



UM10470

LPC178x/7x 用户手册

修订版 1.5 — 2011 年 7 月 6 日

用户手册

文档信息

信息	内容
关键字	LPC1788FBD208 、 LPC1788FET208 、 LPC1788FET180 、 LPC1788FBD144 、 LPC1787FBD208 、 LPC1786FBD208 、 LPC1785FBD208 、 LPC1778FBD208 、 LPC1778FET208 、 LPC1778FET180 、 LPC1778FBD144 、 LPC1777FBD208 、 LPC1776FBD208、LPC1776FET180、LPC1774FBD208、LPC1774FBD144、ARM、 ARM Cortex-M3、32 位、USB、以太网、LCD、CAN、I ² C、I ² S、Flash、EEPROM、 微控制器
摘要	LPC178x/7x 用户手册



修订历史

版本	日期	说明
1.5	20110706	增加了 Power Boost 功能的描述，以及其它一些小的更新和改正。
1.4	20110610	发布官方版 LPC178x/7x 用户手册。增加了事件监控器/记录器。一些小的更新和改正。
1.3	20110307	更换了缺失的图 10。一些小的更新和改正。
1.2	20110225	去除了反映产品初始发布的 SPIFI。一些小的更新和改正。
1.1	20110125	一些小的更新和改正。
1.0	20101022	首次发布 LPC178x/7x 用户手册。

This translated version is for reference only, and the English version shall prevail in case of any discrepancy between the translated and English versions.

版权所有 2012 恩智浦有限公司 未经许可，禁止转载

联络信息

更多信息请访问：<http://www.nxp.com>

销售办事处地址请发邮件至：salesaddresses@nxp.com

1.1 简介

LPC178x/177x 是基于 ARM Cortex-M3 的微控制器，用于处理要求高集成度和低功耗的嵌入式应用。

Cortex-M3 是下一代内核，在相同的时钟速率下能提供比 ARM7 更高的性能，并提供了系统增强型特性，如现代化调试特性和支持更高级别的块集成。Cortex-M3 CPU 具有 3 级流水线 and 哈佛结构，带独立的本地指令总线与数据总线，以及用于外设的性能略低的第三条总线。Cortex-M3 CPU 还包括一个支持随机跳转的内部预取单元。LPC178x/177x 增加了一个专用的 Flash 加速器，使 Flash 中代码执行达到最佳性能。LPC178x/177x 在最差的商用条件下的操作频率可以高达 120MHz。

LPC178x/177x 的外设组件包括高达 512kB 的 Flash 存储器、高达 96kB 的数据存储器、4kB 的 EEPROM 存储器、一个用于 SDRAM 和静态存储器存取的外部存储器控制器、一个 LCD 面板控制器、一个以太网 MAC、一个通用 DMA 控制器、一个 USB 设备/主机/OTG 接口、5 个 UART、3 个 SSP 控制器、3 个 I²C 接口、一个 I²S 串行音频接口、一个双通道 CAN 接口、一个 SD 卡接口、一个 8 通道 12 位 ADC、一个 10 位 ADC、一个电机控制 PWM、一个正交编码器接口、4 个通用定时器、一个 6 输出的通用 PWM、一个带独立电池电源和事件监控器/记录器的超低功耗 RTC、一个窗口式看门狗定时器、一个 CRC 计算引擎、多达 165 个通用 I/O 管脚，等等。LPC178x/177x 的管脚引出线可与 LPC24xx 和 LPC23xx 保持管脚功能兼容。

1.2 特性

具体型号的详细特性，见 [1.4](#) 节。

- 与 LPC23xx 和 24xx 系列器件可进行功能替换。
- ARM Cortex-M3 处理器，可在高至 120MHz 的频率下运行。Cortex-M3 执行 Thumb®-2 指令集，以实现最佳操作与代码长度，包含硬件除法、单周期乘法，以及位字段操作等。同时还包括一个支持 8 个区的存储器保护单元（MPU）。
- ARM Cortex-M3 内置了可嵌套向量中断控制器（NVIC）。
- 具有高达 512kB 的片上 Flash 程序存储器，具有在系统编程(ISP)和在应用编程(IAP)功能。把增强型的 Flash 存储加速器和 Flash 存储器在 CPU 本地代码/数据总线上的位置进行结合，则 Flash 可提供高性能的代码。
- 高达 96kB 的片上 SRAM，包括：
 - 64kB SRAM 可供高性能 CPU 通过本地代码/数据总线访问。
 - 2 个 16kB SRAM 模块，带独立访问路径，可进行更高吞吐量的操作。
这些 SRAM 模块可用于以太网、USB、LCD 以及 DMA 存储器，以及通用指令和数据存储。
 - 4kB 片上 EEPROM。
- 外部存储器控制器，支持异步静态存储器件，如 RAM、ROM 和最多 64MB 的 Flash，以及像单数据速率 SDRAM 这种动态存储器。
- AHB 多层矩阵上具有 8 通道的通用 DMA 控制器（GPDMA），它可结合 SSP、I²S、UART、SD/MMC、CRC 引擎、模数与数模转换器外设、定时器匹配信号和 GPIO 使用，并可用于存储器到存储器的传输。
- 多层 AHB 矩阵内部连接，为每个 AHB 主机提供独立的总线。
AHB 主机包括 CPU、通用 DMA 控制器、以太网 MAC、LCD 控制器，以及 USB 接口。这个内部连接特性提供无仲裁延迟的通信，除非 2 个主机尝试同时访问同一个从机。
- 分离的 APB 总线使 CPU 与 DMA 之间减少了延迟，获得更高的吞吐量。如果 APB 不忙，则单级写入缓存使 CPU 能够连续工作，而无需等待 APB 写操作完成。
- LCD 控制器，同时支持超扭曲向列（STN）与薄膜晶体管（TFT）液晶显示屏。
 - 专用的 DMA 控制器。
 - 可选显示分辨率（最高 1024 × 768 像素）。
 - 支持高达 24 位真彩色模式。
- 串行接口：

- 以太网 MAC 带 MII/RMII 接口与专用的 DMA 控制器。
- USB 2.0 全速从机/主机/OTG 控制器，带有用于从机与主机功能的片上 PHY 和一个专用 DMA 控制器。
- 5 个 UART，带小数波特率发生功能、内部 FIFO、IrDA、DMA 支持，以及 RS-485/EIA-485 支持。UART1 还有全套的调制解调器握手信号。UART4 包含一个同步模式和一个支持 ISO 7816-3 的智能卡模式。144 管脚封装的器件提供 4 个 UART。
- 3 个 SSP 控制器，带有 FIFO，可按多种协议进行通信。SSP 接口可以与 GPDMA 控制器一起使用。
- 3 个增强型 I²C 总线接口，其中 1 个具有开漏输出功能，支持整个 I²C 规范和数据速率为 1MBit/s 的快速模式；另外 2 个具有标准的端口管脚。增强型特性包括多地址识别功能与监控模式。
- 双通道 CAN 控制器。
- 用于数字音频输入或输出的 I²S（IC 之间音频）接口，带小数速率控制功能。I²S 接口可以与 GPDMA 一起使用。I²S 接口支持 3 线的数据传输与接收，或 4 线的联合式传输与接收连接，以及主时钟输出。
- 其它外设：
 - SD 卡接口，同时支持 MMC 卡。
 - 通用 I/O（GPIO）管脚，带可配置的上拉/下拉电阻、开漏模式，以及转发器模式。所有 GPIO 位于 AHB 总线上，以进行快速访问，并支持 Cortex-M3 位带宽（bit-banding）。通过通用 DMA 控制器就可以访问 GPIO。端口 0 和 2 的任何管脚均可生成中断。208 管脚封装上有 165 个 GPIO；180 管脚封装上有 141 个 GPIO；144 管脚封装上有 109 个 GPIO。
 - 12 位模数转换器(ADC)，可在 8 只管脚之间实现多路输入，转换速率高达 400kHz，并具有多个结果寄存器。12 位 ADC 可以与 GPDMA 控制器一起使用。
 - 10 位数模转换器（DAC），具有专门的转换定时器，并支持 DMA 操作。
 - 4 个通用定时器/计数器，共有 8 个捕获输入和 10 个比较输出。每个定时器模块都具有一个外部计数输入。可以选择特定的定时器事件来生成 DMA 请求。
 - 1 个电机控制 PWM，支持三相电机控制。
 - 正交编码器接口，可监控一个外接的正交编码器。
 - 2 个标准的 PWM/定时器模块，带外部计数输入。

- 带有独立电源域的实时时钟（RTC）。RTC 通过专用的 RTC 振荡器来驱动。RTC 模块包括 20 字节的电池供电备份寄存器，当芯片其它部分掉电时，允许系统状态存储在该寄存器中。电池电源可由标准的 3V 锂电池供电。当电池电压掉至 2.1V 的低电压时，RTC 仍能继续工作。RTC 中断可将 CPU 从任何低功耗模式中唤醒。
- 事件监控器/记录器，当 3 个输入的任何一个是发生事件时，它可以捕获 RTC 的值。事件标识与发生时间都存储在寄存器中。事件监控器/记录器使用 RTC 电源域，因此只要 RTC 有供电，它就能工作。
- 窗口式看门狗定时器（WWDG）。窗口化运行、专用的内部振荡器、看门狗警告中断，以及安全特性等。
- CRC 引擎模块可以根据提供的数据，根据 3 种标准多项式中的一种，计算出 CRC。CRC 引擎可以与 DMA 控制器联合使用，因此在数据传输中无需 CPU 介入，就能生成一个 CRC。
- Cortex-M3 系统节拍定时器，包括外部时钟输入选项。
- 标准的 JTAG 测试/调试接口，以及串行线调试与串行线跟踪端口选项。
- 支持实时跟踪的仿真跟踪模块。
- 单个 3.3V 电源（2.4V—3.6V）。温度范围-40°C~85°C。
- 4 个低功耗模式：睡眠、深度睡眠、掉电、深度掉电。
- 通过降低片上稳压器的输出电压，可在 100MHz 或以下做省电运行。
- 4 个外部中断输入，可配置为边沿/电平触发。PORT0 和 PORT2 上的全部管脚均可用做边沿触发的中断源。
- 不可屏蔽中断（NMI）输入。
- 时钟输出功能，可反映主振荡器时钟、IRC 时钟、RTC 时钟、CPU 时钟、USB 时钟或看门狗定时器时钟的输出状态。
- 唤醒中断控制器（WIC）允许 CPU 从时钟在深度睡眠、掉电，深度掉电模式下停止时发生的任何优先级中断中自动唤醒。
- 在处于掉电模式时，可通过中断将处理器从掉电模式中的唤醒，这些中断包括外部中断、RTC 中断、USB 活动中断、以太网唤醒中断、CAN 总线活动中断、PORT0/2 管脚中断和 NMI 等。
- 带掉电检测功能，可对掉电中断和强制复位分别设置阈值。
- 片上有上电复位电路。
- 片上晶振工作频率为 1MHz~25MHz。
- 12MHz 内部 RC 振荡器（IRC）可在±1%的精度内调整，可选择用做系统时钟。
- 通过片上 PLL，没有高频晶振，CPU 也可以最高频率运转。可以从主振荡器或内部 RC 振荡器上运行。

- 第二个专用的 PLL 可用于 USB 接口，从而增加了主 PLL 设置的灵活性。
- 多功能管脚功能选择特性使片上外设功能的使用有了多种可能性。
- 简化电路板测试的边界扫描功能。
- 唯一的器件串行号码，用于识别。
- 现有 208 管脚 LQFP、208 管脚 TFBGA、180 管脚 TFBGA，以及 144 管脚 LQFP 封装。

1.3 应用

- **通信**
 - 销售点终端、Web 服务器、多协议网桥
- **工业/医疗**
 - 自动化控制器、应用控制、机器人控制、HVAC、PLC、变频器、断路器、医疗扫描、安保监控、电机传动，以及视频对讲等
- **消费/家电**
 - 音频、MP3 解码器、警报系统、显示器、打印机、扫描仪、小家电，以及健身设备
- **汽车**
 - 零部件、汽车防盗、GPS/车队监控器

1.4 订购信息

表1. 订购信息

型号	封装		
	名称	描述	版本
LPC1788			
LPC1788FBD208	LQFP208	低矮方形扁平塑料封装；208 根引线；本体为 28x28x1.4 mm	SOT459-1
LPC1788			
LPC1788FBD208	LQFP208	低矮方形扁平塑料封装；208 根引线；本体为 28x28x1.4 mm	SOT459-1
LPC1788FET208	TFBGA208	超细间距球栅阵列塑料封装；208 只球；本体为 15x15x0.7 mm	SOT950-1
LPC1788FET180	TFBGA180	超细间距球栅阵列封装；180 只球；本体为 12x12x0.8 mm	SOT570-2
LPC1788FBD144	LQFP144	低矮四方扁平塑料封装；144 根引线；本体为 20x20x1.4 mm	SOT486-1
LPC1787			
LPC1787FBD208	LQFP208	低矮方形扁平塑料封装；208 根引线；本体为 28x28x1.4 mm	SOT459-1
LPC1786			
LPC1786FBD208	LQFP208	低矮四方扁平塑料封装；208 根引线；本体为 28x28x1.4 mm	SOT459-1
LPC1785			
LPC1785FBD208	LQFP208	低矮方形扁平塑料封装；208 根引线；本体为 28x28x1.4 mm	SOT459-1
LPC1778			
LPC1778FBD208	LQFP208	低矮方形扁平塑料封装；208 根引线；本体为 28x28x1.4 mm	SOT459-1
LPC1788FET208	TFBGA208	超细间距球栅阵列塑料封装；208 只球；本体为 15x15x0.7 mm	SOT950-1
LPC1788FET180	TFBGA180	超细间距球栅阵列封装；180 只球；本体为 12x12x0.8 mm	SOT570-2
LPC1778FBD144	LQFP144	低矮方形扁平塑料封装；144 根引线；本体为 20x20x1.4 mm	SOT486-1
LPC1777			
LPC1777FBD208	LQFP208	低矮方形扁平塑料封装；208 根引线；本体为 28x28x1.4 mm	SOT459-1
LPC1776			
LPC1776FBD208	LQFP208	低矮方形扁平塑料封装；208 根引线；本体为 28x28x1.4 mm	SOT459-1
LPC1788FET180	TFBGA180	超细间距球栅阵列封装；180 只球；本体为 12x12x0.8 mm	SOT570-2
LPC1774			
LPC1774FBD208	LQFP208	低矮方形扁平塑料封装；208 根引线；本体为 28x28x1.4 mm	SOT459-1
LPC1774FBD144	LQFP144	低矮方形扁平塑料封装；144 根引线；本体为 20x20x1.4 mm	SOT486-1

1.4.1 器件选项汇总

表2. LPC178x/177x 器件订购选项

型号 ^[1]	Flash (kB)	SRAM (kB)	EEPROM (kB)	以太网	USB	UART	外部存储器 总线 ^[2]	LCD	QEI	SD
LPC178x										
LPC1788	512	96 ^[5]	4	有	H/O/D	5	32-bit/ 16-bit ^[3] / 8-bit ^[4]	有	有	有
LPC1787	512	96 ^[5]	4	无	H/O/D	5	32-bit	有	有	有
LPC1786	256	80 ^[6]	4	有	H/O/D	5	32-bit	有	有	有
LPC1785	256	80 ^[6]	4	无	H/O/D	5	32-bit	无	有	有
LPC177x										
LPC1778	512	96 ^[5]	4	有	H/O/D	5	32-bit/ 16-bit ^[3] / 8-bit ^[4]	无	有	有
LPC1777	512	96 ^[5]	4	无	H/O/D	5	32-bit	无	有	有
LPC1776	256	80 ^[6]	4	有	H/O/D	5	32-bit/ 16-bit ^[3]	无	有	有
LPC1774	128	40 ^[7]	2	无	D	5/4 ^[4]	32-bit/ 8-bit ^[4]	无	无	无

[1] 所有型号包括 2 个 CAN 通道、3 个 SSP 接口、3 个 I²C 接口、I²S、DAC 和 1 个 8 路 12 位的 ADC。

[2] 可用封装的最大数据总线宽度。可以采用比最大数据总线宽度小的数据总线宽度。欲了解外部总线在不同封装上的具体应用，请参见 10.1 节。

[3] 180 管脚封装的外部总线被限制在 16 位。

[4] 144 管脚封装不包括 UART4 和 DAC，其外部总线被限制在 8 位。

[5] 64kB 的 CPU SRAM 外加 32kB 外设 SRAM。

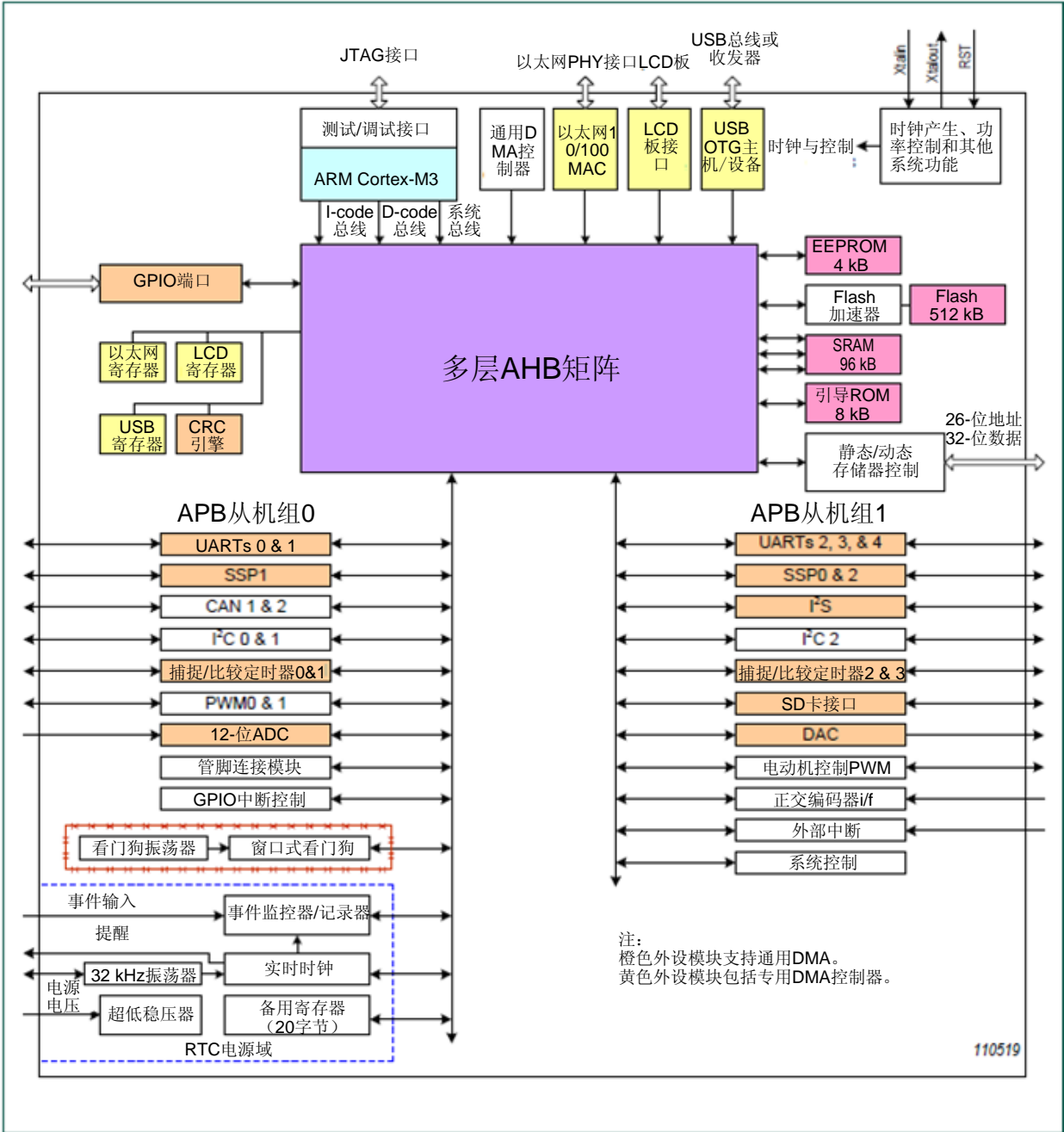
[6] 64kB 的 CPU SRAM 外加 16kB 外设 SRAM。

[7] 32kB 的 CPU SRAM 外加 8kB 外设 SRAM。

[8] 16kB 的 CPU SRAM 外加 8kB 外设 SRAM。

1.5 简化方框图

图1. LPC178x/177x 简化方框图



1.6 结构概述

ARM Cortex-M3 包含三条 AHB-Lite 总线、一条系统总线，以及 I-code 和 D-code 总线，后二者的速率较快，且与 TCM 接口的用法类似：一条总线专用于指令拾取（I-code），另一条总线用于数据访问（D-code）。当对不同的目标设备同时进行操作时，这两条内核总线允许同步操作。

LPC178x/177x 采用多层 AHB 矩阵来连接 Cortex-M3 总线，并以灵活的方式将其它总线主机连接到外设，这种方式允许不同的总线主机同时访问矩阵上不同从机端口的的外设，从而优化了性能。多层矩阵连接的详情见[图 2](#)。

APB 外设使用多层 AHB 矩阵的独立从机端口通过两条 APB 总线连接到 CPU。这样就减少了 CPU 与 DMA 控制器之间的争用，从而获得更好的性能。APB 总线桥被配置为缓冲区写操作，使得 CPU 或 DMA 控制器无需等待 APB 写操作结束。

1.7 ARM Cortex-M3 处理器

ARM Cortex-M3 是一款通用的 32 位微处理器，它具有高性能和超低功耗的特性。Cortex-M3 提供了很多新的特性，包括 Thumb-2 指令集、低中断延时、硬件除法、可中断/可连续的多次装载与存储指令、中断状态的自动保存与恢复、紧密结合中断控制器与唤醒中断控制器，多条内核总线可同时用于访问。

采用了流水线技术，使得处理系统与存储器系统的各个部分都能连续工作。通常情况下，当一个指令正在执行时，第二个指令正在进行解码，而第三个指令正在从存储器中拾取。

ARM Cortex-M3 处理器详情，请见本手册附带的《Cortex-M3 用户指南》。

1.7.1 Cortex-M3 配置选项

LPC178x/177x 使用 Cortex-M3 CPU 的 r2p0 版，它包括一系列可配置选项，具体如下。

系统选项：

- 包括嵌入式向量中断控制器（NVIC）。NVIC 包括了 SYSTICK 定时器。
- 包括唤醒中断控制器（WIC）。WIC 具有将 CPU 从低功耗模式下唤醒的更有效选项。
- 包括存储器保护单元（MPU）。
- 包括了 ROM 表。ROM 表提供了调试部件到外部调试系统的地址。

调试相关选项：

- 包括 JTAG 调试接口。

- 包括串行线调试。串行线调试允许只使用两条线进行调试，简单的跟踪功能可增加第三条线。
- 包括了嵌入式跟踪宏单元（ETM）。ETM 提供指令跟踪功能。
- 包括数据观察点与跟踪（DWT）单元。DWT 允许数据地址或数据值匹配为跟踪信息或触发其它事件。DWT 包括 4 个比较器和计数器，以用于特定的内部事件。
- 包括指令跟踪宏单元（ITM）。软件可写 ITM，以发送消息到跟踪端口。
- 包括了跟踪端口接口单元（TPIU）。TPIU 解码并向外面提供跟踪信息。这可以在串行线浏览器管脚或 4 位并行跟踪端口上实现。
- 包括了 Flash 修补与断点（FPB）。FPB 可以产生硬件断点，并且在代码空间中重新映射到特定的地址到 SRAM 作为更改非易失性代码的临时方法。FPB 包括 2 个文字比较器和 6 个指令比较器。

1.8 片上 Flash 存储器系统

LPC178x/177x 包含了多达 512kB 的片上 Flash 存储器。Flash 存储器加速器实现了 CPU 存取性能的最大化。该存储器可以同时用于代码与数据的存储。对 Flash 存储器的编写有若干种方式来实现。它可通过串行端口来进行在系统编程。应用程序也可以在运行时对 Flash 进行擦除和/或编程，从而为数据存储域固件升级等操作带来极大的灵活性。

1.9 片上静态 RAM

LPC178x/177x 包含多达 96kB 的片上静态 RAM 存储器。高达 64kB 的 SRAM（CPU 与通用 DMA 控制器可访问）位于较高速总线上。另外 2 个高达 16kB 的 SRAM 模块可提供最多 32kB 的 SRAM，主要用于外设数据。当两个 SRAM 均存在时，它们位于 AHB 多层矩阵上的独立从机端口上。

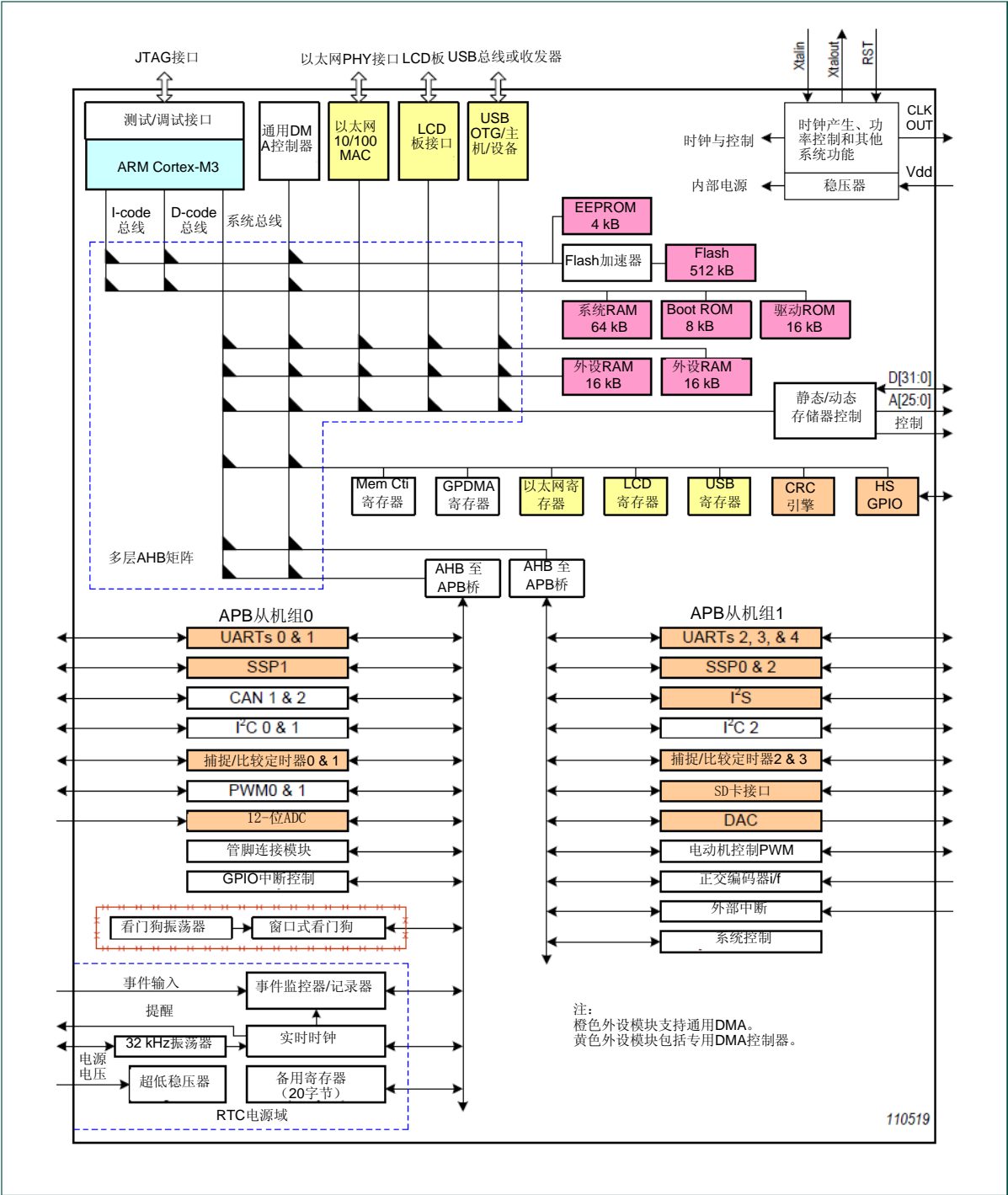
这种结构允许各自执行 CPU 和 DMA 访问操作，从而减少总线主机延迟或没有延迟。它还允许区分不同外设功能的数据，从而提高系统性能。例如，LCD DMA 可以占用一个 SRAM，而以太网 DMA 则占用另一个，与此同时 CPU 则使用系统 SRAM 做数据和/或指令存取。

1.10 片上 EEPROM

LPC178x/177x 包括高达 4kB 的片上 EEPROM 存储器。EEPROM 只允许 CPU 访问。

1.11 详细方框图

图2. LPC178x/177x 方框图，CPU 和总线



2.1 存储器映射与外设寻址

ARM Cortex-M3 处理器含有一个 4GB 的地址空间。下表显示了 LPC178x/177x 如何使用该空间。

表3. LPC178x/177x 存储器使用及明细

地址范围	用途	地址范围明细及描述	
0x0000 0000 to 0x1FFF FFFF	片上非易失性存储器	0x0000 0000—0x0007 FFFF	用于配有 512kB Flash 存储器的设备
		0x0000 0000—0x0003 FFFF	用于配有 256kB Flash 存储器的设备
		0x0000 0000—0x0001 FFFF	用于配有 128kB Flash 存储器的设备
		0x0000 0000—0x0000 FFFF	用于配有 64kB Flash 存储器的设备
	片上 SRAM	0x1000 0000—0x1000 FFFF	用于配有 64kB 本地 SRAM 的设备
		0x1000 0000—0x1000 7FFF	用于配有 32kB 本地 SRAM 的设备
		0x1000 0000—0x1000 3FFF	用于配有 16kB 本地 SRAM 的设备
	引导芯片	0x1FFF 0000—0x1FFF 1FFF	带 Flash 功能的 8kB 引导芯片
0x2000 0000 to 0x3FFF FFFF	片上 SRAM（通常用于存储外设数据）	0x2000 0000—0x2000 1FFF	外设 RAM-Bank0（第一个 8kB）
		0x2002 0000—0x2000 3FFF	外设 RAM-Bank0（第二个 8kB）
		0x2000 4000—0x2000 7FFF	外设 RAM-Bank1（16kB）
	AHB 外设	0x2008 0000—0x200B FFFF	详见 2.3.1 节。
0x4000 0000 to 0x7FFF FFFF	APB 外设	0x4000 0000—0x4007 FFFF	APB0 外设，包含多达 32 个外设模块，每个 16kB
		0x4008 0000—0x400F FFFF	APB1 外设，包含多达 32 个外设模块，每个 16kB
0x8000 0000 to 0xDFFF FFFF	通过外部存储控制器的片外存储器	4 个静态存储器片选：	
		0x8000 0000—0x83FF FFFF	静态存储器片选 0（多达 64MB） ^[1]
		0x9000 0000—0x93FF FFFF	静态存储器片选 1（多达 64MB） ^[2]
		0x9800 0000—0x9BFF FFFF	静态存储器片选 2（多达 64MB）
		0x9C00 0000—0x9FFF FFFF	静态存储器片选 3（多达 64MB）
		4 个动态存储器片选：	
		0xA000 0000—0xAFFF FFFF	动态存储器片选 0（多达 256MB）
		0xB000 0000—0xBFFF FFFF	动态存储器片选 1（多达 256MB）
		0xC000 0000—0xCFFF FFFF	动态存储器片选 2（多达 256MB）
		0xD000 0000—0xDFFF FFFF	动态存储器片选 3（多达 256MB）
0xE000 0000 to 0xE00F FFFF	Cortex-M3 私有外设总线	0xE000 0000—0xE00F FFFF	Cortex-M3 相关功能，包括可嵌套向量中断控制器和系统节拍定时器。

[1] 如果使能了地址转换模式，存储量可以多达 256MB，高位地址为 0x8FFF FFFF。参见 0 位 SCS 寄存器（3.8.1 节）。

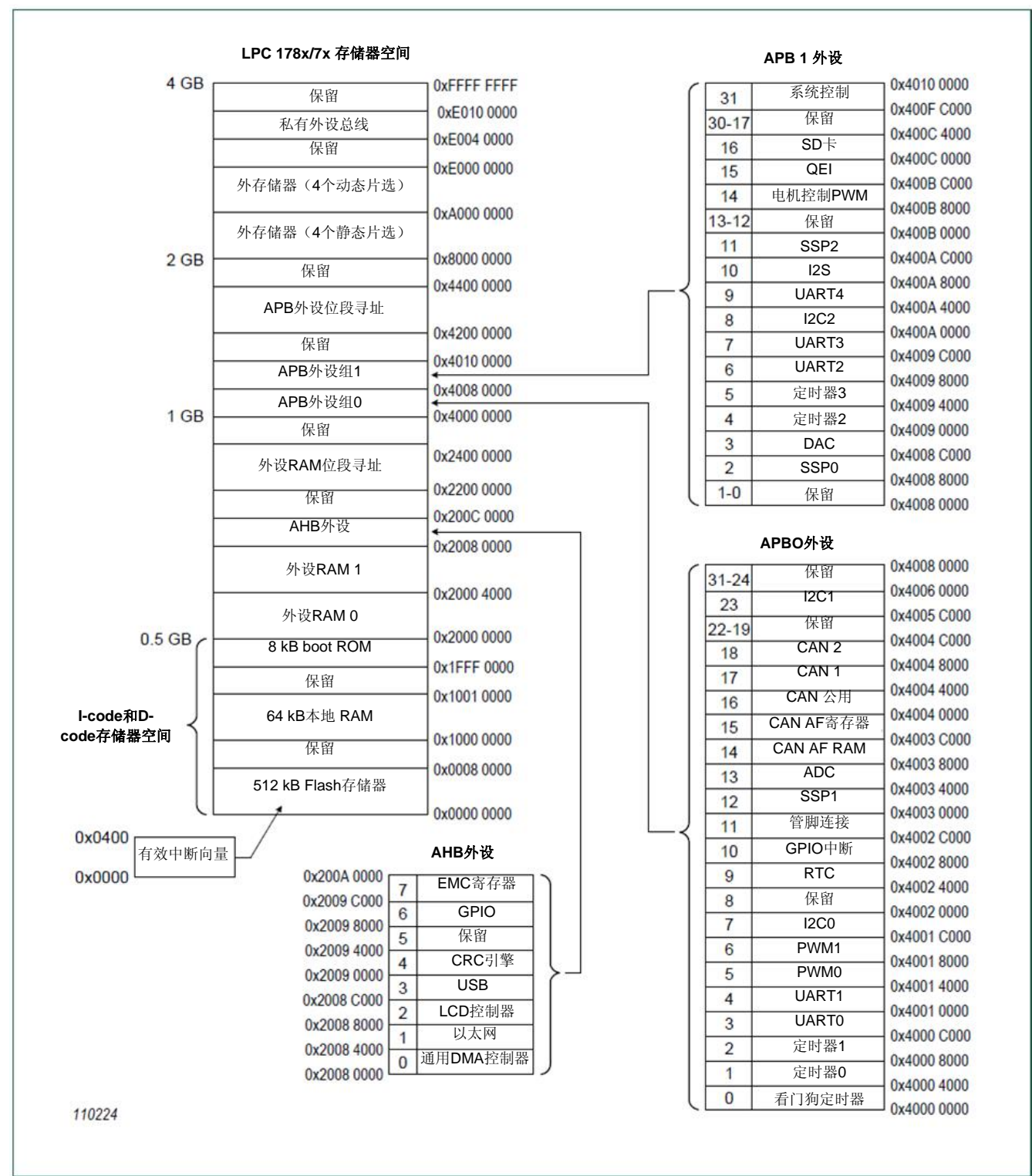
[2] 如果使能了地址转换模式，存储量可以多达 128MB，高位地址为 0x97FF FFFF。参见 0 位 SCS 寄存器（3.8.1 节）。

2.2 存储器映射

LPC178x/177x 包含多个独立的存储区域，如下图所示。[图 3](#)显示的是复位后从用户编程角度所看到的整个地址空间映射。中断向量区支持地址的重新映射，这将在本节后面说明。

[图 3](#)和[表 5](#)显示了从不同角度所观察到的外设地址空间。AHB 外设区域为 2MB，可分配多达 128 个外设。APB 外设区域为 1MB，可分配多达 64 个外设。每个外设空间大小都为 16kB。这样可以简化每个外设的地址解码。

图3. LPC 1788 系统存储器映射



2.3 片上外设

所有外设寄存器不管规格大小，都按照字地址进行分配（32 位边界）。这样就不再需要使用字节定位映射的硬件来进行小边界的字节（8 位）或半字（16 位）访问。这样做的结果是，所有字寄存器与半字寄存器都是一次性访问。例如，不能对一个字寄存器的最高字节执行单独的读或写操作。

2.3.1 AHB 外设

下表给出了 AHB 总线矩阵上的外设功能地址。有关寄存器的完整描述可见相关章节。

表4. AHB 外设及基址

AHB 外设	基址	外设名称
0	0x2008 0000 to 0x2008 3FFF	通用 DMA 控制器
1	0x2008 4000 to 0x2008 7FFF	以太网 MAC
2	0x2008 8000 to 0x2008 BFFF	LCD 控制器
3	0x2008 C000 to 0x2008 FFFF	USB 接口
4	0x2009 0000 to 0x2009 3FFF	CRC 引擎
5	0x2009 4000 to 0x2009 7FFF	保留
6	0x2009 8000 to 0x2009 BFFF	GPIO
7	0x2009 C000 to 0x2009 FFFF	外部存储控制器
8 to 15	0x200A 0000 to 0x200B FFFF	保留

2.3.2 APB 外设地址

下表显示了 2 个 APB 总线的地址映射。APB 外设不会全部用完分配给它们的 16kB 空间。通常，每个器件的寄存器在各个 16kB 范围内的多个位置上采用“别名”或重复。

表5. APB0 外设及基址

APB0 外设	基址	外设名称
0	0x4000 0000	看门狗定时器
1	0x4000 4000	定时器 0
2	0x4000 8000	定时器 1
3	0x4000 C000	通用异步收发传输器 0
4	0x4001 0000	通用异步收发传输器 1
5	0x4001 4000	PWM0
6	0x4001 8000	PWM1
7	0x4001 C000	I ² C0
8	0x4002 0000	保留
9	0x4002 4000	实时时钟和时间监视器/记录器
10	0x4002 8000	GPIO 中断
11	0x4002 C000	管脚连接模块
12	0x4003 0000	串行同步接口 1
13	0x4003 4000	A/D 转换器
14	0x4003 8000	CAN 接收滤波器 RAM
15	0x4003 C000	CAN 接收滤波器寄存器

APB0 外设	基址	外设名称
16	0x4004 0000	CAN 公用寄存器
17	0x4004 4000	CAN 控制器 1
18	0x4004 8000	CAN 控制器 2
19 to 22	0x4004 C000 to 0x4005 8000	保留
23	0x4005 C000	I ² C1
24 to 31	0x4006 0000 to 0x4007 C000	保留

表6. APB1 外设及基址

APB1 外设	基址	外设名称
0 to 1	0x4008 0000 to 0x4008 4000	保留
2	0x4008 8000	串行同步接口 0
3	0x4008 C000	D/A 转换器
4	0x4009 0000	定时器 2
5	0x4009 4000	定时器 3
6	0x4009 8000	通用异步收发传输器 2
7	0x4009 C000	通用异步收发传输器 3
8	0x400A 0000	I ² C2
9	0x400A 4000	通用异步收发传输器 4
10	0x400A 8000	I ² S
11	0x400A C000	串行同步接口 2
12 to 13	0x400B 0000 to 0x400B 4000	保留
14	0x400B 8000	电机控制 PWM
15	0x400B C000	正交编码器接口
16	0x400C 0000	SD 卡接口
17 to 30	0x400D 0000 to 0x400F 8000	保留
31	0x400F C000	系统控制

2.4 存储器重新映射

Cortex-M3 包含了一种允许将中断向量表重新映射到存储器映射空间的备用单元的机制。这通过 Cortex-M3 所包含的向量表偏移寄存器控制。有关向量表偏移功能的详细情况，请见 6.4 节的 NVIC 描述，以及本手册附带的《Cortex-M3 用户指南》中 39.4.3.5 节。

在无 Flash 工作情况，以 0x8000 0000 为起始地址的用户程序必须设置一个中断向量表。有关无 Flash 工作详情，见 7.2 节的“引导控制”。

启动 ROM 的重新映射

在一个硬件复位后，启动 ROM 会临时被映射到地址 0。通常，这个过程用户可见。但是，如果该执行在复位后立即被一个调试程序停止，则应为用户校正映射。见 38.8 节。

2.5 AHB 仲裁

对多层 AHB 矩阵，只有当多个主机试图同时访问同一矩阵从机端口时，才会做主机之间的仲裁。默认情况下，Cortex-M3 的 D-code 总线有最高优先级，然后是 I-code 总线。所有其它主机的优先级较低。

用户可以改变默认优先级。在使用 LCD 接口而无法获取充分的数据时，这种方法尤其有效。

2.5.1 矩阵仲裁寄存器（Matrix_Arb—0x400F C188）

矩阵仲裁寄存器提供了修改 AHB 矩阵仲裁默认优先级的功能。

表7. 矩阵仲裁寄存器（矩阵_仲裁-0x400F C188）位描述

位	符号	描述	复位值
1:0	PRI_ICODE	I-Code 总线优先级应低于 PRI_DCODE 的优先级以保证正常运行。	0x1
3:2	PRI_DCODE	D-Code 总线优先级	0x3
5:4	PRI_SYS	系统总线优先级	0
7:6	PRI_GPDMA	通用 DMA 控制器优先级	0
9:8	PRI_ETH	以太网 DMA 优先级	0
11:10	PRI_LCD	LCD DMA 优先级	0
13:12	PRI_USB	USB DMA 优先级	0
15:14	-	保留。读取值未定义，只写入 0。	无
16	ROM_LAT	只读存储器延迟选择。应固定为 0。	0
31:17	-	保留。读取值未定义，只写入 0。	无

用于指示各优先级的数值为：3=最高，0=最低。

例如，使用 0x0000 0C09 可以赋予 LCD DMA 优先级。这将给予 LCD 最高优先级，D-code 优先级其次，I-code 第三，而所有其它的优先级最低。

通过管理代码与各种数据在存储空间的位置，可有助于减少对仲裁的需求以及出现任何总线主机耗尽的可能性，并且无需改变默认的优先级。例如，LCD 通过连接到 EMC 的片外存储器刷新，同时通过 EMC 执行片外代码则会产生大量的仲裁需求。

3.1 简介

系统控制模块包含了多个系统特性和控制寄存器，它们的许多功能与特定的外设无关。这些功能包括：

- 芯片复位（见 [3.4](#) 节）
- 外设复位控制（见 [3.5](#) 节）
- 掉电检测（见 [3.6](#) 节）
- 外部中断输入（见 [3.7](#) 节）
- 其它系统控制与状态（见 [3.8](#) 节）

必要情况下，每个功能都有自己的寄存器，不需要的位被定义为保留，以支持未来扩展。

3.2 管脚描述

[表 8](#) 所示为与系统控制块功能相关的管脚。

表8. 管脚汇总

管脚名称	管脚方向	管脚描述
EINT0	输入	外部中断输入 0 — 一个低电平/高电平或下降/上升沿有效的通用中断输入。该管脚可用于将处理器从睡眠、深度睡眠或掉电模式中唤醒。
EINT1	输入	外部中断输入 1 — 参见上述 EINT0 描述。
EINT2	输入	外部中断输入 2 — 参见上述 EINT0 描述。
EINT3	输入	外部中断输入 3 — 参见上述 EINT0 描述。
RESET	输入	外部复位输入 — 在这个管脚上的 LOW 使芯片复位,导致 I/O 端口及外设恢复默认状态,并导致处理器从地址 0x0000 0000 开始执行。

3.3 寄存器描述

所有寄存器，无论大小，都按照字地址边界对齐。寄存器详情见各功能说明。

表9. 系统控制模块寄存器汇总

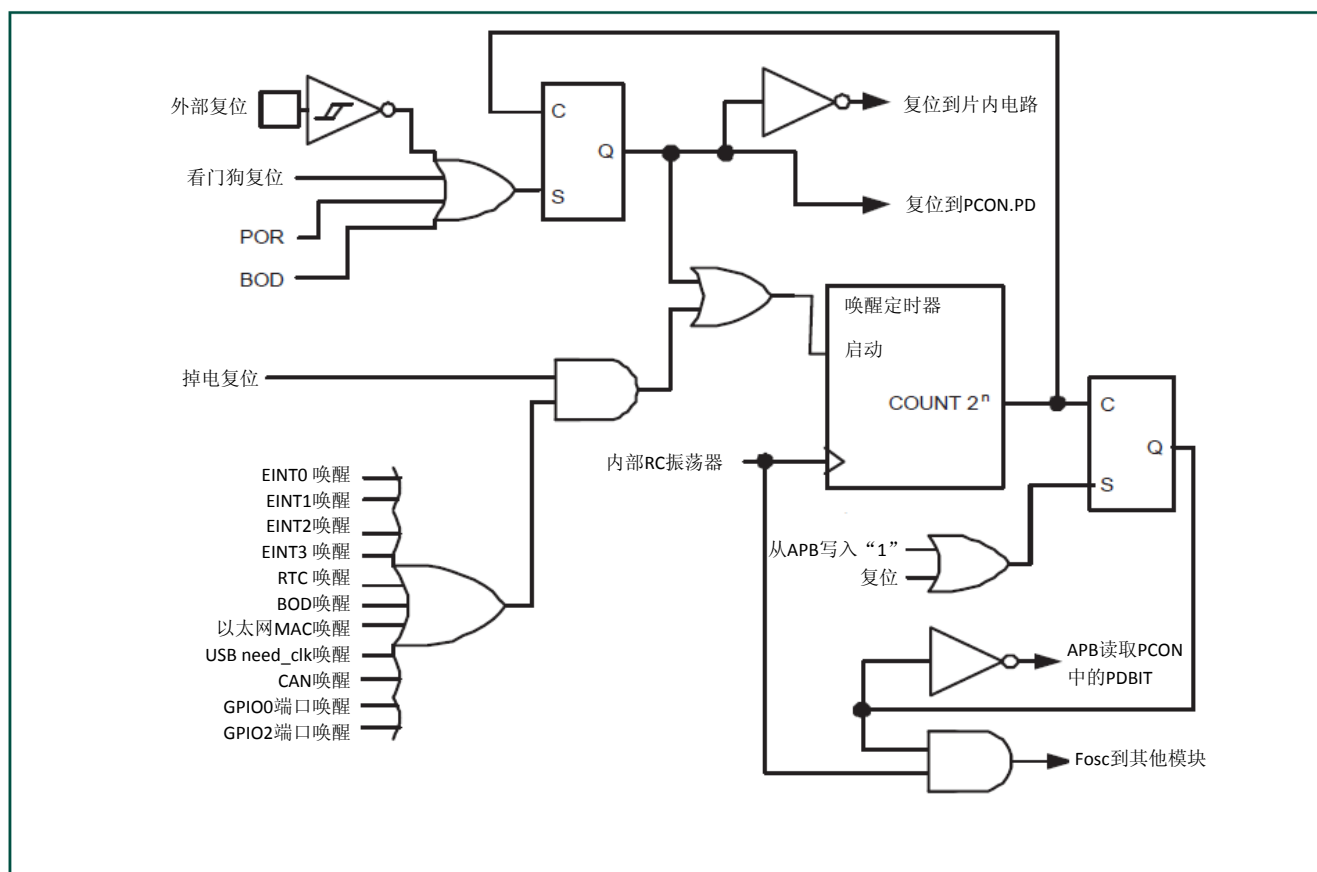
名称	描述	访问	复位值	地址	表
复位					
RSID	复位源标识寄存器	R/W	参见 表 10	0x400F C180	表 10
外设复位控制					
RSTCON0	个别外设复位控制位	R/W	0	0x400F C1CC	表 11
RSTCON1	个别外设复位控制位	R/W	0	0x400F C1D0	表 12
外部中断					
EXTINT	外部中断标志寄存器	R/W	0	0x400F C140	表 14
EXTMODE	外部中断模式寄存器	R/W	0	0x400F C148	表 15
EXTPOLAR	外部中断极性寄存器	R/W	0	0x400F C14C	表 16
Syscon 杂项寄存器					
SCS	系统控制及状态	R/W	0	0x400F C1A0	表 17

3.4 芯片复位

LPC178x/177x 有 6 个复位源： $\overline{\text{RESET}}$ 管脚复位、看门狗复位、上电复位（POR）、掉电检测（BOD）复位、系统复位，以及锁定。

$\overline{\text{RESET}}$ 管脚是一个施密特触发器输入管脚。一旦操作电压达到某个可用电平，则任何复位源激活的芯片复位会启动唤醒定时器（见本章中 4.8 节“唤醒定时器”的描述），复位信号将保持有效，直到外部复位信号被撤销，振荡器开始运行，时钟计数超过了固定的时钟个数，并且 Flash 控制器已完成其初始化。复位逻辑显示在下列方框图中（见图 4）。

图4. 复位模块方框图（包括唤醒定时器）



当一个复位源向 Cortex-M3 CPU 外部发出复位信号时（POR、BOD 复位、外部中断复位以及看门狗复位），IRC 开始起振。IRC 起振（上电后最多 60μs）以及 IRC 提供稳定的时钟输出后，复位信号被锁存并与 IRC 时钟同步。然后将同时启动下面两个序列：

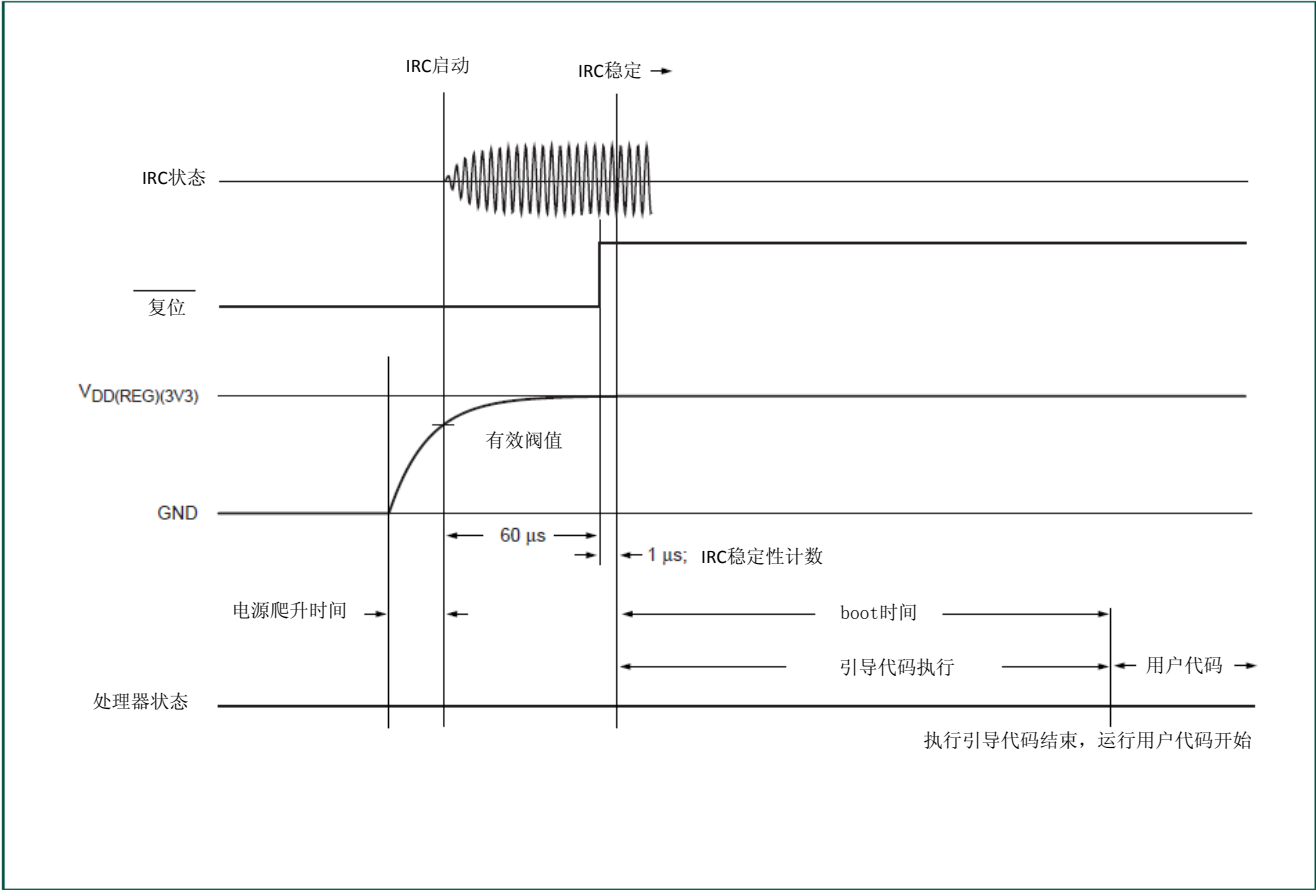
1. 当同步复位无效时，2 位 IRC 唤醒定时器开始计数。2 位 IRC 唤醒定时器的计数结束时，ROM 中的引导代码启动，完成启动工作，就可以跳至 Flash。如果 Flash 未准备就绪接受访问，则 Flash 加速器将插入等待周期，直到 Flash 就绪。
2. 当同步复位无效时，Flash 唤醒定时器（9 位）开始计数。Flash 唤醒定时器产生 100μs

的 Flash 启动时间。一旦达到此时间，Flash 的初始化序列被启动，这将需要大约 250 个周期。这一任务完成后，Flash 加速器将被授予 Flash 访问权。

当内部复位被去除时，处理器会从地址 0 开始执行，这个地址开始是从引导模块映射而来的复位向量。此时，所有处理器与外设寄存器都已被初始化为预先确定的值。

图 5 给出了当 LPC178x/177x 在复位后启动时，RESET、IRC 以及处理器状态之间关系的一个实例。有关由用户代码使能后的主振荡器启动序列，参见 4.3.2 节。

图5. 复位后启动示例



3.4.1
复位源标识寄存器（RSID—0x400F C180）

每个复位源对应该寄存器中某一位，这些标志位通过写入“1”清除。4 个复位源之间的相互关系见下面的说明。

表10. 复位源标识寄存器（RSID—0x400F C180）位描述

位	符号	描述	复位值
0	POR	上电复位（POR）信号有效时该位置位，并清零该寄存器中其它所有的位。但是如果上电复位信号撤销后另外一个复位信号（如外部复位）仍然保持有效，则这个复位信号对应的位置位。POR 位不受其它任何复位源的复位影响。	参见描述
1	EXTR	外部 $\overline{\text{RESET}}$ 信号的激活设置了此位。该位只能由软件或上电复位来清零。	参见描述
2	WDTR	当看门狗定时器溢出并且看门狗模式寄存器的 WDTRESET 位为 1 时，该位置位。该位只能由软件或上电复位来清零。	参见描述
3	BODR	<p>当 $V_{\text{DD(REG)(3V3)}}$ 电压降低于掉电检测（BOD）复位水平（在标称室温条件下，通常是 1.85 伏）时，该位置位。</p> <p>如果 $V_{\text{DD(REG)(3V3)}}$ 电压从 3.3V 降到 BOD 复位触发电平以下后又回升，BODR 位将被设成 1。</p> <p>如果 $V_{\text{DD(REG)(3V3)}}$ 电压从 3.3V 降到 BOD 复位触发电平以下后继续下降到 POR 有效的电压，BODR 位清零。</p> <p>如果 $V_{\text{DD(REG)(3V3)}}$ 电压从 1V 以下持续上升到 BOD 复位触发电平以上，BODR 位将被设成 1。</p> <p>该位只能由软件或上电复位来清零。</p> <p>说明：只有在复位发生并且位 POR=0 时，BODR 位才指示 $V_{\text{DD(REG)(3V3)}}$ 电压是否曾在 BOD 复位触发电平以下。</p>	参见描述
4	SYSRESET	如果处理器由于系统请求复位而被复位，则该位置位。具体描述见 39.4.3.6 节。在 Cortex-M3 AIRCR 寄存器上的 SYSRESETREQ 置位导致了 LPC178x/177x 上的芯片复位。该位只能由软件或上电复位来清零。	参见描述
5	LOCKUP	如果处理器因锁定而复位，则该位置位。具体描述见 39.3.4.4 节。锁定状态导致 LPC178x/177x 上的芯片复位。该位只能由软件或上电复位来清零。	参见描述
31:6	-	保留。读取值未定义，只写入 0。	无

3.5 外设复位控制

LPC17M6x 上的多数外设功能都有一个由软件发起的硬件复位，方法是在 RSTCON0 和 RSTCON1 寄存器中设置相应的位。然后，软件必须清除 RSTCON 寄存器，才能使外设工作。外设会保持在一个硬件复位状态，直到 RSTCON 中的相应位等于 1。

3.5.1 复位控制寄存器 0 (RSTCON0—0x400F C1CC)

使用 RSTCON0 寄存器，可以给很多外设一个硬件复位。见[表 11](#)中的说明。继续采用 RSTCON1 寄存器可以复位更多的外设。

表11. 复位控制寄存器 0 (RSTCON0—0x400F C1CC) 位描述

位	符号	描述	复位值
0	RSTLCD	LCD 控制器复位控制位。	0
1	RSTTIM0	定时器/计数器 0 复位控制位。	0
2	RSTTIM1	定时器/计数器 1 复位控制位。	0
3	RSTUART0	UART0 复位控制位。	0
4	RSTUART1	UART1 复位控制位。	0
5	RSTPWM0	PWM0 复位控制位。	0
6	RSTPWM1	PWM1 复位控制位。	0
7	RSTI2C0	I ² C0 接口复位控制位。	0
8	RSTUART4	UART4 复位控制位。	0
9	RSTRTC	实时时钟 (RTC) 和事件监视器/记录器复位控制位。RTC 复位受限，具体请参见 表 620 。	0
10	RSTSSP1	SSP1 接口复位控制位。	0
11	RSTEMC	外部存储控制器复位控制位。	0
12	RSTADC	A/D 转换器 (ADC) 复位控制位。	0
13	RSTCAN1	CAN 控制器 1 复位控制位。说明：CAN 接收滤波器可能被 RSTCON1 寄存器中的一个独立位复位置零。	0
14	RSTCAN2	CAN 控制器 2 复位控制位。说明：CAN 接收滤波器可能被 RSTCON1 寄存器中的一个独立位复位置零。	0
15	RSTGPIO	GPIO 和 GPIO 中断复位控制位。说明：IOCON 可能被 RSTCON1 寄存器中的一个独立位复位置零。	0
16	-	保留。读取值未定义，只写入 0。	无
17	RSTMCPWM	电机控制 PWM 复位控制位。	0
18	RSTQEI	正交编码器接口复位控制位。	0
19	RSTI2C1	I ² C1 接口复位控制位。	0
20	RSTSSP2	SSP2 接口复位控制位。	0
21	RSTSSP0	SSP0 接口复位控制位。	0
22	RSTTIM2	定时器 2 复位控制位。	0
23	RSTTIM3	定时器 3 复位控制位。	0

位	符号	描述	复位值
24	RSTUART2	UART 2 复位控制位。	0
25	RSTUART3	UART 3 复位控制位。	0
26	RSTI2C2	I2C 接口 2 复位控制位。	0
27	RSTI2S	I2S 接口 2 复位控制位。	0
28	RSTSDC	SD 卡接口复位控制位。	0
29	RSTGPDMA	GPDMA 功能复位控制位。	0
30	RSTENET	以太网模块复位控制位。	0
31	RSTUSB	USB 接口复位控制位。	0

3.5.2 复位控制寄存器 1（RSTCON1—0x400F C1D0）

采用 RSTCON1 寄存器，可以给更多的外设一个硬件复位，见下面的表 12。

表12. 复位控制寄存器 1（RSTCON1—0x400F C1D0）位描述

位	符号	描述	复位值
0	RSTIOCON	IOCON 寄存器复位控制位。	0
1	RSTDAC	D/A 转换器（DAC）复位控制位。	0
2	RSTCANACC	CAN 接收滤波器复位控制位。	0
31:3	-	保留。读取值未定义，只写入 0。	无

3.6 掉电检测

LPC178x/177x 带有掉电检测器 (BOD)，提供了对 $V_{DD(REG)(3V3)}$ 管脚电压的两级监控。如果该管脚电压低于 BOD 中断触发电平(标称室温条件下的典型值为 2.2V)，BOD 会向 NVIC 发出一个中断信号。这个信号可以将 NVIC 中断使能寄存器中的中断使能，从而产生一个 CPU 中断；否则，软件通过读取原始中断状态寄存器，监控该信号。

当管脚上的电压低于 BOD 复位触发电平时（标称室温条件下的典型值为 1.85V），第二级掉电检测发出一个复位信号，停止 LPC178x/177x 的工作。这个复位可防止对 Flash 的修改，因为电压低的情况下，芯片中各部件的运行将变得不可靠。BOD 电路会将这个复位保持在低于 1V，此时上电复位电路维持着整体的复位。

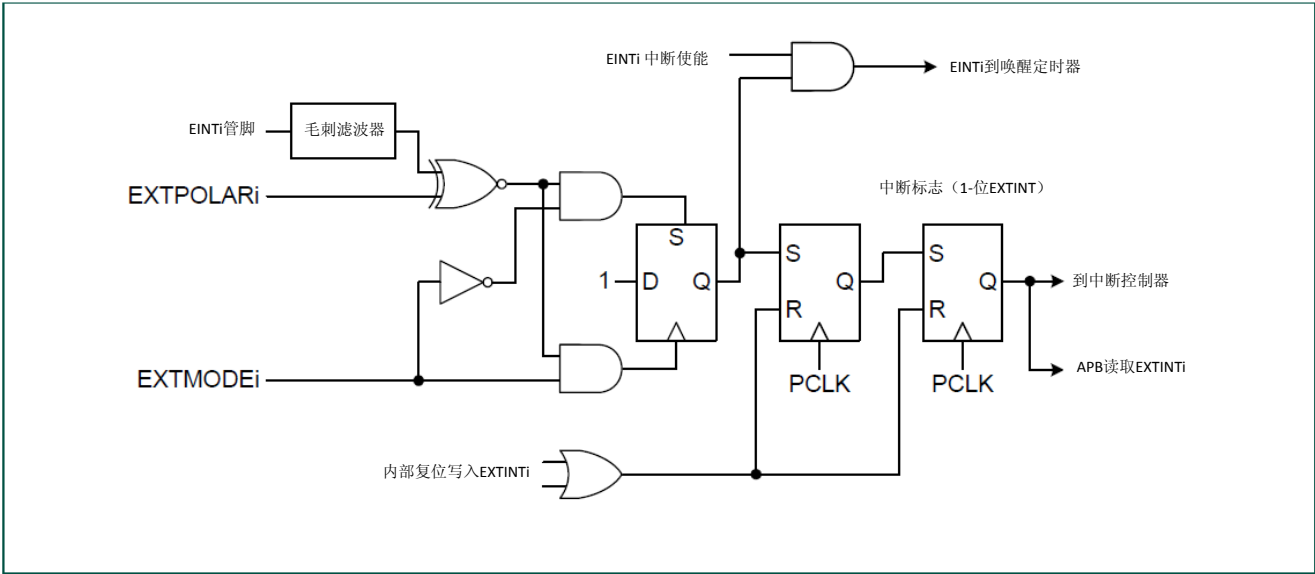
BOD 复位中断电平与 BOD 复位触发电平阈值这两者都含有一些滞后。正常工作情况下，这个滞后能使 BOD 复位中断电平检测可靠的中断，或让一个定期执行的事件循环来探测条件。

但当掉电检测被使能，将 LPC178x/177x 带出掉电模式时（这本身并不是一个有保证的操作——见 4.7.8 节，则在唤醒定时器完成这个延时以前，电源电压就可能从一个瞬变中恢复。此时，瞬变 BOD 的净结果是，在设定掉电模式的指令以后，部件就被唤醒并继续工作，而无需任何中断的发生，以及 RSID 中的 BOD 位为 0。由于所有其它唤醒条件都锁存标志（见 3.7.2 节以及 29.6.2 节），因此无需任何明确的原因，这种类型的唤醒就可以假设为一个已经消失的掉电。

3.7 外部中断输入

LPC178x/177x 有 4 个外部中断输入，作为可选的管脚功能。[图 6](#) 显示了一个外部中断的逻辑。另外，外部中断能够将 CPU 从掉电模式中唤醒。详见 [4.7.6](#) 节。

图6. 外部中断逻辑图



3.7.1 寄存器描述

外部中断功能有 4 个相关的寄存器。EXTINT 寄存器包含中断标志。EXTMODE 与 EXTPOLAR 寄存器设定了电平或边沿触发参数。

表13. 外部中断寄存器

名称	描述	访问	复位值 ^[1]	地址
EXTINT	外部中断标志寄存器包含 EINT0、EINT1、EINT2 和 EINT3 的中断标志。参见表 14。	R/W	0x00	0x400F C140
EXTMODE	外部中断模式寄存器控制每个管脚为边沿触发还是电平触发。参见表 15。	R/W	0x00	0x400F C148
EXTPOLAR	外部中断极性寄存器控制每个管脚为哪种电平或边沿触发。参见表 16。	R/W	0x00	0x400F C14C

[1] 复位值仅反映已使用位中保存的数据，它不包括保留位的内容。

3.7.2 外部中断标志寄存器（EXTINT—0x400F C140）

当一个管脚选择使用外部中断功能时，该管脚上的电平或边沿（通过 EXTPOLAR 和 EXTMODE 寄存器中的相应位决定）将会置位 EXTINT 寄存器的中断标志。这会向 NVIC 提交相应的中断请求，如果管脚中断被能，则将产生中断。

通过向 EXTINT 寄存器中的 EINT0~EINT3 中写入 1 来将其清零。在电平触发模式下，中断只有在管脚处于无效状态时才能被清零。

一旦 EINT0~EINT3 中的一位被设置，且相应代码开始执行（处理唤醒中断和/或外部中断），则 EXTINT 寄存器中的相应位必须清除。否则，在后面 EINT 管脚上触发的事件不会被识别。

重要提示：只要外部中断操作模式（有效电平/边沿）发生变化（包括对外部中断的初始化），则 EXTINT 寄存器中的相应位都必须被清零！详见 3.7.3 节及 3.7.4 节。

例如，如果外部 0 管脚的低电平将系统从掉电模式中唤醒，那么为了将来还能进入掉电模式，唤醒后的程序必须将 EINT0 位复位。如果 EINT0 位仍保持 1，则以后唤醒掉电模式的任何操作都将失败。相同情况也适用于外部中断处理。

有关掉电模式的详情将在以下章节中讨论。

表14. 外部中断标志寄存器（EXTINT—0x400F C140）位描述

位	符号	描述	复位值
0	EINT0	在电平触发模式下，如果管脚的 EINT0 功能被选用且管脚处于有效状态时，该位置位。在边沿触发模式下，如果管脚的 EINT0 功能被选用且管脚上出现所属边沿时，该位置位。 该位通过写入 1 来清零，但在电平触发模式下管脚处于有效状态的情况除外。 ^[1]	0
1	EINT1	在电平触发模式下，如果管脚的 EINT1 功能被选用且管脚处于有效状态时，该位置位。在边沿触发模式下，如果管脚的 EINT1 功能被选用且管脚上出现所属边沿时，该位置位。 该位通过写入 1 来清零，但在电平触发模式下管脚处于有效状态的情况除外。 ^[1]	0
2	EINT2	在电平触发模式下，如果管脚的 EINT2 功能被选用且管脚处于有效状态时，该位置位。在边沿触发模式下，如果管脚的 EINT2 功能被选用且管脚上出现所属边沿时，该位置位。 该位通过写入 1 来清零，但在电平触发模式下管脚处于有效状态的情况除外。 ^[1]	0
3	EINT3	在电平触发模式下，如果管脚的 EINT3 功能被选用且管脚处于有效状态时，该位置位。在边沿触发模式下，如果管脚的 EINT3 功能被选用且管脚上出现所属边沿时，该位置位。 该位通过写入 1 来清零，但在电平触发模式下管脚处于有效状态的情况除外。 ^[1]	0
31:4	-	保留。读取值未定义，只写入 0。	无

[1] 例如，如果选择 EINTx 为低电平触发且在相应的管脚上出现低电平时，该位不能被清零；该位仅在该管脚上的信号变为高电平时才被清零。

3.7.3 外部中断模式寄存器（EXTMODE—0x400F C148）

本寄存器中各个位用来选择各个 EINT 管脚是电平触发还是边沿触发。只有那些选择为 EINT 功能的管脚（见 8.3 节）并在相应 NVIC 寄存器中使能了相应中断，才能产生外部中断。（当然，如果管脚选择用作其它功能，则产生其它功能的中断）。

注：当中断在 NVIC 中被禁能时（可在各个 ISERN/ICERN 寄存器中读出状态），软件应该只改变此寄存器中相应位的值，并且应在使能（初始化）和重新使能中断以前，向 EXTINT 相应位写入 1。通过改变模式和不清零 EXTINT 位，可以设置外部中断。

表15. 外部中断模式寄存器（EXTMODE—0x400F C148）位描述

位	符号	值	描述	复位值
0	EXTMODE0	0	$\overline{\text{EINT0}}$ 选择电平触发。	0
		1	$\overline{\text{EINT0}}$ 为边沿触发。	
1	EXTMODE1	0	$\overline{\text{EINT1}}$ 选择电平触发。	0
		1	$\overline{\text{EINT1}}$ 为边沿触发。	
2	EXTMODE2	0	$\overline{\text{EINT2}}$ 选择电平触发。	0
		1	$\overline{\text{EINT2}}$ 为边沿触发。	
3	EXTMODE3	0	$\overline{\text{EINT3}}$ 选择电平触发。	0
		1	$\overline{\text{EINT3}}$ 为边沿触发。	
31:4 -			保留。读取值未定义，只写入 0。	无

3.7.4 外部中断极性寄存器（EXTPOLAR—0x400F C14C）

在电平触发模式中，本寄存器中的各个位用于选择相应的管脚是高电平有效还是低电平有效。在边沿触发模式中，它们用于选择管脚是上升沿触发还是下降沿触发。只有管脚选择用作 EINT 功能（见 8.3 节），并且在相应 NVIC 寄存器中使能，外部中断功能才能产生中断。（当然，如果管脚选择用作其它功能，则产生其它功能的中断）。

注：当中断在 NVIC 中被禁能时（可在各个 ISER/ICER 寄存器中读出状态），软件只应改变此寄存器中相应位的值，并且应在使能（初始化）或重新使能中断以前，向 EXTINT 相应位写入 1。通过改变模式而不清零 EXTINT 位，可以设置外部中断。

表16. 外部中断极性寄存器（EXTPOLAR—0x400F C14C）位描述

位	符号	值	描述	复位值
0	EXTPOLAR0	0	$\overline{\text{EINT0}}$ 为低电平有效或下降沿触发（由 EXTMODE0 决定）。	0
		1	$\overline{\text{EINT0}}$ 为高电平有效或上升沿触发（由 EXTMODE0 决定）。	
1	EXTPOLAR1	0	$\overline{\text{EINT1}}$ 为低电平有效或下降沿触发（由 EXTMODE1 决定）。	0
		1	$\overline{\text{EINT1}}$ 为高电平有效或上升沿触发（由 EXTMODE1 决定）。	
2	EXTPOLAR2	0	$\overline{\text{EINT2}}$ 为低电平有效或下降沿触发（由 EXTMODE2 决定）。	0
		1	$\overline{\text{EINT2}}$ 为高电平有效或上升沿触发（由 EXTMODE2 决定）。	
3	EXTPOLAR3	0	$\overline{\text{EINT3}}$ 为低电平有效或下降沿触发（由 EXTMODE3 决定）。	0
		1	$\overline{\text{EINT3}}$ 为高电平有效或上升沿触发（由 EXTMODE3 决定）。	
31:4	-		保留。读取值未定义，只写入 0。	无

3.8 其它系统控制与状态标志

下面列出了控制 LPC178x/177x 的一些操作不适用于外设或其他寄存器操作的某些方面。

3.8.1 系统控制与状态寄存器（SCS—0x400F C1A0）

SCS 寄存器包含了与芯片运行各方面有关的特殊控制与状态位。这些功能说明见[表 17](#)。

其中几个位适用于主振荡器。由于芯片开始工作时总是采用内部 RC 振荡器，而主振荡器可能并非用于所有应用中，因此主振荡器只有应软件请求才起振。实现方式是设置 SCS 寄存器中的 OSCEN 位，如表 3-13 所示。主振荡器提供了一个状态标志（SCS 寄存器中的 OSCSTAT 位），这样软件就可以确定振荡器何时运行并且稳定。此时，软件可以控制将主振荡器切换为时钟源。在起振主振荡器以前，必须通过配置 SCS 寄存器中的 OSCRANGE 位，选择一个频率范围。

表17. 系统控制和状态寄存器（SCS—0x400F C1A0）位描述

位	功能	值	描述	访问	复位值
0	EMC 转换控制		控制如何在 EMC 地址管脚上为静态存储器输出地址。 也可参见 EMC 章 10.9 节。	R/W	1
		0	静态存储器地址通过转换以匹配数据总线宽度。例如，当访问 32 位宽的数据总线时，该地址被向右转换 2 个位以使位 2 成为最低有效位。在这种模式下，该设备上的地址位 0 与存储器上的地址位 0 相连，进而简化了存储器连接。这也可能扩大存储器地址范围，因为附加的高位地址位可以在转换后显示在更高的地址管脚上。		
		1	不管数据总线宽度是多少，静态存储器地址总是作为字节地址被输出。例如，当在 32 位总线上访问字数据时，地址位 1 和 0 都将固定为 0。在这种模式下，一个或两个较低地址位可能没有与作为宽超过 8 位的总线一部分的存储器相连。该模式与设备 LPC23xx 和 LPC24xx 的操作相匹配。		
1	EMC 复位去使能 ^[1]		外部存储器控制器复位去使能。也可参见 EMC 章 10.8 节。	R/W	0
		0	当发生任何类型的芯片复位事件时，两个 EMC 复位均有效。在这种模式下，只要有复位发生，所有的 EMC 寄存器和功能都将被初始化。		
		1	很多 EMC 只能由上电或掉电事件进行复位，以使 EMC 可以通过热复位（外部复位或看门狗复位）维持其状态。如果 EMC 配置正确，可以通过热复位保持自动恢复。		
2	EMC 突发控制		外部存储器控制器突发控制。也可参见 EMC 章 10.10 节。	R/W	0
		0	突发使能。		
		1	突发去使能。该模式可以用于防止对内存映射 I/O 设备（连接到 EMC 静态存储器片选上）的多个顺序存储。这些未经请求的访问可能导致一些 I/O 设备出现问题。		

位	功能	值	描述	访问	复位值
3	MCIPWR 有效电平 ^[1]		选择 SD 卡接口信号 SD_PWR 的有效电平。	R/W	1
		0	SD_PWR 低电平有效（SD 卡接口模块的反向输出）。		
		1	SD_PWR 高电平有效（伴随 SD 卡接口模块的输出）。		
4	振荡器范围选择		主振荡器的范围选择。	R/W	0
		0	主振荡器的频率范围是 1MHz 到 20MHz。		
		1	主振荡器的频率范围是 15MHz 到 25MHz。		
5	振荡器使能		主振荡器使能。	R/W	0
		0	主振荡器被禁能。		
		1	主振荡器使能，并且在正确的外部电路连接到 XTAL1 和 XTAL2 引脚的情况下启动。		
6	振荡器状态		主振荡器状态。	RO	0
		0	主振荡器不稳定，不能用作时钟源。		
		1	主振荡器已稳定，能够用作时钟源。主振荡器必须通过 OSCEN 位使能。		
31:7 -			保留。读取值未定义，只写入 0。	-	无

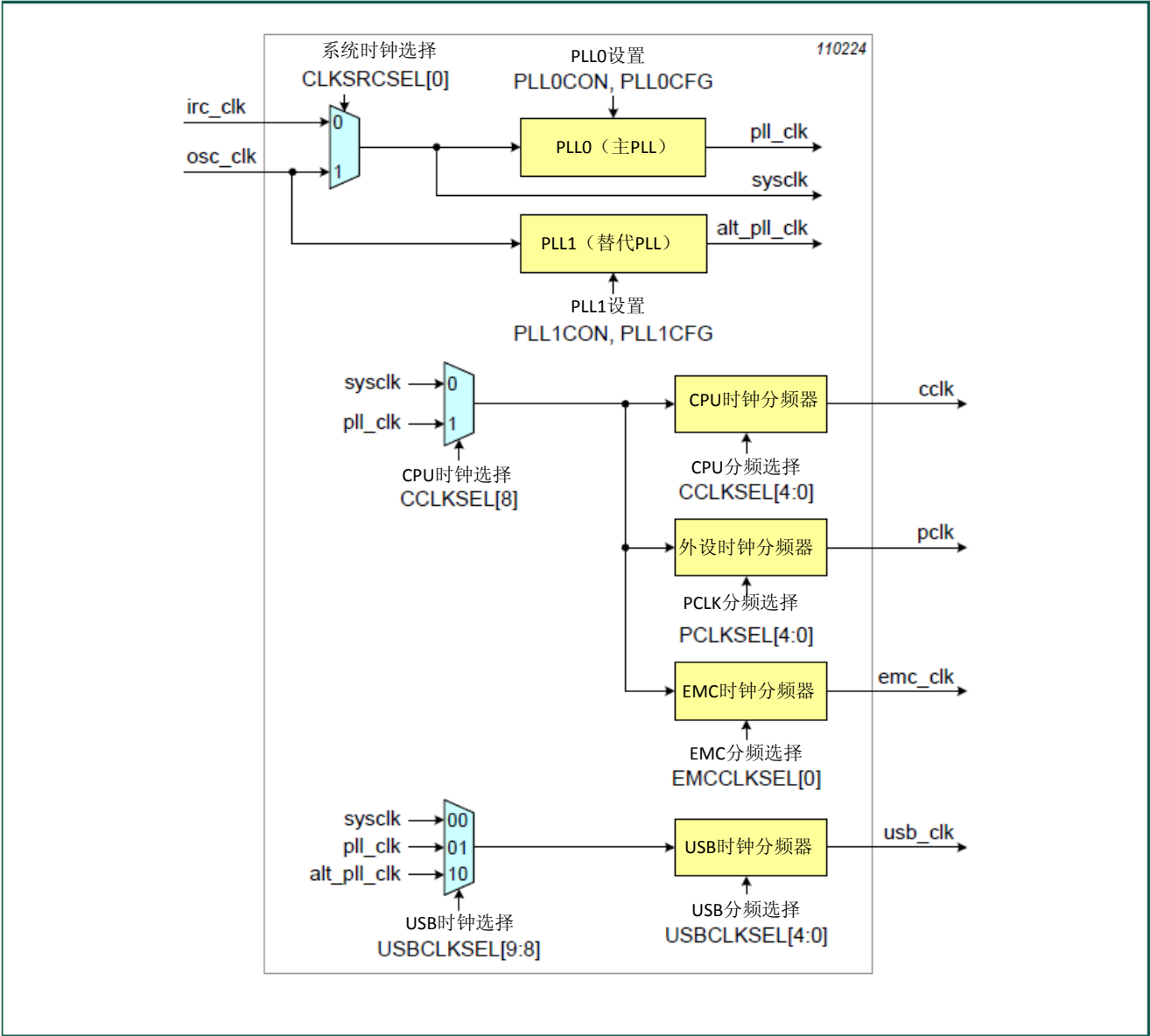
[1] 该位的状态通过软件复位进行保存，并且只有 POR 或 BOD 事件可以将该位复位为默认值。

4.1 计时与功率控制功能汇总

本节描述了 LPC178x/177x 所需要的各种时钟和时钟源选择，以及功率控制和低功耗模式的唤醒。以下章节将描述的功能包括：

- 振荡器（见 [4.3](#) 节）
- 锁相环（PLL）（见 [4.5](#) 节）
- 时钟选择与分频器（见 [4.6](#) 节）
- 功率控制（见 [4.7](#) 节）
- 唤醒定时器（见 [4.8](#) 节）
- 外部时钟输出（见 [4.9](#) 节）

图7. LPC178x/177x 时钟生成



4.2 寄存器描述

所有寄存器无论大小，均按照字地址边界排列。寄存器的详细内容请见各功能描述。

表18. 系统控制寄存器汇总

名称	描述	访问	复位值	地址	表
系统时钟源选择					
CLKSRCSEL	时钟源选择寄存器	R/W	0	0x400F C10C	表 21
锁相环 0 (PLL0,主 PLL)					
PLL0CON	PLL0 控制寄存器	R/W	0	0x400F C080	表 23
PLL0CFG	PLL0 配置寄存器	R/W	0	0x400F C084	表 24
PLL0STAT	PLL0 状态寄存器	RO	0	0x400F C088	表 25
PLL0FEED	PLL0 馈送寄存器	WO	无	0x400F C08C	表 26
锁相环 1 (PLL1, 副 PLL)					
PLL1CON	PLL1 控制寄存器	R/W	0	0x400F C0A0	表 23
PLL1CFG	PLL1 配置寄存器	R/W	0	0x400F C0A4	表 24
PLL1STAT	PLL1 状态寄存器	RO	0	0x400F C0A8	表 25
PLL1FEED	PLL1 馈送寄存器	WO	无	0x400F C0AC	表 26
时钟分频器					
CCLKSEL	CPU 时钟选择寄存器	R/W	1	0x400F C104	表 30
USBCLKSEL	USB 时钟选择寄存器	R/W	0	0x400F C108	表 31
EMCCLKSEL	外部存储控制器时钟选择寄存器	R/W	0	0X400F C100	表 32
PCLKSEL	外设时钟选择寄存器	R/W	0x10	0x400F C1A8	表 33
功率控制					
PCON	功率控制寄存器	R/W	0	0x400F C0C0	表 35
PCONP	外设功率控制	R/W	0x0408 829E	0x400F C0C4	表 37
应用					
CLKOUTCFG	时钟输出配置寄存器	R/W	0	0x400F C1C8	表 40

4.3 振荡器

LPC178x/177x 有 3 个独立的振荡器。它们是主振荡器、内部 RC 振荡器、RTC 振荡器。每个振荡器都可以根据特定的应用需求用于一个或多个用途。如[图 7](#)所示。

复位后，LPC178x/177x 将用内部 RC 振荡器运行，直到被软件切换。这样系统就能在没有任何外部晶振的情况下运行，并且 Boot Loader 代码能够在已知的频率下操作。

4.3.1 内部 RC 振荡器

内部 RC 振荡器（IRC）可以用作时钟，驱动 PLL0 并继后驱动 CPU。IRC 的精度达不到 USB 接口的时间基准精度要求，后者需要更精确时基才能符合 USB 规范（只有主振荡器可以满足这个规范）。另外，如果 CAN 波特率高于 100kBit/s，则 IRC 不应用于 CAN1/2 模块。IRC 的额定频率为 12MHz，出厂时校验的偏差在 $\pm 1\%$ 内。

一旦上电或任何芯片复位，LPC178x/177x 就会将 IRC 用做时钟源。软件可以稍后切换到其它可用的时钟源。

4.3.2 主振荡器

主振荡器可以用做 CPU 的时钟源，可以使用或不使用 PLL0。主振荡器工作频率为 1MHz~25MHz。该频率可通过主 PLL（PLL0）来提高，其值可高达 CPU 操作频率的最大值。振荡器的输出被称作 OSC_CLK。在本文其它地方的速率公式中，PLLCLKIN 选作 PLL0 输入时钟，ARM 处理器的时钟频率则称作 CCLK。在 PLL0 有效且已连接，PLLCLKIN 与 CCLK 的频率相同。详见[4.5](#)节。

LPC178x/177x 板上振荡器可以在以下两种模式下工作：从属模式和振荡模式。

在从属模式下，输入时钟信号应该与一个 100pF 电容（[图 8a](#) 中 C_C ）相连，其幅值为 200mVrms 与 1000mVrms。这相当于一个信号摆幅在 280mV 与 1.4V 之间的方波信号。这种配置中，XTAL2 引脚可以不连接。

振荡模式下使用的外部元件与模型见[图 8b](#)和[图 8c](#)，以及[表 19](#)与[表 20](#)。由于片上集成了反馈电阻，只需要外接一个晶体和电容 C_{X1} 与 C_{X2} ，就可以形成基本模式的振荡电路（基础频率由 L 、 C_L 和 R_S 表示）。[图 8c](#) 中的电容 C_P 表示并联封装电容，其值不应大于 7pF。参数 F_C 、 C_L 、 R_S 和 C_P 由晶体制造商提供。

图8. 振荡器模式和模型：（a）从属模式，（b）振荡模式，（c）外部晶振模型（用来评估 $C_{x1/x2}$ 的值）

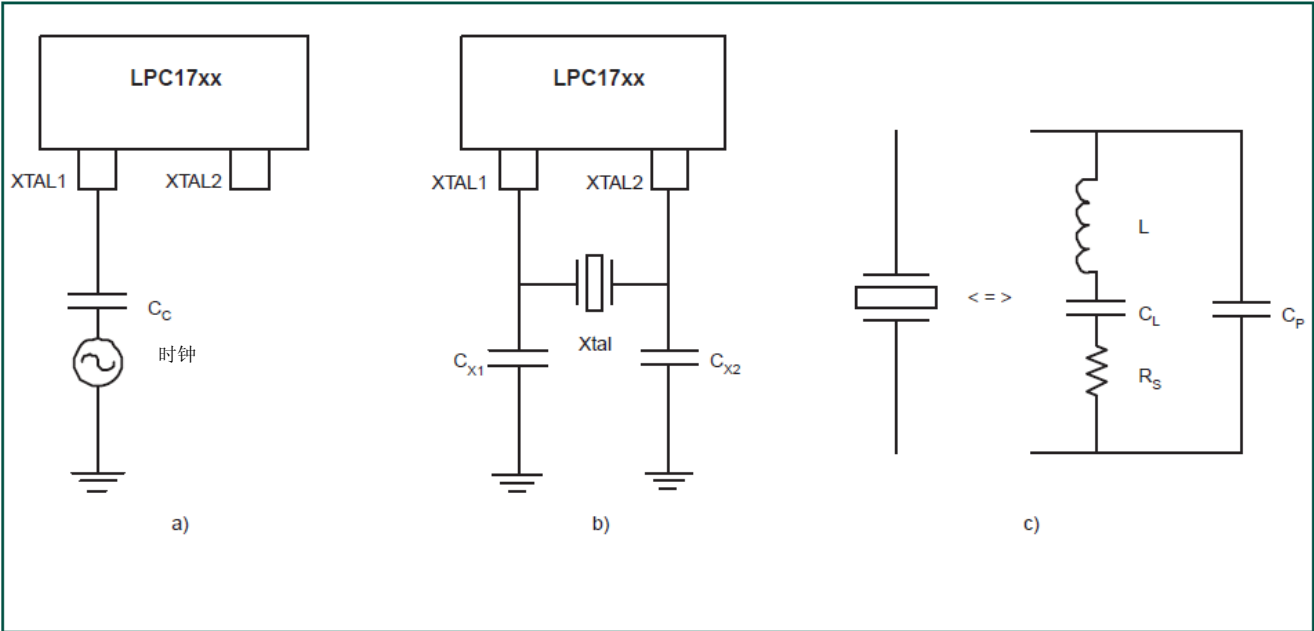


表19. 振荡模式下 $C_{x1/x2}$ 的建议取值（晶体和外部元件参数）低频模式（ $OSCRANGE = 0$ ，参见表17）

基本振荡频率 F_{OSC}	晶体负载电容 C_L	最大晶体串联电阻 R_S	外部负载电容 C_{x1}, C_{x2}
1MHz—5MHz	10pF	< 300 Ω	18pF, 18pF
	20pF	< 300 Ω	39pF, 39pF
	30pF	< 300 Ω	57pF, 57pF
5MHz—10MHz	10pF	< 300 Ω	18pF, 18pF
	20pF	< 200 Ω	39pF, 39pF
	30pF	< 100 Ω	57pF, 57pF
10MHz—15MHz	10pF	< 160 Ω	18pF, 18pF
	20pF	< 60 Ω	39pF, 39pF
15MHz—20MHz	10pF	< 80 Ω	18pF, 18pF

表20. 振荡模式下 $C_{x1/x2}$ 的建议取值（晶体和外部元件参数）高频模式（ $OSCRANGE = 1$ ，参见表17）

基本振荡频率 F_{OSC}	晶体负载电容 C_L	最大晶体串联电阻 R_S	外部负载电容 C_{x1}, C_{x2}
15MHz—20MHz	10pF	< 180 Ω	18pF, 18pF
	20pF	< 100 Ω	39pF, 39pF
20MHz—25MHz	10pF	< 160 Ω	18pF, 18pF
	20pF	< 80 Ω	39pF, 39pF

4.3.2.1 主振荡器的起振

由于芯片总是从内部 RC 振荡器开始工作，而有些应用可能根本就不用主振荡器，因此主振荡器只会应软件的请求而启动。实现方法是设定 SCS 寄存器中的 OSCEN 位，见[表 17](#)中的说明。主振荡器提供了一个状态标志（SCS 寄存器中的 OSCSTAT 位），这样软件就可以确定何时振荡器在运行且稳定。此时，软件可以控制切换到主振荡器，使其作为时钟源。在启动主振荡器以前，必须通过配置 SCS 寄存器中的 OSCRANGE 位，选择一个频率范围。

4.3.3 RTC 振荡器

RTC 振荡器为 RTC 提供 1Hz 到 32kHz 时钟输出，后者可以在 CLKOUT 管脚被输出，从而可以无需探头介入而调节 RTC 振荡器。

4.3.4 看门狗振荡器

看门狗定时器有一个专用的振荡器，为看门狗定时器提供一个 500kHz 的时钟，只要看门狗定时器被使能，这个时钟就永远运行。看门狗振荡时钟可以从 CLKOUT 管脚输出，从而可以观测其频率。

为了使看门狗定时器工作的功耗最低（这在低功耗模式下很关键），看门狗振荡器的频率并未受到严格控制。对某个器件，看门狗振荡器的频率会随温度和供电变化而变化，而不同器件之间则会随工艺而改变。在确定看门狗的重新装载值时，应考虑到这种变化。

对某只器件，温度与电源的变化可能产生 $\pm 17\%$ 的频率变动。相同工作条件下，不同器件之间的频率变动可以达到 $\pm 30\%$ 。

4.4 时钟源选择多路复用

两种时钟源可以选择用来驱动系统时钟（sysclk）和 PLL0。它们是内部 RC 振荡器和主振荡器。

只有在 PLL0 断开连接时，才可更换输入时钟源。如何在系统中如何使用 PLL0 改变时钟源，详见 [4.5.11](#) 节。

4.4.1 时钟源选择寄存器（CLKSRCSEL—0x400F C10C）

CLKSRCSEL 寄存器用于控制 sysclk 和 PLL0 所使用的时钟选择。

表21. 时钟源选择寄存器位描述

位	符号	值	描述	复位值
0	CLKSRC		如下选择系统时钟和 PLL0 的时钟源：	0
		0	选择内部 RC 振荡器作为系统时钟和 PLL0 的时钟源（默认）。	
		1	选择主振荡器作为系统时钟和 PLL0 的时钟源。	
31:1	-		保留。读取值未定义，只写入 0。	无

4.5 PLL0 与 PLL1（锁相环）

PLL0（亦称主 PLL）与 PLL1（亦称副 PLL）的功能相同，但输入可能方式与输出连接方面有略微差异。这些可能方式显示在图 7 中。主 PLL 可以接受来自 IRC 或主振荡器的输入，并且可以为器件上几乎所有部分提供时钟。副 PLL 则只能接受主振荡器的输入，并作为 USB 的替代时钟源。主 PLL 并非总能满足这种外设的计时要求。

复位时两种 PLL 均被禁用和断电。如果副 PLL 被禁用，USB 时钟则可由 PLL0 提供，前提是设置好通过该路径为 USB 提供 48MHz 的时钟。每个时钟的来源都必须通过 CLKSEL 寄存器来选择（见 4.6 节），并可以根据需要，用时钟分频器进一步降低频率。

PLL 的激活由 PLLCON 寄存器控制。PLL 倍频器与分频器值由 PLLCFG 寄存器控制。PLLCFG 寄存器被保护，以防止 PLL 意外关闭或意外改变 PLL 工作参数。该保护通过一个近似于看门狗定时器的馈送序列实现。详见 PLLFEED 寄存器的说明。

PLL0 可接受 IRC 或主振荡器的输入时钟频率。如果只使用主 PLL，则其输出频率必须是系统需要的所有其它时钟的某个整数倍。PLL1 只能从主振荡器获得输入，需要一只 10~25MHz 范围的外部晶振。对于每个 PLL，电流控制振荡器(CCO)的工作频率是 156MHz~320MHz，因此需要额外的分频器，将其输出降低到所需频率。输出分频器的最小值为 2，保证了 PLL 的输出有 50% 占空比。图 9 显示了 PLL 内部连接的方框图。

如果使用了 USB，则 CPU 时钟和其它时钟的可能性就受到了限制，因为 USB 时钟要求有精确的频率和极低的抖动，且 PLL0 输出必须是 48MHz 的倍数。落在 PLL 工作频率范围内的 48MHz 的偶数倍是 192 和 288MHz。另外，只有主振荡器与 PLL 一起使用时，才能满足 USB 的精度与抖动规范。正是基于这些限制，才提供了副 PLL。

副 PLL 仅接受来自主振荡器 10MHz~25MHz 范围内的输入时钟频率。当用做 USB 时钟时，该输入频率被倍增至一个 48MHz 的倍数（如上述的 192 或 288MHz）。

4.5.1 PLL 与启动/引导代码的相互关系

当在用户 Flash 中没有有效代码（由校验和字段决定），或在启动时拉低 ISP 使能引脚（P2[10]）时，芯片将进入 ISP 模式，且引导代码将用 IRC 设置 PLL。因此，当用户启动 JTAG 来调试应用代码时，不能假定主 PLL 被禁能。用户的启动代码必须遵循本章所述步骤，断开主 PLL。

4.5.2 PLL 寄存器描述

PLL 受控于表 22 中所显示的寄存器。更多详细描述如下。

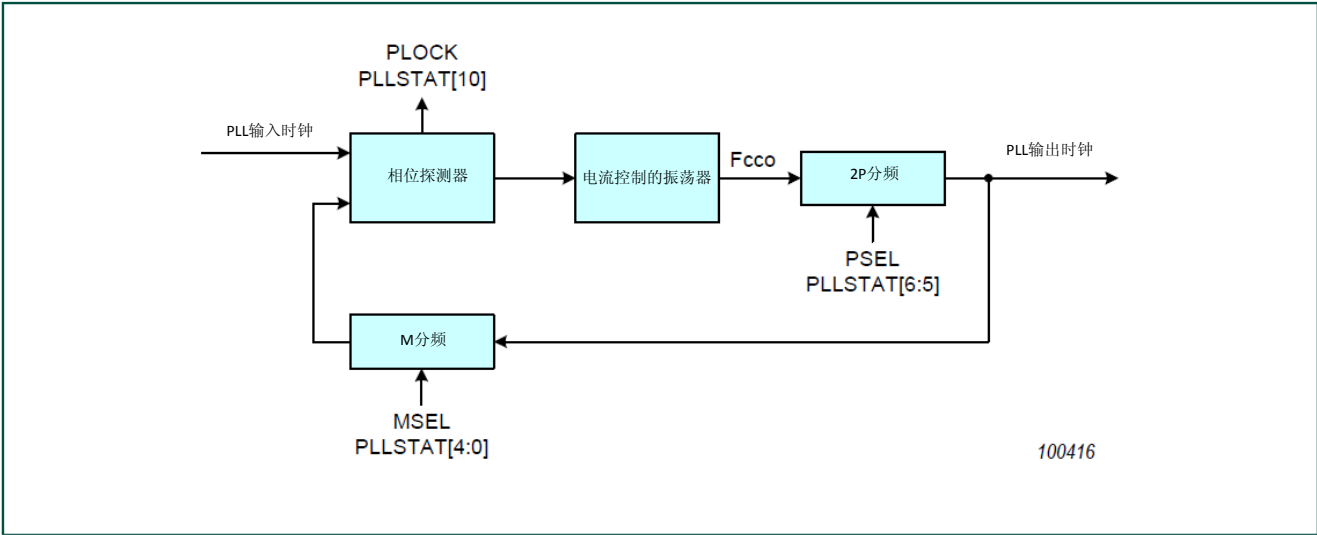
警告：PLL 值设置不正确可能导致 USB 子系统不能正常工作！

表22. PLL1 寄存器

名称	描述	访问	复位值 ^[1]	PLLn 寄存器名称和地址	表
PLLCON	PLL 控制寄存器。保存更新 PLL 控制位的寄存器。写入该寄存器的值只有在发生有效的 PLL 馈送序列后才起作用。	R/W	0	PLL0CON—0x400F C080 PLL1CON—0x400F C0A0	表 23
PLLCFG	PLL 配置寄存器。保存更新 PLL 配置值的寄存器。写入该寄存器的值只有在发生有效的 PLL 馈送序列后才起作用。	R/W	0	PLL0CFG—0x400F C084 PLL1CFG—0x400F C0A4	表 24
PLLSTAT	PLL 状态寄存器。PLL 控制和配置信息的读回寄存器。如果对 PLLCON 或 PLLCFG 执行了写操作,但没有发生 PLL 馈送序列,则这些值将不会反映当前 PLL 的状态。读取该寄存器提供了控制 PLL 和 PLL 状态的实际值。	RO	0	PLL0STAT—0x400F C088 PLL1STAT—0x400F C0A8	表 25
PLLFEED	PLL 馈送寄存器。该寄存器使能 PLL 控制和配置信息的装载,它将 PLLCON 和 PLLCFG 寄存器中的 PLL 控制和配置信息载入到实际影响 PLL 操作的映像寄存器。	WO	无	PLL0FEED-0x400F C08C PLL1FEED-0x400F C0AC	表 26

[1] 复位值仅反映在使用位中保存的数据，它不包括保留位的内容。

图9. PLL1 模块方框图



4.5.3 PLL 控制寄存器（PLL0CON—0x400F C080, PLL1CON—0x400F C0A0）

PLLCON 寄存器包含了使能与连接每个 PLL 的位。使能一个 PLL 后，它就能尝试锁定倍频器与分频器的当前设定值。对于 PLLCON 寄存器的更改，只有在执行了正确的 PLL 馈送序列后才生效（见 4.5.7 节和 4.5.4 节）。

表23. PLL 控制寄存器（PLL0CON—0x400F C080, PLL1CON—0x400F C0A0）位描述

位	符号	描述	复位值
0	PLLE	PLL 使能。当该位为 1 并且在有效的 PLL 馈送之后，该位将激活 PLL 并允许其锁定到指定的频率。参见 PLLSTAT 寄存器，表 25。	0
31:1	-	保留。读取值未定义，只写入 0。	无

每个 PLL 都必须被设置、使能，以及建立了锁定后，才能用作一个时钟源。在 PLL 被选择以前，硬件并不能保证 PLL 的锁定，如果 PLL 脱离锁定状态，硬件也不会自动断开 PLL。

4.5.4 PLL 配置寄存器（PLL0CFG—0x400F C084, PLL1CFG—0x400F C0A4）

PLLCFG 寄存器包含了 PLL 倍频器与分频器值。在被赋予一个正确 PLL 馈送序列以前，改变一个 PLLCFG 寄存器并不会生效（见 4.5.7 节）。有关 PLL 频率、倍频器和分频器值，见 4.5.10 节。

表24. PLL 配置寄存器（PLL0CFG—0x400F C084, PLL1CFG—0x400F C0A4）位描述

位	符号	描述	复位值
4:0	MSEL	PLL 倍频器值。在 PLL 频率计算中提供“M”值。 注：有关 MSEL 正确值的选取，参见 4.5.9 节。	0
6:5	PSEL	PLL 分频器值。在 PLL 频率计算中提供“P”值。 注：有关 PSEL 正确值的选取，参见 4.5.9 节。	0
31:7	-	保留。读取值未定义，只写入 0。	无

4.5.5 PLL 状态寄存器（PLL0STAT—0x400F C088, PLL1STAT—0x400F C0A8）

只读的 PLLSTAT 寄存器提供了当前生效的 PLL 真实工作参数和状态。PLLSTAT 可能与 PLLCON 和 PLLCFG 中的值不同，因为对这两个寄存器的更改只有在执行了正确的 PLL 馈送序列后才生效（见 4.5.7 节）。

表25. PLL 状态寄存器（PLL0STAT—0x400F C088, PLL1STAT—0x400F C0A8）位描述

位	符号	描述	复位值
4:0	MSEL	读回 PLL 倍频器值。这是 PLL 当前使用的值。	0
6:5	PSEL	读回 PLL 分频器值。这是 PLL 当前使用的值。	0
7	-	保留。从保留位读出的值未被定义。	无
8	PLLE_STAT	读回 PLL 使能位。当该位为 1 时，PLL 处于激活状态。 当该位为 0 时，PLL 关闭。当进入掉电模式时，该位自动清零。	0
9	-	保留。从保留位读出的值未被定义。	无

位	符号	描述	复位值
10	PLOCK	反映 PLL 的锁定状态。当该位为 0 时，PLL 未锁定。当该位为 1 时，PLL 锁定到指定的频率。	0
31:11	-	保留。从保留位读出的值未被定义。	无

4.5.6 PLL 中断：PLOCK0 与 PLOCK1

PLLSTAT 寄存器中的 PLOCK 位用于表示相关 PLL1 的锁定状态。当 PLL 首次被使能时，或当其参数被改变时，PLL 需要一些时间来建立新条件下的锁定。通过监控相关的 PLOCK 位，就可以确定何时可以连接并使用 PLL。

每个 PLOCK 位都接到中断控制器上。这样，软件就能使能 PLL 完成其它功能，而无需等待 PLL 锁定。当中断发生时，PLL 可以被选择为一个时钟源，然后禁止中断。PLOCK0 和 PLOCK1 在表 43 中分别被表示为特殊数字 32 和 48。注意，一旦相关 PLL 被锁定，则每个 PLOCK 位都保持有效状态，因此如果使用中断，则中断服务程序必须在退出前禁止中断。

4.5.7 PLL 馈送寄存器(PLL0FEED—0x400F C08C, PLL1FEED—0x400F C0AC)

必须将正确的馈送序列写入 PLLFEED 寄存器，才能让相关的 PLLCON 与 PLLCFG 寄存器生效。馈送序列为：

- 1. 向 PLLFEED 中写入值 0xAA。
- 2. 向 PLLFEED 中写入值 0x55。

两次写入的顺序必须正确，两者之间的相同地址空间（0x400F C000 到 0x400F FFFF）内不得有其它寄存器的访问。因此，如果中断服务程序在对这个空间内的其它寄存器执行写操作，则有在执行 PLL 馈送操作时必须禁止中断。如果两个馈送值中有任何一个不正确，或前述条件之一未得到满足，则对 PLLCON 或 PLLCFG 寄存器的任何修改都不会生效。

表26. PLL 馈送寄存器（PLL0FEED—0x400F C08C, PLL1FEED—0x400F C0AC）位描述

位	符号	描述	复位值
7:0	PLLFEED	PLL 馈送序列必须写入该寄存器才能使 PLL 配置和控制寄存器的更改生效。	0x00
31:8	-	保留。读取值未定义，只写入 0。	无

4.5.8 PLL 与掉电模式

掉电模式会自动关闭并断开已激活的 PLL。从掉电模式的唤醒则不会自动恢复 PLL 设置。恢复 PLL 工作必须由软件完成。一般过程是：一个程序激活 PLL，等待锁定，然后可以在任何中断服务程序开始时调用 PLL，该中断服务程序可以在掉电唤醒时被调用。

如果未选择用 USB 数据线上的活动将微控制器从掉电模式下唤醒（从低功耗模式下唤醒详见 4.7.9 节），则当掉电模式被激活时，主 PLL（PLL0）与副 PLL（PLL1）都将如上所述自动关闭并断开。不过，如果 USB 活动中断被使能，并且 `USB_NEED_CLK = 1`（`USB_NEED_CLK` 的说明见表 255），则不能进入掉电模式，并且任何试图设置 PD 位的尝试都会失败，PLL 保持在当前状态。

4.5.9 PLL 频率计算

主 PLL 与副 PLL 的计算均使用下列参数：

表27. PLL 频率参数

参数	描述
pll_in_clk	PLL 输入时钟的频率
F _{CCO}	PLL 电流控制振荡器的频率
pll_out_clk	PLL 输出频率
M	PLLCFG 寄存器的 MSEL 位的 PLL 倍频器值
P	PLLCFG 寄存器的 PSEL 位的 PLL 分频器值

PLL 输出频率（当 PLL 激活并锁定时）由下式给出：

$$\text{pll_out_clk} = M \times \text{pll_in_clk} \quad \text{或} \quad \text{pll_out_clk} = F_{\text{CCO}} / (2 \times P)$$

计算 CCO 的频率：

$$F_{\text{CCO}} = \text{pll_out_clk} \times 2 \times P \quad \text{或} \quad F_{\text{CCO}} = \text{pll_in_clk} \times M \times 2 \times P$$

PLL 输入与设置必须满足下列条件：

- M 的范围为 1~32。
- P 为 1、2、4、8 之中取一值。
- pll_in_clk 的范围为 10MHz~25MHz。
- F_{CCO} 的范围为 156MHz~320MHz。
- pll_out_clk 的范围为 9.75MHz~160MHz。

4.5.10 确定 PLL 设置的步骤

通常，PLL 设置值可以按下列方式找到：

1. 根据所需的 PLL 输出频率，选择一个振荡器频率（F_{OSC}）。如果使用了 USB 接口，则必须提供一个 12MHz、16MHz 或 24MHz 的外部晶振。12MHz 是建议值，它较省电，同时 PLL 设置也有更多灵活性。
2. 如果系统中使用了 USB 接口，并且 96MHz 或 144MHz 的 PLL 输出可以提供所需的 CPU 时钟频率，则可能仅使用 PLL0。
3. 通过计算 M 的值来配置 MSEL1 位，以获得所需 PLL 输出频率。M = pll_out_clk / pll_in_clk。写入 PLLCFG 寄存器中 MSEL 位的值是 M-1（或见[表 28](#)）。如果两个 PLL 都使用，则此方法同时适用两个 PLL。
4. 寻找一个 P 值来配置 PSEL 位，使 F_{CCO} 在其预定的 156MHz~320MHz 工作频率限制范围内。F_{CCO} 的计算使用下式：F_{CCO} = pll_out_clk x 2 x P。写入 PLLCFG 寄存器中 PSEL 位的值见[表 29](#)。

表28. PLL 倍频器值

M 值	MSEL 位 (PLLCFG 位 [4:0])	MSEL 十六进制
1	00000	0
2	00001	0x01
3	00010	0x02
4	00011	0x03
5	00100	0x04
6	00101	0x05
7	00110	0x06
8	00111	0x07
9	01000	0x08
10	01001	0x09
11	01010	0x0A
12	01011	0x0B
13	01100	0x0C
14	01101	0x0D
15	01110	0x0E
16	01111	0x0F
17	10000	0x10
18	10001	0x11
19	10010	0x12
20	10011	0x13
21	10100	0x14
22	10101	0x15
23	10110	0x16
24	10111	0x17
25	11000	0x18
26	11001	0x19
27	11010	0x1A
28	11011	0x1B
29	11100	0x1C
30	11101	0x1D
31	11110	0x1E
32	11111	0x1F

表29. PLL 分频器值

P 值	PSEL 位（PLLCFG 位 [6:5]）	PSEL 十六进制
1	00	0
2	01	0x1
4	10	0x2
8	11	0x3

4.5.11 PLL 配置序列

以下讨论是针对各个 PLL 以及泛指 PLL 相关寄存器（例如 PLLCFG，而不是 PLL0CFG 或 PLL1CFG）。这些指令必须适用于应用要满足的具体情况。

设置一个 PLL，并将时钟切换至它的输出：

1. 确认 PLL 输出尚未被使用。CCLKSEL 与 USBCLKSEL 寄存器均不得选择被设置的 PLL（见下面的“从 PLL 输出转走时钟”）。如果要写入任何所指的寄存器，则这些寄存器中包含的时钟分频器也要同时做设置。
2. 如果设置了主 PLL，并改变了主时钟源（IRC 与主振荡器之间转换），则先要做的修改是向 CLKSRCSEL 寄存器写入正确的值。
3. 将新的 PLL 设置值写入 PLLCFG 寄存器。在 PLLCON 寄存器中的 PLLE 位写入 1。PLL 馈送序列的完成方式是：向 PLLFEED 寄存器先写入 0xAA，再写入 0x55。
4. 设置所需的时钟分频器。这些寄存器可能包括：CCLKSEL、PCLKSEL、EMCCLKSEL，以及 USBCLKSEL 寄存器。
5. 等待 PLL 锁定。方法可以是轮询 PLLSTAT 寄存器并测试 PLOCK = 1，或者使用 PLL 锁定中断。
6. 在相应的位置选择 PLL 的输出，连接 PLL。这些位置可能包括 CCLKSEL 和 USBCLKSEL 寄存器。

从 PLL 输出转走时钟：

1. 如要切换成一个不使用 PLL 的模式，需向 CCLKSEL 与 USBCLKSEL 寄存器中的任一个或全部写入适当的值，从而选择一个不同的时钟源。
2. 然后，就可以写入 PLLCON 寄存器以关闭该 PLL，也可以做一个 PLL 馈送序列，写入 PLLCFG 寄存器而重新配置该 PLL。

4.5.12 PLL 配置示例

以下例子表示了如何根据不同系统需求选择 PLL 值。

例 1)
假设：

- 系统设计计划使用 IRC 来生成 CPU 时钟。
- CPU 时钟频率要求尽可能接近 80MHz。
- 在两个 PLL 中，只有 PLL0 可提供 CPU 时钟，因此本示例是针对 PLL0。12MHz IRC

频率最接近于 80MHz 的整倍数是 84MHz。由于 $\text{pll_out_clk} = M \times \text{pll_in_clk}$ ，于是 $M = \text{pll_out_clk} / \text{pll_in_clk} = 84 / 12 = 7$ 。

- 现在，必须找到一个 P 值，使 F_{CCO} 能落在 156MHz~320MHz 的 PLL 工作范围内。 $F_{\text{CCO}} = \text{pll_out_clk} \times 2 \times P$ 。先用 $P = 1$ 来找 F_{CCO} 的值，即 $84\text{MHz} \times 2 = 168\text{MHz}$ 。由于这个值已在 PLL 工作范围内，因此不需要做更多工作了。
- PLL 设置为 $M = 7$ 和 $P = 1$ 。这就需要将 6 ($M-1$ ，或见[表 28](#)) 写入 PLL0CFG 寄存器的 MSEL 域。PLL0CFG 的 PSEL 域需要的值是 0 (见[表 29](#))。将两个值一次写入，使 $\text{PLL0CFG} = 0x06$ 。有关 PLL 设置序列的说明见 [4.5.11](#) 节。

例2)

假设：

- 系统设计计划使用一只 12MHz 晶振同时生成 CPU 时钟和 USB 时钟。
- 希望 CPU 时钟接近 100MHz。
- 在两个 PLL 中，仅 PLL0 可以同时提供 CPU 时钟和 USB 时钟，因此本示例适用于 PLL0。PLL 输出必须是 48MHz 的整偶数倍，USB 才能正常工作（即 96MHz 的倍数）。两种 96MHz 的倍数可落在 PLL 工作范围内：192MHz ($2 \times 96\text{MHz}$) 和 288MHz ($3 \times 96\text{MHz}$)。其中，只有 192MHz 可以提供接近 100MHz (96MHz) 的 CPU 时钟。于是，可以用 96MHz 的 PLL 输出，获得两个需要的频率。由于 $\text{pll_out_clk} = M \times \text{pll_in_clk}$ ，则 $M = \text{pll_out_clk} / \text{pll_in_clk} = 96 / 12 = 8$ 。
- 现在，必须找到一个 P 值，使 F_{CCO} 落入 156MHz~320MHz 的 PLL 工作范围。 $F_{\text{CCO}} = \text{pll_out_clk} \times 2 \times P$ 。用 $P = 1$ 开始找 F_{CCO} 的值，其为 $96\text{MHz} \times 2 = 192\text{MHz}$ 。由于这个值已在 PLL 工作范围内，因此不需要做其它工作了。
- PLL 设置为 $M = 8$ 和 $P = 1$ 。这需要将 7 ($M-1$ ，或见[表 28](#)) 写入 PLL0CFG 寄存器的 MSEL 域。PLL0CFG 的 PSEL 域的值为 0 (见[表 29](#))。将两个值一次写入，使 $\text{PLL0CFG} = 0x07$ 。有关 PLL 设置序列的说明见 [4.5.11](#) 节。

例3)

- 假设：
- 系统设计要使用 USB 接口。
- 要求 CPU 时钟保持灵活性，其工作频率能够与 USB 时钟无关。
- 为了保持 CPU 时钟与 USB 时钟的独立性，CPU 将使用 PLL0。对于 USB，PLL1 可以按上例中的相同值做配置。PLL0 可以使用 IRC 或主振荡器工作，获得所需的任何频率，并且 PLL0 可以在不干涉 USB 运行的情况下改变设置。

4.6 时钟选择与分频

每个使用的 PLL 输出都必须经过分频,以获得各个子系统所需要的频率。对 LPC178x/177x,其 CPU、外部存储控制器、USB 接口,以及 APB 总线上的外设都有独立的时钟。独立的时钟选择复用器及时钟分频器为这些时钟的生成提供了灵活性。

4.6.1 CPU 时钟选择寄存器 (CCLKSEL—0x400F C104)

CCLKSEL 寄存器控制着主 PLL 输入的时钟选择,同时也控制 PLL0 输出在 CPU 使用前的分频。当 PLL0 被旁路时,可通过 1 分频。当 PLL0 工作时,输出必须经过分频,才能使 CPU 时钟频率 (CCLK) 在工作限制的范围内。5 位分频器可有一个选项区间,包括降低 CPU 的操作频率来暂时节省功耗而无需关闭 PLL0。注意 CPU 时钟速率设置不应低于外设的时钟速率。

两种时钟源可以选择用于驱动 PLL0 并最终驱动 CPU 与片上外设,它们是主振荡器和内部 RC 振荡器。

只有当 PLL0 未被使用时,才能安全地改变 PLL0 的时钟源选择。对于使用 PLL0 的系统如何改变使用时钟源,详见 4.5.12 节。

注意,关于时钟源的选择有以下限制:

- IRC 振荡器不应 (通过 PLL0) 用做 USB 子系统的时钟源。
- 如果 CAN 波特率高于 100kBit/s,则 IRC 振荡器不应 (通过 PLL0) 用做 CAN 控制器的时钟源。

表30. CPU 时钟选择寄存器 (CCLKSEL—0x400F C104) 位描述

位	符号	值	描述	复位值
4:0	CCLKDIV		从选中的时钟源中选择用于创建 CPU 时钟 (CCLK) 的分频值。	0x01
		0	分频器被关闭,没有时钟提供给 CPU。该设置通常不宜使用,CPU 将停止工作并要求复位以恢复运作。	
		1	输入时钟经 1 分频后产生 CPU 时钟。	
		2	输入时钟经 2 分频后产生 CPU 时钟。	
		3	输入时钟经 3 分频后产生 CPU 时钟。	
		:	:	
		31	输入时钟经 31 分频后产生 CPU 时钟。	
7:5	-		保留。读取值未定义,只写入 0。	无
8	CCLKSEL		为 CPU 时钟分频器选择输入时钟。	0
		0	系统时钟被用作 CPU 时钟分频器的输入。	
		1	主 PLL 输出被用作 CPU 时钟分频器的输入。	
31:9	-		保留。读取值未定义,只写入 0。	无

4.6.2 USB 时钟选择寄存器（USBCLKSEL—0x400F C108）

USBCLKSEL 寄存器控制着对 USB 子系统时钟的选择，而且还控制着时钟的分频，然后提供给 USB 使用。被选 PLL 的输出必须做分频，才能获得 50%占空比的 48MHz USB 时钟频率。分频器能够从 PLL 工作范围内的任何 48MHz 偶倍数频率（即 96MHz 的任何倍数），获得正确的 USB 时钟。

注：内部 RC 振荡器的时钟不应用于 USB 子系统。

表31. USB 时钟选择寄存器（USBCLKSEL—0x400F C108）位描述

位	符号	值	描述	复位值
4:0	USBDIV		从 PLL0 输出中选择用于创建 USB 时钟的分频值。只有下面所列出的值可以从 PLL0 输出中生成 48MHz 偶倍数频率。 警告： 不恰当设置该值，将导致 USB 接口操作错误。只有主振荡器与 PLL0 或 PLL1 配合在一起才可以提供符合 USB 精度和振荡规格的时钟。	0
		0	分频器被关闭，没有时钟提供给 USB 子系统。	
		4	PLL0 输出经 4 分频。PPL0 输出必须是 192MHz。	
		6	PLL0 输出经 6 分频。PPL0 输出必须是 288MHz。	
		其他	其他值不能生成适用于 USB 操作的 48MHz 时钟。	
7:5	-		保留。读取值未定义，只写入 0。	无
9:8	USBSEL		为 USB 时钟分频器选择输入时钟。	0
		00	系统时钟被用作 USB 时钟分频器的输入。当该时钟被选中时，USB 可以通过软件访问，但是 USB 功能不可用。	
		01	主 PLL 输出被用作 USB 时钟分频器的输入。	
		10	副 PLL 输出被用作 USB 时钟分频器的输入。	
		11	保留，不应使用该设置。	
31:10	-		保留。读取值未定义，只写入 0。	无

4.6.3 EMC 时钟选择寄存器（EMCCLKSEL—0x400F C100）

EMCCLKSEL 寄存器控制着 EMC 使用前的时钟分频。EMC 使用的基础时钟与 CPU 和 APB 外设相同。EMC 时钟可以与 CPU 时钟相等，或是其一半。它主要用于当 CPU 运行速度快于外部总线可以支持的速度的情况。

表32. EMC 时钟选择寄存器（EMCCLKSEL—0x400F C100）位描述

位	符号	值	描述	复位值
0	EMCDIV		选择与 CPU 时钟相对应的 EMC 时钟率。	0
		0	EMC 和 CPU 使用同一个时钟。	
		1	EMC 使用 CPU 率一半的时钟。	
31:1	-		保留。读取值未定义，只写入 0。	无

4.6.4 外设时钟选择寄存器（PCLKSEL—0x400F C1A8）

PCLKSEL 寄存器控制着所有 APB 外设使用的基础时钟。5 位分频器允许使用一系列频率。
注：外设时钟速率不应高于 CPU 时钟速率。

表33. 外设时钟选择寄存器（PCLKSEL—0x400F C1A8）位描述

位	符号	值	描述	复位值
4:0	PCLKDIV		选择在所有 APB 外设使用的时钟的分频值。	0x04
		0	分频器被关闭，没有时钟提供给 APB 外设。	
		1	输入时钟经 1 分频后产生 APB 外设时钟。	
		2	输入时钟经 2 分频后产生 APB 外设时钟。	
		3	输入时钟经 3 分频后产生 APB 外设时钟。	
		4	输入时钟经 4 分频后产生 APB 外设时钟。	
		:	:	
		31	输入时钟经 31 分频后产生 APB 外设时钟。	
31:5	-		保留。读取值未定义，只写入 0。	无

4.7 功率控制

LPC178x/177x 支持多种功率控制特性：睡眠模式、深度睡眠模式、掉电模式，以及深度掉电模式。CPU 时钟速率也可以通过改变时钟源、重新配置 PLL 值，和/或改变 CPU 时钟分频值来控制。这样就能根据应用需求，在功耗与处理速度之间做出一种权衡。另外，外设功率控制允许关闭单个片上外设的时钟，通过消除那些无用外设的动态功耗，精细地调节功耗。功率提升特性可将运行速度提高到 120MHz，或当工作速度不高于 100MHz 时低功耗。

通过 Cortex-M3 执行一条 WFI（等待中断）或 WFE（等待异常）指令进入任何低功耗模式。Cortex-M3 内部支持两种低功耗模式：睡眠与深度睡眠。它们通过 Cortex-M3 系统控制寄存器中的 SLEEPDEEP 位来选择。掉电与深度掉电模式则通过 PCON 寄存器中的位来选择。见表 35。相同寄存器中还包含了标志，用于指示每种低功耗模式的进入是否已真实发生。

LPC178x/177x 还实现了分离的电源域，能够关断其他设备的电源，而仍维持实时时钟的运行。

低功耗模式在调试期间有一些限制，详见 38.7 节。

4.7.1 睡眠模式

注：LPC178x/177x 的睡眠模式对应于 LPC2xxx 系列器件的空闲模式。更改名字的原因是 ARM 在 Cortex-M3 中结合了很多低功耗模式的控制。LPC178x/177x 文档在适当的地方会采用 Cortex-M3 的术语。

当进入睡眠模式时，内核时钟停止，PCON 中的 SMFLAG 位置位，见表 35。从睡眠模式中恢复不需要任何特殊序列，但要重新使能 ARM 内核的时钟。

在睡眠模式下，指令的执行被挂起，直到出现复位或中断。外设睡眠模式期间会继续工作，并可能生成使处理器恢复执行的中断。睡眠模式下处理器内核本身、存储器系统及相关控制器，以及内部总线的动态功耗会降低。

睡眠模式下，DMA 控制器可以继续工作，访问外设 RAM 与所有外设寄存器。Flash 存储器与主 SRAM 在睡眠模式下不可用，它们均被禁用以减少功耗。

一旦发生任何使能的中断，CPU 内核将从睡眠模式中被唤醒。

4.7.2 深度睡眠模式

注：LPC178x/177x 的深度睡眠模式对应于 LPC23xx 和 LPC224xx 系列器件的睡眠模式。更改名字的原因是 ARM 在 Cortex-M3 中结合了低功耗模式的控制。LPC178x/177x 文档在适当的地方会采用 Cortex-M3 的术语。

当芯片进入深度睡眠模式时，主振荡器停振，且几乎所有时钟都停止，PCON 的 DSFLAG 位置位，见表 35。为快速启动使用，IRC 仍继续运行。32kHz 的 RTC 振荡器不停止工作，RTC 中断可以用做一个唤醒源。Flash 存储器保持为待机模式，以备快速唤醒。PLL 自动关闭，时钟选择复用器被设定为使用 sysclk（复位状态）。时钟分频控制寄存器自动地复位

为零。

在深度睡眠模式下，处理器状态与寄存器、外设寄存器，以及内部 SRAM 的值均被保留，芯片管脚的逻辑电平也保持为静态。两种方式可以终止深度睡眠模式并恢复正常工作，一是通过复位，另一个是能在无时钟状态下产生的某些特定中断。由于芯片的所有动态操作均已被挂起，所以深度睡眠模式能将芯片功耗降低到极小的值。

在从深度睡眠模式唤醒时，如果进入深度睡眠模式以前使用的是 IRC，则一个 2 位 IRC 定时器开始计数，计数完成时（4 个周期）将恢复代码的执行与外设的活动。如果进入深度睡眠模式以前使用的是主振荡器，则 12 位主振荡器定时器开始计数，计数完成时（4096 个周期）恢复代码的执行。用户必须记住，唤醒以后要重新配置任何所需的 PLL 以及时钟分频器。

只要相关的中断使能时，器件就可以从深度睡眠模式中唤醒。这些中断包括 NMI、外部中断 EINT0 到 EINT3、GPIO 中断、以太网 Wake-on-LAN 中断、掉电检测、RTC 报警中断、USB 输入管脚跳变（USB 活动中断）、CAN 输入管脚跳变，或看门狗定时器超时等。

一旦有任何已使能的中断出现，唤醒就将发生。

4.7.3 掉电模式

掉电模式执行在深度睡眠模式下的所有操作，但也关闭了 Flash 存储器。进入掉电模式会使 PCON 的 PDFLAG 位置位，见[表 35](#)。这样节省了更多功耗，但唤醒后要等待 Flash 恢复工作，然后才能执行 Flash 存储器中的代码或访问其中的数据。

当芯片进入掉电模式时，IRC、主振荡器以及所有时钟均停止。如果 RTC 已使能，则继续运行，RTC 中断可以用于唤醒 CPU。Flash 被强迫进入掉电模式。PLL 自动关闭，时钟选择复用器被设为使用 sysclk（复位状态）。时钟分频控制寄存器自动复位到零。如果看门狗定时器正在运行，则它会在掉电模式下继续工作。

在从掉电模式下唤醒时，如果进入掉电模式以前使用了 IRC，则在 IRC 启动时间后（大约 60μs），2 位的 IRC 定时器开始计数，并在 4 个周期后结束。IRC 定时器到时后，如果代码是从 SRAM 中运行，则代码执行可以立即恢复。同时，Flash 唤醒定时器会测量大约 100μs 的 Flash 启动时间。当定时器超时时，可以访问 Flash。用户必须记住，在唤醒后要重新配置任何需用的 PLL 以及时钟分频器。

只要相关的中断使能时，器件就可以从掉电模式下唤醒。这些中断包括：NMI、外部中断 EINT0~EINT3、GPIO 中断、以太网 Wake-on-LAN 中断、掉电检测、RTC 报警中断、USB 输入管脚跳变（USB 活动中断），或 CAN 输入管脚跳变。

4.7.4 深度掉电模式

在深度掉电模式时，整个芯片的电源被关断（实时时钟、RESET 管脚、WIC 和 RTC 备用寄存器除外）。进入深度掉电模式会导致 PCON 的 DPDFLAG 位置位，见[表 35](#)。

为了优化功率，用户可以选择关闭或保留 32kHz 振荡器的电源。另外还可以使用外接电路在进入深度掉电模式后，通过 V_{DD(REG)(3V3)} 管脚关断片上稳压器的电源，和/或通过 V_{DD(REG)(3V3)} 管脚的 I/O 电源。在器件重新工作以前，必须恢复供电。

当发出了外部复位信号时，或当 RTC 中断已使能并生成了 RTC 中断时，将发生从深度掉电模式中的唤醒。

4.7.5 外设功率控制

外设的功率控制功能允许在应用中关闭不必要的外设，从而更好地节省功耗。详见 PCONP 寄存器的描述。

4.7.6 功率提升

功率提升特性允许工作频率超过 100MHz 的器件达到 120MHz 的上限。当一个芯片复位后开始执行用户代码时，这种提升特性默认为打开。当工作频率在 100MHz 或更低时，关闭这种模式可以节省功耗。见 4.7.12 节。

4.7.7 寄存器描述

功率控制功能所使用的寄存器见表 34。详情如下。

表34. 功率控制寄存器

名称	描述	访问	复位值 ^[1]	PLLn 寄存器名称和地址	表
PCON	功率控制寄存器。该寄存器含有使能 LPC178x/177x 系列的一些低功耗模式的控制位。见表 35。	R/W	0	0x400F C0C0	表 35
PCONP	外设寄存器的功率控制。该寄存器含有使能和禁能各个外设功能的控制位，可通过关闭应用中不必要的外设来减少功耗。见表 37。	R/W	0x0408 829E	0x400F C0C4	表 37
PBOOST	功率提升控制寄存器。该寄存器含有主片上调节器的输出，允许选择 100MHz 以上的高速操作或不高于 100MHz 的低功耗操作。	R/W	0x3	0x400F C1B0	表 38

[1] 复位值仅反映在使用位中保存的数据，它不包括保留位的内容。

4.7.8 功率模式控制寄存器（PCON—0x400F C0C0）

PCON 寄存器中包含了对某些低功耗模式的控制，以及其它与功率相关的控制，见[表 35](#)。

表35. 功率模式控制寄存器（PCON—0x400F C0C0）位描述

位	符号	描述	复位值
0	PM0	功率模式控制位 0。该位控制进入掉电模式。详细内容见 4.7.8.1 节。	
1	PM1	功率模式控制位 1。该位控制进入深度掉电模式。详细内容见 4.7.8.1 节。	0
2	BODRPM	掉电低功耗模式。当 BODRPM 为 1 时，掉电检测电路将在芯片进入掉电模式或深度睡眠模式时关断，使功耗进一步降低。 此时，不能使用掉电检测作为掉电模式的唤醒源。 当该位为 0 时，掉电检测功能在掉电模式和深度睡眠模式中保持有效。 有关掉电检测的详细内容请参见“系统控制模块”。	0
3	BOGD	掉电全局禁能。当 BOGD 为 1 时，掉电检测电路一直被完全禁止，且不消耗功率。 当该位为 0 时，掉电检测电路被使能。 有关掉电检测的详细内容请参见“系统控制模块”。	0
4	BORD	掉电复位禁能。当 BORD 为 1 时，V _{DD(REG)(3V3)} 电压降到 BOD 复位触发电平以下不会导致芯片复位。 掉电中断不受影响。 当 BORD 为 0 时，BOD 复位被使能。 有关掉电检测的详细内容见 3.6 节。	0
7:3	-	保留。读取值未定义，只写入 0。	无
8	SMFLAG	睡眠模式进入标志。当成功进入睡眠模式时该位置位。通过向该位写入 1 由软件将其清零。	0 ^{[1][2]}
9	DSFLAG	深度睡眠进入标志。当成功进入深度睡眠模式时该位置位。通过向该位写入 1 由软件将其清零。	0 ^{[1][2]}
10	PDFLAG	掉电进入标志。当成功进入掉电模式时该位置位。通过向该位写入 1 由软件将其清零。	0 ^{[1][2]}
11	DPDFLAG	深度掉电进入标志。当成功进入深度掉电模式时该位置位。通过向该位写入 1 由软件将其清零。	0 ^{[1][3]}
31:12	-	保留。读取值未定义，只写入 0。	无

[1] 在特定的时间内只有一个标志有效。
[2] 硬件复位值仅用于内核电源的上电或通过掉电检测事件进行硬件复位。
[3] 硬件复位值仅用于上电事件（Vbat）。

4.7.8.1 低功耗模式的编码

PCON 中的 PM1 与 PM0 位允许根据需求进入低功耗模式。这些位的编码能够与以前仅支持睡眠模式和掉电模式的器件保持反向兼容。下面的[表 36](#)显示了 LPC178x/177x 所支持三种低功耗模式的编码。

表36. 低功耗模式的编码

PM1, PM0	描述
00	正如 Cortex-M3 系统控制寄存器的 SLEEPDEEP 位所定义，执行 WFI 或 WFE 进入睡眠或深度睡眠模式。
01	如果 Cortex-M3 系统控制寄存器的 SLEEPDEEP 位为 1，则执行 WFI 或 WFE 进入掉电模式。
10	保留，不应使用该设置。
11	如果 Cortex-M3 系统控制寄存器的 SLEEPDEEP 位为 1，则执行 WFI 或 WFE 进入深度掉电模式。

4.7.9 从低功耗模式中唤醒

任何已使能的中断都可以将 CPU 从睡眠模式中唤醒。某些中断可以唤醒处于深度睡眠模式或掉电模式中的处理器。

如果中断已被使能，则在深度睡眠模式或掉电模式下发生的中断就将唤醒 CPU。唤醒以后，执行将持续到相应的中断服务程序。这些中断是：NMI、外部中断 EINT0~EINT3、GPIO 中断、以太网 Wake-on-LAN 中断、掉电检测中断、RTC 报警中断、CAN 活动中断，以及 USB 活动中断。相应中断必须已在 NVIC 中被使能，唤醒的过程才能发生。对于管脚有关的外设功能，相关功能也必须映射到管脚上。

CAN 活动中断由 CAN 总线管脚上的活动产生，USB 活动中断则由 USB 总线管脚上的活动产生。这些中断只用于唤醒那些处于深度睡眠或掉电模式下的 CPU，且外设功能已上电但但未激活。通常，如果要使用这些中断，则在使能这些中断和进入所需低功耗模式以前，应对这些中断的标志做轮询。这样可以避免立即唤醒，从而节省时间和能耗。一旦被唤醒，中断服务就可以关闭相关的活动中断，对应用进行具体设置，并退出等待正常的外设中断。

在深度掉电模式下，器件的大部分内部供电已关断，这限制了从该模式下唤醒的可能性。对于 LPC178x/177x，外部复位可以唤醒器件。另外，在 RTC 运行并已被设置为产生一个中断时，RTC 事件也可以唤醒器件。

4.7.10 外设功率控制寄存器（PCONP—0x400F C0C4）

PCONP 寄存器可以关闭所选择外设功能的电源，从而达到节电目的。其实现方法是门控关闭特定外设模块的时钟源。有些外设功能不可以关闭（如看门狗定时器和系统控制模块）。

有些外设的功耗可能与时钟无关，尤其是那些包含了模拟功能的外设。这些外设可能有一个独立的禁用控制，它能关闭多余的电路以降低功耗。这种情况下，外设应首先内部禁用，然后关闭使用的 PCONP，这样能最大化地节省功耗。有关节省外设功耗的具体特性，详见外设的具体描述章节。

PCONP 的每个位都控制着一个外设，见[表 37](#)。

如果某个外设控制位为 1，则该外设被使能。如果某个外设控制位为 0，则该外设时钟被禁用（门控关闭），以节省功耗。例如，如果第 19 位为 1，则 I²C1 接口被使能。如果第 19 位为 0，则 I²C1 接口被禁用。

重要提示：仅当外设设在 PCONP 寄存器中使能时，才能对该外设寄存器做有效的读写操作！

表37. 外设功率控制寄存器（PCONP—0x400F C0C4）位描述

位	符号	描述	复位值
0	PCLCD	LCD 控制器功率/时钟控制位。	0
1	PCTIM0	定时器/计数器 0 功率/时钟控制位	1
2	PCTIM1	定时器/计数器 1 功率/时钟控制位。	1
3	PCUART0	UART0 功率/时钟控制位。	1
4	PCUART1	UART1 功率/时钟控制位。	1
5	PCPWM0	PWM0 功率/时钟控制位。	0
6	PCPWM1	PWM1 功率/时钟控制位。	0
7	PCI2C0	I ² C0 接口功率/时钟控制位。	1
8	PCUART4	UART4 功率/时钟控制位。	0
9	PCRTC	RTC 和事件监视器/记录器功率/时钟控制位。	1
10	PCSSP1	SSP1 接口功率/时钟控制位。	0
11	PCEMC	外部存储器控制器功率/时钟控制位。	0
12	PCADC	A/D 转换器（ADC）功率/时钟控制位。	0
注：在清零该位之前清零 AD0CR 中的 PDN 位；在置位 PDN 之前置位该位。			
13	PCCAN1	CAN 控制器 1 功率/时钟控制位。	0
14	PCCAN2	CAN 控制器 2 功率/时钟控制位。	0
15	PCGPIO	IOCON、GPIO 和 GPIO 中断功率/时钟控制位。	1
16	-	保留。读取值未定义，只写入 0。	无
17	PCMCPWM	电机控制 PWM 功率/时钟控制位。	0
18	PCQEI	正交编码器接口功率/时钟控制位。	0
19	PCI2C1	I ² C1 接口功率/时钟控制位。	1
20	PCSSP2	SSP2 接口功率/时钟控制位。	0
21	PCSSP0	SSP0 接口功率/时钟控制位。	0
22	PCTIM2	定时器 2 功率/时钟控制位。	0
23	PCTIM3	定时器 3 功率/时钟控制位。	0
24	PCUART2	UART2 功率/时钟控制位。	0
25	PCUART3	UART3 功率/时钟控制位。	0
26	PCI2C2	I ² C 接口 2 功率/时钟控制位。	1
27	PCI2S	I ² S 接口功率/时钟控制位。	0
28	PCSDC	SD 卡接口功率/时钟控制位。	0
29	PCGPDMA	GP DMA 功能功率/时钟控制位。	0
30	PCENET	以太网模块功率/时钟控制位。	0
31	PCUSB	USB 接口功率/时钟控制位。	0

注：DAC 外设 在 PCONP 中没有控制位。要 使能 DAC，必须配置相关的 IOCON 寄存器，在相关的管脚 P0[26]上选择其输出。见 8.4.1 节。

4.7.11 功率控制注意事项

每次复位后，PCONP 寄存器包含了使能所选接口与外设的值，这些接口与外设由 PCONP 控制而使能。因此，除了通过外设专门寄存器进行正确配置外，用户应用程序也必须访问 PCONP，以使能对应的外设。

在需要控制功率的系统中，需要在 PCONP 寄存器中使能必要的外设，寄存器的其他“保留”位或当前不使用的外设所对应的位都必须清零。

4.7.12 功率提升控制寄存器（PBOOST—0x400F C1B0）

功率提升控制寄存器通过控制片上主稳压器的输出，允许选择 100MHz 以上的高速运行，或 100MHz 及以下的低功耗运行。复位后首次执行用户代码时，提升特性被启动。然后，如果 CPU 时钟速率总在 100MHz 甚至更低，就可以通过用户代码关闭功率提升特性，从而节省 100MHz 以上运行时的能量功耗。详见表 38。

表38. 功率提升控制寄存器（PBOOST—0x400F C1B0）位描述

位	符号	描述	复位值
1:0	Boost	提升控制位。 00: 提升关闭，操作功耗必须在 100MHz 以下。 11: 提升开启，操作功耗允许高达 120MHz。 不允许其他值。	0x3
31:2	-	保留。读取值未定义，只写入 0。	无

4.7.12.1 低功耗模式的编码

PCON 的 PM1 和 PM0 位允许进入所需的低功耗模式。这些位的编码与以前仅支持睡眠模式和掉电模式的器件保持反向兼容。下面的[表 36](#)给出了 LPC178x/177x 所支持三种低功耗模式的编码。

表39. 低功耗模式的编码

PM1, PM0	描述
00	正如 Cortex-M3 系统控制寄存器的 SLEEPDEEP 位所定义，执行 WFI 或 WFE 进入睡眠或深度睡眠模式。
01	如果 Cortex-M3 系统控制寄存器的 SLEEPDEEP 位为 1, 则执行 WFI 或 WFE 进入掉电模式。
10	保留，不应使用该设置。
11	如果 Cortex-M3 系统控制寄存器的 SLEEPDEEP 位为 1, 则执行 WFI 或 WFE 进入深度掉电模式。

4.7.13 电源域

LPC178x/177x 提供了两个独立的电源域，能够断开设备电源，同时维持实时时钟的运行。

VBAT 管脚仅为 RTC 域提供电源。RTC 功耗很低，可由外部电池供电。当器件内核电源存在时（该电源亦可为 RTC 供电），当主电源有效时无需使用电池的电源。

4.8 唤醒定时器

在上电或使用 12MHz IRC 振荡器作为时钟源将 LPC178x/177x 从掉电模式中唤醒时，LPC178x/177x 就开始运行。这样芯片可以快速地启动工作。如果应用需要主振荡器或一至两个 PLL，则软件要使能这些功能，等待它们的稳定，然后再将其作为时钟源。

主振荡器开始被激活时，唤醒定时器使软件确认主振荡器完全工作，然后处理器才能将主振荡器用作时钟源并开始执行指令。这对于上电、所有类型的复位，以及任何原因导致上述功能关闭的情况下非常重要。由于在掉电模式下，振荡器和其它功能均被关断，因此处理器从掉电模式中的任何唤醒都必须使用唤醒定时器。

唤醒定时器通过监控晶体振荡器，以确定开始代码执行是否安全。当芯片加电时，或某些事件使芯片退出掉电模式时，振荡器需要一些时间才能产生足够振幅的信号以驱动时钟逻辑。这个时间量取决于很多因素，包括 $V_{DD(REG)(3V3)}$ 的上升速率（上电时）、晶振的类型及其电气特性（如果使用了石英晶振）、任何其它外接电路（如电容），以及振荡器在现有环境条件下自身的特性。

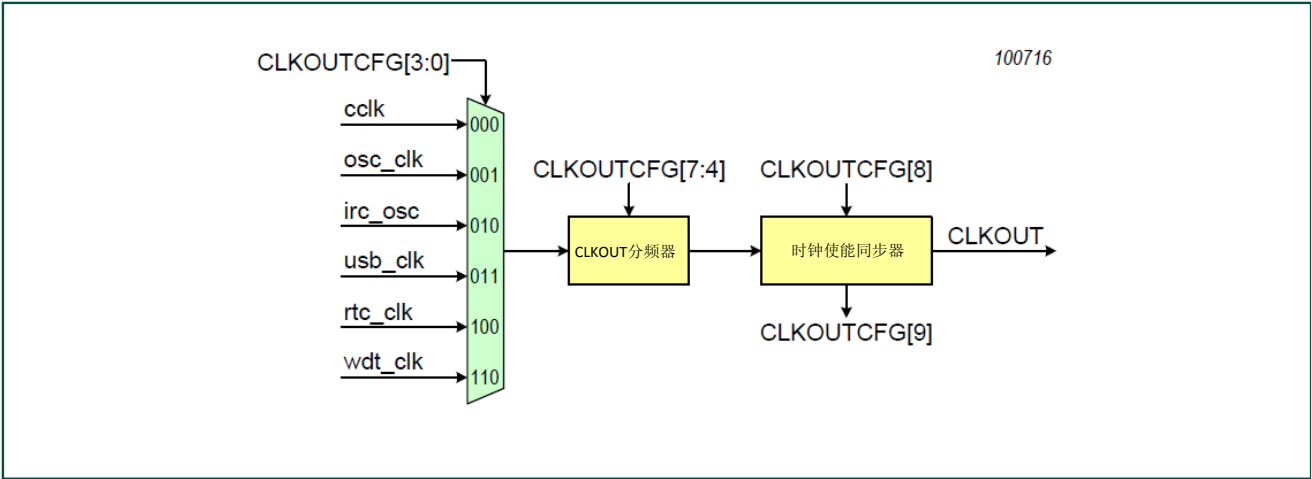
一旦检测到了一个时钟，唤醒定时器就对固定的时钟数（4096 个时钟）进行计数，然后将设置标志（SCS 寄存器中的 OSCSTAT 位）表示主振荡器已准备就绪。然后，软件就可以切换到主振荡器，并启动任何需要的 PLL。详见本章有关主振荡器的描述。

4.9 外部时钟输出管脚

为了便于系统测试与开发，任何一种内部时钟均可引入 CLKOUT 功能（在 P1[25]或 P1[27] 管脚可使用），如图 10 所示。

可以通过 CLKOUT 观察到的时钟是 CPU 时钟（cclk）、主振荡器（osc_clk）、内部 RC 振荡器（irc_osc）、USB 时钟（usb_clk）、RTC 时钟（rtc_clk），以及看门狗振荡器（wdt_clk）。

图10. CLKOUT 选择



4.9.1 时钟输出配置寄存器 CLKOUTCFG—0x400F C1C8）

CLKOUTCFG 控制选择出现在 CLKOUT 管脚的内部时钟，允许通过一个整数值（最高 16）对该时钟分频。分频器可以用于提供一个与某个片上时钟相关的系统时钟。对于多数时钟源，可由 1 分频。当选择了 CPU 时钟，并且其频率高于 50MHz 时，输出前必须经过分频，这样才能使频率在管脚的能力范围内（以合理的逻辑电平作转换）。如果所选时钟没有运行，则 CLKOUT 上没有信号。

注：CLKOUT 复用器主要用于在可能的时钟源之间完全切换而不受干扰。分频器也可用来改变分频值而不受干扰。

表40. 时钟输出配置寄存器（CLKOUTCFG—0x400F C1C8）位描述

位	符号	值	描述	复位值
3:0	CLKOUTSEL		选择 CLKOUT 功能的时钟源。	0
		0000	选择 CPU 时钟作为 CLKOUT 的时钟源。	
		0001	选择主振荡器作为 CLKOUT 的时钟源。	
		0010	选择内部 RC 振荡器作为 CLKOUT 的时钟源。	
		0011	选择 USB 时钟作为 CLKOUT 的时钟源。	
		0100	选择 RTC 振荡器作为 CLKOUT 的时钟源。	
		0110	选择看门狗振荡器作为 CLKOUT 的时钟源。	
		其它	保留，不使用这些设置。	
7:4	CLKOUTDIV		分频值为输出时钟的整数值减 1。	0
		0000	时钟经过 1 分频。	
		0001	时钟经过 2 分频。	
		0010	时钟经过 3 分频。	
		
		1111	时钟经过 16 分频。	
8	CLKOUT_EN		CLKOUT 使能控制，允许切换 CLKOUT 源而不受干扰。 清零该位在下一个下降沿停止 CLKOUT。置位该位使能 CLKOUT。	0
9	CLKOUT_ACT		CLKOUT 有效指示。当 CLKOUT 使能时读为 1。当 CLKOUT 禁止时读为 0，通过 CLKOUT_EN 位进行该操作，并且时钟已停止。	0
31:10	-		保留。读取值未定义，只写入 0。	无

5.1 简介

LPC178x/177x 的 Flash 加速器能够使 Cortex-M3 处理器在运行 Flash 代码时，获得最高的性能，同时节省功耗。Flash 加速器还改进了访问 Flash 存储器数据的速度与功率。

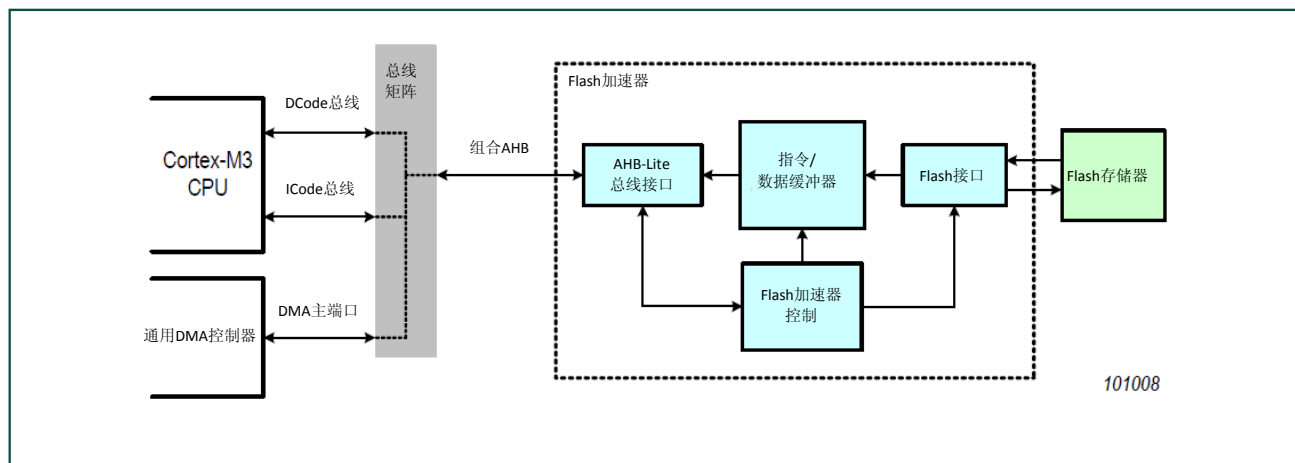
5.2 Flash 加速器模块

Flash 加速器分成以下几个功能模块：

- AHB-Lite 总线接口，可接受 Cortex-M3 的 I-code 与 D-code 总线以及通用 DMA 控制器的访问
- 八个 128 位缓冲区数组
- Flash 加速器控制逻辑，包括地址比较与 Flash 控制
- Flash 存储器接口

图 11 给出了 Flash 加速器模块与数据通路的简化方框图。

图11. Flash 加速器简化模块方框图（显示潜在总线连接）



在以下描述中，“取指”（fetch）一词表示 CPU 发出的一个直接的 Flash 读请求。而“预取指”（prefetch）表示对当前处理器取指地址之后的地址执行 Flash 读操作。

5.2.1 Flash 存储器组

LPC178x/177x Flash 加速器仅控制一组 Flash 存储器。

Flash 编程操作不受 Flash 加速器的控制，而是作为一个独立的功能进行处理。“Boot ROM”包含了 Flash 编程算法，此算法可以作为应用程序的一部分被调用；ROM 中还有一个允许 Flash 存储器编程的装载程序。

5.2.2 Flash 编程问题

在编程和擦除操作期间，不允许访问 Flash 存储器，因此，如果在 Flash 存储器忙于处理编程的时候请求对 Flash 地址进行存储器访问，Flash 加速器需要强制 CPU 等待。在某些情况下，这种延迟可能导致看门狗超时。用户必须知道存在着这种可能性，并采取一些措施，从而确保在编程或擦除 Flash 存储器过程中，不会因非预期的看门狗复位而导致系统故障。

为了防止从 Flash 存储器中读取无效的数据，在 Flash 编程或擦除操作开始后，LPC178x/177x Flash 加速器的缓冲区自动失效。任何对 Flash 地址的读操作都将会在完成 Flash 编程操作之后启动新的取指操作。

5.3 寄存器描述

Flash 加速器由[表 41](#)所示寄存器控制。更多详情如下。

表41. 存储器加速模块寄存器汇总

名称	描述	访问	复位值 ^[1]	地址
FLASHCFG	存储器加速模块配置寄存器。 控制存储器加速计时。见 表 42 。	R/W	0x303A	0x400F C000

[1] 复位值仅反映在使用位中保存的数据，它不包括保留位的内容。

5.4 Flash 加速器配置寄存器（FLASHCFG—0x400F C000）

寄存器中的配置位用于选择 Flash 访问时间，如表 42 所示。FLASHCFG 中的低位用于控制内部 Flash 加速器功能，并且不能更改。

复位后，Flash 加速器被使能，Flash 访问时间被设定为默认值（4 个时钟）。

改变 FLASHCFG 寄存器的值会导致 Flash 加速器所有锁存的内容无效，需要执行新的 Flash 读操作。这就保证了 Flash 加速器与 CPU 操作同步。

表42. 存储器加速模块配置寄存器（FLASHCFG—0x400F C000）位描述

位	符号	值	描述	复位值
11:0	-	-	保留，用户软件不应从复位值处更改这些位。	0x03A
15:12	FLASHTIM		Flash 访问时间。该字段的值加 1 等于访问 Flash 所用到的 CPU 时钟数。 警告： 不恰当设置该值可能会致使器件操作错误。	0x3
		0000	Flash 访问使用 1 个 CPU 时钟。用于高达 20MHz 的 CPU 时钟（参见 4.7.6 节）。	
		0001	Flash 访问使用 2 个 CPU 时钟。用于高达 40MHz 的 CPU 时钟（参见 4.7.6 节）。	
		0010	Flash 访问使用 3 个 CPU 时钟。用于高达 60MHz 的 CPU 时钟（参见 4.7.6 节）。	
		0011	Flash 访问使用 4 个 CPU 时钟。用于高达 80MHz 的 CPU 时钟（参见 4.7.6 节）。使用该设置进行高达 100MHz 到 120MHz 的操作。	
		0100	Flash 访问使用 5 个 CPU 时钟。用于高达 100MHz 的 CPU 时钟（参见 4.7.6 节）。	
		0101	Flash 访问使用 6 个 CPU 时钟。该“安全”设置在任何条件下都有效。其它用于未来更高速的器件	
31:16	-		保留。读取值未定义，只写入 0。	无

5.5 运行

简言之，Flash 加速器试图需要及时锁存的下一条 Cortex-M3 指令，以防止 CPU 取指延迟。LPC178x/177x 使用 Flash 存储器组。Flash 加速器带有一个含 8 个 128 位缓冲区数组，该数组能够以一种可配置的方式，同时存储指令与数据。数组中的每个 128 位缓冲区都可以包含 4 个 32 位指令、8 个 16 位指令，或者两者的某种组合。在执行序列代码过程中，缓冲区通常包含了当前指令，以及该指令所在的整个 Flash 行，或者是一个包含前次请求地址的整个 Flash 数据行。缓冲区的标记是按照其使用方式（是指令缓冲区还是数据缓冲区）以及存取时间。这个信息用于执行缓冲区置换策略。

Cortex-M3 为代码存储空间中的指令存取和数据存取提供了独立的总线，分别是 I-code 和 D-code。这些总线再加上通用 DMA 控制器的主端口均由 AHB 多层矩阵做仲裁。对 Flash 地址空间的任何存取都要提交给 Flash 加速器。

如果 CPU 同时给出了一个 Flash 指令取指和一个 Flash 数据存取，则多层矩阵会给予数据存取优先权。这是因为，数据存取延迟一定会减缓执行速度，而指令取指延迟则通常不会。当 Flash 数据存取结束时，再重新启动已进行中的任何 Flash 取指或预取指。

如发生分支以及其它程序流的变化，则上述指令取指的顺序流会发生中断。Flash 加速器中的缓冲区置换策略会尝试尽可能地将可复用的信息保留到下次需要时。

如果尝试直接写入 Flash 存储器，而未采用正常的 Flash 编程接口（通过 Boot ROM 函数调用），则 Flash 加速器会产生一个错误情况。CPU 将这种错误看作一个数据异常。GPDMA 对错误情况的处理见 [34.4.1.6.3](#) 节。

当缓冲数组中的当前内容不能满足一个指令取指要求时，或发起的预取指并非针对该 Flash 行时，CPU 将延迟，同时启动一个针对相关 128 位 Flash 行的读取。如果已经启动了一个预取指但还没完成，则 CPU 的延迟会更短，因为所需的 Flash 存取已在进行中了。

通常情况下，对刚预取指过的地址存取，或是对一个缓冲区，而其后续者还未不在其它缓冲区中，则会开始 Flash 预取指。一个进行中的预取指可以被一个数据存取中止，以尽量减少 CPU 的延迟。

预取指 Flash 行被锁存在 Flash 存储器内，但在 CPU 给出一个包含该预取指 Flash 行的地址以前，Flash 加速器并不会将此行捕捉到一个缓冲区内。如果内核给出了一个尚未在缓冲区中的指令地址，并且该地址也不包含在预取指 Flash 行中，则该预取指行将被丢弃。

有些特殊下，例如 CPU 请求访问一个已包含在指令缓冲区中的地址的数据。此时，数据将从缓冲区被读取，此缓冲区就好象一个数据缓冲区。还有相反的情况，如果 CPU 请求一个指令地址，而一个现有的数据缓冲区可以满足这个地址，于是指令就由数据缓冲区提供，该缓冲区变为一个指令缓冲区。这让缓冲区的处理变得很困难，因为 Flash 加速器要决定接下来应覆盖哪个缓冲区。

6.1 特性

- 可嵌套向量中断控制器（NVIC）是 ARM Cortex-M3 的一个组成部分。
- 与内核紧密耦合的中断控制器，可支持低中断延时。
- 控制着系统异常与外设中断。
- 在 LPC178x/177x 中，NVIC 支持 40 个向量中断。
- 32 个可编程的中断优先级，带硬件优先级屏蔽。
- 可重新定位的向量表。
- 不可屏蔽中断。
- 软件中断生成。

6.2 描述

可嵌套向量中断控制器（NVIC）是 Cortex-M3 的一个组成部分。它与 CPU 的紧密耦合实现了低中断延时，使新近中断可以得到高效处理。NVIC 除处理系统异常外，还处理中断。处理的异常包括复位、NMI、硬件故障、MemManage 故障、总线故障、使用故障、SVCall、调试监测器、PendSV 和 SysTick。

关于 NVIC 详细操作工作，参见《Cortex-M3 用户指南》[39.4.2](#) 节。

6.3 中断源

[表 43](#) 列出了各个外设功能对应的中断源。每个外设都可能有一条或多条连到向量中断控制器的中断线。每条线可能表示一个以上的中断源，详见说明。

异常号码对应异常向量表中存储的表项。中断号码用于其它环境，如软件中断。

注意，系统异常是进入 Cortex-M3 的硬接线，且表中未显示。有关 SysTick 中断的其它信息可见“系统节拍定时器”一章的 [25.1](#) 节。

此外，NVIC 还可处理不可屏蔽中断（NMI）。为了使 NMI 能按照外部信号工作，NMI 功能必须连接到相关的器件管脚（P2[10] / EINT0n / NMI）。连接以后，该管脚上的逻辑 1 将产生 NMI 等待处理。详细内容参见本用户手册附带的《Cortex-M3 用户指南》。

表43. 连接到向量中断控制器的中断源

中断 ID	异常号	向量偏移量	功能	标志
0	16	0x40	看门狗定时器	看门狗中断 (WDINT)
1	17	0x44	定时器 0	匹配 0—1 (MR0、MR1) 捕获 0—1 (CR0、CR1)
2	18	0x48	定时器 1	匹配 0—2 (MR0、MR1、MR2) 捕获 0—1 (CR0、CR1)
3	19	0x4C	定时器 2	匹配 0-3 捕获 0-1
4	20	0x50	定时器 3	匹配 0-3 捕获 0-1
5	21	0x54	UART0	Rx 线状态 (RLS) 发送保持寄存器空 (THRE) Rx 数据可用 (RDA) 字符超时指示 (CTI) 自动波特率结束 (ABEO) 自动波特率超时 (ABTO)
6	22	0x58	UART1	Rx 线状态 (RLS) 发送保持寄存器空 (THRE) Rx 数据可用 (RDA) 字符超时指示 (CTI) 调制解调器控制改变 自动波特率结束 (ABEO) 自动波特率超时 (ABTO)
7	23	0x5C	UART 2	Rx 线状态 (RLS) 发送保持寄存器空 (THRE) Rx 数据可用 (RDA) 字符超时指示 (CTI) 自动波特率结束 (ABEO) 自动波特率超时 (ABTO)
8	24	0x60	UART 3	Rx 线状态 (RLS) 发送保持寄存器空 (THRE) Rx 数据可用 (RDA) 字符超时指示 (CTI) 自动波特率结束 (ABEO) 自动波特率超时 (ABTO)
9	25	0x64	PWM1	PWM1 的匹配 0- 6 PWM1 的捕获 0- 1
10	26	0x68	I ² C0	SI (状态改变)
11	27	0x6C	I ² C1	SI (状态改变)
12	28	0x70	I ² C2	SI (状态改变)
13	29	0x74	(未使用)	-

中断 ID	异常号	向量偏移量	功能	标志
14	30	0x78	SSP0	SSP0 的 Tx FIFO 一半为空 SSP0 的 Rx FIFO 一半为满 SSP0 的接收超时 SSP0 的接收溢出
15	31	0x7C	SSP 1	Tx FIFO 一半为空 Rx FIFO 一半为满 接收超时 接收溢出
16	32	0x80	PLL0 (主 PLL)	PLL0 锁定 (PLOCK0)
17	33	0x84	RTC 和事件监视器/记录器	计数器增加 (RTCCIF) 报警(RTCALF) EV0、EV1、EV2
18	34	0x88	外部中断	外部中断 0 (EINT0)
19	35	0x8C	外部中断	外部中断 1 (EINT1)
20	36	0x90	外部中断	外部中断 2 (EINT2)
21	37	0x94	外部中断	外部中断 3 (EINT3)
22	38	0x98	ADC	A/D 转换器转换结束
23	39	0x9C	BOD	掉电检测
24	40	0xA0	USB	USB_INT_REQ_LP、USB_INT_REQ_HP、 USB_INT_REQ_DMA、USB_HOST_INT、USB_ATX_INT、 USB_OTG_INT、USB_I2C_INT
25	41	0xA4	CAN	CAN Common、CAN 0 Tx、CAN 0 Rx、CAN 1 Tx、CAN 1 Rx
26	42	0xA8	DMA 控制器	所有 DMA 通道的中断状态
27	43	0xAC	I ² S	Irq、dmareq1、dmareq2
28	44	0xB0	以太网	WakeUpInt、SoftInt、TxDoneInt、TxFinishedInt、TxErrorInt、 TxUnderrunInt、RxDoneInt、RxFinishedInt、RxErrorInt、 RxOverrunInt
29	45	0xB4	SD 卡接口	RxDataAvbl、TxDataAvbl、RxFifoEmpty、TxFifoEmpty、 RxFifoFull、TxFifoFull、RxFifoHalfFull、TxFifoHalfEmpty、 RxActive、TxActive、CmdActive、DataBlockEnd、StartBitErr、 DataEnd、CmdSent、CmdRespEnd、RxOverrun、TxUnderrun、 DataTimeOut、CmdTimeOut、DataCrcFail、CmdCrcFail
30	46	0xB8	电机控制 PWM	IPER[2:0]、IPW[2:0]、ICAP[2:0]、FES
31	47	0xBC	正交编码器	INX_Int、TIM_Int、VELC_Int、DIR_Int、ERR_Int、ENCLK_Int、 POS0_Int、POS1_Int、POS2_Int、REV_Int、POS0REV_Int、 POS1REV_Int、POS2REV_Int
32	48	0xC0	PLL1 (备 PLL)	PLL1 锁定 (PLOCK1)
33	49	0xC4	USB 活动中断	USB_NEED_CLK
34	50	0xC8	CAN 活动中断	CAN1WAKE、CAN2WAKE
35	51	0xCC	UART4	Rx 线状态 (RLS) 发送保持寄存器空 (THRE) Rx 数据可用 (RDA) 字符超时指示 (CTI) 自动波特率结束 (ABEO) 自动波特率超时 (ABTO)

中断 ID	异常号	向量偏移量	功能	标志
36	52	0xD0	SSP2	SSP2 的 Tx FIFO 一半为空 SSP2 的 Rx FIFO 一半为满 SSP2 的接收超时 SSP2 的接收溢出
37	53	0xD4	LCD 控制器	BER、VCompl、LNBUI、FUF1、Crsrl
38	54	0xD8	GPIO 中断	P0xREI、P2xREI、P0xFEI、P2xFEI
39	55	0xDC	PWM0	PWM0 的匹配 0- 6 PWM0 的捕获 0- 1
40	56	0xE0	EEPROM	EE_PROG_DONE、EE_RW_DONE

6.4 向量表的重新映射

Cortex-M3 有一种机制，能够将中断向量表重新映射到存储器映射中的其它位置。这由 Cortex-M3 中的向量表偏移寄存器（VTOR）控制。

向量表可以位于 Cortex-M3 地址空间的 1GB 底部的任何位置。向量表应与 256 字（1024 字节）边界对齐，从而确保 LPC178x/177x 系列器件的一致性。有关向量表的偏移特性，详见本手册附带的《Cortex-M3 用户指南》的 [39.4.3.5](#) 节。

ARM 将 VTOR 第 29 位（TBLOFF）描述为选择一个存储区，可以是代码或 SRAM。为简单起见，这个位可以简单看成地址偏移的一部分，因为“代码”空间与“SRAM”空间之间的分界出现在与存储器地址第 29 位相应的位置。

例如：

向量表要置于“本地”静态 RAM 的起始处，起始地址为 0x1000 0000，则将值 0x1000 0000 置于 VTOR 寄存器中。这表示了代码空间的地址 0x1000 0000，因为 VTOR 的第 29 位等于 0。

向量表要置于外设 RAM 的起始处，起始地址为 0x2000 0000，则将值 0x2000 0000 置于 VTOR 寄存器中。这表示了 SRAM 空间中的地址 0x2000 0000，因为 VTOR 的第 29 位等于 1。

6.5 寄存器描述

下表概述了 LPC178x/177x 中所实现 NVIC 的寄存器。《Cortex-M3 用户指南》的 [39.4.2](#) 节提供了有关 NVIC 的功能描述。

表44. NVIC 寄存器映射

名称	描述	访问	复位值	地址	表
ISER0 to ISER1	中断使能设置寄存器。这些寄存器允许使能中断并读回特定外设功能的中断使能。	RW	0	ISER0—0xE000 E100	表 45
				ISER1—0xE000 E104	表 46
ICPR0 to ICPR1	中断使能清除寄存器。这些寄存器允许禁能中断并读回特定外设功能的中断使能。	RW	0	ICER0—0xE000 E180	表 47
				ICER1—0xE000 E184	表 48
ISPR0 to ISPR1	中断暂挂设置寄存器。这些寄存器允许更改中断状态为暂挂并读回特定外设功能的中断暂挂状态。	RW	0	ISPR0—0xE000 E200	表 49
				ISPR1—0xE000 E204	表 50
ICPR0 to ICPR1	中断暂挂清除寄存器。这些寄存器允许更改中断状态为非暂挂并读回特定外设功能的中断暂挂状态。	RW	0	ICPR0—0xE000 E280	表 51
				ICPR1—0xE000 E284	表 52
IABR0 to IABR1	中断活动位寄存器。这些寄存器允许读取特定外设功能的当前中断活动状态。	RO	0	IABR0—0xE000 E300	表 53
				IABR1—0xE000 E304	表 54
IPR0 to IPR10	中断优先级寄存器。这些寄存器允许向各个中断分配一个优先级。每个寄存器包含 4 个中断的 5 位的优先级字段。	RW	0	IPR0—0xE000 E400	表 55
				IPR1—0xE000 E404	表 56
				IPR2—0xE000 E408	表 57
				IPR3—0xE000 E40C	表 58
				IPR4—0xE000 E410	表 59
				IPR5—0xE000 E414	表 60
				IPR6—0xE000 E418	表 61
				IPR7—0xE000 E41C	表 62
				IPR8—0xE000 E420	表 63
				IPR9—0xE000 E424	表 64
STIR	软件触发中断寄存器。该寄存器允许软件生成中断。	WO	0	IPR10—0xE000 E428	表 65
				STIR—0xE000 EF00	表 66

6.5.1 中断使能设置寄存器 0 (ISER0—0xE000 E100)

ISER0 寄存器可以使能前 32 个外设中断，或读出这些中断的使能状态。其余中断则通过 ISER1 寄存器使能 (6.5.2 节)。禁用这些中断是通过 ICER0 和 ICER1 寄存器 (6.5.3 节和 5.4 节)。

表45. 中断使能设置寄存器 0 寄存器 (ISER0—0xE000 E100)

位	名称	功能
0	ISE_WDT	看门狗定时器中断使能。 写入：写入 0 无效，写入 1 使能中断。 读取：0 表示中断被禁能，1 表示中断被使能。
1	ISE_TIMER0	定时器 0 中断使能。参见位 0 功能描述。
2	ISE_TIMER1	定时器 1 中断使能。参见位 0 功能描述。
3	ISE_TIMER2	定时器 2 中断使能。参见位 0 功能描述。
4	ISE_TIMER3	定时器 3 中断使能。参见位 0 功能描述。
5	ISE_UART0	UART0 中断使能。参见位 0 功能描述。
6	ISE_UART1	UART1 中断使能。参见位 0 功能描述。
7	ISE_UART2	UART2 中断使能。参见位 0 功能描述。
8	ISE_UART3	UART3 中断使能。参见位 0 功能描述。
9	ISE_PWM1	PWM1 中断使能。参见位 0 功能描述。
10	ISE_I2C0	I2C0 中断使能。参见位 0 功能描述。
11	ISE_I2C1	I2C1 中断使能。参见位 0 功能描述。
12	ISE_I2C2	I2C2 中断使能。参见位 0 功能描述。
13	-	保留。读取值未定义，只写入 0。
14	ISE_SSP0	SSP0 中断使能。参见位 0 功能描述。
15	ISE_SSP1	SSP1 中断使能。参见位 0 功能描述。
16	ISE_PLL0	PLL0 (主 PLL) 中断使能。参见位 0 功能描述。
17	ISE_RTC	实时时钟 (RTC) 和事件监测器/记录器中断使能。参见位 0 描述。
18	ISE_EINT0	外部中断 0 中断使能。参见位 0 功能描述。
19	ISE_EINT1	外部中断 1 中断使能。参见位 0 功能描述。
20	ISE_EINT2	外部中断 2 中断使能。参见位 0 功能描述。
21	ISE_EINT3	外部中断 3 中断使能。参见位 0 功能描述。
22	ISE_ADC	A/D 转换器 (ADC) 中断使能。参见位 0 功能描述。
23	ISE_BOD	掉电检测 (BOD) 中断使能。参见位 0 功能描述。
24	ISE_USB	USB 中断使能。参见位 0 功能描述。
25	ISE_CAN	CAN 中断使能。参见位 0 功能描述。
26	ISE_DMA	GPDMA 中断使能。参见位 0 功能描述。
27	ISE_I2S	I2S 中断使能。参见位 0 功能描述。
28	ISE_ENET	以太网中断使能。参见位 0 功能描述。
29	ISE_SD	SD 卡接口中断使能。参见位 0 功能描述。
30	ISE_MCPWM	电机控制 PWM 中断使能。参见位 0 功能描述。
31	ISE_QEI	正交编码器接口中断使能。参见位 0 功能描述。

6.5.2 中断使能设置寄存器 1 (ISER1—0xE000 E104)

ISER1 寄存器可以使能第二组外设中断，或读出这些中断的使能状态。禁用这些中断是通过 ICER0 和 ICER1 寄存器（[6.5.3](#) 节和 [5.4](#) 节）。

表46. 中断使能设置寄存器 1 寄存器 (ISER1—0xE000 E104)

位	名称	功能
0	ISE_PLL1	PLL1（副 PLL）中断使能。 写入：写入 0 无效，写入 1 使能中断。 读取：0 表示中断被禁能，1 表示中断被使能。
1	ISE_USBACT	USB 活动中断使能。参见位 0 功能描述。
2	ISE_CANACT	CAN 活动中断使能。参见位 0 功能描述。
3	ISE_UART4	UART4 中断使能。参见位 0 功能描述。
4	ISE_SSP2	SSP2 中断使能。参见位 0 功能描述。
5	ISE_LCD	LCD 中断使能。参见位 0 功能描述。
6	ISE_GPIO	GPIO 中断使能。参见位 0 功能描述。
7	ISE_PWM0	PWM0 中断使能。参见位 0 功能描述。
8	ISE_FLASH	Flash 和 EEPROM 中断使能。参见位 0 功能描述。
31:9	-	保留。读取值未定义，只写入 0。

6.5.3 中断使能清除寄存器 0 (ICER0—0xE000 E180)

ICER0 寄存器可以禁用前 32 个外设中断，或读出这些中断的使能状态。其余中断通过 ICER1 寄存器禁用 (5.4 节)。中断的使能要通过 ISER0 和 ISER1 寄存器 (6.5.1 节和 6.5.2 节)。

表47. 中断使能清除寄存器 0 (ICER0—0xE000 E180)

位	名称	功能
0	ICE_WDT	看门狗定时器中断禁能。 写入：写入 0 无效，写入 1 禁能中断。 读取：0 表示中断被禁能，1 表示中断被使能。
1	ICE_TIMER0	定时器 0 中断禁能。参见位 0 功能描述。
2	ICE_TIMER1	定时器 1 中断禁能。参见位 0 功能描述。
3	ICE_TIMER2	定时器 2 中断禁能。参见位 0 功能描述。
4	ICE_TIMER3	定时器 3 中断禁能。参见位 0 功能描述。
5	ICE_UART0	UART0 中断禁能。参见位 0 功能描述。
6	ICE_UART1	UART1 中断禁能。参见位 0 功能描述。
7	ICE_UART2	UART2 中断禁能。参见位 0 功能描述。
8	ICE_UART3	UART3 中断禁能。参见位 0 功能描述。
9	ICE_PWM1	PWM1 中断禁能。参见位 0 功能描述。
10	ICE_I2C0	I2C0 中断禁能。参见位 0 功能描述。
11	ICE_I2C1	I2C1 中断禁能。参见位 0 功能描述。
12	ICE_I2C2	I2C2 中断禁能。参见位 0 功能描述。
13	-	保留。读取值未定义，只写入 0。
14	ICE_SSP0	SSP0 中断禁能。参见位 0 功能描述。
15	ICE_SSP1	SSP1 中断禁能。参见位 0 功能描述。
16	ICE_PLL0	PLL0 (主 PLL) 中断禁能。参见位 0 功能描述。
17	ICE_RTC	RTC 和事件监测器/记录器中断禁能。参见位 0 描述。
18	ICE_EINT0	外部中断 0 中断禁能。参见位 0 功能描述。
19	ICE_EINT1	外部中断 1 中断禁能。参见位 0 功能描述。
20	ICE_EINT2	外部中断 2 中断禁能。参见位 0 功能描述。
21	ICE_EINT3	外部中断 3 中断禁能。参见位 0 功能描述。
22	ICE_ADC	ADC 中断禁能。参见位 0 功能描述。
23	ICE_BOD	BOD 中断禁能。参见位 0 功能描述。
24	ICE_USB	USB 中断禁能。参见位 0 功能描述。
25	ICE_CAN	CAN 中断禁能。参见位 0 功能描述。
26	ICE_DMA	GPDMA 中断禁能。参见位 0 功能描述。
27	ICE_I2S	I2S 中断禁能。参见位 0 功能描述。
28	ICE_ENET	以太网中断禁能。参见位 0 功能描述。
29	ICE_SD	SD 卡接口中断禁能。参见位 0 功能描述。
30	ICE_MCPWM	电机控制 PWM 中断禁能。参见位 0 功能描述。
31	ICE_QEI	正交编码器接口中断禁能。参见位 0 功能描述。

6.5.4 中断使能清除寄存器 1 (ICER1—0xE000 E184)

ICER1 寄存器可以禁用第二组外设中断，或读出这些中断的使能状态。中断的使能要通过 ISER0 和 ISER1 寄存器 (6.5.1 节和 6.5.2 节)。

表48. 中断使能清除寄存器 1 寄存器 (ICER1—0xE000 E184)

位	名称	功能
0	ICE_PLL1	PLL1 (副 PLL) 中断禁能。 写入: 写入 0 无效, 写入 1 禁能中断。 读取: 0 表示中断被禁能, 1 表示中断被使能。
1	ICE_USBACT	USB 活动中断禁能。参见位 0 功能描述。
2	ICE_CANACT	CAN 活动中断禁能。参见位 0 功能描述。
3	ICE_UART4	UART4 中断禁能。参见位 0 功能描述。
4	ICE_SSP2	SSP2 中断禁能。参见位 0 功能描述。
5	ICE_LCD	LCD 中断禁能。参见位 0 功能描述。
6	ICE_GPIO	GPIO 中断禁能。参见位 0 功能描述。
7	ICE_PWM0	PWM0 中断禁能。参见位 0 功能描述。
8	ICE_EEPROM	EEPROM 中断禁能。参见位 0 功能描述。
31:9	-	保留。读取值未定义, 只写入 0。

6.5.5 中断暂挂设置寄存器 0 (ISPR0—0xE000 E200)

ISPR0 寄存器可以设置前 32 个外设中断的暂挂状态，或读出这些中断的暂挂状态。其余中断可以通过 ISPR1 寄存器设置各自的暂挂状态（[6.5.6](#) 节）。清除中断的暂挂状态要通过 ICPR0 和 ICPR1 寄存器（[6.5.7](#) 节和 [6.5.8](#) 节）。

表49. 中断暂挂设置寄存器 0 寄存器 (ISPR0—0xE000 E200)

位	名称	功能
0	ISP_WDT	看门狗定时器中断暂挂设置。 写入：写入 0 无效，写入 1 更改中断状态为暂挂。 读取：0 表示中断未暂挂，1 表示中断暂挂。
1	ISP_TIMER0	定时器 0 中断暂挂设置。参见位 0 功能描述。
2	ISP_TIMER1	定时器 1 中断暂挂设置。参见位 0 功能描述。
3	ISP_TIMER2	定时器 2 中断暂挂设置。参见位 0 功能描述。
4	ISP_TIMER3	定时器 3 中断暂挂设置。参见位 0 功能描述。
5	ISP_UART0	UART0 中断暂挂设置。参见位 0 功能描述。
6	ISP_UART1	UART1 中断暂挂设置。参见位 0 功能描述。
7	ISP_UART2	UART2 中断暂挂设置。参见位 0 功能描述。
8	ISP_UART3	UART3 中断暂挂设置。参见位 0 功能描述。
9	ISP_PWM1	PWM1 中断暂挂设置。参见位 0 功能描述。
10	ISP_I2C0	I ² C0 中断暂挂设置。参见位 0 功能描述。
11	ISP_I2C1	I ² C1 中断暂挂设置。参见位 0 功能描述。
12	ISP_I2C2	I ² C2 中断暂挂设置。参见位 0 功能描述。
13	-	保留。读取值未定义，只写入 0。
14	ISP_SSP0	SSP0 中断暂挂设置。参见位 0 功能描述。
15	ISP_SSP1	SSP1 中断暂挂设置。参见位 0 功能描述。
16	ISP_PLL0	PLL0（主 PLL）中断暂挂设置。参见位 0 功能描述。
17	ISP_RTC	RTC 和事件监测器/记录器中断暂挂设置。参见位 0 描述。
18	ISP_EINT0	外部中断 0 中断暂挂设置。参见位 0 功能描述。
19	ISP_EINT1	外部中断 1 中断暂挂设置。参见位 0 功能描述。
20	ISP_EINT2	外部中断 2 中断暂挂设置。参见位 0 功能描述。
21	ISP_EINT3	外部中断 3 中断暂挂设置。参见位 0 功能描述。
22	ISP_ADC	ADC 中断暂挂设置。参见位 0 功能描述。
23	ISP_BOD	BOD 中断暂挂设置。参见位 0 功能描述。
24	ISP_USB	USB 中断暂挂设置。参见位 0 功能描述。
25	ISP_CAN	CAN 中断暂挂设置。参见位 0 功能描述。
26	ISP_DMA	GPDMA 中断暂挂设置。参见位 0 功能描述。
27	ISP_I2S	I ² S 中断暂挂设置。参见位 0 功能描述。
28	ISP_ENET	以太网中断暂挂设置。参见位 0 功能描述。
29	ISP_SD	SD 卡接口中断暂挂设置。参见位 0 功能描述。
30	ISP_MCPWM	电机控制 PWM 中断暂挂设置。参见位 0 功能描述。
31	ISP_QEI	正交编码器接口中断暂挂设置。参见位 0 功能描述。

6.5.6 中断暂挂设置寄存器 1 (ISPR1—0xE000 E204)

ISPR1 寄存器可以设置第二组外设中断的暂挂状态，或读出这些中断的暂挂状态。清除中断的暂挂状态要通过 ICPR0 和 ICPR1 寄存器（[6.5.7](#) 节和 [6.5.8](#) 节）。

表50. 中断暂挂设置寄存器 1 寄存器 (ISPR1—0xE000 E204)

位	名称	功能
0	ISP_PLL1	PLL1（备 PLL）中断暂挂设置。 写入：写入 0 无效，写入 1 更改中断状态为暂挂。 读取：0 表示中断未暂挂，1 表示中断暂挂。
1	ISP_USBACT	USB 活动中断暂挂设置。参见位 0 功能描述。
2	ISP_CANACT	CAN 活动中断暂挂设置。参见位 0 功能描述。
3	ISP_UART4	UART4 中断暂挂设置。参见位 0 功能描述。
4	ISP_SSP2	SSP2 中断暂挂设置。参见位 0 功能描述。
5	ISP_LCD	LCD 中断暂挂设置。参见位 0 功能描述。
6	ISP_GPIO	GPIO 中断暂挂设置。参见位 0 功能描述。
7	ISP_PWM0	PWM0 中断暂挂设置。参见位 0 功能描述。
8	ISP_EEPROM	EEPROM 中断暂挂设置。参见位 0 功能描述。
31:9	-	保留。读取值未定义，只写入 0。

6.5.7 中断暂挂清除寄存器 0 (ICPR0—0xE000 E280)

ICPR0 寄存器可以清除前 32 个外设中断的暂挂状态，或读出这些中断的暂挂状态。其余中断可以通过 ICPR1 寄存器清除其暂挂状态 (6.5.8 节)。中断暂挂状态的设置要通过 ISPR0 和 ISPR1 寄存器 (6.5.5 节和 6.5.6 节)。

表51. 中断暂挂清除寄存器 0 寄存器 (ICPR0—0xE000 E280)

位	名称	功能
0	ICP_WDT	看门狗定时器中断暂挂清除。 写入：写入 0 无效，写入 1 更改中断状态为未暂挂。 读取：0 表示中断未暂挂，1 表示中断暂挂。
1	ICP_TIMER0	定时器 0 中断暂挂清除。参见位 0 功能描述。
2	ICP_TIMER1	定时器 1 中断暂挂清除。参见位 0 功能描述。
3	ICP_TIMER2	定时器 2 中断暂挂清除。参见位 0 功能描述。
4	ICP_TIMER3	定时器 3 中断暂挂清除。参见位 0 功能描述。
5	ICP_UART0	UART0 中断暂挂清除。参见位 0 功能描述。
6	ICP_UART1	UART1 中断暂挂清除。参见位 0 功能描述。
7	ICP_UART2	UART2 中断暂挂清除。参见位 0 功能描述。
8	ICP_UART3	UART3 中断暂挂清除。参见位 0 功能描述。
9	ICP_PWM1	PWM1 中断暂挂清除。参见位 0 功能描述。
10	ICP_I2C0	I2C0 中断暂挂清除。参见位 0 功能描述。
11	ICP_I2C1	I2C1 中断暂挂清除。参见位 0 功能描述。
12	ICP_I2C2	I2C2 中断暂挂清除。参见位 0 功能描述。
13	-	保留。读取值未定义，只写入 0。
14	ICP_SSP0	SSP0 中断暂挂清除。参见位 0 功能描述。
15	ICP_SSP1	SSP1 中断暂挂清除。参见位 0 功能描述。
16	ICP_PLL0	PLL0 (主 PLL) 中断暂挂清除。参见位 0 功能描述。
17	ICP_RTC	RTC 和事件监测器/记录器中断暂挂清除。参见位 0 描述。
18	ICP_EINT0	外部中断 0 中断暂挂清除。参见位 0 功能描述。
19	ICP_EINT1	外部中断 1 中断暂挂清除。参见位 0 功能描述。
20	ICP_EINT2	外部中断 2 中断暂挂清除。参见位 0 功能描述。
21	ICP_EINT3	外部中断 3 中断暂挂清除。参见位 0 功能描述。
22	ICP_ADC	ADC 中断暂挂清除。参见位 0 功能描述。
23	ICP_BOD	BOD 中断暂挂清除。参见位 0 功能描述。
24	ICP_USB	USB 中断暂挂清除。参见位 0 功能描述。
25	ICP_CAN	CAN 中断暂挂清除。参见位 0 功能描述。
26	ICP_DMA	GPDMA 中断暂挂清除。参见位 0 功能描述。
27	ICP_I2S	I2S 中断暂挂清除。参见位 0 功能描述。
28	ICP_ENET	以太网中断暂挂清除。参见位 0 功能描述。
29	ICP_SD	SD 卡接口中断暂挂清除。参见位 0 功能描述。
30	ICP_MCPWM	电机控制 PWM 中断暂挂清除。参见位 0 功能描述。
31	ICP_QEI	正交编码器接口中断暂挂清除。参见位 0 功能描述。

6.5.8 中断暂挂清除寄存器 1（ICPR1—0xE000 E284）

ICPR1 寄存器可以清除第二组外设中断的暂挂状态，或读出这些中断的暂挂状态。中断暂挂状态的设置要通过 ISPR0 和 ISPR1 寄存器（[6.5.5](#) 节和 [6.5.6](#) 节）。

表52. 中断清除暂挂寄存器 1 寄存器（ICPR1—0xE000 E284）

位	名称	功能
0	ICP_PLL1	PLL1（副 PLL）中断暂挂清除。 写入：写入 0 无效，写入 1 更改中断状态为未暂挂。 读取：0 表示中断未暂挂，1 表示中断暂挂。
1	ICP_USBACT	USB 活动中断暂挂清除。参见位 0 功能描述。
2	ICP_CANACT	CAN 活动中断暂挂清除。参见位 0 功能描述。
3	ICP_UART4	UART4 中断暂挂清除。参见位 0 功能描述。
4	ICP_SSP2	SSP2 中断暂挂清除。参见位 0 功能描述。
5	ICP_LCD	LCD 中断暂挂清除。参见位 0 功能描述。
6	ICP_GPIO	GPIO 中断暂挂清除。参见位 0 功能描述。
7	ICP_PWM0	PWM0 中断暂挂清除。参见位 0 功能描述。
8	ICP_EEPROM	EEPROM 中断暂挂清除。参见位 0 功能描述。
31:9	-	保留。读取值未定义，只写入 0。

6.5.9 中断活动位寄存器 0 (IABR0—0xE000 E300)

IABR0 是一个只读寄存器,用于读出前 32 个外设中断的活动状态。当中断服务程序执行时, IABR 中的相应位置位。其它中断可以通过 IABR1 寄存器读出各自的活动状态(6.5.10 节)。

表53. 中断活动位寄存器 0 (IABR0—0xE000 E300)

位	名称	功能
0	IAB_WDT	看门狗定时器中断激活。 读取: 0 表示中断未激活, 1 表示中断激活。
1	IAB_TIMER0	定时器 0 中断激活。参见位 0 功能描述。
2	IAB_TIMER1	定时器 1 中断激活。参见位 0 功能描述。
3	IAB_TIMER2	定时器 2 中断激活。参见位 0 功能描述。
4	IAB_TIMER3	定时器 3 中断激活。参见位 0 功能描述。
5	IAB_UART0	UART0 中断激活。参见位 0 功能描述。
6	IAB_UART1	UART1 中断激活。参见位 0 功能描述。
7	IAB_UART2	UART2 中断激活。参见位 0 功能描述。
8	IAB_UART3	UART3 中断激活。参见位 0 功能描述。
9	IAB_PWM1	PWM1 中断激活。参见位 0 功能描述。
10	IAB_I2C0	I2C0 中断激活。参见位 0 功能描述。
11	IAB_I2C1	I2C1 中断激活。参见位 0 功能描述。
12	IAB_I2C2	I2C2 中断激活。参见位 0 功能描述。
13	-	保留。读取值未定义, 只写入 0。
14	IAB_SSP0	SSP0 中断激活。参见位 0 功能描述。
15	IAB_SSP1	SSP1 中断激活。参见位 0 功能描述。
16	IAB_PLL0	PLL0 (主 PLL) 中断激活。参见位 0 功能描述。
17	IAB_RTC	RTC 和事件监测器/记录器中断激活。参见位 0 描述。
18	IAB_EINT0	外部中断 0 中断激活。参见位 0 功能描述。
19	IAB_EINT1	外部中断 1 中断激活。参见位 0 功能描述。
20	IAB_EINT2	外部中断 2 中断激活。参见位 0 功能描述。
21	IAB_EINT3	外部中断 3 中断激活。参见位 0 功能描述。
22	IAB_ADC	ADC 中断激活。参见位 0 功能描述。
23	IAB_BOD	BOD 中断激活。参见位 0 功能描述。
24	IAB_USB	USB 中断激活。参见位 0 功能描述。
25	IAB_CAN	CAN 中断激活。参见位 0 功能描述。
26	IAB_DMA	GPDMA 中断激活。参见位 0 功能描述。
27	IAB_I2S	I2S 中断激活。参见位 0 功能描述。
28	IAB_ENET	以太网中断激活。参见位 0 功能描述。
29	IAB_SD	重复中断定时器中断激活。参见位 0 功能描述。
30	IAB_MCPWM	电机控制 PWM 中断激活。参见位 0 功能描述。
31	IAB_QEI	正交编码器接口中断激活。参见位 0 功能描述。

6.5.10 中断活动位寄存器 1 (IABR1—0xE000 E304)

IABR1 是一个只读寄存器，用于读出第二组外设中断的活动状态。当中断服务程序执行时，IABR 中的相应位置位。

表54. 中断活动位寄存器 1 (IABR1—0xE000 E304)

位	名称	功能
0	IAB_PLL1	PLL0（副 PLL）中断激活。 读取：0 表示中断未激活，1 表示中断激活。
1	IAB_USBACT	USB 活动中断激活。参见位 0 功能描述。
2	IAB_CANACT	CAN 活动中断激活。参见位 0 功能描述。
3	IAB_UART4	UART4 中断激活。参见位 0 功能描述。
4	IAB_SSP2	SSP2 中断激活。参见位 0 功能描述。
5	IAB_LCD	LCD 中断激活。参见位 0 功能描述。
6	IAB_GPIO	GPIO 中断激活。参见位 0 功能描述。
7	IAB_PWM0	PWM0 中断激活。参见位 0 功能描述。
8	IAB_EEPROM	EEPROM 中断激活。参见位 0 功能描述。
31:9	-	保留。从保留位读出的值未被定义。

6.5.11 中断优先级寄存器 0 (IPR0—0xE000 E400)

IPR0 寄存器控制着前 4 个外设中断的优先级。每个中断可以拥有 32 个优先级中的一个，其中 0 为最高优先级。

表55. 中断优先级寄存器 0 (IPR0—0xE000 E400)

位	名称	功能
2:0	未执行	这些位忽略写入，读取值为 0。
7:3	IP_WDT	看门狗定时器中断优先级。0=最高优先级。31(0x1F)=最低优先级。
10:8	未执行	这些位忽略写入，读取值为 0。
15:11	IP_TIMER0	定时器 0 中断优先级。参见位 7-3 功能描述。
18:16	未执行	这些位忽略写入，读取值为 0。
23:19	IP_TIMER1	定时器 1 中断优先级。参见位 7-3 功能描述。
26:24	未执行	这些位忽略写入，读取值为 0。
31:27	IP_TIMER2	定时器 2 中断优先级。参见位 7-3 功能描述。

6.5.12 中断优先级寄存器 1 (IPR1—0xE000 E404)

IPR1 寄存器控制着第二组 4 个外设中断的优先级。每个中断可以拥有 32 个优先级中的一个，其中 0 为最高优先级。

表56. 中断优先级寄存器 1 (IPR1—0xE000 E404)

位	名称	功能
2:0	未执行	这些位忽略写入，读取值为 0。
7:3	IP_TIMER3	定时器 3 中断优先级。0=最高优先级。31(0x1F)=最低优先级。
10:8	未执行	这些位忽略写入，读取值为 0。
15:11	IP_UART0	UART0 中断优先级。参见位 7-3 功能描述。
18:16	未执行	这些位忽略写入，读取值为 0。
23:19	IP_UART1	UART1 中断优先级。参见位 7-3 功能描述。
26:24	未执行	这些位忽略写入，读取值为 0。
31:27	IP_UART2	UART2 中断优先级。参见位 7-3 功能描述。

6.5.13 中断优先级寄存器 2 (IPR2—0xE000 E408)

IPR2 寄存器控制着第三组 4 个外设中断的优先级。每个中断可以拥有 32 个优先级中的一个，其中 0 为最高优先级。

表57. 中断优先级寄存器 2 (IPR2—0xE000 E408)

位	名称	功能
2:0	未执行	这些位忽略写入，读取值为 0。
7:3	IP_UART3	UART3 中断优先级。0=最高优先级。31(0x1F)=最低优先级。
10:8	未执行	这些位忽略写入，读取值为 0。
15:11	IP_PWM1	PWM1 中断优先级。参见位 7-3 功能描述。
18:16	未执行	这些位忽略写入，读取值为 0。
23:19	IP_I2C0	I ² C0 中断优先级。参见位 7-3 功能描述。
26:24	未执行	这些位忽略写入，读取值为 0。
31:27	IP_I2C1	I ² C1 中断优先级。参见位 7-3 功能描述。

6.5.14 中断优先级寄存器 3 (IPR3—0xE000 E40C)

IPR3 寄存器控制着第四组 4 个外设中断的优先级。每个中断可以拥有 32 个优先级中的一个，其中 0 为最高优先级。

表58. 中断优先级寄存器 3 (IPR3—0xE000 E40C)

位	名称	功能
2:0	未执行	这些位忽略写入，读取值为 0。
7:3	IP_I2C2	I ² C2 中断优先级。0=最高优先级。31(0x1F)=最低优先级。
10:8	未执行	这些位忽略写入，读取值为 0。
15:11	-	保留。读取值未定义，只写入 0。
18:8	未执行	这些位忽略写入，读取值为 0。
23:19	IP_SSP0	SSP0 中断优先级。参见位 7-3 功能描述。
26:24	未执行	这些位忽略写入，读取值为 0。
31:27	IP_SSP1	SSP1 中断优先级。参见位 7-3 功能描述。

6.5.15 中断优先级寄存器 4 (IPR4—0xE000 E410)

IPR4 寄存器控制着第五组 4 个外设中断的优先级。每个中断可以拥有 32 个优先级中的一个，其中 0 为最高优先级。

表59. 中断优先级寄存器 4 (IPR4—0xE000 E410)

位	名称	功能
2:0	未执行	这些位忽略写入，读取值为 0。
7:3	IP_PLL0	PLL0（主 PLL）中断优先级。0=最高优先级。31(0x1F)=最低优先级。
10:8	未执行	这些位忽略写入，读取值为 0。
15:11	IP_RTC	RTC 中断优先级。参见位 7-3 功能描述。
18:16	未执行	这些位忽略写入，读取值为 0。
23:19	IP_EINT0	外部中断 0 中断优先级。参见位 7-3 功能描述。
26:24	未执行	这些位忽略写入，读取值为 0。
31:27	IP_EINT1	外部中断 1 中断优先级。参见位 7-3 功能描述。

6.5.16 中断优先级寄存器 5 (IPR5—0xE000 E414)

IPR5 寄存器控制着第六组 4 个外设中断的优先级。每个中断可以拥有 32 个优先级中的一个，其中 0 为最高优先级。

表60. 中断优先级寄存器 5 (IPR5—0xE000 E414)

位	名称	功能
2:0	未执行	这些位忽略写入，读取值为 0。
7:3	IP_EINT2	外部中断 2 中断优先级。0=最高优先级。31(0x1F)=最低优先级。
10:8	未执行	这些位忽略写入，读取值为 0。
15:11	IP_EINT3	外部中断 3 中断优先级。参见位 7-3 功能描述。
18:16	未执行	这些位忽略写入，读取值为 0。
23:19	IP_ADC	ADC 中断优先级。参见位 7-3 功能描述。
26:24	未执行	这些位忽略写入，读取值为 0。
31:27	IP_BOD	BOD 中断优先级。参见位 7-3 功能描述。

6.5.17 中断优先级寄存器 6 (IPR6—0xE000 E418)

IPR6 寄存器控制着第七组 4 个外设中断的优先级。每个中断可以拥有 32 个优先级中的一个，其中 0 为最高优先级。

表61. 中断优先级寄存器 6 (IPR6—0xE000 E418)

位	名称	功能
2:0	未执行	这些位忽略写入，读取值为 0。
7:3	IP_USB	USB 中断优先级。0=最高优先级。31(0x1F)=最低优先级。
10:8	未执行	这些位忽略写入，读取值为 0。
15:11	IP_CAN	CAN 中断优先级。参见位 7-3 功能描述。
18:16	未执行	这些位忽略写入，读取值为 0。
23:19	IP_DMA	GPDMA 中断优先级。参见位 7-3 功能描述。
26:24	未执行	这些位忽略写入，读取值为 0。
31:27	IP_I2S	I ² S 中断优先级。参见位 7-3 功能描述。

6.5.18 中断优先级寄存器 7 (IPR7—0xE000 E41C)

IPR7 寄存器控制着第八组 4 个外设中断的优先级。每个中断可以拥有 32 个优先级中的一个，其中 0 为最高优先级。

表62. 中断优先级寄存器 7 (IPR7—0xE000 E41C)

位	名称	功能
2:0	未执行	这些位忽略写入，读取值为 0。
7:3	IP_ENET	以太网中断优先级。0=最高优先级。31(0x1F)=最低优先级。
10:8	未执行	这些位忽略写入，读取值为 0。
15:11	IP_SD	SD 卡接口中断优先级。参见位 7-3 功能描述。
18:16	未执行	这些位忽略写入，读取值为 0。
23:19	IP_MCPWM	电机控制 PWM 中断优先级。参见位 7-3 功能描述。
26:24	未执行	这些位忽略写入，读取值为 0。
31:27	IP_QEI	正交编码器接口中断优先级。参见位 7-3 功能描述。

6.5.19 中断优先级寄存器 8 (IPR8—0xE000 E420)

IPR8 寄存器控制着第九组 4 个外设中断的优先级。每个中断可以拥有 32 个优先级中的一个，其中 0 为最高优先级。

表63. 中断优先级寄存器 8 (IPR8—0xE000 E420)

位	名称	功能
2:0	未执行	这些位忽略写入，读取值为 0。
7:3	IP_PLL1	PLL1 (副 PLL) 中断优先级。0=最高优先级。31(0x1F)=最低优先级。
10:8	未执行	这些位忽略写入，读取值为 0。
15:11	IP_USBACT	USB 活动中断优先级。参见位 7-3 功能描述。
18:16	未执行	这些位忽略写入，读取值为 0。
23:19	IP_CANACT	CAN 活动中断优先级。参见位 7-3 功能描述。
26:24	未执行	这些位忽略写入，读取值为 0。
31:27	IP_UART4	UART4 中断优先级。参见位 7-3 功能描述。

6.5.20 中断优先级寄存器 9 (IPR9—0xE000 E424)

IPR9 寄存器控制着第十组 4 个外设中断的优先级。每个中断可以拥有 32 个优先级中的一个，其中 0 为最高优先级。

表64. 中断优先级寄存器 9 (IPR9—0xE000 E424)

位	名称	功能
2:0	未执行	这些位忽略写入，读取值为 0。
7:3	IP_SSP2	SSP2 中断优先级。0=最高优先级。31(0x1F)=最低优先级。
10:8	未执行	这些位忽略写入，读取值为 0。
15:11	IP_LCD	LCD 控制器中断优先级。参见位 7-3 功能描述。
18:16	未执行	这些位忽略写入，读取值为 0。
23:19	IP_GPIO	GPIO 中断优先级。参见位 7-3 功能描述。
26:24	未执行	这些位忽略写入，读取值为 0。
31:27	IP_PWM0	PWM0 中断优先级。参见位 7-3 功能描述。

6.5.21 中断优先级寄存器 10（IPR10—0xE000 E428）

IPR10 寄存器控制着第十一组 4 个外设中断的优先级。每个中断可以拥有 32 个优先级中的一个，其中 0 为最高优先级。

表65. 中断优先级寄存器 10（IPR10—0xE000 E428）

位	名称	功能
2:0	未执行	这些位忽略写入，读取值为 0。
7:3	IP_EEPROM	EEPROM 编程中断。0=最高优先级。31(0x1F)=最低优先级。
31:8	未执行	这些位忽略写入，读取值为 0。

6.5.22 软件触发中断寄存器（STIR—0xE000 EF00）

除了使用 ISPR 寄存器以外，STIR 寄存器提供了一种让软件产生中断的方式。这种机制仅能用于产生外设中断，而不是系统异常。

默认情况下，仅特权软件可以写入 STIR 寄存器。如果特权软件将 CCR 寄存器中的 USERSETMPEND 位置位，则无特权软件也可以赋予这种能力（见 [39.4.2.8](#) 节和 [39.4.3.8](#) 节）。

表66. 软件触发中断寄存器（STIR—0xE000 EF00）

位	名称	功能
8:0	INTID	向该字段写入值，在指定中断 ID 生成中断（参见 表 43 ）。
31:9	-	保留。读取值未定义，只写入 0。

7.1 LPC178x/177x 管脚配置

有关各个 LPC178x/177x 器件的信息，请见具体数据表。[表 67](#) 按照管脚名排序，列出了所有管脚，并且包含了相关管脚功能的描述。

每种 LPC178x/177x 器件对所需功能的管脚配置方法，见 IOCON 寄存器 ([8.4.1](#) 节)。

LPC178x/177x 的 I/O 脚均能承受 5V，除下表中特别指出以外，这些管脚均有输入滞后。晶振管脚、电源管脚以及基准电压管脚不可承受 5V。另外，当管脚被选做模数转换器输入端时，它们将不再能承受 5V，而必须限制为 ADC 正基准管脚上的电压 (VREFP)。

表67. LPC178x/177x 管脚描述

符号	类型	IOCON	选择 ^[1]	描述
P0[0] to P0[31]	I/O			Port 0: 端口 0 根据封装最多可以提供 32 个 I/O 管脚。每个管脚都有各自的方向控制、管脚模式配置和功能选择。
P0[0]/ CAN_RD1/	I/O	0		P0[0] —通用数字 I/O 管脚。
U3_TXD/	I	1		CAN_RD1 —CAN1 接收器输入。
I2C1_SDA/	O	2		U3_TXD —UART3 的发送器输出。
U0_TXD	I/O	3		I2C1_SDA —I2C1 数据 I/O (该管脚不使用专门的 I2C 焊盘, 详情参见 22.1 节)。
	I/O	4		U0_TXD —UART 0 的发送器输出。
P0[1]/ CAN1_TD/	I/O	0		P0[1] —通用数字 I/O 管脚。
U3_RXD/	O	1		CAN1_TD —CAN1 发送器输出。
I2C1_SCL/	I	2		U3_RXD —UART3 的接收器输入。
U0_RXD	I/O	3		I2C1_SCL —I2C1 时钟 I/O (该管脚不使用专门的 I2C 焊盘, 详情参见 22.1 节)。
	I	4		U0_RXD —UART0 的接收器输入。
P0[2]/ U0_TXD/	I/O	0		P0[2] —通用数字 I/O 管脚。
U3_TXD	O	1		U0_TXD —UART 0 的发送器输出。用于 ISP 通信参见 37.1 节。
	O	2		U3_TXD —UART3 的发送器输出。
P0[3]/ U0_RXD/	I/O	0		P0[3] —通用数字 I/O 管脚。
U3_RXD	I	1		U0_RXD —UART0 的接收器输入。用于 ISP 通信参见 37.1 节。
	I	2		U3_RXD —UART3 的接收器输入。
P0[4]/	I/O	0		P0[4] —通用数字 I/O 管脚。
I2S_RX_SCK/	I/O	1		I2S_RX_SCK —I2S 接收时钟。它由主机驱动, 从机接收。该信号对应于《I2S-总线规范》中的信号 SCK。
CAN_RD2/				
T2_CAP0/	I	2		CAN_RD2 —CAN1 接收器输入。
LCD_VD[0]	I	3		T2_CAP0 —定时器 2 的捕获输入, 通道 0。
	O	7		LCD_VD[0] —LCD 数据。
P0[5]/ I2S_RX_WS/	I/O	0		P0[5] —通用数字 I/O 管脚。
CAN_TD2/	I/O	1		I2S_RX_WS —I2S 接收字选。它由主机驱动, 从机接收。该信号对应于《I2S-总线规范》中的信号 WS。
T2_CAP1/	O	2		CAN_TD2 —CAN2 发送器输出。
LCD_VD[1]	I	3		T2_CAP1 —定时器 2 的捕获输入, 通道 1。
	O	7		LCD_VD[1] —LCD 数据。
P0[6]/	I/O	0		P0[6] —通用数字 I/O 管脚。
I2S_RX_SDA/	I/O	1		I2S_RX_SDA —I2S 接收数据。它由发送器驱动, 接收器读取。该信号对应于《I2S-总线规范》中的信号 SD。
SSP1_SSEL/				
T2_MAT0/ U1_RTS/	I/O	2		SSP1_SSEL1 —SSP1 的从选择。
LCD_VD[8]	O	3		T2_MAT0 —定时器 2 的匹配输出, 通道 0。
		4		U1_RTS —UART1 的请求发送输出。该信号也可以被配置为 UART1 的 RS-485/EIA-485 输出使能信号。
	O	7		LCD_VD[8] —LCD 数据。
P0[7]/	I/O	0		P0[7] —通用数字 I/O 管脚。
I2S_TX_SCK/	I/O	1		I2S_TX_SCK —I2S 发送时钟。它由主机驱动, 从机接收。该信号对应于《I2S-总线规范》中的信号 SCK。
SSP1_SCK/				
T2_MAT1/	I/O	2		SSP1_SCK —SSP1 的 SCK。
RTC_EV0/	O	3		T2_MAT1 —定时器 2 的匹配输出, 通道 1。
LCD_VD[9]	I	4		RTC_EV0 —事件监测器/记录器的事件输入 0。
	O	7		LCD_VD[9] —LCD 数据。
P0[8]/ I2S_TX_WS/	I/O	0		P0[8] —通用数字 I/O 管脚。
SSP1_MISO/	I/O	1		I2S_TX_WS —I2S 发送字选。它由主机驱动, 从机接收。该信号对应于《I2S-总线规范》中的信号 WS。
T2_MAT2/				

符号	类型	IOCON 选择 ¹	描述
RTC_EV1/ LCD_VD[16]	I/O	2	SSP1_MISO —SSP1 的主机输入, 从机输出。
	O	3	RTC_EV1 —事件监测器/记录器的事件输入 1。
	I	4	P0[8] —通用数字 I/O 管脚。
	O	7	LCD_VD[16] —LCD 数据。
P0[9]/ I2S_TX_SDA/ SSP1_MOSI/ T2_MAT3/ RTC_EV2/ LCD_VD[17]	I/O	0	P0[9] —通用数字 I/O 管脚。
	I/O	1	I2S_TX_SDA —I2S 发送数据。它由发送器驱动, 接收器读取。该信号对应于《I2S-总线规范》中的信号 SD。
	I/O	2	SSP1_MOSI —SSP1 的主机输出, 从机输入。
	O	3	T2_MAT3 —定时器 2 的匹配输出, 通道 3。
	I	4	RTC_EV2 —事件监测器/记录器的事件输入 2。
	O	7	LCD_VD[17] —LCD 数据。
P0[10]/ U2_TXD/ I2C2_SDA/ T3_MAT0	I/O	0	P0[10] —通用数字 I/O 管脚。
	O	1	U2_TXD —UART2 的发送器输出。
	I/O	2	I2C2_SDA —I2C2 数据 I/O (该管脚不使用专门的 I2C 焊盘, 详见 22.1 节)。
	O	3	T3_MAT0 —定时器 3 的匹配输出, 通道 0。
P0[11]/ U2_RXD/ I2C2_SCL/ T3_MAT1	I/O	0	P0[11] —通用数字 I/O 管脚。
	I	1	U2_RXD —UART2 的接收器输入。
	I/O	2	I2C2_SCL —I2C2 时钟 I/O (该管脚不使用专门的 I2C 焊盘, 详见 22.1 节)。
	O	3	T3_MAT1 —定时器 3 的匹配输出, 通道 1。
P0[12]/ USB_PPWR2 /	I/O	0	P0[12] —通用数字 I/O 管脚。
	O	1	USB_PPWR2 —USB 端口 2 的端口功率使能信号。
SSP1_MISO/ AD0[6]	I/O	2	SSP1_MISO — SSP1 的主机输入, 从机输出。
	I	3	AD0[6] —A/D 转换器 0, 输入 6。当配置为 ADC 输入时, 管脚的数字功能必须禁用 (详见 8.4.1 节)。
P0[13]/ USB_UP_LED2/ SSP1_MOSI/ AD0[7]	I/O	0	P0[13] —通用数字 I/O 管脚。
	O	1	USB_UP_LED2 —USB 端口 2 GoodLink LED 指示器。当设备被配置时 (非控制端点使能), 它为低电平。当设备没有被配置或在全局挂起期间, 它为高电平。
	I/O	2	SSP1_MOSI —SSP1 的主机输出, 从机输入。
	I	3	AD0[7] —A/D 转换器 0, 输入 7。当配置为 ADC 输入时, 管脚的数字功能必须禁用 (详见 8.4.1 节)。
P0[14]/ USB_HSTEN2 /	I/O	0	P0[14] —通用数字 I/O 管脚。
	O	1	USB_HSTEN2 —USB 端口 2 的主机使能状态。
SSP1_SSEL/ USB_CONNECT2	I/O	2	SSP1_SSEL —SSP1 的从选择。
	O	3	USB_CONNECT2 —USB 端口 2 的 SoftConnect 控制。在软件控制下, 该信号用于转换一个 1.5kΩ 的外部电阻。它可以与 SoftConnect USB 特性配合使用。
P0[15]/ U1_TXD/ SSP0_SCK	I/O	0	P0[15] —通用数字 I/O 管脚。
	O	1	U1_TXD —UART1 的发送器输出。
	I/O	2	SSP0_SCK —SSP0 的 SCK。
P0[16]/ U1_RXD/ SSP0_SSEL/	I/O	0	P0[16] —通用数字 I/O 管脚。
	I	1	U1_RXD —UART1 的接收器输入。
	I/O	2	SSP0_SSEL —SSP0 的从选择。
P0[17]/ U1_CTS/ SSP0_MISO	I/O	0	P0[17] —通用数字 I/O 管脚。
	I	1	U1_CTS —UART1 的清除发送输入。
	I/O	2	SSP1_MISO —SSP0 的主机输入, 从机输出。
P0[18]/ U1_DCD/ SSP0_MOSI	I/O	0	P0[18] —通用数字 I/O 管脚。
	I	1	U1_DCD —UART1 的数据载波检测输入。
	I/O	2	SSP0_MOSI —SSP0 的主机输出, 从机输入。
P0[19]/ U1_DSR/ SD_CLK/	I/O	0	P0[19] —通用数字 I/O 管脚。
	I	1	U1_DSR —UART1 的数据设置就绪输入。

符号	类型	IOCON 选择 ^[1]	描述
I2C1_SDA	O	2	SD_CLK —SD 卡接口的时钟输出线。
	I/O	3	I2C1_SDA —I2C1 数据 I/O（该管脚不使用专门的 I2C 焊盘，详情参见 22.1 节）。
P0[20]/ U1_DTR/ SD_CMD/ I2C1_SCL	I/O	0	P0[20] —通用数字 I/O 管脚。
	O	1	U1_DTR —UART1 的数据终端就绪输出。该信号也可以被配置为 UART1 的 RS-485/EIA-485 输出使能信号。
	I/O	2	SD_CMD —SD 卡接口的命令行。
	I/O	3	I2C1_SCL —I2C1 时钟 I/O（该管脚不使用专门的 I2C 焊盘，详情参见 22.1 节）。
P0[21]/ U1_RI/ SD_PWR/ U4_OE/ CAN_RD1	I/O	0	P0[21] —通用数字 I/O 管脚。
	I	1	U1_RI —UART1 的振铃指示输入。
	O	2	SD_PWR —外部 SD 卡电源的电源使能信号。
	O	3	U4_OE —UART4 的 RS-485/EIA-485 输出使能信号。
	I	4	CAN_RD1 —CAN1 接收器输入。
	I/O	5	U4_SCL —同步模式下 UART4 时钟输入或输出。
P0[22]/ U1_RTS/ SD_DAT[0]/ U4_TXD/ CAN_TD1	I/O	0	P0[22] —通用数字 I/O 管脚。
	O	1	U1_RTS —UART1 的请求发送输出。
	I/O	2	SD_DAT[0] —SD 卡接口的数据线 0。
	O	3	U4_TXD —UART4 的发送器输出（在智能卡模式下 I/O）。
	O	4	CAN_TD1 —CAN1 发送器输出。
P0[23]/ AD0[0]/ I2S_RX_SCK/ T3_CAP0	I/O	0	P0[23] —通用数字 I/O 管脚。
	I	1	AD0[0] —A/D 转换器 0，输入 0。当配置为 ADC 输入时，管脚的数字功能必须禁用（详见 8.4.1 节）。
	I/O	2	I2S_RX_SCK —接收时钟。它由主机驱动，从机接收。该信号对应于《I2S-总线规范》中的信号 SCK。
	I	3	T3_CAP0 —定时器 3 的捕获输入，通道 0。
P0[24]/ AD0[1]/ I2S_RX_WS/ T3_CAP1	I/O	0	P0[24] —通用数字 I/O 管脚。
	I	1	AD0[1] —A/D 转换器 0，输入 1。当配置为 ADC 输入时，管脚的数字功能必须禁用（详见 8.4.1 节）。
	I/O	2	I2S_RX_WS —接收字选。它由主机驱动，从机接收。该信号对应于《I2S-总线规范》中的信号 WS。
	I	3	T3_CAP1 —定时器 3 的捕获输入，通道 1。
P0[25]/ AD0[2]/ I2S_RX_SDA/ U3_TXD	I/O	0	P0[25] —通用数字 I/O 管脚。
	I	1	AD0[2] —A/D 转换器 0，输入 2。当配置为 ADC 输入时，管脚的数字功能必须禁用（详见 8.4.1 节）。
	I/O	2	I2S_RX_SDA —接收数据。它由发送器驱动，接收器读取。该信号对《I2S-总线规范》中的信号 SD。
	O	3	U3_TXD —UART3 的发送器输出。
P0[26]/ AD0[3]/ DAC_OUT/ U3_RXD	I/O	0	P0[26] —通用数字 I/O 管脚。
	I	1	AD0[3] —A/D 转换器 0，输入 3。当配置为 ADC 输入时，管脚的数字功能必须禁用（详见 8.4.1 节）。
	O	2	DAC_OUT —D/A 转换器输出。当配置成 DAC 输出时，管脚的数字功能必须禁用（详见 8.4.1 节）。
	I	3	U3_RXD —UART3 的接收器输入。
P0[27]/ I2C0_SDA/ USB_SDA	I/O	0	P0[27] —通用数字 I/O 管脚。
	I/O	1	I2C0_SDA —I2C0 数据 I/O。（该管脚使用专门的 I2C 焊盘，详见 22.1 节）。
	I/O	2	USB_SDA —I2C 串行数据，支持外部 USB 收发器通讯。
P0[28]/ I2C0_SCL/ USB_SCL	I/O	0	P0[28] —通用数字 I/O 管脚。
	I/O	1	I2C0_SCL0 —I2C0 时钟 I/O（该管脚使用专门的 I2C 焊盘，详见 22.1 节）。
	I/O	2	USB_SCL —I2C 串行时钟，支持外部 USB 收发器通讯。
P0[29]/ USB_D+1/ EINT0	I/O	0	P0[29] —通用数字 I/O 管脚。当用作 GPIO 时，P0[29]与 P0[30]共用一个方向控制。

符号	类型	IOCON 选择 ¹	描述
P0[30]/ USB_D1/ EINT1	I/O	1	USB_D+1 —USB 端口 1 双向 D+线。
	I	2	EINT0 —外部中断 0 输入。
	I/O	0	P0[30] —通用数字 I/O 管脚。当用作 GPIO 时，P0[30]与 P0[29]共用一个方向控制。
	I/O	1	USB_D-1 —USB 端口 1 双向 D-线。
P0[31]/ USB_D+2	I	2	EINT1 —外部中断 1 输入。
	I/O	0	P0[31] —通用数字 I/O 管脚。
P1[0] to P1[31]	I/O	1	USB_D+2 —USB 端口 2 双向 D+线。
	I/O		Port 1 :端口 1 根据封装最多可以提供 32 个 I/O 管脚。每个管脚都有各自的方向控制、管脚模式配置和功能选择。
P1[0]/ ENET_TXD0/ T3_CAP1/ SSP2_SCK	I/O	0	P1[0] —通用数字 I/O 管脚。
	O	1	ENET_TXD0 —以太网发送数据 0 (RMII/MII 接口)。
	I	3	T3_CAP1 —定时器 3 的捕获输入，通道 1。
	I/O	4	SSP2_SCK —SSP2 的串行时钟。
P1[1]/ ENET_TXD1/ T3_MAT3/ SSP2_MOSI	I/O	0	P1[1] —通用数字 I/O 管脚。
	O	1	ENET_TXD1 —以太网发送数据 1 (RMII/MII 接口)。
	O	3	T3_MAT3 —定时器 3 的匹配输出，通道 3。
	I/O	4	SSP2_MOSI —SSP2 的主机输出从机输入。
P1[2]/ ENET_TXD2/ SD_CLK/ PWM0[1]	I/O	0	P1[2] —通用数字 I/O 管脚。
	O	1	ENET_TXD2 —以太网发送数据 2 (MII 接口)。
	O	2	SD_CLK —SD 卡接口的时钟输出线。
	O	3	PWM0[1] —脉冲宽度调制器 0 输出 1。
P1[3]/ ENET_TXD3/ SD_CMD/ PWM0[2]	I/O	0	P1[3] —通用数字 I/O 管脚。
	O	1	ENET_TXD3 —以太网发送数据 3 (MII 接口)。
	I/O	2	SD_CMD —SD 卡接口的命令行。
	O	3	PWM0[2] —脉冲宽度调制器 0，输出 2。
P1[4]/ ENET_TX_EN /	I/O	0	P1[4] —通用数字 I/O 管脚。
	O	1	ENET_TX_EN —以太网发射数据使能 (RMII/MII 接口)。
T3_MAT2/ SSP2_MISO	O	3	T3_MAT2 —定时器 3 的匹配输出，通道 2。
	I/O	4	SSP2_MISO —SSP2 的主机输入，从机输出。
P1[5]/ ENET_TX_ER/ SD_PWR/ PWM0[3]	I/O	0	P1[5] —通用数字 I/O 管脚。
	O	1	ENET_TX_ER —以太网发送数据错误提示信号 (MII 接口)。
	O	2	SD_PWR —外部 SD 卡电源的电源使能信号。
P1[6]/ ENET_TX_CLK/ SD_DAT[0]/ PWM0[4]	O	3	PWM0[3] —脉冲宽度调制器 0，输出 3。
	I/O	0	P1[6] —通用数字 I/O 管脚。
	I	1	ENET_TX_CLK —以太网发送时钟 (MII 接口)。
P1[7]/ ENET_COL/ SD_DAT[1]/ PWM0[5]	I/O	2	SD_DAT[0] —SD 卡接口的数据线 0。
	O	3	PWM0[4] —脉冲宽度调制器 0，输出 4。
	I/O	0	P1[7] —通用数字 I/O 管脚。
P1[8]/ ENET_CRS (ENET_CRS_DV)/ T3_MAT1/ SSP2_SSEL	I	1	ENET_COL —以太网冲突检测信号 (MII 接口)。
	I/O	2	SD_DAT[1] —SD 卡接口的数据线 1。
	O	3	PWM0[5] —脉冲宽度调制器 0，输出 5。
P1[9]/ ENET_RXD0/ T3_MAT0	I/O	0	P1[8] —通用数字 I/O 管脚。
	I	1	ENET_CRS (ENET_CRS_DV) —以太网载波侦听信号 (MII 接口) 或以太网载波侦听、数据有效信号 (RMII 接口)。
	O	3	T3_MAT1 —定时器 3 的匹配输出，通道 1。
P1[9]/ ENET_RXD0/ T3_MAT0	I/O	4	SSP2_SSEL —SSP2 的从选择。
	I/O	0	P1[9] —通用数字 I/O 管脚。
	I	1	ENET_RXD0 —以太网接收数据 0 (RMII/MII 接口)。
	O	3	T3_MAT0 —定时器 3 的匹配输出，通道 0。

符号	类型	IOCON 选择 ¹	描述
P1[10]/ ENET_RXD1/ T3_CAP0	I/O I I	0 1 3	P1[10] —通用数字 I/O 管脚。 ENET_RXD1 —以太网接收数据 1 (RMII/MII 接口)。 T3_CAP0 —定时器 3 的捕获输入, 通道 0。
P1[11]/ ENET_RXD2/ SD_DAT[2]/ PWM0[6]	I/O I I/O O	0 1 2 3	P1[11] —通用数字 I/O 管脚。 ENET_RXD2 —以太网接收数据 2 (MII 接口)。 SD_DAT[2] —SD 卡接口的数据线 2。 PWM0[6] —脉冲宽度调制器 0, 输出 6。
P1[12]/ ENET_RXD3/ SD_DAT[3]/ PWM0_CAP0	I/O I I/O I	0 1 2 3	P1[12] —通用数字 I/O 管脚。 ENET_RXD3 —以太网接收数据 (MII 接口)。 SD_DAT[3] —SD 卡接口的数据线 3。 PWM0_CAP0 —PWM0 的捕获输入, 通道 0。
P1[13]/ ENET_RX_DV	I/O I	0 1	P1[13] —通用数字 I/O 管脚。 ENET_RX_DV —以太网接收数据有效信号 (MII 接口)。
P1[14]/ ENET_RX_ER/ T2_CAP0	I/O I I	0 1 3	P1[14] —通用数字 I/O 管脚。 ENET_RX_ER —以太网接收数据错误提示信号 (RMII/MII 接口)。 T2_CAP0 —定时器 2 的捕获输入, 通道 0。
P1[15]/ ENET_RX_CLK (ENET_REF_CLK)/ I2C2_SDA	I/O I I/O	0 1 3	P1[15] —通用数字 I/O 管脚。 ENET_RX_CLK (ENET_REF_CLK) —以太网接收时钟 (MII 接口) 或以太网参考时钟 (RMII 接口)。 I2C2_SDA —I ² C2 数据 I/O (该管脚不使用专门的 I ² C 焊盘, 详情参见 22.1 节)。
P1[16]/ ENET_MDC/ I2S_TX_MCLK	I/O O O	0 1 2	P1[16] —通用数字 I/O 管脚。 ENET_MDC —以太网 MIIM 时钟。 I2S_TX_MCLK —I ² S 发送器主时钟输出。
P1[17]/ ENET_MDIO/ I2S_RX_MCLK	I/O I/O O	0 1 2	P1[17] —通用数字 I/O 管脚。 ENET_MDIO —以太网 MIIM 数据 I/O。 I2S_TX_MCLK —I ² S 接收器主时钟输出。
P1[18]/ USB_UP_LED1/ PWM1[1]/ T1_CAP0/ SSP1_MISO	I/O O O I O	0 1 2 3 5	P1[18] —通用数字 I/O 管脚。 USB_UP_LED1 —USB 端口 1 GoodLink LED 指示器。当设备被配置时 (非控制端点使能), 它为低电平。当设备没有被配置或在全局挂起期间, 它为高电平。 PWM1[1] —脉冲宽度调制器 1 输出, 通道 1。 T1_CAP0 —定时器 1 的捕获输入, 通道 0。 SSP1_MISO —SSP1 的主机输入, 从机输出。
P1[19]/ USB_TX_E1 / USB_PPWR1 /	I/O O O	0 1 2	P1[19] —通用数字 I/O 管脚。 USB_TX_E1 —USB 端口 1 的发送使能信号 (OTG 收发器)。 USB_PPWR1 —USB 端口 1 的端口功率使能信号。
T1_CAP1/ MC_0A/ SSP1_SCK/ U2_OE	I O O O	3 4 5 6	T1_CAP1 —定时器 1 的捕获输入, 通道 1。 MC_0A —电机控制 PWM 通道 0, 输出 A。 SSP1_SCK —SSP1 的串行时钟。 U2_OE —UART2 的 RS-485/EIA-485 输出使能信号。
P1[20]/ USB_TX_DP1/ PWM1[2]/ QE1_PHA/ MC_FB0/ SSP0_SCK/ LCD_VD[6]/ LCD_VD[10]	I/O O O I I I/O O O	0 1 2 3 4 5 6 7	P1[20] —通用数字 I/O 管脚。 USB_TX_DP1 —USB 端口 1 的 D+ 发送数据 (OTG 收发器)。 PWM1[2] —脉冲宽度调制器 1, 通道 2 输出。 QE1_PHA —正交编码器接口 PHA 输入。 MC_FB0 —电机控制 PWM 通道 0 反馈输入。 SSP0_SCK —SSP0 的串行时钟。 LCD_VD[6] —LCD 数据。 LCD_VD[10] —LCD 数据。
P1[21]/ USB_TX_DM1/	I/O O	0 1	P1[21] —通用数字 I/O 管脚。 USB_TX_DM1 —USB 端口 1 的 D- 发送数据 (OTG 收发器)。

符号	类型	IOCON 选择	描述
PWM1[3]/ SSP0_SSEL/	O	2	PWM1[3]—脉冲宽度调制器 1，通道 3 输出。
MC_ABORT /	I/O	3	SSP0_SSEL—SSP0 的从选择。
LCD_VD[7]/ LCD_VD[11]	I	4	MC_ABORT —电机控制 PWM，低电平有效快速中止。
	O	6	LCD_VD[7]—LCD 数据。
	O	7	LCD_VD[11]—LCD 数据。
P1[22]/ USB_RCV1/ USB_PWRD1/ T1_MAT0/ MC_0B/ SSP1_MOSI/ LCD_VD[8]/ LCD_VD[12]	I/O	0	P1[22]—通用数字 I/O 管脚。
	I	1	USB_RCV1—USB 端口 1 的微分接收数据（OTG 收发器）。
	I	2	USB_PWRD1—USB 端口 1 的电源状态（主机电源开关）。
	O	3	T1_MAT0—定时器 1 的匹配输出，通道 0。
	O	4	MC_0B—电机控制 PWM 通道 0，输出 B。
	I/O	5	SSP1_MOSI—SSP1 的主机输出，从机输入。
	O	6	LCD_VD[8]—LCD 数据。
	O	7	LCD_VD[12]—LCD 数据。
P1[23]/ USB_RX_DP1/ PWM1[4]/ QEI_PHB/ MC_FB1/ SSP0_MISO/ LCD_VD[9]/ LCD_VD[13]	I/O	0	P1[23]—通用数字 I/O 管脚。
	I	1	USB_RX_DP1—USB 端口 1 的 D+接收数据（OTG 收发器）。
	O	2	PWM1[4]—脉冲宽度调制器 1，通道 4 输出。
	I	3	QEI_PHB—正交编码器接口 PHB 输入。
	I	4	MC_FB1—电机控制 PWM 通道 1，反馈输出。
	I/O	5	SSP0_MISO—SSP0 的主机输入，从机输出。
	O	6	LCD_VD[9]—LCD 数据。
	O	7	LCD_VD[13]—LCD 数据。
P1[24]/ USB_RX_DM1/ PWM1[5]/ QEI_IDX/ MC_FB2/ SSP0_MOSI/ LCD_VD[10]/ LCD_VD[14]	I/O	0	P1[24]—通用数字 I/O 管脚。
	I	1	USB_RX_DM1—USB 端口 1 的 D-接收数据（OTG 收发器）。
	O	2	PWM1[5]—脉冲宽度调制器 1，通道 5 输出。
	I	3	QEI_IDX —正交编码器接口 INDEX 输入。
	I	4	MC_FB2—电机控制 PWM 通道 2 反馈输入。
	I/O	5	SSP0_MOSI—SSP0 的主机输出，从机输入。
	O	6	LCD_VD[10]/LCD_VD[14]—LCD 数据。
	O	7	LCD_VD[10]/LCD_VD[14]—LCD 数据。
P1[25]/ USB_LS1 / USB_HSTEN1 /	I/O	0	P1[25]—通用数字 I/O 管脚。
	O	1	USB_LS1 —USB 端口 1 的低速状态（OTG 收发器）。
	O	2	USB_HSTEN1 —USB 端口 1 的主机使能状态。
T1_MAT1/ MC_1A/ CLKOUT/ LCD_VD[11]/ LCD_VD[15]	O	3	T1_MAT1—定时器 1 的匹配输出，通道 1。
	O	4	MC_1A—电机控制 PWM 通道 1，输出 A。
	O	5	CLKOUT—可选时钟输出。
	O	6	LCD_VD[11]—LCD 数据。
	O	7	LCD_VD[15]—LCD 数据。
P1[26]/ USB_SSPND1/ PWM1[6]/ T0_CAP0/ MC_1B/ SSP1_SSEL/ LCD_VD[12]/ LCD_VD[20]	I/O	0	P1[26]—通用数字 I/O 管脚。
	O	1	USB_SSPND1—USB 端口 1 总线挂起状态（OTG 收发器）。
	O	2	PWM1[6]—脉冲宽度调制器 1，通道 6 输出。
	I	3	T0_CAP0—定时器 0 的捕获输入，通道 0。
	O	4	MC_1B—电机控制 PWM 通道 1，输出 B。
	I/O	5	SSP1_SSEL—SSP1 的从选择。
	O	6	LCD_VD[12]—LCD 数据。
	O	7	LCD_VD[20]—LCD 数据。
P1[27]/ USB_INT1 /	I/O	0	P1[27]—通用数字 I/O 管脚。
	I	1	USB_INT1 —USB 端口 1 OTG 收发器中断（OTG 收发器）。

符号	类型	IOCON 选择	描述
USB_OVRCCR1 /	I	2	USB_OVRCCR1 —USB 端口 1 过流状态。
T0_CAP1/	I	3	T0_CAP1 —定时器 0 的捕获输入，通道 1。
CLKOUT/	O	4	CLKOUT —可选时钟输出。
LCD_VD[13]/	O	6	LCD_VD[13] —LCD 数据。
LCD_VD[21]	O	7	LCD_VD[21] —LCD 数据。
P1[28]/ USB_SCL1/	I/O	0	P1[28] —通用数字 I/O 管脚。
PWM1_CAP0/	I/O	1	USB_SCL1 —USB 端口 1 I ² C 串行时钟（OTG 收发器）。
T0_MAT0/ MC_2A/	I	2	PWM1_CAP0 —PWM1 的捕获输入，通道 0。
SSP0_SSEL/	O	3	T0_MAT0 —定时器 0 的匹配输出，通道 0。
LCD_VD[14]/	O	4	MC_2A —电机控制 PWM 通道 2，输出 A。
LCD_VD[22]	O	5	SSP0_SSEL —SSP0 的从选择。
	O	6	LCD_VD[14] —LCD 数据。
	O	7	LCD_VD[22] —LCD 数据。
P1[29]/ USB_SDA1/	I/O	0	P1[29] —通用数字 I/O 管脚。
PWM1_CAP1/	I/O	1	USB_SDA1 —USB 端口 1 I ² C 串行数据（OTG 收发器）。
T0_MAT1/ MC_2B/	I	2	PWM1_CAP1 —PWM1 的捕获输入，通道 1。
U4_TXD/	O	3	T0_MAT1 —定时器 0 的匹配输出，通道 1。
LCD_VD[15]/	O	4	MC_2B —电机控制 PWM 通道 2，输出 B。
LCD_VD[23]	O	5	U4_TXD —UART4 的发送器输出（在智能卡模式下 I/O）。
	O	6	LCD_VD[15] —LCD 数据。
	O	7	LCD_VD[23] —LCD 数据。
P1[30]/	I/O	0	P1[30] —通用数字 I/O 管脚。
USB_PWRD2/	I	1	USB_PWRD2 —USB 端口 2 的电源状态。
USB_VBUS/	I	2	USB_VBUS —监控 USB 总线电源的存在。
AD0[4]/			Note: 该信号必须是高电平，USB 才能复位。
I2C0_SDA/ U3_OE	I	3	AD0[4] —A/D 转换器 0，输入 4。当配置为 ADC 输入时，管脚的数字功能必须禁用（详见 8.4.1 节）。
	I/O	4	I2C2_SDA —I ² C2 数据 I/O（该管脚不使用专门的 I ² C 焊盘，详情参见 22.1 节）。
	O	5	U3_OE —UART3 的 RS-485/EIA-485 输出使能信号。
P1[31]/	I/O	0	P1[31] —通用数字 I/O 管脚。
USB_OVRCCR2 /	I	1	USB_OVRCCR2 —USB 端口 2 的过流状态。
SSP1_SCK/ AD0[5]/	I/O	2	SSP1_SCK —SSP1 的串行时钟。
I2C0_SCL	I	3	AD0[5] —A/D 转换器 0，输入 5。当配置为 ADC 输入时，管脚的数字功能必须禁用（详见 8.4.1 节）。
	I/O	4	I2C0_SCL —I ² C0 时钟 I/O（该管脚不使用专门的 I ² C 焊盘，详情参见 22.1 节）。
P2[0] to P2[31]	I/O		Port 2: 端口 2 根据封装最多可以提供 32 个 I/O 管脚。每个管脚都有各自的方向控制、管脚模式配置和功能选择。
P2[0]/ PWM1[1]/	I/O	0	P2[0] —通用数字 I/O 管脚。
U1_TXD/	O	1	PWM1[1] —脉冲宽度调制器 1，通道 1 输出。
LCD_PWR	O	2	U1_TXD —UART1 的发送器输出。
	O	7	LCD_PWR —LCD 面板功率使能。
P2[1]/ PWM1[2]/	I/O	0	P2[1] —通用数字 I/O 管脚。
U1_RXD/ LCD_LE	O	1	PWM1[2] —脉冲宽度调制器 1，通道 2 输出。
	I	2	U1_RXD —UART1 的接收器输入。
	O	7	LCD_LE —行末信号。
P2[2]/ PWM1[3]/	I/O	0	P2[2] —通用数字 I/O 管脚。
U1_CTS/ T2_MAT3/	O	1	PWM1[3] —脉冲宽度调制器 1，通道 3 输出。
TRACEDATA[3]/	I	2	U1_CTS —UART1 的清除发送输入。
LCD_DCLK	O	3	T2_MAT3 —定时器 2 的匹配输出，通道 3。

符号	类型	IOCON 选择 ¹	描述
	O	5	TRACEDATA[3]—跟踪数据，位 3。
	O	7	LCD_DCLK—LCD 平板时钟。
P2[3]/ PWM1[4]/ U1_DCD/ T2_MAT2/ TRACEDATA[2]/ LCD_FP	I/O	0	P2[3]—通用数字 I/O 管脚。
	O	1	PWM1[4]—脉冲宽度调制器 1，通道 4 输出。
	I	2	U1_DCD—UART1 的数据载波检测输入。
	O	3	T2_MAT2—定时器 2 的匹配输出，通道 2。
	O	5	TRACEDATA[2]—跟踪数据，位 2。
	O	7	LCD_FP—帧脉冲（STN）。垂直同步脉冲（TFT）。
P2[4]/ PWM1[5]/ U1_DSR/ T2_MAT1/ TRACEDATA[1]/ LCD_ENAB_M	I/O	0	P2[4]—通用数字 I/O 管脚。
	O	1	PWM1[5]—脉冲宽度调制器 1，通道 5 输出。
	I	2	U1_DSR—UART1 的数据设置就绪输入。
	O	3	T2_MAT1—定时器 2 的匹配输出，通道 1。
	O	5	TRACEDATA[1]—跟踪数据，位 1。
	O	7	LCD_ENAB_M—STN 交流偏压驱动或 TFT 数据使能输出。
P2[5]/ PWM1[6]/ U1_DTR/ T2_MAT0/ TRACEDATA[0]/ LCD_LP	I/O	0	P2[5]—通用数字 I/O 管脚。
	O	1	PWM1[6]—脉冲宽度调制器 1，通道 6 输出。
	O	2	U1_DTR—UART1 的数据终端就绪输出。该信号也可以被配置为 UART1 的 RS-485/EIA-485 输出使能信号。
	O	3	T2_MAT0—定时器 2 的匹配输出，通道 0。
	O	5	TRACEDATA[0]—跟踪数据，位 0。
	O	7	LCD_LP—行同步脉冲（STN）。水平同步脉冲（TFT）。
P2[6]/ PWM1_CAP0/ U1_RI/ T2_CAP0/ U2_OE/ TRACECLK/ LCD_VD[0]/ LCD_VD[4]	I/O	0	P2[6]—通用数字 I/O 管脚。
	I	1	PWM1_CAP0—PWM1 的捕获输入，通道 0。
	I	2	U1_RI—UART1 的铃声指示输入。
	I	3	T2_CAP0—定时器 2 的捕获输入，通道 0。
	O	4	U2_OE—UART2 的 RS-485/EIA-485 输出使能信号。
	O	5	TRACECLK—跟踪时钟。
	O	6	LCD_VD[0]—LCD 数据。
	O	7	LCD_VD[4]—LCD 数据。
P2[7]/ CAN_RD2/ U1_RTS/ LCD_VD[1]/ LCD_VD[5]	I/O	0	P2[7]—通用数字 I/O 管脚。
	I	1	CAN_RD2—CAN2 接收器输入。
	O	2	U1_RTS—UART1 的请求发送输出。该信号也可以被配置为 UART1 的 RS-485/EIA-485 输出使能信号。
	O	6	LCD_VD[1]—LCD 数据。
	O	7	LCD_VD[5]—LCD 数据。
P2[8]/ CAN_TD2/ U2_TXD/ U1_CTS/ ENET_MDC/ LCD_VD[2]/ LCD_VD[6]	I/O	0	P2[8]—通用数字 I/O 管脚。
	O	1	CAN_TD2—CAN2 发送器输出。
	O	2	U2_TXD—UART2 的发送器输出。
	I	3	U1_CTS—UART1 的清除发送输入。
	O	4	ENET_MDC—以太网 MIIM 时钟。
	O	6	LCD_VD[2]—LCD 数据。
	O	7	LCD_VD[6]—LCD 数据。
P2[9]/ USB_CONNECT1/ U2_RXD/ U4_RXD/ ENET_MDIO/ LCD_VD[3]/ LCD_VD[7]	I/O	0	P2[9]—通用数字 I/O 管脚。
	O	1	USB_CONNECT1—USB1 SoftConnect 控制。在软件控制下，该信号用于转换一个 1.5kΩ 的外部电阻。它可以与 SoftConnect USB 特性配合使用。
	I	2	U2_RXD—UART2 的接收器输入。
	I	3	U4_RXD—UART4 的接收器输入。
	I/O	4	ENET_MDIO—以太网 MIIM 数据 I/O。
	I	6	LCD_VD[3]—LCD 数据。
	I	7	LCD_VD[7]—LCD 数据。

注：当 RESET 为低电平时，该管脚的低电平强制片上引导加载器控制复位后的

符号	类型	IOCON 选择 ¹⁾	描述
			器件进入 ISP 模式，详见 37.3 节。
P2[10]/ EINT0/ NMI	I/O	0	P2[10] —通用数字 I/O 管脚。该管脚具有 5ns 的输入毛刺滤波器。
	I	1	EINT0 —外部中断 0 输入。
	I	2	NMI —非屏蔽中断输入。
P2[11]/ EINT1/ SD_DAT[1]/ I2S_TX_SCK/ LCD_CLKIN	I/O	0	P2[11] —通用数字 I/O 管脚。该管脚具有 5ns 的输入毛刺滤波器。
	I	1	EINT1 —外部中断 1 输入。
	I/O	2	SD_DAT[1] —SD 卡接口的数据线 1。
	I/O	3	I2S_TX_SCK —发送时钟。它由主机驱动，从机接收。该信号对应《I ² S-总线规范》中的信号 SCK。
	O	7	LCD_CLKIN —LCD 时钟。
P2[12]/ EINT2/ SD_DAT[2]/ I2S_TX_WS/ LCD_VD[4]/ LCD_VD[3]/ LCD_VD[8]/ LCD_VD[18]	I/O	0	P2[12] —通用数字 I/O 管脚。该管脚具有 5ns 的输入毛刺滤波器。
	I	1	INT2 —外部中断 2 输入。
	I/O	2	SD_DAT[2] —SD 卡接口的数据线 2。
	I/O	3	I2S_TX_WS —发送字选。它由主机驱动，从机接收。该信号对应《I ² S-总线规范》中的信号 WS。
	O	4	LCD_VD[4] —LCD 数据。
	O	5	LCD_VD[3] —LCD 数据。
	O	6	LCD_VD[8] —LCD 数据。
	O	7	LCD_VD[18] —LCD 数据。
P2[13]/ EINT3/ SD_DAT[3]/ I2S_TX_SDA/ LCD_VD[5]/ LCD_VD[9]/ LCD_VD[19]	I/O	0	P2[13] —通用数字 I/O 管脚。该管脚具有 5ns 的输入毛刺滤波器。
	I	1	EINT3 —外部中断 3 输入。
	I/O	2	SD_DAT[3] —SD 卡接口的数据线 3。
	I/O	3	I2S_TX_SDA —发送数据。它由发送器驱动，接收器读取。该信号对应《I ² S-总线规范》中的信号 SD。
	O	5	LCD_VD[5] —LCD 数据。
	O	6	LCD_VD[9] —LCD 数据。
	O	7	LCD_VD[19] —LCD 数据。
P2[14]/ EMC_CS2/ I2C1_SDA/ T2_CAP0	I/O	0	P2[14] —通用数字 I/O 管脚。
	O	1	EMC_CS2 —低电平有效片选 2 信号。
	I/O	2	I2C1_SDA —I ² C1 数据 I/O（该管脚不使用专门的 I ² C 焊盘，详见 22.1 节）。
	I	3	T2_CAP0 —定时器 2 的捕获输入，通道 0。
P2[15]/ EMC_CS3/ I2C1_SCL/ T2_CAP1	I/O	0	P2[15] —通用数字 I/O 管脚。
	O	1	EMC_CS3 —低电平有效片选 3 信号。
	I/O	2	I2C1_SCL —I ² C1 时钟 I/O（该管脚不使用专门的 I ² C 焊盘，详见 22.1 节）。
	I	3	T2_CAP1 —定时器 2 的捕获输入，通道 1。
P2[16]/ <u>EMC_CAS</u>	I/O	0	P2[16] —通用数字 I/O 管脚。
	O	1	EMC_CAS —低电平有效 SDRAM 列地址选通脉冲。
P2[17]/ <u>EMC_RAS</u>	I/O	0	P2[17] —通用数字 I/O 管脚。
	O	1	EMC_RAS —低电平有效 SDRAM 行地址选通脉冲。
P2[18]/ EMC_CLK0	I/O	0	P2[18] —通用数字 I/O 管脚。
	O	1	EMC_CLK0 —SDRAM 时钟 0。
P2[19]/ EMC_CLK1	I/O	0	P2[19] —通用数字 I/O 管脚。
	O	1	EMC_CLK1 —SDRAM 时钟 1。
P2[20]/ <u>EMC_DYCS0</u>	I/O	0	P2[20] —通用数字 I/O 管脚。
	O	1	EMC_DYCS0 —SDRAM 片选 0。
P2[21]/ <u>EMC_DYCS1</u>	I/O	0	P2[21] —通用数字 I/O 管脚。
	O	1	EMC_DYCS1 —SDRAM 片选 1。

符号	类型	IOCON 选择 ¹	描述
P2[22]/	I/O	0	P2[22] —通用数字 I/O 管脚。
EMC_DYCS2 /	O	1	EMC_DYCS2 —SDRAM 片选 2。
SSP0_SCK/	I/O	2	SSP0_SCK —SSP0 的串行时钟。
T3_CAP0	I	3	T3_CAP0 —定时器 3 的捕获输入，通道 0。
P2[23]/	I/O	0	P2[23] —通用数字 I/O 管脚。
EMC_DYCS3 /	O	1	EMC_DYCS3 —SDRAM 片选 3。
SSP0_SSEL/	I/O	2	SSP0_SSEL —SSP0 的从选择。
T3_CAP1	I	3	T3_CAP1 —定时器 3 的捕获输入，通道 1。
P2[24]/ EMC_CKE0	I/O	0	P2[24] —通用数字 I/O 管脚。
	O	1	EMC_CKE0 —SDRAM 时钟使能 0。
P2[25]/ EMC_CKE1	I/O	0	P2[25] —通用数字 I/O 管脚。
	O	1	EMC_CKE1 —SDRAM 时钟使能 1。
P2[26]/	I/O	0	P2[26] —通用数字 I/O 管脚。
EMC_CKE2/	O	1	EMC_CKE2 —SDRAM 时钟使能 2。
SSP0_MISO/	I/O	2	SSP0_MISO —SSP0 的主机输入，从机输出。
T3_MAT0	O	3	T3_MAT0 —定时器 3 的匹配输出，通道 0。
P2[27]/	I/O	0	P2[27] —通用数字 I/O 管脚。
EMC_CKE3/	O	1	EMC_CKE3 —SDRAM 时钟使能 3。
SSP0_MOSI/	I/O	2	SSP0_MOSI —SSP0 的主机输出，从机输入。
T3_MAT1	O	3	T3_MAT1 —定时器 3 的匹配输出，通道 1。
P2[28]/	I/O	0	P2[28] —通用数字 I/O 管脚。
EMC_DQM0	O	1	EMC_DQM0 —数据屏蔽 0，配合 SDRAM 和静态设备使用。
P2[29]/	I/O	0	P2[29] —通用数字 I/O 管脚。
EMC_DQM1	O	1	EMC_DQM1 —数据屏蔽 1，配合 SDRAM 和静态设备使用。
P2[30]/	I/O	0	P2[30] —通用数字 I/O 管脚。
EMC_DQM2/	O	1	EMC_DQM2 —数据屏蔽 2，配合 SDRAM 和静态设备使用。
I2C2_SDA/	I/O	2	I2C2_SDA —I2C2 数据 I/O（该管脚不使用专门的 I2C 焊盘，详情参见 22.1 节）。
T3_MAT2	O	3	T3_MAT2 —定时器 3 的匹配输出，通道 2。
P2[31]/	I/O	0	P2[31] —通用数字 I/O 管脚。
EMC_DQM3/	O	1	EMC_DQM3 —数据屏蔽 3，配合 SDRAM 和静态设备使用。
I2C2_SCL/	I/O	2	I2C2_SCL —I2C2 时钟 I/O（该管脚不使用专门的 I2C 焊盘，详情参见 22.1 节）。
T3_MAT3	O	3	T3_MAT3 —定时器 3 的匹配输出，通道 3。
P3[0] to P3[31]	I/O		Port 3: 端口 3 根据封装最多可以提供 32 个 I/O 管脚。每个管脚都有各自的方向控制、管脚模式配置和功能选择。
P3[0]/ EMC_D[0]	I/O	0	P3[0] —通用数字 I/O 管脚。
	I/O	1	EMC_D[0] —外部存储器数据线 0。
P3[1]/ EMC_D[1]	I/O	0	P3[1] —通用数字 I/O 管脚。
	I/O	1	EMC_D[1] —外部存储器数据线 1。
P3[2]/ EMC_D[2]	I/O	0	P3[2] —通用数字 I/O 管脚。
	I/O	1	EMC_D[2] —外部存储器数据线 2。
P3[3]/ EMC_D[3]	I/O	0	P3[3] —通用数字 I/O 管脚。
	I/O	1	EMC_D[3] —外部存储器数据线 3。
P3[4]/ EMC_D[4]	I/O	0	P3[4] —通用数字 I/O 管脚。
	I/O	1	EMC_D[4] —外部存储器数据线 4。
P3[5]/ EMC_D[5]	I/O	0	P3[5] —通用数字 I/O 管脚。
	I/O	1	EMC_D[5] —外部存储器数据线 5。
P3[6]/ EMC_D[6]	I/O	0	P3[6] —通用数字 I/O 管脚。
	I/O	1	EMC_D[6] —外部存储器数据线 6。
P3[7]/ EMC_D[7]	I/O	0	P3[7] —通用数字 I/O 管脚。
	I/O	1	EMC_D[7] —外部存储器数据线 7。

符号	类型	IOCON 选择 ¹	描述
P3[8]/ EMC_D[8]	I/O	0	P3[8] —通用数字 I/O 管脚。
	I/O	1	EMC_D[8] —外部存储器数据线 8。
P3[9]/ EMC_D[9]	I/O	0	P3[9] —通用数字 I/O 管脚。
	I/O	1	EMC_D[9] —外部存储器数据线 9。
P3[10]/ EMC_D[10]	I/O	0	P3[10] —通用数字 I/O 管脚。
	I/O	1	EMC_D[10] —外部存储器数据线 10。
P3[11]/ EMC_D[11]	I/O	0	P3[11] —通用数字 I/O 管脚。
	I/O	1	EMC_D[11] —外部存储器数据线 11。
P3[12]/ EMC_D[12]	I/O	0	P3[12] —通用数字 I/O 管脚。
	I/O	1	EMC_D[12] —外部存储器数据线 12。
P3[13]/ EMC_D[13]	I/O	0	P3[13] —通用数字 I/O 管脚。
	I/O	1	EMC_D[13] —外部存储器数据线 13。
P3[14]/ EMC_D[14]	I/O	0	P3[14] —通用数字 I/O 管脚。
	I/O	1	EMC_D[14] —外部存储器数据线 14。当上电复位时, 该管脚为 BOOT0 管脚 (详见以下 P3[15] 的描述)。
P3[15]/ EMC_D[15]	I/O	0	P3[15] —通用数字 I/O 管脚。
	I/O	1	EMC_D[15] —外部存储器数据线 15。当上电复位时, 该管脚为 BOOT1 管脚 (仅针对无 Flash 器件)。BOOT[1:0] = 00 选择 EMC_CS1 上 8 位外部存储器。 BOOT[1:0] = 01 被保留。不使用。 BOOT[1:0] = 10 选择 EMC_CS1 上 32 位外部存储器。 BOOT[1:0] = 11 选择 EMC_CS1 上 16 位外部存储器。
P3[16]/ EMC_D[16]/	I/O	0	P3[16] —通用数字 I/O 管脚。
PWM0[1]/ U1_TXD	I/O	1	EMC_D[16] —外部存储器数据线 16。
	O	2	PWM0[1] —脉冲宽度调制器 0, 输出 1。
	O	3	U1_TXD —UART1 的发送器输出。
P3[17]/ EMC_D[17]/	I/O	0	P3[17] —通用数字 I/O 管脚。
PWM0[2]/ U1_RXD	I/O	1	EMC_D[17] —外部存储器数据线 17。
	O	2	PWM0[2] —脉冲宽度调制器 0, 输出 2。
	I	3	U1_RXD —UART1 的接收器输入。
P3[18]/ EMC_D[18]/	I/O	0	P3[18] —通用数字 I/O 管脚。
PWM0[3]/ U1_CTS	I/O	1	EMC_D[18] —外部存储器数据线 18。
	O	2	PWM0[3] —脉冲宽度调制器 0, 输出 3。
	I	3	U1_CTS —UART1 的清除发送输入。
P3[19]/ EMC_D[19]/	I/O	0	P3[19] —通用数字 I/O 管脚。
PWM0[4]/ U1_DCD	I/O	1	EMC_D[19] —外部存储器数据线 19。
	O	2	PWM0[4] —脉冲宽度调制器 0, 输出 4。
	I	3	U1_DCD —UART1 的数据载波检测输入。
P3[20]/ EMC_D[20]/	I/O	0	P3[20] —通用数字 I/O 管脚。
PWM0[5]/ U1_DSR	I/O	1	EMC_D[20] —外部存储器数据线 20。
	O	2	PWM0[5] —脉冲宽度调制器 0, 输出 5。
	I	3	U1_DSR —UART1 的数据设置就绪输入。
P3[21]/ EMC_D[21]/	I/O	0	P3[21] —通用数字 I/O 管脚。
PWM0[6]/ U1_DTR	I/O	1	EMC_D[21] —外部存储器数据线 21。
	O	2	PWM0[6] —脉冲宽度调制器 0, 输出 6。
	O	3	U1_DTR —UART1 的数据终端就绪输出。该信号也可以被配置为 UART1 的 RS-485/EIA-485 输出使能信号。
P3[22]/ EMC_D[22]/	I/O	0	P3[22] —通用数字 I/O 管脚。
PWM0_CAP0/	I/O	1	EMC_D[22] —外部存储器数据线 22。
U1_RI	I	2	PWM0_CAP0 —PWM0 的捕获输入, 通道 0。
	I	3	U1_RI —UART1 的振铃指示输入。
P3[23]/ EMC_D[23]/	I/O	0	P3[23] —通用数字 I/O 管脚。

符号	类型	IOCON 选择	描述
PWM1_CAP0/	I/O	1	EMC_D[23]—外部存储器数据线 23。
T0_CAP0	I	2	PWM1_CAP0—PWM1 的捕获输入, 通道 0。
	I	3	T0_CAP0—定时器 0 的捕获输入, 通道 0。
P3[24]/ EMC_D[24]/	I/O	0	P3[24]—通用数字 I/O 管脚。
PWM1[1]/ T0_CAP1	I/O	1	EMC_D[24]—外部存储器数据线 24。
	O	2	PWM1[1]—脉冲宽度调制器 1, 输出 1。
	I	3	T0_CAP1—定时器 0 的捕获输入, 通道 1。
P3[25]/ EMC_D[25]/	I/O	0	P3[25]—通用数字 I/O 管脚。
PWM1[2]/ T0_MAT0	I/O	1	EMC_D[25]—外部存储器数据线 25。
	O	2	PWM1[2]—脉冲宽度调制器 1, 输出 2。
	O	3	T0_MAT0—定时器 0 的匹配输出, 通道 0。
P3[26]/ EMC_D[26]/	I/O	0	P3[26]—通用数字 I/O 管脚。
PWM1[3]/	I/O	1	EMC_D[26]—外部存储器数据线 26。
T0_MAT1/	O	2	PWM1[3]—脉冲宽度调制器 1, 输出 3。
STCLK	O	3	T0_MAT1—定时器 0 的匹配输出, 通道 1。
	I	4	STCLK—系统节拍定时器时钟输入。
P3[27]/ EMC_D[27]/	I/O	0	P3[27]—通用数字 I/O 管脚。
PWM1[4]/ T1_CAP0	I/O	1	EMC_D[27]—外部存储器数据线 27。
	O	2	PWM1[4]—脉冲宽度调制器 1, 输出 4。
	I	3	T1_CAP0—定时器 1 的捕获输入, 通道 0。
P3[28]/ EMC_D[28]/	I/O	0	P3[28]—通用数字 I/O 管脚。
PWM1[5]/ T1_CAP1	I/O	1	EMC_D[28]—外部存储器数据线 28。
	O	2	PWM1[5]—脉冲宽度调制器 1, 输出 5。
	I	3	T1_CAP1—定时器 1 的捕获输入, 通道 1。
P3[29]/ EMC_D[29]/	I/O	0	P3[29]—通用数字 I/O 管脚。
PWM1[6]/ T1_MAT0	I/O	1	EMC_D[29]—外部存储器数据线 29。
	O	2	PWM1[6]—脉冲宽度调制器 1, 输出 6。
	O	3	T1_MAT0—定时器 1 的匹配输出, 通道 0。
P3[30]/ EMC_D[30]/	I/O	0	P3[30]—通用数字 I/O 管脚。
U1_RTS/ T1_MAT1	I/O	1	EMC_D[30]—外部存储器数据线 30。
	O	2	U1_RTS—UART1 的请求发送输出。该信号也可以被配置为 UART1 的 RS-485/EIA-485 输出使能信号。
	O	3	T1_MAT1—定时器 1 的匹配输出, 通道 1。
P3[31]/ EMC_D[31]/	I/O	0	P3[31]—通用数字 I/O 管脚。
T1_MAT2	I/O	1	EMC_D[31]—外部存储器数据线 31。
	O	3	T1_MAT2—定时器 1 的匹配输出, 通道 2。
P4[0] to P4[31]	I/O		Port 4: 端口 4 根据封装最多可以提供 32 个 I/O 管脚。每个管脚都有各自的方向控制、管脚模式配置和功能选择。
P4[0]/ EMC_A[0]	I/O	0	P4[0]—通用数字 I/O 管脚。
	I/O	1	EMC_A[0]—外部存储器地址线 0。
P4[1]/ EMC_A[1]	I/O	0	P4[1]—通用数字 I/O 管脚。
	I/O	1	EMC_A[1]—外部存储器地址线 1。
P4[2]/ EMC_A[2]	I/O	0	P4[2]—通用数字 I/O 管脚。
	I/O	1	EMC_A[2]—外部存储器地址线 2。
P4[3]/ EMC_A[3]	I/O	0	P4[3]—通用数字 I/O 管脚。
	I/O	1	EMC_A[3]—外部存储器地址线 3。
P4[4]/ EMC_A[4]	I/O	0	P4[4]—通用数字 I/O 管脚。
	I/O	1	EMC_A[4]—外部存储器地址线 4。
P4[5]/ EMC_A[5]	I/O	0	P4[5]—通用数字 I/O 管脚。
	I/O	1	EMC_A[5]—外部存储器地址线 5。
P4[6]/ EMC_A[6]	I/O	0	P4[6]—通用数字 I/O 管脚。

符号	类型	IOCON 选择 ^[1]	描述
	I/O	1	EMC_A[6]—外部存储器地址线 6。
P4[7]/ EMC_A[7]	I/O	0	P4[7]—通用数字 I/O 管脚。
	I/O	1	EMC_A[7]—外部存储器地址线 7。
P4[8]/ EMC_A[8]	I/O	0	P4[8]—通用数字 I/O 管脚。
	I/O	1	EMC_A[8]—外部存储器地址线 8。
P4[9]/ EMC_A[9]	I/O	0	P4[9]—通用数字 I/O 管脚。
	I/O	1	EMC_A[9]—外部存储器地址线 9。
P4[10]/ EMC_A[10]	I/O	0	P4[10]—通用数字 I/O 管脚。
	I/O	1	EMC_A[10]—外部存储器地址线 10。
P4[11]/ EMC_A[11]	I/O	0	P4[11]—通用数字 I/O 管脚。
	I/O	1	EMC_A[11]—外部存储器地址线 11。
P4[12]/ EMC_A[12]	I/O	0	P4[12]—通用数字 I/O 管脚。
	I/O	1	EMC_A[12]—外部存储器地址线 12。
P4[12]/ EMC_A[12]	I/O	0	P4[13]—通用数字 I/O 管脚。
	I/O	1	EMC_A[13]—外部存储器地址线 13。
P4[14]/ EMC_A[14]	I/O	0	P4[14]—通用数字 I/O 管脚。
	I/O	1	EMC_A[14]—外部存储器地址线 14。
P4[15]/ EMC_A[15]	I/O	0	P4[15]—通用数字 I/O 管脚。
	I/O	1	EMC_A[15]—外部存储器地址线 15。
P4[16]/ EMC_A[16]	I/O	0	P4[16]—通用数字 I/O 管脚。
	I/O	1	EMC_A[16]—外部存储器地址线 16。
P4[17]/ EMC_A[17]	I/O	0	P4[17]—通用数字 I/O 管脚。
	I/O	1	EMC_A[17]—外部存储器地址线 17。
P4[18]/ EMC_A[18]	I/O	0	EMC_A[18]—通用数字 I/O 管脚。
	I/O	1	EMC_A[18]—外部存储器地址线 18。
P4[19]/ EMC_A[19]	I/O	0	P4[19]—通用数字 I/O 管脚。
	I/O	1	EMC_A[19]—外部存储器地址线 19。
P4[20]/ EMC_A[20]/	I/O	0	P4[20]—通用数字 I/O 管脚。
I2C2_SDA/	I/O	1	EMC_A[20]—外部存储器地址线 20。
SSP1_SCK	I/O	2	I2C2_SDA—I2C2 数据 I/O（这个管脚不使用专门的 I2C 焊盘，详见 22.1 节）。
	I/O	3	SSP1_SCK—SSP1 的串行时钟。
P4[21]/ EMC_A[21]/	I/O	0	P4[21]—通用数字 I/O 管脚。
I2C2_SCL/	I/O	1	EMC_A[21]—外部存储器地址线 21。
SSP1_SSEL	I/O	2	I2C2_SCL—I2C2 时钟 I/O（这个管脚不使用专门的 I2C 焊盘，详见 22.1 节）。
	I/O	3	SSP1_SSEL—SSP1 的从选择。
P4[22]/ EMC_A[22]/	I/O	0	P4[22]—通用数字 I/O 管脚。
U2_TXD/	I/O	1	EMC_A[22]—外部存储器地址线 22。
SSP1_MISO	O	2	U2_TXD—UART2 的发送器输出。
	I/O	3	SSP1_MISO—SSP1 的主机输入，从机输出。
P4[23]/ EMC_A[23]/	I/O	0	P4[23]—通用数字 I/O 管脚。
U2_RXD/	I/O	1	EMC_A[23]—外部存储器地址线 23。
SSP1_MOSI	I	2	U2_RXD—UART2 的接收器输入。
	I/O	3	SSP1_MOSI—SSP1 的主机输出，从机输入。
P4[24]/ $\overline{\text{EMC_OE}}$	I/O	0	P4[24]—通用数字 I/O 管脚。
	O	1	$\overline{\text{EMC_OE}}$ —低电平有效输出使能信号。
P4[25]/ $\overline{\text{EMC_WE}}$	I/O	0	P4[25]—通用数字 I/O 管脚。
	O	1	$\overline{\text{EMC_WE}}$ —低电平有效写入使能信号。
P4[26]/	I/O	0	P4[26]—通用数字 I/O 管脚。
$\overline{\text{EMC_BLS0}}$	O	1	$\overline{\text{EMC_BLS0}}$ —低电平有效字节通道选择信号 0。

符号	类型	IOCON 选择 ¹	描述
P4[27]/ EMC_BLS1	I/O O	0 1	P4[27] —通用数字 I/O 管脚。 EMC_BLS1 —低电平有效字节通道选择信号 1。
P4[28]/ EMC_BLS2 /	I/O O	0 1	P4[28] —通用数字 I/O 管脚。 EMC_BLS2 —低电平有效字节通道选择信号 2。
U3_TXD/ T2_MAT0/ LCD_VD[6]/ LCD_VD[10]/ LCD_VD[2]	O O O O	2 3 5 6 7	TXD3 —UART3 的发送器输出。 T2_MAT0 —定时器 2 的匹配输出，通道 0。 LCD_VD[6] —LCD 数据。 LCD_VD[10] —LCD 数据。 LCD_VD[2] —LCD 数据。
P4[29]/ EMC_BLS3/ U3_RXD/ T2_MAT1/ I2C2_SCL/ LCD_VD[7]/ LCD_VD[11]/ LCD_VD[3]	I/O O I O I/O O O O	0 1 2 3 4 5 6 7	P4[29] —通用数字 I/O 管脚。 EMC_BLS3 —低电平有效字节通道选择信号 3。 U3_RXD —UART3 的接收器输入。 T2_MAT1 —定时器 2 的匹配输出，通道 1。 I2C2_SCL —I ² C2 时钟 I/O（该管脚不使用专门的 I ² C 焊盘，详情参见 22.1 节）。 LCD_VD[7] —LCD 数据。 LCD_VD[11] —LCD 数据。 LCD_VD[3] —LCD 数据。
P4[30]/ EMC_CS0	I/O O	0 1	P4[30] —通用数字 I/O 管脚。 EMC_CS0 —低电平有效片选 0 信号。
P4[31]/ EMC_CS1	I/O O	0 1	P4[31] —通用数字 I/O 管脚。 EMC_CS1 —低电平有效片选 1 信号。
P5[0] to P5[4]	I/O		Port 5: 端口 5 根据封装最多可以提供 5 个 I/O 管脚。每个管脚都有各自的方向控制、管脚模式配置和功能选择。
P5[0]/ EMC_A[24]/ T2_MAT2	I/O I/O O	0 1 3	P5[0] —通用数字 I/O 管脚。 EMC_A[24] —外部存储器地址线 24。 T2_MAT2 —定时器 2 的匹配输出，通道 2。
P5[1]/ EMC_A[25]/ T2_MAT3	I/O I/O O	0 1 3	P5[1] —通用数字 I/O 管脚。 EMC_A[25] —外部存储器地址线 25。 T2_MAT3 —定时器 2 的匹配输出，通道 3。
P5[2]/ T3_MAT2/ I2C0_SDA	I/O O I/O	0 3 5	P5[2] —通用数字 I/O 管脚。 T3_MAT2 —定时器 3 的匹配输出，通道 2。 I2C0_SDA —I ² C0 数据 I/O（该管脚使用专门的支持 I ² C FM+模式的 I ² C 焊盘）。
P5[3]/ U4_RXD/ I2C0_SCL	I/O I I/O	0 4 5	P5[3] —通用数字 I/O 管脚。 U4_RXD —UART4 的接收器输入。 I2C0_SCL0 —I ² C0 时钟 I/O（该管脚使用专门的支持 I ² C FM+模式的 I ² C 焊盘）。
P5[4]/ U0_OE/ T3_MAT3/ U4_TXD	I/O O O O	0 1 3 4	P5[4] —通用数字 I/O 管脚。 U0_OE —UART0 的 RS-485/EIA-485 输出使能信号。 T3_MAT3 —定时器 3 的匹配输出，通道 3。 U4_TXD —UART4 的发送器输出（在智能卡模式下 I/O）。
RTC_ALARM	O		RTC_ALARM —RTC 受控输出。该管脚具有低电平驱动强度，由 V _{BAT} 驱动（详情参见数据表）。当 RTC 报警产生时，该管脚被驱动为低电平。
USB_D-2	I/O		USB_D-2 —USB 端口 2 双向 D-线。
JTAG_TDO（SWO）	O		JTAG_TDO —JTAG 接口的测试数据输出。 SWO —串行线跟踪输出。
JTAG_TDI	I I		TDI —JTAG 接口的测试数据输入。该管脚具有内部上拉电阻，详见 38.1 节。 TMS —JTAG 接口的测试模式选择。该管脚具有内部上拉电阻，详见 38.1 节。
JTAG_TRST	I		SWDIO —串行线调试数据 I/O。 TRST —JTAG 接口的测试复位。该管脚具有内部上拉电阻，详见 38.1 节。

符号	类型	IOCON 选择 ^[1]	描述
	I		TCK —JTAG 接口的测试时钟。该时钟必须比 CPU 时钟（CCLK）慢 1/6，这样 JTAG 接口才能运作。 SWDCLK —串行线时钟。
$\overline{\text{RSTOUT}}$	O		复位状态输出。该管脚为低电平输出时表明该器件正因某种理由处于复位状态。反映了 $\overline{\text{RESET}}$ 输入管脚和所有内部复位源。
$\overline{\text{RESET}}$	I		外部复位输入。该管脚为低电平时将器件复位，并使 I/O 端口和外设恢复默认状态，处理器从地址 0 开始执行。该管脚具有 20ns 的输入毛刺滤波器。
XTAL1 ^[2]	I		振荡器电路和内部时钟发生器电路的输入。
XTAL2 ^[2]	O		振荡器放大器的输出。
RTCX1 ^[2]	I		RTC 32kHz 超低功率振荡器电路的输入。
RTCX2 ^[2]	O		RTC 32kHz 超低功率振荡器电路的输出。
V _{SS} ^[2]	I		地： 数字 IO 管脚的 OV 参考点。
V _{SSREG} ^[2]	I		地： 内部逻辑的 OV 参考点。
V _{SSA} ^[2]	I		模拟地： A/D 转换器的 0V 电源和参考点。它应与 V _{SS} 的电压相同，但应当互相隔离以减少噪音和故障。
V _{DD} (3V3) ^[2]	I		3.3V 电源电压： 这仅是用于 I/O 的电源电压而不是用于在 VBAT 领域的管脚。
V _{DD} (REG) (3V3) ^[2]	I		3.3V 稳压器电源电压： 这仅是用于提供内部逻辑的片上电压稳压器的电源。
V _{DD} ^[2]	I		模拟 3.3V 焊盘电源电压： 这可以和 V _{DD} (3V3) 一样的电源连接，但应当互相隔离以减少噪音和故障。该电压用来给 A/D 转换器供电。 注：如果不使用 A/D 转换器，该管脚应和 3.3V 电压相连接。
VREFP ^[2]	I		ADC 积极参考电压： 它应与 V _{DDA} 的电压相同，但应当互相隔离以减少噪音和故障。该管脚上的电平用作 A/D 转换器的参考电平。 注：如果不使用 A/D 转换器，该管脚应和 3.3V 电压相连接。
VBAT ^[2]	I		RTC 电源： 该管脚上的 3.3V 电压给 RTC 供电。

[1] 这些值用于 IOCON 寄存器的 FUNC 字段，详见 8.4.1 节。

[2] 这些管脚提供特别的模拟功能。

7.2 引导控制

无 Flash 的 LPC178x/177x 器件的 P3[15]/D15 和 P3[14]/D14 管脚用于在引导过程中配置外部存储器总线。这些管脚在上电复位（POR）期间采样。有关引导控制管脚的相关设置，见[表 68](#)。

表68. P3[15]/D15 和 P3[14]/D14 管脚的引导控制

P3[15]/D15 BOOT1	P3[14]/D14 BOOT0	说明
0	0	来自 $\overline{\text{EMC_CS1}}$ 上 8 位外部存储器的引导。对 POR 信号进行采样。
0	1	保留。不使用。
1	0	来自 $\overline{\text{EMC_CS1}}$ 上 32 位外部存储器的引导。对 POR 信号进行采样。
1	1	来自 $\overline{\text{EMC_CS1}}$ 上 16 位外部存储器的引导。对 POR 信号进行采样。

在引导过程中，外部存储器组 0 和 1 均配置为相同的数据总线宽度，并由 P3[15]/D15 和 P3[14]/D14 两个引导管脚的设置所决定。未使用的地址管脚（当从 16 位宽度外部存储器引导时是 A0，当从 32 位宽度存储器引导时是 A1 和 A0）配置为 GPIO。

在地址 0x8000 0000 时，引导加载器会分支到外部的用户代码。这个代码必须设置一个中断向量表（见 [2.4](#) 节）。外部引导存储器必须接到静态片选 1 上（ $\overline{\text{EMC_CS1}}$ ）。注意， $\overline{\text{EMC_CS1}}$ 被镜像到 $\overline{\text{EMC_CS0}}$ ，因为在 POR 期间，EMCControl 寄存器中的地址镜像位置位，见[表 114](#)。因此，外部引导存储器中的用户代码必须从地址 0x8000 0000 处（EMC 组 0 地址）链接去执行。

注：只有无 Flash 器件才支持外部引导选项

8.1 本章阅读方法

LPC178x/177x 为每个 GPIO 管脚的配置提供了一个独立的寄存器。这个配置包括：哪个内部功能连接到该管脚、输出模式（普通、上拉、下拉，或转发器）、开路漏极模式控制、迟滞使能、转换速率控制，以及模拟功能的缓冲设置。有些管脚还有其他特殊控制，如针对 I²C 缓冲模式。这些寄存器汇总见[表 69](#)。

表69. I/O 管脚配置寄存器汇总

端口	寄存器	明细表
端口 0 管脚	IOCON_P0_nn, nn 为端口管脚编码，从 0 到 31 [1]	表 70
端口 1 管脚	IOCON_P1_nn, nn 为端口管脚编码，从 0 到 31 [1]	表 71
端口 2 管脚	IOCON_P2_nn, nn 为端口管脚编码，从 0 到 31 [1]	表 72
端口 3 管脚	IOCON_P3_nn, nn 为端口管脚编码，从 0 到 31 [1]	表 73
端口 4 管脚	IOCON_P4_nn, nn 为端口管脚编码，从 0 到 31 [1]	表 74
端口 5 管脚	IOCON_P5_nn, nn 为端口管脚编码，从 0 到 4 [1]	表 75

[1] 哪种管脚适用取决于器件编号和封装组合。

8.2 描述

管脚连接模块可以让微控制器的多数管脚拥有一个以上的功能。配置寄存器控制着多路复用器，用于在管脚与片上设备之间建立连接。

外设在被激活以及任何相关中断被使能之前，都应先连接到相应的管脚上。如已使能的外设功能尚未映射到一个相关管脚上，则该功能的任何活动均应看作尚未定义。

一个端口管脚选择了某个功能后，相同管脚就不能使用其它外设功能。但是，GPIO 输入保持连接，且可以由软件读取，或用于实现 GPIO 中断特性。

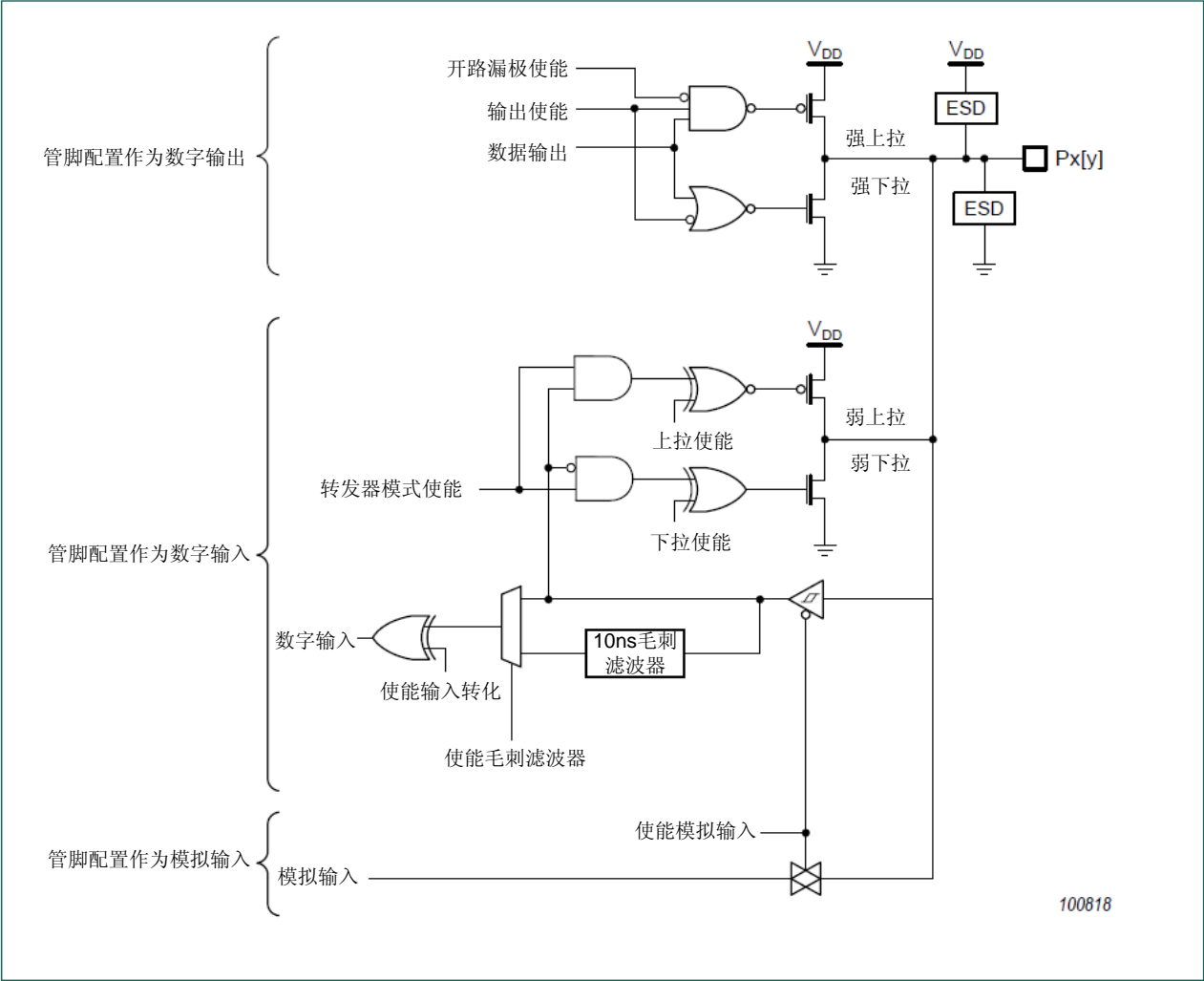
8.3 IOCON 寄存器

IOCON 寄存器控制着器件管脚的功能。每个 GPIO 管脚都有一个专门的控制寄存器，用于选择其功能与特性。每个管脚都有一组唯一的功能。并非所有管脚都能选择所有特性。例如，有 I²C 功能的管脚可以配置为不同的 I²C 总线模式，而有模拟替代功能的管脚则可以选择模拟模式。关于 IOCON 寄存器，详见[8.4.1](#)节。以下章节将描述各管脚的具体特性。

多连接

由于某个外设功能允许出现在多个管脚上，可以配置多个管脚执行同一功能。如果一个外设输出功能被配置到多个管脚上，它实际上会到达这些管脚。如果某个外设输入功能被定义为有一个以上的源，则这些值将做逻辑组合，有可能导致外设工作异常。因此应注意避免这种情况。

图12. I/O 配置



8.3.1 管脚功能

IOCON 寄存器中的 FUNC 位可以设置为 GPIO（典型值为 000），或某个特殊功能。对于设为 GPIO 的管脚，FIONDIR 寄存器决定了该管脚是配置为输入还是输出（见 9.5.1 节）。对于任何的特殊功能，管脚方向均根据其功能而自动控制。FIONDIR 寄存器对特殊功能没有作用。

8.3.2 管脚模式

IOCON 寄存器中的 **MODE** 位可为每个管脚选择片上上拉电阻或下拉电阻，或选择转发器模式。

可能的片上电阻配置为：上拉使能、下拉使能，或没有上拉/下拉。默认值是上拉使能。

转发器模式在管脚为高电平时使能上拉电阻，在管脚为低电平时使能下拉电阻。这样，如果管脚被配置为一个输入，而不是外部驱动，则管脚维持其已知的最后状态。这种状态保持不适用于深度掉电模式。转发器模式通常用于防止某个管脚在暂时无驱动情况下的悬浮状态（如果管脚悬浮到一个不确定状态，则可能会产生大量功耗）。

8.3.3 迟滞

数字功能的输入缓冲可以配置为有迟滞和无迟滞。具体数值参见相应器件的数据手册。

8.3.4 输入反相

如果外部源的极性与输入端相反，这个选项能使用户不必增加外接反相器。不要将此选项设置在 **GPIO** 输出上。否则，相同端口上其它管脚工作而选择了输入反相时，会造成输出的意外翻转。例如，如果软件读一个 **GPIO** 端口寄存器，修改寄存器值中的其它位/输出，并将其值写回端口寄存器时，则已选输入反相那个端口的任何输出都会改变状态。

8.3.5 模拟/数字模式

在模拟模式下，模拟输入连接被使能。在数字模式下，模拟输入连接被禁用。这为模拟输入端提供了保护，防止电压超出模拟电源以及基准电压范围，这些电压有时会出现于数字管脚上，因为数字管脚一般能承受 5V。

如果选择了模拟模式，则 **MODE** 域应为“无效”（Inactive）（00）；**HYS**、**INV**、**FILTR**、**SLEW** 和 **OD** 设置均无效。对于一个未连接的模拟功能管脚，要将 **ADMODE** 位的设置保持为 1（数字模式），并在 **MODE** 域中选择上拉或下拉模式。

8.3.6 输入滤波器

A 型和 W 型管脚带有一个可选使能的滤波器。该滤波器可抑制小于 10ns 的输入脉冲。

8.3.7 输出转换速率

对于不需要极快切换状态的数字输出，**SLEW** 位应设为“标准”（standard）。这个设置能够让多个输出同时切换，而不会明显地降低器件电源/地的分配水平，并且对信号转换时间只有少许影响。当应用对模拟精度有较高要求时，这个指标尤其重要。更多细节参见相关具体器件的数据手册。

8.3.8 I²C 模式

带特殊焊盘电路并支持 I²C 的管脚（P0[27]、P0[28]、P5[2]和 P5[3]）有额外的配置位。它们不是硬接线的，因此更容易用于非 I²C 功能。

HS 位适用于标准、FM 模式，以及 FM+模式的 I²C，并可用于上述所有管脚。

HIDRIVE 位仅适用于管脚 P5[2]与 P5[3]，并用于选择标准模式和 FM 模式 I²C 或 FM+模式 I²C。

- 对于任何 I²C 模式，清除 HS 位都会使能输入毛刺滤波器。如果管脚存在 HIDRIVE 位，则清除它会为标准模式或 FM 模式 I²C 选择正确的驱动强度。
- 对于 FM+模式的 I²C 运行，设置 HIDRIVE 位可为 FM+模式 I²C 选择正确的驱动强度。
- 对于非 I²C 运行，这些管脚保持为开路漏极，只能驱动为低电平，而与 HS 和 HIDRIVE 如何设置无关。如果作为输出，则它们通常要用一支外接上拉电阻，除非只用于拉入电流。为保持与其它 GPIO 管脚的最大兼容性，令 HS = 1 和 HIDRIVE = 0（如适用）。

8.3.9 开漏极模式

在选择输出时，可以在 FUNC 域中选择一个特定功能，或为一个其 FIONDIR 寄存器有一个 1 的管脚选择 GPIO 功能，OD 位的 1 选择了开路漏极工作，即 1 禁用了高电平驱动的晶体管。这个选项对主要 I²C 管脚没有作用。注意，在这种模拟式的开路漏极模式下，管脚的属性与真正的开路漏极输出有所不同。

8.3.10 DAC 使能

可用于某种功能的 DAC 输出管脚包含一个对该功能的使能，如果要用到 DAC 的输出，必须设置这个使能。

8.4 寄存器描述

管脚连接模块中包含了一个 I/O 控制寄存器（IOCON），针对每个有可编程属性的管脚，可选择该管脚相应的外设功能。这些寄存器在表 8–表 70~表 8–表 75 中按 GPIO 端口号显示。

表70. 端口 0 的 I/O 控制寄存器

端口管脚	寄存器	访问	复位值 ^[1]	地址	IOCON 类型 ^[2]	封装
P0[0]	IOCON_P0_00	R/W	0x030	0x4002 C000	D (表 76、表 77)	全部
P0[1]	IOCON_P0_01	R/W	0x030	0x4002 C004	D (表 76、表 77)	全部
P0[2]	IOCON_P0_02	R/W	0x030	0x4002 C008	D (表 76、表 77)	全部
P0[3]	IOCON_P0_03	R/W	0x030	0x4002 C00C	D (表 76、表 77)	全部
P0[4]	IOCON_P0_04	R/W	0x030	0x4002 C010	D (表 76、表 77)	全部
P0[5]	IOCON_P0_05	R/W	0x030	0x4002 C014	D (表 76、表 77)	全部
P0[6]	IOCON_P0_06	R/W	0x030	0x4002 C018	D (表 76、表 77)	全部
P0[7]	IOCON_P0_07	R/W	0x0A0	0x4002 C01C	W (表 84、表 85)	全部
P0[8]	IOCON_P0_08	R/W	0x0A0	0x4002 C020	W (表 84、表 85)	全部
P0[9]	IOCON_P0_09	R/W	0x0A0	0x4002 C024	W (表 84、表 85)	全部
P0[10]	IOCON_P0_10	R/W	0x030	0x4002 C028	D (表 76、表 77)	全部
P0[11]	IOCON_P0_11	R/W	0x030	0x4002 C02C	D (表 76、表 77)	全部
P0[12]	IOCON_P0_12	R/W	0x1B0	0x4002 C030	A (表 78、表 79)	144、180、208
P0[13]	IOCON_P0_13	R/W	0x1B0	0x4002 C034	A (表 78、表 79)	144、180、208
P0[14]	IOCON_P0_14	R/W	0x030	0x4002 C038	D (表 76、表 77)	144、180、208
P0[15]	IOCON_P0_15	R/W	0x030	0x4002 C03C	D (表 76、表 77)	全部
P0[16]	IOCON_P0_16	R/W	0x030	0x4002 C040	D (表 76、表 77)	全部
P0[17]	IOCON_P0_17	R/W	0x030	0x4002 C044	D (表 76、表 77)	全部
P0[18]	IOCON_P0_18	R/W	0x030	0x4002 C048	D (表 76、表 77)	全部
P0[19]	IOCON_P0_19	R/W	0x030	0x4002 C04C	D (表 76、表 77)	全部
P0[20]	IOCON_P0_20	R/W	0x030	0x4002 C050	D (表 76、表 77)	全部
P0[21]	IOCON_P0_21	R/W	0x030	0x4002 C054	D (表 76、表 77)	全部
P0[22]	IOCON_P0_22	R/W	0x030	0x4002 C058	D (表 76、表 77)	全部
P0[23]	IOCON_P0_23	R/W	0x1B0	0x4002 C05C	A (表 78、表 79)	全部
P0[24]	IOCON_P0_24	R/W	0x1B0	0x4002 C060	A (表 78、表 79)	全部
P0[25]	IOCON_P0_25	R/W	0x1B0	0x4002 C064	A (表 78、表 79)	全部
P0[26]	IOCON_P0_26	R/W	0x1B0	0x4002 C068	A (表 78、表 79)	全部
P0[27]	IOCON_P0_27	R/W	0	0x4002 C06C	I (表 82、表 83)	全部
P0[28]	IOCON_P0_28	R/W	0	0x4002 C070	I (表 82、表 83)	全部
P0[29]	IOCON_P0_29	R/W	0	0x4002 C074	U (表 80、表 81)	全部
P0[30]	IOCON_P0_30	R/W	0	0x4002 C078	U (表 80、表 81)	全部
P0[31]	IOCON_P0_31	R/W	0	0x4002 C07C	U (表 80、表 81)	144、180、208

[1] 复位值仅反映已使用位中保存的数据。它不包括保留位的内容。

[2] IOCON 类型是 D（标准数字管脚），其他管脚具有某个专门功能：A（模拟）、U（USB）、I（I²C）、W。

表71. 端口 1 的 I/O 控制寄存器

端口管脚	寄存器	访问	复位值 ^[1]	地址	IOCON 类型	封装
P0[0]	IOCON_P1_00	R/W	0x030	0x4002 C080	D (表 76、表 77)	全部
P0[1]	IOCON_P1_01	R/W	0x030	0x4002 C084	D (表 76、表 77)	全部
P0[2]	IOCON_P1_02	R/W	0x030	0x4002 C088	D (表 76、表 77)	180、208
P0[3]	IOCON_P1_03	R/W	0x030	0x4002 C08C	D (表 76、表 77)	180、208
P0[4]	IOCON_P1_04	R/W	0x030	0x4002 C090	D (表 76、表 77)	全部
P0[5]	IOCON_P1_05	R/W	0x030	0x4002 C094	D (表 76、表 77)	180、208
P0[6]	IOCON_P1_06	R/W	0x030	0x4002 C098	D (表 76、表 77)	180、208
P0[7]	IOCON_P1_07	R/W	0x0A0	0x4002 C09C	D (表 76、表 77)	180、208
P0[8]	IOCON_P1_08	R/W	0x0A0	0x4002 C0A0	D (表 76、表 77)	全部
P0[9]	IOCON_P1_09	R/W	0x0A0	0x4002 C0A4	D (表 76、表 77)	全部
P0[10]	IOCON_P1_10	R/W	0x030	0x4002 C0A8	D (表 76、表 77)	全部
P0[11]	IOCON_P1_11	R/W	0x030	0x4002 C0AC	D (表 76、表 77)	180、208
P0[12]	IOCON_P1_12	R/W	0x1B0	0x4002 C0B0	D (表 76、表 77)	180、208
P0[13]	IOCON_P1_13	R/W	0x1B0	0x4002 C0B4	D (表 76、表 77)	180、208
P0[14]	IOCON_P1_14	R/W	0x030	0x4002 C0B8	D (表 76、表 77)	全部
P0[15]	IOCON_P1_15	R/W	0x030	0x4002 C0BC	D (表 76、表 77)	全部
P0[16]	IOCON_P1_16	R/W	0x030	0x4002 C0C0	D (表 76、表 77)	全部
P0[17]	IOCON_P1_17	R/W	0x030	0x4002 C0C4	D (表 76、表 77)	全部
P0[18]	IOCON_P1_18	R/W	0x030	0x4002 C0C8	D (表 76、表 77)	全部
P0[19]	IOCON_P1_19	R/W	0x030	0x4002 C0CC	D (表 76、表 77)	全部
P0[20]	IOCON_P1_20	R/W	0x030	0x4002 C0D0	D (表 76、表 77)	全部
P0[21]	IOCON_P1_21	R/W	0x030	0x4002 C0D4	D (表 76、表 77)	全部
P0[22]	IOCON_P1_22	R/W	0x030	0x4002 C0D8	D (表 76、表 77)	全部
P0[23]	IOCON_P1_23	R/W	0x030	0x4002 C0DC	D (表 76、表 77)	全部
P0[24]	IOCON_P1_24	R/W	0x030	0x4002 C0E0	D (表 76、表 77)	全部
P0[25]	IOCON_P1_25	R/W	0x030	0x4002 C0E4	D (表 76、表 77)	全部
P0[26]	IOCON_P1_26	R/W	0x030	0x4002 C0E8	D (表 76、表 77)	全部
P0[27]	IOCON_P1_27	R/W	0x030	0x4002 C0EC	D (表 76、表 77)	全部
P0[28]	IOCON_P1_28	R/W	0x030	0x4002 C0F0	D (表 76、表 77)	全部
P0[29]	IOCON_P1_29	R/W	0x030	0x4002 C0F4	D (表 76、表 77)	全部
P0[30]	IOCON_P1_30	R/W	0x1B0	0x4002 C0F8	A (表 78、表 79)	全部
P0[31]	IOCON_P1_31	R/W	0x1B0	0x4002 C0FC	A (表 78、表 79)	全部

[1] 复位值仅反映已使用位中保存的数据。它不包括保留位的内容。

表72. 端口 2 的 I/O 控制寄存器

端口管脚	寄存器	访问	复位值 ^[1]	地址	IOCON 类型	封装
P2[0]	IOCON_P2_00	R/W	0x030	0x4002 C100	D (表 76、表 77)	全部
P2[1]	IOCON_P2_01	R/W	0x030	0x4002 C104	D (表 76、表 77)	全部
P2[2]	IOCON_P2_02	R/W	0x030	0x4002 C108	D (表 76、表 77)	全部
P2[3]	IOCON_P2_03	R/W	0x030	0x4002 C10C	D (表 76、表 77)	全部
P2[4]	IOCON_P2_04	R/W	0x030	0x4002 C110	D (表 76、表 77)	全部
P2[5]	IOCON_P2_05	R/W	0x030	0x4002 C114	D (表 76、表 77)	全部
P2[6]	IOCON_P2_06	R/W	0x030	0x4002 C118	D (表 76、表 77)	全部
P2[7]	IOCON_P2_07	R/W	0x030	0x4002 C11C	D (表 76、表 77)	全部
P2[8]	IOCON_P2_08	R/W	0x030	0x4002 C120	D (表 76、表 77)	全部
P2[9]	IOCON_P2_09	R/W	0x030	0x4002 C124	D (表 76、表 77)	全部
P2[10]	IOCON_P2_10	R/W	0x030	0x4002 C128	D (表 76、表 77)	全部
P2[11]	IOCON_P2_11	R/W	0x030	0x4002 C12C	D (表 76、表 77)	全部
P2[12]	IOCON_P2_12	R/W	0x030	0x4002 C130	D (表 76、表 77)	全部
P2[13]	IOCON_P2_13	R/W	0x030	0x4002 C134	D (表 76、表 77)	全部
P2[14]	IOCON_P2_14	R/W	0x030	0x4002 C138	D (表 76、表 77)	208
P2[15]	IOCON_P2_15	R/W	0x030	0x4002 C13C	D (表 76、表 77)	208
P2[16]	IOCON_P2_16	R/W	0x030	0x4002 C140	D (表 76、表 77)	180、208
P2[17]	IOCON_P2_17	R/W	0x030	0x4002 C144	D (表 76、表 77)	180、208
P2[18]	IOCON_P2_18	R/W	0x030	0x4002 C148	D (表 76、表 77)	180、208
P2[19]	IOCON_P2_19	R/W	0x030	0x4002 C14C	D (表 76、表 77)	180、208
P2[20]	IOCON_P2_20	R/W	0x030	0x4002 C150	D (表 76、表 77)	180、208
P2[21]	IOCON_P2_21	R/W	0x030	0x4002 C154	D (表 76、表 77)	180、208
P2[22]	IOCON_P2_22	R/W	0x030	0x4002 C158	D (表 76、表 77)	208
P2[23]	IOCON_P2_23	R/W	0x030	0x4002 C15C	D (表 76、表 77)	208
P2[24]	IOCON_P2_24	R/W	0x030	0x4002 C160	D (表 76、表 77)	180、208
P2[25]	IOCON_P2_25	R/W	0x030	0x4002 C164	D (表 76、表 77)	180、208
P2[26]	IOCON_P2_26	R/W	0x030	0x4002 C168	D (表 76、表 77)	208
P2[27]	IOCON_P2_27	R/W	0x030	0x4002 C16C	D (表 76、表 77)	208
P2[28]	IOCON_P2_28	R/W	0x030	0x4002 C170	D (表 76、表 77)	180、208
P2[29]	IOCON_P2_29	R/W	0x030	0x4002 C174	D (表 76、表 77)	180、208
P2[30]	IOCON_P2_30	R/W	0x030	0x4002 C178	D (表 76、表 77)	208
P2[31]	IOCON_P2_31	R/W	0x030	0x4002 C17C	D (表 76、表 77)	208

[1] 复位值仅反映已使用位中保存的数据。它不包括保留位的内容。

表73. 端口 3 的 I/O 控制寄存器

端口管脚	寄存器	访问	复位值 ^[1]	地址	IOCON 类型	封装
P3[0]	IOCON_P3_00	R/W	0x030	0x4002 C180	D (表 76、表 77)	144、180、208
P3[1]	IOCON_P3_01	R/W	0x030	0x4002 C184	D (表 76、表 77)	144、180、208
P3[2]	IOCON_P3_02	R/W	0x030	0x4002 C188	D (表 76、表 77)	144、180、208
P3[3]	IOCON_P3_03	R/W	0x030	0x4002 C18C	D (表 76、表 77)	144、180、208
P3[4]	IOCON_P3_04	R/W	0x030	0x4002 C190	D (表 76、表 77)	144、180、208
P3[5]	IOCON_P3_05	R/W	0x030	0x4002 C194	D (表 76、表 77)	144、180、208
P3[6]	IOCON_P3_06	R/W	0x030	0x4002 C198	D (表 76、表 77)	144、180、208
P3[7]	IOCON_P3_07	R/W	0x0A0	0x4002 C19C	D (表 76、表 77)	144、180、208
P3[8]	IOCON_P3_08	R/W	0x0A0	0x4002 C1A0	D (表 76、表 77)	180、208
P3[9]	IOCON_P3_09	R/W	0x0A0	0x4002 C1A4	D (表 76、表 77)	180、208
P3[10]	IOCON_P3_10	R/W	0x030	0x4002 C1A8	D (表 76、表 77)	180、208
P3[11]	IOCON_P3_11	R/W	0x030	0x4002 C1AC	D (表 76、表 77)	180、208
P3[12]	IOCON_P3_12	R/W	0x1B0	0x4002 C1B0	D (表 76、表 77)	180、208
P3[13]	IOCON_P3_13	R/W	0x1B0	0x4002 C1B4	D (表 76、表 77)	180、208
P3[14]	IOCON_P3_14	R/W	0x030	0x4002 C1B8	D (表 76、表 77)	180、208
P3[15]	IOCON_P3_15	R/W	0x030	0x4002 C1BC	D (表 76、表 77)	180、208
P3[16]	IOCON_P3_16	R/W	0x030	0x4002 C1C0	D (表 76、表 77)	208
P3[17]	IOCON_P3_17	R/W	0x030	0x4002 C1C4	D (表 76、表 77)	208
P3[18]	IOCON_P3_18	R/W	0x030	0x4002 C1C8	D (表 76、表 77)	208
P3[19]	IOCON_P3_19	R/W	0x030	0x4002 C1CC	D (表 76、表 77)	208
P3[20]	IOCON_P3_20	R/W	0x030	0x4002 C1D0	D (表 76、表 77)	208
P3[21]	IOCON_P3_21	R/W	0x030	0x4002 C1D4	D (表 76、表 77)	208
P3[22]	IOCON_P3_22	R/W	0x030	0x4002 C1D8	D (表 76、表 77)	208
P3[23]	IOCON_P3_23	R/W	0x030	0x4002 C1DC	D (表 76、表 77)	144、180、208
P3[24]	IOCON_P3_24	R/W	0x030	0x4002 C1E0	D (表 76、表 77)	144、180、208
P3[25]	IOCON_P3_25	R/W	0x030	0x4002 C1E4	D (表 76、表 77)	全部
P3[26]	IOCON_P3_26	R/W	0x030	0x4002 C1E8	D (表 76、表 77)	全部
P3[27]	IOCON_P3_27	R/W	0x030	0x4002 C1EC	D (表 76、表 77)	208
P3[28]	IOCON_P3_28	R/W	0x030	0x4002 C1F0	D (表 76、表 77)	208
P3[29]	IOCON_P3_29	R/W	0x030	0x4002 C1F4	D (表 76、表 77)	208
P3[30]	IOCON_P3_30	R/W	0x1B0	0x4002 C1F8	D (表 76、表 77)	208
P3[31]	IOCON_P3_31	R/W	0x1B0	0x4002 C1FC	D (表 76、表 77)	208

[1] 复位值仅反映已使用位中保存的数据。它不包括保留位的内容。

表74. 端口 4 的 I/O 控制寄存器

端口管脚	寄存器	访问	复位值 ^[1]	地址	IOCON 类型	封装
P4[0]	IOCON_P4_00	R/W	0x030	0x4002 C200	D (表 76、表 77)	144、180、208
P4[1]	IOCON_P4_01	R/W	0x030	0x4002 C204	D (表 76、表 77)	144、180、208
P4[2]	IOCON_P4_02	R/W	0x030	0x4002 C208	D (表 76、表 77)	144、180、208
P4[3]	IOCON_P4_03	R/W	0x030	0x4002 C20C	D (表 76、表 77)	144、180、208
P4[4]	IOCON_P4_04	R/W	0x030	0x4002 C210	D (表 76、表 77)	144、180、208
P4[5]	IOCON_P4_05	R/W	0x030	0x4002 C214	D (表 76、表 77)	144、180、208
P4[6]	IOCON_P4_06	R/W	0x030	0x4002 C218	D (表 76、表 77)	144、180、208
P4[7]	IOCON_P4_07	R/W	0x030	0x4002 C21C	D (表 76、表 77)	144、180、208
P4[8]	IOCON_P4_08	R/W	0x030	0x4002 C220	D (表 76、表 77)	144、180、208
P4[9]	IOCON_P4_09	R/W	0x030	0x4002 C224	D (表 76、表 77)	144、180、208
P4[10]	IOCON_P4_10	R/W	0x030	0x4002 C228	D (表 76、表 77)	144、180、208
P4[11]	IOCON_P4_11	R/W	0x030	0x4002 C22C	D (表 76、表 77)	144、180、208
P4[12]	IOCON_P4_12	R/W	0x030	0x4002 C230	D (表 76、表 77)	144、180、208
P4[13]	IOCON_P4_13	R/W	0x030	0x4002 C234	D (表 76、表 77)	144、180、208
P4[14]	IOCON_P4_14	R/W	0x030	0x4002 C238	D (表 76、表 77)	144、180、208
P4[15]	IOCON_P4_15	R/W	0x030	0x4002 C23C	D (表 76、表 77)	144、180、208
P4[16]	IOCON_P4_16	R/W	0x030	0x4002 C240	D (表 76、表 77)	180、208
P4[17]	IOCON_P4_17	R/W	0x030	0x4002 C244	D (表 76、表 77)	180、208
P4[18]	IOCON_P4_18	R/W	0x030	0x4002 C248	D (表 76、表 77)	180、208
P4[19]	IOCON_P4_19	R/W	0x030	0x4002 C24C	D (表 76、表 77)	180、208
P4[20]	IOCON_P4_20	R/W	0x030	0x4002 C250	D (表 76、表 77)	208
P4[21]	IOCON_P4_21	R/W	0x030	0x4002 C254	D (表 76、表 77)	208
P4[22]	IOCON_P4_22	R/W	0x030	0x4002 C258	D (表 76、表 77)	208
P4[23]	IOCON_P4_23	R/W	0x030	0x4002 C25C	D (表 76、表 77)	208
P4[24]	IOCON_P4_24	R/W	0x030	0x4002 C260	D (表 76、表 77)	144、180、208
P4[25]	IOCON_P4_25	R/W	0x030	0x4002 C264	D (表 76、表 77)	144、180、208
P4[26]	IOCON_P4_26	R/W	0x030	0x4002 C268	D (表 76、表 77)	180、208
P4[27]	IOCON_P4_27	R/W	0x030	0x4002 C26C	D (表 76、表 77)	180、208
P4[28]	IOCON_P4_28	R/W	0x030	0x4002 C270	D (表 76、表 77)	全部
P4[29]	IOCON_P4_29	R/W	0x030	0x4002 C274	D (表 76、表 77)	全部
P4[30]	IOCON_P4_30	R/W	0x030	0x4002 C278	D (表 76、表 77)	144、180、208
P4[31]	IOCON_P4_31	R/W	0x030	0x4002 C27C	D (表 76、表 77)	144、180、208

[1] 复位值仅反映已使用位中保存的数据。它不包括保留位的内容。

表75. 端口 5 的 I/O 控制寄存器

端口管脚	寄存器	访问	复位值 ^[1]	地址	IOCON 类型	封装
P5[0]	IOCON_P5_00	R/W	0x030	0x4002 C280	D (表 76、表 77)	全部, TFBGA100 除外
P5[1]	IOCON_P5_01	R/W	0x030	0x4002 C284	D (表 76、表 77)	144、180、208
P5[2]	IOCON_P5_02	R/W	0	0x4002 C288	I (表 82、表 83)	144、180、208
P5[3]	IOCON_P5_03	R/W	0	0x4002 C28C	I (表 82、表 83)	144、180、208
P5[4]	IOCON_P5_04	R/W	0x030	0x4002 C290	D (表 76、表 77)	全部

[1] 复位值仅反映已使用位中保存的数据。它不包括保留位的内容。

8.4.1 I/O 配置寄存器内容（IOCON）

以下章节描述了 IOCON 寄存器中各个位对每个 GPIO 端口管脚的功能。与大多数其它端口管脚相比，特殊端口管脚的 IOCON 有些不同。这些差异包括支持模拟功能（如 ADC 输入与 DAC 输出）、USB D+/D-管脚，以及特殊的 I²C 管脚：

- [“D 型 IOCON 寄存器（适用于大多数 GPIO 端口管脚）”](#)
- [“A 型 IOCON 寄存器（适用于包含模拟功能的管脚）”](#)
- [“U 型 IOCON 寄存器（适用于包含 USB D+或 D-功能的管脚）”](#)
- [“I 型 IOCON 寄存器（适用于包含专用 I²C 功能的管脚）”](#)
- [“W 型 IOCON 寄存器（这些管脚其它方面与 D 型相同，但包含一个可选输入毛刺滤波器，默认其上拉/下拉被禁用）”](#)

8.4.1.1 D 型 IOCON 寄存器（适用于大多数 GPIO 端口管脚）

该 IOCON 表适用于所有端口管脚，除：P0[7~9]、P0[12~13]、P0[23~31]、P1[30~31]，以及 P5[2~3]。这些管脚包含了 DAC、ADC、USB、I²C，或输入毛刺滤波器功能，它们会改变相关 IOCON 寄存器的内容。

表76. D 类型 IOCON 寄存器位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能。关于具体值，参见表 77。	000
4:3	MODE		选择功能模式（片上上拉/下拉电阻控制）。详见 8.3.2 节。	10
		00	无效（未使能上拉/下拉电阻）。	
		01	下拉电阻使能。	
		10	上拉电阻使能。	
		11	转发器模式。	
5	HYS		迟滞。详见 8.3.3 节。	1
		0	禁能。	
		1	使能。	
6	INV		输入极性。详见 8.3.4 节。	0
		0	输入未被反相（管脚上是高电平时，读值为 1）	
		1	输入被反相（管脚上是高电平时，读值为 0）	
8:7	-		保留。读取值未定义，只写入 0。	无
9	SLEW		驱动器转换速率。详见 8.3.7 节。	0
		0	标准模式下，输出转换速率控制使能。更多输出可被同时切换。	
		1	快速模式下，转换速率控制被禁用。更多详细信息，参见适当的器件数据表。	
10	OD		控制开漏极模式。详见 8.3.9 节。	0
		0	正常的推拉式输出	
		1	模拟的开漏极输出（高驱动禁用）	
31:11	-		保留。读取值未定义，只写入 0。	无

表77. D 型 I/O 控制寄存器：FUNC 值和管脚功能

寄存器	IOCON 寄存器内 FUNC 字段值							
	000	001	010	011	100	101	110	111
IOCON_P0_0	P0[0]	CAN_RD1	U3_TXD	I2C1_SDA	U0_TXD			
IOCON_P0_1	P0[1]	CAN_TD1	U3_RXD	I2C1_SCL	U0_RXD			
IOCON_P0_2	P0[2]	U0_TXD	U3_TXD					
IOCON_P0_3	P0[3]	U0_RXD	U3_RXD					
IOCON_P0_4	P0[4]	I2S_RX_SCK	CAN_RD2	T2_CAP0				LCD_VD[0]
IOCON_P0_5	P0[5]	I2S_RX_WS	CAN_TD2	T2_CAP1				LCD_VD[1]
IOCON_P0_6	P0[6]	I2S_RX_SDA	SSP1_SSEL	T2_MAT0	U1_RTS			LCD_VD[8]
IOCON_P0_10	P0[10]	U2_TXD	I2C2_SDA	T3_MAT0				
IOCON_P0_11	P0[11]	U2_RXD	I2C2_SCL	T3_MAT1				
IOCON_P0_14	P0[14]	USB_HSTEN2	SSP1_SSEL	USB_CONNECT2				
IOCON_P0_15	P0[15]	U1_TXD	SSP0_SCK					
IOCON_P0_16	P0[16]	U1_RXD	SSP0_SSEL					
IOCON_P0_17	P0[17]	U1_CTS	SSP0_MISO					
IOCON_P0_18	P0[18]	U1_DCD	SSP0_MOSI					
IOCON_P0_19	P0[19]	U1_DSR	SD_CLK	I2C1_SDA				
IOCON_P0_20	P0[20]	U1_DTR	SD_CMD	I2C1_SCL				
IOCON_P0_21	P0[21]	U1_RI	SD_PWR	U4_OE	CAN_RD1	U4_CLK		
IOCON_P0_22	P0[22]	U1_RTS	SD_DAT[0]	U4_TXD	CAN_TD1			
IOCON_P1_0	P1[0]	ENET_TXD0		T3_CAP1	SSP2_SCK			
IOCON_P1_1	P1[1]	ENET_TXD1		T3_MAT3	SSP2_MOSI			
IOCON_P1_2	P1[2]	ENET_TXD2	SD_CLK	PWM0[1]				
IOCON_P1_3	P1[3]	ENET_TXD3	SD_CMD	PWM0[2]				
IOCON_P1_4	P1[4]	ENET_TX_EN		T3_MAT2	SSP2_MISO			
IOCON_P1_5	P1[5]	ENET_TX_ER	SD_PWR	PWM0[3]				
IOCON_P1_6	P1[6]	ENET_TX_CLK	SD_DAT[0]	PWM0[4]				
IOCON_P1_7	P1[7]	ENET_COL	SD_DAT[1]	PWM0[5]				
IOCON_P1_8	P1[8]	ENET_CRS		T3_MAT1	SSP2_SSEL			
IOCON_P1_9	P1[9]	ENET_RXD0		T3_MAT0				
IOCON_P1_10	P1[10]	ENET_RXD1		T3_CAP0				

IOCON 寄存器内 FUNC 字段值								
寄存器	000	001	010	011	100	101	110	111
IOCON_P1_11	P1[11]	ENET_RXD2	SD_DAT[2]	PWM0[6]				
IOCON_P1_12	P1[12]	ENET_RXD3	SD_DAT[3]	PWM0_CAP0				
IOCON_P1_13	P1[13]	ENET_RX_DV						
IOCON_P1_14	P1[14]	ENET_RX_ER		T2_CAP0				
IOCON_P1_15	P1[15]	ENET_RX_CLK		I2C2_SDA				
IOCON_P1_16	P1[16]	ENET_MDC	I2S_TX_MCLK					
IOCON_P1_17	P1[17]	ENET_MDIO	I2S_RX_MCLK					
IOCON_P1_18	P1[18]	USB_UP_LED1	PWM1[1]	T1_CAP0		SSP1_MISO		
IOCON_P1_19	P1[19]	USB_TX_E1	USB_PPWR1	T1_CAP1	MC_0A	SSP1_SCK	U2_OE	
IOCON_P1_20	P1[20]	USB_TX_DP1	PWM1[2]	QE1_PHA	MC_FB0	SSP0_SCK	LCD_VD[6]	LCD_VD[10]
IOCON_P1_21	P1[21]	USB_TX_DM1	PWM1[3]	SSP0_SSEL	MC_ABORT		LCD_VD[7]	LCD_VD[11]
IOCON_P1_22	P1[22]	USB_RCV1	USB_PWRD1	T1_MAT0	MC_0B	SSP1_MOSI	LCD_VD[8]	LCD_VD[12]
IOCON_P1_23	P1[23]	USB_RX_DP1	PWM1[4]	QE1_PHB	MC_FB1	SSP0_MISO	LCD_VD[9]	LCD_VD[13]
IOCON_P1_24	P1[24]	USB_RX_DM1	PWM1[5]	QE1_IDX	MC_FB2	SSP0_MOSI	LCD_VD[10]	LCD_VD[14]
IOCON_P1_25	P1[25]	USB_LS1	USB_HSTEN1	T1_MAT1	MC_1A	CLKOUT	LCD_VD[11]	LCD_VD[15]
IOCON_P1_26	P1[26]	USB_SSPND1	PWM1[6]	T0_CAP0	MC_1B	SSP1_SSEL	LCD_VD[12]	LCD_VD[20]
IOCON_P1_27	P1[27]	USB_INT1	USB_OVRCCR1	T0_CAP1	CLKOUT		LCD_VD[13]	LCD_VD[21]
IOCON_P1_28	P1[28]	USB_SCL1	PWM1_CAP0	T0_MAT0	MC_2A	SSP0_SSEL	LCD_VD[14]	LCD_VD[22]
IOCON_P1_29	P1[29]	USB_SDA1	PWM1_CAP1	T0_MAT1	MC_2B	U4_TXD	LCD_VD[15]	LCD_VD[23]
IOCON_P2_0	P2[0]	PWM1[1]	U1_TXD					LCD_PWR
IOCON_P2_1	P2[1]	PWM1[2]	U1_RXD					LCD_LE
IOCON_P2_2	P2[2]	PWM1[3]	U1_CTS	T2_MAT3		TRACEDATA[3]		LCD_DCLK
IOCON_P2_3	P2[3]	PWM1[4]	U1_DCD	T2_MAT2		TRACEDATA[2]		LCD_FP
IOCON_P2_4	P2[4]	PWM1[5]	U1_DSR	T2_MAT1		TRACEDATA[1]		LCD_ENAB_M
IOCON_P2_5	P2[5]	PWM1[6]	U1_DTR	T2_MAT0		TRACEDATA[0]		LCD_LP
IOCON_P2_6	P2[6]	PWM1_CAP0	U1_RI	T2_CAP0	U2_OE	TRACECLK	LCD_VD[0]	LCD_VD[4]
IOCON_P2_7	P2[7]	CAN_RD2	U1_RTS				LCD_VD[1]	LCD_VD[5]
IOCON_P2_8	P2[8]	CAN_TD2	U2_TXD	U1_CTS	ENET_MDC		LCD_VD[2]	LCD_VD[6]
IOCON_P2_9	P2[9]	USB_CONNECT1	U2_RXD	U4_RXD	ENET_MDIO		LCD_VD[3]	LCD_VD[7]
IOCON_P2_10	P2[10]	EINT0	NMI					

IOCON 寄存器内 FUNC 字段值								
寄存器	000	001	010	011	100	101	110	111
IOCON_P2_11	P2[11]	EINT1	SD_DAT[1]	I2S_TX_SCK				LCD_CLKIN
IOCON_P2_12	P2[12]	EINT2	SD_DAT[2]	I2S_TX_WS	LCD_VD[4]	LCD_VD[3]	LCD_VD[8]	LCD_VD[18]
IOCON_P2_13	P2[13]	EINT3	SD_DAT[3]	I2S_TX_SDA		LCD_VD[5]	LCD_VD[9]	LCD_VD[19]
IOCON_P2_14	P2[14]	EMC_CS2	I2C1_SDA	T2_CAP0				
IOCON_P2_15	P2[15]	EMC_CS3	I2C1_SCL	T2_CAP1				
IOCON_P2_16	P2[16]	EMC_CAS						
IOCON_P2_17	P2[17]	EMC_RAS						
IOCON_P2_18	P2[18]	EMC_CLK0						
IOCON_P2_19	P2[19]	EMC_CLK1						
IOCON_P2_20	P2[20]	EMC_DYCS0						
IOCON_P2_21	P2[21]	EMC_DYCS1						
IOCON_P2_22	P2[22]	EMC_DYCS2	SSP0_SCK	T3_CAP0				
IOCON_P2_23	P2[23]	EMC_DYCS3	SSP0_SSEL	T3_CAP1				
IOCON_P2_24	P2[24]	EMC_CKE0						
IOCON_P2_25	P2[25]	EMC_CKE1						
IOCON_P2_26	P2[26]	EMC_CKE2	SSP0_MISO	T3_MAT0				
IOCON_P2_27	P2[27]	EMC_CKE3	SSP0_MOSI	T3_MAT1				
IOCON_P2_28	P2[28]	EMC_DQM0						
IOCON_P2_29	P2[29]	EMC_DQM1						
IOCON_P2_30	P2[30]	EMC_DQM2	I2C2_SDA	T3_MAT2				
IOCON_P2_31	P2[31]	EMC_DQM3	I2C2_SCL	T3_MAT3				
IOCON_P3_0	P3[0]	EMC_D[0]						
IOCON_P3_1	P3[1]	EMC_D[1]						
IOCON_P3_2	P3[2]	EMC_D[2]						
IOCON_P3_3	P3[3]	EMC_D[3]						
IOCON_P3_4	P3[4]	EMC_D[4]						
IOCON_P3_5	P3[5]	EMC_D[5]						
IOCON_P3_6	P3[6]	EMC_D[6]						
IOCON_P3_7	P3[7]	EMC_D[7]						
IOCON_P3_8	P3[8]	EMC_D[8]						

UM10470
All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2011. All rights reserved.

			IOCON 寄存器内 FUNC 字段值					
寄存器	000	001	010	011	100	101	110	111
IOCON_P3_9	P3[9]	EMC_D[9]						
IOCON_P3_10	P3[10]	EMC_D[10]						
IOCON_P3_11	P3[11]	EMC_D[11]						
IOCON_P3_12	P3[12]	EMC_D[12]						
IOCON_P3_13	P3[13]	EMC_D[13]						
IOCON_P3_14	P3[14]	EMC_D[14]						
IOCON_P3_15	P3[15]	EMC_D[15]						
IOCON_P3_16	P3[16]	EMC_D[16]	PWM0[1]	U1_TXD				
IOCON_P3_17	P3[17]	EMC_D[17]	PWM0[2]	U1_RXD				
IOCON_P3_18	P3[18]	EMC_D[18]	PWM0[3]	U1_CTS				
IOCON_P3_19	P3[19]	EMC_D[19]	PWM0[4]	U1_DCD				
IOCON_P3_20	P3[20]	EMC_D[20]	PWM0[5]	U1_DSR				
IOCON_P3_21	P3[21]	EMC_D[21]	PWM0[6]	U1_DTR				
IOCON_P3_22	P3[22]	EMC_D[22]	PWM0_CAP0	U1_RI				
IOCON_P3_23	P3[23]	EMC_D[23]	PWM1_CAP0	T0_CAP0				
IOCON_P3_24	P3[24]	EMC_D[24]	PWM1[1]	T0_CAP1				
IOCON_P3_25	P3[25]	EMC_D[25]	PWM1[2]	T0_MAT0				
IOCON_P3_26	P3[26]	EMC_D[26]	PWM1[3]	T0_MAT1	STCLK			
IOCON_P3_27	P3[27]	EMC_D[27]	PWM1[4]	T1_CAP0				
IOCON_P3_28	P3[28]	EMC_D[28]	PWM1[5]	T1_CAP1				
IOCON_P3_29	P3[29]	EMC_D[29]	PWM1[6]	T1_MAT0				
IOCON_P3_30	P3[30]	EMC_D[30]	U1_RTS	T1_MAT1				
IOCON_P3_31	P3[31]	EMC_D[31]		T1_MAT2				
IOCON_P4_0	P4[0]	EMC_A[0]						
IOCON_P4_1	P4[1]	EMC_A[1]						
IOCON_P4_2	P4[2]	EMC_A[2]						
IOCON_P4_3	P4[3]	EMC_A[3]						
IOCON_P4_4	P4[4]	EMC_A[4]						
IOCON_P4_5	P4[5]	EMC_A[5]						
IOCON_P4_6	P4[6]	EMC_A[6]						

			IOCON 寄存器内 FUNC 字段值						
寄存器	000	001	010	011	100	101	110	111	
IOCON_P4_7	P4[7]	EMC_A[7]							
IOCON_P4_8	P4[8]	EMC_A[8]							
IOCON_P4_9	P4[9]	EMC_A[9]							
IOCON_P4_10	P4[10]	EMC_A[10]							
IOCON_P4_11	P4[11]	EMC_A[11]							
IOCON_P4_12	P4[12]	EMC_A[12]							
IOCON_P4_13	P4[13]	EMC_A[13]							
IOCON_P4_14	P4[14]	EMC_A[14]							
IOCON_P4_15	P4[15]	EMC_A[15]							
IOCON_P4_16	P4[16]	EMC_A[16]							
IOCON_P4_17	P4[17]	EMC_A[17]							
IOCON_P4_18	P4[18]	EMC_A[18]							
IOCON_P4_19	P4[19]	EMC_A[19]							
IOCON_P4_20	P4[20]	EMC_A[20]	I2C2_SDA	SSP1_SCK					
IOCON_P4_21	P4[21]	EMC_A[21]	I2C2_SCL	SSP1_SSEL					
IOCON_P4_22	P4[22]	EMC_A[22]	U2_TXD	SSP1_MISO					
IOCON_P4_23	P4[23]	EMC_A[23]	U2_RXD	SSP1_MOSI					
IOCON_P4_24	P4[24]	EMC_OE							
IOCON_P4_25	P4[25]	EMC_WE							
IOCON_P4_26	P4[26]	EMC_BLS0							
IOCON_P4_27	P4[27]	EMC_BLS1							
IOCON_P4_28	P4[28]	EMC_BLS2	U3_TXD	T2_MAT0		LCD_VD[6]	LCD_VD[10]	LCD_VD[2]	
IOCON_P4_29	P4[29]	EMC_BLS3	U3_RXD	T2_MAT1	I2C2_SCL	LCD_VD[7]	LCD_VD[11]	LCD_VD[3]	
IOCON_P4_30	P4[30]	EMC_CS0							
IOCON_P4_31	P4[31]	EMC_CS1							
IOCON_P5_0	P5[0]	EMC_A[24]		T2_MAT2					
IOCON_P5_1	P5[1]	EMC_A[25]		T2_MAT3					
IOCON_P5_4	P5[4]	U0_OE		T3_MAT3	U4_TXD				

UM10470

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2011. All rights reserved.

8.4.1.2 A 型 IOCON 寄存器（适用于包含模拟功能的管脚）

该 IOCON 表适用于管脚 P0[12~13]、P0[23~26]，以及 P1[30~31]。P0[26]管脚因表示为 DAC 输出而有略微差异，见下面第 16 位的描述。

表78. A 型 IOCON 寄存器位描述

位	符号	值	描述	复位值 ^[1]
2:0	FUNC		选择管脚功能。关于具体值，参见表 79。	0
4:3	MODE		选择功能模式（片上上拉/下拉电阻控制）。参见 8.3.2 节。	10
		00	无效（未使能下拉/上拉电阻）。	
		01	下拉电阻使能。	
		10	上拉电阻使能。	
		11	转发器模式。	
5	-		保留。读取值未定义，只写入 0。	无
6	INVERT		输入极性。参见 8.3.4 节。	0
		0	输入未反相（管脚上是高电平时，读取值为 1）	
		1	输入反相（管脚上是高电平时，读取值为 0）	
7	ADMODE		选择模拟/数字模式。参见 8.3.5 节。	1
		0	模拟模式	
		1	数字模式	
8	FILTER		控制毛刺滤波器。参见 8.3.6 节。	1
		0	低于约 10ns 的噪音脉冲被过滤	
		1	没有输入过滤	
9	-		保留。读取值未定义，只写入 0。	无
10	OD		控制开路漏极模式。参见 8.3.9 “开漏极模式”。	0
		0	正常的推拉式输出	
		1	模拟的开路漏极输出（高电平驱动禁能）	
14:11	-		保留。读取值未定义，只写入 0。	无
16	DACEN		DAC 使能控制。这一位仅适用于 P0[26]。P0[26]包括 DAC 输出功能 DAC_OUT。参见 8.3.10 节。	0
		0	DAC 禁能	
		1	DAC 使能	
31:17	-		保留。读取值未定义，只写入 0。	无

[1] 复位值仅反映已使用位中保存的数据，它不包括保留位的内容。

表79. A 型 I/O 控制寄存器：FUNC 值和管脚功能

IOCON 寄存器内 FUNC 字段值								
寄存器	000	001	010	011	100	101	110	111
IOCON_P0_12	P0[12]	USB_PPWR2	SSP1_MISO	ADC0[6]				
IOCON_P0_13	P0[13]	USB_UP_LED2	SSP1_MOSI	ADC0[7]				
IOCON_P0_23	P0[23]	ADC0[0]	I2S_RX_SCK	T3_CAP0				
IOCON_P0_24	P0[24]	ADC0[1]	I2S_RX_WS	T3_CAP1				
IOCON_P0_25	P0[25]	ADC0[2]	I2S_RX_SDA	U3_TXD				
IOCON_P0_26	P0[26]	ADC0[3]	DAC_OUT	U3_RXD				
IOCON_P1_30	P1[30]	USB_PWRD2	USB_VBUS	ADC[4]	I2C0_SDA	U3_OE		
IOCON_P1_31	P1[31]	USB_OVRCR2	SSP1_SCK	ADC[5]	I2C0_SCL			

8.4.1.3 U 型 IOCON 寄存器（适用于包含 USB D+或 D-功能的管脚）

该 IOCON 表适用于管脚 P0[29]、P0[30]，以及 P0[31]。这些特殊功能管脚不包含可选模式和其它管脚选项。

表80. U 型 IOCON 寄存器位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能。关于具体值，参见表 81。	000
31:3	-		保留。读取值未定义，只写入 0。	无

表81. U 型 I/O 控制寄存器：FUNC 值和管脚功能

寄存器	IOCON 寄存器内 FUNC 字段值							
	000	001	010	011	100	101	110	111
IOCON_P0_29	P0[29]	USB_D+1	EINT0					
IOCON_P0_30	P0[30]	USB_D-1	EINT1					
IOCON_P0_31	P0[31]	USB_D+2						

8.4.1.4 I 型 IOCON 寄存器（适用于包含专用 I²C 功能的管脚）

该 IOCON 表适用于管脚 P0[27~28]和 P5[2~3]。

表82. I 型 IOCON 寄存器位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能。关于具体值，参见表 83。	0
5:3	-		保留。读取值未定义，只写入 0。	无
6	INVERT		输入极性。参见 8.3.4 节。	0
		0	输入未反相（管脚上为高电平时，读取值为 1）	
		1	输入反相（管脚上为高电平时，读取值为 0）	
7	-		保留。读取值未定义，只写入 0。	无
8	HS		为标准模式、FM 及 FM+模式配置 I ² C 特性。参见 8.3.8 节。	0
		0	I ² C 50ns 毛刺滤波器及转换速率控制使能。	
		1	I ² C 50ns 毛刺滤波器及转换速率控制禁能。	
9	HIDRIVE		控制管脚的灌电流能力，仅针对 P5[2] 和 P5[3]而言。参见 8.3.8 节。	0
		0	输出驱动灌电流为 4mA。其足够用于标准及 FM 模式 I ² C。	
		1	输出驱动灌电流为 20mA。这一电流是 FM+模式 I ² C 所需的。更多详细信息，参见适当的设备数据表。	
31:10	-		保留。读取值未定义，只写入 0。	无

[1] 复位值仅反映已使用位中保存的数据，它不包括保留位的内容。

表83. I 型 I/O 控制寄存器：FUNC 值和管脚功能

IOCON 寄存器内 FUNC 字段值								
寄存器	000	001	010	011	100	101	110	111
IOCON_P0_27	P0[27]	I2C0_SDA	USB_SDA1					
IOCON_P0_28	P0[28]	I2C0_SCL	USB_SCL1					
IOCON_P5_2	P5[2]			T3_MAT2		I2C0_SDA		
IOCON_P5_3	P5[3]				U4_RXD	I2C0_SCL		

8.4.1.5 W 型 IOCON 寄存器(这些管脚其它方面与 D 型相同,但包含一个可选输入毛刺滤波器,默认其上拉/下拉被禁用)

该 IOCON 表适用于管脚 P0[7]、P0[8]和 P0[9]。

表84. W 型 IOCON 寄存器位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能。关于具体值, 参见表 85。	000
4:3	MODE		选择管脚的输出功能模式 (片上上拉/下拉电阻控制)。参见 8.3.2 节。	00
		00	无效 (未使能上拉/下拉电阻)	
		01	下拉电阻使能。	
		10	上拉电阻使能。	
		11	转发器模式。	
5	HYS		迟滞。参见 8.3.3 节。	1
		0	禁能。	
		1	使能。	
6	INV		输入极性。参见 8.3.4 节。	0
		0	输入未反相 (管脚上为高电平时, 读取值为 1)	
		1	输入反相 (管脚上为高电平时, 读取值为 0)	
7	-		说明: 为确保正常运行, 该位必须设为“1”。	1
8	FILTER		控制毛刺滤波器。参见 8.3.6 节。	0
		0	低于约 10 ns 的噪音脉冲被过滤	
		1	没有进行输入过滤	
9	SLEW		驱动器转换速率。参见 8.3.7 节。	0
		0	标准模式下, 输出转换速率控制使能。更多输出可被同时切换。	
		1	高速模式下, 转换速率控制被禁能。更多详细信息, 参见适当的设备数据表。	
10	OD		控制开路漏极模式。参见 8.3.9 节。	0
		0	正常的推拉式输出	
		1	模拟的开路漏极输出 (高电平驱动禁能)	
31:11	-		保留。读取值未定义, 只写入 0。	无

表85. W 型 I/O 控制寄存器: FUNC 值和管脚功能

IOCON 寄存器内 FUNC 字段值								
寄存器	000	001	010	011	100	101	110	111
IOCON_P0_7	P0[7]	I2S_TX_SCK	SSP1_SCK	T2_MAT1	RTC_EV0			LCD_VD[9]
IOCON_P0_8	P0[8]	I2S_TX_WS	SSP1_MISO	T2_MAT2	RTC_EV1			LCD_VD[16]
IOCON_P0_9	P0[9]	I2S_TX_SDA	SSP1_MOSI	T2_MAT3	RTC_EV2			LCD_VD[17]

9.1 基本配置

GPIO 的配置使用下列寄存器：

1. 功率：在 PCONP 寄存器中（[表 37](#)），设置位 PCGPIO。它使能 GPIO 自身、GPIO 中断，以及 IOCON 模块。
2. 管脚：GPIO 管脚及其模式见 [8.4.1](#) 节。
3. 唤醒：GPIO 端口 0 和 2 可在需要时用于唤醒，见（[4.7.9](#) 节）。
4. 中断：在 IO0/2IntEnR（[表 100](#)）或 IO0/2IntEnF（[表 102](#)）中使能 GPIO 中断。在 NVIC 中使用相应的中断使能设置寄存器使能中断。

9.2 特性

9.2.1 数字 I/O 端口

- 加速的 GPIO 功能：
 - GPIO 寄存器位于外设 AHB 总线上，以实现高速 I/O 时序。
 - 屏蔽寄存器可在其它位不变的情况下，将一组端口位作为一个组。
 - 所有 GPIO 寄存器均可按字节、半字和字的方式寻址。
 - 整个端口值可用单个指令写入。
 - GPIO 寄存器可以由 GPDMA 访问。
- 位电平置位和清零寄存器能够用单个指令置位清零一个端口的任何位。
- 所有 GPIO 寄存器都支持 Cortex-M3 位带操作。
- GPIO 可由 GPDMA 控制器访问，允许对 GPIO 进行 DMA 数据操作，使之与 DMA 请求同步。
- 每个端口位的方向可控制。
- 所有 I/O 口复位后均默认为上拉输入。

9.2.2 可产生中断的数字端口

- Port 0 和 Port 2 端口的每个管脚都可以提供中断功能。
- 每个端口管脚都可以被编程为上升沿、下降沿或边沿产生中断。
- 边沿检测是非同步的，因此可以在没有时钟的情况下（如在掉电模式下）操作。有了这一特性，就不需要电平触发中断。
- 每个已使能中断均可产生唤醒信号，使器件退出掉电模式。
- 寄存器为软件提供挂起的上升沿中断、挂起的下降沿中断，以及整个挂起的 GPIO 中断。
- GPIO 中断功能并不要求管脚配置为 GPIO。这就允许作为工作外设接口一部分的管脚改变时发生中断。

9.3 应用

- 通用 I/O
- 驱动 LED 或其它指示器
- 控制片外器件
- 探测数字输入信号及边沿信号
- 使器件从掉电模式中唤醒

9.4 管脚描述

表86. GPIO 管脚描述

管脚名称	类型	描述
P0[31:0]; P1[31:0]; P2[31:0]; P3[31:0]; P4[31:0]; P5[4:0]	I/O	通用输入/输出管脚。通常与其他外设功能共用，因此，并非全部 GPIO 都可 在一个应用中使用。在一个特定的器件中，封装选项会影响可用的 GPIO 数目。 某些管脚可能会受到管脚可选功能要求的限制。例如，I ² C 管脚是特殊管脚，用于 该管脚上可选择的任何其他功能。更多详细信息，参见 7.1 节。

9.5 寄存器描述

表 87 中的寄存器表示了适用于所有 GPIO 端口的 GPIO 增强特性。这些寄存器位于一个 AHB 总线上, 以实现快速的读写时序。它们都可以按字节、半字和字的方式寻址。屏蔽寄存器可以从一个 GPIO 端口的其它位独立地访问相同端口中的一组位。

表87. GPIO 寄存器映射 (局部总线可访问寄存器—增强型 GPIO 特性)

通用名称	描述	访问	复位值 ^[1]	PORTn 寄存器名称&地址	表
FIODIR	高速GPIO端口方向控制寄存器。该寄存器单独控制每个端口管脚的方向。	R/W	0	FIO0DIR—0x2009 8000 FIO1DIR—0x2009 8020 FIO2DIR—0x2009 8040 FIO3DIR—0x2009 8060 FIO4DIR—0x2009 8080 FIO5DIR—0x2009 80A0	表 89
FIOMASK	高速端口屏蔽寄存器。写、置位、清零和读端口 (通过写 FIOPIN、FIOSET、FIOCLR和读FIOPIN来执行) 改变或返回时, 只对该寄存器中为“0”的位有效。	R/W	0	FIO0MASK—0x2009 8010 FIO1MASK—0x2009 8030 FIO2MASK—0x2009 8050 FIO3MASK—0x2009 8070 FIO4MASK—0x2009 8090 FIO5MASK—0x2009 80B0	表 97
FIOPIN	高速端口管脚值寄存器, 与FIOMASK结合使用。不管管脚方向或可选的功能选择如何, 数字端口管脚的当前状态可从该寄存器中读出 (只要管脚不配置为ADC的输入)。通过与FIOMASK寄存器进行反相与 (AND) 来屏蔽读出某些位。写该寄存器, 向FIOMASK中为“0”的位填入对应的值。 注: 如果读该寄存器, 那么不管物理管脚的状态如何, 在 FIOMASK 寄存器中被“1”屏蔽的位将始终读出 0。	R/W	0	FIO0PIN—0x2009 8014 FIO1PIN—0x2009 8034 FIO2PIN—0x2009 8054 FIO3PIN—0x2009 8074 FIO4PIN—0x2009 8094 FIO5PIN—0x2009 80B4	表 95
FIOSET	高速端口输出置位寄存器, 与FIOMASK结合使用。该寄存器控制输出管脚的状态。写1使相应的端口管脚产生高电平。写入0无效。读该寄存器返回端口输出寄存器的当前内容。只可以更改FIOMASK中为0的位, 即非屏蔽位。	R/W	0	FIO0SET—0x2009 8018 FIO1SET—0x2009 8038 FIO2SET—0x2009 8058 FIO3SET—0x2009 8078 FIO4SET—0x2009 8098 FIO5SET—0x2009 80B8	表 91
FIOCLR	高速端口输出清零寄存器, 与FIOMASK结合使用。该寄存器控制输出管脚的状态。写1使相应的端口管脚产生低电平。写0无效。只可以更改FIOMASK中为0的位, 即非屏蔽位。	WO	0	FIO0CLR—0x2009 801C FIO1CLR—0x2009 803C FIO2CLR—0x2009 805C FIO3CLR—0x2009 807C FIO4CLR—0x2009 809C FIO5CLR—0x2009 80BC	表 93

[1] 复位值仅反映已使用位中保存的数据, 它不包括保留位的内容。

表88. GPIO 中断映射寄存器

通用名称	描述	访问	复位值 ^[1]	PORTn 寄存器名称&地址	表
IntEnR	上升沿的 GPIO 中断使能。	R/W	0	IO0IntEnR—0x4002 8090	表 100
				IO2IntEnR—0x4002 80B0	表 101
IntEnF	下降沿的 GPIO 中断使能。	R/W	0	IO0IntEnR—0x4002 8094	表 102
				IO2IntEnR—0x4002 80B4	表 103
IntStatR	上升沿的 GPIO 中断状态。	RO	0	IO0IntStatR—0x4002 8084	表 104
				IO2IntStatR—0x4002 80A4	表 105
IntStatF	下降沿的 GPIO 中断状态。	RO	0	IO0IntStatF—0x4002 8088	表 106
				IO2IntStatF—0x4002 80A8	表 107
IntClr	GPIO 中断清零。	WO	0	IO0IntClr—0x4002 808C	表 108
				IO2IntClr—0x4002 80AC	表 109
IntStatus	GPIO 整体中断状态。	RO	0	IOIntStatus—0x4002 8080	表 99

[1] 复位值仅反映已使用位中保存的数据，它不包括保留位的内容。

9.5.1 GPIO 端口方向寄存器 FIOxDIR (FIO0DIR~FIO5DIR—0x2009 8000~0x2009 80A0)

这是一个可按字访问的寄存器，当管脚被配置为 GPIO 端口管脚时，该寄存器用于控制管脚的方向。任何管脚的方向位必须按照管脚的功能置位。

注: GPIO 管脚 P0[29]和 P0[30]是与 USB_D+和 USB_D-管脚共用的，必须有相同的方向。如果 FIO0DIR 的位 29 或位 30 中的一个配置为 0，则 P0[29]和 P0[30]都将为输入。如果 FIO0DIR 的位 29 或位 30 均配置为 1，则 P0[29]和 P0[30]都将为输出。

表89. 高速 GPIO 端口方向寄存器 (FIO0DIR~FIO5DIR—0x2009 8000~0x2009 80A0) 位描述

位	符号	值	描述	复位值
31:0	FIO0DIR	0	高速 GPIO 方向 PORTx 控制位。FIOxDIR 的位 0 控制管脚 Px[0]。FIOxDIR 的位 31 控制管脚 Px[31]。	0x0
	FIO1DIR			
	FIO2DIR	1	控制的管脚为输入管脚。	
	FIO3DIR			
	FIO4DIR			
	FIO5DIR			

除了 32 位长和只能按字访问的 FIODIR 寄存器以外，每个高速 GPIO 端口还可以通过几个按字节和半字访问的寄存器来控制，这些寄存器列在[表 90](#)中。除了提供与 FIODIR 寄存器相同的功能以外，这些额外的寄存器还能更容易、更高速地访问物理端口管脚。

表90. 高速 GPIO 端口方向控制字节和半字访问寄存器描述

通用名称	描述	寄存器长度 (位) &访问	复位值	PORTn 寄存器名称&地址
FIOxDIR0	高速 GPIO 端口 x 方向控制寄存器 0。FIOxDIR0 寄存器中的位 0 对应管脚 Px[0]位 7 对应管脚 Px[7]。	8 (位) R/W	0x00	FIO0DIR0—0x2009 8000 FIO1DIR0—0x2009 8020 FIO2DIR0—0x2009 8040 FIO3DIR0—0x2009 8060 FIO4DIR0—0x2009 8080 FIO5DIR0—0x2009 80A0
FIOxDIR1	高速 GPIO 端口 x 方向控制寄存器 1。FIOxDIR0 寄存器中的位 0 对应管脚 Px[8].....位 7 对应管脚 Px[15]。	8 (位) R/W	0x00	FIO0DIR1—0x2009 8001 FIO1DIR1—0x2009 8021 FIO2DIR1—0x2009 8041 FIO3DIR1—0x2009 8061 FIO4DIR1—0x2009 8081 FIO5DIR1—0x2009 80A1
FIOxDIR2	高速 GPIO 端口 x 方向控制寄存器 2。FIOxDIR0 寄存器中的位 0 对应管脚 Px[16].....位 7 对应管脚 Px[23]。	8 (位) R/W	0x00	FIO0DIR2—0x2009 8002 FIO1DIR2—0x2009 8022 FIO2DIR2—0x2009 8042 FIO3DIR2—0x2009 8062 FIO4DIR2—0x2009 8082 FIO5DIR2—0x2009 80A2
FIOxDIR3	高速 GPIO 端口 x 方向控制寄存器 3。FIOxDIR0 寄存器中的位 0 对应管脚 Px[24].....位 7 对应管脚 Px[31]。	8 (位) R/W	0x00	FIO0DIR3—0x2009 8003 FIO1DIR3—0x2009 8023 FIO2DIR3—0x2009 8043 FIO3DIR3—0x2009 8063 FIO4DIR3—0x2009 8083 FIO5DIR3—0x2009 80A3
FIOxDIRL	高速 GPIO 端口 x 方向控制低半字寄存器。FIOxDIRL 寄存器中的位 0 对应管脚 Px[0].....位 15 对应管脚 Px[15]。	16 (半字) R/W	0x0000	FIO0DIRL—0x2009 8000 FIO1DIRL—0x2009 8020 FIO2DIRL—0x2009 8040 FIO3DIRL—0x2009 8060 FIO4DIRL—0x2009 8080 FIO5DIRL—0x2009 80A0
FIOxDIRU	高速 GPIO 端口 x 方向控制高半字寄存器。FIOxDIRL 寄存器中的位 0 对应管脚 Px[16].....位 15 对应管脚 Px[31]。	16 (半字) R/W	0x0000	FIO0DIRU—0x2009 8002 FIO1DIRU—0x2009 8022 FIO2DIRU—0x2009 8042 FIO3DIRU—0x2009 8062 FIO4DIRU—0x2009 8082 FIO5DIRU—0x2009 80A2

9.5.2 GPIO 端口输出设置寄存器 FIOxSET (FIO0SET~FIO5SET—0x2009 8018~0x2009 80B8)

此寄存器用于在输出模式下，为配置成 GPIO 的端口管脚提供一个高电平输出。写入 1 会在相应端口管脚产生高电平。写入 0 则无效。如果任何管脚被配置为输入或第二种功能，则在 FIOxSET 中的相应位写入 1 无效。

读取 FIOxSET 寄存器可返回该寄存器的值，该值由前次向 FIOxSET 和 FIOxCLR（或上述的 FIOxPIN）的写入所决定。这个值并不反映任何外部环境对 I/O 管脚的影响。

通过 FIOxMASK 寄存器的相应位，可以约束 FIOxSET 寄存器对一个端口管脚的访问（见 [9.5.5](#) 节）。

表91. 高速 GPIO 端口输出设置寄存器（FIO0SET~FIO4SET—0x2009 8018~0x2009 8098）位描述

位	符号	值	描述	复位值
31:0	FIO0SET	0	高速 GPIO 输出值设置位。FIOxSET 的位 0 控制管脚 Px[0]，FIOxSET 的位 31 控制管脚 Px[31]。	0x0
	FIO1SET			
	FIO2SET		控制的管脚输出不改变。	
	FIO3SET	1	控制的管脚输出被设为高电平。	
	FIO4SET			
	FIO5SET			

除了 32 位长和只能按字访问的 FIOxSET 寄存器以外，每个高速 GPIO 端口还可以由 [表 92](#) 所列的几种按字节和半字访问的寄存器所控制。除了提供与 FIOxSET 寄存器相同的功能以外，这些额外的寄存器还能更容易、更高速地访问物理端口的管脚。

表92. 高速 GPIO 端口输出设置字节和半字访问寄存器描述

通用名称	描述	寄存器长度 (位) & 访问	复位值	PORTn 寄存器 名称&地址
FIOxSET0	高速 GPIO 端口 x 输出设置寄存器 0。FIOxSET0 寄存器中的位 0 对应管脚 Px[0].....位 7 对应管脚 Px[7]。	8（位） R/W	0x00	FIO0SET0—0x2009 8018 FIO1SET0—0x2009 8038 FIO2SET0—0x2009 8058 FIO3SET0—0x2009 8078 FIO4SET0—0x2009 8098 FIO5SET0—0x2009 80B8
FIOxSET1	高速 GPIO 端口 x 输出设置寄存器 1。FIOxSET1 寄存器中的位 0 对应管脚 Px[8].....位 7 对应管脚 Px[15]。	8（位） R/W	0x00	FIO0SET1—0x2009 8019 FIO1SET1—0x2009 8039 FIO2SET1—0x2009 8059 FIO3SET1—0x2009 8079 FIO4SET1—0x2009 8099 FIO5SET1—0x2009 80B9
FIOxSET2	高速 GPIO 端口 x 输出设置寄存器 2。FIOxSET2 寄存器中的位 0 对应管脚 Px[16].....位 7 对应管脚 Px[23]。	8（位） R/W	0x00	FIO0SET2—0x2009 801A FIO1SET2—0x2009 803A FIO2SET2—0x2009 805A FIO3SET2—0x2009 807A FIO4SET2—0x2009 809A FIO5SET2—0x2009 80BA

通用名称	描述	寄存器长度 (位) &访问	复位值	PORTn 寄存器 名称&地址
FIOxSET3	高速 GPIO 端口 x 输出设置寄存器 3。FIOxSET3 寄存器中的位 0 对应管脚 Px[24].....位 7 对应管脚 Px[31]。	8 (位) R/W	0x00	FIO0SET3—0x2009 801B FIO1SET3—0x2009 803B FIO2SET3—0x2009 805B FIO3SET3—0x2009 807B FIO4SET3—0x2009 809B FIO5SET3—0x2009 80BB
FIOxSETL	高速 GPIO 端口 x 输出设置低半字寄存器。 FIOxSETL 寄存器中的位 0 对应管脚 Px[0].....位 15 对应管脚 Px[15]。	16 (半字) R/W	0x0000	FIO0SETL—0x2009 8018 FIO1SETL—0x2009 8038 FIO2SETL—0x2009 8058 FIO3SETL—0x2009 8078 FIO4SETL—0x2009 8098 FIO5SETL—0x2009 80B8
FIOxSETU	高速 GPIO 端口 x 输出设置高半字寄存器。 FIOxSETU 寄存器中的位 0 对应管脚 Px[16].....位 31 对应管脚 Px[31]。	16 (半字) R/W	0x0000	FIO0SETU—0x2009 801A FIO1SETU—0x2009 803A FIO2SETU—0x2009 805A FIO3SETU—0x2009 807A FIO4SETU—0x2009 809A FIO5SETU—0x2009 80BA

9.5.3 GPIO 端口输出清零寄存器 FIOxCLR (FIO0CLR~FIO5CLR—0x2009 801C~0x2009 80BC)

此寄存器用于在输出模式下，为配置成 GPIO 的端口管脚提供一个低电平输出。写入 1 会在相应端口管脚产生一个低电平，并清零 FIOxSET 寄存器中的相应位。写入 0 则无效。如果任何管脚被配置为输入或第二种功能，则写入 FIOxCLR 无效。

通过 FIOxMASK 寄存器的相应位，可以约束 FIOxCLR 寄存器对一个端口管脚的访问（见 9.5.5 节）。

表93. 高速 GPIO 端口输出清零寄存器（FIO0CLR~FIO4CLR—0x2009 801C~0x2009 809C）位描述

位	符号	值	描述	复位值
31:0	FIO0CLR	0	高速 GPIO 输出值清零位。 FIOxCLR 的位 0 控制管脚 Px[0]，FIOxCLR 的位 31 控制管脚 Px[31]。	0x0
	FIO1CLR			
	FIO2CLR	1	控制的管脚输出不改变。	
	FIO3CLR		控制的管脚输出被设为低电平。	
	FIO4CLR			
	FIO5CLR			

除了 32 位长和只能按字访问的 FIOxCLR 寄存器以外，每个高速 GPIO 端口还可以由表 94 所列的几种按字节和半字访问的寄存器所控制。除了提供与 FIOxCLR 寄存器相同的功能以外，这些额外的寄存器还能更容易、更高速地访问物理端口管脚。

表94. 高速 GPIO 端口输出清零字节和半字访问寄存器描述

通用名称	描述	寄存器长度 (位) &访问	复位值	PORTn 寄存器 名称&地址
FIOxCLR0	高速 GPIO 端口 x 输出清零寄存器 0。 FIOxCLR0 寄存器中的位 0 对应管脚 Px[0].....位 7 对应管脚 Px[7]。	8 (位) WO	0x00	FIO0CLR0—0x2009 801C FIO1CLR0—0x2009 803C FIO2CLR0—0x2009 805C FIO3CLR0—0x2009 807C FIO4CLR0—0x2009 809C FIO5CLR0—0x2009 80BC
FIOxCLR1	高速 GPIO 端口 x 输出清零寄存器 1。 FIOxCLR1 寄存器中的位 0 对应管脚 Px[8].....位 7 对应管脚 Px[15]。	8 (位) WO	0x00	FIO0CLR1—0x2009 801D FIO1CLR1—0x2009 803D FIO2CLR1—0x2009 805D FIO3CLR1—0x2009 807D FIO4CLR1—0x2009 809D FIO5CLR1—0x2009 80BD
FIOxCLR2	高速 GPIO 端口 x 输出清零寄存器 2。 FIOxCLR2 寄存器中的位 0 对应管脚 Px[16].....位 7 对应管脚 Px[23]。	8 (位) WO	0x00	FIO0CLR2—0x2009 801E FIO1CLR2—0x2009 803E FIO2CLR2—0x2009 805E FIO3CLR2—0x2009 807E FIO4CLR2—0x2009 809E FIO5CLR2—0x2009 80BE
FIOxCLR3	高速 GPIO 端口 x 输出设置寄存器 3。 FIOxCLR3 寄存器中的位 0 对应管脚 Px[24].....位 7 对应管脚 Px[31]。	8 (位) WO	0x00	FIO0CLR3—0x2009 801F FIO1CLR3—0x2009 803F FIO2CLR3—0x2009 805F FIO3CLR3—0x2009 807F FIO4CLR3—0x2009 809F FIO5CLR3—0x2009 80BF
FIOxCLRL	高速 GPIO 端口 x 输出清零低半字寄存器。 FIOxCLRL 寄存器中的位 0 对应管脚 Px[0].....位 15 对应管脚 Px[15]。	16 (半字) WO	0x0000	FIO0CLRL—0x2009 801C FIO1CLRL—0x2009 803C FIO2CLRL—0x2009 805C FIO3CLRL—0x2009 807C FIO4CLRL—0x2009 809C FIO5CLRL—0x2009 80BC
FIOxCLRU	高速 GPIO 端口 x 输出清零高半字寄存器。 FIOxCLRU 寄存器中的位 0 对应管脚 Px[16].....位 15 对应管脚 Px[31]。	16 (半字) WO	0x0000	FIO0CLRU—0x2009 801E FIO1CLRU—0x2009 803E FIO2CLRU—0x2009 805E FIO3CLRU—0x2009 807E FIO4CLRU—0x2009 809E FIO5CLRU—0x2009 80BE

9.5.4 GPIO 端口管脚值寄存器 FIOxPIN (FIO0PIN~FIO5PIN — 0x2009 8014~0x2009 80B4)

该寄存器提供了端口管脚值，可配置这些值来执行仅为数字的功能。寄存器将给出管脚的逻辑值，而与该管脚配置为输入或输出、GPIO 或其他可选数字功能无关。例如，某个特定端口管脚可以将 GPIO 输入、GPIO 输出、UART 接收，以及 PWM 输出作为可选功能。该管脚的任何配置均将允许从相应的 FIOxPIN 寄存器读出其当前逻辑状态。

假设某个管脚配置为模拟功能，如果选择的是模拟配置，则该管脚状态不可读。将管脚选为 A/D 输入会断开该管脚的数字特性。此时，FIOxPIN 寄存器中读出的管脚值无效。

写入 FIOxPIN 寄存器会将其值存储在端口输出寄存器中，而无需同时使用 FIOxSET 寄存器和 FIOxCLR 寄存器获得完整的写入值。在应用中应小心采用这个特性，因为它会影响到整个端口。

通过 FIOxMASK 寄存器的相应位，可以约束由 FIOxPIN 寄存器对一个端口管脚的访问（见 9.5.5 节）。

只有在屏蔽寄存器中用 0 屏蔽的那些管脚（见 9.5.5 节），才会关联到高速 GPIO 端口管脚值寄存器中的当前内容。

表95. 高速 GPIO 端口管脚值寄存器（FIO0PIN~FIO4PIN—0x2009 8014~0x2009 8094）位描述

位	符号	值	描述	复位值
31:0	FIO0VAL	0 1	高速 GPIO 输出值设置位。FIOxCLR 寄存器中的位 0 对应管脚 Px[0].....位 31 对应管脚 Px[31]。	0x0
	FIO1VAL			
	FIO2VAL		控制的管脚输出被设为低电平。	
	FIO3VAL			
	FIO4VAL		控制的管脚输出被设为高电平。	
	FIO5VAL			

除了 32 位长和只能按字访问的 FIOxPIN 寄存器以外，每个高速 GPIO 端口还可以由表 96 所列的几种按字节和半字访问的寄存器所控制。除了提供与 FIOxPIN 寄存器相同的功能以外，这些额外的寄存器还能更容易、更高速地访问物理端口管脚。

表96. 高速 GPIO 端口管脚值字节和半字访问寄存器描述

通用名称	描述	寄存器长度 (位) &访问	复位值	PORTn 寄存器 名称&地址
FIOxPIN0	高速 GPIO 端口 x 管脚值寄存器 0。FIOxPIN0 寄存器中的位 0 对应管脚 Px[0].....位 7 对应管脚 Px[7]。	8 (位) R/W	0x00	FIO0PIN0—0x2009 8014 FIO1PIN0—0x2009 8034 FIO2PIN0—0x2009 8054 FIO3PIN0—0x2009 8074 FIO4PIN0—0x2009 8094 FIO5PIN0—0x2009 80B4
FIOxPIN1	高速 GPIO 端口 x 管脚值寄存器 1。FIOxPIN1 寄存器中的位 0 对应管脚 Px[8].....位 7 对应管脚 Px[15]。	8 (位) R/W	0x00	FIO0PIN1—0x2009 8015 FIO1PIN1—0x2009 8035 FIO2PIN1—0x2009 8055 FIO3PIN1—0x2009 8075 FIO4PIN1—0x2009 8095 FIO5PIN1—0x2009 80B5
FIOxPIN2	高速 GPIO 端口 x 管脚值寄存器 2。FIOxPIN2 寄存器中的位 0 对应管脚 Px[16].....位 7 对应管脚 Px[23]。	8 (位) R/W	0x00	FIO0PIN2—0x2009 8016 FIO1PIN2—0x2009 8036 FIO2PIN2—0x2009 8056 FIO3PIN2—0x2009 8076 FIO4PIN2—0x2009 8096 FIO5PIN2—0x2009 80B6

通用名称	描述	寄存器长度 (位) &访问	复位值	PORTn 寄存器 名称&地址
FIOxPIN3	高速 GPIO 端口 x 管脚值寄存器 3。FIOxPIN3 寄存器中的位 0 对应管脚 Px[24].....位 7 对应管脚 Px[31]。	8（位） R/W	0x00	FIO0PIN3—0x2009 8017 FIO1PIN3—0x2009 8037 FIO2PIN3—0x2009 8057 FIO3PIN3—0x2009 8077 FIO4PIN3—0x2009 8097 FIO5PIN3—0x2009 80B7
FIOxPINL	高速 GPIO 端口 x 管脚值低半字寄存器。FIOxPINL 寄存器中的位 0 对应管脚 Px[0].....位 15 对应管脚 Px[15]。	16（半字） R/W	0x0000	FIO0PINL—0x2009 8014 FIO1PINL—0x2009 8034 FIO2PINL—0x2009 8054 FIO3PINL—0x2009 8074 FIO4PINL—0x2009 8094 FIO5PINL—0x2009 80B4
FIOxPINU	高速 GPIO 端口 x 管脚值高半字寄存器。FIOxPINU 寄存器中的位 0 对应管脚 Px[16].....位 15 对应管脚 Px[31]。	16（半字） R/W	0x0000	FIO0PINU—0x2009 8016 FIO1PINU—0x2009 8036 FIO2PINU—0x2009 8056 FIO3PINU—0x2009 8076 FIO4PINU—0x2009 8096 FIO5PINU—0x2009 80B6

9.5.5 高速 GPIO 端口屏蔽寄存器 FIOxMASK（FIO0MASK~FIO5MASK—0x2009 8010~0x2009 80B0）

此寄存器用于选择哪些端口管脚将能够或不能够被 FIOxPIN、FIOxSET 或 FIOxCLR 寄存器写入访问。当读取 FIOxPIN 寄存器时，屏蔽寄存器还会过滤端口的内容。

通过读或写访问，此寄存器中为“0”的位可以使能相应物理管脚的访问。如果寄存器中的位是“1”，则相应管脚将不能通过写访问而修改，如果是读访问，也不会反映在更新后的 FIOxPIN 寄存器中。软件实例见 9.6 节。

表97. 高速 GPIO 端口屏蔽寄存器（FIO0MASK~FIO4MASK—0x2009 8010~0x2009 8090）位描述

位	符号	值	描述	复位值
31:0	FIO0MASK	0	高速 GPIO 物理管脚访问控制。	0x0
	FIO1MASK		控制的管脚受到 FIOxSET、FIOxCLR 和 FIOxPIN 寄存器的写操作影响。管脚的当前状态可从 FIOxPIN 寄存器中读出。	
	FIO2MASK			
	FIO3MASK	1	控制的管脚不受 FIOxSET、FIOxCLR 和 FIOxPIN 寄存器的写操作影响。读取 FIOxPIN 寄存器时，该位不会通过物理管脚的状态更新。	
	FIO4MASK			
	FIO5MASK			

除了 32 位长和只能按字访问的 FIOxMASK 寄存器以外，每个高速 GPIO 端口还可以由表 98 所列的几种按字节和半字访问的寄存器所控制。除了提供与 FIOxMASK 寄存器相同的功能以外，这些额外的寄存器还能更容易、更高速地访问物理端口管脚。

表98. 高速 GPIO 端口屏蔽字节和半字访问寄存器描述

通用寄存器名称	描述	寄存器长度 (位) &访问	复位值	PORTn 寄存器 名称&地址
FIOxMASK0	高速 GPIO 端口 x 屏蔽寄存器 0。FIOxMASK0 寄存器中的位 0 对应管脚 Px[0].....位 7 对应管脚 Px[7]。	8 (位) R/W	0x0	FIO0MASK0—0x2009 8010 FIO1MASK0—0x2009 8030 FIO2MASK0—0x2009 8050 FIO3MASK0—0x2009 8070 FIO4MASK0—0x2009 8090 FIO5MASK0—0x2009 80B0
FIOxMASK1	高速 GPIO 端口 x 屏蔽寄存器 1。FIOxMASK1 寄存器中的位 0 对应管脚 Px[8].....位 7 对应管脚 Px[15]。	8 (位) R/W	0x0	FIO0MASK1—0x2009 8011 FIO1MASK1—0x2009 8031 FIO2MASK1—0x2009 8051 FIO3MASK1—0x2009 8071 FIO4MASK1—0x2009 8091 FIO5MASK1—0x2009 80B1
FIOxMASK2	高速 GPIO 端口 x 屏蔽寄存器 2。FIOxMASK2 寄存器中的位 0 对应管脚 Px[16].....位 7 对应管脚 Px[23]。	8 (位) R/W	0x0	FIO0MASK2—0x2009 8012 FIO1MASK2—0x2009 8032 FIO2MASK2—0x2009 8052 FIO3MASK2—0x2009 8072 FIO4MASK2—0x2009 8092 FIO5MASK2—0x2009 80B2
FIOxMASK3	高速 GPIO 端口 x 屏蔽寄存器 3。FIOxMASK3 寄存器中的位 0 对应管脚 Px[24].....位 7 对应管脚 Px[31]。	8 (位) R/W	0x0	FIO0MASK3—0x2009 8013 FIO1MASK3—0x2009 8033 FIO2MASK3—0x2009 8053 FIO3MASK3—0x2009 8073 FIO4MASK3—0x2009 8093 FIO5MASK3—0x2009 80B3
FIOxMASKL	高速 GPIO 端口 x 屏蔽低半字寄存器。FIOxMASKL 寄存器中的位 0 对应管脚 Px[0].....位 15 对应管脚 Px[15]。	16 (半字) R/W	0x0	FIO0MASKL—0x2009 8010 FIO1MASKL—0x2009 8030 FIO2MASKL—0x2009 8050 FIO3MASKL—0x2009 8070 FIO4MASKL—0x2009 8090 FIO5MASKL—0x2009 80B0
FIOxMASKU	高速 GPIO 端口 x 屏蔽高半字寄存器。FIOxMASKU 寄存器中的位 0 对应管脚 Px[16].....位 15 对应管脚 Px[31]。	16 (半字) R/W	0x0	FIO0MASKU—0x2009 8012 FIO1MASKU—0x2009 8032 FIO2MASKU—0x2009 8052 FIO3MASKU—0x2009 8072 FIO4MASKU—0x2009 8092 FIO5MASKU—0x2009 80B2

9.5.6 GPIO 中断寄存器

下列寄存器用于配置 Port 0 和 Port 2 端口，以产生中断。

9.5.6.1 GPIO 整体中断状态寄存器 (IOIntStatus—0x4002 8080)

这是个只读的寄存器，它反映了所有支持 GPIO 中断的 GPIO 端口上挂起的中断。每端口只使用一个状态位。

表99. GPIO 整体中断状态寄存器 (IOIntStatus—0x4002 8080) 位描述

位	符号	值	描述	复位值
0	P0Int		Port 0 GPIO 中断挂起。	0
		0	在 Port 0 上没有挂起的中断。	
		1	在 Port 0 上至少有一个挂起的中断。	
1	-		保留。从保留位中读出的值未定义。	无
2	P2Int		Port 2 GPIO 中断挂起。	0
		0	在 Port 2 上没有挂起的中断。	
		1	在 Port 2 上至少有一个挂起的中断。	
31:2	-		保留。从保留位中读出的值未定义。	无

9.5.6.2 端口 0 上升沿的 GPIO 中断使能 (IO0IntEnR—0x4002 8090)

这些读写寄存器的每个位用于使能端口 0 相应管脚的上升沿中断。

表100. 端口 0 上升沿的 GPIO 中断使能 (IO0IntEnR—0x4002 8090) 位描述

位	符号	值	描述 ^[1]	复位值
0	P0.0ER		使能 P0[0]上升沿中断。	0
		0	上升沿中断在 P0[0]上禁能。	
		1	上升沿中断在 P0[0]上使能。	
1	P0.1ER		使能 P0[1]上升沿中断。	0
2	P0.2ER		使能 P0[2]上升沿中断。	0
3	P0.3ER		使能 P0[3]上升沿中断。	0
4	P0.4ER		使能 P0[4]上升沿中断。	0
5	P0.5ER		使能 P0[5]上升沿中断。	0
6	P0.6ER		使能 P0[6]上升沿中断。	0
7	P0.7ER		使能 P0[7]上升沿中断。	0
8	P0.8ER		使能 P0[8]上升沿中断。	0
9	P0.9ER		使能 P0[9]上升沿中断。	0
10	P0.10ER		使能 P0[10]上升沿中断。	0
11	P0.11ER		使能 P0[11]上升沿中断。	0
12	P0.12ER		使能 P0[12]上升沿中断。	0
13	P0.13ER		使能 P0[13]上升沿中断。	0
14	P0.14ER		使能 P0[14]上升沿中断。	0
15	P0.15ER		使能 P0[15]上升沿中断。	0
16	P0.16ER		使能 P0[16]上升沿中断。	0
17	P0.17ER		使能 P0[17]上升沿中断。	0
18	P0.18ER		使能 P0[18]上升沿中断。	0
19	P0.19ER		使能 P0[19]上升沿中断。	0
20	P0.20ER		使能 P0[20]上升沿中断。	0
21	P0.21ER		使能 P0[21]上升沿中断。	0
22	P0.22ER		使能 P0[22]上升沿中断。	0
23	P0.23ER		使能 P0[23]上升沿中断。	0
24	P0.24ER		使能 P0[24]上升沿中断。	0
25	P0.25ER		使能 P0[25]上升沿中断。	0
26	P0.26ER		使能 P0[26]上升沿中断。	0
27	P0.27ER		使能 P0[27]上升沿中断。	0
28	P0.28ER		使能 P0[28]上升沿中断。	0
29	P0.29ER		使能 P0[29]上升沿中断。	0
30	P0.30ER		使能 P0[30]上升沿中断。	0
31	P0.31ER		使能 P0[31]上升沿中断。	0

[1] 哪种管脚适用取决于产品型号和封装组合。更多详细信息，参见具体设备数据表。

9.5.6.3 端口 2 上升沿的 GPIO 中断使能 (IO2IntEnR—0x4002 80B0)

这些读写寄存器的每个位用于使能端口 2 相应管脚的上升沿中断。

表101. 端口 2 上升沿的 GPIO 中断使能 (IO2IntEnR—0x4002 80B0) 位描述

位	符号	值	描述 ^[1]	复位值
0	P2.0ER		使能 P2[0]上升沿中断。	0
		0	上升沿中断在 P2[0]上禁能。	
		1	上升沿中断在 P2[0]上使能。	
1	P2.1ER		使能 P2[1]上升沿中断。	0
2	P2.2ER		使能 P2[2]上升沿中断。	0
3	P2.3ER		使能 P2[3]上升沿中断。	0
4	P2.4ER		使能 P2[4]上升沿中断。	0
5	P2.5ER		使能 P2[5]上升沿中断。	0
6	P2.6ER		使能 P2[6]上升沿中断。	0
7	P2.7ER		使能 P2[7]上升沿中断。	0
8	P2.8ER		使能 P2[8]上升沿中断。	0
9	P2.9ER		使能 P2[9]上升沿中断。	0
10	P2.10ER		使能 P2[10]上升沿中断。	0
11	P2.11ER		使能 P2[11]上升沿中断。	0
12	P2.12ER		使能 P2[12]上升沿中断。	0
13	P2.13ER		使能 P2[13]上升沿中断。	0
14	P2.14ER		使能 P2[14]上升沿中断。	0
15	P2.15ER		使能 P2[15]上升沿中断。	0
16	P2.16ER		使能 P2[16]上升沿中断。	0
17	P2.17ER		使能 P2[17]上升沿中断。	0
18	P2.18ER		使能 P2[18]上升沿中断。	0
19	P2.19ER		使能 P2[19]上升沿中断。	0
20	P2.20ER		使能 P2[20]上升沿中断。	0
21	P2.21ER		使能 P2[21]上升沿中断。	0
22	P2.22ER		使能 P2[22]上升沿中断。	0
23	P2.23ER		使能 P2[23]上升沿中断。	0
24	P2.24ER		使能 P2[24]上升沿中断。	0
25	P2.25ER		使能 P2[25]上升沿中断。	0
26	P2.26ER		使能 P2[26]上升沿中断。	0
27	P2.27ER		使能 P2[27]上升沿中断。	0
28	P2.28ER		使能 P2[28]上升沿中断。	0
29	P2.29ER		使能 P2[29]上升沿中断。	0
30	P2.30ER		使能 P2[30]上升沿中断。	0
31	P2.31ER		使能 P2[31]上升沿中断。	0

[1] 哪种管脚适用取决于产品型号和封装组合。更多详细信息，参见具体设备数据表。

9.5.6.4 端口 0 下降沿的 GPIO 中断使能 (IO0IntEnF—0x4002 8094)

这些读写寄存器的每个位用于使能端口 0 相应管脚的下降沿中断。

表102. 端口 0 下降沿的 GPIO 中断使能 (IO0IntEnF—0x4002 8094) 位描述

位	符号	值	描述 ^[1]	复位值
0	P0.0EF		使能 P0[0]下降沿中断。	0
		0	下降沿中断在 P0[0]上禁能。	
		1	下降沿中断在 P0[0]上使能。	
1	P0.1EF		使能 P0[1]下降沿中断。	0
2	P0.2EF		使能 P0[2]下降沿中断。	0
3	P0.3EF		使能 P0[3]下降沿中断。	0
4	P0.4EF		使能 P0[4]下降沿中断。	0
5	P0.5EF		使能 P0[5]下降沿中断。	0
6	P0.6EF		使能 P0[6]下降沿中断。	0
7	P0.7EF		使能 P0[7]下降沿中断。	0
8	P0.8EF		使能 P0[8]下降沿中断。	0
9	P0.9EF		使能 P0[9]下降沿中断。	0
10	P0.10EF		使能 P0[10]下降沿中断。	0
11	P0.11EF		使能 P0[11]下降沿中断。	0
12	P0.12EF		使能 P0[12]下降沿中断。	0
13	P0.13EF		使能 P0[13]下降沿中断。	0
14	P0.14EF		使能 P0[14]下降沿中断。	0
15	P0.15EF		使能 P0[15]下降沿中断。	0
16	P0.16EF		使能 P0[16]下降沿中断。	0
17	P0.17EF		使能 P0[17]下降沿中断。	0
18	P0.18EF		使能 P0[18]下降沿中断。	0
19	P0.19EF		使能 P0[19]下降沿中断。	0
20	P0.20EF		使能 P0[20]下降沿中断。	0
21	P0.21EF		使能 P0[21]下降沿中断。	0
22	P0.22EF		使能 P0[22]下降沿中断。	0
23	P0.23EF		使能 P0[23]下降沿中断。	0
24	P0.24EF		使能 P0[24]下降沿中断。	0
25	P0.25EF		使能 P0[25]下降沿中断。	0
26	P0.26EF		使能 P0[26]下降沿中断。	0
27	P0.27EF		使能 P0[27]下降沿中断。	0
28	P0.28EF		使能 P0[28]下降沿中断。	0
29	P0.29EF		使能 P0[29]下降沿中断。	0
30	P0.30EF		使能 P0[30]下降沿中断。	0
31	P0.31EF		使能 P0[31]下降沿中断。	0

[1] 哪种管脚适用取决于产品型号和封装组合。更多详细信息，参见具体设备数据表。

9.5.6.5 端口 2 下降沿的 GPIO 中断使能 (IO2IntEnF—0x4002 80B4)

这些读写寄存器的每个位用于使能端口 2 相应管脚的下降沿中断。

表103. 端口 2 下降沿的 GPIO 中断使能 (IO2IntEnF—0x4002 80B4) 位描述

位	符号	值	描述 ^[1]	复位值
0	P2.0EF		使能 P2[0]下降沿中断。	0
		0	下降沿中断在 P2[0]上禁能。	
		1	下降沿中断在 P2[0]上使能。	
1	P2.1EF		使能 P2[1]下降沿中断。	0
2	P2.2EF		使能 P2[2]下降沿中断。	0
3	P2.3EF		使能 P2[3]下降沿中断。	0
4	P2.4EF		使能 P2[4]下降沿中断。	0
5	P2.5EF		使能 P2[5]下降沿中断。	0
6	P2.6EF		使能 P2[6]下降沿中断。	0
7	P2.7EF		使能 P2[7]下降沿中断。	0
8	P2.8EF		使能 P2[8]下降沿中断。	0
9	P2.9EF		使能 P2[9]下降沿中断。	0
10	P2.10EF		使能 P2[10]下降沿中断。	0
11	P2.11EF		使能 P2[11]下降沿中断。	0
12	P2.12EF		使能 P2[12]下降沿中断。	0
13	P2.13EF		使能 P2[13]下降沿中断。	0
14	P2.14EF		使能 P2[14]下降沿中断。	0
15	P2.15EF		使能 P2[15]下降沿中断。	0
16	P2.16EF		使能 P2[16]下降沿中断。	0
17	P2.17EF		使能 P2[17]下降沿中断。	0
18	P2.18EF		使能 P2[18]下降沿中断。	0
19	P2.19EF		使能 P2[19]下降沿中断。	0
20	P2.20EF		使能 P2[20]下降沿中断。	0
21	P2.21EF		使能 P2[21]下降沿中断。	0
22	P2.22EF		使能 P2[22]下降沿中断。	0
23	P2.23EF		使能 P2[23]下降沿中断。	0
24	P2.24EF		使能 P2[24]下降沿中断。	0
25	P2.25EF		使能 P2[25]下降沿中断。	0
26	P2.26EF		使能 P2[26]下降沿中断。	0
27	P2.27EF		使能 P2[27]下降沿中断。	0
28	P2.28EF		使能 P2[28]下降沿中断。	0
29	P2.29EF		使能 P2[29]下降沿中断。	0
30	P2.30EF		使能 P2[30]下降沿中断。	0
31	P2.31EF		使能 P2[31]下降沿中断。	0

[1] 哪种管脚适用取决于产品型号和封装组合。更多详细信息，参见具体设备数据表。

9.5.6.6 端口 0 上升沿中断的 GPIO 中断状态 (IO0IntStatR—0x4002 8084)

这些只读寄存器中的每个位都表示端口 0 的上升沿中断状态。

表104. 端口 0 上升沿中断的 GPIO 中断状态 (IO0IntStatR—0x4002 8084) 位描述

位	符号	值	描述 ^[1]	复位值
0	P0.0REI		P0[0]上升沿中断状态。	0
		0	尚未检测到 P0[0]上有上升沿。	
		1	P0[0]上的上升沿导致中断产生。	
1	P0.1REI		P0[1]上升沿中断状态。	0
2	P0.2REI		P0[2]上升沿中断状态。	0
3	P0.3REI		P0[3]上升沿中断状态。	0
4	P0.4REI		P0[4]上升沿中断状态。	0
5	P0.5REI		P0[5]上升沿中断状态。	0
6	P0.6REI		P0[6]上升沿中断状态。	0
7	P0.7REI		P0[7]上升沿中断状态。	0
8	P0.8REI		P0[8]上升沿中断状态。	0
9	P0.9REI		P0[9]上升沿中断状态。	0
10	P0.10REI		P0[10]上升沿中断状态。	0
11	P0.11REI		P0[11]上升沿中断状态。	0
12	P0.12REI		P0[12]上升沿中断状态。	0
13	P0.13REI		P0[13]上升沿中断状态。	0
14	P0.14REI		P0[14]上升沿中断状态。	0
15	P0.15REI		P0[15]上升沿中断状态。	0
16	P0.16REI		P0[16]上升沿中断状态。	0
17	P0.17REI		P0[17]上升沿中断状态。	0
18	P0.18REI		P0[18]上升沿中断状态。	0
19	P0.19REI		P0[19]上升沿中断状态。	0
20	P0.20REI		P0[20]上升沿中断状态。	0
21	P0.21REI		P0[21]上升沿中断状态。	0
22	P0.22REI		P0[22]上升沿中断状态。	0
23	P0.23REI		P0[23]上升沿中断状态。	0
24	P0.24REI		P0[24]上升沿中断状态。	0
25	P0.25REI		P0[25]上升沿中断状态。	0
26	P0.26REI		P0[26]上升沿中断状态。	0
27	P0.27REI		P0[27]上升沿中断状态。	0
28	P0.28REI		P0[28]上升沿中断状态。	0
29	P0.29REI		P0[29]上升沿中断状态。	0
30	P0.30REI		P0[30]上升沿中断状态。	0
31	P0.31REI		P0[31]上升沿中断状态。	0

[1] 哪种管脚适用取决于产品型号和封装组合。更多详细信息，参见适当的设备数据表。

9.5.6.7 端口 2 上升沿中断的 GPIO 中断状态（IO2IntStatR—0x4002 80A4）

这些只读寄存器中的每个位都表示端口 2 的上升沿中断状态。

表105. 端口 2 上升沿中断的 GPIO 中断状态（IO2IntStatR—0x4002 80A4）位描述

位	符号	值	描述 ^[1]	复位值
0	P2.0REI		P2[0]上升沿中断状态。	0
		0	尚未检测到 P2[0]上有上升沿。	
		1	P2[0]上的上升沿导致中断产生。	
1	P2.1REI		P2[1]上升沿中断状态。	0
2	P2.2REI		P2[2]上升沿中断状态。	0
3	P2.3REI		P2[3]上升沿中断状态。	0
4	P2.4REI		P2[4]上升沿中断状态。	0
5	P2.5REI		P2[5]上升沿中断状态。	0
6	P2.6REI		P2[6]上升沿中断状态。	0
7	P2.7REI		P2[7]上升沿中断状态。	0
8	P2.8REI		P2[7]上升沿中断状态。	0
9	P2.9REI		P2[9]上升沿中断状态。	0
10	P2.10REI		P2[10]上升沿中断状态。	0
11	P2.11REI		P2[11]上升沿中断状态。	0
12	P2.12REI		P2[12]上升沿中断状态。	0
13	P2.13REI		P2[13]上升沿中断状态。	0
14	P2.14REI		P2[14]上升沿中断状态。	0
15	P2.15REI		P2[15]上升沿中断状态。	0
16	P2.16REI		P2[16]上升沿中断状态。	0
17	P2.17REI		P2[17]上升沿中断状态。	0
18	P2.18REI		P2[18]上升沿中断状态。	0
19	P2.19REI		P2[19]上升沿中断状态。	0
20	P2.20REI		P2[20]上升沿中断状态。	0
21	P2.21REI		P2[21]上升沿中断状态。	0
22	P2.22REI		P2[22]上升沿中断状态。	0
23	P2.23REI		P2[23]上升沿中断状态。	0
24	P2.24REI		P2[24]上升沿中断状态。	0
25	P2.25REI		P2[25]上升沿中断状态。	0
26	P2.26REI		P2[26]上升沿中断状态。	0
27	P2.27REI		P2[27]上升沿中断状态。	0
28	P2.28REI		P2[28]上升沿中断状态。	0
29	P2.29REI		P2[29]上升沿中断状态。	0
30	P2.30REI		P2[30]上升沿中断状态。	0
31	P2.31REI		P2[31]上升沿中断状态。	0

[1] 哪种管脚适用取决于产品型号和封装组合。更多详细信息，参见适当的设备数据表。

9.5.6.8 端口 0 下降沿中断的 GPIO 中断状态 (IO0IntStatF—0x4002 8088)

这些只读寄存器中的每个位都表示端口 0 的下降沿中断状态。

表106. 端口 0 下降沿中断的 GPIO 中断状态 (IO0IntStatF—0x4002 8088) 位描述

位	符号	值	描述 ^[1]	复位值
0	P0.0FEI		P0[0]下降沿中断状态。	0
		0	尚未检测到 P0[0]上有下降沿。	
		1	P0[0]上的下降沿导致中断产生。	
1	P0.1FEI		P0[1]下降沿中断状态。	0
2	P0.2FEI		P0[2]下降沿中断状态。	0
3	P0.3FEI		P0[3]下降沿中断状态。	0
4	P0.4FEI		P0[4]下降沿中断状态。	0
5	P0.5FEI		P0[5]下降沿中断状态。	0
6	P0.6FEI		P0[6]下降沿中断状态。	0
7	P0.7FEI		P0[7]下降沿中断状态。	0
8	P0.8FEI		P0[8]下降沿中断状态。	0
9	P0.9FEI		P0[9]下降沿中断状态。	0
10	P0.10FEI		P0[10]下降沿中断状态。	0
11	P0.11FEI		P0[11]下降沿中断状态。	0
12	P0.12FEI		P0[12]下降沿中断状态。	0
13	P0.13FEI		P0[13]下降沿中断状态。	0
14	P0.14FEI		P0[14]下降沿中断状态。	0
15	P0.15FEI		P0[15]下降沿中断状态。	0
16	P0.16FEI		P0[16]下降沿中断状态。	0
17	P0.17FEI		P0[17]下降沿中断状态。	0
18	P0.18FEI		P0[18]下降沿中断状态。	0
19	P0.19FEI		P0[19]下降沿中断状态。	0
20	P0.20FEI		P0[20]下降沿中断状态。	0
21	P0.21FEI		P0[21]下降沿中断状态。	0
22	P0.22FEI		P0[22]下降沿中断状态。	0
23	P0.23FEI		P0[23]下降沿中断状态。	0
24	P0.24FEI		P0[24]下降沿中断状态。	0
25	P0.25FEI		P0[25]下降沿中断状态。	0
26	P0.26FEI		P0[26]下降沿中断状态。	0
27	P0.27FEI		P0[27]下降沿中断状态。	0
28	P0.28FEI		P0[28]下降沿中断状态。	0
29	P0.29FEI		P0[29]下降沿中断状态。	0
30	P0.30FEI		P0[30]下降沿中断状态。	0
31	P0.31FEI		P0[31]下降沿中断状态。	0

[1] 哪种管脚适用取决于产品型号和封装组合。更多详细信息，参见适当的设备数据表。

9.5.6.9 端口 2 下降沿中断的 GPIO 中断状态 (IO2IntStatF—0x4002 80A8)

这些只读寄存器中的每个位都表示端口 2 的下降沿中断状态。

表107. 端口 2 下降沿中断的 GPIO 中断状态 (IO2IntStatF—0x4002 80A8) 位描述

位	符号	值	描述 ^[1]	复位值
0	P2.0FEI		P2[0]下降沿中断状态。	0
		0	尚未检测到 P2[0]上有下降沿。	
		1	P2[0]上的下降沿导致中断产生。	
1	P2.1FEI		P2[1]下降沿中断状态。	0
2	P2.2FEI		P2[2]下降沿中断状态。	0
3	P2.3FEI		P2[3]下降沿中断状态。	0
4	P2.4FEI		P2[4]下降沿中断状态。	0
5	P2.5FEI		P2[5]下降沿中断状态。	0
6	P2.6FEI		P2[6]下降沿中断状态。	0
7	P2.7FEI		P2[7]下降沿中断状态。	0
8	P2.8FEI		P2[8]下降沿中断状态。	0
9	P2.9FEI		P2[9]下降沿中断状态。	0
10	P2.10FEI		P2[10]下降沿中断状态。	0
11	P2.11FEI		P2[11]下降沿中断状态。	0
12	P2.12FEI		P2[12]下降沿中断状态。	0
13	P2.13FEI		P2[13]下降沿中断状态。	0
14	P2.14FEI		P2[14]下降沿中断状态。	0
15	P2.15FEI		P2[15]下降沿中断状态。	0
16	P2.16FEI		P2[16]下降沿中断状态。	0
17	P2.17FEI		P2[17]下降沿中断状态。	0
18	P2.18FEI		P2[18]下降沿中断状态。	0
19	P2.19FEI		P2[19]下降沿中断状态。	0
20	P2.20FEI		P2[20]下降沿中断状态。	0
21	P2.21FEI		P2[21]下降沿中断状态。	0
22	P2.22FEI		P2[22]下降沿中断状态。	0
23	P2.23FEI		P2[23]下降沿中断状态。	0
24	P2.24FEI		P2[24]下降沿中断状态。	0
25	P2.25FEI		P2[25]下降沿中断状态。	0
26	P2.26FEI		P2[26]下降沿中断状态。	0
27	P2.27FEI		P2[27]下降沿中断状态。	0
28	P2.28FEI		P2[28]下降沿中断状态。	0
29	P2.29FEI		P2[29]下降沿中断状态。	0
30	P2.30FEI		P2[30]下降沿中断状态。	0
31	P2.31FEI		P2[31]下降沿中断状态。	0

[1] 哪种管脚适用取决于产品型号和封装组合。更多详细信息，参见适当的设备数据表。

9.5.6.10 端口 0 的 GPIO 中断清零寄存器 (IO0IntClr—0x4002 808C)

向这个只写寄存器中的位写入 1 可清零相应端口 0 管脚的任何中断。

表108. 端口 0 的 GPIO 中断清零寄存器 (IO0IntClr—0x4002 808C) 位描述

位	符号	值	描述 ^[1]	复位值
0	P0.0CI		P0[0]清零 GPIO 端口中断。	0
		0	IOxIntStatR 和 IOxIntStatF 中的相应位不变。	
		1	IOxIntStatR 和 IOxIntStatF 中的相应位清零。	
1	P0.1CI		P0[1]清零 GPIO 端口中断。	0
2	P0.2CI		P0[2]清零 GPIO 端口中断。	0
3	P0.3CI		P0[3]清零 GPIO 端口中断。	0
4	P0.4CI		P0[4]清零 GPIO 端口中断。	0
5	P0.5CI		P0[5]清零 GPIO 端口中断。	0
6	P0.6CI		P0[6]清零 GPIO 端口中断。	0
7	P0.7CI		P0[7]清零 GPIO 端口中断。	0
8	P0.8CI		P0[8]清零 GPIO 端口中断。	0
9	P0.9CI		P0[9]清零 GPIO 端口中断。	0
10	P0.10CI		P0[10]清零 GPIO 端口中断。	0
11	P0.11CI		P0[11]清零 GPIO 端口中断。	0
12	P0.12CI		P0[12]清零 GPIO 端口中断。	0
13	P0.13CI		P0[13]清零 GPIO 端口中断。	0
14	P0.14CI		P0[14]清零 GPIO 端口中断。	0
15	P0.15CI		P0[15]清零 GPIO 端口中断。	0
16	P0.16CI		P0[16]清零 GPIO 端口中断。	0
17	P0.17CI		P0[17]清零 GPIO 端口中断。	0
18	P0.18CI		P0[18]清零 GPIO 端口中断。	0
19	P0.19CI		P0[19]清零 GPIO 端口中断。	0
20	P0.20CI		P0[20]清零 GPIO 端口中断。	0
21	P0.21CI		P0[21]清零 GPIO 端口中断。	0
22	P0.22CI		P0[22]清零 GPIO 端口中断。	0
23	P0.23CI		P0[23]清零 GPIO 端口中断。	0
24	P0.24CI		P0[24]清零 GPIO 端口中断。	0
25	P0.25CI		P0[25]清零 GPIO 端口中断。	0
26	P0.26CI		P0[26]清零 GPIO 端口中断。	0
27	P0.27CI		P0[27]清零 GPIO 端口中断。	0
28	P0.28CI		P0[28]清零 GPIO 端口中断。	0
29	P0.29CI		P0[29]清零 GPIO 端口中断。	0
30	P0.30CI		P0[30]清零 GPIO 端口中断。	0
31	P0.31CI		P0[31]清零 GPIO 端口中断。	0

[1] 哪种管脚适用取决于产品型号和封装组合。更多详细信息，参见适当的设备数据表。

9.5.6.11 端口 2 的 GPIO 中断清零寄存器 (IO2IntClr—0x4002 80AC)

向这个只写寄存器中的位写入 1 可清零相应端口 2 管脚的任何中断。

表109. 端口 2 的 GPIO 中断清零寄存器 (IO2IntClr—0x4002 80AC) 位描述

位	符号	值	描述 ^[1]	复位值
0	P2.0CI		P2[0]清零 GPIO 端口中断。	0
		0	IOxIntStatR 和 IOxIntStatF 中的相应位不变。	
		1	IOxIntStatR 和 IOxIntStatF 中的相应位清零。	
1	P2.1CI		P2[1]清零 GPIO 端口中断。	0
2	P2.2CI		P2[2]清零 GPIO 端口中断。	0
3	P2.3CI		P2[3]清零 GPIO 端口中断。	0
4	P2.4CI		P2[4]清零 GPIO 端口中断。	0
5	P2.5CI		P2[5]清零 GPIO 端口中断。	0
6	P2.6CI		P2[6]清零 GPIO 端口中断。	0
7	P2.7CI		P2[7]清零 GPIO 端口中断。	0
8	P2.8CI		P2[8]清零 GPIO 端口中断。	0
9	P2.9CI		P2[9]清零 GPIO 端口中断。	0
10	P2.10CI		P2[10]清零 GPIO 端口中断。	0
11	P2.11CI		P2[11]清零 GPIO 端口中断。	0
12	P2.12CI		P2[12]清零 GPIO 端口中断。	0
13	P2.13CI		P2[13]清零 GPIO 端口中断。	0
14	P2.14CI		P2[14]清零 GPIO 端口中断。	0
15	P2.15CI		P2[15]清零 GPIO 端口中断。	0
16	P2.16CI		P2[16]清零 GPIO 端口中断。	0
17	P2.17CI		P2[17]清零 GPIO 端口中断。	0
18	P2.18CI		P2[18]清零 GPIO 端口中断。	0
19	P2.19CI		P2[19]清零 GPIO 端口中断。	0
20	P2.20CI		P2[20]清零 GPIO 端口中断。	0
21	P2.21CI		P2[21]清零 GPIO 端口中断。	0
22	P2.22CI		P2[22]清零 GPIO 端口中断。	0
23	P2.23CI		P2[23]清零 GPIO 端口中断。	0
24	P2.24CI		P2[24]清零 GPIO 端口中断。	0
25	P2.25CI		P2[25]清零 GPIO 端口中断。	0
26	P2.26CI		P2[26]清零 GPIO 端口中断。	0
27	P2.27CI		P2[27]清零 GPIO 端口中断。	0
28	P2.28CI		P2[28]清零 GPIO 端口中断。	0
29	P2.29CI		P2[29]清零 GPIO 端口中断。	0
30	P2.30CI		P2[30]清零 GPIO 端口中断。	0
31	P2.31CI		P2[31]清零 GPIO 端口中断。	0

[1] 哪种管脚适用取决于产品型号和封装组合。更多详细信息，参见适当的设备数据表。

9.6 GPIO 使用注意事项

9.6.1 实例: GPIO 端口上 0s 和 1s 的瞬时输出

方法 1: 使用 32 位 (字) 可访问的高速 GPIO 寄存器

```
FIOOMASK = 0xFFFF00FF;  
FIOOPIN  = 0x0000A500;
```

方法 2: 使用 16 位 (半字) 可访问的高速 GPIO 寄存器

```
FIOOMASKL = 0x00FF;  
FIOOPINL  = 0xA500;
```

方法 3: 使用 8 位 (字节) 可访问的高速 GPIO 寄存器

```
FIOOPIN1  = 0xA5;
```

9.6.2 写入 FIOSET/FIOCLR 与 FIOPIN 的比较

写入 FIOSET/FIOCLR 寄存器能够使一个程序方便地将端口的输出管脚同时修改为高电平和低电平。当使用 FIOSET 或 FIOCLR 时, 只有写为 1 的管脚/位将做改变, 而写为 0 的那些则不受影响。

写入 FIOPIN 寄存器可实现一个需要值在并行 GPIO 上的即时输出。写入 FIOPIN 寄存器的数据将影响到该端口上配置为输出的所有管脚: 0 值将产生低电平的管脚输出, 而 1 值则产生高电平的管脚输出。

用 FIOOMASK 寄存器定义哪些管脚要受影响, 就可以修改端口管脚的一个子集。FIOOMASK 在要修改管脚的对应位置 0, 其它所有位为 1。上述 [9.6.1](#) 节中的方法 2 表示在 PORT0 管脚 15 至 8 上输出 0xA5, 而 PORT0 上所有其它输出管脚则保持原值。

10.1 如何阅读本章

本章描述了支持 LP178x/7x 设备外存的外部存储控制器（EMC）。对于支持外存的器件，其 EMC 配置会随不同的封装而变化，见[表 110](#)。

表110. EMC 配置

设备封装	所支持的数据总线宽度	适用的管脚	动态存储器配置寄存器 [1][2]	静态存储器配置寄存器 [1][3]	外部存储器连接
144-管脚	8 位	EMC_A[15:0] EMC_D[7:0] <u>EMC_OE</u> <u>EMC_WE</u> <u>EMC_CS1:0</u>		EMCStaticConfig1/0 EMCStaticWaitWen1/0 EMCStaticWaitOen1/0 EMCStaticWaitRd1/0 EMCStaticWaitPage1/0 EMCStaticWaitWr1/0 EMCStaticWaitTurn1/0	参见 10.13.3 节。
180-管脚	16 位, 8 位	EMC_A[19:0] EMC_D[15:0] <u>EMC_OE</u> <u>EMC_WE</u> <u>EMC_BLS1:0</u> <u>EMC_CS1:0</u> <u>EMC_DYCS1:0</u> <u>EMC_CAS</u> <u>EMC_RAS</u> EMC_CLK1:0 EMC_CKE1:0 EMC_DQM1:0	EMCDynamicConfig1/0 EMCDynamicRasCas1/0	EMCStaticConfig1/0 EMCStaticWaitWen1/0 EMCStaticWaitOen1/0 EMCStaticWaitRd1/0 EMCStaticWaitPage1/0 EMCStaticWaitWr1/0 EMCStaticWaitTurn1/0	参见 10.13.2 节。 参见 10.13.3 节。

设备封装	所支持的数据总线宽度	适用的管脚	动态存储器配置寄存器 ^{[1][2]}	静态存储器配置寄存器 ^{[1][3]}	外部存储器连接
208-管脚	32 位, 16 位, 8 位	EMC_A[25:0] EMC_D[31:0] <u>EMC_OE</u> <u>EMC_WE</u> <u>EMC_BLS3:0</u> <u>EMC_CS3:0</u> <u>EMC_DYCS3:0</u> <u>EMC_CAS</u> <u>EMC_RAS</u> EMC_CLK1:0 EMC_CKE3:0 EMC_DQM3:0	EM CDynamicConfig3/2/1/0 EMCDynamicRasCas3/2/1/0	EMCStaticConfig3/2/1/0 EMCStaticWaitWen3/2/1/0 EMCStaticWaitOen3/2/1/0 EMCStaticWaitRd3/2/1/0 EMCStaticWaitPage3/2/1/0 EMCStaticWaitWr3/2/1/0 EMCStaticWaitTurn3/2/1/0	参见 10.13.1 节。 参见 10.13.2 节。 参见 10.13.3 节。

- [1] 除了普遍用于所有 EMC 操作的寄存器: EMCControl 和 EMCConfig。
- [2] 除了普遍用于所有 EMC 动态片选的寄存器: EMCDynamicControl、EMCDynamicRefresh、EMCDynamicReadConfig、EMCDynamicRP、EMCDynamicRAS、EMCDynamicSREX、EMCDynamicAPR、EMCDynamicDAL、EMCDynamicWR、EMCDynamicRC、EMCDynamicRFC、EMCDynamicXSR、EMCDynamicRRD 和 EMCDynamicMRD。
- [3] 除了适用于静态片选的 EMCStaticExtendedWait 寄存器。

10.2 基本配置

EMC 配置使用下列寄存器:

1. 功率: PCONP 寄存器 ([表 37](#)), 置位 PCEMC 位。
注: EMC 会因复位被使能 (PCEMC = 1)。在 POR 和热复位时, EMC 亦被使能, 见 [表 114](#) 和 [表 117](#)。
2. 时钟: EMC 时钟可以与 CPU 时钟相同 (默认), 或为 CPU 时钟的一半。当 CPU 运行速度快于外部总线能够支持的速度时, 应首先使用较低速率。EMC 的时钟选择见 [4.6.3](#) 节。
3. 管脚: 通过相关的 IOCON 寄存器选择 EMC 管脚与管脚模式 ([8.4.1](#) 节)
4. 配置: 见 [表 114](#) 至 [表 117](#)。更多 EMC 配置另见 [3.8.1](#) 节。特别要保证为应用硬件正确地配置地址移位模式。
5. MPU: Cortex-M3 的默认存储空间许可不允许执行包含有动态存储器片选的地址区间内的程序。通过对 MPU 编程可以修改这些许可 (见 [39.4.5](#) 节)。

10.3 简介

LPC178x/177x 外部存储控制器 (EMC) 是一个 ARM PrimeCell™ 多端口存储控制器外设, 它支持异步静态存储设备, 如 RAM、ROM 和 Flash, 以及像单数据速率 SDRAM 这种动态存储器。EMC 是一个符合先进微控制器总线架构 (AMBA) 规范的外设。

10.4 特性

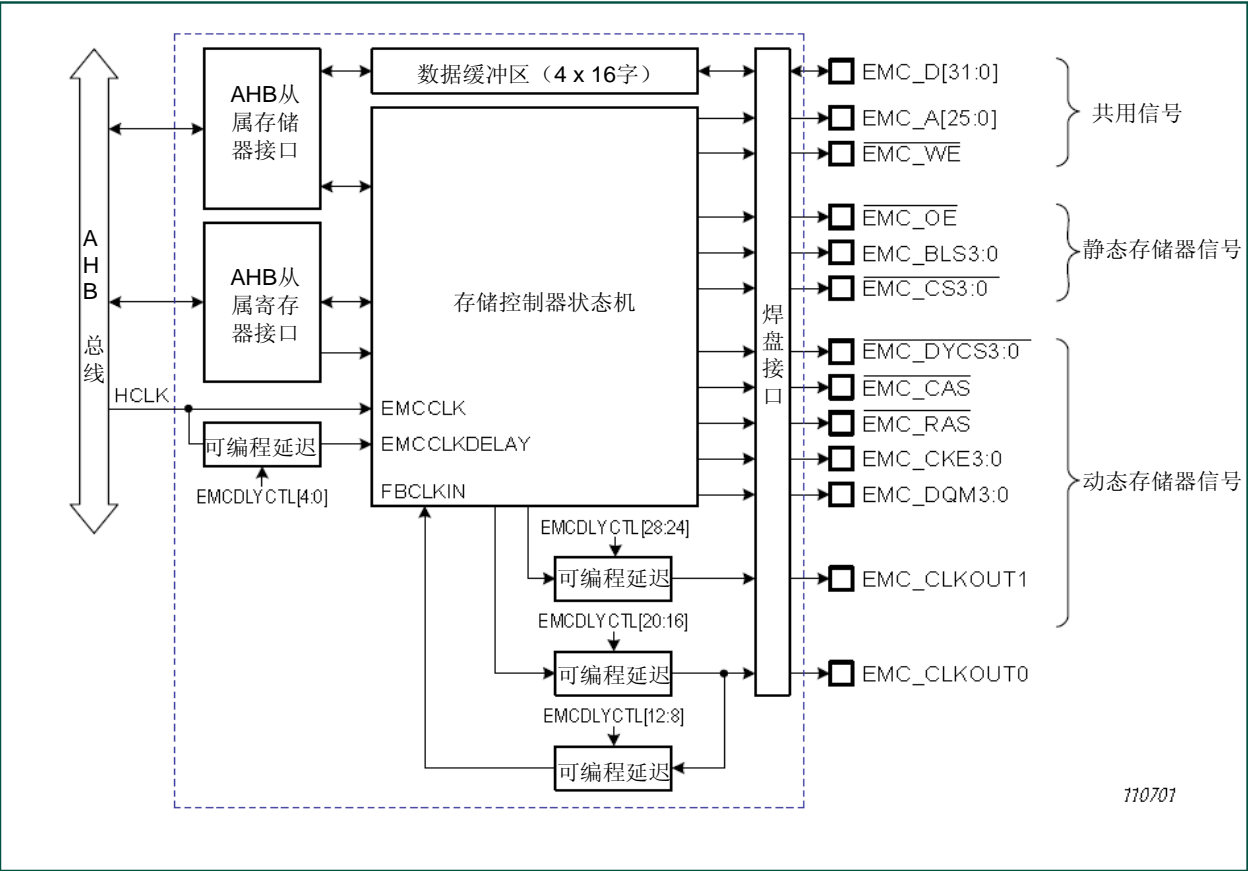
- 静态片选支持高达 64MB 数据。通过使能地址移位模式, 静态片选 0 可以支持高达 256MB 数据, 而静态片选 1 则可以支持高达 128MB 数据 (见 SCS 寄存器位 0 (3.8.1 节))。
- 动态片选支持高达 256MB 数据。
- 支持动态存储器接口, 包括单数据速率 SDRAM。
- 支持异步静态存储设备, 包括 RAM、ROM, 以及 Flash, 可以有或没有异步页读模式。
- 低处理延时。
- 减少延时并提高性能的读写缓冲区。
- 支持 8 位、16 位和 32 位宽静态存储器。
- 支持 16 位和 32 位宽片选的 SDRAM 存储器。
- 静态存储器特性包括:
 - 异步页读模式读取
 - 可编程等待状态
 - 总线周转延迟
 - 输出使能与写使能延迟
 - 延长的等待
- 4 个同步存储器片选, 4 个静态存储器片选。
- 动态控制 SDRAM 的 CKE 与 CLKOUT 的节电模式。
- 软件控制的动态存储器自刷新模式。
- 控制器可支持 2 kbit、4 kbit 和 8 kbit 原地址同步存储器件。
一般是 512Mb、256Mb 和 128Mb 器件, 每器件有 4、8、16 或 32 个数据位。
- 独立的复位域, 必要时可通过一个芯片复位进行自动刷新。
- 可编程延迟元件, 允许精细的 EMC 时序调整。

注: 不支持同步静态存储设备 (同步突发模式)。

10.5 EMC 功能描述

[图 13](#) 为 EMC 的框图。

图13. EMC 方框图



EMC 模块的功能描述分为以下部分:

- AHB 从寄存器接口
- AHB 从存储器接口
- 数据缓冲区
- 存储控制器状态机
- 焊盘接口

注: 对于 32 位宽的片选, 数据与动态存储器之间的传输是采用长度为 4 的 SDRAM 突发。对 16 位宽的片选, 采用的是长度为 8 的 SDRAM 突发。

10.5.1 AHB 从寄存器接口

AHB 从寄存器接口模块用于使能 EMC 寄存器使其被编程。这个模块还包括了大多数寄存器, 用于完成大部分寄存器地址解码工作。

为了避免出现字节序 (endianness) 问题, 所有从 EMC 寄存器输入或输出的数据必须为 32 位宽。

注: 如果尝试使用非一个字 (32 位) 的长度访问, 则会引起 AHB 总线产生 ERROR 响应, 传输被终止。

10.5.2 AHB 从存储器接口

AHB 从存储器接口允许访问外部存储器。

10.5.2.1 存储器传输的字节序

数据输入和输出外部存储器的字节序由 EMCConfig 寄存器中的 Endian 模式 (N) 位决定。

注: 在改变字节序以前, 存储控制器必须是空闲状态 (见 EMCStatus 寄存器的忙域), 这样数据才能正确传输。

10.5.2.2 存储器传输的大小

存储器事务处理的宽度可以是 8、16 或 32 位。任何尝试以大于一个字 (32 位) 方式访问都会在 AHB 总线产生 ERROR 响应, 传输被中止。

10.5.2.3 写保护的存储区

对写保护存储区执行写操作会在 AHB 总线产生一个 ERROR 响应, 传输被中止。

10.5.3 焊盘接口

焊盘接口模块用于为焊盘提供接口。焊盘接口使用了一个反馈时钟 FBCLKIN, 它来自于

EMC 的 CLKOUT0 输出, 用于片外读取的 SDRAM 数据与片上域的再同步。

10.5.4 数据缓冲区

AHB 接口通过缓冲区完成读写操作, 提高了存储器带宽, 减少了处理事务延时。EMC 带有 4 个 16 字的缓冲区。这些缓冲区可以用作读缓冲区、写缓冲区, 或两者组合使用。缓冲区的分配是自动完成的。

在 SDRAM 初始化期间, 缓冲区必须禁用。在正常工作期间, 缓冲区必须使能。

使用 EMCStaticConfig 寄存器, 可以为静态存储器使能或禁能缓冲区。

10.5.4.1 写缓冲区

写缓冲区用于:

- 合并写入事务, 这样就能尽量减少外部事务数量。
在 EMC 完成写入事务前, 缓冲数据, 从而改善 AHB 的写入延时。
将所有动态存储器的写入事务转换为外部存储器接口的 4 字脉冲串(quadword burst)。这样就改进了动态存储器的传输效率。
- 减少外部存储器的通信。从而提高了存储器带宽, 降低了功耗。

写缓冲区的操作:

- 如果缓冲区使能, 则 AHB 写操作会写入到最近使用最少(LRU)的缓冲区(如果为空)。如果 LRU 缓冲区不为空, 则缓冲区内容被输出到存储器, 为 AHB 写入数据释放空间。
- 如果缓冲区包含的写入数据已被标记为“dirty”, 则其内容要先写入存储器, 然后就可以重新分配缓冲区。

一旦出现下列情况, 则写缓冲区被刷新输出:

- 存储控制器状态机未忙于访问外部存储器。
存储控制器状态机未忙于访问外部存储器, AHB 接口正写入另外的缓冲区。

注: 对于动态存储器, 缓冲区的最小输出是 4 字数据。对于静态存储器, 缓冲区的最小输出是 1 字节数据。

10.5.4.2 读缓冲区

读缓冲区用于:

- 缓冲来自存储器的读请求。这样, 以后写入缓冲区的读请求就会从缓冲区中读数据, 而不是从存储器, 从而减小了事务延时。
将所有读事务转换为外部存储器接口的 4 字脉冲串。这样能提高动态存储器的传输效率。
- 减少外部存储器的通信。这样能改善存储器带宽, 降低功耗。

读缓冲区的操作:

- 如果缓冲区使能, 要读取的数据已在某个缓冲区内, 则数据将从缓冲区中直接读出。
- 如果要读取的数据不在缓冲区内, 则会选择 LRU 缓冲区。如果缓冲区为 “dirty” (包含了写入数据), 则写入的数据被输出至存储器。当一个空缓冲区可用时, 读命令被提交给存储器。

通过读取存储器而填充的缓冲区被标记为 “not-dirty” (不包含写入的数据), 其内容不会传回存储控制器, 除非其后 AHB 传输执行对该缓冲区的写操作。

10.5.5 存储控制器状态机

存储控制器状态机包括一个静态存储控制器和一个动态存储控制器。

10.5.6 通过可编程延时元素的时序控制

可编程延时元素用于精细调整与 SDRAM 存储器连接的 EMC 操作中各方面的时序。

- 对于时钟延迟工作模式, 为每个可能的时钟输出 (CLKOUT0 和 CLKOUT1) 提供一个独立的可编程延迟。
- 对于命令延迟工作模式, 为所有命令输出提供一个可编程的控制延迟。
- 对于这两种工作模式, 提供用于控制时间的一个可编程的延迟, 这个时间是从 SDRAM 存储器采样输入数据的时间。

可编程延迟的位置见 EMC 总框图 (图 13)。更多信息, 见 EMCDLYCTL 与 EMCCAL 寄存器的描述。

10.6 低功率操作

很多系统在低功率睡眠模式时必须维持存储器系统中的内容。EMC 提供了一种机制, 可将动态存储器置于自刷新 (self-refresh) 模式。

通过软件置位 EMCDynamicControl 寄存器中的 SREFREQ 位, 并轮询 EMCStatus 寄存器的 SREFACK 位, 即可进入自刷新模式。

当存储控制器处于自刷新模式时, 产生的任何存储器事务都将被拒绝, 并且 AHB 总线产生一个错误响应。清零 EMCDynamicControl 寄存器中的 SREFREQ 位, 可使存储器返回到正常工作状态。有关刷新的要求, 见存储器的数据表。

注: 当 SDRAM 存储器处于自刷新模式时, 静态存储器能够正常存取。

10.6.1 低功率 SDRAM 深度睡眠模式

EMC 支持 JEDEC 的低功率 SDRAM 深度睡眠模式, 进入方式是通过置位 EMCDynamicControl 寄存器中的深度睡眠模式 (DP) 位、动态存储器时钟使能 (CE) 位,

以及动态时钟控制（CS）位。然后，设备就进入低功率模式，此时设备掉电，不再刷新。存储器中所有数据均丢失。

10.6.2 低功率 SDRAM 部分数组刷新

EMC 支持 JEDEC 的低功率 SDRAM 部分数组刷新。通过对 SDRAM 存储器件的适当初始化，可以编程设定部分数组刷新。当存储器件进入自刷新模式时，只有指定的存储器通道被刷新。未得到刷新的存储器通道将丢失其数据内容。

10.7 存储器组选择

支持 8 个可独立配置的存储器片选：

- 管脚 `EMC_CS3` 至 `EMC_CS0` 用于选择静态存储器件。
- 管脚 `EMC_DYCS3` 至 `EMC_DYCS0` 用于选择动态存储器件。

静态存储器片选大小是每个 64M 字节，而动态存储器片选是每个 256M 字节。[表 111](#) 给出了片选的地址区间。

表111. 存储器组的选择

片选管脚	地址范围	存储器类型	范围大小
<code>EMC_CS0</code>	0x8000 0000—0x83FF FFFF	静态	64Mb
<code>EMC_CS1</code>	0x9000 0000—0x93FF FFFF	静态	64Mb
<code>EMC_CS2</code>	0x9800 0000—0x9BFF FFFF	静态	64Mb
<code>EMC_CS3</code>	0x9C00 0000—0x9FFF FFFF	静态	64Mb
<code>EMC_DYCS0</code>	0xA000 0000—0xAFFF FFFF	动态	256Mb
<code>EMC_DYCS1</code>	0xB000 0000—0xBFFF FFFF	动态	256Mb
<code>EMC_DYCS2</code>	0xC000 0000—0xCFFF FFFF	动态	256Mb
<code>EMC_DYCS3</code>	0xD000 0000—0xDFFF FFFF	动态	256Mb

10.8 EMC 复位

EMC 接收两种复位信号。一种是上电复位，当芯片使用电源时以及当检测到掉电状态时（有关掉电检测详见“系统控制模块”一章）有效。另一种复位则来自外部复位管脚和看门狗定时器。

SCS 寄存器中有一个配置位 `EMC_Reset_Disable`，用于控制 EMC 复位的方式（见 [3.8.1](#) 节）。默认的配置（`EMC_Reset_Disable = 0`）是：当发生任何类型的复位事件时，两种 EMC 复位均有效。这种模式下，EMC 的所有寄存器与功能都会在任何复位条件下初始化。

如果 `EMC_Reset_Disable` 设为 1，则 EMC 的大部分功能只在上电或掉电事件时才复位，这样，EMC 就可以通过热复位（外部复位或看门狗复位）保持状态。如果 EMC 的配置正

确，通过热复位时可以维持自动刷新。

10.9 地址移位模式

EMC 对静态存储器支持可选的地址移位模式，它可以简化电路板设计，某些情况下有可能增加外部存储器的寻址空间。后一种情况见本手册“存储器映射”一章中[表 3](#)的脚注。

地址移位模式由 SCS 寄存器中的一个配置位控制，该位为 EMC 移位控制（见 [3.8.1](#) 节）。

当地址移位模式无效时（SCS 寄存器中的 EMC 移位控制位为 1），静态存储器地址输出为字节地址。这意味着对宽度大于一个字节的存储器，不能使用一或两根地址线，并且电路板设计必须移位存储器件的地址连线。例如，如果连接了一个 32 位宽的存储系统，则存储器件的最低地址线会跳过位 0 和位 1，而连接到 EMC 位 2 地址线。

当地址移位模式有效时（SCS 寄存器中的 EMC 移位控制位为 0），静态存储器地址被移动，与总线宽度所要求的最低地址位相匹配。这样，存储器件的最低地址线就总是连接到 EMC 的位 0 地址线。

10.10 存储器映射 I/O 与突发禁能

默认情况下，EMC 采用缓冲的方式获得更好的外部存储器存取性能。不过，对于存储器映射的 I/O 设备，因缓冲而产生的预读操作可能会引起设备的问题。问题可能来源于，读另一个寄存器时使某个寄存器的状态发生变化，或因为在专门读取设备中其它寄存器时，引起了对某个数据 FIFO 的误读。

为防止出现这种问题，可以禁能对真实 CPU 内存读请求的预读。控制这一功能的配置位是 EMC 突发控制（EMC Burst Control），它位于 SCS 寄存器中（见 [3.8.1](#) 节）。

10.11 管脚描述

[表 112](#) 列出了 EMC 的接口与控制信号管脚。

表112. 焊盘接口和控制信号描述

名称	类型	上电复位值	自刷新期间值	描述
EMC_A[23:0]	输出	0	取决于静态存储器存取	外部存储器地址输出，用于静态和 SDRAM 设备。SDRAM 存储器仅使用位[14:0]。
EMC_D[31:0]	输入/输出	数据输出= 0	取决于静态存储器存取	外部存储器数据线。当数据从外部存储器中读取时，它们为输入；当数据被写入外部存储器时，它们为输出。
$\overline{\text{EMC_OE}}$	输出	1	取决于静态存储器存取	静态存储设备的低电平有效输出使能。
EMC_BLS3:0	输出	0xF	取决于静态存储器存取	低电平有效字节通道选择，用于静态存储设备。
$\overline{\text{EMC_WE}}$	输出	1	取决于静态存储器存取	低电平有效写入使能，用于 SDRAM 和静态存储器。
$\overline{\text{EMC_CS3:0}}$	输出	0xF	取决于静态存储器存取	静态存储器片选，默认为低电平有效，用于静态存储设备。

名称	类型	上电复位值	自刷新期间值	描述
			器存取	
EMC_DYCS3:0	输出	0xF	0xF	SDRAM 片选, 用于 SDRAM 设备。
EMC_CAS	输出	1	1	列地址选通脉冲, 用于 SDRAM 设备。
EMC_RAS	输出	1	1	行地址选通脉冲, 用于 SDRAM 设备。
EMC_CLK1:0	输出	遵循 CCLK	遵循 CCLK	SDRAM 时钟, 用于 SDRAM 设备。
EMC_CKE3:0	输出	0xF	0x0	SDRAM 时钟使能, 用于 SDRAM 设备。每个片选配置一个。
EMC_DQM3:0	输出	0xF	0xF	数据屏蔽输出至 SDRAM, 用于 SDRAM 和静态存储器。

10.12 寄存器描述

本章描述了 EMC 寄存器, 并提供了微控制器编程所需要的细节。EMC 寄存器见[表 113](#)。

表113. EMC 寄存器汇总

地址	寄存器名称	描述	热复位值 ^[1]	上电复位值 ^[1]	类型	表
0x2009 C000	EMCControl	控制存储器控制器的操作。	0x1	0x3	R/W	表 114
0x2009 C004	EMCStatus	提供 EMC 状态信息。	-	0x5	RO	表 115
0x2009 C008	EMCConfig	配置存储器控制器的操作	-	0x0	R/W	表 116
0x2009 C020	EMCDynamicControl	控制动态存储器操作。	-	0x006	R/W	表 117
0x2009 C024	EMCDynamicRefresh	配置动态存储器刷新操作。	-	0x0	R/W	表 118
0x2009C028	EMCDynamicReadConfig	配置动态存储器的读取模式。	-	0x0	R/W	表 119
0x2009 C030	EMCDynamicRP	选择预充电的命令周期。	-	0x0F	R/W	表 120
0x2009 C034	EMCDynamicRAS	选择有效至预充电的命令周期。	-	0xF	R/W	表 121
0x2009 C038	EMCDynamicSREX	选择自刷新退出时间。	-	0xF	R/W	表 122
0x2009 C03C	EMCDynamicAPR	选择最后数据输出至有效命令时间。	-	0xF	R/W	表 123
0x2009 C040	EMCDynamicDAL	选择数据进入至有效命令时间。	-	0xF	R/W	表 124
0x2009 C044	EMCDynamicWR	选择写入恢复时间。	-	0xF	R/W	表 125
0x2009 C048	EMCDynamicRC	选择有效至有效命令周期。	-	0x1F	R/W	表 126
0x2009 C04C	EMCDynamicRFC	选择自刷新周期。	-	0x1F	R/W	表 127
0x2009 C050	EMCDynamicXSR	选择退出自刷新至有效命令时间。	-	0x1F	R/W	表 128
0x2009 C054	EMCDynamicRRD	选择有效组 A 至有效组 B 的延时。	-	0xF	R/W	表 129
0x2009 C058	EMCDynamicMRD	选择装载模式寄存器至有效命令时间。	-	0xF	R/W	表 130
0x2009 C080	EMCStaticExtendedWait	选择长静止存储器的读写传输时间。	-	0x0	R/W	表 131
0x2009 C100	EMCDynamicConfig0	选择 EMC_DYCS0 配置信息。	-	0x0	R/W	表 132
0x2009 C104	EMCDynamicRasCas0	选择 EMC_DYCS0 的 RAS 和 CAS 延迟。	-	0x303	R/W	表 134
0x2009 C120	EMCDynamicConfig1	选择 EMC_DYCS1 配置信息。	-	0x0	R/W	表 132
0x2009 C124	EMCDynamicRasCas1	选择 EMC_DYCS1 的 RAS 和 CAS 延迟。	-	0x303	R/W	表 134
0x2009 C140	EMCDynamicConfig2	选择 EMC_DYCS2 配置信息。	-	0x0	R/W	表 132

地址	寄存器名称	描述	热复位值 ^[1]	上电复位值 ^[1]	类型	表
0x2009 C144	EMCDynamicRasCas2	选择 $\overline{\text{EMC_DYCS2}}$ 的 RAS 和 CAS 延迟。	-	0x303	R/W	表 134
0x2009 C160	EMCDynamicConfig3	选择 $\overline{\text{EMC_DYCS3}}$ 配置信息。	-	0x0	R/W	表 132
0x2009 C164	EMCDynamicRasCas3	选择 $\overline{\text{EMC_DYCS3}}$ 的 RAS 和 CAS 延迟。	-	0x303	R/W	表 134
0x2009 C200	EMCStaticConfig0	选择 $\overline{\text{EMC_CS0}}$ 存储器配置。	-	0x0	R/W	表 135
0x2009 C204	EMCStaticWaitWen0	选择从 $\overline{\text{EMC_CS0}}$ 到写使能的延迟。	-	0x0	R/W	表 136
0x2009 C208	EMCStaticWaitOen0	选择从 $\overline{\text{EMC_CS0}}$ 或地址变化（两者中较晚的）到输出使能的延迟。	-	0x0	R/W	表 137
0x2009 C20C	EMCStaticWaitRd0	选择从 $\overline{\text{EMC_CS0}}$ 到读取的延迟。	-	0x1F	R/W	表 138
0x2009 C210	EMCStaticWaitPage0	选择 $\overline{\text{EMC_CS0}}$ 异步页读模式的顺序存取延迟。	-	0x1F	R/W	表 139
0x2009 C214	EMCStaticWaitWr0	选择从 $\overline{\text{EMC_CS0}}$ 到写入的延迟。	-	0x1F	R/W	表 140
0x2009 C218	EMCStaticWaitTurn0	选择 $\overline{\text{EMC_CS0}}$ 总线的周转周期数。	-	0xF	R/W	表 141
0x2009 C220	EMCStaticConfig1	选择 $\overline{\text{EMC_CS1}}$ 存储器配置。	-	0x0	R/W	表 135
0x2009 C224	EMCStaticWaitWen1	选择 $\overline{\text{EMC_CS1}}$ 到写使能的延迟。	-	0x0	R/W	表 136
0x2009 C228	EMCStaticWaitOen1	选择从 $\overline{\text{EMC_CS1}}$ 或地址变化（两者中较晚的）至输出使能的延迟。	-	0x0	R/W	表 137
0x2009 C22C	EMCStaticWaitRd1	选择从 $\overline{\text{EMC_CS1}}$ 到读取的延迟。	-	0x1F	R/W	表 138
0x2009 C230	EMCStaticWaitPage1	选择 $\overline{\text{EMC_CS1}}$ 异步页读模式的顺序存取延迟。	-	0x1F	R/W	表 139
0x2009 C234	EMCStaticWaitWr1	选择从 $\overline{\text{EMC_CS1}}$ 到写入的延迟。	-	0x1F	R/W	表 140
0x2009 C238	EMCStaticWaitTurn1	选择 $\overline{\text{EMC_CS1}}$ 总线的周转周期数。	-	0xF	R/W	表 141
0x2009 C240	EMCStaticConfig2	选择 $\overline{\text{EMC_CS2}}$ 存储器配置。	-	0x0	R/W	表 135
0x2009 C244	EMCStaticWaitWen2	选择从 $\overline{\text{EMC_CS2}}$ 到写使能的延迟。	-	0x0	R/W	表 136
0x2009 C248	EMCStaticWaitOen2	选择从 $\overline{\text{EMC_CS2}}$ 或地址变化（两者中较晚的）至输出使能的延迟。	-	0x0	R/W	表 137
0x2009 C24C	EMCStaticWaitRd2	选择从 $\overline{\text{EMC_CS2}}$ 到读取的延迟。	-	0x1F	R/W	表 138
0x2009 C250	EMCStaticWaitPage2	选择 $\overline{\text{EMC_CS2}}$ 异步页读模式的顺序存取延迟。	-	0x1F	R/W	表 139
0x2009 C254	EMCStaticWaitWr2	选择从 $\overline{\text{EMC_CS2}}$ 到写入的延迟。	-	0x1F	R/W	表 140
0x2009 C258	EMCStaticWaitTurn2	选择 $\overline{\text{EMC_CS2}}$ 总线的周转周期数。	-	0xF	R/W	表 141
0x2009 C260	EMCStaticConfig3	选择 $\overline{\text{EMC_CS3}}$ 存储器配置。	-	0x0	R/W	表 135
0x2009 C264	EMCStaticWaitWen3	选择从 $\overline{\text{EMC_CS3}}$ 到写使能的延迟。	-	0x0	R/W	表 136

地址	寄存器名称	描述	热复位值 ^[1]	上电复位值 ^[1]	类型	表
0x2009 C268	EMCStaticWaitOen3	选择从 $\overline{\text{EMC_CS3}}$ 或地址变化（两者中较晚的）至输出使能的延迟。	-	0x0	R/W	表 137
0x2009 C26C	EMCStaticWaitRd3	选择从 $\overline{\text{EMC_CS3}}$ 到读取的延迟。	-	0x1F	R/W	表 138
0x2009 C270	EMCStaticWaitPage3	选择 $\overline{\text{EMC_CS3}}$ 异步页模式的顺序存取延迟。	-	0x1F	R/W	表 139
0x2009 C274	EMCStaticWaitWr3	选择从 $\overline{\text{EMC_CS3}}$ 到写入的延迟。	-	0x1F	R/W	表 140
0x2009 C278	EMCStaticWaitTurn3	选择 $\overline{\text{EMC_CS3}}$ 总线的周转周期数。	-	0xF	R/W	表 141
0x400F C1DC	EMCDLYCTL	选择与 SDRAM 运行相关的 4 个可编程延迟值。	-	0x210	R/W	表 142
0x400F C1E0	EMCCAL	控制可编程延迟的校准计数器并返回结果值。	-	0x1F00	R/W	表 143

[1] 复位值仅反映使用位中保存的数据，它并不包括保留位的内容。

10.12.1 EMC 控制寄存器（EMCControl—0x2009 C000）

EMCControl 寄存器是一个读写寄存器，控制存储控制器的操作。在正常操作期间，可以改变控制位。[表 114](#) 给出了 EMCControl 寄存器各位的分配。

表114. EMC 控制寄存器位描述（EMCControl-0x2009 C000）

位	符号	值	描述	复位值
0	EMC Enable (E)		表明 EMC 是否被使能或禁能：	1
		0	禁能	
		1	使能（POR 和热复位值） EMC 禁能可降低功耗。当存储器控制器禁能时，存储器不能刷新。通过设定使能或复位来使能存储器控制器。 该位必须在 EMC 为空闲状态时进行修改。 ^[1]	
1	Address mirror (M)		表明正常或复位的存储器映射：	1
		0	正常的存储器映射。	
		1	复位存储器映射。静态存储器 $\overline{\text{EMC_CS1}}$ 被镜像到 $\overline{\text{EMC_CS0}}$ 和 $\overline{\text{EMC_DYCS0}}$ 上（上电复位值）。 当上电复位时， $\overline{\text{EMC_CS1}}$ 被镜像到 $\overline{\text{EMC_CS0}}$ 和 $\overline{\text{EMC_DYCS0}}$ 存储器区域。将 M 位清零使能对 $\overline{\text{EMC_CS0}}$ 和 $\overline{\text{EMC_DYCS0}}$ 存储器的存取。	
2	Low-power mode (L)		表明正常或低功率模式：	0
		0	正常模式（热复位值）	
		1	低功率模式。 进入低功率模式可降低存储器控制器的功耗。必要时对动态存储器进行刷新。通过将低功率模式位或上电复位清零，使存储器控制器返回正常的功能模式。 该位必须在 EMC 为空闲状态时进行修改。 ^[1]	
31:3	-		保留。读取值未定义，只写入 0。	无

[1] 在低功率或禁能状态下不能访问外部存储器。如果访问存储器，在 AHB 总线上会出现 ERROR 响应。在低功率和/或禁能状态下，可对 EMC 寄存器进行编程设定。

10.12.2 EMC 状态寄存器（EMCStatus—0x2009 C004）

只读的 EMCStatus 寄存器提供了 EMC 状态信息。[表 115](#) 给出了 EMCStatus 寄存器各位的分配。

表115. EMC 状态寄存器位描述（EMCStatus—0x2009 C008）

位	符号	值	描述	复位值
0	Busy (B)		该位用来确保存储器控制器完全进入了低功率或禁能模式，这一操作是通过确定存储器控制器是否繁忙实现的：	0
		0	EMC 是空闲的（热复位值）	
		1	EMC 正忙于执行存储器事务、命令、自刷新周期，或正处于自刷新模式（上电复位值）。	
1	Write buffer status(S)		该位使 EMC 完全进入低功率或禁能模式：	0
		0	写缓冲区为空（上电复位值）	
		1	写缓冲区包含数据。	
2	Self-refresh acknowledge (SA)		该位表明 EMC 的工作模式：	1
		0	正常模式	
		1	自刷新模式（上电复位值）。	
31:3	-		保留。从保留位读出的值未定义。	无

10.12.3 EMC 配置寄存器（EMCConfig—0x2009 C008）

EMCConfig 寄存器用于配置存储控制器的操作。建议对该寄存器的修改应在系统初始化期间，或在不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态，然后进入低功率或禁能模式可以保证这一点。访问这个寄存器需要一个等待状态。[表 116](#) 给出了 EMCConfig 寄存器各位的分配。

表116. EMC 配置寄存器位描述（EMCConfig—0x2009 C008）

位	符号	值	描述	复位值
0			字节序模式：	0
		0	小端字节模式（上电复位值）。	
		1	大端字节模式。	
			在上电复位处，字节序位的值为 0。在小端和大端模式进行切换前，所有数据必须输出至 EMC。	
7:1	-		保留。读取值未定义，只写入 0。	无
8			CCLK: CLKOUT 比率：	0
		0	1:1（上电复位值）	
		1	1:2（该选项对于 LPC178x/177x 不可用）	
			为确保 EMC 的正常操作，该位必须包含 0。	
31:9	-		保留。读取值未定义，只写入 0。	无

10.12.4 动态存储器控制寄存器（EMCDynamicControl—0x2009 C020）

EMCDynamicControl 寄存器控制着动态存储器的操作。在正常操作期间都可以改变控制位。[表 117](#) 给出了 EMCDynamicControl 寄存器各位的分配。

表117. 动态控制寄存器位描述 (EMCDynamicControl-0x2009 C020)

位	符号	值	描述	复位值
0	Dynamic memory clock enable (CE)	0	空闲设备的时钟使能被禁止，以节省功率（上电复位值）。	0
		1	所有时钟使能持续受到高电平的驱动 ^[1]	
1	Dynamic memory clock control (CS)	0	当所有 SDRAM 处于空闲且处于自刷新的模式时，CLKOUT 停止。	1
		1	CLKOUT 持续运行（上电复位值） 当时钟控制为低电平时，输出时钟 CLKOUT 在无 SDRAM 事务时停止。在自刷新模式下时钟也会停止。	
2	Self-refresh request, EMCSREFREQ (SR)	0	正常模式。	1
		1	进入自刷新模式（上电复位值）。 向该位写入 1，则可在软件控制下进入自刷新模式。将 0 写入该位，EMC 返回正常模式。 必须对 EMCStatus 寄存器中的自刷新确定位进行轮询，以发现 EMC 的当前工作模式。 ^[2]	
4:3	-		保留。读取值未定义，只写入 0.	无
5	Memory control clock (MMC)	0	CLKOUT 使能（上电复位值）。	0
		1	CLKOUT 禁能 ^[3] 。	
6	-		保留。读取值未定义，只写入 0。	无
8:7	SDRAM initialization (I)	00	发布 SDRAM NORMAL 操作命令（上电复位值）。	0
		01	发布 SDRAM MODE 命令。	
		10	发布 SDRAM PALL（全部预充电）命令。	
		11	发布 SDRAM NOP（不操作）命令。	
12:9	-		保留。读取值未定义，只写入 0。	无
13	Low-power SDRAM deep-sleep mode	0	正常操作（上电复位值）。	0
		1	进入深度睡眠模式	
31:14	-		保留。读取值未定义，只写入 0。	无

- [1] 在 SDRAM 初始化过程中，时钟使能必须为高电平。
- [2] 存储器控制器从上电复位退出，自刷新位为高电平。为进入正常功能模式，将该位设为低电平。
- [3] 若无 SDRAM 存储器事务，可执行 CLKOUT 禁能。若使能，该位可与动态存储器时钟控制字段共用。

注：将深度睡眠模式（DP）位、动态存储器时钟使能（CE）位，以及动态时钟控制（CS）位设置为 1，就可以进入深度睡眠模式。然后设备进入一种低功率模式，设备掉电，不再刷新。存储器中所有的数据均丢失。

10.12.5 动态存储器刷新定时器寄存器（EMCDynamicRefresh—0x2009 C024）

EMCDynamicRefresh 寄存器用于配置动态存储器的操作。建议对该寄存器的修改应在系统初始化期间，或不存在电流和未完成事务时进行。通过等待 EMC 至空闲，然后进入低功率或禁能模式可以保证这一点。不过，必要时，这些控制位也可以在正常操作期间改变。访问这个寄存器需要一个等待状态。

注：此寄存器用于所有 4 个动态存储器片选。因此必须写入所有片选的最差情况值。[表 118](#) 给出了 EMCDynamicRefresh 寄存器各个位的分配。

表118. 动态存储器刷新定时器寄存器位描述 (EMCDynamicRefresh—0x2009 C024)

位	符号	值	描述	复位值
10:0	Refresh timer (REFRESH)		表明 SDRAM 刷新周期之间 16 个 CCLK 的倍数。	0
		0x0	刷新禁能（上电复位值）。	
		0x1	0x7FF = n x16 =SDRAM 刷新周期中的 16n 个 CCLK。	
			例如： 0x1 = 1 x 16 = SDRAM 刷新周期中的 16 个 CCLK。 0x8 = 8 x 16 = SDRAM 刷新周期中的 128 个 CCLK。	
31:11	-		保留。读取值未定义，只写入 0。	无

例如，刷新周期为 16 μs，CCLK 频率为 50 MHz 时，必须在这个寄存器中编入下列值：

$$(16 \times 10^{-6} \times 50 \times 10^6)/16 = 50 \text{ 或 } 0x32$$

如果要求热复位期间自动刷新（置位 EMC_Reset_Disable 位），则必须调整自动刷新的时序，使得时钟速率在复位周期唤醒期间下降时仍保持足够的刷新率。在这个周期内，EMC（以及 LPC178x/177x 有时钟的所有其它部分）在 12 MHz 的 IRC 振荡器下运行。因此，如果热复位时要求自动刷新，则必须在刷新计算时将 12 MHz 作为 CCLK 速率。

注：刷新周期是平均分配的。但是，当根据存储控制器的状态，发出自动刷新命令时，可能会有少许的变化。

10.12.6 动态存储器读取配置寄存器 (EMCDynamicReadConfig—0x2009 C028)

EMCDynamicReadConfig 寄存器用于配置动态存储器的读取模式。此寄存器只能在系统初始化期间修改。访问该寄存器需要一个等待状态。

注：此寄存器用于所有 4 个动态存储器片选。因此必须写入所有片选的最差情况值。

表 119 给出了 EMCDynamicReadConfig 寄存器各位的分配。

表119. 动态存储器读取配置寄存器位描述 (EMCDynamicReadConfig—0x2009 C028)

位	符号	值	描述	复位值
1:0	Read data strategy (RD)	00	时钟输出延时策略，使用 CLKOUT（命令未延时，时钟输出延时）。上电复位值。	0x0
		01	命令延时策略，使用 EMCCLKDELAY（命令延时，时钟输出未延时）。	
		10	命令延时策略与一个时钟周期，使用 EMCCLKDELAY（命令延时，时钟输出未延时）。	
		11	命令延时策略与二个时钟周期，使用 EMCCLKDELAY（命令延时，时钟输出未延时）。	
31:2	-		保留。读取值未定义，只写入 0。	无

10.12.7 动态存储器预充电命令周期寄存器 (EMCDynamicTRP—0x2009 C030)

EMCDynamicTRP 寄存器能够编写预充电的命令周期——tRP。此寄存器只能在系统初始

化期间修改。此值（tRP）一般可以在 SDRAM 的数据手册中找到。访问该寄存器需要一个等待状态。

注：此寄存器用于所有 4 个动态存储器片选。因此必须写入所有片选的最差情况值。

[表 120](#) 给出了 EMCDynamicTRP 寄存器各位的分配。

表120. 动态存储器预充电命令周期寄存器位描述（EMCDynamicTRP—0x2009 C030）

位	符号	值	描述	复位值
3:0	Precharge command period (tRP)	0x0 -	n + 1 个时钟周期。EMCCLK 周期内存在延时。	0x0F
		0xE		
		0xF		
31:4	-	保留。读取值未定义，只写入 0。		无

10.12.8 动态存储器有效至预充电命令周期寄存器（EMCDynamicTRAS—0x2009 C034）

EMCDynamicTRAS 寄存器能够编写有效至预充电的命令周期——tRAS。建议对该寄存器的修改应在系统初始化期间，或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态，然后进入低功率或禁用模式可以保证这一点。此值（tRAS）一般可以在 SDRAM 的数据手册中找到。访问该寄存器需要一个等待状态。

注：此寄存器用于所有 4 个动态存储器片选。因此必须写入所有片选的最差情况值。

[表 121](#) 给出了 EMCDynamicTRAS 寄存器各位的分配。

表121. 动态存储器有效到预充电命令周期寄存器位描述（EMCDynamicTRAS – 0x2009 C034）

位	符号	值	描述	复位值
3:0	Active to precharge command period (tRAS)	0x0 -	n + 1 个时钟周期。EMCCLK 周期内存在延时。	0xF
		0xE		
		0xF		
31:4	-	保留。读取值未定义，只写入 0。		无

10.12.9 动态存储器自刷新退出时间寄存器（EMCDynamicSREX—0x2009 C038）

EMCDynamicTSREX 寄存器能够编写自刷新退出时间——tSREX。建议对该寄存器的修改应在系统初始化期间，或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态，然后进入低功率或禁用模式可以保证这一点。此值（tSREX）一般可以在 SDRAM 的数据手册中找到，对于没有此参数的器件，可以使用 tXSR。访问该寄存器需要一个等待状态。

注：此寄存器用于所有 4 个动态存储器片选。因此必须写入所有片选的最差情况值。

[表 122](#) 给出了 EMCDynamicTSREX 寄存器各位的分配。

表122. 动态存储器自刷新退出时间寄存器位描述 (EMCDynamicSREX—0x2009 C038)

位	符号	值	描述	复位值
3:0	Self-refresh exit time (tSREX)	0x0 -	n + 1 个时钟周期。CCLK 周期内存在延时。	0xF
		0xE		
		0xF	16 个时钟周期 (上电复位值)。	
31:4	-		保留。读取值未定义, 只写入 0。	无

10.12.10 动态存储器最后数据输出至有效时间寄存器 (EMCDynamicTAPR—0x2009 C03C)

EMCDynamicTAPR 寄存器能够编写最后数据输出到有效的命令时间——tAPR。建议对该寄存器的修改应在系统初始化期间, 或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态, 然后进入低功率或禁用模式可以保证这一点。此值 (tAPR) 一般可以在 SDRAM 的数据手册中找到。访问该寄存器需要一个等待状态。

注: 此寄存器用于所有 4 个动态存储器片选。因此必须写入所有片选的最差情况值。

[表123](#)给出了 EMCDynamicTAPR 寄存器各位的分配。

表123. 动态存储器最后数据输出至有效时间寄存器的位描述 (EMCDynamicTAPR— 0x2009 C03C)

位	符号	值	描述	复位值
3:0	Last-data-out to active command time (tAPR)	0x0 -	n + 1 个时钟周期。CCLK 周期内存在延时。	0xF
		0xE		
		0xF	16 个时钟周期 (上电复位值)。	
31:4	-		保留。读取值未定义, 只写入 0。	无

10.12.11 动态存储器数据输入至有效命令时间寄存器 (EMCDynamicTDAL—0x2009 C040)

EMCDynamicTDAL 寄存器编写数据输入到有效的命令时间——tDAL。建议对该寄存器的修改应在系统初始化期间, 或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态, 然后进入低功率或禁用模式可以保证这一点。此值 (tDAL 或 tAPW) 一般可以在 SDRAM 的数据手册中找到。访问该寄存器需要一个等待状态。

注: 此寄存器用于所有 4 个动态存储器片选。因此必须写入所有片选的最差情况值。

[表 124](#)给出了 EMCDynamicTDAL 寄存器各位的分配。

表124. 动态存储器数据输入至有效命令时间寄存器位描述 (EMCDynamicTDAL—0x2009 C040)

位	符号	值	描述	复位值
3:0	Data-in to active command (tDAL)	0x0 -	n 个时钟周期。CCLK 周期内存在延时。	0xF
		0xE		
		0xF	15 个时钟周期 (上电复位值)。	
31:4	-		保留。读取值未定义, 只写入 0。	无

10.12.12 动态存储器写入恢复时间寄存器（EMCDynamicTWR—0x2009 C044）

EMCDynamicTWR 寄存器能够编写写入恢复时间——tWR。建议对该寄存器的修改应在系统初始化期间，或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态，然后进入低功率或禁用模式可以保证这一点。此值(tWR、tDPL、tRWL, 或 tRDL)一般可以在 SDRAM 的数据手册中找到。访问该寄存器需要一个等待状态。

注：此寄存器用于所有 4 个动态存储器片选。因此必须写入所有片选的最差情况值。

表 125 给出了 EMCDynamicTWR 寄存器各位的分配。

表125. 动态存储器写入恢复时间寄存器（EMCDynamicTWR— 0x2009 C044）位描述

位	符号	值	描述	复位值
3:0	Write recovery time (tWR)	0x0 -	n + 1 个时钟周期。CCLK 周期内存在延时。	0xF
		0xE		
		0xF	16 个时钟周期（上电复位值）。	
31:4	-		保留。读取值未定义，只写入 0。	无

10.12.13 动态存储器有效至有效命令周期寄存器（EMCDynamicTRC—0x2009 C048）

EMCDynamicTRC 寄存器能够编写有效至有效命令周期——tRC。建议对该寄存器的修改应在系统初始化期间，或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态，然后进入低功率或禁用模式可以保证这一点。此值(tRC)一般可以在 SDRAM 的数据手册中找到。访问该寄存器需要一个等待状态。

注：此寄存器用于所有 4 个动态存储器片选。因此必须写入所有片选的最差情况值。

表 126 给出了 EMCDynamicTRC 寄存器各位的分配。

表126. 动态存储器有效至有效命令周期寄存器位描述（EMCDynamicTRC—0x2009 C048）

位	符号	值	描述	复位值
4:0	Active to active command period (tRC)	0x0 -	n + 1 个时钟周期。CCLK 周期内存在延时。	0x1F
		0x1E		
		0xF	32 个时钟周期（上电复位值）。	
31:5	-		保留。读取值未定义，只写入 0。	无

10.12.14 动态存储器自动刷新周期寄存器（EMCDynamictRFC—0x2009 C04C）

EMCDynamictRFC 寄存器能够编写自动刷新周期以及有效命令自动刷新周期——tRFC。建议对该寄存器的修改应在系统初始化期间，或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态，然后进入低功率或禁用模式可以保证这一点。此值（tRFC，有时为 tRC）一般可以在 SDRAM 的数据手册中找到。访问该寄存器需要一个等待状态。

注：此寄存器用于所有 4 个动态存储器片选。因此必须写入所有片选的最差情况值。

表 127 给出了 EMCDynamictRFC 寄存器各位的分配。

表127. 动态存储器自刷新周期寄存器（EMCDynamictRFC—0x2009 C04C）位描述

位	符号	值	描述	复位值
4:0	Auto-refresh period and auto-refresh to active command period (tRFC)	0x0 -	n + 1 个时钟周期。CCLK 周期内存在延时。	0x1F
		0x1E		
		0xF	32 个时钟周期（上电复位值）。	
31:5	-	保留。读取值未定义，只写入 0。		无

10.12.15 动态存储器退出自刷新寄存器（EMCDynamictXSR—0x2009 C050）

EMCDynamictXSR 寄存器能够编写从自刷新退出到有效的命令时间——tXSR。建议对该寄存器的修改应在系统初始化期间，或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态，然后进入低功率或禁用模式可以保证这一点。此值（tXSR）一般可以在 SDRAM 的数据手册中找到。访问该寄存器需要一个等待状态。

注：此寄存器用于所有 4 个动态存储器片选。因此必须写入所有片选的最差情况值。

表 128 给出了 EMCDynamictXSR 寄存器各位的分配。

表128. 动态存储器退出自刷新寄存器位描述（EMCDynamictXSR—0x2009 C050）

位	符号	值	描述	复位值
4:0	Exit self-refresh to active command time (tXSR)	0x0 -	n + 1 个时钟周期。CCLK 周期内存在延时。	0x1F
		0x1E		
		0xF		
31:5	-	保留。读取值未定义，只写入 0。		无

10.12.16 动态存储器有效组 A 至有效组 B 的时间寄存器
(EMCDynamicTRRD—0x2009 C054)

EMCDynamicTRRD 寄存器能够编写有效组 A 至有效组 B 的延时——tRRD。建议对该寄存器的修改应在系统初始化期间，或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态，然后进入低功率或禁用模式可以保证这一点。此值（tRRD）一般可以在 SDRAM 的数据手册中找到。访问该寄存器需要一个等待状态。

注：此寄存器用于所有 4 个动态存储器片选。因此必须写入所有片选的最差情况值。

表 129 给出了 EMCDynamicTRRD 寄存器各位的分配。

表129. 动态存储器有效组 A 至有效组 B 时间寄存器位描述 (EMCDynamicTRRD—0x2009 C054)

位	符号	值	描述	复位值
3:0	Active bank A to active bank B latency (tRRD)	0x0 -	n + 1 个时钟周期。CCLK 周期内存在延时。	0xF
		0xE		
		0xF	16 个时钟周期（上电复位值）。	
31:4	-		保留。读取值未定义，只写入 0。	无

10.12.17 动态存储器装载模式寄存器至有效命令时间寄存器
(EMCDynamicMRD—0x2009 C058)

EMCDynamicTMRD 寄存器能够编写装载模式寄存器到有效的命令时间——tMRD。建议对该寄存器的修改应在系统初始化期间，或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态，然后进入低功率或禁用模式可以保证这一点。此值（tMRD 或 tRSA）一般可以在 SDRAM 的数据手册中找到。访问该寄存器需要一个等待状态。

注：此寄存器用于所有 4 个动态存储器片选。因此必须写入所有片选的最差情况值。

表 130 给出了 EMCDynamicTMRD 寄存器各位的分配。

表130. 动态存储器装载模式寄存器至有效命令时间位描述 (EMCDynamicMRD—0x2009 C058)

位	符号	值	描述	复位值
3:0	Load mode register to active command time (tMRD)	0x0 -	n + 1 个时钟周期。CCLK 周期内存在延时。	0xF
		0xE		
		0xF	16 个时钟周期（上电复位值）。	
31:4	-		保留。读取值未定义，只写入 0。	无

10.12.18 静态存储器延长等待寄存器（EMCStaticExtendedWait—0x2009 C080）

EMCStaticConfig 寄存器中的 ExtendedWait（EW）位置位。建议对该寄存器的修改应在系统初始化期间，或不存在电流和未完成事务时进行。但是，如有必要，这些控制位也可以在正常操作期间修改。访问该寄存器需要一个等待状态。

表 131 给出了 EMCStaticExtendedWait 寄存器各位的分配。

表131. 静态存储器延长等待寄存器位描述（EMCStaticExtendedWait—0x2009 C080）

位	符号	值	描述	复位值
9:0	Extended wait time out (EXTENDEDWAIT)	0x0	16 个时钟周期（上电复位值）。CCLK 周期内存在延时。	
		0x1	(n+1) x16 个时钟周期。	
31:10	-		保留。读取值未定义，只写入 0。	无

例如，对于刷新周期为 16μs，CCLK 频率为 50MHz 的静态存储器读写传输时间，必须在这个寄存器中编入下列值：(16 x 10⁻⁶ x 50 x 10⁶)/16 - 1 = 49。

10.12.19 动态存储器配置寄存器（EMCDynamicConfig0~3—0x2009 C100, 120, 140, 160）

EMCDynamicConfig0~3 寄存器能够编写相关动态存储器片选的配置信息。这些寄存器一般只在系统初始化期间修改。访问该寄存器需要一个等待状态。

表 132 给出了 EMCDynamicConfig0~3 寄存器各位的分配。

表132. 动态存储器配置寄存器位描述（EMCDynamicConfig0~3—0x2009 C100, 0x2009 C120,0x2009 C140, 0x2009 C160）

位	符号	值	描述	复位值
2:0	-		保留。读取值未定义，只写入 0。	无
4:3	Memory device (MD)	00	SDRAM（上电复位值）。	00
		01	低功率 SDRAM。	
		1x	保留。	
6:5	-		保留。读取值未定义，只写入 0。	无
12:7	Address mapping (AM)		参见表 133。	0
		0	000000 =复位值 ^[1] 。	
13	-		保留。读取值未定义，只写入 0。	无
14	Address mapping (AM)		参见表 133。	0
		0	0 =复位值。	
18:15	-		保留。读取值未定义，只写入 0。	无
19	Buffer enable (B)	0	缓冲禁能，以读取该片选（上电复位值）。	0
		1	缓冲使能，以读取该片选 ^[2] 。	

位	符号	值	描述	复位值
20	Write protect (P)	0	写未受保护（上电复位值）。	0
		1	写保护。	
31:21	-	保留。读取值未定义，只写入 0。		无

- [1] SDRAM 列和行宽以及组的数量可通过地址映射自动计算。
- [2] 在 SDRAM 初始化过程中，缓冲必须禁能。在正常运行时，缓冲必须使能。

未出现在[表 133](#)中的地址映射已被保留。

表133. 地址映射

14	12	11:9	8:7	描述
16 位外部总线地址映射（行，组，列）				
0	0	000	00	16Mb (2Mx8)，2 组，行长度= 11，列长度 = 9
0	0	000	01	16Mb (1Mx16)，2 组，行长度 = 11，列长度 = 8
0	0	001	00	64Mb (8Mx8)，4 组，行长度 = 12，列长度 = 9
0	0	001	01	64Mb (4Mx16)，4 组，行长度 = 12，列长度 = 8
0	0	010	00	128Mb (16Mx8)，4 组，行长度 = 12，列长度 = 10
0	0	010	01	128Mb (8Mx16)，4 组，行长度= 12，列长度= 9
0	0	011	00	256Mb (32Mx8)，4 组，行长度=13，列长度=10
0	0	011	01	256Mb (16Mx16)，4 组，行长度=13，列长度=9
0	0	100	00	512Mb (64Mx8)，4 组，行长度=13，列长度=11
0	0	100	01	512Mb (32Mx16)，4 组，行长度=13，列长度=10
16 位外部总线地址映射（组，行，列）				
0	1	000	00	16Mb (2Mx8)，2 组，行长度=11，列长度=9
0	1	000	01	16Mb (1Mx16)，2 组，行长度=11，列长度=8
0	1	001	00	64Mb (8Mx8)，4 组，行长度=12，列长度= 9
0	1	001	01	64Mb (4Mx16)，4 组，行长度=12，列长度= 8
0	1	010	00	128Mb (16Mx8)，4 组，行长度= 12，列长度= 10
0	1	010	01	128Mb (8Mx16)，4 组，行长度=12，列长度= 9
0	1	011	00	256Mb (32Mx8)，4 组，行长度=13，列长度=10
0	1	011	01	256Mb (16Mx16)，4 组，行长度=13，列长度=9
0	1	100	00	512Mb (64Mx8)，4 组，行长度=13，列长度=11
0	1	100	01	512Mb (32Mx16)，4 组，行长度=13，列长度=10
32 位外部总线地址映射（行，组，列）				
1	0	000	00	16Mb (2Mx8)，2 组，行长度=11，列长度=9
1	0	000	01	16Mb (1Mx16)，2 组，行长度=11，列长度=8
1	0	001	00	64Mb (8Mx8)，4 组，行长度=12，列长度= 9
1	0	001	01	64Mb (4Mx16)，4 组，行长度=12，列长度= 8
1	0	001	10	64Mb (2Mx32)，4 组，行长度=11，列长度= 8
1	0	010	00	128Mb (16Mx8)，4 组，行长度= 12，列长度= 10
1	0	010	01	128Mb (8Mx16)，4 组，行长度=12，列长度= 9

14	12	11:9	8:7	描述
1	0	010	10	128Mb (4Mx32), 4 组, 行长度=12, 列长度= 8
1	0	011	00	256Mb (32Mx8), 4 组, 行长度=13, 列长度=10
1	0	011	01	256Mb (16Mx16), 4 组, 行长度=13, 列长度=9
1	0	011	10	256Mb (8Mx32), 4 组, 行长度=13, 列长度= 8
1	0	100	00	512Mb (64Mx8), 4 组, 行长度=13, 列长度=11
1	0	100	01	512Mb (32Mx16), 4 组, 行长度=13, 列长度=10
32 位外部总线地址映射 (组, 行, 列)				
1	1	000	00	16Mb (2Mx8), 2 组, 行长度=11, 列长度=9
1	1	000	01	16Mb (1Mx16), 2 组, 行长度=11, 列长度=8
1	1	001	00	64Mb (8Mx8), 4 组, 行长度=12, 列长度= 9
1	1	001	01	64Mb (4Mx16), 4 组, 行长度=12, 列长度= 8
1	1	001	10	64Mb (2Mx32), 4 组, 行长度=11, 列长度= 8
1	1	010	00	128Mb (16Mx8), 4 组, 行长度= 12, 列长度= 10
1	1	010	01	128Mb (8Mx16), 4 组, 行长度=12, 列长度= 9
1	1	010	10	128Mb (4Mx32), 4 组, 行长度=12, 列长度= 8
1	1	011	00	256Mb (32Mx8), 4 组, 行长度=13, 列长度=10
1	1	011	01	256Mb (16Mx16), 4 组, 行长度=13, 列长度=9
1	1	011	10	256Mb (8Mx32), 4 组, 行长度=13, 列长度= 8
1	1	100	00	512Mb (64Mx8), 4 组, 行长度=13, 列长度=11
1	1	100	01	512Mb (32Mx16), 4 组, 行长度=13, 列长度=10

一个片选可以连接到单只存储设备上, 但条件是片选的数据总线宽度与设备宽度一致。另外, 片选还可以连接到几只外接设备上。此时, 片选的数据总线宽是存储器件数据总线宽之和。

例如, 对于一个连接到下列设备上的片选:

- 对 32 位宽的存储设备, 选择 32 位宽的地址映射。
- 对 16 位宽的存储设备, 选择 16 位宽的地址映射。
- 对 4 x 8 位宽的存储设备, 选择 32 位宽的地址映射。
- 对 2 x 8 位宽的存储设备, 选择 16 位宽的地址映射。

SDRAM 组选择管脚 BA1 和 BA0 分别连接到地址线 A14 和 A13。

10.12.20 动态存储器 RAS & CAS 延时寄存器 (EMCDynamicRASCAS0~3—0x2009C104, 124, 144, 164)

EMCDynamicRasCas0~3 寄存器能够编写相关动态存储器的 RAS 与 CAS 延时。建议对该寄存器的修改应在系统初始化期间, 或不存在电流和未完成事务时进行。通过等待 EMC 至

空闲状态, 然后进入低功率或禁用模式可以保证这一点。访问该寄存器需要一个等待状态。

注: 这些寄存器中设置的值必须与 SDRAM 存储器件初始化时使用的值一致。

[表134](#)给出了 EMCDynamicRasCas0~3 寄存器各位的分配。

表134. 动态存储器 RAS & CAS 延迟寄存器位描述 (EMCDynamicRasCas0~3—0x2009 C104, 0x2009 C124, 0x2009 C144, 0x2009 C164)

位	符号	值	描述	复位值
1:0	RAS latency (active to read/write delay) (RAS)	00	保留。	11
		01	1 个 CCLK 周期。	
		10	2 个 CCLK 周期。	
		11	3 个 CCLK 周期 (上电复位值)。	
7:2	-		保留。读取值未定义, 只写入 0。	无
9:8	CAS latency (CAS)	00	保留。	11
		01	1 个 CCLK 周期。	
		10	2 个 CCLK 周期。	
		11	3 个 CCLK 周期 (上电复位值)。	
31:10	-		保留。读取值未定义, 只写入 0。	无

10.12.21 静态存储器配置寄存器(EMCStaticConfig0~3—0x2009 C200, 220, 240, 260)

EMCStaticConfig0~3 寄存器用于静态存储器的配置。建议对该寄存器的修改应在系统初始化期间, 或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态, 然后进入低功率或禁用模式可以保证这一点。访问该寄存器需要一个等待状态。

[表135](#)给出了 EMCStaticConfig0~3 寄存器各位的分配。注意, 不支持存储设备的同步突发模式。

表135. 静态存储器配置寄存器位描述 (EMCStaticConfig0~3—0x2009 C200, 0x2009 C220, 0x2009 C240, 0x2009 C260)

位	符号	值	描述	复位值
1:0	Memory width (MW)	00	8 位 (上电复位值)。	0
		01	16 位。	
		10	32 位。	
		11	保留。	
2	-		保留。读取值未定义, 只写入 0。	无
3	Page mode (PM)		在页读模式下, EMC 突发为 4 个外部访问。因此, 支持具有异步页读模式突发 4 个或更多设备。不支持具有异步页读模式突发 2 个设备, 且必须正常地访问此类设备。	
		0	禁能 (上电复位值)。	

位	符号	值	描述	复位值
		1	异步页读模式使能（页长为 4）。	
5:4	-		保留。读取值未定义，只写入 0。	无
6	Chip select polarity (PC)		上电复位上的片选极性值为 0	0
		0	低电平有效片选。	
		1	高电平有效片选。	
7	Byte lane state (PB)		字节通道状态位 PB 使不同类型的存储器连接。对于采用字节宽度的静态存储器，来自 EMC 的 $\overline{\text{BLS3:0}}$ 信号通常被连接至 $\overline{\text{WE}}$ （写使能）。	0
			在这种情况下，为执行读取操作，所有的 $\overline{\text{BLS3:0}}$ 位必须为高电平。这就意味着，字节通道状态（PB）位必须为低电平。	
			16 位宽的静态存储器设备通常使 $\overline{\text{BLS3:0}}$ 信号连接至静态存储器中的 UBn 和 LBn （高字节和低字节）信号。在这种情况下，向特殊字节执行写操作必须将适当的 UBn 或 LBn 信号设为低电平。为执行读取操作，所有 UB 和 LB 信号必须设置为低电平，以驱动总线。在这种情况下，字节通道状态（PB）位必须为高电平。	
		0	为读取， $\overline{\text{BLS3:0}}$ 中的所有位为高电平。为写入， $\overline{\text{BLS3:0}}$ 中各自有效位为低电平（上电复位值）。	
8	Extended wait (EW)	1	为读取， $\overline{\text{BLS3:0}}$ 中各自有效位为低电平。为写入， $\overline{\text{BLS3:0}}$ 中各自有效位为低电平。	0
			延长等待（EW）使用 <code>EMCStaticExtendedWait</code> 寄存器为读写传输计时，而非使用 <code>EMCStaticWaitRd</code> 和 <code>EMCStaticWaitWr</code> 。这使能了超长的事务处理。 [1]	
		0	延长等待禁能（上电复位值）。	
		1	延长等待使能。	
			保留。读取值未定义，只写入 0。	
18:9	-		保留。读取值未定义，只写入 0。	无
19	Buffer enable (B) [2]	0	缓冲禁能（上电复位值）。	0
		1	缓冲使能。	
20	Write protect (P)	0	写未受保护（上电复位值）。	0
		1	写保护。	
31:21	-		保留。读取值未定义，只写入 0。	无

[1] 不得同时选择延长等待和页读模式。
[2] 即使缓冲区使能位被清零，EMC 也可以执行突发读取。

10.12.22 静态存储器写使能延迟寄存器（EMCStaticWaitWen0~3—0x2009 C204, 224, 244, 264）

EMCStaticWaitWen0~3 寄存器能够编写从片选到写使能的延迟。建议对该寄存器的修改应

在系统初始化期间,或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态,然后进入低功率或禁用模式可以保证这一点。访问该寄存器需要一个等待状态。

[表 136](#) 给出了 EMCStaticWaitWen0~3 寄存器各位的分配。

表136. 静态存储器写使能延迟寄存器位描述 (EMCStaticWaitWen0~3—0x2009 C204,0x2009 C224,0x2009 C244, 0x2009 C264)

位	符号	值	描述	复位值
3:0	Wait write enable (WAITWEN)		从片选设置到写使能的延时。	0x0
		0x0	片选设置与写使能的 1 个 CCLK 周期延时 (上电复位值)。	
		0x1-0xF (n + 1) CCLK 周期延时。延时为(WAITWEN +1) x tCCLK。		
31:4	-		保留。读取值未定义,只写入 0。	无

10.12.23 静态存储器输出使能延迟寄存器 (EMCStaticWaitOen0~3—0x2009 C208, 228, 248, 268)

EMCStaticWaitOen0~3 寄存器能够编写来自片选或地址变化 (两者中较晚的) 到输出使能的延时。建议对该寄存器的修改应在系统初始化期间,或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态,然后进入低功率或禁用模式可以保证这一点。访问该寄存器需要一个等待状态。

[表 137](#) 给出了 EMCStaticWaitOen0~3 寄存器各位的分配。

表137. 静态存储器输出使能延迟寄存器位描述 (EMCStaticWaitOen0~3—0x2009 C208, 0x2009 C228,0x2009 C248, 0x2009 C268)

位	符号	值	描述	复位值
3:0	Wait output enable (WAITOEN)		从片选设置到输出使能的延时。	0x0
		0x0	无延时 (上电复位值)。	
		0x1 - n	周期延时。延时为 WAITOEN x tCCLK。	
		0xF		
31:4	-		保留。读取值未定义,只写入 0。	无

10.12.24 静态存储器读取延迟寄存器 (EMCStaticWaitRd0~3—0x2009 C20C, 22C, 24C, 26C)

EMCStaticWaitRd0~3 寄存器能够编写从片选到读取的延迟。建议对该寄存器的修改应在系统初始化期间,或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态,然后进入低功率或禁用模式可以保证这一点。如果 EMCStaticConfig0~3 寄存器中的延长等待位使能,则该寄存器不使用。访问该寄存器需要一个等待状态。

[表 138](#) 给出了 EMCStaticWaitRd0~3 寄存器各位的分配。

表138. 静态存储器读取延迟寄存器位描述 (EMCStaticWaitRd0~3—0x2009 C20C, 0x2009 C22C,0x2009 C24C, 0x2009 C26C)

位	符号	值	描述	复位值
4:0	Non-page mode read wait states or asynchronous page mode read first access wait state (WAITRD)		非页读模式读取或异步页读模式读取，仅第一次读取：	0x1F
		0x0 - 0x1E	(n + 1)读取 CCLK 周期。对于非顺序性读取而言，等待状态时间为 (WAITRD + 1) x tCCLK。	
		0x1F	32 个读取 CCLK 周期（上电复位值）。	
31:5	-		保留。读取值未定义，只写入 0。	无

10.12.25 静态存储器页读模式读取延迟寄存器 (EMCStaticwaitPage0~3—0x2009 C210, 230, 250, 270)

EMCStaticWaitPage0~3 能够编写异步页读模式的顺序存取延迟。建议对该寄存器的修改应在系统初始化期间，或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态，然后进入低功率或禁用模式可以保证这一点。访问该寄存器需要一个等待状态。

表 139 给出了 EMCStaticWaitPage0~3 寄存器各位的分配。

表139. 静态存储器页读模式读取延迟寄存器 0~3 位描述 (EMCStaticWaitPage0~3—0x2009 C210,0x2009 C230, 0x2009 C250, 0x2009 C270)

位	符号	值	描述	复位值
4:0	Asynchronous page mode read after the first read wait states (WAITPAGE)		第一次读取后异步页读模式读取的等待状态数量：	0x1F
		0x0 - 0x1E	(n+ 1) CCLK 周期读取时间。对于针对顺序性读取的异步页读模式读取而言，第一次读取后页读模式读取的等待状态时间为 (WAITPAGE + 1) x tCCLK。	
		0x1F	32 个 CCLK 周期读取时间（上电复位值）。	
31:5	-		保留。读取值未定义，只写入 0。	无

10.12.26 静态存储器写入延迟寄存器(EMCStaticWaitWr0~3—0x2009 C214, 234, 254, 274)

EMCStaticWaitWr0~3 寄存器能够编写从片选到写入的延迟。建议对该寄存器的修改应在系统初始化期间，或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态，然后进入低功率或禁用模式可以保证这一点。如果在 EMCStaticConfig 寄存器中使能了延长等待(EW)位，则不使用该寄存器。访问该寄存器需要一个等待状态。

[表 140](#) 给出了 EMCStaticWaitWr0~3 寄存器各位的分配。

表140. 静态存储器写延迟寄存器 0~3 位描述(EMCStaticWaitWr—0x2009 C214, 0x2009 C234,0x2009 C254, 0x2009 C274)

位	符号	值	描述	复位值
4:0	Write wait states (WAITWR)		第一次读取后写入的 SRAM 等待状态时间:	0x1F
		0x0 -	(n + 2) CCLK 周期读取时间。第一次读取后写入的等待状态时间为	
		0x1E	WAITWR (n + 2) x tCCLK。	
		0x1F	33 CCLK 周期写入时间（上电复位值）。	
31:5	-		保留。读取值未定义，只写入 0。	无

10.12.27 静态存储器周转延迟寄存器（EMCStaticWaitTurn0~3—0x2009 C218, 238, 258, 278）

EMCStaticWaitTurn0~3 寄存器能够编写总线的周转周期数。建议对该寄存器的修改应在系统初始化期间，或不存在电流和未完成事务时进行。通过等待 EMC 至空闲状态，然后进入低功率或禁用模式可以保证这一点。访问该寄存器需要一个等待状态。

[表 141](#) 给出了 EMCStaticWaitTurn0~3 寄存器各位的分配。

表141. 静态存储器周转延迟寄存器 0~3 位描述（EMCStaticWaitTurn0~3—0x2009 C218,0x2009 C238, 0x2009 C258, 0x2009 C278）

位	符号	值	描述	复位值
3:0	Bus turnaround cycles (WAITTURN)	0x0	保留，不使用该值。	0xF
		0x1 -	(n + 1)个 CCLK 周转周期。总线周转时间为(WAITTURN + 1) x	
		0xE	tCCLK。	
		0xF	16 个 CCLK 周转周期（上电复位值）。	
31:4	-		保留。读取值未定义，只写入 0。	无

为防止外部存储器数据总线上的总线竞用，WAITTURN 域控制着静态存储器读、写操作之间增加的总线周转周期数。WAITTURN 还控制着静态存储器与动态存储器存取之间的周转周期数。

10.12.28 延迟控制寄存器（EMCDLYCTL—0x400F C1DC）

EMCDLYCTL 寄存器控制着片上可编程延迟，可用于精确调整外部 SDRAM 存储器的时序。延迟能以约 250ps 的增量配置，最高可达约 7.75ns。有关可编程延迟的概述见 [10.5.6](#) 节。[图 14](#)显示了可编程延迟的详细连接。[表 142](#)显示了 EMCDLYCTL 中各位的分配。

图14. EMC 可编程延迟

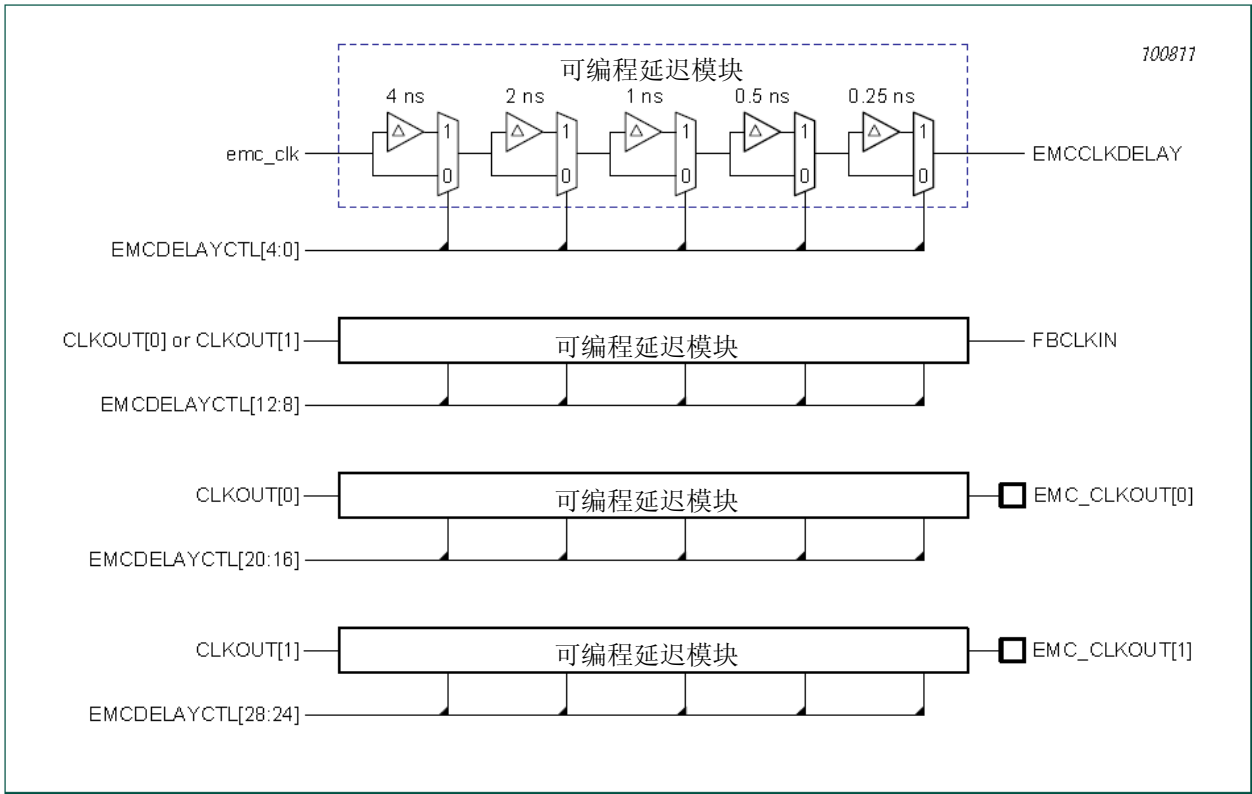


表142. 延迟控制寄存器位描述 (EMCDLYCTL—0x400F C1DC)

位	符号	描述	复位值
4:0	CMDDLY	命令延迟模式下 EMC 输出的可编程延迟值。参见 10.12.6 节。延迟时间约为 (CMDDLY+1) x 250ps。	0x10
7:5	-	保留。读取值未定义，只写入 0。	无
12:8	FBCLKDLY	控制输入数据采样的反馈时钟的可编程延迟值。参见 10.5.3 节。延迟时间约为 (FBCLKDLY+1) x 250ps。	0x02
15:13	-	保留。读取值未定义，只写入 0。	无
20:16	CLKOUT0DLY	CLKOUT0 输出的可编程延迟值。这通常用于时钟延迟模式。参见 10.12.6 节。延迟时间约为 (CLKOUT0DLY+1) x 250ps。	0
23:21	-	保留。读取值未定义，只写入 0。	无
28:24	CLKOUT1DLY	CLKOUT1 输出的可编程延迟值。这通常用于时钟延迟模式。参见 10.12.6 节。延迟时间约为 (CLKOUT1DLY+1) x 250ps。	0
31:29	-	保留。读取值未定义，只写入 0。	无

10.12.29 EMC 校准寄存器 (EMCCAL—0x400F C1E0)

EMCCAL 寄存器能够实时提供 EMC 可编程延迟的值，从而校准这些延迟。可以对应用中

使用的延迟设置做校准，以补偿不同器件之间的固有差异，以及环境条件的变化。下面的图 15 给出了延迟校准电路。表 143 显示了 EMCCAL 各位的分配。

图15. EMC 延迟校验

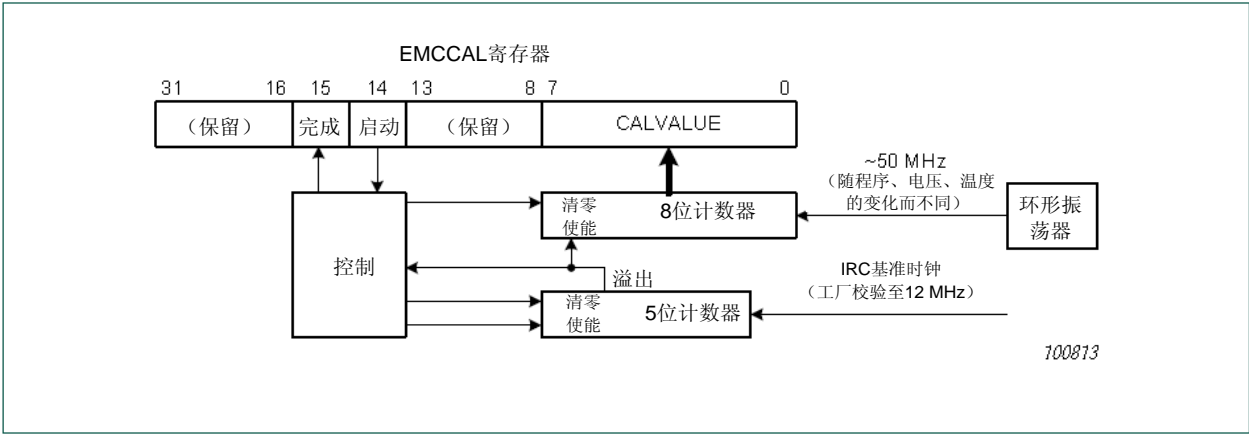


表143. EMC 校准寄存器位描述（EMCCAL—0x400F C1E0）

位	符号	描述	复位值
7:0	CALVALUE	返回约 50MHz 环形振荡器的计数，这发生在 IRC 振荡器的 32 个时钟。这显示了处理变化、内部稳压器和环境温度的复合作用。	0
13:8	-	保留。读取值未定义，只写入 0。	无
14	START	EMC 校准计数器的开始控制位。将 1 写入该位开始测量过程。当测量完成时，该位自动清零。	0
15	DONE	测量完成标志。当校准测量完成时，该位置位。当 START 位置位时，该位自动清零。	0
31:16	-	保留。读取值未定义，只写入 0。	无

校准可编程延迟的步骤:

1. 向 EMCCAL 寄存器的 START 位写入 1。
2. 等待，直到同一寄存器的 DONE 位变为 1。在这期间可进行其他操作，校准需要 12MHz IRC 时钟的 32 个时钟，或大约 2.7μs。
3. 从 EMCCAL 寄存器中读取底部 8 位的校准值。在室温下，该值一般为 0×86。
4. 必要时，根据校准结果调整一个或多个可编程延迟。

通常，根据应用环境下周围条件改变的速度，校准步骤应周期性重复。

10.13外部存储器接口

外部存储器接口取决于组的宽度(通过相应 EMCStaticConfig 寄存器中的 MW 位,选择 32、16 或 8 位)。

如果一个存储器组被配置为 32 位宽，则地址线 A0 和 A1 可以用做非地址线。如果一个存

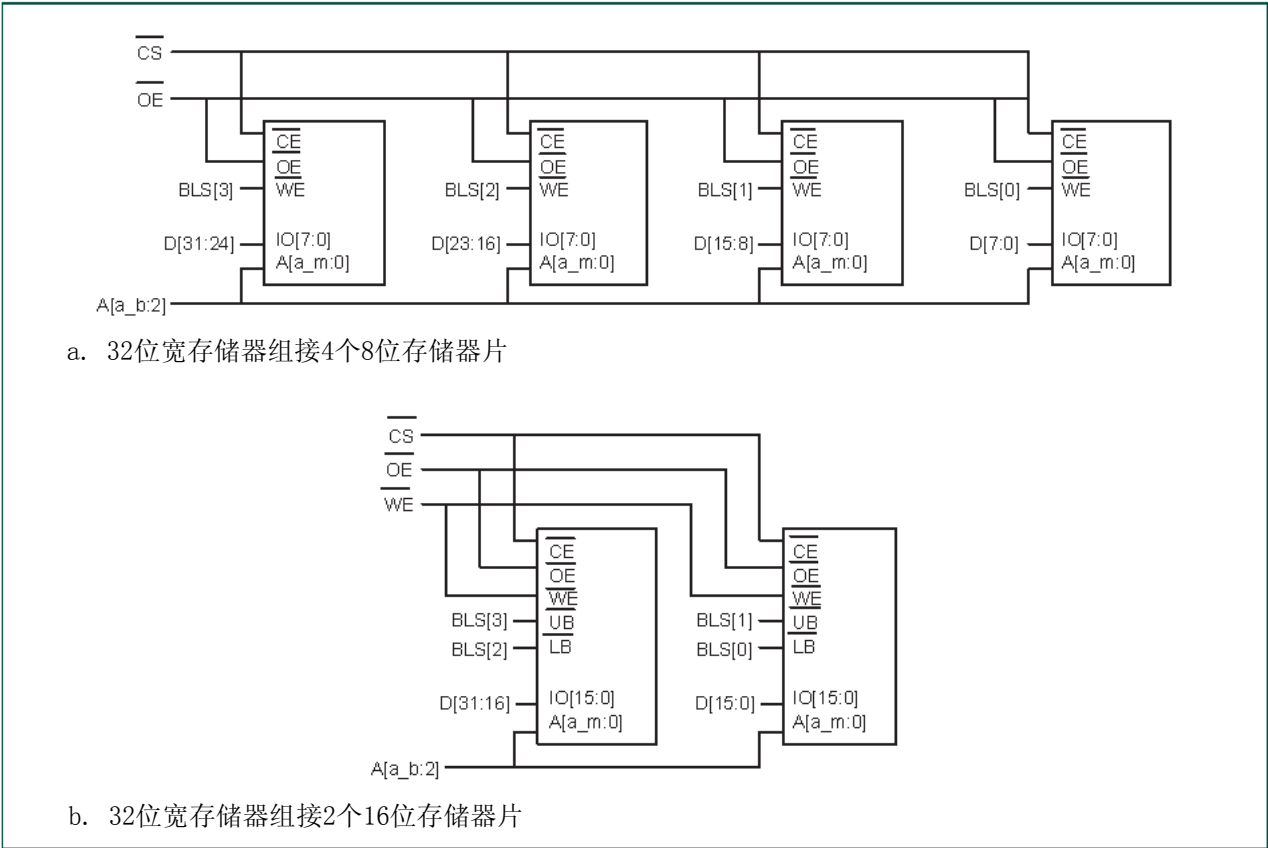
存储器通道被配置为 16 位宽，则不需要 A0。不过，8 位宽的存储器组要使用直到 A0 的全部地址线。使用 IOCON 寄存器就可以配置 A1 和/或 A0 线，使它们提供地址功能或非地址功能（见 8.4.1 节）。

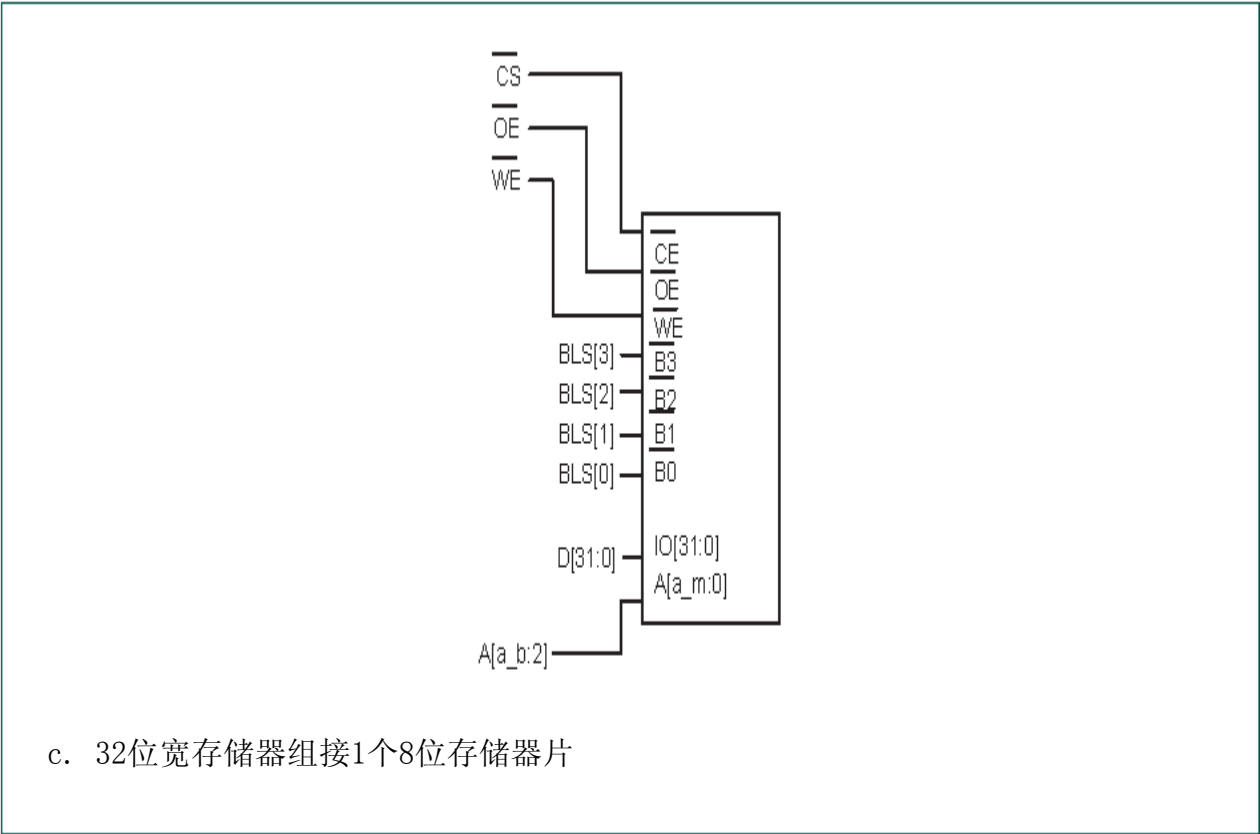
下图中的符号“a_b”指数据总线中的最高顺序地址线。符号“a_m”指外部存储器接口中所使用存储器芯片的最高顺序地址线。

如果将外部存储器用于封闭式（flashless）设备的外部引导存储器，有关与 EMC 的连接可参见 7.2 节。另可见 7.2 节“引导控制”。存储器组 1 和 2 的宽度由两个 BOOT1/0 管脚的设置所决定。

10.13.1 32 位宽存储器组连接

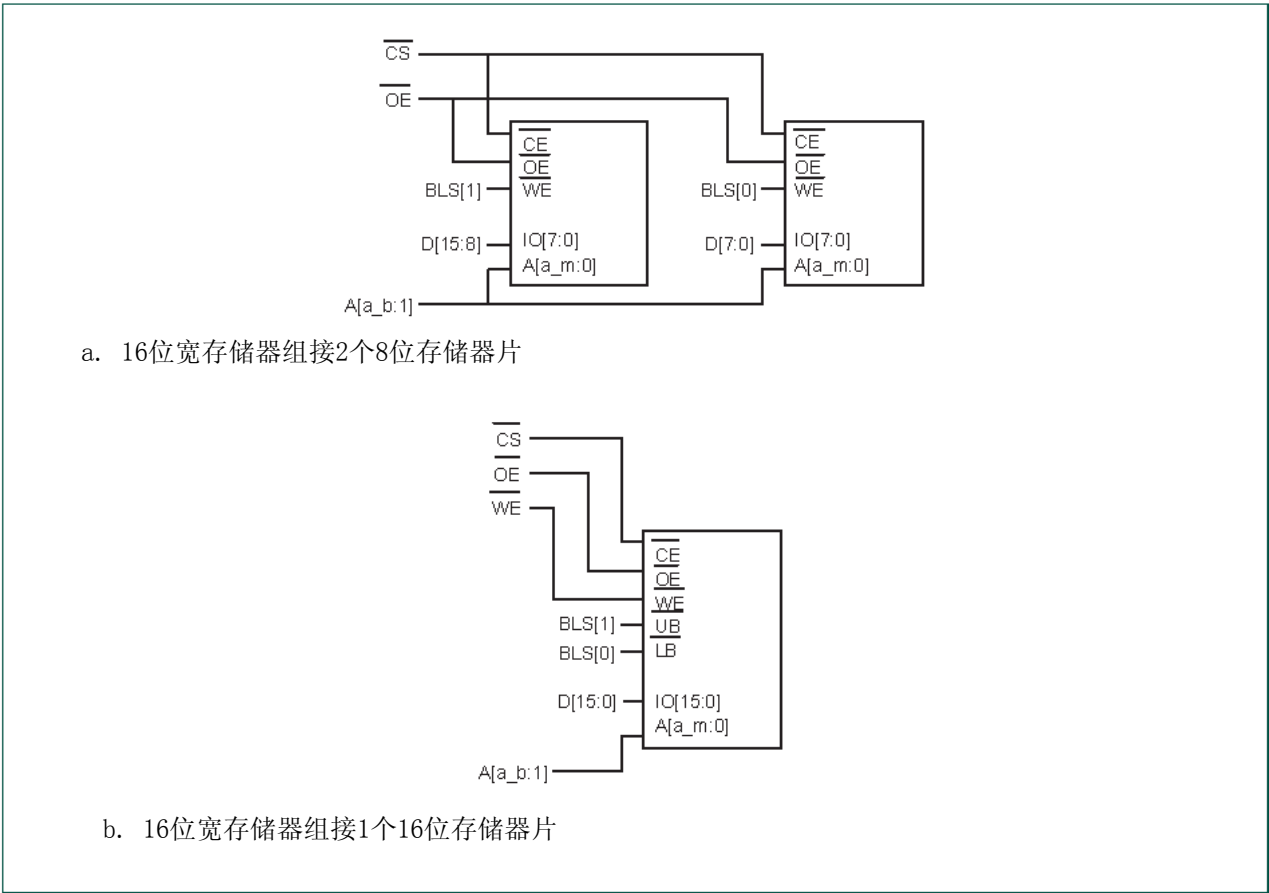
图16. 32 位存储器组的外部存储器接口（bits MW=10）





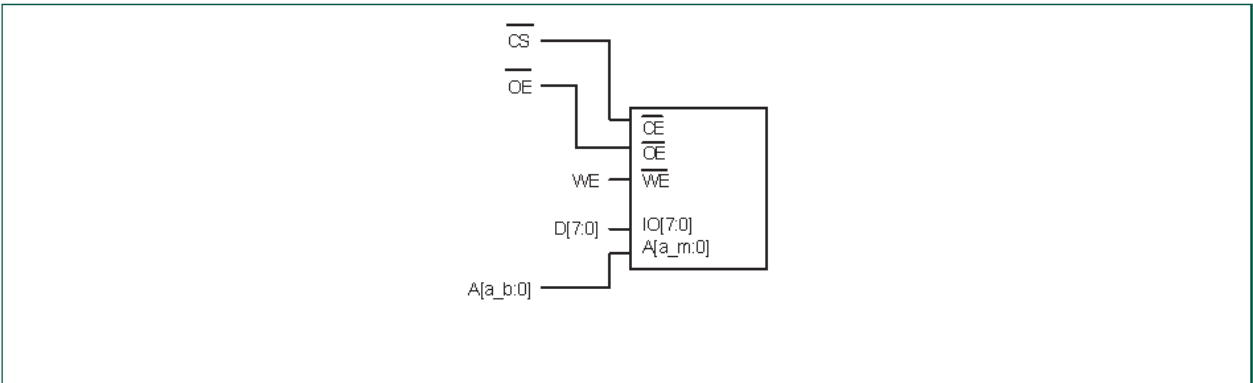
10.13.2 16 位宽存储器组连接

图17. 位存储器组的外部存储器接口 (bits MW = 01)



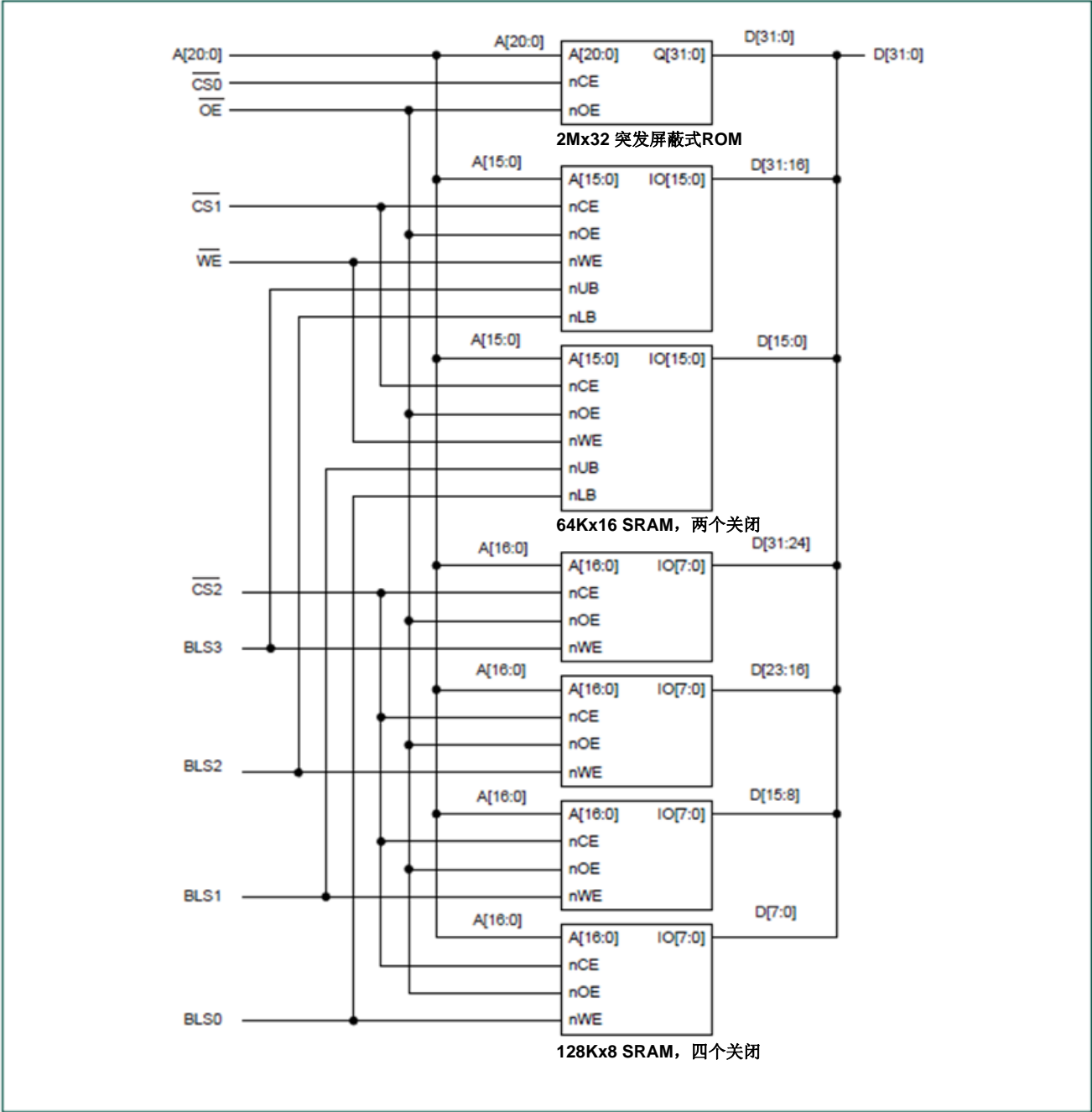
10.13.3 位宽存储器组连接

图18. 位存储器组的外部存储器接口 (bits MW = 00)



10.13.4 存储器配置实例

图19. 典型存储器配置图



11.1 基本配置

使用以下寄存器对以太网控制器进行配置：

1. 功率：在 PCONP 寄存器中（[表 37](#)）置位 PCENET。
注：复位后，以太网模块是禁能的（PCENET = 0）。
2. 时钟：见[表 30](#)。
3. 管脚：通过 IOCON 寄存器使能以太网管脚并选择其模式，见 [8.4.1](#) 节。
4. 唤醒：以太网端口上的活动可将微控制器从掉电模式下唤醒，见 [4.7.9](#) 节。
5. 中断：利用相应的中断置位使能寄存器使能 NVIC 中的中断。
6. 初始化：见 [11.17.2](#) 节。

11.2 简介

以太网模块包含了一个全功能的 10Mbps 或 100Mbps 以太网 MAC（媒体访问控制器），通过 DMA 硬件加速来优化性能。其特性包括：大量的控制寄存器组、半双工或全双工操作、流控制、控制帧、重发硬件加速、接收包过滤，以及 LAN 上的唤醒。采用分散—集中式（Scatter-Gather）DMA 进行自动帧发送与接收，减轻了 CPU 的工作量。

以太网模块是一个 AHB 主机，驱动 AHB 总线矩阵。通过矩阵，能访问到所有片上 RAM 存储器。以太网建议的 RAM 使用法是将一个 RAM 模块专门用来处理以太网通信。这个 RAM 只能由以太网和 CPU 访问，也可能包括 GPDMA，从而获取以太网功能的最大带宽。

以太网模块接口采用了 MII（媒体独立接口）或 RMII（简化的媒体独立接口）协议，以及片上 MIIM（媒体独立接口管理）串行总线，还有 MDIO（管理数据输入/输出）来实现与片外以太网 PHY 之间的连接。

表144. 以太网的缩写词与定义

缩写词	定义
AHB	先进的高性能总线
CRC	循环冗余校验
DMA	直接存储器访问
Double-word	64 位实体
FCS	帧校验序列（CRC）
Fragment	一个以太网帧或其中的一部分；一个以太网帧可以是一个或多个片段。
Frame	一个以太网帧由目标地址、源地址、长度/类型区、有效载荷以及帧校验序列组成。
Half-word	16 位实体
LAN	局域网
MAC	媒体访问控制子层
MII	媒体独立接口
MIIM	MII 管理
Octet	8 位数据实体，在 IEEE 802.3 中用作“字节”。
Packet	通过以太网传输的帧；一个包由导言、起始帧定界符和以太网帧组成。
PHY	以太网物理层
RMII	简化的 MII
Rx	接收
TCP/IP	传输控制协议/网际协议。以太网使用的最常规的高级协议。
Tx	发送
VLAN	虚拟局域网
WoL	LAN 上唤醒
Word	32 位实体

11.3 特性

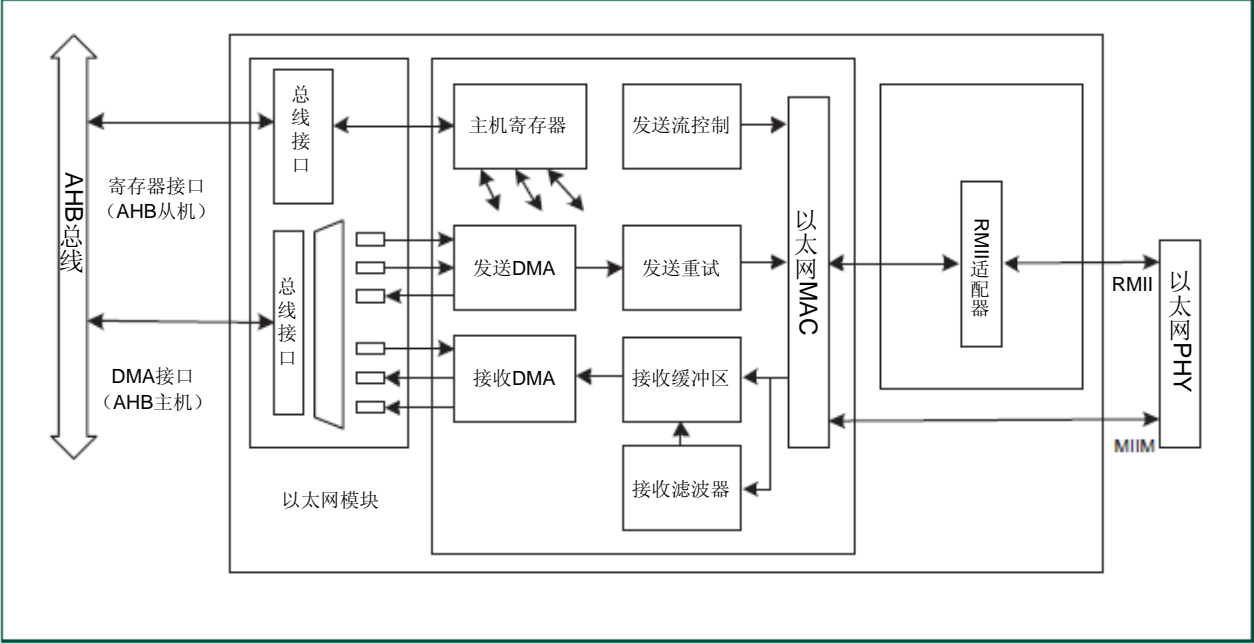
- 以太网标准支持：
 - 支持 10Mbps 或 100Mbps PHY 设备, 包括 10 Base-T、100 Base-TX、100 Base-FX, 以及 100 Base-T4。
 - 完全符合 IEEE 标准 802.3。
 - 完全符合 802.3x 的全双工流控制与半双工背压。
 - 灵活的发送与接收帧选项。
 - 支持 VLAN 帧。
- 存储器管理：
 - 映射至共享 SRAM 的独立的发送与接收缓冲区存储器。
 - 带分散/集中式 DMA 的 DMA 管理器以及帧描述符数组。
 - 通过缓冲和预取来优化的存储器通信。
- 增强的以太网特性：
 - 接收过滤。
 - 发送与接收时均支持多播帧与广播帧。
 - 发送操作可选择自动插入 FCS（CRC）。
 - 可选择在发送操作时自动帧填充。
 - 发送与接收均支持超长帧，可以采用任何长度的帧。

- 多种接收模式。
- 出现冲突时自动后退以及帧重发。
- 包括采用时钟切换的功率管理。
- 支持“LAN 上唤醒”的功率管理功能，以便将系统唤醒，该功能可使用接收滤波器或魔法帧检测滤波器实现。
- 物理接口：
 - 通过标准的 MII 或简化的 MII（RMII）接口连接外部 PHY 芯片。
 - 通过媒体独立管理（MIIM）接口，访问 PHY 寄存器。

11.4 结构与操作

图 20 显示了以太网模块的内部结构。

图20. 以太网方框图



以太网模块的框图包括：

- 主机寄存器模块，包括了软件视图中的寄存器，以及用于处理 AHB 对以太网模块访问的寄存器。主机寄存器与发送与接收数据通道以及 MAC 连接。
- 至 AHB 的 DMA 接口。它提供了一个 AHB 主机连接，使以太网模块能够访问片上 SRAM，实现描述符的读操作、状态的写操作，以及数据缓冲区的读写操作。
- 以太网 MAC，通过 MII 或 RMII 接口与片外 PHY 相连。
- 发送数据通道，包括：

- 发送 DMA 管理器，用于从存储器中读取描述符和数据，以及将状态写入存储器。
- 发送重试模块，处理以太网重试以及中止情况。
- 发送流量控制模块，可以插入以太网暂停帧。
- 接收数据通道，包括：
 - 接收 DMA 管理器，用于从存储器中读取描述符和数据，以及将数据与状态写入存储器。
 - 以太网 MAC，通过解析帧头中的部分信息来检测帧的类型。
 - 接收滤波器，采用不同的过滤机制，滤除特定的以太网帧。
 - 接收缓冲区，实现接收帧的延迟，这样滤波器就能滤掉特定帧，然后再将接收帧保存至存储器。

11.5 DMA 引擎功能

以太网模块的设计采用了 DMA 硬件加速，提供最佳性能。连接到 AHB 总线上独立的分散/集中 DMA 引擎可减轻 CPU 的数据传输负荷。

描述符保存在存储器中，它包含了有关进出以太网帧的片段的信息。一个片段可能是一个整帧，或是一个小得多的数据量。每个描述符都包括一个指向某个存储缓冲区的指针，该缓冲区包含了与片段有关的数据、片段缓冲区的大小，以及片段将如何被发送或接收的细节。

描述符存放在由以太网模块中的指针寄存器设定的存储器数组中。其它寄存器决定了数组的大小，并指向被 DMA 引擎使用的每个数组中的下一个描述符，以及将被以太网设备驱动使用的每个数组中的下一个描述符。

11.6 DMA 操作概述

DMA 引擎利用存储器中的“接收描述符数组”和“发送描述符数组”来管理收发数据，与描述符对应的存储器缓冲区可以保存一个完整的以太网帧或其中一部分。在发送以太网帧时，发送 DMA 引擎会利用“发送描述符”（一个或多个）将多个存储器片段中的数据“集中”起来，并将它们按顺序发送。在接收时，接收 DMA 引擎也按需要使用的“接收描述符”（一个或多个），将接收到的数据“分散”保存到多个存储器片段中。

描述符数组的基址寄存器，表示描述符数组入口数目的寄存器以及描述符数组输入/输出指针都包含在以太网模块中。描述符入口与所有发送和接收包数据均保存在存储器内，存储器不属于以太网模块的一部分。描述符入口用于指示相关数据帧在存储器中的保存位置，数据处理方式，以及每个以太网处理结果的状态。

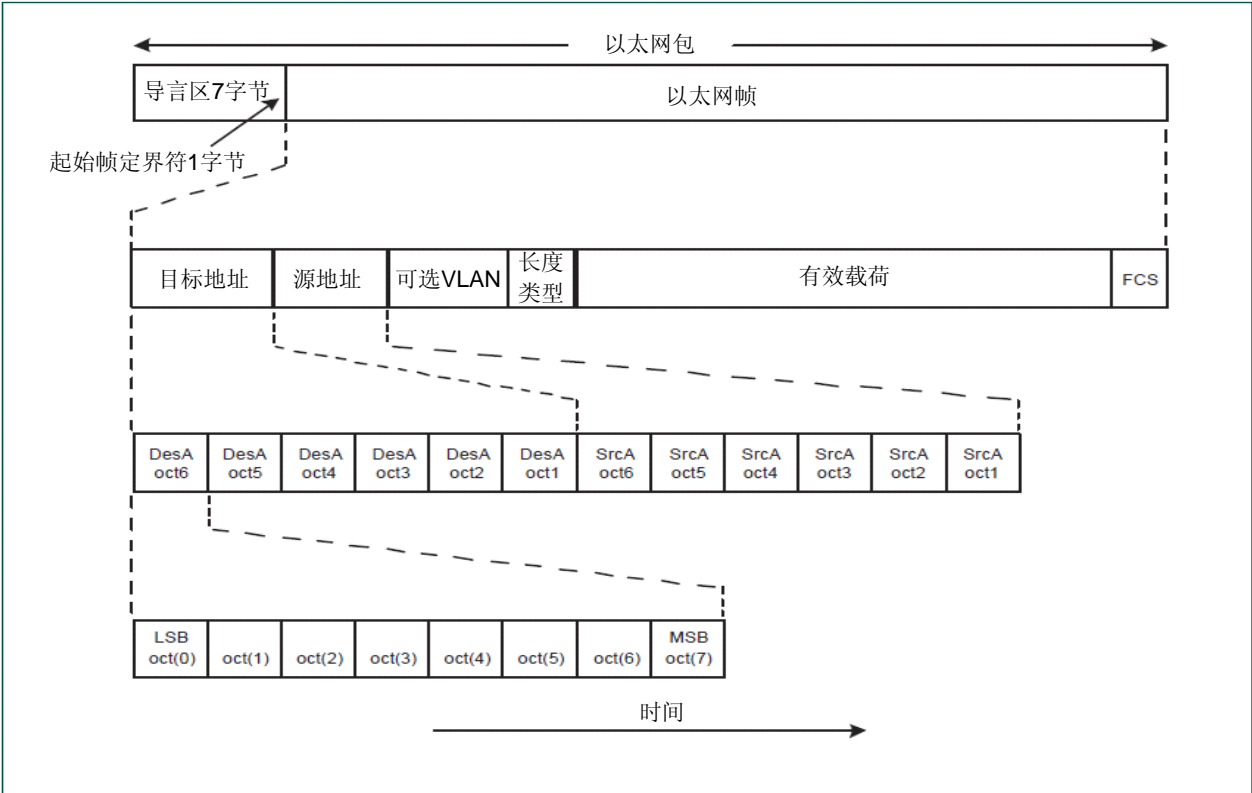
DMA 引擎的硬件用于控制如何将来自以太网 MAC 的数据保存在存储器中，保存与片段相关的状态，并针对输入数据让硬件接收指针向前移动。驱动软件必须处理所接收数据的位置，改变描述符的数据地址（以避免不必要的移动），并前移软件接收指针。两个指针在描述符数组中建立了一个环形队列，使 DMA 硬件与驱动软件都能知道哪个描述符（如果有）是可用的，包括描述符数组是为空还是为满。

同样，驱动软件必须为将由以太网 MAC 发送的数据设置指针，给出每个数据片段的指令，并针对输出数据让软件发送指针向前移动。DMA 引擎中的硬件读取这个信息，可能的情况下将数据发送到以太网 MAC 接口，更新状态，并前移硬件传送指针。

11.7 以太网包

图 21 显示了以太网包的不同区域。

图21. 以太网包字段



一个以太网包包括：一个导言，一个起始帧定界符和一个以太网帧。

以太网帧包括：目标地址、源地址、可选 VLAN 区、长度/类型区、有效载荷，以及帧校验序列。

每个地址包含 6 个字节，每个字节包含 8 个位。传输操作从最低有效位开始。

11.8 综述

11.8.1 分区

以太网模块与相关设备驱动软件提供了在 OSI 参考模型（见 IEEE 标准 802.3）中数据链路层的媒体访问控制（MAC）子层的功能。MAC 子层提供了向下一个更高的协议级，即 MAC

客户层，通常是逻辑链路控制子层，提供发送和接收信息帧的服务。设备驱动软件实现了与 MAC 客户层的接口。它在以太网模块中建立寄存器，维护指向存储器中的帧的描述符数组，并通过中断接收来自以太网模块的返回结果。当发送帧时，软件建立以太网帧中的一部分信息，方法是提供指向目标地址区域、源地址区域、长度/类型区域、MAC 客户数据区域，以及可选在帧检验序列字段中 CRC 的指针。最好应采用以太网内核的分散/集中功能将帧区域串联起来，以避免不必要的复制。硬件会加上“导言”和“起始帧定界符”，如有软件请求，还可选择加上 CRC。当接收信息包时，硬件会除去“导言”和“起始帧定界符”，将信息包剩余部分（即包括目标地址、源地址、长度/类型区域、MAC 客户数据，以及帧校验序列（FCS）的以太网帧）传递给设备驱动程序。

除了 MAC 以外，以太网模块还包含接收与发送 DMA 管理器，它控制着 MAC 与 AHB 接口之间的接收与发送数据流。帧信息的传送是通过主机存储器中的描述符数组，这样硬件就可以无需软件/CPU 的支持自行处理多个帧。帧可以包含多个片段，通过分散/集中 DMA 可以访问这些片段。DMA 管理器采用预取指和缓冲处理来优化存储器的带宽。

接收过滤模块用于对接收到的帧进行过滤，判别是否被寻址到这个以太网站，从而可以丢弃不符合要求的数据帧。Rx 滤波器包括一个完全地址滤波器和一个 Hash 滤波器。

以太网模块支持“LAN 上唤醒”功率管理功能，能在收到 LAN 上的“唤醒帧”时，将系统从某种掉电模式中（此时一些时钟被关闭）唤醒。“唤醒帧”通过“接收滤波器模块”或通过“魔法帧检测技术”识别。触发一个中断可产生系统唤醒。

“中断逻辑模块”用于发出中断和屏蔽中断，并跟踪中断的原因。中断模块向主机系统发送中断请求信号。用软件可以使能、清除和设置中断。

流量控制模块中实现了对 IEEE 802.3/条款 31 的流量控制功能。接收流的控制帧由 MAC 自动处理。传送流的控制帧由软件发起。在半双工模式下，流量控制模块通过只发送连续的“导言”产生“背压”，在发送操作时，流量控制由暂停中断实现，以避免 jabber 超过限制。

以太网模块同时拥有标准的媒体独立接口（MII）总线和简化的媒体独立接口（RMII），连接到一个外部以太网 PHY 芯片。通过命令寄存器中的 RMII 位，可以选择 MII 或 RMII 模式。标准的半字节宽度 MII 接口能用于与 PHY 芯片的低速数据连接，即 10Mbps 时 2.5MHz，或 100Mbps 时 25MHz。RMII 接口以低的管脚数实现了与 PHY 的双倍时钟数据连接。通过 MIIM 总线（通常操作频率为 2.5MHz）的串行管理连接，用 AHB 接口可访问 PHY 芯片中的寄存器。

11.8.2 PHY 设备实例

[表 145](#) 给出了一些兼容 PHY 设备的实例。

表145. PHY 设备实例

制造商	设备型号
Broadcom	BCM5221
ICS	ICS1893
Intel	LXT971A
LSI Logic	L80223、L80225、L80227
Micrel	KS8721
National	DP83847、DP83846、DP83843
SMSC	LAN83C185

11.9 管脚描述

[表 146](#) 是媒体独立接口（MII）用于连接外部 PHY 的信号，而[表 147](#) 则是简化的媒体独立接口（RMII）用于连接外部 PHY 的信号。

表146. 以太网 MII 的管脚描述

管脚名	类型	管脚描述
ENET_TX_EN	输出	发送数据使能，低电平有效。
ENET_TXD3:0	输出	发送数据，4 位。
ENET_TX_ER	输出	发送错误。
ENET_TX_CLK	输入	发送器时钟。
ENET_RX_DV	输入	接收数据有效。
ENET_RXD3:0	输入	接收数据，4 位。
ENET_RX_ER	输入	接收错误。
ENET_RX_CLK	输入	接收时钟。
ENET_COL	输入	冲突检测。
ENET_CRS	输入	载波侦听。

表147. 以太网 RMII 的管脚描述

管脚名	类型	管脚描述
ENET_TX_EN	输出	发送数据使能，低电平有效。
ENET_TXD1:0	输出	发送数据，2 位。
ENET_RXD1:0	输入	接收数据，2 位。
ENET_RX_ER	输入	接收错误。
ENET_CRS	输入	ENET_CRS_DV，载波侦听/数据有效。
ENET_RX_CLK	输入	ENET_REF_CLK，参考时钟。

[表 148](#) 所示为用于外部 PHY 的媒体独立接口管理（MIIM）的信号。

表148. 以太网 MIIM 的管脚描述

管脚名	类型	管脚描述
ENET_MDC	输出	MIIM 时钟。
ENET_MDIO	输入/输出	MI 数据输入和输出。

11.10 寄存器与软件接口

以太网模块的软件接口包括一个寄存器以及发送与接收描述符的格式定义。以下两节将讨论这两部分内容。

11.10.1 寄存器映射

[表 149](#) 列出了寄存器、寄存器地址以及其它基本信息。全部 AHB 地址空间需要 4KB。

在一个硬件复位或由命令寄存器中 **RegReset** 位进行软件复位之后，所有寄存器中的全部位都被复位为“0”，有特殊说明的除外。

有些寄存器中含有未使用的位，在通过 AHB 接口对这些位执行读操作时将返回“0”。对未使用的位执行写操作时，不会产生任何影响。

寄存器映射中包括在以太网 MAC 中的寄存器和内核周围的寄存器，内核周围的寄存器用于控制 DMA 传输、流控制和过滤。

对保留地址或保留位执行读操作时，所得的数据是不可预知的。而对保留地址或保留位执行写操作将没有影响。

读取只写的寄存器会在 AHB 接口上返回一个读错误。写入只读的寄存器会在 AHB 接口上返回一个写错误。

表149. 以太网寄存器描述

名称	描述	访问	复位值	地址	表
MAC 寄存器					
MAC1	MAC 配置寄存器 1	R/W	0x8000	0x2008 4000	表 150
MAC2	MAC 配置寄存器 2	R/W	0	0x2008 4004	表 151
IPGT	连续两包的内部包间隙寄存器	R/W	0	0x2008 4008	表 153
IPGR	非连续两包的内部包间隙寄存器	R/W	0	0x2008 400C	表 154
CLRT	冲突窗口/重试寄存器	R/W	0x370F	0x2008 4010	表 155
MAXF	最大帧寄存器	R/W	0x0600	0x2008 4014	表 156
SUPP	PHY 支持寄存器	R/W	0	0x2008 4018	表 157
TEST	测试寄存器	R/W	0	0x2008 401C	表 158
MCFG	MII Mgmt 配置寄存器	R/W	0	0x2008 4020	表 159
MCMD	MII Mgmt 命令寄存器	R/W	0	0x2008 4024	表 161
MADR	MII Mgmt 地址寄存器	R/W	0	0x2008 4028	表 162
MWTD	MII Mgmt 写数据寄存器	WO	0	0x2008 402C	表 163
MRDD	MII Mgmt 读数据寄存器	RO	0	0x2008 4030	表 164
MIND	MII Mgmt 指示寄存器	RO	0	0x2008 4034	表 165
SA0	站地址 0 寄存器	R/W	0	0x2008 4040	表 166
SA1	站地址 1 寄存器	R/W	0	0x2008 4044	表 167
SA2	站地址 2 寄存器	R/W	0	0x2008 4048	表 168
控制寄存器					
命令	命令寄存器	R/W	0	0x2008 4100	表 169

名称	描述	访问	复位值	地址	表
Status	状态寄存器。	RO	0	0x2008 4104	表 170
RxDescriptor	接收描述符基址寄存器。	R/W	0	0x2008 4108	表 171
RxStatus	接收状态基址寄存器。	R/W	0	0x2008 410C	表 172
RxDescriptorNumber	接收描述符数目寄存器。	R/W	0	0x2008 4110	表 173
RxProduceIndex	接收产生索引寄存器。	RO	0	0x2008 4114	表 174
RxConsumeIndex	接收消耗索引寄存器。	R/W	0	0x2008 4118	表 175
TxDescriptor	发送描述符基址寄存器。	R/W	0	0x2008 411C	表 176
TxStatus	发送状态基址寄存器。	R/W	0	0x2008 4120	表 177
TxDescriptorNumber	发送描述符数目寄存器。	R/W	0	0x2008 4124	表 178
TxProduceIndex	发送产生索引寄存器。	R/W	0	0x2008 4128	表 179
TxConsumeIndex	发送消耗索引寄存器。	RO	0	0x2008 412C	表 180
TSV0	发送状态向量 0 寄存器。	RO	0	0x2008 4158	表 181
TSV1	发送状态向量 1 寄存器。	RO	0	0x2008 415C	表 182
RSV	接收状态向量寄存器。	RO	0	0x2008 4160	表 183
FlowControlCounter	流量控制计数器寄存器。	R/W	0	0x2008 4170	表 184
FlowControlStatus	流量控制状态寄存器。	RO	0	0x2008 4174	表 185
接收过滤寄存器					
RxFilterCtrl	接收滤波器控制寄存器。	R/W	0	0x2008 4200	表 186
RxFilterWoLStatus	接收滤波器 WoL 状态寄存器。	RO	0	0x2008 4204	表 187
RxFilterWoLClear	接收滤波器 WoL 清零寄存器。	WO	0	0x2008 4208	表 188
HashFilterL	Hash 滤波器表 LSB 寄存器。	R/W	0	0x2008 4210	表 189
HashFilterH	Hash 滤波器表 MSB 寄存器。	R/W	0	0x2008 4214	表 190
模块控制寄存器					
IntStatus	中断状态寄存器。	RO	0	0x2008 4FE0	表 191
IntEnable	中断使能寄存器。	R/W	0	0x2008 4FE4	表 192
IntClear	中断清零寄存器。	WO	0	0x2008 4FE8	表 193
IntSet	中断置位寄存器。	WO	0	0x2008 4FEC	表 194
PowerDown	掉电寄存器。	R/W	0	0x2008 4FF4	表 195

表中的第 3 列是寄存器的访问属性：只读、只写、读/写。
 所有 AHB 寄存器的写操作（除访问中断寄存器外）都被记录（post），即 AHB 操作将在写数据真正提交给寄存器之前完成。而对中断寄存器的访问只能在将数据提交给寄存器之后通过接受写入的数据来完成。

11.11 以太网 MAC 寄存器定义

本节定义了以太网模块寄存器映射中各个寄存器的位。

11.11.1 MAC 配置寄存器 1（MAC1—0x2008 4000）

MAC 配置寄存器 1（MAC1）的地址为 0x2008 4000。各个位的定义见[表 150](#)。

表150. MAC 配置寄存器 1 位描述（MAC1—0x2008 4000）

位	符号	功能	复位值
0	RECEIVE ENABLE	将该位置位可允许对接收帧进行接收。MAC 在内部将该控制位与输入的接收信息流同步。	0
1	PASS ALL RECEIVE FRAMES	当该位使能（设置为“1”）时，MAC 将传递所有的帧信息，而不考虑类型（常规帧与控制帧）。当该位禁能时，MAC 不传递有效的控制帧。	0
2	RX FLOW CONTRO	当该位使能（设置为“1”）时，MAC 遵照接收到的 PAUSE 流控制帧采取行动。当该位禁能时，忽略收到的 PAUSE 流控制帧。	0
3	TX FLOW CONTROL	当该位使能（设置为“1”）时，允许发送 PAUSE 流控制帧。当该位禁能时，阻止流控制帧。	0
4	LOOPBACK	该位置位将导致 MAC 发送接口被回送到 MAC 接收接口。该位清零将执行正常操作。	0
7:5	-	未使用	0x0
8	RESET TX	该位置位将使发送功能逻辑进入复位状态。	0
9	RESET MCS / TX	该位置位将使 MAC 控制子层/发送逻辑复位。MCS 逻辑执行流控制。	0
10	RESET RX	该位置位将使以太网接收逻辑进入复位状态。	0
11	RESET MCS / RX	该位置位将使 MAC 控制子层/接收逻辑复位。MCS 逻辑执行流控制。	0x0
13:12	-	保留。读取值未定义，只写入 0。	0x0
14	SIMULATION RESET	该位置位将使发送功能中的随机数发生器复位。	0
15	SOFT RESET	该位置位将使 MAC 内除主机接口以外的所有模块进入复位状态。	1
31:16	-	保留。读取值未定义，只写入 0。	0x0

11.11.2 MAC 配置寄存器 2（MAC2—0x2008 4004）

MAC 配置寄存器 2（MAC2）的地址为 0x2008 4004。各个位的定义见[表 151](#)。

表151. MAC 配置寄存器 2 位描述（MAC2—0x2008 4004）

位	符号	功能	复位值
0	FULL-DUPLEX	当该位使能（设置为“1”）时，MAC 工作在全双工模式下。当该位禁能时，MAC 工作在半双工模式下。	0
1	FRAME LENGTH CHECKING	当该位使能（设置为“1”）时，将发送帧和接收帧的长度与长度/类型区域进行比较。如果长度/类型区域表示的是长度，则执行校验操作。对于每一个接收到的帧，不匹配的状态将在 StatusInfo 字中报告。	0
2	HUGE FRAME ENABLE	当该位使能（设置为“1”）时，可发送和接收任意长度的帧。	0
3	DELAYED CRC	如果 IEEE 802.3 帧的起始处包含专有的头信息，则该位可确定该信息的字节数。当该位为 1 时，添加 4 个字节的头信息（CRC 功能会忽略这几个字节）。当该位为 0 时，没有头信息。	0
4	CRC ENABLE	该位置位时将在每帧上添加 CRC，而不管是否需要。如果 PAD/CRC ENABLE 置位，则该位必须置位。如果提交给 MAC 的帧包含 CRC，则将该位清零。	0

位	符号	功能	复位值
5	PAD / ENABLE	CRC 该位置位使得 MAC 填充（pad）所有的短帧。如果提交给 MAC 的帧含有有效的长度，则将该位清零。 该位与 AUTO PAD ENABLE 和 VLAN PAD ENABLE 一起使用。填充功能的详细信息请见表 153。	0
6	VLAN ENABLE	PAD 该位置位可使 MAC 将所有的短帧填充到 64 字节并添加一个有效的 CRC。各种填充特性的详细信息请见表 153。 注：如果 PAD / CRC ENABLE 位清零，则忽略该位。	0
7	AUTO DETECT PAD ENABLE	该位置位可使 MAC 自动检测帧类型。通过将源地址之后的两个字节与 0x8100（VLAN 协议 ID）进行比较，可以将帧类型确定为被标记还是没有被标记，然后再进行填充。表 153 根据该寄存器的配置，提供了有关填充功能的描述。 注：如果 PAD / CRC ENABLE 位清零，则忽略该位	0
8	PURE PREAMBLE ENFORCEMENT	该位使能当（设置为“1”）时，MAC 将验证导言的内容，以确保包含 0x55 并且无误。导言有误的包会被丢弃。当该位禁能时，不对导言进行检验。	0
9	LONG PREAMBLE ENFORCEMENT	当该位使能（设置为“1”）时，MAC 只允许接收含有的导言字节小于 12 的包。当该位禁能时，该位使能（设置为“1”）时，MAC 按照标准，允许导言为任意长度。	0
11:10	-	保留。读取值未定义，只写入 0。	0x0
12	NO BACKOFF	当该位使能（设置为“1”）时，MAC 将在冲突之后立即重发，而不是使用标准中指定的二进制指数后退（Binary Exponential Backoff）算法。	0
13	BACK PRESSURE / NO BACKOFF	当该位使能（设置为“1”）时，MAC 在背压过程中偶然导致了一次冲突之后将立即重发，无需后退，从而减少进一步冲突的机会，确保发送包能够发送出去。	0
14	EXCESS DEFER	当该位使能（设置为“1”）时，MAC 将按照标准，服从载波侦听的结果。当该位禁能时，MAC 将在延迟超出限制时中止。	0
31:15	-	保留。读取值未定义，只写入 0。	0x0

表152. 填充操作

类型	自动检测填充使能 MAC2 [7]	VLAN 填充使能 MAC2 [6]	填充/CRC 使能 MAC2 [5]	动作
任意	x	x	0	不填充或不添加 CRC 校验
任意	0	0	1	填充到 60 字节，添加 CRC
任意	x	1	1	填充到 64 字节，添加 CRC
任意	1	0	1	如果没有被标记，则填充到 60 字节并添加 CRC。如果被标记 VLAN，则填充到 64 字节并添加 CRC。

11.11.3 连续两包的内部包间隔寄存器（IPGT—0x2008 4008）

连续两包（Back-to-Back）的内部包间隔寄存器（IPGT）的地址为 0x2008 4008。各个位的定义见[表 153](#)。

表153. 连续两包的内部包间隔寄存器位描述（IPGT—0x2008 4008）

位	符号	功能	复位值
6:0	BACK-TO-BACK INTER-PACKET-GAP	这是一个可编程的字段，表示在任何已发送的包的结尾与下一个包的开始之间的最小可能的时间间隔。在全双工模式中，该寄存器的值应该是所需的时间间隔减去 3。在半双工模式中，该寄存器的值应该是所需的时间间隔减去 6。在全双工模式中，建议的设置为 0x15（21d），它表示最小的 IPG 为 960ns（在 100Mbps 模式下）或 9.6μs（在 10Mbps 模式下）。在半双工模式中，建议的设置为 0x12（18d），它表示最小的 IPG 为 960ns（在 100Mbps 模式下）或 9.6μs（在 10Mbps 模式下）。	0x0
31:7	-	保留。读取值未定义，只写入 0。	0x0

11.11.4 非连续两包的内部间隔寄存器（IPGR—0x2008 400C）

非连续两包的内部间隔寄存器（IPGR）的地址为 0x2008 400C。各个位的定义见[表 154](#)。

表154. 非连续两包的内部包间隔寄存器位描述（IPGR—0x2008 400C）

位	符号	功能	复位值
6:0	NON-BACK-TO-BACK INTER-PACKET-GAP PART2	这是一个可编程的字段，表示非连续两包的内部包间隔。建议的值为 0x12（18d），它表示最小的 IPG 为 960ns（在 100Mbps 模式下）或 9.6μs（在 10Mbps 模式下）。	0x0
7	-	保留。读取值未定义，只应写入 0。	0x0
14:8	NON-BACK-TO-BACK INTER-PACKET-GAP PART1	这是一个可编程的字段，表示在 IEEE 802.3/4.2.3.2.1 “服从载波（Carrier Deference）”中涉及到的可选的 CarrierSense 窗口。如果在 IPGR1 时间段内检测到载波，则 MAC 认为载波存在。但如果载波是在 IPGR1 之后变为有效，则 MAC 继续计时 IPGR2 并发送，这样必然引起一次冲突，确保对媒体的访问失败。该字段的范围为 0x0~IPGR2。建议的值为 0xC（12d）。	0x0
31:15	-	保留。读取值未定义，只写入 0。	0x0

11.11.5 冲突窗口/重试寄存器（CLRT—0x2008 4010）

冲突窗口/重试寄存器（CLRT）的地址为 0x2008 4010。各个位的定义见[表 155](#)。

表155. 冲突窗口/重试寄存器位描述（CLRT—0x2008 4010）

位	符号	功能	复位值
3:0	RETRANSMISSI N MMAXIMUM	这是一个可编程的字段，表示在由于冲突过多而中止发送包之前，一次冲突之后尝试重新发送的次数。标准规定尝试的次数为 0xF（15d）。见 IEEE 802.3/4.2.3.2.5。	0xF
7:4	-	保留。读取值未定义，只写入 0。	0x0
13:8	COLLISION WINDOW	这是一个可编程的字段，表示在适当配置网络中发生冲突的时间槽（slot time）或冲突窗口。该字段的默认值为 0x37（55d），表示在导言和 SFD 之后有 56 个字节窗口。	0x37
31:14	-	保留。读取值未定义，只写入 0。	无

11.11.6 最大帧寄存器（MAXF—0x2008 4014）

最大帧寄存器（MAXF）的地址为 0x2008 4014。各个位的定义见[表 156](#)。

表156. 最大帧寄存器位描述（MAXF—0x2008 4014）

位	符号	功能	复位值
15:0	MAXIMUM FRAME LENGTH	该字段的复位值为 0x0600，它表示最大的接收帧为 1536 个字节。没有被标记的最大以太网帧为 1518 个字节。被标记的帧会加上 4 个字节，总共 1522 个字节。如果所需的最大长度限制比复位值小，则可对该 16 位字段进行编程。	0x0600
31:16	-	未使用	0x0

11.11.7 PHY 支持寄存器（SUPP—0x2008 4018）

PHY 支持寄存器（SUPP）的地址为 0x2008 4018。SUPP 寄存器用于对 RMII 接口进行附加控制。此寄存器的位定义见[表 157](#)。

表157. PHY 支持寄存器位描述（SUPP—0x2008 4018）

位	符号	功能	复位值
7:0	-	未使用	0x0
8	SPEED	该位配置简化的 MII 逻辑以用于当前的操作速率。当该位置位时，选择 100Mbps 模式。当该位清零时，选择 10Mbps 模式。	0
31:9	-	未使用	0x0

PHY 支持寄存器中未使用的位应保留为 0。

11.11.8 测试寄存器（TEST—0x2008 401C）

测试寄存器（TEST）的地址为 0x2008 401C。此寄存器的位定义见[表 158](#)。这些位仅用于测试目的。

表158. 测试寄存器位描述（TEST—0x2008 401C）

位	符号	功能	复位值
0	SHORTCUT PAUSE QUANTA	该位将有效的 PAUSE 量子从 64 字节时间减少到 1 字节时间。	0
1	TEST PAUSE	该位使 MAC 控制子层禁止传输，就好像接收到了暂停时间参数为非零的 PAUSE 接收控制帧。	0
2	TEST BACKPRESSURE	该位置位将使 MAC 在链路上产生背压。背压时将发送导言，唤起载波侦听。来自系统的发送包将在背压过程中发送出去。	0
31:3	-	未使用	0x0

11.11.9 MII Mgmt 配置寄存器（MCFG—0x2008 4020）

MII Mgmt 配置寄存器（MCFG）的地址为 0x2008 4020。此寄存器的位定义见[表 159](#)。

表159. MII Mgmt 配置寄存器位描述（MCFG—0x2008 4020）

位	符号	功能	复位值
0	SCAN INCREMENT	该位置位使得 MII 管理硬件越过 PHY 来执行读周期。当该位置位时，II 管理硬件通过 PHY ADDRESS[4:0]中设置的值从地址 1 执行读周期。	0
1	SUPPRESS PREAMBLE	该位置位使得 MII 管理硬件执行不带有 32 位导言的读/写周期。该位清零使周期可正常执行。有一部分 PHY 是禁止导言的。	0
5:2	CLOCK SELECT	该字段由时钟分频逻辑在创建 MII 管理时钟（MDC）时使用。IEEE 802.3u 规定该时钟不能超过 2.5 MHz。但是，有部分 PHY 支持高达 12.5 MHz 的时钟速率。AHB 总线时钟（HCLK）被特定的数值分频。有关该字段的值的定义请参见 表 160 。	0
14:6	-	未使用	0x0
15	RESET MII MGMT	该位将 MII 管理硬件复位。	0
31:16	-	未使用	0x0

表160. 时钟选择的编码

时钟选择	位 5	位 4	位 3	位 2	最大 AHB 时钟
主机时钟 4 分频	0	0	0	x	10
主机时钟 6 分频	0	0	1	0	15
主机时钟 8 分频	0	0	1	1	20
主机时钟 10 分频	0	1	0	0	25
主机时钟 14 分频	0	1	0	1	35
主机时钟 20 分频	0	1	1	0	50
主机时钟 28 分频	0	1	1	1	70
主机时钟 36 分频	1	0	0	0	80 ^[1]
主机时钟 40 分频	1	0	0	1	90 ^[1]
主机时钟 44 分频	1	0	1	0	100 ^[1]
主机时钟 48 分频	1	0	1	1	120 ^[1]
主机时钟 52 分频	1	1	0	0	130 ^[1]
主机时钟 56 分频	1	1	0	1	140 ^[1]
主机时钟 60 分频	1	1	1	0	150 ^[1]
主机时钟 64 分频	1	1	1	1	160 ^[1]

[1] 允许的最大 AHB 时钟速率不超过设备的最大 CPU 时钟速率。

11.11.10 MII Mgmt 命令寄存器（MCMD—0x2008 4024）

II Mgmt 命令寄存器（MCMD）的地址为 0x2008 4024。此寄存器的位定义见[表 161](#)。

表161. MII Mgmt 命令寄存器位描述（MCMD—0x2008 4024）

位	符号	功能	复位值
0	READ	该位促使 MII 管理硬件执行一次读周期。读取的数据在寄存器 MRDD（II Mgmt 读数据）中返回。	0
1	SCAN	该位促使 MII 管理硬件连续地执行读周期。该特性是非常有用的，例如可用于监控链路的失败。	0
31:2	-	未使用	0x0

11.11.11 MII Mgmt 地址寄存器（MADR—0x2008 4028）

II Mgmt 地址寄存器（MADR）的地址为 0x2008 4028。此寄存器的位定义见[表 162](#)。

表162. MII Mgmt 地址寄存器位描述（MADR—0x2008 4028）

位	符号	功能	复位值
4:0	REGISTER ADDRESS	该字段表示 Mgmt 周期的 5 位寄存器地址。最多可访问 32 个寄存器。	0x0
7:5	-	未使用	0x0
12:8	PHY ADDRESS	该字段表示 Mgmt 周期的 5 位 PHY 地址。最多可寻址 31 个 PHY（0 被保留）。	0x0
31:13	-	未使用	0x0

11.11.12 MII Mgmt 写数据寄存器（MWTD—0x2008 402C）

II Mgmt 写数据寄存器（MWTD）是一个只写寄存器，地址为 0x2008 402C。此寄存器的位定义见[表 163](#)。

表163. MII Mgmt 写数据寄存器位描述（MWTD—0x2008 402C）

位	符号	功能	复位值
15:0	WRITE DATA	当对该字段执行写操作时，II Mgmt 使用这 16 位数据以及在 II Mgmt 地址寄存器（MADR）0x0 中预先配置的 PHY 和寄存器地址来执行写周期。	0x0
31:16	-	未使用	0x0

11.11.13 MII Mgmt 读数据寄存器（MRDD—0x2008 4030）

MII Mgmt 读数据寄存器（MRDD）是一个只读寄存器，地址为 0x2008 4030。此寄存器的位定义见[表 164](#)。

表164. MII Mgmt 读数据寄存器（MRDD—0x2008 4030）位描述

位	符号	功能	复位值
15:0	READ DATA	在 MII Mgmt 的一个读周期之后，能从这个字段中读取 16 位数据。	0x0
31:16	-	未使用	0x0

11.11.14 MII Mgmt 指示寄存器（MIND—0x2008 4034）

MII Mgmt 指示寄存器（MIND）是一个只读寄存器，地址为 0x2008 4034。此寄存器的位定义见[表 165](#)。

表165. MII Mgmt 指示寄存器位描述（MIND—0x2008 4034）

位	符号	功能	复位值
0	BUSY	当该位返回 1 时，表示 MII Mgmt 当前正在执行 MII Mgmt 读或写周期。	0
1	SCANNING	当该位返回 1 时，表示扫描操作（连续的 MII Mgmt 读周期）正在进行。	0
2	NOT VALID	当该位返回 1 时，表示 MII Mgmt 读周期还没有完成，读数据也是无效的	0
3	MIILink Fail	当该位返回 1 时，表示出现 MII Mgmt 链接失败。	0
31:4	-	未使用	0x0

以下是两个通过 MII 管理控制器访问 PHY 的实例。

对于不使用扫描时执行 PHY 写操作：

1. 向 MCMD 写入 0；
2. 向 PHY 地址与寄存器地址写入 MADR；
3. 向 MWTD 写入数据；
4. 等待 MIND 中的繁忙位清零。

对于不使用扫描时执行 PHY 读操作：

1. 向 MCMD 写入 1；
2. 向 PHY 地址与寄存器地址写入 MADR；
3. 等待 MIND 中的繁忙位清零；
4. 向 MCMD 写入 0；
5. 从 MRDD 读数据。

11.11.15 站地址 0 寄存器（SA0—0x2008 4040）

站地址 0 寄存器（SA0）的地址为 0x2008 4040。此寄存器的位定义见[表 166](#)。

表166. 站地址寄存器位描述（SA0—0x2008 4040）

位	符号	功能	复位值
7:0	STATION ADDRESS, 第 2 个字节	该字段包含站地址的第 2 个字节。	0x0
15:8	STATION ADDRESS, 第 1 个字节	该字段包含站地址的第 1 个字节。	0x0
31:16	-	未使用	0x0

站地址用于完全的地址过滤，以及发送暂停控制帧。数据包中的字节顺序参见[图 21](#)。

11.11.16 站地址 1 寄存器（SA1—0x2008 4044）

站地址 1 寄存器（SA1）的地址为 0x2008 4044。此寄存器的位定义见[表 167](#)。

表167. 站地址寄存器位描述（SA1—0x2008 4044）

位	符号	功能	复位值
7:0	STATION ADDRESS, 第 4 个字节	该字段包含站地址的第 4 个字节。	0x0
15:8	STATION ADDRESS, 第 3 个字节	该字段包含站地址的第 3 个字节。	0x0
31:16	-	未使用	0x0

站地址用于完全的地址过滤，以及发送暂停控制帧。数据包中的字节顺序参见[图 21](#)。

11.11.17 站地址 2 寄存器（SA2—0x2008 4048）

站地址 2 寄存器（SA2）的地址为 0x2008 4048。此寄存器的位定义见[表 168](#)。

表168. 站地址寄存器位描述（SA2—0x2008 4048）

位	符号	功能	复位值
7:0	STATION ADDRESS, 第 6 个字节	该字段包含站地址的第 6 个字节。	0x0
15:8	STATION ADDRESS, 第 5 个字节	该字段包含站地址的第 5 个字节。	0x0
31:16	-	未使用	0x0

站地址用于完全的地址过滤，以及发送暂停控制帧。数据包中的字节顺序参见[图 21](#)。

11.12控制寄存器定义

11.12.1 命令寄存器（Command—0x2008 4100）

命令寄存器（Command）的地址为 0x2008 4100。各个位的定义见[表 169](#)。

表169. 命令寄存器位描述（Command—0x2008 4100）

位	符号	功能	复位值
0	RxEnable	接收使能。	0
1	TxEnable	发送使能。	0
2	-	未使用	0x0
3	RegReset	向该位写入“1”时，所有的通道和主机寄存器均复位。MAC 需要单独进行复位。	0
4	TxReset	向该位写入“1”时，发送通道复位。	0
5	RxReset	向该位写入“1”时，接收通道复位。	0
6	PassRuntFrame	该位被设为“1”时，将小于 64 字节的短帧传递到寄存器中，除非该短帧的 CRC 有误。如果该位被设成“0”，则将短帧滤除。	0
7	PassRxFilter	该位被设为“1”时，禁止对接收过滤，即把所有接收到的帧都写入存储器中。	0
8	TxFlowControl	使能 IEEE 802.3/条款 31 的流控，即在全双工下发送暂停控制帧，在半双工下发送连续的导言。	0
9	RMII	该位被设为“1”时，选择 RMII 模式；该位被设为“0”时，选择 MII 模式。	0
10	FullDuplex	该位被设为“1”表示在全双工模式下操作。	0
31:11	-	未使用	0x0

所有位均可以执行读写操作。Tx/RxReset 位是只写的，读操作将返回 0。

11.12.2 状态寄存器（Status—0x2008 4104）

状态寄存器（Status）是只读寄存器，地址为 0x2008 4104。各个位的定义见[表 170](#)。

表170. 状态寄存器位描述（Status—0x2008 4104）

位	符号	功能	复位值
0	RxStatus	如果该位为 1，接收通道处于活动状态。如果该位为 0，接收通道不工作。	0
1	TxStatus	如果该位为 1，发送通道处于活动状态。如果该位为 0，发送通道不工作。	0
31:2	-	未使用	0x0

该寄存器的值表示了两个通道的状态。当状态为 1 时，通道处于活动状态，表明：

- 在发送或接收帧信息的同时，通道使能，且命令寄存器中的 Rx/TxEnable 位置位，否则通道是禁止的。
- 对于发送通道，发送队列不为空，即 ProduceIndex != ConsumeIndex。
- 对于接收通道，接收队列不为空，即 ProduceIndex != ConsumeIndex - 1。

如果 Command 寄存器中 Rx/TxEnable 位的软件复位禁用了通道，并且通道已将当前帧的状态与数据提交给了存储器，则状态从活动转变为静止。如果“发送队列”为空或“接收队列”为满，并且状态和数据已提交给存储器，则通道状态也会转变为静止。

11.12.3 接收描述符基址寄存器（RxDescriptor—0x2008 4108）

接收描述符基址寄存器（RxDescriptor）的地址为 0x2008 4108。各个位的定义见[表 171](#)。

表171. 接收描述符基址寄存器位描述（RxDescriptor—0x2008 4108）

位	符号	功能	复位值
1:0	-	固定为 “00”	-
31:2	RxDescriptor	接收描述符基址的 MSB。	0x0

接收描述符的基址是一个字边界对齐的字节地址，即 LSB 1:0 固定为 “00”。该寄存器含有描述符数组的最低地址。

11.12.4接收状态基址寄存器（RxStatus—0x2008 410C）

接收描述符的基址是一个字边界对齐的字节地址，即 LSB 1:0 固定为 “00”。此寄存器包含了描述符数组中的最低地址。

表172. 接收状态基址寄存器位描述（RxStatus—0x2008 410C）

位	符号	功能	复位值
2:0	-	固定为 “00”	-
31:3	RxStatus	接收状态基址的 MSB。	0x0

接收状态的基址是一个双字对齐的字节地址，即 LSB 2:0 固定为 “000”。

11.12.5接收描述符数目寄存器（RxDescriptor—0x2008 4110）

接收描述符数目寄存器（RxDescriptorNumber）的地址为 0x2008 4110。各个位的定义见[表 173](#)。

表173. 接收描述符数目寄存器位描述（RxDescriptor—0x2008 4110）

位	符号	功能	复位值
15:0	RxDescriptorNumber	在以 RxDescriptor 为基址的描述符数组中的描述符数目。描述符的数目为 0x0 减 1 编码。	0x0
31:16	-	未使用	0x0

接收描述符数目寄存器定义了以 RxDescriptor 为基址的描述符数组中的描述符数量。描述符的数目应与状态数目相匹配。寄存器采用了减 1 编码，即，如果数组有 8 个元素，则寄存器中的值应为 7。

11.12.6 接收产生索引寄存器（RxProduceIndex—0x2008 4114）

接收产生索引寄存器（RxProduceIndex）是一个只读寄存器，地址为 0x2008 4114。各个位的定义见[表 174](#)。

表174. 接收产生索引寄存器位描述（RxProduceIndex—0x2008 4114）

位	符号	功能	复位值
15:0	RxProduceIndex	下一次将被接收通道填充的描述符的索引。	0x0
31:16	-	未使用	0x0

接收产生索引寄存器定义了下一个要被硬件接收处理填充的描述符。在收到一个帧后，硬件将索引加 1。一旦达到 RxDescriptorNumber 的值，则寄存器值回为 0。如果 RxProduceIndex 等于 RxConsumeIndex – 1，则数组满，再接收的任何帧都会造成缓冲区溢出错误。

11.12.7 接收消耗索引寄存器（RxConsumeIndex—0x2008 4118）

接收消耗索引寄存器（RxConsumeIndex）的地址为 0x2008 4118。各个位的定义见[表 175](#)。

表175. 接收消耗索引寄存器位描述（RxConsumeIndex—0x2008 4118）

位	符号	功能	复位值
15:0	RxConsumeIndex	下一次将被接收处理的描述符的索引	
31:16	-	未使用	0x0

接收消耗索引寄存器定义了下一个要被软件接收驱动处理的描述符。只要 RxProduceIndex 等于 RxConsumeIndex，接收数组就为空。如果数组不空，软件就可以处理由 RxConsumeIndex 指向的帧。软件处理完一个帧后，应将 RxConsumeIndex 加 1。一旦寄存器达到 RxDescriptorNumber 的值，则寄存器值必须回到 0。如果 RxProduceIndex 等于 RxConsumeIndex – 1，则数组满，再接收的任何帧都会造成缓冲区溢出错误。

11.12.8 发送描述符基址寄存器（TxDescriptor—0x2008 411C）

发送描述符基址寄存器（TxDescriptor）的地址为 0x2008 411C。各个位的定义见[表 176](#)。

表176. 发送描述符基址寄存器位描述（TxDescriptor—0x2008 411C）

位	符号	功能	复位值
1:0	-	固定为“00”	-
31:2	TxDescriptor	发送描述符基址的 MSB。	0x0

发送描述符基址是一个与字边界对齐的字节地址，即 LSB 1:0 固定为“00”。此寄存器包含了描述符数组中的最低地址。

11.12.9 发送状态基址寄存器（TxStatus—0x2008 4120）

发送状态基址寄存器（TxStatus）的地址为 0x2008 4120。各个位的定义见[表 177](#)。

表177. 发送状态基址寄存器位描述（TxStatus—0x2008 4120）

位	符号	功能	复位值
1:0	-	固定为“00”	-
31:2	TxStatus	发送状态基址的 MSB。	0x0

发送状态的基址是一个与字边界对齐的字节地址，即 LSB 1:0 固定为“00”。此寄存器包含了状态数组中的最低地址。

11.12.10 发送描述符数目寄存器（TxDescriptorNumber—0x2008 4124）

发送描述符数目寄存器（TxDescriptorNumber）的地址为 0x2008 4124。各个位的定义见[表 178](#)。

表178. 发送描述符数目寄存器位描述（TxDescriptorNumber—0x2008 4124）

位	符号	功能	复位值
15:0	TxDescriptorNumber	在以 TxDescriptor 为基址的描述符数组中的描述符数目。该寄存器采用减 1 编码。	
31:16	-	未使用	0x0

发送描述符数目寄存器定义了以 TxDescriptor 为基址的描述符数组中的描述符数目。描述符数目应与状态数目相匹配。寄存器采用了减 1 编码，即，如果数组有 8 个元素，则寄存器中的值应为 7。

11.12.11 发送产生索引寄存器（TxProduceIndex—0x2008 4128）

发送产生索引寄存器（TxProduceIndex）的地址为 0x2008 4128。各个位的定义见[表 179](#)。

表179. 发送产生索引寄存器（TxProduceIndex—0x2008 4128）位描述

位	符号	功能	复位值
15:0	TxProduceIndex	下一次将被发送软件驱动程序填充的描述符的索引。	0x0
31:16	-	未使用	0x0

发送产生索引寄存器定义了下一个要由软件发送驱动填充的描述符。如果 TxProduceIndex 等于 TxConsumeIndex 时，传送描述符数组就为空。如果发送硬件使能，则一旦描述符数组不为空时，就开始传送帧。当软件处理完一个帧后，会将 TxProduceIndex 加 1。一旦达到了 TxDescriptorNumber 的值，寄存器值必须回到 0。如果 TxProduceIndex 等于 TxConsumeIndex - 1，则描述符数组为满，软件应停止产生新的描述符，直到硬件已发送了一些帧，并更新 TxConsumeIndex。

11.12.12 发送消耗索引寄存器（TxConsumeIndex—0x2008 412C）

发送消耗索引寄存器（TxConsumeIndex）是一个只读寄存器，地址为 0x2008 412C。各个位的定义见[表 180](#)。

表180. 发送消耗索引寄存器位描述（TxConsumeIndex—0x2008 412C）

位	符号	功能	复位值
15:0	TxConsumeIndex	下一次将被发送通道发送的描述符的索引。	0x0
31:16	-	未使用	0x0

发送消耗索引寄存器定义了下一次将被硬件发送处理发送描述符。当发送完一帧之后，硬件将 TxConsumeIndex 加 1。如果它与 TxDescriptorNumber 的值相等，则该寄存器的值回到 0。如果 TxConsumeIndex 等于 TxProduceIndex，则描述符数组为空，发送通道将停止发送，直到软件产生新的描述符。

11.12.13 发送状态向量 0 寄存器（TSV0—0x2008 4158）

发送状态向量 0 寄存器（TSV0）是一个只读寄存器，地址为 0x2008 4158。发送状态向量寄存器保存着由 MAC 返回的最新发送状态。由于状态向量超过了 4 个字节，因此状态被分配到两个寄存器 TSV0 和 TSV1 中。这些寄存器是供调试使用，因为驱动软件与以太网模块之间的通信主要是通过帧描述符来实现。只要 MAC 的内部状态有效，则状态寄存器的内容也有效，当发送与接收处理都暂停时，状态寄存器通常只执行读操作。

[表 181](#) 列出了 TSV0 寄存器各位的定义。

表181. 发送状态向量 0 寄存器位描述（TSV0—0x2008 4158）

位	符号	功能	复位值
0	CRC error	包中附带的 CRC 与内部产生的 CRC 不相等。	0
1	Length check error	表示帧长度区域的值与实际的数据个数不相等，并且此时的帧长度区域不表示类型。	0
2	Length out of range ^[1]	表示帧类型/长度区域的值大于 1500 个字节。	0
3	Done	包发送完成。	0
4	Multicast	包的目标地址为多播地址。	0
5	Broadcast	包的目标地址为广播地址。	0
6	Packet Defer	包至少被延迟了一次尝试，但还没有达到延迟期限。	0
7	Excessive Defer	包在 100Mbps 下的延迟时间超出了 6071 个半字节时间，在 10Mbps 下的延迟时间超出了 24287 个位时间。	0
8	Excessive Collision	包由于超过最大允许的冲突次数而被中止。	0
9	Late Collision	产生的冲突超出了冲突窗口，即超出了 512 个位时间。	0
10	Giant	帧中的字节数大于 TSV1 的发送字节计数字段中能够表示的字节数。	0
11	Underrun	主机方引起的缓冲区下溢。	0
27:12	Total bytes	包括冲突尝试在内的传输字节总数。	0x0
28	Control frame	该帧是一个控制帧。	0
29	Pause	该帧是一个带有有效 PAUSE 操作码的控制帧。	0
30	Backpressure	前面应用了载波侦听方式的背压。	0
31	VLAN	帧长度/类型区域含有 0x8100，它是 VLAN 的协议标识符。	0

[1] EMAC 不区分帧类型和帧长度。例如，当接收到 IP(0x8000)或 ARP(0x0806)包时，EMAC 将帧类型与最大长度进行比较并给出“长度超出范围”错误。事实上，该位不是一个错误指示，而仅仅是由芯片产生的、关于接收帧状态的一个说明。

11.12.14 发送状态向量 1 寄存器 (TSV1—0x2008 415C)

发送状态向量 1 寄存器 (TSV1) 是一个只读寄存器，地址为 0x2008 415C。发送状态向量寄存器保存着由 MAC 返回的最新发送状态。由于状态向量超过了 4 个字节，因此状态被分配到两个寄存器 TSV0 和 TSV1 中。这些寄存器是供调试使用的，因为驱动软件与以太网模块之间的通信主要是通过帧描述符来实现。只要 MAC 的内部状态有效，则状态寄存器的内容也有效，当发送与接收过程暂停时，状态寄存器通常只执行读操作。[表 182](#) 列出了 TSV1 寄存器各位的定义。

表182. 发送状态向量 1 寄存器位描述 (TSV1—0x2008 415C)

位	符号	功能	复位值
15:0	Transmit byte count	发送帧中的字节总数，不包括冲突的字节数。	0x0
19:16	Transmit collision count	当前帧在发送过程中遇到的冲突次数。该值不能达到冲突的最大次数 (16)。	0x0
31:20	-	未使用	0x0

11.12.15 接收状态向量寄存器 (RSV—0x2008 4160)

接收状态向量寄存器 (RSV) 是一个只读寄存器，地址为 0x2008 4160。接收状态向量寄存器保存着由 MAC 返回的最新接收状态。此寄存器是供调试使用的，因为驱动软件与以太网模块之间的通信主要是通过帧描述符来实现。只要 MAC 的内部状态有效，则状态寄存器的内容也有效，当发送与接收过程暂停时，状态寄存器通常才执行读操作。

[表 183](#) 列出了 RSV 寄存器各位的定义。

表183. 接收状态向量寄存器位描述 (RSV—address 0x2008 4160)

位	符号	功能	复位值
15:0	Received byte count	表示接收到的帧信息的长度。	0x0
16	Packet previously ignored	表示漏掉 (drop) 了一个包。	0
17	RXDV event previously seen	表示上一次发现的接收事件其长度不够，不能成为一个有效的包。	0
18	Carrier event previously seen	表示上一次接收统计之后的某个时候，检测到载波事件。	0
19	Receive code violation	表示接收到的 PHY 数据不代表一个有效的接收代码。	0
20	CRC error	包中附带的 CRC 与内部产生的 CRC 不相等。	0
21	Length check error	表示帧长度区域与实际的数据个数不相等，且此时的帧长度区域不表示类型。	0
22	Length out of range ^[1]	表示帧类型/长度区域的值大于 1518 个字节。	0
23	Receive OK	表示接收包含有效的 CRC 并且没有符号错误。	0
24	Multicast	包的目标地址为多播地址。	0
25	Broadcast	包的目标地址为广播地址。	0
26	Dribble Nibble	表示接收到包之后又接收到另一个 1~7 位的数据。此时形成了一个 nibble，称作 dribble nibble，但没有发送出去。	0
27	Control frame	该帧是一个控制帧。	0
28	PAUSE	该帧是一个带有有效 PAUSE 操作码的控制帧。	0
29	Unsupported Opcode	当前帧是控制帧，但含有未知的操作码。	0
30	VLAN	帧长度/类型区域含有 0x8100，它是 VLAN 的协议标识符。	0
31	-	未使用	0x0

[1] EMAC 不区分帧类型和帧长度。例如，当接收到 IP(0x8000)或 ARP(0x0806)包时，EMAC 将帧

类型与最大长度进行比较并给出“长度超出范围”错误。事实上，该位不是一个错误指示，而只仅仅是由芯片产生的、关于接收帧状态的一个说明。

11.12.16 流控制计数器寄存器（FlowControlCounter—0x2008 4170）

流控制计数器寄存器（FlowControlCounter）的地址为 0x2008 4170。[表 184](#) 列出了该寄存器的位定义。

表184. 流控制计数器寄存器位描述（FlowControlCounter—0x2008 4170）

位	符号	功能	复位值
15:0	MirrorCounter	在全双工模式下，该字段指定了重新发送暂停控制帧之前的周期数。	0x0
31:16	PauseTimer	在全双工模式下，该字段指定了插入暂停流控制帧的暂停定时器区域的值。 在半双工模式下，该字段指定了背压周期数。	0x0

11.12.17 流控制状态寄存器（FlowControlStatus—0x2008 4174）

流控制状态寄存器（FlowControlStatus）是一个只读寄存器，地址为 0x2008 4174。[表 185](#) 列出了寄存器各位的定义。

表185. 流控制状态寄存器位描述（FlowControlStatus—0x2008 4174）

位	符号	功能	复位值
15:0	MirrorCounterCurrent	在全双工模式下，该字段表示数据通道的镜像计数器（mirror counter）的当前值，该计数器最高可达到流控制计数器寄存器的 MirrorCounter 字段指定的值。在半双工模式下，该字段的值可达到流控制计数器寄存器的 PauseTimer 字段的值。	0x0
31:16	-	未使用	0x0

11.13接收过滤寄存器定义

11.13.1接收滤波器控制寄存器（RxFilterCtrl—0x2008 4200）

接收滤波器控制寄存器（RxFilterCtrl）的地址为 0x2008 4200。[表 186](#)列出了寄存器中各个位的定义。

表186. 接收滤波器控制寄存器位描述（RxFilterCtrl—0x2008 4200）

位	符号	功能	复位值
0	AcceptUnicastEn	当该位设为“1”时，接受所有的单播帧。	0
1	AcceptBroadcastEn	当该位设为“1”时，接受所有的广播帧。	0
2	AcceptMulticastEn	当该位设为“1”时，接受所有的多播帧。	0
3	AcceptUnicastHashEn	当该位设为“1”时，接受通过不完全 Hash 滤波器的单播帧。	0
4	AcceptMulticastHashEn	当该位设为“1”时，接受通过不完全 Hash 滤波器的多播帧。	0
5	AcceptPerfectEn	当该位设为“1”时，接受目标地址与站地址相同的帧。	0
11:6	-	保留。读取值未定义，只写入 0。	无
12	MagicPacketEnWoL	当该位设为“1”时，魔法包滤波器的结果在匹配时将产生一个 WoL 中断。	0
13	RxFilterEnWoL	当该位设为“1”时，完全地址匹配滤波器与不完全 Hash 滤波器的结果在匹配时将产生一个 WoL 中断。	0
31:14	-	未使用	0x0

11.13.2接收滤波器 WoL 状态寄存器（RxFilterWoLStatus—0x2008 4204）

接收滤波器 LAN 上唤醒状态寄存器(RxFilterWoLStatus)是一个只读寄存器,地址为 0x2008 4204。

[表 187](#)列出了寄存器中各个位的定义。

表187. 接收滤波器 WoL 状态寄存器位描述（RxFilterWoLStatus—0x2008 4204）

位	符号	功能	复位值
0	AcceptUnicastWoL	当该值为“1”时，一个单播帧引起 WoL。	0
1	AcceptBroadcastWoL	当该值为“1”时，一个广播帧引起 WoL。	0
2	AcceptMulticastWoL	当该值为“1”时，一个多播帧引起 WoL。	0
3	AcceptUnicastHashWoL	当该值为“1”时，一个通过不完全 Hash 滤波器的单播帧引起 WoL。	0
4	AcceptMulticastHashWoL	当该值为“1”时，一个通过不完全 Hash 滤波器的多播帧引起 WoL。	0
5	AcceptPerfectWoL	当该值为“1”时，完全地址匹配滤波器引起 WoL。	0
6	-	未使用	0x0
7	RxFilterWoL	当该值为“1”时，接收滤波器引起 WoL。	0
8	MagicPacketWoL	当该值为“1”时，魔法包滤波器引起 WoL。	0
31:9	-	未使用	0x0

该寄存器中的位记录了产生 WoL 的原因。这些位可通过对 RxFilterWoLClear 寄存器执行写操作来清零。

11.13.3接收滤波器 WoL 清零寄存器（RxFilterWoLClear—0x2008 4208）

接收滤波器 LAN 上唤醒清零寄存器(RxFilterWoLClear)是一个只写寄存器,地址为 0x2008 4208。

表 188 列出了寄存器中各个位的定义。

表188. 接收滤波器 WoL 清零寄存器位描述（RxFilterWoLClear—0x2008 4208）

位	符号	功能	复位值
0	AcceptUnicastWoLCIr	向位 0~5 的其中一位写入“1”时, RxFilterWoLStatus 寄存器中对应的状态位被清零。	0
1	AcceptBroadcastWoLCIr		0
2	AcceptMulticastWoLCIr		0
3	AcceptUnicastHashWoLCIr		0
4	AcceptMulticastHashWoLCIr		0
5	AcceptPerfectWoLCIr		0
6	-	未使用	0x0
7	RxFilterWoLCIr	向位 7 和/或 8 写入“1”时, RxFilterWoLStatus 寄存器中对应的状态位被清零。	0
8	MagicPacketWoLCIr		0
31:9	-	未使用	0x0

此寄存器中各个位都是只写的；写入“1”将清零 RxFilterWoLStatus 寄存器中的相应位。

11.13.4Hash 滤波器表 LSB 寄存器（HashFilterL—0x2008 4210）

Hash 滤波器表 LSB 寄存器（HashFilterL）的地址为 0x2008 4210。表 189 列出了寄存器中各个位的定义。Hash 滤波器表的使用详见 11.17.10 节。

表189. Hash 滤波器表 LSB 寄存器位描述（HashFilterL—0x2008 4210）

位	符号	功能	复位值
31:0	HashFilterL	用于接收过滤的不完全滤波器 Hash 表的位 31:0。	0x0

11.13.5Hash 滤波器表 MSB 寄存器（HashFilterH—0x2008 4214）

Hash 滤波器表 MSB 寄存器（HashFilterH）的地址为 0x2008 4214。表 190 列出了寄存器中各个位的定义。Hash 滤波器表的使用详见 11.17.10 节。

表190. Hash 滤波器表 MSB 寄存器位描述（HashFilterH—0x2008 4214）

位	符号	功能	复位值
31:0	HashFilterH	用于接收过滤的不完全滤波器 Hash 表的位 63:32。	0x0

11.14 模块控制寄存器定义

11.14.1 中断状态寄存器（IntStatus—0x2008 4FE0）

中断状态寄存器（IntStatus）是一个只读寄存器，地址为 0x2008 4FE0。中断状态寄存器的位定义见表 191。注意，所有位都是带异步置位的触发器，这样，如果当时钟禁用时出现了一个唤醒事件，就能够产生中断。

表191. 中断状态寄存器位描述（IntStatus—0x2008 4FE0）

位	符号	功能	复位值
0	RxOverrunInt	在接收队列中出现重大的溢出错误时中断置位。这个重大的中断应该通过 Rx 软件复位来解决。该位在出现一个非重大的溢出错误时不会置位。	0
1	RxErrorInt	接收出现错误时中断触发。接收错误包括：AlignmentError、RangeError、LengthError、SymbolError、CRCError 或 NoDescriptor 或 Overrun。	0
2	RxFinishedInt	当所有的接收描述符均已处理完时，即当传输满足 ProduceIndex == ConsumeIndex 时中断触发。	0
3	RxDoneInt	在接收描述符处理完成，并且描述符控制区域中的中断位被置位时中断触发。	0
4	TxUnderrunInt	在发送队列中出现重大的溢出错误时中断置位。这个重大的中断应该通过 Tx 软件复位来解决。该位在出现一个非重大的溢出错误时不会置位。	0
5	TxErrorInt	发送出现错误时中断触发。发送错误包括：LateCollision、ExcessiveCollision 和 ExcessiveDefer、NoDescriptor 或 Underrun。	0
6	TxFinishedInt	当所有的发送描述符均已处理完时，即当传输满足 ProduceIndex == ConsumeIndex 时中断触发。	0
7	TxDoneInt	在描述符已发送完成，并且描述符控制区域中的中断位被置位时中断触发。	0
11:8	-	未使用	0x0
12	SoftInt	软件向 IntSet 寄存器的 SoftIntSet 位写入 1 时中断触发。	0
13	WakeUpInt	接收滤波器检测到一个唤醒事件触发的中断。	0
31:14	-	未使用	0x0

中断状态寄存器为只读寄存器。通过 IntSet 寄存器可实现置位操作，通过 IntClear 寄存器可实现复位操作。

11.14.2 中断使能寄存器（IntEnable—0x2008 4FE4）

中断使能寄存器（IntEnable）的地址为 0x2008 4FE4。中断使能寄存器各位的定义见表 192。

表192. 中断使能寄存器位描述（intEnable—0x2008 4FE4）

位	符号	功能	复位值
0	RxOverrunIntEn	使能在接收缓冲区溢出或描述符下溢时的中断触发。	0
1	RxErrorIntEn	使能接收错误时的中断触发。	0
2	RxFinishedIntEn	使能当所有接收描述符已完成，即传输满足 ProduceIndex == ConsumeIndex 时的中断触发。	0
3	RxDoneIntEn	使能在接收描述符已处理完成并且描述符控制区域中的中断位置位时的中断触发。	0
4	TxUnderrunIntEn	使能在发送缓冲区溢出或描述符下溢时的中断触发。	0
5	TxErrorIntEn	使能发送错误时的中断触发。	0
6	TxFinishedIntEn	使能当所有发送描述符已处理完，即传输满足 ProduceIndex == ConsumeIndex 时的中断触发。	0

位	符号	功能	复位值
7	TxDoneIntEn	使能在描述符已发送完成并且描述符控制区域中的中断位置位时的中断触发。	0
11:8	-	未使用	0x0
12	SoftIntEn	使能由 IntStatus 寄存器中的 SoftInt 位触发的中断，通过软件向 IntSet 寄存器的 SoftIntSet 位写入“1”来产生中断触发。	0
13	WakeupIntEn	使能由接收滤波器检测到的唤醒事件触发的中断。	0
31:14	-	未使用	0x0

11.14.3 中断清零寄存器（IntClear—0x2008 4FE8）

中断清零寄存器（IntClear）是一个只写寄存器，地址为 0x2008 4FE8。中断清零寄存器各位的定义见[表 193](#)。

表193. 中断清零寄存器位描述（IntClear—0x2008 4FE8）

位	符号	功能	复位值
0	RxOverrunIntClr	向位 0~7 的其中一位写入“1”可将中断状态寄存器 IntStatus 中的对应位清零。	0
1	RxErrorIntClr		0
2	RxFinishedIntClr		0
3	RxDoneIntClr		0
4	TxUnderrunIntClr		0
5	TxErrorIntClr		0
6	TxFinishedIntClr		0
7	TxDoneIntClr		0
11:8	-	未使用	0x0
12	SoftIntClr	向位 12 和/或 13 写入“1”可将中断状态寄存器 IntStatus 中的对应位清零。	0
13	WakeupIntClr		0
31:14	-	未使用	0x0

中断清零寄存器为只写寄存器。向 IntClear 寄存器的一个位写入 1，会清零状态寄存器中的相应位。写入 0 则对中断状态没有影响。

11.14.4 中断置位寄存器（IntSet—0x2008 4FEC）

中断置位寄存器（IntSet）是一个只写寄存器，地址为 0x2008 4FEC。中断置位寄存器各位的定义见[表 194](#)。

表194. 中断置位寄存器位描述（IntSet—0x2008 4FEC）

位	符号	功能	复位值
0	RxOverrunIntSet	向位 0~7 的其中一位写入“1”可将中断状态寄存器（IntStatus）中的对应位置位。	0
1	RxErrorIntSet		0
2	RxFinishedIntSet		0
3	RxDoneIntSet		0

位	符号	功能	复位值
4	TxUnderrunIntSet		0
5	TxErrorIntSet		0
6	TxFinishedIntSet		0
7	TxDoneIntSet		0
11:8	-	未使用	0x0
12	SoftIntSet	向位 12 和/或 13 写入“1”可将中断状态寄存器（IntStatus）中的对应位置位。	0
13	WakeUpIntSet		0
31:14	-	未使用	0x0

中断置位寄存器为只写寄存器。向 IntSet 寄存器的一个位写入 1，会置位状态寄存器中的相应位。写入 0 则对中断状态没有影响。

11.14.5掉电寄存器（PowerDown—0x2008 4FF4）

掉电寄存器（PowerDown）用于阻止除对掉电寄存器以外的所有 AHB 访问。此寄存器的地址为 0x2008 4FF4。寄存器各位的定义见[表 195](#)。

表195. 掉电寄存器位描述（PowerDown—0x2008 4FF4）

位	符号	功能	复位值
30:0	-	未使用	0x0
31	PowerDownMACAHB	如果该位为“真”，则除了访问掉电寄存器之外的所有 AHB 访问都将返回读/写错误。	0

当位 31 置位时，对 MACAHB 接口上（访问 PowerDown 寄存器除外）的所有读和写访问都将返回一个错误。

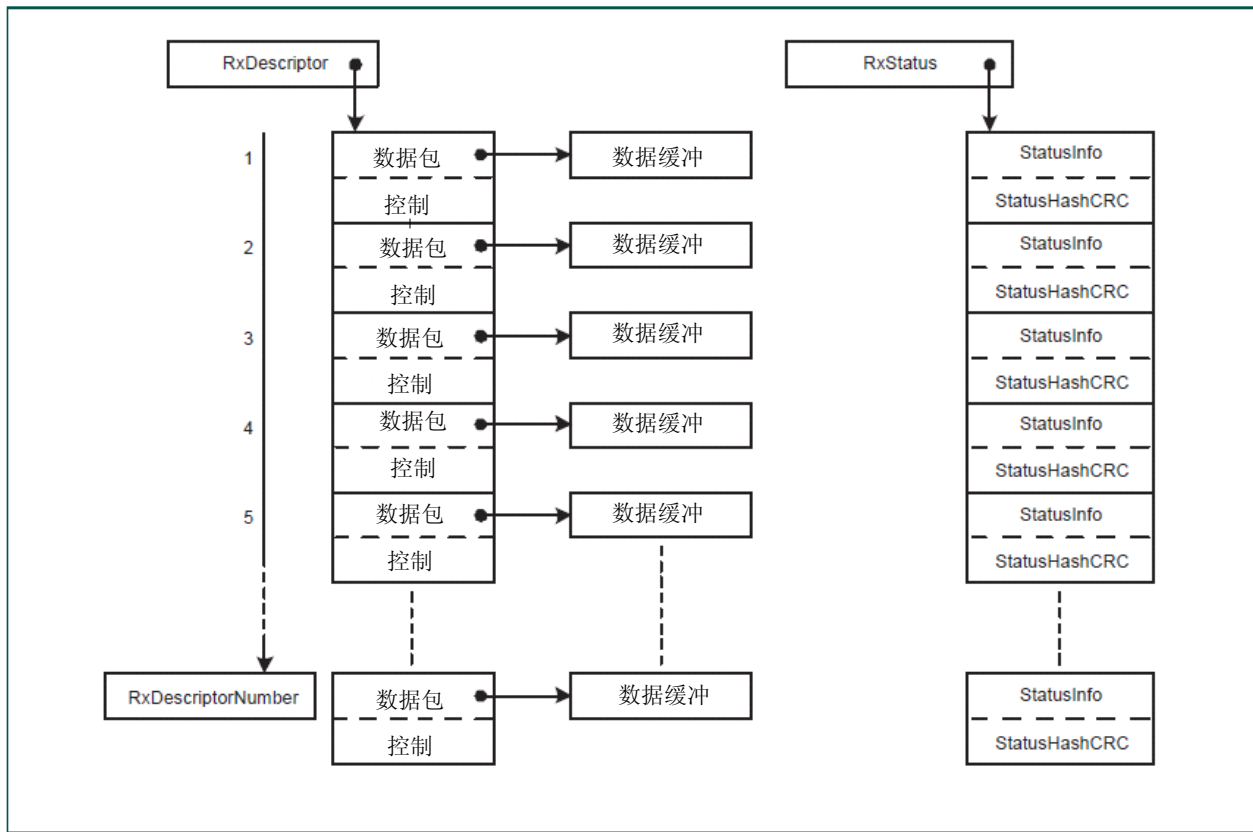
11.15描述符与状态格式

本节定义了发送与接收分散/集中 DMA 引擎的描述符格式。每个以太网帧都可以包括一个或多个片段。每个片段都对应于单一的描述符。以太网模块中的 DMA 管理器用于分散（用于接收）和集中（用于发送）单个以太网帧的多个片段。

11.15.1接收描述符与状态

[图 22](#) 描述了接收描述符在存储器中的布局。

图22. 接收描述符在存储器中的规划



接收描述符保存在存储器的一个数组中。数组的基址保存在 **RxDescriptor** 寄存器中，应与一个 4 字节地址边界对齐。数组的描述符数量以减 1 编码方式，保存在 **RxDescriptorNumber** 寄存器中，即：如果数组有 8 个元素，则寄存器值应为 7。与描述符平行的是一个状态数组。对于描述符数组中的每个元素，状态数组中都有一个相应的状态区域。状态数组的基址保存在 **RxStatus** 寄存器中，并且必须与 8 字节的地址边界对齐。在操作过程中（此时接收数据通道使能），**RxDescriptor**、**RxStatus** 和 **RxDescriptorNumber** 均不得被修改。

RxConsumeIndex 与 **RxProduceIndex** 这两个寄存器定义了将被硬件和软件使用的下一个描述符单元。两个寄存器均作为起始为 0 的计数器，当它们到达 **RxDescriptorNumber** 的值时，再返回从 0 计数。**RxProduceIndex** 包含了将被接收到的下一个帧数据填充的描述符索引。**RxConsumeIndex** 由软件编写，是软件接收驱动下一个要处理的描述符索引。当 **RxProduceIndex** == **RxConsumeIndex** 时，接收缓冲区为空。当 **RxProduceIndex** == **RxConsumeIndex** - 1 时（考虑到它是一个封包设计），接收缓冲区为满，此时除非软件驱动程序释放一个或多个描述符，否则新接收的数据将产生溢出。

每个“接收描述符”都占用存储器中两个字（8 字节）。同样，每个状态区域也占用存储器中两个字（8 字节）。每个接收描述符都包括一个“数据包指针（**Packet**）”和一个“控制字（**Control**）”组成，指针指向用来存放接收数据的数据缓冲区，控制字包含的是控制信息。针对表 196 中定义的描述符地址，**Packet** 区域的地址偏移量为 0，而 **Control** 区域的地址

偏移量为 4 字节。

表196. 接收描述符的区域

符号	地址偏移量	字节	描述
Packet	0x0	4	用来存放接收数据的数据缓冲区的基址。
Control	0x4	4	控制信息，见表 197。

数据缓冲区指针（**Packet**）是一个按字节对齐的 32 位地址值，包含了数据缓冲区的基址。控制字各位的定义见表 197。

表197. 接收描述符控制字

位	符号	描述
10:0	Size	数据缓冲区的字节数。这是设备驱动程序为一帧或帧片段保留的缓冲区字节数，即被“数据包区域”指向的缓冲区的字节数。 Size 的值采用的是减 1 编码，例如，如果缓冲区为 8 字节，则 Size 的值为 7。
30:11	-	未使用
31	Interrupt	该位表示当该帧或帧片段中的数据以及相关的状态信息已提交给寄存器时，是否确实产生了一个 RxDone 中断。

表 198 列出了状态数组中接收状态单元的各个区域。

表198. 接收状态的区域

符号	地址偏移量	字节	描述
StatusInfo	0x0	4	接收状态的返回标志，见表 200。
StatusHashCRC	0x4	4	目标地址 Hash CRC 和源地址 Hash CRC 的串联。

每个接收状态都包含两个字。**StatusHashCRC** 包含两个串联的 9 位 Hash CRC，计算自接收帧中所包含的目标地址和源地址。在检测了目标地址和源地址后，**StatusHashCRC** 进行一次计算，然后保存该值，供相同帧的所有片段使用。

两个串联的 CRC 见表 199。

表199. 接收状态 Hash CRC 字

位	符号	描述
8:0	SAHashCRC	从源地址中计算而得的 Hash CRC。
15:9	-	未使用
24:16	DAHashCRC	从目标地址中计算而得的 Hash CRC。
31:25	-	未使用

StatusInfo 字中包含了 MAC 返回的标志，以及由接收数据通道生成的标志，它们反映了接收的状态。表 200 列出了 **StatusInfo** 字中各位的定义。

表200. 接收状态信息字

位	符号	描述
10:0	RxSize	传输给一个片段缓冲区的实际数据的字节数。换句话说，它是 DMA 管理器针对一个描述符实际写入的帧或片段的字节数。该值可能与描述符控制区域中的 Size 位（表示器件驱动程序分配的缓冲区大小）的值有所不同。该字段采用减 1 编码，例如，如果缓冲区有 8 个字节，则 RxSize 的值为 7。
17:11	-	未使用
18	ControlFrame	表示这一个用于流控制的控制帧，它可以是一个暂停帧也可以是一个带有不支持的操作码的帧。
19	VLAN	表示一个 VLAN 帧。
20	FailFilter	表示这帧信息的 Rx 过滤失败。这样的帧将不能正常地传递到存储器中。但由于缓冲区大小的限制，帧中可能已有一部分信息传递到了存储器。一旦发现某帧的 Rx 过滤失败，就将该帧的剩余部分丢弃，而不传递到存储器中。但如果命令寄存器中的 PassRxFilter 位置位，则整帧都将传递到存储器中。
21	Multicast	当接收到一个多播帧时置位。
22	Broadcast	当接收到一个广播帧时置位。
23	CRCErr	接收到的帧有一个 CRC 错误。
24	SymbolError	在接收过程中，PHY 通过 PHY 接口报告有一个位错误。
25	LengthError	该帧的帧长度区域指定了一个有效的帧长度，但它与实际的数据长度不相等。
26	RangeError	接收到的包超出了包长度的最大限制。
27	AlignmentError	当检测到 dribble 位和一个 CRC 错误时，将“对齐错误”作上标记。这与 IEEE std.802.3/条款 4.3.2 是一致的。
28	Overrun	接收溢出。适配器不能接收数据流。
29	NoDescriptor	没有新的 Rx 描述符可用，并且对于当前的接收描述符中的缓冲区大小来说，帧信息太长。
30	LastFlag	该位设为“1”时，表示这个描述符是一帧中的最后一个片段。如果一帧只有一个片段组成，则该位也设成“1”。
31	Error	表示在该帧的接收过程中出现错误。它是 AlignmentError、RangeError、LengthError、SymbolError、CRCErr 和 Overrun 逻辑相“或”的结果。

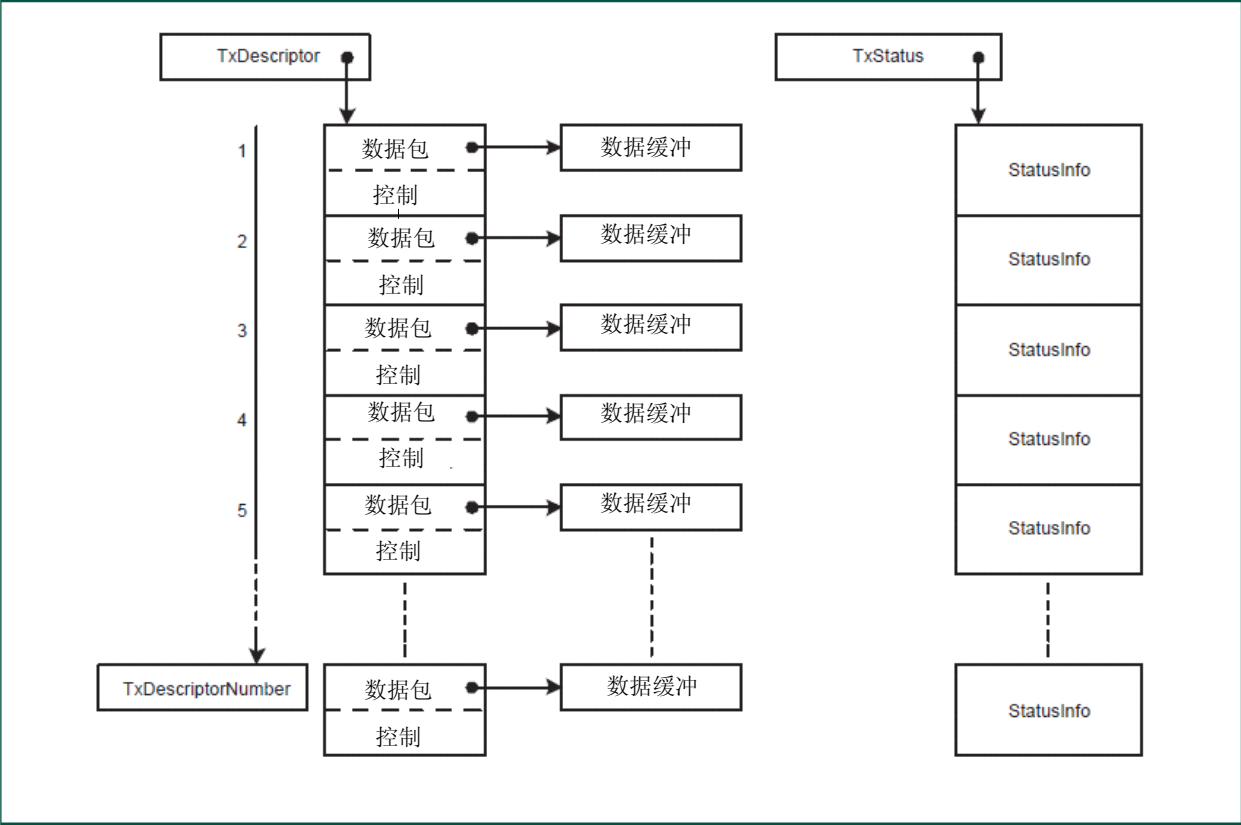
[1] EMAC 不区分帧类型和帧长度。例如，当接收到 IP(0x8000)或 ARP(0x0806)包时，EMAC 将帧类型与最大长度进行比较并给出“长度超出范围”错误。事实上，该位不是一个错误指示，而仅仅是由芯片产生的、关于接收帧状态的一个说明。

对于多片段的帧，AlignmentError、RangeError、LengthError、SymbolError 以及 CRCErr 各位的值中，除了对帧中最后一个片段以外都将为 0；同样，FailFilter、Multicast、Broadcast、VLAN 与 ControlFrame 各位的值未定义。帧中最后一个片段的状态将从 MAC 复制这些位的值。所有片段的状态都有有效的 LastFrag、RxSize、Error、Overrun 与 NoDescriptor 位。

11.15.2 发送描述符与状态

图 23 给出了发送描述符在存储器中的布局。

图23. 发送描述符在存储器中的规划



发送描述符保存在存储器的一个数组中。发送描述符数组的最低地址保存在 **TxDescriptor** 寄存器中，并且必须与一个 4 字节的地址边界对齐。数组中的描述符数量使用减 1 编码，保存在 **TxDescriptorNumber** 寄存器中，即：如果数组有 8 个元素，则寄存器值应为 7。与描述符平行的是一个“状态数组”。对于描述符数组中的每个元素，“状态数组”中都有一个相应的状态区域。状态数组的基址保存在 **TxStatus** 寄存器中，并且必须与 4 字节的地址边界对齐。在操作过程中(此时发送数据通道使能)，**TxDescriptor**、**TxStatus** 和 **TxDescriptorNumber** 均不得被修改。

TxConsumeIndex 与 **TxProduceIndex** 这两个寄存器定义了下一个将被硬件和软件使用的描述符单元索引。两个寄存器均作为起始为 0 的计数器，当它们到达 **TxDescriptorNumber** 的值时，再返回 0 计数。**TxProduceIndex** 包含了下一个将被软件驱动所填充的描述符索引。**TxConsumeIndex** 包含了下一个将被硬件发送的描述符索引。当 **TxProduceIndex == TxConsumeIndex** 时，发送缓冲区为空。当 **TxProduceIndex == TxConsumeIndex - 1** 时(考虑到它是一个封包设计)，发送缓冲区为满，除非硬件发送了一个或多个帧释放一些描述符，否则软件驱动不能够增加新的描述符。

每个发送描述符都占用存储器中两个字(8 字节)。同样，每个状态区域占用存储器中一个字(4 字节)。每个发送描述符都包括一个指针和一个控制字组成，指针指向发送数据缓冲区，控制字包含的是控制信息。数据包区域有一个 0 地址偏移量，而控制区域则有一个 4

字节的地址偏移量，见[表 201](#)。

表201. 发送描述符的区域

符号	地址偏移量	字节	描述
Packet	0x0	4	包含发送数据的数据缓冲区的基址。
Control	0x4	4	控制信息，见 表 202 。

数据缓冲区指针（**Packet**）是一个 32 位按字节对齐的地址，包含了数据缓冲区的基址。控制字各位的定义见[表 202](#)。

表202. 发送描述符的控制字

位	符号	描述
10:0	Size	数据缓冲区的字节数。这是帧或片段需被 DMA 管理器取出时的字节数。在大多数情况下，该值与由描述符数据包区域指向的数据缓冲区的字节数相等。 Size 的值采用减 1 编码，例如，如果缓冲区为 8 字节，则 Size 的值为 7。
25:11	-	未使用
26	Override	忽略（ override ）每一帧。如果为“真”，则位 30:27 将不考虑 MAC 内部寄存器的默认值。如果为“假”，则位 30:27 将被忽略并使用 MAC 的默认值。
27	Huge	如果为“真”，则使能超长帧，不限制帧长度。如果为“假”，将发送的字节数限制到最大的帧长度（ MAXF[15:0] 的值）。
28	Pad	如果为“真”，将短帧填充到 64 字节。
29	CRC	如果为“真”，将一个硬件 CRC 添加到帧内。
30	Last	如果为“真”，表示这是用于发送帧中最后一个片段的描述符。如果为“假”，则表示应添加来自下一个描述符的片段。
31	Interrupt	如果为“真”，表示当该帧或帧片段中的数据已发送完成，并且相关的状态信息已提交给存储器时，将产生一个 TxDone 中断。

[表 203](#)显示了一个区域的发送状态

表203. 发送状态的区域

符号	地址偏移量	字节	描述
StatusInfo	0x0	4	发送状态的返回标志，见 表 204 。

发送状态由 1 个 **StatusInfo** 字组成。该字包含由 MAC 返回的标志，以及由发送数据通道产生的反映发送状态的标志。[表 204](#)列出了 **StatusInfo** 字各位的定义。

表204. 发送状态信息字

位	符号	描述
20:0	-	未使用
24:21	CollisionCount	这个包遭遇的冲突次数，该值可高达重新发送的最大值。
25	Defer	这个包由于媒体被占据而遭遇延迟。该延迟不是一个错误，除非出现延迟超出限制的情况。
26	ExcessiveDefer	这个包遭遇的延迟超出了最大的延迟限制并被中止。
27	ExcessiveCollision	表示这个包超出了最大的冲突限制并被自助。
28	LateCollision	冲突窗口超出范围，导致发送包中止。
29	Underrun	由于适配器没有产生发送数据而出现 Tx 下溢。
30	NoDescriptor	发送流由于描述符不可用而被中断。

位	符号	描述
31	Error	发送过程中出现的错误。它是 Underrun、LateCollision、ExcessiveCollision、ExcessiveDefer 逻辑相“或”的结果。

对于多片段的帧，LateCollision、ExcessiveCollision、ExcessiveDefer、Defer 以及 CollisionCount 各位的值中，除了帧中最后一个片段以外都将为 0。帧中最后片段的状态将从 MAC 复制这些位的值。所有片段的状态都将有有效的 Error、NoDescriptor 与 Underrun 位。

11.16以太网模块功能描述

本节定义了能够处理 10/100 以太网 MAC 的 DMA 功能。在介绍了以太网模块的 DMA 概念以及基本的发送与接收功能以后，本节还将详细描述流控制、接收过滤等高级特性。

11.16.1 概述

以太网模块通过 MII/RMII 接口与片外 PHY 连接，传送与接收以太网包。MII 或 RMII 模式由软件选择。

以太网模块通常在系统启动过程中初始化。以太网模块的软件初始化应包括对描述符与状态数组的初始化，以及接收器片段缓冲区的初始化。

注：在初始化以太网模块时，重要的是首先配置 PHY，确保参考时钟（RMII 模式下的 ENET_REF_CLK 信号，或 MII 模式下的 ENET_RX_CLK 与 ENET_TX_CLK 信号）在外部管脚上，并连接到 EMAC 模块（用 IOCON 寄存器选择相应管脚），然后再继续以太网的配置。否则，CPU 可能会锁死，不能实现更多功能。如果使用了调试模式，这将使 JTAG 失去与目标的通信。

在发送一个数据包时，软件驱动必须设置相应的 Control 寄存器，以及一个指向数据包缓冲区的描述符，然后才能使 TxProduceIndex 加 1，将数据包发送给硬件。发送以后，硬件将使 TxConsumeIndex 加 1，并可选产生一个中断。

硬件将接收来自 PHY 的数据包，并按软件驱动的配置进行过滤。在接收一个数据包时，硬件将从存储器中读取一个描述符，以找到相应接收器数据缓冲区的位置。接收的数据被写入数据缓冲区，接收状态返回到接收描述符的状态字中。可以选择产生一个中断，告知软件已接收到了一个数据包。注意，DMA 管理器将对高达 3 个描述符进行预取指和缓冲操作。

11.16.2 AHB 接口

以太网模块的寄存器连接到一个 AHB 从接口，允许 CPU 访问寄存器。

AHB 接口有一个仅支持字访问的 32 位数据通道，有 4kB 的地址空间。[表 149](#) 列出了以太网模块的寄存器。

除了访问 IntSet、IntClear 和 IntEnable 寄存器以外，所有 AHB 对寄存器的写访问都具有“加速”模式。AHB 写操作必须按顺序执行。

如果 PowerDown 寄存器的 PowerDown 位置位, 则除了对 PowerDown 寄存器的访问以外, 所有的 AHB 读写访问都将返回一个读错误或写错误。

总线错误

以太网模块对下列情况会产生错误:

- 当对一个只写寄存器执行 AHB 读访问时, AHB 接口将返回一个读错误; 同样, 当对一个只读寄存器做一个 AHB 写访问时, AHB 接口将返回一个写错误。当对保留寄存器做 AHB 读或写访问时, AHB 将返回读错误或写错误。这些错误均传回 CPU。定义为只读和只写的寄存器列在表 149 中。
- 如果 PowerDown 位置位, 则除了对 PowerDown 的访问以外, 对 AHB 寄存器的所有访问都将返回一个错误响应。

11.17 中断

以太网模块有一个中断请求, 输出给 CPU (通过 NVIC)。

中断服务程序必须读取 IntStatus 寄存器, 以决定中断源。所有中断状态都可以由软件写入到 IntSet 寄存器来置位; 软件写入 IntClear 寄存器就可以清零中断状态。

发送与接收数据通道只能将中断状态置位, 而不能将状态清零。SoftInt 中断不能由硬件置位, 而可以被软件用于测试目的。

11.17.1 直接存储器访问 (DMA)

描述符数组

以太网模块包含两个 DMA 管理器。DMA 管理器可在几乎没有处理器支持, 并且无需为每个帧触发一个中断的情况下, 将帧直接传出或传入存储器。

DMA 管理器工作时使用保存在存储器中的帧描述符数组与状态数组。描述符与状态作为以太网硬件与设备驱动软件之间的一个接口。接收帧有一个描述符数组, 而发送帧也有一个描述符数组。通过帧描述符的缓冲功能, 可以减少描述符所占用的存储器通信量与存储器带宽。

每个帧描述符都包括两个 32 位区域: 第一个区域是指向数据缓冲区的指针, 缓冲区含有一个帧或片段, 而第二个区域是一个与该帧或片段相关的控制字。

软件驱动必须在 TxDescriptor/RxDescriptor 和 TxStatus/RxStatus 寄存器中写入描述符与状态数组的基址。每个数组中描述符/状态的数目也必须写入 TxDescriptorNumber/RxDescriptorNumber 寄存器中。一个数组中描述符的数目对应于相关状态数组中的状态数目。

发送描述符数组、接收描述符数组, 以及发送状态数组都必须与一个 4 字节 (32 位) 的地址边界对齐, 而接收状态数组则必须与一个 8 字节 (64 位) 的地址边界对齐。

描述符的拥有权

设备驱动软件与以太网硬件可以同时描述符数组执行读和写操作，从而产生和消耗描述符。一个描述符的“拥有者”可以是设备驱动软件，也可以是以太网硬件。只有描述符的“拥有者”才能读写其值。通常情况下，使用以及拥有描述符与状态的顺序如下：由设备驱动软件拥有并建立描述符；其次，描述符/状态的“拥有权”由设备驱动传给以太网模块，后者读取描述符，将信息写入状态区域；以太网模块将描述符的“拥有权”传回设备驱动，后者使用状态信息，然后重复利用描述符以供其它帧使用。软件必须预先分配好用于保存描述符数组的存储器。

通过将 TxProduceIndex/RxConsumeIndex 寄存器加 1（如果是在数组边界上则返回），软件可以将描述符与状态的拥有权移交给硬件。硬件通过更新 TxConsumeIndex/RxProduceIndex 寄存器，将描述符和状态拥有权移交给软件。

在将一个描述符移交给接收与发送 DMA 硬件以后，设备驱动软件不能修改描述符或不应通过将 TxProduceIndex/RxConsumeIndex 寄存器减 1 回收描述符，因为描述符可能已被硬件预取指了。此时，设备驱动软件必须等待，直到帧被发送完成，或设备驱动必须对发送和/或接收的数据通道进行软复位，从而将描述符数组复位。

连续的顺序及封包设计

当从数组中读取描述符以及将状态写入数组时，这些操作是按连续的顺序并采用的是封包设计（wrap-around）。连续顺序意思是，当以太网模块完成了一个描述符/状态的读写时，下一个要读写位于下一个更高的相邻存储器地址中的描述符/状态。封包意思是，当以太网模块完成了对数组（最高存储器地址）中最后一个描述符/状态的读写时，它将读写位于数组基址的第一个描述符/状态。

描述符数组的满状态与空状态

描述符数组可以为空状态、部分满状态或满状态。当所有描述符均为产生者拥有时，描述符数组为空。如果产生者与消耗者都拥有部分描述符，并且两方都在忙于处理自己的描述符时，描述符数组为部分满状态。当所有描述符（除一个外）均由消耗者拥有时，该描述符数组为满，此时，产生者没有更多空间处理帧信息。描述符的拥有权通过“消耗索引”与“产生索引”来指示。“产生索引”是产生者拥有的第一个元素。它也是下一个要被帧产生者使用的数组元素索引（它可能已使用这个和下一个元素）。“消耗索引”是由消耗者拥有的数组第一个元素。它也是下一个要被帧消费者消耗的数组元素的编号（它和下一个元素都可能正在被消耗处理过程中）。如果“消耗索引”与“产生索引”相等，则描述符数组为空，所有数组元素均由产生者拥有。如果“消耗索引”等于“产生索引”加 1，则数组为满，所有数组元素（除了“产生索引”处的那一个以外）均由消耗者拥有。对于一个满的描述符数组，仍有一个数组元素保持为空，通过查看“产生索引”和“消耗索引”的值，就能够简单地地区分出满和空的状态。一个数组必须至少有 2 个元素，才能表示一个满的描述符数组，此时，“产生索引”值为 0 和“消耗索引”值为 1。当要确定一个描述符数组是否为满时，要考虑到数组的封包因素，因此，如果一个“产生索引”指向数组的最后一个元素，而一个“消耗索引”指向数组的第一个元素，这也意味着描述符数组为满。当“产生索引”与“消耗索引”不相同，并且“消耗索引”不等于“产生索引”加 1 时（考虑了封包设计），则描述符数组为部分满状态，消耗者与产生者均拥有描述符，能够对描述符数

组进行有效操作。

中断位

描述符有一个中断位，由软件编程。当以太网模块在处理一个描述符，并发现该位处于置位状态时，将允许触发一个中断（在将状态提交给存储器后），方法是将 **IntStatus** 寄存器中的 **RxDoneInt** 或 **TxDoneInt** 位传递给中断输出管脚。如果描述符中没有置位中断位，则 **RxDoneInt** 或 **TxDoneInt** 也不置位，不触发中断（注意，**IntEnable** 中的相应位也必须置位才能触发中断）。这样就为描述符数组的管理提供了灵活的方式。例如，设备驱动可以为 **Tx** 描述符数组增加 10 个帧，可将描述符数组中描述符编号 5 的中断位置位。这样可以在发送完编号为 5 的描述符后触发中断，调用中断服务程序。设备驱动程序可以为描述符数组增加另一组帧，而不会中断帧的连续传输。

帧片段

为了获得帧存储的最大灵活性，帧可以被分为多个帧片段，这些片段放在存储器中的不同位置。此时，每个帧片段要使用一个描述符。因此，一个描述符可以指向一个帧，或指向一个帧的某个片段。通过采用片段方式，就可以实现分散/集中 **DMA**，即：发送的帧由存储器中的多个片段集中获得，而接收帧可以分散至存储器中的多个片段。

将多个片段串接起来，就可以在小存储空间创建长帧。片段的另一种用法是能够将帧头和有效载荷分配到不同位置，而在设备驱动中无需复制操作就能将它们连接起来。

在发送时，描述符控制区域中的 **Last** 位指示了该片段是否为一帧的最后一个片段；对接收帧，状态字的 **StatusInfo** 区域中的 **LastFrag** 位指示了片段是否为一帧信息的最后一个片段。如果 **Last (Frag)** 位为 0，则下一个描述符属于相同的以太网帧，如果 **Last (Frag)** 为 1，则下一个描述符属于一个新的以太网帧。

11.17.2 初始化

在复位后，以太网软件驱动要对以太网模块进行初始化。在初始化期间，软件需要完成下列工作：

- 从 **MAC** 中移除软件复位条件；
- 通过 **MAC** 的 **MIIM** 接口配置 **PHY**；
注：重要的是配置 **PHY**，确保基准时钟（**RMII** 模式是 **ENET_REF_CLK** 信号，**MII** 模式是 **ENET_RX_CLK** 与 **ENET_TX_CLK** 信号）在外部管脚上，并连接到 **EMAC** 模块（用 **IOCON** 寄存器选择相应的管脚），然后再继续以太网的配置。否则，**CPU** 可能锁死，不可能完成其它功能。如果使用了调试模式，这可能还会使 **JTAG** 失去与目标的通信。
- 选择 **MII** 或 **RMII** 模式；
- 配置发送与接收 **DMA** 引擎，包括描述符数组；
- 配置 **MAC** 中的主机寄存器（**MAC1**、**MAC2** 等）；
- 使能接收与发送数据通道。

根据 **PHY** 情况，软件需要通过 **MII** 管理接口初始化 **PHY** 中的寄存器。软件可以通过编写

MAC 中的 MCFG、MCMD、MADR 寄存器读写 PHY 寄存器。数据应写入 MWTD 寄存器；数据与状态信息应从 MRDD 和 MIND 寄存器读取。

以太网模块支持 MII 和 RMII PHY。在初始化期间，软件必须通过编写命令寄存器来选择 MII 或 RMII 模式。

在切换到 RMII 模式以前，必须使默认的软件复位（MAC1 寄存器的位 15）失效。在这个操作期间，PH 必须要运行 PHY 时钟，且将其内部连接上。

发送与接收 DMA 引擎应由设备驱动做初始化，方法是在存储器中分配描述符与状态数组。发送与接收功能都有各自专门的描述符与状态数组。这些数组的基址要编写在 TxDescriptor/TxStatus 和 RxDescriptor/RxStatus 寄存器中。一个数组中的描述符数目要与数组中的状态数目相匹配。

请注意，1 个发送描述符、接收描述符与接收状态均为 8 个字节，而 1 个发送状态则是 4 个字节。所有描述符数组与发送状态都必须是 4 字节边界对齐；接收状态必须是 8 字节边界对齐。描述符数组的描述符数目需要采用减 1 编码写入 TxDescriptorNumber/RxDescriptorNumber 寄存器中，即：寄存器中的值等于描述符数目减 1。如果描述符数组有 4 个描述符，则该寄存器的值应为 3。

在建立了描述符数组后，需要为接收描述符分配帧缓冲区，然后才使能接收数据通道。接收描述符的数据包区域需要填充该描述符帧缓冲区的基址。其中，接收描述符中的控制区域需要采用减 1 编码，包含数据缓冲区的大小。

接收数据通道有一个可配置的过滤功能，用于丢弃/忽略指定的以太网帧。过滤功能也应在初始化期间进行配置。

在一个硬件复位有效后，MAC 中的软件复位位将为有效。必须先移除软件复位条件，然后才可以使能以太网模块。

接收功能的使能位于两个位置。一个是接收 DMA 管理器需使能，另一个是 MAC 的接收数据通道需使能。为防止接收 DMA 引擎的溢出，应通过将命令寄存器中的 RxEnable 位置位，使能接收 DMA 引擎，然后再将 MAC1 寄存器中的 RECEIVE ENABLE 位置位，使能 MAC 中的接收数据通道。

在任何时候，都可以通过将命令寄存器中的 TxEnable 位置位，使能传送 DMA 引擎。

在使能数据通道以前，可以编写 MAC 中的多个选项，如自动流控制、发送到接收的回送（用于验证）、全双工/半双工模式等。

未对接收和发送数据通道进行（软）复位时，不能修改描述符数组的基址与描述符数组的大小。

11.17.3 发送过程

概述

本节将简要地描述发送过程。

设备驱动程序建立描述符与数据

如果一个描述符数组已满，设备驱动程序应等待，直到描述符数组成为非满状态，然后再写入描述符数组中的一个描述符。如果描述符数组不为满，则设备驱动程序应使用由 **TxDestructor** 所指数组中第 **TxProduceIndex** 个编号的描述符。

描述符中的 **Packet** 指针设定为指向即将发送的一个数据帧或帧片段。描述符命令区域中的 **Size** 应设置为片段缓冲区的字节数，它使用减 1 编码。其它控制信息可以在描述符的控制区域中指示（**Interrupt**、**Last**、**CRC**、**Pad** 位）。

在写入描述符后，需要使 **TxProduceIndex** 寄存器加 1（考虑封包），将描述符移交给硬件。

如果发送数据通道被禁能，则设备驱动应使能该传送数据通道，方法是设置命令寄存器中的 **TxEnable** 位。

对于多片段发送（不包括最后一个片段），则描述符中的 **Last** 位必须设置为 0；对于最后一个片段，**Last** 位必须设置为 1。当帧已发送，并且发送状态已提交给存储器，要触发一个中断时，要将描述符控制区域中的中断位设为 1。如要让硬件在此以太网帧的帧序控制区域中增加一个 **CRC**，则要设置描述符中的 **CRC** 位。如果软件尚未添加 **CRC**，则必须执行硬件添加 **CRC** 功能。为使能短帧的自动填充以达到最小必需的帧长度，将描述符控制区域中的 **Pad** 位设为 1。在一般应用中，**CRC** 位和 **Pad** 位都设为 1。

设备驱动可以使用 **IntEnable** 寄存器建立中断，以等待来自硬件的一个完成信号，或可以定期检查（轮询）发送的过程。另外，还可以在硬件开始消耗数组开始的描述符时，在描述符数组末尾增加新的帧。

设备驱动将命令寄存器中的 **TxEnable** 位重新置为 0，就可以停止发送过程。发送将不会立即停止；已在发送的帧将被发送完毕，并将状态提交给存储器，然后数据通道失效。设备驱动读取 **Status** 寄存器中的 **TxStatus** 位，就可以监控发送数据通道的状态。

一旦传送数据通道使能，并且相应的 **TxConsumeIndex** 与 **TxProduceIndex** 不相等，即硬件仍然需要处理来自描述符数组的帧，则 **Status** 寄存器中的 **TxStatus** 位将返回“1”（工作状态）。

Tx DMA 管理器读取 Tx 描述符数组

当 **TxEnable** 位置位时，Tx DMA 管理器从 **TxDestructor** 和 **TxConsumeIndex** 确定的存储器地址处读取描述符。请求的描述符数目由硬件拥有的描述符总数所决定，即：**TxProduceIndex** – **TxConsumeIndex**。描述符的块传输能够尽量减少存储器的负荷。从存储器读取返回的数据被缓冲，并根据需求被消耗。

Tx DMA 管理器发送数据

在读取了描述符以后，发送 DMA 引擎从存储器中读取相关的帧数据并发送。在发送完成后，Tx DMA 管理器将状态信息写回到状态区域的 **StatusInfo** 与 **StatusHashCRC** 字中。只有在状态信息已提交给存储器后，**TxConsumeIndex** 的值才被更新，这个状态信息由存储器接口中的一个内部标记协议检验。Tx DMA 管理器继续发送帧，直到描述符数组为空。如果传送描述符数组为空，则 **Status** 寄存器的 **TxStatus** 位将返回到“0”（停止状态）。如描述符数组为空，以太网硬件将置位 **IntStatus** 寄存器中的 **TxFinishedInt** 位。传送数据通道仍是使能的。

Tx DMA 管理器在加载描述符时，会检查描述符控制区域的 **Last** 位。如果 **Last** 位为 0，则表示帧中包含了多个片段。Tx DMA 管理器从主机存储器中收集所有片段，访问一串帧描述符，并将它们作为以太网线路上的一个以太网帧发送出去。当 Tx DMA 管理器发现一个控制区域中 **Last** 位设为 1 的描述符时，就表示这是帧的最后一个片段，于是就找到了帧的结尾。

更新 ConsumeIndex

每次 Tx DMA 管理器将一个状态字提交给存储器时，它就完成了一个描述符的发送，它将 **TxConsumeIndex** 加 1（考虑封包因素），将描述符交还给设备驱动软件。软件在收到硬件的移交后，可以将描述符重新用于新的发送。

设备驱动软件可以跟踪 DMA 管理器的进度，方式是读取 **TxConsumeIndex** 寄存器的值，查看发送过程的进度。当 Tx 描述符数组为空时，**TxConsumeIndex** 寄存器保留其最后一个值。

写发送状态

当通过 MII/RMII 总线发送了帧以后，DMA 管理器更新帧描述符的 **StatusInfo** 字。

如果描述符是用于帧的最后一个片段（或没有划分片段，是对整个帧），则根据帧发送的成功或失败情况，决定是否将状态寄存器中的错误标志（**Error**、**LateCollision**、**ExcessiveCollision**、**Underrun**、**ExcessiveDefer**、**Defer**）置位。**CollisionCount** 区域表示帧发送时遇到的冲突次数，最大值为 MAC 的 **Collision** 窗口/重试寄存器中所编写的 **Retransmission Maximum** 值。

一旦帧中的数据已被 Tx DMA 管理器接收，则除最后一个片段以外，帧中所有片段的状态都将被写入存储器。即使描述符是对一个帧片段，而不是最后一个片段，也会通过 AHB 接口返回错误标志。如果以太网模块在一次（多片段）帧发送期间检测到了一个发送错误，仍会通过 AHB 接口读取该帧的所有其余片段。在一次错误后，余下的发送数据会被以太网模块丢弃。如果在多片段帧的发送期间出现一个错误，则错误状态会重复到帧的最后一个片段。一旦帧中的数据已被 Tx DMA 管理器接收，则除了帧的最后一个片段以外，所有片段的状态都将被写入存储器。如果很早就检测到了错误，则状态中将包括错误信息。只有在以太网线路的发送完成以后，才会写入帧最后一个片段的状态。因此，最后一个片段的状态总能反映出帧中任何地方出现的任何错误。

通过读取 **TSV0** 和 **TSV1** 寄存器，也可以查询上一个帧的发送状态。这些寄存器不会报告

一个单独片段的状态，也不会保存前一个发送帧的信息。它们主要供调试使用的，因为驱动软件与以太网模块之间的通信是通过帧描述符完成的。只要 MAC 的内部状态有效，则状态寄存器也有效，通常当发送与接收过程暂停时，才读取这些状态寄存器。

发送错误处理

如果在发送过程中发生了一个错误，则 Tx DMA 管理器会通过状态数组中写入的发送状态字 (StatusInfo)，以及中断状态寄存器 (IntStatus) 来报告错误。

一般发送错误有几种类型：LateCollision、ExcessiveCollision、ExcessiveDefer、Underrun 和 NoDescriptor。所有这些错误在发送状态字 (StatusInfo) 中都有相应的位。除了 StatusInfo 字中的各个位以外，LateCollision、ExcessiveCollision 和 ExcessiveDefer 相“或”，成为 Status 中的 Error 位。错误位会传递到 IntStatus 寄存器；当出现 LateCollision、ExcessiveCollision、ExcessiveDefer 或 NoDescriptor 错误时，IntStatus 寄存器中的 TxError 位置位；Underrun 错误则在 IntStatus 寄存器中的 TxUnderrun 位中报告。

Underrun 错误可能有以下三种原因：

- 在一个多片段发送中，没有可用的下一个片段。这是一种非致命性错误。NoDescriptor 状态将在前一个片段上返回，IntStatus 中的 TxError 位将会置位。
- 当以太网模块已经开始发送帧时，发送片段的数据不可用。这是一种非致命性错误。Underrun 状态将在传输时返回，IntStatus 中的 TxError 位将会置位。
- 当前一个状态仍然等待通过存储器接口进行传输时，发送状态流停止，并且必须写入一个新的状态。这是一种致命性错误，只能通过硬件的软复位来解决。

第一、二种情况是非致命性的，设备驱动程序必须重新发送帧，或让更高的软件层重新发送帧。在第三种情况下，硬件处于未定义状态，需要将命令寄存器中的 TxReset 位置位以实现软复位。

在报告了 LateCollision、ExcessiveCollision、ExcessiveDefer 或 Underrun 错误以后，错误帧的发送将中止，剩余的发送数据与帧片段被丢弃，并继续发送描述符中的下一帧。

设备驱动程序应获取发送错误信息，并采取必要动作。

发送触发中断

发送数据通道可以产生 4 种不同类型的中断：

- 如果描述符控制区域中的中断位置位，则 Tx DMA 将在发送片段并将相应发送状态提交给存储器后，将 IntStatus 寄存器中的 TxDoneInt 位置位。即使不是最后一个片段的描述符，描述符中的中断位也可以用来生成一个中断。
- 如果当以太网硬件使能时，描述符数组为空，则硬件将置位 IntStatus 寄存器的 TxFinishedInt 位。
- 如果 AHB 接口没有足够高的带宽来消耗发送状态，则当 IntStatus 寄存器中的 TxUnderrun 位置位时，发送可能出现下溢。这是一个致命性错误，需要对发送队列做一个软复位。

- 在出现发送错误（LateCollision、ExcessiveCollision 或 ExcessiveDefer）或对于多片段帧设备驱动程序只提供了初始帧的描述符而不提供帧剩余部分的描述符（即出现 NoDescriptor 错误），或出现一个非致命性溢出错误时，硬件将置位 IntStatus 寄存器中的 TxErrorInt 位。

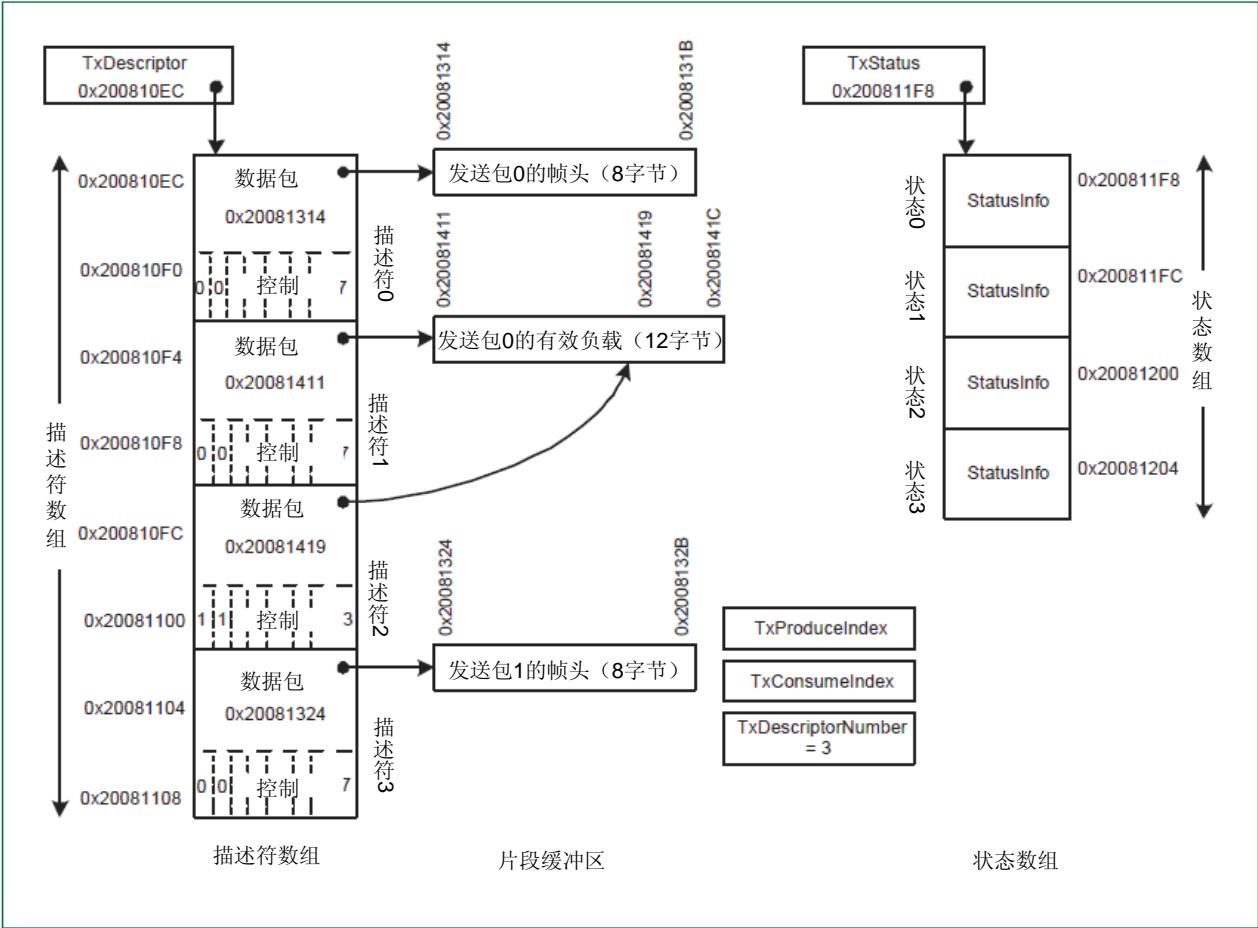
所有以上中断都可以通过置位或复位 IntEnable 寄存器中的相应位来使能或禁能。使能或禁能并不影响 IntStatus 寄存器的内容，而仅是将中断状态传递给 CPU（通过 NVIC）。

中断是 DMA 管理器与设备驱动之间通信的一个良好方式，无论是对单个帧还是整个列表，它可以触发设备驱动程序去检查已处理的描述符状态字。

发送实例

图 24 给出了一个发送过程例子，传送采用的是一个 8 字节帧头和 12 字节的帧有效负载。

图24. 发送示例的存储器和寄存器



复位后，DMA 寄存器的值将为 0。在初始化期间，设备驱动将在存储器中分配描述符和状态数组。在本例中，分配的是 4 个描述符的数组；数组为 4x2x4 字节，是 4 字节的地址边界对齐。由于描述符数目与状态数目相匹配，因此状态数组包含 4 个元素；状态数组为 4x1x4

字节，是 4 字节的地址边界对齐。设备驱动将描述符数组的基址（0x2008 10EC）写入 **TxDestructor** 寄存器中，将状态数组的基址（0x2008 11F8）写入 **TxStatus** 寄存器中。设备驱动程序将描述符与状态数目减 1 后（3），写入 **TxDestructorNumber** 寄存器。数组中的描述符与状态不需要初始化。

此时，可以置位命令寄存器中的 **TxEnable** 位，使能发送数据通道。如果发送数据通道使能时，没有更多需要发送的帧，则 **TxFinishedInt** 中断标志将置位。为减少处理器的中断负荷，可以置位 **IntEnable** 寄存器中的相应位，从而只使能那些需要的中断。

现在，假设应用软件希望用 TCP/IP 协议发送一个 12 字节的帧（实际应用中的帧将大于 12 字节）。TCP/IP 栈将给帧加一个帧头。在存储器中，帧头并不需要紧接在有效载荷的前面。设备驱动可以对 **Tx DMA** 编程，将帧头与有效载荷数据集中起来。为此，设备驱动编写第一个描述符，使之指向帧头；描述符的 **Last** 标志将设为 **false/0**，表示是一个多片段的发送。设备驱动程序编写下一个描述符，使之指向实际的有效负载数据。一个有效负载缓冲区的最大容量为 **2kB**，因此一个描述符就足够用来描述有效负载缓冲区。为了举例，有效负载被分配在两个描述符上。数组中的第一个描述符用于描述帧头，数组中的第二个描述符则用于说明有效负载的前 8 个字节；数组中的第三个描述符用于说明帧的剩余 4 个字节。在第三个描述符中，控制字中的 **Last** 位被设为 **true/1**，表示它是帧中的最后一个描述符。在此例中，描述符控制区域中的中断位在帧的最后一个片段，从而在发送完成时触发一个中断。描述符控制字的 **Size** 为片段缓冲区中的字节数，采用的是减 1 编码。

注意，在实际的设备驱动程序中，有效负载通常只在大于 **2kB** 时才分布到多个描述符中。另外需要注意的是，发送的有效负载数据是直接发给硬件，而不经设备驱动程序的复制（零复制设备驱动）。

在建立了描述符以后，由于设备驱动程序已经对 3 个描述符进行了编程，会将 **TxProduceIndex** 寄存器加 3。如果在初始化期间没有使能发送数据通道，则设备驱动程序需要立即使能该通道。

如果发送数据通道已使能，以太网模块一旦检测到 **TxProduceIndex** 不等于 **TxConsumeIndex**（两者在复位后均为 0），就会开始发送帧。**Tx DMA** 开始从存储器中读取描述符。存储系统将返回描述符，以太网模块会逐一地接收这些描述符，同时读取发送数据的片段。

一旦从存储器中返回了发送的读数据，以太网模块就会尝试通过 **MII/RMII** 接口，在以太网线路上启动发送。

在发送了一帧的每个片段以后，**Tx DMA** 会将片段的发送状态写入状态数组。一旦帧中数据被 **Tx DMA** 管理器接受，则除了帧中的最后一个片段以外，所有片段的状态都要被写入。只有在以太网线路的发送全部完成后，才写入帧最后一个片段的状态。

由于最后一个片段描述符中的中断位置位，在将最后一个片段的状态提交给存储器后，以太网模块会触发一个 **TxDoneInt** 中断，它触发设备驱动去检查状态信息。

在本例中，只要以太网模块还没有将 **TxConsumeIndex** 加 1，设备驱动程序就不能增加新的描述符，因为描述符数组已满（尽管还有一个描述符尚未被编写）。只有在硬件将第一个片段的状态提交给存储器，并且 **DMA** 管理器将 **TxConsumeIndex** 设为 1 以后，设备驱动

程序才能够编写下一个（第 4 个）描述符。第 4 个描述符可以在第一帧发送完成以前进行设置。

本例中，硬件会为帧添加 CRC。如果由设备驱动软件添加 CRC，则 CRC 帧尾可以被看作另一个帧片段，它可以在进行另一个集中 DMA 时添加。

每个数据字节都是通过 MII 接口以两个 4 位值发送，或通过 RMII 接口以四个 2 位值发送。以太网模块会加上导言、帧头定界符，如果使能了硬件 CRC，还会加上 CRC 帧尾。一旦 MII/RMII 接口上的发送开始，除非产生下溢错误，否则就不能中断，这就是为什么描述符与数据读取命令要尽可能快地发送和传输。

使用 MII PHY 时，以太网模块与 PHY 之间的数据通信操作频率为 25MHz。使用 RMII PHY 时，以太网模块与 PHY 之间的数据通信频率为 50MHz。在 10Mbps 模式下，数据只能每 10 个时钟周期发送一次。

11.17.4 接收过程

本节概述了接收的过程，包括设备驱动软件的活动。

设备驱动软件建立描述符

在初始化了接收描述符与状态数组，使之能从以太网连接接收帧以后，应在 MAC1 寄存器和控制寄存器中使能接收数据通道。

在初始化期间，描述符中每个指针（Packet 区域）设置为指向数据片段缓冲区。缓冲区的大小为描述符控制区域中 Size 位的值。另外，描述符中的控制区域有一个中断位。中断位能够在一片段缓冲区被填满，并将其状态提交给存储器之后产生中断。

在初始化并使能了接收数据通道以后，接收硬件就拥有了所有描述符，并且不可以通过软件来修改，硬件将 RxProduceIndex 加 1（这表示已接收了一个帧）后，便将描述符的拥有权移交给设备驱动程序。设备驱动程序可在（软）复位接收数据通道后修改描述符。

Rx DMA 管理器读取 Rx 描述符数组

当命令寄存器中的 RxEnalbe 位置位时，Rx DMA 管理器从存储器中读取描述符，存储器地址由 RxDescriptor 和 RxProduceIndex 决定。即使 MII/RMII 接口上的实际接收数据尚未到达，以太网模块也会开始读取描述符（描述符预取）。被读取的描述符的大小由硬件拥有的描述符总数决定： $RxConsumeIndex - RxProduceIndex - 1$ 。描述符的块传输可降低存储器负担。从存储器返回的数据根据需求进行缓冲和消耗。

RX DMA 管理器接收数据

读取描述符后，接收 DMA 引擎等待 MAC 从 MII/RMII 接口接收通过接收滤波器返回的数据。不符合过滤标准的接收帧不会保存到存储器。一旦某个帧通过了接收滤波器，数据就被写入描述符相关的片段缓冲区中。Rx DMA 的写操作不会超过缓冲区的容量。当接收的帧超过一个描述符的片段缓冲区时，该帧将写入到连续描述符的多个片段缓冲区内。对于多片段接收的情况，除帧内的最后一个片段以外，所有片段都将返回一个状态，其中 LastFrag

位被置为 0。只有对一个帧的最后一个片段，状态中的 **LastFrag** 位才会置为 1。帧信息的最后一个片段数据可能不会填满一个缓冲区，再接收其它帧时，会将接收数据写入下一个描述符的片段缓冲区内。

接收一个片段后，**Rx DMA** 管理器会将状态信息写回到状态数组的 **StatusInfo** 与 **StatusHashCRC** 字中。以太网模块将描述符片段缓冲区的字节数写入状态字的 **RxSize** 中。只有在片段数据与片段状态信息均已提交给存储器以后，才会更新 **RxProduceIndex** 的值，检查方法是使用存储器接口的一个内部标记协议。**Rx DMA** 管理器继续接收各个帧，直到描述符数组为满。如果描述符数组为满，则以太网硬件将置位 **IntStatus** 寄存器的 **RxFinishedInt** 位。接收数据通道仍是使能状态。如果接收描述符数组为满，则任何新接收到的数据都会产生一个溢出错误和中断。

更新 ProduceIndex

每当 **Rx DMA** 管理器向存储器提交一个数据片段以及相关的状态字时，它就完成了一个描述符的接收，将 **RxProduceIndex** 加 1（考虑封包因素），从而将描述符移交给设备驱动软件。当接收的数据已经处理时，软件通过将描述符移交给硬件使得描述符能够重新用于新的接收操作。

设备驱动软件可以跟踪 **DMA** 管理器的进度，方法是读取 **RxProduceIndex** 寄存器，看接收处理进行的程度。当 **Rx** 描述符数组为空时，**RxProduceIndex** 保留其最后的值。

写接收状态

当从 **MII/RMII** 总线接收了帧以后，**DMA** 管理器会更新帧描述符中的 **StatusInfo** 和 **StatusHashCRC** 字。

如果描述符是用于一帧的最后一个片段（如果没有片段，用于整个帧），根据帧接收的成功与否，在 **StatusInfo** 中设置错误标志（**Error**、**NoDescriptor**、**Overrun**、**AlignmentError**、**RangeError**、**LengthError**、**SymbolError** 或 **CRCErr**）。**RxSize** 区域的值为实际写入到片段缓冲区的字节数，采用的是减 1 编码。如果不是帧中最后一个片段，则 **RxSize** 将与缓冲区大小相匹配。对所有属于相同数据包的帧来说，目标地址与源地址的 **Hash CRC** 均计算一次，然后保存在与对应片段相关的状态字 **StatusHashCRC** 中。如果接收报告了一个错误，则接收帧中的任何其余数据均被丢弃，接收状态字段中的 **LastFrag** 位置位，于是，除最后一个片段以外，一帧中所有片段的错误标志将始终为 0。

最后一个接收帧的状态也可以由读取 **RSV** 寄存器来查询。该寄存器不会报告每个片段的状态，也不会保存以前所接收帧的信息。**RSV** 主要用于调试，因为驱动软件与以太网模块之间的通信要通过帧描述符实现。

接收错误处理

当接收过程中出现一个错误时，**Rx DMA** 管理器会通过接收 **Status** 寄存器中的 **StatusInfo** 以及 **IntStatus** 中断状态寄存器报告错误。

接收过程可以产生多种类型的错误：**AlignmentError**、**RangeError**、**LengthError**、**SymbolError**、**CRCErr**、**Overrun** 与 **NoDescriptor**。所有这些在接收 **StatusInfo** 中都有相应的

位。除了 **StatusInfo** 中有单独的位以外，**StatusInfo** 中的 **Error** 位还是 **AlignmentError**、**RangeError**、**LengthError**、**SymbolError** 和 **CRCErr** 相“或”的结果。错误亦会传递给 **IntStatus** 寄存器；如果有 **AlignmentError**、**RangeError**、**LengthError**、**SymbolError**、**CRCErr** 或 **NoDescriptor** 错误，则 **IntStatus** 寄存器中的 **RxError** 位置位；非致命性超时错误报告在 **IntStatus** 寄存器的 **RxError** 位；致命性超时错误报告在 **IntStatus** 寄存器的 **RxOverrun** 位。出现致命性溢出错误时，需要通过置位命令寄存器中的 **RxReset** 位进行软复位。

溢出错误有下列三种原因：

- 在多片段接收情况下，下一个描述符丢失。此时，前一个描述符的状态字中的 **NoDescriptor** 置位，**IntStatus** 寄存器中的 **RxError** 位也置位。这个错误是非致命性的。
- 接收器数据接口上数据流暂停，数据包损坏。此时，状态字中的溢出位置位，**IntStatus** 寄存器中的 **RxError** 位也置位。这个错误是非致命性的。
- 当前一个状态仍等待通过存储器接口进行传输的同时，接收状态流暂停并且必须要写入一个新的状态。这个错误会破坏硬件状态，需要硬件进行软复位。这个错误被检测并将 **IntStatus** 寄存器中的溢出位置位。

第一种溢出条件将导致帧信息的不完整，**NoDescriptor** 状态与 **IntStatus** 中的 **RxError** 位置位。软件应丢弃部分接收的帧。第二种溢出条件下，帧数据会被损坏，从而使状态字中的溢出状态位置位，同时 **IntError** 中断位也置位。在第三种情况下，接收错误不能在接收状态数组中报告，它会破坏硬件状态；该错误还将在 **IntStatus** 寄存器的溢出位报告。此时，应使用命令寄存器中的 **RxReset** 位对硬件进行软复位。

设备驱动软件应获取上述接收错误，并采取动作。

接收触发中断

接收数据通道可以产生 4 种不同类型的中断：

- 如果描述符控制区域的中断位置位，则在接收了一个片段，并将相关数据与状态提交给存储器后，**Rx DMA** 将置位 **IntStatus** 寄存器中的 **RxDoneInt** 位。即使某个描述符（片段）不是一个多片段帧中的最后一个，也可以用描述符中的中断位产生一个中断。
- 如果描述符数组为满时以太网硬件仍是使能，则硬件会置位 **IntStatus** 寄存器中的 **RxFinishedInt** 位。
- 如果 AHB 接口不以一个足够高的带宽消耗接收状态，则接收状态将溢出，此时 **IntStatus** 寄存器中的 **RxOverrun** 位将置位。
- 如有出现接收错误（**AlignmentError**、**RangeError**、**LengthError**、**SymbolError** 或 **CRCErr**）或者对多片段帧设备驱动软件只提供初始片段的描述符而不提供剩余片段的描述符，或者发生一个非致命性数据溢出时，硬件会将 **IntStatus** 寄存器中的 **RxErrorInt** 位置位。

所有上述中断都可以通过置位或复位 **IntEnable** 寄存器中的相应位使能或禁能。使能或禁能并不影响 **IntStatus** 寄存器的内容，而只影响向 CPU 传递中断的状态（通过 NVIC）。

无论是单个帧引起的中断还是整个序列引起的中断，它们都是 **DMA** 管理器与设备驱动软件

之间通信的良好方式，可以触发设备驱动软件去查询已处理的描述符的状态字。

设备驱动程序处理接收数据

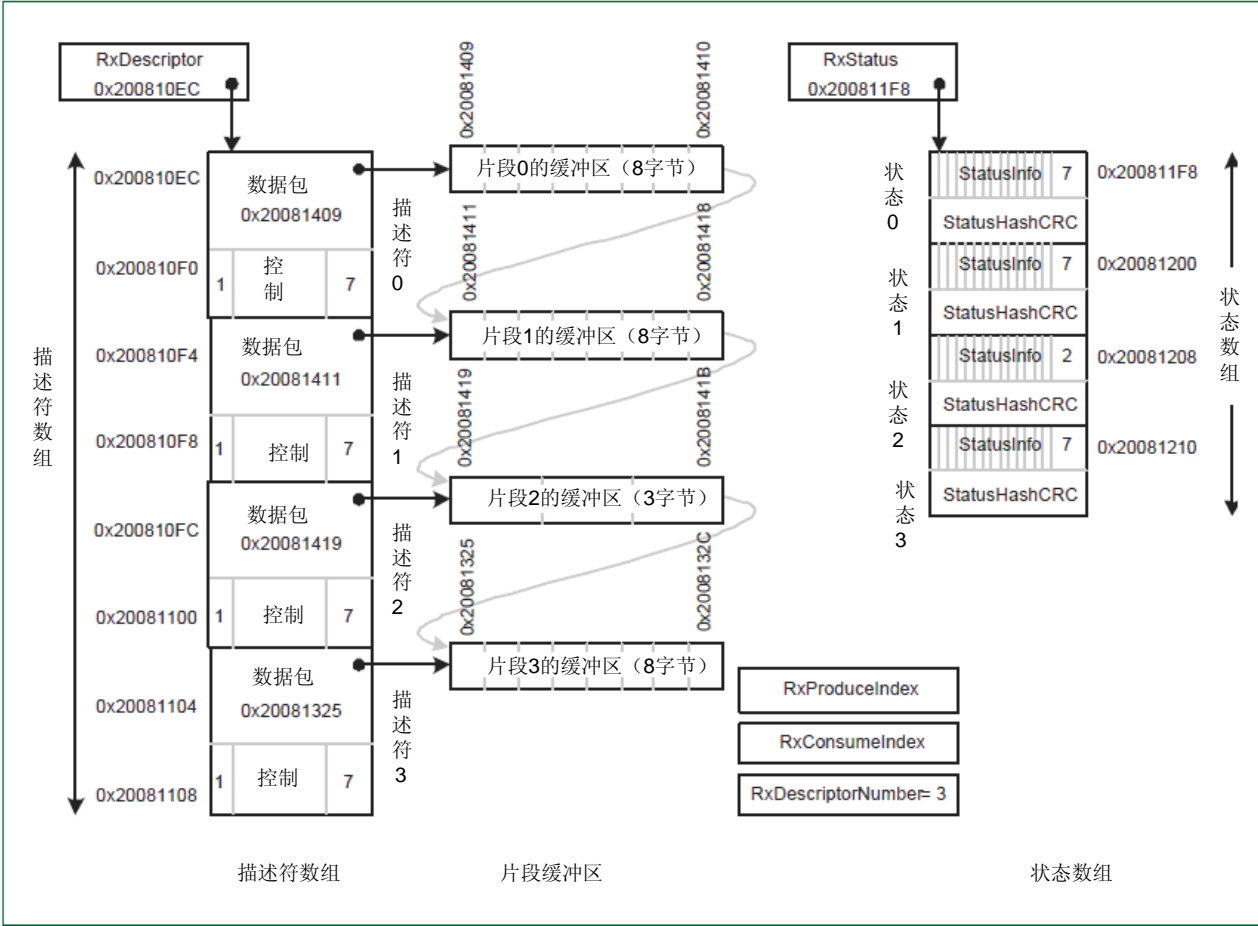
作为对状态中断（如 RxDoneInt）或对 RxProduceIndex 的轮询的一个响应，设备驱动可以读取已被硬件移交的描述符（RxProduceIndex—RxConsumeIndex）。设备驱动程序应检查状态数组中的状态字，看是否有多片段接收与接收错误。

设备驱动可以将接收数据与状态转发给上层的软件。在处理了数据与状态后，描述符、状态与数据缓冲区都可以循环使用，并通过将 RxConsumeIndex 加 1 移交给硬件。

接收示例

图 25 通过一个接收示例来阐明接收过程，该示例中，一帧信息为 19 字节。

图25. 接收示例的存储器和寄存器



复位后，DMA 寄存器的值将为 0。在初始化期间，设备驱动将在存储器中分配描述符及状态数组。在本例中，分配的是一个 4 个描述符的数组；数组为 4x2x4 字节，是 4 字节的地址边界对齐。由于描述符的数目与状态数目相匹配，因此状态数组包含了 4 个元素；数组

为 4x2x4 字节，是 8 字节的地址边界对齐。设备驱动将描述符数组的基址（0x2008 10EC）写入 RxDescriptor 寄存器，并将状态数组的基址（0x2008 11F8）写入 RxStatus 寄存器。设备驱动软件在 RxDescriptorNumber 寄存器中写入描述符与状态数量减 1（3）。数组中的描述符与状态不需要初始化。

描述符分配以后，需要为每个描述符分配一个片段缓冲区。每个片段缓冲区可以在 1 字节至 2k 字节之间。片段缓冲区的基址存放在描述符的 Packet 区域。片段缓冲区的字节数存放在描述符控制字的 Size 区域。描述符控制字的中断区域可以设定为：一旦描述符被接收过程填充时，就产生一个中断。在本例中，片段缓冲区为 8 字节，因此描述符控制字区域的 Size 值为 7。注意，在本例中，片段缓冲区实际上是一个连续的存储器空间；即使帧信息被分布在多个片段上时，它通常也是在一个线性的连续存储器空间上；当描述符在描述符数组的末尾对其封包（wrap）时，帧就不占用连续的存储器空间了。

设备驱动应通过向命令寄存器的 RxEnable 位写入“1”，使能接收过程，然后需要向 MAC 配置寄存器的“RECEIVE ENABLE”位写入“1”，使能 MAC。现在，以太网模块就可以开始接收以太网帧了。为了减少处理器的中断负担，可以通过置位 IntEnable 寄存器中的相应位，禁能某些中断。

Rx DMA 管理器使能后，它会开始发出描述符读命令。在本例中，描述符的数目为 4。开始时，RxProduceIndex 和 RxConsumeIndex 都是 0。当 RxProduceIndex = RxConsumeIndex - 1 时，则认为描述符数组已满，Rx DMA 管理器只能读（RxConsumeIndex - RxProduceIndex - 1 =）3 个描述符；要注意封包因素。

使能了 MAC 中的接收功能以后，数据接收将在下个帧开始进行，即，如果接收功能使能时，MII/RMII 接口正在接收一帧信息，则该帧将被丢弃，接收从下一帧开始。以太网模块将从帧中删除导言和帧起始定界符。如果帧通过了接收过滤，则 Rx DMA 管理器将开始将帧写入第一个片段缓冲区。

假设帧长度为 19 字节。考虑到本例中设定的缓冲区大小，帧将被分布到 3 个片段缓冲区上。在第一个片段缓冲区中写入初始 8 字节后，将写入第一个片段缓冲区的状态，RxDMA 继续填充第二个片段缓冲区。由于这是一个多片段接收，第一个片段的状态在 StatusInfo 字的 LastFrag 位上为 0；而 RxSize 字段则设为 7（8 - 1，采用减 1 编码）。在第二个片段写入 8 字节后，Rx DMA 将继续对第三个片段执行写操作。第二个片段的状态与第一个片段的状态类似：LastFrag = 0，RxSize = 7。在将 3 个字节写入第三个片段缓冲区之后，帧信息到达末尾，接着写入第三个片段的状态，其状态为：LastFrag 位设为 1 且 RxSize 等于 2（3，采用减 1 编码）。

从 MII/RMII 接口接收的下一个帧将写入第四个片段缓冲区，即第三个缓冲区中的 5 个字节留空未用。

Rx DMA 管理器采用存储器接口上的一种内部标记协议，以检查接收的数据与状态是否已被提交给存储器。片段的状态被提交给存储器后，就会触发一个 RxDoneInt 中断，它激活设备驱动软件，去检查状态信息。在本例中，所有描述符的控制字中都置位了中断位，即，在向存储器提交了数据和状态后，所有描述符都会产生一个中断。

本例中，在设备驱动软件没有将 RxConsumeIndex 加 1 以前，接收功能不可以读取新的描述符，因为描述符数组已满（即使还有一个描述符尚未编写）。只有在设备驱动软件将接收

的数据移交给应用软件后，并且设备驱动软件更新了 `RxConsumeIndex`（加 1），以太网模块才可以继续读取描述符以及接收数据。设备驱动软件有可能将 `RxConsumeIndex` 加 3，因为驱动软件会将包含有 3 个片段的整个帧移交给应用，因此同时释放了 3 个描述符。

由于接收滤波器的过滤与缓冲模块的延时（128 或 136 个周期），在 MII 接口上传输的每一对半字节（或在 RMII 接口上传输的 4 对位）都会在数据写接口上以一个字节的方式传输。以太网模块会从数据中删除导言、帧起始定界符以及 CRC，并校验 CRC。为了减少出现缓冲区的 `NoDescriptor` 错误的可能性，3 个描述符都要进行缓冲。只有当状态信息被提交给存储器后，`RxProduceIndex` 的值才更新，它可以由存储器接口的一个内部标记协议来检验。软件设备驱动将处理接收的数据，然后，设备驱动会更新 `RxConsumeIndex`。

对于 RMII PHY，以太网模块与 PHY 之间的数据通信是占一半数据带宽，两倍时钟频率（50 MHz）。

11.17.5 发送重试

如果在以太网上发生冲突，通常会出现一个帧前 64 字节的冲突窗口内。如果检测到了冲突，则以太网模块会重试发送。为此，帧的前 64 字节被缓冲，于是可以在重试期间使用这个数据。对应用和设备驱动软件来说，一个帧前 64 字节的发送重试是完全透明的。

当冲突发生在 64 字节冲突窗口之外时，会触发一个 `LateCollision` 错误，并且发送操作中止。在出现 `LateCollision` 错误之后，发送帧中余下的数据被丢弃。以太网模块将帧中状态区域里的 `Error` 和 `LateCollision` 位置位。`IntStatus` 寄存器中的 `TxError` 位也将置位。如果 `IntEnable` 寄存器中的相应位置位，则 `IntStatus` 寄存器中的 `TxError` 位被传给 CPU（通过 NVIC）。设备驱动软件应获取该中断，并采取相应的行动。

`CLRT` 寄存器中的“`RETRANSMISSION MAXIMUM`”字段可以用于配置重试的最大次数，超过此次数后将放弃发送。

11.17.6 状态 Hash CRC 计算

对于每个接收到的帧，以太网模块都能够探测出目标地址与源地址，并从它们计算出相应的 Hash CRC。在计算时，以太网模块会采用两个内部模块：一个是与每帧开始与结尾同步的控制器，第二个是 CRC 计算器。

当检测到一个新帧时，内部信号会通知控制器。控制器开始对与目标地址的字节对应的输入帧字节进行计数。当计数到第 6 个（也是最后一个）字节时，控制器通知计算器将相应的 32 位 CRC 保存在第一个内部寄存器中。然后，控制器重复地统计下一个输入的字节，从而与源地址同步。当统计完源地址的最后字节时，控制器再次通知 CRC 计算器停止工作到下一个新帧。当计算器接收到第二个通知时，它就将当前的 32 位 CRC 保存在第二个内部寄存器中。然后，各自寄存器中的 CRC 均保持不变，直到收到新的通知。

写入 `StatusHashCRC` 字的目标地址与源地址 Hash CRC 是由 CRC 计算器计算得到的 32 位 CRC 的 9 个最高有效位。

11.17.7 双工模式

以太网模块有全双工模式和半双工模式。半双工或全双工模式需要由设备驱动软件在初始化过程中配置。

对于一个全双工连接，需要将命令寄存器中的 **FullDuplex** 位设置为 1，并且 **MAC2** 配置寄存器的 **FULL-DUPLEX** 位设置为 1；对于半双工，需将这两个位设置为 0。

11.17.8 IEEE 802.3/条款 31 流控制

概述

对于全双工连接，以太网模块遵循“IEEE 802.3/条款 31”中的规定，使用“暂停帧”进行流控制。这种流控制可用于全双工的点对点连接。流控制可以使接收器暂停发送器，如当接收缓冲区（几乎）为满的情况。为此，接收器会向发送器发出一个“暂停帧”。

暂停帧使用 512 个位时间单元，相当于 128 个 rx_clk/tx_clk 周期。

接收流控制

在全双工模式下，以太网模块接收到“暂停帧”时，会将发送操作挂起。**Rx** 流控制由发送操作的接收端发起。通过置位 **MAC1** 配置寄存器中的“**RX FLOW CONTROL**”位来使能 **Rx** 流控制。如果“**RX FLOW CONTROL**”位为 0，则以太网模块会忽略接收到的暂停控制帧。当以太网模块的接收端接收到暂停帧时，发送端的发送操作将在当前发送帧发送完毕后被中断，中断时间由接收到的暂停帧的时间总量决定。发送数据通道将在 512 个位槽（bit slot）时间内停止发送数据，暂停时间由接收到的暂停控制帧的暂停定时器区域设置。

默认情况下，不将接收到的暂停控制帧传递到设备驱动程序。如要将接收到的流控制帧传递给设备驱动程序，则要置位 **MAC1** 配置寄存器中的“**PASS ALL RECEIVE FRAMES**”位。

发送流控制

如果设备驱动需要停止接收数据（如因为软件缓冲区已满），则以太网模块可以发送暂停控制帧。发送流控制需要由设备驱动软件发起；没有由硬件（如 **DMA** 管理器）发起的 **IEEE 802.3/31** 流控制。

通过软件流控制，设备驱动软件可以根据实际情况，决定是否需要通过发出 **Tx** 暂停帧来中断帧的接收进程。注意，由于以太网有延迟，因此在流控制生效，并且接收流停止以前，仍然会收到一些帧。

发送流控制是向命令寄存器的 **TxFlowControl** 位写入 1 来激活。当以太网模块工作在全双工模式时，这将导致 **IEEE 802.3/31** 暂停帧的发送。流控制会持续生效，直到向命令寄存器的 **TxFlowControl** 位写入 0 为止。

如果 **MAC** 工作在全双工模式，则置位命令寄存器 **TxFlowControl** 位会开始发送一个暂停帧。所发送暂停帧的暂停定时器区域中的值通过 **FlowControlCounter** 寄存器的 **PauseTimer[15:0]** 位来设置。当 **TxFlowControl** 位无效时，会自动发送另外一个暂停定时器

值为 0x0000 的暂停帧，以中止流控制并恢复发送操作。

如果必须延长流控制时间，就必须发送一个暂停帧序列。利用镜像计数器机制可以支持该操作。要使能镜像计数，可在 FlowControlCounter 寄存器的 MirrorCounter[15:0]位中写入一个非零的值。当 TxFowControl 位有效时，发送一个暂停帧。暂停帧发送完成后，内部镜像计数器初始化为 0。内部镜像计数器每隔 512 个位槽（bit slot）时间加 1。当内部镜像计数器达到 MirrorCounter 值时，发送另一个暂停帧，其暂停定时器值等于 FlowControlCounter 寄存器的 PauseTimer 字段值，内部镜像计数器复位为 0，重新开始计数。寄存器 MirrorCounter[15:0]的设定值通常要小于寄存器 PauseTimer[15:0]的值，以确保镜像计数器更早到达计算终点，这样，就可以在另一方恢复发送操作以前，有时间先发送一个新的暂停帧。在发送方完成暂停定时器的计数前，持续发送暂停帧，就可以延续暂停，只要 TxFowControl 一直有效。这种延续可以一直持续到 TxFowControl 无效时，此时，会自动发送一个暂停定时器值为 0x0000 的最后一个暂停帧，中止流控制并恢复发送操作。如要禁能镜像计数器机制，可向 FlowControlCounter 寄存器中的 MirrorCounter 位写入 0。在使用镜像计数器机制时，要保守地编写 MirrorCounter，要考虑飞行时间（time-of-flight）延迟、帧发送时间、队列延迟、晶体频率公差，以及响应时间延迟等，一般 MirrorCounter 值是 PauseTimer 值的约 80%。

如果软件设备驱动将 FlowControlCounter 寄存器的 MirrorCounter 字段的值设为 0，则以太网模块将只发送一个暂停控制帧。在发送了暂停帧以后，内部暂停计数器初始化为 0；内部暂停计数器每 512 个位槽（bit slot）的时间执行加 1 操作。一旦内部暂停计数器达到了 PauseTimer 寄存器的值，命令寄存器中的 TxFowControl 位将复位。软件设备驱动可以轮询 TxFowControl 位，检测暂停何时完成。

流控制模块中的内部计数器值可以通过 FlowControlStatus 寄存器读取。如果 MirrorCounter 非零，则 FlowControlStatus 寄存器将返回内部镜像计数器的值；如果 MirrorCounter 为 0，则 FlowControlStatus 寄存器将返回内部暂停计数器的值。

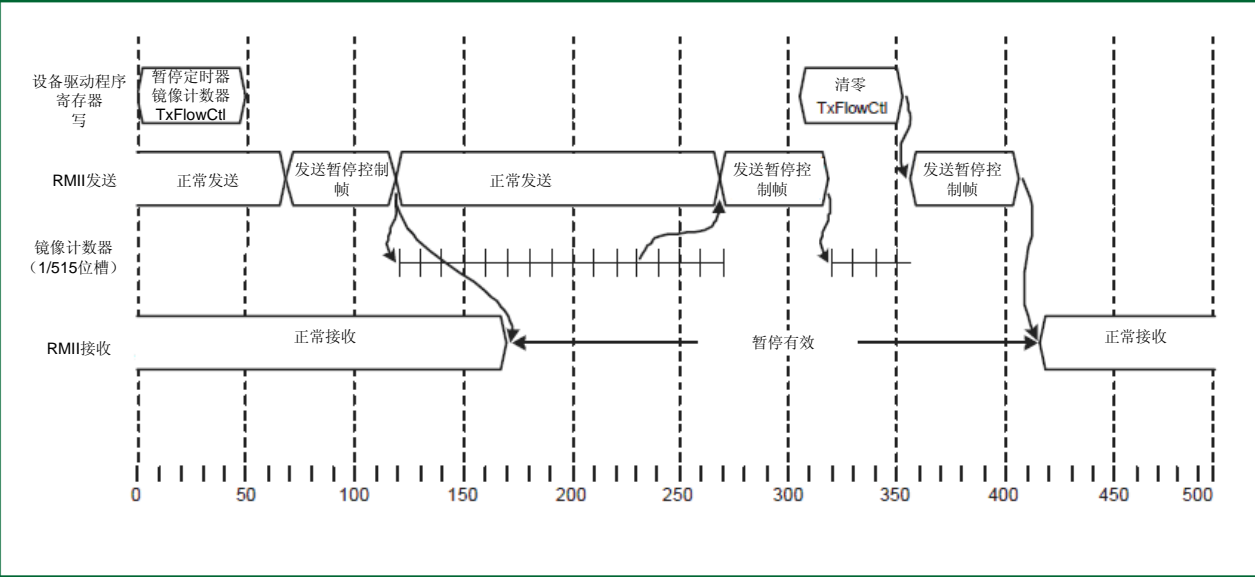
设备驱动软件能够动态地修改 MirrorCounter 寄存器的值，并在零 MirrorCounter 模式和非零 MirrorCounter 模式之间切换。

发送流控制的使能是通过置位 MAC1 配置寄存器中的“TX FLOW CONTROL”位。如果“TX FLOW CONTROL”位为 0，则 MAC 就不会发送暂停控制帧，软件不能启动暂停帧的发送，并且命令寄存器中的 TxFowControl 位应为 0。

发送流控制实例

[图 26](#) 所示为发送流控制。

图26. 发送流控制



本例中，当发送一个帧时，也接收一个帧（全双工模式）。设备驱动软件检测到有些缓冲区可能溢出，并通过编写 **FlowControlCounter** 寄存器的 **PauseTimer** 和 **MirrorCounter** 字段使能发送流控制，然后，它通过置位命令寄存器中的 **TxFlowControl** 位，使能传送流控制。

作为对流控制使能的响应，在当前发送帧完成以后，会发送一个暂停控制帧。当暂停帧发送完成时，内部镜像计数器将开始统计位槽（bit slot）；一旦计数器达到了 **MirrorCounter** 字段中的值，则发送另一个暂停帧。在计数过程中，发送数据通道会继续正常的发送操作。

一旦软件禁用了发送流控制，则会发送一个零值暂停控制帧，恢复接收过程。

11.17.9半双工模式背压

当采用半双工模式时，可以生成背压来暂停接收数据包，方法是发送连续的导言，它基本上阻塞了以太网媒体上的其它传输。当以太网模块在半双工模式下工作时，命令寄存器中的 **TxFlowControl** 位有效，即可在以太网线路上施加连续的导言，实际上就阻止了相同网段上其它以太网站的通信。

在半双工模式下，当 **TxFlowControl** 位为高电平时，就会发送连续的导言，直到 **TxFlowControl** 位失效。如果媒体空闲，则以太网模块开始发送导言，从而唤起载波侦听，使所有其它站都推迟发送。当前导言的发送产生一个冲突时，背压会“穿越”(ride through) 冲突。发生冲突的站将后退，然后遵循背压。如果在背压期间，用户希望发送一个帧，则可以将背压中止，发送帧以后再恢复背压。如果 **TxFlowControl** 的有效时间长于 3.3ms（在 10Mbps 模式中）或 0.33ms（在 100Mbps 模式中），背压将停止发送导言并持续几个字节的时间，以避免超出 jabber 限制。

11.17.10 接收过滤

接收过滤的特性

以太网 MAC 有多个接收数据包过滤功能，可以由软件驱动进行下列配置：

- 完全地址过滤：能识别出与站地址完全匹配的包，并传递到软件驱动程序中。
- Hash 表过滤：对基于站地址的数据包进行不完全过滤。
- 单播/多播/广播包过滤：允许传递所有单播、多播和/或广播的数据包。
- 魔法包过滤：检测魔法包，产生一个“LAN 上唤醒”中断。

过滤功能可以进行逻辑组合，从而产生复杂的过滤功能。此外，以太网模块可以传递或抵制小于 64 字节的短包；混合模式则允许所有数据包传递到软件中。

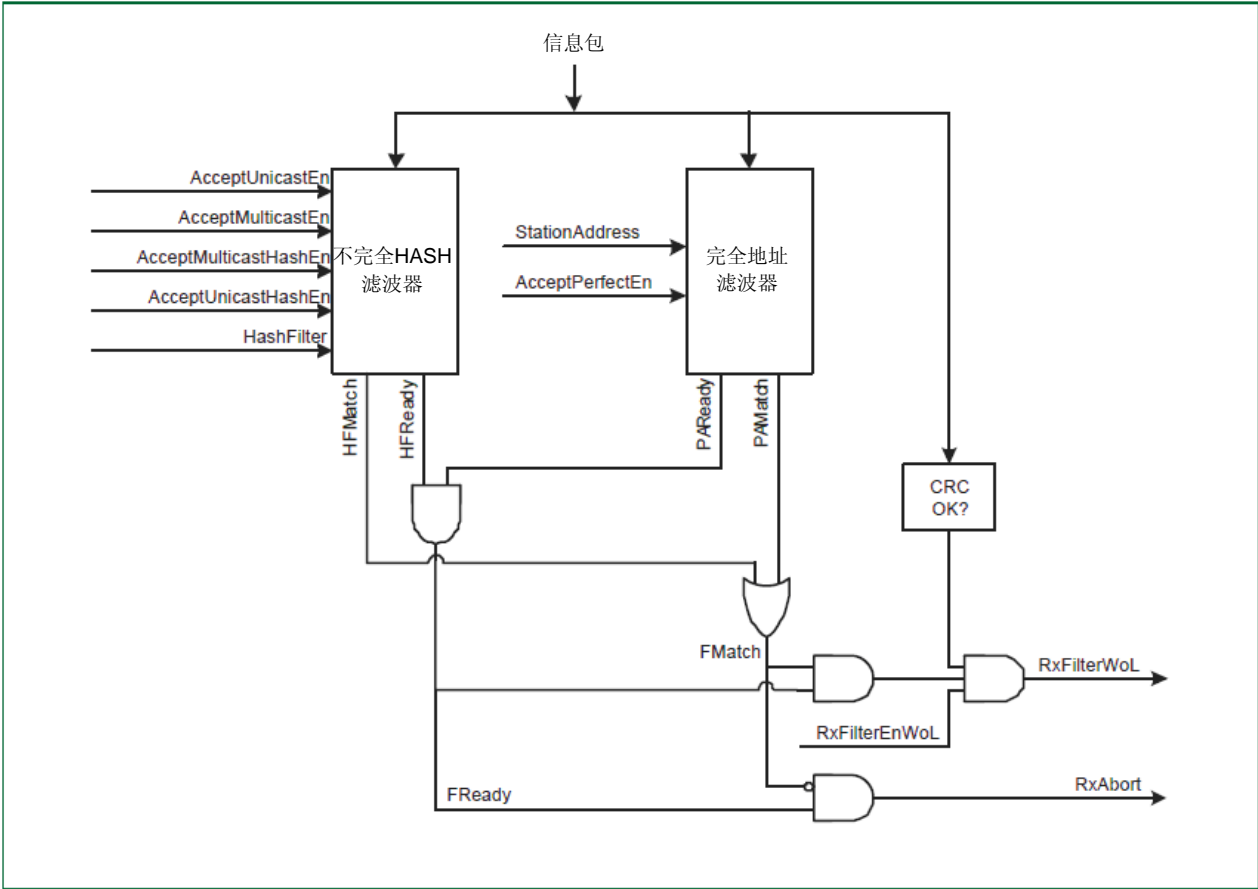
概述

以太网模块能够通过分析以太网帧内的目标地址，过滤接收到的帧。这种功能大大降低了主机系统的负荷，因为如果没有接收过滤，那么必须由设备驱动软件来检测和拒收寻址到其它站的以太网帧。这会占用带宽、存储空间和主机 CPU 时间。采用完全地址滤波器或（不完全的）Hash 滤波器，就可以实现地址过滤功能。后一种方法会产生一个 6 位的 Hash 编码，用做一个 64 入口的可编程 Hash 表的索引。[图 27](#) 给出了接收滤波器的功能图。

在图的顶端，以太网接收的帧进入滤波器。每个滤波器均受控制寄存器的信号控制；每个滤波器提供一个“就绪”（Ready）输出，和一个“匹配”（Match）输出。如果“Ready”是 0，则 Match 值就是“无关”（don't care）；如果滤波器完成了过滤，则使其 Ready 输出为有效；如果滤波器发现了一个匹配帧，则将与 Ready 输出一起使 Match 输出有效。滤波器的结果通过逻辑功能组合，成为一个单一的 RxAbort 输出。如果 RxAbort 输出有效，则该帧就不需要被接收。

为了减少存储器通信，接收数据通道有一个 68 字节的缓冲区。以太网 MAC 只有在 68 字节延迟后，才开始将一个帧写入存储器。如果在帧的前 68 字节中 RxAbort 信号有效，则该帧可以被丢弃，从缓冲区中清除，而根本不会保存至存储器，也不会用掉接收描述符等。如果在一个帧的前 68 字节后 RxAbort 信号才有效（可能因为收到了一个魔法帧），由于部分帧已经写入了存储器，以太网 MAC 将停止将帧中其余数据写入存储器；帧状态字中的 FailFilter 位将置位，表示设备驱动软件可以立即丢弃该帧。

图27. 接收滤波器方框图



单播、广播与多播

基于帧类型(单播、多播或广播)的一般过滤可以通过 RxFilterCtrl 寄存器的 AcceptUnicastEn、AcceptMulticastEn 或 AcceptBroadcastEn 位置位。置位 AcceptUnicast、AcceptMulticast 和 AcceptBroadcast 位会分别接收所有类型的单播、多播和广播帧，而忽略帧内的以太网目标地址。如要设置混合模式（即接收所有类型的帧），则将所有 3 个位都设为 1。

完全地址匹配

当收到一个有单播目标地址的帧时，完全滤波器会比较目标地址与站地址寄存器 SA0、SA1 和 SA2 中设置的 6 字节站地址。如果 RxFilterCtrl 寄存器中的 AcceptPerfectEn 位设为 1，并且地址匹配，则接受该帧。

不完全 Hash 过滤

不完全滤波器的使用基于一种 Hash 机制。此滤波器对目标地址使用一个 Hash 函数，用 Hash 访问一个表，该表指示出是否应接收此帧。这种类型滤波器的优点是，一个小表就可以覆盖任何可能的地址。缺点是过滤是不完全的，即有时应该丢弃的帧也被接受了。

- Hash 函数：
 - 从以太网帧的 6 个字节目标地址中，计算标准的以太网循环冗余检查（CRC）函数（在计算整帧 CRC 时总是要计算此 CRC），然后，取 32 位 CRC 结果的[28:23]位组成 Hash。6 位 Hash 用于访问 Hash 表：它用作 64 位 HashFilter 寄存器的一个索引，该寄存器已经编好了接受值。如果所选接受值为 1，则帧被接受。
 - 设备驱动软件可以初始化 Hash 滤波器，方法是写入 HashFilterL 和 HashfilterH 寄存器。HashFilterL 包含了表中的位 0 至 31，而 HashFilterH 包含表中的位 32 至 63。因此，Hash 值 0 对应于 HashfilterL 寄存器中的位 0，而 Hash 值 63 则对应于 HashFilterH 寄存器的位 31。
- 多播与单播
 - 不完全 Hash 滤波器可以用于多播地址，方法是置位 RxFilter 寄存器中的 AcceptMulticastHashEn 位为 1。
 - 同样，用于多播地址的不完全 Hash 滤波器也可以用于单播地址。它能响应大量的单播地址，而无需使能所有单播地址。通过将 RxFilter 寄存器的 AcceptUnicastHashEn 位设为 1，Hash 滤波器就可以应用于单播地址。

过滤的使能与禁能

通过置位命令寄存器中的 PassRxFilter 位，可以旁路以上章节所定义的滤波器。当 PassRxFilter 位置位时，所有接收的帧都会旁路到存储器。此时，设备驱动软件必须用软件来实现所有过滤功能。PassRxFilter 位置位不会影响到下一节所定义的短帧过滤。

短帧

小于 64 字节（或对 VLAN 帧是 68 字节）的帧比最小的以太网帧还要小，因而被认为是错误帧；它们可能会与片段发生冲突。接收数据通道会自动过滤并丢弃这些短帧，而不将它们写入存储器，也不使用接收描述符。

如果一个短帧有一个正确的 CRC，这个短帧可能是有用的。设备驱动软件可以接收有正确 CRC 的短帧，方法是置位命令寄存器中的 PassRunFrame 位为 1。

11.17.11 功率管理

以太网模块以时钟切换方式支持功率管理。以太网内核中所有时钟均可以关断。如果需要“LAN 上唤醒”，则应关断 rx_clk。

11.17.12 LAN 上唤醒

概述

以太网模块支持远程“LAN 上唤醒”的功率管理。主机系统可以掉电，甚至包括以太网模块自身也可以掉电，同时以太网模块连续监听 LAN 上的数据包。以太网模块可以接收并识别特定类型的数据包，用于触发主机系统，将其从掉电状态下唤醒。

系统的唤醒要通过中断而生效。当检测到一个唤醒事件时，`IntStatus` 寄存器中的 `WakeupInt` 位置位。如果 `IntEnable` 寄存器中的 `WakeupIntEn` 位置位，则中断状态将触发一个中断。系统功率管理逻辑应使用这个中断来唤醒系统。

在掉电状态下，LAN 上产生唤醒事件的包会丢失。

以太网包有两种方式可以触发唤醒事件：“LAN 上的通用唤醒”和“魔法包”。魔法包过滤要使用一个额外的滤波器，用于魔法包的检测。对于这两种情况，只有当触发数据包是有效 CRC 时，才能触发“LAN 上唤醒”事件。[图 27](#) 显示了唤醒信号的产生过程。

软件可以读取 `RxFilterWoLStatus` 寄存器，以检查唤醒事件的原因。在掉电以前，功率管理软件应通过写 `RxFilterWoLClear` 寄存器将其清除。

注：当进入掉电模式时，一个接收的帧不能完全保存在 Rx 缓冲区内。此时，在退出掉电模式后，下一个接收帧会被损坏，因为前一个帧的数据加到了后一个接收帧的前面。软件驱动必须在退出掉电模式后，立即将接收数据通道复位。

以下描述了两种 LAN 上唤醒机制。

通过过滤实现 WoL

接收过滤功能可以用于生成 LAN 上唤醒事件。如果 `RxFilterCtrl` 寄存器的 `RxFilterEnWoL` 位置位，则当接收一个通过滤波器的帧时，接收滤波器将 `IntStatus` 寄存器中的 `WakeupInt` 位置位。只有当帧的 CRC 正确时，才会产生中断。

魔法包 WoL

以太网模块支持采用魔法包技术的唤醒（见 AMD 公司的“魔法包技术”）。一个魔法包是一种特殊构成的数据包，专门用于唤醒。这种包可以被以太网模块接收、分析和识别，并用于触发一个唤醒事件。

魔法包数据部分包含了 16 个重复的、无间隔也无中断的站地址，在此之前是 6 个魔法包同步字节，其值均为 0xFF。其它数据可以处于包的数据部分中魔法包模式的周围。整个数据包必须是一个结构完好的以太网帧。

魔法包检测单元用于分析以太网包、提取包的地址，并检查魔法包模式的有效载荷。包内的地址用于模式的匹配（不是在 SA0/1/2 寄存器中的地址）。如果数据包通过了接收滤波器，则魔法包只设置唤醒中断状态位。如图 27 所示：接收滤波器的结果与魔法包滤波器的结果相“与”，得到最后结果。

魔法包过滤的使能是通过设置 RxFilterCtrl 寄存器中的 MagicPacketEnWoL 位。注意，在执行魔法包 WoL 功能时，RxFilterCtrl 寄存器中的 RxFilterEnWoL 位应为 0。将 RxFilterEnWoL 设为 1 会接收来自一个匹配地址的所有数据包，而不仅是魔法包，因此，采用魔法包实现 WoL 时要更严格。

当检测到一个魔法包时，除了 IntStatus 寄存器中的 WakeupInt 位置位以外，RxFilterWoLStatus 寄存器中的 MagicPacketWoL 位也要置位。软件可以向 RxFilterWoLClear 寄存器的相应位写入 1 将其复位。

举例：下面是一个站地址为 0x11 0x22 0x33 0x44 0x55 0x66 的魔法包实例（MISC 表示包内各种附加的数据字节）：

```
<DESTINATION> <SOURCE> <MISC>
FF FF FF FF FF FF
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
11 22 33 44 55 66 11 22 33 44 55 66
<MISC> <CRC>
```

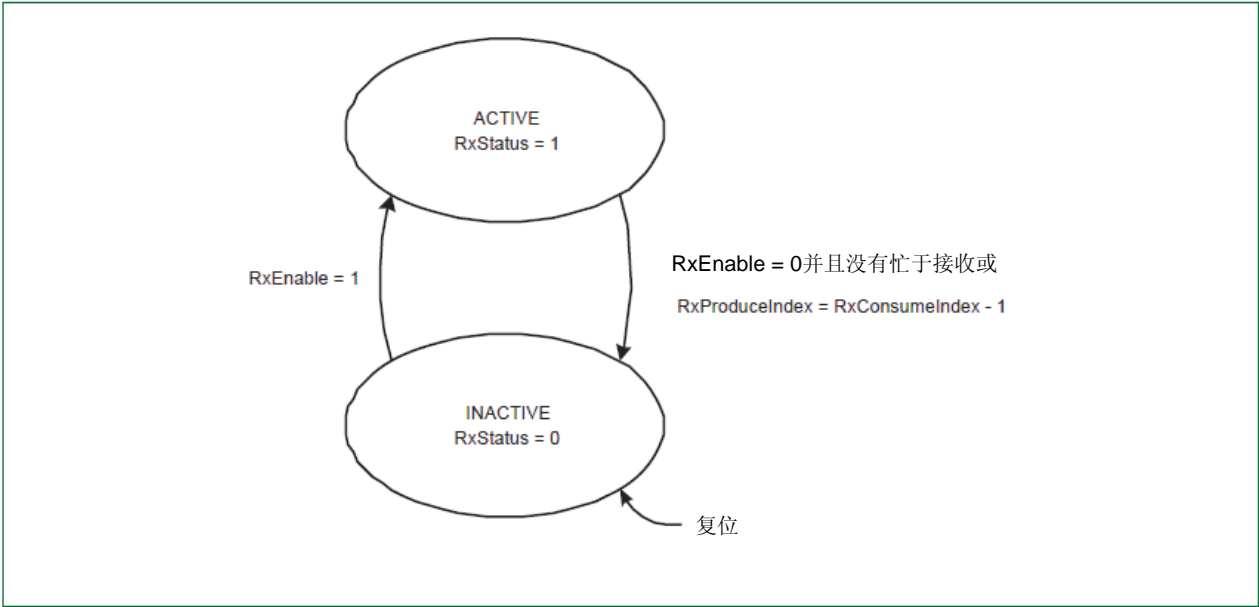
11.17.13 接收与发送的使能与禁能

接收的使能与禁能

复位以后，以太网模块的接收功能被禁能。接收功能的使能方法是，通过设备驱动程序将命令寄存器中的 RxEnable 位以及 MAC1 配置寄存器中的“RECEIVE ENABLE”位置位（按此顺序执行）。

设备驱动可以监控接收数据通道的状态，方法是读取 Status 寄存器的 RxStatus 位。图 28 表示了用于产生 RxStatus 位的状态机。

图28. 接收有效/无效的状态机



复位后，状态机处于 INACTIVE（无效）状态。一旦命令寄存器中的 RxEnable 位置位，则状态机就转换为 ACTIVE（有效）状态。当 RxEnable 位清零时，状态机又回到 INACTIVE 状态。如果当接收数据通道被禁能时，接收数据通道正忙于接收一个数据包，则该数据包会被完整接收，与其状态一起保存在存储器中，然后再回到 INACTIVE 状态。另外，如果接收描述符数组已满，则状态机也会返回 INACTIVE 状态。

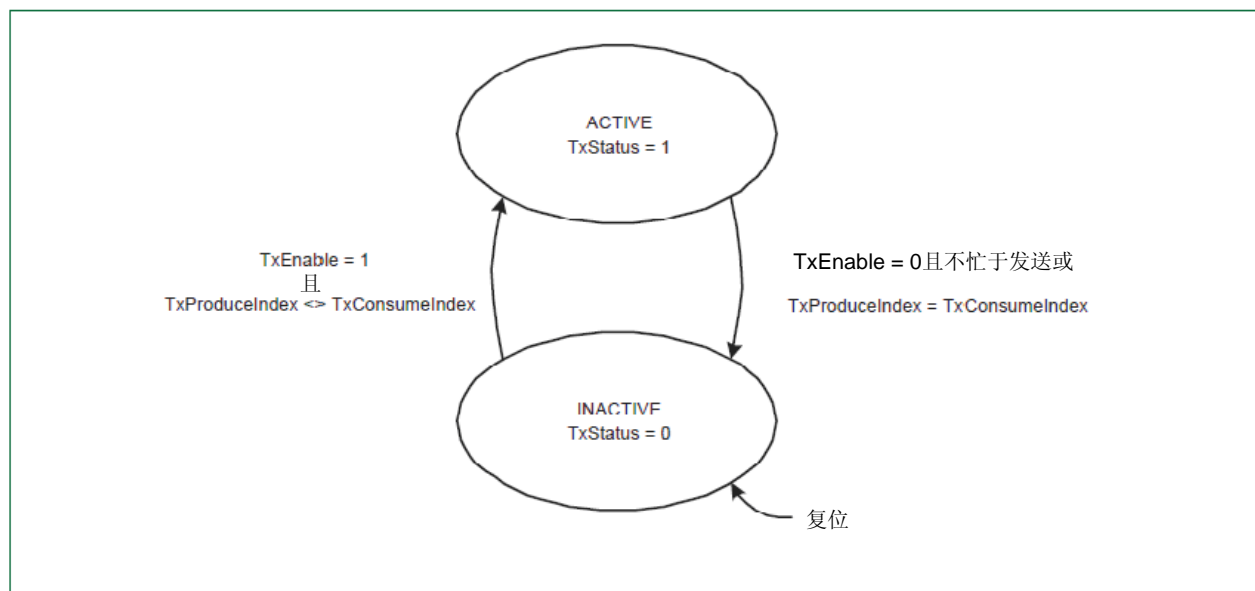
对于图 28 所示的状态机，软复位与硬复位同样有效，即，在软复位后，接收数据通道就为无效状态，直到数据通道被再次使能。

发送的使能与禁能

复位后，以太网模块的发送功能被禁能。设备驱动将命令寄存器中的 TxEnalbe 位设为 1，就可以使能 Tx 传送数据通道。

设备驱动通过读取 Status 寄存器的 TxStatus 位，就可以监控传送数据通道的状态。图 29 表示了产生 TxStatus 位的状态机。

图29. 发送有效/无效的状态机



复位后，状态机处于 INACTIVE（无效）状态。一旦命令寄存器中的 TxEnable 位置位，并且 Produce 和 Consume 两个索引不相等，则状态机就转换为 ACTIVE（有效）状态。当 TxEnable 位清零，并且发送数据通道已完成了所有挂起的发送操作，包括将发送状态提交给存储器，则状态机回到 INACTIVE 状态。如果 Produce 和 Consume 索引再次相等，即所有帧均已发送，则状态机也返回 INACTIVE 状态。

对于图 29 所示的状态机，软复位和硬复位同样有效，即在软复位后，发送数据通道就为无效状态，直到数据通道被再次使能。

11.17.14 发送填充与 CRC

当出现一个帧小于 60 字节（或对 VLAN 帧是 64 字节）的情况时，以太网模块会将帧填充到 64 或 68 字节，包括一个 4 字节的 CRC 帧校验序列（FCS）。影响填充功能的是：MAC2 配置寄存器中的“**AUTO DETECT PAD ENABLE**”（ADPEN）、“**VLAN PAD ENABLE**”（VLPEN）与“**PAD/CRC ENABLE**”（PADEN）位，以及发送描述符控制字的 Override 与 Pad 位。影响 CRC 产生的有：MAC2 配置寄存器的“**CRC ENABLE**”（CRCE）和“**DELAYED CRC**”（DCRC）位，以及传送描述符控制字的 Override 与 CRC 位。

如果描述符中 Override 位为 0，填充使能（EPADEN）的作用实际上等效于 MAC2 寄存器的“PAD/CRC ENABLE”位。如果 Override 位为 1，则 EPADEN 将取自描述符的 Pad 位。同样，如果 Override 位为 0，则 CRC 使能（ECRCE）实际等效于 CRCE，否则它等于描述符的 CRC 位。

如果需要使用填充功能，则被填充的帧信息上将始终添加 CRC。对于没有填充的帧，只有当 ECRCE 位置位时才添加 CRC。

如果 EPADEN 为 0，除非 ECRCE 位置位，否则帧不会被填充，也不会添加 CRC。

如果 EPADEN 为 1，则短帧将被填充，并总是给填充帧添加 CRC。此时，如果 ADPEN 与 VLPEN 都是 0，则帧会填充到 60 字节，再加一个 CRC，构成 64 字节的帧；如果 VLPEN 为 1，则帧会填充到 64 字节，再加一个 CRC，构成 68 字节的帧；如果 ADPEN 为 1，当 VLPEN 为 0 时，VLAN 帧将填充到 64 字节，非 VLAN 帧将填充到 60 字节，并会给填充的值添加 CRC，构成 64 或 68 字节的填充帧。

如果“生成 CRC”被使能，通过置位 MAC2 寄存器中的 DELAYED CRC 位，可以将 CRC 生成延迟 4 个字节，这样可跳过特定的帧头信息。

11.17.15 超长帧与帧长度检验

将 MAC2 配置寄存器中的“HUGE FRAME ENABLE”位设置为 1 可以使能任何长度帧的发送与接收。超长帧的发送可以逐帧使能，方法是将发送描述符中控制字的 Override 与 Huge 位置位。

当使能了超长帧时，以太网模块就不会检验帧长度，不会报告帧长度的错误（RangeError 与 LengthError）。如果使能了超长帧，RSV 寄存器中的已接收字节计数就可能无效，因为帧可能超过最大长度限制；而接收状态数组中的 RxSize 字段是有效的。

帧长度的检查方法是，将帧的长度/类型字段与帧中的实际字节数进行比较。通过置位接收 StatusInfo 字中的相应位，可以报告 LengthError。

MAXF 寄存器能够使设备驱动程序指定帧的最大字节数。以太网模块会将实际的接收帧与 MAXF 的值进行比较，并且当帧信息的实际字节数更大时，在接收 StatusInfo 字中报告 RangeError。

11.17.16 统计计数器

一般来说，以太网应用包括多个计数器，用于跟踪以太网通信的统计。针对这些计数器有很多标准，如 IEEE 标准 802.3/条款 30。其它标准如 RFC 2665 和 RFC 2233。

本文档采取的方法是，默认所有计数器均由软件实现。根据帧状态中的 StatusInfo 字段，在标准中列出的很多重要的统计事件都可以由软件计数。

11.17.17 MAC 状态向量

寄存器 TSV0、TSV1 和 RSV 中有可供 MAC 检测的发送与接收状态信息，这样软件就可以轮询它们。正常情况下应限制使用这些寄存器，因为驱动程序与以太网模块之间的通信主要是通过帧描述符发生的。统计事件可以在设计驱动程序中由软件来计数。而在调试时，发送与接收状态向量均设置为可见。只要 MAC 的内部状态为有效，它们也就有效，并且一般只应在发送和接收过程停止时读取。

11.17.18 复位

以太网模块有一个硬复位输入，它连接到芯片的复位脚上，还有几个软复位，可由寄存器中相应位的设置而激活。以太网模块中的所有寄存器在硬复位后都为 0，除非特别说明。

硬复位

硬复位后，所有寄存器都将设为自己的默认值。

软复位

通过置位命令寄存器以及 **MAC1** 配置寄存器中的位，可以对部分以太网模块进行软复位。**MAC1** 寄存器有 6 个不同的复位位：

- **SOFT RESET**: 该位置位将使 **MAC** 中的所有模块复位，除了 **MAC** 寄存器以外（地址为 0x000 至 0x0FC）。在一个硬复位生效后，软复位的值为 1，即在硬复位后，需要清零软复位。
- **SIMULATION RESET**: 将发送功能中的随机数发生器复位。硬复位生效后，该值为 0。
- **RESET MCS/Rx**: 该位置位将复位 **MAC** 的控制子层（暂停帧逻辑），以及 **MAC** 中的接收功能。硬复位生效后，该值为 0。
- **RESET Rx**: 该位置位将复位 **MAC** 中的接收功能。硬复位生效后，该值为 0。
- **RESET MCS/Tx**: 该位置位将复位 **MAC** 的控制子层（暂停帧逻辑），以及 **MAC** 中的发送功能。硬复位生效后，该值为 0。
- **RESET Tx**: 该位置位将复位 **MAC** 中的发送功能。硬复位生效后，该值为 0。

以上各复位位必须由软件清零。

命令寄存器有 3 个不同的复位位：

- **TxReset**: 向 **TxReset** 位写入“1”将复位发送数据通道，这包括除 **MAC** 部分以外的所有发送数据通道中（只读）的寄存器，以及主机寄存器模块中的 **TxProduceIndex** 寄存器。对发送数据通道的软复位将中止发送数据通道的所有 **AHB** 处理。复位位将由以太网模块自动清零。**Tx** 数据通道的软复位将清零 **Status** 寄存器中的 **TxStatus** 位。
- **RxReset**: 向 **RxReset** 位写入“1”将复位接收数据通道，这包括除 **MAC** 部分以外的所有接收数据通道中（只读）的寄存器，以及主机寄存器模块中的 **RxConsumeIndex** 寄存器。对接收数据通道的软复位将中止接收数据通道的所有 **AHB** 处理。复位位将由以太网模块自动清零。**Rx** 数据通道的软复位将清零 **Status** 寄存器中的 **RxStatus** 位。
- **RegReset**: 复位主机寄存器模块中的所有数据通道与寄存器，除了 **MAC** 中的寄存器以外。寄存器的软复位也将中止发送与接收数据通道的所有 **AHB** 处理。复位位将由以太网模块自动清零。

为了实现以太网模块的完全软复位，设备驱动软件必须完成下列操作：

- 将 **MAC1** 中的“**SOFT RESET**”位设为 1。
- 将命令寄存器中的 **RegReset** 位置位，该位是自动清零的。
- 重新初始化 **MAC** 寄存器（0x000 至 0x0FC）。
- 复位 **MAC1** 寄存器中的“**SOFT RESET**”位为 0。

如要只复位发送通道，设备驱动软件必须完成下列操作：

- 将 **MAC1** 寄存器中的“**RESET MCS/Tx**”位设为 1。

- 将命令寄存器中的 TxEnable 位设为 0，禁能 Tx DMA 管理器。
- 将命令寄存器中的 TxReset 位置位，该位是自动清零的
- 复位 MAC1 寄存器中的“RESET MCS/Tx”位为 0。

如要只复位接收通道，设备驱动软件必须完成下列操作：

- 复位 MAC1 配置寄存器中的“RECEIVE ENABLE”位，复位命令寄存器中的 RxEnable 位，禁能接收功能。
- 将 MAC1 寄存器中的“RESET MCS/Rx”位设为 1。
- 将命令寄存器中的 RxReset 位置位（该位自动清零）。
- 复位 MAC1 寄存器中的“RESET MCS/Rx”位为 0。

11.17.19 以太网错误

以太网模块会在以下情况下产生错误：

- 接收操作可能导致的错误：AlignmentError、RangeError、LengthError、SymbolError、CRCError、NoDescriptor 或 Overrun。这些错误在接收 StatusInfo 以及中断状态寄存器（IntStatus）中报告。
- 发送操作可能导致的错误：LateCollision、ExcessiveCollision、ExcessiveDefer、NoDescriptor 或 Underrun。这些错误将在发送 StatusInfo 以及中断状态寄存器（IntStatus）中报告。

11.18 AHB 带宽

以太网模块连接到一个 AHB 总线上，AHB 总线必须承载以太网通信相关的所有信息和控制信息，以及 CPU 访问所需要的信息以操作以太网模块并处理信息内容。

11.18.1 DMA 访问

假设

通过一些假设，可以计算出各种 AHB 传输需要的带宽，并将它们相加，得到总的带宽需求。

以太网模块使用描述符非常灵活，允许定义一定大小的缓冲区。为了分析总线的带宽需求，必须对这些缓冲区进行一些假设，使“最坏的情况”不会出现。因为这种情况下，所有描述符均指向单字节缓冲区，而大部分存储器用来保存描述符和极少的数据。显然，AHB 无法处理由这种低级（并且是不合逻辑）的情况引起的超大量总线通信。

在对此进行分析时，我们假设以太网包是一个 64 字节的帧，且发送通道与接收通道的通信都是连续的。

本分析并不反映以太网通信在时间上的流程，这样，在发送通道与接收通道中将采用内部包间隙，从而减少了一个较大时间帧的带宽需求。

DMA 访问类型及其带宽要求

与外部以太网 PHY 的接口是 RMII。RMII 工作在 50MHz，每 4 个时钟周期传输一个字节。数据传输速率为 12.5Mbps。

与外部以太网 PHY 的接口是 MII 或 RMII。MII 接口工作在 25MHz，每 2 个时钟周期传输一个字节。RMII 工作在 50MHz，每 4 个时钟周期传输一个字节。两种情况下数据传输速率同为 12.5Mbps。

下列情况下，以太网模块启动 DMA 访问：

- Tx 描述符读操作：
 - 发送描述符占用存储器的 2 个字（8 字节），每用一个描述符读取一次。
 - 每传输 64 字节（16 个字）的数据，2 个字被读取一次。
 - 因此，读描述符的速率为数据速率的 1/8，即 1.5625Mbps。
- Rx 描述符读操作：
 - 接收描述符占用存储器的 2 个字（8 字节），每用一个描述符读取一次。
 - 每接收 64 字节（16 个字）的数据，2 个字被读取一次。
 - 因此，读描述符的速率为数据速率的 1/8，即 1.5625Mbps。
- Tx 状态写操作：
 - 发送状态占用存储器的 1 个字（4 字节），每用一个描述符写一次。
 - 每传输 64 字节（16 个字）的数据，这 1 个字被读取一次。
 - 因此，写发送状态的速率为数据速率的 1/16，即 0.7813Mbps。

- Rx 状态写操作：
 - 接收状态占用存储器的 2 个字（8 字节），每用一个描述符写一次。
 - 每传输 64 字节（16 个字）的数据，这 2 个字被读取一次。
 - 因此，写发送状态的速率为数据速率的 1/8，即 1.5625Mbps。
- Tx 数据读操作：
 - 数据以一个以太网帧发送，大小可变。
 - 基本以太网速率为 12.5Mbps。
- Rx 数据写操作：
 - 数据以一个以太网帧接收，大小可变。
 - 基本以太网速率为 12.5Mbps。

这样，以太网 DMA 功能所产生的通信总速率为 30.5Mbps。

11.18.2 CPU 访问的类型

- 镜像了各种 DMA 类型的访问：
 - 必须读取所有状态值或部分状态值，每次使用后，需要对所有描述符或部分描述符进行写操作，发送的数据必须由 CPU 保存在存储器中，最后接收的数据必须由 CPU 从存储器中获取。
 - 这样就得到了与组合 DMA 功能大致相同或略低的速率，即 30.5Mbps。
- 访问以太网模块中的寄存器：
 - CPU 必须读取 RxProduceIndex、TxConsumeIndex 和 IntStatus 寄存器，并同时读写 RxConsumeIndex 与 TxProduceIndex 寄存器。
 - 每发送和接收 64 字节（16 个字）的数据，就执行一次 7 字读/写操作。
 - 因此，访问速率为数据速率的 7/16，即 5.4688Mbps。

综上所述，以太网 DMA 功能所产生的通信总速率为 36Mbps。

11.18.3 总带宽

AHB 上总的通信是 DMA 访问速率与 CPU 访问速率之和，大约为 66.5 MB/s。

峰值带宽需求可能略高，因为使用了部分存储器缓冲区，例如用于保存常用的地址（如站地址）。驱动软件可以决定如何在不过度使用 AHB 的情况下高效地建立帧。

AHB 总线上可用的带宽取决于系统时钟频率。举例来说，假设系统时钟设定为 60 MHz。所有或几乎所有与以太网相关的总线访问都是字传输。原始 AHB 带宽中基本上每两个系统时钟传输 4 个字节，即相当于系统时钟速率的两倍。在 60 MHz 系统时钟下，带宽约为 120 MB/s，在同时发送和接收情况下，以太网通信的 AHB 总线利用率大约是 55%。这表明，不需要为以太网使用最大的 CPU 频率，就有足够的带宽余量。

11.19CRC 计算

计算用于下列目的:

- 在以太网帧末尾产生 FCS。
- 生成用于 Hash 表过滤的 Hash 表索引。
- 生成目标地址与源地址的 Hash CRC。

下列的 C 伪代码函数逐帧地计算 CRC (无 FCS), 并将帧内的字节数作为参数。函数返回的 CRC 是一个 32 位整数。

```
int crc_calc(char frame_no_fcs[], int frame_len) {
    int i; // iterator
    int j; // another iterator
    char byte; // current byte
    int crc; // CRC result
    int q0, q1, q2, q3; // temporary variables
    crc = 0xFFFFFFFF;
    for (i = 0; i < frame_len; i++) {
        byte = *frame_no_fcs++;
        for (j = 0; j < 2; j++) {
            if (((crc >> 28) ^ (byte >> 3)) & 0x00000001) {
                q3 = 0x04C11DB7;
            } else {
                q3 = 0x00000000;
            }
            if (((crc >> 29) ^ (byte >> 2)) & 0x00000001) {
                q2 = 0x09823B6E;
            } else {
                q2 = 0x00000000;
            }
            if (((crc >> 30) ^ (byte >> 1)) & 0x00000001) {
                q1 = 0x130476DC;
            } else {
                q1 = 0x00000000;
            }
            if (((crc >> 31) ^ (byte >> 0)) & 0x00000001) {
                q0 = 0x2608EDB8;
            } else {
                q0 = 0x00000000;
            }
            crc = (crc << 4) ^ q3 ^ q2 ^ q1 ^ q0;
            byte >>= 4;
        }
    }
    return crc;
}
```

对于 FCS 计算, 将帧中的第一个字节和帧长度 (无 FCS) 传递给函数。

对于 Hash 过滤, 将帧的目标地址部分传递给函数, 而只对 6 个地址字节计算 CRC。Hash 滤波器使用[28:23]位, 作为 64 位{ HashFilterH, HashFilterL }向量的索引。如果对应的位置

位，则数据包通过，否则被 Hash 滤波器拒绝。

为了得到目标地址与源地址的 Hash CRC，此函数先计算两个 32 位 CRC，然后提取每个 32 位 CRC 的 9 个最高有效位，将它们连接起来，写入每个帧状态的 StatusHashCRC 字中。

12.1 如何阅读本章

某些 LPC178x/177x 设备上有 LCD 控制器，详见 [1.4](#) 节。

12.2 基本配置

LCD 控制器的配置使用下列寄存器：

1. 功率：PCONP 寄存器（见 [4.7.10](#) 节），置位 PCLCD 位。
注：复位后 LCD 被禁用（PCLCD = 0）。
亦见 [12.6.12](#) 节的上电步骤。
2. 时钟：见表 [226](#) 和表 [223](#)。
3. 管脚：通过相应的 IOCON 寄存器，选择 LCD 管脚与管脚模式（见 [8.4.1](#) 节）。
4. 中断：利用相应的中断置位使能寄存器使能 NVIC 中的中断。

12.3 简介

LCD 控制器为接口提供了所有需要的控制信号，接口直接连接各种彩色与单色 LCD 面板。

12.4 特性

- AHB 总线主接口，用于访问帧缓冲区。
- 独立的 AHB 从接口，用于设置与控制。
- 双 16 深度的可编程 64 位宽 FIFO，用于缓冲输入的显示数据。
- 支持接口为 4 位或 8 位的单面板与双面板单色超扭曲向列（STN）型显示器。
- 支持单面板和双面板的彩色 STN 显示器。
- 支持薄膜晶体管（TFT）彩色显示器。
- 可编程设定的显示分辨率，包括但不限于：320x200、320x240、640x200、640x240、640x480、800x600 与 1024x768。
- 支持单面板显示器的硬件光标。
- 支持 15 灰度级单色、3375 种彩色 STN，以及 32K 种颜色调色板 TFT。
- 单色 STN 的 1、2 或 4 位/像素（bpp）调色板显示器。
- 彩色 STN 和 TFT 的 1、2、4 或 8 位/像素（bpp）调色板彩色显示器。
- 16 bpp 真彩无调色板，支持彩色 STN 与 TFT。
- 24 bpp 真彩无调色板，支持彩色 TFT。
- 对不同显示面板的可编程时序。
- 256 个表项的 16 位调色 RAM，排列为一个 128x32 位 RAM。
- 帧、行与像素时钟信号。
- STN 的 AC 偏压信号，TFT 面板的数据使能信号。
- 支持小端字节序（little-endian）和大端字节序（big-endian），以及 Windows CE 数据格式。
- LCD 面板时钟可以来自外设时钟，或一个时钟输入管脚。

12.4.1 可编程参数

以下是可以编程的显示器与控制器关键参数：

- 水平前沿与后沿
- 水平同步脉冲宽度
- 每行像素数
- 垂直前沿与后沿
- 垂直同步脉冲宽度
- 每面板行数
- 每行像素时钟数
- 硬件光标控制
- 信号极性，高电平有效或低电平有效
- AC 面板偏压
- 面板时钟频率
- 每像素位数
- 显示器类型：STN 单色、STN 彩色或 TFT
- STN 的 4 位或 8 位接口模式

- STN 单、双面板模式
- 小端字节序、大端字节序或 Windows CE 模式
- 中断产生事件

12.4.2 硬件光标支持

硬件光标特性减少了在 LCD 帧缓冲区中维持一个光标图像所需要的软件开销。

如果没有这一特性, 软件要做下列工作:

- 保存下一个光标位置所在区域的图像。
- 用光标图像刷新该区域。
- 用以前保存的图像, 修复前一个光标的位置。

此外, LCD 驱动还必须检查图像操作是否覆盖了该光标, 并做出修正。当光标尺寸为 64x64, 颜色为 24 位彩色时, 每次光标移动都要读写大约 75kB 数据。

硬件光标则无需这种管理工作, 它为光标提供了一个完全独立的图像缓冲区, 在当前光标的 (X, Y) 坐标上, 将光标图像重叠在 LCD 输出流上。

移动硬件光标时, 软件驱动提供一个新的光标坐标。帧缓冲区无需修改。这样就大大降低了软件开销。

光标图像保存在 LCD 控制器内的一个 256x32 位缓冲区中。

12.4.3 支持的 LCD 面板类型

LCD 控制器支持下列类型的 LCD 面板:

- 有源矩阵 TFT 面板, 高达 24 位的总线接口。
- 单面单色 STN 面板 (4 位和 8 位总线接口)。
- 双面单色 STN 面板 (每面板 4 位和 8 位总线接口)。
- 单面彩色 STN 面板, 8 位总线接口。
- 双面彩色 STN 面板, 每面板 8 位总线接口。

12.4.4 TFT 面板

TFT 面板支持下列一种或多种彩色模式:

- 1 bpp、调色、从可用颜色中选择 2 色。
- 2 bpp、调色、从可用颜色中选择 4 色。
- 4 bpp、调色、从可用颜色中选择 16 色。
- 8 bpp、调色、从可用颜色中选择 256 色。
- 12 bpp、直接 4:4:4 RGB。
- 16 bpp、直接 5:5:5 RGB, 1 bpp 通常不使用。这个像素仍有输出, 可以用作一个亮

度位，连接到一个 6:6:6 TFT 面板 RGB 分量的最低有效位（LSB）。

- 16 bpp、直接 5:6:5 RGB。
- 24 bpp、直接 8:8:8 RGB，提供 1600 多万种颜色。

每个 16 位调色板入口都包含 5 bpp（RGB），再加一个公共的亮度位。与全 6 bpp 结构相比，这种方式提供了更好的存储器使用率及性能。如果使用了亮度位，并同时适用于所有三个颜色分量，则所支持的颜色总数可以从 32K 加倍到 64K。

另外，可以用 16 个信号驱动一块 5:6:5 的面板，附加位只用于绿色通道。

12.4.5 彩色 STN 面板

彩色 STN 面板支持下列一种或多种彩色模式：

- 1 bpp、调色、从 3375 种颜色中选择 2 色。
- 2 bpp、调色、从 3375 种颜色中选择 4 色。
- 4 bpp、调色、从 3375 种颜色中选择 16 色。
- 8 bpp、调色、从 3375 种颜色中选择 256 色。
- 16 bpp、直接 4:4:4 RGB，4 bpp 未使用。

12.4.6 单色 STN 面板

单色 STN 面板支持下列一种或多种模式：

- 1 bpp、调色、从 15 个灰度级中选择 2 个。
- 2 bpp、调色、从 15 个灰度级中选择 4 个。
- 4 bpp、调色、从 15 个灰度级中选择 16 个。

对于单色面板，可以编程到 4 bpp 以上，但使用这些模式并没有益处，因为显示器支持的最大灰度级是 15。

12.5 管脚描述

LCD 控制器的最大配置采用了 31 个管脚。对单色 STN 面板，有多种只使用 10 只管脚的变化形式。根据所选择的配置，管脚会划分成组。所有 LCD 功能都与其它芯片功能共享。[表 205](#) 中只显示了与 LCD 有关的部分管脚名称。

注：LCD 控制器连接至所需管脚的方式见 [8.3](#) 节。

表205. LCD 控制器管脚

管脚名称	类型	功能
LCD_PWR	输出	LCD 面板功率使能。
LCD_DCLK	输出	LCD 面板时钟。
LCD_ENAB_M	输出	STN AC 偏压驱动或 TFT 数据使能输出。
LCD_FP	输出	帧脉冲(STN)。垂直同步脉冲（TFT）

管脚名称	类型	功能
LCD_LE	输出	线端信号
LCD_LP	输出	线同步脉冲（STN）。水平同步脉冲（TFT）
LCD_VD[23:0]	输出	LCD 面板数据。所使用的位取决于面板配置。
LCD_CLKIN	输入	可选时钟输入。

12.5.1 信号使用

各种类型显示器使用的信号在以下章节中说明。

12.5.1.1 单面板 STN 显示器使用的信号

单面板 STN 显示器使用的信号见[表 206](#)。UD 指的是上面板的数据。

表206. 单面板 STN 显示器使用的管脚

管脚名称	4 位单色 (10 个管脚)	8 位单色 (14 个管脚)	彩色 (14 个管脚)
LCD_PWR	是	是	是
LCD_DCLK	是	是	是
LCD_ENAB_M	是	是	是
LCD_FP	是	是	是
LCD_LE	是	是	是
LCD_LP	是	是	是
LCD_VD[3:0]	UD[3:0]	UD[3:0]	UD[3:0]
LCD_VD[7:4]	-	UD[7:4]	UD[7:4]
LCD_VD[23:8]	-	-	-

12.5.1.2 双面板 STN 显示器使用的信号

双面板 STN 显示器使用的信号见[表 207](#)。UD 指的是上面板的数据，LD 指的是下面板的数据。

表207. 双面板 STN 显示器使用的管脚

管脚名称	4 位单色 (14 个管脚)	8 位单色 (22 个管脚)	彩色 (22 个管脚)
LCD_PWR	是	是	是
LCD_DCLK	是	是	是
LCD_ENAB_M	是	是	是
LCD_FP	是	是	是
LCD_LE	是	是	是
LCD_LP	是	是	是
LCD_VD[3:0]	UD[3:0]	UD[3:0]	UD[3:0]
LCD_VD[7:4]	-	UD[7:4]	UD[7:4]
LCD_VD[11:8]	LD[3:0]	LD[3:0]	LD[3:0]
LCD_VD[15:12]	-	LD[7:4]	LD[7:4]
LCD_VD[23:16]	-	-	-

12.5.1.3 TFT 显示器使用的信号

TFT 显示器使用的信号见[表 208](#)。

表208. TFT 显示器使用的管脚

管脚名称	12 位，4:4:4 模式 (18 个管脚)	16 位，5:6:5 模式 (22 个管脚)	16 位，1:5:5:5 模式 (24 个管脚)	24 位 (30 个管脚)
LCD_PWR	是	是	是	是
LCD_DCLK	是	是	是	是
LCD_ENAB_M	是	是	是	是
LCD_FP	是	是	是	是
LCD_LE	是	是	是	是
LCD_LP	是	是	是	是
LCD_VD[1:0]	-	-	-	红[1:0]
LCD_VD[2]	-	-	亮度	红[2]
LCD_VD[3]	-	红[0]	红[0]	红[3]
LCD_VD[7:4]	红[3:0]	红[4:1]	红[4:1]	红[7:4]
LCD_VD[9:8]	-	-	-	绿[1:0]
LCD_VD[10]	-	绿[0]	亮度	绿[2]
LCD_VD[11]	-	绿[1]	绿[0]	绿[3]
LCD_VD[15:12]	绿[3:0]	绿[5:2]	绿[4:1]	绿[7:4]
LCD_VD[17:16]	-	-	-	蓝[1:0]
LCD_VD[18]	-	-	亮度	蓝[2]
LCD_VD[19]	-	蓝[0]	蓝[0]	蓝[3]
LCD_VD[23:20]	蓝[3:0]	蓝[4:1]	蓝[4:1]	蓝[7:4]

12.6 LCD 控制器功能描述

LCD 控制器将以像素编码的数据转换为所需要的格式和时序，用于驱动各种单双面板的单色和彩色 LCD。

以像素编码的数据包通过 AHB 接口发送到两个独立可编程的 32 位宽 DMA FIFO，FIFO 用作输入数据流的缓冲区。

然后，使用一个像素串行器，将缓冲区的像素编码数据解包。

根据不同的 LCD 类型与模式，解包后的数据可以表示：

- 一种实际的显示灰度值或颜色值。
- 一个表示 256x16 位宽调色 RAM 地址的灰度值或颜色值。

如果是 STN 显示器，可以从所指调色位置获得一个值，也可以将实际值传递给灰度级发生器。在一些已设定数量的帧上，硬编码的灰度级算法逻辑对寻址像素的活动做出排序，以提供有效的显示表现。

对于 TFT 显示器，无论是寻址调色值还是实际颜色值，都直接转送给输出显示驱动器，而

绕过灰度级算法逻辑。

除了数据格式化以外，LCD 控制器还提供一组可编程的显示控制信号，包括：

- LCD 面板功率使能。
- 像素时钟。
- 水平与垂直同步脉冲。
- 显示偏压。

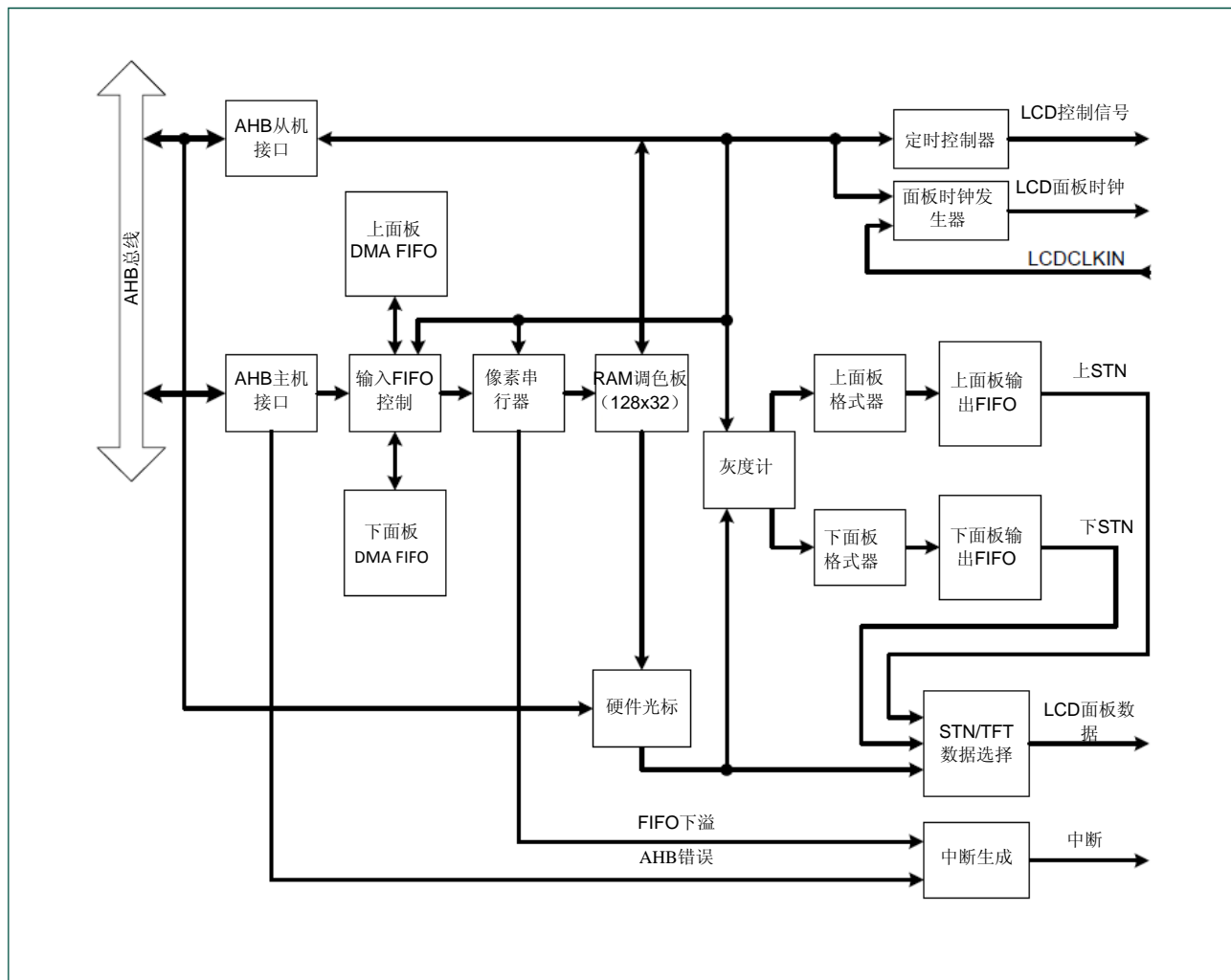
LCD 控制器可为下列事件产生单独中断：

- 上、下面板的 DMA FIFO 下溢。
- 基址刷新有效。
- 垂直比较。
- 总线错误。

当任何单个中断有效时，还可使能一个联合中断。

[图 30](#) 显示了 LCD 控制器的简化框图。

图30. LCD 控制器方框图



12.6.1 AHB 接口

LCD 控制器包含两个独立的 AHB 接口。第一个为 AHB 从接口，主要被 CPU 用于访问 LCD 控制器内部的控制与数据寄存器。第二个为 AHB 主接口，被 LCD 控制器用于对系统其它位置存储器中保存显示数据的 DMA 访问。LCD DMA 控制器只能访问 AHB1 上的 10kB SRAM 和外部存储器。

12.6.1.1 AMBA AHB 从接口

AHB 从接口将 LCD 控制器连接到 AHB 总线，并提供 CPU 对寄存器和调色 RAM 的访问。

12.6.1.2 AMBA AHB 主接口

AHB 主接口将显示数据从一个选定的从（存储器）传送给 LCD 控制器的 DMA FIFO。它的数据来源可以配置为：AHB1 上的 16kB 片上 SRAM、各种类型片外静态存储器，或片外 SDRAM。

在双面板模式下，DMA FIFO 通过单一 DMA 请求，采用交替方式填充。在单面板模式下，DMA FIFO 由单一 DMA 请求，采用连续方式填充。

内置的 AHB 主接口状态机完成下列功能：

- 在识别了一个新帧时，将上面板的基址加载到 AHB 地址增量器。
- 同时监控上、下 DMA FIFO 的水平，并提出一个 DMA 请求，申请存储器中的显示数据，将显示数据填入上述已设定的水印（watermark）。当两个 FIFO（双面板模式）中某一个至少有 4 个可用位置时，再次提出 DMA 请求。
- 在定长的突发传输期间，检查 1kB 的边界，相应地调节这类事件中的地址。
- 为定长与未定义的突发传输生成地址队列。
- 控制存储器与 DMA FIFO 之间的握手。如果 FIFO 尚未完成自己的同步及刷新队列，则插入忙周期。
- 在双面板模式下，以一种不同于单面板 DMA 请求的方式，填充 DMA FIFO。
- 如果在一个有效突发传输期间出现了一个错误，则产生一个总线错误中断。
- 响应重试命令，重新启动失败的访问。这会导致再同步期间产生一些忙周期。

12.6.2 双 DMA FIFO 与相关控制逻辑

从存储器获取的像素数据由两个 DMA FIFO 缓冲，FIFO 可以单独控制，从而覆盖单、双面板 LCD 类型。每个 FIFO 为 16 字深和 64 位宽，可以级联，在单面板模式下组成一个相当于 32 个双字深度的 FIFO。

同步逻辑将像素数据从 AHB 模块域传送到 LCD 控制器的时钟域。每个 FIFO 都设置了水平标记，这样，当至少有 4 个位置可用时，每个 FIFO 就会请求数据。

如果当两个 DMA FIFO 中任一空（发生一个下溢出状况），而做读取尝试时，就会产生一个中断信号。

12.6.3 像素串行器

这个模块从 DMA FIFO 的输出端口读取 32 位宽的 LCD 数据，根据当前的工作模式，提取出 24、16、8、4、2 或 1 bpp 数据。LCD 控制器支持大端字节序、小端字节序和 Windows CE 数据格式。

根据不同工作模式，提取的数据可以用于指向调色 RAM 中的一个颜色值或灰度级值，或可以用做一个实际颜色值，直接加到 LCD 面板的输入上。

[表 209](#) 至 [表 211](#) 给出了对应于字节序与 bpp 组合的各种 DMA FIFO 字的数据结构。对于这三种支持的数据格式，所需要的各种面板显示像素数据都必须从数据字中提取。

表209. 与小端字节和小端像素次序相对应的 FIFO 位

FIFO 位	1 bpp	2 bpp	4 bpp	8 bpp	16 bpp	24 bpp								
0	p0	p0	p0	p0	p0	p0								
1	p1													
2	p2	p1	p1											
3	p3													
4	p4	p2												
5	p5													
6	p6	p3		p1	p1									
7	p7													
8	p8	p4	p2											
9	p9													
10	p10	p5												
11	p11													
12	p12	p6	p3	p2	p1									
13	p13													
14	p14	p7												
15	p15													
16	p16	p8	p4	p2	p1	p1								
17	p17													
18	p18	p9												
19	p19													
20	p20	p10	p5											
21	p21													
22	p22	p11												
23	p23													
24	p24	p12	p6	p3	p1									
25	p25													
26	p26	p13												
27	p27													
28	p28	p14	p7											
29	p29													
30	p30	p15												
31	p31													

表210. 与大端字节和大端像素次序相对应的 FIFO 位

FIFO 位	1 bpp	2 bpp	4 bpp	8 bpp	16 bpp	24 bpp
0	p31	p15	p7	p3	p1	p0
1	p30					
2	p29					
3	p28	p14				
4	p27		p6			
5	p26					
6	p25					
7	p24	p12				
8	p23		p5			
9	p22					
10	p21					
11	p20	p2				
12	p19		p10			
13	p18					
14	p17					
15	p16	p9	p4			
16	p15			p8		
17	p14					
18	p13	p7				
19	p12		p6			
20	p11					
21	p10			p5		
22	p9	p4				
23	p8					
24	p7		p3			
25	p6	p1				
26	p5					
27	p4			p2		
28	p3	p1				
29	p2					
30	p1		p0			
31	p0					

表211. 与小端字节和大端像素次序相对应的 FIFO 位

FIFO 位	1 bpp	2 bpp	4 bpp	8 bpp	16 bpp	24 bpp								
0	p7	p3	p1	p0	p0	p0								
1	p6													
2	p5	p2												
3	p4													
4	p3	p1	p0											
5	p2													
6	p1	p0												
7	p0													
8	p15	p7	p3	p1	p0									
9	p14													
10	p13	p6												
11	p12													
12	p11	p5	p2											
13	p10													
14	p9	p4												
15	p8													
16	p23	p11	p5	p2	p1									
17	p22													
18	p21	p10												
19	p20													
20	p19	p9	p4											
21	p18													
22	p17	p8												
23	p16													
24	p31	p15	p7	p3										
25	p30													
26	p29	p14												
27	p28													
28	p27	p13	p6											
29	p26													
30	p25	p12												
31	p24													

表 212 显示了 RGB 模式下每个 DMA FIFO 字的数据结构。

表212. RGB 模式下的数据格式

FIFO 数据	24 位 RGB	16 位（1:5:5:5 RGB）	16 位（5:6:5 RGB）	16 位（4:4:4 RGB）
0	p0, 红 0	p0, 红 0	p0, 红 0	p0, 红 0
1	p0, 红 1	p0, 红 1	p0, 红 1	p0, 红 1
2	p0, 红 2	p0, 红 2	p0, 红 2	p0, 红 2
3	p0, 红 3	p0, 红 3	p0, 红 3	p0, 红 3
4	p0, 红 4	p0, 红 4	p0, 红 4	p0, 绿 0
5	p0, 红 5	p0, 绿 0	p0, 绿 0	p0, 绿 1
6	p0, 红 6	p0, 绿 1	p0, 绿 1	p0, 绿 2
7	p0, 红 7	p0, 绿 2	p0, 绿 2	p0, 绿 3
8	p0, 绿 0	p0, 绿 3	p0, 绿 3	p0, 蓝 0
9	p0, 绿 1	p0, 绿 4	p0, 绿 4	p0, 蓝 1
10	p0, 绿 2	p0, 蓝 0	p0, 绿 5	p0, 蓝 2
11	p0, 绿 3	p0, 蓝 1	p0, 蓝 0	p0, 蓝 3
12	p0, 绿 4	p0, 蓝 2	p0, 蓝 1	-
13	p0, 绿 5	p0, 蓝 3	p0, 蓝 2	-
14	p0, 绿 6	p0, 蓝 4	p0, 蓝 3	-
15	p0, 绿 7	p0 亮度位	p0, 蓝 4	-
16	p0, 蓝 0	p1, 红 0	p1, 红 0	p1, 红 0
17	p0, 蓝 1	p1, 红 1	p1, 红 1	p1, 红 1
18	p0, 蓝 2	p1, 红 2	p1, 红 2	p1, 红 2
19	p0, 蓝 3	p1, 红 3	p1, 红 3	p1, 红 3
20	p0, 蓝 4	p1, 红 4	p1, 红 4	p1, 绿 0
21	p0, 蓝 5	p1, 绿 0	p1, 绿 0	p1, 绿 1
22	p0, 蓝 6	p1, 绿 1	p1, 绿 1	p1, 绿 2
23	p0, 蓝 7	p1, 绿 2	p1, 绿 2	p1, 绿 3
24	-	p1, 绿 3	p1, 绿 3	p1, 蓝 0
25	-	p1, 绿 4	p1, 绿 4	p1, 蓝 1
26	-	p1, 蓝 0	p1, 绿 5	p1, 蓝 2
27	-	p1, 蓝 1	p1, 蓝 0	p1, 蓝 3
28	-	p1, 蓝 2	p1, 蓝 1	-
29	-	p1, 蓝 3	p1, 蓝 2	-
30	-	p1, 蓝 4	p1, 蓝 3	-
31	-	p1 亮度位	p1, 蓝 4	-

12.6.4 RAM 调色板

基于 RAM 的调色板是一个 256 x 16 位的双口 RAM，物理结构为 128 x 32 位。由一个字的写访问就可以将两个表项写入调色板。串行像素数据的最低有效位（LSB）用于选择调色 RAM 的上半部分和下半部分。选择结果取决于字节序的模式。在小端字节序模式下，设置 LSB 选择的是上半部分，而在大端字节序模式下，则选择调色板的下半部分。

通过 AHB 从接口，可以写入并校验像素数据值。对于所支持的彩色的信息，参见本章前面有关面板类型的章节。

调色 RAM 是一种双口 RAM，每个端口都可以独立地控制与寻址。端口 1 用作一个读写端口，连接到 AHB 从接口。通过这个端口，可以写入并校验调色板表项。端口 2 用作一个只读端口，连接到解包器和灰度计。对于小于 16 bpp 的彩色模式，调色板能够将每个像素值映射到一个 16 位颜色：

- 对 TFT 显示器，16 位值被直接发送至像素串行器。
- 对 STN 显示器，16 位值先用灰度计转换。

[表 213](#) 给出了调色板数据各位的表述。调色板的 16 位输出使用了 TFT 1:5:5:5 数据格式。在 16 及 24 bpp TFT 模式下，调色板被旁路，像素串行的输出被用作 TFT 面板数据。

表213. TFT 模式下的调色板数据存储

位	名称 (RGB 格式)	描述 (RGB 格式)	名称 (BGR 格式)	描述 (BGR 格式)
4:0	R[4:0]	红色调色板数据	B[4:0]	蓝色调色板数据
9:5	G[4:0]	绿色调色板数据	G[4:0]	绿色调色板数据
14:10	B[4:0]	蓝色调色板数据	R[4:0]	红色调色板数据
15	I	亮度/未使用	I	亮度/未使用
20:16	R[4:0]	红色调色板数据	B[4:0]	蓝色调色板数据
25:21	G[4:0]	绿色调色板数据	G[4:0]	绿色调色板数据
30:26	B[4:0]	蓝色调色板数据	R[4:0]	红色调色板数据
31	I	亮度/未使用	I	亮度/未使用

红色与蓝色像素数据可以相互交换，以支持使用控制器位 8 的 BGR 数据格式（BGR）。详见 LCD_CTRL 寄存器的描述。[表 214](#) 给出了在 STN 彩色模式下，调色板数据各位的表述。

表214. STN 彩色模式下的调色板数据存储

位	名称 (RGB 格式)	描述 (RGB 格式)	名称 (BGR 格式)	描述 (BGR 格式)
0	R[0]	未使用	B[0]	未使用
4:1	R[4:1]	红色调色板数据	B[4:1]	蓝色调色板数据
5	G[0]	未使用	G[0]	未使用
9:6	G[4:1]	绿色调色板数据	G[4:1]	绿色调色板数据
10	B[0]	未使用	R[0]	未使用
14:11	B[4:1]	蓝色调色板数据	R[4:1]	红色调色板数据
15	I	未使用	I	未使用
16	-	未使用	-	未使用
20:17	R[3:0]	红色调色板数据	B[3:0]	蓝色调色板数据
21	-	未使用	-	未使用
25:22	G[3:0]	绿色调色板数据	G[3:0]	绿色调色板数据
26	-	未使用	-	未使用
30:27	B[3:0]	蓝色调色板数据	R[3:0]	红色调色板数据
31	-	未使用	-	未使用

对于单色 STN 模式，只使用红色调色板域的位[4:1]。但是，在 STN 彩色模式下，还要使用绿色和蓝色[4:1]。每种颜色只使用 4 位，因为灰度计对每种颜色只支持 16 级灰度。

[表 215](#) 给出了在 STN 单色模式下，调色板数据的各位表述。

表215. STN 单色模式下的调色板数据存储

位	名称	描述
0	-	未使用
4:1	Y[3:0]	亮度数据
16:5	-	未使用
20:17	Y[3:0]	亮度数据
31:21	-	未使用

12.6.5 硬件光标

硬件光标是 LCD 控制器的内在组成部分。它采用 LCD 时序模块，提供了一个对当前扫描位置坐标的指示，并拦截调色逻辑与灰度/输出复用器之间的像素流。

通过 LCD 从接口可以访问所有光标编程寄存器。这亦提供了一个对光标图像 RAM 的读/写端口。

12.6.5.1 光标的操作

硬件光标位于一个双口 RAM 内。它由软件通过 AHB 从接口编程。AHB 从接口亦提供对硬件光标控制器的访问。通过这些寄存器能够修改光标的位置，以及完成其它功能。

在使能后，硬件光标会使用水平同步信号与垂直同步信号，还有一个像素时钟使能及各种显示参数，计算出当前的扫描坐标。

当显示点位于光标图像边界以内时，光标会用光标像素替代帧缓冲区的像素。

当显示了光标的最后像素时，产生一个中断，软件可以将此中断用作一个指示，这个指示表明此时修改光标图像是安全的。这样，软件就可以控制动画的完成，而不会出现帧同步光标闪烁的情况。

12.6.5.2 光标尺寸

支持两种光标尺寸，见[表 216](#)。

表216. STN 单色模式下的调色板数据存储

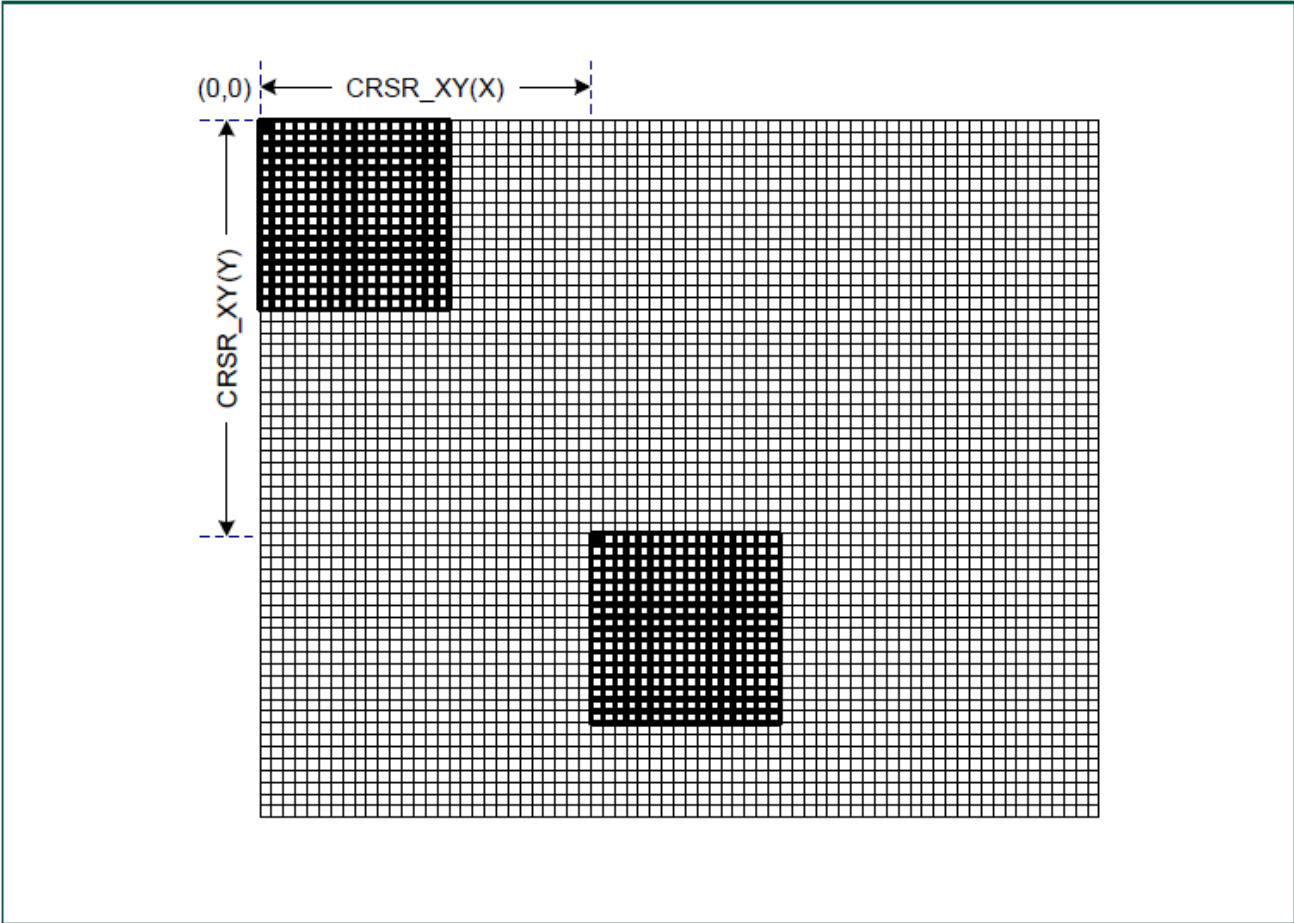
X 像素	Y 像素	每像素位数	每线字数	光标图像中的字数
32	32	2	2	64
64	64	2	4	256

12.6.5.3 光标的移动

接下来的描述假定屏幕和光标源均在可视屏幕的左上角（第一个可视像素每帧）。[图 31](#) 显

示了每个像素坐标是如何被假呈现在像素的左上角的。

图31. 光标移动



12.6.5.4 光标的 XY 定位

CRSR_XY 寄存器控制着在光标覆盖处的光标位置（见光标 XY 位置寄存器）。它为 X、Y 坐标提供了独立的字段。

CRSR_CFG 寄存器（见光标配置寄存器）提供了一个 FrameSync 位，用于控制光标的可见属性。

当 FrameSync 无效时，光标可对所编写 CRSR_XY 值的任何变化做出立即响应。如果光标移过 LCD 扫描线，可能会看到一些瞬时的拖尾效应。

当 FrameSync 有效时，光标只有在发生一次垂直同步时，才会刷新自己的位置。这就提供了干净的光标运动，但光标每帧只刷新一次。

12.6.5.5 光标的剪裁

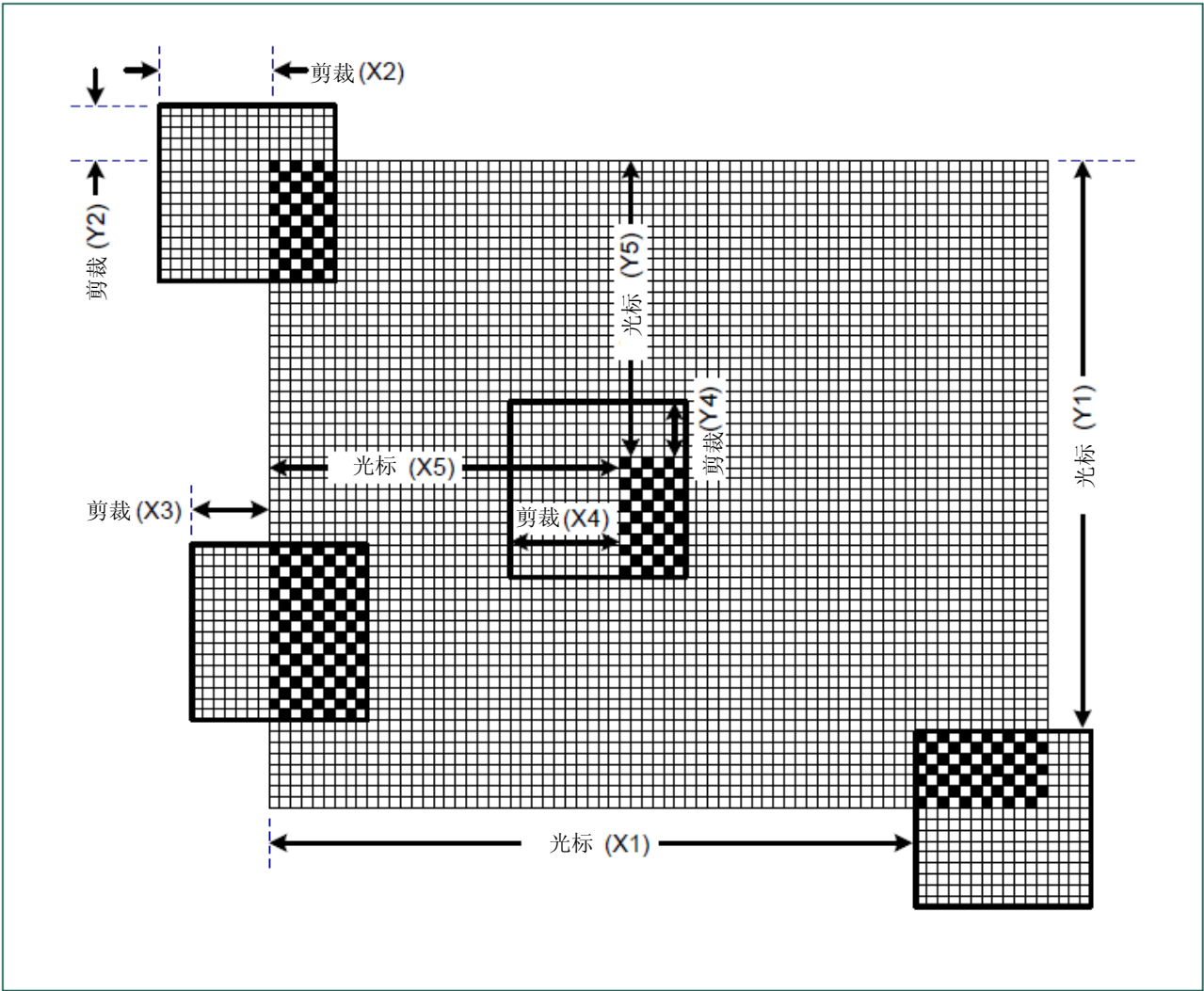
CRSR_XY 寄存器（见光标 XY 位置寄存器）使用正向二进制值编写，这样光标图像就能够

位于可见屏幕图像的任何位置。当光标位于屏幕极限处，超出了右侧或底侧的屏幕图像时，光标图像会自动剪裁（见图 32 中的 X1、Y1）。经检查的模式显示光标的可见部分。

因为 CRSR_XY 寄存器的值为正整数，为了模拟光标在屏幕左侧和顶侧的剪裁情况，提供了一个剪裁位置寄存器 CRSR_CLIP。该寄存器控制着光标图像的哪个点位于 CRSR_CLIP 坐标上。对于 Y 轴上的剪裁功能，CRSR_XY(X)为 0，而 Clip(X)被编程，以提供进入光标图像的偏移（X2 和 X3）。对显示屏顶部，提供了等效的功能，用于 X 轴的剪裁（Y2）。

对于没有在 X=0 或 Y=0 线上被剪裁的光标，用 0 编写剪裁位置寄存器的 X 和 X 字段，以正确地显示光标。错误编写的结果见剪裁（X4，Y4）。

图32. 光标剪裁



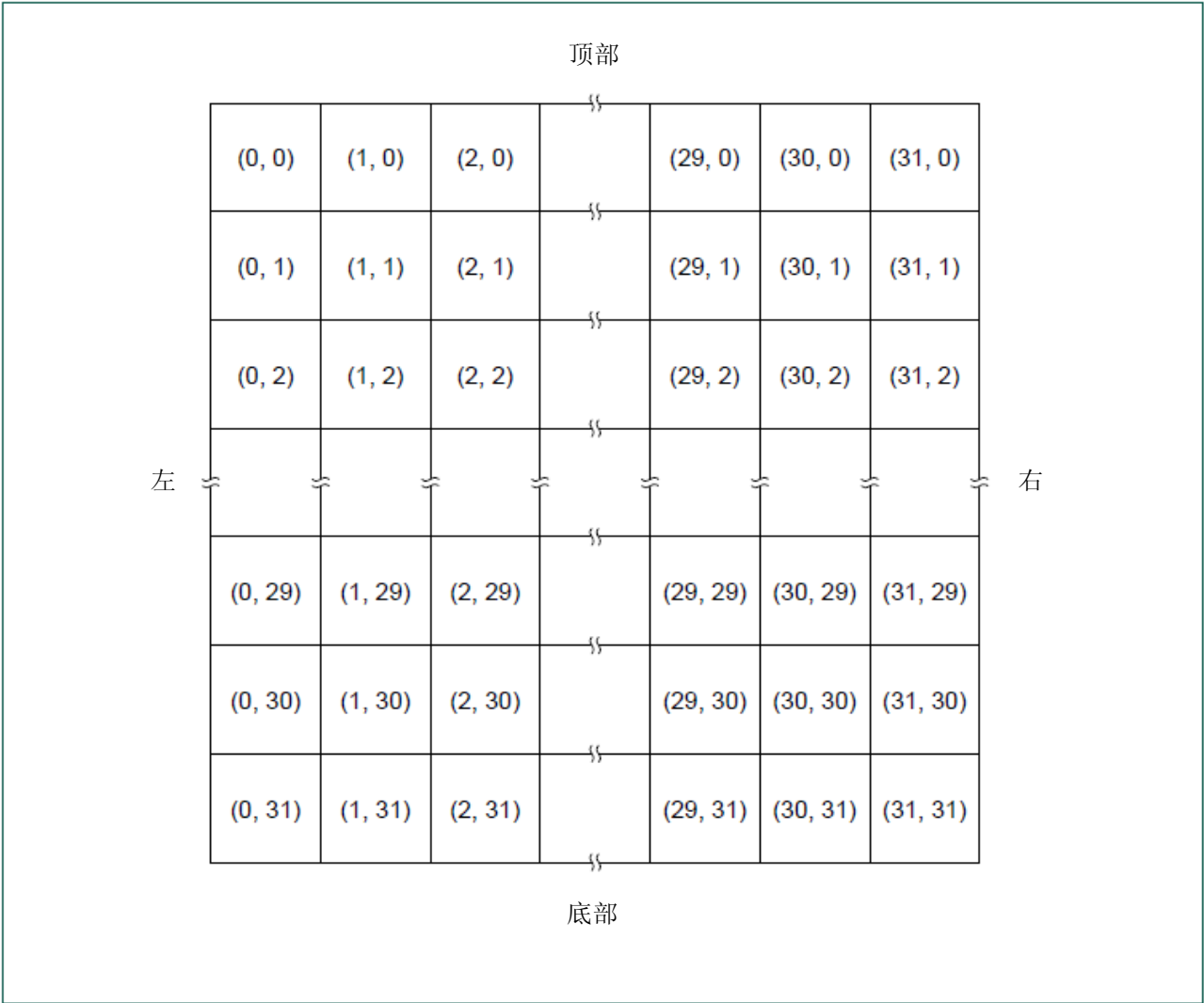
12.6.5.6 光标图像格式

LCD 帧缓冲区支持 3 种封包格式，但对硬件光标图像的要求已简化为只支持 LBBP。这是小端字节序、大端字节序，用于 Windows CE 模式。

图像 RAM 的起始地址是从 LCD 基址偏移 0x800，见本章中的寄存器描述。

显示光标的坐标系统以 (X, Y) 表示。64 x 64 是图 33 所示 32 x 32 格式的一个扩展。

图33. 光标图像格式



32 x 32 像素格式

存储器中保存了 4 种光标，每种有相同的像素格式。[表 217](#) 列出了 4 种光标的基址。

表217. 32 x 32 光标地址

地址	描述
0x2008 8800	光标 0 的起始地址。
0x2008 8900	光标 1 的起始地址。
0x2008 8A00	光标 2 的起始地址。
0x2008 8B00	光标 3 的起始地址。

[表 218](#) 给出了光标 0 缓冲到像素的映射。

表218. 32 x 32 像素光标格式下缓冲到像素映射

数据位	光标存储器的偏移量					
	0	4	(8 * y)	(8 * y) + 4	F8	FC
1:0	(3, 0)	(19, 0)	(3, y)	(19, y)	(3, 31)	(19, 31)
3:2	(2, 0)	(18, 0)	(2, y)	(18, y)	(2, 31)	(18, 31)
5:4	(1, 0)	(17, 0)	(1, y)	(17, y)	(1, 31)	(17, 31)
7:6	(0, 0)	(16, 0)	(0, y)	(16, y)	(0, 31)	(16, 31)
9:8	(7, 0)	(23, 0)	(7, y)	(23, y)	(7, 31)	(23, 31)
11:10	(6, 0)	(22, 0)	(6, y)	(22, y)	(6, 31)	(22, 31)
13:12	(5, 0)	(21, 0)	(5, y)	(21, y)	(5, 31)	(21, 31)
15:14	(4, 0)	(20, 0)	(4, y)	(20, y)	(4, 31)	(20, 31)
17:16	(11, 0)	(27, 0)	(11, y)	(27, y)	(11, 31)	(27, 31)
19:18	(10, 0)	(26, 0)	(10, y)	(26, y)	(10, 31)	(26, 31)
21:20	(9, 0)	(25, 0)	(9, y)	(25, y)	(9, 31)	(25, 31)
23:22	(8, 0)	(24, 0)	(8, y)	(24, y)	(8, 31)	(24, 31)
25:24	(15, 0)	(31, 0)	(15, y)	(31, y)	(15, 31)	(31, 31)
27:26	(14, 0)	(30, 0)	(14, y)	(30, y)	(14, 31)	(30, 31)
29:28	(13, 0)	(29, 0)	(13, y)	(29, y)	(13, 31)	(29, 31)
31:30	(12, 0)	(28, 0)	(12, y)	(28, y)	(12, 31)	(28,31)

64 x 64 像素格式

存储器空间中只有一个 64 x 64 的光标，详见[表 219](#)。

表219. 64x 64 像素光标格式下缓冲到像素映射

数据位	光标存储器的偏移量								FC
	0	4	8	12	(16 * y)	(16 * y) + 4	(16 * y) + 8	(16 * y) + 12	
1:0	(3, 0)	(19, 0)	(35, 0)	(51, 0)	(3, y)	(19, y)	(35, y)	(51, y)	(51, 63)
3:2	(2, 0)	(18, 0)	(34, 0)	(50, 0)	(2, y)	(18, y)	(34, y)	(50, y)	(50, 63)
5:4	(1, 0)	(17, 0)	(33, 0)	(49, 0)	(1, y)	(17, y)	(33, y)	(49, y)	(49, 63)
7:6	(0, 0)	(16, 0)	(32, 0)	(48, 0)	(0, y)	(16, y)	(32, y)	(48, y)	(48, 63)
9:8	(7, 0)	(23, 0)	(39, 0)	(55, 0)	(7, y)	(23, y)	(39, y)	(55, y)	(55, 63)
11:10	(6, 0)	(22, 0)	(38, 0)	(54, 0)	(6, y)	(22, y)	(38, y)	(54, y)	(54, 63)
13:12	(5, 0)	(21, 0)	(37, 0)	(53, 0)	(5, y)	(21, y)	(37, y)	(53, y)	(53, 63)
15:14	(4, 0)	(20, 0)	(36, 0)	(52, 0)	(4, y)	(20, y)	(36, y)	(52, y)	(52, 63)
17:16	(11, 0)	(27, 0)	(43, 0)	(59, 0)	(11, y)	(27, y)	(43, y)	(59, y)	(59, 63)
19:18	(10, 0)	(26, 0)	(42, 0)	(58, 0)	(10, y)	(26, y)	(42, y)	(58, y)	(58, 63)
21:20	(9, 0)	(25, 0)	(41, 0)	(57, 0)	(9, y)	(25, y)	(41, y)	(57, y)	(57, 63)
23:22	(8, 0)	(24, 0)	(40, 0)	(56, 0)	(8, y)	(24, y)	(40, y)	(56, y)	(56, 63)
25:24	(15, 0)	(31, 0)	(47, 0)	(63, 0)	(15, y)	(31, y)	(47, y)	(63, y)	(63, 63)
27:26	(14, 0)	(30, 0)	(46, 0)	(62, 0)	(14, y)	(30, y)	(46, y)	(62, y)	(62, 63)
29:28	(13, 0)	(29, 0)	(45, 0)	(61, 0)	(13, y)	(29, y)	(45, y)	(61, y)	(61, 63)
31:30	(12, 0)	(28, 0)	(44, 0)	(60, 0)	(12, y)	(28, y)	(44, y)	(60, y)	(60, 63)

光标像素编码

光标的每个像素都需要两位的信息。它们可被解析为彩色 0、彩色 1、透明，以及透明反转。

在编码方法里，位 1 在彩色与透明之间选择（AND 掩码），位 0 选择变型（XOR 掩码）。

[表 220](#) 给出了像素编码各位的定义。

表220. 像素编码

值	描述
00	彩色 0。光标按照 CRSR_PAL0 寄存器里所设定的红/绿/蓝（RGB）值来显示颜色。
01	彩色 1。光标按照 CRSR_PAL1 寄存器里所设定的 RGB 值来显示颜色。
10	透明。光标像素透明，因此显示的颜色没有变化。这使得可视光标的形状为非正方形。
11	透明反转。光标像素是显示出的帧像素的互补色。这可以确保光标可见，无论帧缓冲区分区图像是什么颜色。

12.6.6 灰度计

一个专利的灰度计算法用于驱动单色和彩色 STN 面板。它为单色显示器提供 15 个灰度级。对 STN 彩色显示器，则是为 3 个彩色分量（RGB）都提供灰度级。这样可供使用的颜色就有 3375（15x15x15）种。灰度计将每个 4 位灰度值转换为一个对多个帧的各像素活动的序列（某种程度上依赖于显示器的特性），以表示出灰度级和彩色。

12.6.7 上、下面板格式器

格式器用于 STN 模式，将灰度计输出转化为显示器需要的并行格式。对于单色显示器，位宽为 4 或 8 位，而对彩色显示器，则位宽为 8 位。[表 221](#) 显示了一个彩色显示器，它以一个重复序列中相当于 2 2/3 像素的数据为驱动。

表221. 2 2/3 像素数据驱动的彩色显示器

字节	CLD[7]	CLD[6]	CLD[5]	CLD[4]	CLD[3]	CLD[2]	CLD[1]	CLD[0]
0	P2[绿]	P2[红]	P1[蓝]	P1[绿]	P1[红]	P0[蓝]	P0[绿]	P0[红]
1	P5[红]	P4q[蓝]	P4[绿]	P4[红]	P3[蓝]	P3[绿]	P3[红]	P2[蓝]
2	P7[蓝]	P7[绿]	P7[红]	P6[蓝]	P6[绿]	P6[红]	P5[蓝]	P5[绿]

每个格式器都包括 3 个 3 位（RGB）左移位寄存器。来自灰度计的 RGB 像素数据位的值被同时移入相应的寄存器。当有足够数据时，就构建出一个字节，方法是将寄存器的数据复用到正确的位，以满足 LCD 面板的 RGB 数据模式。这个字节被发送至 3 字节的 FIFO，它有足够的空间存储 8 个彩色像素。

12.6.8 面板时钟发生器

面板时钟发生器模块的输出是面板时钟，即 LCD_DCLK 管脚。面板时钟可以基于 LCD 模块的外设时钟，也可以基于 LCD 的外部输入时钟，即 LCD_CLKIN 管脚。无论选择了哪个来源，都可以分频，从而产生出内部 LCD 时钟 LCDCLK。

通过对面板时钟发生器的编程，可以在 LCDCLK/2 至 LCDCLK/1025 的区间内，输出 LCD 面板时钟，以配合所用 LCD 面板的 bpp 数据速率。

LCD_POL 寄存器中的 CLKSEL 位决定了所使用的基时钟是 CCLK 管脚还是 LCD_CLKIN 管脚。

12.6.9 时序控制器

时序控制器模块的主要功能是生成面板的水平与垂直时序信号。它还提供面板偏压以及使能信号。这些时序均可通过寄存器编程。

12.6.10STN 与 TFT 的数据选择

提供对无源超扭曲向列（STN）以及有源薄膜晶体管（TFT）型 LCD 显示器的支持。

12.6.10.1 STN 显示器

STN 显示板需要计算像素模式的生成，为单色显示器提供伪灰度级，或为彩色显示器提供颜色创建。

12.6.10.2 TFT 显示器

TFT 显示面板需要将每个像素的数字颜色值加到显示数据输入端。

12.6.11 中断产生

LCD 控制器生成 4 个中断，以及一个联合中断。4 个中断是：

- 主机总线错误中断。
- 垂直比较中断。
- 下一个基址更新中断。
- FIFO 下溢中断。

4 个独立可屏蔽中断中，每一个均能通过 LCD_INT_MSK 寄存器中的屏蔽位被使能或禁用。这些中断还组合成一个总中断，如果这些中断中的任何一个中断有效和未屏蔽，则这个组合中断也有效。除了提供单个输出以外，还提供一个联合中断输出，这样在处理中断时，既能使用一个全局中断服务程序，也能使用模块的设备驱动。

单个中断源的状态可从 LCD_INTRAW 寄存器读出。

12.6.11.1 主机总线错误中断

当主接口在与从接口的一个传输期间收到了 ERROR 响应时，就产生一个主机总线错误中断。遇到这种错误时，主接口进入一个错误状态，直到收到错误清除信号。当相关的中断服务程序完成时，可以向 LCD_INTCLR 寄存器中的 BERIC 位写入 1，清除主机总线错误中断。这个动作将主接口从 ERROR 状态释放出来，启动 FRAME 状态，从而能够发起新的数据显示帧。

12.6.11.2 垂直比较中断

当 4 个垂直显示区中（用 LCD_CTRL 寄存器选择）有一个达到时，就产生垂直比较中断。可以让中断在以下的开始处产生：

- 垂直同步。
- 后沿。
- 活动视频。
- 前沿。

向 LCD_INTCLR 寄存器的 VcompIC 位写入 1，可以清除中断。

12.6.11.2.1 下一个基址更新中断

当 LCDUPBASE 或 LCDLPBASE 值已分别传送给 LCDUPCURR 或 LCDLPCURR 增量器时，就产生 LCD 下一个基址的更新中断。这给系统发送了信号：系统可以使用新帧的基址（如需要）安全地更新 LCDUPBASE 或 LCDLPBASE 寄存器。

向 LCD_INTCLR 寄存器的 LNBUIC 位写入 1，就可以清除中断。

12.6.11.2.2 FIFO 下溢中断

当一个空 DMA FIFO 请求内部数据时,就产生 FIFO 下溢中断。内部产生上、下面板的 DMA FIFO 下溢中断信号。

向寄存器 LCD_INTCLR 的 FUFIC 位写入 1, 就可以清除中断。

12.6.12 LCD 上电与掉电顺序

LCD 控制器上电时需要按照以下顺序进行:

1. 加电时, 下列信号保持为低电平:

- LCD_LP
- LCD_DCLK
- LCD_FP
- LCD_ENAB_M
- LCD_VD[23:0]
- LCD_LE

2. 当 LCD 功率稳定时, 向 LCD_CTRL 寄存器的 LcdEn 位写入 1, 这样会使能下列信号, 使其进入有效状态:

- LCD_LP
- LCD_DCLK
- LCD_FP
- LCD_ENAB_M
- LCD_LE

LCD_VD[23:0]信号维持在无效状态。

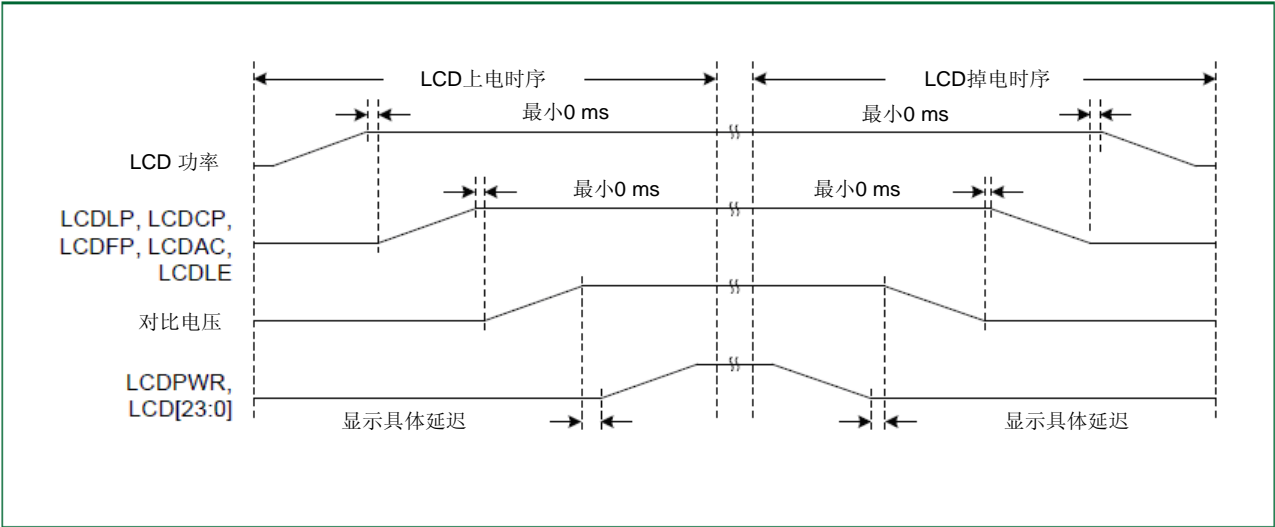
3. 当第 2 步中的信号稳定时, 为 LCD 面板加对比度电压 (LCD 控制器不控制或不提供这个电压)。

4. 如有必要, 可以用一个软件或硬件定时器, 在给面板加控制信号与加电源的中间, 加入一个具体的最小显示延迟时间。这个时间结束后, 再向 LCD_CTRL 寄存器中的 LcdPwr 位写 1 为面板加电, 同时将 LCD_PWR 信号设为高电平, 并将 LCD_VD[23:0]信号使能到有效状态。LCD_PWR 用于对 LCD 面板功率的控制。

掉电顺序与上述 4 个步骤顺序相反, 必须严格遵守, 此时是向相应的寄存器位中写入 0。

[图 34](#) 显示了上电与掉电顺序。

图34. 上电和掉电时序



12.7 寄存器描述

[表 222](#) 给出了与 LCD 控制器有关的寄存器，以及它们功能的综述。表后是每个寄存器的详情。

表222. LCD 控制器寄存器汇总

地址	名称	描述	复位值 ^[1]	访问	表
0x400F C1B8	LCD_CFG	LCD 配置和计时控制	0	R/W	表 223
0x2008 8000	LCD_TIMH	水平时序控制寄存器	0	R/W	表 224
0x2008 8004	LCD_TIMV	垂直时序控制寄存器	0	R/W	表 225
0x2008 8008	LCD_POL	时钟与信号极性控制寄存器	0	R/W	表 226
0x2008 800C	LCD_LE	线端控制寄存器	0	R/W	表 227
0x2008 8010	LCD_UPBASE	上面板帧基址寄存器	0	R/W	表 228
0x2008 8014	LCD_LPBASE	下面板帧基址寄存器	0	R/W	表 229
0x2008 8018	LCD_CTRL	LCD 控制寄存器	0	R/W	表 230
0x2008 801C	LCD_INTMSK	中断屏蔽寄存器	0	R/W	表 231
0x2008 8020	LCD_INTRAW	原始中断状态寄存器	0	RO	表 232
0x2008 8024	LCD_INTSTAT	被屏蔽中断状态寄存器	0	RO	表 233
0x2008 8028	LCD_INTCLR	中断清零寄存器	0	WO	表 234
0x2008 802C	LCD_UPCURR	上面板当前地址寄存器	0	RO	表 235
0x2008 8030	LCD_LPCURR	下面板当前地址寄存器	0	RO	表 236
0x2008 8200 - 0x2008 83FC	LCD_PAL	256 x 16 位彩色调色板寄存器	0	R/W	表 237
0x2008 8800 - 0x2008 8BFC	CRSR_IMG	光标图像寄存器	0	R/W	表 238
0x2008 8C00	CRSR_CTRL	光标控制寄存器	0	R/W	表 239
0x2008 8C04	CRSR_CFG	光标配置寄存器	0	R/W	表 240
0x2008 8C08	CRSR_PAL0	光标调色板寄存器 0	0	R/W	表 241
0x2008 8C0C	CRSR_PAL1	光标调色板寄存器 1	0	R/W	表 242
0x2008 8C10	CRSR_XY	光标 XY 位置寄存器	0	R/W	表 243
0x2008 8C14	CRSR_CLIP	光标剪裁位置寄存器	0	R/W	表 244
0x2008 8C20	CRSR_INTMSK	光标中断屏蔽寄存器	0	R/W	表 245
0x2008 8C24	CRSR_INTCLR	光标中断清零寄存器	0	WO	表 246
0x2008 8C28	CRSR_INTRAW	光标原始中断状态寄存器	0	RO	表 247
0x2008 8C2C	CRSR_INTSTAT	光标屏蔽中断状态寄存器	0	RO	表 248

[1] 复位值仅反映在使用位中保存的数据，它不包括保留位的内容。

12.7.1 LCD 配置寄存器（LCD_CFG, RW—0x400F C1B8）

LCD_CFG 寄存器控制着用于 LCD 数据生成的时钟预分频（prescaling）。

LCD_CFG 寄存器的内容描述见[表 223](#)。

表223. LCD 配置寄存器（LCD_CFG, RW — 0x400F C1B8）

位	功能	描述	复位值
4:0	CLKDIV	LCD 面板时钟预分频器选择。 该寄存器的值加 1 被用来对选中的输入时钟进行分频（参见 LCD_POL 寄存器中的 CLKSEL 位），以生成面板时钟。	0
31:5	-	保留。读取值未定义，只写入 0。	-

12.7.2 水平时序寄存器（LCD_TIMH, RW — 0x2008 8000）

LCD_TIMH 寄存器控制水平同步脉冲宽度（HSW）、水平前沿（HFP）周期、水平后沿（HBP）周期，以及每行像素（PPL）。

LCD_TIMH 寄存器的内容描述见[表 224](#)。

表224. 水平时序寄存器（LCD_TIMH, RW — 0x2008 8000）

位	功能	描述	复位值
1:0	-	保留。读取值未定义，只写入 0。	-
7:2	PPL	每行像素。PPL 位字段指定了屏幕上每一线或行中像素的数目。PPL 是一个 6 位的值，它代表每行有 16 到 1024 个像素。PPL 对 HFP 应用之前像素时钟的数目进行计数。该位的值为要求值经 16 分频，再减 1。每行像素的实际数目 = 16 x (PPL + 1)。例如，为了实现每行 320 个像素，将 PPL 的程序设为 (320/16) - 1 = 19。	0
15:8	HSW	水平同步脉冲宽度。 8 位的 HSW 字段指定了被动模式下线时钟的脉冲宽度，或主动模式下水平同步脉冲。用期望值减 1 进行程序设定。	0
23:16	HFP	水平前沿。8 位的 HFP 字段设定了在 LCD 总线时钟脉冲之前，每一线或行像素结尾的像素时钟间隔数目。当一整行像素被发送至 LCD 驱动时，HFP 中的值将对像素时钟的数目进行计数并等待总线时钟生效。HFP 可以生成一个含有 1 到 256 像素时钟周期的周期。用期望值减 1 进行程序设定。	0
31:24	HBP	水平后沿。 8 位的 HBP 字段用来指定在每一线或行像素开始时的像素时钟周期的数目。在前一行中的总线时钟失效之后，HBP 中的值将对像素时钟的数目进行计数并等待下一显示行启动。HBP 可以生成一个含有 1 到 256 像素时钟周期的延迟。用期望值减 1 进行程序设定。	0

12.7.2.1 水平时序限制

DMA 在一条水平显示线的开始请求新的数据。必须给 LCD 接口上的 DMA 传输以及传播到 FIFO 通道的数据提供一些时间。数据通道等待时间限制了 STN 模式下的水平沿宽度可用最小值。最小值为 HSW = 2 和 HBP = 2。

单面板模式：

- HSW = 3 个像素时钟周期

- HBP = 5 个像素时钟周期
- HFP = 5 个像素时钟周期
- 面板时钟分频器 (PCD) = 1 (LCDCLK / 3)

双面板模式：

- HSW = 3 个像素时钟周期
- HBP = 5 个像素时钟周期
- HFP = 5 个像素时钟周期
- PCD = 5 (LCDCLK / 7)

如果在显示线开始时给出了足够的时间，例如，设定 HSW = 6、HBP = 10，则对 PCD = 4 的最小值，数据不会损坏。

12.7.3 垂直时序寄存器 (LCD_TIMV, RW — 0x2008 8004)

LCD_TIMV 寄存器控制着垂直同步脉冲宽度 (VSW)、垂直前沿 (VFP) 周期、垂直后沿 (VBP) 周期，以及每面板线数 (LPP)。

LCD_TIMV 寄存器的内容描述见[表 225](#)。

表225. 垂直时序寄存器 (LCD_TIMV, RW — 0x2008 8004)

位	功能	描述	复位值
9:0	LPP	每面板线数。 该位表示每个显示屏上的有效线数目。LPP 字段指定了被控制的 LCD 面板上的线或行数目。LPP 是一个 10 位的介于 1 和 1024 线之间的值。用每 LCD 面板线数减 1 对该寄存器进行程序设定。对于双面板显示器，用在每个上和下面板上的线数对该寄存器进行程序设定。	0
15:10	VSW	垂直同步脉冲宽度。 该位表示水平同步线数。6 位的 VSW 字段指定了垂直同步脉冲的宽度。用要求的线数减 1 对该寄存器进行程序设定。水平同步线数对于无源 STN LCD 而言必须要小（例如，可以将其设为 0）。值越高，在 STN LCD 上的对比度越差。	0
23:16	VFP	垂直前沿。 该位表示在垂直同步周期之前，每个帧结尾的无效线数目。8 位的 VFP 字段指定了嵌在每个帧结尾的线时钟数目。当一整帧像素被发送至 LCD 显示器上时，VFP 中的值将对需等待的线时钟周期的数目进行计数。在计数结束后，垂直同步信号 LCD_FP 在主动模式下被激活，或者按照 VSW 位字段所指定的那样额外的线时钟在被动模式下被激活。VFP 生成 0 到 255 个线时钟周期。在被动式显示器上将该位置成 0 以提高对比度。	0
31:24	VBP	垂直后沿。 该位表示在垂直同步周期之后，每个帧起始时的无效线数目。8 位的 VBP 字段指定了嵌在每个帧起始的线时钟数目。VBP 计数在前一帧的垂直同步信号在主动模式下被取消之后，或者按照 VSW 位字段所指定的那样额外的线时钟在被动模式下被嵌入后马上开始。在这之后，VBP 中的计数值设定了嵌入在下一帧之前的线时钟周期的数目。VBP 生成 0 到 255 个额外的线时钟周期。在被动式显示器上将该位置成 0 以提高对比度。	0

12.7.4 时钟与信号极性寄存器（LCD_POL，RW — 0x2008 8008）

LCD_POL 寄存器控制着时钟时序与信号极性等细节。

LCD_POL 寄存器的内容描述见[表 226](#)。

表226. 时钟和信号极性寄存器（LCD_POL，RW — 0x2008 8008）

位	功能	描述	复位值
4:0	PCD_LO	面板时钟分频器的下 5 位。 10 位的包含 PCD_HI（该寄存器的位 31:27）和 PCD_LO 的 PCD 字段被用来从输入时钟 LCD_DCLK = LCDCLK/(PCD+2)上生成 LCD 面板时钟频率 LCD_DCLK。 对于带有一个 4 或 8 位接口的单色 STN 显示器，面板时钟是实际单个像素时钟率的四分之一或八分之一。对于彩色 STN 显示器，每个 LCD_DCLK 周期输出 22/3 个像素，因此面板时钟是像素率的 0.375 倍。 对于 TFT 显示器，像素时钟分频器可以通过设定该寄存器中的 BCD 位而被旁路。 注：数据通道等待时间限制了 STN 模式下面板时钟分频器的可用最小值： 单面板彩色模式，PCD = 1（LCD_DCLK = LCDCLK/3）。 双面板彩色模式，PCD = 4（LCD_DCLK = LCDCLK/6）。 单面板单色 4 位接口模式，PCD = 2（LCD_DCLK = LCDCLK/4）。 双面板单色 4 位接口模式和单面板单色 8 位接口模式，PCD = 6（LCD_DCLK = LCDCLK/8）。 双面板单色 8 位接口模式，PCD = 14（LCD_DCLK = LCDCLK/16）。	0
5	CLKSEL	时钟选择。 该位控制 LCDCLK 源的选择。 0 表示 LCD 模块的时钟源为 CCLK。 1 表示 LCD 模块的时钟源为 LCD_CLKIN（LVD 的外部时钟输入）。	0
10:6	ACB	AC 偏压管脚频率。 AC 偏压管脚频率仅应用于 STN 显示器。这些显示器要求像素电压极性周期性地转换以防止 DC 电荷积聚而造成危害。用要求值减 1 对这一字段进行设定，以在 AC 偏压管脚 LCD_ENAB_M 的各个反转之间应用线时钟数目。当 LCD_ENAB_M 管脚被用作数据使能信号时，如果 LCD 在 TFT 模式下处于工作状态，则该字段无效。	0
11	IVS	反相垂直同步。 IVS 位将 LCD_FP 信号的极性反相。 0 表示 LCD_FP 管脚为高电平有效，低电平无效。 1 表示 LCD_FP 管脚为低电平有效，高电平无效。	0
12	IHS	反相水平同步。 IHS 位将 LCD_LP 信号的极性反相。 0 表示 LCD_LP 管脚为高电平有效，低电平无效。 1 表示 LCD_LP 管脚为低电平有效，高电平无效。	0
13	IPC	反相面板时钟。 IPC 位选择面板时钟沿，在这个沿上像素数据被驱动到 LCD 数据线上。 0 表示数据被驱动到 LCD_DCLK 上升沿的 LCD 数据线上。 1 表示数据被驱动到 LCD_DCLK 下降沿的 LCD 数据线上。	0
14	IOE	反相输出使能。 该位选择在 TFT 模式下的输出使能信号的有效极性。在该模式下，当有效显示数据可用时，LCD_ENAB_M 管脚被用来使能 LCD 面板。在有效显示器模式下，当 LCD_ENAB_M 处于激活状态时，数据被驱动到 LCD_DCLK 被编程沿的 LCD 数据线上。 0 表示在 TFT 模式下，LCD_ENAB_M 输出管脚为高电平有效。 1 表示在 TFT 模式下，LCD_ENAB_M 输出管脚为低电平有效。	0

位	功能	描述	复位值
15	-	保留。读取值未定义，只写入 0。	-
25:16	CPL	每线时钟。 该字段指定了每线上 LCD 面板相连的实际 LCD_DCLK 时钟数目。该字段的值等于 PPL 的数目除以 1（针对 TFT 模式）、4 或 8（针对单色无源模式），2 2/3（针对彩色无源模式），减 1。为了使 LCD 显示器能正常工作，必须准确地在 LCD_TIMH 寄存器中设定该字段和 PPL 位。	0
26	BCD	旁路像素时钟分频器。 将该位设成 1 将旁路像素时钟分频器逻辑。该位主要用于 TFT 显示器。	0
31:27	PCD_HI	面板时钟分频器的上 5 位。 参见该寄存器位 [4:0]中的 PCD_LO 描述。	0

12.7.5 线端控制寄存器（LCD_LE，RW — 0x2008 800C）

LCD_LE 寄存器控制着线端信号 LCD_LE 的使能。当信号使能时，在每个显示线的最后像素上，经过一个可编程的延迟后，LCD_LE 上输出 4 个 LCDCLK 周期宽度的正脉冲。如果线端信号被禁用，则永远保持为低电平。
LCD_LE 寄存器的内容描述见[表 227](#)。

表227. 线端控制寄存器（LCD_LE，RW — 0x2008 800C）

位	功能	描述	复位值
6:0	LED	线端延迟。 控制从最后一个面板时钟 LCD_DCLK 的上升沿的线端信号延迟。用 LCDCLK 时钟周期数减 1 对该位进行程序设定。	0
15:7	-	保留。读取值未定义，只写入 0。	-
16	LEE	LCD 线端使能。 0 表示 LCD_LE 被禁能（维持低电平）。 1 表示 LCD_LE 信号有效。	0
31:17	-	保留。读取值未定义，只写入 0。	-

12.7.6 上面板帧基址寄存器（LCD_UPBASE，RW — 0x2008 8010）

LCD_UPBASE 寄存器是彩色 LCD 上面板的 DMA 基址寄存器，用于为上面板的帧缓冲区编写基址。LCD_UPBASE（以及双面板时的 LCD_LPBase）都必须在使能 LCD 控制器以前初始化。基址必须按双字（doubleword）对齐。

可选情况下，该值也可以在帧中改变，从而建立双缓冲的视频显示。这些寄存器均在每个 LCD 垂直同步时，复制到当时的相应寄存器上。这个事件会置位 LNBU 位，并产生一个可选中断。在生成双缓冲视频时，此中断可以用于重新编写基址。

LCD_UPBASE 寄存器的内容描述见[表 228](#)。

表228. 上面板帧基址寄存器（LCD_UPBASE，RW — 0x2008 8010）

位	功能	描述	复位值
2:0	-	保留。读取值未定义，只写入 0。	-
31:3	LCDUPBASE	LCD 上面板基址。 该位是存储器中的上面板帧数据的起始地址，双字对齐。	0

12.7.7 下面板帧基址寄存器（LCD_LPBASE，RW — 0x2008 8014）

LCD_LPBASE 寄存器是彩色 LCD 下面板的 DMA 基址寄存器，用于为下面板的帧缓冲区编写基址。LCD_LPBASE 必须在使能 LCD 控制器以前初始化。基址必须按双字对齐。

可选情况下，该值也可以在帧中改变，从而建立双缓冲的视频显示。这些寄存器均在每个 LCD 垂直同步时，复制到当时的相应寄存器上。这个事件会置位 LNBU 位，并产生一个可选中断。在生成双缓冲视频时，此中断可以用于重新编写基址。

LCD_LPBASE 寄存器的内容描述见[表 229](#)。

表229. 下面板帧基址寄存器（LCD_LPBASE，RW — 0x2008 8014）

位	功能	描述	复位值
2:0	-	保留。读取值未定义，只写入 0。	-
31:3	LCDLPBASE	LCD 下面板基址。 该位是存储器中的下面板帧数据的起始地址，双字对齐。	0

12.7.8 LCD 控制寄存器（LCD_CTRL，RW — 0x2008 8018）

LCD_CTRL 寄存器控制 LCD 的工作模式以及面板像素参数。

LCD_CTRL 寄存器的内容描述见[表 230](#)。

表230. LCD 控制寄存器（LCD_CTRL，RW — 0x2008 8018）

位	功能	描述	复位值
0	LcdEn	LCD 使能控制位。 0 表示 LCD 被禁能。LCD_LP、LCD_DCLK、LCD_FP、LCD_ENAB_M 和 LCD_LE 为低电平有效信号。 1 表示 LCD 被使能。LCD_LP、LCD_DCLK、LCD_FP、LCD_ENAB_M 和 LCD_LE 为高电平有效信号。 关于 LCD 功率排序的详细内容，参见 LCD 上电和掉电序列。	0
3:1	LcdBpp	每像素 LCD 位： 选择每 LCD 像素的位数目： 000 = 1 bpp。 001 = 2 bpp。 010 = 4 bpp。 011 = 8 bpp。 100 = 16 bpp。 101 = 24 bpp（仅针对 TFT 面板）。 110 = 16 bpp，5:6:5 模式。 111 = 12 bpp，4:4:4 模式。	0
4	LcdBW	STN LCD 单色/彩色选择。 0 表示 STN LCD 为彩色。 1 表示 STN LCD 为单色。 该位在 TFT 模式下没有意义。	0
5	LcdTFT	LCD 面板 TFT 类型选择。 0 表示 LCD 是一个 STN 显示器。使用灰度计。 1 表示 LCD 是一个 TFT 显示器。不使用灰度计	0
6	LcdMono8	单色 LCD 接口宽度。 该位控制一个单色 STN LCD 是否使用 4 或 8 位并行接口。该位在其他模式下没有意义，且必须被设为 0。 0 表示单色 LCD 使用一个 4 位接口。 1 表示单色 LCD 使用一个 8 位接口。	0
7	LcdDual	单或双 LCD 面板选择。 STN LCD 接口： 0 表示单面板。 1 表示双面板。	0

位	功能	描述	复位值
8	BGR	彩色格式选择。 0 = RGB：正常输出。 1 = BGR：红蓝交换。	0
9	BEBO	大端字节顺序。 控制存储器中的字节排序。 0 表示小端字节序。 1 表示大端字节序。	0
10	BEPO	大端像素排序。 控制一个字节中的像素排序。 0 表示一个字节中的小端字节排序。 1 表示一个字节中的大端字节排序。 BEPO 位在 1、2 和 4 bpp 显示器模式下的小端和大端像素包之间进行选择，在 8 或 16 bpp 显示器模式下该位无效。 关于数据格式的更多信息，参见像素串行器。	0
11	LcdPwr	LCD 功率使能。 0 表示电源没有通到 LCD 面板上，LCD_VD[23:0]信号被禁能，（维持低电平）。 1 表示电源通到 LCD 面板上，LCD_VD[23:0]信号被使能，（激活）。关于 LCD 功率排序的详细内容，参见 LCD 上电和掉电序列。	0
13:12	LcdVComp	LCD 垂直比较中断。 生成 VComp 中断在： 00 = 垂直同步起始。 01 = 后沿起始。 10 = 活动视频起始。 11 = 前沿起始。	0
15:14	-	保留。读取值未定义，只写入 0。	-
16	WATERMARK	LCD DMA FIFO 水印（watermark）水平。 DMA 请求生成时控制。 0 表示 DMA FIFO 有 4 个或更多空位时，生成一个 LCD DMA 请求。 1 表示 DMA FIFO 有 8 个或更多空位时，生成一个 LCD DMA 请求。	0
31:17	-	保留。读取值未定义，只写入 0。	-

12.7.9 中断屏蔽寄存器（LCD_INTMSK, RW — 0x2008 801C）

LCD_INTMSK 寄存器控制各个 LCD 中断是否发生。置位此寄存器中的位能够将相应的原始中断 LCD_INTRAW 状态位值传递给 LCD_INTSTAT 寄存器以作为中断进行处理。

LCD_INTMSK 寄存器的内容描述见[表 231](#)。

表231. 中断屏蔽寄存器（LCD_INTMSK, RW — 0x2008 801C）

位	功能	描述	复位值
0	-	保留。读取值未定义，只写入 0。	-
1	FUFIM	FIFO 下溢中断使能。 0: FIFO 下溢中断被禁能。 1: FIFO 下溢时产生中断。	0
2	LNBUIM	LCD 下一基址更新中断使能。 0: 基址更新中断被禁能。 1: 当 LCD 基址寄存器从下一地址寄存器更新时产生中断。	0
3	VCompIM	垂直比较中断使能。 0: 垂直比较时间中断被禁能。 1: 当垂直比较时间（由 LCD_CTRL 寄存器中的 LcdVComp 字段定义）达到时产生中断。	0
4	BERIM	AHB 主错误中断使能。 0: AHB 主错误中断被禁能。 1: 当 AHB 主错误发生时产生中断。	0
31:5	-	保留。读取值未定义，只写入 0。	-

12.7.10 原始中断状态寄存器（LCD_INTRAW, RW — 0x2008 8020）

LCD_INTRAW 寄存器包含了各种 LCD 控制器事件的状态标志。如果被 LCD_INTMSK 寄存器中的屏蔽位使能，则这些标志可以产生一个中断。

LCD_INTRAW 寄存器的内容描述见[表 232](#)。

表232. 原始中断状态寄存器（LCD_INTRAW, RW — 0x2008 8020）

位	功能	描述	复位值
0	-	保留。读取值未定义，只写入 0。	-
1	FUFRIS	FIFO 下溢原始中断状态。 当空导致一个下溢条件发生时，上或下 DMA FIFO 已经被读访问时置位。 LCD_INTMSK 寄存器中的 FUFIM 位置位时产生中断。	
2	LNBURIS	LCD 下一基址更新原始中断状态。 取决于模式。当前基址寄存器成功被下一地址寄存器更新时该位置位。该位表示如果使用了双缓冲，一个新的下一地址将能够被加载。 LCD_INTMSK 寄存器中的 LNBUIM 位置位时产生中断。	0
3	VCompRIS	垂直比较原始中断状态。 按 LCD_CTRL 寄存器中的 LcdVComp 位所选择的，当到达 4 个垂直区中的一个区，该位置位。 LCD_INTMSK 寄存器中的 VCompIM 位置位时产生中断。	0
4	BERRAW	AHB 主机总线错误原始中断状态。 当 AHB 主接口从一个从接口接收到一个总线错误响应时该位置位。LCD_INTMSK 寄存器中的 BERIM 位置位时产生中断。	0
31:5	-	保留。读取值未定义，只写入 0。	-

12.7.11 被屏蔽中断状态寄存器（LCD_INTSTAT, RW — 0x2008 8024）

LCD_INTSTAT 寄存器为只读寄存器，包括了对 LCD_INTRAW 寄存器与 LCD_INTMASK 寄存器的逐位逻辑相“与”（AND）运算。所有中断的逻辑相“或”（OR）运算被提供给系统中断控制器。

LCD_INTSTAT 寄存器的内容描述见[表 233](#)。

表233. 被屏蔽中断状态寄存器（LCD_INTSTAT, RW — 0x2008 8024）

位	功能	描述	复位值
0	-	保留。从保留位读出的值未被定义。	-
1	FUFMIS	FIFO 下溢屏蔽中断状态。 当 LCD_INTRAW 寄存器中的 FUFRIS 位和 LCD_INTMSK 寄存器中的 FUFIM 位同时置位时该位置位。	0
2	LNBUMIS	LCD 下一基址更新屏蔽中断状态。 当 LCD_INTRAW 寄存器中的 LNBURIS 位和 LCD_INTMSK 寄存器中的 LNBUIM 位同时置位时该位置位。	0
3	VCompMIS	垂直比较屏蔽中断状态。 当 LCD_INTRAW 寄存器中的 VCompRIS 位和 LCD_INTMSK 寄存器中的 VCompIM 位同	0

位	功能	描述	复位值
		时置位时该位置位。	
4	BERMIS	AHB 主机总线错误屏蔽中断状态。 当 LCD_INTRAW 寄存器中的 BERRAW 位和 LCD_INTMSK 寄存器中的 BERIM 位同时置位时该位置位。	0
31:5	-	保留。读取值未定义，只写入 0。	-

12.7.12 中断清零寄存器（LCD_INTCLR, RW — 0x2008 8028）

LCD_INTCLR 寄存器为只写寄存器。向相应位写入逻辑 1，可清除相应的中断。

LCD_INTCLR 寄存器的内容描述见[表 234](#)。

表234. 中断清零寄存器（LCD_INTCLR, RW — 0x2008 8028）

位	功能	描述	复位值
0	-	保留。读取值未定义，只写入 0。	-
1	FUFIC	FIFO 下溢中断清除。 向该位写入 1 清除 FIFO 下溢中断。	0
2	LNBUIC	LCD 下一基址更新中断清除。 向该位写入 1 清除 LCD 下一基址更新中断。	0
3	VCompIC	垂直比较中断清除。 向该位写入 1 清除垂直比较中断。	0
4	BERIC	AHB 主错误中断清除。 向该位写入 1 清除 AHB 主错误中断。	0
31:5	-	保留。读取值未定义，只写入 0。	-

12.7.13 上面板当前地址寄存器（LCD_UPCURR, RW — 0x2008 802C）

LCD_UPCURR 寄存器为只读寄存器，包含了读取时上面板数据 DMA 的近似地址值。

注：此寄存器可以在任何时候修改，因此只用于显示位置的粗略指示。

LCD_UPCURR 寄存器的内容描述见[表 235](#)。

表235. 上面板当前地址寄存器（LCD_UPCURR, RW — 0x2008 802C）

位	功能	描述	复位值
31:0	LCDUPCURR	LCD 上面板当前地址。 包含当前 LCD 上面板数据 DMA 地址。	0

12.7.14 下面板当前地址寄存器（LCD_LPCURR, RW - 0x2008 8030）

LCD_LPCURR 寄存器为只读寄存器，包含了读取时上面板数据 DMA 的近似地址值。

注：此寄存器可以在任何时候修改，因此只用于显示位置的粗略指示。

LCD_LPCURR 寄存器的内容描述见[表 236](#)。

表236. 下面板当前地址寄存器（LCD_LPCURR, RW — 0x2008 8030）

位	功能	描述	复位值
31:0	LCDLPCURR	LCD 下面板当前地址。 包含当前 LCD 下面板数据 DMA 地址。	0

12.7.15 彩色调色板寄存器（LCD_PAL, RW — 0x2008 8200 to 0x2008 83FC）

LCD_PAL 寄存器包含了 256 个调色板表项，组织成每字两个表项的 128 个位置。

每个字的位置都包含两个调色板表项。这意味着，调色板使用了 128 个字的位置。当配置为小端字节序时，位[15:0]是低编码的调色板表项，位[31:16]是高编码的调色板表项。当配置为大端字节序时，顺序颠倒，即位[31:16]是低编码的调色板表项，位[15:0]是高编码的调色板表项。

注：只有 TFT 显示器会用到全部的调色板表项位。

LCD_PAL 寄存器的内容描述见[表 237](#)。

表237. 彩色调色板寄存器（LCD_PAL, RW — 0x2008 8200 到 0x2008 83FC）

位	功能	描述	复位值
4:0	R[4:0]	红色调色板数据。 对于 STN 显示器，只有 4 个位为[4:1]的 MSB 被使用。对于单色显示器，只有红色调色板数据被使用。所有的调色板寄存器都使用同样的位字段。	0
9:5	G[4:0]	绿色调色板数据。	0
14:10	B[4:0]	蓝色调色板数据。	0
15	I	亮度/未使用位。 可以被用作 6:6:6 模式下 TFT 显示器上红、绿和蓝输入的 LSB，使颜色数目翻倍至 64K，每种颜色有两种不同的亮度。	0
20:16	R[4:0]	红色调色板数据。 对于 STN 显示器，只有 4 个位为[4:1]的 MSB 被使用。对于单色显示器，只有红色调色板数据被使用。所有的调色板寄存器都使用同样的位字段。	0
25:21	G[4:0]	绿色调色板数据。	0
30:26	B[4:0]	蓝色调色板数据。	0
31	I	亮度/未使用位。 可以被用作 6:6:6 模式下 TFT 显示器上红、绿和蓝输入的 LSB，使颜色数目翻倍至 64K，每种颜色有两种不同的亮度。	0

12.7.16 光标图像寄存器（CRSR_IMG, RW — 0x2008 8800 至 0x2008 8BFC）

CRSR_IMG 寄存器区包含 256 个字宽的值，被硬件光标机制用于定义图像或重叠在显示器上的图像。图像必须永远以 LBBP 模式（小端字节，大端像素）保存，如 12.6.5.6 节所述。两个位用于光标中每个像素颜色与透明度的编码。

根据 CRSR_CFG 寄存器中位 0 的状态（见光标配置寄存器的描述），光标图像 RAM 可包含 4 个 32x32 光标图像，或 1 个 64x64 光标图像。

光标定义的颜色被映射到 CRSR_PAL0 和 CRSR_PAL0 寄存器的值上（见光标调色板寄存器的描述）。

CRSR_IMG 寄存器的内容描述见表 238。

表238. 光标图像寄存器（CRSR_IMG, RW — 0x2008 8800 到 0x2008 8BFC）

位	功能	描述	复位值
31:0	CRSR_IMG	光标图像数据。 光标图像寄存器的 256 个字定义了一个 64x64 光标或 4 个 32x32 光标的外观。	0

12.7.17 光标控制寄存器（CRSR_CTRL, RW — 0x2008 8C00）

CRSR_CTRL 寄存器提供了对常用光标功能的访问，如光标的显示开/关控制，以及光标号。

如果选择的是 32x32 光标，则可以在 4 种 32x32 光标中使能一种。每种图像占据四分之一的图像存储器，光标 0 从地址 0 起，然后光标 1 从地址 0x100 起，光标 2 从地址 0x200 起，光标 3 从地址 0x300 起。如果选择的是 64x64 光标，则只有一个光标适合图像缓冲区，没有其他选择。

适用于光标坐标的类似帧同步规则也适用于光标号。如果 CsrFramesync 为 1，则只有在垂直帧消隐周期内，才能改变显示的光标图像。如果 CsrFramesync 为 0，则光标图像索引会立即改变，即使光标正在被扫描。

CRSR_CTRL 寄存器的内容描述见表 239。

表239. 光标控制寄存器（CRSR_CTRL, RW — 0x2008 8C00）

位	功能	描述	复位值
0	CsrOn	光标使能。 0 表示光标未显示。 1 表示光标被显示。	0
3:1	—	保留。读取值未定义，只写入 0。	0
5:4	CsrNum[1:0]	光标图像号。 如果所选光标的尺寸为 64x64，该字段无效。如果所选光标的尺寸为 32x32： 00 = 光标 0。 01 = 光标 1。	0

位	功能	描述	复位值
31:6	—	10 = 光标 2。	0
		11 = 光标 3。	
		保留。读取值未定义，只写入 0。	

12.7.18 光标配置寄存器（CRSR_CFG, RW — 0x2008 8C04）

CRSR_CFG 寄存器为硬件光标提供所有配置信息。

CRSR_CFG 寄存器的内容描述见[表 240](#)。

表240. 光标配置寄存器（CRSR_CFG, RW — 0x2008 8C04）

位	功能	描述	复位值
0	CrsrSize	光标尺寸选择。 0 表示 32x32 像素光标。考虑到 4 个定义光标。 1 表示 64x64 像素光标。	0
1	FrameSync	光标帧同步类型。 0 表示光标协同是异步的。 1 表示光标协同与帧同步脉冲同步。	0
31:2	-	保留。读取值未定义，只写入 0。	-

12.7.19 光标调色板寄存器 0（CRSR_PAL0, RW— 0x2008 8C08）

光标调色板寄存器提供了光标可见颜色的彩色调色板信息。Color0 通过 CRSR_PAL0 映射。

此寄存器以帧缓冲区调色板输出所显示的相同方式，根据 LCD 面板的性能，提供要显示的 24 位 RGB 值。

在单色 STN 模式下，只使用了红色域的前 4 位。在 STN 彩色模式下，使用了红、蓝和绿域的前 4 位。在每像素 24 位模式下，调色板寄存器的所有 24 位都是有效的。

CRSR_PAL0 寄存器的内容描述见[表 241](#)。

表241. 光标调色板寄存器 0（CRSR_PAL0, RW — 0x2008 8C08）

位	功能	描述	复位值
7:0	红	红色元件	0
15:8	绿	绿色元件	0
23:16	蓝	蓝色元件。	0
31:24	-	保留。读取值未定义，只写入 0。	-

12.7.20 光板调色板寄存器 1（CRSR_PAL1, RW — 0x2008 8C0C）

光标调色板寄存器提供了光标可见颜色的彩色调色板信息。Color1 通过 CRSR_PAL1 映射。

此寄存器以帧缓冲区调色板输出所显示的相同方式，根据 LCD 面板的性能，提供要显示的 24 位 RGB 值。

在单色 STN 模式下，只使用了红色域的前 4 位。在 STN 彩色模式下，使用了红、蓝和绿域的前 4 位。在每像素 24 位模式下，调色板寄存器的所有 24 位都是有效的。

CRSR_PAL1 寄存器的内容描述见[表 242](#)。

表242. 光标调色板寄存器 1（CRSR_PAL1，RW — 0x2008 8C0C）

位	功能	描述	复位值
7:0	红	红色元件	0
15:8	绿	绿色元件	0
23:16	蓝	蓝色元件。	0
31:24	-	保留。读取值未定义，只写入 0。	-

12.7.21 光标 XY 位置寄存器（CRSR_XY，RW — 0x2008 8C10）

CRSR_XY 寄存器定义了光标的左上边沿与光标覆盖区的左上边沿之间的距离。详见光标剪裁一节。

如果 CRSR_CFG 寄存器中的 FrameSync 位为 0，则光标位置会立即改变，即使光标正在被扫描。如果 FrameSync 为 1，则只有在下一个垂直帧消隐周期内，光标位置才会改变。

CRSR_XY 寄存器的内容描述见[表 243](#)。

表243. 光标 XY 位置寄存器（CRSR_XY，RW — 0x2008 8C10）

位	功能	描述	复位值
9:0	CrsrX	像素中所测量到的光标源的 X 纵坐标。 当该位为 0 时，光标的左边缘在显示器的左侧。	0
15:10	-	保留。读取值未定义，只写入 0。	-
25:16	CrsrY	像素中所测量到的光标源的 Y 纵坐标。 当该位为 0 时，光标的上边缘在显示器的顶部。	0
31:26	-	保留。读取值未定义，只写入 0。	-

12.7.22 光标剪裁位置寄存器（CRSR_CLIP，RW — 0x2008 8C14）

CRSR_CLIP 寄存器定义了光标图像的左上边沿与光标图像中的第一个显示像素之间的距离。

光标剪裁寄存器使用的同步规则不同于光标坐标。如果 CRSR_CFG 中的 FrameSync 位为 0，则光标剪裁点会立即改变，即使光标正在被扫描。

如果 CRSR_CFG 中的 FrameSync 位为 1，则只有在垂直帧消隐周期内，显示的光标图像才会改变，前提是剪裁寄存器设置后，光标位置已经更新。在编程设定时，剪裁寄存器必须在位置寄存器(ClcdCrsrXY)以前被写入，以确保给定帧的剪裁与对应位置信息之间的一致性。

CRSR_CLIP 寄存器的内容描述见[表 244](#)。

表244. 光标剪裁位置寄存器（CRSR_CLIP, RW — 0x2008 8C14）

位	功能	描述	复位值
5:0	CrsrClipX	X 方向的光标剪裁位置。 从光标图像的左边缘到光标中第一个显示出的像素之间的距离。当该位为 0 时，显示光标线中的第一个像素。	0
7:6	-	保留。读取值未定义，只写入 0。	-
13:8	CrsrClipY	Y 方向的光标剪裁位置。从光标图像的顶部到光标中第一个显示出的像素之间的距离。当该位为 0 时，第一个显示出的像素从光标图像的顶部线开始。	0
31:14	-	保留。读取值未定义，只写入 0。	-

12.7.23 光标中断屏蔽寄存器（CRSR_INTMSK, RW — 0x2008 8C20）

CRSR_INTMSK 寄存器用于使能或禁能光标对处理器的中断。

CRSR_INTMSK 寄存器的内容描述见[表 245](#)。

表245. 光标中断屏蔽寄存器（CRSR_INTMSK, RW — 0x2008 8C20）

位	功能	描述	复位值
0	CrsrIM	光标中断屏蔽。 当该位清零时，光标不会中断处理器。 当该位置位时，光标在从光标图像上读取完最后一个字之后马上中断处理器。	0
31:1	-	保留。读取值未定义，只写入 0。	-

12.7.24 光标中断清零寄存器（CRSR_INTCLR, RW — 0x2008 8C24）

CRSR_INTCLR 寄存器被软件用于清零光标中断状态，以及给处理器的光标中断信号。

CRSR_INTCLR 寄存器的内容描述见[表 246](#)。

表246. 光标中断清零寄存器（CRSR_INTCLR, RW — 0x2008 8C24）

位	功能	描述	复位值
0	CrsrIC	光标中断清除。 向该位写入 0 无效。 向该位写入 1 导致光标中断状态被清除。	0
31:1	—	保留。读取值未定义，只写入 0。	-

12.7.25 光标原始中断状态寄存器（CRSR_INTRAW, RW — 0x2008 8C28）

CRSR_INTRAW 寄存器设定为指示一个光标的中断状态。当通过 CRSR_INTMSK 寄存器中的 CrsrIM 位使能时，它为系统中断控制器提供中断。

CRSR_INTRAW 寄存器的内容描述见[表 247](#)。

表247. 光标原始中断状态寄存器（CRSR_INTRAW, RW — 0x2008 8C28）

位	功能	描述	复位值
0	CrsrRIS	光标原始中断状态。 光标中断状态从现有帧的光标图像上读取最后一份数据之后置位。 该位通过向 CRSR_INTCLR 寄存器中的 CrsrIC 位写入信息来清除。	0
31:1	—	保留。读取值未定义，只写入 0。	—

12.7.26 光标屏蔽中断状态寄存器（CRSR_INTSTAT, RW — 0x2008 8C2C）

CRSR_INTSTAT 寄存器用于指示一个光标的中断，前提是该中断未在 CRSR_INTMSK 寄存器中被屏蔽。

CRSR_INTSTAT 寄存器的内容描述见[表 248](#)。

表248. 光标屏蔽中断状态寄存器（CRSR_INTSTAT, RW — 0x2008 8C2C）

位	功能	描述	复位值
0	CrsrMIS	光标屏蔽中断状态。 如果 CRSR_INTMSK 寄存器中的相应位被置位，光标中断状态从现有帧的光标图像上读取最后一份数据之后置位。 如果 CRSR_INTMSK 寄存器被清除，则该位维持清零状态。该位通过向 CRSR_INTCLR 寄存器写入信息来清除。	0
31:1	-	保留。读取值未定义，只写入 0。	-

12.8 LCD 时序图

图35.STN 显示器的水平时序

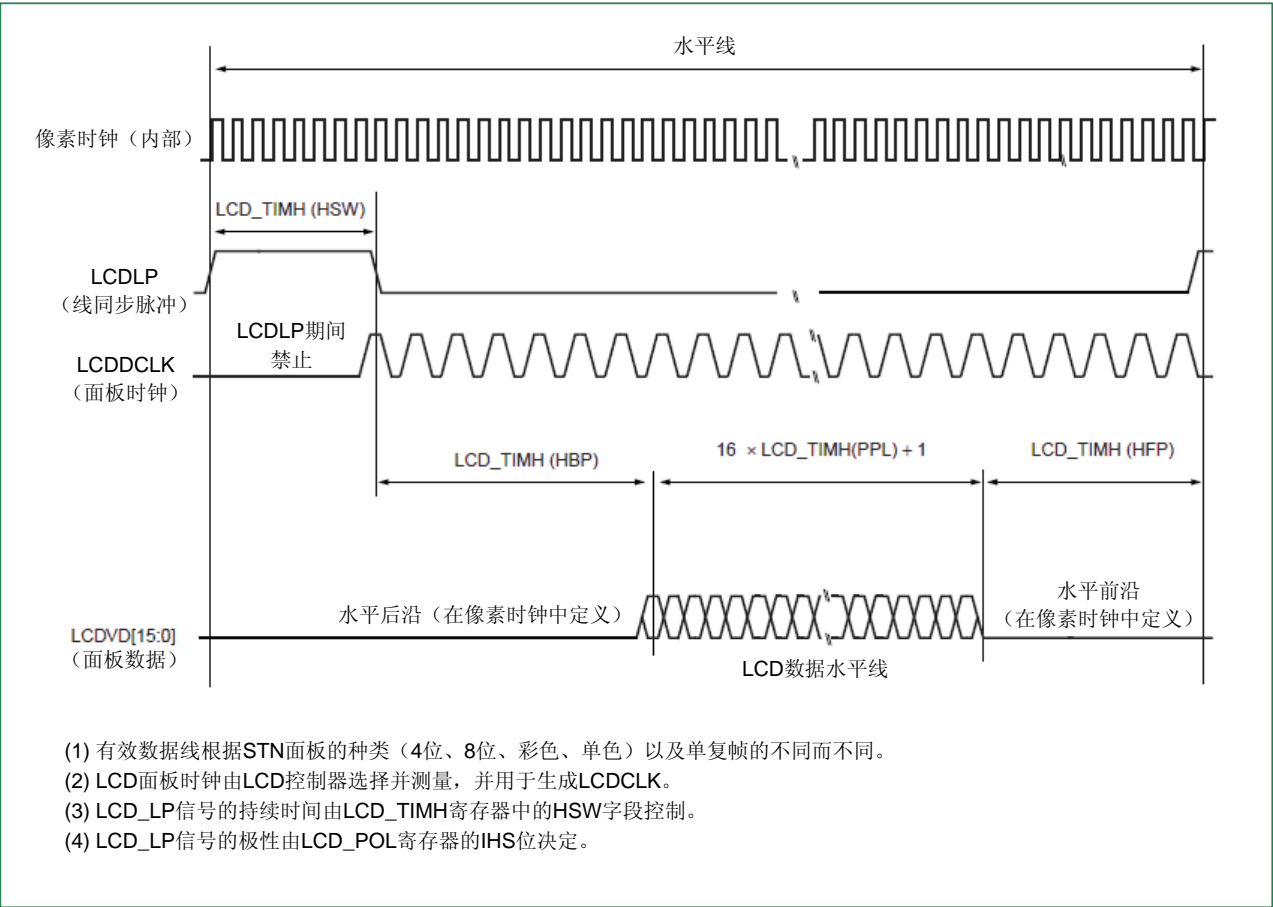


图36. STN 显示器的垂直时序

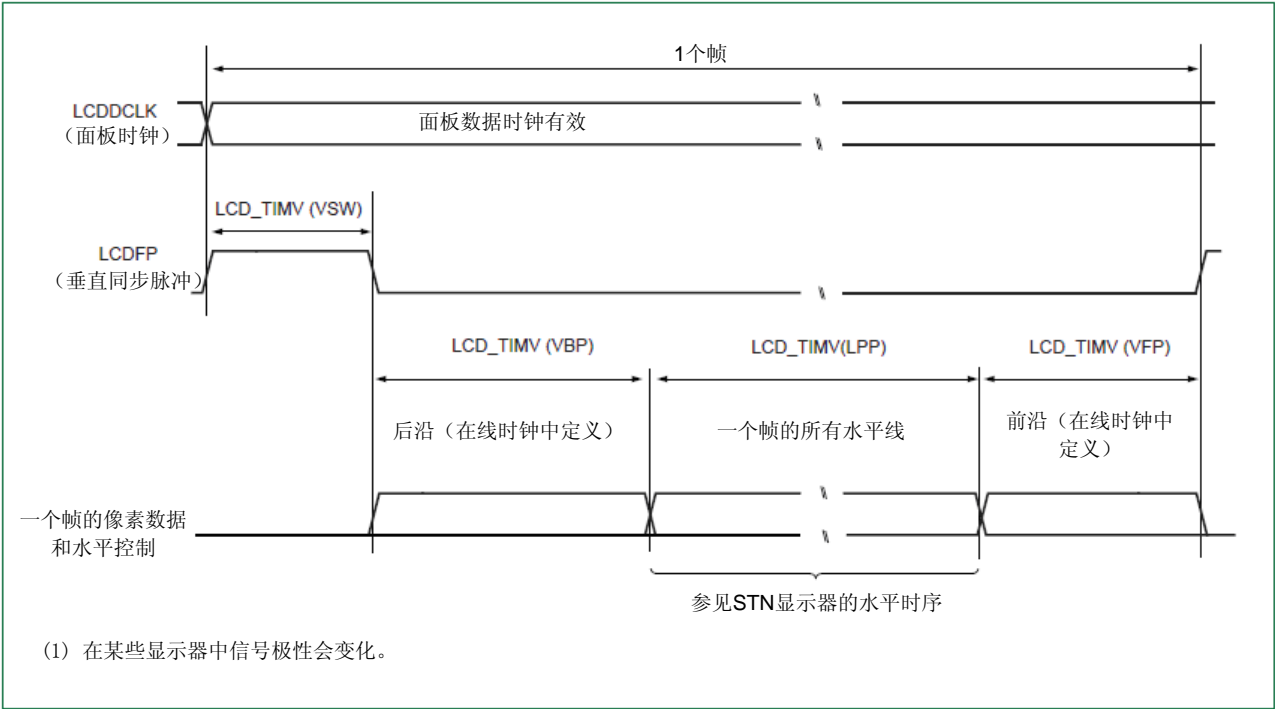


图37. TFT 显示器的水平时序

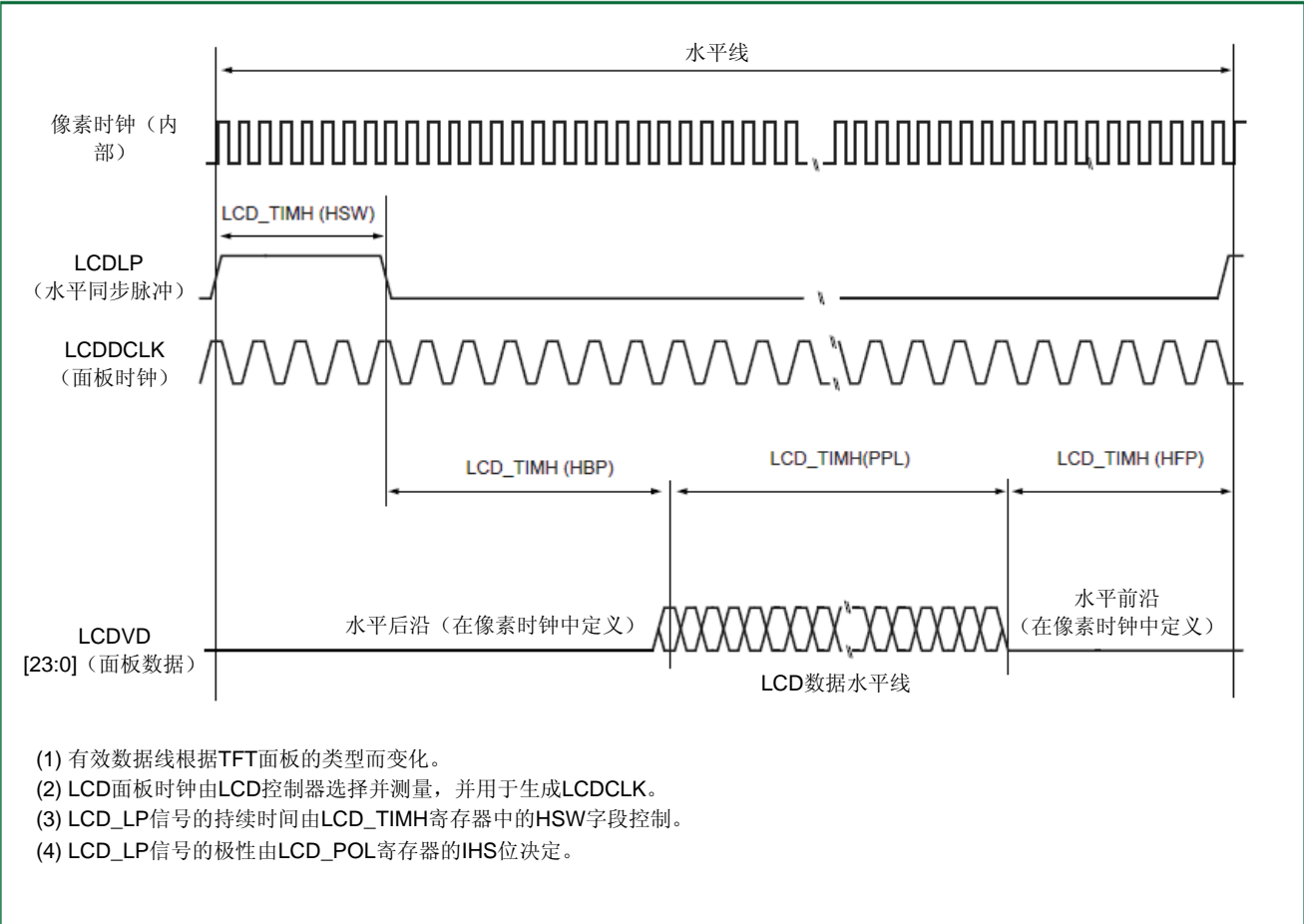
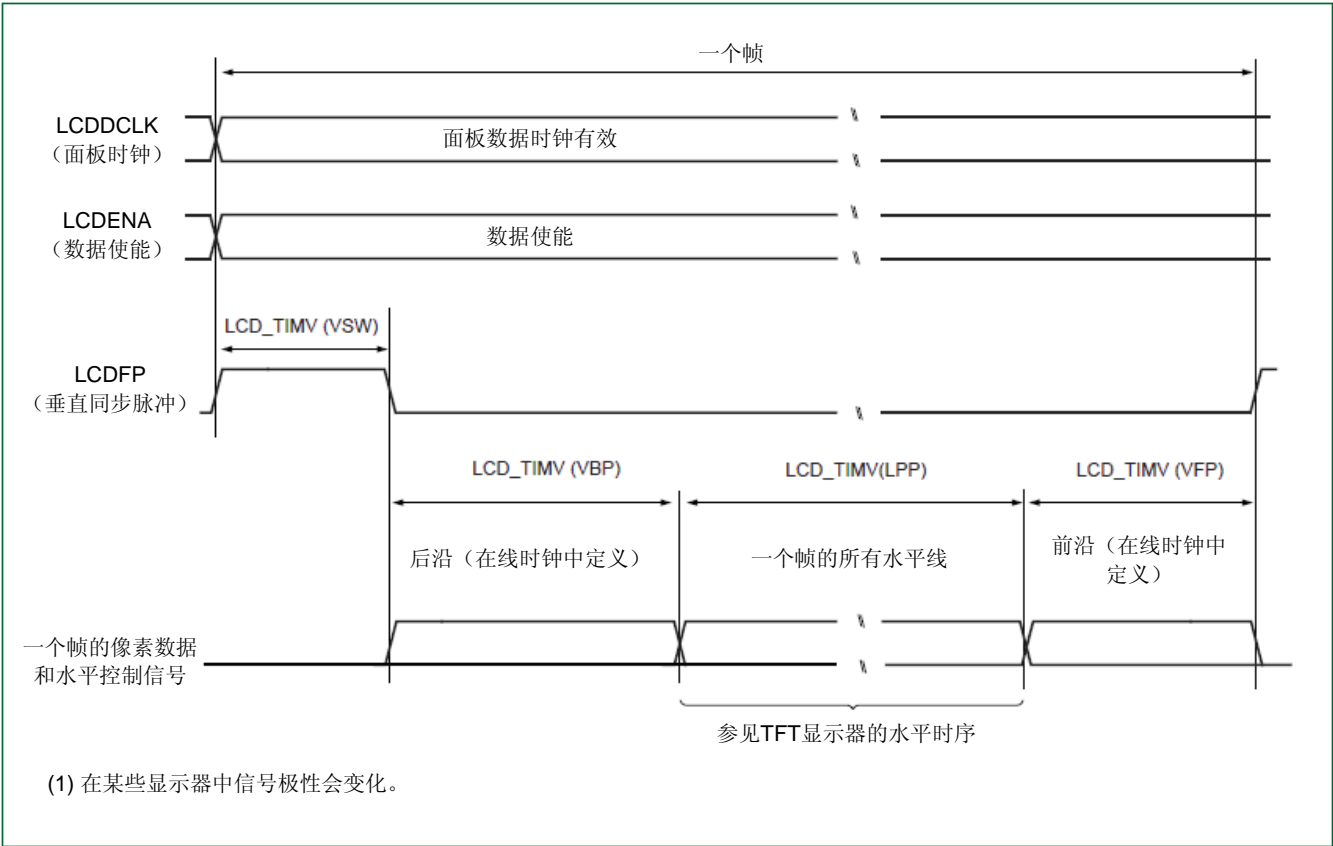


图38. TFT 显示器的垂直时序



12.9 LCD 面板的信号使用

表249. STN 单面板模式下 LCD 面板连接

外部管脚	4 位单 STN 单面板		8 位单 STN 单面板		彩色 STN 单面板	
	使用的管脚	LCD 功能	使用的管脚	LCD 功能	使用的管脚	LCD 功能
LCD_VD[8] - LCD_VD[23]	-	-	-	-	-	-
LCD_VD[7]	-	-	P4[29]	UD[7]	P4[29]	UD[7]
LCD_VD[6]	-	-	P4[28]	UD[6]	P4[28]	UD[6]
LCD_VD[5]	-	-	P2[13]	UD[5]	P2[13]	UD[5]
LCD_VD[4]	-	-	P2[12]	UD[4]	P2[12]	UD[4]
LCD_VD[3]	P2[9]	UD[3]	P2[9]	UD[3]	P2[9]	UD[3]
LCD_VD[2]	P2[8]	UD[2]	P2[8]	UD[2]	P2[8]	UD[2]
LCD_VD[1]	P2[7]	UD[1]	P2[7]	UD[1]	P2[7]	UD[1]
LCD_VD[0]	P2[6]	UD[0]	P2[6]	UD[0]	P2[6]	UD[0]
LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP
LCD_ENAB_M	P2[4]	LCD_ENAB_M	P2[4]	LCD_ENAB_M	P2[4]	LCD_ENAB_M
LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP
LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK
LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE
LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD_PWR
LCD_CLKIN	P2[11]	LCD_CLKIN	P2[11]	LCD_CLKIN	P2[0]	LCD_PWR

表250. TN 双面板模式下 LCD 面板连接

外部管脚	4 位单 STN 双面板		8 位单 STN 双面板		彩色 STN 双面板	
	使用的管脚	LCD 功能	使用的管脚	LCD 功能	使用的管脚	LCD 功能
LCD_VD[16] - LCD_VD[23]	-	-	-	-	-	-
LCD_VD[15]	-	-	P1[29]	LD[7]	P1[29]	LD[7]
LCD_VD[14]	-	-	P1[28]	LD[6]	P1[28]	LD[6]
LCD_VD[13]	-	-	P1[27]	LD[5]	P1[27]	LD[5]
LCD_VD[12]	-	-	P1[26]	LD[4]	P1[26]	LD[4]
LCD_VD[11]	P4[29]	LD[3]	P1[25]	LD[3]	P1[25]	LD[3]
LCD_VD[10]	P4[28]	LD[2]	P1[24]	LD[2]	P1[24]	LD[2]
LCD_VD[9]	P2[13]	LD[1]	P1[23]	LD[1]	P1[23]	LD[1]
LCD_VD[8]	P2[12]	LD[0]	P1[22]	LD[0]	P1[22]	LD[0]
LCD_VD[7]	-	-	P1[21]	UD[7]	P1[21]	UD[7]
LCD_VD[6]	-	-	P1[20]	UD[6]	P1[20]	UD[6]
LCD_VD[5]	-	-	P2[13]	UD[5]	P2[13]	UD[5]
LCD_VD[4]	-	-	P2[12]	UD[4]	P2[12]	UD[4]
LCD_VD[3]	P2[9]	UD[3]	P2[9]	UD[3]	P2[9]	UD[3]
LCD_VD[2]	P2[8]	UD[2]	P2[8]	UD[2]	P2[8]	UD[2]
LCD_VD[1]	P2[7]	UD[1]	P2[7]	UD[1]	P2[7]	UD[1]
LCD_VD[0]	P2[6]	UD[0]	P2[6]	UD[0]	P2[6]	UD[0]
LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP
LCD_ENAB_M	P2[4]	LCD_ENAB_M	P2[4]	LCD_ENAB_M	P2[4]	LCD_ENAB_M
LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP
LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK
LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE
LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD-PWR
LCD_CLKIN	P2[11]	LCD_CLKIN	P2[11]	LCD_CLKIN	P2[11]	LCD_CLKIN

表251. TFT 面板所使用的 LCD 面板连接

外部管脚	TFT12 位 (4:4:4 模式)		TFT16 位 (5:6:5 模式)		TFT16 位 (1:5:5:5 模式)		TFT24 位	
	使用的管脚	LCD 功能	使用的管脚	LCD 功能	使用的管脚	LCD 功能	使用的管脚	LCD 功能
LCD_VD[23]	P1[29]	蓝 3	P1[29]	蓝 4	P1[29]	蓝 4	P1[29]	蓝 7
LCD_VD[22]	P1[28]	蓝 2	P1[28]	蓝 3	P1[28]	蓝 3	P1[28]	蓝 6
LCD_VD[21]	P1[27]	蓝 1	P1[27]	蓝 2	P1[27]	蓝 2	P1[27]	蓝 5
LCD_VD[20]	P1[26]	蓝 0	P1[26]	蓝 1	P1[26]	蓝 1	P1[26]	蓝 4
LCD_VD[19]	-	-	P2[13]	蓝 0	P2[13]	蓝 0	P2[13]	蓝 3
LCD_VD[18]	-	-	-	-	P2[12]	亮度	P2[12]	蓝 2
LCD_VD[17]	-	-	-	-	-	-	P0[9]	蓝 1
LCD_VD[16]	-	-	-	-	-	-	P0[8]	蓝 0
LCD_VD[15]	P1[25]	绿 3	P1[25]	绿 5	P1[25]	绿 4	P1[25]	绿 7
LCD_VD[14]	P1[24]	绿 2	P1[24]	绿 4	P1[24]	绿 3	P1[24]	绿 6
LCD_VD[13]	P1[23]	绿 1	P1[23]	绿 3	P1[23]	绿 2	P1[23]	绿 5
LCD_VD[12]	P1[22]	绿 0	P1[22]	绿 2	P1[22]	绿 1	P1[22]	绿 4
LCD_VD[11]	-	-	P1[21]	绿 1	P1[21]	绿 0	P1[21]	绿 3
LCD_VD[10]	-	-	P1[20]	绿 0	P1[20]	亮度	P1[20]	绿 2
LCD_VD[9]	-	-	-	-	-	-	P0[7]	绿 1
LCD_VD[8]	-	-	-	-	-	-	P0[6]	绿 0
LCD_VD[7]	P2[9]	红 3	P2[9]	红 4	P2[9]	红 4	P2[9]	红 7
LCD_VD[6]	P2[8]	红 2	P2[8]	红 3	P2[8]	红 3	P2[8]	红 6
LCD_VD[5]	P2[7]	红 1	P2[7]	红 2	P2[7]	红 2	P2[7]	红 5
LCD_VD[4]	P2[6]	红 0	P2[6]	红 1	P2[6]	红 1	P2[6]	红 4
LCD_VD[3]	-	-	P2[12]	红 0	P4[29]	红 0	P4[29]	红 3
LCD_VD[2]	-	-	-	-	P4[28]	亮度	P4[28]	红 2
LCD_VD[1]	-	-	-	-	-	-	P0[5]	红 1
LCD_VD[0]	-	-	-	-	-	-	P0[4]	红 0
LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP	P2[5]	LCD_LP
LCD_ENAB_M	P2[4]	LCD_ENAB_M	P2[4]	LCD_ENAB_M	P2[4]	LCD_ENAB_M	P2[4]	LCD_ENAB_M
LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP	P2[3]	LCD_FP
LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK	P2[2]	LCD_DCLK
LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE	P2[1]	LCD_LE
LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD_PWR	P2[0]	LCD_PWR
LCD_CLKIN	P2[11]	LCD_CLKIN	P2[11]	LCD_CLKIN	P2[11]	LCD_CLKIN	P2[11]	LCD_CLKIN

13.1 如何阅读本章

本章描述了某些 LPC178x/177x 器件使用的 USB 设备控制器（详见 [1.4](#) 节）。在某些 LPC178x/177x 系列器件上，USB 控制器亦可以配置为主机运行或 OTG 运行。

13.2 基本配置

USB 控制器的配置使用了下列寄存器：

1. 功率：PCONP 寄存器（[表 37](#)）中置位 PCUSB。
注：复位后，USB 模块被禁用（PCUSB = 0）。
2. 时钟：USB 模块可以与主 PLL（PLL0）或副 PLL（PLL1）一起使用，从而得到 USB 时钟。见 [4.5.2](#) 节。
3. 管脚：在相应的 IOCON 寄存器中，选择所需要 的 USB 管脚及其模式（见 [8.4.1](#) 节）。
4. 唤醒：USB 总线端口上的活动可以将微控制器从掉电模式中唤醒，见 [4.7.9](#) 节。
5. 中断：通过适当的中断设置使能寄存器，可在 NVIC 中使能中断。
6. 初始化：见 [13.13](#) 节。

13.3 简介

通用串行总线（USB）为 4 线总线，它支持一个主机与一个或多个（最多 127 个）外设之间的通信。主控制器通过一种基于令牌的协议，为连接的设备分配 USB 带宽。总线支持设备的热插拔以及动态配置。所有事务均由主控制器发起。

主机将事务安排在 1ms 的帧中。每个帧都包含一个帧起始（SOF）标记，以及将数据从设备端点传入或传出的事务。每个设备最多可以有 16 个逻辑端点或 32 个物理端点。针对端点定义了 4 种传输类型。控制传输用于设备的配置。中断传输用于间发的周期数据传输。批量传输用于对传输速率没有严格要求的情况。同步传输可以保证传输时间，但没有纠错功能。

有关通用串行总线的更多信息，请见 USB 应用者论坛（USB Implementers Forum）网站。

LPC178x/177x 上的 USB 设备控制器能够与 USB 主控制器实现全速（12Mb/s）的数据交换。

表252. 本章用到的与 USB 相关的首字母缩写词、简写以及定义

缩写词	定义
AHB	先进高性能总线
ATLE	自动传输长度提取
ATX	模拟收发器
DD	DMA 描述符
DDP	DMA 描述指针
DMA	直接存储器访问
EOP	数据包结尾
EP	端点
EP_RAM	端点 RAM
FS	全速
LED	发光二极管
LS	低速
MPS	最大数据包容量
NAK	否定应答
PLL	锁相环
RAM	随机访问存储器
SOF	帧开始
SIE	串行接口引擎
SRAM	同步 RAM
UDCA	USB 设备通信区域
USB	通用串行总线

13.4 特性

- 完全符合 USB2.0 全速规范。
- 支持 32 个物理（16 个逻辑）端点。
- 支持控制、批量、中断与同步端点。
- 运行时，可调整使用的端点。
- 运行时，可通过软件选择端点的最大数据包长度（直至 USB 规范规定的最大长度）。
- 支持 SoftConnect 与 GoodLink 特性。
- 所有非控制端点支持 DMA 传输。
- 允许在 CPU 控制与 DMA 模式之间的动态切换。
- 实现了批量端点与同步端点的双缓冲。

13.5 固定端点配置

[表 253](#) 显示了所有支持的端点配置。端点在运行时使用端点实现寄存器使用端点并进行配置，见 [13.10.5](#) 节。

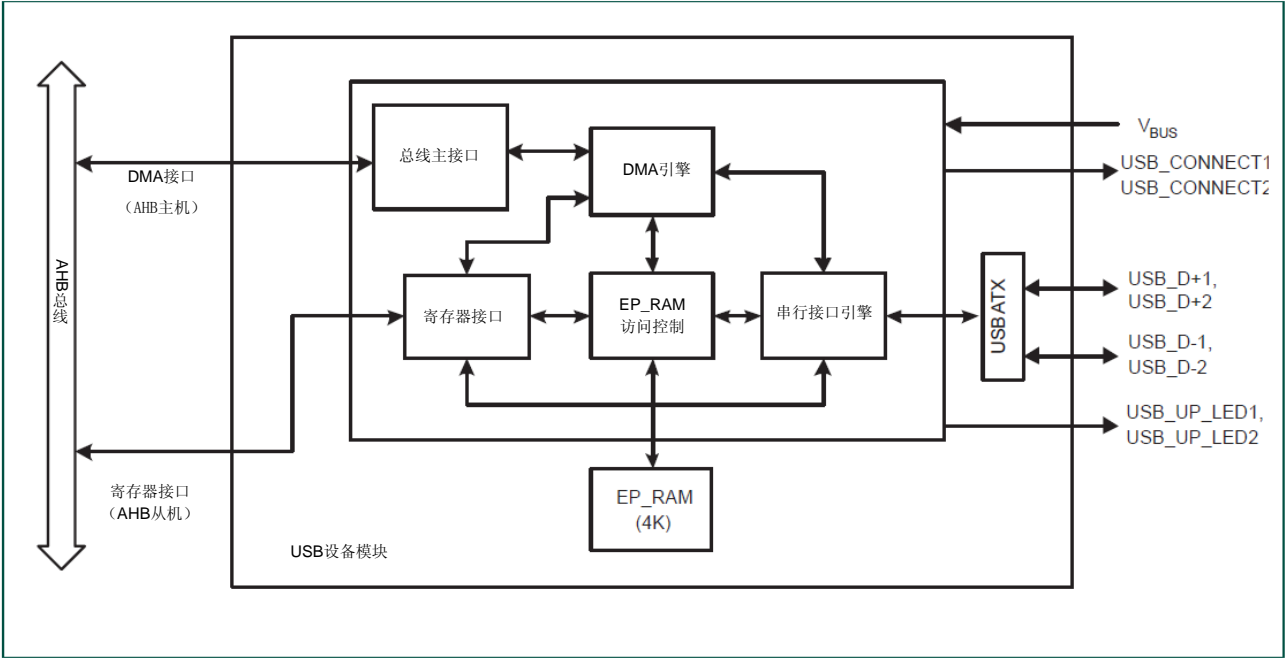
表253. 固定的端点配置

逻辑端点	物理端点	端点类型	方向	数据包长度（字节）	双缓冲
0	0	控制	出	8/16/32/64	无
0	1	控制	入	8/16/ 32/64	无
1	2	中断	出	1~64	无
1	3	中断	入	1~64	无
2	4	批量	出	8/16/32/64	有
2	5	批量	入	8/16/32/64	有
3	6	同步	出	1~1023	有
3	7	同步	入	1~1023	有
4	8	中断	出	1~64	无
4	9	中断	入	1~64	无
5	10	批量	出	8/16/32/64	有
5	11	批量	入	8/16/32/64	有
6	12	同步	出	1~1023	有
6	13	同步	入	1~1023	有
7	14	中断	出	1~64	无
7	15	中断	入	1~64	无
8	16	批量	出	8/16/32/64	有
8	17	批量	入	8/16/32/64	有
9	18	同步	出	1~1023	有
9	19	同步	入	1~1023	有
10	20	中断	出	1~64	无
10	21	中断	入	1~64	无
11	22	批量	出	8/16/32/64	有
11	23	批量	入	8/16/32/64	有
12	24	同步	出	1~1023	有
12	25	同步	入	1~1023	有
13	26	中断	出	1~64	无
13	27	中断	入	1~64	无
14	28	批量	出	8/16/32/64	有
14	29	批量	入	8/16/32/64	有
15	30	批量	出	8/16/32/64	有
15	31	批量	入	8/16/32/64	有

13.6 功能描述

USB 设备控制器的架构见下 图 39。

图39. USB 设备控制器方框图



13.6.1 模拟收发器

USB 设备控制器有一个内置的模拟收发器 (ATX)。USB ATX 用于发送/接收 USB 总线的双向 D+ 与 D- 信号。

13.6.2 串行接口引擎 (SIE)

SIE 实现了全部 USB 协议层。从速度角度考虑，它采用全硬接线式 (hardwired)，无需固件干预。它用于处理 EP_RAM 中的端点缓冲区与 USB 总线之间的数据传输。这个模块的功能包括：同步模式识别、并行/串行转换、位填充 (bit stuffing) /解除填充、CRC 校验/产生、PID 验证/生成、地址识别，以及握手信号的评估/生成。

13.6.3 端点 RAM (EP_RAM)

每个端点缓冲区都以基于 FIFO 的 SRAM 形式实现。这个专用的 SRAM 叫做 EP_RAM。每个已实现的端点在 EP_RAM 中都有一个保留空间。总的 EP_RAM 空间需求取决于已实现的端点数、端点最大数据包长度，以及端点是否支持双缓冲。

13.6.4 EP_RAM 访问控制

EP_RAM 访问控制逻辑用于处理进出 EP_RAM 的数据传输，它的访问来源有三种：CPU（通过寄存器接口）、SIE 和 DMA 引擎。

13.6.5 DMA 引擎与总线主接口

当某个端点的 DMA 引擎使能后，可在 AHB 总线上的 RAM 与 EP_RAM 中的端点缓冲区之间传输数据。所有端点之间共享一个 DMA 通道。在传输数据时，DMA 引擎通过总线主机接口作为 AHB 总线上的主机运行。

13.6.6 寄存器接口

寄存器接口使 CPU 能够控制 USB 设备控制器的操作。它还提供了一种向控制器写入传输数据，以及从控制器读出接收数据的方式。

13.6.7 SoftConnect

与 USB 的连接方式是通过一只 1.5kΩ 上拉电阻，将 D+（对全速设备）拉至高电平。SoftConnect 功能可用于让软件完成其初始化序列后，再确定与 USB 建立连接。也可以在不重新拔插电缆情况下，完成 USB 总线连接的重新初始化。

使用 SoftConnect 功能时，CONNECT 信号应控制一个外部开关，开关连接着 D+ 与 +3.3V 之间的 1.5kΩ 电阻。这样，软件可以使用 SIE 的“设置设备状态”命令写入 CON 位，从而控制 CONNECT 信号。

13.6.8 GoodLink

GoodLink 技术可用于指示 USB 连接是否良好。当设备已成功地被清点和配置后，LED 指示灯常亮。在挂起期间，LED 灯熄灭。

这一特性对 USB 设备的状态提供了一种用户友好的指示。它是一种有用的区域诊断工具，能隔离出故障设备。

使用 GoodLink 特性时，UP_LED 信号用于控制 LED。使用 SIE 的“配置设备”命令控制 UP_LED 信号。

13.7 操作概述

USB 总线事务在设备端点与主机之间传输数据。事务的方向由主机确定。OUT 事务是从主机向设备传输数据。IN 事务是从设备向主机传输数据。所有事务均由主控制器启动。

对于 OUT 事务，USB ATX 接收 USB 总线上的双向 D+与 D-信号。串行接口引擎（SIE）接收来自 ATX 的串行数据，并将其转换为一个并行数据流。并行数据被写入 EP_RAM 中相应的端点缓冲区。

对于 IN 事务，SIE 从 EP_RAM 中的端点缓冲区读出并行数据，将其转换为串行数据，然后使用 USB ATX 将其传输到 USB 总线上。

一旦数据被接收或发送完毕，端点缓冲区就可以做读写。其实现方式取决于端点的类型与工作模式。每个端点都有两种工作模式：从模式（CPU 控制），以及 DMA 模式。

在从模式下，CPU 使用寄存器接口，在 RAM 与端点缓冲区之间传输数据。有关本模式的详情见 [13.14](#) 节。

在 DMA 模式下，DMA 在 RAM 与端点缓冲区之间传输数据。有关本模式的详情见 [13.15](#) 节。

13.8 管脚描述

表254. USB 外部接口

名称	方向	描述
V _{BUS}	入	V _{BUS} 状态输入。在没有通过对应的 IOCON 寄存器将功能使能时，它在内部驱动为高电平。
USB_CONNECT1, USB_CONNECT2	出	SoftConnect 控制信号
USB_UP_LED1, USB_UP_LED2	出	GoodLink LED 控制信号
USB_D+1, USB_D+2	入/出	正向差分数据
USB_D-1, USB_D-2	入/出	反向差分数据

13.9 时钟与功率管理

本节描述 USB 设备控制器的时钟与功率管理特性。

13.9.1 功率要求

USB 协议坚持要求对设备进行功率管理。如果设备是从总线获取功率（总线供电设备），这点就使功率管理变得尤为重要。总线供电设备应满足以下约束条件：

- 1. 未配置状态的设备从总线获取的最大电流为 100mA。
- 2. 已配置设备所获电流应不大于配置描述符中 Max Power 字段的规定值。最大电流为 500mA。
- 3. 处于挂起状态的设备可以获取最大电流为 500μA。

13.9.2 时钟

USB 设备控制器的时钟见[表 255](#)。

表255. USB 设备控制器时钟源

时钟源	描述
AHB 主机时钟	AHB 主机总线接口和 DMA 的时钟
AHB 从机时钟	AHB 从机接口的时钟
usbclk	来自 USB 特定 PLL 或主 PLL 的 48MHz 时钟，用于恢复来自 USB 总线的 12MHz 时钟。

13.9.3 功率管的支持

为节省功率，USB 设备控制器会在不使用时自动禁用 AHB 主时钟与 usbclk。

当 USB 设备控制器进入挂起状态时（总线在 3ms 内保持空闲），则设备控制器的 usbclk 输入被自动禁能，以帮助节省功率。不过，如果软件要访问设备控制器的寄存器时，usbclk 必须有效。为了在挂起状态时也能允许访问设备控制器的寄存器，提供了 USBClkCtrl 和 USBClkSt 两个寄存器。

当软件希望访问设备控制器的寄存器时，它应确认 usbclk 已使能，方法是：首先置位 USBClkCtrl 寄存器中的 DEV_CLK_EN 位，然后轮询 USBClkSt 中相应的 DEV_CLK_ON 位，直至置位为止。一旦置位以后，usbclk 会一直保持使能状态，直到 DEV_CLK_EN 被软件清零。

当进行 DMA 传输时，设备控制器会自动开启 AHB 主时钟。主时钟一旦有效，会维持有效状态至少 2ms（2 帧），这有助于确保 DMA 吞吐量不受 AHB 主时钟关闭的影响。在上一次 DMA 访问后 2ms，AHB 主时钟自动禁能以节省功率。如有必要，软件也能够使用 USBClkCtrl 寄存器，强制此时钟保持在使能状态。

注意，只要 PCONP 寄存器的 PCUSB 位置位，则 AHB 从时钟总是使能的。当设备控制器

未在使用中时，清零 PCUSB 就可以禁能所有该设备控制器的时钟。

使用 USB_NEED_CLK 信号可以使芯片进入掉电模式或从掉电模式下被唤醒变得容易。如果 USBClkSt 寄存器中的任何位有效，则 USB_NEED_CLK 也有效。

当 DEV_CLK_EN 与 AHB_CLK_EN 被清零而进入挂起状态，DEV_CLK_ON 与 AHB_CLK_ON 在相应的时钟被关闭后都会被清零。当两位均为 0 时，USB_NEED_CLK 将为低电平，表示可以通过写入 PCON 寄存器将芯片置于掉电模式。从 USBIntSt 寄存器可以读出 USB_NEED_CLK 的状态。

挂起状态下的任何总线活动都会使 USB_NEED_CLK 信号变为有效。当芯片处于掉电模式下，USB 中断被使能时，USB_NEED_CLK 有效会使芯片从掉电模式中唤醒。

13.9.4 远程唤醒

USB 设备控制器支持软件启动的远程唤醒。远程唤醒的含意是恢复 USB 总线上由设备启动的信号。做法是清零 SIE 设置设备状态寄存器中的 SUS 位。在写入寄存器以前，必须用 USBClkCtrl 寄存器使能设备控制器的所有时钟。

13.10 寄存器描述

表 256 显示了可由 CPU 直接访问的 USB 设备控制器寄存器。串行接口引擎（SIE）还有一些其它寄存器，可通过 SIE 命令寄存器间接地访问。详见 13.12 节。

表256. USB 设备寄存器映射

名称	描述	访问	复位值 ^[1]	地址	表
时钟选择寄存器					
USBPortSel ^[2]	USB 端口选择	R/W	0	0x2008 C110	表 257
时钟控制寄存器					
USBClkCtrl	USB 时钟控制	R/W	0	0x2008 CFF4	表 258
USBClkSt	USB 时钟状态	RO	0	0x2008 CFF8	表 259
设备中断寄存器					
USBIntSt	USB 中断状态	R/W	0x8000 0000	0x400F C1C0	表 260
USBDevIntSt	USB 设备中断状态	RO	0x10	0x2008 C200	表 261
USBDevIntEn	USB 设备中断使能	R/W	0	0x2008 C204	表 263
USBDevIntClr	USB 设备中断清零	WO	0	0x2008 C208	表 265
USBDevIntSet	USB 设备中断设置	WO	0	0x2008 C20C	表 267
USBDevIntPri	USB 设备中断优先级	WO	0	0x2008 C22C	表 269
端点中断寄存器					
USBEPIntSt	USB 端点中断状态	RO	0	0x2008 C230	表 270
USBEPIntEn	USB 端点中断使能	R/W	0	0x2008 C234	表 272
USBEPIntClr	USB 端点中断清零	WO	0	0x2008 C238	表 274
USBEPIntSet	USB 端点中断设置	WO	0	0x2008 C23C	表 276
USBEPIntPri	USB 端点中断优先级	WO ^[3]	0	0x2008 C240	表 278
端点使用寄存器					
USBReEp	USB 使用端点	R/W	0x3	0x2008 C244	表 280
USBEPIn	USB 端点索引	WO ^[3]	0	0x2008 C248	表 282
USBMaxPSize	USB 最大包长度	R/W	0x8	0x2008 C24C	表 283
USB 传输寄存器					
USBRxData	USB 接收数据	RO	0	0x2008 C218	表 284
USBRxPLen	USB 接收包长度	RO	0	0x2008 C220	表 285
USBTxData	USB 发送数据	WO ^[3]	0	0x2008 C21C	表 286
USBTxPLen	USB 发送包长度	WO ^[3]	0	0x2008 C224	表 287
USBCtrl	USB 控制	R/W	0	0x2008 C228	表 288
SIE 命令寄存器					
USBCmdCode	USB 命令代码	WO ^[3]	0	0x2008 C210	表 289
USBCmdData	USB 命令数据	RO	0	0x2008 C214	表 290
DMA 寄存器					
USBDMASt	USB DMA 请求状态	RO	0	0x2008 C250	表 291
USBDMAClr	USB DMA 请求清零	WO ^[3]	0	0x2008 C254	表 293
USBDMASet	USB DMA 请求设置	WO ^[3]	0	0x2008 C258	表 294
USBUDCAH	USB UDCA Head	R/W	0	0x2008 C280	表 295
USBEPDMASt	USB 端点 DMA 状态	RO	0	0x2008 C284	表 296
USBEPDMAEn	USB 端点 DMA 使能	WO ^[3]	0	0x2008 C288	表 297
USBEPDMADis	USB 端点 DMA 禁能	WO ^[3]	0	0x2008 C28C	表 298

名称	描述	访问	复位值 ^[1]	地址	表
USBDMAIntSt	USB DMA 中断状态	RO	0	0x2008 C290	表 299
USBDMAIntEn	USB DMA 中断使能	R/W	0	0x2008 C294	表 300
USBEoTIntSt	USB 传输结束中断状态	RO	0	0x2008 C2A0	表 301
USBEoTIntClr	USB 传输结束中断清除	WO ^[3]	0	0x2008 C2A4	表 302
USBEoTIntSet	USB 传输结束中断设置	WO ^[3]	0	0x2008 C2A8	表 303
USBNDDRIntSt	USB 新 DD 请求中断状态	RO	0	0x2008 C2AC	表 304
USBNDDRIntClr	USB 新 DD 请求中断清除	WO ^[3]	0	0x2008 C2B0	表 305
USBNDDRIntSet	USB 新 DD 请求中断设置	WO ^[3]	0	0x2008 C2B4	表 306
USBSysErrIntSt	USB 系统错误中断状态	RO	0	0x2008 C2B8	表 307
USBSysErrIntClr	USB 系统错误中断清除	WO ^[3]	0	0x2008 C2BC	表 308
USBSysErrIntSet	USB 系统错误中断设置	WO ^[3]	0	0x2008 C2C0	表 309

- [1] 复位值仅反映已使用位中保存的数据，它不包括保留位的内容。
- [2] USBPortSel 寄存器与 OTGStCtrl 寄存器相同（见 [15.8.6](#) 节）。在设备操作中，该寄存器中只有位 0 和 1 被用来控制 USB 管脚与端口 1 或端口 2 的连接。
- [3] 读取 WO 寄存器将返回一个无效值。

13.10.1 端口选择寄存器

13.10.1.1 USB 端口选择寄存器（USBPortSel—0x2008 C110）

此寄存器用于选择 USB 设备信号要连接到的 USB 端口管脚。USBPortSel 是一个读写寄存器。

表257. USB 端口选择寄存器（USBPortSel—0x2008 C110）位描述

位	符号	值	描述	复位值
1:0	PORTSEL		选择设备控制器信号所映射的 USB 端口。	0
		0x0	USB 设备控制器信号映射到 U1 端口：USB_CONNECT1、USB_UP_LED1、USB_D+1、USB_D-1。	
		0x3	USB 设备控制器信号映射到 U2 端口：USB_CONNECT2、USB_UP_LED2、USB_D+2、USB_D-2。	
		其他	保留。	
31:2	-		保留。读取值未定义，只写入 0。	无

13.10.2 时钟控制寄存器

13.10.2.1 USB 时钟控制寄存器（USBClkCtrl—0x2008 CFF4）

此寄存器控制着 USB 设备控制器的时钟。无论何时软件要访问设备控制器的寄存器时，DEV_CLK_EN 与 AHB_CLK_EN 位必须置位。只有当访问 USBPortSel 寄存器时，才需要将 PORTSEL_CLK_EN 位置位。

只要相应的 USBClkCtrl 位已置位，则软件就不需要在每次寄存器访问时重复这一操作。需要注意的是，只有 PCONP 的 PCUSB 位置位时，此寄存器才正常工作；当 PCUSB 清零时，设备控制器的所有时钟均被禁能，不管此寄存器中是什么内容。USBClkCtrl 是一个读写寄存器。

表258. USBClkCtrl 寄存器（USBClkCtrl—0x2008 CFF4）位描述

位	符号	描述	复位值
0	-	保留。读取值未定义，只写入 0。	无
1	DEV_CLK_EN	设备时钟使能。使能设备控制器的 usbclk 输入。	0
2	-	保留。读取值未定义，只写入 0。	无
3	PORTSEL_CLK_EN	端口选择寄存器的时钟使能。	无
4	AHB_CLK_EN	AHB 时钟使能。	0
31:5	-	保留。读取值未定义，只写入 0。	无

13.10.2.2 USB 时钟状态寄存器（USBClkSt—0x2008 CFF8）

此寄存器用于保存时钟可用的状态。寄存器的位相“或”，形成 USB_NEED_CLK 信号。当通过 USBClkCtrl 使能一个时钟时，软件应轮询 USBClkSt 中的相应位。如果它已置位，则软件可以继续寄存器访问。只要 USBClkCtrl 位没收到打扰，软件就不需要为每次访问重复轮询操作。USBClkSt 是一个只读寄存器。

表259. USB 时钟状态寄存器（USBClkSt—0x2008 CFF8）位描述

位	符号	描述	复位值
0	-	保留。读取值未定义，只写入 0。	无
1	DEV_CLK_ON	设备时钟开启。设备控制器的 usbclk 输入有效。	0
2	-	保留。读取值未定义，只写入 0。	无
3	PORTSEL_CLK_ON	端口选择寄存器时钟开启。	无
4	AHB_CLK_ON	AHB 时钟开启。	0
31:5	-	保留。从保留位读出的值未被定义。	无

13.10.3 设备中断寄存器

13.10.3.1 USB 中断状态寄存器（USBIntSt—0x2008 C1C0）

USB 设备控制器有 3 条中断线。该寄存器能使软件通过一次读操作判断所有中断线的状态。所有 3 条中断线相“或”，连接到向量中断控制器的通道。此寄存器还包含了 USB_NEED_CLK 状态位与 EN_USB_INTS 控制位。USBIntSt 是一个读写寄存器。

表260. USB 中断状态寄存器（USBIntSt—0x2008 C1C0）位描述

位	符号	描述	复位值
0	USB_INT_REQ_LP	低优先级中断线的状态。该位为只读位。	0
1	USB_INT_REQ_HP	高优先级中断线的状态。该位为只读位。	0
2	USB_INT_REQ_DMA	DMA 中断线的状态。该位为只读位。	0
7:3	-	保留。读取值未定义，只写入 0。	无
8	USB_NEED_CLK	USB 需要时钟指示器。当检测到 USB 活动或检测到 USB 数据管脚上的状态发生改变时，该位被设置为 1，而且它表示需要一个 48 MHz 的 PLL 支持时钟。 一旦该位为 1，它就在接收到/发送最后包的 5ms 后或在挂起改变（SUS_CH）中断发生的 2ms 后复位为 0。 如果选择 USB 总线上的活动从掉电模式中唤醒器件，那么该位从 0 到 1 的变化可唤醒微控制器（详细信息请参考 4.7.9 节）。 有关 PLL 和请求掉电模式的描述也可参考 4.5.8 节和 4.7.10 节。该位为只读位。	1
30:9	-	保留。读取值未定义，只写入 0。	无
31	EN_USB_INTS	使能所有 USB 中断。当该位清零时，向量中断控制器将无法检测到 USB 中断线相或后的输出。	1

13.10.3.2 USB 设备中断状态寄存器（USBDevIntSt—0x2008 C200）

USBDevIntSt 寄存器记录着每个中断的状态。0 表示未产生中断，1 表示存在着中断。USBDevIntSt 是一个只读寄存器。

表261. USB 设备中断状态寄存器（USBDevIntSt—0x2008 C200）位分配

复位值： 0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	-	-	-	-
位	23	22	21	20	19	18	17	16
符号	-	-	-	-	-	-	-	-
位	15	14	13	12	11	10	9	8
符号	-	-	-	-	-	-	ERR_INT	EP_RLZED
位	7	6	5	4	3	2	1	0
符号	TxENDPKT	Rx ENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

表262. USB 设备中断状态寄存器（USBDevIntSt—0x2008 C200）位描述

位	符号	描述	复位值
0	FRAME	每隔 1ms 产生一次帧中断。该位用在同步包的传输中。	0
1	EP_FAST	端点的快速中断。如果端点中断在端点中断优先级寄存器（USBEPIntPri）中相应的位被置位，则该端点中断与 EP_FAST 位相关。	0
2	EP_SLOW	端点的慢速中断。如果端点中断在端点中断优先级寄存器（USBEPIntPri）中相应的位未置位，则该端点中断与 EP_SLOW 位相关。	0
3	DEV_STAT	该位在 USB 总线复位、USB 挂起改变或连接改变时置位。 请参考 13.12.6 节。	0
4	CCEMPTY	命令代码寄存器（USBCmdCode）为空（可写入新的命令）。	1
5	CDFULL	命令数据寄存器（USBCmdData）已满（现在可以读取数据）。	0
6	RxENDPKT	在端点缓冲区中的当前数据包已传送给 CPU。	0
7	TxENDPKT	传输到端点缓冲区的数据字节数与 TxPacket 长度寄存器（USBTxPLen）中编程设定的字节数相等。	0
8	EP_RLZED	端点被使用。该位在使用端点寄存器（USBReEp）或 MaxPacketSize 寄存器（USBMaxPSize）更新且相应的操作完成时置位。	0
9	ERR_INT	错误中断。USB 设备的任何总线错误中断。请参考 13.12.9 节。	0
31:10	-	保留。从保留位读出的值未被定义。	无

13.10.3.3 USB 设备中断使能寄存器（USBDevIntEn—0x2008 C204）

向此寄存器写入 1，可以使能 USBDevIntSt 中的相应位，则可以在其中某个中断线上产生中断。默认情况下，中断被发送到 USB_INT_REQ_LP 中断线。通过改变 USBDevIntPri 的值，可以选择将 EP_FAST 或 FRAME 中断发送到中断线。USBDevIntEn 是一个读写寄存器。

表263. USB 设备中断使能寄存器（USBDevIntEn—0x2008 C204）位分配

复位值： 0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	-	-	-	-
位	23	22	21	20	19	18	17	16
符号	-	-	-	-	-	-	-	-
位	15	14	13	12	11	10	9	8
符号	-	-	-	-	-	-	ERR_INT	EP_RLZED
位	7	6	5	4	3	2	1	0
符号	TxENDPKT	Rx ENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

表264. USB 设备中断使能寄存器（USBDevIntEn—0x2008 C204）位描述

位	符号	值	描述	复位值
31:0	见上表的位分配	0	没有中断产生。	0
		1	当设备中断状态（USBDevIntSt）寄存器（表 261）中的对应位置位时，中断产生。默认是将中断发送到 USB_INT_REQ_LP 中断线。另外，通过改变 USBDevIntPri 的值，也可以将 EP_FAST 或 FRAME 中断发送到 USB_INT_REQ_HP 中断线。	

13.10.3.4 USB 设备中断清除寄存器（USBDevIntClr—0x2008 C208）

向此寄存器的某个位写入 1，可将 USBDevIntSt 中的相应位清零。写入 0 则无效。

注：在清零 EP_SLOW 或 EP_FAST 中断位以前，应先清零 USBEpIntSt 中相应端点的中断。

USBDevIntClr 是一个只写寄存器。

表265. USB 设备中断清除寄存器（USBDevIntClr—0x2008 C208）位分配

复位值： 0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	-	-	-	-
位	23	22	21	20	19	18	17	16
符号	-	-	-	-	-	-	-	-
位	15	14	13	12	11	10	9	8
符号	-	-	-	-	-	-	ERR_INT	EP_RLZED
位	7	6	5	4	3	2	1	0
符号	TxENDPKT	Rx ENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

表266. USB 设备中断清零寄存器（USBDevIntClr—0x2008 C208）位描述

位	符号	值	描述	复位值
31:0	见上表的位分配	0	无效。	0
		1	USBDevIntSt 寄存器（见 13.10.3.2 节）中的相应位被清零。	

13.10.3.5 USB 设备中断设置寄存器（USBDevIntSet—0x2008 C20C）

向此寄存器的某个位写入 1，可将 USBDevIntSt 中的相应位置位。写入 0 则无效。

USBDevIntSet 是一个只写寄存器。

表267. USB 设备中断设置寄存器（USBDevIntSet—0x2008 C20C）位分配

复位值: 0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	-	-	-	-
位	23	22	21	20	19	18	17	16
符号	-	-	-	-	-	-	-	-
位	15	14	13	12	11	10	9	8
符号	-	-	-	-	-	-	ERR_INT	EP_RLZED
位	7	6	5	4	3	2	1	0
符号	TxENDPKT	Rx ENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

表268. USB 设备中断设置寄存器（USBDevIntSet—0x2008 C20C）位描述

位	符号	值	描述	复位值
31:0	见上表 USBDevIntSet 的位分配	0	无效。	0
		1	USBDevIntSt 寄存器（见 13.10.3.2 节）中的相应位被置位。	

13.10.3.6 USB 设备中断优先级寄存器（USBDevIntPri—0x2008 C22C）

向此寄存器的某个位写入 1，可将相应中断发送到 USB_INT_REQ_HP 中断线。写入 0 将中断发送到 USB_INT_REQ_LP 中断线。EP_FAST 或 FRAME 中断都可以发送到 USB_INT_REQ_HP 中断线，但不能同时进行。如果软件试图将两位同时设为 0，则没有中断被发送到 USB_INT_REQ_HP。USBDevIntPri 是一个只写寄存器。

表269. USB 设备中断优先级寄存器（USBDevIntPri—0x2008 C22C）位描述

位	符号	值	描述	复位值
0	FRAME	0	FRAME 中断将进入 USB_INT_REQ_LP。	0
		1	FRAME 中断将进入 USB_INT_REQ_HP。	
1	EP_FAST	0	EP_FAST 中断将进入 USB_INT_REQ_LP。	0
		1	EP_FAST 中断将进入 USB_INT_REQ_LP。	
31:2	-		保留。读取值未定义，只写入 0。	无

13.10.4 端点中断寄存器

本组中的寄存器用于简化对端点中断的处理。端点中断在从模式中使用。

13.10.4.1 USB 端点中断状态寄存器（USBEPINTST—0x2008 C230）

每个物理的非同步端点都在此寄存器中对应一个位，用来指示端点产生的中断。当准确接收到一个数据包时，所有非同步 OUT 端点产生一个中断。当成功发送一个数据包或 NAK 握手信号发送到总线上（如果 NAK 中断已使能）时，所有非同步 IN 端点也产生一个中断（见 [339 页 13.12.3 节](#)。此寄存器中的一个位为 1 时，会导致 USBDEVINTST 的 EP_FAST 或 EP_SLOW 位置位，具体是哪个位置位将由 USBEPDEVINTPRI 的相应位的值决定。USBEPINTST 是一个只读寄存器。

注意，对于同步端点，对数据包的处理是在出现 FRAME 中断时。

表270. USB 端点中断状态寄存器（USBEPINTST—0x2008 C230）位分配

复位值: 0x0000 0000

位	31	30	29	28	27	26	25	24
符号	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
位	23	22	21	20	19	18	17	16
符号	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
位	15	14	13	12	11	10	9	8
符号	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
位	7	6	5	4	3	2	1	0
符号	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

表271. USB 端点中断状态寄存器（USBEPINTST—0x2008 C230）位描述

位	符号	描述	复位值
0	EP0RX	端点 0，接收完数据中断位。	0
1	EP0TX	端点 0，发送完数据中断位或发送一个 NAK。	0
2	EP1RX	端点 1，接收完数据中断位。	0
3	EP1TX	端点 1，发送完数据中断位或发送一个 NAK。	0
4	EP2RX	端点 2，接收完数据中断位。	0
5	EP2TX	端点 2，发送完数据中断位或发送一个 NAK。	0
6	EP3RX	端点 3，同步端点。	无
7	EP3TX	端点 3，同步端点。	无
8	EP4RX	端点 4，接收完数据中断位。	0
9	EP4TX	端点 4，发送完数据中断位或发送一个 NAK。	0
10	EP5RX	端点 5，接收完数据中断位。	0
11	EP5TX	端点 5，发送完数据中断位或发送一个 NAK。	0
12	EP6RX	端点 6，同步端点。	无
13	EP6TX	端点 6，同步端点。	无
14	EP7RX	端点 7，接收完数据中断位。	0

位	符号	描述	复位值
15	EP7TX	端点 7，发送完数据中断位或发送一个 NAK。	0
16	EP8RX	端点 8，接收完数据中断位。	0
17	EP8TX	端点 8，发送完数据中断位或发送一个 NAK。	0
18	EP9RX	端点 9，同步端点。	无
19	EP9TX	端点 9，同步端点。	无
20	EP10RX	端点 10，接收完数据中断位。	0
21	EP10TX	端点 10，发送完数据中断位或发送一个 NAK。	0
22	EP11RX	端点 11，接收完数据中断位。	0
23	EP11TX	端点 11，发送完数据中断位或发送一个 NAK。	0
24	EP12RX	端点 12，同步端点。	无
25	EP12TX	端点 12，同步端点。	无
26	EP13RX	端点 13，接收完数据中断位。	0
27	EP13TX	端点 13，发送完数据中断位或发送一个 NAK。	0
28	EP14RX	端点 14，接收完数据中断位。	0
29	EP14TX	端点 14，发送完数据中断位或发送一个 NAK。	0
30	EP15RX	端点 15，接收完数据中断位。	0
31	EP15TX	端点 15，发送完数据中断位或发送一个 NAK。	0

13.10.4.2 USB 端点中断使能寄存器（USBEPIntEn—0x2008 C234）

将此寄存器的一个位设为 1 使 USBEPIntSt 寄存器中的相应位在出现与该端点相关的中断时置位。将此寄存器的一个位设为 0，将使 USBDMARSt 寄存器中的相应位在出现与该端点相关的中断时置位。USBEPIntEn 是一个读写寄存器。

表272. USB 端点中断使能寄存器（USBEPIntEn—0x2008 C234）位分配

复位值： 0x0000 0000

位	31	30	29	28	27	26	25	24
符号	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
位	23	22	21	20	19	18	17	16
符号	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
位	15	14	13	12	11	10	9	8
符号	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
位	7	6	5	4	3	2	1	0
符号	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

表273. USB 端点中断使能寄存器（USBEPIntEn—0x2008 C234）位描述

位	符号	值	描述	复位值
31:0 的位分配	USBEPIntEn	0	USBDMARSt 中的对应位在出现该端点的中断时置位。	0
		1	USBEPIntSt 中的对应位在出现该端点的中断时置位。 暗示该端点处于从模式。	

13.10.4.3 USB 端点中断清除寄存器（USBEPINTCLR—0x2008 C238）

向此寄存器的一个位写入 1，可对相应物理端点执行 SIE “选择端点/清除中断”命令（表 317）。写入 0 无效。在执行“选择端点/清除中断”命令前，硬件要清零 USBDEVINTST 寄存器中的 CDFULL 位。在执行完命令时，CDFULL 位置位，USBCMDDATA 中包含了端点的状态，而 USBEPINTST 中的相应位被清零。

注：

- 当使用 USBEPINTCLR 清除中断时，软件应等待 CDFULL 位置位，确保相应中断已被清零后再继续操作。
- 虽然能够同时将 USBEPINTCLR 中的多个位置位，但建议不要这样做；在操作结束时，在所有清零的多个位中，只有最低有效中断位对应的端点状态可用。
- 另外，SIE “选择端点/清除中断”命令可以通过 SIE 命令寄存器直接调用，但建议采用 USBEPINTCLR 寄存器，因为它的使用更加方便。

每个物理端点在此寄存器中都有自己的保留位。各个位的定义与 USBEPINTST 的位相同，见表 270。USBEPINTCLR 是一个只写寄存器。

表274. USB 端点中断清除寄存器（USBEPINTCLR—0x2008 C238）位分配

复位值： 0x0000 0000

位	31	30	29	28	27	26	25	24
符号	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
位	23	22	21	20	19	18	17	16
符号	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
位	15	14	13	12	11	10	9	8
符号	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
位	7	6	5	4	3	2	1	0
符号	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

表275. USB 端点中断清除寄存器（USBEPINTCLR—0x2008 C238）位描述

位	符号	值	描述	复位值
31:0	见上表 USBEPINTCLR 的位分配	0	无效。	0
		1	通过执行对应端点的 SIE 选择端点/清除中断命令，可令 USBEPINTST 中的对应位清零。	

13.10.4.4 USB 端点中断设置寄存器（USBEPINTSET—0x2008 C23C）

此寄存器中的一个位写入 1，可置位 USBEPINTST 中的相应位。写入 0 无效。每个端点在此寄存器中都有自己的位。USBEPINTSET 是一个只写寄存器。

表276. USB 端点中断设置寄存器（USBEPIntSet—0x2008 C23C）位分配

复位值: 0x0000 0000

位	31	30	29	28	27	26	25	24
符号	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
位	23	22	21	20	19	18	17	16
符号	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
位	15	14	13	12	11	10	9	8
符号	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
位	7	6	5	4	3	2	1	0
符号	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

表277. USB 端点中断设置寄存器（USBEPIntSet—0x2008 C23C）位描述

位	符号	值	描述	复位值
31:0	见上表 USBEPIntSet 的位分配	0	无效。	0
		1	将 USBEPIntSt 中的对应位置位。	

13.10.4.5 USB 端点中断优先级寄存器（USBEPIntPri—0x2008 C240）

此寄存器决定了一个端点中断是进入 USBDevIntSt 的 EP_FAST 还是 EP_SLOW 位。如果此寄存器中的一个位被设为 1，则相应的端点中断进入 EP_FAST，如果是 0 则进入 EP_SLOW。也可以有多个端点的中断进入 EP_FAST 或 EP_SLOW。

注意，USBDevIntPri 寄存器决定了 EP_FAST 中断是进入 USB_INT_REQ_HP 中断线还是 USB_INT_REQ_LP 中断线。

USBEPIntPri 是一个只写寄存器。

表278. USB 端点中断优先级寄存器（USBEPIntPri—0x2008 C240）位分配

复位值: 0x0000 0000

位	31	30	29	28	27	26	25	24
符号	EP15TX	EP15RX	EP14TX	E14RX	EP13TX	EP13RX	EP12TX	EP12RX
位	23	22	21	20	19	18	17	16
符号	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
位	15	14	13	12	11	10	9	8
符号	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
位	7	6	5	4	3	2	1	0
符号	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

表279. USB 端点中断优先级寄存器（USBEPIntPri—0x2008 C240）位描述

位	符号	值	描述	复位值
31:0	见上表 USBepIntPri 的位分配	0	对应中断进入 USBDevIntSt 寄存器的 EP_SLOW 位	0
		1	对应中断进入 USBDevIntSt 寄存器的 EP_FAST 位	
USBDevIntSt				

13.10.5 端点使用寄存器

本组中的寄存器主要用于在运行时使用端点以及对端点进行配置。

13.10.5.1 EP RAM 要求

USB 设备控制器使用一个基于 FIFO 的 RAM 作为端点的缓冲区。这个专用 RAM 叫做端点 RAM（EP_RAM）。每个端点在 EP_RAM 中都有保留的空间。每个端点需要的 EP_RAM 空间取决于其 MaxPacketSize 的值，以及是否具有双缓冲。设备使用 32 个字的 EP_RAM 保存端点缓冲区的指针。EP_RAM 是按字对齐的，但 MaxPacketSize 是以字节定义的，因此 RAM 深度必须调整到下一个字的边界。另外，每个缓冲区都有一个字的头信息(header)，用以显示所接收到的数据包的长度。

物理端点所需要的 EP_ RAM 空间（以字为单位）可以表示为：

$$EPRAMspace = \left(\frac{MaxPacketSize + 3}{4} + 1 \right) \times dbstatus$$

其中：单缓冲端点的 dbstatus 为 1，双缓冲端点的 dbstatus 为 2。

由于所有实现的端点都占用 EP_RAM 空间，因此总的 EP_RAM 要求是：

$$TotalEPRAMspace = 32 + \sum_{n = 0}^N EPRAMspace(n)$$

其中：N 是已使用的端点数量。总 EP_RAM 空间不应超过 4096 个字节（4kB，1k 字）。

13.10.5.2 USB 使用端点寄存器（USBReEp—0x2008 C244）

向此寄存器的一个位写入 1，可使用相应的端点。写入 0 不能使用相应的端点。当发生总线复位时，此寄存器返回自己的复位状态。USBReEp 是一个读写寄存器。

表280. USB 使用端点寄存器（USBReEp—0x2008 C244）位分配

复位值: 0x0000 0003

位	31	30	29	28	27	26	25	24
符号	EP31	EP30	EP29	EP28	EP27	EP26	EP25	EP24
位	23	22	21	20	19	18	17	16
符号	EP23	EP22	EP21	EP20	EP19	EP18	EP17	EP16
位	15	14	13	12	11	10	9	8
符号	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8
位	7	6	5	4	3	2	1	0
符号	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0

表281. USB 使用端点寄存器（USBReEp—0x2008 C244）位描述

位	符号	值	描述	复位值
0	EP0	0	不使用控制端点 EP0。	1
		1	使用控制端点 EP0。	
1	EP1	0	不使用控制端点 EP1。	1
		1	使用控制端点 EP1。	
31:2	EPxx	0	不使用端点 EPxx。	0
		1	使用端点 EPxx。	

在复位时，只有控制端点已使用。其它端点如有必要，可通过设置 USBReEp 中的相应位使用。计算已使用端点所需的 EP_RAM 空间，见 13.10.5.1 节。

端点使用是多周期操作。下面显示的是端点使用的伪代码。

```
Clear EP_RLZED bit in USBDevIntSt;

for every endpoint to be realized,
{
    /* OR with the existing value of the Realize Endpoint register */
    USBReEp |= (UInt32) ((0x1 << endpt));
    /* Load Endpoint index Reg with physical endpoint no.*/
    USBEpIn = (UInt32) endpointnumber;

    /* load the max packet size Register */
    USBEpMaxPSize = MPS;

    /* check whether the EP_RLZED bit in the Device Interrupt Status register is set
    */
    while (!(USBDevIntSt & EP_RLZED))
    {
        /* wait until endpoint realization is complete */
    }
    /* Clear the EP_RLZED bit */
    Clear EP_RLZED bit in USBDevIntSt;
}
```

设备将不会对未使用端点的任何事务做出响应。SIE 配置设备命令只会使已使用和已使能的端点对事务做出响应。详见[表 312](#)。

13.10.5.3 USB 端点索引寄存器（USBEPIn—0x2008 C248）

每个端点都有一个寄存器，用来记录端点的最大包长度值（MaxPacketSize）值。这实际上是一个寄存器数组。因此在写操作之前，要通过 USBEPIn 寄存器“寻址”该寄存器。

USBEPIn 寄存器中保存了物理端点的编号。写入 USBMaxPSize 会设置由 USBEPIn 指定的数组单元。USBEPIn 是一个只写寄存器。

表282. USB 端点索引寄存器（USBEPIn—0x2008 C248）位描述

位	符号	描述	复位值
4:0	PHY_EP	物理端点的编号（0~31）	0
31:5	-	保留。读取值未定义，只写入 0。	无

13.10.5.4 USB MaxPacketSize 寄存器（USBMaxPSize—0x2008 C24C）

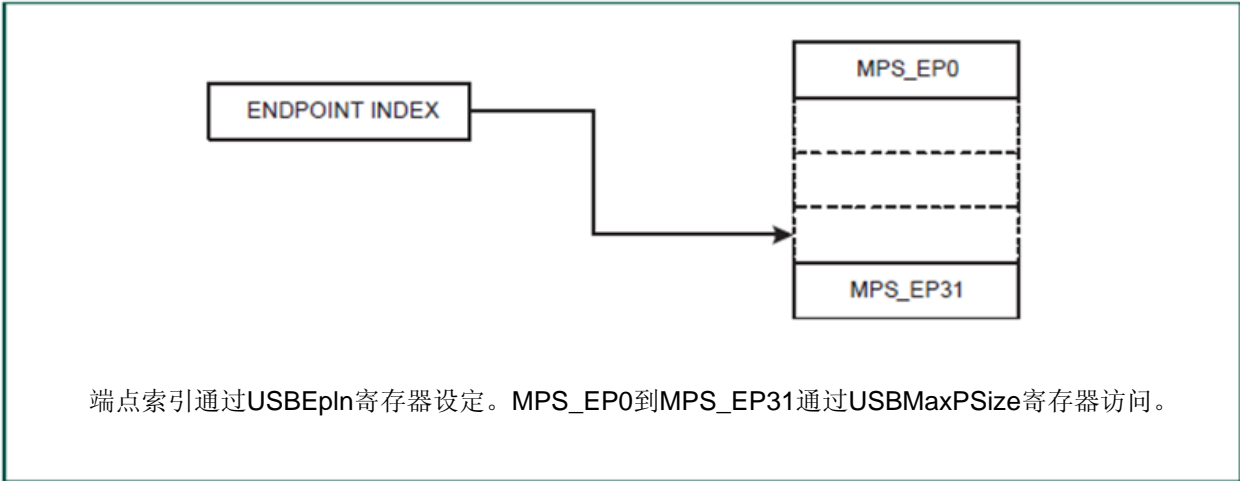
复位时，控制端点的最大数据包长度为 8 字节。其它端点设置为 0。如果修改 USBMaxPSize 的值，需要重新计算 EP_RAM 中的端点缓冲区地址。这是一个多周期处理操作。在结束时，USBDevIntSt 中的 EP_RLZED 位将置位（[表 261](#)）。USBMaxPSize 数组索引见[图 40](#)。USBMaxPSize 是一个读写寄存器。

表283. USB MaxPacketSize 寄存器（USBMaxPSize—0x2008 C24C）位描述

位	符号	描述	复位值
9:0	MPS	最大包长度值。	0x008 ^[1]
31:10	-	保留。读取值未定义，只写入 0。	无

[1] 该值为 EP0 和 EP1 的复位值。其它所有端点的复位值为 0x0。

图40. USB MaxPacketSize 寄存器数组索引



13.10.6USB 传输寄存器

在从模式中操作时，本组寄存器用于在端点缓冲区与 RAM 之间传输数据。见 13.14 节。

13.10.6.1 USB 接收数据寄存器（USBRxData—0x2008 C218）

在 OUT 数据传输中，CPU 从此寄存器中读出端点缓冲区数据。在读此寄存器以前，应相应地设置 USBCtrl 寄存器中的 RD_EN 位和 LOG_ENDPOINT 字段。此寄存器读操作将从所选择的端点缓冲区中提取数据。数据采用小端格式，即：从 USB 总线接收的第一个字节将是 USBRxData 的最低有效字节。USBRxData 是一个只读寄存器。

表284. USB 接收数据寄存器（USBRxData—0x2008 C218）位描述

位	符号	描述	复位值
31:0	RX_DATA	接收数据。	0

13.10.6.2 USB 接收包长度寄存器（USBRxPLen—0x2008 C220）

此寄存器给出了正在传输的当前数据包剩余在端点缓冲区（当前通过 USBRxData 寄存器读取的数据包所在的端点缓冲区）的字节数，并且有一位用于指示数据包是否有效。在读此寄存器以前，应相应设置 USBCtrl 寄存器中的 RD_EN 位和 LOG_ENDPOINT 字段。每次读取 USBRxData 寄存器都会更新 USBRxPLen 寄存器。USBRxPLen 是一个只读寄存器。

表285. USB 接收包长度寄存器（USBRxPLen—0x2008 C220）位描述

位	符号	值	描述	复位值
9:0	PKT_LNGTH	-	可以从当前所选的缓冲区中读出的剩余字节数。当该字段的值减少到 0 时，在 USBDevIntSt 寄存器中的 RxENDPKT 位将置位。	0
10	DV	-	数据有效。该位对于同步端点非常有用。非同步端点在接收到错误的数据包时不会产生中断。但是无效数据包可能和总线复位一同产生。对于同步端点，即使接收到错误的数据包，数据传输仍可继续进行。这种情况下数据包的 DV 位不能置位。	0
		0	数据无效。	
		1	数据有效。	
11	PKT_RDY	-	PKT_LNGTH 字段有效，并且数据包准备就绪可以读出。	0
31:12	-	-	保留。从保留位读出的值未被定义。	无

13.10.6.3 USB 发送数据寄存器（USBTxData—0x2008 C21C）

在 IN 数据传输中，CPU 将端点数据写入此寄存器。在写入寄存器以前，应相应设置 USBCtrl 寄存器中的 WR_EN 位和 LOG_ENDPOINT 字段，数据包长度应写入到 USBTxPLen 寄存器。在写入此寄存器时，数据被写入到所选择的端点缓冲区。数据采用小端格式，即：USB 总线上发送的第一个字节将成为 USBTxData 的最低有效字节。USBTxData 是一个只写寄存器。

表286. USB 发送数据寄存器（USBTxData—0x2008 C21C）位描述

位	符号	描述	复位值
31:0	TX_DATA	发送数据。	0

13.10.6.4 USB 发送包长度寄存器（USBTxPLen—0x2008 C224）

此寄存器包含了从 CPU 传输到所选择的端点缓冲区的字节数。在数据写入 USBTxData 以前，软件应首先向此寄存器中写入数据包长度（≤MaxPacketSize）。在每次对 USBTxData 进行写操作以后，硬件将 USBTxPLen 中的值减去 4。在此过程开始前，USBCtrl 寄存器的 WR_EN 位和 LOG_ENDPOINT 字段应被置位，以选择所需要的端点缓冲区。

如果数据缓冲区的长度大于端点的 MaxPacketSize，软件应提交数据包长度为 MaxPacketSize 的数据，而在最后数据包中发送剩下的多余字节。例如，如果 MaxPacketSize 为 64 字节，要发送的数据缓冲区长度为 130 字节，则软件将发送两个 64 位数据包，并在最后一个包中发送余下的 2 字节。因此，USB 上总共发送了 3 个数据包。USBTxPLen 是一个只写寄存器。

表287. USB 发送包长度寄存器（USBTxPLen—0x2008 C224）位描述

位	符号	值	描述	复位值
9:0	PKT_LENGTH	-	可以写入所选的端点缓冲区的剩余字节数。每次写 USBTxData 寄存器，该域的值由硬件减 4。当该字段减到 0 时，在 USBDevIntSt 寄存器中的 TxENDPKT 位被置位。	0
31:10	-	-	保留。读取值未定义，只写入 0。	无

13.10.6.5 USB 控制寄存器（USBCtrl—0x2008 C228）

此寄存器控制着 USB 设备的数据传输操作。它选择在 USBRxData 和 USBTxData 寄存器访问时的端点缓冲区，并使能它们的读写操作。USBCtrl 是一个读写寄存器。

表288. USB 控制寄存器（USBCtrl—0x2008 C228）位描述

位	符号	值	描述	复位值
0	RD_EN		读模式控制。使能使用 USBRxData 寄存器从 OUT 端点缓冲区中读取数据，端点由本寄存器的 LOG_ENDPOINT 域指定。在从 USBRxData 寄存器中读取当前数据包的最后一个字时，硬件将该位清零。	0
		0	读模式禁能。	
		1	读模式使能。	
1	WR_EN		写模式控制。使能使用 USBTxData 寄存器将数据写入 IN 端点缓冲区。端点由本寄存器的 LOG_ENDPOINT 指定。当 USBTxLen 中的字节数已发送完时，硬件将该位清零。	0
		0	写模式禁能。	
		1	写模式使能。	
5:2	LOG_ENDPOINT	-	逻辑端点编号。	0
31:6	-	-	保留。读取值未定义，只写入 0。	无

13.10.7 SIE 命令代码寄存器

SIE 命令代码寄存器用于与串行接口引擎的通信。更多信息见 [13.12](#) 节。

13.10.7.1 USB 命令代码寄存器（USBCmdCode—0x2008 C210）

此寄存器用于向 SIE 发送命令和写入数据。寄存器中写入的命令被传输到 SIE 并在 SIE 中执行。命令执行后，寄存器变为空，USBDevIntSt 寄存器的 CCEMPTY 位置位。详见 [13.12](#) 节。USBCmdCode 是一个只写寄存器。

表289. USB 命令代码寄存器（USBCmdCode—0x2008 C210）位描述

位	符号	值	描述	复位值
7:0	-		保留。读取值未定义，只写入 0。	无
15:8	CMD_PHASE		命令阶段：	0
		0x02	读	
		0x01	写	
		0x05	命令	
23:16	CMD_CODE/ CMD_WDATA		这是一个多用途域。当 CMD_PHASE 为命令或读状态时，该域包含着命令代码（CMD_CODE）。当 CMD_PHASE 为写状态时，该域包含着命令写数据（CMD_WDATA ）。	0
31:24	-		保留。读取值未定义，只写入 0。	无

13.10.7.2 USB 命令数据寄存器（USBCmdData—0x2008 C214）

此寄存器包含了执行 SIE 命令后所获取的数据。当数据已准备好被读取时，USBDevIntSt 寄存器的 CD_FULL 位置位。详见[表 261](#)。USBCmdData 是一个只读寄存器。

表290. USB 命令数据寄存器（USBCmdData—0x2008 C214）位描述

位	符号	描述	复位值
7:0	CMD_RDATA	命令读数据。	0
31:8	-	保留。从保留位读出的值未被定义。	无

13.10.8 DMA 寄存器

本组寄存器用于 DMA 模式操作（见 [13.15](#) 节）。

13.10.8.1 USB DMA 请求状态寄存器（USBDMARSt—0x2008 C250）

当发生端点中断（见 USBEpIntSt 的描述），且 USBEpIntEn 中的相应位为 0 时，硬件设置此寄存器中与某个非同步端点相关的位。当 USBEpIntEn 中的相应位为 0，且发生 FRAME 中断时，与某个同步端点相关的位置位。如果对于 USBEpDMASr 寄存器中相应端点来说 DMA 是使能的，则该寄存器中为 1 的位可用作 DMA 引擎开始数据传输标志。DMA 不能使能用于控制端点（EP0 和 EP1）。USBDMARSt 是一个只读寄存器。

表291. USB DMA 请求状态寄存器（USBDMARSt—0x2008 C250）位分配

复位值: 0x0000 0000

位	31	30	29	28	27	26	25	24
符号	EP31	EP30	EP29	EP28	EP27	EP26	EP25	EP24
位	23	22	21	20	19	18	17	16
符号	EP23	EP22	EP21	EP20	EP19	EP18	EP17	EP16
位	15	14	13	12	11	10	9	8
符号	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8
位	7	6	5	4	3	2	1	0
符号	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0

表292. USB DMA 请求状态寄存器（USBDMARSt—0x2008 C250）位描述

位	符号	值	描述	复位值
0	EP0	0	控制端点 OUT（对于这个端点，DMA 不可以使能，EP0 位必须为 0）。	0
1	EP1	0	控制端点 IN（对于这个端点，DMA 不可以使能，EP1 位必须为 0 ）。	0
31:2	EPxx		端点 xx（2≤xx≤31）的 DMA 请求。	0
		0	端点 xx 没有请求 DMA。	
		1	端点 xx 请求 DMA。	

[1] 对于这个端点，DMA 不可以使能，USBDMARSt 中的对应位必须为 0。

13.10.8.2 USB DMA 请求清除寄存器（USBDMARClr—0x2008 C254）

向此寄存器的一个位写入 1，将清零 USBDMARSt 寄存器中的相应位。写入 0 无效。

在使能端点的 DMA 操作前，应用此寄存器进行初始化。当使能了端点的 DMA 操作时，硬件会在完成一个数据包传输后将 USBDMARSt 中的相应位清零。因此，在端点的 DMA 操作使能时，软件不应使用该寄存器进行清零操作。

USBDMARClr 是一个只写寄存器。

USBDMARClr 寄存器各位的分配与 USBDMARSt 寄存器相同（[表 291](#)）。

表293. USB DMA 请求清除寄存器（USBDMARClr—0x2008 C254）位描述

位	符号	值	描述	复位值
0	EP0	0	控制端点 OUT（对于这个端点，DMA 不可以使能，EP0 位必须为 0 ）。	0
1	EP1	0	控制端点 IN（对于这个端点，DMA 不可以使能，EP1 位必须为 0 ）。	0
31:2	EPxx		清除端点 xx（2≤xx≤31）的 DMA 请求。	0
		0	无效。	
		1	清除 USBDMARSt 中的相应位。	

13.10.8.3 USB DMA 请求设置寄存器（USBDMARSet—0x2008 C258）

向此寄存器的一个位写入 1，可将 USBDMARSt 寄存器中的相应位置位。写入 0 无效。

此寄存器允许软件提出 DMA 请求。这一特性在某个端点由从模式切换到 DMA 模式时非常有用：如果在 USBEpIntEn 的相应位清零以前，就收到了一个要在 DMA 模式下处理的数据包，则硬件不会提出 DMA 请求。此时软件可以使用此寄存器手动启动 DMA 传输。

软件也可以用此寄存器启动 DMA 传输，从而能够在收到主机的 IN 令牌包以前，就预先填充 IN 端点缓冲区。

USBDMARSet 是一个只写寄存器。
USBDMARSet 寄存器各位的分配与 USBDMARSt 寄存器相同（[表 291](#)）。

表294. USB DMA 请求设置寄存器（USBDMARSet—0x2008 C258）位描述

位	符号	值	描述	复位值
0	EP0	0	控制端点 OUT（对于这个端点，DMA 不可以使能，EP0 位必须为 0）。	0
1	EP1	0	控制端点 IN（对于这个端点，DMA 不可以使能，EP1 位必须为 0 ）。	0
31:2	EPxx		设置端点 xx（2≤xx≤31）的 DMA 请求。	0
		0	无效。	
		1	设置 USBDMARSt 中的相应位。	

13.10.8.4 USB UDCA Head 寄存器（USBUDCAH—0x2008 C280）

UDCA（USB 设备通信区域）Head 寄存器保留 UDCA 在 RAM 中的地址。有关 UDCA 与 DMA 描述符的详细说明，见 [13.15.2](#) 节和 [13.15.4](#) 节。USBUDCAH 是一个读写寄存器。

表295. USB UDCA Head 寄存器（USBUDCAH—0x2008 C280）位描述

位	符号	描述	复位值
6:0	-	保留。读取值未定义，只写入 0。UDCA 以 128 字节为边界对齐。	0
31:7	UDCA_ADDR	UDCA 的起始地址。	0

13.10.8.5 USB EP DMA 状态寄存器（USBEPDMASt—0x2008 C284）

此寄存器中的各个位指示着相应端点的 DMA 操作是否已使能。只有此寄存器中的相应位置位的情况下，端点的 DMA 传输才能启动。USBEPDMASt 是一个只读寄存器。

表296. USB EP DMA 状态寄存器（USBEPDMASt—0x2008 C284）位描述

位	符号	值	描述	复位值
0	EP0_DMA_ENABLE	0	控制端点 OUT（对于这个端点，DMA 不可以使能，EP0_DMA_ENABLE 位必须为 0）。	0
1	EP1_DMA_ENABLE	0	控制端点 IN（对于这个端点，DMA 不可以使能，EP1_DMA_ENABLE 位必须为 0）。	0
31:2	EPxx_DMA_ENABLE		表示端点 xx（2≤xx≤31）的 DMA 是否使能。	0
		0	禁止端点 EPxx 的 DMA 操作。	
		1	使能端点 EPxx 的 DMA 操作。	

13.10.8.6 USB EP DMA 使能寄存器（USBEPDMAEn—0x2008 C288）

向此寄存器中的一个位写入 1，将使能相应端点的 DMA 操作。写入 0 无效。控制端点 EP0 和 EP1 的 DMA 操作不能使能。USBEPDMAEn 是一个只写寄存器。

表297. USB EP DMA 使能寄存器（USBEPDMAEn—0x2008 C288）位描述

位	符号	值	描述	复位值
0	EP0_DMA_ENABLE	0	控制端点 OUT（对于这个端点，DMA 不可以使能，EP0_DMA_ENABLE 位必须为 0）。	0
1	EP1_DMA_ENABLE	0	控制端点 IN（对于这个端点，DMA 不可以使能，EP1_DMA_ENABLE 位必须为 0）。	0
31:2	EPxx_DMA_ENABLE		端点 xx（2≤xx≤31）的 DMA 使能控制位。	0
		0	无效。	
		1	使能端点 EPxx 的 DMA 操作。	

13.10.8.7 USB EP DMA 禁能寄存器（USBEPDMADis—0x2008 C28C）

向此寄存器中的一个位写入 1，可将 USBEPDMASt 中的相应位清零。写入 0 则对 USBEPDMASt 的相应位无效。对此寄存器的任何写入都会清零内部 DMA_PROCEED 标志。有关 DMA_PROCEED 标志的更多信息参见 [13.15.5.4](#) 节。如果在相应位清零时，一个端点正在处理 DMA 传输，则在传输完成后，DMA 才被禁能。当 DMA 传输过程中检测到错误条件时，硬件将清零相应的位。USBEPDMADis 是一个只写寄存器。

表298. USB EP DMA 禁能寄存器（USBEPDMADis—0x2008 C28C）位描述

位	符号	值	描述	复位值
0	EP0_DMA_DISABLE	0	控制端点 OUT（对于这个端点，DMA 不可以使能，EP0_DMA_DISABLE 位必须为 0）。	0
1	EP1_DMA_DISABLE	0	控制端点 IN（对于这个端点，DMA 不可以使能，EP1_DMA_DISABLE 位必须为 0）。	0
31:2	EPxx_DMA_DISABLE		端点 xx（2≤xx≤31）的 DMA 禁能控制位。	0
		0	无效。	
		1	禁能端点 EPxx 的 DMA 操作。	

13.10.8.8 USB DMA 中断状态寄存器（USBDMAIntSt—0x2008 C290）

此寄存器的每个位反映在相应中断状态寄存器的 32 个位中，是否有任何一个位置位。USBDMAIntSt 是一个只读寄存器。

表299. USB DMA 中断状态寄存器（USBDMAIntSt—0x2008 C290）位描述

位	符号	值	描述	复位值
0	EOT		传输结束中断位。	0
		0	USBEoTIntSt 寄存器中的所有位都为 0。	
		1	USBEoTIntSt 寄存器中至少有一个位为 1。	
1	NDDR		新的 DD 请求中断位。	0
		0	USBNDDRIntSt 寄存器中的所有位都为 0。	
		1	USBNDDRIntSt 寄存器中至少有一个位为 1。	
2	ERR		系统错误中断位。	0
		0	USBSysErrIntSt 寄存器中的所有位都为 0。	
		1	USBSysErrIntSt 寄存器中至少有一个位为 1。	
31:3	-		保留。从保留位读出的值未被定义。	无

13.10.8.9 USB DMA 中断使能寄存器（USBDMAIntEn—0x2008 C294）

向此寄存器的一个位写入 1，使能 USBDMAIntSt 中的相应位，从而在 USB_INT_REQ_DMA 中断线上（已使能情况下）产生一个中断。USBDMAIntEn 是一个读写寄存器。

表300. USB DMA 中断使能寄存器（USBDMAIntEn—0x2008 C294）位描述

位	符号	值	描述	复位值
0	EOT		传输结束中断使能位。	0
		0	传输结束中断禁能。	
		1	传输结束中断使能。	
1	NDDR		新的 DD 请求中断使能位。	0
		0	新的 DD 请求中断禁能。	
		1	新的 DD 请求中断使能。	
2	ERR		系统错误中断使能位。	0
		0	系统错误中断禁能。	
		1	系统错误中断使能。	
31:3	-		保留。读取值未定义，只写入 0。	无

13.10.8.10 USB 传输结束中断状态寄存器（USBEOIntSt—0x2008 C2A0）

在当前 DMA 描述符的 DMA 传输完成时，无论是正常结束（描述符退出）还是出现错误，都会设置将寄存器中相应端点的位置位。中断的原因记录在描述符的 DD_status 区域中。USBEOIntSt 是一个只读寄存器。

表301. USB 传输结束中断状态寄存器（USBEOIntSt—0x2008 C2A0）位描述

位	符号	值	描述	复位值
31:0	EPxx		端点 xx (2≤xx≤31) 的传输结束中断请求。	0
		0	没有端点 xx 的传输结束中断请求。	
		1	有端点 xx 的传输结束中断请求。	

13.10.8.11 USB 传输结束中断清零寄存器（USBEOIntClr—0x2008 C2A4）

向此寄存器的一个位写入 1，会将 USBEOIntSt 寄存器中的相应位清零。写入 0 无效。USBEOIntClr 是一个只写寄存器。

表302. USB 传输结束中断清零寄存器（USBEOIntClr—0x2008 C2A4）位描述

位	符号	值	描述	复位值
31:0	EPxx		清除端点 xx (2≤xx≤31) 的传输结束中断请求。	0
		0	无效。	
		1	清除 USBEOIntSt 寄存器中 EPxx 传输结束的中断请求。	

13.10.8.12 USB 传输结束中断置位寄存器（USBEOIntSet—0x2008 C2A8）

向此寄存器的一个位写入 1，会将 USBEOIntSt 寄存器中的相应位置位。写入 0 无效。USBEOIntSet 是一个只写寄存器。

表303. USB 传输结束中断置位寄存器（USBEOIntSet—0x2008 C2A8）位描述

位	符号	值	描述	复位值
31:0	EPxx		设置端点 xx (2≤xx≤31) 的传输结束中断请求。	0
		0	无效。	
		1	将 USBEOIntSt 寄存器中 EPxx 传输结束的中断请求置位。	

13.10.8.13 USB 新 DD 请求中断状态寄存器（USBNDDRIntSt—0x2008 C2AC）

当从 USB 中请求一次传输并且相应端点未探测到有效 DD 时，此寄存器的位置位。USBNDDRIntSt 是一个只读寄存器。

表304. USB 新 DD 请求中断状态寄存器（USBNDDRIntSt—0x2008 C2AC）位描述

位	符号	值	描述	复位值
31:0	EPxx		端点 xx (2≤xx≤31) 的新 DD 中断请求。	0
		0	没有端点 xx 的新 DD 中断请求。	
		1	有端点 xx 的新 DD 中断请求。	

13.10.8.14 USB 新 DD 请求中断清零寄存器（USBNDDRIntClr—0x2008 C2B0）

向此寄存器的一个位写入 1，会将 USBNDDRIntSt 寄存器中的相应位清零。写入 0 无效。USBNDDRIntClr 是一个只写寄存器。

表305. USB 新 DD 请求中断清零寄存器（USBNDDRIntClr—0x2008 C2B0）位描述

位	符号	值	描述	复位值
31:0	EPxx		清除端点 xx ($2 \leq xx \leq 31$) 的新 DD 中断请求。	0
		0	无效。	
		1	清除 USBNDDRIntSt 寄存器中的端点 xx 的新 DD 中断请求。	

13.10.8.15 USB 新 DD 请求中断置位寄存器（USBNDDRIntSet—0x2008 C2B4）

向此寄存器的一个位写入 1，会将 USBNDDRIntSt 寄存器中的相应位置位。写入 0 无效。USBNDDRIntSet 是一个只写寄存器。

表306. USB 新 DD 请求中断置位寄存器（USBNDDRIntSet—0x2008 C2B4）位描述

位	符号	值	描述	复位值
31:0	EPxx		设置端点 xx ($2 \leq xx \leq 31$) 的新 DD 中断请求。	0
		0	无效。	
		1	置位 USBNDDRIntSt 寄存器中的端点 xx 的新 DD 中断请求。	

13.10.8.16 USB 系统错误中断状态寄存器（USBSysErrIntSt—0x2008 C2B8）

如果在数据传输时或当获取或更新 DD 时，发生了一个系统错误（AHB 总线错误），则此寄存器的相应位置位。USBSysErrIntSt 是一个只读寄存器。

表307. USB 系统错误中断状态寄存器（USBSysErrIntSt—0x2008 C2B8）位描述

位	符号	值	描述	复位值
31:0	EPxx		端点 xx ($2 \leq xx \leq 31$) 的系统错误中断请求。	0
		0	没有端点 xx 的系统错误中断请求。	
		1	有端点 xx 的系统错误中断请求。	

13.10.8.17 USB 系统错误中断清零寄存器（USBSysErrIntClr—0x2008 C2BC）

向此寄存器的一个位写入 1，会将 USBSysErrIntSt 寄存器中的相应位清零。写入 0 无效。USBSysErrIntClr 是一个只写寄存器。

表308. USB 系统错误中断清零寄存器（USBSysErrIntClr—0x2008 C2BC）位描述

位	符号	值	描述	复位值
31:0	EPxx		清除端点 xx ($2 \leq xx \leq 31$) 的系统错误中断请求。	0
		0	无效。	
		1	清除 USBSysErrIntSt 寄存器中的端点 xx 的系统错误中断请求。	

13.10.8.18 USB 系统错误中断置位寄存器（USBSysErrIntSet—0x2008 C2C0）

向此寄存器的一个位写入 1，会将 USBSysErrIntSt 寄存器中的相应位置位。写入 0 无效。
USBSysErrIntSet 是一个只写寄存器。

表309. USB 系统错误中断置位寄存器（USBSysErrIntSet—0x2008 C2C0）位描述

位	符号	值	描述	复位值
31:0	EPxx		设置端点 xx（2≤xx≤31）的系统错误中断请求。	0
		0	无效。	
		1	将 USBSysErrIntSt 寄存器中的端点 xx 的系统错误中断请求置位。	

13.11 中断处理

本节描述了如何将任何端点上的中断事件发送到可嵌套向量中断控制器（NVIC）。中断事件的处理如[图 41](#)所示。

所有非同步 OUT 端点（控制、批量与中断端点）在成功地收到一个数据包时会产生中断。所有非同步 IN 端点会在成功发送了一个包时，或发送了一个 NAK 信号且通过 SIE 设置模式命令使能了 NAK 中断时产生中断，见[13.12.3](#)节。对于同步端点，每 1ms 生成一个帧中断。

从模式下和 DMA 模式的中断处理是不同的。

从模式

如果端点上发生了一个中断事件，并且在 USBEpIntEn 寄存器中使能了端点中断，则 USBEpIntSt 中的相应状态位置位。对于非同步端点，所有端点的中断事件通过相应的 USBEpIntPri[n]寄存器被分为两种类型：快速端点中断事件和慢速端点中断事件。所有快速端点中断事件相“或”，并发送到 USBDevIntSt 寄存器的 EP_FAST 位。所有慢速端点中断事件相“或”，并发送到 USBDevIntSt 寄存器的 EP_SLOW 位。

对于同步端点，USBDevIntSt 中的 FRAME 位每 1ms 设置一次。

USBDevIntSt 寄存器保存着所有端点中断事件的状态，以及各种其它中断的状态（见[13.10.3.2](#)节）。默认情况下，所有中断（如果已在 USBDevIntEn 中使能）发送到 USBIntSt 寄存器中的 USB_INT_REQ_LP 位，请求低优先级的中断处理。但是，USBDevIntPri 寄存器可以将 FRAME 或 EP_FAST 位发送到 USBIntSt 寄存器的 USB_INT_REQ_HP 位。

EP_FAST 和 FRAME 中断事件中只有一个可以发送到 USB_INT_REQ_HP 位。如果试图将两个位都发送到 USB_INT_REQ_HP，则两个中断事件都会被发送到 USB_INT_REQ_LP。

慢速端点中断事件总是直接发送到 USB_INT_REQ_LP 位，由软件请求低优先级的中断处理。

发送到 NVIC 的最后中断信号由 USBIntSt 寄存器中的 EN_USB_INTS 位控制。只有当 EN_USB_INTS 位置位时，USB 中断才会发送到 NVIC。

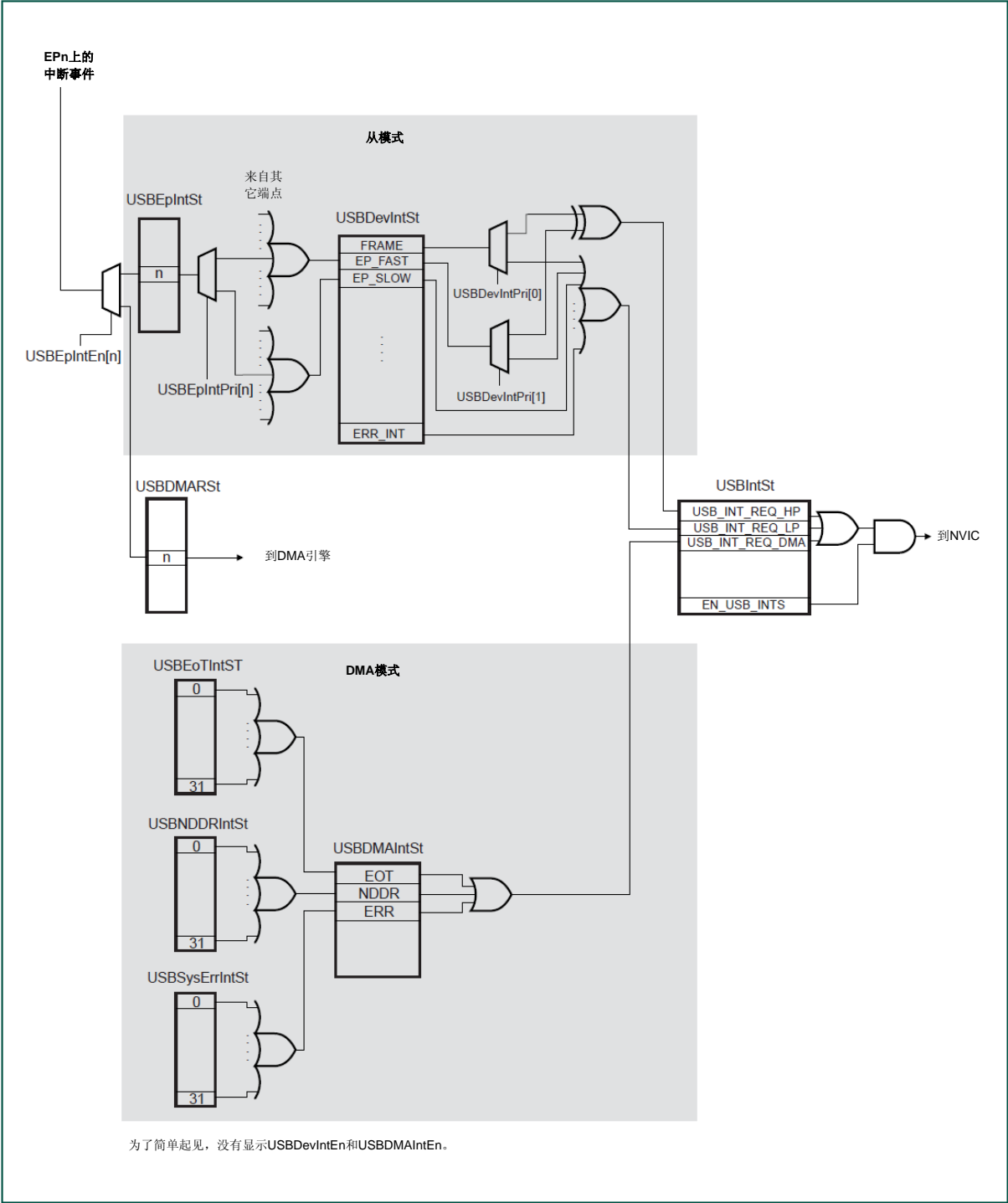
DMA 模式

如果中断事件发生在一个非控制端点，并且在 USBEpIntEn 寄存器中未使能端点中断，则 USBDMARSt 中的相应状态位由硬件置位。如果 USBEpDMASt 寄存器中使能了相应端点的 DMA 传输，则上述状态位作为 DMA 引擎传输数据的标志。

在 DMA 模式下，每个端点的数据传输可以产生 3 种类型的中断：传输结束中断、新 DD 请求中断，以及系统错误中断。这些中断事件会分别把 USBEoTIntSt、USBNDDRIntSt 和 USBSysErrIntSt 寄存器中与各个端点对应的位位置。然后，所有端点的传输结束中断相“或”，并发送至 USBDMAIntSt 的 EOT 位。同样，所有新 DD 请求中断和系统错误中断事件分别发送到 USBDMAStInt 寄存器的 NDDR 位和 ERR 位。

EOT、NDDR 和 ERR 位（如果在 USBDMAIntEn 中使能）相“或”以将 USBIntSt 寄存器中的 USB_INT_REQ_DMA 位置位。如果 USBIntSt 中的 EN_USB_INTS 位置位，则中断发送到 NVIC。

图41. 中断事件处理



13.12 串行接口引擎命令描述

串行接口引擎 (SIE) 的函数与寄存器都通过命令访问, 命令由命令代码组成, 后面是可选的数据字节 (读操作或写操作)。在执行上述访问时将使用 USBCmdCode (表 289) 与 USBCmdData (表 290) 寄存器。

一次完整访问包括两个阶段:

1. **命令阶段:** 将 USBCmdCode 寄存器 CMD_PHASE 字段设置为 0x05 (命令), CMD_CODE 字段设置为所需命令代码。命令执行完成后, USBDevIntSt 的 CCEMPTY 位置位。
2. **数据阶段 (可选):** 如果是写操作, 将 USBCmdCode 寄存器 CMD_PHASE 字段设置为 0x01 (写), 而 CMD_WDATA 字段设置为所需的写数据。写操作完成后, USBDevIntSt 的 CCEMPTY 位置位。如果是读操作, 将 USBCmdCode 寄存器 CMD_PHASE 字段设置为 0x02 (读), 并用与相应的命令代码设置 CMD_CODE 字段。读操作完成后, USBDevIntSt 的 CDFULL 位置位, 表示 USBCmdData 寄存器中的数据已可用于读入。在执行读操作时是可用的。对于多字节寄存器, 先访问最低有效字节。

表 310 给出了可用的命令。

以下是“读取当前帧编号”命令的例子 (读 2 个字节):

```
USBDevIntClr = 0x30;           // Clear both CCEMPTY & CDFULL
USBCmdCode = 0x00F50500;      // CMD_CODE=0xF5, CMD_PHASE=0x05 (Command)
while (!(USBDevIntSt & 0x10)); // Wait for CCEMPTY.
USBDevIntClr = 0x10;           // Clear CCEMPTY interrupt bit.
USBCmdCode = 0x00F50200;      // CMD_CODE=0xF5, CMD_PHASE=0x02 (Read)
while (!(USBDevIntSt & 0x20)); // Wait for CDFULL.
USBDevIntClr = 0x20;           // Clear CDFULL.
CurFrameNum = USBCmdData;     // Read Frame number LSB byte.
USBCmdCode = 0x00F50200;      // CMD_CODE=0xF5, CMD_PHASE=0x02 (Read)
while (!(USBDevIntSt & 0x20)); // Wait for CDFULL.
Temp = USBCmdData;             // Read Frame number MSB byte
USBDevIntClr = 0x20;           // Clear CDFULL interrupt bit.
CurFrameNum = CurFrameNum | (Temp << 8);
```

下面是设置地址命令的例子 (写 1 个字节):

```
USBDevIntClr = 0x10;           // Clear CCEMPTY.
USBCmdCode = 0x00D00500;      // CMD_CODE=0xD0, CMD_PHASE=0x05 (Command)
while (!(USBDevIntSt & 0x10)); // Wait for CCEMPTY.
USBDevIntClr = 0x10;           // Clear CCEMPTY.
USBCmdCode = 0x008A0100;      // CMD_WDATA=0x8A (DEV_EN=1, DEV_ADDR=0xA),
                                // CMD_PHASE=0x01 (Write)
while (!(USBDevIntSt & 0x10)); // Wait for CCEMPTY.
USBDevIntClr = 0x10;           // Clear CCEMPTY.
```

表310. SIE 命令代码表

命令名称	接受者	代码 (Hex)	数据阶段
设备命令			
设置地址	设备	D0	写 1 个字节
配置设备	设备	D8	写 1 个字节
设置模式	设备	F3	写 1 个字节
读取当前帧号	设备	F5	读 1 或 2 个字节
读测试寄存器	设备	FD	读 2 个字节
设置设备状态	设备	FE	写 1 个字节
获得设备状态	设备	FE	读 1 个字节
获得错误代码	设备	FF	读 1 个字节
读错误状态	设备	FB	读 1 个字节
端点命令			
选择端点	端点 0	00	读 1 个字节 (可选)
	端点 1	01	读 1 个字节 (可选)
	端点 xx	xx	读 1 个字节 (可选)
选择端点/清除中断	端点 0	40	读 1 个字节
	端点 1	41	读 1 个字节
	端点 xx	xx+40	读 1 个字节
设置端点状态	端点 0	40	写 1 个字节
	端点 1	41	写 1 个字节
	端点 xx	xx+40	写 1 个字节
清空缓冲区	所选的端点	F2	读 1 个字节 (可选)
确认缓冲区	所选的端点	FA	无

13.12.1 设置地址（命令：0xD0，数据：写 1 个字节）

“设置地址”命令用于设置 USB 分配的地址，并使能（嵌入的）函数。设备设置的地址将在控制事务的状态阶段以后生效。在总线复位后，DEV_ADDR 被设为 0x00，DEV_EN 被设为 1。设备将响应函数地址 0x00，端点 0（默认端点）的数据包。

表311. 设置地址命令位描述

位	符号	描述	复位值
6:0	DEV_ADDR	由软件设置的设备地址。总线复位之后，该字段的值为 0x00。	0
7	DEV_EN	设备使能。总线复位之后，该位为 1。	0
		0: 设备不会响应任何包。	
		1: 设备将响应函数地址为 DEV_ADDR 的数据包。	

13.12.2 配置设备（命令：0xD8，数据：写 1 个字节）

向寄存器写入 1 表示该设备已被配置，且所有已使能的非控制端点都将响应。默认情况下，即使设备没有配置，控制端点也始终是使能的并会作出响应。

表312. 配置设备命令位描述

位	符号	描述	复位值
0	CONF_DEVICE	对设备进行配置。所有使能的非控制端点将作出响应。在总线复位时，该位由硬件清零。 当该位置位时，如果设备不是在挂起状态（SUS=0），则 UP_LED 信号被驱动为低电平。	0
7:1	-	保留。读取值未定义，只写入 0。	无

13.12.3 设置模式（命令：0xF3，数据：写 1 个字节）

表313. 设置模式命令位描述

位	符号	值	描述	复位值
0	AP_CLK		始终是 PLL 时钟。	0
		0	USB_NEED_CLK 有效；当设备进入挂起状态时，可以将 48MHz 时钟停止。	
		1	USB_NEED_CLK 固定为 1；当设备进入挂起状态时，不可以将 48MHz 时钟停止。	
1	INAK_CI		控制 IN 端点的 NAK 中断。	0
		0	只有在成功处理时才产生中断。	
		1	当 IN 处理成功完成以及获得 NAK 应答时都产生中断。	
2	INAK_CO		控制 OUT 端点的 NAK 中断。	0
		0	只有在成功处理时才产生中断。	
		1	当 OUT 处理成功完成以及获得 NAK 应答时都产生中断。	
3	INAK_II		中断 IN 端点的 NAK 中断。	0
		0	只有在成功处理时才产生中断。	
		1	当 IN 处理成功完成以及获得 NAK 应答时都产生中断。	
4	INAK_IO ^[1]		中断 OUT 端点的 NAK 中断。	0
		0	只有在成功处理时才产生中断。	
		1	当 OUT 处理成功完成以及获得 NAK 应答时都产生中断。	
5	INAK_BI		批量 IN 端点的 NAK 中断。	0
		0	只有在成功处理时才产生中断。	
		1	当 IN 处理成功完成以及获得 NAK 应答时都产生中断。	
6	INAK_BO ^[2]		批量 OUT 端点的 NAK 中断。	0
		0	只有在成功处理时才产生中断。	
		1	当 OUT 处理成功完成以及获得 NAK 应答时都产生中断。	
7	-		保留。读取值未定义，只写入 0。	无

[1] 如果 DMA 对于任何的中断 OUT 端点来说都是使能的，则该位应复位为 0。
[2] 如果 DMA 对于任何的批量 OUT 端点来说都是使能的，则该位应复位为 0。

13.12.4 读当前帧编号（命令：0xF5，数据：读 1 或 2 个字节）

返回上一次成功接收的 SOF 的帧编号。帧编号为 11 位宽，首先返回最低有效字节。如果用户只需要帧编号的低 8 位，则只需要读第一个字节。

- 如果在一帧开始时，设备没有收到 SOF，则返回的帧编号是上一次成功接收的 SOF 的帧编号。
- 如果 SOF 帧编号中有 CRC 错误，则设备收到的返回编号将是损坏的。

13.12.5 读测试寄存器（命令：0xFD，数据：读 2 个字节）

测试寄存器为 16 位宽。如果 USB 时钟（usbclk 与 AHB 从机时钟）正在运行，则它将返回 0xA50F 值。

13.12.6 设置设备状态（命令：0xFE，数据：写 1 个字节）

“设置设备状态”命令用于设置“设备状态寄存器”中的位。

表314. 设置设备状态命令位描述

位	符号	值	描述	复位值
0	CON		连接位表示设备的当前连接状态。它对用于 SoftConnect 的 CONNECT 输出管脚进行控制。读取连接位时将返回当前的连接状态。当 V _{BUS} 状态输入为低电平并持续 3ms 以上时，该位由硬件清零。	0
		0	向该位写入 0 将使 CONNECT 管脚变为高电平。	
		1	向该位写入 1 将使 CONNECT 管脚变为低电平。	
1	CON_CH		连接发生改变。	0
		0	该位在读操作时清零。	
		1	当设备的上拉电阻由于 V _{BUS} 消失而断开连接时，该位置位。当该位为 1 时，产生 DEV_STAT 中断。	
2	SUS		挂起：挂起位表示当前的挂起状态。 当设备被挂起（SUS=1）并且 CPU 向 SUS 位写入 0 时，该设备将产生一个远程唤醒。这只有在设备被连接时（CON=1）才发生。 当设备没有连接或没有挂起时，向该位写入 0 是无效的。 向该位写入 1 也无效。	0
		0	出现任何活动时，该位复位为 0。	
		1	当设备在其上行端口（upstream port）上，持续 3ms 以上都没有看到任何活动时，该位置位。	
3	SUS_CH		挂起位（SUS）变化指示器。SUS 位在以下情况下会翻转： 设备进入挂起状态。 设备断开连接。 设备在其上行端口上接收到恢复信号。该位在读操作时清零。	0
		0	SUS 位没有改变。	
		1	SUS 位发生改变。同时产生一个 DEV_STAT 中断。	

位 符号	值 描述	复位值
4 RST	总线复位位。在总线复位时，设备将自动进入默认状态。在默认状态下： <ul style="list-style-type: none">• 设备没有配置。• 将对地址 0 作出响应。• 控制端点将处于暂停状态。• 除了控制端点 EP0 和 EP1 以外，所有端点都没有实现。• 所有端点的数据切换（data toggling）均被复位。• 所有缓冲区被清零。• 端点中断状态没有发生改变。• 产生 DEV_STAT 中断。 注：当设备没有被连接（CON=0）时将忽略总线复位。	0
	0 该位在读操作时清零。	
	1 在设备接收到总线复位时，该位置位。产生 DEV_STAT 中断。	
7:5 -	保留。读取值未定义，只写入 0。	无

13.12.7 获得设备状态（命令：0xFE，数据：读 1 个字节）

“获得设备状态”命令返回“设备状态寄存器”中的值。读设备状态将返回 1 个字节的数
据。位定义与“设置设备状态寄存器”相同，见[表 314](#)。

注：为保证操作正确，必须在执行“获得设备状态”命令前，将 USBDevIntSt 中的 DEV_STAT
位清零。

13.12.8 获得错误代码（命令：0xFF，数据：读 1 个字节）

SIE 中可能产生不同的错误状态。“获得错误代码”命令返回上一个已发生的错误代码。错误
代码由 4 个最低有效位构成。

表315. 获得错误代码命令位描述

位	符号	值	描述	复位值
3:0	EC		错误代码。	0
		0000	无错误。	
		0001	PID 编码错误。	
		0010	未知的 PID。	
		0011	意外的数据包—任何违反规范的包序列。	
		0100	令牌 CRC 中的错误。	
		0101	数据 CRC 中的错误。	
		0110	超时错误。	
		0111	多路串扰 (Babble)。	
		1000	数据包结尾时的错误。	
		1001	发送/接收 NAK。	
		1010	发送暂停。	
		1011	缓冲区溢出错误。	
		1100	发送空包 (只针对 ISO 端点)。	
		1101	位填充错误。	
		1110	同步时的错误。	
		1111	数据 PID 中的 Toggle 位错误, 数据无效。	
4	EA	-	一旦读该寄存器, Error Active 位将被复位。	无
7:5	-		保留。读取值未定义, 只写入 0。	

13.12.9 读错误状态（命令：0xFB，数据：读 1 个字节）

此命令从 USB 设备读取 8 位错误寄存器。该寄存器记录了 SIE 上最近发生的错误事件。如果其中任一位置位, 则 USBDevIntSt 的 ERR_INT 位置位。在对此寄存器进行读操作后, 错误位清零。

表316. 读错误状态命令位描述

位	符号	描述	复位值
0	PID_ERR	PID 编码错误、未知的 PID 或未知的令牌 CRC。	0
1	UEPKT	意外的数据包—任何违反规范的数据包序列。	0
2	DCRC	数据 CRC 错误。	0
3	TIMEOUT	超时错误。	0
4	EOP	数据包结尾错误。	0
5	B_OVRN	缓冲区溢出。	0
6	BTSTF	位填充错误。	0
7	TGL_ERR	数据 PID 中的错误翻转位 (toggle bit), 数据无效。	0

13.12.10 选择端点（命令：0x00—0x1F，数据：读 1 个字节（可选））

“选择端点”命令将一个内部指针初始化为指向 EP_RAM 中的所选缓冲区的起始字节。这个命令可以选跟随读数据操作, 从而在端点缓冲区的数据包上返回一些附加信息。“选择端点”命令的命令代码与物理端点编号相同。对于单缓冲端点, B_2_FULL 位无效。

表317. 选择端点命令位描述

位	符号	值	描述	复位值
0	FE		满/空。该位表示端点缓冲区的满/空状态。对于 IN 端点，FE 位是 B_1_FULL 和 B_2_FULL 位相与的结果。对于 OUT 端点，FE 位是 B_1_FULL 和 B_2_FULL 位相或的结果。对于单缓冲端点，该位只简单地反映 B_1_FULL 的状态。	0
		0	对于 IN 端点，至少有一个端点写缓冲区是空的。	
		1	对于 OUT 端点，至少有一个端点读缓冲区是满的。	
1	ST		暂停的端点指示器。	0
		0	所选的端点没有暂停。	
		1	所选的端点被暂停。	
2	STP		SETUP 位：该位的值在每次成功地接收到数据包（即在所选物理端点上获得 ACK 应答的封包）之后更新。	0
		0	在所选端点上执行选择端点/清除中断命令时，STP 位清零。	
		1	所选端点上一次接收到的包为 SETUP 包。	
3	PO		包覆盖（over-written）位。	0
		0	PO 位由“选择端点/清除中断”命令来清零。	
		1	之前接收到的包被 SETUP 包覆盖。	
4	EPN		EP NAKed 位表示发送一个 NAK。如果主机向已满的 OUT 缓冲区发送一个 OUT 包，则设备返回 NAK。如果主机向空的 IN 缓冲区发送一个 IN 令牌包，则设备返回 NAK。	0
		0	当设备在接收到 OUT 包之后发送一个 ACK，或者当设备在发送 IN 包之后看到一个 ACK 时，EPN 位复位。	
		1	当设备发送一个 NAK 并且 NAK 特性的中断使能时，EPN 位置位。	
5	B_1_FULL		缓冲区 1 的状态。	0
		0	缓冲区 1 为空。	
		1	缓冲区 1 为满。	
6	B_2_FULL		缓冲区 2 的状态。	0
		0	缓冲区 2 为空。	
		1	缓冲区 2 为满。	
7	-		保留。读取值未定义，只写入 0。	无

13.12.11 选择端点/清除中断（命令：0x40—0x5F，数据：读 1 个字节）

命令 0x40 至 0x5F 对于各个选择端点来说是相同的，不同的地方在于：

- 它们会将 USBEplntSt 寄存器中对应于端点的位清零。
- 如果是控制 OUT 端点，则会将相应的选择端点寄存器中的 STP 和 PO 位清零。
- 必须读取 1 个字节。

注：通过使用 USBCmdCode 和 USBCmdData 寄存器，或通过 USBEplntClr 中的相应位置位可以调用该命令。为方便起见，建议使用 USBEplntClr 寄存器。

13.12.12 设置端点状态（命令：0x40—0x55，数据：写 1 个字节（可选））

“设置端点状态”命令用于设置端点的状态位 7:5 和 0。“设置端点状态”的命令代码等于 0x40 与物理端点编号（十六进制）之和。并非所有类型端点都可以设置所有的位。

表318. 设置端点状态命令位描述

位	符号	值	描述	复位值
0	ST		暂停的端点位。当一个暂停的端点接收到 SETUP 令牌时，不管接收到的 SETUP 包的内容如何，该端点都会自动退出暂停状态。如果端点应该停留在其暂停状态，则 CPU 可以通过将该位置位使得该端点再次暂停。当一个暂停的端点由于设置端点状态命令或接收到一个 SETUP 令牌而退出暂停状态时，该端点也会被重新初始化。这样可以将缓冲区清空：如果是 OUT 缓冲区，则它将等待 DATA 0 PID ；如果是 IN 缓冲区，则它会写 DATA 0 PID 。 端点的中断状态不会发生改变。若端点已经退出暂停状态，则向该位写入 0 将对端点进行初始化。若通过置位端点状态命令将端点暂停，则该端点也会被重新初始化。	0
		0	端点没有暂停。	
		1	端点被暂停。	
4:1	-		保留。读取值未定义，只写入 0 。	无
5	DA		禁能端点位。	0
		0	端点使能。	
		1	端点禁能。	
6	RF_MO		速率反馈（rate feedback）模式。	0
		0	中断端点位于翻转模式（Toggle mode）中。	
		1	中断端点位于速率反馈模式中。这意味着无需数据翻转位（data toggle bit）就可以进行传输。	
7	CND_ST		条件暂停位。	0
		0	两个控制端点都没有暂停。	
		1	将两个控制端点暂停，除非选择端点寄存器中的 STP 位是置位的。只是针对控制 OUT 端点而定义的。	

13.12.13 清空缓冲区（命令：0xF2，数据：读 1 个字节（可选））

当主机发送的 **OUT** 包被成功接收时，内部硬件 **FIFO** 状态 **Buffer_Full** 标志置位。通过返回一个 **NAK** 应答，所有后到的包均被拒绝。当设备软件读取数据时，应发出“清空缓冲区”命令将缓冲区清空。这也会将内部 **Buffer_Full** 标志清零。当缓冲区为空时，就可以接收新的数据包了。

当可选数据字节的位 **0** 为 **1** 时，前面接收的数据包被 **SETUP** 包覆盖(over-written)。Packet Over-written 位只用于控制传输。根据 **USB** 规范，无论缓冲区的状态如何，都应接受 **SETUP** 包。软件在读取 **SETUP** 数据后，应总是检查 **PO** 位的状态。如果该位置位，则应丢弃前面读取的数据，发出“选择端点/清除中断”命令将 **PO** 位清零，读取新的 **SETUP** 数据，并再次检查 **PO** 位。

关于本命令的使用描述，见 [13.14](#) 节。

表319. 清空缓冲区命令位描述

位	符号	值	描述	复位值
0	PO		包覆盖位。该位只适用于控制端点 EP0。	0
		0	之前接收到的数据包保持完好。	
		1	之前接收到的数据包被后面的 SETUP 包覆盖。	
7:1	-		保留。读取值未定义，只写入 0。	无

13.12.14 确认缓冲区（命令：0xFA，数据：无）

当 CPU 已将数据写入一 IN 缓冲区时，软件应发出一个“确认缓冲区”命令。这个命令告知硬件，缓冲区已准备好在 USB 总线上进行发送操作。当收到下一个 IN 令牌包时，硬件将发送缓冲区中的内容。

内部有一个硬件 FIFO 状态标志，叫做 Buffer_Full。这个标志由确认缓冲区命令置位且当数据已在 USB 总线上发送时清零，缓冲区为空。

当控制 IN 缓冲区对应的 OUT 缓冲区的 Packet Over-written (PO) 位置位（见“清空缓冲区寄存器”），或包含一个挂起的 SETUP 包时，则不能确认控制 IN 缓冲区有效。对于控制端点，当收到 SETUP 包时，已确认的缓冲区将无效。

本命令的使用描述见 [13.14](#) 节。

13.13 USB 设备控制器的初始化

LPC178x/177x USB 设备控制器的初始化包括下列步骤：

1. 通过置位 PCONP 的 PCUSB 位使能设备控制器。
2. 配置并使能 PLL 与时钟分频器，以提供 48MHz 的 usbclk 和所需的 cclk 频率。有关 PLL 设置与配置的步骤，见 [4.5.10](#) 节。
3. 通过置位 USBClkCtrl 寄存器中的 DEV_CLK_EN 位和 AHB_CLK_EN 位使能设备控制器的时钟。轮询 USBClkSt 寄存器中相应的时钟位，直到它们被置位。
4. 通过使用 USBPortSel 寄存器选择所需要的 USB 端口管脚。必须先置位 USBClkCtrl 中的 PORTSEL_CLK_EN 位，然后才能访问 USBPortSel，并应在访问 USBPortSel 后清零 PORTSEL_CLK_EN 位。
5. 写入相应的 IOCON 寄存器，使能 USB 管脚的功能。
6. 使用相应的 IOCON 寄存器，使管脚进入“纯输入”模式，禁用 V_{BUS} 管脚的上拉和下拉电阻。见 [8.4.1](#) 节。
7. 为 EP0 和 EP1 设置 USBEPIn 和 USBMaxPSize 寄存器，并等待 USBDevIntSt 中的 EP_RLZED 位置位，实现 EP0 和 EP1。
8. 使能端点中断（从模式）：
 - 使用 USBEPIntClr 将所有端点的中断清零。
 - 使用 USBDevIntClr 将所有设备的中断清零。

- 通过置位 USBEpIntEn 中的相应位，使能所需端点的从模式。
- 使用 USBEpIntPri 设置每个已使能中断的优先级。
- 使用 SIE 的“设置模式”命令，配置所需中断的模式。
- 使用 USBDevIntEn，使能设备中断（一般是 DEV_STAT 和 EP_SLOW，也可能是 EP_FAST）。

9. 配置 DMA（DMA 模式）：

- 使用 USBEpDMADis，禁用所有端点的 DMA 操作。
- 使用 USBDMARClr，清除所有挂起的 DMA 请求。
- 使用 USBEoTIntClr、USBNDDRIntClr 和 USBSysErrIntClr，清除所有 DMA 中断。
- 在系统存储器中准备 UDCA。
- 将所需的 UDCA 地址写入 USBUDCAH。
- 使用 USBEpDMAEn，使能所需端点的 DMA 操作。
- 置位 USBDMAIntEn 中的 EOT、DDR 和 ERR 位。

10. 在 NVIC 中安装 USB 中断处理器，方法是将对应的地址写入相应的向量表单元，并使能 NVIC 中的 USB 中断。

11. 使用 SIE “设置地址”命令，将默认 USB 地址设为 0x0，DEV_EN 设为 1。总线复位也会实现上述设置。

12. 使用 SIE “设置设备状态”命令，将 CON 位设为 1，使 CONNECT 有效。

端点的配置变化取决于软件应用。默认情况下，所有端点都是禁能的，除了控制端点 EP0 和 EP1 以外。当软件收到来自主机的 SET_CONFIGURATION 或 SET_INTERFACE 设备请求后，可以使能并配置更多的端点。

13.14 从模式操作

在从模式下，CPU 使用寄存器接口，在 RAM 与端点缓冲区之间传输数据。

13.14.1 中断的产生

在从模式下，RAM 与端点缓冲区之间的数据包传输可以在出现端点中断时启动来相应中断。端点中断的使能要采用 USBEpIntEn 寄存器，并可在 USBEpIntSt 寄存器中进行查询。

所有非同步 OUT 端点都会在成功接收到一个数据包时产生端点中断。所有非同步的 IN 端点在成功发送一个数据包时，或在总线上发送一个 NAK 握手信号且 NAK 中断特性使能时产生中断。

对于同步端点，当发生 FRAME 中断时（在 USBDevIntSt 中），做数据的传输。

13.14.2 OUT 端点的数据传输

当软件要从一个端点缓冲区读取数据时，应将 USBCtrl 寄存器中的 RD_EN 位置位，并将 LOG_ENDPOINT 字段设置为所需的端点编号。控制逻辑将获取数据包长度输入到 USBRxPLen 寄存器，并将 PKT_RDY 位置位（表 285）。

现在，软件可以开始从 USBRxData 寄存器读取数据（表 284）。当到达数据包的结尾时，RD_EN 位清零，并且 USBDevSt 寄存器中的 RxENDPKT 位置位。此时，软件发出“清空缓冲区”（见表 319）命令。端点准备好接收下一个数据包。对于 OUT 同步端点，无论缓冲区是否被清空，都会接收下一个包。在帧结束以前尚未从缓冲区读取的数据将丢失。详见 13.16 节。

如果软件在整个包被读取以前就清零 RD_EN，则读操作中止，数据保留在端点缓冲区中。当该端点 RD_EN 位再次置位时，数据将从开始处读起。

13.14.3 IN 端点的数据传输

当向端点缓冲区写入数据时，WR_EN 位（见 13.10.6.5 节）置位，软件将数据包中即将被发送的字节数写入 USBTxPLen 寄存器（13.10.6.4 节）。然后，它就可以持续地向 USBTxData 寄存器写入数据。

如果向 USBTxData 写入的数据量已经达到 USBTxPLen 寄存器中设置的字节数，WR_EN 位清零，并 USBDevIntSt 寄存器中的 TxENDPKT 位置位。软件发出一个“确认缓冲区”命令（13.12.14 节）。现在，端点已准备好发送数据包。对于 IN 同步端点，只有在下个 FRAME 中断发生以前确认了缓冲区后，缓冲区中的数据才会被发送；否则，下一个帧中将发送一个空包。如果软件在整个包写入前清零了 WR_EN，则当下次为此端点设置 WR_EN 时，会从头开始执行写操作。

对于同一个逻辑端点，RD_EN 与 WR_EN 可以同时为高电平。因此可以交错地执行读和写操作。

13.15 DMA 操作

在 DMA 模式中，DMA 在 RAM 与端点缓冲区之间传输数据。

以下章节描述了 DMA 模式的操作。背景信息见 [13.15.2](#) 节以及 [13.15.3](#) 节。DMA 描述符各字段描述见 [13.15.4](#) 节。最后三节将描述 DMA 操作：[13.15.5](#) 节、[13.15.6](#) 节，以及 [13.15.7](#) 节。

13.15.1 传输术语

在本节中，要提到 3 种传输类型：

1. **USB 传输：**通过 USB 总线传输数据。USB2.0 规格将其简称为传输。在本节中，它们被称为“USB 传输”，以区别于“DMA 传输”。USB 传输由多个事务构成。每个事务都包括多个数据包。
2. **DMA 传输：**端点缓冲区与系统存储器（RAM）之间的数据传输。
3. **数据包传输：**本节中，数据包传输是指一个数据包在一个端点缓冲区与系统存储器（RAM）之间的传输。DMA 传输包括一个或多个数据包传输。

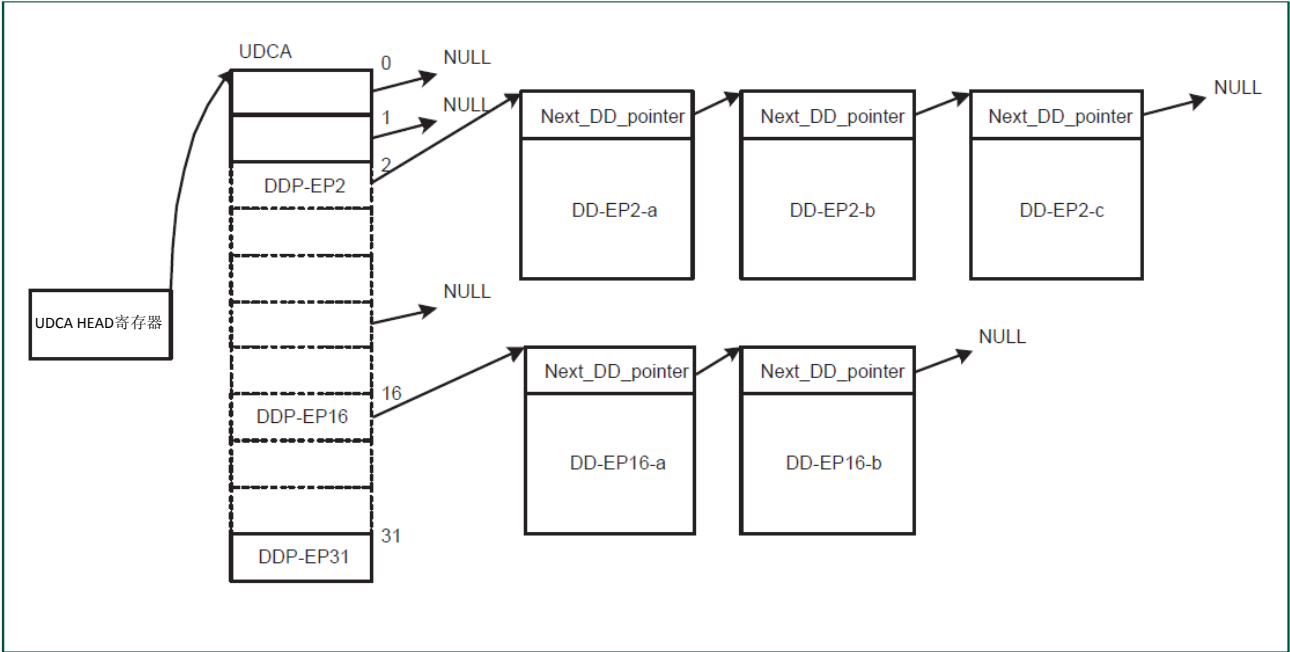
13.15.2 USB 设备通信区域

CPU 与 DMA 控制器之间的通信是通过一个公共存储区域，这个区域称作“USB 设备通信区域”，即 UDCA。UDCA 是一个 32 字的 DMA 描述符指针（DDP）数组，每个 DDP 都对应一个物理端点。如果针对端点定义了一个这样的数组，每个 DDP 都指向一个 DMA 描述符的起始地址。对于未实现的端点以及禁能 DMA 操作的端点，DDP 被忽略且可以设置为 NULL（0x0）。

UDCA 的起始地址保存在 USBUDCAH 寄存器中。UDCA 可以位于 RAM 的任何 128 字节边界，可通过 CPU 与 DMA 控制器访问。

[图 42](#) 显示了 UDCA 及其与 UDCA Head（USBUDCAH）寄存器与 DMA 描述符之间的关系。

图42. UDCA Head 寄存器和 DMA 描述符



13.15.3 触发 DMA 引擎

通过设置 USBEpIntEn 寄存器中的相应位为 0 来禁止从模式操作时 ([13.10.4.2](#) 节)，对应的端点提出 DMA 请求，并产生一个端点中断（见 [13.10.8.1](#) 节）。

当在 USBEpDMASt 中使能了端点的 DMA 操作时，端点的 DMA 传输开始，USBDMARSt 中的相应位置位，并为端点找到一个有效的 DD。

所有端点共享单一的 DMA 通道以减少硬件开销。如果 USBDMARSt 中有多个有效的 DMA 请求，则先处理物理端点编号最小的那个端点。

在 DMA 模式中，应使用 SIE 设置模式命令，将对应于批量 OUT 与中断 OUT 端点 NAK 中断的位（INAK_BO 和 INAK_IO）设置为 0 ([13.12.3](#) 节)。

13.15.4 DMA 描述符

对 DMA 传输的描述采用了一种名为 DMA 描述符（DD）的数据结构。

DD 存放在 RAM 中。这些描述符可以位于片上 RAM 中的任何字对齐的地址中。

对于非同步端点，DD 为 4 个字长。对于同步端点，DD 为 5 个字长。

与 DMA 传输相关的参数为：

- DMA 缓冲区的起始地址；
- DMA 缓冲区的长度；
- 下一个 DMA 描述符的起始地址；
- 控制信息；
- 计数信息（已传输的字节数）；
- 状态信息。

表 320 列出了 DMA 描述符的各个字段。

表320. DMA 描述符

字的位置	访问 (H/W)	访问 (S/W)	位的位置	描述
0	R	R/W	31:0	Next_DD_pointer
1	R	R/W	1:0	DMA_mode（00 为正常模式；01 为 ATLE 模式）
	R	R/W	2	Next_DD_valid（1 为有效；0 为无效）
	-	-	3	保留。读取值未定义，只写入 0。
	R	R/W	4	Isochronous_endpoint（1 为同步端点；0 为非同步端点）
	R	R/W	15:5	Max_packet_size
	R/W ^[1]	R/W	31:16	DMA_buffer_length 对于非同步端点，该值表示的是字节数；对于同步端点，该值表示的是包的个数。
2	R/W	R/W	31:0	DMA_buffer_start_addr
3	R/W	R/I	0	DD_retired（初始化为 0）
	W	R/I	4:1	DD_status（初始化为 0000）： 0000: NotServiced 0001: BeingServiced 0010: NormalCompletion 0011: DataUnderrun（短包） 1000: DataOverrun 1001: SystemError
	W	R/I	5	Packet_valid（初始化为 0）
	W	R/I	6	LS_byte_extracted（ATLE 模式）（初始化为 0）
W	R/I	7	MS_byte_extracted（ATLE 模式）（初始化为 0）	
R	W	13:8	Message_length_position（ATLE 模式）	
-	-	15:14	保留。读取值未定义，只写入 0。	
R/W	R/I	31:16	Present_DMA_count（初始化为 0）	
4	R/W	R/W	31:0	Isochronous_packetsize_memory_address

[1] 在 ATLE 模式中, 该字段的访问属性为只写。
表注: R-读; W-写; I-初始化。

13.15.4.1 Next_DD_pointer

指向下一个要提取的 DMA 描述符在存储器中的位置。

13.15.4.2 DMA_mode

设定 DMA 的操作模式。定义了两种模式: 正常模式和自动传输长度提取 (ATLE) 模式。在正常模式下, 软件为 OUT 端点初始化 DMA_buffer_length。在 ATLE 模式下, DMA_buffer_length 从输入的数据中提取。详见 13.15.7 节。

13.15.4.3 Next_DD_valid

此位表示软件是否已准备了下一个 DMA 描述符。如果该位置位, 则 DMA 引擎在完成当前描述符后提取下一个描述符。

13.15.4.4 Isochronous_endpoint

该位置位用于指示描述符属于一个同步端点。因此，在提取描述符时必须读取 5 个字。

13.15.4.5 Max_packet_size

该字段表示端点的最大数据包容量。这个参数用于 IN 端点从存储器传输数据时。对 OUT 端点，它用于检测短包。此字段仅适用于非同步端点。该字段的值应该与使用 USBMaxPSize 寄存器分配给端点的 MPS 值。

13.15.4.6 DMA_buffer_length

该字段表示被传输数据所占用的 DMA 缓冲区的深度。当达到这一限制时，DMA 引擎将停止使用这个描述符并寻找下一个描述符。

在正常模式下，软件会同时为 IN 端点和 OUT 端点设置此值。在 ATLE 模式下，软件仅为 IN 端点设置此值。对于 OUT 端点，硬件使用数据流中提取的长度设置此值。

对于同步端点，DMA_buffer_length 指的是包的数量，对于非同步端点，该字段的值用字节数表示。

13.15.4.7 DMA_buffer_start_addr

读取数据或写入数据的地址。每次 DMA 引擎完成一个数据包的传输时，都会更新这个字段。

13.15.4.8 DD_retired

当 DMA 引擎用完当前的描述符时，由硬件设置此位。当缓冲区达到末尾、传输一个短包（非同步端点），或检测到一个错误条件时，该位置位。

13.15.4.9 DD_status

DMA 传输的状态在这个字段中编码。定义了下列编码：

- **NotServiced:** 没有传输数据包。
- **BeingServiced:** 至少传输了一个数据包。
- **NormalCompletion:** DD 是在到达了缓冲区末尾且没有发生错误的情况下退出的。同时 DD_retired 位置位。
- **DataUnderrun:** 在到达 DMA 缓冲区末尾以前，USB 传输由于接收到一个短包而结束。同时 DD_retired 位置位。
- **DataOverrun:** 当数据包正在传输时，DMA 缓冲区到达末尾。这是一个错误情况。DD_retired 位置位。当前的 DMA 计数字段的值等于 DMA_buffer_length 的值。该数据包必须在下一次 DMA 传输时从端点缓冲区重新发送。USBEPDMASt 中相应的 EPxx_DMA_ENABLE 位清零。
- **SystemError:** 由于 AHB 总线上的错误，正在处理的 DMA 传输被终止。此时 DD_retired 位不置位。USBEPDMASt 中相应的 EPxx_DMA_ENABLE 位清零。由于在更新 DD 过程

中也可能发生系统错误，因此 RAM 中的 DD 字段并不可靠。

13.15.4.10 Packet_valid

此位用于同步端点。它表示上一次传输到存储器中的包是否是在无错误的情况下接收到的。如果数据包是有效的，则此位置位，即表示无错接收。有关同步端点的操作，见 [13.15.6](#) 节。

这个位对非同步端点没有必要，因为只有无错的包才会产生 DMA 请求，因此，当产生了请求时，Packet_valid 总是置位的。

13.15.4.11 LS_byte_extracted

用于 ATLE 模式。当置位时，此位表示已提取了传输长度中的最低有效字节（LSB）。提取的长度反映在 DMA_buffer_length 字段的 23:16 位。

13.15.4.12 MS_byte_extracted

用于 ATLE 模式。该位置位表示已提取了传输长度中的最高有效字节（MSB）。提取的长度反映在 DMA_buffer_length 字段的 31:24 位。当 LS_Byte_extracted 和 MS_byte_extracted 位设置时，提取操作停止。

13.15.4.13 Present_DMA_count

DMA 引擎传输的字节数。DMA 引擎在完成每个数据包的传输后更新此字段。

对于同步端点，Present_DMA_count 是已传输的数据包个数。对于非同步端点，Present_DMA_count 是字节数。

13.15.4.14 Message_length_position

用于 ATLE 模式。这个字段给出了嵌入到输入数据包中的消息长度位置的偏移量。它只适用于 OUT 端点。0 偏移表示消息长度从第一个包的第一字节开始。

13.15.4.15 Isochronous_packetsize_memory_address

该字段为即将进行传输或被提取的数据包容量信息以及帧编号所在的存储缓冲区的地址。见 [图 43](#)。它只适用于同步端点。

13.15.5 非同步端点操作

13.15.5.1 设置 DMA 传输

软件为那些即将使能的 DMA 传输的物理端点准备 DMA 描述符 (DD)。这些 DD 放在片上 RAM 中。第一个 DD 的起始地址写入 UDCA 中对应端点的 DMA 描述指针 (DDP) 单元。然后，软件将 USBEPDMAEn 寄存器中对应端点的 EPxx_DMA_ENABLE 位 ([13.10.8.6](#) 节) 置位。对于正常模式，描述符中的 DMA_mode 位设置为 “00”。所有其它 DD 字段均按照[表 320](#) 所示进行初始化。

物理端点 0 和 1 (默认的控制端点) 不支持 DMA 操作。

13.15.5.2 查找 DMA 描述符

当某端点的 DMA 传输被触发时，DMA 引擎将首先确定是否必须提取一个新的描述符。如果上一个传输的包是针对同一端点，且 DD 不是退出状态，则不必提取新的描述符。识别这种情况可以使用一个名为 DMA_PROCEED 的内部标志 (见 [13.15.5.4](#) 节)。

如果必须读一个新的描述符，则 DMA 引擎会计算该端点的 DDP 位置，并从此位置取出 DD 的起始地址。DD 起始地址如果在位置 0，则该地址被认为是无效的。这种情况下，会产生 NDDR 中断。所有字对齐的其它地址都是有效的。

在提取 DD 时，先要读取 DD 状态字 (字 3)，并检查 DD_retired 位的状态。如果 DD_retired 未置位，则 DDP 指向一个有效 DD。如果 DD_retired 已置位，则 DMA 引擎会读取 DD 的控制字 (字 1)。

如果 Next_DD_valid 位置位，则 DMA 引擎将取出 DD 的 Next_DD_pointer 字段 (字 0)，并将其加载到 DDP。新的 DDP 被写入 UDCA 区域。

然后，从 DDP 中的地址处提取全部 DD (4 个字)。DD 将给出要完成 DMA 传输的细节。DMA 引擎将与从 DD 提取的信息 (起始地址，DMA 计数等) 一起加载到它的硬件资源中。

如果 Next_DD_valid 未置位，而 DD_retired 位置位，则 DMA 引擎会唤起该端点的 NDDR 中断，并清零相应的 EPxx_DMA_ENABLE 位。

13.15.5.3 传输数据

对于 OUT 端点，DMA 引擎从 EP_RAM 中读出当前包，并传输到从 DMA_buffer_start_addr 开始的片上 RAM 存储单元中。对于 IN 端点，数据从 DMA_buffer_start_addr 开始的片上 RAM 中提取并写入 EP_RAM 中。DMA_buffer_start_addr 和 Present_DMA_count 字段在每个包的传输后更新。

13.15.5.4 优化的描述符读取操作

DMA 传输通常是多包传输。除非端点有变化，否则硬件将不会从存储器中读取新的 DD。为了指示多包传输正在进行，硬件会将置位 DMA_PROCEED 内部标志。

当 DMA_buffer_length 字段中规定的字节数被传输完成后, DMA_PROCEED 标志被清零。当软件对 USBEPDMADis 寄存器执行写操作时, 该标志也要清零。由于软件具有清除 DMA_PROCEED 标志的能力, 因此软件能强制为下一个包的传输重新获取 DD。向 USBEPDMADis 寄存器中写入全 0 可清零 DMA_PROCEED 标志, 且不会禁用任何端点的 DMA 操作。

13.15.5.5 结束数据包传输

在完成数据包传输时, DMA 引擎把更新的状态信息的 DD 写回到读操作时的相同存储单元。DD 中的 DMA_buffer_start_addr、Present_DMA_count 与 DD_status 字段被更新。

DD 在退出时有以下几种类型:

正常结束: 如果当前包已全部传输, 且 Present_DMA_count 字段等于 DMA_buffer_length, 则 DD 正常结束。DD 将被写回到存储器, DD_retired 和 DD_status 设置为 NormalCompletion。该端点产生 EOT 中断。

USB 传输结束: 如果当前包已全部传输, 而包的容量小于 Max_packet_size 字段值, 并且仍未达到 DMA 缓冲区的末尾, 则是 USB 传输结束。DD 将被写回到存储器, DD_retired 和 DD_Status 设置为 DataUnderrun。该端点产生 EOT 中断。

错误结束: 如果当前包没有完全传输, 即在数据包传输过程中达到 DMA 缓冲区的末尾, 则发生错误情况。DD 被写回, DD_retired 与 DD_status 设置为 DataOverrun 状态代码。该端点生成 EOT 中断, 并且 USBEPDMASt 寄存器中的相应位被清零。当使用 USBEPDMAEn 寄存器再次将相应 EPxx_DMA_ENABLE 位置位时, 没有完全传输的数据包将从端点缓冲区中重新向存储器发送。

13.15.5.6 No_Packet DD

对于 IN 传输, 如果系统暂时没有可发送的数据, 它通过设置 No_Packet DD, 系统可以响应 NDDR 中断。方法是将 DD 中的 Max_packet_size 和 DMA_buffer_length 字段同时设为 0。在处理一个 No_Packet DD 时, DMA 引擎在没有传输数据包的情况下将 USBDMARSt 中与端点对应的 DMA 请求清零。DD 退出, 此时的状态代码为 NormalCompletion。这个步骤可以根据需要而重复。设备将在 USB 总线上以 NAK 响应 IN 令牌包, 直到带有一个数据包 DD 被设置, 且 DMA 将数据包传输到端点缓冲区。

13.15.6 同步端点操作

对于同步端点, 每个包的长度是可以变化的。对于每个帧来说, 每个同步端点有一个数据包。

13.15.6.1 设置 DMA 传输

软件将 DD 中的同步端点位设置为 1, 并设置 Isochronous_packetsize_memory_address 字段的初始值。所有其它字段的初始化均与非同步端点相同。

对于同步端点，字段 `DMA_buffer_length` 和 `Present_DMA_count` 的值用帧数目而不是字节数表示。

13.15.6.2 查找 DMA 描述符

查找描述符的方法与查找非同步端点的方法相同。

DMA 使能的同步端点可以在每个 FRAME 中断时产生 DMA 请求。在处理 DMA 请求时，DMA 引擎将读取描述符，如果 `Isochronous_endpoint` 位置位，则 DMA 引擎将从 DD 的第 5 个字中读取 `Isochronous_packetsize_memory_address`。

13.15.6.3 传输数据

数据在存储单元 `DMA_buffer_start_addr` 中进行传输。在包传输结束时，`Present_DMA_count` 的值加 1。

同步包的容量保存在存储器中，如图 43 所示。数据包容量存储器中的每个字都划分为多个字段：`Frame_number`（31 至 17 位），`Packet_valid`（16 位），以及 `Packet_length`（15 至 0 位）。对于某个给定 DD，分配给数据包容量的存储器的空间应是 `DMA_buffer_length` 个字，每个传输包用一个字。

OUT 端点

在每个帧信息传输结束时，数据包容量被写入 `Isochronous_packet_size_memory_address` 中的地址单元，并且 `Isochronous_packet_size_memory_address` 的值加 4。

IN 端点

在同步数据包容量中，只需要使用 `Packet_length` 字段。对于每个帧，USB 设备将传输一个端点数据包传输到主机，该数据包的容量在 `Packet_length` 字段中指定，在包传输结束时，将 `Isochronous_packet_size_memory_address` 的值加 4。如果 `Packet_length` 为 0，则 USB 设备将发一个空包。

13.15.6.4 DMA 描述符结束

用于同步端点的 DD 只能以 `NormalCompletion` 状态代码结束，因为同步端点没有短包，而 USB 传输会无休止地持续下去，直到发生一个系统错误。所以不会存在同步端点的 `DataOverrun` 检测。

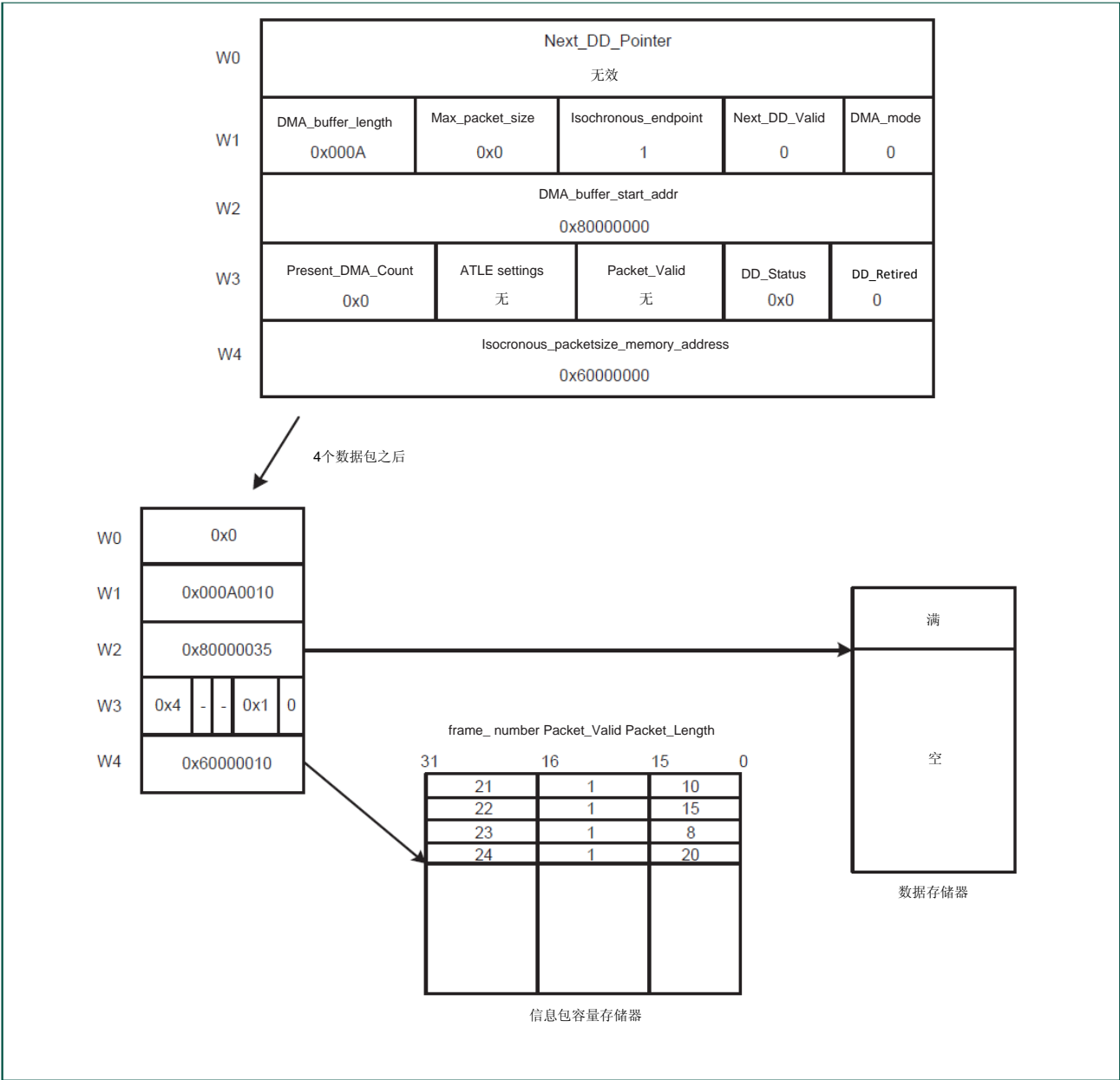
13.15.6.5 同步 OUT 端点操作举例

假设将同步端点设置为传输 10 帧，当帧编号为 21 时传输开始。在传输了 4 个数据包长度分别为 10、15、8 和 20 字节的帧信息之后，描述符与存储器映射如图 43 所示。

$\text{The_total_number_of_bytes_transferred} = 0x0A + 0x0F + 0x08 + 0x14 = 0x35$ 。

数据包长度存储器中所有字的 `Packet_valid` 位（位 16）均设置为 1。

图43. 同步 OUT 端点操作举例



13.15.7 自动长度传输提取（ATLE）模式操作

有些主机驱动程序能够将小容量的 USB 传输(delta 传输)连接起来, 形成一个大容量的 USB 传输, 这些驱动程序如网络驱动程序接口规范 (NDIS) 主机驱动程序。对于 OUT USB 传输, 设备硬件必须将这个已连接的传输分解开, 返回到原来的 delta 传输, 然后将它们传输到独立的 DMA 缓冲区。这个操作的实现方式是在 DMA 描述符中, 将 DMA 模式设置为自动传输长度提取 (ATLE) 模式。只有批量端点支持 ATLE 模式。

ATLE 模式中的 OUT 传输

图44. ATLE 模式中的数据传输

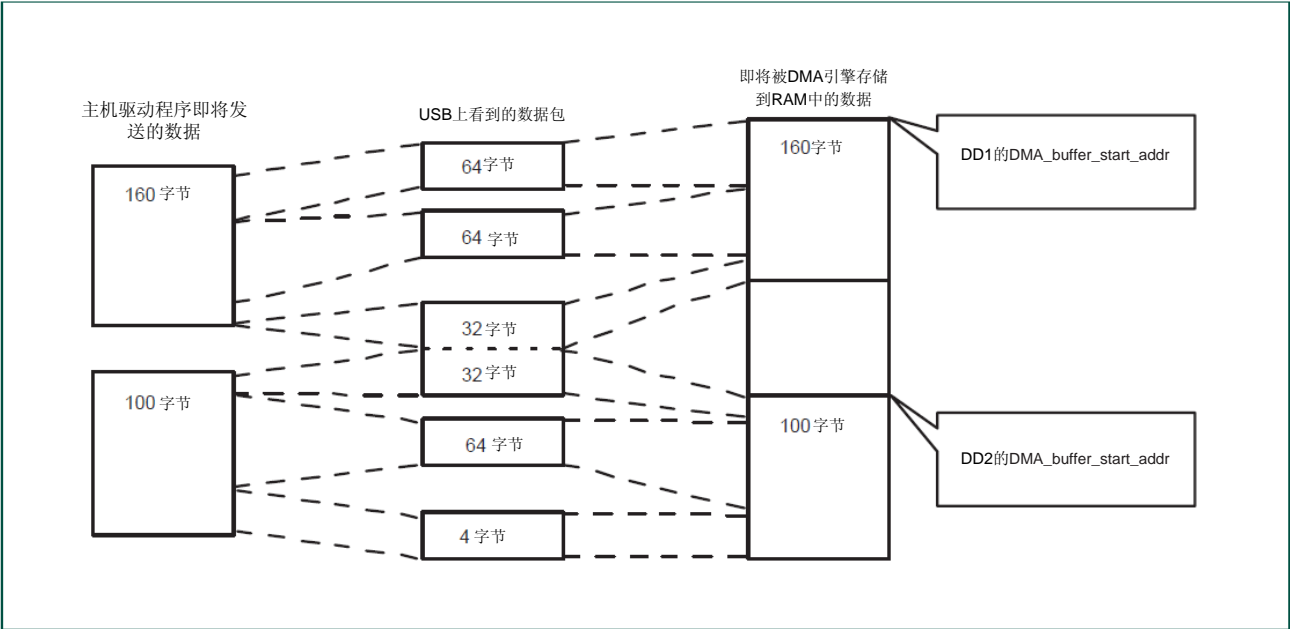


图 44 显示了 ATLE 模式下的一个典型 OUT USB 传输, 其中主机连接了两个分别为 160 字节和 100 字节的 USB 传输。由于 MaxPacketSize 为 64, 因此设备硬件将这个 USB 传输解释为 4 个 64 字节的数据包和一个 4 字节的短包。第 3、4 个数据包被连接起来。注意, 在正常模式下, USB 传输会被解释为 64、64、32、64 和 36 字节。

现在, DMA 引擎负责将两个 USB 传输分开, 将它们放在 DMA 描述符 1 (DD1) 和 DMA 描述符 2 (DD2) 中由 DMA_buffer_start_addr 字段所指定的存储单元中。

硬件在 Message_length_position 所规定的偏移量 (从 USB 传输的起始字节开始), 读出两字节宽的 DMA_buffer_length, 并将其写入 DD 的 DMA_buffer_length 字段。如果要提取的 DMA_buffer_length 两个字节分在两个包中, 为确保正确的提取, 硬件在提取了各个字节后, 将 LS_byte_extracted 和 MS_byte_extracted 标志置位。提取了 MS 字节后, DMA 继续进行, 与在正常模式中相同。

当准备一个新 DD 时, 软件将 LS_byte_extracted 和 MS_byte_extracted 标志清零。因此, 如果 DD 退出, 则硬件必须为下一个 DD 再次提取传输长度。

如果在一个连接包（如图 44 中的第 3 个包）的传输过程中 DD1 退出，并且没有设置 DD2（DD1 的 Next_DD_valid 字段为 0），则 DD1 退出时的 DD_status 状态代码设置为 DataOverrun。这将被看作是一个错误情况，USBepDMASt 的相应 EPxx_DMA_ENABLE 位被硬件清零。

在 ATLE 模式下，即将传输的最后缓冲区长度总是以一个短包或空包结束，用以表示 USB 传输的结束。如果相连的传输长度以 MaxPacketSize 包边界结束，则（NDIS）主机将发送一个空包来标记 USB 传输的结束。

ATLE 模式中的 IN 传输

对于从设备到主机的 IN USB 传输，DMA_buffer_length 由设备软件设置，与正常模式相同。

在 ATLE 模式中，设备将来自多个 DD 的数据连接起来，构成一个 USB 传输。如果 DD 在一个数据包（包容量小于 MaxPacketSize）的传输过程中退出，则提取由 Next_DD_pointer 所指的下一个 DD，MaxPacketSize 包中剩余字节将从下一个 DD 的缓冲区中传输。

如果没有设置下一个 DD（即 DD 中的 Next_DD_valid 字段为 0），且当前 DD 的 DMA 缓冲区长度在到达 MaxPacketSize 包边界以前已经结束，则当前 DD 的可用字节被作为一个短包在 USB 上发送，以标志主机 USB 传输的结束。

如果最后缓冲区长度以 MaxPacketSize 的边界结束，设备软件必须将下一个 DD 的 DMA_buffer_length 字段设置为 0，这样设备发出一个空包，来标志主机的 USB 传输的结束。

13.15.7.1 设置 DMA 传输

对于 OUT 端点，主机硬件需要设置 DD 中的 Message_length_position 字段。它表示消息长度在输入数据包中的起始位置。另外，设备软件还必须将 OUT 端口的 DMA_buffer_length 字段设为 0，因为设备硬件会在提取缓冲区长度以后更新这个字段。

对于 IN 端点，描述符的设置与正常模式相同。

由于一个包可以被划分到两个 DD，软件应总是保持两个可用 DD，用一个短包或空包结束的最后 DMA 传输除外。

13.15.7.2 查找 DMA 描述符

DMA 描述符的查找方式与正常模式相同。

13.15.7.3 传输数据

OUT 端点

如果状态字段中的 LS_byte_extracted 或 MS_byte_extracted 位都未设置，则硬件将从数据流中提取传输长度，并设置 DMA_buffer_length。一旦提取完成，则 LS_byte_extracted 和 MS_byte_extracted 位都将置位。

IN 端点

IN 端点的 DMA 传输处理与正常模式下的处理相同，并持续到传输字节数等于 DMA_buffer_length 字段的值为止。

13.15.7.4 结束包传输

DMA 引擎持续与片上 RAM 进行传输，直到传输的字节数已达到 DMA_buffer_length 字段中指定的字节数。然后将产生 EOT 中断。如果它发生在数据包的中间，则加载相连的 DD，且该包中剩余部分从新 DD 指向的地址进行传输。

OUT 端点

如果链接的 DD 无效，并且包已部分传输给存储器，则 DD 以 DataOverrun 状态代码结束，且该端点的 DMA 将禁能。否则，DD_status 将更新为 NormalCompletion 状态代码。

IN 端点

如果链接的 DD 无效，并且包已部分传输给 USB，则 DD 以 DD_status 字段中设置 NormalCompletion 状态代码结束。这种情况与 USB 传输结束一致，数据包将作为短包发送。另外，当链接的 DD 有效，且缓冲区长度为 0 时，将发送一个空包表示 USB 传输结束。

13.16 双缓冲的端点操作

USB 设备控制器的批量端点与同步端点采用双缓冲，以增加数据吞吐量。

当实现双缓冲端点时，设备将在 EP_RAM 中自动分配两个端点缓冲区的空间。见 [13.10.5.1](#) 节。

在以下讨论中，当前可由 CPU 或 DMA 引擎访问读写的端点缓冲区被称为有效缓冲区。

13.16.1 批量端点

对于批量端点，用 SIE “清空缓冲区”或“确认缓冲区”命令切换有效的端点缓冲区。

以下表示在从模式下，批量 OUT 端点的双缓冲是如何工作的。

假设缓冲区 1 (B_1) 和缓冲区 2 (B_2) 均为空，有效缓冲区为 B_1。

1. 主机向端点发送一个数据包。设备硬件将包放入 B_1，产生一个端点中断。
2. 软件清除端点中断，开始从 B_1 中读取包的数据。当 B_1 仍在被读取时，主机发送第 2 个包，设备硬件将其放入 B_2，并产生一个端点中断。
3. 软件仍在读 B_1，此时主机尝试发送第 3 个包。由于 B_1 和 B_2 均满，设备硬件响应一个 NAK。
4. 软件从 B_1 完成了第 1 个包的读取操作，发送一个 SIE “清空缓冲区”命令，释放 B_1 以接收其它包。B_2 成为有效缓冲区。
5. 软件发出 SIE “选择端点”命令来读取“选择端点寄存器”，并测试 FE 位。软件发现

有效缓冲区 (B_2) 中已经有数据 (FE=1)。软件清除端点中断，并开始读 B_2 中的内容。

6. 主机重新发送第 3 个包，设备硬件将该包放入 B_1。产生一个端点中断。
7. 软件从 B_2 完成了第 2 个包的读取操作，发送一个 SIE “清空缓冲区” 命令，释放 B_2 以接收其它包。B_1 成为有效缓冲区。软件等待下一个端点中断（该中断在第 6 步时已经产生）。
8. 软件将端点中断清零作为对它的相应，并开始从 B_1 中读第 3 个包。
9. 软件从 B_1 完成了第 3 个包的读取操作，发送一个 SIE “清空缓冲区” 命令，清空 B_1 以接收其它包。B_2 成为有效缓冲区。
10. 软件测试 FE 位，发现有效缓冲区 (B_2) 为空 (FE=0)。

B_1 和 B_2 均为空。软件等待下一个端点中断的发生。现在有效缓冲区是 B_2。主机发送的下一个数据包将放入 B_2。

以下表示在从模式下，批量 IN 端点的双缓冲是如何工作的。

假设缓冲区 1 (B_1) 和缓冲区 2 (B_2) 均为空，有效缓冲区为 B_1。NAK 中断特性被使能。

1. 主机发送 IN 令牌包来请求一个数据包。设备用 NAK 作响应并产生一个端点中断。
2. 软件清除端点中断。设备有 3 个待发包。软件将第 1 个包填入 B_1，并发送一个 SIE “确认缓冲区” 命令。有效缓冲区切换为 B_2。
3. 软件发送 SIE “选择端点” 命令来读取选择端点寄存器并测试 FE 位。它发现 B_2 为空 (FE=0)，用第 2 个包填充 B_2。软件发送一个 SIE “确认缓冲区” 命令，有效缓冲区切换为 B_1。
4. 软件等待端点中断的发生。
5. 设备成功地发送了 B_1 中的包，并清空缓冲区。发生一个端点中断。
6. 软件清除端点中断。软件用第 3 个包填充 B_1，并用 SIE “确认缓冲区” 命令确认 B_1。有效缓冲区切换为 B_2。
7. 设备成功地从 B_2 发送了第 2 个包，并生成一个端点中断。
8. 软件没有更多要发送的包，因此只清除中断。
9. 设备成功地从 B_1 发送了第 3 个包，并生成一个端点中断。
10. 软件没有更多要发送的包，因此只清除中断。
11. B_1 与 B_2 均为空，有效缓冲区是 B_2。下一个包将被软件写入 B_2。

在 DMA 模式下，有效缓冲区的切换由硬件自动处理。对于批量 IN 端点，通过使用 USBDMARSet 寄存器手动地开始一个包的传输，可以提前对一个端点缓冲区进行填充，从而充分利用双缓冲的功能。

13.16.2 同步端点

对于同步端点，当发生 FRAME 中断时，由硬件切换有效数据缓冲区。SIE “清空缓冲区” 与 “确认缓冲区” 命令都不会引起有效缓冲区的切换。

当正在总线上发送或接受另一个缓冲区中的数据包时，双缓冲使能软件充分利用帧的间隔，将一个包写入有效缓冲区或从有效缓冲区中读取一个包。

对于 OUT 同步端点，当缓冲区切换时，在帧结束前尚未从有效缓冲区中读出的任何数据都将丢失。

对于一个 IN 同步端点，如果在帧结束前，未确认有效缓冲区，则当有效缓冲区切换时，总线上会发送一个空包，当该缓冲区再次变为有效时，其内容将被覆盖。

14.1 如何阅读本章

本章描述了某些 LPC178x/177x 器件上的 USB 主机控制器（详见 [1.4](#) 节）。在这些器件上，可以将 USB 控制器配置为设备、主机或 OTG 等运行方式。

14.2 基本配置

使用下列寄存器配置 USB 控制器：

1. 功率：在 PCONP 寄存器中（[表 37](#)），设置 PCUSB 位。
注：复位后，USB 模块是禁用的（PCUSB = 0）。
2. 时钟：USB 模块可以使用副 PLL（PLL1）获得 USB 时钟，也可以用主 PLL（PLL0）。见 [4.5.2](#) 节。
3. 管脚：在相关 IOCON 寄存器中选择 USB 管脚及其模式（[8.4.1](#) 节）。
4. 唤醒：USB 总线端口上的活动可以将微控制器从掉电模式中唤醒，见 [4.7.9](#) 节。
5. 中断：使用相关的中断设置使能寄存器，可以在 NVIC 中使能中断。
6. 初始化：见 [15.11](#) 节。

14.3 简介

本节描述了 USB 2.0 OTG 双角色（dual role）内核的主机部分，它集成了主机控制器（符合 OHCI）、设备控制器，以及 I²C 接口。I²C 接口用于控制外部 OTG ATX。

USB 是一种 4 线总线，支持一个主机与多个（最多 127 个）外设之间的通信。主机控制器通过一种基于令牌的协议，为连接的设备分配 USB 带宽。USB 总线支持设备的热插拔以及动态配置。所有事务均由主机控制器发起。

主机控制器允许与连接到总线上的 USB 设备的数据交换。它包括寄存器接口、串行接口引擎，以及 DMA 控制器。寄存器接口符合 OHCI 规范。

表321. 本章用到的与 USB（OHCI）相关的首字母缩写词/简写

首字母缩写词/简写	描述
AHB	先进的高性能总线
ATX	模拟收发器
DMA	直接存储器访问
FS	全速
LS	低速
OHCI	开放式主机控制器接口
USB	通用串行总线

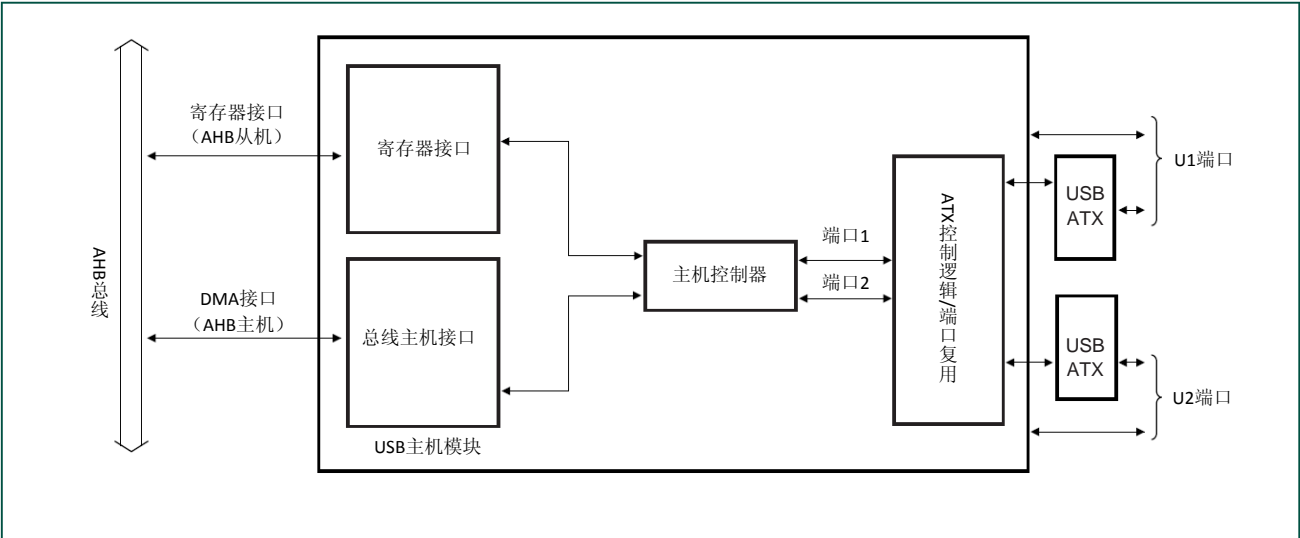
14.3.1 特性

- 遵循 OHCI 规范。
- OpenHCI 规定了 USB 主机控制器及 SW 驱动的操作和接口。
 - USBOperational：处理列表并生成 SOF 令牌。
 - USBReset：强制总线上的复位信号，禁用 SOF。
 - USBSuspend：监控 USB 上的唤醒活动。
 - USBResume：强制总线上的恢复信号。
- 主机控制器有 4 个 SW 驱动程序可见的 USB 状态。
- HCCA 寄存器指向中断以及同步描述符列表。
- ControlHeadED 与 BulkHeadED 寄存器指向控制与批量描述符列表。

14.3.2 结构

USB 主机控制器的结构如图 45 所示。

图45. USB 主机控制器方框图



14.4 接口

OTG 控制器有 2 个 USB 端口,用 USB 管脚名的后缀 1、2 表示,以下称为 USB 端口 1(U1)和 USB 端口 2 (U2)。

14.4.1 管脚描述

表322. USB OTG 端口管脚

管脚名称	方向	描述	类型
V _{BUS}	I	V _{BUS} 状态输入。当该功能未通过相应 IOCON 寄存器使能时,由内部高电平驱动。	USB 连接器
端口 U1			
USB_D+1	I/O	正向差分数据	USB 连接器
USB_D-1	I/O	反向差分数据	USB 连接器
USB_CONNECT1	O	SoftConnect 控制信号	控制
USB_UP_LED1	O	GoodLink LED 控制信号	控制
USB_INT1	I	OTG ATX 中断	外部 OTG 收发器
USB_SCL1	I/O	I ² C 串行时钟	外部 OTG 收发器
USB_SDA1	I/O	I ² C 串行数据	外部 OTG 收发器
USB_TX_E1	O	发送使能	外部 OTG 收发器
USB_TX_DP1	O	D+发送数据	外部 OTG 收发器
USB_TX_DM1	O	D-发送数据	外部 OTG 收发器
USB_RCV1	I	差分接收数据	外部 OTG 收发器
USB_RX_DP1	I	D+接收数据	外部 OTG 收发器
USB_RX_DM1	I	D-接收数据	外部 OTG 收发器
USB_LS1	O	低速状态 (仅应用于主机功能)	外部 OTG 收发器
USB_SSPND1	O	总线挂起状态	外部 OTG 收发器
USB_PPWR1	O	端口电源使能	主机电源切换
USB_PWRD1	I	端口电源状态	主机电源切换
USB_OVRCR1	I	过流状态	主机电源切换
USB_HSTEN1	O	主机使能状态	
端口 U2			
USB_D+2	I/O	正向差分数据	USB 连接器
USB_D2	I/O	反向差分数据	USB 连接器
USB_CONNECT2	O	SoftConnect 控制信号	控制
USB_UP_LED2	O	GoodLink LED 控制信号	控制
USB_PPWR2	O	端口电源使能	主机电源切换
USB_PWRD2	I	端口电源状态	主机电源切换
USB_OVRCR2	I	过流状态	主机电源切换
USB_HSTEN2	O	主机使能状态	控制

14.4.1.1 USB 主机使用注意事项

两个端口均可配置为 USB 主机。对于如何连接 USB 端口的细节，见 USB OTG 一章的 15.7 节。

USB 设备/主机/OTG 控制器在复位后都被禁能，必须在 PCONP 寄存器的 PCUSB 位中写入 1 使能，见表 37。

14.4.2 软件接口

USB 主机模块的软件接口包括一个寄存器视图，以及端点描述符的格式定义。关于这两方面的详细情况，见 OHCI 规范。下一节为寄存器映射。

14.4.2.1 寄存器映射

以下寄存器位于 AHB 时钟的“cclk”域。它们可以被处理器直接访问。所有寄存器均为 32 位宽，以字地址边界排列。

表323. USB 主机寄存器地址定义

名称	功能	地址	R/W ^[1]	复位值
HcRevision	通过主机控制器实现的 HCI 规范版本的 BCD 表示法。	0x2008 C000	R	0x10
HcControl	定义 HC 的工作模式。	0x2008 C004	R/W	0
HcCommandStatus	该寄存器用于接收主机控制器驱动器（HCD）的命令。它也可以表示 HC 的状态。	0x2008 C008	R/W	0
HcInterruptStatus	表示不同事件的状态，这些事件通过设置相应的位来产生硬件中断。	0x2008 C00C	R/W	0
HcInterruptEnable	控制 HcInterruptStatus 寄存器的位并指示哪个事件将产生硬件中断。	0x2008 C010	R/W	0
HcInterruptDisable	该位用来禁能 HcInterruptStatus 寄存器中对应的位并反过来禁能产生硬件中断的事件。	0x2008 C014	R/W	0
HcHCCA	包含主机控制器通信区域的物理地址。	0x2008 C018	R/W	0
HcPeriodCurrentED	包含当前同步或中断端点描述符的物理地址。	0x2008 C01C	R	0
HcControlHeadED	包含控制列表的第一个端点描述符的物理位置。	0x2008 C020	R/W	0
HcControlCurrentED	包含控制列表的当前端点描述符的物理位置。	0x2008 C024	R/W	0
HcBulkHeadED	包含批量列表的第一个端点描述符的物理位置。	0x2008 C028	R/W	0
HcBulkCurrentED	包含批量列表的当前端点描述符的物理位置。	0x2008 C02C	R/W	0
HcDoneHead	包含添加到“Done”队列的最后一个传输描述符的物理位置。	0x2008 C030	R	0
HcFmInterval	定义在一帧中的位时间间隔，以及不会产生过载的全速最大包容量。	0x2008 C034	R/W	0x2EDF
HcFmRemaining	14 位的计数器，表示在当前帧中剩余的位时间。	0x2008 C038	R	0
HcFmNumber	包含 16 位的计数器，并在 HC 和 HCD 出现的事件中提供计时参考。	0x2008 C03C	R	0

名称	功能	地址	R/W ^[1]	复位值
HcPeriodicStart	包含可编程的 14 位值，该值确定 HC 开始处理周期列表的最早时间。	0x2008 C040	R/W	0
HcLSThreshold	包含 11 位值，HC 通过该值来确定在 EOF 之前是否执行最大的 8 字节 LS 包的传输。	0x2008 C044	R/W	0x628h
HcRhDescriptorA	描述根集线器特性的第一个寄存器。	0x2008 C048	R/W	0xFF000902
HcRhDescriptorB	描述根集线器特性的第二个寄存器。	0x2008 C04C	R/W	0x60000h
HcRhStatus	该寄存器划分为两部分。低 D 字表示集线器状态字段，高字表示集线器状态的变化字段。	0x2008 C050	R/W	0x6000
HcRhPortStatus[1]	控制并报导每个端口上的端口事件。	0x2008 C054	R/W	0
HcRhPortStatus[2]	控制并报导每个端口上的端口事件。	0x2008 C058	R/W	0
Module_ID/ Ver_Rev_ID	IP 编号，yy（0x00）是唯一的版本号，zz（0x00）是唯一的版本号。	0x2008 C0FC	R	0x3505yyzz

- [1] [表 323](#) 中的 R/W 栏列出了寄存器的可访问性。
- (a) 访问标记“R”的寄存器将在读取时返回它们的当前值。
 - (b) 标记“R/W”的寄存器允许进行读写操作。

14.4.2.2 USB 主机寄存器定义

关于寄存器定义的内容，参见 Compaq 公司网站上的 OHCI 规范文档。

15.1 如何阅读本章

本章描述了某些 LPC178x/177x 器件上的 USB OTG 控制器(详见 [1.4](#) 节)。在这些器件上，USB 控制器可以配置为设备、主机或 OTG 运行。

15.2 基本配置

USB 控制器使用下列寄存器配置：

1. 功率：在 PCONP 寄存器中 ([表 37](#))，置位 PCUSB。
注：复位后，USB 模块被禁用 (PCUSB = 0)。
2. 时钟：USB 时钟可以用副 PLL (PLL1) 生成，也可以用主 PLL (PLL0)。见 [4.5.2](#) 节。
3. 管脚：在相关 IOCON 寄存器中，选择 USB 管脚及其模式 ([8.4.1](#) 节)。
4. 唤醒：USB 总线端口上的活动可以将微控制器从掉电模式中唤醒(见 [15.10.2](#) 节与 [4.7.9](#) 节)。
5. 中断：使用相应的中断设置使能寄存器，在 NVIC 中使能中断。
6. 初始化：见 [15.11](#) 节。

15.3 简介

本节描述了 USB 2.0 OTG 双角色 (dual role) 设备控制器的 OTG 与 I²C 部分，它集成了 (OHCI) 主机控制器、设备控制器，以及 I²C。I²C 接口是 USB 模块的一部分，用于控制外部 OTG 收发器，不同于 [22.1](#) 节中描述的 I²C 外设。

USB OTG (On-The-Go) 是 USB 2.0 规范的一个增补，它为 USB 外设的连接增加了主机功能，从而扩充了现有移动设备及 USB 外设的能力。有关 USB OTG 规范及更多信息见 USB 应用者论坛 (USB Implementers Forum) 网站。

15.4 特性

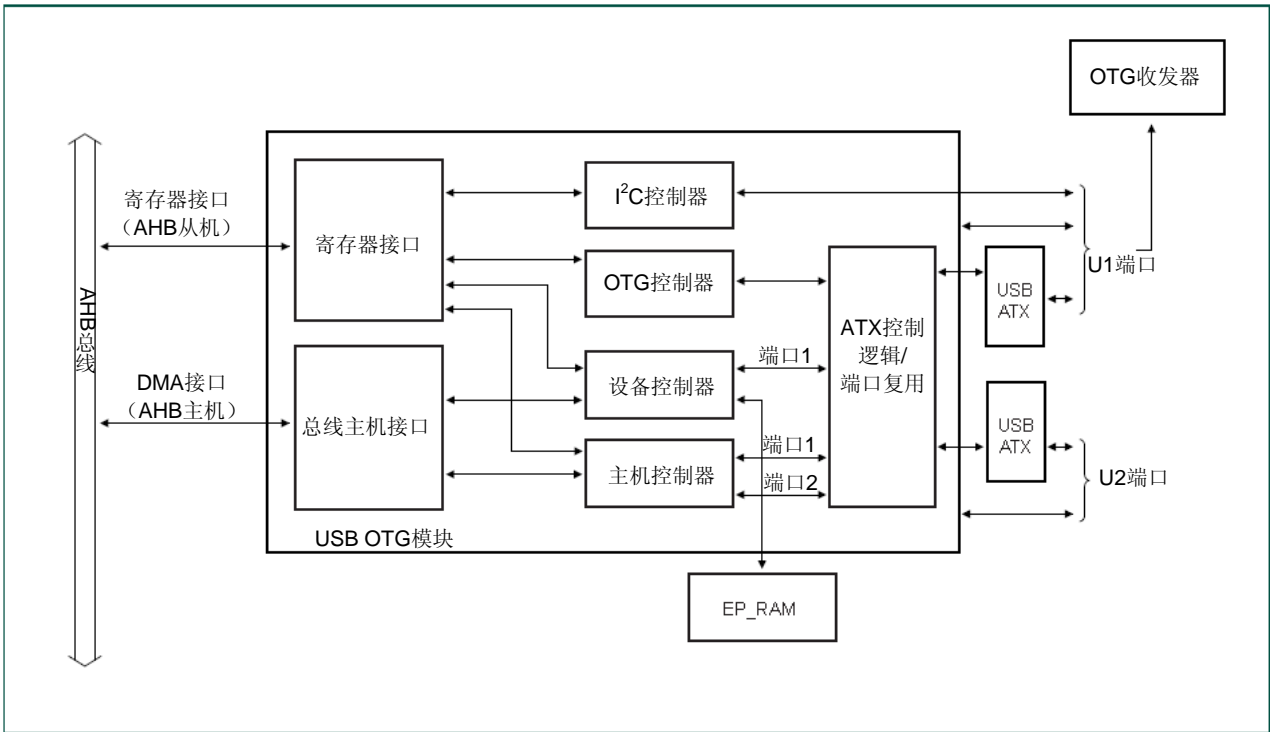
- 完全符合 USB 2.0 补充规范：USB OTG Rev. 1.0a。
- 硬件支持主机协商协议 (HNP)。
- 包含 HNP 和 SRP 要求的可编程定时器。
- 支持任何符合“OTG 收发器规范 (CEA-2011)，Rev. 1.0”的 OTG 收发器。

15.5 结构

USB OTG 控制器的结构如下面框图所示。主机、设备、OTG 与 I²C 控制器可通过寄存器接口进行编程。OTG 控制器能够通过 HNP 协议使能主机与设备功能角色的动态转换。一个端口可以连接到一个外部 OTG 收发器上,以支持一个 OTG 连接。寄存器接口与外部 OTG 收发器之间的通信可通过 I²C 接口以及外部 OTG 收发器的中断信号处理。

对于仅使用设备或主机控制器（非 OTG）的 USB 连接，端口使用嵌入的 USB 模拟收发器（ATX）。

图46. USB OTG 控制器方框图



15.6 操作模式

OTG 控制器能够在下列模式中操作：

- 一个双角色 OTG 端口，并可选其它主机端口（见[图 47](#)和[图 48](#)）
- 两个主机端口（见[图 49](#)）
- 一个主机端口和一个设备端口（见[图 50](#)）

15.7 管脚配置

OTG 控制器有两个 USB 端口，以 USB 管脚名的后缀 1、2 来表示，以下称为 USB 端口 1（U1）和 USB 端口 2（U2）。

表324. USB OTG 端口 1 的管脚

管脚名称	方向	描述	管脚类别
V _{BUS}	I	V _{BUS} 状态输入。当该功能未通过相应 IOCON 寄存器使能时，由内部高电平驱动。	USB 连接器
端口 U1			
USB_D+1	I/O	正向差分数据	USB 连接器
USB_D-1	I/O	负向差分数据	USB 连接器
USB_CONNECT1	O	SoftConnect 控制信号	控制
USB_UP_LED1	O	GoodLink LED 控制信号	控制
$\overline{\text{USB_INT1}}$	I	OTG ATX 中断	外部 OTG 收发器
USB_SCL1	I/O	I ² C 串行时钟	外部 OTG 收发器
USB_SDA1	I/O	I ² C 串行数据	外部 OTG 收发器
$\overline{\text{USB_TX_E1}}$	O	发送使能	外部 OTG 收发器
USB_TX_DP1	O	D+发送数据	外部 OTG 收发器
USB_TX_DM1	O	D-发送数据	外部 OTG 收发器
USB_RCV1	I	差分接收数据	外部 OTG 收发器
r USB_RX_DP1	I	D+接收数据	外部 OTG 收发器
r USB_RX_DM1	I	D-接收数据	外部 OTG 收发器
$\overline{\text{r USB_LS1}}$	O	低速状态（仅应用于主机功能）	外部 OTG 收发器
$\overline{\text{r USB_SSPND1}}$	O	总线挂起状态	外部 OTG 收发器
$\overline{\text{r r USB_PPWR1}}$	O	端口电源使能	主机电源切换
USB_PWRD1	I	端口电源状态	主机电源切换
$\overline{\text{USB_OVRCCR1}}$	I	过流状态	主机电源切换
$\overline{\text{USB_HSTEN1}}$	O	主机使能状态	
端口 U2			
USB_D+2	I/O	正向差分数据	USB 连接器
USB_D-2	I/O	负向差分数据	USB 连接器
USB_CONNECT2	O	SoftConnect 控制信号	控制
USB_UP_LED2	O	GoodLink LED 控制信号	控制
$\overline{\text{USB_PPWR2}}$	O	端口电源使能	主机电源切换
USB_PWRD2	I	端口电源状态	主机电源切换
$\overline{\text{USB_OVRCCR2}}$	I	过流状态	主机电源切换
$\overline{\text{USB_HSTEN2}}$	O	主机使能状态	控制

15.7.1 连接端口 U1 到 OTG

下图显示了使用端口 U1 和 U2 连接到 USB 设备的不同实现方法。在该示例中，外部 OTG 收发器使用了 ISP1302 (ST-Ericsson)，USB 主机电源开关是 LM3526-L (美国国家半导体公司)。有两种方法可以连接 OTG 收发器：

1. 使用内部 USB 收发器做 USB 信令, 外部 OTG 收发器仅用于 OTG 功能(见[图 47](#))。这种选择在 VP/VM 模式下使用内部收发器。
2. 在 VP/VM 模式下, 将内部 OTG 收发器用于 OTG 功能, 以及 USB 信令 (见[图 48](#))。

两种情况下，端口 U2 都连接为一个主机。第一种方案使用的管脚较少。

图47. USB OTG 端口配置：端口 U1 为 OTG 双角色设备，端口 U2 为主机。

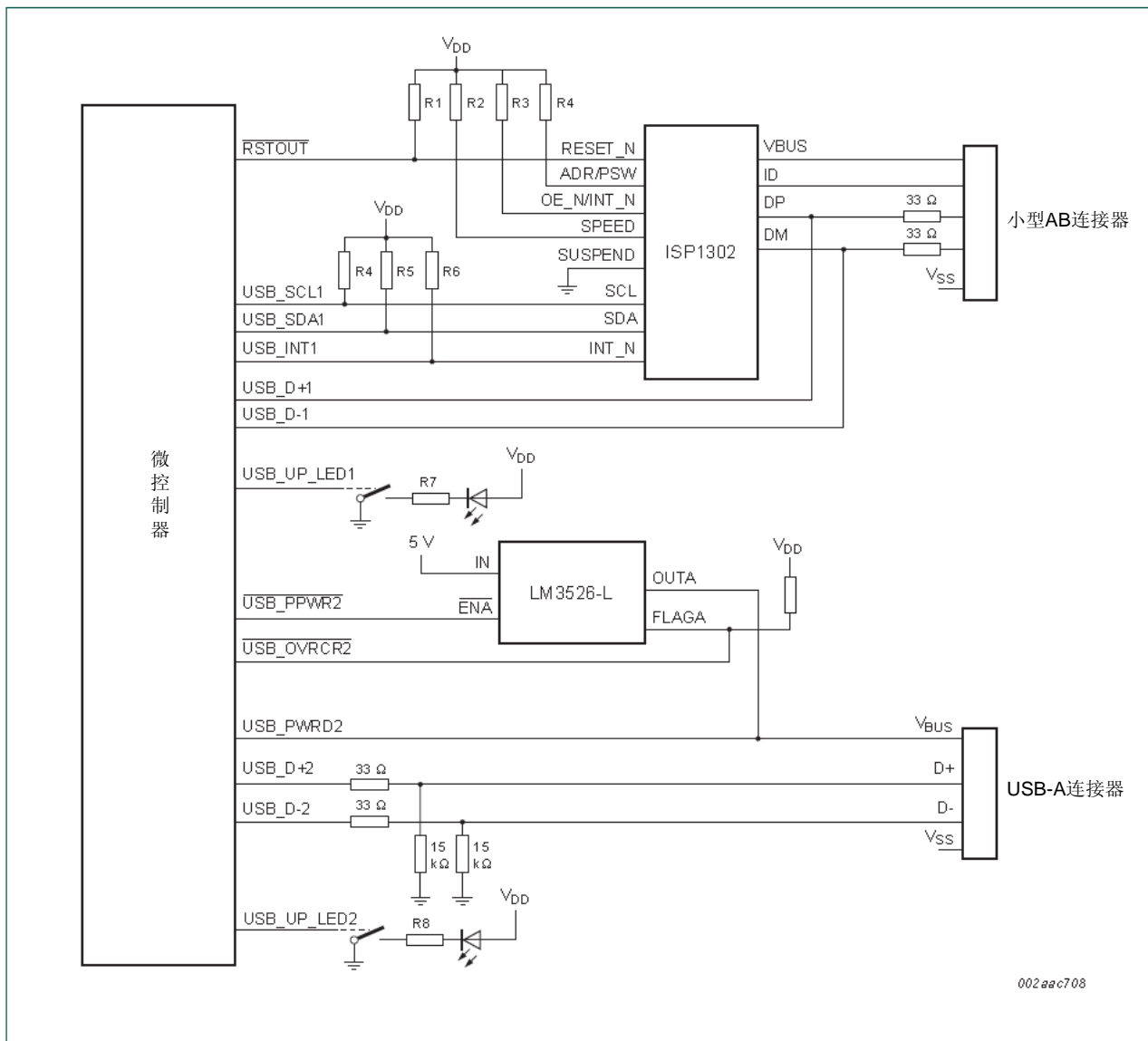
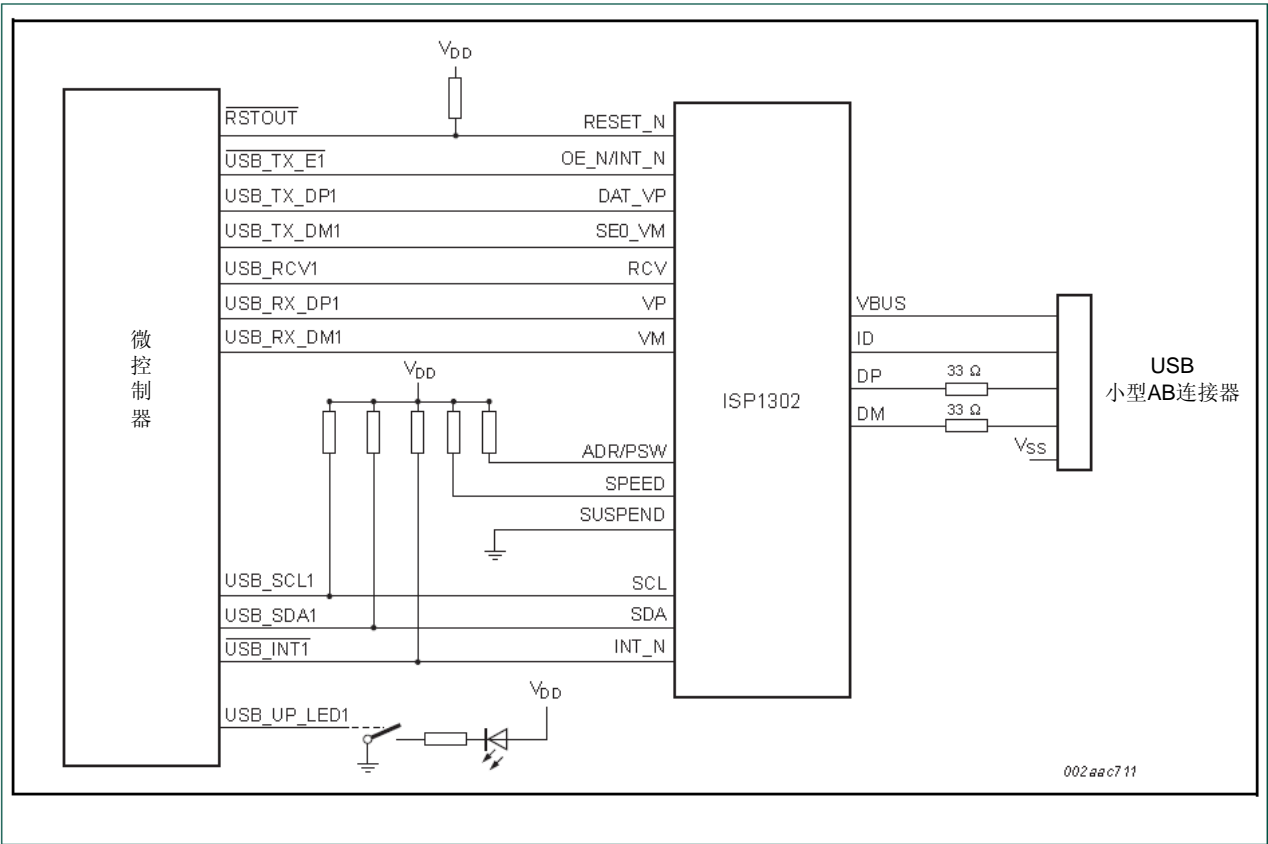


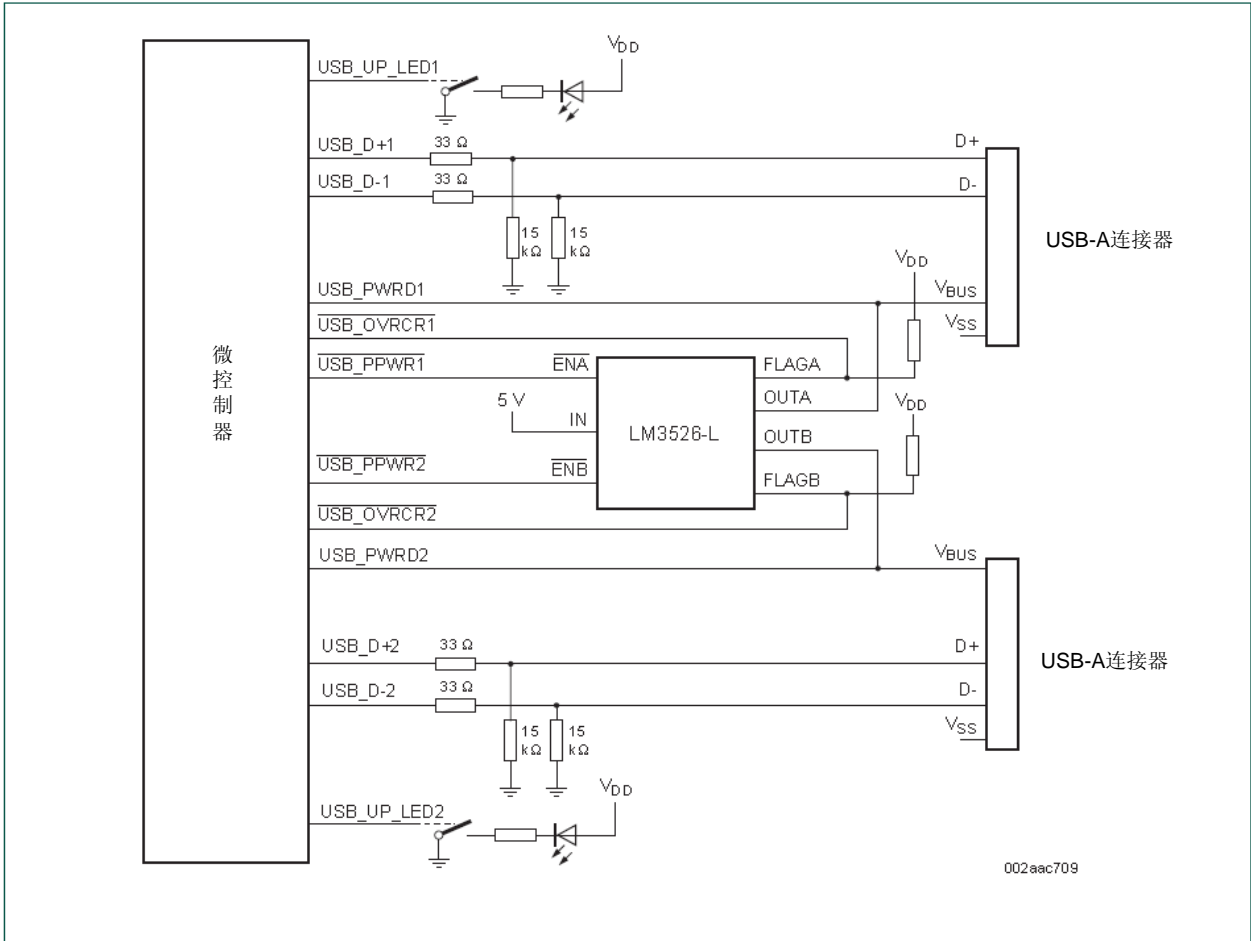
图48. USB OTG 端口配置: VP_VM 模式



15.7.2 将端口 U1 和 U2 作为主机的连接

利用嵌入的 USB 收发器可将端口 U1 和 U2 作为主机进行连接。端口不具有 OTG 功能。

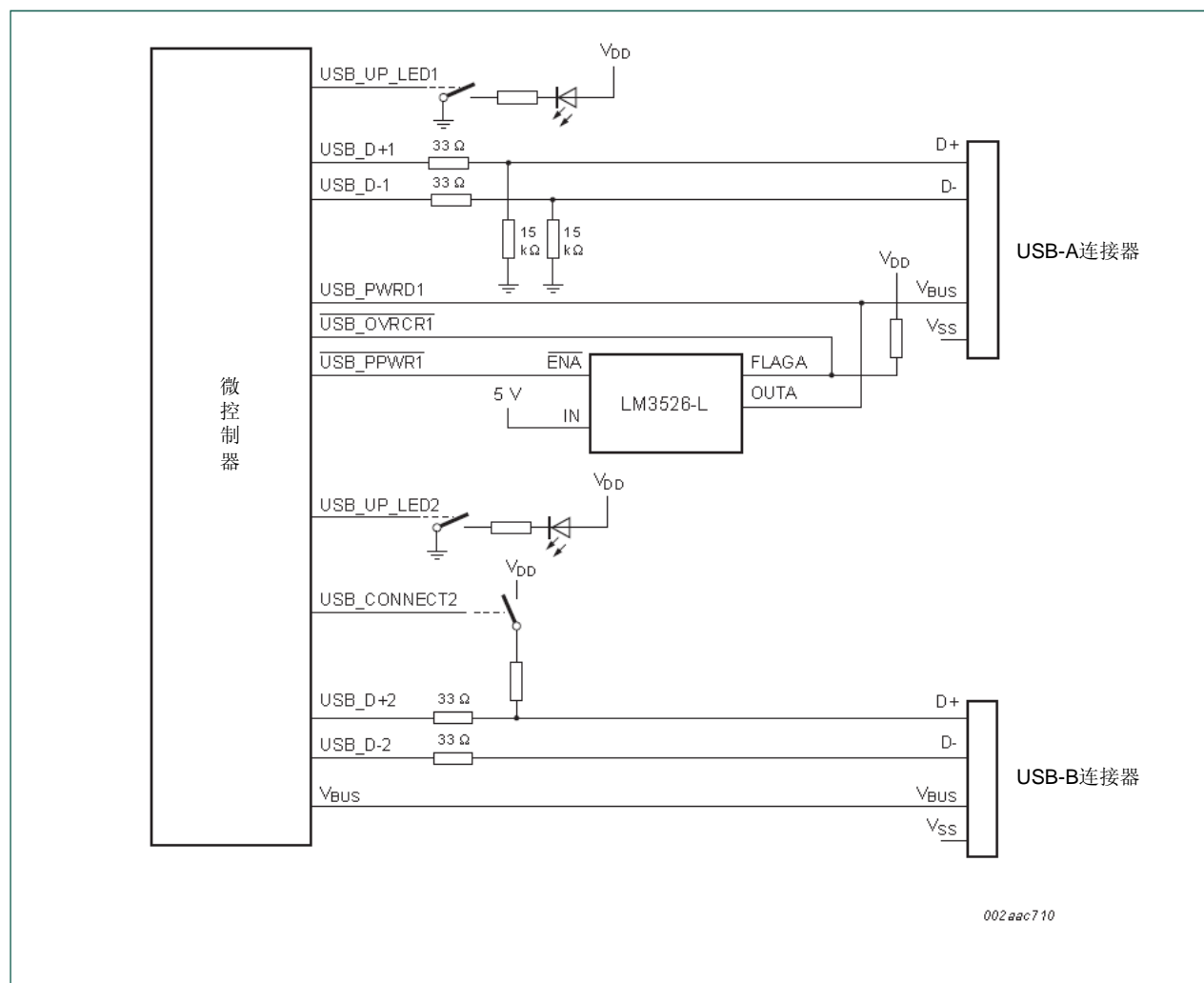
图49. USB 主机端口配置：端口 U1、U2 作为主机



15.7.3 将端口 U1 用作主机、端口 U2 用作设备的连接

端口 U2 连接为设备，端口 U1 连接为主机。两个端口均使用嵌入的 USB 收发器。两个端口上均无 OTG 功能。

图50. USB 设备端口配置：端口 U1 为主机和端口 U2 为设备



15.8 寄存器描述

下表是对 OTG 与 I²C 寄存器的总结。

设备寄存器与主机寄存器分别在“USB 设备控制器”与“USB 主机（OHCI）控制器”章节的[表 256](#)和[表 323](#)中描述。所有寄存器均为 32 位宽，沿字地址边界排列。

表325. USB OTG 和 I²C 寄存器地址定义

名称	描述	访问	复位值	地址	表
中断寄存器					
USBIntSt	USB 中断状态	R/W	0x8000 0100	0x400F C1C0	表 326
OTG 寄存器					
OTGIntSt	OTG 中断状态	RO	0	0x2008 C100	表 327
OTGIntEn	OTG 中断使能	R/W	0	0x2008 C104	表 327
OTGIntSet	OTG 中断设置	WO	无	0x2008 C108	表 327
OTGIntClr	OTG 中断清零	WO	无	0x2008 C10C	表 327
OTGStCtrl ^[1]	OTG 状态和控制	R/W	0	0x2008 C110	表 328
OTGTmr	OTG 定时器	R/W	0xFFFF	0x2008 C114	表 329
I ² C 寄存器					
I2C_RX	I ² C 接收	RO	无	0x2008 C300	表 332
I2C_TX	I ² C 发送	WO	无	0x2008 C300	表 333
I2C_STS	I ² C 状态	RO	0x0A00	0x2008 C304	表 334
I2C_CTL	I ² C 控制	R/W	0	0x2008 C308	表 335
I2C_CLKHI	I ² C 时钟高电平	R/W	0xB9	0x2008 C30C	表 336
I2C_CLKLO	I ² C 时钟低电平	WO	0xB9	0x2008 C310	表 337
时钟控制寄存器					
OTGClkCtrl	OTG 时钟控制器	R/W	0	0x2008 CFF4	表 330
OTGClkSt	OTG 时钟状态	RO	0	0x2008 CFF8	表 331

[1] 仅在设备应用中，该寄存器的位 0 和 1 被用作控制 USB 管脚与端口 1 和 2 的连接（参见[13.10.1](#)节）。

15.8.1 USB 中断状态寄存器（USBIntSt—0x2008 C1C0）

USB OTG 控制器有 7 条中断线。这些中断线相“或”，连接到向量中断控制器的一个通道中。此寄存器使软件能够通过一次读操作确定 7 条中断线的状态。

表326. USB 中断状态寄存器（USBIntSt—0x2008 C1C0）位描述

位	符号	描述	复位值
0	USB_INT_REQ_LP	低优先级中断线的状态。该位只读。	0
1	USB_INT_REQ_HP	高优先级中断线的状态。该位只读。	0
2	USB_INT_REQ_DMA	DMA 中断线的状态。该位只读。	0
3	USB_HOST_INT	USB 主机中断线的状态。该位只读。	0
4	USB_ATX_INT	外部 ATX 中断线的状态。该位只读。	0
5	USB_OTG_INT	OTG 中断线的状态。该位只读。	0
6	USB_I2C_INT	I ² C 模块中断线的状态。该位只读。	0
7	-	保留。读取值未定义，只写入 0。	无
8	USB_NEED_CLK	USB 所需时钟指示器。该位只读。	1
30:9	-	保留。读取值未定义，只写入 0。	无
31	EN_USB_INTS	使能所有 USB 中断。当该位清零时，向量中断控制器看不到 USB 中断线的“或”输出。	1

15.8.2 OTG 中断状态寄存器（OTGIntSt—0x2008 C100）

当在 HNP 主机交换序列期间发生中断事件时，本寄存器中的各个位由硬件置位。这些位的更多设置信息见 [15.9](#) 节。

表327. OTG 中断状态寄存器（OTGIntSt—0x2008 C100）位描述

位	符号	描述	复位值
0	TMR	定时器超时。	0
1	REMOVE_PU	移除上拉电阻。该位由硬件置位来表示软件需要禁能 D+ 上拉电阻。	0
2	HNP_FAILURE	HNP 失败。该位由硬件置位来表示 HNP 切换失败。	0
3	HNP_SUCCESS	HNP 成功。该位由硬件置位来表示 HNP 切换成功。	0
31:4	-	保留。读取值未定义，只写入 0。	无

15.8.3 OTG 中断使能寄存器（OTGIntEn—0x2008 C104）

向此寄存器中的一个位写入 1，可使能 OTGIntSt 中的相应位，从而在某根中断线上产生一个中断。中断与 USBIntSt 寄存器中的 USB_OTG_INT 中断线连接。

OTGIntEn 中各位的分配及复位值与 OTGIntSt 相同。

15.8.4 OTG 中断设置寄存器（OTGIntSet—0x2008 C20C）

向此寄存器中的一个位写入 1，可设置 OTGIntSt 寄存器中的相应位。写入 0 无效。OTGIntSet 中各位的分配与 OTGIntSt 相同。

15.8.5 OTG 中断清除寄存器（OTGIntClr—0x2008 C10C）

向此寄存器中的一个位写入 1，可清零 OTGIntSt 寄存器中的相应位。写入 0 无效。OTGIntClr 中各位的分配与 OTGIntSt 相同。

15.8.6 OTG 状态与控制寄存器（OTGStCtrl—0x2008 C110）

OTGStCtrl 寄存器能够在 HNP 移交序列过程中使能硬件跟踪、控制 OTG 定时器、监控定时器计数，以及控制映射至端口 U1 和 U2 的功能。

OTG 定时器控制着移交序列过程中的时间关键事件。定时器可以在两种模式下工作：

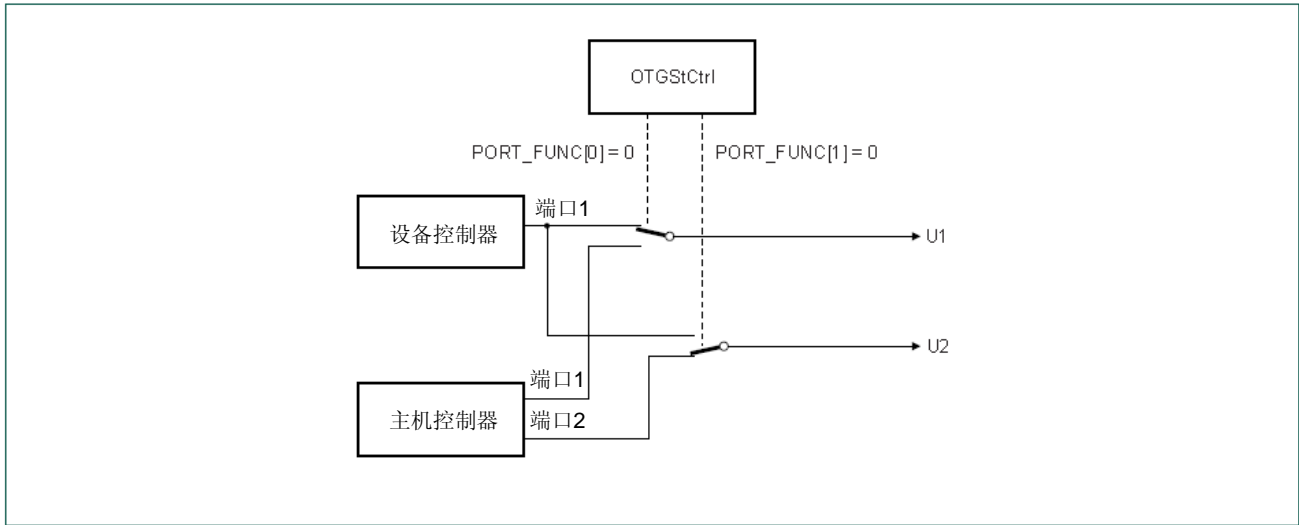
- 1. 单次触发模式：在 TIMEOUT_CNT 的结束产生中断（见 15.8.7 节），OTGIntSt 中的 TMR 置位，定时器将被禁能。
- 2. 自由运行模式：在 TIMEOUT_CNT 的结束产生中断（15.8.7 节），TMR 置位，定时器值重新装入计数器。此模式下，定时器不被禁能。

表328. OTG 状态控制寄存器（OTGStCtrl—0x2008 C110）位描述

位	符号	描述	复位值
1:0	PORT_FUNC	控制 USB 函数的连接（参见表 51）。当 B_HNP_TRACK 或 A_HNP_TRACK 置位且 HNP 成功移交时，位 0 通过硬件置位或清零。参见 15.9 节。 00: U1 = 设备（OTG），U2 = 主机 01: U1 = 主机（OTG），U2 = 主机 10: 保留 11: U1 = 主机，U2 = 设备	0
3:2	TMR_SCALE	定时器调节选择。该字段确定了每个定时器的持续时间。 00: 10 μs（100 KHz） 01: 100 μs（10 KHz） 10: 1000 μs（1 KHz） 11: 保留	0
4	TMR_MODE	定时器模式选择。 0: 单次触发模式 1: 自由运行模式	
5	TMR_EN	定时器使能。该位置位时，TMR_CNT 加 1。该位清零时，TMR_CNT 复位为 0。	0
6	TMR_RST	定时器复位。向该位写入 1 会将 TMR_CNT 复位为 0。 这可以使软件对某个位进行单独控制以在定时器使能时重启定时器。	0
7	-	保留。读取值未定义，只写入 0。	无
8	B_HNP_TRACK	使能 B 设备（外设）的 HNP 跟踪，请参见 15.9 节。当 HNP_SUCCESS 或 HNP_FAILYRE 置位时，硬件会清零该位。	0
9	A_HNP_TRACK	使能 A 设备（主机）的 HNP 跟踪，请参见 15.9 节。当 HNP_SUCCESS 或 HNP_FAILYRE 置位时，硬件会清零该位。	0
10	PU_REMOVED	当 B 设备从外设变为主机时，软件在其移除 D+ 上拉时将该位置位，请参见 15.9 节。	0

位	符号	描述	复位值
		当 HNP_SUCCESS 或 HNP_FAILYRE 置位时，硬件会清零该位。	
15:11	-	保留。读取值未定义，只写入 0。	
31:16	TMR_CNT	当前定时器的计数值。	0

图51. PORT_FUNC bit 0 = 0 和 PORT_FUNC bit 1 = 0 端口选择



15.8.7 OTG 定时器寄存器（OTGTmr—0x2008 C114）

表329. OTG 定时器寄存器（OTGTmr—0x2008 C114）位描述

位	符号	描述	复位值
15:0	TIMEOUT_CNT	TMR 中断在计数值达到该值时置位。	0xFFFF
31:16	-	保留。读取值未定义，只写入 0。	无

15.8.8 OTG 时钟控制寄存器（OTGClkCtrl—0x2008 CFF4）

此寄存器控制着 OTG 控制器的时钟。当软件要访问寄存器时，需要设置相应的时钟控制位。只要相应的 OTGClkCtrl 位已经置位，软件就不必对每次寄存器访问都重复该操作。

表330. OTG 时钟控制寄存器（OTGClkCtrl—0x2008 CFF4）位描述

位	符号	值	描述	复位值
0	HOST_CLK_EN		主机时钟使能	0
		0	禁能主机时钟。	
		1	激活主机时钟。	
1	DEV_CLK_EN		设备时钟使能。	0
		0	禁能设备时钟。	
		1	激活设备时钟。	
2	I2C_CLK_EN		I2C 时钟使能	0
		0	禁能 I2C 时钟。	
		1	激活 I2C 时钟。	

位	符号	值	描述	复位值
3	OTG_CLK_EN		OTG 时钟使能	0
		0	禁能 OTG 时钟。	
		1	激活 OTG 时钟。	
4	AHB_CLK_EN		AHB 主机时钟使能	0
		0	禁能 AHB 主机时钟。	
		1	激活 AHB 主机时钟。	
31:5	-		保留。读取值未定义，只写入 0。	无

15.8.9 OTG 时钟状态寄存器（OTGClkSt—0x2008 CFF8）

此寄存器保存着时钟的有效状态。当通过 OTGClkCtrl 使能了时钟时，软件应轮询这个寄存器中的相应位。如果该位已置位，则软件可以继续对寄存器的访问。只要相应的 OTGClkCtrl 位已经置位，软件就不必对每次寄存器访问都重复该操作。

表331. OTG 时钟状态寄存器（OTGClkSt—0x2008 CFF8）位描述

位	符号	值	描述	复位值
0	HOST_CLK_ON		主机时钟的状态。	0
		0	主机时钟不可用。	
		1	主机时钟可用。	
1	DEV_CLK_ON		设备时钟的状态。	0
		0	设备时钟不可用。	
		1	设备时钟可用。	
2	I2C_CLK_ON		I2C 时钟的状态。	0
		0	I2C 时钟不可用。	
		1	I2C 时钟可用。	
3	OTG_CLK_ON		OTG 时钟的状态。	0
		0	OTG 时钟不可用。	
		1	OTG 时钟可用。	
4	AHB_CLK_ON		AHB 主机时钟的状态。	0
		0	AHB 时钟不可用。	
		1	AHB 时钟可用。	
31:5	-		保留。读取值未定义，只写入 0。	无

15.8.10 I2C 接收寄存器（I2C_RX—0x2008 C300）

此寄存器是接收 FIFO 的顶部字节。接收 FIFO 的深度为 4 字节。Rx FIFO 可通过一次硬复位或一个软复位（I2C_CTL 的第 7 位）刷新。读一个空的 FIFO 会得到不可预期的数据结果。

表332. I2C 接收寄存器（I2C_RX—address 0x2008 C300）位描述

位	符号	描述	复位值
7:0	RX Data	接收数据。	-

15.8.11 I2C 发送寄存器（I2C_TX—0x2008 C300）

此寄存器是传送 FIFO 的顶部字节。传送 FIFO 的深度为 4 字节。

Tx FIFO 可由一个硬复位和软复位（I2C_CTL 第 7 位）刷新，或当发生一次仲裁失败时（I2C_STS 第 3 位）刷新。向一个满 FIFO 写入的数据会被忽略。

必须写 I2C_TX 进行读写操作来传输每个字节。主机接收操作时，位[7:0]被忽略。主机接收器必须为它期望在 Rx FIFO 中接收到的每个字节写一个伪字节(dummy byte)到 Tx FIFO。当停止位或起始位置位，从而对一个写入 Tx FIFO 的字节制造产生了重启条件时（主机接收器），则向从机读取的字节不会被应答。即，主机接收到的最后字节不会被应答。

表333. I²C 发送寄存器（I2C_TX—0x2008 C300）位描述

位	符号	描述	复位值
7:0	TX Data	发送数据。	-
8	START	为 1 时，在发送该字节前发送一个起始条件。	-
9	STOP	为 1 时，在发送完该字节后发送一个停止条件。	-
31:10	-	保留。读取值未定义，只写入 0。	-

15.8.12 I²C 状态寄存器（I2C_STS—0x2008 C304）

I2C_STS 寄存器提供了 TX 和 RX 模块的状态，以及外部总线当前状态的信息。I2C_CTL 寄存器的各个位被中断使能，并连接到 USBIntSt 中的位 I2C_USB_INT。

表334. I²C 状态寄存器（I2C_STS—0x2008 C304）位描述

位	符号	值	描述	复位值
0	TDI		处理结束中断。处理成功完成时该位置位。向状态寄存器的位 0 写入 1 可清除中断。从机操作对其没有影响。	0
		0	处理还没有完成。	
		1	处理已完成。	
1	AFI		仲裁失败中断。发送时，若 SDAOUT 为高时 SDA 为低，I ² C 会失去对另一个总线设备的仲裁，这时仲裁失败位置位。向状态寄存器的位 1 写入 1 可清除中断。	0
		0	在最后一次发送时没有仲裁失败。	
		1	在最后一次发送时出现仲裁失败。	
2	NAI		无应答中断。每发送完一个字节后，发送器都希望从接收器接收一个应答。若没有收到应答，该位置位。当主机 TX FIFO 有字节写入时中断被清除。	0
		0	发送完最后一个字节后接收到了应答。	
		1	发送完最后一个字节后没有收到应答。	
3	DRMI		主机请求数据中断。一旦传输开始，只要它后面没有跟随停止条件，发送器就必须有足够的数	0
			据来发送，或者在有数据可发送之前一直将 SCL 线保持为低电平。当主机发送器的数据发送完毕时主机数据请求位就会置位。若主机 TX FIFO 没有数据（即为空）且发送完最后一个字节后没有停止条件标志，那么 SCL 就必须一直保持低电平直至 CPU 写入新的发送字节。当有字节写入主机 TX FIFO 时该位会清零。	
		0	主机发送器不需要数据。	
		1	主机发送器需要数据。	
4	DRSI		从机请求数据中断。一旦传输开始，只要它后面没有跟随停止条件，发送器就必须有足够的数	0
			据来发送，或者在有数据可发送之前一直将 SCL 线保持为低电平。当从机发送器的数据发送完毕时从机数据请求位就会置位。若从机 TX FIFO 没有数据（即为空）且发送完最后一个字节后没有停止条件标志，那么 SCL 就必须一直保持低电平直至 CPU 写入新的发送字节。当有字节写入从机 TX FIFO 时该位会清零。	

位	符号	值	描述	复位值
		0	从机发送器不需要数据。	
		1	从机发送器需要数据。	
5	Active		表示总线是否忙碌。该位在起始条件出现时置位，在停止条件出现时清零。	0
6	SCL		SCL 信号的当前值。	-
7	SDA		SDA 信号的当前值。	-
8	RFF		接收 FIFO 满（RFF）。当 RX FIFO 满且不能再接收任何数据时该位置位。 当 RX FIFO 不满时该位清零。若在接收 FIFO 满时还有数据达到，则 SCL 保持低电平直至 CPU 读取 RX FIFO 并为该数据腾出空间为止。	0
		0	RX FIFO 不满	
		1	RX FIFO 满	
9	RFE		接收 FIFO 空。该位在 RX FIFO 空时置位，在 RX FIFO 有有效数据时清零。	1
		0	RX FIFO 有数据。	
		1	RX FIFO 空	
10	TFF		发送 FIFO 满。该位在 TX FIFO 满时置位，在 TX FIFO 不满时清零。	0
		0	TX FIFO 不满。	
		1	TX FIFO 满	
11	TFE		发送 FIFO 空。该位在 TX FIFO 空时置位，在 TX FIFO 有有效数据时清零。	1
		0	TX FIFO 有有效数据。	
		1	TX FIFO 空	
31:12	-		保留。读取值未定义，只写入 0。	无

15.8.13 I²C 控制寄存器（I2C_CTL—0x2008 C308）

I2C_CTL 用于使能中断和复位 I²C 状态机。置位时，使能的中断会使 USB_I2C_INT 中断输出线被拉高。

表335. I²C 控制寄存器（I2C_CTL—0x2008 C308）位描述

位	符号	描述	复位值
0	TDIE	发送完成中断使能。该位可使能 TDI 中断发信号示意 I ² C 发出一个停止条件。	0
		0	禁能 TDI 中断。
		1	使能 TDI 中断。
1	AFIE	发送器仲裁失败中断使能。当试图把 SDA 设置为高电平时，该位使能在发送过程中被拉高0的 AFI 中断，但总线通过另一个器件被驱动为低电平。	
		0	禁能 AFI。
		1	使能 AFI。
2	NAIE	发送器无应答中断使能。该位可使能 NAI 中断发信号示意发已送的字节未被应答。	0
		0	禁能 NAI。
		1	使能 NAI。
3	DRMIE	主机发送器数据请求中断使能。该位可使能 DRMI 中断通知主机发送器已用完数据、尚未发0起停止条件，并保持 SCL 线为低电平。	
		0	禁能 DRMI 中断。
		1	使能 DRMI 中断。
4	DRSIE	从机发送器数据请求中断使能。该位可使能 DRSI 中断通知从机发送器已用完数据、尚未发0	

位	符号	描述	复位值
		起停止条件，并保持 SCL 线为低电平。	
		0 禁能 DRSI 中断。	
		1 使能 DRSI 中断。	
5	REFIE	接收 FIFO 满中断使能。该位可使能接收 FIFO 满中断以示意接收 FIFO 不能再接收数据。	0
		0 禁能 RFFI。	
		1 使能 RFFI。	
6	RFDAIE	有可用的接收数据中断使能。该位可使能 DAI 中断来表示接收 FIFO 中有可用数据（即不空）。	0
		0 禁能 DAI。	
		1 使能 DAI。	
7	TFFIE	发送 FIFO 不满中断使能。该位可使能发送 FIFO 不满中断以示意可以向发送 FIFO 写入数据。 注：该 FIFO 不满。这有助于 CPU 向 I ² C 模块写数据（FIFO 有空间时），而且不需要轮询状态寄存器。	0
		0 禁能 TFFI。	
		1 使能 TFFI。	
8	SRST	软复位。只有在少许情况下才需要软复位。如：器件发起了起始条件确没发送停止条件。若总线保持忙状态的时间长于超时周期，系统定时器就会复位 I ² C 总线。在软复位时，TX 和 Rx FIFO 被刷新，I2C_STS 寄存器清空，所有内部状态机被复位以出现空闲。软复位不能修改 I2C_CLKHI、I2C_CLKLO 和 I2C_CTL（除软复位位以外）。	0
		0 请见文本。	
		1 将 I ² C 总线复位（闲置）。自清零。	
31:9	-	保留。读取值未定义，只写入 0。	无

15.8.14 I²C 时钟高电平寄存器（I2C_CLKHI—0x2008 C30C）

CLK 寄存器中包含了对 48MHz 时钟周期的最终计数，来创建低速 I²C 串行时钟 SCL 的高电平周期。

表336. I²C_CLKHI 寄存器（I2C_CLKHI—address 0x2008 C30C）位描述

位	符号	描述	复位值
7:0	CDHI	时钟分频器高电平。该值是 48MHz 时钟的数目，串行时钟（SCL）将为高电平周期。	0xB9

15.8.15 I²C 时钟低电平寄存器（I2C_CLKLO—0x2008 C310）

CLK 寄存器中包含了对 48MHz 时钟周期的最终计数，来创建低速 I²C 串行时钟 SCL 的低电平周期。

表337. I²C_CLKLO 寄存器（I2C_CLKLO—address 0x2008 C310）位描述

位	符号	描述	复位值
7:0	CDLO	时钟分频器低电平。该值是 48MHz 时钟的数目，串行时钟（SCL）将为低电平周期。	0xB9

15.8.16 中断处理

在 HNP 切换过程中，置位并清零 OTGIntSt 寄存器中设置的中断。所有与 OTG 有关的中断（如已使能）均被连接到 USBIntSt 寄存器的位 USB_OTG_INT。

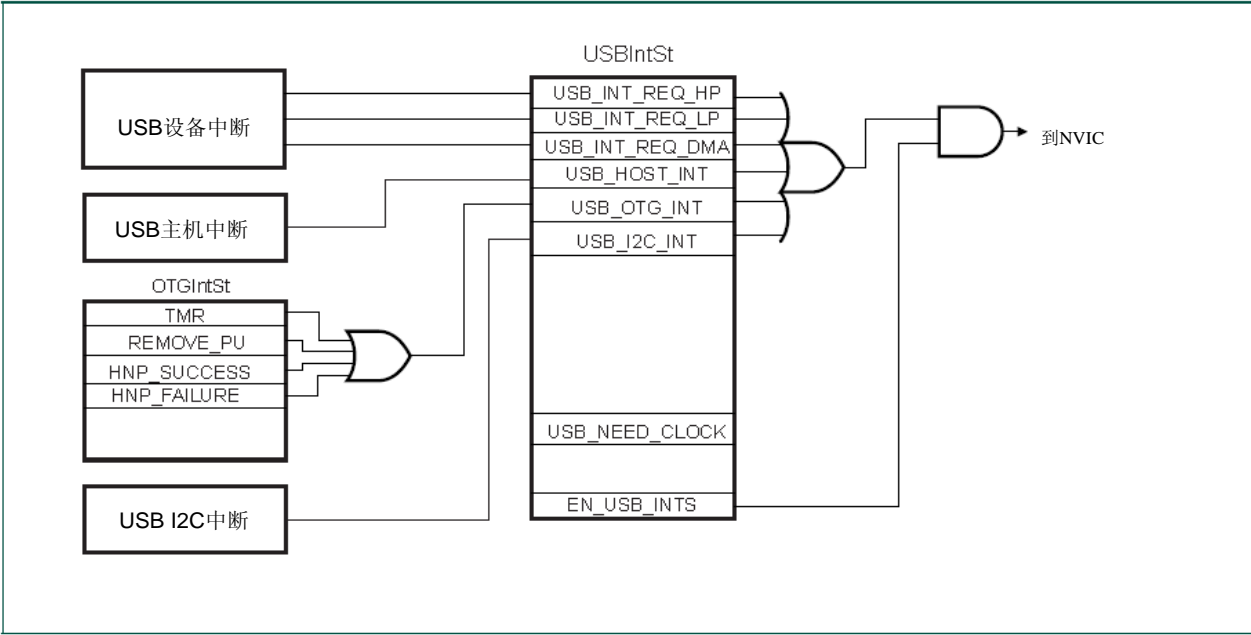
与 I²C 有关的中断在 I2C_STS 寄存器中置位，如果使能，则通过 I2C_CTL 连接到位 USB_I2C_INT。

有关设备控制器产生的中断的更多细节，见 USB 设备一章。对于主机控制器产生的中断，见 OHCI 规范。

USBIntSt 寄存器中的 EN_USB_INTS 位用于使能连接任何 USB 相关的中断到 NVIC 控制器（见图 52）。

注：当 HNP 在主机与设备之间切换过程中（OTG 栈有效），一个操作可将中断的级别提高。建议让 OTG 栈启动基于中断的操作，忽略设备与主机级别的中断。这意味着在 HNP 切换过程中，OTG 栈要提供与主机和设备控制器之间的通信。

图52. USB OTG 中断处理



15.9 支持 HNP

本节描述了 OTG 控制器所提供的对主机协商协议（HNP）的硬件支持。

当两个双角色 OTG 设备相互连接时，mini-AB 插座的插入就决定了各台设备的默认角色。用 mini-A 插头的设备默认为主机（A 设备），用 mini-B 插头的设备默认为从机（B 设备）。

一旦连接以后，默认主机（A 设备）与默认从机（B 设备）就可以通过使用 HNP 切换主从机角色。

图 53 显示了 OTG 控制器的操作流程。每个控制器（主机、设备或 OTG）均通过一系列状态与控制寄存器以及中断来和它们的软件栈通信。另外，OTG 软件栈可以通过 I²C 接口和外部收发器中断信号，与外部 OTG 收发器通信。

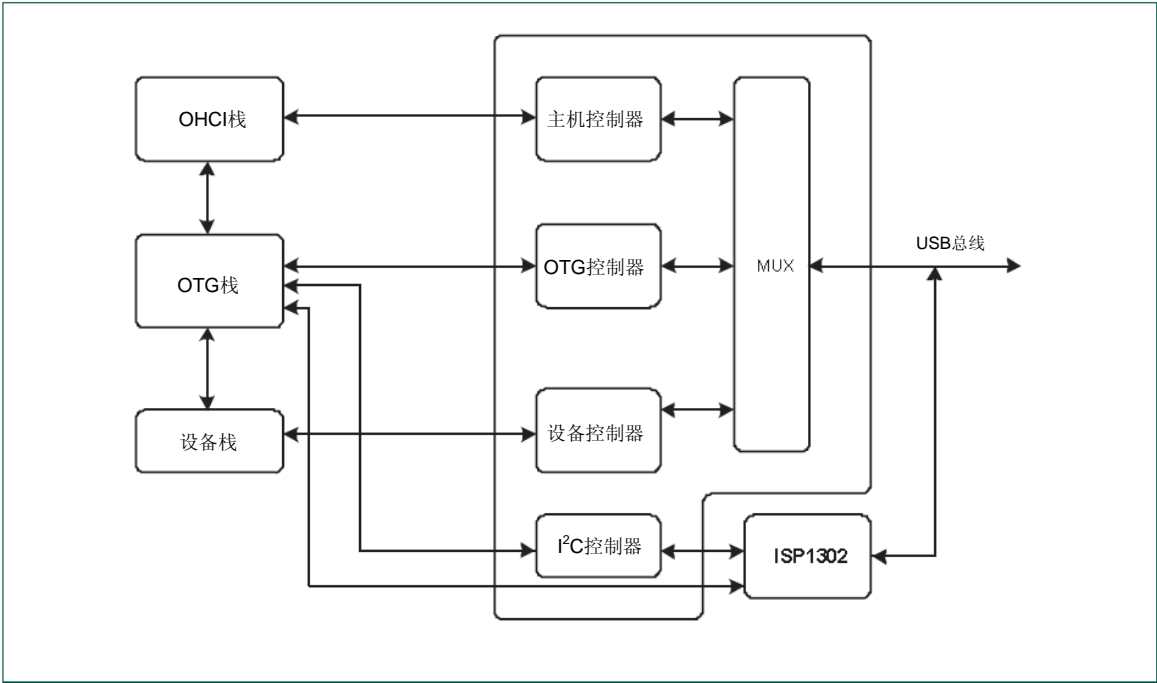
OTG 软件栈遵守 USB 2.0 规范的 On-The-Go 增补，负责执行 HNP 状态机。

OTG 控制器硬件为 HNP 状态机中的一些状态切换提供支持，如下列章节所示。

以下文档中有关于 USB 状态机、HNP 切换，以及 USB 控制器之间通信的详细内容。

- USB OHCI 规范
- USB OTG 增补，版本 1.2
- USB 2.0 规范
- ISP1302 数据手册与用户手册

图53. 有软件栈的 USB OTG 控制器



15.9.1 B 设备：外设到主机的切换

这种情况下，OTG 控制器属于 B 设备，默认角色是外设，现在要将其角色从外设切换到主机。

On-The-Go 增补使用一个状态框图定义了 HNP 切换过程中双角色 B 设备的行为。OTG 软件栈负责双角色 B 设备状态图中所有状态的实现。

OTG 控制器硬件支持双角色 B 设备状态框图中 b_peripheral、b_wait_acon 与 b_host 之间的状态切换。置位 OTGStCtrl 寄存器中的 B_HNP_TRACK 位，使硬件支持 B 设备从外设到主机的切换。置位该位后的硬件操作如[图 54](#)所示。

图54. B-设备从外设状态切换到主机状态的硬件支持

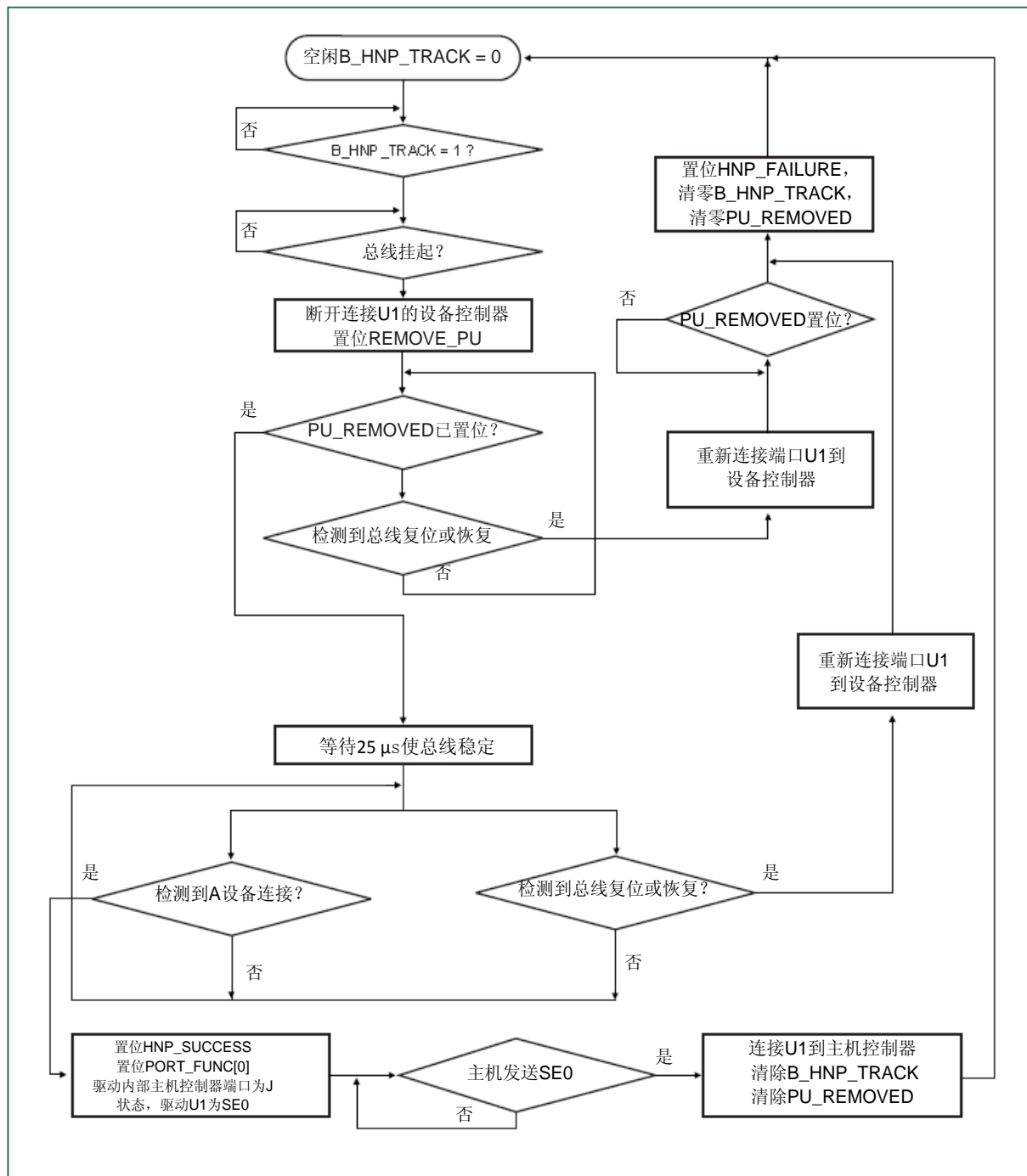


图55. B-设备从外设状态切换到主机状态时软件的状态转变

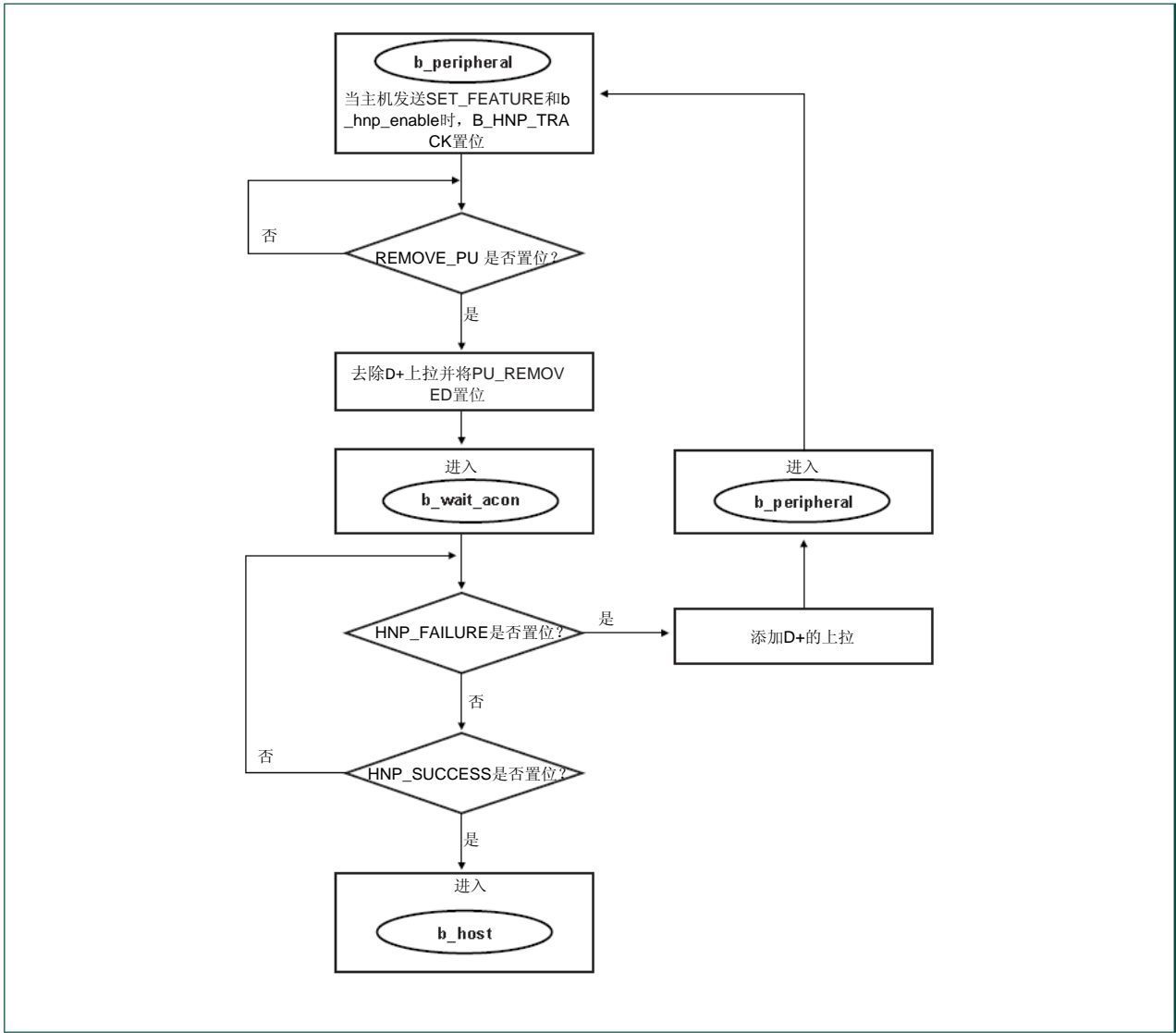


图 55 显示了 OTG 软件栈为了响应硬件设置 REMOVE_PU、HNP_SUCCESS、AND HNP_FAILURE 的操作而必须要执行的操作。另外还给出了软件操作与双角色 B 设备状态的关系。B 设备状态已用黑体标出，并用椭圆圈住。

注意，图中只显示了一部分硬件支持的 B 设备的 HNP 状态和状态切换。软件负责执行所有 HNP 状态。

图 55 可能表示出应轮询 REMOVE_PU 这样的中断位，但如果相应中断已被使能，则不必要。

下面的代码示例显示出如何完成图 55 中的操作。在该示例中，假设外部 OTG 收发器使用的是 ISP1302。

移除 D+上拉电阻

```
/* Remove D+ pull-up through ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x007; // Send OTG Control (Clear) register address
OTG_I2C_TX = 0x201; // Clear DP_PULLUP bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

增加 D+上拉电阻

```
/* Add D+ pull-up through ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x006; // Send OTG Control (Set) register address
OTG_I2C_TX = 0x201; // Set DP_PULLUP bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

15.9.2 A 设备：主机到外设的 HNP 切换

这种情况下，OTG 控制器的角色 A 设备，默认为主机。现在要将 A 设备从主机角色切换为外设。

On-The-Go 增补中用一个状态机图定义了双角色 A 设备在 HNP 切换期间的行为。在双角色 A 设备状态图中，OTG 软件栈负责实现所有状态。

OTG 控制器硬件支持双角色 A 设备状态图中状态 a_host、a_suspend、a_wait_vfall 和 a_peripheral 之间的切换。置位 OTGStCtrl 寄存器中的 A_HNP_TRACK，使硬件允许 A 设备从主机状态切换成设备状态。该位置位后的硬件操作如图 56 所示。

图56. A-设备从主机状态切换到外设状态的硬件支持

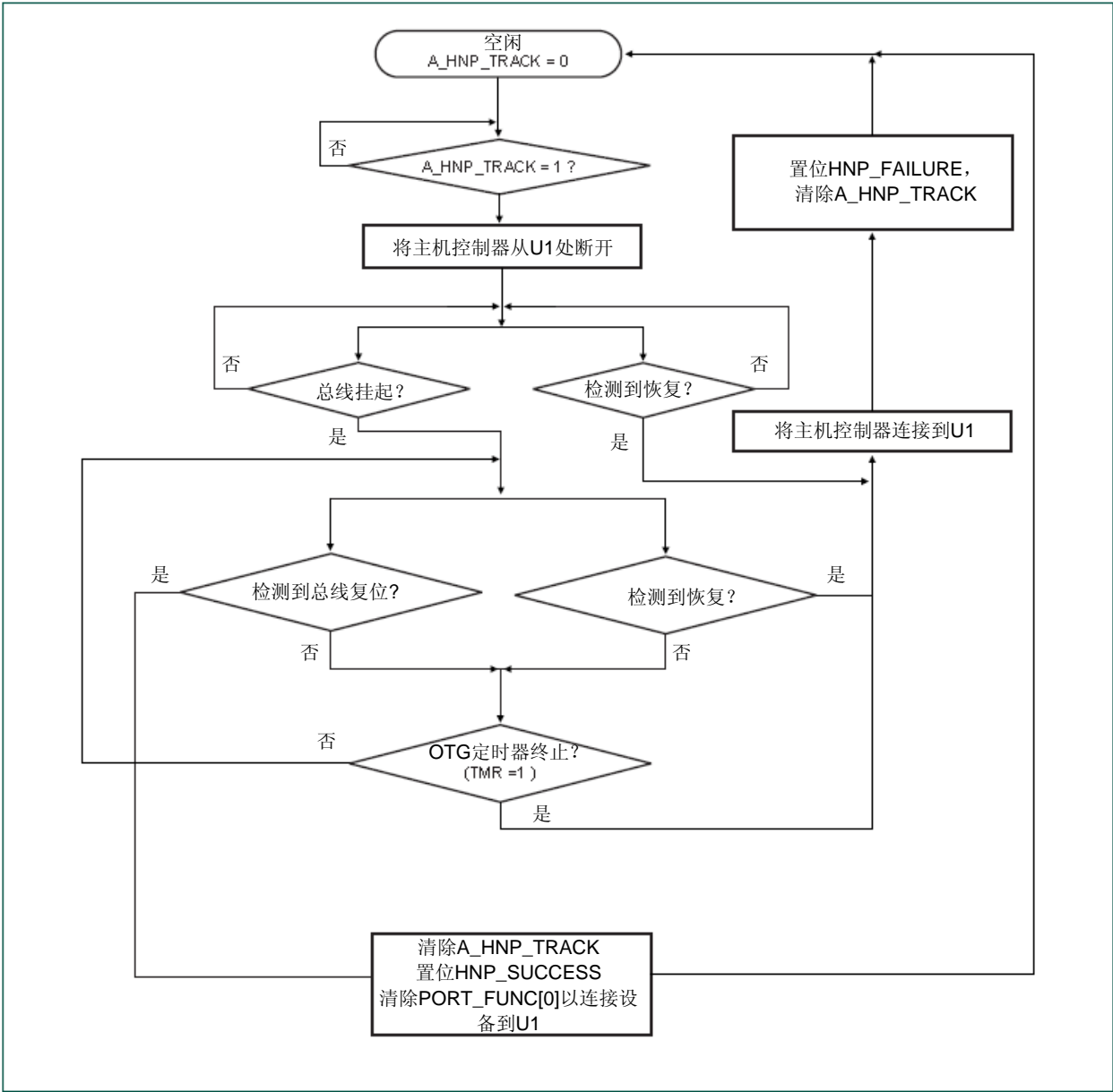
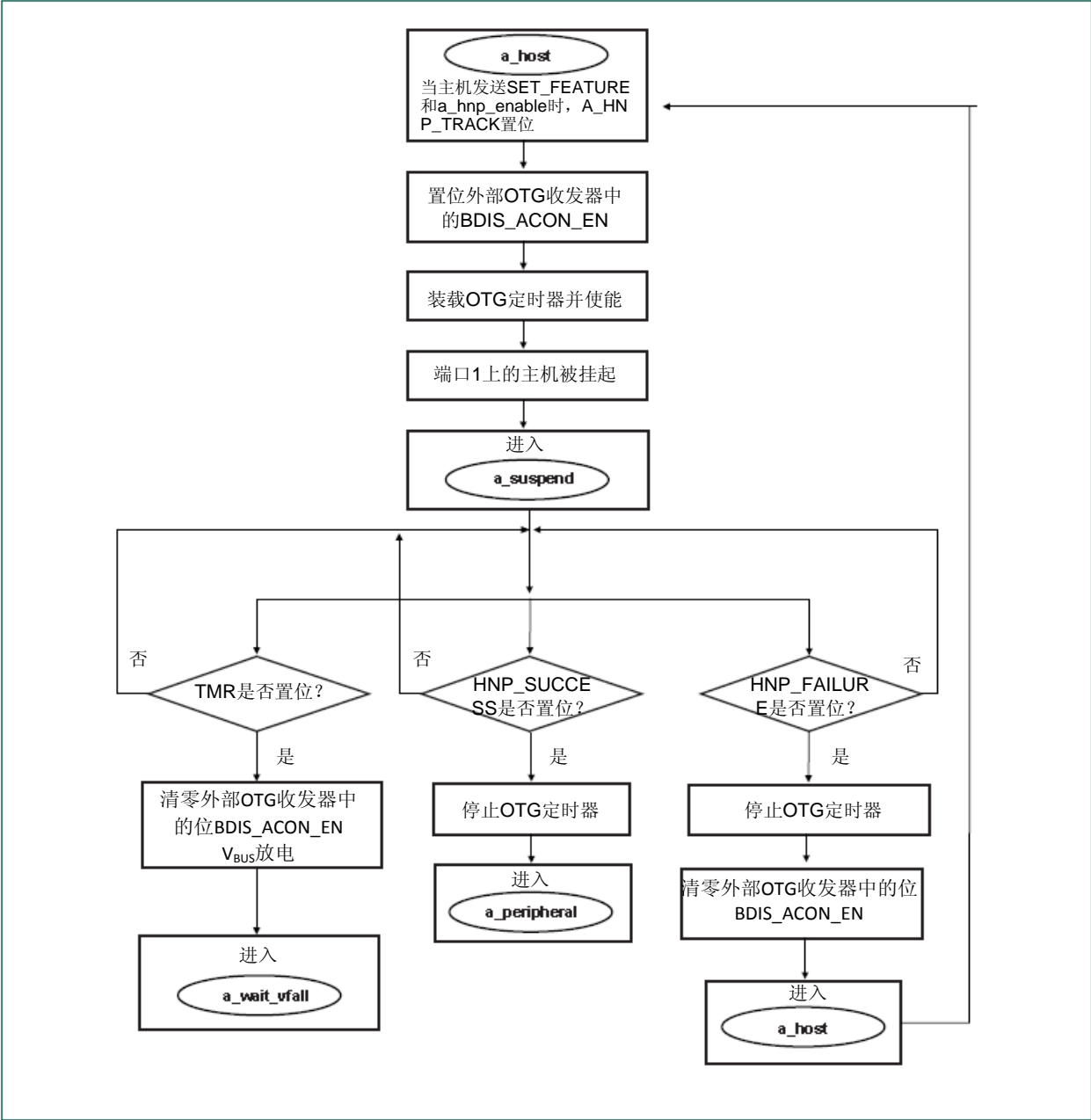


图 57 显示了为响应硬件对 TMR、HNP_SUCCESS 和 HNP_FAILURE 的操作，OTG 软件栈应执行的操作。同时还显示了软件与双角色 A 设备状态之间的关系。A 设备的状态已用黑体标出，并用椭圆圈住。

图57. A-设备从主机状态切换到外设状态时软件的状态转变



注意, 图中只显示了一部分硬件支持的 A 设备 HNP 状态和状态转换。软件负责所有 HNP 状态的实现。

[图 57](#) 可能暗示应轮询 TMR 等中断位, 但如果相应中断已使能, 就不需要轮询了。

下面给出了完成[图 57](#)操作所需的代码示例。在该示例中, 假设外部 OTG 收发器使用的是 ISP1302。

置位外部 OTG 收发器的位 BDIS_ACON_EN

```
/* Set BDIS_ACON_EN in ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x004; // Send Mode Control 1 (Set) register address
OTG_I2C_TX = 0x210; // Set BDIS_ACON_EN bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

清零外部 OTG 收发器的位 BDIS_ACON_EN

```
/* Set BDIS_ACON_EN in ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x005; // Send Mode Control 1 (Clear) register address
OTG_I2C_TX = 0x210; // Clear BDIS_ACON_EN bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;
```

V_{BUS} 放电

```
/* Clear the VBUS_DRV bit in ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x007; // Send OTG Control (Clear) register address
OTG_I2C_TX = 0x220; // Clear VBUS_DRV bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));

/* Clear TDI */
OTG_I2C_STS = TDI;

/* Set the VBUS_DISCHRG bit in ISP1302 */
OTG_I2C_TX = 0x15A; // Send ISP1302 address, R/W=0
OTG_I2C_TX = 0x006; // Send OTG Control (Set) register address
OTG_I2C_TX = 0x240; // Set VBUS_DISCHRG bit, send STOP condition

/* Wait for TDI to be set */
while (!(OTG_I2C_STS & TDI));
```



```
/* Clear TDI */
OTG_I2C_STS = TDI;
```

装载并使能 OTG 定时器

```
/* The following assumes that the OTG timer has previously been */
/* configured for a time scale of 1ms (TMR_SCALE = "10") */
/* and monoshot mode (TMR_MODE = 0) */

/* Load the timeout value to implement the a_aidl_bdis_tmr timer */
/* the minimum value is 200ms */
OTG_TIMER = 200;

/* Enable the timer */
OTG_STAT_CTRL |= TMR_EN;
```

停止 OTG 定时器

```
/* Disable the timer - causes TMR_CNT to be reset to 0 */
OTG_STAT_CTRL &= ~TMR_EN;

/* Clear TMR interrupt */
OTG_INT_CLR = TMR;
```

端口 1 上的主机被挂起

```
/* Write to PortSuspendStatus bit to suspend host port 1 - */
/* this example demonstrates the low-level action software needs to
take. */
/* The host stack code where this is done will be somewhat more involved.
*/
HC_RH_PORT_STAT1 = PSS;
```

15.10 时钟与功率管理

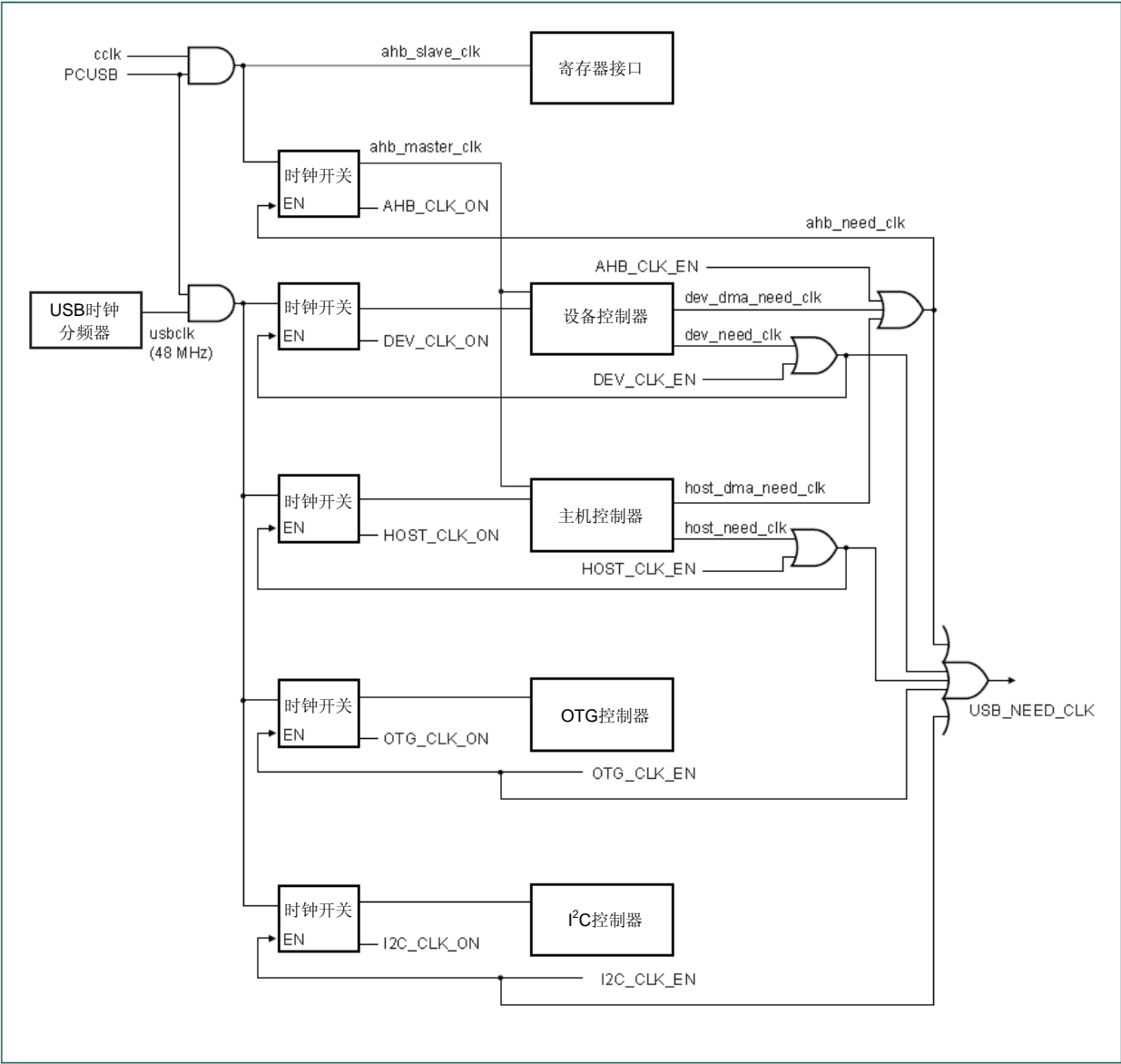
OTG 控制器的时钟如[图 58](#)所示。

除 `ahb_slave_clk` 以外, 时钟开关控制着每个时钟。当时钟开关的使能有效时, 其时钟输出就会开启, `CLK_ON` 输出有效。OTGClkSt 寄存器中可观察到 `CLK_ON` 信号。

为了节省功耗, 可以将不使用的设备、主机、OTG 以及 I²C 控制器的时钟关闭, 方法是将 OTGClkCtrl 寄存器中相应的 `CLK_EN` 位清零。当整个 USB 模块停止运行时, 清零 PCONP 寄存器中的 `PCUSB` 位可以关闭所有 USB 时钟。

当软件要访问某个控制器中的寄存器时, 应首先确保已使能了相关控制器的 48MHz 时钟, 方法是置位 OTGClkCtrl 寄存器中的 `CLK_EN` 位, 然后轮询 OTGClkSt 中的 `CLK_ON` 位, 直到其置位为止。一旦 `CLK_ON` 置位, 控制器的时钟就将一直保持有效, 直到 `CLK_EN` 位被软件清零。如在 48MHz 时钟未使能情况下访问某个控制器的寄存器, 则会导致数据异常中断。

图58. 时钟和功率的控制



15.10.1 设备时钟请求信号

设备控制器有两个时钟请求信号：`dev_need_clk` 和 `dev_dma_need_clk`。当这些信号有效时，它们会分别开启设备时钟（48MHz）和 `ahb_master_clk`。

当设备未挂起，或设备挂起但 USB 总线上检测到有活动时，`dev_need_clk` 信号有效。如果检测到一个断线设备断开连接（“SIE 获取设备状态寄存器”中清零了 `CON` 位—[13.10.7](#) 节），则 `dev_need_clk` 信号无效。软件不需要访问设备控制器寄存器时，这个信号允许 `DEV_CLK_EN` 在正常操作中清零。设备将继续正常工作，并在设备挂起或断开连接时，自动关闭时钟。

设备控制器对存储器的任何 DMA 访问都会使 `dev_dma_need_clk` 置为有效。一旦该信号使能，它会保持 2ms（2 帧）的有效状态，以确保 DMA 吞吐量不会因再使能 `ahb_master_clk` 所产生任何延时而受到影响。DMA 访问结束后的 2ms，`dev_dma_need_clk` 失效，以节省功耗。该信号允许在正常工作期间清零 `AHB_CLK_EN`。

15.10.1.1 主机时钟请求信号

主机控制器有两个时钟请求信号：`host_need_clk` 和 `host_dma_need_clk`。这些信号有效时，会分别开启主机时钟（48 MHz）和 `ahb_master_clk`。

当主机控制器功能状态不是挂起，或者功能状态是挂起，而 USB 总线上检测到了一个恢复信号或一个中断时，`host_need_clk` 信号有效。当软件不需要访问主控制器寄存器时，这个信号允许 `HOST_CLK_EN` 在正常操作中清零。主机会继续正常工作，并在设备挂起或断开连接时自动关闭时钟。

当主控制器有任何对存储器的 DMA 访问时，都会使 `host_dma_need_clk` 有效。一旦有效，它会保持 2ms（2 帧）的有效状态，以确保 DMA 吞吐量不会因再使能 `ahb_master_clk` 所产生任何延时而受到影响。DMA 访问结束后 2ms，`host_dma_need_clk` 失效，以节省功耗。该信号允许 `AHB_CLK_EN` 在正常操作中清零。

15.10.2 掉电模式支持

LPC178x/177x 可以配置为在 USB 总线活动时从掉电模式中唤醒。当芯片处于掉电模式，且 USB 中断被使能时，`USB_NEED_CLK` 的有效可将芯片从掉电模式中唤醒。

当 USB 活动中断被使能时，在进入掉电模式以前，必须使 `USB_NEED_CLK` 无效。实现方法是清零 `OTGClkCtrl` 中的所有 `CLK_EN` 位，使主机控制器进入挂起功能状态。如果需要等待 `dma_need_clk` 信号或 `dev_need_clk` 无效，则可以轮询 `USBIntSt` 中的 `USB_NEED_CLK` 状态，以确认它们何时全部无效。

15.11 USB OTG 控制器初始化

LPC178x/177x OTG 设备控制器的初始化包括下列步骤：

1. 设置 PCONP 中的 PCUSB 位，使能设备控制器。
2. 配置并使能副 PLL（PLL1）或主 PLL（PLL0），以获得 usbcclk（48MHz），和期望的 cclk 频率。关于确定 PLL 设置与配置的步骤，见 [4.5.10](#) 节。
3. 置位 USBCIkCtrl 寄存器中的相应 CLK_EN 位，使能所需的控制器时钟。轮询 USBCIkSt 寄存器中的 CLK_ON 位，直到它们置位为止。
4. 写入相应的 IOCON 寄存器，使能所需的 USB 管脚功能。
5. 按 [13.13](#) 节中的相关步骤，初始化设备控制器。
6. 遵循 OpenHCI 规范中给出的指导，初始化主机控制器。

16.1 基本配置

SD 卡接口（也称为 MCI 或多媒体卡接口）的配置需使用以下寄存器：

1. 电源：在 PCONP 寄存器（[表 37](#)）中，设置 PC_SD 位。
注：复位时，SD 卡接口会被禁用（PCSD = 0）。
2. 外设时钟：SD 卡接口使用公共 PCLK，它既用于总线接口，也用于大多数 APB 外设的功能部分。参见 [4.6.4](#) 节。
3. 管脚：通过相关的 IOCON 寄存器，选择 SD 卡接口管脚及其模式（[8.4.1](#) 节）。
4. 利用相应的中断置位使能寄存器来使能 NVIC 中的中断。

16.2 简介

SD（Secure Digital）卡接口是先进外设总线（Advanced Peripheral Bus, APB）系统总线与多媒体和/或 SD 存储卡之间的接口。它包括两个部分：

- SD 卡接口提供 SD 存储卡专用的所有功能，如时钟生成单元、电源管理控制、命令和数据传输。另外，该接口支持多媒体卡接口（Multimedia Card Interface）。
- APB 接口访问 SD 卡接口寄存器，并产生中断和 DMA 请求信号。

16.3 特性

SD 卡接口提供下列特性：

- 符合 SD 存储卡物理层规范，v0.96（*Secure Digital Memory Card Physical Layer Specification, v0.96*）。
- 符合多媒体卡规范，v2.11（*Multimedia Card Specification v2.11*）。
- 用作多媒体卡总线或 SD 存储卡总线主机。它可以连接到多个多媒体卡，也可以将其与单个 SD 存储卡连接。
- 通过通用 DMA 控制器（General Purpose DMA Controller）提供 DMA 支持。

16.4 管脚描述

表338. SD/MMC 卡接口管脚描述

管脚名称	类型	描述
SD_CLK	输出	时钟输出
SD_CMD	输入	命令输入/输出。
SD_DAT[3:0]	输出	数据线。仅 SD_DAT[0]被用于多媒体卡。
SD_PWR	输出	外部卡电源的电源使能管脚。

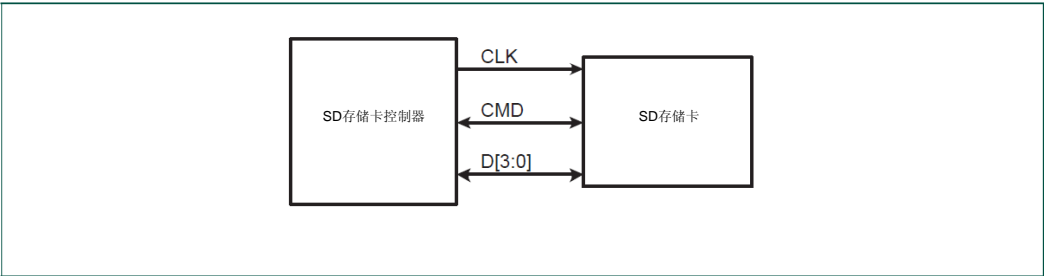
16.5 功能概述

SD 卡接口可以用作一个 SD 存储卡总线主机（参见 16.5.1 节），也可以用作多媒体卡总线主机（参见 16.5.2 节）。可以连接单个 SD 存储卡或最多 4 个多媒体卡（取决于系统板的负载能力）。

16.5.1 SD 存储卡

图 59 中显示 SD 存储卡的连接。

图59. SD 存储卡连接



16.5.1.1 SD 存储卡总线信号

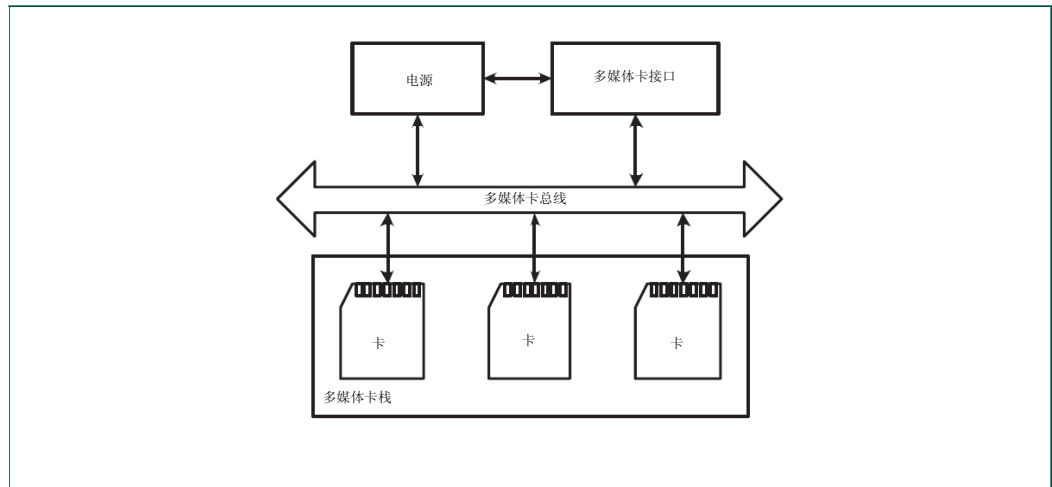
SD 存储卡总线使用下列信号：

- SD_CLK 主机至卡的时钟信号
- SD_CMD 双向命令/响应信号
- SD_DAT[3:0]双向数据信号

16.5.2 多媒体卡

图 60 中显示多媒体卡系统。

图60. 多媒体卡系统



多媒体卡按照其功能可分为三种类型：

- 只读存储（ROM）卡，其中包含预编程数据
- 读/写（R/W）卡，用于大容量存储
- 输入/输出（I/O）卡，用于通信

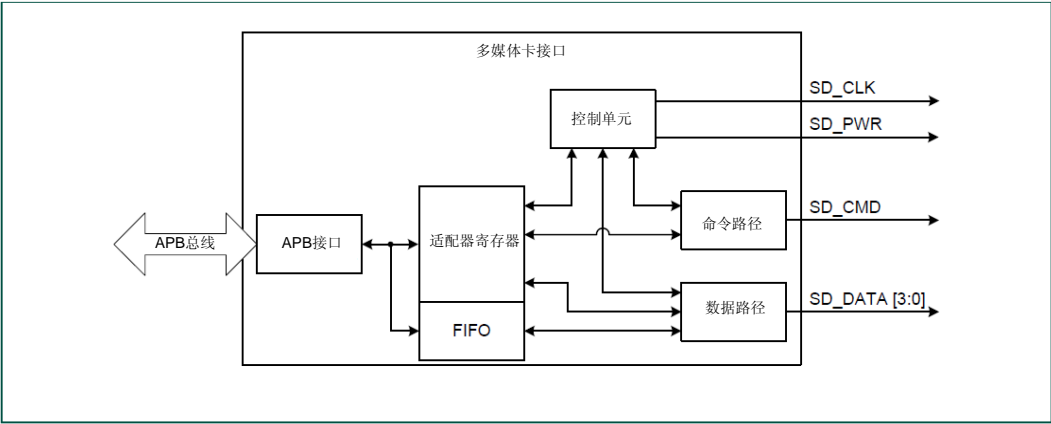
多媒体卡系统使用三条信号线来传输命令：

- CLK: 每个时钟周期同时通过命令线和数据线传输一个位。时钟频率在 0MHz 到 20MHz（多媒体卡）或 0MHz 到 25MHz（SD 存储卡）之间变化。
- CMD: 对卡进行初始化并传输命令的双向命令信道。CMD 有两种工作模式：
 - 初始化的漏极开路（Open-drain）模式
 - 命令传输的推挽（Push-pull）模式
- DAT: 双向数据信道，工作在推挽模式下

16.5.3 SD 卡接口详细信息

[图 61](#) 中显示简化的 SD 卡接口框图。

图61. SD 卡接口



SD 卡接口是一个 SD/多媒体存储卡总线主控，提供与一个多媒体卡堆或 SD 存储卡的接口。它包括五个子单元：

- 适配器寄存器模块
- 控制单元
- 命令路径
- 数据路径
- 数据 FIFO

16.5.3.1 适配器寄存器模块

适配器寄存器模块包括所有的系统寄存器。另外，该模块生成用于清除多媒体卡中静态标志的信号。将 1 写入 MCIClear 寄存器中的相应位时，就会生成清除信号。

16.5.3.2 控制单元

控制单元包括电源管理功能以及用于存储卡时钟的时钟分频器。

共有三个供电阶段：

- 电源关闭
- 上电
- 电源启动

电源管理逻辑控制着一个外部电源单元，在电源关闭或上电阶段，禁用卡总线的输出信号。上电阶段是电源关闭与电源启动阶段之间的转换阶段，使外部电源能够达到卡总线的工作电压。一个设备驱动程序用于确保在外部电源达到工作电压以前，使接口保持在上电阶段。

时钟管理逻辑生成并控制 SD_CLK 信号。SD_CLK 输出可使用时钟分频或时钟旁路模式。下列情况下时钟输出无效：

- 复位后
- 在电源关闭或上电阶段
- 如果使能了节电模式，而且卡总线处于 IDLE 状态（在命令和数据路径子单元进入 IDLE 阶段后的八个时钟周期）

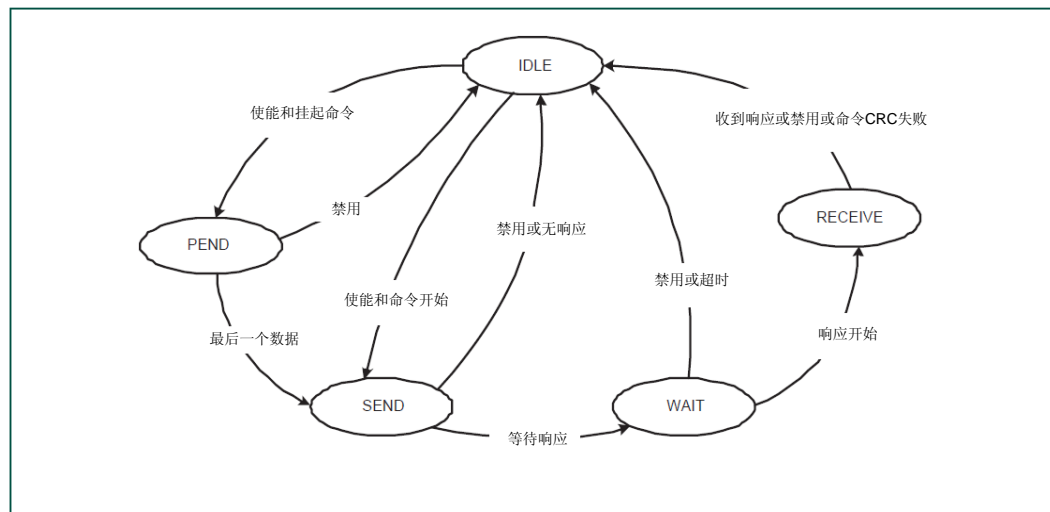
16.5.3.3 命令路径

命令路径子单元向卡发送命令，并从卡接收响应。

16.5.3.4 命令路径状态机

当命令寄存器被写入，并设置了使能位以后，命令传输开始。在传输命令时，如果不需要响应，则“命令路径状态机”（CPSM）设置状态标志，进入 IDLE 状态。如果需要响应，则状态机等待响应（见图 62）。收到响应时，会将收到的 CRC 代码与内部生成的代码进行比较，并设置适当的状态标志。

图62. 命令路径状态机



当进入等待（WAIT）状态时，命令定时器开始运行。如果在 CPSM 进入接收（RECEIVE）状态前就超时¹，则设置超时标志，进入空闲（IDLE）²状态。

如果在命令寄存器中设置了中断位，则定时器会被禁用，CPSM 等待其中某个卡的中断请求。如果在命令寄存器中设置了挂起位，CPSM 会进入挂起（PEND）状态，并等待数据路径子单元的 CmdPend 信号。检测到 CmdPend 时，CPSM 进入发送（SEND）状态。这会使命数据计数器，触发停止命令的传输。

¹ 超时期限为一个 64 个 SD_CLK 时钟周期的固定值。

² CPSM 在 IDLE 状态保持至少八个 SD_CLK 周期，以符合 Ncc 和 Nrc 时序约束。

图 63 中显示命令传输。

图63. 命令传输



16.5.3.5 命令格式

命令路径工作在半双工模式，因此既可以发送也可以接收命令和响应。如果 CPSM 未处于 SEND 状态，SD_CMD 输出处于 HI-Z 状态，如 图 63 中所示。则 SD_CMD 上的数据与 SD_CLK 上升沿同步。所有命令长度都是固定的 48 位。表 339 中显示命令格式。

表339. 命令格式

位位置	宽度	值	描述
0	1	1	停止位。
7:1	7	-	CRC7
39:8	32	-	参数。
45:40	6	-	命令索引。
46	1	1	发送位。
47	1	0	起始位。

SD 卡接口支持两种响应类型。都采用 CRC 错误校验：

- 48 位短响应（见 表 340）
- 136 位长响应（见 表 341）

注：如果响应不包含 CRC（CMD1 响应），则设备驱动程序必须忽略 CRC 失败状态。

表340. 简单响应格式

位位置	宽度	值	描述
0	1	1	停止位。
7:1	7	-	CRC7（或 1111111）。
39:8	32	-	参数。
45:40	6	-	命令索引。
46	1	0	发送位。
47	1	0	起始位。

表341. 长响应格式

位位置	宽度	值	描述
0	1	1	停止位。
127:1	127	-	CID 或 CSD（包含内部 CRC7）。
133:128	6	111111	保留。
134	1	1	发送位。
135	1	0	起始位。

命令寄存器中包含了命令索引（发送到卡的六个位）和命令类型。它们决定了命令是否需要响应，以及响应长度是 48 位还是 136 位（有关详细信息，参见 16.6.4 节）。命令路径实现了表 342 中所所示的状态标志（有关详细信息，参见 16.6.11 节）。

表342. 命令路径状态标志

标志	描述
CmdRespEnd	若响应 CRC 为 OK，则该标志置位。
CmdCrcFail	若响应 CRC 失败，则该标志置位。
CmdSent	当命令（不需要响应）发出时，该标志置位。
CmdTimeOut	响应超时。
CmdActive	命令传输正在进行中。

CRC 发生器对 CRC 代码之前所有位，计算 CRC 校验和。这包括起始位、发射器位、命令索引，和命令参数（或卡状态）。对于长响应格式，会计算 CID 或 CSD 的前 120 位的 CRC 校验和。请注意，计算 CRC 时不使用起始位、发射器位和六个保留位。

CRC 校验和是一个 7 位的值：

CRC[6:0] = 余数 [(M(x) × x₇)/G(x)]

G(x) = x₇ + x₃ + 1

M(x) = （起始位） × x₃₉ + ... + （CRC 前最后一位） × x₀，或者

M(x) = （起始位） × x₁₁₉ + ... + （CRC 前最后一位） × x₀

16.5.3.6 数据路径

使用时钟控制寄存器可以编程设定卡的数据总线宽度。如果使能了宽总线模式，会在所有四个数据信号上，以每个时钟周期四个位的方式传输数据（SD_DAT[3:0]）。如果未使能宽总线模式，则每个时钟周期仅传输一个位 SD_DAT[0]。

根据传输方向（发送或接收），“数据路径状态机”（DPSM）在使能后会进入 WAIT_S 或 WAIT_R 状态：

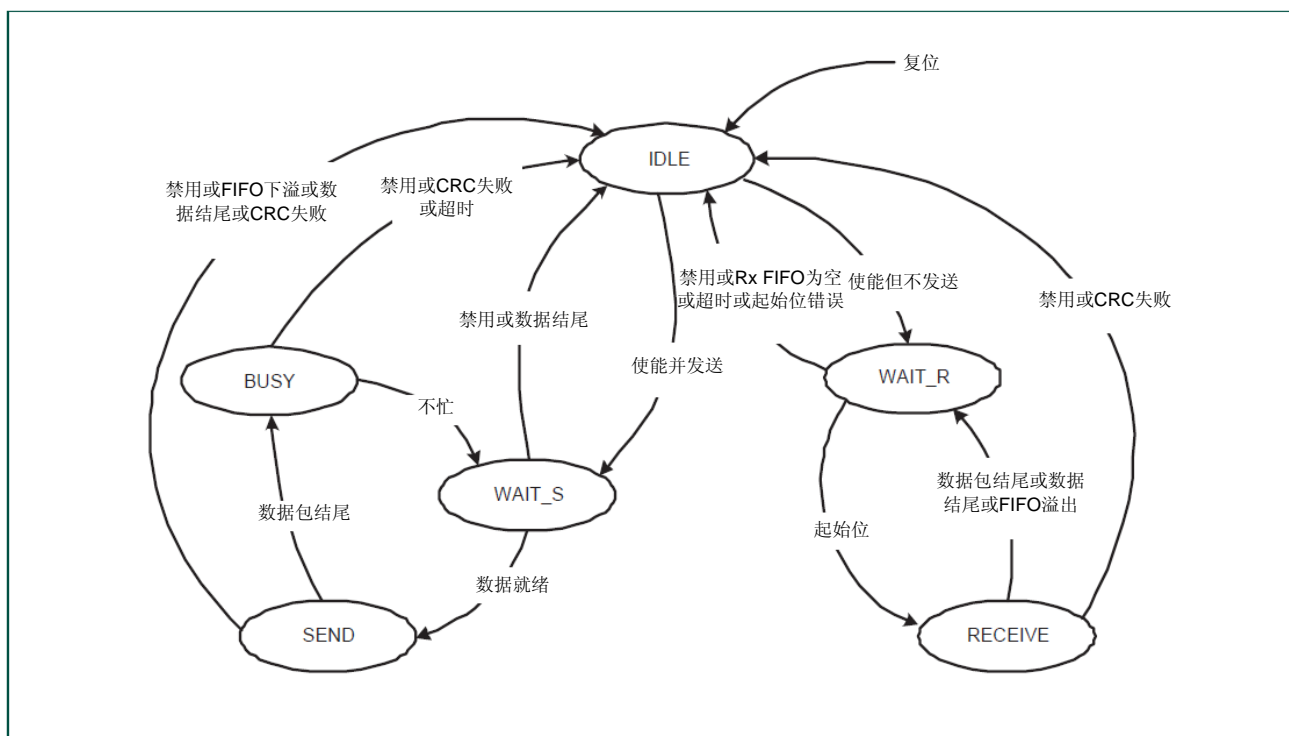
- 发送：DPSM 进入 WAIT_S 状态。如果发送 FIFO 中有数据，则 DPSM 进入 SEND 状态，且数据路径子单元会开始向卡发送数据。

- 接收: DPSM 进入 WAIT_R 状态, 并等待一个起始位。当接收到起始位时, DPSM 进入 RECEIVE 状态, 且数据路径子单元开始从卡接收数据。

16.5.3.7 数据路径状态机

DPSM 工作在 SD_CLK 频率下。卡总线信号中的数据与 SD_CLK 的上升沿同步。DPSM 有六种状态, 如图 64 中所示。

图64. 数据路径状态机



- IDLE**: 数据路径未激活, 且 SD_DAT[3:0]输出为 HI-Z。当写入数据控制寄存器并设置使能位后, DPSM 在数据计数器中加载新值, 然后根据数据方向位, 进入 WAIT_S 或 WAIT_R 状态。

WAIT_R: 如果数据计数器为 0, 则当接收 FIFO 为空时, DPSM 进入 IDLE 状态。如果数据计数器不为 0, 则 DPSM 会等待 SD_DAT 上的一个起始位。

如果 DPSM 在超时之前接收到了一个起始位, 则它进入 RECEIVE 状态, 并加载数据模块计数器。如果它在检测到起始位之前超时, 或发生了一个起始位错误, 则它进入 IDLE 状态并设置超时状态标志。

- RECEIVE**: 从卡接收到的串行数据组成字节, 并写入数据 FIFO 中。根据数据控制寄存器中的传输模式位, 数据传输模式可以是块或流:
 - 在块模式下, 当数据块计数器达到零时, DPSM 会一直等待, 直至接收到 CRC 代码。如果接收到的代码与内部生成的 CRC 代码匹配, 则 DPSM 进入 WAIT_R 状态。如果不匹配, 则设置 CRC 失败状态标志, 且 DPSM 进入 IDLE 状态。

- 在流模式下，DPSM 会在数据计数器不为零时接收数据。当计数器为零时，移位寄存器中的剩余数据会写入数据 FIFO，且 DPSM 进入 WAIT_R 状态。

如果发生 FIFO 溢出错误，DPSM 会设置 FIFO 错误标志并进入 WAIT_R 状态。

- WAIT_S: 如果数据计数器为零，DPSM 进入 IDLE 状态。如果计数器不为零，则它会等待到数据 FIFO 空的标志无效时，进入 SEND 状态。

注：DPSM 会继续保持在 WAIT_S 状态至少两个时钟周期，以满足 Nwr 时序约束。

- SEND: DPSM 开始发送数据到卡。根据数据控制寄存器中的传输模式位，数据传输模式可以是块或流：
 - 在块模式下，当数据块计数器达到零时，DPSM 会发送一个内部生成的 CRC 代码和停止位，并进入 BUSY 状态。
 - 在流模式下，当使能位为高电平（HIGH）且数据计数器不为零时，DPSM 发送数据到卡。然后，它会进入 IDLE 状态。

如果发生 FIFO 下溢出错误，DPSM 会设置 FIFO 错误标志并进入 IDLE 状态。

- BUSY: DPSM 等待 CRC 状态标志：
 - 如果它没有接收到正 CRC 状态，则进入 IDLE 状态并设置 CRC 失败状态标志。
 - 如果它接收到一个正 CRC 状态，如果 SD_DAT[0]不为低电平（卡不处于忙状态）时进入 WAIT_S 状态。

如果在 DPSM 处于 BUSY 状态时发生一个超时，则它会设置数据超时标志，并进入 IDLE 状态。

当 DPSM 处于 WAIT_R 或 BUSY 状态时，数据定时器被使能，生成数据超时错误：

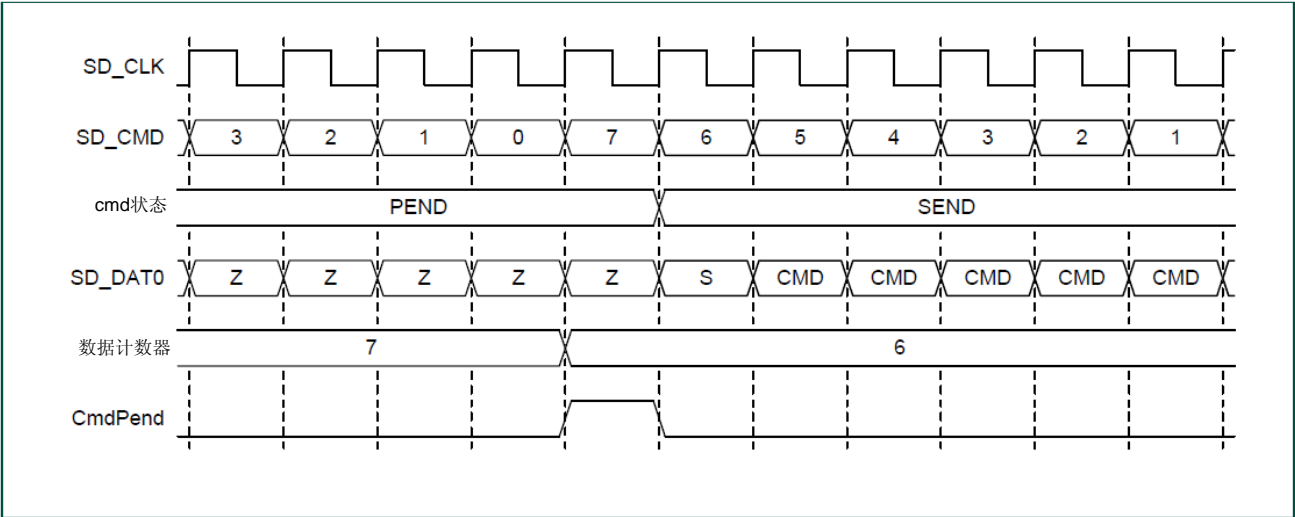
- 在发送数据时，如果 DPSM 在 BUSY 状态的时间超过了设定的超时期限，则发生超时。
- 在接收数据时，如果数据结尾不为真，且 DPSM 在 WAIT_R 状态的时间超过了设定的超时期限，则发生超时。

16.5.3.8 数据计数器

数据计数器有两个功能：

- 当值达到零时，停止数据传输。这是数据终止的条件。
- 开始传输挂起的命令（参见图 65）。该功能用于发送一个针对流数据传输的停止命令。

图65. 挂起命令启动



数据块计数器决定了一个数据块的结尾。如果计数器为 0，则数据终止条件为 TRUE（有关详细信息，参见 16.6.9 节）。

16.5.3.9 总线模式

在宽总线模式下，全部四个数据信号（SD_DAT[3:0]）都用于传输数据，并且为每个数据信号单独计算 CRC 代码。将数据块发送到卡时，仅将 SD_DAT[0]用于 CRC 令牌和忙信号。起始位必须（在同一时钟周期内）同时发送给全部四个数据信号。如果接收数据时，在同一时钟沿处没有对全部数据信号检测到起始位，则 DPSM 设置起始位错误标志并进入 IDLE 状态。

数据路径同样工作在半双工模式，数据可以发送到卡，也可以从卡接收数据。当未传输时，SD_DAT[3:0]处于 HI-Z 状态。

这些信号中的数据与时钟周期的上升沿同步。

如果选择标准总线模式，则 SD_DAT[3:1]输出始终处于 HI-Z 状态，而且发送数据时仅 SD_DAT[0]输出被驱动为低电平。

设计说明：如果选择宽总线模式，则所有数据输出要同时使能。如果未选择宽总线模式，则 SD_DAT[3:1]输出始终是关闭的，而且传输数据时仅 SD_DAT[0]输出被驱动为低电平。

16.5.3.10 CRC 令牌状态

CRC 令牌状态与各个写入数据块一致，并决定一个卡是否正确地接收了数据块。接收到令牌后，卡会通过将 SD_DAT[0]驱动为低电平，使忙信号有效。表 343 中显示 CRC 令牌状态的值。

表343. CRC 令牌状态

令牌	描述
010	卡已接收无错误数据块。
101	卡已检测到 CRC 错误。

16.5.3.11 状态标志

表 344 中列出数据路径的状态标志（有关详细信息，参见 16.6.11 节）。

表344. 数据路径状态标志

标志	描述
TxFifoFull	发送 FIFO 为满。
TxFifoEmpty	发送 FIFO 为空。
TxFifoHalfEmpty	发送 FIFO 为半满。
TxDataAvlbl	发送 FIFO 数据可用。
TxUnderrun	发送 FIFO 下溢错误。
RxFifoFull	接收 FIFO 为满。
RxFifoEmpty	接收 FIFO 为空。
RxFifoHalfFull	接收 FIFO 为半满。
RxDataAvlbl	接收 FIFO 数据可用。
RxOverrun	接收 FIFO 溢出错误。
DataBlockEnd	数据块发送/接收。
StartBitErr	在宽总线模式下，所有数据信号上未检测到起始位。
DataCrcFail	数据包 CRC 失败。
DataEnd	数据终止（数据计数器为 0）。
DataTimeOut	数据超时。
TxActive	命令发送正在进行中。
RxActive	命令接收正在进行中。

16.5.3.12 CRC 发生器

CRC 发生器仅对单个块中的数据位计算 CRC 校验和，在数据流模式下会被忽略。校验和是一个 16 位的值：

CRC[15:0] = 余数 [(M(x) × x¹⁵)/G(x)]

G(x) = x¹⁶ + x¹² + x⁵ + 1

M(x) - （第一个数据位） × xⁿ + ... + （最后数据位） × x⁰

16.5.3.13 数据 FIFO

数据 FIFO（先入先出）子单元是带有发送和接收逻辑的数据缓冲区。

FIFO 包含一个 32 位宽、16 字深度的数据缓冲区，以及发送和接收逻辑。因为数据 FIFO 在 APB 时钟域（PCLK）中运行，所以，所有来自 SD 卡接口时钟域（MCLK）中子单元的信号都要重新同步。

根据 TxActive 和 RxActive，可以禁用 FIFO，使能发送或接收。TxActive 和 RxActive 由数据路径子单元驱动，而且二者互斥：

- 当 TxActive 有效时，发送 FIFO 是作为发送逻辑和数据缓冲区（参见 16.5.3.14 节）
- 当 RxActive 有效时，接收 FIFO 是作为接收逻辑和数据缓冲区（参见 16.5.3.15 节）

16.5.3.14 发送 FIFO

SD 卡接口使能发送后，就可以通过 APB 接口将数据写入发送 FIFO。

发送 FIFO 的访问方式是通过 16 个连续的地址（参见 16.6.15 节）。发送 FIFO 包含一个数据输出寄存器，其中存储着读取指针所指向的数据字。当数据路径子单元加载其移位寄存器时，它会使读取指针值加 1，并给出新的数据。

如果发送 FIFO 被禁用，所有状态标志都无效。数据路径子单元在发送数据时会使 TxActive 有效。表 345 中列出发送 FIFO 的状态标志。

表345. 发送 FIFO 状态标志

标志	描述
TxFifoFull	当所有 16 个发送 FIFO 字包含有效数据时，设为高电平。
TxFifoEmpty	当所有发送 FIFO 不包含有效数据时，设为高电平。
TxHalfEmpty	当 8 个或 8 个以上的发送 FIFO 字为空时，设为高电平。这一标志可用作 DMA 请求。
TxDataAvlbl	当发送 FIFO 包含有效数据时，设为高电平。这一标志和 TxFifoEmpty 标志相反。
TxUnderrun	当下溢错误发生时，设为高电平。通过对 MCIClear 寄存器进行写操作将该标志清零。

16.5.3.15 接收 FIFO

在数据路径子单元接收到一个数据字时，它会将数据驱动在写入数据总线上，并使写入使能信号有效。该信号与 PCLK 域同步。写入完成后，写入指针加 1，接收 FIFO 的控制逻辑使 RxWrDone 有效，然后使写入使能信号无效。

在读取端，读取指针的当前值所指向的 FIFO 字内容会被驱动到读取数据总线上。当 APB 总线接口使 RxRdPrtInc 有效时，读取指针加 1。

如果接收 FIFO 被禁用，所有状态标志均为无效，且读取和写入指针被复位。数据路径子单元在接收数据时使 RxActive 有效。表 353 中列出接收 FIFO 的状态标志。

接收 FIFO 的访问方式是通过 16 个连续的地址（参见 16.6.15 节）。

如果接收 FIFO 被禁用，所有状态标志均无效，且读取和写入指针被复位。数据路径子单元在接收数据时使 RxActive 有效。表 346 中列出接收 FIFO 的状态标志。

表346. 接收 FIFO 状态标志

符号	描述
RxFifoFull	当所有 16 个接收 FIFO 字包含有效数据时，设为高电平。
RxFifoEmpty	当接收 FIFO 不包含有效数据时，设为高电平。

符号	描述
RxHalfFull	当所有 8 个或 8 个以上的接收 FIFO 字包含有效数据时，设为高电平。这一标志可用作 DMA 请求。
RxDataAvlbl	当接收 FIFO 不为空时，设为高电平。这一标志和 RxFifoEmpty 标志相反。
RxOverrun	当溢出错误发生时，设为高电平。通过对 MCIClear 寄存器进行写操作将该标志清零。

16.5.3.16 APB 接口

APB 接口生成中断和 DMA 请求，并访问 SD 卡接口寄存器和数据 FIFO。该接口包括一个数据路径、寄存器解码器，以及中断/DMA 逻辑。DMA 由通用 DMA 控制器进行控制，有关详细信息，参见对应章节。

16.5.3.17 中断逻辑

中断逻辑生成一个中断请求信号。当至少有一个所选的状态标志为高电平时，该信号有效。一个屏蔽寄存器用于选择生成某个中断的条件。如果设置了对应的屏蔽标志，则状态标志会生成中断请求。

16.6 寄存器说明

SD 卡接口的寄存器如[表 347](#) 中所示。

表347. 卡接口寄存器汇总

名称	描述	访问	宽度	复位值 ^[1]	地址	表
MCIPower	电源控制寄存器	R/W	8	0	0x400C 0000	表 348
MCIClock	时钟控制寄存器	R/W	12	0	0x400C 0004	表 349
MCIArgument	参数寄存器	R/W	32	0	0x400C 0008	表 350
MMCCommand	命令寄存器	R/W	11	0	0x400C 000C	表 351
MCIRespCmd	响应命令寄存器	RO	6	0	0x400C 0010	表 353
MCIResponse0	响应寄存器	RO	32	0	0x400C 0014	表 354
MCIResponse1	响应寄存器	RO	32	0	0x400C 0018	表 354
MCIResponse2	响应寄存器	RO	32	0	0x400C 001C	表 354
MCIResponse3	响应寄存器	RO	31	0	0x400C 0020	表 354
MCIDataTimer	数据定时器	R/W	32	0	0x400C 0024	表 356
MCIDataLength	数据长度寄存器	R/W	16	0	0x400C 0028	表 357
MCIDataCtrl	数据控制寄存器	R/W	8	0	0x400C 002C	表 358
MCIDataCnt	数据计数器	RO	16	0	0x400C 0030	表 360
MCIStatus	状态寄存器	RO	22	0	0x400C 0034	表 361
MCIClear	清零寄存器	WO	11	-	0x400C 0038	表 362
MCIMask0	中断 0 屏蔽寄存器	R/W	22	0	0x400C 003C	表 363
MCI FifoCnt	FIFO 计数器	RO	15	0	0x400C 0048	表 364
MCI FIFO	数据 FIFO 寄存器	R/W	32	0	0x400C 0080 to 0x400C 00BC	表 365

[1] 复位值仅反映已使用位中保存的数据，它不包括保留位的内容。

16.6.1 电源控制寄存器（MCIPWR—0x400C 0000）

MCIPWR 寄存器控制外部电源。可以将电源启动和关闭，还可以调节输出电压。[表 348](#) 中显示 MCIPower 寄存器的位分配情况。

用 SCS 寄存器的第 3 位可以选择 SD_PWR 管脚的有效电平（有关详细信息，参见 [3.8.1](#) 节）。

表348. 电源控制寄存器（MCIPWR—0x400C 0000）位描述

位	符号	值	描述	复位值
1:0	Ctrl	00	电源关闭	00
		01	保留	
		10	上电	
		11	电源启动	
5:2	-		保留。读取值未定义，只写入 0。	无
6	OpenDrain		SD_CMD 输出控制。	0
7	Rod		棒控制。	0
31:8	-		保留。读取值未定义，只写入 0。	无

在外部电源启动时，软件首先进入上电阶段，等到电源输出稳定后，再进入电源启动阶段。在上电期间，SD_PWR 设置为高电平。在上述两个阶段中，卡总线插槽都被禁用。

注：一次数据写入后，在三个 MCLK 时钟周期加两个 PCLK 时钟周期内，数据无法写入到该寄存器中。

16.6.2 时钟控制寄存器（MCIClock—0x400C 0004）

MCIClock 寄存器控制 SD_CLK 输出。[表 349](#) 中显示时钟控制寄存器的位分配情况。

表349. 时钟控制寄存器（MCIClock—0x400C 0004）位描述

位	符号	值	描述	复位值
7:0	ClkDiv		总线时钟周期： SD_CLK 频率 = MCLK/[2x(ClkDiv+1)]	0
8	Enable		使能 SD 卡总线时钟：	0
		0	时钟禁能。	
		1	时钟使能。	
9	PwrSave		当总线处于空闲状态时，禁止 SD_CLK 输出：	0
		0	总是使能。	
		1	当总线被激活时，使能时钟。	
10	Bypass		使能时钟分频逻辑旁路：	0
		0	禁能旁路。	
		1	使能旁路。MCLK 驱动至卡总线输出（SD_CLK）。	
11	WideBus		使能宽总线模式：	0
		0	标准总线模式（仅使用 SD_DAT[0]）。	
		1	宽总线模式（使用 SD_DAT[3:0]）。	
31:12	-		保留。读取值未定义，只写入 0。	无

当 SD 卡接口处于标识模式时，SD_CLK 的频率必须低于 400kHz。当相关卡地址被分配给了所有卡时，可以将时钟频率更改为最大卡总线频率。

注：一次数据写入后，在三个 MCLK 时钟周期加两个 PCLK 时钟周期内，数据无法写入到该寄存器中。

16.6.3 参数寄存器（MCIArgment—0x400C 0008）

MCIArgment 寄存器包含一个 32 位的命令参数，该参数作为命令消息的一部分发送到卡。
[表 350](#) 中显示 MCIArgment 寄存器的位分配情况。

表350. 参数寄存器（MCIArgment—0x400C 0008）位描述

位	符号	描述	复位值
31:0	CmdArg	命令参数	0x0000 0000

如果命令中包含参数，则必须将其装入到参数寄存器中，然后才能将命令写入命令寄存器。

16.6.4 命令寄存器（MCICommand—0x400C 000C）

MCICommand 寄存器包含命令索引和命令类型位：

- 命令索引作为命令消息的一部分发送到卡。
- 命令类型位控制着“命令路径状态机”（CPSM）。将 1 写入使能位可以启动命令发送操作，而清除该位会禁用 CPSM。

[表 351](#) 中显示 MCICommand 寄存器的位分配情况。

表351. 命令寄存器（MCICommand—0x400C 000C）位描述

位	符号	描述	复位值
5:0	CmdIndex	命令索引。	0
6	Response	若该位置位，CPSM 等待响应。	0
7	LongRsp	若该位置位，CPSM 接收一个 136 位长响应。	0
8	Interrupt	若该位置位，CPSM 禁用命令定时器并等待中断请求。	0
9	Pending	若该位置位，CPSM 在开始发送命令前等待 CmdPend。	0
10	Enable	若该位置位，CPSM 被使能。	0
31:11	-	保留。读取值未定义，只写入 0。	无

注：一次数据写入，在三个 MCLK 时钟周期加两个 PCLK 时钟周期内，数据无法写入到该寄存器中。

[表 352](#) 中给出响应类型。

表352. 命令响应类型

响应	长响应	描述
0	0	无响应，期待出现 CmdSent 标志。
0	1	无响应，期待出现 CmdSent 标志。
1	0	短响应，期待出现 CmdRespEnd 或 CmdCrcFail 标志。
1	1	长响应，期待出现 CmdRespEnd 或 CmdCrcFail 标志。

16.6.5 命令响应寄存器（MCIRspCommand—0x400C 0010）

MCIRspCommand 寄存器包含了所收到最后命令响应的命令索引字段。[表 351](#) 中显示 MCIRspCommand 寄存器的位分配情况。

表353. 命令响应寄存器（MCIRspCommand—0x400C 0010）位描述

位	符号	描述	复位值
5:0	RespCmd	响应命令索引	0
31:6	-	保留。读取值未定义，只写入 0。	无

如果命令响应发送中不包含命令索引字段（长响应），RespCmd 字段是未知的，不过它必须包含 11111（响应的保留字段值）。

16.6.6 响应寄存器（MCIResponse0-3—0x400C 0014、0x400C 0018、0x400C 001C 和 0x400C 0020）

MCIResponse0-3 寄存器包含卡的状态，这是接收响应的一部分。[表 354](#) 中显示 MCIResponse0-3 寄存器的位分配情况。

表354. 响应寄存器（MCIResponse0-3—0x400C 0014，0x400C 0018，0x400C 001C 和 0x400C 0020）位描述

位	符号	描述	复位值
31:0	Status	卡状态	0

卡状态的长度可以是 32 位或 127 位，取决于响应类型（参见[表 355](#)）。

表355. 响应寄存器类型

描述	短响应	长响应
MCIResponse0	卡状态[31:0]	卡状态[127:96]
MCIResponse1	未使用	卡状态[95:64]
MCIResponse2	未使用	卡状态[63:32]
MCIResponse3	未使用	卡状态[31:1]

首先接收的是卡状态的最高有效位。MCIResponse3 寄存器中的 LSBit 始终为 0。

16.6.7 数据定时器寄存器（MCIDataTimer—0x400C 0024）

MCIDataTimer 寄存器包含了数据超时周期，在卡总线的时钟周期。[表 356](#) 中显示 MCIDataTimer 寄存器的位分配情况。

表356. 数据定时器寄存器（MCIDataTimer—0x400C 0024）位描述

位	符号	描述	复位值
31:0	DataTime	数据超时周期。	0

计数器从数据定时器寄存器中加载值，并在“数据路径状态机”（DPSM）进入 WAIT_R 或

BUSY 状态时开始递减。如果定时器达到 0 时，DPSM 处于上述任一状态，则会设置超时状态标志。

数据传输必须被写入数据定时器寄存器和数据长度寄存器，然后才能写入数据控制寄存器。

16.6.8 数据长度寄存器（MCIDataLength—0x400C 0028）

MCIDataLength 寄存器包含要传输的数据字节数。当数据传输开始时，该值会加载到数据计数器中。[表 357](#) 中显示 MCIDataLength 寄存器的位分配情况。

表357. 数据长度寄存器（MCIDataLength—0x400C 0028）位描述

位	符号	描述	复位值
15:0	DataLength	数据长度值	0
31:16	-	保留。读取值未定义，只写入 0。	无

对于块数据传输，数据长度寄存器中的值必须是块大小的倍数（参见 [16.6.9](#) 节）。

如需启动数据传输，要对数据定时器寄存器和数据长度寄存器进行写操作，然后对数据控制寄存器进行写操作。

16.6.9 数据控制寄存器（MCIDataCtrl—0x400C 002C）

MCIDataCtrl 寄存器控制着 DPSM。[表 358](#) 中显示 MCIDataCtrl 寄存器的位分配情况。

表358. 数据控制寄存器（MCIDataCtrl—0x400C 002C）位描述

位	符号	符号	描述	复位值
0	Enable		数据传输使能。	0
1	Direction		数据传输方向：	0
		0	从控制器到卡。	
		1	从卡到控制器。	
2	Mode		数据传输模式：	0
		0	块数据传输	
		1	流数据传输	
3	DMAEnable		使能 DMA：	0
		0	DMA 禁能。	
		1	DMA 使能。	
7:4	BlockSize		数据块长度	0
31:8	-		保留。读取值未定义，只写入 0。	无

注：在一次数据写操作完成后，该寄存器在 3 个 MCLK 时钟周期加上 2 个 PCLK 时钟周期的时间内无法写入数据。

如果将 1 写入使能位，则数据传输开始。根据方向位，DPSM 可进入 WAIT_S 或 WAIT_R 状态。在数据传输开始后，不需要清除使能位。如果 Mode 位为 0，BlockSize 控制数据块的长度，如[表 359](#) 中所示。

表359. 数据块长度

块大小	块长度
0	2 ⁰ =1 字节。
1	2 ¹ =2 字节。
...	-
11	2 ¹¹ =2048 字节。
12:15	保留。

16.6.10 数据计数器寄存器（MCIDataCnt—0x400C 0030）

在 DPSM 从 IDLE 状态进入 WAIT_R 或 WAIT_S 状态时，MCIDataCnt 寄存器从数据长度寄存器中加载值（参见 16.6.8 节“数据长度寄存器（MCIDataLength—0x400C 0028）”）。在传输数据的过程中，计数器的值会递减，直至为 0。然后，DPSM 进入 IDLE 状态，并设置数据状态结束标志。表 360 中显示 MCIDataCnt 寄存器的位分配情况。

表360. 数据计数器寄存器（MCIDataCnt—0x400C 0030）位描述

位	符号	描述	复位值
15:0	DataCount	剩余的数据	0
31:16	-	保留。读取值未定义，只写入 0。	无

注：该寄存器仅在数据传输完成的情况下可读。

16.6.11 状态寄存器（MCIStatus—0x400C 0034）

MCIStatus 寄存器是一个只读寄存器。它包含两种类型的标志：

- 静态[10:0]：这些标志会保持为有效，直至通过写入清除寄存器将它们清除（参见 16.6.12 节）。
- 动态[21:11]：这些更改状态取决于底层逻辑的状态（例如，在写入 FIFO 时 FIFO 满和空标志会作为数据被设为有效和无效）。

表 361 中显示 MCIStatus 寄存器的位分配情况。

表361. 状态寄存器（MCIStatus—0x400C 0034）位描述

位	符号	描述	复位值
0	CmdCrcFail	命令响应接收（未通过 CRC 校验）。	0
1	DataCrcFail	数据块发送/接收（未通过 CRC 校验）。	0
2	CmdTimeOut	命令响应超时。	0
3	DataTimeOut	数据超时。	0
4	TxUnderrun	发送 FIFO 下溢错误。	0
5	RxOverrun	接收 FIFO 溢出错误。	0
6	CmdRespEnd	命令响应接收（通过 CRC 校验）。	0
7	CmdSent	命令发送（无需响应）。	0
8	DataEnd	数据终止（数据计数器为 0）。	0

位	符号	描述	复位值
9	StartBitErr	在宽总线模式下，所有数据信号上未检测到起始位。	0
10	DataBlockEnd	数据块发送/接收（通过 CRC 校验）。	0
11	CmdActive	命令传输正在进行中。	0
12	TxActive	数据发送正在进行中。	0
13	RxActive	数据接收正在进行中。	0
14	TxFifoHalfEmpty	发送 FIFO 为半空。	0
15	RxFifoHalfFull	接收 FIFO 为半满。	0
16	TxFifoFull	发送 FIFO 为满。	0
17	RxFifoFull	接收 FIFO 为满。	0
18	TxFifoEmpty	发送 FIFO 为空。	0
19	RxFifoEmpty	接收 FIFO 为空。	0
20	TxDataAvlbl	发送 FIFO 中存在可用数据。	0
21	RxDataAvlbl	接收 FIFO 中存在可用数据。	0
31:22	-	保留。从保留位读出的值未定义。	无

16.6.12 清零寄存器（MCIClear—0x400C 0038）

MCIClear 寄存器是一个只写寄存器。向寄存器中的相应位写入 1，可清零相应的静态状态标志。[表 362](#) 中显示 MCIClear 寄存器的位分配情况。

表362. 清零寄存器（MCIClear—0x400C 0038）位描述

位	符号	描述	复位值
0	CmdCrcFailClr	清零 CmdCrcFail 标志。	-
1	DataCrcFailClr	清零 DataCrcFail 标志。	-
2	CmdTimeOutClr	清零 CmdTimeOut 标志。	-
3	DataTimeOutClr	清零 DataTimeOut 标志。	-
4	TxUnderrunClr	清零 TxUnderrun 标志。	-
5	RxOverrunClr	清零 RxOverrun 标志。	-
6	CmdRespEndClr	清零 CmdRespEnd 标志。	-
7	CmdSentClr	清零 CmdSent 标志。	-
8	DataEndClr	清零 DataEnd 标志。	-
9	StartBitErrClr	清零 StartBitErr 标志。	-
10	DataBlockEndClr	清零 DataBlockEnd 标志。	-
31:11	-	保留。读取值未定义，只写入 0。	无

16.6.13 中断屏蔽寄存器（MCIMask0—0x400C 003C）

中断屏蔽寄存器通过将相应位设置为 1，决定哪些状态标志生成中断请求。[表 363](#) 中显示 MCIMaskx 寄存器的位分配情况。

表363. 中断屏蔽寄存器（MCIMask0—0x400C 003C）位描述

位	符号	描述	复位值
0	Mask0	屏蔽 CmdCrcFail 标志。	0
1	Mask1	屏蔽 DataCrcFail 标志。	0
2	Mask2	屏蔽 CmdTimeOut 标志。	0
3	Mask3	屏蔽 DataTimeOut 标志。	0
4	Mask4	屏蔽 TxUnderrun 标志。	0

位	符号	描述	复位值
5	Mask5	屏蔽 RxOverrun 标志。	0
6	Mask6	屏蔽 CmdRespEnd 标志。	0
7	Mask7	屏蔽 CmdSent 标志。	0
8	Mask8	屏蔽 DataEnd 标志。	0
9	Mask9	屏蔽 StartBitErr 标志。	0
10	Mask10	屏蔽 DataBlockEnd 标志。	0
11	Mask11	屏蔽 CmdActive 标志。	0
12	Mask12	屏蔽 TxActive 标志。	0
13	Mask13	屏蔽 RxActive 标志。	0
14	Mask14	屏蔽 TxFifoHalfEmpty 标志。	0
15	Mask15	屏蔽 RxFifoHalfFull 标志。	0
16	Mask16	屏蔽 TxFifoFull 标志。	0
17	Mask17	屏蔽 RxFifoFull 标志。	0
18	Mask18	屏蔽 TxFifoEmpty 标志。	0
19	Mask19	屏蔽 RxFifoEmpty 标志。	0
20	Mask20	屏蔽 TxDataAvlbl 标志。	0
21	Mask21	屏蔽 RxDataAvlbl 标志。	0
31:22 -		保留。读取值未定义，只写入 0。	无

16.6.14 FIFO 计数器寄存器（MCIFifoCnt—0x400C 0048）

MCIFifoCnt 寄存器包含要写入 FIFO 或要从 FIFO 读取的剩余字数。当数据控制寄存器中设置了使能位时，FIFO 计数器会从数据长度寄存器中加载值（参见 16.6.8 节）。如果数据长度不是字对齐（4 的倍数），则将剩余的 1 到 3 个字节被当作一个字。表 364 中显示 MCIFifoCnt 寄存器的位分配情况。

表364. FIFO 计数器寄存器（MCIFifoCnt—0x400C 0048）位描述

位	符号	描述	复位值
14:0	DataCount	剩余的数据	0
31:15 -		保留。读取值未定义，只写入 0。	无

16.6.15 数据 FIFO 寄存器（MCIFIFO—0x400C 0080 至 0x400C 00BC）

接收和发送 FIFO 可以作为 32 位宽寄存器进行读写操作。FIFO 包含在 16 个顺序地址上的 16 个入口。这样，微处理器就可以用它加载和存储多个操作数，以读取/写入 FIFO。表 365 中显示 MCIFIFO 寄存器的位分配情况。

表365. 数据 FIFO 寄存器（MCIFIFO—0x400C 0080 至 0x400C 00BC）位描述

位	符号	描述	复位值
31:0	Data	FIFO 数据。	0

17.1 基本配置

UART1 外设的配置需要使用下列寄存器：

1. 功率：在 PCONP 寄存器（[表 37](#)）中，设置 PCUART1 位。
注：复位时，UART1 会被使能（PCUART1=1）。
2. 外设时钟：UART1 使用通用 PCLK，它既用于总线接口，也用于大多数 APB 外设的功能部分。参见 [4.6.4](#) 节。
3. 波特率：在 U1LCR 寄存器（[表 376](#)）中，将位 DLAB 设置为 1。由此可启用对寄存器 DLL（[表 370](#)）和 DLM（[表 371](#)）的访问，以设置波特率。如果需要，还可以在分数分频寄存器（[表 383](#)）中设置分数波特率。
4. UART FIFO：使用 U1FCR 寄存器（[表 375](#)）中的 FIFO 使能位（位 0）使能 FIFO。
5. 管脚：通过相关的 IOCON 寄存器（[8.4.1](#) 节）选择 UART 管脚和管脚模式。
注：不应为 UART 接收管脚启用下拉电阻。
6. 中断：要使能 UART 中断，将 U1LCR 寄存器（[表 376](#)）中的位 DLAB 设置为 0。由此可启用对 U1IER（[表 372](#)）的访问。中断的使能是通过在 NVIC 中使用相应的中断设置使能寄存器来实现的。
7. DMA：UART1 的发送和接收功能可通过 GPDMA 控制器进行操作（参见[表 664](#)）。

17.2 特性

- 完全 Modem 控制握手信号。
- 数据大小为 5、6、7 和 8 位。
- 奇偶生成和校验：奇校验（Odd）、偶校验（Even）、1 校验（Mark）、0 校验（Space）或无校验（None）。
- 一个或两个停止位。
- 16 字节收发 FIFO。
- 内置波特率发生器，包含一个多功能性的分数波特率分频器。
- 支持 DMA 发送和接收。
- 自动波特率功能。
- 中断生成和检测。
- 多处理器寻址模式。
- 支持 RS-485/EIA-485。

17.3 结构

UART1 的结构如下面的框图所示。

APB 接口提供 CPU 或主机与 UART1 之间的通信连接。

UART1 接收器模块 (U1RX) 监控串行输入线路 RXD1 的有效输入。UART1 RX 移位寄存器 (U1RSR) 通过 RXD1 接收有效字符。当 U1RSR 汇编完一个有效字符时，它将该字符传送到 UART1 RX 缓冲寄存器 FIFO 中，等待 CPU 或主机通过通用主机接口进行访问。

UART1 发送器模块 U1TX 接收 CPU 或主机写入的数据，并且将数据缓存到 UART1 TX 保持寄存器 FIFO (U1THR) 中。UART1 TX 移位寄存器 (U1TSR) 读取 U1THR 中存储的数据，并对数据进行汇编，通过串行输出管脚 TXD1 将数据发送出去。

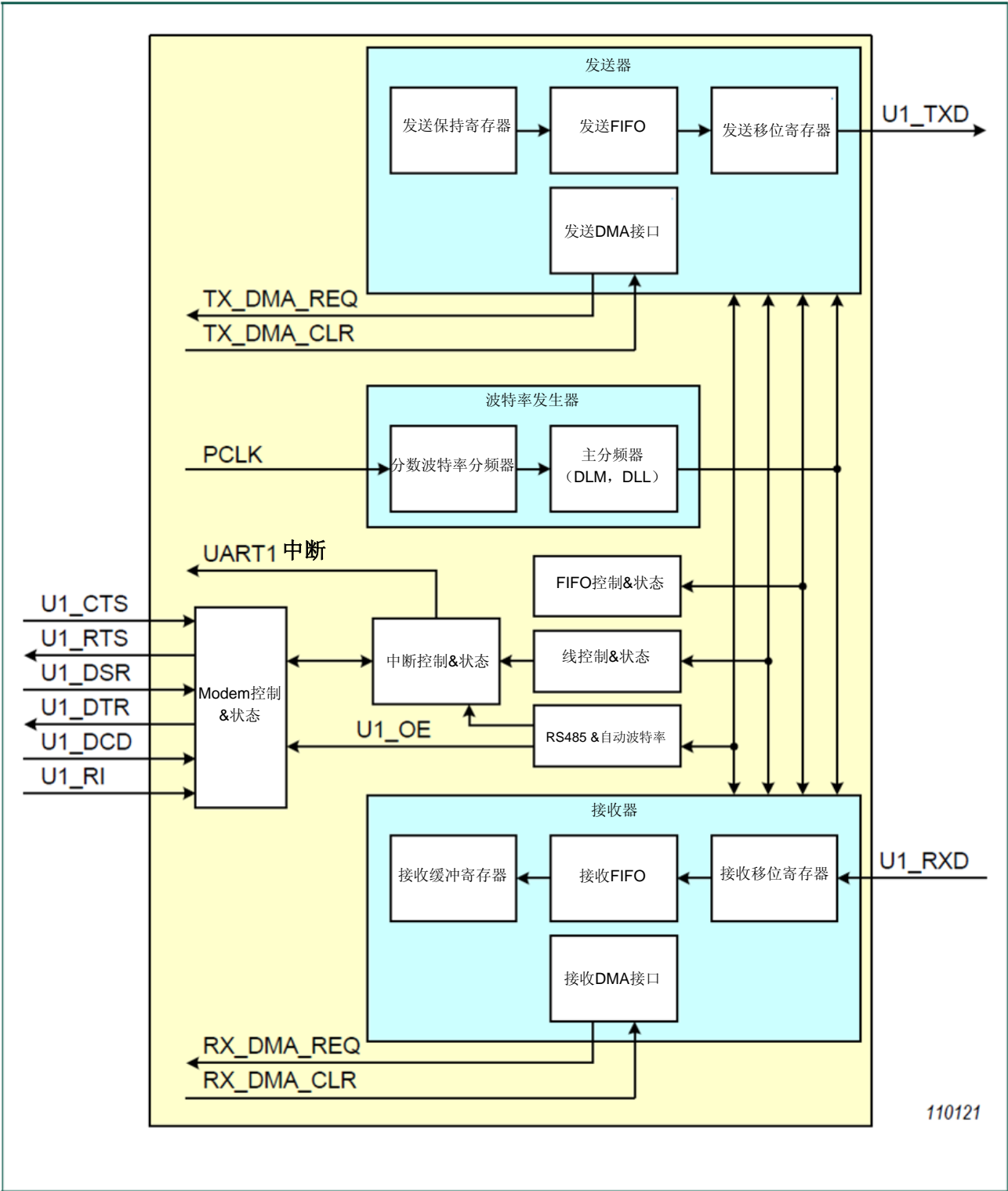
UART1 波特率发生器模块 U1BRG，产生 UART1 TX 模块所使用的时序使能信号。U1BRG 时钟输入源为 APB 时钟 (PCLK)。主时钟由 U1DLL 和 U1DLM 寄存器中指定的除数相除。得到的时钟是 16x 过采样时钟。

Modem 接口包括 U1MCR 和 U1MSR 寄存器。该接口完成 Modem 外设与 UART1 之间的握手信号交换。

中断接口包括 U1IER 和 U1IIR 寄存器。中断接口接收若干个 U1TX 和 U1RX 模块发出的单时钟宽度使能信号。

U1TX 和 U1RX 发送的状态信息会存储在 U1LSR 中。U1TX 和 U1RX 的控制信息会存储在 U1LCR 中。

图66. UART1 框图



17.4 管脚描述

表366. UART1 管脚描述

管脚	类型	描述
U1_RXD	输入	串行输入管脚。串行接收数据。
U1_TXD	输出	串行输出管脚。串行发送数据。
U1_CTS	输入	<p>消除发送。低电平有效信号，指示外部 Modem 是否已经准备好接收数据，UART1 数据可通过 TXD1 发送。在 Modem 接口的正常操作模式（U1MCR[4]=0）下，该信号的补码值会被存放在 U1MSR[4] 中。而状态改变信息会被存放到 U1MSR[0]中，如果第 4 级优先级中断使能（U1IER[3]=1），状态改变信息将作为中断源。</p> <p>消除发送。CTS1 是一个异步、低电平有效的 Modem 状态信号。该信号的状态可通过对 Modem 状态寄存器的位 4（CTS）进行查询来获知。Modem 状态寄存器（MSR）的位 0（DCTS）指示 CTS1 的状态在上次读 MSR 后是否发生了变化。如果使能了 Modem 状态中断，当 CTS1 改变电平状态且 auto-CTS 模式又未启动时，中断就会产生。CTS1 在 auto-CTS 模式下还会被用于控制发送器。</p>
U1_DCD	输入	<p>数据载波检测。低电平有效信号，指示外部 Modem 是否与 UART1 建立了通信连接，可以进行数据交换。在 Modem 接口的正常操作模式（U1MCR[4]=0）下，该信号的补码会被存放在 U1MSR[7]中。而状态改变信息会被存放到 U1MSR3 中，如果第 4 级优先级中断使能（U1IER[3]=1），状态改变信息将作为中断源。</p>
U1_DSR	输入	<p>数据设置就绪。低电平有效信号，指示外部 Modem 是否准备好与 UART1 建立通信连接。在 Modem 接口的正常操作模式（U1MCR[4]=0）下，该信号的补码值会被存放在 U1MSR[5]中。而状态改变信息会被存放到 U1MSR[1]中，如果第 4 级优先级中断使能（U1IER[3]=1），状态改变信息将作为中断源。</p>
U1_DTR	输出	<p>数据终端就绪。低电平有效信号，指示 UART1 已准备好与外部 Modem 建立连接。该信号的补码值会被存放在 U1MCR[0]中。</p> <p>DTR 管脚还可作为 RS-485/EIA-485 输出使能信号管脚使用。</p>
U1_RI	输入	<p>铃响指示。低电平有效信号，指示 Modem 检测到电话铃响信号。在 Modem 接口的正常操作模式（U1MCR[4]=0）下，该信号的补码值会被存放在 U1MSR[6]中。而状态改变信息会被存放到 U1MSR[2] 中，如果第 4 级优先级中断使能（U1IER[3]=1），状态改变信息将作为中断源。</p>
U1_RTS	输出	<p>请求发送。低电平有效信号，指示 UART1 打算发送数据给外部 Modem。该信号的补码值会被存放在 U1MCR[1]中。</p> <p>在 auto-RTS 模式下，RTS1 被用于控制发送 FIFO 阈值逻辑。</p> <p>请求发送。RTS1 是一个低电平有效信号，告知 Modem 或数据集 UART 已经准备好接收数据。将 RTS Modem 控制寄存器中相应的位置位，RTS1 输出有效（低）电平；在系统复位后或在回写模式操作过程中，或者将 MCR 的位 1（RTS）清零，RTS 输出无效（高）电平。在 auto-RTS 模式下，RTS1 受发送 FIFO 阈值逻辑控制。</p> <p>RTS 管脚还可作为 RS-485/EIA-485 输出使能信号管脚使用。</p>

17.5 寄存器描述

UART1 所包含的寄存器结构如[表 367](#) 中所示。除数锁存器访问位 (DLAB) 包含在 U1LCR[7] 中，能够使能对除数锁存器的访问。

表367. UART1 寄存器映射

名称	描述	访问	复位值 ^[1]	地址	表
U1RBR (DLAB =0)	接收缓冲寄存器。内含下一个要读取的已接收字符。	RO	无	0x4001 0000	表 368
U1THR (DLAB =0)	发送保持寄存器。在此写入下一个要发送的字符。	WO	无	0x4001 0000	表 369
U1DLL (DLAB =1)	除数锁存器 LSB。波特率除数值的最低有效字节。分数波特率分频器是使用整个除数来产生波特率的。	R/W	0x01	0x4001 0000	表 370
U1DLM (DLAB =1)	除数锁存器 MSB。波特率除数值的最高有效字节。分数波特率分频器是使用整个除数来产生波特率的。	R/W	0	0x4001 0004	表 371
U1IER (DLAB =0)	中断使能寄存器。包含 7 个独立的中断使能位对应 7 个潜在的 UART1 中断。	R/W	0	0x4001 0004	表 372
U1IIR	中断 ID 寄存器。识别处于挂起状态的中断。	RO	0x01	0x4001 0008	表 373
U1FCR	FIFO 控制寄存器。控制 UART1 FIFO 的使用和模式。	WO	0	0x4001 0008	表 375
U1LCR	线控制寄存器。包含帧格式控制和间隔产生控制。	R/W	0	0x4001 000C	表 376
U1MCR	Modem 控制寄存器。包含流控制握手控制和回写模式控制。	R/W	0	0x4001 0010	表 377
U1LSR	线状态寄存器。包含发送和接收的状态标志（包括线错误）。	RO	0x60	0x4001 0014	表 379
U1MSR	Modem 状态寄存器。包含握手信号状态标志。	RO	0	0x4001 0018	表 380
U1SCR	高速缓存寄存器。8 位的临时存储空间，供软件使用。	R/W	0	0x4001 001C	表 381
U1ACR	自动波特率控制寄存器。包含了自动波特率的特性控制。	R/W	0	0x4001 0020	表 382
U1FDR	分数分频器寄存器。为波特率分频器产生时钟输入。	R/W	0x10	0x4001 0028	表 383
U1TER	发送使能寄存器。关闭 UART 发送器，使用软件流控制。	R/W	0x80	0x4001 0030	表 385
U1RS485CTRL	RS-485/EIA-485 控制。包含了 RS-485/EIA-485 模式多方面的配置控制。	R/W	0	0x4001 004C	表 386
U1ADRMATCH	RS-485/EIA-485 地址匹配。包含了 RS-485/EIA-485 模式所使用的地址匹配值。	R/W	0	0x4001 0050	表 387
U1RS485DLY	RS-485/EIA-485 方向控制延迟。	R/W	0	0x4001 0054	表 388

[1] 复位值仅反映已使用位中保存的数据，它不包括保留位的内容。

17.5.1 UART1 接收器缓冲寄存器（U1RBR—0x4001 0000，DLAB = 0）

U1RBR 是 UART1 RX FIFO 的最高字节。RX FIFO 的最高字节包含最早接收到的字符，并且可通过总线接口读取。LSB（位 0）代表“最早”接收到的数据位。如果接收到的字符少于 8 位，则未使用的 MSB 使用 0 填充。

如果要访问 U1RBR，U1LCR 中的除数锁存器访问位（DLAB）必须为 0。U1RBR 始终为只读寄存器。

由于 PE、FE 和 BI 位与 RBR FIFO 顶部的字节（即下次读取 RBR 时获取的字节）对应，因此，要正确地成对读出有效的已接收字节及其状态位，应先读取 U1LSR 寄存器的内容，然后读取 U1RBR 中的字节。

表368. UART1 接收器缓冲寄存器位描述（U1RBR – 0x4001 0000，DLAB = 0）

位	符号	描述	复位值
7:0	RBR	UART1 接收缓冲寄存器包含了 UART1 RX FIFO 当中最早接收到的字节。	未定义
31:8	-	保留，从保留位读出的值未定义。	无

17.5.2 UART1 发送保持寄存器（U1THR—0x4001 0000，DLAB = 0）

U1THR 只写寄存器是 UART1 TX FIFO 的最高字节。最高字节是 TX FIFO 中的最新字节，可以通过总线接口写入。LSB 代表要发送的第一个位。

如果要访问 U1THR，U1LCR 中的除数锁存器访问位（DLAB）必须为 0。U1THR 为只写寄存器。

表369. UART1 发送保持寄存器位描述（U1THR – 0x4001 0000，DLAB = 0）

位	符号	描述	复位值
7:0	THR	写 UART1 发送保持寄存器会使数据保存到 UART1 发送 FIFO 中。 当字节达到 FIFO 的底部并且发送器就绪时，字节就会被发送。	无
31:8	-	保留。读取值未定义，只写入 0。	无

17.5.3 UART1 除数锁存器 LSB 和 MSB 寄存器（U1DLL—0x4001 0000 和 U1DLM—0x4001 0004，DLAB = 1）

UART1 除数锁存器是 UART1 波特率发生器的一部分，它与分数分频器一同使用，保持产生波特率时钟的 APB 时钟(PCLK)的分频值，波特率时钟必须是所需波特率的 16x 倍。U1DLL 和 U1DLM 寄存器一起构成一个 16 位除数，其中 U1DLL 包含除数的低 8 位，而 U1DLM 包含除数的高 8 位。分频值 0x0000 会被作为 0x0001 处理，因为除数不能为 0。如果要访问 UART1 除数锁存器，U1LCR 中的除数锁存器访问位（DLAB）必须为 1。有关如何为 U1DLL 和 U1DLM 选择正确值的详细信息，参见 [17.5.16](#) 节。

表370. UART1 除数锁存器 LSB 寄存器位描述 (U1DLL – 0x4001 0000, DLAB = 1)

位	符号	描述	复位值
7:0	DLLSB	UART1 除数锁存器 LSB 寄存器与 U1DLM 寄存器一起决定 UART1 的波特率。	0x01
31:8	-	保留。读取值未定义，只写入 0。	无

表371. UART1 除数锁存器 MSB 寄存器位描述 (U1DLM – 0x4001 0004, DLAB = 1)

位	符号	描述	复位值
7:0	DLMSB	UART1 除数锁存器 MSB 寄存器与 U1DLL 寄存器一起决定 UART1 的波特率。	0
31:8	-	保留。读取值未定义，只写入 0。	无

17.5.4 UART1 中断使能寄存器 (U1IER—0x4001 0004, DLAB = 0)

U1IER 用于使能四个 UART1 中断源。

表372. UART1 中断使能寄存器位描述 (U1IER – 0x4001 0004, DLAB = 0)

位	符号	值	描述	复位值
0	RBR Interrupt Enable		使能 UART1 的接收数据可用中断。它还控制着字符接收超时中断。	0
		0	禁止 RDA 中断。	
		1	使能 RDA 中断。	
1	THRE Interrupt Enable		使能 UART1 的 THRE 中断。该中断的状态可从 U1LSR[5]中读出。	0
		0	禁止 THRE 中断。	
		1	使能 THRE 中断。	
2	RX Line Interrupt Enable		使能 UART1 的 RX 线状态中断。该中断的状态可从 U1LSR[4:1]中读出。	0
		0	禁止 RX 线状态中断。	
		1	使能 RX 线状态中断。	
3	Modem Status Interrupt Enable		使能 Modem 中断。该中断状的状态可从 U1MSR[3:0]中读出。	0
		0	禁止 Modem 中断。	
		1	使能 Modem 中断。	
6:4	-		保留。读取值未定义，只写入 0。	无
7	CTS Interrupt Enable		如果使能了 auto-CTS 模式，当 CTS1 信号发生跳变时，该位会使能/禁止 Modem 状态中 0	0
			断的产生。如果 auto-CTS 模式被禁止，在 Modem 状态中断使能位 (U1IER[3]) 被置位的	
			情况下，CTS1 信号跳变将会触发中断。	
			在正常操作模式下，CTS1 信号跳变将会触发 Modem 状态中断，除非中断已被禁止 (U1IER	
			寄存器中的 U1IER[3]被清零)。在 auto-CTS 模式下，只有当 U1IER[3]和 U1IER[7]位同时	
			被置位时，CTS1 位上的跳变才会触发中断。	
		0	禁止 CTS 中断。	
		1	使能 CTS 中断。	
8	ABEOIntEn		使能 auto-baud 结束中断。	0
		0	禁止 auto-baud 结束中断。	
		1	使能 auto-baud 结束中断。	
9	ABTOIntEn		使能 auto-baud 超时中断。	0
		0	禁止 auto-baud 超时中断。	
		1	使能 auto-baud 超时中断。	
31:10	-		保留。读取值未定义，只写入 0。	无

17.5.5 UART1 中断标识寄存器（U1IIR—0x4001 0008）

U1IIR 提供一个状态代码，用于指示一个挂起中断的优先级和中断源。在访问 U1IIR 的过程中，中断被冻结。如果在访问 U1IIR 的过程中产生了中断，该中断会被记录以用于下次 U1IIR 访问。

表373. UART1 中断标识寄存器位描述（U1IIR – 0x4001 0008）

位	符号	值	描述	复位值
0	IntStatus		中断状态。注：U1IIR[0]为低电平有效。挂起的中断可通过评估 U1IIR[3:1]确定。	1
		0	至少有一个中断被挂起。	
		1	没有挂起的中断。	
3:1	IntId		中断标识。U1IER[3:1]指示对应 UART1 Rx 或 TX FIFO 的中断。 下面未列出的 U1IER[3:1]的其他组合都为保留值（100,101,111）。	0
		011	1 – 接收线状态（RLS）。	
		010	2a – 接收数据可用（RDA）。	
		110	2b – 字符超时指示器（CTI）。	
		001	3 – THRE 中断。	
		000	4 – Modem 中断。	
5:4	-		保留。读取值未定义，只写入 0。	无
7:6	FIFO Enable		这些位等效于 U1FCR[0]。	0
8	ABEOInt		Auto-baud 结束中断。若 auto-baud 已成功结束且中断被使能，则为真。	0
9	ABTOInt		Auto-baud 超时中断。若 auto-baud 已超时且中断被使能，则为真。	0
31:10	-		保留。读取值未定义，只写入 0。	无

位 U1IIR[9:8]由自动波特率功能设置，用于发布超时信号或自动波特率结束条件信号。而自动波特率中断条件的清除是通过设置自动波特率控制寄存器中的相应清除（Clear）位来实现的。

如果 IntStatus 位为 1，表示没有中断被挂起，此时 IntId 位为 0。如果 IntStatus 位为 0，表示有一个非自动波特率中断被挂起，此时 IntId 位会指出中断的类型以及表 374 中所示的处理。知道了 U1IIR[3:0]的状态，中断处理程序就能确定中断原因以及如何清除激活的中断。在退出中断服务程序之前，必须读取 U1IIR 来清除中断。

UART1 RLS 中断（U1IIR[3:1]=011）是最高优先级的中断，只要 UART1RX 输入上产生以下四个错误条件中的任意一个，该位就会被置位：溢出错误（OE）、校验错误（PE）、帧错误（FE）以及间隔中断（BI）。通过 U1LSR[4:1]可查看设置中断的 UART1 Rx 错误条件。读取 U1LSR 时，中断会被清除。

UART1 RDA 中断（U1IIR[3:1]=010）与 CTI 中断（U1IIR[3:1] = 110）共用第二优先级。当 UART1 Rx FIFO 深度达到 U1FCR7:6 中定义的触发值时，RDA 会被激活，而当 UART1 Rx FIFO 深度低于触发值时，RDA 复位。当 RDA 中断有效时，CPU 可读出由触发值定义的一个数据块。

CTI 中断（U1IIR[3:1]=110）是一个第二优先级中断，当 UART1 Rx FIFO 内含有至少一个字符并且在接收到第 3.5 到 4.5 个字符时间内没有发生 UART1 Rx FIFO 动作的情况下，该中断被设置。任何 UART1 Rx FIFO 动作（UART1 RSR 的读取或写入）都会清除该中断。当

接收到的消息不是触发值的倍数时，该中断会清空 UART1 RBR。例如，如果外设要发送一条长度为 105 个字符的消息，而触发值是 10 个字符，那么 CPU 接收到 10 个 RDA 中断的结果是 100 个字符的传输，而接收 1 至 5 个 CTI 中断（取决于服务程序）的结果是剩下 5 个字符的传输。

表374. UART1 中断处理

U1IIR[3:0]值 ^[1]	优先级	中断类型	中断源	中断复位
0001	-	无	无	-
0110	最高	RX 线状态/错误	OE ^[2] 、PE ^[2] 、FE ^[2] 或 BI ^[2]	U1LSR 读操作 ^[2]
0100	第二	RX 数据可用	Rx 数据可用或达到 FIFO 的触发值（U1FCR0=1）	U1RBR 读操作 ^[3] 或 UART1 FIFO 低于触发值
1100	第二	字符超时指示	RX FIFO 中至少有一个字符，并且在一段时间内没有字符输入或移出，该时间的长短取决于 FIFO 中的字符数以及在 3.5 到 4.5 字符的时间内的触发值。 实际时间为：[(字长度) × 7 - 2] × 8 + [(触发值 - 字符数) × 8 + 1] RCLK	U1RBR 读操作 ^[3]
0010	第三	THRE	THRE ^[2]	U1IIR 读操作 ^[4] （如果 U1IIR 是中断源）或 THR 写操作
0000	第四	Modem 状态	CTS、DSR、RI 或 DCD	读 MSR

[1] “0000”、“0011”、“0101”、“0111”、“1000”、“1001”、“1010”、“1011”、“1101”、“1110”、“1111” 均为保留值。

[2] 详细信息，参见 17.5.10 节。

[3] 详细信息，参见 17.5.1 节。

[4] 详细信息，参见 17.5.5 节和 17.5.2 节。

UART1 THRE 中断（U1IIR[3:1] = 001）是第三优先级中断，在 UART1 THR FIFO 为空并满足特定的初始化条件时，该中断会被激活。这些初始化条件是为了让 UART1 THR FIFO 有机会填入数据，以免在系统启动时发生多次 THRE 中断。THRE=1，且在上次 THRE=1 事件后，U1THR 中没有同时出现至少两个字符的情况下，这些初始化条件就会实现一个字符延时减去停止位。当没有解码和服务的 THRE 中断时，该延时为 CPU 提供了将数据写入 U1THR 的时间。如果 UART1 THR FIFO 同时保持两个或更多字符，且当前 U1THR 为空时，THRE 中断就会立即被设置。当发生 U1THR 写操作或 U1IIR 读操作，并且 THRE 是最高优先级的中断（U1IIR[3:1]=001）时，THRE 中断复位。

Modem 状态中断是优先级最低的中断，只要 Modem 的输入管脚、DCD、DSR 或 CTS 的状态有任何变化，该中断就会被激活。此外，当 Modem 输入 RI 上低电平到高电平的跳变时，也会触发 Modem 中断。中断源可通过查询 U1MSR[3:0]来确定。读取 U1MSR 会清除 Modem 中断。

17.5.6 UART1 FIFO 控制寄存器（U1FCR—0x4001 0008）

只写的 U1FCR 控制 UART1 RX 和 TX FIFO 的操作。

表375. UART1 FIFO 控制寄存器位描述（U1FCR – 0x4001 0008）

位	符号	值	描述	复位值
0	FIFO Enable	0	UART1 FIFO 被禁止。禁止在应用中使用。	0
		1	高电平有效，使能对 UART1 Rx 和 TX FIFO 以及 U1FCR[7:1]的访问。该位必须置位以实现正确的 UART1 操作。该位的任何变化都将使 UART1 FIFO 自动清空。	
1	RX FIFO Reset	0	对两个 UART1 FIFO 均无影响。	0
		1	写逻辑 1 到 U1FCR[1]将会清零 UART1 Rx FIFO 中所有字节，并复位指针逻辑。该位会自动清零。	
2	TX FIFO Reset	0	对两个 UART1 FIFO 均无影响。	0
		1	写逻辑 1 到 U1FCR[2]将会清零 UART1 TX FIFO 中所有字节，并复位指针逻辑。该位会自动清零。	
3	DMA Mode Select		当 FIFO 使能位（该寄存器的位 0）被置位时，该位选择 DMA 模式。请参考 17.5.6.1 节。	0
5:4	-		保留。读取值未定义，只写入 0。	无
7:6	RX Trigger Level		这两个位决定了接收 UART1 FIFO 在激活中断前必须写入的字符数量。	0
		00	触发点 0（1 字节或 0x01）	
		01	触发点 1（4 字节或 0x04）	
		10	触发点 2（8 字节或 0x08）	
		11	触发点 3（14 字节或 0x0E）	
31:8	-		保留。读取值未定义，只写入 0。	无

17.5.6.1 DMA 操作

用户可以选择用 DMA 进行 UART 的发送和/或接收。DMA 模式由 FCR 寄存器中的 DMA 模式选择位决定。只有在 FCR 寄存器中的 FIFO 使能位使能了 FIFO 时，该位才有用。

UART 接收 DMA

在 DMA 模式下，如果接收 FIFO 水平大于或等于触发值，或者在发生一次字符超时的情况下，接收 DMA 请求才会生效。请参见上文中对 RX 触发值的描述。接收 DMA 请求由 DMA 控制器清除。

UART 发送 DMA

在 DMA 模式下，当发送 FIFO 变为未满时，发送 DMA 请求就会生效。发送 DMA 请求由 DMA 控制器清除。

17.5.7 UART1 线控制寄存器（U1LCR—0x4001 000C）

U1LCR 决定待发送或接收的数据字符的格式。

表376. UART1 线控制寄存器位描述（U1LCR—0x4001 000C）

位	符号	值	描述	复位值
1:0	Word Length Select	00	5 位字符长度。	0
		01	6 位字符长度。	
		10	7 位字符长度。	
		11	8 位字符长度。	
2	Stop Bit Select	0	1 个停止位。	0
		1	2 个停止位（当 U1LCR[1:0]=00 时，为 1.5 个）。	
3	Parity Enable	0	禁止奇偶产生和校验。	0
		1	使能奇偶产生和校验。	
5:4	Parity Select	00	奇校验。发送字符和添加的校验位所包含的“1”的数量是一个奇数。	0
		01	偶校验。发送字符和添加的校验位所包含的“1”的数量是一个偶数。	
		10	强制为“1” stick 校验	
		11	强制为“0” stick 校验	
6	Break Control	0	禁止间隔发送	0
		1	使能间隔发送。当 U1LCR[6]为高电平（有效）时，输出管脚 UART1 TXD 强制为逻辑 0。	
7	Divisor Latch Access Bit (DLAB)	0	禁止访问除数锁存器	0
		1	使能访问除数锁存器	
31:8	-		保留。读取值未定义，只写入 0。	无

17.5.8 UART1 Modem 控制寄存器（U1MCR—0x4001 0010）

U1MCR 使能 Modem 的回写模式，并控制 Modem 的输出信号。

表377. UART1 Modem 控制寄存器位描述（U1MCR—0x4001 0010）

位	符号	值	描述	复位值
0	DTR Control		选择 Modem 输出管脚 DTR。当 Modem 回写模式激活时，该位读出为 0。	0
1	RTS Control		选择 Modem 输出管脚 RTS。当 Modem 回写模式激活时，该位读出为 0。	0
3-2	-		保留。读取值未定义，只写入 0。	0
4	Loopback Mode Select		Modem 回写模式提供了执行回写测试的诊断机制。发送器输出的串行数据在内部连接到接收器的串行输入端。输入管脚 RXD1 对回写操作无影响，而输出管脚 TXD1 保持为标记状态。4 个 Modem 输入端（CTS、DSR、RI 和 DCD）与外部断开。从外部看来，Modem 输出端（RTS、DTR）无效。而从内部来看，4 个 Modem 输出都连接到 4 个 Modem 输入上。这样连接的结果将导致 U1MSR 的高 4 位由 U1MCR 的低 4 位驱动，而不是在正常模式下由 4 个 Modem 输入驱动。这样在回写模式下，写 U1MCR 的低 4 位可产生 Modem 状态中断。	0
		0	禁能 Modem 回写模式。	
		1	使能 Modem 回写模式。	
5	-		保留。读取值未定义，只写入 0。	0
6	RTSen	0	禁止 auto-RTS 流控制。	0
		1	使能 auto-RTS 流控制。	
7	CTSen	0	禁止 auto-RTS 流控制。	0
		1	使能 auto-RTS 流控制。	
31:8	-		保留。读取值未定义，只写入 0。	无

17.5.9 自动流控制

如果启用了 Auto-RTS 模式，那么 UART1 的接收 FIFO 硬件将控制 UART1 的 RTS1 输出。如果启用了 Auto-CTS 模式，那么 UART1 的 U1TSR 硬件仅在 CTS1 输入信号有效的情况下启动发送。

17.5.9.1 Auto-RTS

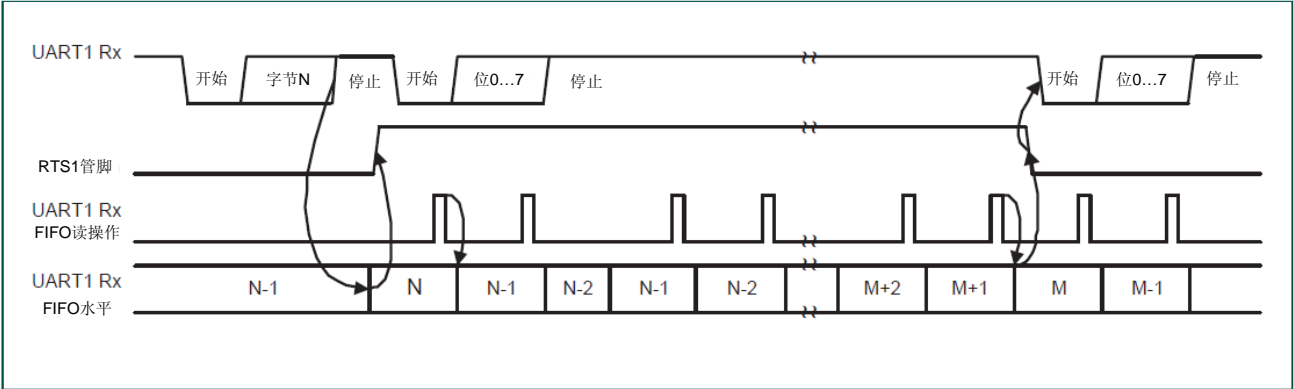
Auto-RTS 功能通过设置 RTSen 位启用。Auto-RTS 数据流控制在 U1RBR 模块中产生，链接到编程的接收 FIFO 触发值。如果启用了 Auto-RTS，则数据流的控制过程如下所述：

当接收 FIFO 水平达到编程的触发值时，RTS1 变为无效（变为高电平）。在达到触发值后，发送 UART 可能还会发送一个额外的字节（假设发送 UART 还有一个字节要发送），因为 UART 在开始发送额外字节之前可能不知道 RTS1 信号已无效。一旦接收 FIFO 达到之前的触发值，RTS1 就会自动重新生效（变为低电平）。RTS1 重新生效指示发送 UART 继续发送数据。

如果禁用了 Auto-RTS 模式，则 RTSen 位控制 UART1 的 RTS1 输出。如果启用了 Auto-RTS 模式，则硬件控制 RTS1 输出，而且 RTS1 的实际值会复制到 UART1 的 RTS 控制位（RTS Control）。只要启用了 Auto-RTS 模式，软件就只能对 RTS 控制位的值进行读取。

示例：假设 UART1 工作在 550 模式下，将 U1FCR 中的触发值设置为 0x2，此时如果使能了 auto-RTS 模式，当接收 FIFO 内出现了 8 个字节时，UART1 将会把 RTS1 设为无效（参见表 375）。当接收 FIFO 达到之前的触发值（4 字节）时，RTS1 输出将会再次生效。

图67. Auto-RTS 功能时序



17.5.9.2 Auto-CTS

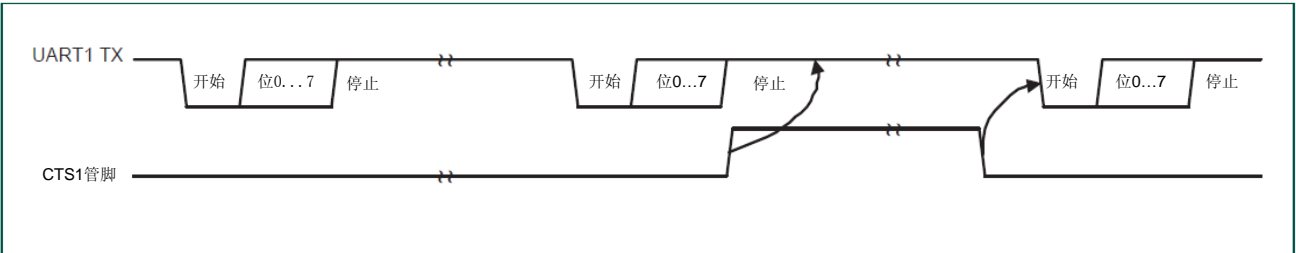
Auto-CTS 功能通过设置 CTSen 位启用。如果启用了 Auto-CTS 功能，U1TSR 模块中的发送电路会在发送下一个数据字节之前检查 CTS1 输入。当 CTS1 有效（低电平）时，发送器就会将下一个字节发送出去。如果要使发送器停止发送后续字节，必须在末尾的停止位发送出一半之前释放 CTS1。在 Auto-CTS 模式下，即使 U1MSR 中的 Delta CTS 位被置位，CTS1 信号的变化也不会触发 Modem 状态中断，除非 CTS 中断启用位 (CTS Interrupt Enable) 被置位。表 378 所示为生成一个 Modem 状态中断的条件。

表378. Modem 状态中断产生

使能 Modem 状态中断 (U1ER[3])	CTSen (U1MCR[7])	CTS 中断使能 (U1IER[7])	Delta CTS (U1MSR[0])	Delta DCD 或后沿 RI 或 Delta DSR (U1MSR[3] 或 U1MSR[2] 或 U1MSR[1])	Modem 状态中断
0	x	x	x	x	否
1	0	x	0	0	否
1	0	x	1	x	是
1	0	x	x	1	是
1	1	0	x	0	否
1	1	0	x	1	是
1	1	1	0	0	否
1	1	1	1	x	是
1	1	1	x	1	是

Auto-CTS 功能会减少对主机系统的中断。当流控制被使能时，CTS1 状态的变化不会触发主机中断，因为设备会自动控制其发送器。如果不启用 Auto-CTS，发送器会将任何存放在发送 FIFO 中的数据都发出去，从而导致接收器发生溢出错误。图 68 给出了 Auto-CTS 的功能时序。

图68. Auto-CTS 功能时序



在开始发送初始字符时，CTS1 信号生效。一旦挂起的数据传输结束了，发送将立即停止。CTS1 无效（变为高电平）时，UART 会不断地发送“1”。一旦 CTS1 获知挂起的数据传输恢复了，起始位就会被发送，然后是下一个字符的数据位。

17.5.10UART1 线状态寄存器（U1LSR—0x4001 0014）

U1LSR 是一个只读寄存器，提供 UART1 TX 和 RX 模块的状态信息。

表379. UART1 线状态寄存器位描述（U1LSR—0x4001 0014）

位	符号	值	描述	复位值
0	Receiver Data Ready (RDR)		当 U1RBR 包含一个未读字符时，U1LSR[0]就会被置位；当 UART1 RBR FIFO 为空 0 时，U1LSR[0]就会被清零。	0
		0	UART1 接收 FIFO 为空。	
		1	UART1 接收 FIFO 不为空。	
1	Overrun Error (OE)		溢出错误条件在错误发生后立即设置。读 U1LSR 会清零 U1LSR[1]。当 UART1 RSR 0 已有新的字符汇集，而 UART1 RBR FIFO 已满时，U1LSR[1]会被置位。此时，UART1 RBR FIFO 将不会被覆盖，UART1 RSR 内的字符将会丢失。	0
		0	溢出错误状态未激活。	
		1	溢出错误状态激活。	
2	Parity Error (PE)		当接收字符的校验位处于错误状态时，校验错误就会产生。读 U1LSR 会清零 0 U1LSR[2]。校验错误检测时间取决于 U1FCR[0]。 注：校验错误与 UART1 RBR FIFO 顶部的字符相关。	0
		0	校验错误状态未激活。	
		1	校验错误状态激活。	
3	Framing Error (FE)		当接收字符的停止位为逻辑 0 时，就会发生帧错误。读 U1LSR 会清零 U1LSR[3]。 0 帧错误检测时间取决于 U1FCR0。当检测到有帧错误时，RX 会尝试与数据重新同步，并假设错误的停止位实际上是一个超前的起始位。但即使没有出现帧错误，它也无法假设下一个接收到的字节是正确的。 注：帧错误与 UART1 RBR FIFO 顶部的字符相关。	0
		0	帧错误状态未激活。	
		1	帧错误状态激活。	
4	Break Interrupt		在发送整个字符（起始位、数据、校验位以及停止位）过程中，RXD1 如果保持在间 0	0

位	符号	值	描述	复位值
	(BI)		隔状态（全“0”），则产生间隔中断。一旦检测到间隔条件，接收器立即进入空闲状态，直到 RXD1 进入标记状态（全“1”）。读 U1LSR 会清零该状态位。间隔检测时间取决于 U1FCR[0]。 注： 间隔中断与 UART1 RBR FIFO 顶部的字符相关。	
		0	间隔中断状态未激活。	
		1	间隔中断状态激活。	
5	Transmitter Holding Register Empty (THRE)		当检测到 UART1 THR 已空时，THRE 就会立即被设置。写 U1THR 会清零 THRE。	1
		0	U1THR 包含有效数据。	
		1	U1THR 为空。	
6	Transmitter Empty (TEMT)		当 U1THR 和 U1TSR 同时为空时，TEMT 就会被设置；而当 U1TSR 或 U1THR 任一一个包含有效数据时，TEMT 就会被清零。	1
		0	U1THR 和/或 U1TSR 包含有效数据。	
		1	U1THR 和 U1TSR 为空。	
7	Error in RX FIFO (RXFE)		当一个带有 RX 错误（如：帧错误、校验错误或间隔中断）的字符载入到 U1RBR 中 0 时，U1LSR[7]就会被设置。当 U1LSR 寄存器被读取并且 UART1 FIFO 中不再有错误时，该位就会清零。	0
		0	U1RBR 中没有 UART1 RX 错误或 U1FCR[0]=0。	
		1	UART1 RBR 中包含至少一个 UART1 RX 错误。	
31:8			保留，从保留位读出的值未定义。	

17.5.11 UART1 Modem 状态寄存器（U1MSR—0x4001 0018）

U1MSR 是一个只读寄存器，提供 Modem 输入信号的状态信息。读 U1MSR 会清零 U1MSR[3:0]。请注意，Modem 信号对 UART1 操作没有直接影响，Modem 信号的操作是通过软件实现的。

表380. UART1 Modem 状态寄存器位描述（U1MSR – 0x4001 0018）

位	符号	值	描述	复位值
0	Delta CTS		当输入端 CTS 的状态改变时，该位置位。读 U1MSR 会清零该位。	0
		0	没有检测到 Modem 输入端 CTS 上的状态变化。	
		1	检测到 Modem 输入端 CTS 上的状态变化。	
1	Delta DSR		当输入端 DSR 的状态改变时，该位置位。读 U1MSR 会清零该位。	0
		0	没有检测到 Modem 输入端 DSR 上的状态变化。	
		1	检测到 Modem 输入端 DSR 上的状态变化。	
2	Trailing Edge RI		当输入端 RI 上低电平到高电平的跳变时，该位置位。读 U1MSR 会清零该位。	0
		0	没有检测到 Modem 输入端 RI 上的状态变化。	
		1	检测到 RI 上的低电平到高电平跳变的变化。	
3	Delta DCD		当输入端 DCD 的状态改变时，该位置位。读 U1MSR 会清零该位。	0
		0	没有检测到 Modem 输入端 DCD 上的状态变化。	
		1	检测到 Modem 输入端 DCD 上的状态变化。	
4	CTS		清除发送状态。输入信号 CTS 的补码。在 Modem 回写模式下，该位连接到 U1MCR[1]。	0

位	符号	值	描述	复位值
5	DSR		数据设置就绪状态。输入信号 DSR 的补码。在 Modem 回写模式下，该位连接到 U1MCR[0]。	0
6	RI		铃响指示状态。输入 RI 的补码。在 Modem 回写模式下，该位连接到 U1MCR[2]。	0
7	DCD		数据载波检测状态。输入 DCD 的补码。在 Modem 回写模式下，该位连接到 U1MCR[3]。	0
31:8	-		保留，从保留位读出的值未定义。	无

17.5.12 UART1 高速缓存寄存器（U1SCR—0x4001 001C）

U1SCR 对 UART1 操作没有影响。用户可以自由地读写该寄存器。不提供中断接口向主机指示 U1SCR 所发生的读或写操作。

表381. UART1 高速缓存寄存器位描述（U1SCR – 0x4001 0014）

位	符号	描述	复位值
7:0	Pad	一个可读、可写的字节	0
31:8	-	保留。读取值未定义，只写入 0。	无

17.5.13 UART1 自动波特率控制寄存器（U1ACR—0x4001 0020）

在用户测量波特率的输入时钟/数据速率期间，整个测量过程就是由 UART1 自动波特率控制寄存器（U1ACR）进行控制的。用户可以自由地读写该寄存器。

表382. 自动波特率控制寄存器位描述（U1ACR—0x4001 0020）

位	符号	值	描述	复位值
0	Start		在 auto-baud 功能结束后，该位会自动清零。	0
		0	Auto-baud 功能停止（auto-baud 功能不运行）。	
		1	Auto-baud 功能启动（auto-baud 功能正在运行）。Auto-baud 运行位。该位会在 auto-baud 功能结束后自动清零。	
1	Mode		Auto-baud 模式选择位。	0
		0	模式 0。	
		1	模式 1。	
2	AutoRestart	0	不重新启动	0
		1	如果超时则重新启动（计数器会在下一个 UART1 Rx 下降沿重新启动）	0
7:3	-		保留。读取值未定义，只写入 0。	0
8	ABEOIntClr		Auto-baud 结束中断清零位（仅可写访问）。	0
		0	写 0 无影响。	
		1	写 1 将 U1IIR 中相应的中断清除。	
9	ABTOIntClr		Auto-baud 超时中断清零位（仅可写访问）。	0
		0	写 0 无影响。	
		1	写 1 将 U1IIR 中相应的中断清除。	
31:10	-		保留。读取值未定义，只写入 0。	0

17.5.14 自动波特率

UART1 自动波特率 (auto-baud) 功能可用于测量基于 “AT” 协议 (Hayes 命令) 的输入波特率。如果自动波特率功能被使能, 那么自动波特率功能将测量接收数据流中的 1 位所消耗的时间, 并根据这个结果来设置除数锁存器寄存器 U1DLM 和 U1DLL。

注: 在自动波特率操作期间不会连接分数波特率分频器, 所以在需要自动波特率功能时不应使用分数波特率分频器。

自动波特率功能是通过置位 U1ACR 起始位来启动的, 并通过清零 U1ACR 起始位来停止。自动波特率一旦结束, 起始位会被立即清零, 如果此时读取该起始位, 会返回自动波特率的状态 (挂起/完成)。

可通过设置 U1CAR 模式位来使用两种自动波特率测量模式。在模式 0 下, 波特率是通过对 UART1 Rx 管脚上两个连续的下降沿进行测量 (起始位的下降沿和最低有效位的下降沿) 来得到的。在模式 1 下, 波特率则是通过测量 UART1 Rx 管脚上的下降沿与后续的上升沿之间的时间 (起始位的长度) 来得到的。

如果发生超时 (速率测量计数器溢出), 可以使用 U1ACR AutoRestart 位来自动重启波特率测量。如果该位被置位, 速率测量将会在下一次 UART1 Rx 管脚的下一个下降沿重新启动。

自动波特率功能会产生两种中断:

- 如果使能了中断 (U1IER ABTOIntEn 置位且自动波特率测量计数器溢出), 则 U1IIR ABTOInt 中断将被设置。
- 如果使能了中断 (U1IER ABEOIntEn 置位且自动波特率已成功完成), 则 U1IIR ABEOInt 中断将被设置。

必须通过置位相应的 U1ACR ABTOIntClr 位和 ABEOIntEn 位, 才能清除自动波特率中断。

在执行自动波特率期间, 分数波特率发生器通常被禁用 (即 DIVADDVAL=0)。但是, 如果使能了分数波特率发生器 (即 DIVADDVAL>0), 那么它将影响 UART1 Rx 管脚波特率的测量, 但 U1FDR 寄存器的值在测量波特率以后不会被修改。此外, 当使用自动波特率时, 任何对 U1DLM 和 U1DLL 寄存器的写操作都必须在写 U1ACR 寄存器之前完成。UART1 支持的最小和最大波特率是 PCLK、数据位数、停止位数以及校验位数的函数。

(1)

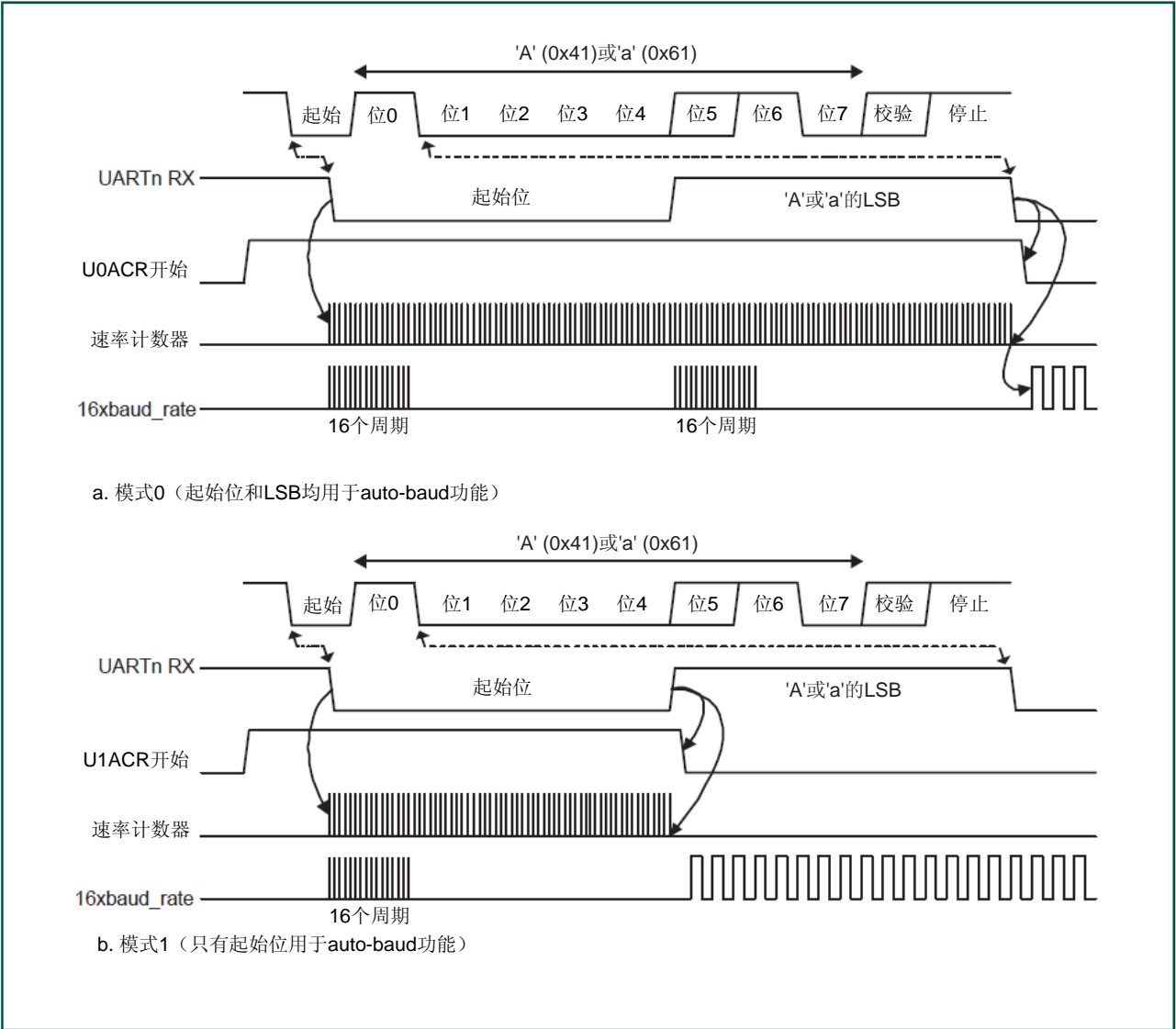
$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq_{UART1} baudrate \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

17.5.15 自动波特率模式

当软件等待一个“AT”命令时，它采用期望的字符格式配置 UART1，并置位 U1ACR 起始位。用户不必关心除数锁存器 U1DLM 和 U1DLL 中的初始值。由于字母“A”或“a”的 ASCII 编码（“A”=0x41，“a”=0x61）的关系，UART1 Rx 管脚检测到的起始位和期望字符的 LSB 是由两个下降沿限定的。当 U1ACR 起始位被置位时，自动波特率协议将执行以下步骤：

1. 一旦 U1ACR 起始位被置位，波特率测量计数器会立即复位，同时 UART1 U1RSR 复位。U1RSR 波特率切换为最高速率。
2. UART1 Rx 管脚的下降沿会触发起始位的开始。波特率测量计数器将开始对 PCLK（可选择被分数波特率发生器预分频）进行计数。
3. 在接收起始位的过程中，RSR 波特率输入端会产生 16 个脉冲，频率与（经分数波特率发生器预分频的）UART1 输入时钟相同，这样可保证起始位存储在 U1RSR 中。
4. 在接收起始位的过程中（以及模式 0 下的字符 LSB），速率计数器将继续随着被预分频的 UART1 输入时钟（PCLK）递增。
5. 如果 Mode=0，则速率计数器将在 UART1 Rx 管脚的下一个下降沿停止。如果 Mode=1，则速率计数器将在 UART1 Rx 管脚的下一个上升沿停止。
6. 速率计数器的值会载入到 U1DLM/U1DLL 中，并且波特率将会切换到正常操作模式。在设置完 U1DLM/U1DLL 后，如果自动波特率结束中断被使能，U1IIR ABEOInt 将会被置位。接着，U1RSR 将继续接收“A/a”字符剩余的位。

图69. 自动波特率 a)模式 0 和 b)模式 1 波形图



17.5.16 UART1 分数分频器寄存器 (U1FDR—0x4001 0028)

UART1 分数分频器寄存器 (U1FDR) 控制着用于波特率生成的时钟预分频器，并且用户可自由对该寄存器进行读写操作。该预分频器使用 APB 时钟并根据指定的分数要求产生一个输出时钟。

重要提示：如果分数分频器是有效的 (DIVADDVAL>0) 且 DLM=0，则 DLL 寄存器的值必须大于 2。

表383. UART1 分数分频器寄存器位描述（U1FDR – 0x4001 0028）

位	功能	值	描述	复位值
3:0	DIVADDVAL	0	产生波特率的预分频除数值。如果该字段为 0，分数波特率发生器将不会影响 UARTn 的波特率。	0
7:4	MULVAL	1	波特率预分频乘数值。不管是否使用分数波特率发生器，为了让 UARTn 正常工作，该字段必须大于或等于 1。	1
31:8	-	-	保留。读取值未定义，只写入 0。	0

该寄存器控制着用于波特率生成的时钟预分频器。该寄存器的复位值会让 UART1 的分数功能保持在禁用状态，以确保 UART1 与没有配备该特性的 UART 保持全面的软件和硬件兼容。

UART1 波特率的计算（n = 1）：(2)

$$UART1_{baudrate} = \frac{PCLK}{16 \times (256 \times U1DLM + U1DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

其中 PCLK 为外设时钟，U1DLM 和 U1DLL 为标准 UART1 波特率分频器寄存器，而 DIVADDVAL 和 MULVAL 为 UART1 分数波特率发生器的指定参数。

MULVAL 和 DIVADDVAL 的值应符合以下条件：

- 1. 1≤MULVAL≤15
- 2. 0≤DIVADDVAL≤14
- 3. DIVADDVAL<MULVAL

U1FDR 的值在发送/接收数据的过程中不应进行修改，否则可能导致数据丢失或损坏。

如果 U1FDR 寄存器的值不符合上述两个要求，则分数分频器输出为未定义。如果 DIVADDVAL 为 0，则分数分频器会被禁用，并且不会对时钟进行分频。

17.5.16.1 波特率计算

UART1 可以与分数分频器一起工作，也可以不使用分数分频器。在实际应用中，使用几种不同的分数分频器设置很可能会获得目标波特率。以下算法给出了一个得到 DLM、DLL、MULVAL 和 DIVADDVAL 参数集的方法。该参数集允许实际波特率与目标波特率之间存在小于 1.1%的相对误差。

图70. 设置 UART 分频器的算法

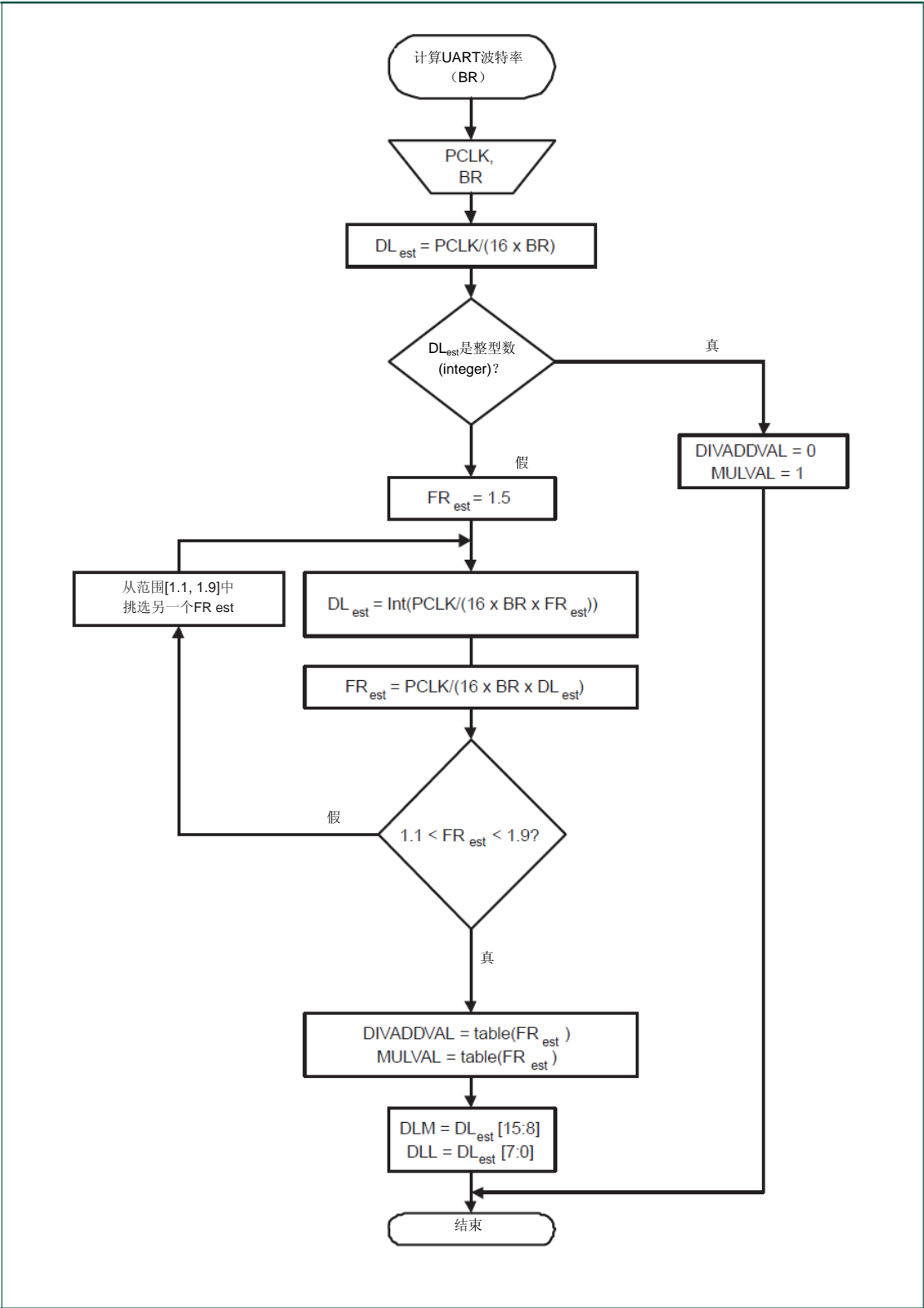


表384. 分数分频器设置查找表

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

17.5.16.1.1 示例 1: PCLK=14.7456 MHz，BR=9600

根据提供的算法， $DL_{est}=PCLK/(16 \times BR)=14.7456\text{ MHz}/(16 \times 9600)=96$ 。因为该算式中的 DL_{est} 是一个整型数，所以 $DIVADDVAL=0$ ， $MULVAL=1$ ， $DLM=0$ 且 $DLL=96$ 。

17.5.16.1.2 示例 2: PCLK=12 MHz，BR=115200

根据提供的算法， $DL_{est}=PCLK/(16 \times BR)=12\text{ MHz}/(16 \times 115200)=6.51$ 。该算式中的 DL_{est} 不是整型数，因此下一步要估算 FR 参数。使用 $FR_{est}=1.5$ 进行首次估算，得到新的 $DL_{est}=4$ ，然后重新计算 FR_{est} 得到 $FR_{est}=1.628$ 。因为 $FR_{est}=1.628$ 是在 1.1 到 1.9 的指定范围之内，因此 $DIVADDVAL$ 和 $MULVAL$ 的值可通过附带的查找表获得。

在表 384 中，最接近 $FR_{est}=1.628$ 的值是 $FR=1.625$ 。也就是说， $DIVADDVAL=5$ ，而 $MULVAL=8$ 。

基于这些查找结果，建议将 UART 设置为： $DLM=0$ ， $DLL=4$ ， $DIVADDVAL=5$ 和 $MULVAL=8$ 。根据公式 2，UART 的波特率为 115384。该速率与原来指定的 115200 之间存在 0.16% 的相对误差。

17.5.17UART1 发送使能寄存器（U1TER—0x4001 0030）

除配备了完整的硬件流控制（上述的 Auto-CTS 和 Auto-RTS 机制）外，U1TER 还可以实现软件流控制。当 TxEn=1 时，只要数据可用，UART1 发送器就会一直发送数据。一旦 TxEn 变为 0，UART1 就会停止数据传输。

[表 385](#) 描述了如何使用 TxEn 位来实现硬件流控制。但我们极力建议用户采用 UART1 硬件所实现的自动流控制功能处理硬件流控制，并限制 TxEn 位对软件流控制的范围。

U1TER 可以实现软件和硬件流控制。当 TXEn=1 时，只要数据可用，UART1 发送器就会一直发送数据。一旦 TXEn 变为 0，UART1 就会停止数据传输。

表385. UART1 发送使能寄存器位描述（U1TER – 0x4001 0030）

位	符号	描述	复位值
6:0	-	保留。读取值未定义，只写入 0。	无
7	TXEN	该位为 1 时（复位后），一旦先前的数据都被发送出去后，写入 THR 的数据就会在 TXD 管脚上输出。如果在发送某字符时该位被清零，那么将该字符发送完毕后就不再发送数据，直到该位再次被置位。也就是说，该位为 0 时会阻止字符从 THR 或 TX FIFO 传输到发送移位寄存器。当检测到硬件握手 TX-permit 信号（CTS）变为假时，或者接收到 XOFF 字符（DC3）时，软件通过执行软件握手可将该位清零。当检测到 TX-permit 信号为真时，或者在接收到 XON（DC1）字符时，软件又能将该位重新置位。	1
31:8	-	保留。读取值未定义，只写入 0。	无

17.5.18UART1 RS485 控制寄存器（U1RS485CTRL—0x4001 004C）

U1RS485CTRL 寄存器控制 UART 在 RS-485/EIA-485 模式下的配置。

表386. UART1 RS485 控制寄存器位描述（U1RS485CTRL—0x4001 004C）

位	符号	值	描述	复位值
0	NMMEN	0	禁能 RS-485/EIA-485 普通多点模式（NMM）。	0
		1	使能 RS-485/EIA-485 普通多点模式。在该模式下，当接收字节的校验位为 1 时，对地址进行检测，从而产生一个接收数据中断。参见 17.5.21 节。	
1	RXDIS	0	使能接收器。	0
		1	禁能接收器。	
2	AADEN	0	禁止自动地址检测（AAD）。	0
		1	使能自动地址检测。	
3	SEL	0	如果使能了方向控制（位 DCTRL=1），管脚 $\overline{\text{RTS}}$ 会被用于方向控制。	0
		1	如果使能了方向控制（位 DCTRL=1），管脚 $\overline{\text{DTR}}$ 会被用于方向控制。	
4	DCTRL	0	禁能自动方向控制。	0
		1	使能自动方向控制。	
5	OINV		该位保留了 RTS（或 DTR）管脚上方向控制信号的极性。	0
		0	当发送器有数据要发送时，方向控制管脚会被驱动为逻辑“0”。在最后一个数据位被发送出去后，该位就会被驱动为逻辑“1”。	
		1	当发送器有数据要发送时，方向控制管脚会被驱动为逻辑“1”。在最后一个数据位被发送出去后，该位就会被驱动为逻辑“0”。	
31:6	-		保留。读取值未定义，只写入 0。	无

17.5.19 UART1 RS-485 地址匹配寄存器（U1RS485ADRMATCH -0x4001 0050）

U1RS485ADRMATCH 寄存器包含了 RS-485/EIA-485 模式的地址匹配值。

表387. UART1 RS-485 地址匹配寄存器位描述（U1RS485ADRMATCH—0x4001 0050）

位	符号	描述	复位值
7:0	ADRMATCH	包含了地址匹配值。	0
31:8	-	保留。读取值未定义，只写入 0。	无

17.5.20 UART1 RS-485 延时值寄存器（U1RS485DLY—0x4001 0054）

对于最后一个停止位离开 TXFIFO 到使 $\overline{\text{RTS}}$ （或 $\overline{\text{DTR}}$ ）信号无效之间的延时，用户可对 8 位的 RS485DLY 寄存器进行编程。该延时时间是以波特率时钟周期为单位的。可以设定任何从 0 到 255 位时间的延时。

表388. UART1 RS-485 延时值寄存器位描述（U1RS485DLY—0x4001 0054）

位	符号	描述	复位值
7:0	DLY	包含了方向控制（RTS 或 DTR）延时值。该寄存器与一个 8 位计数器一起工作。	0
31:8	-	保留。读取值未定义，只写入 0。	无

17.5.21 RS-485/EIA-485 操作模式

RS-485/EIA-485 特性允许将 UART 配置为可寻址从机。可寻址从机是由同一主机控制的多个从机之一。

UART 主机发送器通过将校验位设置为“1”来标识地址字符。对于数据字符，校验位会被设置为“0”。

每个 UART 从机接收器都会被配给一个唯一地址。从机可编程为手动或自动地丢弃那些地址与本机地址不符的数据。

RS-485/EIA-485 普通多点模式（NMM）

该模式是通过置位 RS485CTRL 的位 0 来启用的。在该模式下，校验位被用于另一个用途，即区分所接收数据中的地址和数据。

如果接收器被禁能（RS485CTRL 位 1=“1”），所有接收到的字节都会被忽略且不会存储在 RXFIFO 中。当检测到一个地址字节（校验位=“1”）时，接收到的数据会被存储在 RXFIFO 中，并生成 Rx 数据就绪中断。此时，处理器可以读出地址字节，并决定是否使能接收器接收后面的数据。

当接收器被使能（RS485CTRL 位 1=“0”）时，所有接收到的字节（无论是数据还是地址）都会被接收并存储在 RXFIFO 中。

RS-485/EIA-485 自动地址检测（AAD）模式

如果同时设置 RS485CTRL 寄存器的位 0（9 位模式使能）和位 2（AAD 模式使能），则 UART 处于自动地址检测模式。

在该模式下，接收器会将接收到的所有地址字节（校验位=“1”）与 RS485ADRMATCH 寄存器中编程的 8 位值进行比较。

如果接收器被禁能（RS485CTRL 位 1=“1”），则任何接收到的字节，无论是数据字节还是地址字节，只要与 RS485ADRMATCH 的值不匹配，都会被丢弃。

当检测到匹配的地址字符时，地址字符和校验位就会被一起存储到 RXFIFO 中，此外，接收器将会自动使能（RS485CTRL 位 1 将会由硬件清零）。接收器还会产生 n Rx 数据就绪中断。

当接收器被使能（RS485CTRL 位 1=“0”）时，所有接收到的字节都会被接收并存储在 RXFIFO 中，直到接收到的地址字节与 RS485ADRMATCH 的值不匹配为止。出现不匹配的地址字节时，接收器会通过硬件自动禁能（RS485CTRL 位 1 将会置位）。所接收到的非匹配地址字符不会存储在 RXFIFO 中。

RS-485/EIA-485 自动方向控制

RS485/EIA-485 模式包含有一个选项，允许发送器自动控制 $\overline{\text{RTS}}$ （或 $\overline{\text{DTR}}$ ）管脚的状态作为一个方向控制输出信号。

该特性是通过将 RS485CTRL 的位 4 设置为“1”来使能的。

如果方向控制使能，当 RS485CTRL 的位 3 等于“0”时，则使用 $\overline{\text{RTS}}$ 管脚。当 RS485CTRL 的位 3=“1”时，则使用 $\overline{\text{DTR}}$ 管脚。

自动方向控制使能后，当 CPU 写数据到 TXFIFO 时，所选管脚变为有效（驱动为低电平）。最后一个数据位一旦被发送出去，该管脚就会被设为无效（驱动为高电平）。参见 RS485CTRL 寄存器的位 4 和位 5。

除了回写模式，RS485CTRL 的位 4 对 RTS（或 DTR）的控制优先于其他所有机制。

RS485/EIA-485 驱动器延时

驱动器延时是指最后一个停止位离开 TXFIFO 到使 $\overline{\text{RTS}}$ （或 $\overline{\text{DTR}}$ ）信号无效之间的延时。该延时可以在 8 位 RS485DLY 寄存器中进行编程。该延时时间是以波特率时钟周期为单位的。可以设定任何从 0 到 255 位时间的延时。

RS485/EIA-485 输出反相

$\overline{\text{RTS}}$ （或 $\overline{\text{DTR}}$ ）管脚上的方向控制信号的极性可通过对 U1RS485CTRL 寄存器中的位 5 进行编程来实现反相。当该位置位时，方向控制管脚将在发送器有数据要发送时就会驱动为逻辑 1。在最后一个数据发送出去后，方向控制管脚会驱动为逻辑 0。

18.1 如何阅读本章

LPC178x/177x 系列中的大部分设备都包含 5 个 UART。少数设备仅包含 4 个 UART。有关详细信息,请参阅具体的设备数据手册。UART0、2 和 3 与 UART1 基本相同,但没有 Modem/流控制信号。在下一章介绍的 UART4 中,新增了同步模式和智能卡模式。

18.2 基本配置

UART0/2/3 外设的配置需要使用下列寄存器:

1. 功率: 在 PCONP 寄存器 ([表 37](#)) 中, 设置 PCUART0/2/3 位。
注: 复位时, UART0 会被使能(PCUART0=1), 而 UART2/3 会被禁能(PCUART2/3=0)。
2. 外设时钟: 这些 UART 使用公共 PCLK, 它既用于总线接口, 也用于大多数 APB 外设的功能部分。参见 [4.6.4](#) 节。
3. 波特率: 在寄存器 U0/2/3LCR ([表 399](#)) 中, 将位 DLAB 设置为 1, 从而能够对寄存器 DLL ([表 393](#)) 和 DLM ([表 394](#)) 进行访问, 以设置波特率。同时, 如果有需要的话, 还可以在分数分频器寄存器 ([表 403](#)) 中设置分数波特率。
4. UART FIFO: 在寄存器 U0/2/3FCR ([表 398](#)) 中使用 FIFO 使能位(位 0)可使能 FIFO。
5. 管脚: 通过相关的 IOCON 寄存器 ([8.4.1](#) 节) 选择 UART 管脚和管脚模式。
注: UART 接收管脚不应使能下拉电阻。
5. 中断: 要使能 UART 中断, 在寄存器 U0/2/3LCR ([表 399](#)) 中将位 DLAB 设置为 0。由此使能了对 U0/2/3IER ([表 395](#)) 的访问。利用相应的中断置位使能寄存器来使能 NVIC 中的中断。
6. DMA: UART0/2/3 的发送和接收功能可通过 GPDMA 控制器进行操作 (参见[表 664](#))。

18.3 特性

- 数据大小为 5、6、7 和 8 位。
- 奇偶生成和校验：奇校验（Odd）、偶校验（Even）、1 校验（Mark）、0 校验（Space）或无校验（None）。
- 一个或两个停止位。
- 16 字节收发 FIFO。
- 内置波特率发生器，包含一个多功能的分数波特率分频器。
- 支持 DMA 发送和接收。
- 自动波特率功能。
- 中断生成和检测。
- 多处理器寻址模式。
- 支持软件流控制。
- 支持 RS-485/EIA-485。

18.4 结构

UART0、2 和 3 的结构如下面的框图所示。

APB 接口提供 CPU 或主机与 UART 之间的通信连接。

UARTn 接收器模块（UnRX）监控串行输入线 RXDn 的有效输入。UARTn RX 移位寄存器（UnRSR）通过 RXDn 接收有效字符。当 UnRSR 汇编完一个有效字符时，它将该字符传送到 UARTn RX 缓冲寄存器 FIFO 中，等待 CPU 或主机通过通用主机接口进行访问。

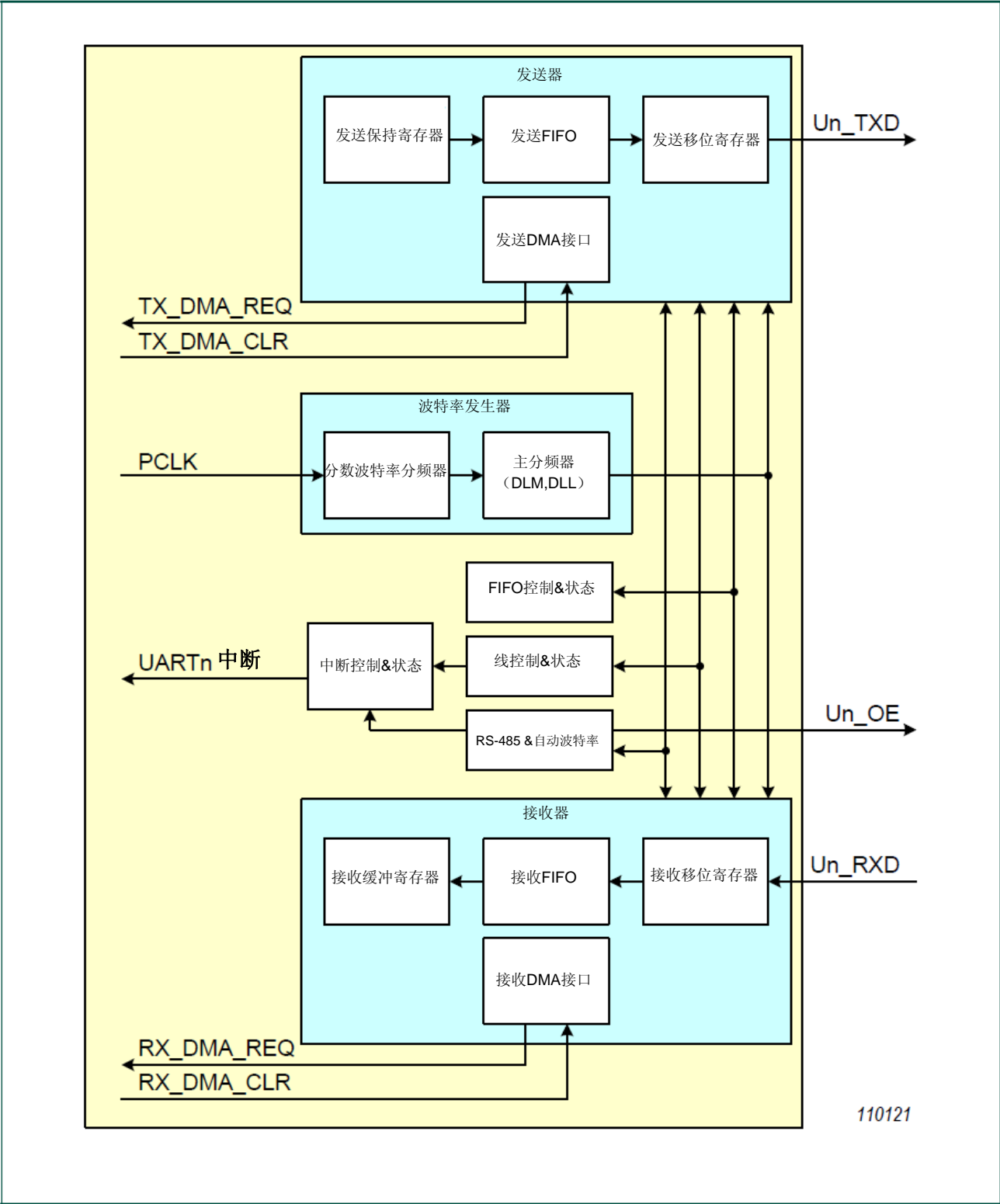
UARTn 发送器模块 UnTX 接收 CPU 或主机写入的数据，并且将数据缓存到 UARTn TX 保持寄存器 FIFO（UnTHR）中。UARTn TX 移位寄存器（UnTSR）读取 UnTHR 中存储的数据，并对数据进行汇编，通过串行输出管脚 TXDn 将数据发送出去。

UARTn 波特率发生器模块 UnBRG，产生 UARTn TX 模块所使用的时序使能信号。UnBRG 时钟输入源为 APB 时钟（PCLK）。主时钟由 UnDLL 和 UnDLM 寄存器中指定的除数相除。得到的时钟是 16x 过采样时钟。

中断接口包括 UnIER 和 UnIIR 寄存器。中断接口接收若干个 UnTX 和 UnRX 模块发出的单时钟宽度使能信号。

UnTX 和 UnRX 发送的状态信息会存储在 UnLSR 中。UnTX 和 UnRX 的控制信息会存储在 UnLCR 中。

图71. UART0、2、3 框图



18.5 管脚描述

表389. UARTn 管脚描述

管脚	类型	描述
U0_RXD, U2_RXD, U3_RXD	输入	串行输入管脚。串行接收数据。
U0_TXD, U2_TXD, U3_TXD	输出	串行输出管脚。串行发送数据。
U0_OE, U2_OE, U3_OE	输出	输出使能。 RS-485/EIA-485 输出使能。

18.6 寄存器描述

每个 UART 所包含的寄存器如表 390 中所示。除数锁存器访问位（DLAB）在 UnLCR7 中，能够使能对除数锁存器的访问。

表390. UART0/2/3 寄存器映射

通用名称	描述	访问	复位值 ^[1]	UARTn 寄存器名称&地址	表
RBR (DLAB =0)	接收缓冲寄存器。内含下一个要读取的已接收字符。	RO	无	U0RBR - 0x4000 C000 U2RBR - 0x4009 8000 U3RBR - 0x4009 C000	表 391
THR (DLAB =0)	发送保持寄存器。在此写入下一个要发送的字符。	WO	无	U0THR - 0x4000 C000 U2THR - 0x4009 8000 U3THR - 0x4009 C000	表 392
DLL (DLAB =1)	除数锁存器 LSB。波特率除数值的最低有效字节。 分数波特率分频器是使用整个除数来产生波特率的。	R/W	0x01	U0DLL - 0x4000 C000 U2DLL - 0x4009 8000 U3DLL - 0x4009 C000	表 393
DLM (DLAB =1)	除数锁存器 MSB。波特率除数值的最高有效字节。 分数波特率分频器是使用整个除数来产生波特率的。	R/W	0	U0DLM - 0x4000 C004 U2DLM - 0x4009 8004 U3DLM - 0x4009 C004	表 394
IER (DLAB =0)	中断使能寄存器。包含 7 个独立的中断使能位对应 7 个潜在的 UART 中断。	R/W	0	U0IER - 0x4000 C004 U2IER - 0x4009 8004 U3IER - 0x4009 C004	表 395
IIR	中断 ID 寄存器。识别处于挂起状态的中断。	RO	0x01	U0IIR - 0x4000 C008 U2IIR - 0x4009 8008 U3IIR - 0x4009 C008	表 396
FCR	FIFO 控制寄存器。控制 UART FIFO 的使用和模式。	WO	0	U0FCR - 0x4000 C008 U2FCR - 0x4009 8008 U3FCR - 0x4009 C008	表 398
LCR	线控制寄存器。包含帧格式控制和间隔产生控制。	R/W	0	U0LCR - 0x4000 C00C U2LCR - 0x4009 800C U3LCR - 0x4009 C00C	表 399
LSR	线状态寄存器。包含发送和接收的状态标志（包括线错误）。	RO	0x60	U0LSR - 0x4000 C014 U2LSR - 0x4009 8014 U3LSR - 0x4009 C014	表 400
SCR	高速缓存寄存器。8 位的临时存储空间，供软件使用。	R/W	0	U0SCR - 0x4000 C01C U2SCR - 0x4009 801C U3SCR - 0x4009 C01C	表 401
ACR	自动波特率控制寄存器。包含了自动波特率的特性控制。	R/W	0	U0ACR - 0x4000 C020 U2ACR - 0x4009 8020 U3ACR - 0x4009 C020	表 402

通用名称	描述	访问	复位值 ^[1]	UARTn 寄存器名称&地址	表
FDR	分数分频器寄存器。为波特率分频器产生时钟输入。	R/W	0x10	U0FDR - 0x4000 C028 U2FDR - 0x4009 8028 U3FDR - 0x4009 C028	表 403
TER	发送使能寄存器。关闭 UART 发送器，使用软件流控制。	R/W	0x80	U0TER - 0x4000 C030 U2TER - 0x4009 8030 U3TER - 0x4009 C030	表 405
RS485CTRL	RS-485/EIA-485 控制。包含了 RS-485/EIA-485 模式多方面的配置控制。	R/W	0	U0RS485CTRL - 0x4000 C04C U2RS485CTRL - 0x4009 804C U3RS485CTRL - 0x4009 C04C	表 406
ADRMATCH	RS-485/EIA-485 地址匹配。包含了 RS-485/EIA-485 模式所使用的地址匹配值。	R/W	0	U0ADRMATCH - 0x4000 C050 U2ADRMATCH - 0x4009 8050 U3ADRMATCH - 0x4009 C050	表 407
RS485DLY	RS-485/EIA-485 方向控制延迟。	R/W	0	U0RS485DLY - 0x4000 C054 U2RS485DLY - 0x4009 8054 U3RS485DLY - 0x4009 C054	表 408

[1] 复位值仅指使用位中保存的数据，它不包括保留位的内容。

18.6.1 UARTn 接收器缓冲寄存器(U0RBR—0x4000 C000, U2RBR—0x4009 8000, U3RBR—0x4009 C000, DLAB = 0)

UnRBR 是 UARTn RX FIFO 的最高字节。它包含了最早接收到的字符，并且可通过总线接口进行读取。LSB（位 0）代表“最早”接收到的数据位。如果接收到的字符少于 8 位，则未使用的 MSB 使用 0 填充。

如果要访问 UnRBR，LCR 中的除数锁存器访问位（DLAB）必须为 0。UnRBR 始终为只读寄存器。

由于 PE、FE 和 BI 位与 RBR FIFO 顶部的字节（即下次读取 RBR 时获取的字节）相关，因此，要正确地成对读出有效的已接收字节及其状态位，应先读取 U0LSR 寄存器的内容，然后读取 UnRBR 中的字节。

表391. UARTn 接收器缓冲寄存器（U0RBR—0x4000 C000, U2RBR—0x4009 8000, U3RBR—04009 C000, DLAB = 0）位描述

位	符号	描述	复位值
7:0	RBR	UARTn 接收缓冲寄存器包含了 UARTn Rx FIFO 当中最早接收到的字节。	未定义
31:8	-	保留，从保留位读出的值未定义。	无

18.6.2 UARTn 发送保持寄存器（U0THR—0x4000 C000, U2THR—0x4009 8000, U3THR—0x4009 C000, DLAB = 0）

UnTHR 是 UARTn TX FIFO 的最高字节。它是 TX FIFO 中的最新字符，可以通过总线接口写入。LSB 代表第一个要发送出去的位。

如果要访问 UnTHR，UnLCR 中的除数锁存器访问位（DLAB）必须为 0。UnTHR 始终是只写寄存器。

表392. UARTn 发送保持寄存器位描述（U0THR—0x4000 C000，U2THR—0x4009 8000，U3THR —0x4009 C000，DLAB = 0）

位	符号	描述	复位值
7:0	THR	写 UARTn 发送保持寄存器会使数据保存到 UARTn 发送 FIFO 中。 当字节达到 FIFO 的底部并且发送器就绪时，字节就会被发送。	无
31:8	-	保留。读取值未定义，只写入 0。	无

18.6.3 UARTn 除数锁存器 LSB 寄存器（U0DLL—0x4000 C000，U2DLL—0x4009 8000，U3DLL—0x4009 C000，DLAB = 1）和 UARTn 除数锁存器 MSB 寄存器（U0DLM—0x4000 C004，U2DLM—0x4009 8004，U3DLM—0x4009 C004，DLAB = 1）

UARTn 除数锁存器是 UARTn 波特率发生器的一部分，它与分数分频器一同使用，保持产生波特率时钟的 APB 时钟(PCLK)的分频值，波特率时钟必须是所需波特率的 16x 倍。UnDLL 和 UnDLM 寄存器一起构成一个 16 位除数，其中 UnDLL 包含除数的低 8 位，而 UnDLM 包含除数的高 8 位。分频值 0x0000 会被作为 0x0001 处理，因为除数不能为 0。如果要访问 UARTn 除数锁存器，UnLCR 中的除数锁存器访问位（DLAB）必须为 1。有关如何为 UnDLL 和 UnDLM 选择正确值的详细信息，参见 [18.6.11](#) 节。

表393. UARTn 除数锁存器 LSB 寄存器位描述（U0DLL—0x4000 C000，U2DLL—0x4009 8000，U3DLL —0x4009 C000，DLAB = 1）

位	符号	描述	复位值
7:0	DLLSB	UARTn 除数锁存器 LSB 寄存器与 UnDLM 寄存器一起决定 UARTn 的波特率。	0x01
31:8	-	保留。读取值未定义，只写入 0。	无

表394. UARTn 除数锁存器 MSB 寄存器位描述（U0DLM—0x4000 C004，U2DLL—0x4009 8004，U3DLL —0x4009 C004，DLAB = 1）

位	符号	描述	复位值
7:0	DLMSB	UARTn 除数锁存器 MSB 寄存器与 U0DLL 寄存器一起决定 UARTn 的波特率。	0
31:8	-	保留。读取值未定义，只写入 0。	无

18.6.4 UARTn 中断使能寄存器（U0IER—0x4000 C004，U2IER—0x4009 8004，U3IER—0x4009 C004，DLAB = 0）

UnIER 用于使能三个 UARTn 中断源。

表395. UARTn 中断使能寄存器位描述（U0IER—0x4000 C004，U2IER—0x4009 8004，U3IER—0x4009 C004，DLAB = 0）

位	符号	值	描述	复位值
0	RBR Interrupt Enable	0	使能 UARTn 的接收数据可用中断。它还控制着字符接收超时中断。	0
		0	禁止 RDA 中断。	

位	符号	值	描述	复位值
1	THRE Interrupt Enable	1	使能 RDA 中断。	0
		0	禁止 THRE 中断。	
		1	使能 THRE 中断。	
2	RX Line Status Interrupt Enable	1	使能 UARTn 的 RX 线状态中断。该中断的状态可从 UnLSR[4:1]中读出。	0
		0	禁止 RX 线状态中断。	
		1	使能 RX 线状态中断。	
7:3	-		保留。读取值未定义，只写入 0。	无
8	ABEOIntEn		使能 auto-baud 结束中断。	0
		0	禁止 auto-baud 结束中断。	
		1	使能 auto-baud 结束中断。	
9	ABTOIntEn		使能 auto-baud 超时中断。	0
		0	禁止 auto-baud 超时中断。	
		1	使能 auto-baud 超时中断。	
31:10	-		保留。读取值未定义，只写入 0。	无

18.6.5 UARTn 中断标识寄存器（U0IIR—0x4000 C008，U2IIR—0x4009 8008，U3IIR—0x4009 C008）

UnIIR 提供一个状态代码，用于指示一个挂起中断的优先级和中断源。在访问 UnIIR 的过程中，中断被冻结。如果在访问 UnIIR 的过程中产生了中断，该中断会被记录以用于下次 UnIIR 访问。

表396. UARTn 中断标识寄存器位描述（U0IIR —0x4000 C008，U2IIR—0x4009 8008，U3IIR—0x4009 C008）

位	符号	值	描述	复位值
0	IntStatus		中断状态。注：UnIIR[0]为低电平有效。挂起的中断可通过 UnIIR[3:1]确定。	1
		0	至少有一个中断被挂起。	
		1	没有挂起的中断。	
3:1	IntId		中断标识。UnIER[3:1]指示对应 UARTn Rx 或 TX FIFO 的中断。下面未列出的 UnIER[3:1]的其他组合都为保留值（000,100,101,111）。	0
		011	1 – 接收线状态(RLS)	
		010	2a – 接收数据可用 (RDA)	
		110	2b – 字符超时指示器 (CTI)	
		001	3 – THRE 中断	
5:4	-		保留。读取值未定义，只写入 0。	无
7:6	FIFO Enable		这些位等效于 UnFCR[0]。	0
8	ABEOInt		Auto-baud 结束中断。若 auto-baud 已成功结束且中断被使能，则为真。	0
9	ABTOInt		Auto-baud 超时中断。若 auto-baud 已超时且中断被使能，则为真。	0
31:10	-		保留。读取值未定义，只写入 0。	无

位 UnIIR[9:8]由自动波特率功能设置，用于发布超时信号或自动波特率结束条件信号。而自动波特率中断条件的清除是通过设置自动波特率控制寄存器中的相应清除（Clear）位来实现的。

如果 `IntStatus` 位为 1，表示没有中断被挂起，此时 `IntId` 位为 0。如果 `IntStatus` 位为 0，表示有一个非自动波特率中断被挂起，此时 `IntId` 位会标识中断的类型以及[表 397](#)中所示的处理。知道了 `UnIIR[3:0]`的状态，中断处理程序就能确定中断原因以及如何清除有效的中断。在退出中断服务程序之前，必须读取 `UnIIR` 来清除中断。

UARTn RLS 中断 (`UnIIR[3:1]=011`) 是最高优先级的中断，只要 UARTn RX 输入上产生以下四个错误条件中的任意一个，该位就会被置位：溢出错误 (OE)、校验错误 (PE)、帧错误 (FE) 以及间隔中断 (BI)。通过 `UnLSR[4:1]`可查看设置中断的 UARTn Rx 错误条件。读取 `UnLSR` 时，中断会被清除。

UARTn RDA 中断 (`UnIIR[3:1]=010`) 与 CTI 中断 (`UnIIR[3:1]=110`) 共用第二优先级。当 UARTn Rx FIFO 深度到达 `UnFCR[7:6]`中定义的触发值时,RDA 就会被激活,而当 UARTn Rx FIFO 深度低于触发值时, RDA 复位。当 RDA 中断有效时, CPU 可读出由触发值定义的一个数据块。

CTI 中断 (`UnIIR[3:1]=110`) 是一个第二优先级中断，当 UARTn Rx FIFO 内含有至少一个字符并且在接收到 3.5 到 4.5 字符时间内没有发生 UARTn Rx FIFO 动作时，该中断被设置。任何 UARTn Rx FIFO 动作 (UARTn RSR 的读取或写入) 都会清除该中断。当接收到的消息不是触发值大小的倍数时，该中断会清空 UARTn RBR。例如，如果外设想要发送一条长度为 105 个字符的消息，而触发值为 10 个字符，那么 CPU 接收到 10 个 RDA 中断的结果是 100 个字符的传输，而接收 1 至 5 个 CTI 中断 (取决于服务程序) 的结果是剩下 5 个字符的传输。

表397. UARTn 中断处理

U0IIR[3:0]值 ^[1]	优先级	中断类型	中断源	中断复位
0001	-	无	无	-
0110	最高	RX 线状态/错误	OE ^[2] 、PE ^[2] 、FE ^[2] 或 BI ^[2]	UnLSR 读操作 ^[2]
0100	第二	RX 数据可用	Rx 数据可用或达到 FIFO 的触发点 (<code>UnFCR0=1</code>)	UnRBR 读操作 ^[3] 或 UARTn FIFO 低于触发值
1100	第二	字符超时指示	RX FIFO 中至少有一个字符，并且在一段时间内没有字符输入或移出，该时间的长短取决于 FIFO 中的字符数以及在 3.5 到 4.5 字符的时间内的触发值。 实际时间为：[(字长度) × 7-2] × 8 + [(触发值-字符数) × 8 + 1] RCLKs	UnRBR 读操作 ^[3]
0010	Third	THRE	THRE ^[2]	UnIIR 读操作 (如果 UnIIR 是中断源) 或 THR 写操作 ^[4]

[1] “0000”、“0011”、“0101”、“0111”、“1000”、“1001”、“1010”、“1011”、“1101”、“1110”、“1111” 均为保留值。

[2] 详细信息，参见 [18.6.8](#)。

[3] 详细信息，参见 [18.6.1](#)。

[4] 详细信息，参见 [18.6.5](#) 和 [18.6.2](#)。

UARTn THRE 中断 (`UnIIR[3:1]=001`) 是第三优先级中断，当 UARTn THR FIFO 为空且满

足特定的初始化条件时，该中断激活。这些初始化条件是为了让 UARTn THR FIFO 有机会填入数据，以免在系统启动时产生许多 THRE 中断而规定的。当 THRE=1 时，且在上次 THRE=1 事件后，UnTHR 中没有同时出现至少两个字符的情况下，这些初始化条件就会实现一个字符延时减去停止位。当没有解码和服务 THRE 中断时，该延时为 CPU 提供了写数据到 UnTHR 的时间。如果 UARTn THR FIFO 中同时保持两个或更多字符，而当前的 UnTHR 为空时，THRE 中断会立即被设置。当发生 UnTHR 写操作或 UnHIR 读操作，并且 THRE 是最高优先级中断（UnHIR[3:1]=001）时，THRE 中断复位。

18.6.6 UARTn FIFO 控制寄存器（U0FCR—0x4000 C008, U2FCR—0x4009 8008, U3FCR—0x4009 C008）

UnFCR 为只写寄存器，控制 UARTn RX 和 TX FIFO 的操作。

表398. UARTn FIFO 控制寄存器位描述（U0FCR—0x4000 C008, U2FCR—0x4009 8008, U3FCR—0x4007 C008）

位	符号	值	描述	复位值
0	FIFO Enable	0	UARTn FIFO 被禁止。禁止在应用中使用。	0
		1	高电平有效，使能对 UARTn Rx 和 TX FIFO 以及 UnFCR[7:1]的访问。该位必须置位以实现正确的 UART 操作。该位的任何变化都将使相关的 UART FIFO 自动清空。	
1	RX FIFO Rese	0	对两个 UARTn FIFO 均无影响。	0
		1	写逻辑 1 到 UnFCR[1]将会清零 UARTn Rx FIFO 中所有字节，并复位指针逻辑。该位会自动清零。	
2	TX FIFO Reset	0	对两个 UARTn FIFO 均无影响。	0
		1	写逻辑 1 到 UnFCR[2]将会清零 UARTn TX FIFO 中所有字节，并复位指针逻辑。该位会自动清零。	
3	DMA Mode Select		当 FIFO 使能位（该寄存器的位 0）被置位时，该位选择 DMA 模式。参见 18.6.6.1 节。	0
5:4	-		保留。读取值未定义，只写入 0。	无
7:6	RX Trigger Level		这两个位决定了接收 UARTn FIFO 在激活中断或 DMA 请求前必须写入的字符数量。	0
		00	触发点 0（1 字符或 0x01）	
		01	触发点 1（4 字符或 0x04）	
		10	触发点 2（8 字符或 0x08）	
		11	触发点 3（14 字符或 0x0E）	
31:8	-		保留。读取值未定义，只写入 0。	无

18.6.6.1 DMA 操作

用户可以选择用 DMA 进行 UART 的发送和/或接收。DMA 模式由 FCR 寄存器中的 DMA 模式选择位（DMA Mode Select）决定。只有在 FCR 寄存器中的 FIFO 使能位使能了 FIFO 时，该位才有用。

UART 接收 DMA

在 DMA 模式下，当接收 FIFO 水平大于或等于触发值时，或者在发生一次字符超时的情况下，接收 DMA 请求才会生效。请参见上文中对 RX 触发值的描述。接收 DMA 请求由 DMA 控制器清除。

UART 发送 DMA

在 DMA 模式下，当发送 FIFO 变为未滿时，发送 DMA 请求就会生效。发送 DMA 请求由 DMA 控制器清除。

18.6.7 UARTn 线控制寄存器（U0LCR—0x4000 C00C，U2LCR—0x4009 800C，U3LCR—0x4009 C00C）

UnLCR 决定待发送或接收数据字符的格式。

表399. UARTn 线控制寄存器位描述（U0LCR—0x4000 C00C，U2LCR—0x4009 800C，U3LCR—0x4009 C00C）

位	符号	值	描述	复位值
1:0	Word Length Select	00	5 位字符长度	0
		01	6 位字符长度	
		10	7 位字符长度	
		11	8 位字符长度	
2	Stop Bit Select	0	1 个停止位。	0
		1	2 个停止位（当 UnLCR[1:0]=00 时，为 1.5 个）。	
3	Parity Enable	0	禁止奇偶产生和校验。	0
		1	使能奇偶产生和校验。	
5:4	Parity Select	00	奇校验。发送字符和添加的校验位所包含的“1”的数量是一个奇数。	0
		01	偶校验。发送字符和添加的校验位所包含的“1”的数量是一个偶数。	
		10	强制为“1”stick 校验。	
		11	强制为“0”stick 校验。	
6	Break Control	0	禁止间隔发送。	0
		1	使能间隔发送。当 UnLCR[6]为高电平（有效）时，输出管脚 UARTn TXD 强制为逻辑 0。	
7	Divisor Latch	0	禁止访问除数锁存器。	0
	Access Bit (DLAB)	1	使能访问除数锁存器。	
31:8	-		保留。读取值未定义，只写入 0。	无

18.6.8 UARTn 线状态寄存器（U0LSR—0x4000 C014，U2LSR—0x4009 8014，U3LSR—0x4009 C014）

UnLSR 是一个只读寄存器，提供 UARTn TX 和 RX 模块的状态信息。

表400. UARTn 线状态寄存器位描述（U0LSR—0x4000 C014，U2LSR—0x4009 8014，U3LSR—0x4009 C014）

位	符号	值	描述	复位值
0	Receiver Data Ready (RDR)		当 UnRBR 包含一个未读字符时，UnLSR[0]就会被置位；当 UARTn RBR FIFO 为 0 空时，UnLSR[0]就会被清零。	0
		0	UARTn 接收 FIFO 为空。	
		1	UARTn 接收 FIFO 不为空。	
1	Overrun Error (OE)		溢出错误条件在错误发生后立即设置。读 UnLSR 会清零 UnLSR[1]。当 UARTn RSR 0 已有新的字符汇集，而 UARTn RBR FIFO 已满时，UnLSR[1]会被置位。此时，UARTn RBR FIFO 将不会被覆盖，UARTn RSR 内的字符将会丢失。	0
		0	溢出错误状态未激活。	
		1	溢出错误状态激活。	

位	符号	值	描述	复位值
2	Parity Error (PE)		当接收字符的校验位处于错误状态时，校验错误就会产生。读 UnLSR 会清零 0 UnLSR[2]。校验错误检测时间取决于 UnFCR[0]。 注： 校验错误与在 UARTn RBR FIFO 顶部的字符相关。	
		0	校验错误状态未激活。	
		1	校验错误状态激活。	
3	Framing Error (FE)		当接收字符的停止位为逻辑 0 时，就会发生帧错误。读 UnLSR 会清零 UnLSR[3]。0 帧错误检测时间取决于 UnFCR[0]。当检测到有帧错误时，RX 会尝试与数据重新同步，并假设错误的停止位实际上是一个超前的起始位。但即使没有出现帧错误，它也无法假设下一个接收到的字符是正确的。 注： 帧错误与 UARTn RBR FIFO 顶部的字符相关。	
		0	帧错误状态未激活。	
		1	帧错误状态激活。	
4	Break Interrupt (BI)		在发送整个字符（起始位、数据、校验位以及停止位）过程中，RXDn 如果保持在 0 间隔状态（全“0”），则产生间隔中断。一旦检测到间隔条件，接收器立即进入空闲状态，直到 RXDn 进入标记状态（全“1”）。读 UnLSR 会清零该状态位。间隔检测时间取决于 UnFCR[0]。 注： 间隔中断与 UARTn RBR FIFO 顶部的字符相关。	
		0	间隔中断状态未激活。	
		1	间隔中断状态激活。	
5	Transmitter Holding Register Empty (THRE))		当检测到 UARTn THR 已空时，THRE 就会立即被设置。写 UnTHR 会清零 THRE。1	
		0	UnTHR 包含有效数据。	
		1	UnTHR 为空。	
6	Transmitter Empty (TEMT)		当 UnTHR 和 UnTSR 同时为空时，TEMT 就会被设置；而当 UnTSR 或 UnTHR 任一包含有效数据时，TEMT 就会被清零。	
		0	UnTHR 和/或 UnTSR 包含有效数据。	
		1	UnTHR 和 UnTSR 为空。	
7	Error in RX FIFO (RXFE)		当一个带有 Rx 错误（如：帧错误、校验错误或间隔中断）的字符载入到 UnRBR 中 0 时，UnLSR[7]就会被设置。当 UnLSR 寄存器被读取并且 UARTn FIFO 中不再有错误时，该位就会清零。	
		0	UnRBR 中没有 UARTn RX 错误或 UnFCR[0]=0。	
		1	UARTn RBR 包含至少一个 UARTn RX 错误。	
31:8	-		保留。从保留位读出的值未定义。	无

18.6.9 UARTn 高速缓存寄存器（U0SCR—0x4000 C01C，U2SCR—0x4009 801C U3SCR—0x4009 C01C）

UnSCR 不会对 UARTn 操作有影响。用户可以自由地读写该寄存器。不提供中断接口向主机指示 UnSCR 所发生的读或写操作。

表401. UARTn 高速缓存寄存器位描述（U0SCR—0x4000 C01C，U2SCR—0x4009 801C，U3SCR—0x4009 C01C）

位	符号	描述	复位值
7:0	Pad	一个可读、可写的字节。	0
31:8	-	保留。读取值未定义，只写入 0。	无

18.6.10 UARTn 自动波特率控制寄存器（U0ACR—0x4000 C020, U2ACR—0x4009 8020, U3ACR—0x4009 C020）

在用户测量波特率的输入时钟/数据速率期间，整个测量过程就是由 UARTn 自动波特率控制寄存器（UnACR）进行控制的。用户可以自由地读写该寄存器。

表402. UARTn 自动波特率控制寄存器位描述（U0ACR—0x4000 C020, U2ACR—0x4009 8020, U3ACR—0x4009 C020）

位	符号	值	描述	复位值
0	Start		在 auto-baud 功能结束后，该位会自动清零。	0
		0	Auto-baud 功能停止（auto-baud 功能不运行）。	
		1	Auto-baud 功能启动（auto-baud 功能正在运行）。Auto-baud 运行位。该位会在 auto-baud 功能结束后自动清零。	
1	Mode		Auto-baud 模式选择位。	0
		0	模式 0。	
		1	模式 1。	
2	AutoRestart	0	不重新启动。	0
		1	如果超时则重新启动（计数器会在下一个 UARTn Rx 下降沿重新启动）。	0
7:3	-		保留。读取值未定义，只写入 0。	无
8	ABEOIntClr		Auto-baud 结束中断清零位（仅可写访问）。写 1 将 UnIIR 中相应的中断清除。写 0 无影响。	0
9	ABTOIntClr		Auto-baud 超时中断清零位（仅可写访问）。写 1 将 UnIIR 中相应的中断清除。写 0 无影响。	0
31:10	-		保留。读取值未定义，只写入 0。	无

18.6.10.1 自动波特率

UARTn 自动波特率功能可用于测量基于“AT”协议（Hayes 命令）的输入波特率。如果自动波特率功能被使能，那么自动波特率功能将测量接收数据流中的一个位所消耗的时间，并根据这个结果来设置除数锁存器寄存器 UnDLM 和 UnDLL。

注：在自动波特率操作期间不会连接分数波特率分频器，所以在需要自动波特率功能时不应使用分数波特率分频器。

自动波特率功能通过置位 UnACR 起始位来启动的，并通过清零 UnACR 起始位来停止。自动波特率一旦结束，起始位就将自动清零，并且对该起始位进行读取将会返回自动波特率的状态（挂起/完成）。

可通过设置 UnACR 模式位来使用两种自动波特率测量模式。在模式 0 下，波特率是通过对 UARTn Rx 管脚上两个连续的下降沿进行测量（起始位的下降沿和最低有效位的下降沿）来得到的。在模式 1 下，波特率则是通过测量 UARTn Rx 管脚上的下降沿与后续的上升沿之间的时间（起始位的长度）来得到的。

如果发生超时（速率测量计数器溢出），UnACR AutoRestart 位可用于自动重启波特率测量。如果该位被置位，速率测量将会在 UARTn Rx 管脚的下一个下降沿重新启动。

自动波特率功能会产生两种中断：

- 如果使能了中断 (UnIER ABTOIntEn 置位且自动波特率测量计数器溢出), 则 UnIIR ABTOInt 中断将会被设置。
- 如果使能了中断 (UnIER ABEOIntEn 置位且自动波特率已成功完成), 则 UnIIR ABEOInt 中断将会被设置。

自动波特率中断必须通过置位相应的 UnACR ABTOIntClr 和 ABEOIntEn 位来清零。

在自动波特率期间, 分数波特率发生器通常会被禁用 (DIVADDVAL=0)。但是, 如果使能了分数波特率发生器 (DIVADDVAL>0), 那么它将影响 UARTn Rx 管脚波特率的测量, 但 UnFDR 寄存器的值在速率测量后不会被修改。此外, 当使用自动波特率时, 任何对 UnDLM 和 UnDLL 寄存器的写操作都必须在写 UnACR 寄存器之前完成。UARTn 支持的最小和最大波特率是 PCLK、数据位数、停止位数和校验位数的函数。

(3)

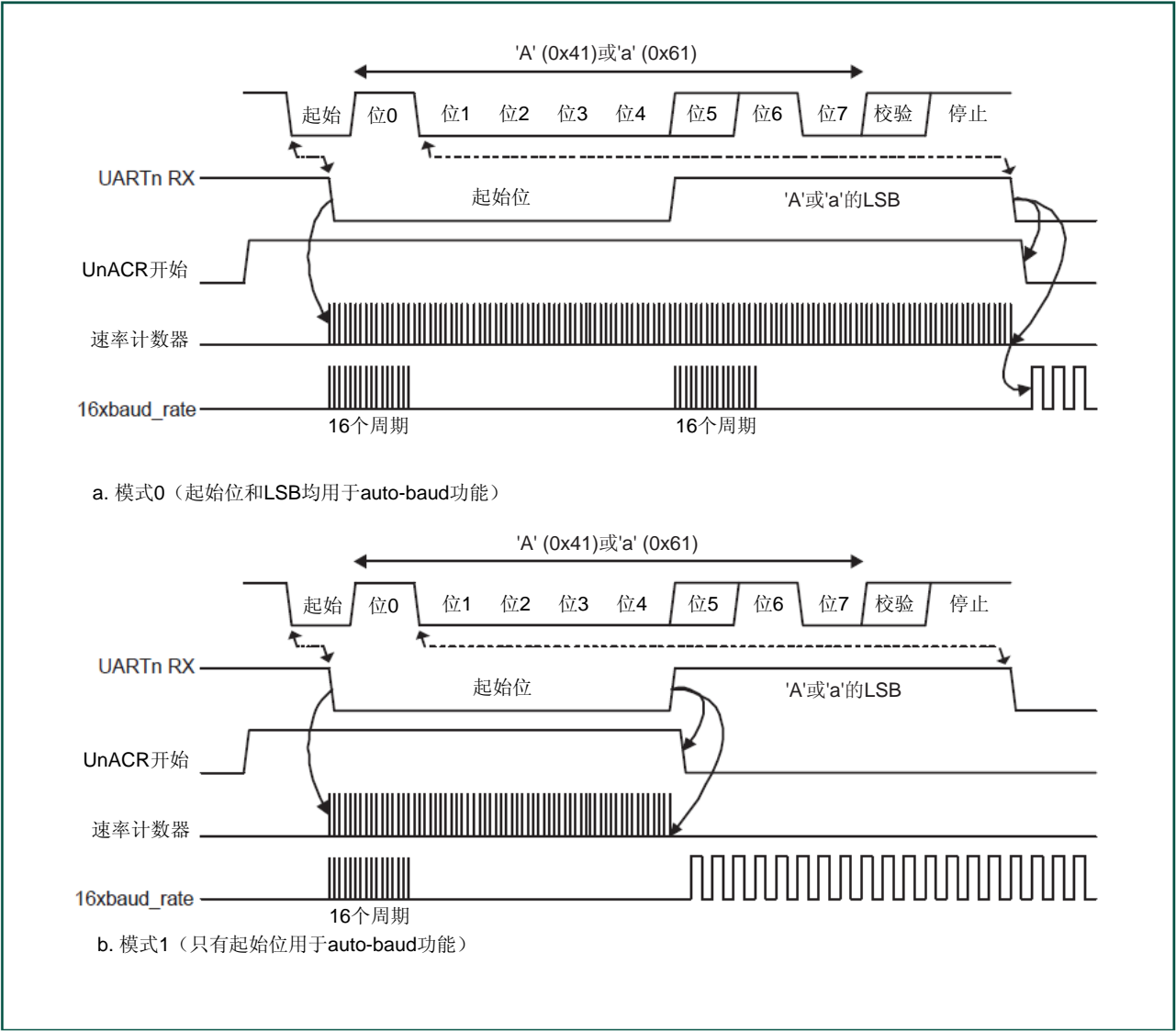
$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UARTn_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

18.6.10.2 自动波特率模式

当软件等待一个“AT”命令时, 它采用期望的字符格式配置 UARTn, 并置位 UnACR 起始位。用户不必关心除数锁存器 UnDLM 和 UnDLL 内的初始值。由于字母“A”或“a”的 ASCII 编码 (“A”=0x41, “a”=0x61) 的关系, UARTn Rx 管脚检测到的起始位和期望字符的 LSB 是由两个下降沿限定的。当 UnACR 起始位被置位时, 自动波特率协议将执行以下步骤:

1. UnACR 起始位一置位, 波特率测量计数器会复位, 同时 UARTn UnRSR 复位。UnRSR 波特率切换为最高速率。
2. UARTn Rx 管脚上的下降沿会触发起始位的开始。速率测量计数器将开始对 PCLK (可选择被分数波特率发生器预分频) 进行计数。
3. 在接收起始位的过程中, RSR 波特率输入端会产生 16 个脉冲, 频率与 (经分数波特率发生器预分频的) UARTn 输入时钟相同, 这样, 保证了起始位存储在 UnRSR 中。
4. 在接收起始位的过程中 (以及模式 0 下的字符 LSB), 速率计数器将继续随着被预分频的 UARTn 输入时钟 (PCLK) 而继续递增。
5. 如果 Mode=0, 则速率计数器将在 UARTn Rx 管脚的下一个下降沿停止。如果 Mode=1, 则速率计数器将在 UARTn Rx 管脚的下一个上升沿停止。
6. 速率计数器的值被载入到 UnDLM/UnDLL 中, 并且波特率将会切换到正常操作模式。在设置完 UnDLM/UnDLL 后, 如果自动波特率结束中断被使能, UnIIR ABEOInt 将会被置位。UnRSR 将继续接收“A/a”字符剩余的位。

图72. 自动波特率 a)模式 0 和 b)模式 1 波形图



18.6.11 UARTn 分数分频器寄存器 (U0FDR—0x4000 C028, U2FDR—0x4009 8028, U3FDR—0x4009 C028)

UARTn 分数分频器寄存器 (UnFDR) 控制着用于波特率生成的时钟预分频器，并且用户可自由对该寄存器进行读写操作。预分频器使用 APB 时钟并根据指定的分数要求产生一个输出时钟。

重要提示：如果分数分频器是有效的 (DIVADDVAL>0) 且 DLM=0，则 DLL 寄存器的值必须大于 2。

表403. UARTn 分数分频器寄存器位描述（U0FDR—0x4000 C028, U2FDR—0x4009 8028, U3FDR—0x4009 C028）

位	功能	值	描述	复位值
3:0	DIVADDVAL	0	产生波特率的预分频除数值。如果该字段为 0，分数波特率发生器将不会影响 UARTn 的波特率。	0
7:4	MULVAL	1	波特率预分频乘数值。不管是否使用分数波特率发生器，为了让 UARTn 正常工作，该字段必须大于或等于 1。	1
31:8	-		保留。读取值未定义，只写入 0。	0

该寄存器控制着用于波特率生成的时钟预分频器。该寄存器的复位值会使 UART 的分数功能保持在禁用状态，从而确保此 UART 完全软件和硬件兼容没有配备该特性的 UART。

UART 波特率的计算（n=0/2/3）:

(4)

$$UARTn_{baudrate} = \frac{PCLK}{16 \times (256 \times UnDLM + UnDLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

其中，PCLK 为外设时钟，UnDLM 和 UnDLL 为标准 UART 波特率分频器寄存器，而 DIVADDVAL 和 MULVAL 为 UART 分数波特率发生器的指定参数。

MULVAL 和 DIVADDVAL 的值应符合以下条件:

- 1. 1≤MULVAL≤15
- 2. 0≤DIVADDVAL≤14
- 3. DIVADDVAL<MULVAL

UnFDR 的值在发送/接收数据的过程中不应进行修改，否则可能导致数据丢失或损坏。

如果 UnFDR 寄存器的值不符合上述两个要求，则分数分频器输出为未定义。如果 DIVADDVAL 为 0，则分数分频器会被禁能，并且不会对时钟进行分频。

18.6.11.1 波特率计算

UARTn 可以与分数分频器一起工作，也可以不使用分数分频器。在实际应用中，可能使用几种不同的分数分频器设置都可以获得目标波特率。以下算法给出了一个得到 DLM、DLL、MULVAL 和 DIVADDVAL 参数集的方法。该参数集允许实际波特率与目标波特率之间存在小于 1.1%的相对误差。

图73. 设置 UART 分频器的算法

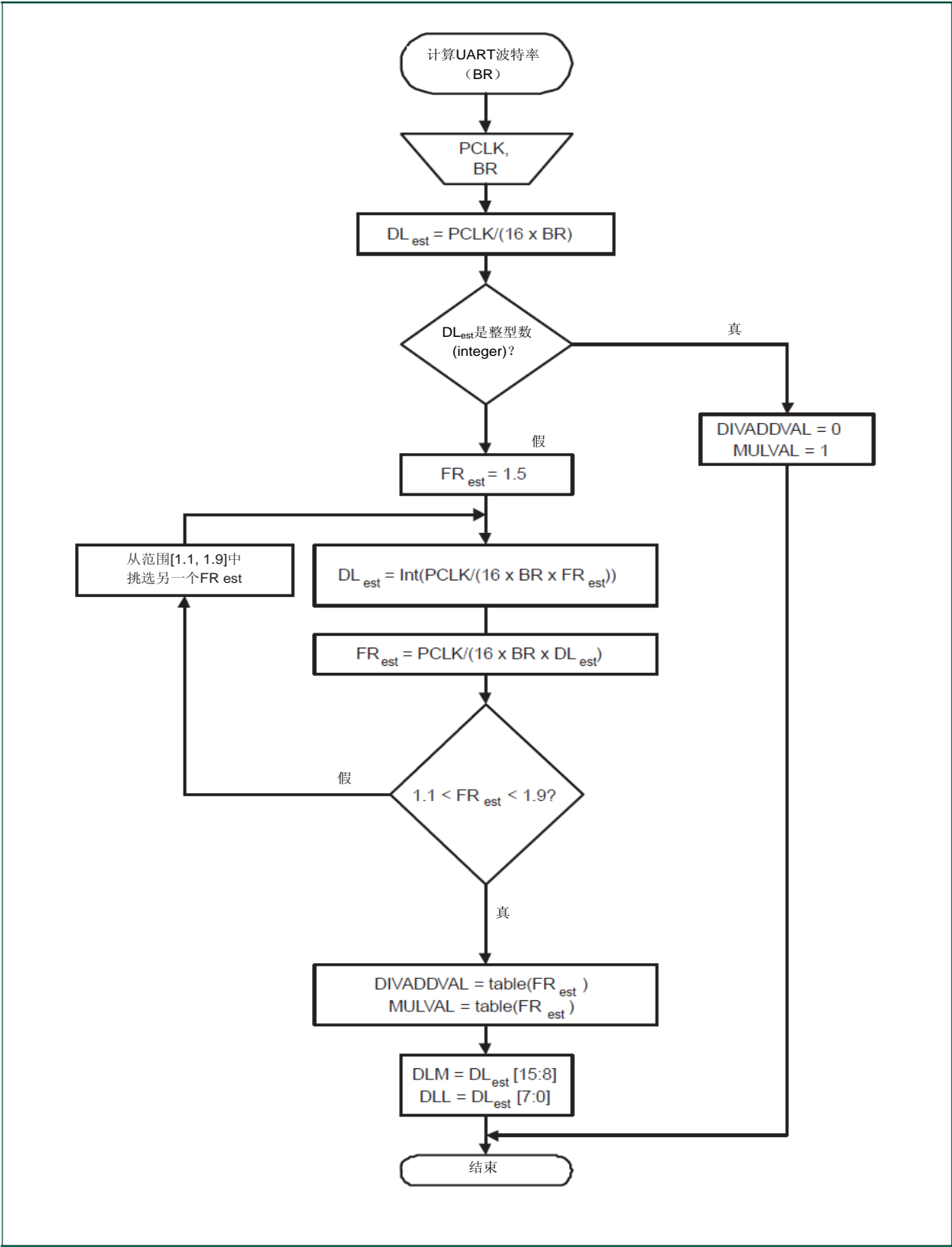


表404. 分数分频器设置查找表

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

18.6.11.1.1 示例 1: PCLK=14.7456 MHz，BR=9600

根据提供的算法， $DL_{est}=PCLK/(16 \times BR)=14.7456\text{ MHz}/(16 \times 9600)=96$ 。因为该算式中的 DL_{est} 是一个整型数，所以 $DIVADDVAL=0$ ， $MULVAL=1$ ， $DLM=0$ 且 $DLL=96$ 。

18.6.11.1.2 示例 2: PCLK=12 MHz，BR=115200

根据提供的算法， $DL_{est}=PCLK/(16 \times BR)=12\text{ MHz}/(16 \times 115200)=6.51$ 。该算式中的 DL_{est} 不是整型数，因此下一步要估算 FR 参数。使用 $FR_{est}=1.5$ 进行首次估算，计算新的 $DL_{est}=4$ ，然后重新计算 FR_{est} 得到 $FR_{est}=1.628$ 。因为 $FR_{est}=1.628$ 是在 1.1 到 1.9 的指定范围之内，因此 $DIVADDVAL$ 和 $MULVAL$ 的值可通过附带的查找表获得。

在查找表 404 中，最接近 $FR_{est}=1.628$ 的值是 $FR=1.625$ 。也就是说， $DIVADDVAL=5$ ，而 $MULVAL=8$ 。

基于这些查找结果，建议将 UART 设置为： $DLM=0$ ， $DLL=4$ ， $DIVADDVAL=5$ 和 $MULVAL=8$ 。根据公式 4，UART 的波特率为 115384。该速率与原来指定的 115200 之间存在 0.16% 的相对误差。

18.6.12 UARTn 发送使能寄存器 (U0TER—0x4000 C030, U2TER—0x4009 8030, U3TER—0x4009 C030)

UnTER 寄存器可以实现软件流控制。当 TXEn=1 时，只要数据可用，UARTn 发送器就会

一直发送数据。一旦 TXEn 变为 0，UARTn 就会停止发送。

[表 405](#) 描述了如何利用 TXEn 位来实现软件流控制。

表405. UARTn 发送使能寄存器位描述（U0TER—0x4000 C030, U2TER—0x4009 8030, U3TER—0x4009 C030）

位	符号	描述	复位值
6:0	-	保留。读取值未定义，只写入 0。	无
7	TXEN	该位为 1 时（复位后），一旦先前的数据都被发送出去后，写入 THR 的数据就会在 TXD 管脚上输出。如果在发送某字符时该位被清零，那么将该字符发送完毕后就不再发送数据，直到该位再次被置位。也就是说，该位为 0 时会阻止字符从 THR 或 TX FIFO 传输到发送移位寄存器。当接收到 XOFF 字符（DC3）时，软件通过执行软件握手可以将该位清零。当接收到 XON 字符（DC1）时，软件又能将该位重新置位。	1

18.6.13 UARTn RS485 控制寄存器（U0RS485CTRL—0x4000 C04C，U2RS485CTRL—0x4009 804C，U3RS485CTRL—0x4009 C04C）

UnRS485CTRL 寄存器控制 RS-485/EIA-485 模式的配置。

表406. UARTn RS485 控制寄存器位描述（U0RS485CTRL—0x4000 C04C, U2RS485CTRL—0x4009 804C, U3RS485CTRL—0x4009 C04C）

位	符号	值	描述	复位值
0	NMMEN	0	禁止 RS-485/EIA-485 普通多点模式（NMM）。	0
		1	使能 RS-485/EIA-485 普通多点模式。在该模式下，当接收字节的校验位为 1 时，对地址进行检测，从而产生一个接收数据中断。参见 18.6.16 节。	
1	RXDIS	0	使能接收器。	0
		1	禁能接收器。	
2	AADEN	0	禁止自动地址检测（AAD）。	0
		1	使能自动地址检测。	
3	-		保留。读取值未定义，只写入 0。	无
4	DCTRL	0	禁能自动方向控制。	0
		1	使能自动方向控制。	
5	OINV		该位保留了 Un_OE 管脚上方向控制信号的极性。	0
		0	当发送器有数据要发送时，方向控制管脚会被驱动为逻辑“0”。在最后一个数据位被发送出去后，该位就会被驱动为逻辑“1”。	
		1	当发送器有数据要发送时，方向控制管脚会被驱动为逻辑“1”。在最后一个数据位被发送出去后，该位就会被驱动为逻辑“0”。	
31:6	-		保留。读取值未定义，只写入 0。	无

18.6.14 UARTn RS-485 地址匹配寄存器（U0RS485ADRMATCH—0x4000 C050，U2RS485ADRMATCH—0x4009 8050，U3RS485ADRMATCH—0x4009 C050）

UnRS485ADRMATCH 寄存器包含 RS-485/EIA-485 模式的地址匹配值。

表407. UARTn RS-485 地址匹配寄存器位描述(U0RS485ADRMATCH—0x4000 C050, U2RS485ADRMATCH—0x4009 8050, U3RS485ADRMATCH—0x4009 C050)

位	符号	描述	复位值
7:0	ADRMATCH	包含了地址匹配值。	0
31:8	-	保留。读取值未定义，只写入 0。	无

18.6.15 UARTn RS-485 延时值寄存器（U0RS485DLY—0x4000 C054，U2RS485DLY—0x4009 8054, U3RS485DLY—0x4009 C054）

对于最后一个停止位离开 TXFIFO 到使 Un_OE 信号无效之间的延时，用户在 8 位的 RS485DLY 寄存器内进行编程。该延迟时间是以波特率时钟周期为单位的。可以设定任何从 0 到 255 位时间的延时。

表408. UARTn RS-485 延时值寄存器位描述（U0RS485DLY—0x4000 0054U2RS485DLY—0x4009 8054, U3RS485DLY—0x4009 C054）

位	符号	描述	复位值
7:0	DLY	包含了方向控制（UnOE）延时值。该寄存器与一个 8 位计数器一起工作。	0
31:8	-	保留。读取值未定义，只写入 0。	无

18.6.16 RS-485/EIA-485 操作模式

RS-485/EIA-485 特性允许将 UART 配置为可寻址从机。可寻址从机是同一主机控制的多个从机之一。

UART 主机发送器通过将校验位设置为“1”来标识地址字符。对于数据字符，校验位会设置为“0”。

每一个 UART 从机接收器都会被分配给一个唯一的地址。从机可设定为手动或自动丢弃那些地址与本机地址不相符的数据。

RS-485/EIA-485 普通多点模式（NMM）

该模式是通过置位 RS485CTRL 的位 0 来启动的。在该模式下，校验位用于另一个用途，即区分所接收数据中的地址和数据。

如果接收器被禁能（RS485CTRL 位 1=“1”），所有接收到的字节都会被忽略且不会存储在 RXFIFO 中。当检测到一个地址字节（校验位=“1”）时，接收到的数据会被存储在 RXFIFO 中，并生成 Rx 数据就绪中断。此时，处理器可以读出地址字节，并决定是否使能接收器接收后面的数据。

当接收器被使能（RS485CTRL 位 1=“0”）时，所有接收到的字节（无论是数据还是地址）都会被接收并存储在 RXFIFO 中。

RS-485/EIA-485 自动地址检测（AAD）模式

如果同时设置 RS485CTRL 寄存器的位 0（9 位模式使能）和位 2（AAD 模式使能），则 UART 处于自动地址检测模式。

在该模式下，接收器会将接收到的所有地址字节（校验位=“1”）与 RS485ADRMATCH 寄存器中编程的 8 位值进行比较。

如果接收器被禁能（RS485CTRL 位 1=“1”），则任何接收到的字节，无论是数据字节还是地址字节，只要与 RS485ADRMATCH 的值不匹配，都会被丢弃。

当检测到匹配的地址字符时，地址字符和校验位就会被一起存储到 RXFIFO 中，此外，接收器将会自动使能（RS485CTRL 位 1 将会由硬件清零）。接收器还会产生 n Rx 数据就绪中断。

当接收器被使能（RS485CTRL 位 1=“0”）时，所有接收到的字节都会被接收并存储在 RXFIFO 中，直到接收到的地址字节与 RS485ADRMATCH 的值不匹配为止。出现不匹配的地址字节时，接收器会通过硬件自动禁能（RS485CTRL 位 1 将会置位）。所接收到的非匹配地址字符不会存储在 RXFIFO 中。

RS-485/EIA-485 自动方向控制

RS485/EIA-485 模式含有一个选项，允许发送器自动控制 UnOE 管脚的状态作为一个方向控制输出信号。

该特性是通过将 RS485CTRL 的位 4 设置为“1”来使能的。

自动方向控制使能后，当 CPU 写数据到 TXFIFO 时，UnOE 管脚变为有效（驱动为低电平）最后一个数据位一旦被发送出去，该管脚就会被设为无效（驱动为高电平）。参见 RS485CTRL 寄存器中的位 4 和位 5。

自动方向控制使能后，当 CPU 写数据到 TXFIFO 时，所选管脚变为有效（驱动为低电平）最后一个数据位一旦被发送出去，该管脚就会被设为无效（驱动为高电平）。参见 RS485CTRL 寄存器中的位 4 和位 5。

RS485/EIA-485 驱动器延时

驱动器延时是指最后一个停止位移出 TXFIFO 到使 UnOE 信号无效之间的延时。该延时可以在 8 位 RS485DLY 寄存器中进行编程。该延时时间是以波特率时钟周期为单位的。可以设定任何从 0 到 255 位时间的延时。

RS485/EIA-485 输出反相

UnOE 管脚上的方向控制信号的极性可通过对 UnRS485CTRL 寄存器中的位 5 进行编程来实现反相。当该位置位时，方向控制管脚将在发送器有数据要发送时被驱动为逻辑 1。在最后一个数据发送出去后，方向控制管脚会被驱动为逻辑 0。

19.1 如何阅读本章

LPC178x/177x 系列中的大部分设备都包含 5 个 UART。少数设备不包含 UART4。有关详细信息，请参阅 [1.4.1](#) 节和具体的设备数据手册。UART4 与 UART02/3 基本相同，但新增了同步模式和智能卡模式。

19.2 基本配置

UART4 的配置需要使用下列寄存器：

1. 功率：在 PCONP 寄存器（[表 37](#)）中，设置 PCUART4 位。
注：复位时，UART4 被禁用（PCUART4 = 0）。
2. 外设时钟：该 UART 使用公共的 PCLK，它既用于总线接口，也用于大多数 APB 外设的功能部分。参见 [4.6.4](#) 节。
3. 波特率：在寄存器 U4LCR（[表 419](#)）中，将 DLAB 位设置为 1，从而能够对寄存器 DLL（[表 413](#)）和 DLM（[表 414](#)）进行访问，以设置波特率。同时，如果有需要的话，还可以在分数分频器寄存器（[表 425](#)）中设置分数波特率。
4. UART FIFO：在寄存器 U4FCR（[表 418](#)）中使用 FIFO 使能位（位 0）可使能 FIFO。
5. 管脚：通过相关的 IOCON 寄存器（[8.4.1](#) 节）选择 UART 管脚和管脚模式。
注：UART 接收管脚不应使能下拉电阻。
6. 中断：要使能 UART 中断，在寄存器 U4LCR（[表 419](#)）中将位 DLAB 设置为 0。从而能够对 U4IER（[表 415](#)）进行访问。利用相应的中断置位使能寄存器来使能 NVIC 中的中断。
7. DMA：UART4 的发送和接收功能可通过 GPDMA 控制器进行操作（参见[表 664](#)）。

19.3 特性

- 数据大小为 5、6、7 和 8 位。
- 奇偶的生成和校验：奇校验、偶校验、1 校验、0 校验或无校验。
- 一个或两个停止位。
- 16 字节收发 FIFO。
- 内置波特率发生器，包含一个多功能的分数波特率分频器。
- 支持 DMA 发送和接收。
- 自动波特率功能。
- 中断生成和检测。
- 多处理器寻址模式。
- IrDA 模式，支持红外通信。
- 支持软件流控制。
- 支持带输出使能的 RS-485/EIA-485 的 9 位模式。
- 可选的同步发送或接收模式。
- 可选的符合 ISO 7816-3 标准的智能卡接口。

19.4 结构

UART4 的结构如下面的框图所示。

APB 接口提供 CPU 或主机与 UART 之间的通信连接。

UART4 接收器模块 (U4RX) 监控串行输入线 RXDn 的有效输入。UART4 RX 移位寄存器 (U4RSR) 通过 RXD4 接收有效字符。当 U4RSR 汇编完一个有效字符时，它将该字符传送到 UART4 RX 缓冲寄存器 FIFO 中，等待 CPU 或主机通过通用主机接口进行访问。

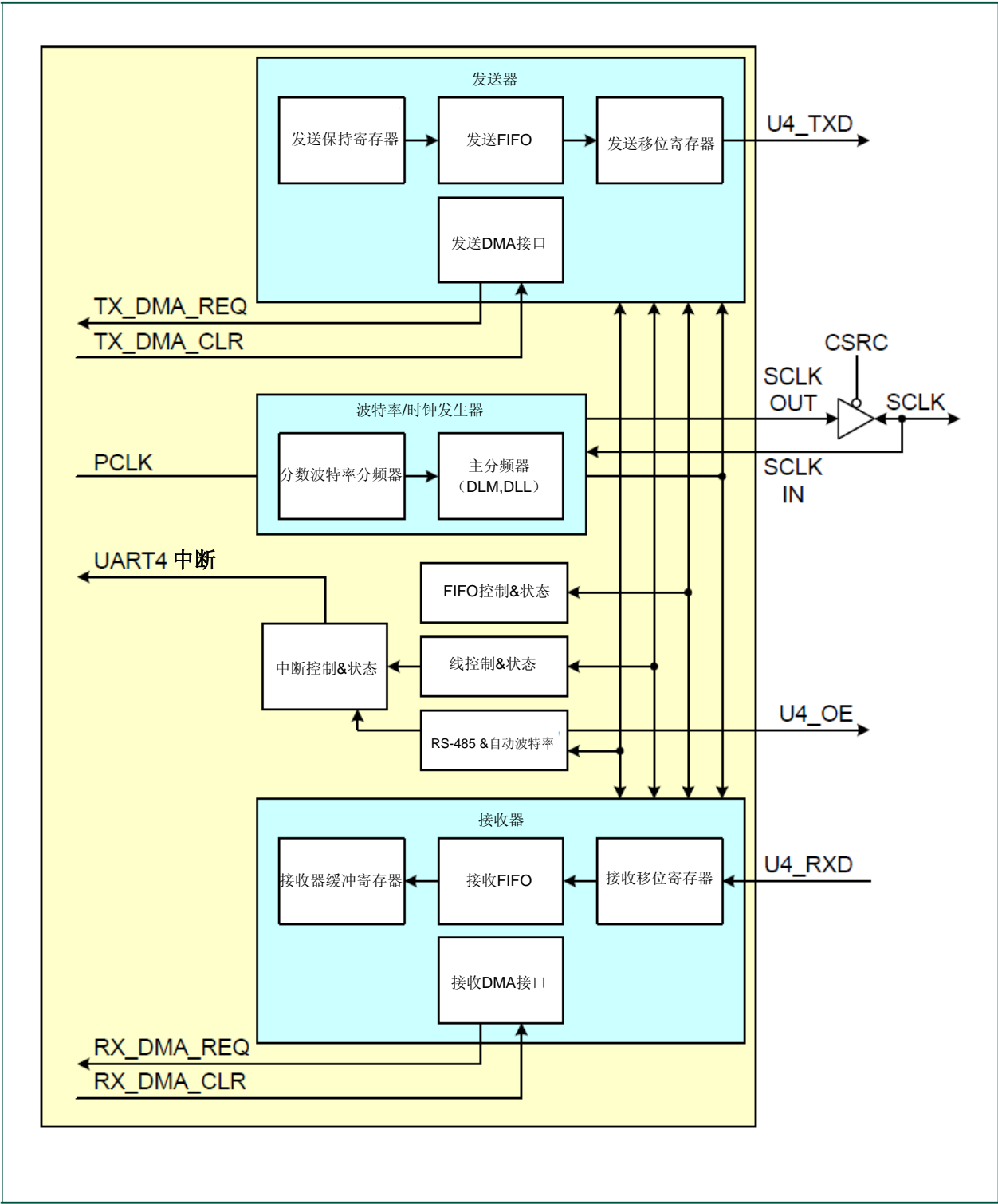
UART4 发送器模块 U4TX 接收 CPU 或主机写入的数据，并且将数据缓存到 UART4 TX 保持寄存器 FIFO (U4THR) 中。UART4 TX 移位寄存器 (U4TSR) 读取 U4THR 中存储的数据，并对数据进行汇编，通过串行输出管脚 TXDn 发送出去。

UART4 波特率发生器模块 U4BRG 产生 UART4 TX 模块所使用的时序使能信号。U4BRG 时钟输入源为 APB 时钟 (PCLK)。主时钟由 U4DLL 和 U4DLM 寄存器中指定的除数相除。得到的时钟是 16x 过采样时钟。

中断接口包括 U4IER 和 U4IIR 寄存器。中断接口接收若干个 U4TX 和 U4RX 模块发出的单时钟宽度使能信号。

U4TX 和 U4RX 发送的状态信息会存储在 U4LSR 中。U4TX 和 U4RX 的控制信息存储在 U4LCR 中。

图74. UART4 框图



19.5 管脚描述

表409. UART4 管脚描述

管脚	类型	描述
U4_RXD	输入	串行输入管脚。串行接收数据。
U4_TXD	输出	串行输出管脚。串行发送数据（智能卡模式下的输入/输出）。
U4_OE	输出	输出使能。RS-485/EIA-485 输出使能。
U4_SCLK	输入/输出	串行时钟。同步模式及智能卡模式下的时钟输入或输出。

19.6 寄存器描述

UART4 包含的寄存器如[表 410](#)中所示。除数锁存器访问位（DLAB）在 U4LCR7 中，能够使能对除数锁存器的访问。

表410. UART4 寄存器映射

通用名称	描述	访问	复位值 ^[1]	寄存器地址	表
U4RBR (DLAB =0)	接收器缓冲寄存器。内含下一个要读取的已接收字符。	RO	无	0x400A 4000	表 411
U4THR (DLAB =0)	发送保持寄存器。在此写入下一个要发送的字符。	WO	无	0x400A 4000	表 412
U4DLL (DLAB =1)	除数锁存器 LSB。波特率除数值的最低有效字节。分数波特率分频器是使用整个除数来产生波特率的。	R/W	0x01	0x400A 4000	表 413
U4DLM (DLAB =1)	除数锁存器 MSB。波特率除数值的最高有效字节。分数波特率分频器是使用整个除数来产生波特率的。	R/W	0	0x400A 4004	表 414
U4IER (DLAB =0)	中断使能寄存器。包含 7 个独立的中断使能位对应 7 个潜在的 UART 中断。	R/W	0	0x400A 4004	表 415
U4IIR	中断 ID 寄存器。识别处于挂起状态的中断。	RO	0x01	0x400A 4008	表 416
U4FCR	FIFO 控制寄存器。控制 UART FIFO 的使用和模式。	WO	0	0x400A 4008	表 418
U4LCR	线控制寄存器。包含帧格式控制和间隔产生控制。	R/W	0	0x400A 400C	表 419
U4LSR	线状态寄存器。包含发送和接收的状态标志（包括线错误）。	RO	0x60	0x400A 4014	表 420
U4SCR	高速缓存寄存器。8 位的临时存储空间，供软件使用。	R/W	0	0x400A 401C	表 421
U4ACR	自动波特率控制寄存器。包含了自动波特率的特性控制。	R/W	0	0x400A 4020	表 422
U4ICR	IrDA 控制寄存器。使能和配置 IrDA 模式。	R/W	0	0x400A 4024	表 423
U4FDR	分数分频器寄存器。为波特率分频器产生时钟输入。	R/W	0x10	0x400A 4028	表 425
U4OSR	过采样寄存器。控制各个位时间（bit time）的过采样程度。	R/W	0xF0	0x400A 402C	表 427
U4SCICTRL	智能卡接口控制寄存器。使能和配置智能卡接口特性。	R/W	0	0x400A 4048	表 428
U4RS485CTRL	RS-485/EIA-485 控制。包含了 RS-485/EIA-485 模式多方面的配置控制。	R/W	0	0x400A 404C	表 429
U4ADRMATCH	RS-485/EIA-485 地址匹配。包含了 RS-485/EIA-485 模式所使用的地址匹配值。	R/W	0	0x400A 4050	表 430
U4RS485DLY	RS-485/EIA-485 方向控制延迟。	R/W	0	0x400A 4054	表 431
U4SYNCCTRL	同步模式控制寄存器。	R/W	0	0x400A 4058	表 432
U4TER	发送使能寄存器。关闭 UART 发送器，使用软件流控制。	R/W	0x01	0x400A 405C	表 433

[1] 复位值仅反映使用位中保存的数据，不包括保留位的内容。

19.6.1 UART4 接收器缓冲寄存器（U4RBR—0x400A C000，DLAB = 0）

U4RBR 是 UART4 RX FIFO 的最高字节。它包含了最早接收到的字符，并且可通过总线接口进行读取。LSB（位 0）代表“最早”接收到的数据位。如果接收到的字符少于 8 位，则未使用的 MSB 使用 0 填充。

如果要访问 U4RBR，LCR 中的除数锁存器访问位（DLAB）必须为 0。U4RBR 始终为只读寄存器。

由于 PE、FE 和 BI 位与 RBR FIFO 顶部的字节（即下次读取 RBR 时获取的字节）对应，因此，要正确地成对读出有效的已接收字节及其状态位，应先读取 U0LSR 寄存器的内容，然后读取 U4RBR 中的字节。

表411. UART4 接收器缓冲寄存器位描述（U4RBR—0x400A 4000，DLAB = 0）

位	符号	描述	复位值
7:0	RBR	UART4 接收器缓冲寄存器包含了 UART4 Rx FIFO 当中最早接收到的字节。	未定义
31:8	-	保留，从保留位读出的值未定义。	无

19.6.2 UART4 发送保持寄存器（U4THR—0x400A C000，DLAB = 0）

U4THR 是 UART4 TX FIFO 的最高字节。它是 TX FIFO 中的最新字符，可以通过总线接口写入。LSB 代表要发送的第一个位。

如果要访问 U4THR，U4LCR 中的除数锁存器访问位（DLAB）必须为 0。U4THR 始终是只写寄存器。

表412. UART4 发送保持寄存器位描述（U4THR—0x400A 4000，DLAB = 0）

位	符号	描述	复位值
7:0	THR	写 UART4 发送保持寄存器会使数据保存到 UART4 发送 FIFO 中。 当字节达到 FIFO 的底部并且发送器就绪时，字节就会被发送。	无
31:8	-	保留。读取值未定义，只写入 0。	无

19.6.3 UART4 除数锁存器 LSB 寄存器（U4DLL—0x400A 4000，DLAB = 1）和
UART4 除数锁存器 MSB 寄存器（U4DLM—0x400A C004，DLAB = 1）

UART4 除数锁存器是 UART4 波特率发生器的一部分，它与分数分频器一同使用，保持产生波特率时钟的 APB 时钟(PCLK)的分频值，波特率时钟必须是所需波特率的 16x 倍。U4DLL 和 U4DLM 寄存器一起构成一个 16 位除数，其中 U4DLL 包含除数的低 8 位，而 U4DLM 包含除数的高 8 位。分频值 0x0000 会被作为 0x0001 处理，因为除数不能为 0。如果要访问 UART4 除数锁存器，U4LCR 中的除数锁存器访问位（DLAB）必须为 1。有关如何为 U4DLL 和 U4DLM 选择正确值的详细信息，参见 [19.6.12](#) 节。

表413. UART1 除数锁存器 LSB 寄存器位描述（U4DLL—0x400A 4000，DLAB = 1）

位	符号	描述	复位值
7:0	DLLSB	UART4 除数锁存器 LSB 寄存器与 U4DLM 寄存器一起决定 UART4 的波特率。	0x01
31:8	-	保留。读取值未定义，只写入 0。	无

表414. UART4 除数锁存器 MSB 寄存器位描述（U4DLL—0x400A 4004，DLAB = 1）

位	符号	描述	复位值
7:0	DLMSB	UART4 除数锁存器 MSB 寄存器与 U0DLL 寄存器一起决定 UART4 的波特率。	0
31:8	-	保留。读取值未定义，只写入 0。	无

19.6.4 UART4 中断使能寄存器（U4IER—0x400A C004，DLAB = 0）

U4IER 用于使能三个 UART4 中断源。

表415. UART4 中断使能寄存器位描述（U4IER—0x400A 4004，DLAB = 0）

位	符号	值	描述	复位值
0	RBR Interrupt Enable		使能 UART4 的接收数据可用中断。它还控制着字符接收超时中断。	0
		0	禁止 RDA 中断。	
		1	使能 RDA 中断。	
1	THRE Interrupt Enable		使能 UART4 的 THRE 中断。该中断的状态可从 U4LSR[5]中读出。	0
		0	禁止 THRE 中断。	
		1	使能 THRE 中断。	
2	RX Line Status Interrupt Enable		使能 UART4 的 RX 线状态中断。该中断的状态可从 U4LSR[4:1]中读出。	0
		0	禁止 RX 线状态中断。	
		1	使能 RX 线状态中断。	
7:3	-		保留。读取值未定义，只写入 0。	无
8	ABEOIntEn		使能 auto-baud 结束中断。	0
		0	禁止 auto-baud 结束中断。	
		1	使能 auto-baud 结束中断。	
9	ABTOIntEn		使能 auto-baud 超时中断。	0
		0	禁止 auto-baud 超时中断。	
		1	使能 auto-baud 超时中断。	
31:10	-		保留。读取值未定义，只写入 0。	无

19.6.5 UART4 中断标识寄存器（U4IIR—0x400A C008）

U4IIR 提供一个状态代码，用于指示一个挂起中断的优先级和中断源。在访问 U4IIR 的过程中，中断被冻结。如果在访问 U4IIR 的过程中产生了中断，该中断会被记录以用于下次 U4IIR 访问。

表416. UART4 中断标识寄存器位描述（U4IIR—0x400A 4008）

位	符号	值	描述	复位值
0	IntStatus		中断状态。注：U4IIR[0]为低电平有效。挂起的中断可通过 U4IIR[3:1]确定。	1
		0	至少有一个中断被挂起。	
		1	没有挂起的中断。	
3:1	IntId		中断标识。U4IER[3:1]指示对应 UART4 Rx 或 TX FIFO 的中断。下面未列出的 U4IER[3:1]的其他组合都为保留值(000,100,101,111)。	0
		011	1 – 接收线状态（RLS）。	
		010	2a – 接收数据可用（RDA）。	
		110	2b – 字符超时指示器（CTI）。	
		001	3 – THRE 中断	
5:4	-		保留。读取值未定义，只写入 0。	无
7:6	FIFO Enable		这些位等效于 U4FCR[0]。	0
8	ABEOInt		Auto-baud 结束中断。若 auto-baud 已成功结束且中断被使能，则为真。	0
9	ABTOInt		Auto-baud 超时中断。若 auto-baud 已超时且中断被使能，则为真。	0
31:10	-		保留。读取值未定义，只写入 0。	无

位 U4IIR[9:8]由自动波特率功能设置，用于发布超时信号或自动波特率结束条件信号。而自动波特率中断条件的清除是通过设置自动波特率控制寄存器中相应清除（Clear）位来实现。

如果 IntStatus 位为 1，表示没有中断被挂起，此时 IntId 位为 0。如果 IntStatus 为 0，表示有一个非自动波特率中断被挂起，此时 IntId 位会标识中断的类型以及表 417 中所示的处理。知道了 U4IIR[3:0]的状态，中断处理程序就能确定中断原因以及如何清除激活的中断。在退出中断服务程序之前，必须读取 U4IIR 来清除中断。

UART4 RLS 中断（U4IIR[3:1] = 011）是最高优先级的中断，只要 UART4 Rx 输入上产生以下四个错误条件中的任意一个，该位就会被置位：溢出错误（OE）、校验错误（PE）、帧错误（FE）以及间隔中断（BI）。通过 U0LSR[4:1]可查看设置中断的 UART4 Rx 错误条件。读取 U4LSR 时，中断会被清除。

UART4 RDA 中断（U4IIR[3:1] = 010）与 CTI 中断（U4IIR[3:1] = 110）共用第二优先级。当 UART4 Rx FIFO 达到 U4FCR[7:6]中定义的触发值时，RDA 被激活，而当 UART4 Rx FIFO 深度低于触发值时，RDA 复位。当 RDA 中断有效时，CPU 可读出由触发值定义的一个数据块。

CTI 中断（U4IIR[3:1] = 110）是一个第二优先级中断，当 UART4 Rx FIFO 内含有至少一个字符并且在 3.5 到 4.5 个字符时间内没有发生 UART4 Rx FIFO 动作，则该中断被设置。任何 UART4 Rx FIFO 动作（UART4 RSR 的读取或写入）都会清除该中断。当接收到的消息不是触发值的倍数时，该中断会清空 UART4 RBR。例如，如果外设要发送一条长度为 105 个字符的消息，而触发值为 10 个字符，那么 CPU 接收到 10 个 RDA 中断的结果是 100 个字符的传输，而接收 1 至 5 个 CTI 中断（取决于服务程序）的结果是剩下 5 个字符的传输。

表417. UART4 中断处理

U1IIR[3:0]值 ^[1]	优先级	中断类型	中断源	中断复位
0001	-	无	无	-
0110	最高	RX 线状态/错误	OE ^[2] 、PE ^[2] 、FE ^[2] 或 BI ^[2]	U4LSR 读操作 ^[2]
0100	第二	RX 数据可用	Rx 数据可用或达到 FIFO 的触发值（U4FCR0=1）	U4RBR 读操作 ^[3] 或 UART4 FIFO 低于触发值
1100	第二	字符超时指示	Rx FIFO 中至少有一个字符，并且在一段时间内没有字符输入或移出，该时间的长短取决于 FIFO 中的字符数以及在 3.5 到 4.5 字符的时间内的触发值。 实际时间为：[（字长度）× 7- 2] × 8 + [（触发值- 字符数）× 8 + 1] RCLKs	U4RBR 读操作 ^[3]
0010	第三	THRE	THRE ^[2]	U4IIR 读操作 ^[4] （如果 U4IIR 是中断源）或 THR 写操作 ^[4]

- [1] “0000”、“0011”、“0101”、“0111”、“1000”、“1001”、“1010”、“1011”、“1101”、“1110”、“1111” 均为保留值。
- [2] 详细信息，参见 19.6.8 节。
- [3] 详细信息， 参见 19.6.1 节。
- [4] 详细信息，参见 19.6.5 节和 19.6.2 节。

UART4 THRE 中断（U4IIR[3:1] = 001）是第三优先级中断，在 UART4 THR FIFO 为空并满足特定的初始化条件时，该中断会被激活。这些初始化条件是为了让 UART4 THR FIFO 有机会填入数据，以免在系统启动时产生许多 THRE 中断而规定的。当 THRE = 1 时，且在上次 THRE = 1 事件后，U4THR 中没有同时出现至少两个字符的情况下，这些初始化条件就会实现一个字符延时减去停止位。当没有解码和服务 THRE 中断时，该延时为 CPU 提供了写数据到 U4THR 的时间。如果 UART4 THR FIFO 中同时保持两个或更多字符，而当前的 U4THR 为空时，THRE 中断会立即被设置。当发生 U4THR 写操作或 U4IIR 读操作，并且 THRE 是最高优先级的中断（U4IIR[3:1] = 001）时，THRE 中断复位。

19.6.6 UART4 FIFO 控制寄存器（U4FCR—0x400A C008）

只写的 U4FCR 控制 UART4 RX 和 TX FIFO 的操作。

表418. UART4 FIFO 控制寄存器位描述（U4FCR—0x400A 4008）

位	符号	值	描述	复位值
0	FIFO Enable	0	UART4 FIFO 被禁止。禁止在应用中使用。	0
		1	高电平有效，使能对 UART4 Rx 和 TX FIFOs 以及 U4FCR[7:1]的访问。该位必须置位以实现正确的 UART 操作。该位的任何变化都将使 UART FIFO 清空。	
1	RX FIFO Reset	0	对两个 UART4 FIFO 均无影响。	0
		1	写逻辑 1 到 U4FCR[1]将会清零 UART4 Rx FIFO 中所有字节，并复位指针逻辑。该位会自动清零。	

位	符号	值	描述	复位值
2	TX FIFO Reset	0	对两个 UART4 FIFO 均无影响。	0
		1	写逻辑 1 到 U4FCR[2]将会清零 UART4 TX FIFO 中所有字节，并复位指针逻辑。该位会自动清零。	
3	DMA Mode Select		当 FIFO 使能位（该寄存器的位 0）被置位时，该位选择 DMA 模式。参见 19.6.6.1 节。	0
5:4	-		保留。读取值未定义，只写入 0。	无
7:6	RX Trigger Level		这两个位决定了接收 UART4 FIFO 在激活中断或 DMA 请求前必须写入的字符数量。	0
		00	触发点 0（1 字节或 0x01）	
		01	触发点 1（4 字节或 0x04）	
		10	触发点 2（8 字节或 0x08）	
		11	触发点 3（14 字节或 0x0E）	
31:8	-		保留。读取值未定义，只写入 0。	无

19.6.6.1 DMA 操作

用户可以选择用 DMA 进行 UART 的发送和/或接收。DMA 模式由 FCR 寄存器中的 DMA 模式选择位决定。只有在 FCR 寄存器中的 FIFO 使能位使能了 FIFO 时，该位才有用。

UART 接收 DMA

在 DMA 模式下，当接收 FIFO 水平大于或等于触发值时，或者在发生一次字符超时情况下，接收 DMA 请求才会生效。请参见上文中对 RX 触发值的描述。接收 DMA 请求由 DMA 控制器清除。

UART 发送 DMA

在 DMA 模式下，当发送 FIFO 变为未滿时，发送 DMA 请求就会有效。发射器 DMA 请求由 DMA 控制器清除。

19.6.7 UART4 线控制寄存器（U4LCR—0x400A C00C）

U4LCR 决定待发送或接收的数据字符的格式。

表419. UART4 线控制寄存器位描述（U4LCR—0x400A 400C）

位	符号	值	描述	复位值
1:0	Word Length Select	00	5 位字符长度 1:0	0
		01	6 位字符长度	
		10	7 位字符长度	
		11	8 位字符长度	
2	Stop Bit Select	0	1 个停止位。	0
		1	2 个停止位（当 U4LCR[1:0]=00 时，为 1.5 个）。	
3	Parity Enable	0	禁止奇偶产生和校验。	0
		1	使能奇偶产生和校验。	

位	符号	值	描述	复位值
5:4	Parity Select	00	奇校验。发送字符和添加的校验位所包含的“1”的数量是一个奇数。	0
		01	偶校验。发送字符和添加的校验位所包含的“1”的数量是一个偶数。	
		10	强制为“1”stick 校验。	
		11	强制为“0”stick 校验。	
6	Break Control	0	禁止间隔发送。	0
		1	使能间隔发送。当 U4LCR[6]为高电平（有效）时，输出管脚 UART4 TXD 强制为逻辑 0。	
7	Divisor Latch Access Bit (DLAB)	0	禁止访问除数锁存器。	0
		1	使能访问除数锁存器。	
31:8	-		保留。读取值未定义，只写入 0。	无

19.6.8 UART4 线状态寄存器（U4LSR—0x400A C014）

U4LSR 是一个只读寄存器，提供 UART4 TX 和 RX 模块的状态信息。

表420. UART4 线状态寄存器位描述（U4LSR—0x400A 4014）

位	符号	值	描述	复位值
0	Receiver Data Ready (RDR)		当 U4RBR 包含一个未读字符时，U4LSR[0]就会被置位；当 UART4 RBR FIFO 为空 0 时，U4LSR[0]就会被清零。	0
		0	UART4 接收 FIFO 为空。	
		1	UART4 接收 FIFO 不为空。	
1	Overrun Error (OE)		溢出错误条件在错误发生后立即设置。读 U4LSR 会清零 U4LSR[1]。当 UART4 RSR 0 已有新的字符汇集，而 UART4 RBR FIFO 已满时，U4LSR[1]会被置位。此时，UART4 RBR FIFO 将不会被覆盖，UART4 RSR 内的字符将会丢失。	0
		0	溢出错误状态未激活。	
		1	溢出错误状态激活。	
2	Parity Error (PE)		当接收字符的校验位处于错误状态时，校验错误就会产生。读 U4LSR 会清零 U4LSR[2]。校验错误检测时间取决于 U4FCR[0]。 注：校验错误与 UART4 RBR FIFO 顶部的字符相关。	0
		0	校验错误状态未激活。	
		1	校验错误状态激活。	
3	Framing Error (FE)		当接收字符的停止位为逻辑 0 时，就会发生帧错误。读 U4LSR 会清零 U4LSR[3]。0 帧错误检测时间取决于 U4FCR[0]。当检测到有帧错误时，Rx 会尝试与数据重新同步，并假设错误的停止位实际上是一个超前的起始位。但即使没有出现帧错误，它也无法假设下一个接收到的字符是正确的。 注：帧错误与 UART4 RBR FIFO 顶部的字符相关。	0
		0	帧错误状态未激活。	
		1	帧错误状态激活。	
4	Break Interrupt (BI)		在发送整个字符（起始位、数据、校验位以及停止位）过程中，RXDn 如果保持在 0 间隔状态（全“0”），则产生间隔中断。一旦检测到间隔条件，接收器立即进入空闲状态，直到 RXDn 进入标记状态（全“1”）。读 U4LSR 会清零该状态位。间隔检测时间取决于 U4FCR[0]。 注：间隔中断与 UART4 RBR FIFO 顶部的字符相关。	0
		0	间隔中断状态未激活。	
		1	间隔中断状态激活。	
5	Transmitter		当检测到 UART4 THR 已空时，THRE 就会立即被设置。写 U4THR 会清零 THRE。1	1

位	符号	值	描述	复位值
	Holding Register Empty (THRE))	0	U4THR 包含有效字符。	
		1	U4THR 为空。	
6	Transmitter Empty (TEMT)		当 U4THR 和 U4TSR 同时为空时，TEMT 就会被设置；而当 U4TSR 或 U4THR 任一包含有效数据时，TEMT 就会被清零。	1
		0	U4THR 和/或 U4TSR 包含有效数据。	
		1	U4THR 和 U4TSR 为空。	
7	Error in RX FIFO (RXFE)		当一个带有 Rx 错误（如：帧错误、校验错误或间隔中断）的字符载入到 U4RBR 中 0 时，U4LSR[7]就会被设置。当 U4LSR 寄存器被读取并且 UART4 FIFO 中不再有错误时，该位就会清零。	
		0	U4RBR 中没有 UART4 RX 错误或 U4FCR[0]=0。	
		1	UART4 RBR 中包含至少一个 UART4 RX 错误。	
31:8	-		保留。从保留位读取的值未定义。	无

19.6.9 UART4 高速缓存寄存器（U4SCR—0x400A C01C）

U4SCR 对 UART4 操作没有影响。用户可以自由地读写该寄存器。不提供中断接口向主机指示 U4SCR 所发生的读或写操作。

表421. UART4 高速缓存寄存器位描述（U4SCR—0x400A 401C）

位	符号	描述	复位值
7:0	Pad	一个可读、可写的字节。	0
31:8	-	保留。读取值未定义，只写入 0。	无

19.6.10 UART4 自动波特率控制寄存器（U4ACR—0x400A C020）

在用户测量波特率的输入时钟/数据速率期间，整个测量过程就是由 UART4 自动波特率控制寄存器（U4ACR）进行控制的。用户可以自由地读写该寄存器。

表422. UART4 自动波特率控制寄存器位描述（U4ACR—0x400A 4020）

位	符号	值	描述	复位值
0	Start		在 auto-baud 功能结束后，该位会自动清零。	0
		0	Auto-baud 功能停止（auto-baud 功能不运行）。	
		1	Auto-baud 功能启动（auto-baud 功能正在运行）。Auto-baud 运行位。在 auto-baud 功能结束后，该位会自动清零。	
1	Mode		Auto-baud 模式选择位。	0
		0	模式 0。	
		1	模式 1。	
2	AutoRestart	0	不重新启动。	0
		1	如果超时则重新启动（计数器会在下一个 UART4 Rx 下降沿重新启动）。	0
7:3	-		保留。读取值未定义，只写入 0。	无
8	ABEOIntClr		Auto-baud 结束中断清零位（仅可写访问）。写 1 将 U4IIR 中相应的中断清除。写 0 无影响。	0
9	ABTOIntClr		Auto-baud 超时中断清零位（仅可写访问）。写 1 将 U4IIR 中相应的中断清除。写 0 无影响。	0
31:10	-		保留。读取值未定义，只写入 0。	无

19.6.10.1 自动波特率

UART4 自动波特率功能可用于测量基于“AT”协议（Hayes 命令）的输入波特率。如果自动波特率功能被使能，那么自动波特率功能将测量接收数据流中的一个位所消耗的时间，并根据这个结果来设置除数锁存器寄存器 U4DLM 和 U4DLL。

注：在自动波特率操作期间不会连接分数波特率分频器，所以在需要自动波特率功能时不应使用分数波特率分频器。

自动波特率功能通过置位 U4ACR 起始位来启动，并通过清零 U4ACR 起始位来停止。自动波特率一旦结束，起始位会被立即清零，如果此时读取该起始位，会返回自动波特率的状态（挂起/完成）。

可通过设置 U4ACR 模式位来使用两种自动波特率测量模式。在模式 0 下，波特率是通过对 UART4 Rx 管脚上两个连续的下降沿进行测量（起始位的下降沿和最低有效位的下降沿）来得到的。在模式 1 下，波特率则是通过测量 UART4 Rx 管脚上的下降沿与后续的上升沿之间的时间（起始位的长度）来得到的。

如果发生超时（速率测量计数器溢出），U4ACR AutoRestart 位可用于自动重启波特率测量。如果该位被置位，速率测量将会在 UART4 Rx 管脚的下一个下降沿重新启动。

自动波特率功能会产生两种中断：

- 如果使能了中断(U4IER ABTOIntEn 置位且自动波特率速率测量计数器溢出),则 U4IIR ABTOInt 中断将被设置。
- 如果使能了中断(U4IER ABEOIntE 置位且自动波特率已成功完成),则 U4IIR ABEOInt 中断将被设置。

必须通过置位相应的 U4ACR ABTOIntClr 和 ABEOIntEn 位，才能清除自动波特率中断。

在执行自动波特率期间，分数波特率发生器通常会被禁用（即 DIVADDVAL = 0）。但是，如果使能了分数波特率发生器（DIVADDVAL > 0），那么它会影响 UART4 Rx 管脚波特率的测量，但 U4FDR 寄存器的值在速率测量后不会被修改。此外，当使用自动波特率时，任何对 U4DLM 和 U4DLL 寄存器的写操作都必须在写 U4ACR 寄存器之前完成。UART4 支持的最小和最大波特率是 PCLK、数据位数、停止位数和校验位数的函数。

(5)

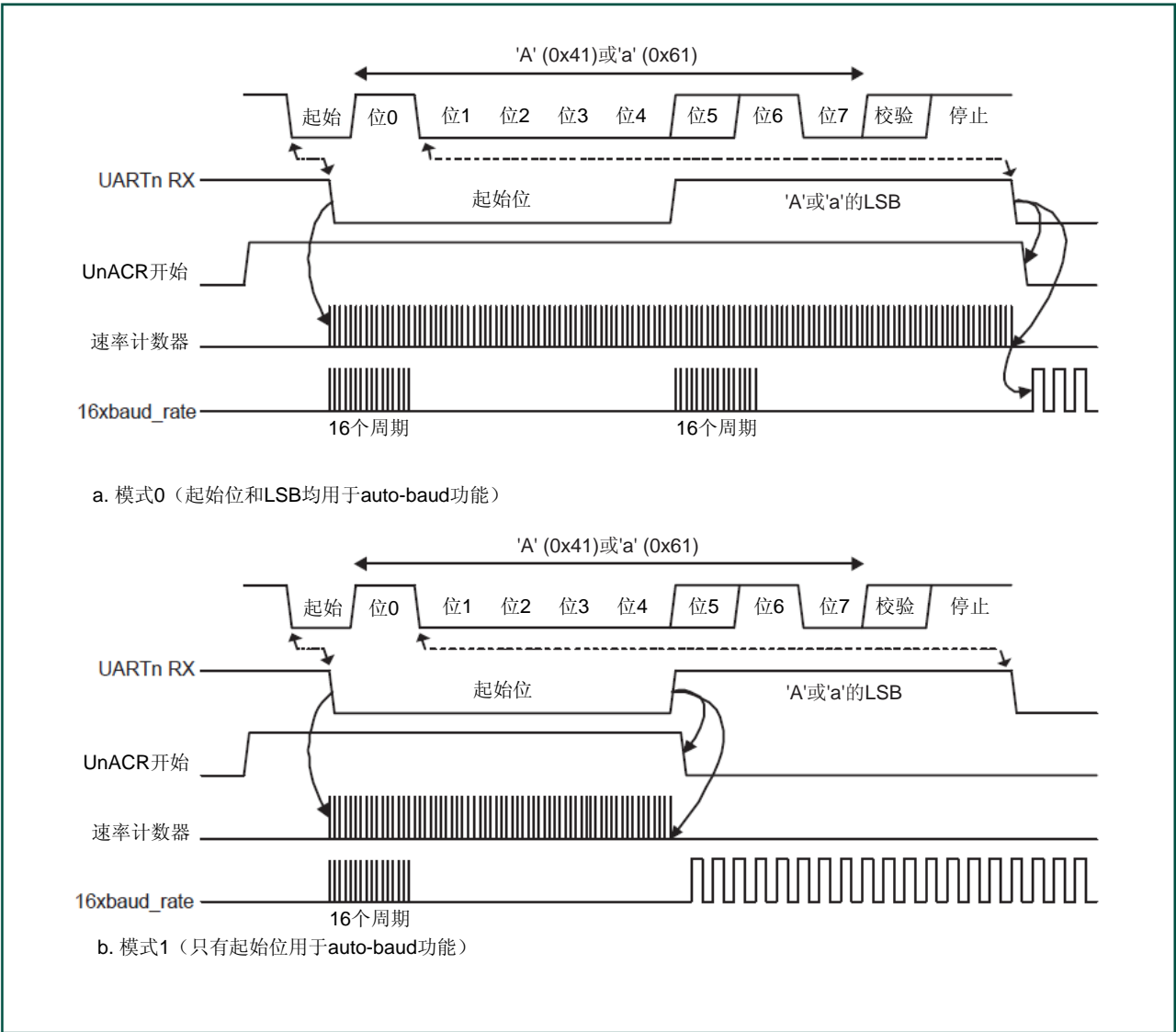
$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART4_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

19.6.10.2 自动波特率模式

当软件等待一个“AT”命令时，它采用期望的字符格式配置 UART4，并设置 U4ACR 起始位。用户不必关心除数锁存器 U4DLM 和 U4DLL 中的初始值。由于字母“A”或“a”的 ASCII 编码（“A” = 0x41，“a” = 0x61）的关系，UART4 Rx 管脚检测到的起始位和期望字符的 LSB 由两个下降沿限定。当 U4ACR 起始位被置位时，自动波特率协议将执行以下步骤：

1. 一旦 U4ACR 起始位被置位，波特率测量计数器复位，UART4 U4RSR 复位。U4RSR 波特率切换为最高速率。
2. UART4 Rx 管脚上的下降沿会触发起始位的开始。速率测量计数器将开始对 PCLK（可选择被分数波特率发生器预分频）进行计数。
3. 在接收起始位的过程中，RSR 波特率输入端产生 16 个脉冲，频率是（经分数波特率预分频的）UART4 输入时钟，这样可保证起始位存储在 U4RSR 中。
4. 在接收起始位的过程中（以及模式 0 下的字符 LSB），速率计数器将继续随着被预分频的 UART4 输入时钟（PCLK）递增。
5. 如果 Mode = 0，则速率计数器将在 UART4 Rx 管脚的下一个下降沿停止。如果 Mode = 1，则速率计数器将在 UART4 Rx 管脚的下一个上升沿停止。
6. 速率计数器的值被载入到 U4DLM/U4DLL 中，并且波特率将会切换到正常操作模式。在设置完 U4DLM/U4DLL 后，如果自动波特率结束中断被使能，U4IIR ABEOInt 将会被置位。接着，U4RSR 将继续接收“A/a”字符剩余的位。

图75. Auto-baud a) 模式 0 和 b) 模式 1 波形图



19.6.11 UART4 IrDA 控制寄存器 (U4ICR—0x400A C024)

IrDA 控制寄存器用于使能和配置每个 UART 上的 IrDA 模式。在发送或接收数据时，不应更改 U4ICR 的值，否则会造成数据丢失或损坏。

表423. UART4 IrDA 控制寄存器位描述 (U4ICR—0x400A 4024)

位	符号	值	描述	复位值
0	IrDAEn	0	UART4 上的 IrDA 模式被禁能，UART4 充当一个标准 UART。	0
		1	UART4 上的 IrDA 模式被使能。	
1	IrDAInv		该位为 1 时，串行输入被反相。这对串行输入无影响。该位为 0 时，串行输入未被反相。	0
2	FixPulseEn		该位为 1 时，使能 IrDA 固定脉冲宽度模式。	0

位	符号	值	描述	复位值
5:3	PulseDiv		当 FixPulseEn = 1 时，配置脉冲。详细信息，参见下文。	0
31:6	-		保留。读取值未定义，只写入 0。	0

当 IrDA 模式下使用固定脉冲宽度模式时（IrDAEn = 1 且 FixPulseEn = 1），U4ICR 中的 PulseDiv 位会被用来选择脉冲宽度。这些位应该被置位，这样得到的脉冲宽度将至少为 1.63 μs。表 424 所示为可能的脉冲宽度。

表424. IrDA 脉冲宽度

FixPulseEn	PulseDiv	IrDA 发送脉冲宽度(μs)
0	x	3 / (16x波特率)
1	0	2xT _{PCLK}
1	1	4xT _{PCLK}
1	2	8xT _{PCLK}
1	3	16xT _{PCLK}
1	4	32xT _{PCLK}
1	5	64xT _{PCLK}
1	6	128xT _{PCLK}
1	7	256xT _{PCLK}

19.6.12UART4 分数分频器寄存器（U4FDR—0x400A C028）

UART4 分数分频器寄存器（U4FDR）控制着用于波特率生成的时钟预分频器，用户可以自由地进行读写操作。该预分频器使用 APB 时钟并根据指定的分数要求产生一个输出时钟。

重要提示：如果分数分频器是有效的（DIVADDVAL > 0）且 DLM = 0，则 DLL 寄存器的值必须大于 2。

表425. UART4 分数分频器寄存器位描述（U4FDR—0x400A 4028）

位	功能	值	描述	复位值
3:0	DIVADDVAL	0	产生波特率的预分频除数值。如果该字段为 0，分数波特率发生器将不会影响 UART4 的波特率。	0
7:4	MULVAL	1	波特率预分频乘数值。不管是否使用分数波特率发生器，为了让 UART4 正常工作，该字段必须大于或等于 1。	1
31:8	-		保留。读取值未定义，只写入 0。	0

该寄存器控制着用于波特率生成的时钟预分频器。该寄存器的复位值会让 UART 的分数功能保持在禁用状态，以确保此 UART 与没有配备该特性的 UART 保持全面的软件和硬件兼容。

UART 波特率可以用下面的公式进行计算：

(6)

$$UART4_{baudrate} = \frac{PCLK}{16 \times (256 \times U4DLM + U4DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

其中，PCLK 为外设时钟，U4DLM 和 U4DLL 为标准 UART 波特率分频器寄存器，而 DIVADDVAL 和 MULVAL 为 UART 分数波特率发生器的指定参数。

MULVAL 和 DIVADDVAL 的值应符合以下条件：

1. $1 \leq \text{MULVAL} \leq 15$
2. $0 \leq \text{DIVADDVAL} \leq 14$
3. $\text{DIVADDVAL} < \text{MULVAL}$

在发送/接收数据的过程中，不应修改 U4FDR 的值，否则可能导致数据丢失或损坏。

如果 U4FDR 寄存器的值不符合上述两个要求，则分数分频器输出为未定义。如果 DIVADDVAL 为 0，则分数分频器会被禁用，并且不会对时钟进行分频。

19.6.12.1 波特率计算

UART4 可以与分数分频器一起工作，也可以不使用分数分频器。在实际应用中，使用几种不同的分数分频器设置很可能会获得目标波特率。以下算法给出了一个得到 DLM、DLL、MULVAL 和 DIVADDVAL 参数集的方法。该参数集允许实际波特率与目标波特率之间存在小于 1.1% 的相对误差。

图76. 设置 UART 分频器的算法

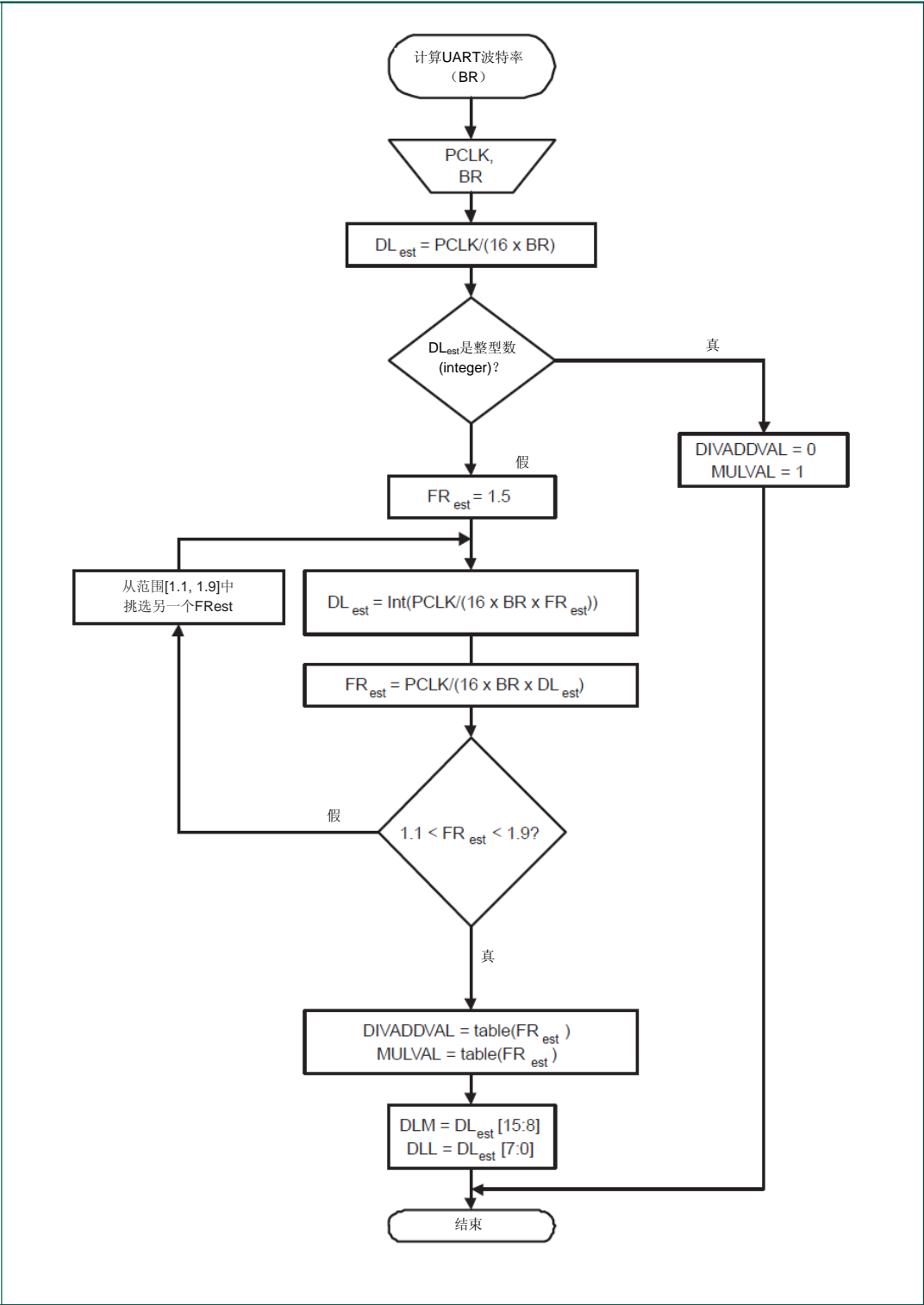


表426. 分数分频器设置查找表

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

19.6.12.1.1 示例 1: PCLK = 14.7456 MHz, BR = 9600

根据提供的算法， $DL_{est} = PCLK / (16 \times BR) = 14.7456 \text{ MHz} / (16 \times 9600) = 96$ 。因为该算式中的 DL_{est} 是一个整型数，所以 $DIVADDVAL = 0$ ， $MULVAL = 1$ ， $DLM = 0$ 且 $DLL = 96$ 。

19.6.12.1.2 示例 2: PCLK = 12 MHz, BR = 115200

根据提供的算法， $DL_{est} = PCLK / (16 \times BR) = 12 \text{ MHz} / (16 \times 115200) = 6.51$ 。该算式中的 DL_{est} 不是整型数，因此下一步要估算 FR 参数。使用 $FR_{est} = 1.5$ 进行首次估算，得到新的 $DL_{est} = 4$ ，然后重新计算 FR_{est} 得到 $FR_{est} = 1.628$ 。因为 $FR_{est} = 1.628$ 是在 1.1 到 1.9 的指定范围之内，因此 $DIVADDVAL$ 和 $MULVAL$ 的值可通过附带的查找表获得。

在表 426 中，最接近 $FR_{est} = 1.628$ 的值是 $FR = 1.625$ 。也就是说， $DIVADDVAL = 5$ ，而 $MULVAL = 8$ 。

基于这些查找结果，建议将 UART 设置为： $DLM = 0$ ， $DLL = 4$ ， $DIVADDVAL = 5$ 和 $MULVAL = 8$ 。根据公式 6，UART 的波特率为 115384。该速率与原来指定的 115200 之间存在 0.16% 的相对误差。

19.6.13 UART4 过采样寄存器（U4OSR—0x400A 402C）

在大多数应用中，UART 在一个额定位时间内要接收数据进行 16 次采样，并发送 16 个输入时钟宽度的位。该寄存器允许软件控制输入时钟与位时钟之间的比率。在智能卡模式下需要使用该功能，而在其他模式下，该功能提供了一种替代分数分频的方法。

表427. UART4 过采样寄存器位描述（U4OSR—0x400A 402C）

位	符号	描述	复位值
0	-	保留。读取值未定义，只写入 0。	无
3:1	OSFrac	过采样比率的小数部分，为一个输入时钟周期的 1/8 (001 =0.125, ..., 111 = 0.875)	0
7:4	OSInt	过采样比率的整数部分，减去 1。复位值等同于正常操作模式下每位时间（bit time）16 个输入时钟。	0xF
14:8	FDInt	在智能卡模式下，这些位作为 OSInt 字段的更为有效的延伸位，实现了 ISO7816-3 所要求的高达 2048 的过采样比率。在智能卡模式下，位 14:4 的初始值应被设为 371，由此得出过采样比率为 372。	0
31:15	-	保留。读取值未定义，只写入 0。	无

例：当波特率为 3.25 Mbps 而时钟频率为 24 MHz UART 时，理想的过采样比率是 24/3.25 或 7.3846。将 OSInt 设置为 0110 是 7 个时钟/位，而将 OSFrac 设置为 011 是 0.375 个时钟/位，最终过采样比率是 7.375。

在智能卡模式下，OSInt 由 FDInt 扩展。最大可以将过采样扩展到 2048，这是支持 ISO 7816-3 的要求。注意，当 D<0 时可以设置大于该值的值，但 UART 不支持该设置。当使能了智能卡模式时，应该将 OSInt 和 FDInt 的初始值编程为“00101110011”（372 减 1）。

19.6.14 UART4 智能卡接口控制寄存器（U4SCICTRL—0x400A4048）

该寄存器允许 UART 用于符合 ISO7816-3 标准的异步智能卡应用中。

表428. UART4 智能卡接口控制寄存器位描述（U4SCICTRL—0x400A 4048）

位	符号	值	描述	复位值
0	SCIEN		智能卡接口使能。	0
		0	智能卡接口禁能。	
		1	异步半双工智能卡接口使能。	
1	NACKDIS		NACK 响应禁能。仅在 T=0 时可用。	0
		0	NACK 响应使能。	
		1	NACK 响应禁止。	
2	PROTSEL		ISO7816-3 标准中定义的协议选择位。	0
		0	T = 0	
		1	T = 1	
7:5	TXRETRY		当协议选择位 T（如上所示）为 0 时，若远程设备信号 NACK，则由字段控制 UART - 所尝试进行的最大重发次数。当 NACK 发生时，这一次数加 1，LSR 中的 Tx 错误位置位，要求产生中断（若使能），并且 UART 会被锁定直至 FIFO 清零。	
15:8	XTRAGUARD		当协议选择位 T（如上所示）为 0 时，该字段显示位时间的数量。根据该位的值，在 UART 发送一个字符后，保护时间不得超过标称的 2 位时间。该字段中的 0xFF 可表明在一个字符和 11 位时间/字符后，仅存在一个单一位。	
31:16	-		保留。读取值未定义，只写入 0。	无

19.6.14.1 智能卡连接

如果按照上文所述设置 U4SCICTRL 寄存器中的位 SCIEN，UART 在 TXD 管脚上提供双向串行数据。当 SCIEN 为 1 时不使用 RXD 管脚。如果在 I/O 配置（I/O Configuration）块中使能了 UART SCLK 功能，则该管脚上会输出一个串行时钟：对智能卡来说，这个时钟的使用是可选的。软件必须使用定时器实现字符和块的等待时间（UART 中不提供触发信号的硬件支持）。可以使用 GPIO 管脚来控制智能卡复位和电源管脚。

19.6.14.2 智能卡设置

在智能卡应用中必须做如下设置：

- 在必要时，如 3.5 节中所述将 UART 复位。
- 对一个 IOCON 寄存器进行编程来使能一种 UART TXD 功能。
- 如果要进行通信的智能卡需要（或者有可能需要）一个时钟，则对一个 IOCON 寄存器进行编程以使能 UART SCLK 功能。UART 将使用它作为一个输出。
- 使能 UART 时钟并设置 UART 计时模式，以获得 3.58 MHz 的初始 UART 频率。
- 对 OSR 进行编程（参见 19.6.13 节）以获得 372x 过采样。
- 对 LCR（参见 19.6.7 节）进行编程以获得 8 位字符、使能校验和偶校验。
- 对与智能卡相关的 GPIO 信号进行编程以实现（按以下顺序）如下目的：
 - 复位为低。
 - 为卡提供 VCC（GPIO 管脚没有所需的 200 mA 驱动）。
 - VPP（如果已提供给卡）处于“空闲”状态。
- 对 SCICTRL 进行编程（参见 19.6.14 节）以使能带有所需功能选项的智能卡特性。
- 设置一个或多个定时器，为 ISO 7816 的启动提供所需时序。

之后，软件应监控 UART 和定时器的状态，以便按照 ISO 7816 3.2.b 及后续版本中的说明与智能卡进行交互。

19.6.15 UART4 RS485 控制寄存器（U4RS485CTRL—0x400A 404C）

U4RS485CTRL 寄存器控制 RS-485/EIA-485 模式的配置。

表429. UART4 RS485 控制寄存器位描述（U4RS485CTRL—0x400A 404C）

位	符号	值	描述	复位值
0	NMMEN	0	禁止 RS-485/EIA-485 普通多点模式（NMM）。	0
		1	使能 RS-485/EIA-485 普通多点模式。在该模式下，当接收字节的校验位为 1 时，对地址进行检测,从而产生一个接收数据中断。参见 19.6.18 节。	
1	RXDIS	0	使能接收器。	0
		1	禁能接收器。	
2	AADEN	0	禁止自动地址检测（AAD）。	0
		1	使能自动地址检测。	
3	-	-	保留。读取值未定义，只写入 0。	无

位	符号	值	描述	复位值
4	DCTRL	0	禁能自动方向控制。	0
		1	使能自动方向控制。	
5	OINV		该位保留了 U4_OE 管脚上方向控制信号的极性。	0
		0	当发送器有数据要发送时，方向控制管脚会被驱动为逻辑“0”。在最后一个数据位被发送出去后，该位就会被驱动为逻辑“1”。	
		1	当发送器有数据要发送时，方向控制管脚会被驱动为逻辑“1”。在最后一个数据位被发送出去后，该位就会被驱动为逻辑“0”。	
31:6	-		保留。读取值未定义，只写入 0。	无

19.6.16UART4 RS-485 地址匹配寄存器（U4RS485ADRMATCH—0x400A 4050）

U4RS485ADRMATCH 寄存器包含 RS-485/EIA-485 模式的地址匹配值。

表430. UART4 RS-485 地址匹配寄存器位描述（U4RS485ADRMATCH—0x400A 4050）

位	符号	描述	复位值
7:0	ADRMATCH	包含了地址匹配值。	0
31:8	-	保留。读取值未定义，只写入 0。	无

19.6.17UART4 RS-485 延时值寄存器（U4RS485DLY—0x400A 4054）

对于最后一个停止位离开 TXFIFO 到撤销 U4_OE 信号之间的延时，用户可对 8 位的 RS485DLY 寄存器进行编程。该延时时间以波特率时钟周期为单位的。可以设定任何从 0 到 255 位时间的延时。

表431. UART4 RS-485 延时值寄存器位描述（U4RS485DLY—0x400A 4054）

位	符号	描述	复位值
7:0	DLY	包含了方向控制（U4OE）延时值。该寄存器与一个 8 位计数器一起工作。	0
31:8	-	保留。读取值未定义，只写入 0。	无

19.6.18RS-485/EIA-485 操作模式

RS-485/EIA-485 特性允许将 UART 配置为可寻址从机。可寻址从机是同一主机控制的多个从机之一。

UART 主机发送器通过将校验位设置为“1”来标识地址字符。对于数据字符，校验位会设置为“0”。

每一个 UART 从机接收器都会被分配给一个唯一的地址。从机可设定为手动或自动丢弃那些与本机地址不相符的数据。

RS-485/EIA-485 正常多点模式（NMM）

该模式是通过置位 RS485CTRL 的位 0 来启动的。在该模式下，校验位用于另一个用途，即区分所接收数据中的地址和数据。

如果接收器被禁能（RS485CTRL 位 1=“1”），则所有接收到的数据字节都会被忽略且不会

存储在 RXFIFO 中。当检测到一个地址字节时（校验位=“1”）时，接收到的数据会被存储在 RXFIFO 中，并生成 Rx 数据就绪中断。此时，处理器可以读出地址字节，并决定是否使能接收器接收后面的数据。

当接收器被使能（RS485CTRL 位 1=“0”）时，所有接收到的字节（无论是数据还是地址）都会被接收并存储在 RXFIFO 中。

RS-485/EIA-485 自动地址检测（AAD）模式

如果同时设置了 RS485CTRL 寄存器的位 0（9 位模式使能）和位 2（AAD 模式使能），则 UART 处于自动地址检测模式。

在该模式下，接收器会将接收到的所有地址字节（校验位=“1”）与 RS485ADRMATCH 寄存器中编程的 8 位值进行比较。

如果接收器被禁能（RS485CTRL 位 1=“1”），则任何接收到的字节，无论是数据字节还是地址字节，只要与 RS485ADRMATCH 的值不匹配，都会被丢弃。

当检测到匹配的地址字符时，地址字符和校验位就会被一起存储到 RXFIFO 中，此外，接收器将被自动使能（RS485CTRL 位 1 将会由硬件清零）。接收器还会产生 n Rx 数据就绪中断。

当接收器被使能（RS485CTRL 位 1=“0”）时，所有接收到的字节都会被接收并存储在 RXFIFO 中，直到接收到的地址字节与 RS485ADRMATCH 的值不匹配为止。出现不匹配的地址字节时，接收器会通过硬件自动禁能（RS485CTRL 位 1 将会置位），所接收到的非匹配地址字符不会存储在 RXFIFO 中。

RS-485/EIA-485 自动方向控制

RS485/EIA-485 模式包含有一个选项，允许发送器自动控制 U4OE 管脚的状态作为一个方向控制输出信号。

该特性是通过将 RS485CTRL 的位 4 设置为“1”来使能的。

自动方向控制使能后，当 CPU 向 TXFIFO 中写入数据时，U4OE 管脚变为有效（驱动为低电平）。最后一个数据位一旦被发送出去，该管脚就会被设为无效（驱动为高电平）。参见 RS485CTRL 寄存器中的位 4 和位 5。

自动方向控制使能后，当 CPU 向 TXFIFO 中写入数据时，所选管脚将变为有效（驱动为低电平）。最后一个数据位一旦被发送出去，该管脚就会被设为无效（驱动为高电平）。参见 RS485CTRL 寄存器中的位 4 和位 5。

RS485/EIA-485 驱动器延时

驱动器延时是指最后一个停止位离开 TXFIFO 到使 U4OE 无效之间的延时。该延时可以在 8 位 RS485DLY 寄存器中进行编程。该延时时间是以波特率时钟周期为单位。可以设定任何从 0 到 255 位时间的延时。

RS485/EIA-485 输出反置

U4OE 管脚上方向控制信号的极性可通过对 U4RS485CTRL 寄存器中的位 5 进行编程来实现反置。当该位置位时，方向控制管脚将在发送器有数据要发送时被驱动为逻辑 1。在最后一个数据发送出去后，方向控制管脚会被驱动为逻辑 0。

19.6.19 UART4 同步模式控制寄存器（U4SYNCCTRL—0x400A 4058）

SYNCCTRL 寄存器控制同步模式。当该模式生效时，UART 在 SCLK 管脚上生成或接收一个位时钟，并将其应用于发送和接收移位寄存器。

表432. UART4 同步模式控制寄存器位描述（U4SYNCCTRL—0x400A 4058）

位	符号	值	描述	复位值
0	SYNC		使能同步模式。	0
		0	禁能	
		1	使能	
1	CSRC		时钟源选择。	0
		0	同步从模式（SCLK 入）	
		1	同步主模式（SCLK 出）	
2	FES		下降沿采样。	0
		0	在 SCLK 的上升沿对 RxD 进行采样	
		1	在 SCLK 的下降沿对 RxD 进行采样	
3	TSBYPASS		同步从模式下的发送同步旁路。	0
		0	在被用于时钟沿检测逻辑前，输入时钟首先会被同步。	
		1	在被用于时钟沿检测逻辑前，输入时钟没有进行同步。这种情况下就可以以潜在的亚稳度为代价来获得更高的输入时钟率。	
4	CSCEN		持续主时钟使能（仅在 CSRC 为 1 的情况下使用）	0
		0	仅当字符被发送至 TxD 时在 SCLK 上生成时钟	
		1	SCLK 持续运行（RxD 上字符的接收可独立于 TxD 上发送情况而进行）	
5	SSDIS		起始/停止位	0
		0	与在其他模式下一样发送起始位和停止位。	
		1	不发送起始/停止位。	
6	CCCLR		持续时钟清零	0
		0	CSCEN 受软件控制。	
		1	每个字符被接收后，硬件清零 CSCEN。	
31:6	-		保留。从保留位读取的值未定义。	无

在复位后，同步模式被禁用。同步模式可通过置位 SYNC 位来使能。当 SYNC 为 1 时，UART 按如下步骤进行操作：

- 1. CSRC 位控制 UART 在 SCLK 管脚上是发送（主模式）还是接收（从模式）一个串行位时钟。
- 2. 当 CSRC 为 1 选择了主模式时，如果 CSCEN=1，UART 在 SCLK 上连续生成时钟；如果 CSCEN=0，仅当 TxD 上有数据正在发送时，UART 才会在 SCLK 上生成时钟。
- 3. SSDIS 位用于控制是否使用起始位和停止位。当 SSDIS 为 0 时，UART 与在其他模式下一样，发送并采样起始位和停止位。当 SSDIS 为 1 时，UART 既不发送也不采样起始位和停止位，且 SCLK 的每个下降沿在 RxD 上采样一个数据位到接收移位寄存器中，并且将发送移位寄存器移位。

在本节的剩余部分将详细说明当 SYNC 为 1 时的操作。

TxD 上的数据在 SCLK 的下降沿发生变化。当 SSDIS 为 0 时，FES 位控制 UART 是在 SCLK 的上升沿还是下降沿采样 RxD 的串行数据。当 SSDIS 为 1 时，UART 会忽略 FES，并始终在 SCLK 的下降沿对 RxD 进行采样。

当 SYNC=1、CSRC=1、CSCEN=1 且 SSDIS=1 时，操作模式非常复杂，因为 SCLK 会同时用于数据流的两个方向，而且没有明确的机制可以通知接收器有效数据是在 TxD 中还是 RxD 中。

由于缺少上述机制，SSDIS=1 可以与 CSCEN=0 或 CSRC=0 一起用作一种类似于 SPI 协议的模式，即在 SCLK 上，每 8 个时钟周期组内 UART 会与远程设备“交换”字符（至少是概念上的）。以上操作可称为全双工模式，但在一个较高级协议控制之下，相同硬件模式可以用于一种半双工模式，此时，无论要向哪一方向发送数据，SCLK 源都会每隔 N 个时钟周期切换一次。（N 是位数/字符。）

当 UART4 作为时钟源（CSRC=1）时，这种半双工操作可能需要将一个虚拟字符写入发送保持寄存器，从而生成 8 个时钟以接收一个字符。CCCLR 位提供了一种更简单的方法，可用于设置半双工接收。当较高级协议要求 UART 接收字符时，软件应向 SYNCCTRL 寄存器中写入 CSCEN=1 和 CCCLR=1。当 UART 已发送 N 个时钟周期，从而接收到一个字符之后，它会清零 CSCEN 位。如果之后需要接收更多字符，软件可以重复设置 CSCEN 和 CCCLR。

除上述半双工操作外，CSCEN=1 主要与 SSDIS=0 一起使用，这样起始位就能指示各个方向每个字符的发送情况。

19.6.20UART4 发送使能寄存器（U4TER—0x400A C05C）

U4TER 寄存器用于使能软件流控制。当 TXEn=1 时，只要数据可用，UART4 发送器就一直发送数据。一旦 TXEn 变为 0，UART4 就会停止发送。

[表 433](#) 中描述了如何使用 TXEn 位来实现软件流控制。

表433. UART4 发送使能寄存器位描述（U4TER—0x400A 4030）

位	符号	描述	复位值
0	TXEN	该位为 1 时（复位后），一旦先前的数据都被发送出去后，写入 THR 的数据就会在 TXD 管脚上输出。 如果在发送某字符时该位被清零，那么将该字符发送完毕后就不再发送数据，直到该位被再次被置位。也就是说，该位为 0 时会阻止字符从 THR 或 TX FIFO 传输到发送移位寄存器。当接收到 XOFF 字符（DC3）时，软件通过执行软件握手可以将该位清零。在接收到 XON（DC1）字符时，软件又能将该位重新置位。	1
7:6	-	保留。读取值未定义，只写入 0。	无

20.1 基本配置

CAN1/2 外设使用下列寄存器进行设置：

1. 功率：在 PCONP 寄存器（[表 37](#)）中，设置 PCAN1/2 位。
注：复位时，CAN1/2 模块会被禁能（PCAN1/2=0）。
2. 外设时钟：CAN 接口使用公共 PCLK，它既用于总线接口，也用于大多数 APB 外设的功能部分。参见 [4.6.4](#) 节。
注：如果所需使用的 CAN 波特率必须高于 100kbit/s（参见[表 445](#)），那么就不能选择 IRC 作为时钟源（参见 [4.6](#) 节）。
3. 唤醒：CAN 控制器可将微控制器从掉电模式唤醒，参见 [4.7.9](#) 节。
4. 管脚：在相关的 IOCON 寄存器（[8.4.1](#) 节）中选择 CAN1/2 管脚及其管脚模式。
5. 中断：CAN 中断是通过 CAN1/2IER 寄存器（[表 444](#)）来使能的。
利用相应的中断置位使能寄存器来使能 NVIC 中的中断。
6. CAN 控制器初始化：参见 CANMOD 寄存器（[20.7.1](#) 节）。

20.2 CAN 控制器

控制器局域网络（CAN）是一个用于串行数据通信的高性能通信协议。CAN 控制器提供了一个完整的 CAN 协议（遵循 CAN 规范 V2.0B）实现方案。微控制器包含了该片上 CAN 控制器，用来构建功能强大的局域网，支持极高安全级别的分布式实时控制，可以在汽车、工业环境、各种高速网络以及低成本多路联机的应用中发挥很大的作用。因此，能大大精简线缆（wiring harness），且具有强大的诊断监控功能。

CAN 模块可同时支持多条 CAN 总线，允许设备在多种应用中用作网关、交换机或路由器。

CAN 模块由两部分组成：控制器和验收滤波器。所有的寄存器和 RAM 都以 32 位字宽度来访问。

20.3 特性

20.3.1 通用 CAN 特性

- 兼容 CAN 规范 2.0B、ISO11898-1。
- 多主机结构，带有无破坏性的位仲裁。
- 总线访问优先级由报文标识符（11 位或 29 位）决定。
- 可高优先级报文确保了等待时间。
- 传输速率可编程（高达 1Mbit/s）。
- 多播和广播报文功能。
- 数据长度：0 至 8 字节。
- 强大的错误处理能力。
- 非归零（NRZ）编码/解码，带有位填充。

20.3.2 CAN 控制器特性

- 2 个 CAN 控制器和总线
- 支持 11 位和 29 位的标识符。
- 双重接收缓冲器和三态发送缓冲器。
- 可编程的错误报警界限和可读/写的错误计数器。
- 仲裁丢失捕获和错误代码捕获（带有详细的位位置）。
- 单次触发的发送（不会重复发送）。
- 只听模式（无应答，无活动错误标志）。
- “自身”报文接收（自接收请求）。

20.3.3 验收滤波器特性

- 快速的硬件实现搜索算法，支持大量的 CAN 标识符。
- 全局验收滤波器识别所有 CAN 总线的 11 位和 29 位的 Rx 标识符。
- 允许 11 位和 29 位 CAN 标识符的明确定义和分组定义。
- 验收滤波器可以为选中的标准标识符提供 FullCAN-style 自动接收。

20.4 管脚描述

表434. CAN 管脚描述

管脚名称	类型	描述
CAN_RD1, CAN_RD2	输入	串行输入。来自 CAN 收发器。
CAN_TD1CAN_TD2,	输出	串行输出。输出到 CAN 收发器。

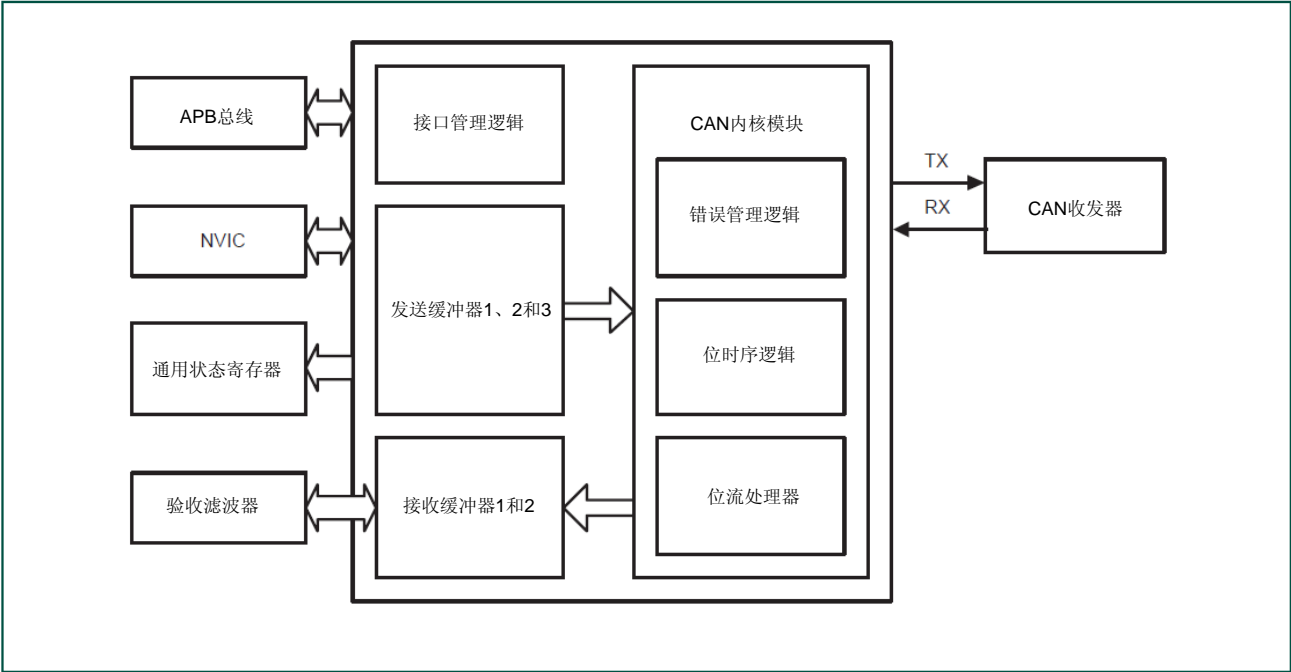
20.5 CAN 控制器结构

CAN 控制器是一个发送和接收缓冲器的完全串行接口，但它不含有验收滤波器。验收滤波器是独立的模块，对所有 CAN 通道进行 CAN 标识符进行过滤。除了报文缓冲和验收过滤之外，CAN 控制器的功能与 PeliCAN 类似。

CAN 控制器模块包括了连接到下列模块的接口：

- APB 接口
- 验收滤波器
- 可嵌套向量中断控制器（NVIC）
- CAN 收发器
- 通用状态寄存器

图77. CAN 控制器模块框图



20.5.1 APB 接口模块（AIB）

APB 接口模块提供对所有 CAN 控制器寄存器的访问。

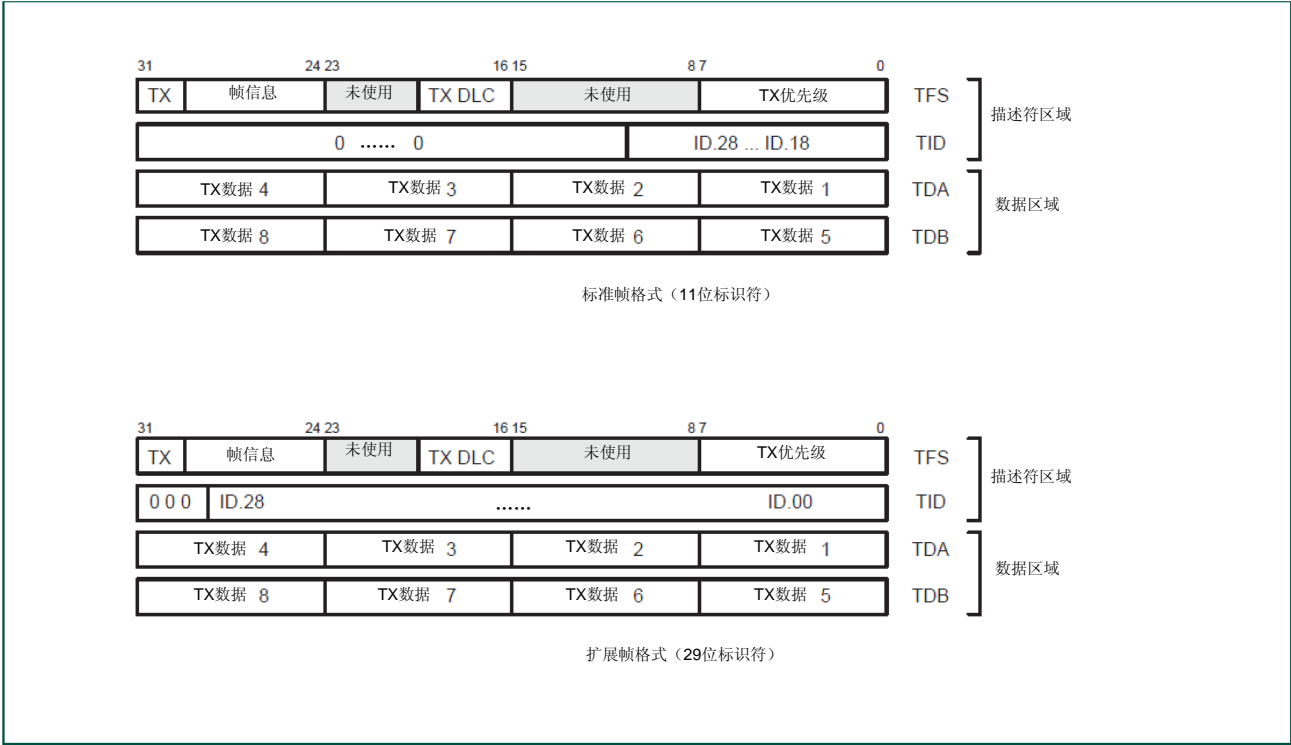
20.5.2 接口管理逻辑（IML）

接口管理逻辑会解读 CPU 发送的命令、控制 CAN 寄存器的内部寻址，向 CPU 提供中断和状态信息。

20.5.3 发送缓冲器（TXB）

TXB 是一个三态发送缓冲器，它位于接口管理逻辑（IML）与比特流处理器（BSP）之间。每个发送缓冲器都可以存储一个将要在 CAN 网络上发送的完整报文。该缓冲器由 CPU 写入，并由 BSP 读出。

图78. 标准和扩展帧格式配置的发送缓冲器的分布



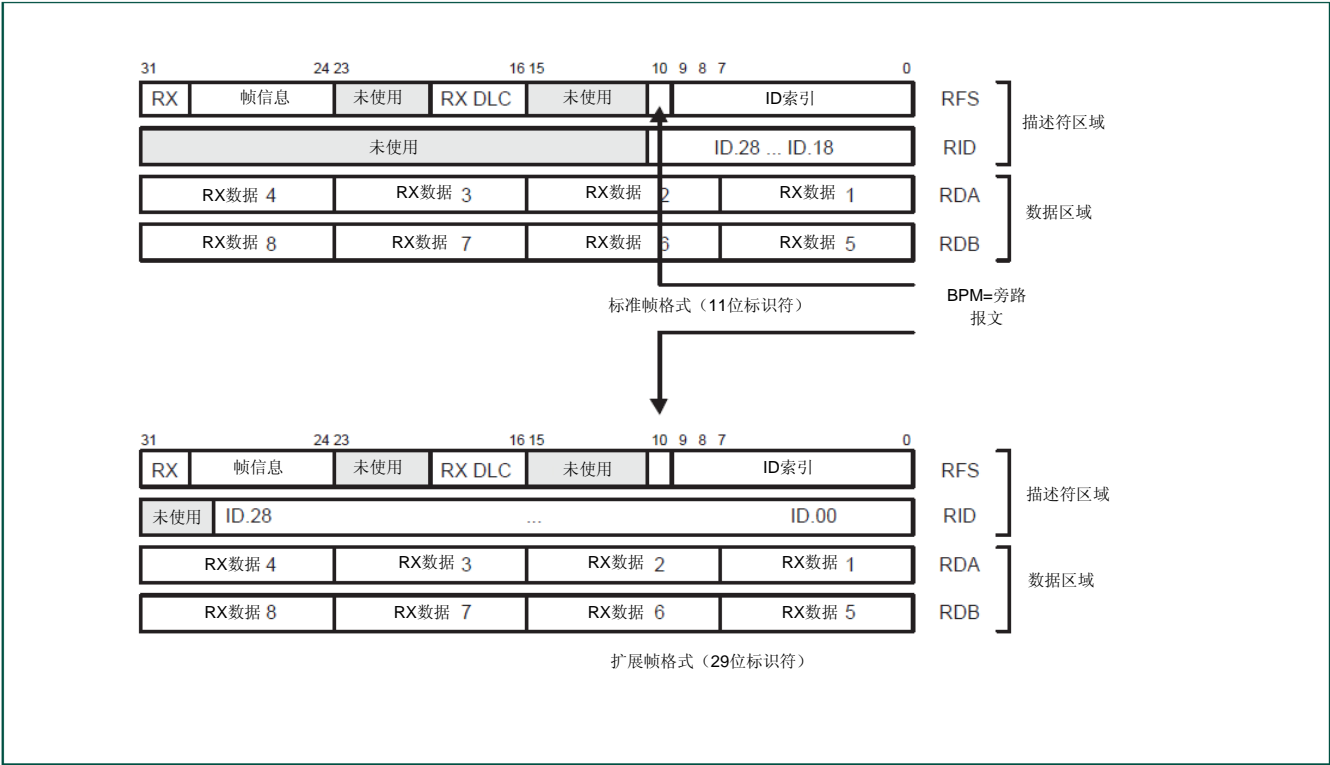
20.5.4 接收缓冲器（RXB）

接收缓冲器（RXB）是一个 CPU 可访问的双重接收缓冲器。它位于 CAN 控制器内核模块与 APB 接口模块之间，其中存储着从 CAN 总线接收到的所有报文。有了这个双重接收缓冲器，CPU 能够在处理一个报文的同时接收另一个报文。

接收缓冲器的整体分布与上文描述的发送缓冲器非常相似。标识符、帧格式、远程传输请求位以及数据长度代码的意义和发送缓冲器中的描述一样。另外，接收缓冲器还包含一个 ID 索引字段（参见 20.7.9.1 节）。

接收到的数据长度代码代表的是实际发送的数据长度代码，其值可能会大于 8，这是由发送 CAN 节点决定的。但是，接收到的最大数据字节为 8，这一点应在读取接收缓冲器的报文时考虑到。如果接收缓冲器中没有足够的空间存放一个新的报文，在这个报文变为有效且通过验收测试时，CAN 控制器就会产生一个数据超载条件。已部分写入接收缓冲器的报文（当数据超载条件发生时）会被删除。并且，该状态会通过状态寄存器和数据超载中断（如果使能了中断）告知 CPU。

图79. 标准和扩展帧格式配置的接收缓冲器的分布



20.5.5 错误管理逻辑（EML）

EML 负责错误界定。 它从 BSP 获取错误报告，然后通知 BSP 和 IML 错误的相关统计。

20.5.6 位时序逻辑（BTL）

位时序逻辑监控串行 CAN 总线，并处理与总线线路相关的位时序。 该逻辑在报文开头的“隐性”到“显性”的跳变时同步到 CAN 总线比特流（硬同步），在接收报文时使用其他跳变来重复同步（软同步）。BTL 还提供了可编程时间段来弥补传输延迟时间和相移（例如：由于振荡器漂移而引起的），从而定义采样点和在一个位时间内的采样次数。

20.5.7 比特流处理器（BSP）

比特流处理器是一个时序器，控制发送缓冲器、接收缓冲器以及 CAN 总线之间的数据流。同时，它还在 CAN 总线上执行错误检测、仲裁、填充以及错误处理。

20.5.8 CAN 控制器自测试

CAN 控制器支持两个不同的自测试：

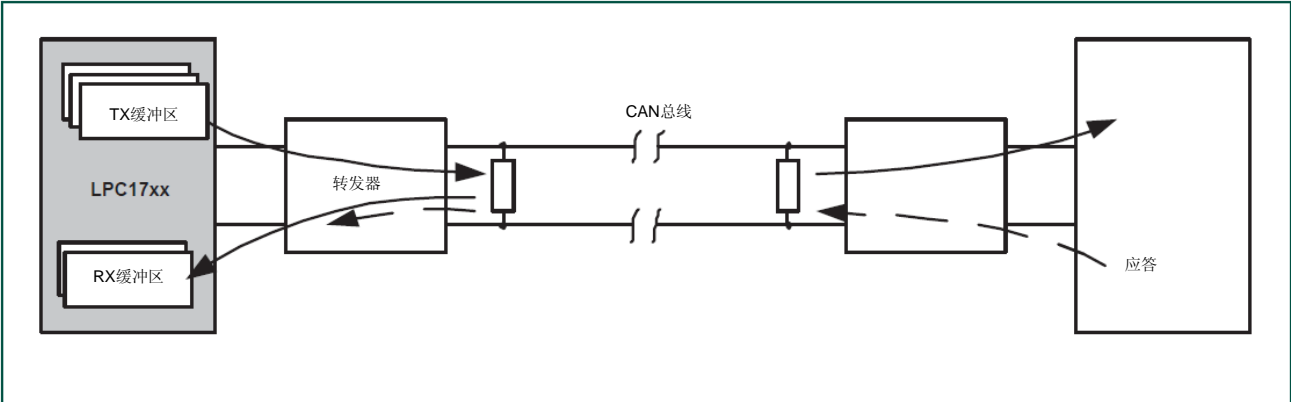
- 全局自测试（在正常操作模式下设置自接收请求位）
- 局部自测试（在自测试模式下设置自接收请求位）

两种自测试都在使用 CAN 控制器的“自接收”功能。 使用自接收请求时，发送的报文被接收并存放 to 接收缓冲器中。 因此，验收滤波器必须进行相应的配置。 CAN 报文一旦发送出去，就会产生发送中断和接收中断（如果中断被使能的话）。

全局自测试

全局自测试可以用来检验在一个给定的 CAN 系统中所选 CAN 控制器配置。如图 80 所示，至少还要有另一个正在应答每个 CAN 报文的 CAN 节点连接到 CAN 总线上。

图80. 全局自测试（以高速 CAN 总线为例）

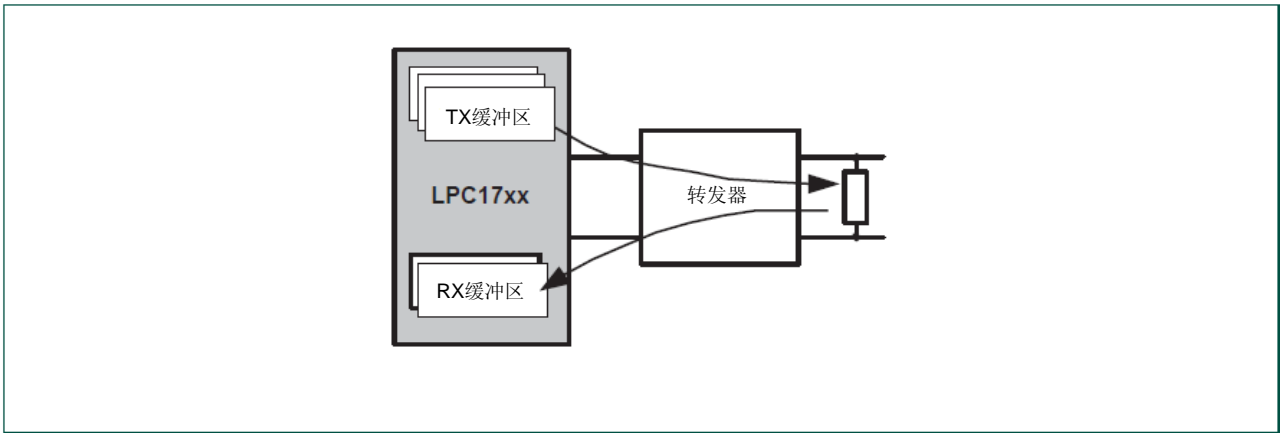


启动一次全局自测试与启动一次正常的 CAN 发送类似。在启动全局自测试的情况下，CAN 报文的发送是通过置位自接收请求位（SRR）和 CAN 控制器命令寄存器（CANCMR）中的报文缓冲位（STB3、STB2 和 STB1）来启动的。

局部自测试

局部自检非常适用于单个节点测试。在这种自测试模式下，不需要其他节点的应答。如下图所示，带有适合 CAN 总线终端的 CAN 收发器必须连接到 LPC178x/177x。CAN 控制器必须通过置位 CAN 控制器模式寄存器（CANMOD）中的 STM 位来进入“自测试”模式。提示： 设置自测试模式位（STM）时，CAN 控制器必须处于复位模式。

图81. 本地自测试（以高速 CAN 总线为例）



报文发送是通过置位自接收请求位（SRR）和选中的报文缓冲器（STB3、STB2 和 STB1）来启动。

20.6 CAN 模块的内存映射

CAN 控制器和验收滤波器占用了大量的 APB 插槽（slot），如下所示：

表435. CAN 模块的内存映像

地址范围	用途
0x4003 8000—0x4003 87FF	验收滤波器 RAM。
0x4003 C000—0x4003 C017	验收滤波器寄存器。
0x4004 0000—0x4004 000B	集中 CAN 寄存器。
0x4004 4000—0x4004 405F	CAN 控制器 1 寄存器。
0x4004 8000—0x4004 805F	CAN 控制器 2 寄存器。
0x400F C110—0x400F C114	CAN 唤醒和睡眠寄存器。

20.7 CAN 控制器寄存器

CAN 模块执行如[表 436](#)和[表 437](#)所示的寄存器。详细描述如下：

表436. CAN 验收滤波器和集中 CAN 寄存器

名称	描述	访问	复位值	地址	表
AFMR	验收滤波器寄存器	R/W	1	0x4003 C000	表 465
SFF_sa	标准帧单个起始地址寄存器	R/W	0	0x4003 C004	表 466
SFF_GRP_sa	标准帧组起始地址寄存器	R/W	0	0x4003 C008	表 467
EFF_sa	扩展帧起始地址寄存器	R/W	0	0x4003 C00C	表 468
EFF_GRP_sa	扩展帧组起始地址寄存器	R/W	0	0x4003 C010	表 469
ENDofTable	AF 表格结束寄存器	R/W	0	0x4003 C014	表 470
LUTerrAd	LUT 错误地址寄存器	RO	0	0x4003 C018	表 471
LUTerr	LUT 错误寄存器	RO	0	0x4003 C01C	表 472
CANTxSR	CAN 集中发送状态寄存器	RO	0x0003 0300	0x4004 0000	表 460
CANRxSR	CAN 集中接收状态寄存器	RO	0	0x4004 0004	表 461
CANMSR	CAN 集中其它状态寄存器	RO	0	0x4004 0008	表 462

表437. CAN1 和 CAN2 控制器寄存器映射

名称	描述	访问	复位值	CAN1&2 寄存器名称&地址	表
MOD	控制 CAN 控制器的操作模式	R/W	1	CAN1MOD—0x4004 4000 CAN2MOD—0x4004 8000	表 440
CMR	影响 CAN 控制器状态的命令位	WO	0	CAN1CMR—0x4004 4004 CAN2CMR—0x4004 8004	表 441
GSR	全局控制器状态和错误计数器	RO ^[1]	0x3C	CAN1GSR—0x4004 4008 CAN2GSR—0x4004 8008	表 442
ICR	中断状态、仲裁丢失捕获、错误代码捕获	RO	0	CAN1ICR—0x4004 400C CAN2ICR—0x4004 800C	表 443
IER	中断使能	R/W	0	CAN1IER—0x4004 4010 CAN2IER—0x4004 8010	表 444

名称	描述	访问	复位值	CAN1&2 寄存器名称&地址	表
BTR	总线时序	R/W ^[2]	0x1C0000	CAN1BTR—0x4004 4014 CAN2BTR—0x4004 8014	表 445
EWL	错误报警界限	R/W ^[2]	0x60	CAN1EWL—0x4004 4018 CAN2EWL—0x4004 8018	表 446
SR	状态寄存器	RO	0x3C3C3C	CAN1SR—0x4004 401C CAN2SR—0x4004 801C	表 447
RFS	接收帧状态	R/W ^[2]	0	CAN1RFS—0x4004 4020 CAN2RFS—0x4004 8020	表 448
RID	接收到的标识符	R/W ^[2]	0	CAN1RID—0x4004 4024 CAN2RID—0x4004 8024	表 449
RDA	接收到的数据字节 1~4	R/W ^[2]	0	CAN1RDA—0x4004 4028 CAN2RDA—0x4004 8028	表 451
RDB	接收到的数据字节 5~8	R/W ^[2]	0	CAN1RDB—0x4004 402C CAN2RDB—0x4004 802C	表 452
TFI1	发送帧信息 (Tx 缓冲器 1)	R/W	0	CAN1TFI1—0x4004 4030 CAN2TFI1—0x4004 8030	表 453
TID1	发送标识符 (Tx 缓冲器 1)	R/W	0	CAN1TID1—0x4004 4034 CAN2TID1—0x4004 8034	表 454
TDA1	发送数据字节 1~4 (Tx 缓冲器 1)	R/W	0	CAN1TDA1—0x4004 4038 CAN2TDA1—0x4004 8038	表 456
TDB1	发送数据字节 5~8 (Tx 缓冲器 1)	R/W	0	CAN1TDB1—0x4004 403C CAN2TDB1—0x4004 803C	表 457
TFI2	发送帧信息 (Tx 缓冲器 2)	R/W	0	CAN1TFI2—0x4004 4040 CAN2TFI2—0x4004 8040	表 453
TID2	发送标识符 (Tx 缓冲器 2)	R/W	0	CAN1TID2—0x4004 4044 CAN2TID2—0x4004 8044	表 454
TDA2	发送数据字节 1~4 (Tx 缓冲器 2)	R/W	0	CAN1TDA2—0x4004 4048 CAN2TDA2—0x4004 8048	表 456
TDB2	发送数据字节 5~8 (Tx 缓冲器 2)	R/W	0	CAN1TDB2—0x4004 404C CAN2TDB2—0x4004 804C	表 457
TFI3	发送帧信息 (Tx 缓冲器 3)	R/W	0	CAN1TFI3—0x4004 4050 CAN2TFI3—0x4004 8050	表 453
TID3	发送标识符 (Tx 缓冲器 3)	R/W	0	CAN1TID3—0x4004 4054 CAN2TID3—0x4004 8054	表 454
TDA3	发送数据字节 1~4 (Tx 缓冲器 3)	R/W	0	CAN1TDA3—0x4004 4058 CAN2TDA3—0x4004 8058	表 456
TDB3	发送数据字节 5~8 (Tx 缓冲器 3)	R/W	0	CAN1TDB3—0x4004 405C CAN2TDB3—0x4004 805C	表 457

[1] 错误计数器只能在 CANMOD 寄存器中的 RM 为 1 时被写入。

[2] 这些寄存器只能在 CANMOD 寄存器中的 RM 为 1 时被写入。

每个 CAN 控制器的内部寄存器都是作为片上存储器映射外部寄存器呈现在 CPU 面前。由于 CAN 控制器可以在不同模式下工作（操作/复位，另请参见 [20.7.1](#) 节），因此它必须分辨不同的内部地址定义。 请注意，某些寄存器的写访问只允许在复位模式下执行。

表438. CAN1 和 CAN2 控制器寄存器概括

通用名称	操作模式 读操作	写操作	复位模式 读操作	写操作
MOD	模式	模式	模式	模式
CMR	0x00	命令	0x00	命令
GSR	全局状态和错误计数器	-	全局状态和错误计数器	只有错误计数器
ICR	中断和捕获	-	中断和捕获	-
IER	中断使能	中断使能	中断使能	中断使能
BTR	总线时序	-	总线时序	总线时序
EWL	错误报警界限	-	错误报警界限	错误报警界限
SR	状态	-	状态	-
RFS	Rx 信息和索引	-	Rx 信息和索引	Rx 信息和索引
RID	Rx 标识符	-	Rx 标识符	Rx 标识符
RDA	Rx 数据	-	Rx 数据	Rx 数据
RDB	Rx 信息和索引	-	Rx 信息和索引	Rx 信息和索引
TFI1	Tx 信息 1	Tx 信息	Tx 信息	Tx 信息
TID1	Tx 标识符	Tx 标识符	Tx 标识符	Tx 标识符
TDA1	Tx 数据	Tx 数据	Tx 数据	Tx 数据
TDB1	Tx 数据	Tx 数据	Tx 数据	Tx 数据

表439. CAN 唤醒和睡眠寄存器

名称	描述	访问	复位值	地址	表
CANSLEEPCLR	允许清除当前 CAN 通道的睡眠状态，并读出该状态。	R/W	0	0x400F C110	表 458
CANWAKEFLAGS	允许读出 CAN 通道的唤醒状态。	R/W	0	0x400F C114	表 459

在下面的寄存器表中，“复位值”列显示了硬件复位对各个位或字段的影响，而“RM 置位”列则显示了软件由于总线关闭条件（Bus-Off condition）而将 RM 位置位时每个位或字段所受的影响。 请注意，当硬件复位 RM 的情况发生时，如果某些位对应的“复位值”列和“RM 置位”列中值不同，则优先采用“复位值”列中的设置。 在这两列中，X 都表明位或字段的值不变。

20.7.1 CAN 模式寄存器（CAN1MOD—0x4004 4000，CAN2MOD—0x4004 8000）

CAN 模式寄存器的内容用于更改 CAN 控制器的行为。 该寄存器的位由 CPU 来置位或复位，CPU 将其用作一个读写存储器。

表440. CAN 模式寄存器位描述（CAN1MOD—地址 0x4004 4000，CAN2MOD—地址 0x4004 8000）

位	符号	值	功能	复位值	RM 置位
0	RM ^{[1][6]}		复位模式。	1	1
		0（正常）	CAN 控制器处于操作模式，某些寄存器不能被写入。		
		1（复位）	禁能 CAN 操作，可写的寄存器可以被写入，终止当前报文的发送/接收。		
1	LOM ^{[3][2][6]}		只听模式	0	x
		0（正常）	CAN 控制器应答 CAN 总线上成功接收到的报文。错误计数		

位	符号	值	功能	复位值	RM 置位
2	STM ^{[3][6]}		器在当前值处停止。		
		1（只听）	控制器不应答，即使成功接收到报文。 报文不能被发送，控制器在“错误被动（error passive）”模式下工作。 该模式用于软件位速率检测和“热插拔”。		
		0（正常）	发送的报文必须被应答才被视为发送成功。即使没有接收到应答，控制器也认定一个 Tx 报文成功。	0	x
3	TPM ^[4]	1（自测）	在该模式下可能使用 CANxCMR 的 SRR 位执行一个完整的节点测试，而不需要总线上其它任何一个有效的节点。		
			发送优先级模式。	0	x
		0（CAN ID）	3 个发送缓冲器的发送优先级由 CAN 标识符决定。		
4	SM ^[5]	1（本地优先级）	3 个发送缓冲器的发送优先级由发送缓冲器内的 Tx 优先级寄存器的内容决定。		
			睡眠模式。	0	0
		0（唤醒）	正常操作。		
5	RPM	1（睡眠）	如果没有 CAN 中断被挂起也没有总线活动，则 CAN 控制器进入睡眠模式。 详见 20.8.2 节睡眠模式的描述。		
			接收极性模式。	0	x
		0（低有效）	RD 输入低有效（显性位=0）		
6	-	1（高有效）	RD 输入高有效（显性位=1）—反极性。		
			保留。 读取值未定义，只写入 0。	0	0
			测试模式。	0	x
7	TM	0（禁能）	正常操作。		
		1（使能）	TD 管脚将反映该位的值（在系统时钟的下一个正相沿的作用下），这个值可以在 RD 管脚上检测到。		
			保留。 读取值未定义，只写入 0。	无	
31: 8	-				

- [1] 在硬件复位过程中或当总线状态位被置为“1”（总线关闭）后，复位模式位就被置为“1”（呈现出来）。 复位模式位被置为“0”时，CAN 控制器将等待：
- 一次总线空闲信号的出现（11 个隐性位），如果上述复位是由硬件复位引起的或者由 CPU 发起的。
 - 128 次总线空闲信号的出现，如果上述复位是在重新进入总线开启模式前已由 CAN 控制器发起总线关闭引起的。
- [2] 该操作模式强制 CAN 控制器进入错误被动模式。 报文不可能被发送。 只听模式可以被用于软件驱动的位速率检测和“热插拔”。
- [3] 只有之前进入了复位模式，才能对 MOD.1 和 MOD.2 位执行写访问。
- [4] 发送优先级模式的详细说明请参见 20.5.3 节。
- [5] 如果睡眠模式位被置为“1”（睡眠），且没有总线活动，也没有 CAN 中断处于挂起状态，CAN 控制器将进入睡眠模式。 在至少一个之前提到的异常有效的情况下，SM 的置位会导致一个唤醒中断。 如果 SM 被设为低（唤醒）或者有总线活动，CAN 控制器将唤醒。 唤醒时会产生一个唤醒中断。 如果一个处于睡眠模式的 CAN 控制器因为总线活动唤醒，该控制器在检测到 11 个连续隐性位（总线空闲序列）前不能收到该报文。 注：在复位模式下 SM 不能置位。 在清除复位模式后，只有重新检测到总线空闲时 SM 才可以被置位。
- [6] 在写操作之前当 RM 位为 1 时，LOM 和 STM 位才能被写入。

20.7.2 CAN 命令寄存器 (CAN1CMR—0x4004 x004, CAN2CMR—0x4004 8004)

对这个只写寄存器执行写操作会启动一个 CAN 控制器传输层的操作。读出的寄存器值为 0。

在两个命令之间，至少需要一个内部时钟周期的处理时间。

表441. CAN 命令寄存器位描述 (CAN1CMR—地址 0x4004 4004, CAN2CMR—地址 0x4004 8004)

位	符号	值	功能	复位值	RM	置位
0 ^{[1][2]}	TR		发送请求。	0	0	
		0 (不存在)	没有发送请求。			
		1 (存在)	1 (存在) 之前已写入到 CANxTFI 和 CANxTID 以及 CANxTDA 和 CANxTDB 寄存器 (可选) 的报文排队等待从所选的发送缓冲器中发送出去。如果写入 TR=1 时, STB1、STB2 和 STB3 位的两个或者三个全部被选择, 则发送缓冲器将根据选定的优先级机制来选择 (详见 20.5.3 节)。			
1 ^{[1][3]}	AT		终止发送。	0	0	
		0 (无动作)	不终止发送。			
		1 (存在)	如果还未在处理过程中, 选定的发送缓冲器的挂起发送请求会被取消。			
2 ^[4]	RRB		释放接收缓冲器。	0	0	
		0 (无动作)	不释放接收缓冲器。			
		1 (释放)	接收缓冲器中的信息 (由 CANxRFS、CANxRID 以及 CANxRDA 和 CANxRDB 寄存器 (如果应用) 的值组成) 被释放, 这些信息可以被下次接收到的帧替换。如果接收到的帧不可用, 那么写入这个命令会清除状态寄存器的 RBS 位。			
3 ^[5]	CDO		清除数据超载位。	0	0	
		0 (无动作)	不清除数据超载位。			
		1 (清除)	清除状态寄存器中的数据超载位。			
4 ^{[1][6]}	SRR		无自接收请求。	0	0	
		0 (不存在)				
		1 (存在)	之前已写入到 CANxTFS 和 CANxTID 以及 CANxTDA 和 CANxTDB 寄存器 (可选) 的报文排队等待从选择的发送缓冲器中发送出去, 并同时接收进来。该位与前述的 TR 位不同, 原因是在这种情况下接收器在报文发送过程中未被禁能。因此, 当接收器的标识符被验收滤波器识别时, 接收器可以接收到报文。			
5	STB1		选择 Tx 缓冲器 1。	0	0	
		0 (不选择)	没有选择 Tx 缓冲器 1 进行发送。			
		1 (已选择)	选择了 Tx 缓冲器 1 进行发送。			
6	STB2		选择 Tx 缓冲器 2。	0	0	
		0 (不选择)	没有选择 Tx 缓冲器 2 进行发送。			
		1 (已选择)	选择了 Tx 缓冲器 2 进行发送。			
7	STB3		选择 Tx 缓冲器 3。	0	0	
		0 (不选择)	没有选择 Tx 缓冲器 3 进行发送。			
		1 (已选择)	选择了 Tx 缓冲器 3 进行发送。			
31: 8	-		保留。读取值未定义, 只写入 0。	无		

- [1] 同时置位命令位 **TR** 和 **AT** 会导致发送一个报文一次。在出错或仲裁丢失的情况下不会执行重复发送（单次触发的发送）。同时置位命令位 **SRR** 和 **TR** 导致使用自接收特性发送一次要发送的报文。在出错或仲裁丢失的情况下不会执行重复发送。
- 同时置位命令位 **TR**、**AT** 和 **SSR** 导致发送一个报文一次，这与前面 **TR** 和 **AT** 描述的情况相同。当状态寄存器的发送状态位置位时，内部发送请求位被自动清零。
- 同时置位命令位 **TR** 和 **SRR** 将会忽略置位的 **SRR** 位。
- [2] 如果在之前的命令中发送请求或自接收请求位被置“1”，就不能通过复位这些位来取消发送。请求的发送只能通过终止发送位来取消。
- [3] 当 **CPU** 需要暂挂之前请求的发送（例如，发送一个更紧急的报文之前）时，终止发送位被使用。已经在处理的发送不会被停止。检查发送结束状态位来判断最初的报文是否已经成功发送或被终止了。这个操作应当在发送缓冲器状态位已经被置“1”或发送中断已经产生之后才执行。
- [4] 在读出接收缓冲器的内容之后，**CPU** 可以通过置位释放接收缓冲器位为“1”来释放这部分存储空间。这会使得另外的报文立刻变得可用。如果没有其它可用的报文，则接收中断位被复位。如果发出 **RRB** 命令，在新的中断产生之前至少需要 2 个内部时钟周期。
- [5] 这个命令位用于清除数据超载状态位的数据超载条件。只要数据超载状态位被置位，就不会再产生数据超载中断。
- [6] 当产生自接收请求时，如果验收滤波器被设置成相应的标识符，则报文被发送并同时被接收。一个接收和发送中断将指示出正确的自接收（参见 [20.7.1](#) 节中自测试模式的描述）。

20.7.3 CAN 全局状态寄存器（CAN1GSR—0x4004 x008, CAN2GSR—0x4004 8008）

全局状态寄存器的内容反映了 CAN 控制器的状态。这是一个只读寄存器，但当 CANMOD 寄存器中的 RM 位被置 1 时，错误计数器可以被写入。没有列出的位读出的为 0，也应当写入 0。

表442. CAN 全局状态寄存器位描述（CAN1GSR—地址 0x4004 4008, CAN2GSR—地址 0x4004 8008）

位	符号	值	功能	复位值	RM 置位
0	RBS ^[1]		接收缓冲器状态。	0	0
		0（空）	没有报文可用。		
		1（满）	双重接收缓冲器至少接收到一个完整的报文，报文可在 CANxRFS、CANxRID 以及 CANxRDA 和 CANxRDB 寄存器（如果应用）中得到。如果后面接收到的报文都不可用，则该位可通过 CANxCMR 的释放接收缓冲器命令清零。		
1	DOS ^[2]		数据超载状态。	0	0
		0（不存在）	自从上一个清除数据超载命令发出或写入到 CANxCMR（或自从复位）就没有出现数据超载。		
		1（超载）	由于前面发送到这个 CAN 控制器的报文没有被尽快读出和释放（双重接收缓冲器没有足够的空间来存放一个新的报文），因此这个报文丢失。		
2	TBS		发送缓冲器状态。	1	1
		0（锁定）	至少有一个发送缓冲器不能供 CPU 使用，即至少有一个先前排队等待的报文等待这个 CAN 控制器的处理而仍然没有发送出去，因此，软件不应该写 Tx 缓冲器 CANxTFI、CANxTID、CANxTDA 和 CANxTDB 寄存器。		
		1（释放）	所有 3 个发送缓冲器都可以供 CPU 使用。任何一个 Tx 缓冲器中都没有发送报文正在挂起等待这个 CAN 控制器的处理，软件可以写 CANxTFI、CANxTID、CANxTDA 或 CANxTDB 寄存器。		
3	TCS ^[3]		发送结束状态。	1	X
		0（未完成）	至少有一个请求的发送还没有成功完成。		
		1（完成）	所有请求的发送都已经成功完成。		
4	RS ^[4]		接收状态。	1	0
		0（空闲）	CAN 控制器空闲。		
		1（接收）	CAN 控制器正在接收一个报文。		
5	TS ^[4]		发送状态。	1	0
		0（空闲）	CAN 控制器空闲。		
		1（发送）	CAN 控制器正在发送一个报文。		
6	ES ^[5]		错误状态。	0	0
		0（OK）	两个错误计数器的值都低于错误报警界限。		
		1（错误）	发送和接收错误计数器中的一个或两个的值都已经到达错误报警界限寄存器中设置的界限。		
7	BS ^[6]		总线状态。	0	0
		0（总线开启）	CAN 控制器正在处理总线活动。 CAN 控制器当前并未在处理/被禁止处理总线活动，因为发送错误计数器的值已经达到界限值 255。		
		1（总线关闭）	CAN 控制器正在处理总线活动。 CAN 控制器当前并未在处理/被禁止处理总线活动，因为发送错误计数器的值已经达到界限值 255。		

位	符号	值	功能	复位值	RM	置位
15: 8	-		保留。读取值未定义，只写入 0。	无		
23: 16	RXERR	-	Rx 错误计数器的当前值（一个 8 位的值）。	0	X	
31: 24	TXERR	-	Tx 错误计数器的当前值（一个 8 位的值）。	0		X

- [1] 在读出所有的报文并用“释放接收缓冲器”命令释放掉它们的存储空间之后，该位被清零。
- [2] 如果接收缓冲器中没有足够的空间存放报文时，则该报文被搁置，一旦报文有效就向 CPU 报告数据超载条件。如果这个报文没有成功完成（例如由于出错的原因），就不报告超载条件。
- [3] 只要发送请求位或自接收请求位因为 3 个发送缓冲器中的至少一个缓冲器而被置为“1”，那么发送结束状态位就被置为“0”（未完成）。发送结束状态位将一直保持为“0”，直到所有的报文都被成功发送。
- [4] 如果接收状态位和发送状态位都为“0”（空闲），CAN 总线就空闲。如果两个位都被置位，控制器就正在等待再次变成空闲。硬件复位后，11 个连续的隐性位被检测，直至达到空闲状态。总线关闭后，这将会花费 128 倍的 11 个连续的隐性位时间。
- [5] 根据 CAN 规范的规定，接收或发送过程中检测到的错误将会影响错误计数器。当有至少一个错误计数器的值到达或超出错误报警界限时，错误状态位被置位。如果中断使能，错误报警中断产生。硬件复位后错误报警界限的默认值是 96（十进制），参见 20.7.7 节。
- [6] 模式位置为“1”（存在），产生错误报警中断（如果中断使能）。然后发送错误计数器被置为“127”，接收错误计数器被清零。CAN 控制器将一直保持这种模式，直到 CPU 清零复位模式位。一旦这个过程结束，CAN 控制器就会等待最短的协议定义时间（总线空闲信号出现 128 次），发送错误计数器进行递减计数。在这之后，总线状态位被清零（总线开启）、错误状态位被置为“0”（OK）、错误计数器被复位并产生一个错误报警中断（如果中断使能）。在这段时间内读取 Tx 错误计数器可以获得总线关闭状态恢复的相关信息。

RX 错误计数器

RX 错误计数器寄存器是状态寄存器的一部分，它反映了接收错误计数器的当前值。在硬件复位后，该寄存器会初始化为 0。在操作模式下，该寄存器是作为一个只读存储器呈现在 CPU 面前的。只能在复位模式下对该寄存器执行写访问。如果出现一个总线关闭事件，RX 错误计数器就会被初始化为 0。只要总线关闭有效，写该寄存器不会产生任何影响。RX 错误计数器由下式决定：

$$\text{RX 错误计数器} = (\text{CANxGSR AND } 0\text{x}00\text{FF}0000) / 0\text{x}00010000$$

注：只有在预先进入复位模式的情况下，处于强制内容的 CPU 才能改变 RX 错误计数器的值。寄存器在复位状态下的内容强制状态，错误状态改变（状态寄存器）、错误报警或错误被动中断（由新寄存器内容强制产生）都将不再出现，直到复位状态被再次取消。

TX 错误计数器

TX 错误计数器寄存器是状态寄存器的一部分，它反映了发送错误计数器的当前值。在操作模式下，该寄存器是作为一个只读存储器呈现在 CPU 面前的。在硬件复位后，该寄存器被初始化为 0。只能在复位模式下对该寄存器进行写访问。如果出现一个总线关闭事件，TX 错误计数器会被初始化为 127，以计算最短的协议定义时间（总线空闲信号出现 128 次）。在这段时间内读取 TX 错误计数器可以获得总线关闭状态恢复的相关信息。如果总线关闭有效，就向 TXERR 写入 0~254 之内的值来清除总线关闭标志，控制器将在复位模式清除后等待一次 11 个连续隐性位（总线空闲）的时间。TX 错误计数器由下式决定：

$$\text{TX 错误计数器} = (\text{CANxGSR AND } 0\text{x}\text{FF}000000) / 0\text{x}01000000$$

向 TXERR 写入 255 来启动一个 CPU 驱动的总线关闭事件。只有在之前已经进入复位状态的情况下,CPU 才能强制改变 TX 错误计数器的值。寄存器在复位状态下为内容强制状态,错误状态改变(状态寄存器)、错误报警或错误被动中断(由新寄存器内容强制产生)都不会出现,直到复位状态被再次取消。

在离开复位状态后,解释新的 TX 计数器内容,执行总线关闭事件(就像它是一个总线错误事件强制执行一样)。这就意味着:再次进入了复位状态、TX 错误计数器被初始化为 127、RX 计数器被清零、所有相关的状态中断寄存器位被置位。现在清除复位状态将会执行协议定义的总线关闭恢复序列(等待总线空闲信号出现 128 次)。如果在总线关闭恢复结束之前(TXERR>0)进入复位状态,则总线关闭保持有效,且 TXERR 被冻结。

20.7.4 CAN 中断和捕获寄存器 (CAN1ICR—0x4004 400C, CAN2ICR—0x4004 800C)

该寄存器中的位指出了 CAN 总线事件的相关信息。这是一个只读寄存器。

中断和捕获寄存器的中断标志允许识别一个中断源。 当一个或多个位置位时,向 CPU 指示一个 CAN 中断。在 CPU 将该寄存器读出以后,除接收中断位之外的所有中断位都被置位。 中断寄存器作为一个只读存储器出现在 CPU 面前。

位 1~10 在被读出时清零。

当出现一个总线错误时,位 16~23 被捕获。同时,如果 CANIER 寄存器中的 BEIE 位为 1,该寄存器中的 BEI 位就会被置位,并且产生一个 CAN 中断。

当 CAN 仲裁丢失时,位 24~31 会被捕获。同时,如果 CANIER 寄存器中的 ALIE 位为 1,该寄存器中的 ALI 位就会被置位,并且产生一个 CAN 中断。一旦这些字节中的任何一个被捕获,其值就保持不变,直到被读出,读出时该值被释放以便捕获一个新的值。

不管读取的是寄存器的部分或整个寄存器,清零位 1~10 以及释放位 16~23 和位 24~31 都在读取 CANxICR 时出现。 也就是说,软件应当将 CANxICR 作为一个字读取,根据应用的需

要来适当地处理寄存器的所有位。

表443. CAN 中断和捕获寄存器位描述 (CAN1ICR—地址 0x4004 400C, CAN2ICR—地址 0x4004 800C)

位	符号	值	功能	复位值	RM	置位
0	RI ^[1]	0 (复位) 1 (置位)	接收中断。该位在 CANxSR 的 RBS 位和 CANxIER 的 RIE 位都为 1 时置位,表明已经接收到一个新的报文并存放在接收缓冲器中。	0		0
1	TI1	0 (复位) 1 (置位)	发送中断 1。当 CANxSR 的 TBS1 位从 0 变为 1 时(当来自 TXB1 的一个报文被成功发送或终止时)该位被置位,表明发送缓冲器 1 现在可用, CANxIER 的 TIE1 位为 1。	0		0
2	EI	0 (复位) 1 (置位)	错误报警中断。该位在 CANxSR 的错误状态位或总线状态位每次发生改变(置位或清零)时置位,在这两位改变时中断使能寄存器的 EIE 位被置位。	0	X	
3	DOI	0 (复位) 1 (置位)	数据超载中断。该位在 CANxSR 的 DOS 位从 0 变为 1 和 CANxIER 的 DOIE 位为 1 时置位。	0		0
4	WUI ^[2]	0 (复位) 1 (置位)	唤醒中断。当 CAN 控制器正处于睡眠状态时检测到总线活动,并且 CANxIER 的 WUIE 位为 1 时,该位置位。	0		0

位	符号	值	功能	复位值	RM 置位
5	EPI	0 (复位) 1 (置位)	错误被动中断。 如果 CANxIER 的 EPIE 位为 1，并且在任何方向上 CAN 控制器都在错误被动和错误主动模式之间切换，则该位置位。 这种情况是：CAN 控制器已经到达错误被动状态（至少有一个错误计数器的值超过了 CAN 协议定义的 127），或者，CAN 控制器处于错误被动状态并再次进入错误主动状态。	0	0
6	ALI	0 (复位) 1 (置位)	仲裁丢失中断。 如果 CANxIER 寄存器中的 ALIE 位被置为 1，并且 CAN 控制器在尝试发送时丢失仲裁时，该位就会被置位。 在这种情况下，CAN 节点变成了一个接收器。	0	0
7	BEI	0 (复位) 1 (置位)	总线错误中断—如果 CANxIER 的 BEIE 位为 1， 并且 CAN 控制器在总线上检测到一个错误时，该位就会被置位。	0	X
8	IDI	0 (复位) 1 (置位)	ID 就绪中断—如果 CANxIER 的 IDIE 位为 1，并且已经接收到一个 CAN 标识符（报文已经成功接收或终止），该位就会被置位。无论报文是否成功接收或者终止，该位置位，且 IDIE 位在 IER 寄存器中置位。	0	0
9	TI2	0 (复位) 1 (置位)	发送中断 2。当 CANxSR 的 TBS2 位从 0 变为 1 时（当来自 TXB2 的报文被成功发送或终止时）该位就会置位，表明发送缓冲器 2 现在可用，CANxIER 的 TIE2 位为 1。	0	0
10	TI3	0 (复位) 1 (置位)	发送中断 3。当 CANxSR 的 TBS3 位从 0 变为 1 时（当来自 TXB3 的报文被成功发送或终止时）该位就会置位，表明发送缓冲器 3 现在可用，CANxIER 的 TIE3 位为 1。	0	0
15: 11	-		保留。 从保留位读取的值未定义。	0	0
20: 16	ERRBIT		错误代码捕获： 当 CAN 控制器检测到一个总线错误时，帧内错误的位置就会被捕获到该字段中。 该位的值反映了内部的状态变量，因此并不完全是线性的：	0	X
4: 0			00011 帧起始 00010 ID28 ... ID21 00110 ID20 ... ID18 00100 起始位 00101 IDE 位 00111 ID17 ... 13 01111 ID12 ... ID5 01110 ID4 ... ID0 01100 RTR 位 01101 保留位 1 01001 保留位 0 01011 数据长度代码 01010 数据字段 01000 CRC 序列 11000 CRC 分隔符号 11001 应答间隙 (slot) 11011 应答分隔符号 11010 帧结束 10010 暂停 10001 主动错误标志 10110 被动错误标志 10011 容许的显性位 10111 错误分隔符号 11100 超载标志		
21	ERRDIR		当 CAN 控制器检测到一个总线错误时，当前位的方向被捕获到该位中。 0 发送过程中出错。	0	X

位	符号	值	功能	复位值	RM	置位
23: 22	ERRC1: 0	1	接收过程中出错。	0		X
		00	当 CAN 控制器检测到一个总线错误时，错误的类型被捕获到该字段中：位错误			
		01	格式错误			
		10	填充错误			
		11	其他错误			
31: 24	ALCBIT ^[4]	-	如果在 CAN 上尝试发送时仲裁被丢失，帧内的位编号就会被捕获到该字段中。在 ALCBIT 的内容被读出后，ALI 位清零，可以出现新的仲裁丢失中断。	0		X
		00	仲裁在标识符的第一位（MS）丢失			
		11	仲裁在 SRTS 位丢失（标准帧报文的 RTR 位）			
		12	仲裁在 IDE 位丢失			
		13	仲裁在标识符的第 12 位丢失（只适用于扩展帧）			
		30	仲裁在标识符的最后一位丢失（只适用于扩展帧）			
		31	仲裁在 RTR 位丢失（只适用于扩展帧）			

- [1] 接收中断位在读出中断寄存器时不会被清零。发出“释放接收缓冲器”命令将暂时清零 RI。如果在释放命令后接收缓冲器中有其它可用的报文，则 RI 再次置位。否则 RI 保持清零状态。
- [2] 如果当 CAN 控制器正在处理总线活动或一个 CAN 中断正在挂起时 CPU 想置位睡眠位，也可以产生唤醒中断。WUI 标志也可以在相应的使能位 WUIE 未置位时变为有效。在这种情况下，唤醒中断不会提交。
- [3] 只要总线错误出现，就会强制产生相应的总线错误中断（如果中断使能）。同时，位流处理器当前的位位置会被捕获到错误码捕获寄存器中。这个寄存器的内容会一直固定不变，直到用户软件将它的内容读出一。从这以后，捕获机制就会被再次激活，即，读 CANxICR 来使能其它总线错误中断。
- [4] 当仲裁丢失时，相应的仲裁丢失中断会被强迫产生（如果中断使能）。在那时，位流处理器的当前位位置会被捕获到仲裁丢失捕获寄存器中。这个寄存器的内容会一直固定不变，直到用户软件将它的内容读出一。从这以后，捕获机制就会被再次激活。

20.7.5 CAN 中断使能寄存器(CAN1IER—0x4004 4010, CAN2IER—0x4004 8010)

这个读/写寄存器控制着 CAN 控制器的各种事件是否会导致产生中断。该寄存器中的位 10:0 与 CANxICR 寄存器中的位 10:0 是一一对应关系。如果 CANxIER 寄存器的某个位是 0，则它对应的中断会被禁能；如果 CANxIER 寄存器中的某个位是 1，则它对应的中断源被使能触发一个中断。

表444. CAN 中断使能寄存器的位描述 (CAN1IER—地址 0x4004 4010, CAN2IER—地址 0x4004 8010)

位	符号	功能	复位值	RM	置位
0	RIE	接收中断使能。当接收缓冲器状态为“满”时，CAN 控制器请求相应的中断。	0	X	
1	TIE1	缓冲器 1 发送中断使能。当 TXB1 的报文已经成功发送或发送缓冲器 1 再次可访问时（例如，在一个终止发送命令之后），CAN 控制器请求相应的中断。	0	X	
2	EIE	错误报警中断使能。如果错误或总线状态改变（参见状态寄存器），CAN 控制器就会请求相应的中断。	0	X	
3	DOIE	数据超载中断使能。如果数据超载状态位置位（参见状态寄存器），CAN 控制器就会请求相应的中断。	0	X	
4	WUIE	唤醒中断使能。如果睡眠的 CAN 控制器唤醒，就会请求相应的中断。	0	X	
5	EPIE	错误被动中断使能。如果 CAN 控制器的错误状态从错误主动变成错误被动（反之亦然），就会请求相应的中断。	0	X	
6	ALIE	仲裁丢失中断使能。如果 CAN 控制器已经丢失了仲裁，就会请求相应的中断。	0	X	
7	BEIE	总线错误中断使能。如果已经检测到一个总线错误，CAN 控制器就请求相应的中断。	0	X	
8	IDIE	ID 就绪中断使能。当已经接收到一个 CAN 标识符时，CAN 控制器就会请求相应的中断。	0	X	
9	TIE2	缓冲器 2 发送中断使能。当 TXB2 的报文已经成功发送或发送缓冲器 2 再次可访问时（例如，在一个终止发送命令之后），CAN 控制器请求相应的中断。	0	X	
10	TIE3	缓冲器 3 发送中断使能。当 TXB3 的报文已经成功发送或发送缓冲器 3 再次可访问时（例如，在一个终止发送命令之后），CAN 控制器请求相应的中断。	0	X	
31: 11	-	保留。读取值未定义，只写入 0。	无		

20.7.6 CAN 总线时序寄存器（CAN1BTR—0x4004 4014，CAN2BTR—0x4004 8014）

该寄存器控制如何从 APB 时钟获得各种 CAN 时序。它定义了波特率预分频（BRP）和同步跳转宽度（SJW）的值。另外，它定义了位时间的长度、采样点的位置以及每个采样点的采样次数。该寄存器可以随时读出，但只能在 CANmod 的 RM 位为 1 时才能被写入。

表445. CAN 总线时序寄存器的位描述（CAN1BTR—地址 0x4004 4014，CAN2BTR—地址 0x4004 8014）

位	符号	值	功能	复位值	RM	置位
9: 0	BRP		波特率预分频。分频 APB 时钟来产生 CAN 时钟，分频值为（该字段的值+1）。	0		X
13: 10	-		保留。读取值未定义，只写入 0。	无		
15: 14	SJW		同步跳转宽度是（该字段的值+1）个 CAN 时钟。	0		X
19: 16	TESG1		从标称同步点到采样点的延时是（该字段的值+1）个 CAN 时钟。	1100		X
22:20	TESG2		从采样点到下个标称同步点的延时是（该字段的值+1）个 CAN 时钟。标称的 CAN 位时间是（该字段的值+TSEG1 的值+3）个 CAN 时钟。	001		X
23	SAM		采样			
		0	总线被采样 1 次（建议用于高速总线的情况）	0		X
		1	总线被采样 3 次（建议用于低到中速总线过滤总线线路尖峰信号的情况）			
31: 24	-		保留。读取值未定义，只写入 0。	无		

波特率预分频

CAN 系统时钟 t_{SCL} 的时间可编程，它决定了独立的位时序。CAN 系统时钟 t_{SCL} 用以下等式计算：

(7)

$$t_{SCL} = t_{CANsuppliedCLK} \times (BRP + 1)$$

同步跳转宽度

为补偿不同总线控制器的时钟振荡器之间的相位漂移，所有总线控制器都必须与当前传输的相关信号边沿再同步。同步跳变宽度 t_{SJW} 定义了可以通过一次再同步来缩短或延长的某个位时间的最大时钟周期：

(8)

$$t_{SJW} = t_{SCL} \times (SJW + 1)$$

时间段 1 和时间段 2

时间段 TSEG1 和 TSEG2 决定了每个位时间的时钟周期数以及采样点的位置：

(9)

$$t_{SYNCSEG} = t_{SCL}$$

(10)

$$t_{TSEG1} = t_{SCL} \times (TSEG1 + 1)$$

(11)

$$t_{TSEG2} = t_{SCL} \times (TSEG2 + 1)$$

20.7.7 CAN 错误报警界限寄存器(CAN1EWL—0x4004 4018, CAN2EWL—0x4004 8018)

该寄存器设置了一个 Tx 或 Rx 错误的界限，到达这个界限可以产生中断。该寄存器可以随时读出，但是只能在 CANmod 中的 RM 位为 1 时才能被写入。

表446. CAN 错误报警界限寄存器的位描述（CAN1EWL—地址 0x4004 4018，CAN2EWL—地址 0x4004 8018）

位	符号	功能	复位值	RM 置位
7: 0	EWL	在 CAN 操作过程中，这个值与 Tx 和 Rx 错误计数器的值相比较。如果其中一个计数器的值与这个值匹配，则 CANSR 的错误状态（ES）位被置位。	96 ₁₀ = 0x60	X
31: 8	-	保留。读取值未定义，只写入 0。	无	

请注意，只有在之前已经进入了复位状态，才能改变错误报警界限寄存器的内容。新的寄存器内容强制的错误状态改变（状态寄存器）和错误报警中断（由新的寄存器内容强制产生）不会出现，直到复位状态再次被取消。

20.7.8 CAN 状态寄存器（CAN1SR—0x4004 401C，CAN2SR—0x4004 801C）

该只读寄存器包含三个状态字节，其中与传输无关的位和全局状态寄存器中的对应位相同，而那些与传输相关的位反映了每个 Tx 缓冲器的状态（共 3 个 Tx 缓冲器）。

表447. CAN 状态寄存器的位描述（CAN1SR—地址 0x4004 401C，CAN2SR—地址 0x4004 801C）

位	符号	值	功能	复位值	RM	置位
0	RBS		接收缓冲器状态。 这一位与 CANxGSR 的 RBS 位相同。	0	0	
1	DOS		数据超载状态。 这一位与 CANxGSR 的 DOS 位相同。	0	0	
2	TBS1 ^[1]		发送缓冲器状态 1。	1	1	
		0（锁定）	软件不能访问 Tx 缓冲器 1，也不能写对应的 CANxTFI、CANxTID、CANxTDA 和 CANxTDB 寄存器，因为报文正在等待发送或正处于发送过程中。			
		1（释放）	软件可以将一个报文写入发送缓冲器 1 和它的 CANxTFI、CANxTID、CANxTDA 和 CANxTDB 寄存器。			
3	TCS1 ^[2]		发送结束状态。	1	x	
		0（未完成）	前面请求的 Tx 缓冲器 1 的发送没有成功完成。			
		1（已完成）	前面请求的 Tx 缓冲器 1 的发送已成功完成。			
4	RS		接收状态。 这一位与 GSR 的 RS 位相同。	1	0	
5	TS1		发送状态 1。	1	0	
		0（空闲）	没有来自 Tx 缓冲器 1 的传输。			
		1（发送）	CAN 控制器正在发送 Tx 缓冲器 1 的一个报文。			
6	ES		错误状态。 这一位与 CANxGSR 的 ES 位相同。	0	0	
7	BS		总线状态。 这一位与 CANxGSR 的 BS 位相同。	0	0	
8	RBS		接收缓冲器状态。 这一位与 CANxGSR 的 RBS 位相同。	0	0	
9	DOS		数据超载状态。 这一位与 CANxGSR 的 DOS 位相同。	0	0	
10	TBS2 ^[1]		发送缓冲器状态 2。	1	1	
		0（锁定）	软件不能访问 Tx 缓冲器 2，也不能写对应的 CANxTFI、CANxTID、CANxTDA 和 CANxTDB 寄存器，因为报文正在等待发送或正处于发送过程中。			
		1（释放）	软件可以将一个报文写入发送缓冲器 2 和它的 CANxTFI、CANxTID、CANxTDA 和 CANxTDB 寄存器。			
11	TCS2 ^[2]		发送结束状态。	1	x	
		0（未完成）	前面请求的 Tx 缓冲器 2 的发送没有成功完成。			
		1（已完成）	前面请求的 Tx 缓冲器 2 的发送已成功完成。			
12	RS		接收状态。 这一位与 GSR 的 RS 位相同。	1	0	
13	TS2		发送状态 2。	1	0	
		0（空闲）	没有来自 Tx 缓冲器 2 的传输。			
		1（发送）	CAN 控制器正在发送 Tx 缓冲器 2 的一个报文。			
14	ES		错误状态。 这一位与 CANxGSR 的 ES 位相同。	0	0	
15	BS		总线状态。 这一位与 CANxGSR 的 BS 位相同。	0	0	
16	RBS		接收缓冲器状态。 这一位与 CANxGSR 的 RBS 位相同。	0	0	
17	DOS		数据超载状态。 这一位与 CANxGSR 的 DOS 位相同。	0	0	

位	符号	值	功能	复位值	RM 置位
18	TBS3 ^[1]		发送缓冲器状态 3。	1	1
		0（锁定）	软件不能访问 Tx 缓冲器 3，也不能写对应的 CANxTFI、CANxTID、CANxTDA 和 CANxTDB 寄存器，因为报文正在等待发送或正处于发送过程中。		
		1（释放）	软件可以将一个报文写入发送缓冲器 3 和它的 CANxTFI、CANxTID、CANxTDA 和 CANxTDB 寄存器。		
19	TCS3 ^[2]		发送结束状态。	1	x
		0（未完成）	前面请求的 Tx 缓冲器 3 的发送没有成功完成。		
		1（已完成）	前面请求的 Tx 缓冲器 3 的发送已成功完成。		
20	RS		接收状态。这一位与 GSR 的 RS 位相同。	1	0
21	TS3		发送状态 3。	1	0
		0（空闲）	没有来自 Tx 缓冲器 3 的传输。		
		1（发送）	CAN 控制器正在发送 Tx 缓冲器 3 的一个报文。		
22	ES		错误状态。这一位与 CANxGSR 的 ES 位相同。	0	0
23	BS		总线状态。这一位与 CANxGSR 的 BS 位相同。	0	0
31: 24	-		保留，从保留位读出的值未定义。	无	

- [1] 如果 CPU 试图在发送缓冲器状态位为“0”（锁定）时写入这个发送缓冲器，则写入的字节不被接受并丢失（不作任何通知）。
- [2] 只要发送请求位或自接收请求位因为这个 Tx 缓冲器而置为“1”，发送结束状态位被置为“0”（未完成）。发送结束状态位将一直保持为“0”，直到一个报文被成功发送。

20.7.9 CAN 接收帧状态寄存器（CAN1RFS—0x4004 4020，CAN2RFS—0x4004 8020）

该寄存器定义了当前接收到的报文的特性。在正常模式下，它是一个只读寄存器，但如果 CANxMOD 中的 RM 位为 1，这个寄存器可以被写入用来测试。

表448. CAN 接收帧状态寄存器的位描述（CAN1RFS—地址 0x4004 4020，CAN2RFS—地址 0x4004 8020）

位	符号	功能	复位值	RM	置位
0	RIE	接收中断使能。当接收缓冲器状态为“满”时，CAN 控制器请求相应的中断。	0	X	
1	TIE1	缓冲器 1 发送中断使能。当 TXB1 的报文已经成功发送或发送缓冲器 1 再次可访问时（例如，在一个终止发送命令之后），CAN 控制器请求相应的中断。	0	X	
2	EIE	错误报警中断使能。如果错误或总线状态改变（参见状态寄存器），CAN 控制器就会请求相应的中断。	0	X	
3	DOIE	数据超载中断使能。如果数据超载状态位置位（参见状态寄存器），CAN 控制器就会请求相应的中断。	0	X	
4	WUIE	唤醒中断使能。如果睡眠的 CAN 控制器唤醒，就会请求相应的中断。	0	X	
5	EPIE	错误被动中断使能。如果 CAN 控制器的错误状态从错误主动变成错误被动（反之亦然），就会请求相应的中断。	0	X	
6	ALIE	仲裁丢失中断使能。如果 CAN 控制器已经丢失了仲裁，就会请求相应的中断。	0	X	
7	BEIE	总线错误中断使能。如果已经检测到一个总线错误，CAN 控制器就请求相应的中断。	0	X	
8	IDIE	ID 就绪中断使能。当已经接收到一个 CAN 标识符时，CAN 控制器就会请求相应的中断。	0	X	
9: 0	ID Index	如果 BP 位（见下行）为 0，该字段的值是从零开始的查找表 RAM 行的编号，验收滤波器利用该值来匹配接收标识符。标准表格中的禁能行也设定有编号，但不参与匹配。有关查 ID 索引值的例子，参见 20.17 节。	0	X	
10	BP	如果该位为 1，则在 AF 旁路模式下接收当前的报文，ID 索引字段（见上行）无意义。	0	X	
15: 11	-	保留。从保留位读出的值未定义。	无		
19: 16	DLC	该字段包含了当前接收到的报文的数据长度代码（DLC）字段。当 RTR=0 时，该字段的值与 CANRDA 和 CANRDB 寄存器中可用的数据字节数有关： 0000-0111=0~7 个字节 1000-1111=8 个字节 RTR=1 时，该字段的值指明了请求发回的数据字节数，使用相同的编码方式。	0	X	
29: 20	-	保留。读取值未定义，只写入 0。	无		
30	RTR	该位包含了当前接收到的报文的远程传输请求位。 表明接收到一个数据帧，如果 DLC 是一个非零值，数据帧中的数据可从 CANRDA 读出，也可能从 CANRDB 寄存器读出。1 表明接收到一个远程帧，这时 DLC 的值用来识别请求发送的数据字节数（使用相同的标识符）。	0	X	
31	FF	该位为 0 时，表明当前接收到的报文包含一个 11 位的标识符；该位为 1 时，表明当前接收到的报文包含一个 29 位的标识符。这一位的值会影响下面将要描述的 CANid 寄存器的内容。	0	X	

20.7.9.1 ID 索引字段

ID 索引是包含 ID 查找表位置的 Info 寄存器的一个 10 位的字段。 软件可使用该索引来简化从接收缓冲器到共享报文存储器的报文传输。只要 CANRFS 寄存器中的 ID 索引的位 10（BP）为 1，当前的 CAN 报文就能在验收滤波器旁路模式下被接收。

20.7.10 CAN 接收标识符寄存器(CAN1RID—0x4004 4024, CAN2RID -0x4004 8024)

该寄存器包含当前接收到的报文的标识符字段。 正常工作模式下这是一个只读寄存器，但是，如果 CANmod 中的 RM 位为 1，这个寄存器就能被写入用来测试。根据 CANRFS 中 FF 位的值，标识符有两种不同的格式。 有关特定 CAN 通道寄存器地址的详细信息，参见 [表 436](#)。

表449. CAN 接收标识符寄存器的位描述（CAN1RID —地址 0x4004 4024， CAN2RID—地址 0x4004 8024）

位	符号	功能	复位值	RM 置位
10: 0	ID	当前接收到的报文的 11 位标识符字段。在 CAN2.0A 中，这些位表示为 ID10-0，而在 CAN2.0B 中，它们表示为 ID29-18。	0	X
31: 11	-	保留。 从保留位读出的值未定义。	无	

表450. FF=1 时的 RX 标识符寄存器

位	符号	功能	复位值	RM 置位
28: 0	ID	当前接收到的报文的 29 位标识符字段。 在 CAN2.0B 中，这些位表示为 ID29-0。	0	X
31: 29	-	保留。 从保留位读出的值未定义。	无	

20.7.11 CAN 接收数据寄存器 A（CAN1RDA—0x4004 4028，CAN2RDA -0x4004 8028）

该寄存器包含了当前接收到的报文的前 1~4 个数据字节。在正常模式下，这是一个只读寄存器，但是，如果 CANMOD 中的 RM 位为 1，这个寄存器就能被写入用来测试。有关特定 CAN 通道寄存器地址的详细信息，参见[表 436](#)。

表451. CAN 接收数据寄存器 A 的位描述（CAN1RDA—地址 0x4004 4028，CAN2RDA—地址 0x4004 8028）

位	符号	功能	复位值	RM	置位
7: 0	Data 1	如果 CANRFS 中的 DLC 字段的值≥0001，这个字段就包含当前接收到的报文的第 1 个数据字节。	0		X
15: 8	Data 2	如果 CANRFS 中的 DLC 字段的值≥0010，这个字段就包含当前接收到的报文的第 2 个数据字节。	0		X
23: 16	Data 3	如果 CANRFS 中的 DLC 字段的值≥0011，这个字段就包含当前接收到的报文的第 3 个数据字节。	0		X
31: 24	Data 4	如果 CANRFS 中的 DLC 字段的值≥0100，这个字段就包含当前接收到的报文的第 4 个数据字节。	0		X

20.7.12 CAN 接收数据寄存器 B（CAN1RDB—0x4004 402C，CAN2RDB—0x4004 802C）

该寄存器包含了当前接收到的报文的前 5~8 个数据字节。在正常模式下，这是一个只读寄存器，但是，如果 CANMOD 中的 RM 位为 1，这个寄存器就能被写入用来测试。有关特定 CAN 通道寄存器地址的详细信息，参见[表 436](#)。

表452. CAN 接收数据寄存器 B 的位描述（CAN1RDB—地址 0x4004 402C，CAN2RDB—地址 0x4004 802C）

位	符号	功能	复位值	RM	置位
7: 0	Data 5	如果 CANRFS 中的 DLC 字段的值≥0101，这个字段就包含当前接收到的报文的第 5 个数据字节。	0		X
15: 8	Data 6	如果 CANRFS 中的 DLC 字段的值≥0110，这个字段就包含当前接收到的报文的第 6 个数据字节。	0		X
23: 16	Data 7	如果 CANRFS 中的 DLC 字段的值≥0111，这个字段就包含当前接收到的报文的第 7 个数据字节。	0		X
31: 24	Data 8	如果 CANRFS 中的 DLC 字段的值≥1000，这个字段就包含当前接收到的报文的第 8 个数据字节。	0		X

20.7.13 CAN 发送帧信息寄存器（CAN1TFI[1/2/3]—0x4004 40[30/ 40/50]，CAN2TFI[1/2/3]—0x4004 80[30/40/50]）

当 CANSR 中相应的 TBS 位为 1 时，软件可以通过对其中一个寄存器进行写操作来定义对应 Tx 缓冲器中发送报文的格式。表中未列出的位读入为 0，也应写入 0。

当使用自接收功能（自测试）时，为了方便比较，发送缓冲器中 CANxTFI 寄存器的保留位的值应该设置成接收缓冲器中希望的值，否则，保留为的值就未定义。

CAN 控制器包括三个发送缓冲器。每个发送缓冲器的长度都是 4 个字，而且能够存储一条完整的 CAN 报文，如图 78 所示。

缓冲器的布局分为描述符区域和数据区域，其中描述符区域的第一个字包括描述帧格式的 Tx 帧信息、数据长度以及 Tx 帧是远程帧还是数据帧。另外，Tx 优先级寄存器允许为每个发送报文定义一个优先级。根据所选的帧格式，紧跟其后的是标准帧格式（SFF）的 11 位标识符或扩展帧格式（EFF）的 29 位标识符。请注意，TID 区域中未使用的位必须定义为 0。TDA 和 TDB 的数据区域可包含最多八个数据字节。

表453. CAN 发送帧信息寄存器位描述（CAN1TFI[1/2/3]—地址 0x4004 40[30/40/50]，CAN2TFI[1/2/3]—地址 0x400480[30/40/50]）

位	符号	功能	复位值	RM	置位
7: 0	PRI0	如果 CANxMOD 寄存器中的 TPM（发送优先级模式）位被置为“1”，已使能的 Tx 缓冲器根据这个字段的值来竞争发送报文的权限。具有最低 TX 优先级值的缓冲器获得优先权，它们的内容也最先被发送。			X
15: 8	-	保留。读取值未定义，只写入 0。	0		
19: 16	DLC	数据长度代码。这个值在下个发送报文的 DLC 字段中被发送。另外，如果 RTR=0，这个值就控制着下个发送报文中发送的数据字节数（CANxTDA 和 CANxTDB 寄存器中的数据字节数）： 000-0111=0~7 个字节 1xxx=8 个字节	0		X
29: 20	-	保留。读取值未定义，只写入 0。	0		
30	RTR	这个值在下个发送报文的 RTR 位中被发送。如果该位为 0，则 DLC 字段提交的数据字节数从 CANxTDA 和 CANxTDB 寄存器中发送出来。如果该位为 1，则一个远程帧被发送（包含一个对应字节数的发送请求）。	0		X
31	FF	如果该位为 0，带有 11 位标识符（标准帧格式）的下个发送报文被发送；如果该位为 1，待发送的是带有 29 位标识符（扩展帧格式）的报文。	0		X

自动发送优先级检测

为了不中断发送报文流，CAN 控制器为所有发送缓冲器提供了自动发送优先级检测。选择好发送优先级模式后，内部优先级由 CAN 标识符或一个用户定义的“本地优先级”决定。如果多个报文被使能发送（TR=1），就要组织内部发送报文队列，例如带有最小 CAN 标识符（TID）或最低“本地优先级”（Tx 优先级）的发送缓冲器赢得优先权，内容被最先发送。

新的 CAN 报文在总线上发送之前要将内部调度过程的结果考虑进来。这个内部调度过程在发送错误出现之后和重新发送之前也存在。

Tx DLC

报文的数据区域中的字节数用数据长度代码（DLC）来编码。 在开始发送远程帧时，因为 RTR 位是“1”（远程），所以不考虑 DLC。 这会将已发送/接收数据的字节数强行设置为 0。但是，如果 2 个 CAN 控制器同时启动具有相同标识符的远程帧发送，那么，为了避免总线错误，DLC 必须正确设置。 为保证兼容性，不能使用 DLC>8 个字节。 如果选择了大于 8 的值，会使用 DLC 中指定的数据长度代码，使用数据帧发送 8 个字节。 数据字节计数的范围是 0~8 个字节，其编码设置如下所示：

(12)

$$DataByteCount = DLC$$

20.7.14 CAN 发送标识符寄存器（CAN1TID[1/2/3]—0x4004 40[34/44/54]，CAN2TID[1/2/3]—0x4004 80[34/44/54]）

当 CANxSR 中相应的 TBS 位为 1 时，软件可以写其中一个寄存器来定义下一条发送报文的标识符区域。 没有列出的位读数为 0，也应当写入 0。根据 CANTFI 中的 FF 位的值，该寄存器使用两种不同的格式。

在标准帧格式的报文中，CAN 标识符由 11 个位（ID.28~ID.18）组成，而在扩展帧格式的报文中，CAN 标识符由 29 个位（ID.28~ID.0）组成。 ID.28 为最高有效位，在仲裁过程中它最先在总线上被发送。 标识符就像是报文的名称，接收器使用它来进行验收过滤，标识符也决定了仲裁过程中总线访问的优先级。

表454. CAN 发送标识符寄存器位描述（CAN1TID[1/2/3]—地址 0x4004 40[34/44/54]，CAN2TID[1/2/3]—地址 0x400480[34/44/54]）

位	符号	功能	复位值	RM	置位
10: 0	ID	下一个发送报文中待发送的 11 位标识符。	0		X
31: 11	-	保留。 读取值未定义，只写入 0。	无		

表455. FF=1 时的发送标识符寄存器

位	符号	功能	复位值	RM	置位
28: 0	ID	下一个发送报文中待发送的 29 位标识符。	0		X
31: 29	-	保留。 读取值未定义，只写入 0。	无		

20.7.15 CAN 发送数据寄存器 A（CAN1TDA[1/2/3]—0x4004 40[38/48/58]，CAN2TDA[1/2/3]—0x4004 80[38/48/58]）

当 CANSR 中相应的 TBS 位为 1 时，软件可以写其中一个寄存器来定义下一条发送报文的前 1~4 个数据字节。 数据长度代码定义所传输的数据字节数。 Tx 数据字节 1 的最高有效位最先被发送。

表456. CAN 发送数据寄存器 A 位描述 (CAN1TDA[1/2/3]—地址 0x4004 40[38/48/58], CAN2TDA[1/2/3]—地址 0x400480[38/48/58])

位	符号	功能	复位值	RM 置位
7: 0	Data 1	如果对应 CANxTFI 中的 RTR=0, DLC≥0001, 这个字节就作为下个发送报文的第 1 个数据字节被发送。	0	X
15:8	Data 2	如果对应 CANxTFI 中的 RTR=0, DLC≥0010, 这个字节就作为下个发送报文的第 2 个数据字节被发送。	0	X
23: 16	Data 3	如果对应 CANxTFI 中的 RTR=0, DLC≥0011, 这个字节就作为下个发送报文的第 3 个数据字节被发送。	0	X
31: 24	Data 4	如果对应 CANxTFI 中的 RTR=0, DLC≥0100, 这个字节就作为下个发送报文的第 4 个数据字节被发送。	0	X

20.7.16 CAN 发送数据寄存器 B (CAN1TDB[1/2/3]—0x4004 40[3C/4C/5C], CAN2TDB[1/2/3]—0x4004 80[3C/4C/5C])

当 CANSR 中相应的 TBS 位为 1 时, 软件可以写其中一个寄存器来定义下一条发送报文的第 5~8 个数据字节。数据长度代码定义了传输的数据字节数。Tx 数据字节 1 的最高有效位最先被发送。

表457. CAN 发送数据寄存器 B 位描述 (CAN1TDB[1/2/3]—地址 0x4004 40[3C/4C/5C], CAN2TDB[1/2/3]—地址 0x400480[3C/4C/5C])

位	符号	功能	复位值	RM 置位
7: 0	Data 5	如果对应 CANTFI 中的 RTR=0, DLC≥0101, 这个字节就作为下个发送报文的第 5 个数据字节被发送。	0	X
15:8	Data 6	如果对应 CANTFI 中的 RTR=0, DLC≥0110, 这个字节就作为下个发送报文的第 6 个数据字节被发送。	0	X
23: 16	Data 7	如果对应 CANTFI 中的 RTR=0, DLC≥0111, 这个字节就作为下个发送报文的第 7 个数据字节被发送。	0	X
31: 24	Data 8	如果对应 CANTFI 中的 RTR=0, DLC≥1000, 这个字节就作为下个发送报文的第 8 个数据字节被发送。	0	X

20.7.17 CAN 睡眠清零寄存器 (CANSLEEPCLR—0x400F C110)

该寄存器给出了两个 CAN 通道当前的睡眠状态, 并提供了在唤醒后恢复通道时钟的方法。更多关于 CAN 睡眠特性的信息, 参见 [20.8.2](#)。

表458. CAN 睡眠清零寄存器的位描述 (CANSLEEPCLR—地址 0x400F C110)

位	符号	功能	复位值
0	-	保留。读取值未定义, 只写入 0。	无
1	CAN1SLEEP	CAN 通道 1 的睡眠状态和控制。 读: 当该位为 1 时, 表明 CAN 通道 1 正处于睡眠模式。写: 写入 1 时, 恢复 CAN 通道 1 的时钟。	0
2	CAN2SLEEP	CAN 通道 2 的睡眠状态和控制。 读: 当该位为 1 时, 表明 CAN 通道 2 正处于睡眠模式。写: 写入 1 时, 恢复 CAN 通道 2 的时钟。	0
31: 3	-	保留。读取值未定义, 只写入 0。	无

20.7.18 CAN 唤醒标志寄存器（CANWAKEFLAGS—0x400F C114）

该寄存器给出了两个 CAN 通道的唤醒状态，并允许清零唤醒事件。更多关于 CAN 睡眠特性的信息，参见 20.8.2 节。

表459. CAN 唤醒标志寄存器的位描述（CANWAKEFLAGS—地址 0x400F C114）

位	符号	功能	复位值
0	-	保留。 读取值未定义，只写入 0。	无
1	CAN1WAKE	CAN 通道 1 的唤醒状态。 读： 当该位为 1 时，表明 CAN 通道 1 的接收数据线出现了下降沿。写： 写入 1 清零该位。	0
2	CAN2WAKE	CAN 通道 2 的唤醒状态。 读： 当该位为 1 时，表明 CAN 通道 2 的接收数据线出现了下降沿。写： 写入 1 清零该位。	0
31: 3 -	-	保留。 读取值未定义，只写入 0。	无

20.8 CAN 控制器操作

20.8.1 错误处理

CAN 控制器根据 CAN 规范 2.0B 的规定计数和处理发送和接收错误。每检测到一个错误，发送和接收错误计数器的值就会递增，如果操作未发生错误，则其值会递减。 如果发送错误计数器的值为 255 时又有一个错误出现，那么 CAN 控制器会被强制进入一个称为总线关闭的状态。 在该状态下，下列寄存器位被置位： CANxSR 中的 BS 位、CANxIR 中的 BEI 和 EI 位（如果已使能），以及 CANxMOD 中的 RM 位。 RM 复位能禁能 CAN 控制器的大部分功能。 而且在这个时候，发送错误计数器被设置为 127，接收错误计数器被清零。 软件必须随后清零 RM 位。 之后，发送错误计数器递减计数 128 次证明是否在总线空闲条件下（11 个连续的隐性位）。 软件可通过读取 Tx 错误计数器来监控这个递减计数。 当递减计数结束时，CAN 控制器会清零 CANxSR 中的 BS 和 ES 位，置位 CANxSR 中的 EI 位（如果 IER 中的 EIE 为 1）。

如果 CANxMOD 中的 RM 位为 1，就可以写 Tx 和 Rx 错误计数器。向 Tx 错误计数器写入 255 将强制 CAN 控制器进入总线关闭状态。 如果总线关闭（CANxSR 中的 BS 位）为 1，向 Tx 错误计数器写入 0~254 之间的任何值都将清除总线关闭状态。 软件清除 CANxMOD 中的 RM 位之后，恢复操作之前只需要一个总线空闲条件（11 个连续的隐性位）。

20.8.2 睡眠模式

如果 CAN 模式寄存器中的 SM 位为 1、没有 CAN 中断正在挂起并且没有 CAN 总线活动，CAN 控制器将进入睡眠模式。软件只能在 CAN 模式寄存器中的 RM 为 0 时才能置位 SM；另外，软件还可以置位 CAN 中断使能寄存器中的 WUIE 位来使能任何唤醒条件引起的中断。

在下列情况下，CAN 控制器会被唤醒（如果 CAN 中断使能寄存器中的 WUIE 位为 1，则置位 CAN 中断寄存器中的 WUI 位）： a）CAN 总线上的显性位； b）软件清零 CAN 模式寄存器中的 SM。 为了响应总线活动而唤醒的睡眠 CAN 控制器，却要在检测到总线空闲（11 个连续的隐性位）以后才能接收初始报文。 如果软件置位 SM 时中断正在挂起或 CAN 总线有效，则可以立即唤醒。

发生唤醒事件时，软件需要执行以下操作：

1. 向 CANSLEEPCLR 寄存器中的相应位写入 1。
2. 向 CAN1MOD 和/或 CAN2MOD 寄存器中的 SM 位写入 0。
3. 向 CANWAKEFLAGS 寄存器中的相应位写入 1。如果该步骤执行失败，之后就会无法进入掉电模式。

如果 LPC178x/177x 处于深度睡眠或掉电模式，在 CAN 活动中断使能的情况下，CAN 活动将唤醒设备。参见 [4.7](#) 节。

20.8.3 中断

每个 CAN 控制器产生 3 个中断请求：接收、发送和“其他状态”。发送中断是 3 个 Tx 缓冲器发送中断相或的结果。所有中断共享一个 NVIC 通道。为 CAN 活动中断提供单独的中断。

20.8.4 发送优先级

如果 CANxMOD 寄存器中的 TPM 位为 0，多个使能的 Tx 缓冲器就会根据它们的 CAN 标识符（TID）来竞争报文发送权。如果 TPM 为 1，Tx 缓冲器就根据它们的 CANxTFS 寄存器中的位 7:0 的 PRIO 字段来竞争发送权。在上述两种情况下，二进制值最小的字段值拥有优先权。如果 2 个（或 3 个）发送使能的缓冲器拥有相同的最小值，则编号最小的缓冲器最先发送。

在发送每个报文之前，CAN 控制器会在多个使能的 Tx 缓冲器中进行动态选择。

20.9 集中 CAN 寄存器

为了简单而又快速地进行访问，每个 CAN 控制器状态寄存器的所有 CAN 控制器状态位都被集中在一起形成下面这些寄存器。下列寄存器的 LS 位中每个已定义的字节包含每个 CAN 控制器的一个特定状态位。

所有的状态寄存器都是只读寄存器，允许对它们进行字节、半字节和字访问。

20.9.1 集中发送状态寄存器（CANTxSR—0x4004 0000）

表460. 集中发送状态寄存器的位描述（CANTxSR—地址 0x4004 0000）

位	符号	描述	复位值
0	TS1	该位为 1 时，表明 CAN 控制器 1 正在发送一个报文（与 CAN1GSR 的 TS 相同）。	0
1	TS2	该位为 1 时，表明 CAN 控制器 2 正在发送一个报文（与 CAN2GSR 的 TS 相同）。	0
7: 2	-	保留，从保留位读出的值未定义。	无
8	TBS1	该位为 1 时，CAN1 控制器的所有 3 个 Tx 缓冲器都可供 CPU 使用（与 CAN1GSR 的 TBS 相同）。1	1
9	TBS2	该位为 1 时，CAN2 控制器的所有 3 个 Tx 缓冲器都可供 CPU 使用（与 CAN2GSR 的 TBS 相同）。1	1
15: 10	-	保留，从保留位读出的值未定义。	无
16	TCS1	该位为 1 时，CAN1 控制器已经成功完成了所有请求的发送（与 CAN1GSR 的 TCS 相同）。	1
17: 16	TCS2	该位为 1 时，CAN2 控制器已经成功完成了所有请求的发送（与 CAN2GSR 的 TCS 相同）。	1
31: 18	-	保留，从保留位读出的值未定义。	无

20.9.2 集中接收状态寄存器（CANRxSR—0x4004 0004）

表461. 集中接收状态寄存器的位描述（CANRxSR—地址 0x4004 0004）

位	符号	描述	复位值
0	RS1	该位为 1 时，表明 CAN1 正在接收一个报文（与 CAN1GSR 的 RS 相同）。	0
1	RS2	该位为 1 时，表明 CAN2 正在接收一个报文（与 CAN2GSR 的 RS 相同）。	0
7: 2	-	保留，从保留位读出的值未定义。	无
8	RB1	该位为 1 时，CAN1 控制器中一个接收到的报文可用（与 CAN1GSR 的 RBS 相同）。	0
9	RB2	该位为 1 时，CAN2 控制器中一个接收到的报文可用（与 CAN2GSR 的 RBS 相同）。	0
15: 10	-	保留，从保留位读出的值未定义。	无
16	DOS1	该位为 1 时，一个报文因为以前传输到 CAN1 控制器的报文没有被迅速读出而丢失（与 CAN1GSR 的 DOS 相同）。	0
17: 16	DOS2	该位为 1 时，一个报文因为以前传输到 CAN2 控制器的报文没有被迅速读出而丢失（与 CAN2GSR 的 DOS 相同）。	0
31: 18	-	保留，从保留位读出的值未定义。	无

20.9.3 集中其他状态寄存器（CANMSR—0x4004 0008）

表462. 集中其它状态寄存器的位描述（CANMSR—地址 0x4004 0008）

位	符号	描述	复位值
0	E1	该位为 1 时，CAN1 Tx 和 Rx 错误计数器中的一个或两个值都已经到达 CAN1EWL 寄存器设置的界限（与 CAN1GSR 的 ES 相同）。	0
1	E2	该位为 1 时，CAN2 Tx 和 Rx 错误计数器中的一个或两个值都已经到达 CAN2EWL 寄存器设置的界限（与 CAN2GSR 的 ES 相同）。	0
7: 2	-	保留，从保留位读出的值未定义。	无
8	BS1	该位为 1 时，CAN1 控制器当前正在处理总线活动（与 CAN1GSR 的 BS 相同）。	0
9	BS2	该位为 1 时，CAN2 控制器当前正在处理总线活动（与 CAN2GSR 的 BS 相同）。	0
31: 10	-	保留，从保留位读出的值未定义。	无

20.10全局验收滤波器

该模块为所有的 CAN 控制器提供了接收标识符的查找（CAN 术语中称之为验收过滤）。它包含一个 512x32（2kB）的 RAM，软件可以在其中存放 1~5 个标识符表。该 RAM 可以容纳多达 1024 个标准标识符或 512 个扩展标识符，或者两种类型混合的标识符。

20.11验收滤波器模式

通过设置验收滤波器模式寄存器（[20.14.1](#) 节）中的 AccOff、AccBP 和 eFCAN 位，验收滤波器可以进入不同的模式。在每种模式下，配置寄存器和 ID 查找表的访问操作都是不同的。

表463. 验收滤波器模式和访问控制

验收滤波器模式	AccOff 位	AccBP 位	验收滤波器状态	ID 查找表 RAM ^[1]	验收滤波器配置寄存器	CAN 控制器报文接收中断
关闭模式	1	0	复位&停止	由 CPU 对其进行读/写	由 CPU 对其进行读/写	不接收报文
旁路模式	X	1	复位&停止	由 CPU 对其进行读/写	由 CPU 对其进行读/写	接收所有的报文
工作模式和 FullCAN 模式	0	0	运行	只能由 CPU 读取 ^[2]	只能由验收滤波器来访问	硬件验收过滤

- [1] 整个 ID 查找表 RAM 只能以字的形式来访问。
- [2] 在验收滤波器的操作模式下，只有在禁能或使能报文时才能访问查找表。

只有在验收滤波器关闭模式和旁路模式中才能对所有区配置寄存器进行写访问。允许在所有的验收滤波器模式下对寄存器进行读访问。

20.11.1 验收滤波器关闭模式

验收滤波器关闭模式通常在初始化阶段使用。在该模式下，可以对所有寄存器和查找表 RAM 进行无条件访问。在验收滤波器关闭模式下，不接收 CAN 报文，所以，这些报文不会存储在有效 CAN 控制器的接收器缓冲器中。

20.11.2 验收滤波器旁路模式

验收滤波器旁路模式可以用来改变一个正在运行的系统中的验收滤波器配置，即，改变 ID 查找表存储器的标识符。在这个重新配置的过程中，必须使用软件验收过滤。

建议使用 ID 就绪中断（ID 索引）和接收中断（RI）。在该模式下，接收所有的 CAN 报文，并将其存储在有效 CAN 控制器的接收缓冲器中。

20.11.3 验收滤波器工作模式

当配置寄存器中的 AccOff 和 AccBP 都未置位且 eFCAN=0 时，验收滤波器处于工作模式。

20.11.4 FullCAN 模式

当配置寄存器中的 AccOff 和 AccBP 未置位且 eFCAN=1 时，验收滤波器处于 FullCAN 模式。有关 FullCAN 模式的更多详细信息，参见 20.16 节。

20.12 ID 查找表 RAM 的各个区

4 个 12 位的区配置寄存器（SFF_sa、SFF_GRP_sa、EFF_sa、EFF_GRP_sa）用来定义 ID 查找表存储器中不同的标识符区的边界。第 5 个 12 位的区配置寄存器是表格结束地址寄存器（ENDofTable），用来定义所有标识符区的结束。另外，表结束地址也用来分配 FullCAN 报文对象区的起始地址（如果 FullCAN 报文对象被使能存放在区中）。

表464. 区配置寄存器设置

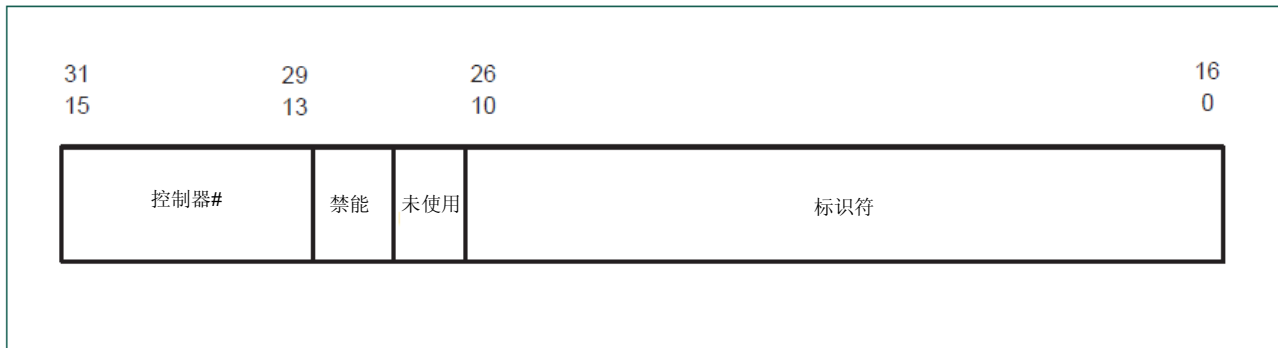
ID 查找表区	寄存器	值	区状态
FullCAN（标准帧格式）标识符区	SFF_sa	= 0x000	禁能
		> 0x000	使能
明确的标准帧格式标识符区	SFF_GRP_sa	= SFF_sa	禁能
		> SFF_sa	使能
标准帧组格式标识符区	EFF_sa	= SFF_GRP_sa	禁能
		> SFF_GRP_sa	使能
明确的扩展帧格式标识符区	EFF_GRP_sa	= EFF_sa	禁能
		> EFF_sa	使能
扩展帧组格式标识符区	ENDofTable	= EFF_GRP_sa	禁能
		> EFF_GRP_sa	使能

20.13 ID 查找表 RAM

整个 ID 查找表 RAM 只能进行字访问，并且只能在验收滤波器关闭或旁路模式下进行写访问。但读访问可以在所有的验收滤波器模式下进行。

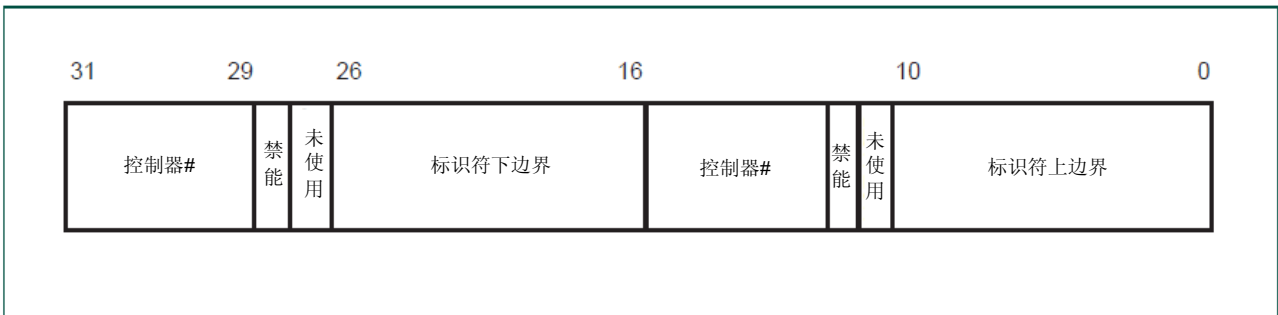
如果应用中使用的是标准标识符（11 位），则验收滤波器 RAM 的 3 个表格中至少有一个必定不为空。如果“FullCAN 模式”选项使能，第 1 个表格存放标准标识符，用于 FullCAN 模式中接收的处理。第 2 个表格存放单个标准标识符，第 3 个表格存放标准标识符的范围，使报文通过 CAN 控制器接收。FullCAN 的单个标准标识符表格必须按升序排列，每半个字为一个标识符，每个字包含 2 个标识符。由于每个 CAN 总线都有自身的地址映射，因此表格的每行都必须包含使用的 CAN 控制器的编号（001~010）。

图82. 一行 FullCAN 模式下单个标准标识符表格



标准标识符范围表格包含标识符的一对上下边界（边界值包含在内），一对范围占用一个字，按升序排列。

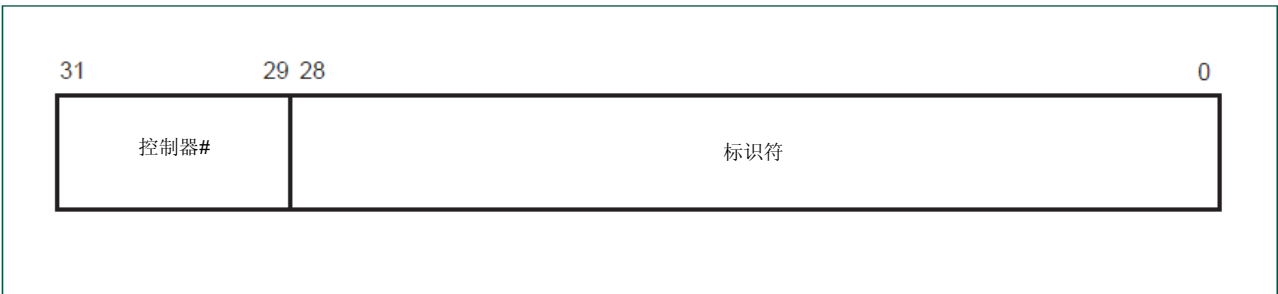
图83. 一行标准标识符范围表格



标准行中的禁能位提供了一种动态开/关特定 CAN 标识符或标识符范围响应的方法。 如果使能了验收滤波器功能，验收滤波器 RAM 中只有禁用位可通过软件来改变。 通过向 RAM 中相应的字内写入 32 位 0 来使能对标准地址范围的响应,通过写入 32 位 1(0xFFFF FFFF) 来关闭该响应。 只有禁能位被改变。 禁止行必须保存按升序排列的标识符。

如果在应用中使用的是扩展标识符（29 位），验收滤波器 RAM 其他的两个表格中至少有 1 个必定不为空，两个表格一个用来存放单个扩展标识符，一个用来存放扩展标识符的范围。 单个扩展标识符表格必须按升序排列。

图84. 一行扩展标识符表格



扩展标识符范围表格必须包含偶数个行，其格式与单个扩展标识符表格相同。与单个扩展标识符表格一样，扩展范围必须按升序排列。表中的第 1 行和第 2 行（第 3 和第 4 行.....）形成一对，作为包括边界在内的扩展地址范围。这样，包含在地址范围内（含边界）的地址就一定能接收到（接受）。软件必须保证表格由一对一对的字组成。

使用 FullCAN 方法来接收扩展标识符信息实现起来并不容易。

五个地址寄存器指向验收滤波器 RAM 中各个表格之间的边界：FullCAN 标准地址、标准单个地址、标准地址范围、扩展单个地址以及扩展地址范围。在存储器中，这些表格必须是连续的。后四个表格的起始地址分别是各自相邻的前一个表格的终止地址。扩展范围表格的结束地址在表格结束寄存器中给出。如果一个表格的起始地址等于下一个表格的起始地址或表格结束寄存器的值，则该表格为空。

当 CAN 控制器的接收方已接收到一个完整的标识符时，它将通知验收滤波器。验收滤波器会响应该信号，并读出控制器编号、标识符尺寸以及来自控制器本身的标识符。然后，验收滤波器继续搜索 RAM，以决定该接收还是忽略这个报文。

如果 FullCAN 模式使能且 CAN 控制器告知当前报文包含一个标准标识符，则验收滤波器首先查询标识符表格，以便接收可在 FullCAN 模式下处理。此外，如果 AF 未在 FullCAN 表格中发现合适的匹配，它将接着查询单个标识符，以获取 CAN 控制器给出的标识符尺寸。只要发现相等的匹配，AF 就通知 CAN 控制器保存报文，并提供一个 ID 索引值使之保存到接收帧状态寄存器中。

如果验收滤波器未在单个标识符表格中找到匹配，它会接着查询标识符范围表格，以获取 CAN 控制器给出的标识符尺寸。如果 AF 在表格范围中发现了合适的匹配，它就通知 CAN 控制器保存报文，并提供一个 ID 索引值使之保存在接收帧状态寄存器中。如果在单个标识符表格或范围表格中，AF 都未找到与接收到的标识符尺寸相匹配的值，AF 会通知 CAN 控制器丢弃/忽略接收到的报文。

20.14 验收滤波器寄存器

20.14.1 验收滤波器模式寄存器（AFMR—0x4003 C000）

验收滤波器模式寄存器的 AccBP 和 AccOff 位用来使验收滤波器进入旁路和关闭模式。模式寄存器的 eFCAN 位可以用来为接收到的 11 位 CAN ID 报文激活一个增强型的 FullCAN 模式。

表465. 验收滤波器模式寄存器的位描述（AFMR—地址 0x4003 C000 ）

位	符号	值	描述	复位值
0	AccOff ^[2]	1	如果 AccBP 为 0，验收滤波器不工作。此时，忽略 CAN 总线上所有的 Rx 报文。	1
1	AccBP ^[1]	1	所有 Rx 报文都被使能的 CAN 控制器接收。在修改下述的所有寄存器的内容以及在以任何方式修改查找表 RAM 的内容而不是在标准标识符行中设置或清除禁能位之前，软件都必须设置该位。当该位和 AccOff 位都为 0 时，验收滤波器屏蔽接收到的 CAN 标识符。	0
2	eFCAN ^[3]	0	软件必须从接收 CAN 控制器读取所有被使能的 CAN 总线上所有使能 ID 的报文。	0
		1	验收滤波器本身会为所选 CAN 总线上选定的标准 ID 值处理接收和保存的报文。参见 20.16 节。	
31: 3	-	-	保留。读取值未定义，只写入 0。	无

- [1] 验收滤波器旁路模式 (AccBP)：通过设置验收滤波器模式寄存器中的 AccBP 位使验收滤波器进入验收滤波器旁路模式。在旁路模式下，验收滤波器的内部状态机被复位和停止。所有接收到的 CAN 报文被接受，验收滤波器可以由软件来处理。
- [2] 验收滤波器关闭模式 (AccOff)：在上电或硬件复位后，验收滤波器将进入关闭模式，验收滤波器模式寄存器 0 的 AccOff 位被置为 1。验收滤波器的内部状态机被复位和停止。如果在非关闭模式中通过硬件或软件置位 AccOff 位，验收滤波器将被强制进入关闭模式。
- [3] FullCAN 模式的增强特性：接收到的 CAN 报文的 FullCAN 模式可以通过置位验收滤波器模式寄存器的 eFCAN 位来使能。

20.14.2 区配置寄存器

ID 查找表 RAM 使用 10 位的区配置寄存器来分别为 11 位 CAN 标识符和 29 位 CAN 标识符的明确 CAN 标识符和 CAN 标识符组指示不同的区边界。10 位宽度区配置寄存器允许使用 512x32（2kB）的查找表 RAM。整个 ID 查找表 RAM 只能使能字访问。全部五个区配置寄存器都包含验收滤波器 RAM 的 APB 地址，但不包含 APB 基地址。只有在验收滤波器的关闭模式和旁路模式下，才允许对所有区配置寄存器进行写访问。允许在所有的验收滤波器模式下对区寄存器进行读访问。

20.14.3 标准帧单个起始地址寄存器（SFF_sa—0x4003 C004）

表466. 标准帧单个起始地址寄存器位描述（SFF_sa—地址 0x4003 C004）

位	符号	描述	复位值
1: 0	-	保留。读取值未定义，只写入 0。	无
10: 2	SFF_sa ^[1]	AF 查找 RAM 中的单个标准标识符表格的起始地址。如果表格为空，则向这个寄存器和下述的 SFF_GRP_sa 寄存器写入相同的值。为了兼容未来的设备，这个寄存器的位 31: 11 和 1: 0 被写入 0。如果 AFMR 的 eFCAN 位为 1，这个值也可指示验收滤波器将要查找的标准 ID 的表格大小，而且，如果找到了匹配的标准 ID，接收到的报文就会被自动保存到验收滤波器 RAM 中。	0
31: 11	-	保留。读取值未定义，只写入 0。	无

- [1] 只有在验收滤波器旁路模式或验收滤波器关闭模式下才能对查找表的区配置寄存器进行写访问。

20.14.4 标准帧组起始地址寄存器（SFF_GRP_sa—0x4003 C008）

表467. 标准帧组起始地址寄存器的位描述（SFF_GRP_sa—地址 0x4003 C008）

位	符号	描述	复位值
1: 0	-	保留。读取值未定义，只写入 0。	
11: 2	SFF_GRP_sa ^[1]	AF 查找 RAM 中标准标识符组表格的起始地址。 如果表格为空，则向这个寄存器和下述的 EFF_sa 寄存器写入相同的值。 只有当标准单个标识符表格被使用并且 AF 查找表 RAM 的最后一个字（地址：0x7FC）被使用时，才允许向这个寄存器写入最大值 0x800。为了兼容未来的设备，这个寄存器的位 31: 12 和 1: 0 被写入 0。	0
31: 12	-	保留。读取值未定义，只写入 0。	无

[1] 只有在验收滤波器旁路模式或验收滤波器关闭模式下才能对查找表的区配置寄存器进行写访问。

20.14.5 扩展帧起始地址寄存器（EFF_sa—0x4003 C00C）

表468. 扩展帧起始地址寄存器的位描述（EFF_sa—地址 0x4003 C00C）

位	符号	描述	复位值
1: 0	-	保留。读取值未定义，只写入 0。	无
10: 2	EFF_sa ^[1]	AF 查找 RAM 中的单个扩展标识符表格的起始地址。 如果表格为空，则向这个寄存器和下述的 EFF_GRP_sa 寄存器写入相同的值。只有当扩展单个标识符表格为空并且 AF 查找表 RAM 的最后一个字（地址：0x7FC）被使用时，才允许向这个寄存器写入最大值 0x800。为了兼容未来的设备，这个寄存器的位 31: 11 和 1: 0 被写入 0。	0
31: 11	-	保留。读取值未定义，只写入 0。	无

[1] 只有在验收滤波器旁路模式或验收滤波器关闭模式下才能对查找表的区配置寄存器进行写访问。

20.14.6 扩展帧组起始地址寄存器（EFF_GRP_sa—0x4003 C010）

表469. 扩展帧组起始地址寄存器的位描述（EFF_GRP_sa—地址 0x4003 C010）

位	符号	描述	复位值
1: 0	-	保留。读取值未定义，只写入 0。	无
11: 2	Eff_GRP_sa ^[1]	AF 查找 RAM 中扩展标识符组表格的起始地址。 如果表格为空，则向这个寄存器和下述的 ENDofTable 寄存器写入相同的值。只有当该表格为空并且 AF 查找表 RAM 的最后一个字（地址：0x7FC）被使用时，才允许向这个寄存器写入最大值 0x800。为了兼容未来的设备，这个寄存器的位 31: 12 和 1: 0 被写入 0。	0
31: 12	-	保留。读取值未定义，只写入 0。	无

[1] 只有在验收滤波器旁路模式或验收滤波器关闭模式下才能对查找表的区配置寄存器进行写访问。

20.14.7 AF 表结束寄存器（ENDofTable—0x4003 C014）

表470. AF 表结束寄存器的位描述（ENDofTable—地址 0x4003 C014）

位	符号	描述	复位值
1: 0	-	保留。读取值未定义，只写入 0。	无
11: 2	EndofTable ^[1]	最后一个有效 AF 表格中最后一个有效地址的上个地址。为了兼容未来的设备，这个寄存器的位 31: 12 和 1: 0 被写入 0。如果 AFMR 的 eFCAN 位为 0，写入这个寄存器的最大值是 0x800，这就允许 AF 查找表 RAM 中的最后一个字（地址 0x7FC）被使用。如果 AFMR 中的 eFCAN 位为 1，这个字段的值就标识着验收滤波器 RAM 区域的起始地址，验收滤波器将自动接收所选 CAN 总线上选定 ID 的报文。在这种情况下，写入这个寄存器的最大值是 0x800 – 6 ×（SFF_sa 的值）。这就允许在这个字段的地址和验收滤波器 RAM 之前保存 12 个字节的报文，并在验收滤波器 RAM 的起始地址和下一个有效 AF 表格之间指定每个标准 ID。	0
31: 12	-	保留。读取值未定义，只写入 0。	无

[1] 只有在验收滤波器旁路模式或验收滤波器关闭模式下才能对查找表的区配置寄存器进行写访问。

20.14.8 状态寄存器

如果在 ID 筛选过程中查找表 RAM 中遇到编程错误，则查找表错误状态寄存器、错误地址以及标志寄存器会提供相关的信息。只能对查找表错误地址和标志寄存器进行读访问。如果检测到错误，LUTerror 标志就会被置位，LUTerrorAddr 寄存器提供了一个地址，在这个地址下，查找表在 ID 筛选过程中遇到了一个错误。LUTerrorAddr 滤波器模块的任何读操作都可用于查找表中断。

20.14.9 LUT 错误地址寄存器（LUTerrAd—0x4003 C018）

表471. LUT 错误地址寄存器的位描述（LUTerrAd—地址 0x4003 C018）

位	符号	描述	复位值
1: 0	-	保留。读取值未定义，只写入 0。	无
10: 2	LUTerrAd	LUT 错误位（如下）为 1，这个只读字段就包含了 AF 查找表 RAM 的一个地址，在这个地址上，验收滤波器遇到了一个表格内容错误。	0
31: 11	-	保留。读取值未定义，只写入 0。	无

20.14.10 LUT 错误寄存器（LUTerr—0x4003 C01C）

表472. LUT 错误寄存器的位描述（LUTerr—地址 0x4003 C01C）

位	符号	描述	复位值
0	LUTerr	如果验收滤波器在 AF RAM 中遇到一个表格内容错误，这个只读位就会被置为 1。当软件读取 LUTerrAd 寄存器时，该位被清零。这一条件与来自 CAN 控制器的“其它 CAN”中断进行相或（OR）以产生一个连接到 NVIC 的请求。	0
31: 1	-	保留，从保留位读出的值未定义。	无

20.14.11 全局 FullCAN 中断使能寄存器（FCANIE—0x4003 C020）

只有在验收滤波器处于关闭模式时，才允许对全局 FullCAN 中断使能寄存器进行写访问。

表473. 全局 FullCAN 中断使能寄存器位描述（FCANIE—地址 0x4003 C020）

位	符号	描述	复位值
0	FCANIE	全局 FullCAN 中断使能。该位为 1 时中断被使能。	0
31: 1	-	保留。读取值未定义，只写入 0。	无

20.14.12 FullCAN 中断和捕获寄存器（FCANIC0—0x4003 C024 与 FCANIC1—0x4003 C028）

有关这两个寄存器的详细描述，参见 20.16.2 节。

表474. FullCAN 中断和捕获寄存器 0 的位描述（FCANIC0—地址 0x4003 C024）

位	符号	描述	复位值
0	IntPnd0	FullCAN 中断挂起位 0。	0
...	IntPndx （0<x<31）	FullCAN 中断挂起位 x。	0
31	IntPnd31	FullCAN 中断挂起位 31。	0

表475. FullCAN 中断和捕获寄存器 1 的位描述（FCANIC1—地址 0x4003 C028）

位	符号	描述	复位值
0	IntPnd32	FullCAN 中断挂起位 32。	0
...	IntPndx （32<x<63）	FullCAN 中断挂起位 x。	0
31	IntPnd63	FullCAN 中断挂起位 63。	0

20.15配置和搜索算法

CAN 标识符查找表存储器可以存放标准和扩展 CAN 帧格式的明确标识符和 CAN 标识符组。它们组成一个排好序的列表或表格，在每个区中源 CAN 通道（SCC）以及 CAN 标识符都按照升序排列。

SCC 值和 CAN 控制器-1 相等，即 SCC=0 和 CAN1 匹配，而 SCC=1 和 CAN2 匹配。

每个 CAN 标识符连接到一个 ID 索引编号。在 CAN 标识符匹配时，匹配的 ID 索引存储在相应 CAN 控制器的帧状态寄存器（CANRFS）中的标识符索引中。

20.15.1 验收滤波器搜索算法

验收滤波器的标识符筛选过程按以下顺序开始：

- 1. FullCAN（标准帧格式）标识符区
- 2. 明确的标准帧格式标识符区
- 3. 标准帧组格式标识符区
- 4. 明确的扩展帧格式标识符区

5. 扩展帧组格式标识符区

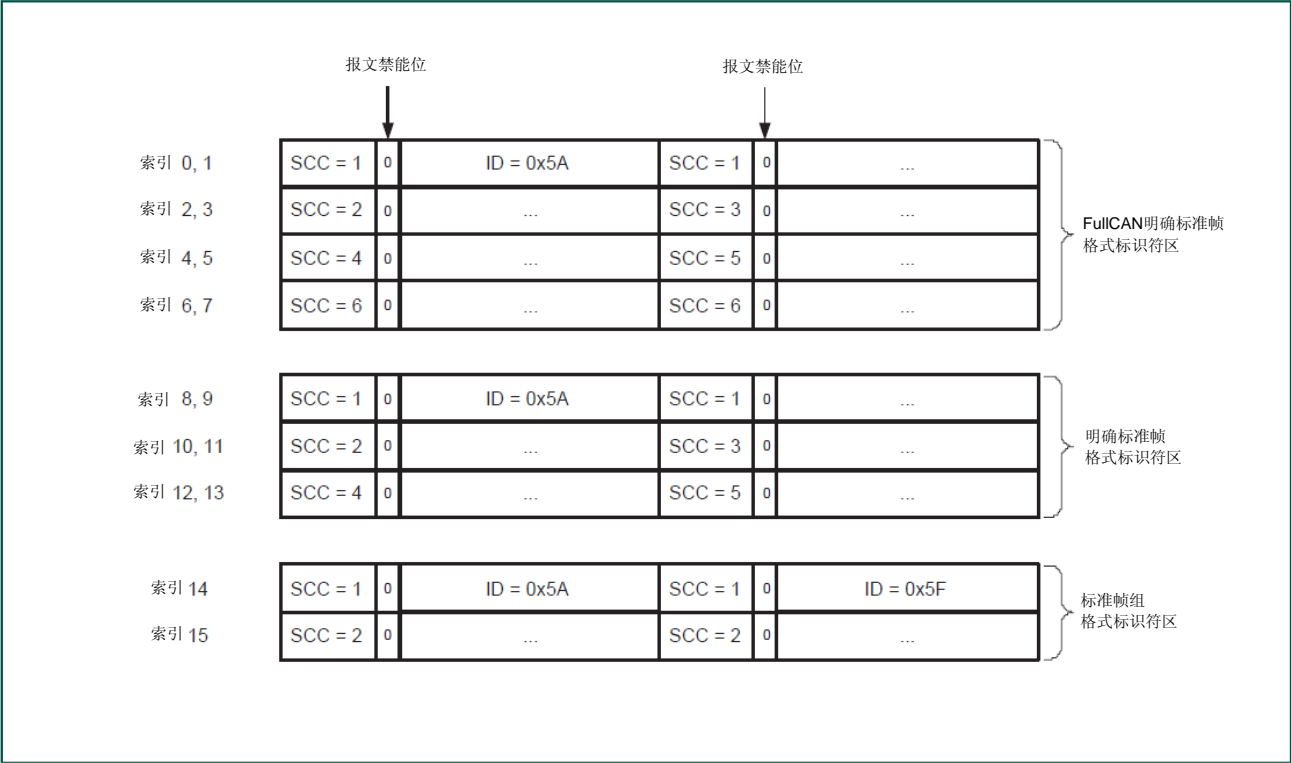
注： 只有激活的区才能参与筛选过程。

如果多个区中定义了具有相同帧格式的同一个报文标识符，则第一次匹配就会终止该标识符的筛选过程。

例如，如果在 FullCAN、明确的标准帧格式和标准帧组格式标识符区中定义了相同的源 CAN 通道以及标识符时，筛选过程在 FullCAN 区匹配时就已经结束。

如图 85 所示的例子中，带有源 CAN 通道的标识符已经在 FullCAN、明确的标准帧格式标识符区以及标准帧组格式标识符区中定义好了。 这个例子相当于有 6 个 CAN 控制器。

图85. 说明查找算法的 ID 查找表示例



全部三个区中都定义了源 CAN 通道 SCC=1 的 CAN 控制器 1 的标识符 0x5A。 如果使用该配置，CAN 控制器 1 上带有 0x5A 标识符的 CAN 报文都能在 FullCAN 区中找到匹配。

可能会有在 FullCAN 区中禁能“0x5A 标识符”的情况。 这时，筛选过程会在明确的标识符区中找到匹配时结束。

分组标识符区的第一组按照下述方式定义：源 CAN 通道的 SCC=1 的 CAN 控制器 1 会接收标识符从 0x5A 到 0x5F 的 CAN 报文。如上所述，标识符 0x5A 已经在 FullCAN 或明确的标识符区找到了匹配（如果使能）。 该组中定义的其它标识符（0x5B 到 0x5F）将会在这个分组标识符区中找到匹配。

利用这种方法，用户可以在不同滤波器模式之间动态地切换，以查找相同的标识符。

20.16 FullCAN 模式

FullCAN 模式以 LPC2xxx 系列产品中使用的 CAN 网关模块提供的功能为基础。CAN 网关模块使用验收滤波器来为 CAN 通道提供过滤。

CAN 网关模块的概念主要来源于 BasicCAN 功能。该概念非常适合网用来在不同的 CAN 通道之间传输的报文或报文数据的系统。只要接收到一个 CAN 报文，BasicCAN 设备就会产生一个接收中断。软件必须将接收到的报文从相应的 CAN 控制器的接收缓冲器移出，存放到用户 RAM 中。

为了适用仪表板之类的应用中，在这类应用中控制器通常从几个 CAN 通道中接收数据进行更进一步的处理，CAN 网关模块扩展了一个被称为 FullCAN 接收的功能。这个增加的特性使用一个内部报文处理程序来将接收到的 FullCAN 报文从相应 CAN 控制器的接收缓冲器移到查找表 RAM 的 FullCAN 报文对象数据区中。

当 FullCAN 模式被使能时，验收滤波器以仿“FullCAN”控制器的形式自己处理所选 CAN 总线上选择的标准 ID 值的报文的接收和保存。

为了设置该位并使用该模式，必须满足与验收滤波器 RAM 内容以及指向它的指针相关的另外两个条件：

- 标准帧单个起始地址寄存器（SFF_sa）的内容必须大于或等于 ID 的数目，以便自动执行接收保存操作，SFF_sa 一般是 2 的倍数。在必要时，SFF_sa 还可以为 4 的倍数。
- EndOfTable 寄存器必须小于或等于 0x800-6x（SFF_sa 的值），允许每个 ID 保存 12 字节的报文，以便自动执行接收保存操作。

当上述条件都满足且 eFCAN 被置位时：

- 验收滤波器 RAM 的起始地址和 SFF_sa 地址之间的区域用作单个标准 ID 表格和 CAN 控制器/总线的识别，这部分区域按升序排序，格式与单个标准 ID 表格相同（参见图 82）。它的行和其它标准表格一样可以标识为“禁能”。如果存在奇数个“FullCAN”ID，则表格中至少有一行被标识为“禁能”。
- 第一个（SFF_sa）/2ID 索引值分配给这些自动存储的 ID。
也就是说，当 eFCAN 为 0 时，ID 不能再在 FullCAN 模式下处理时，Rx 帧状态寄存器中保存的 ID 索引值可在原值的基础上增加（SFF_sa）/2。
- 接收到一个标准 ID 时，验收滤波器先查询这个表格，再查询标准单个标识符和标识符组表格。
- 当这个表格中接收到一个控制器和 ID 的报文时，验收滤波器将接收到的报文从 CAN 控制器中读出并存放到验收滤波器 RAM 中（起始地址为（EndOfTable）+其 ID 索引 x12）。
- 这些报文的格式如表 476 所示。

20.16.1 FullCAN 报文的分布

表476. 自动保存的 Rx 报文的格式

地址	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
0	F	R	0000				SEM	0000				DLC	0000								ID.28	...		ID.18								
	F	T					[1: 0]																									
	R																															
+4	Rx 数据 4						Rx 数据 3								Rx 数据 2																	
+8	Rx 数据 8						Rx 数据 7								Rx 数据 6																	

FF、RTR 和 DLC 字段的描述见[表 448](#)。

验收滤波器和 CPU 都可以访问查找表 RAM 的 FullCAN 报文对象区，有一个方法可以确保当验收滤波器硬件正在写 FullCAN 报文对象时不会出现 CPU 读该报文对象的情况。

为此，验收滤波器使用一个 3 种状态的信号量，对 2 个信号量 SEM1 和 SEM0 进行编码(参见[表 476](#))，使之用于每个报文对象。 该机制为 CPU 提供了 FullCAN 报文对象区中验收滤波器活动的当前状态的相关信息。

信号量工作在下列模式中：

表477. FullCAN 信号量操作

SEM1	SEM0	活动
0	1	验收滤波器正在更新内容。
1	1	验收滤波器已经完成了内容的更新。
0	0	CPU 正在读取验收滤波器。

在将第一个数据字节写入一个报文对象之前，验收滤波器将 FrameInfo 字节写入相应的缓冲器位置，同时设置 SEM[1:0]=01。

而在将最后一个数据字节写入报文对象之后，验收滤波器通过设置 SEM[1:0]=11 来标志更新完毕。

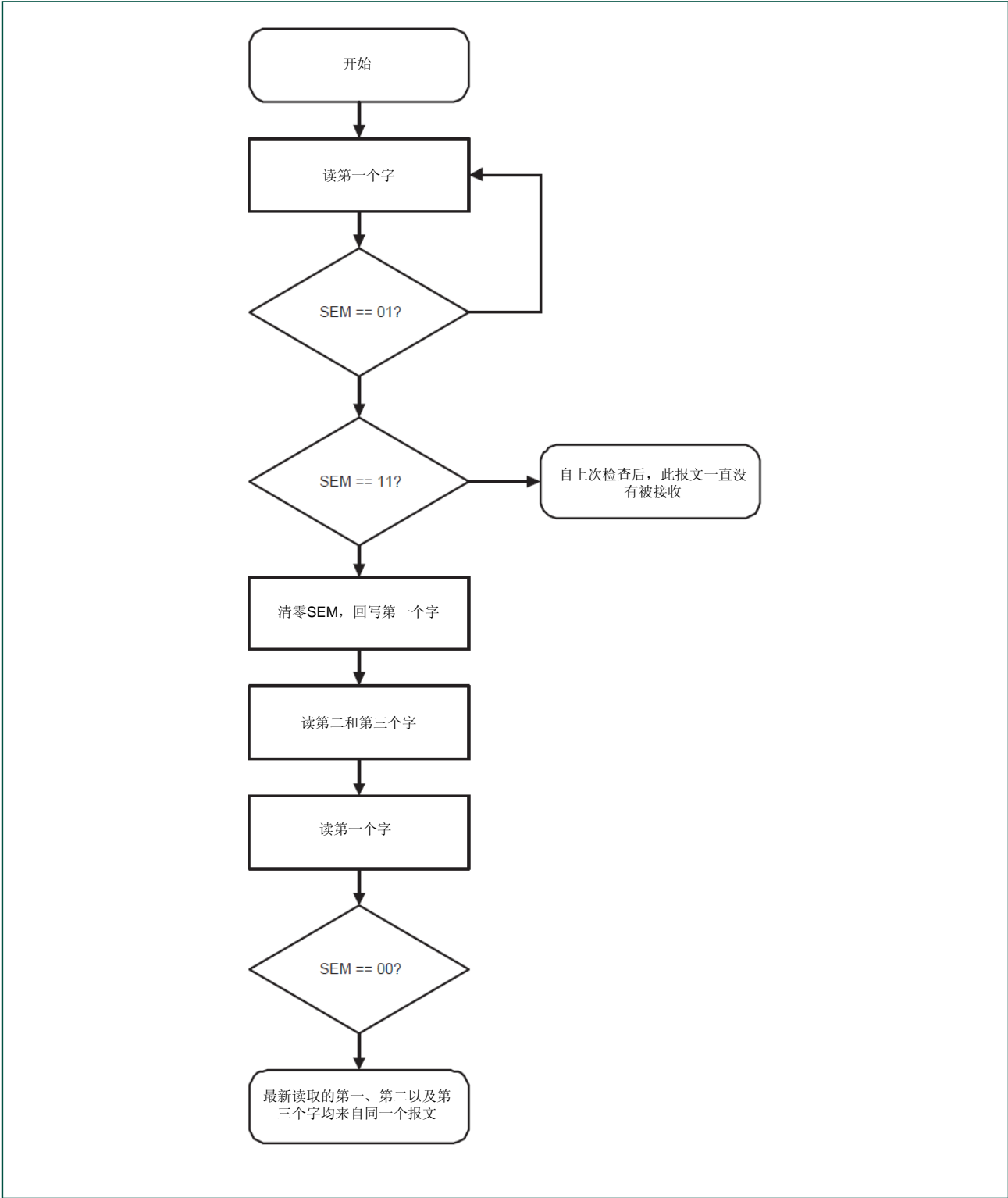
在读取一个报文对象之前，CPU 应读取 SEM[1:0]以确定验收滤波器活动的当前状态。 如果 SEM[1:0]=01，则验收滤波器当正在读取报文对象。 如果 SEM[1:0]=11，则报文对象可以被读取。

因此，CPU 开始读取报文对象前应该清零 SEM[1:0]=00。

当 CPU 完成读取时，它可以再次检查 SEM[1:0]。在这最后一次检查中，如果 SEM[1:0]=01 或 11，则在 CPU 读取报文的过程中验收滤波器已经完成了报文对象的更新，CPU 应该丢弃数据。 相反地，如果 SEM[1:0]像希望的那样为 00 时，则 CPU 已成功地读取有效数据。

[图 86](#) 显示了软件应如何使用 SEM 字段来确保从报文中读取的所有三个字都是来自同一条已接收到的报文。

图86. 读取一个自动存储报文的信号量流程



20.16.2 FullCAN 中断

CAN 网关模块包含一个 2kB 的 ID 查找表 RAM。在这个容量下，如果整个查找表 RAM 只用来存放 FullCAN 对象，则最多可以定义 146 个 FullCAN 对象。只有前 64 个 FullCAN 对象可以被配置参与中断模式。但仍然可以定义多于 64 个 FullCAN 对象。唯一的区别在于，剩余的 FullCAN 对象不能提供 FullCAN 中断。

FullCAN 中断寄存器-组包含（挂起）FullCAN 接收中断的中断标志（IntPndx）。一旦接收到一个 FullCAN 报文，FCAN 中断寄存器中相应的中断位（IntPndx）就变为有效。只要全局 FullCAN 中断使能位置位，FullCAN 接收中断就被传递给向量中断控制器。

应用软件必须解决下列问题：

1. 根据 FCANIC 中断寄存器的位位置来为多个挂起的中断计算索引/对象编号。
2. 多个 FullCAN 接收中断正在挂起时中断优先级的处理。

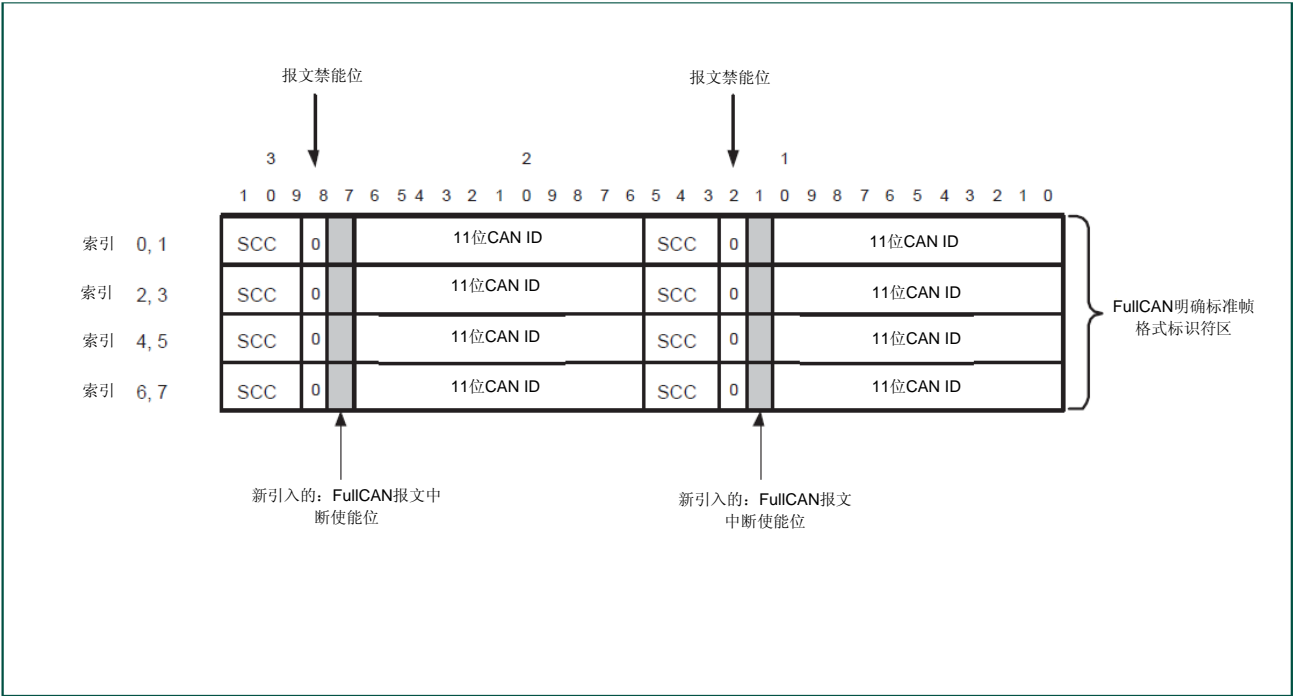
含有中断优先级处理的软件必须为每个 FullCAN 对象分配一个接收中断优先级。如果多个中断正在挂起，则软件必须决定接下来服务哪个接收到的 FullCAN 对象。

给每个 FullCAN 对象增加一个新的 FullCAN 中断使能位（FCANIntxEn），以便可以单独使能或禁能每个对象的 FullCAN 中断。引入一个新的报文丢失标志（MsgLstx）来指示自从上一次这个报文对象被 CPU 读取起是否已经接收到多个 FullCAN 报文。中断使能位和报文丢失位都位于现有的查找表 RAM 中。

20.16.2.1 FullCAN 报文中断使能位

如[图 87](#)所示，在 FullCAN 区中定义了带源 CAN 通道的 8 个 FullCAN 标识符。新引入的 FullCAN 报文中断使能位可以被用来使能每个 FullCAN 报文产生一个中断。

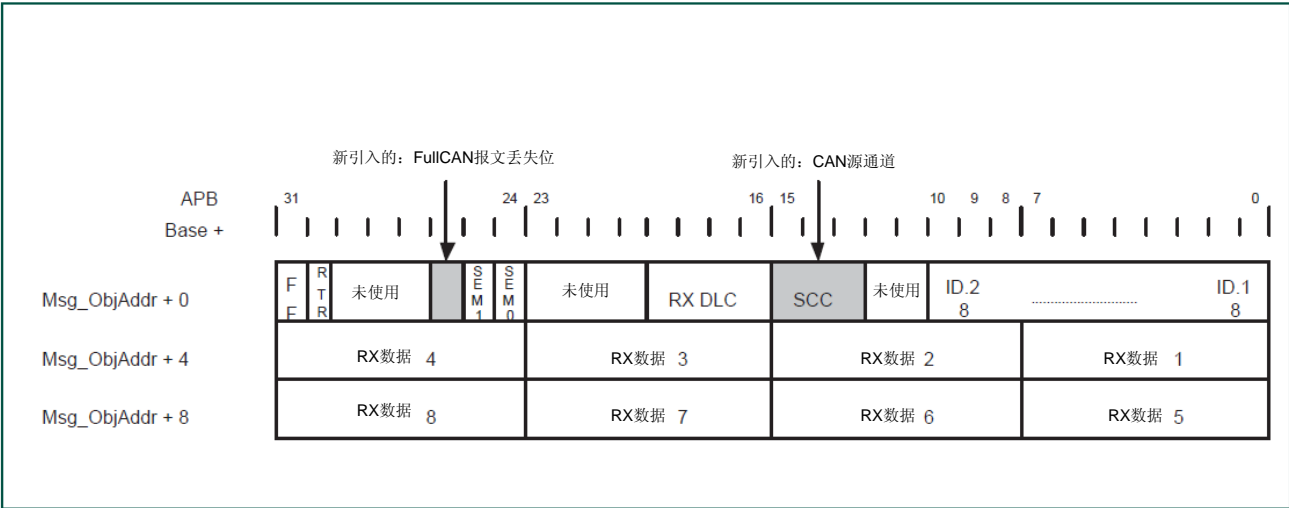
图87. ID 查找表中 FullCAN 区的示例



20.16.2.2 报文丢失位和 CAN 通道编号

图 88 是查找表 FullCAN 报文对象区中存储的一个 FullCAN 报文的详细分布结构。

图88. FullCAN 报文对象分布图



引入一个新的报文丢失标志 (MsgLst) 来指示自从上一次这个报文对象被 CPU 读取起是否已经接收到多个 FullCAN 报文。更多有关接收到的 FullCAN 报文的 CAN 源通道 (SCC) 信息被添加到报文对象中。

20.16.2.3 置位中断挂起位 (IntPnd 63~0)

当接收到一个 FullCAN 报文并且相应的 FullCAN 对象的中断被使能 (使能位 FCANIntxEn 被置位) 时, 中断挂起位 (IntPndx) 变为有效。

在**最后一次写一个 FullCAN 报文对象的数据存储区的过程中**, FullCAN 对象的中断挂起位 (IntPndx) 变为有效。

20.16.2.4 清零中断挂起位 (IntPnd 63~0)

当一个报文对象的信号量位被软件 (ARM CPU) 清零时, 每个 FullCAN 中断挂起请求都会被清除。

20.16.2.5 置位一个 FullCAN 报文对象的报文丢失位 (MsgLost 63~0)

当接收到一个 FullCAN 报文并且该对象的 FullCAN 中断有效时, FullCAN 报文对象的报文丢失位变为有效。

在**第一次写一个 FullCAN 报文对象的数据存储区的过程中**, 如果中断挂起位已经被置位, 则 FullCAN 对象的报文丢失位 (MsgLostx) 变为有效。

20.16.2.6 清零一个 FullCAN 报文对象的报文丢失位 (MsgLost 63~0)

当一个 FullCAN 报文对象的 FullCAN 中断未设为有效, 则该对象的报文丢失位会被清零。

在**第一次写一个 FullCAN 报文对象的数据存储区的过程中**, 如果中断挂起位未被设置, 则该 FullCAN 对象的报文丢失位 (MsgLostx) 会被清零。

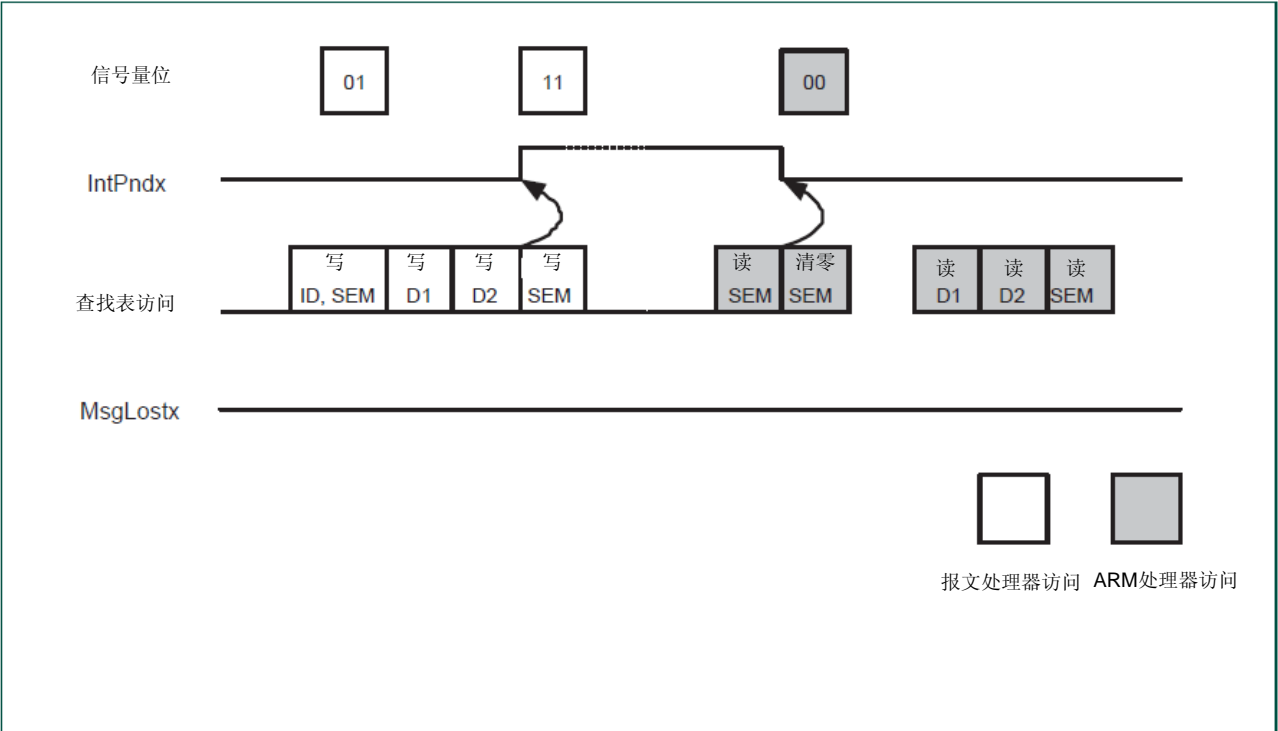
20.16.3 FullCAN 中断的置位和清零机制

内置的 FullCAN 中断置位和清零机制需要采取特殊的预防措施。下面的各小节描述了已经存在的信号量位 (详情参见 [20.16.1](#) 节) 和新引入的特性 (IntPndx、MsgLstx) 将如何作用。

20.16.3.1 状况 1: 正常情况, 无报文丢失

下文中的[图 89](#)显示了一种典型的“正常”情况: 一个接收到的 FullCAN 报文保存在 FullCAN 报文对象区中。保存好后由软件 (ARM CPU) 将报文读出。

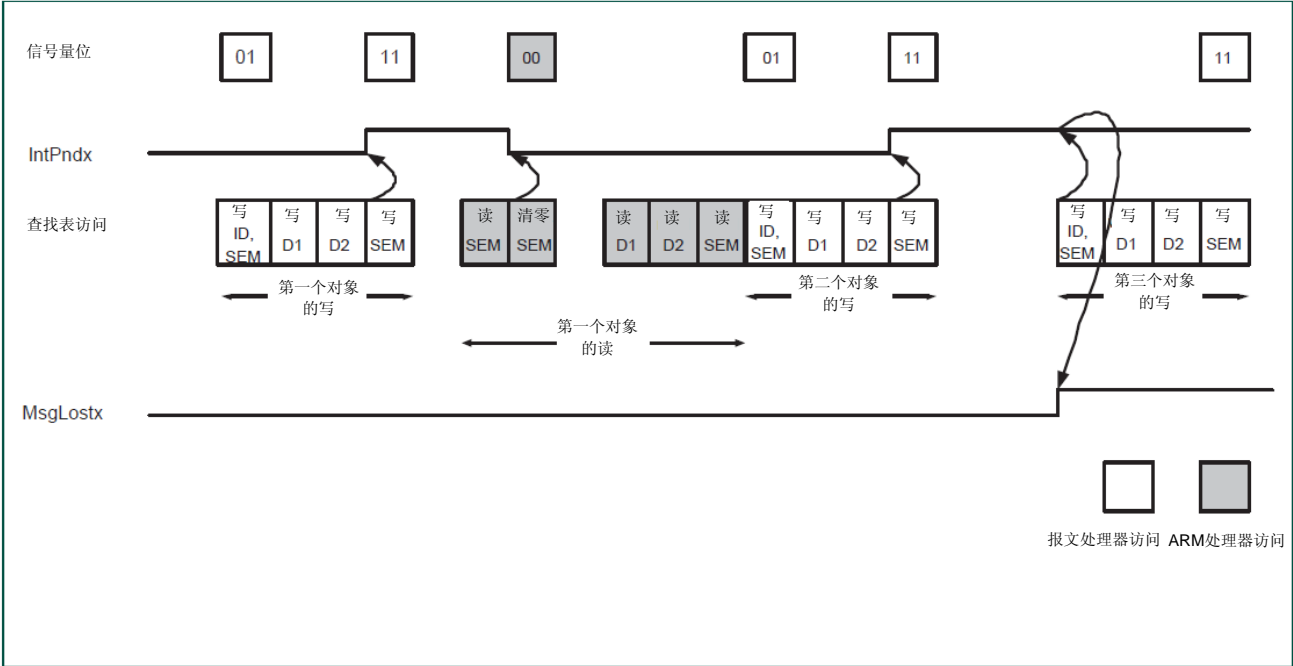
图89. 正常情况（无报文丢失）



20.16.3.2 状况 2：报文丢失

在这种情况下，第一个 FullCAN 报文保存好，然后由软件读出（第一个对象的写和读）。接下来，第二个报文被保存（第二个对象的写），但在第三个报文保存好（第三个对象的写）之前第二个报文并未被读出。由于对象的 FullCAN 中断（IntPndx）已经被声明，因此报文丢失使信号变得有效。

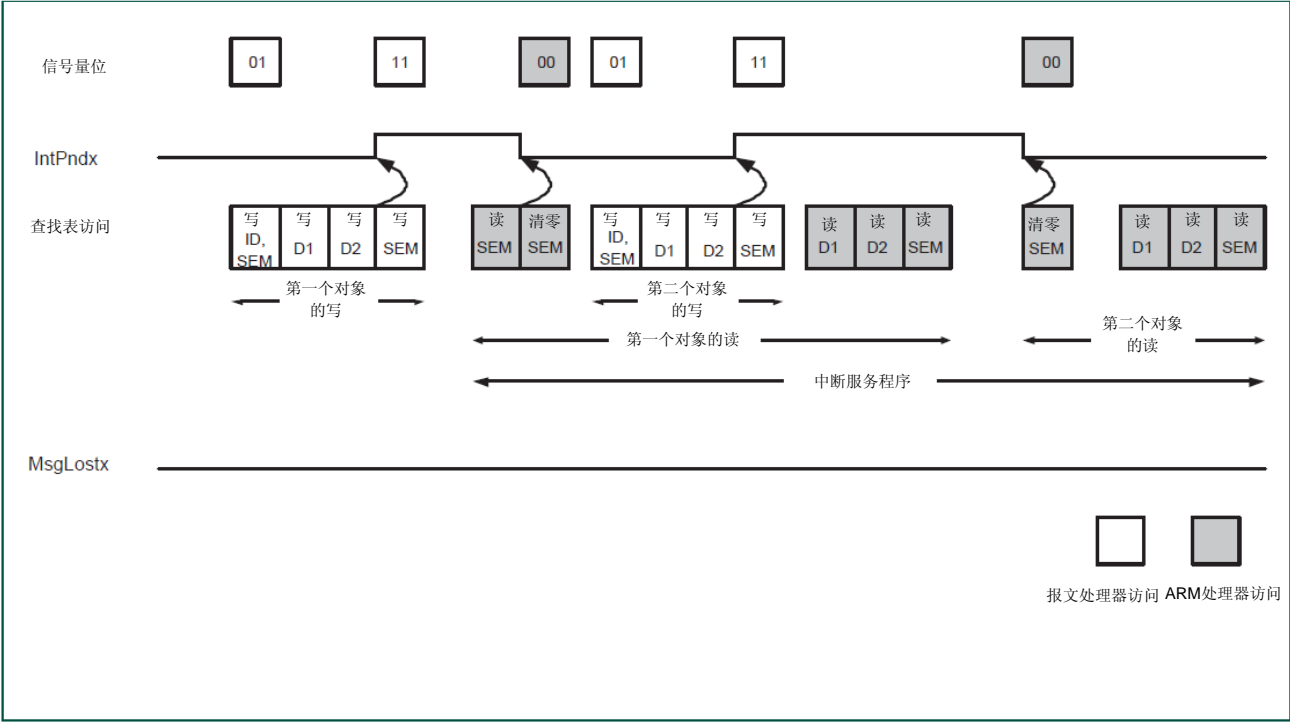
图90. 报文丢失



20.16.3.3 状况 3： 报文被改写，由信号量位来指示

该情况是一个特例，在这种情况下，丢失报文由现有的信号量位来指示。 如果在软件读一个报文对象的过程中另一个新的报文被报文处理程序保存，那么就进入这种情况。 此时，FullCAN 中断位再次被置位，第二个对象被写入。

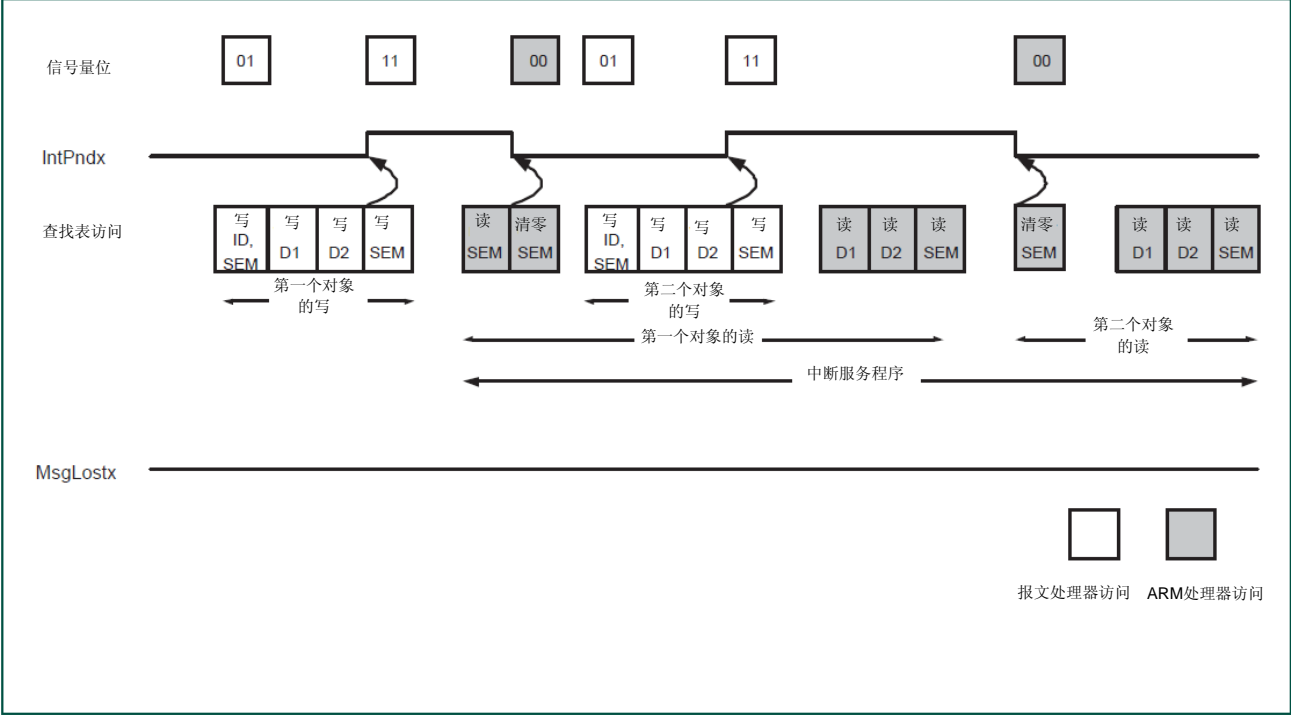
图91. 报文被改写



20.16.3.4 状况 3.1： 报文被改写，由信号量位和报文丢失来指示

这种情况属于状况 3 下面的一种：丢失的报文由现有的信号量位和报文丢失来指示。

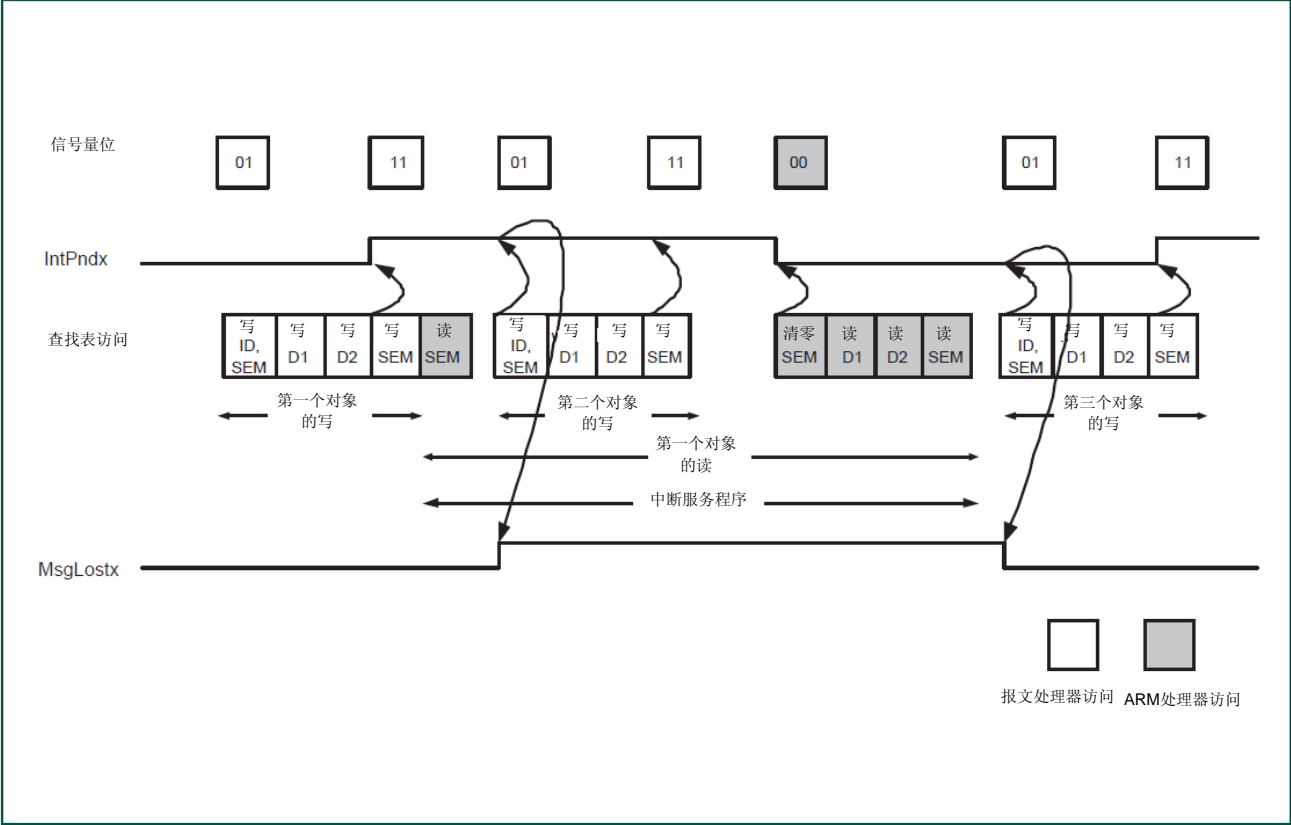
图92. 报文的改写由信号量位和报文丢失来指示



20.16.3.5 状况 3.2： 报文被改写，由报文丢失来指示

这种情况属于状况 3 下面的一种：丢失的报文由报文丢失来指示。

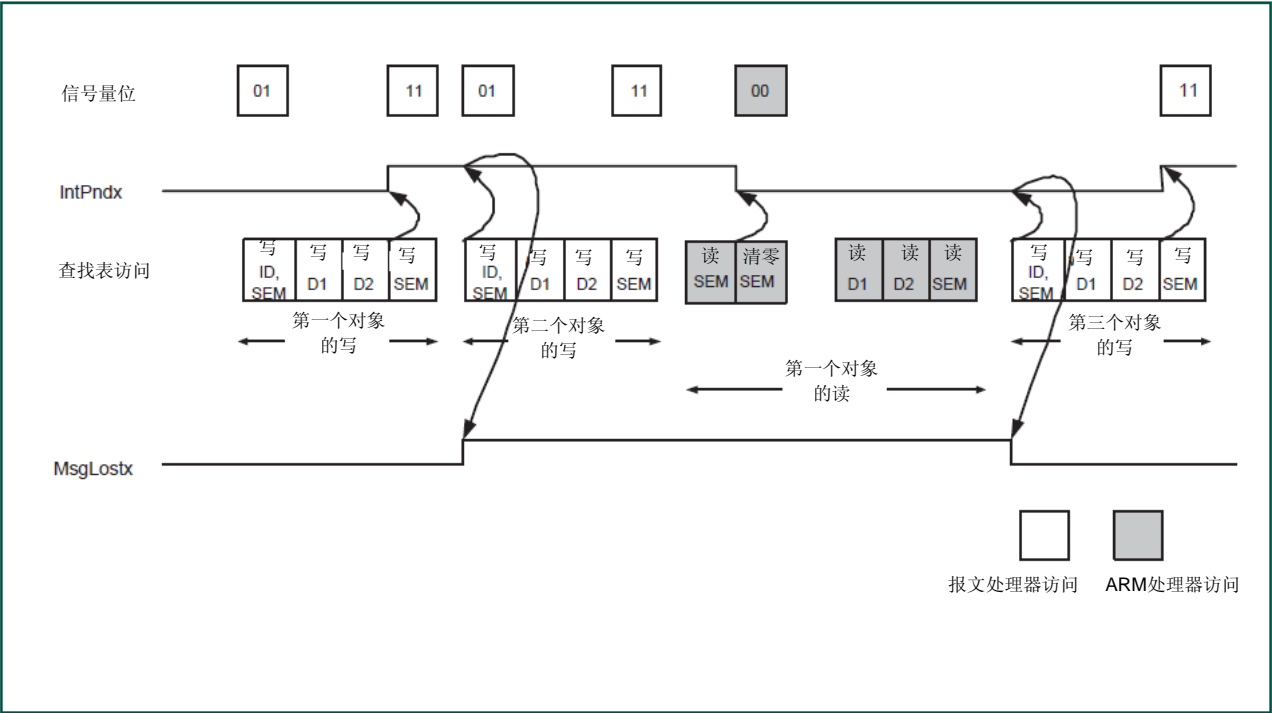
图93. 报文的改写由报文丢失来指示



20.16.3.6 状况 4： 清零报文丢失位

这种情况是一种特例，在这种情况下，一个对象的报文丢失位在一个没有被读取的报文对象被改写的过程中（第二个对象的写）被置位。 接下来软件将对该对象读出时（读第一个对象）清除挂起的中断。写入第三个对象时清零报文丢失位。如果对象的挂起位未被置位，则每次“写 ID, SEM”都会清零报文丢失位。

图94. 清零报文丢失位



20.17 验收滤波器表格和 ID 索引值示例

20.17.1 示例 1： 仅使用一个区

```
SFF_sa      <      ENDofTable      OR
SFF_GRP_sa  <      ENDofTable      OR
EFF_sa      <      ENDofTable      OR
EFF_GRP_sa  <      ENDofTable
```

一个区起始地址低于所有已编程的 CAN 标识符的终止地址。

20.17.2 示例 2： 所有的区都被使用

```
SFF_sa      <      SFF_GRP_sa      AND
SFF_GRP_sa  <      EFF_sa          AND
EFF_sa      <      EFF_GRP_sa      AND
EFF_GRP_sa  <      ENDofTable
```

如果有一个区未被使用，则起始地址必须设置为下一个区的起始地址值。

20.17.3 示例 3： 多个区（但并非所有区）被使用

如果未使用 SFF 组，那么 SFF 组区的起始地址（SFF_GRP_sa 寄存器）必须设置成下一个区的起始地址，在这种情况下，它是 Explicit SFF 区的起始地址（SFF_sa 寄存器）。

在明确的标识符和标识符组都被设定的情况下，CAN 标识符查找必须从明确的标识符区开始。如果在这个区中没有找到合适的匹配，继续查找标识符组区。通使用这种顺序来查找可以确保在找到一个明确标识符的匹配时，随后的软件可以直接对报文进行处理，但如果找到的是标识符组的匹配时，随后的软件需要更多的操作来识别报文。

20.17.4 配置示例 4

假设五个验收滤波器地址寄存器包含的值在下表的第三列中给出。表格的第四和第五列分别是字和行的编号（十进制表示）。如果 CAN 报文的标识符与表格中的某一行相匹配，CANRFS 寄存器的 ID 索引字段将返回 ID 索引列中所给出的值。

表478. 验收滤波器表格和 ID 索引值的示例

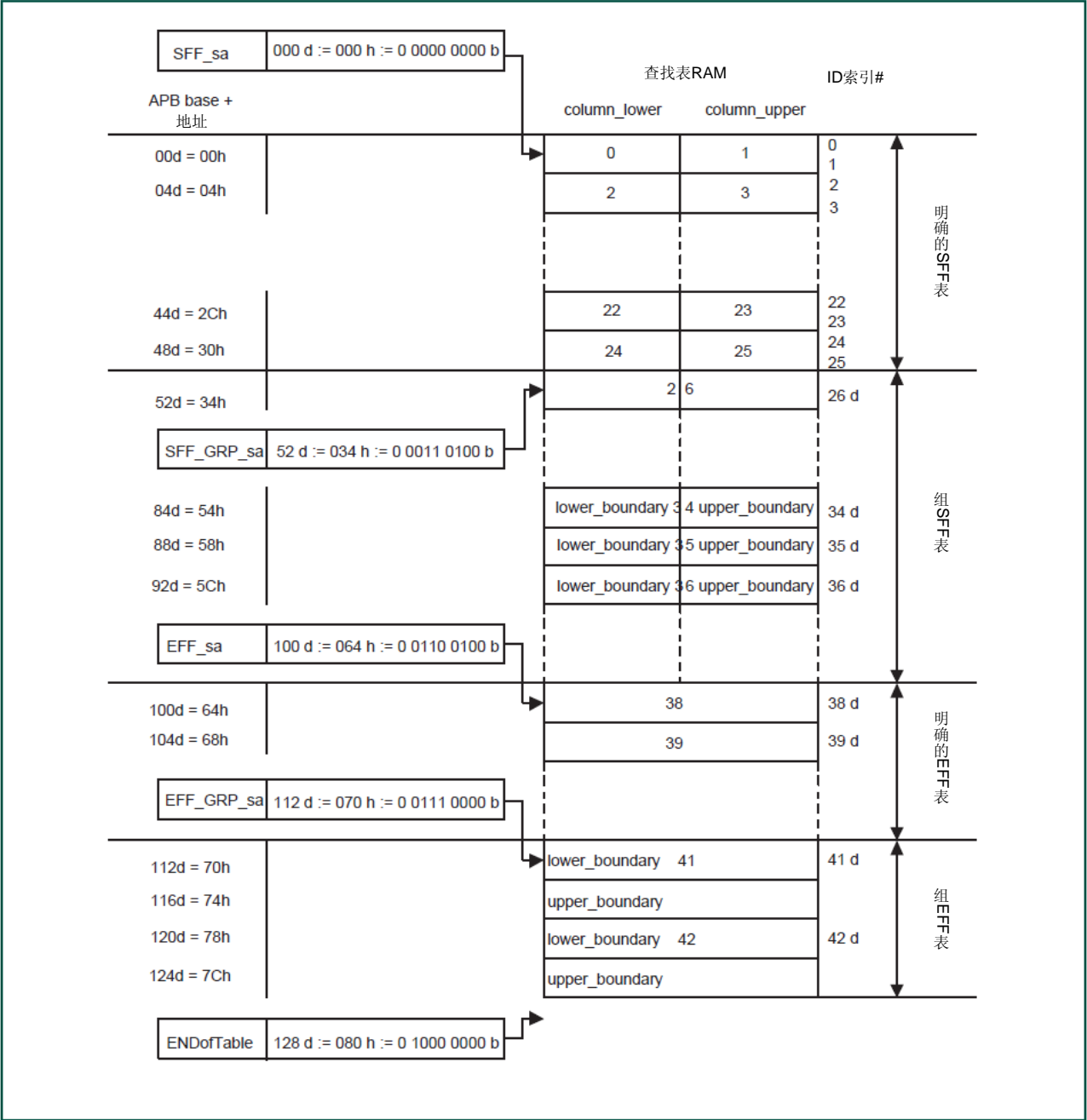
表	寄存器	值	#字	#行	ID 索引
标准单个标识符	SFF_sa	0x040	810	1610	0-1510
标准标识符组	SFF_GRP_sa	0x060	410	410	16-1910
扩展单个标识符	EFF_sa	0x070	810	1610	20-5510
扩展标识符组	EFF_GRP_sa	0x100	810	1610	56-5710
	ENDofTable	0x110			

20.17.5 配置示例 5

[图 95](#) 是地址寄存器、表格分布和 ID 索引值的详细图例。 它列出了：

- 标准单个标识符表格，位于验收滤波器 RAM 起始位置，包含 26 个标识符；后接
- 标准标识符组表格，包含 12 组标识符；后接
- 扩展单个标识符表格，包含 3 个标识符；后接
- 扩展标识符组表格，包含 2 组标识符。

图95. 验收滤波器表格和 ID 索引值的详细示例



20.17.6 配置示例 6

下表列出了被使用和激活的区和 CAN 标识符类型。 该这个例子的 ID 查找表配置如[图 96](#)所示。

表479. 使用的 ID 查找表区

ID 查找表区	状态
FullCAN	未激活
明确的标准帧格式	激活
标准帧组格式	激活
明确的扩展帧格式	激活
扩展帧组格式	激活

明确的标准帧格式标识符区（11 位 CAN ID）:

明确标准帧格式区的起始地址在 SFF_sa 寄存器中定义（值为 0x00）。 这个区的结束用 SFF_GRP_sa 寄存器来定义。在 ID 查找表的明确标准帧格式区中,带有源 CAN 通道(SCC)的两个 CAN 标识符共用一个 32 位的字。 不使用或禁能的 CAN 标识符可以通过置位报文禁能位来标识。

标准帧组格式标识符区（11 位 CAN ID）:

标准帧组格式区的起始地址在 SFF_GRP_sa 寄存器中定义（值为 0x10）。 这个区的结束用 EFF_sa 寄存器来定义。 在标准帧组格式区中，具有相同源 CAN 通道（SCC）的两个 CAN 标识符共用一个 32 位的字，并指示待接收的 CAN 标识符范围。 位 31~16 代表 CAN 标识符范围的低边界，而 15~0 则代表 CAN 标识符范围的高边界。
在这个范围内的所有标识符（包括边界标识符）都将被接收。 验收滤波器通过置位高和低边界标识符的报文禁能位来禁能和停止使用整个组的标识符。 为了存放四组标准帧格式标识符，EFF_sa 寄存器的值被设置为 0x20。 这个区的索引值为 9 的标识符组不使用，因此被禁能。

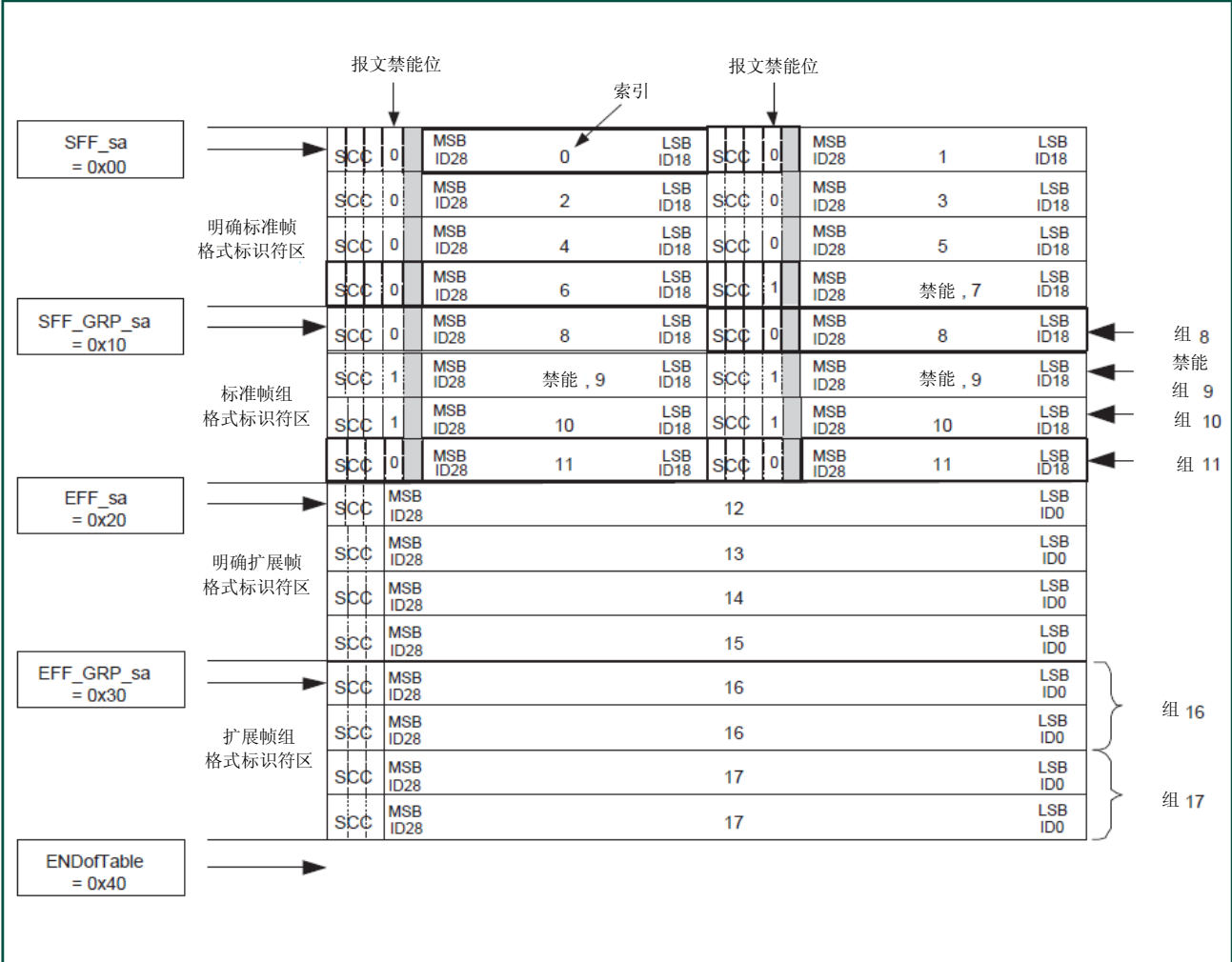
明确的扩展帧格式标识符区（29 位 CAN ID）（见[图 96](#)）

明确的扩展帧格式区的起始地址在 SFF_sa 寄存器中定义（值为 0x20）。 这个区的结束用 EFF_GRP_sa 寄存器来定义。在明确的扩展帧格式区中,每根地址线只有一个带有源 CAN 通道（SCC）的 CAN 标识符被设定。 为了给四个明确扩展帧格式标识符提供存储空间，EFF_GRP_sa 寄存器的值被设置为 0x30。

扩展帧组格式标识符区（29 位 CAN ID）（见[图 96](#)）

扩展帧组格式区的起始地址在 SFF_GRP_sa 寄存器中定义（值为 0x30）。 这个区的结束用表格结束地址寄存器（ENDofTable）来定义。 在扩展帧组格式区中，边界被设定为一对地址线，第一条地址线代表低边界，第二条地址线代表高边界。 为了给两组扩展帧组格式标识符提供存储空间，ENDofTable 寄存器的值被设置为 0x40。

图96. ID 查找表配置示例（无 FullCAN）



20.17.7 配置示例 7

下表列出了被使用和激活的区和 CAN 标识符类型。 这个例子的 ID 查找表配置如图 97 所示。

这个例子使用一个典型的配置，在该配置中定义了 FullCAN 和明确的标准帧格式报文。 正如 20.15.1 节中描述的一样，验收滤波器会按特定顺序执行。 由于 FullCAN 区使能，验收滤波器的标识符筛选过程就首先从 FullCAN 区开始，然后再继续处理剩下的已使能的区。

表480. 使用的 ID 查找表区

ID 查找表区	状态
FullCAN	激活和使能
明确的标准帧格式	激活
标准帧组格式	未激活
明确的扩展帧格式	未激活
扩展帧组格式	未激活

FullCAN 明确的标准帧格式标识符区（11 位 CAN ID）

FullCAN 明确的标准帧格式标识符区的起始地址（自动）设置为 0x00。 这个区的结束用 SFF_sa 寄存器来定义。 在 FullCAN ID 区中，只有 FullCAN 对象的标识符保存下来进行验收过滤。 在这个区中，两个带有源 CAN 通道（SCC）的 CAN 标识符共用一个 32 位的字。 不使用或禁能的 CAN 标识符可以通过置位报文禁能位来标识。 每个已定义的标识符的 FullCAN 对象可以在 FullCAN 报文对象区中找到。 如果在验收滤波器过程中找到一个标识符匹配，接收到的 FullCAN 报文对象数据就从相应的 CAN 控制器的接收缓冲器转移到 FullCAN 报文对象区中。 为了给八个 FullCAN、明确的标准帧格式标识符提供存储空间，SFF_sa 寄存器的值被设置为 0x10。 这个区中索引值为 1 的标识符不使用，因此被禁能。

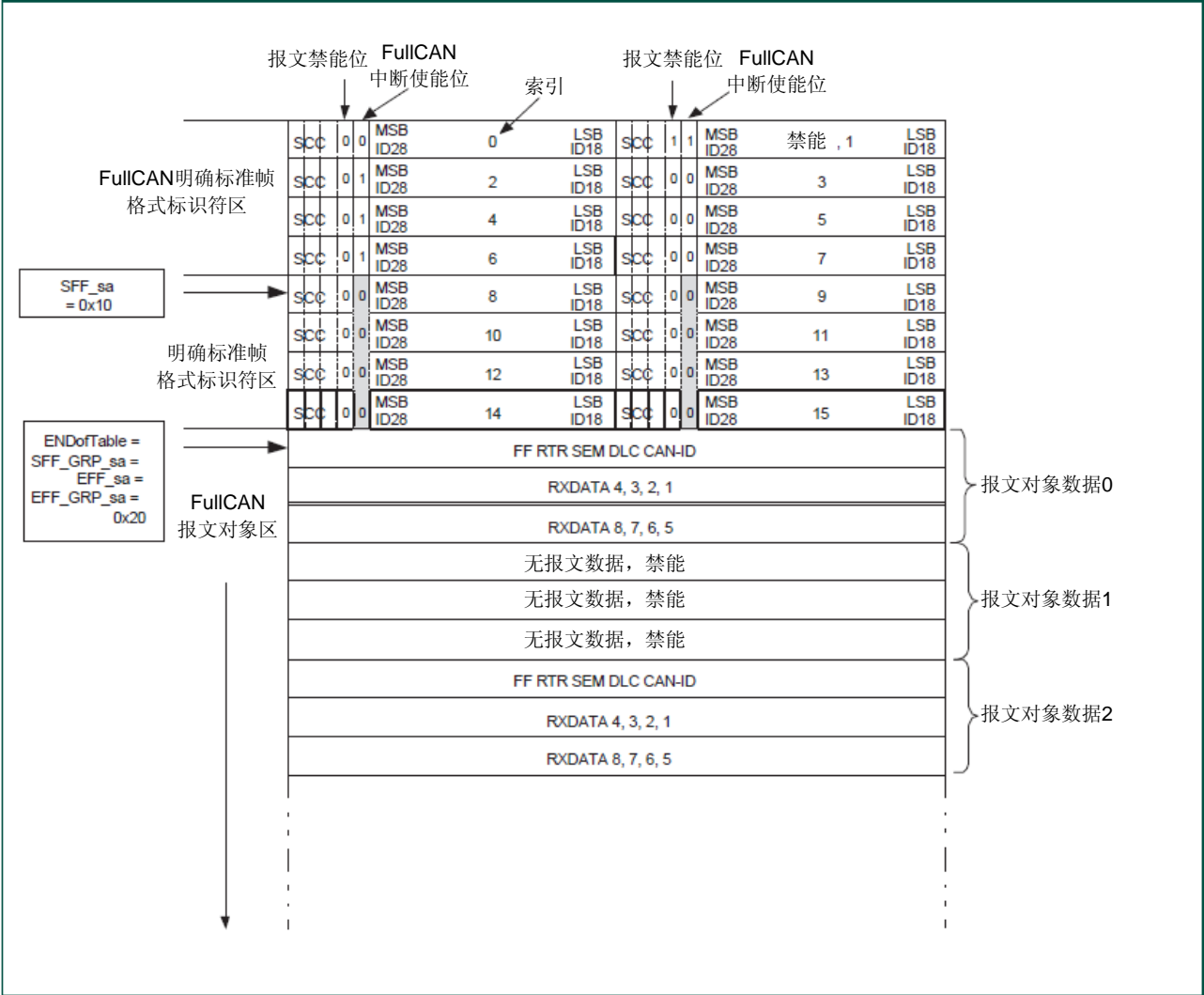
明确的标准帧格式标识符区（11 位 CAN ID）

明确的标准帧格式区的起始地址在 SFF_sa 寄存器中定义（值为 0x10）。 这个区的结束用表格结束地址寄存器（ENDofTable）来定义。 在 ID 查找表的明确标准帧格式区中，带有源 CAN 通道（SCC）的两个 CAN 标识符共用一个 32 位的字。 不使用或禁能的 CAN 标识符可以通过置位报文禁能位来标识。 为了给八个明确的标准帧格式标识符提供存储空间，ENDofTable 寄存器的值被设置为 0x20。

FullCAN 报文对象数据区

FullCAN 报文对象数据区的起始地址用 ENDofTable 寄存器来定义。 使能的 FullCAN 标识符的数量受到 FullCAN 报文对象数据区的可用存储空间的限制。 每个定义的 FullCAN 报文需要三根地址线，供给 FullCAN 报文对象数据区中的报文数据使用。 FullCAN 报文对象区按照下面的方法来组织：FullCAN 标识符区的每个索引编号与 FullCAN 报文对象区中的一个报文对象编号相对应。

图97. ID 查找表配置示例（FullCAN 被激活并使能）



20.17.8 查找表编程准则

ID 查找表的所有标识符区都必须按照下面的方式来编程：每个有效的区组成一个排好序的列表或表格，在每个区中源 CAN 通道（SCC）和 CAN 标识符都按照升序排列。

SCC 值和 CAN 控制器-1 相等，即，SCC=0 和 CAN1 匹配，而 SCC=1 和 CAN2 匹配。

在 ID 查找表中遇到语法错误时，错误线的查找表地址在查找表错误地址寄存器（LUTerrAd）中变得可用。

查找表错误地址寄存器（LUTerrAd）的报告过程是一个“运行时”流程。仅报告存在语法错误的地址线，它们在整个验收过程中被传递。

设定查找表要遵循下列常用规则：

- 每个区组织成一个排好序的列表或表格，在每个区中源 CAN 通道（SCC）和 CAN 标识符都按照升序排列（被禁能的标识符也不例外）。
- 标识符组定义中的高边界和低边界都必须来自同一个源 CAN 通道。
- 如果要禁能一组标识符，高边界和低边界的报文禁能位都必须被置位。

21.1 基本配置

使用下列寄存器来配置 SSP 接口（SSP0、SSP1 和 SSP2）：

1. 功率：在 PCONP 寄存器（[表 37](#)）中，置位 PCSSP0 可使能 SSP0，置位 PCSSP1 可使能 SSP1。
注：复位时，SSP0 和 SSP1 会被使能（PCSSP0/1 = 1），而 SSP2 会被禁用（PCSSP2 = 0）。
2. 外设时钟：SSP 使用公共 PCLK，它既用于总线接口，也用于大多数 APB 外设的功能部分。参见 [4.6.4](#) 节。在主机模式下，必须对时钟进行分频（参见 [21.6.5](#) 节）。
3. 管脚：通过相关的 IOCON 寄存器（[8.4.1](#) 节）选择 SSP 管脚和管脚模式。
4. 中断：SSP0 中断是通过 SSP0IMSC 寄存器来使能的，而 SSP1 中断时通过 SSP1IMSC 寄存器来使能的（[表 488](#)）。利用相应的中断置位使能寄存器来使能 NVIC 中的中断，参见 [表 43](#)。
5. 初始化：有两个控制寄存器可用于每个待配置的 SSP 端口：SSP0CR0 和 SSP0CR1 用于 SSP0，SSP1CR0 和 SSP1CR1 用于 SSP1，SSP2CR0 和 SSP2CR1 用于 SSP2。参见 [21.6.1](#) 节和 [21.6.2](#) 节。
6. DMA：SSP 接口的发送和接收 FIFO 可以连接到 GPDMA 控制器（参见 [21.6.10](#) 节）。
7. 有关 GPDMA 系统的连接，参见 [表 664](#)。

21.2 特性

- 兼容 Motorola SPI、4 线 TI SSI 和 National 半导体的 Microwire 总线。
- 同步串行通信。
- 可选择主机操作或从机操作。
- 8 帧收发 FIFO。
- 4~16 位数据帧。
- GPDMA 支持的 DMA 传输。

21.3 描述

SSP 是一个同步串行端口 (SSP) 控制器, 可控制 SPI、4 线 SSI 或 Microwire 总线的操作。SSP 可以与总线上的多个主机或从机交互。但在一次给定的数据传输过程中, 总线上只能有一个主机和一个从机进行通信。数据传输原则上是全双工模式的, 4~16 位数据的帧由主机发送到从机或由从机发送到主机。但实际上, 在大多数情况下, 只有一个方向上的数据流包含有意义的数据。

LPC178x/177x 系列有三个同步串行端口控制器—SSP0、SSP1 和 SSP2。

21.4 管脚描述

表481. SSP 管脚描述

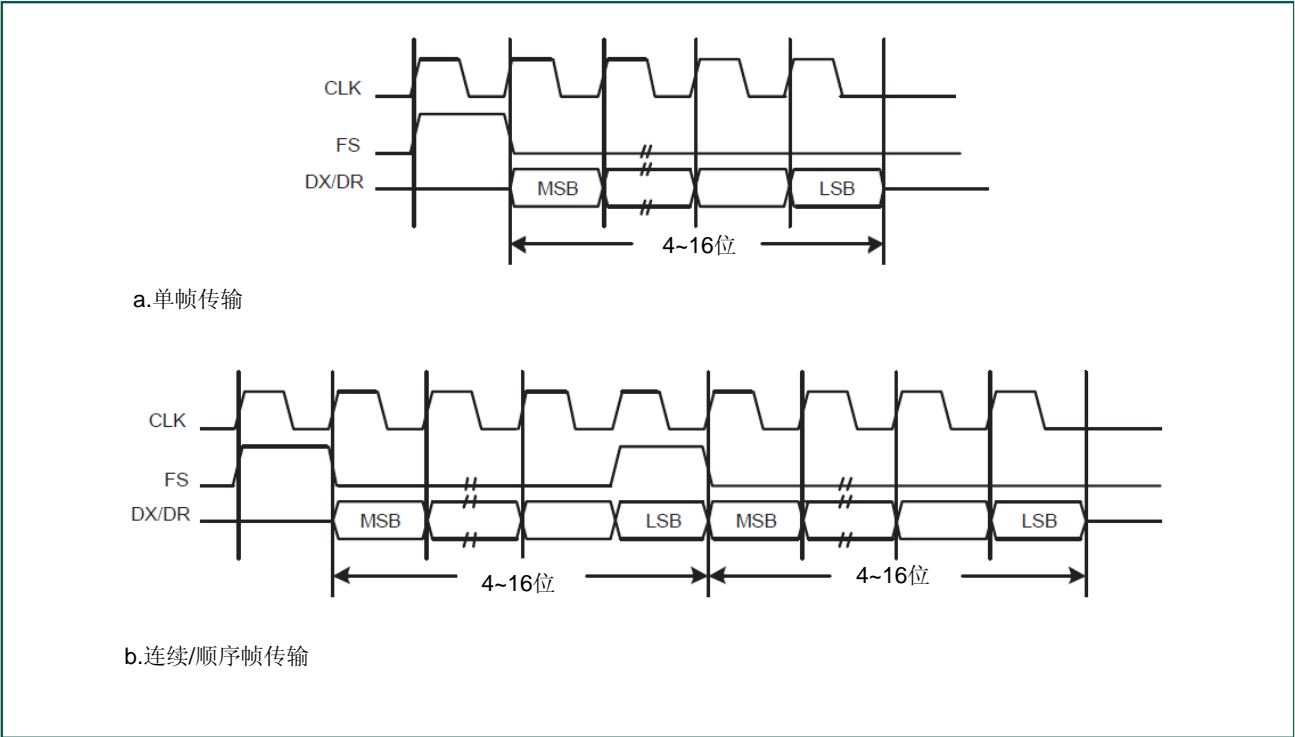
管脚名称	类型	接口管脚名称/功能			管脚描述
		SPI	SSI	Microwire	
SCK0/1/2	I/O	SCK	CLK	SK	串行时钟。 SCK/CLK/SK 是用来同步数据传输的时钟信号。它由主机驱动, 从机接收。当使用 SPI 接口时, 时钟可编程为高有效或低有效, 否则, 时钟总是高有效。 SCK 的状态只能在数据传输过程中改变。在其它时间里, SSPn 接口使其保持无效状态或不驱动它 (使其处于高阻态)。
SSEL0/1/2	I/O	SSEL	FS	CS	帧同步/从机选择。 当 SSPn 接口是总线主机时, 它在串行数据启动前驱动该信号至有效状态, 然后在串行数据传输终止后释放该信号至无效状态。该信号的有效状态可以为高有效或低有效, 取决于所选的总线和模式。当 SSPn 是总线从机时, 该信号根据使用的通信协议来指示主机的数据的存在状况。 当只有一个总线主机和一个总线从机时, 主机的帧同步或从机选择信号直接与从机相应的输入相连。当总线上接有多个从机时, 需要管理好这些从机的帧选择/从机选择输入, 以免一次传输有多个从机响应。
MISO0/1/2	I/O	MISO	DR(M) DX(S)	SI(M) SO(S)	主机输入从机输出。 MISO 信号使串行数据从从机传输到主机。当 SSPn 是从机时, 串行数据从该信号输出。当 SSPn 是主机时, 串行数据从该信号输入。当 SSPn 是从机且未被 FS/SSEL 选择时, 它将不驱动该信号 (使其处于高阻态)。
MOSI0/1/2	I/O	MOSI	DX(M) DR(S)	SO(M) SI(S)	主机输出从机输入。 MOSI 信号使串行数据从主机传输到从机。当 SSPn 是主机时, 串行数据从该信号输出。当 SSPn 是从机时, 串行数据从该信号输入。

21.5 总线描述

21.5.1 TI 同步串行数据帧格式

图 98 显示了 SSP 模块支持的 4 线 Texas Instruments (TI) 同步串行数据帧的格式。

图98. TI 同步串行数据帧的格式：a) 单帧传输 b) 连续/顺序 2 帧传输



对于在该模式下配置为主机的设备，CLK 和 FS 会被强制为低电平，并且只要 SSP 空闲，发送数据线 DX 就会为三态。发送 FIFO 的底部入口一旦有数据装入，FS 立即变为高电平，并维持一个 CLK 周期的时间。要发送的值也会从发送 FIFO 传递到发送逻辑的串行移位寄存器。在下一个 CLK 上升沿，4~16 位数据帧的 MSB 会被输出到 DX 管脚。同样，接收数据的 MSB 由片外串行从设备输入到 DR 管脚。

在每个 CLK 的下降沿，SSP 和片外串行从机设备将每一数据位送入相应的串行移位寄存器。在 LSB 被锁存后，接收到的数据在 CLK 的首个上升沿从串行移位寄存器传递到接收 FIFO。

21.5.2 SPI 帧格式

SPI 接口是一个 4 线接口，它的 SSEL 信号用作从机选择。SPI 格式的主要特性是 SCK 信号的无效状态和相位可通过 SSPCR0 控制寄存器的 CPOL 位和 CPHA 位来编程设定。

21.5.2.1 时钟极性 (CPOL) 和相位 (CPHA) 控制

当 CPOL 时钟极性控制位为 0 时，它会在 SCK 管脚上产生一个稳定的低电平值。如果 CPOL

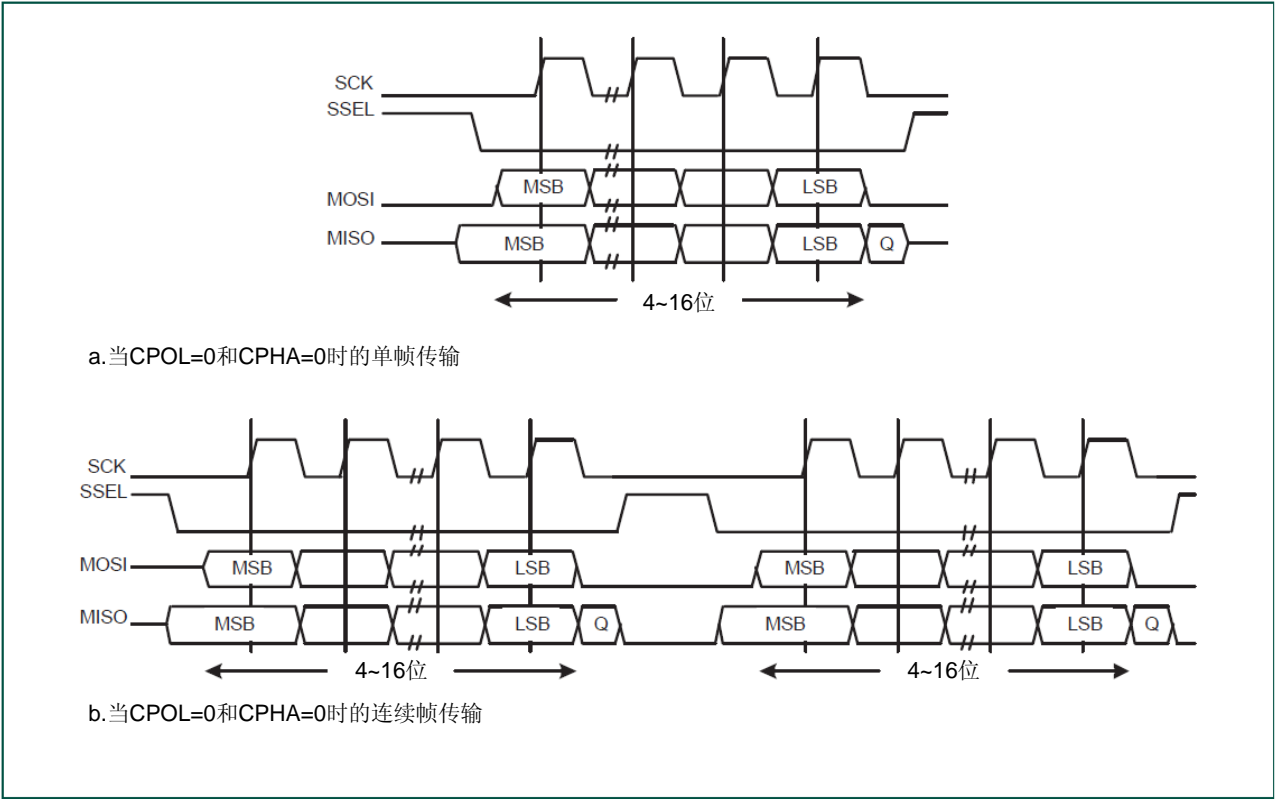
时钟极性控制位为 1，那么当没有数据传输时，它会在 CLK 管脚上产生一个稳定的高电平值。

CPHA 控制位用来选择捕获数据以及允许数据改变状态的时钟沿。通过在第一个数据捕获沿之前允许或不允许时钟变换，从而在第一个被发送的位上产生极大的影响。当 CPHA 相位控制位为 0 时，在时钟沿出现第一次变换时捕获数据。如果 CPHA 时钟相位控制位为 1，则在时钟沿出现第二次变换时捕获数据。

21.5.2.2 CPOL=0, CPHA=0 时的 SPI 格式

CPOL = 0, CPHA = 0 时，SPI 格式的单帧和连续帧传输信号序列如图 99 所示。

图99. CPOL=0 和 CPHA=0 时的 SPI 帧格式 (a) 单帧传输和 b) 连续帧传输)



这种配置在空闲期间：

- CLK 信号强制为低电平。
- SSEL 强制为高电平。
- 发送 MOSI/MISO 焊盘处于高阻态。

如果 SSP 被使能且发送 FIFO 中装载了有效数据，则 SSEL 主信号驱动为低电平，指示数据发送开始。这就使从机的数据被使能并出现在主机的 MISO 输入线上。主机的 MOSI 管脚也被使能。

1/2 个 SCK 周期后，有效的主机数据会传输到 MOSI 管脚。由于主机和从机数据都已设置，因此再过 1/2 个 SCK 周期，SCK 主时钟管脚变为高电平。

此时，数据在 SCK 信号的上升沿被捕获，并保持到 SCK 的下降沿。

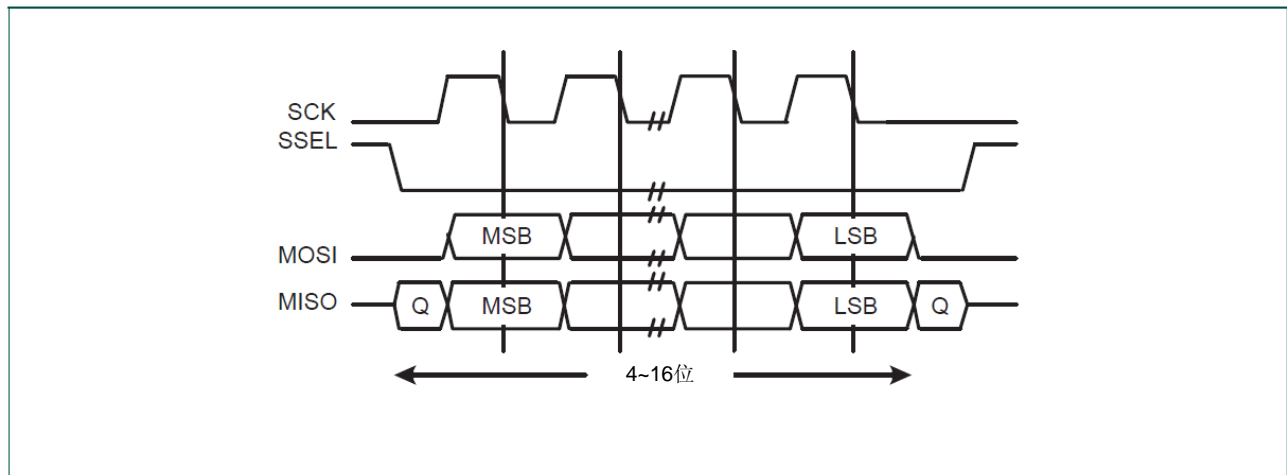
在发送单个字时，当数据字的所有位传输完后，最后一个位被捕获的一个 SCK 周期后，SSEL 线返回到空闲高电平状态。

但是，在连续帧的顺序传输过程中，每个数据字传输之间 SSEL 信号必须为高电平。这是因为当 CPHA 位为逻辑 0 时，从机选择管脚冻结了串行外设寄存器中的数据，不允许改变。因此，在每次数据传输之间主机设备必须拉高从机设备的 SSEL 管脚来使能串行外设数据的写操作。当连续帧传输结束时，最后一位被捕获一个 SCK 周期后，SSEL 管脚返回到空闲状态。

21.5.2.3 CPOL=0, CPHA=1 时的 SPI 格式

CPOL = 0, CPHA = 1 时，SPI 格式的传输信号序列如图 100 所示，它包括了单帧传输和连续帧传输两种方式。

图100. CPOL=0 和 CPHA=1 时的 SPI 帧格式



这种配置在空闲期间：

- CLK 信号强制为低电平。
- SSEL 强制为高电平。
- 发送 MOSI/MISO 焊盘处于高阻态。

如果 SSP 使能且发送 FIFO 中包含有效数据，则 SSEL 主信号驱动为低电平，指示数据发送开始。主机的主 MOSI 管脚也会被使能。再过 1/2 个 SCK 周期，主机和从机的有效数据都被使能输出到各自的发送线上。同时，出现一个上升沿变换时 SCK 被使能。

之后，数据在 SCK 信号的下降沿被捕获，并保持到 SCK 的上升沿。

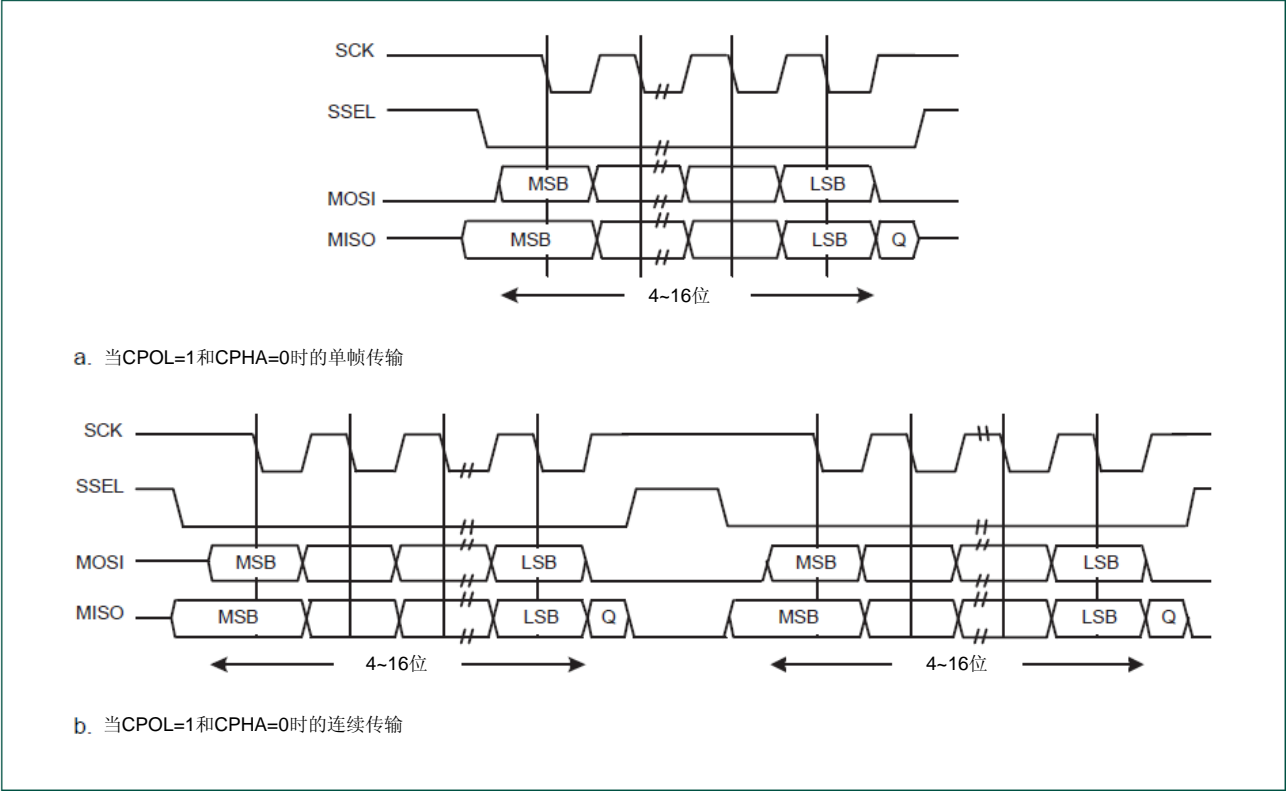
在传输单个字时，当所有位传输结束后，最后一位被捕获后一个 SCK 周期内，SSEL 线返回到空闲高电平状态。

在连续帧的顺序传输过程中，SSEL 管脚在两个连续的数据字传输之间保持低电平，终止传输的方法与单个字传输时相同。

21.5.2.4 CPOL = 1, CPHA = 0 时的 SPI 格式

CPOL = 1, CPHA = 0 时，SPI 格式的单帧和连续帧传输信号序列如图 101 所示。

图101.CPOL=1 和 CPHA=0 时的 SPI 帧格式 (a) 单帧传输 (b) 连续帧传输



这种配置在空闲期间：

- CLK 信号强制为高电平。
- SSEL 强制为高电平。
- 发送 MOSI/MISO 焊盘处于高阻态。

如果 SSP 使能且发送 FIFO 中包含有效数据，则 SSEL 主机信号驱动为低电平，指示数据发送开始，这使得从机数据立即传输到主机的主 MISO 线上。主机的主 MOSI 管脚也被使能。

1/2 个 SCK 周期后，有效的主机数据被传输到 MOSI 线。由于主机和从机数据都已设置，因此再过 1/2 个 SCK 周期后，SCK 主时钟管脚将变为低电平。这意味着，数据在 SCK 信号的下降沿被捕获，并保持到 SCK 的上升沿。

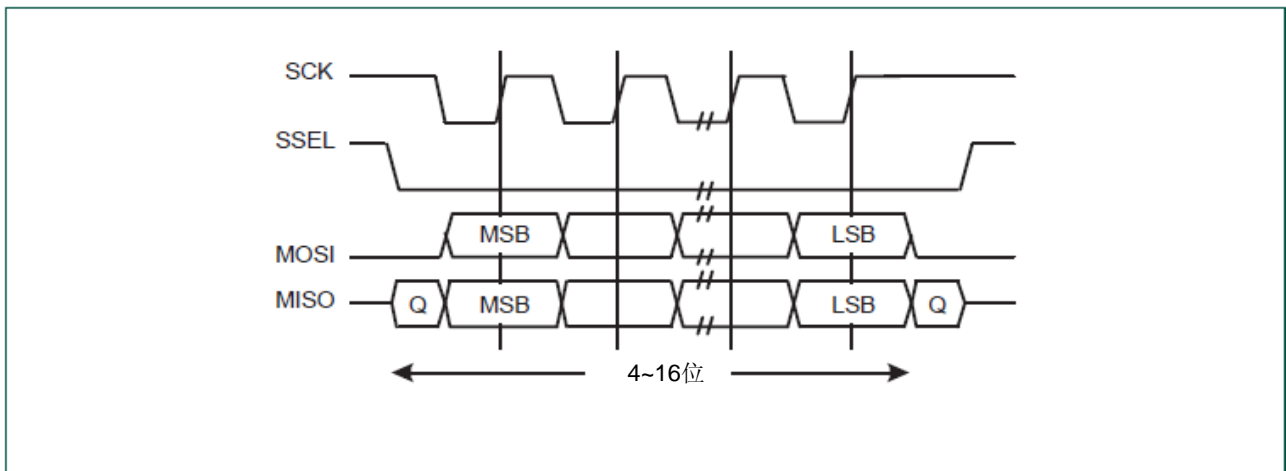
在发送单个字时，当数据字的所有位都传输完，最后一位被捕获的一个 SCK 周期后，SSEL 线返回到空闲高电平状态。

但是，在连续帧的顺序发送过程中，每个数据字传输之间的 SSEL 信号必须为高电平。这是因为当 CPHA 位为逻辑 0 时，从机选择管脚冻结了串行外设寄存器中的数据，不允许改变。因此，在每次数据传输之间主机设备必须拉高从机设备的 SSEL 管脚来使能串行外设数据的写操作。连续传输完成后，最后一位被捕获一个 SCK 周期后，SSEL 管脚返回到空闲状态。

21.5.2.5 CPOL = 1, CPHA = 1 时的 SPI 格式

CPOL = 1, CPHA = 1 时，SPI 格式的传输信号序列如[图 102](#)所示，它包含单帧传输和连续帧传输两种方式。

图102. CPOL=1 和 CPHA=1 时的 SPI 帧格式



这种配置在空闲期间：

- CLK 信号强制为高电平。
- SSEL 强制为高电平。
- 发送 MOSI/MISO 焊盘处于高阻态。

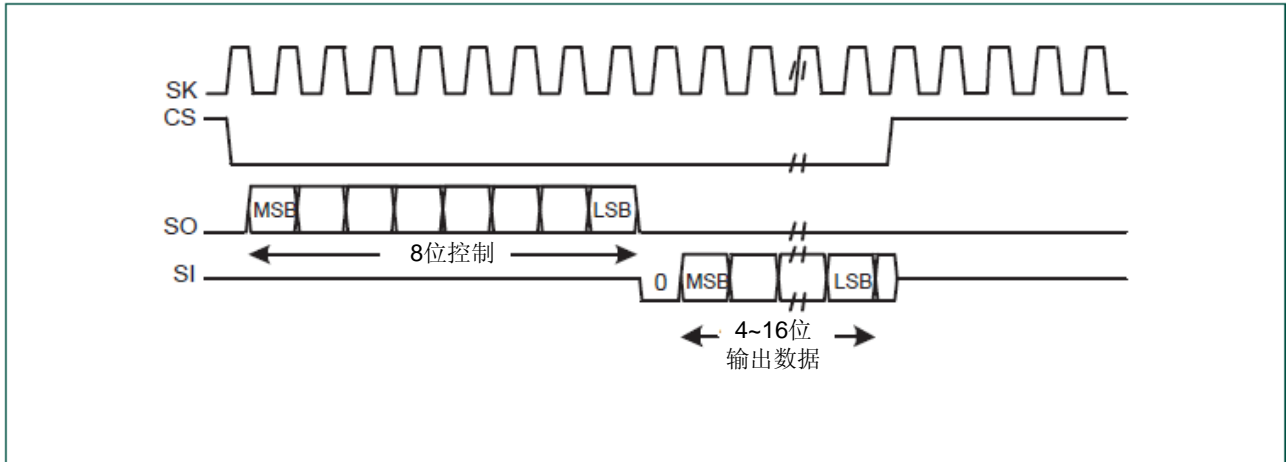
如果 SSP 使能且发送 FIFO 中包含有效数据，则 SSEL 主机信号被驱动为低电平，指示数据发送开始。主机的 MOSI 也被使能。再过 1/2 个 SCK 周期，主机和从机的数据都被使能输出到各自的发送线上。同时，出现下降沿变换时 SCK 被使能。之后，数据会在 SCK 信号的上升沿被捕获，并保持到 SCK 的下降沿。

在传输单个字时，当所有位传输结束，最后一位被捕获一个 SCK 周期后，SSEL 线返回到空闲高电平状态。对于连续帧的顺序发送，SSEL 管脚保持有效的低电平状态，直到最后一个字的最后一位捕获，它再返回到如上所述的空闲状态。总的来说，在连续帧的顺序传输过程中，SSEL 管脚在两个连续的数据字传输之间保持低电平，终止传输的方法与单个字传输相同。

21.5.3 National 半导体 Microwire 帧格式

图 103 所示为 Microwire 帧格式的单帧传输。图 104 所示为 Microwire 帧格式连续帧传输。

图103. Microwire 帧格式（单帧传输）



Microwire 格式与 SPI 格式非常相似，但它是采用半双工模式而非全双工模式发送，并使用主从消息传递技术。每次串行发送以 SSP 发送到片外从机设备的一个 8 位控制字开始。在发送控制字的过程中，SSP 不接收数据。控制字发送结束后，片外从机对其进行解码，在 8 位控制消息的最后一位发送结束的一个串行时钟后，才返回所需的数据。返回的数据长度为 4~16 位，使得总的帧长度在 13~25 位之间。

这种配置在空闲期间：

- SK 信号强制为低电平。
- CS 强制为高电平。
- 发送数据线 SO 可强制为低电平。

发送过程由写入到发送 FIFO 的一个控制字节来触发。CS 的下降沿使发送 FIFO 底部入口的数据发送到发送逻辑的串行移位寄存器，8 位控制帧的 MSB 移到 SO 管脚。CS 在帧发送过程中保持低电平。SI 管脚在发送过程中保持三态。

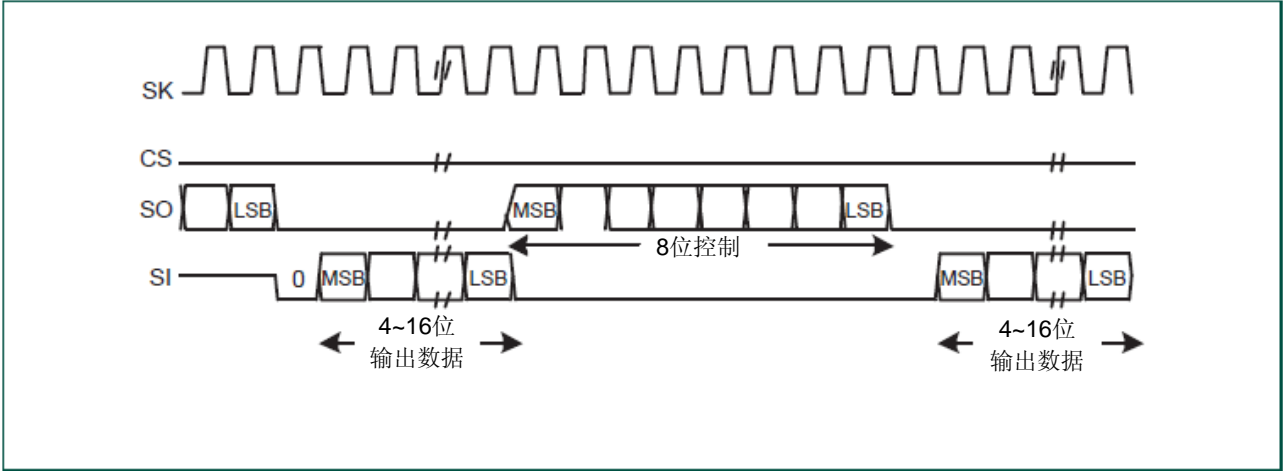
片外串行从机设备在每个 SK 的上升沿将每个控制位锁存到串行移位寄存器中。当从机设备完成最后一位的锁存后，再用一个时钟等待状态周期对控制字节进行解码，然后将数据发回给 SSP。每个位在 SK 的下降沿驱动到 SI 线上。SSP 则在 SK 的上升沿依次将各个位锁存。帧传输结束后，对于单帧传输，在最后一位被锁存到接收串行移位寄存器的一个时钟周期后，CS 信号被置为高电平，使数据传输到接收 FIFO。

注：LSB 被接收移位寄存器锁存后或当 CS 管脚变为高电平时，片外从机设备的接收线在 SK 下降沿都呈现三态。

对于连续传输，数据发送开始和结束的方法都与单帧传输相同。但不同的是在连续传输过程中 CS 线保持有效（保持低电平），且数据连续发送。当前数据帧的 LSB 被接收后，下一

帧控制字节立即发送。在一帧数据的 LSB 被锁存到 SSP 后，每个接收到的值在 SK 下降沿从接收移位寄存器中传递出去。

图104. Microwire 帧格式（连续传输）

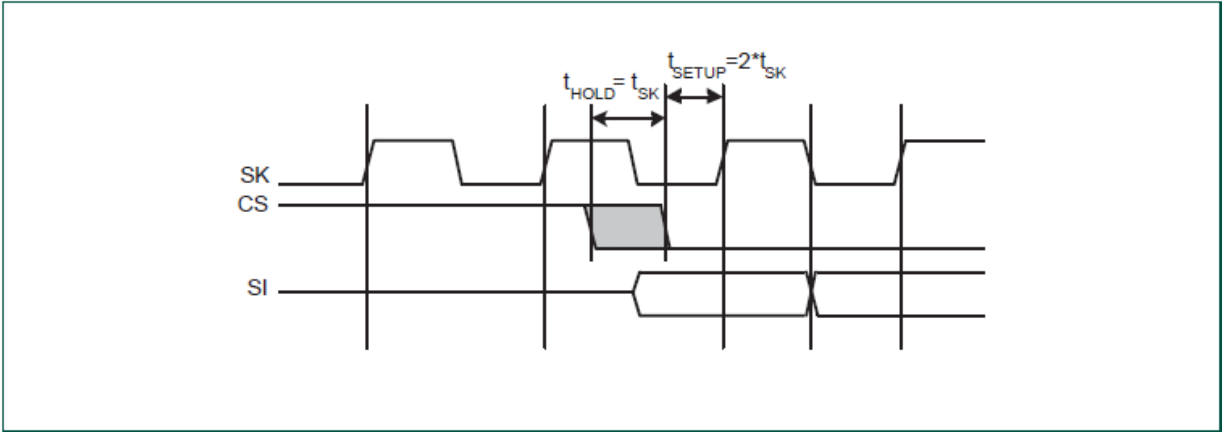


21.5.3.1 Microwire 模式下 CS 相对于 SK 的建立和保持时间要求

在 Microwire 模式下，CS 变为低电平后，SSP 从机在 SK 上升沿对接收数据的首位进行采样。因此，主机驱动 SK 自由运行时，必须确保 CS 信号相对 SK 上升沿有充足的建立和保持时间。

图 105 给出对 CS 建立和保持时间的要求。相对 SSP 从机采样接收数据首位的 SK 上升沿，CS 的建立时间必须至少为 SK（SSP 在 SK 上运行）周期的两倍。相对于之前的 SK 上升沿，CS 保持时间至少为一个 SK 周期。

图105. Microwire 帧格式的建立和保持时间



21.6 寄存器描述

SSP 控制器的寄存器地址如[表 482](#)所示。

表482. SSP 寄存器映射

通用名称	描述	访问	复位值 ^[1]	SSPn 寄存器名称和地址	表
CR0	控制寄存器 0。选择串行时钟速率、总线类型和数据长度。	R/W	0	SSP0CR0—0x4008 8000 SSP1CR0—0x4003 0000 SSP2CR0—0x400A C000	表 483
CR1	控制寄存器 1。选择主机/从机和其它模式。	R/W	0	SSP0CR1—0x4008 8004 SSP1CR1—0x4003 0004 SSP2CR1—0x400A 4004	表 484
DR	数据寄存器。写满发送 FIFO 和读空接收 FIFO。	R/W	0	SSP0DR—0x4008 8008 SSP1DR—0x4003 0008 SSP2DR—0x400A C008	表 485
SR	状态寄存器	RO		SSP0SR—0x4008 800C SSP1SR—0x4003 000C SSP2SR—0x400A C00C	表 486
CPSR	时钟预分频寄存器	R/W	0	SSP0CPSR—0x4008 8010 SSP1CPSR—0x4003 0010 SSP2CPSR—0x400A C010	表 487
IMSC	中断使能置位和清零寄存器	R/W	0	SSP0IMSC—0x4008 8014 SSP1IMSC—0x4003 0014 SSP2IMSC—0x400A C014	表 488
RIS	原始中断状态寄存器	R/W		SSP0RIS—0x4008 8018 SSP1RIS—0x4003 0018 SSP2RIS—0x400A C018	表 489
MIS	使能中断状态寄存器	R/W	0	SSP0MIS—0x4008 801C SSP1MIS—0x4003 001C SSP2MIS—0x400A C01C	表 490
ICR	SSPICR 中断清零寄存器	R/W	无	SSP0ICR—0x4008 8020 SSP1ICR—0x4003 0020 SSP2ICR—0x400A C020	表 491
DMACR	DMA 控制寄存器	R/W	0	SSP0DMACR—0x4008 8024 SSP1DMACR—0x4003 0024 SSP2DMACR—0x400A C024	表 492

[1] 复位值仅反映使用位中保存的数据，不包括保留位的内容。

21.6.1 SSPn 控制寄存器 0(SSP0CR0—0x4008 8000, SSP1CR0—0x4003 0000, SSP2CR0—0x400A C000)

该寄存器控制 SSP 控制器的基本操作。

表483. SSPn 控制寄存器 0 位描述 (SSP0CR0—地址 0x4008 8000, SSP1CR0—地址 0x4003 0000, SSP2CR0—0x400A C000)

位	符号	值	描述	复位值
3:0	DSS		数据长度选择。该字段控制着每帧传输的位数目。不支持且不使用值 0000-0010。	0000
		0011	4 位传输	
		0100	5 位传输	
		0101	6 位传输	
		0110	7 位传输	
		0111	8 位传输	
		1000	9 位传输	
		1001	10 位传输	
		1010	11 位传输	
		1011	12 位传输	
		1100	13 位传输	
		1101	14 位传输	
		1110	15 位传输	
		1111	16 位传输	
5:4	FRF		帧格式。	00
		00	SPI	
		01	TI	
		10	Microwire	
		11	不支持且不应使用这个组合。	
6	CPOL		时钟输出极性。该位只用于 SPI 模式。	0
		0	SSP 控制器使总线时钟在帧传输之间保持低电平。	
		1	SSP 控制器使总线时钟在帧传输之间保持高电平。	
7	CPHA		时钟输出相位。该位只用于 SPI 模式。	0
		0	SSP 控制器在帧传输的第一个时钟跳变沿捕获串行数据，即离开时钟线的帧间状态。	
		1	SSP 控制器在帧传输的第二个时钟跳变沿捕获串行数据，即回到时钟线的帧间状态。	
15:8	SCR		串行时钟频率。SCR 的值为总线上传输的每个数据位对应的预分频器输出时钟数减 1。假设 CPSDVSR 为预分频器分频值，APB 时钟 PCLK 为预分频器的时钟，则位速率为 PCLK/ (CPSDVSR×[SCR+1])。	0
31:8	-		保留。读取值未定义，只写入 0。	NA

21.6.2 SSPn 控制器寄存器 1 (SSP0CR1—0x4008 8004, SSP1CR1—0x4003 0004, SSP2CR1—0x400A C004)

该寄存器控制 SSP 控制器某些方面的操作。

表484. SSPn 控制寄存器 1 位描述（SSP0CR1—地址 0x4008 8004，SSP1CR1—地址 0x4003 0004，SSP2CR1—0x400A C004）

位	符号	值	描述	复位值
0	LBM		回写模式。	0
		0	正常操作模式。	
		1	串行输入管脚可用作串行输出管脚（MOSI 或 MISO），而不是仅用作串行输入管脚（MISO 或 MOSI 分别起作用）。	
1	SSE		SSP 使能。	0
		0	SSP 控制器禁能。	
		1	SSP 控制器可与串行总线上的其它设备相互通信。在置位该位前，软件应将合适的控制器信息写入其它 SSP 寄存器和中断寄存器。	
2	MS		主机/从机模式。该位只能在 SSE 位为 0 时写入。	0
		0	SSP 控制器用作一个总线主机，驱动 SCLK、MOSI 和 SSEL 线并接收 MISO 线。	
		1	SSP 控制器用作一个总线从机，驱动 MISO 线并接收 SCLK、MOSI 和 SSEL 线。	
3	SOD		从机输出禁能。该位只与从机模式有关（MS=1）。如果该位为 1，禁止 SSP 控制器驱动发送数据线（MISO）。	0
31:4	-		保留。读取值未定义，只写入 0。	无

21.6.3 SSPn 数据寄存器（SSP0DR—0x4008 8008，SSP1DR—0x4003 0008，SSP2DR—0x400A C008）

软件可向该寄存器写入要发送的数据，或从该寄存器读取已接收的数据。

表485. SSPn 数据寄存器位描述（SSP0DR—地址 0x4008 8008，SSP1DR—0x4003 0008，SSP2DR—0x400A C008 ）

位	符号	描述	复位值
15:0	DATA	写： 状态寄存器的 TNF 位为 1 指示 Tx FIFO 未滿时，软件可将要发送的帧数据写入该寄存器。如果 Tx FIFO 以前为空且 SSP 控制器空闲，则立即开始发送数据。否则，写入该寄存器的数据要等到所有之前的数据发送（或接收）完成后才能发送。如果数据长度小于 16 位，软件必须对数据进行调整后再写入该寄存器。 读： 状态寄存器的 RNE 位为 1 指示 Rx FIFO 不为空时，软件可读取该寄存器的数据。软件读取该寄存器时，SSP 控制器将返回 Rx FIFO 中的最早收到的一帧数据。如果数据长度小于 16 位，该字段的数据必须进行合适的调整，高位补零。	0
31:16	-	保留。读取值未定义，只写入 0。	无

21.6.4 SSPn 状态寄存器（SSP0SR—0x4008 800C，SSP1SR—0x4003 000C，SSP2SR—0x400A C）

这是一个只读寄存器，反映了 SSP 控制器的当前状态。

表486. SSPn 状态寄存器位描述（SSP0SR—地址 0x4008 800C，SSP1SR—0x4003 000C，SSP2SR—0x400A C00C）

位	符号	描述	复位值
0	TFE	发送 FIFO 空。发送 FIFO 为空时该位为 1，反之为 0。	1
1	TNF	发送 FIFO 未滿。Tx FIFO 满时该位为 0，反之为 1。	1
2	RNE	接收 FIFO 不为空。接收 FIFO 为空时，该位为 0，反之为 1。	0
3	RFF	接收 FIFO 满。接收 FIFO 满时该位为 1，反之为 0。	0
4	BSY	忙。SSPn 控制器空闲时该位为 0，或当前正在发送/接收一帧数据和/或 TxFIFO 不为空时该位为 1。	0
31:5	-	保留。从保留位读取的值未定义。	无

21.6.5 SSPn 时钟预分频寄存器(SSP0CPSR—0x4008 8010，SSP1CPSR—0x4003 0010，SSP2CPSR—0x400A C010)

该寄存器控制着通过预分频器分频 PCLK 来获得预分频时钟的系数，反过来，预分频时钟被 SSPnCR0 中的 SCR 系数分频后得到位时钟。

表487. SSPn 时钟预分频寄存器位描述（SSP0CPSR—地址 0x4008 8010，SSP1CPSR—0x4003 0010，SSP2CPSR—0x400A C010）

位	符号	描述	复位值
7:0	CPSDVSR	这是 2~254 中的一个偶数值。它是 PCLK 的分频因子，PCLK 通过分频后得到预分频器输出时钟。位 0 读出时总是为 0。	0
31:8	-	保留。读取值未定义，只写入 0。	无

重要提示：必须适当地初始化 SSPnCPSR 值，否则 SSP 控制器将无法正确发送数据。

在从机模式下，主机提供的 SSP 时钟速率不能大于 4.6.4 节中选定的外设时钟的 1/12。与 SSPnCPSR 寄存器的内容无关。

在主机模式下，CPSDVSR_{min}=2 或更大的值（只能为偶数）。

21.6.6 SSPn 中断使能置位/清零寄存器（SSP0IMSC—0x4008 8014，SSP1IMSC—0x4003 0014，SSP2IMSC—0x400A C014）

该寄存器控制 SSP 控制器中四个可能中断条件的使能。注意，ARM 所使用的术语“masked”与普通计算机术语中的“masked”的意思是相反的，“masked”在普通计算机术语中是“禁能”的意思，而在 ARM 术语中则是“使能”。为避免混淆，这里不使用“masked”这个词。

表488. SSPn 中断使能置位/清零寄存器位描述（SSP0IMSC—地址 0x4008 8014，SSP1IMSC—0x4003 0014，SSP2IMSC—0x400A C014）

位	符号	描述	复位值
0	RORIM	当接收溢出时，软件置位该位来使能中断。即，当 Rx FIFO 满且完成另一个帧的接收时该位置位。ARM 特别指出，发生接收溢出时新数据帧会将前面的数据帧覆盖。	0
1	RTIM	当接收超时发生时，软件置位该位来使能中断。当 Rx FIFO 不为空且在 32 个位时间内没有	0

位	符号	描述	复位值
		从 Rx FIFO 中读出数据时，产生接收超时。	
2	RXIM	软件置位该位，使得当 Rx FIFO 至少有一半为满时使能中断。	0
3	TXIM	软件置位该位，使得当 Tx FIFO 至少有一半为空时使能中断。	0
31:4	-	保留。读取值未定义，只写入 0。	无

21.6.7 SSPn 原始中断状态寄存器（SSP0RIS—0x4008 8018，SSP1RIS—0x4003 0018，SSP2RIS—0x400A C018）

这是一个只读寄存器，当中断条件出现时，寄存器中相应的位置位为 1，这与是否在 SSPnIMSC 寄存器中使能该中断无关。

表489. SSPn 原始中断状态寄存器位描述（SSP0RIS—地址 0x4008 8018，SSP1RIS—0x4003 0018，SSP2RIS—0x400A C018）

位	符号	描述	复位值
0	RORRIS	当 Rx FIFO 满且又接收到另一帧数据时，该位置位为 1。ARM 特别指出，上述情况发生时新数据帧会将前面的数据帧覆盖。	0
1	RTRIS	当 Rx FIFO 不为空且在 32 个位时间内没有从 Rx FIFO 中读出数据时，该位置位为 1。	0
2	RXRIS	如果 Rx FIFO 至少有一半为满时，该位置位为 1。	0
3	TXRIS	如果 Tx FIFO 至少有一半为空时，该位置位为 1。	1
31:4	-	保留。从保留位读出的值未定义。	无

21.6.8 SSPn 使能中断状态寄存器（SSP0MIS—0x4008 801C，SSP1MIS—0x4003 001C，SSP2MIS—0x400A C01C）

这是一个只读寄存器，当中断条件出现且相应的中断在 SSPnIMSC 中被使能时，该寄存器中相应的位会置位为 1。当 SSP 中断出现时，中断服务程序可通过读该寄存器来判断中断源。

表490. SSPn 使能中断状态寄存器位描述（SSP0MIS—地址 0x4008 801C，SSP1MIS—0x4003 001C，SSP2MIS—0x400A C01C）

位	符号	描述	复位值
0	RORMIS	当 Rx FIFO 满时又接收到另外一帧数据，且中断被使能时，该位置位为 1。	0
1	RTMIS	如果 Rx FIFO 不为空且在 32 个位时间内没有从 Rx FIFO 中读出数据时，该位置位为 1。	0
2	RXMIS	如果 Rx FIFO 至少有一半为满且中断被使能时，该位置位为 1。	0
3	TXMIS	如果 Tx FIFO 至少有一半为空且中断被使能时，该位置位为 1。	0
31:4	-	保留。从保留位读取的值未定义。	无

21.6.9 SSPn 中断清零寄存器(SSP0ICR—0x4008 8020, SSP1ICR—0x4003 0020, SSP2ICR—0x400A C020)

软件可以向该只写寄存器中写入一个或多个 1 来清除 SSP 控制器中相应的中断条件。需要注意的是另外两个中断条件可通过写或读相应的 FIFO 来清除，或通过清除 SSPnIMSC 中对应的位来禁能。

表491. SSPn 中断清零寄存器位描述 (SSP0ICR—地址 0x4008 8020, SSP1ICR—0x4003 0020, SSP2ICR—0x400A C020)

位	符号	描述	复位值
0	RORIC	向该位写“1”来清除“当 RxFIFO 满时帧被接收”中断条件。	无
1	RTIC	向该位写“1”来清除“Rx FIFO 不为空且在 32 个位时间内没有从 Rx FIFO 中读出数据”中断条件。	无
31:2	-	保留。读取值未定义，只写入 0。	无

21.6.10 SSPn DMA 控制寄存器 (SSP0DMACR—0x4008 8024, SSP1DMACR—0x4003 0024, SSP2DMACR—0x400A C024)

SSPnDMACR 寄存器是 DMA 控制寄存器。它是一个读/写寄存器。

表492. SSPn DMA 控制寄存器位描述 (SSP0DMACR—地址 0x4008 8024, SSP1DMACR—0x4003 0024, SSP2DMACR—0x400A C024)

位	符号	描述	复位值
0	接收 DMA 使能 (RXDMAE)	当该位被置“1”时，接收 FIFO 的 DMA 被使能，否则接收 DMA 被禁能。	0
1	发送 DMA 使能 (TXDMAE)	当该位被置“1”时，发送 FIFO 的 DMA 被使能，否则发送 DMA 被禁能。	0
31:2	-	保留。读取值未定义，只写入 0。	无

22.1 基本配置

I²C0/1/2 接口的配置需要使用下列寄存器：

1. 功率：在 PCONP 寄存器（[表 37](#)）中置位 PCI2C0/1/2 位。
注：在复位时，所有 I²C 接口都被使能 (PCI2C0/1/2 = 1)。
2. 外设时钟：I²C 接口使用公共时钟，它同时用于总线接口和大多数 APB 外设的功能部分。参见 [4.6.4](#) 节。
3. 管脚：使用相关的 IOCON 寄存器（参见 [8.4.1](#) 节）选择 I²C0、I²C1 或 I²C2 管脚和管脚模式。

注：管脚 P0[27]和 P0[28]是专用的开路漏极 I²C 管脚，支持全兼容的快速模式（400kHz）和标准模式（100kHz）的 I²C。这些管脚完全没有片上的上拉设备，当用作输出时，一定要做外部的上拉（根据 I²C 总线规范）。管脚 P5[2]和 P5[3]与上述两个管脚相似，但它们还支持增强型快速模式（Fast Mode Plus）（1MHz）的 I²C。与大多数端口的管脚相比，以上两组管脚都有稍微不同的配置选项，详见 [8.4.1](#) 节。所有可用于 I²C 通信的其他管脚，请参见下面的注释。

注：可以通过相关的 IOCON 寄存器将不使用专用 I²C 焊盘的 I²C 管脚（[表 67](#) 中已标识出来）配置为开路漏极模式，而且这些管脚可用于快速模式（400kHz）和标准模式（100kHz）的 I²C。这些管脚不包含用于抑制线路上毛刺的模拟滤波器，而是由 I²C 模块自身的数字滤波器执行相似的功能。这些管脚应配置为：无上拉、无下拉、开路漏极模式。

4. 利用相应的中断置位使能寄存器使能 NVIC 中的中断。
5. 初始化：参见 [22.9.8.1](#) 节和 [22.10.1](#) 节。

22.2 特性

- 支持 1MHz 的增强型快速模式(仅 I²C0 的部分引出线支持)、400kHz 的快速模式和 100kHz 的标准模式。
- 标准的 I²C 兼容总线接口，可配置为主模式、从模式或主/从模式。
- 多个主机同时传输时会进行仲裁，避免了总线上串行数据的冲突。
- 允许对时钟编程，以调整 I²C 的传输速率。
- 数据在主机和从机之间的传输是双向的。
- 串行时钟同步使得不同位速率的设备可通过一个串行总线进行通信。
- 串行时钟同步作为握手机制来挂起和恢复串行传输。
- 从机可有多达四个不同的地址。
- 监控模式允许观察 I²C 总线上所有的流量，忽视从机地址，也不影响实际 I²C 总线的流量。

22.3 应用

该接口用于与外部 I²C 标准器件连接，如串行 RAM、LCD、音调发生器和其他微控制器等，也可以用作诊断/测试总线。

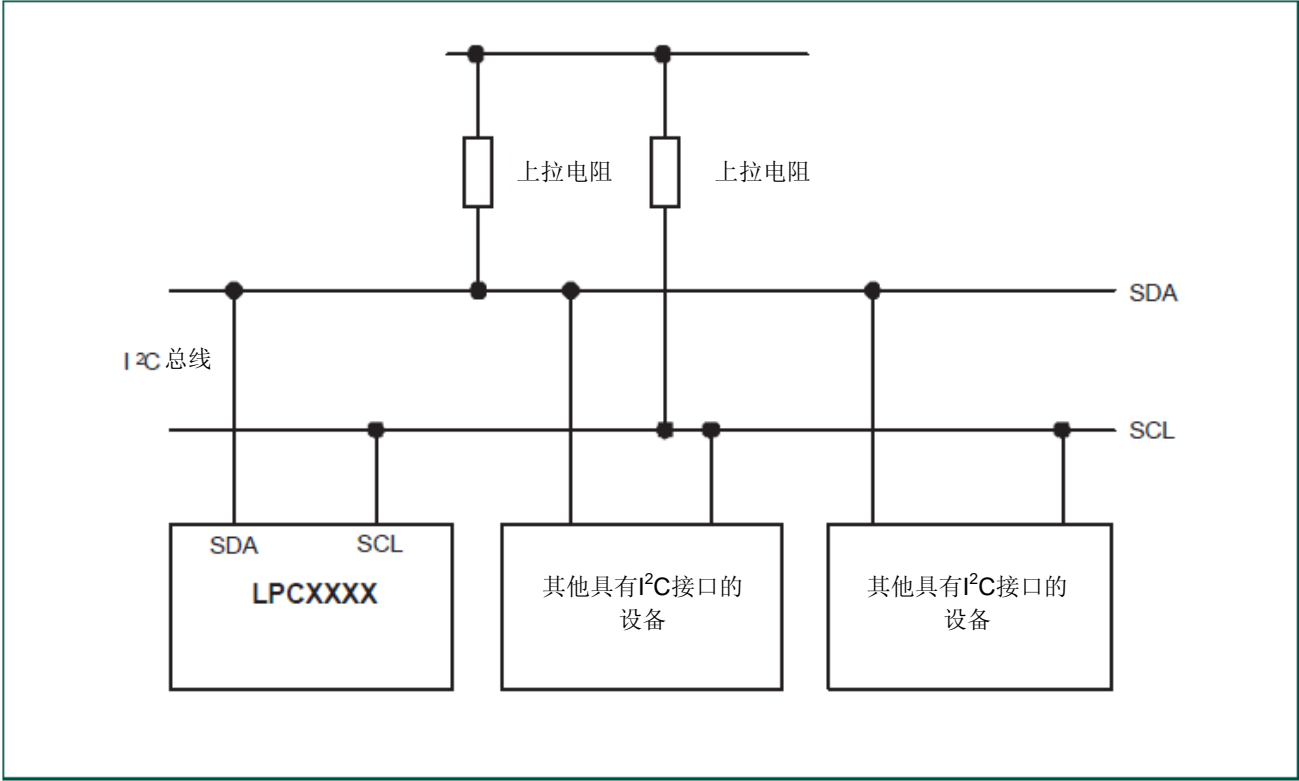
22.4 描述

I²C 总线的典型配置如[图 106](#)所示。根据方向位（R/W）的状态，在 I²C 总线上可能有以下两种类型的数据传输：

- 数据传输从一个主发送器到一个从接收器。主机发送的第一个字节是从机地址。接下来是大量的数据字节。从机在每个字节接收完成后返回一个应答位，直至从设备无法接受更多数据。
- 数据传输从一个从发送器到一个主接收器。主机发送的第一个字节是从机地址，然后从机返回一个应答位。接下来从机向主机发送数据字节。除最后一个字节之外，主机在其他每个字节接收完成之后都会返回一个应答位。接收完最后一个字节后，主机返回一个“非应答位”。主设备产生所有串行时钟脉冲、起始条件以及停止条件。每一帧的传输都是以一个停止条件或一个重复起始条件来结束。由于重复的起始条件也是下一个串行传输的开始，因此 I²C 总线不会被释放。

LPC178x/177x 的 I²C 接口是面向字节的，有四种操作模式：主发送器模式、主接收器模式、从发送器模式以及从接收器模式。

图106. I²C 总线配置



22.4.1 I2C 增强型快速模式

增强型快速模式（Fast Mode Plus）是指以 1Mbit/s 的传输速率与 NXP 半导体公司供应的 I2C 产品进行通信。

要使用增强型快速模式，必须配置 I2C0 管脚，然后选择 400kHz~1MHz 的速率，参见[表 507](#)。要将管脚配置成支持增强型快速模式，必须将 IOCON_P5_02 和 IOCON_P5_03 寄存器中的 I2CMODE 位设置为二进制数 10，参见 [8.4.1](#) 节。

22.5 管脚描述

表493. I2C 管脚描述

管脚	类型	描述
I2C0_SDA	输入/输出	I2C0 串行数据
I2C2_SCL	输入/输出	I2C2 串行时钟
I2C0_SDA	输入/输出	I2C0 串行数据
I2C2_SCL	输入/输出	I2C2 串行时钟
I2C0_SDA	输入/输出	I2C0 串行数据
I2C2_SCL	输入/输出	I2C2 串行时钟

3 个 I2C 接口的内部逻辑是完全相同的。这些接口可通过多种方式引至设备管脚，其中一些具有不同的管脚 I/O 特征。管脚 P0[27]和 P0[28]上的 I2C0 使用专用 I2C 焊盘，支持完全符合规范的快速模式和标准模式的 I2C。管脚 P5[2]和 P5[3]上的 I2C0 同样使用专用 I2C 焊盘。这些焊盘支持增强型快速模式以及上文提到的模式。

除上文所述这两组管脚以外，引至其他管脚的所有 I2C 接口都使用标准 I/O 管脚。这些管脚同样支持快速模式和标准模式下的 I2C 操作。主要不同之处在于，这些管脚上没有专用 I2C 焊盘上的模拟尖峰信号（spike）抑制滤波器。所有 I2C 接口上都带有一个数字滤波器，可用于同一用途。

22.6 I²C 操作模式

在一个指定的应用中，I²C 模块可用作主机、从机或同时用作主机和从机。在从机模式下，I²C 硬件查找其自身的四个从机地址之一和通用调用（General Call）地址。如果检测到其中的一个地址，则产生一个中断请求。如果处理器要成为总线主机，则硬件要等到总线空闲，然后再进入主机模式，以保证不中断从机的操作。如果在主机模式下总线仲裁丢失，则 I²C 模块立刻切换到从机模式并在同一个串行传输中检测自身已配置的从机地址。

22.6.1 主发送器模式

在该模式下，数据从主机发送到从机。在进入主发送器模式之前，I2CONSET 寄存器必须按照表 494 所示进行初始化。I2EN 必须设置为 1 来使能 I²C 功能。如果 AA 位为 0，而另一个设备是总线主机时，I²C 接口将不会对任何地址产生应答，也就是说它无法进入从机模式。STA、STO 和 SI 位必须设置为 0。向 I2CONCLR 寄存器中的 SIC 位写入 1 可清除 SI 位。写入从机地址后，应该清零 STA 位。

表494. 用于配置主机模式的 I2C0CONSET 和 I2C1CONSET

位	7	6	5	4	3	2	1	0
符号	-	I2EN	STA	STO	SI	AA	-	-
值	-	1	0	0	0	0	-	-

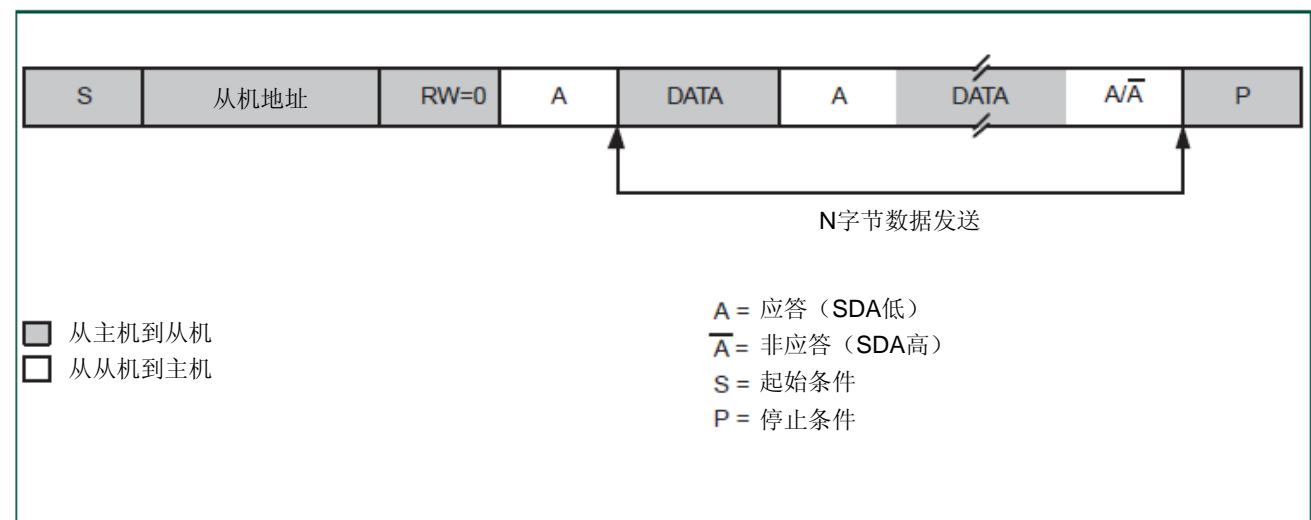
第一个发送的字节包含接收设备的从机地址（7 位）和数据方向位。在该模式下，数据方向位（R/W）应为 0，表示执行写操作。因此第一个发送的字节包含从机地址和写操作位。每次发送 8 位的数据。每发送完一个字节，都会收到一个应答位。输出起始条件和停止条件可用来指示一个串行传输的开始和结束。

当软件设置 STA 位时，I²C 接口将进入主发送器模式。I²C 逻辑在总线空闲后立即发送一个起始条件。在发送完起始条件后，SI 会置位。此时，I2STAT 寄存器中的状态码为 0x08。该状态码用于指向一个状态服务程序。该状态服务程序把从机地址和写操作位加载到 I2DAT 寄存器中，然后清零 SI 位。向 I2CONCLR 寄存器中的 SIC 位写入 1 可清零 SI。

当从机地址和 R/W 位已发送后且接收到一个应答位，SI 位再次置位，并且对于主机模式，可能的状态码是 0x18、0x20 或 0x38，如果已使能了从机模式（AA=1），则可能的状态码是 0x68、0x78 或 0xB0。

每个状态码对应的执行操作如表 511 到表 514 所示。

图107. 主发送器模式的格式



22.6.2 主接收器模式

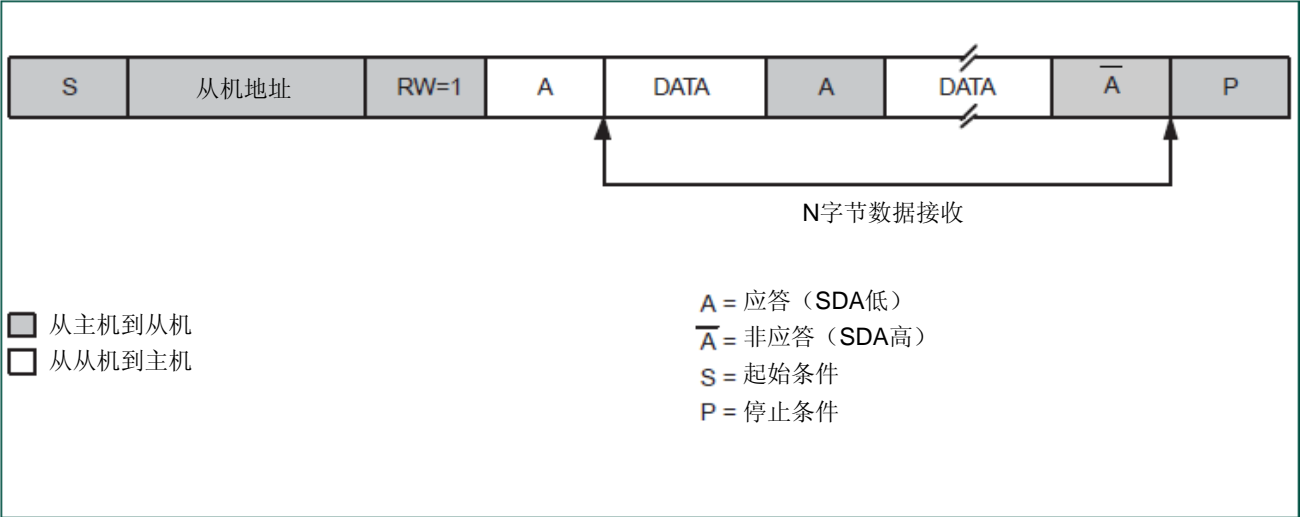
在主接收器模式下，接收的数据来自从发送器。传输的发起方式与主发送器模式中的发起方式相同。在发送完起始条件后，中断服务程序必须把从机地址和数据方向位加载到 I2C 数据寄存器 (I2DAT) 中，然后清零 SI 位。此时，数据方向位 (R/W) 应为 1，表示执行读操作。

当从机地址和数据方向位已发送且接收到一个应答位，SI 位置位，状态寄存器将显示状态码。对于主机模式，可能的状态码是 0x40、0x48 或 0x38。对于从机模式，可能的状态码是 0x68、0x78 或 0xB0。有关详细信息，参见[表 512](#)。

当 LPC178x/177x 需要应答一个接收的字节时，需要先设置相应的 AA 位，然后再清零 SI 位，并启动字节的读操作。当 LPC178x/177x 不需要应答接收的字节时，需要先清零 AA 位，然后再清零 SI 位，并启动字节的读操作。

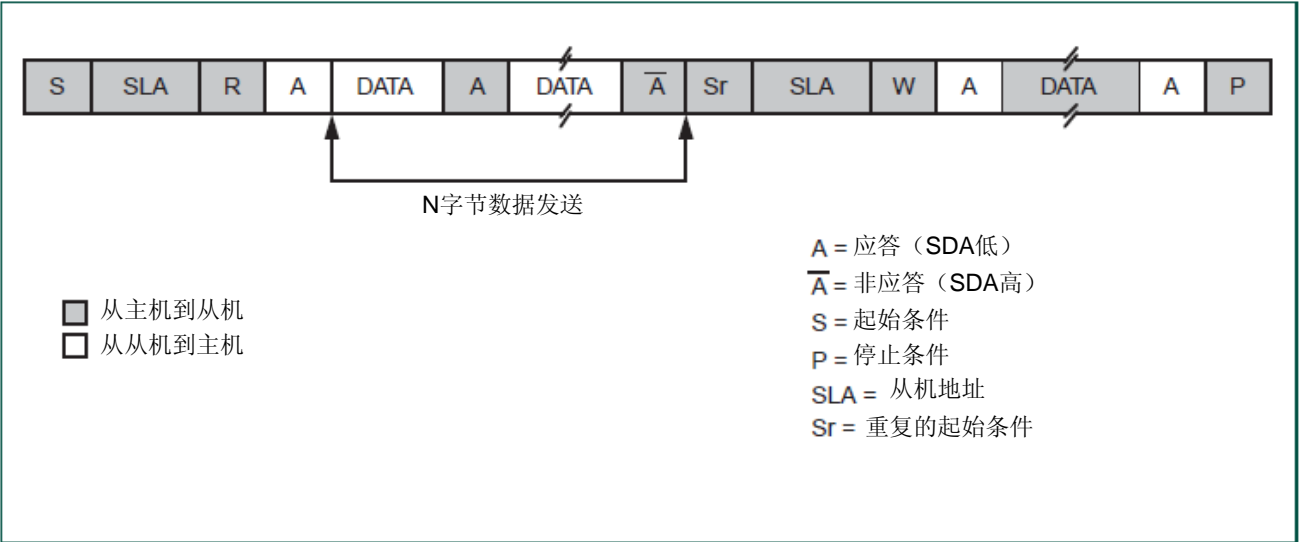
注意，最后一个接收的字节总是跟随一个来自 LPC178x/177x 的非应答位，这样，主机就可以告知从机读序列已完成，它需要发出一个停止命令或重复的起始命令。一旦发送了非应答位且 SI 位置位，LPC178x/177x 就可以发送一个停止位 (STO 位置位) 或重复的起始位 (STA 位置位)。然后，SI 位被清零以启动请求的操作。

图108. 主接收器模式的格式



发送完重复的起始条件后，I²C 可以切换到主发送器模式。

图109. 在发送重复的起始条件后，主接收器切换为主发送器



22.6.3 从接收器模式

在从接收器模式下，接收的数据字节来自于主发送器。要初始化从接收器模式，用户必须对任意一个从地址寄存器（I2ADR0-3）、从机屏蔽寄存器（I2MASK0-3）和 I²C 控制置位寄存器（I2CONSET）执行写操作，如表 495 所示。

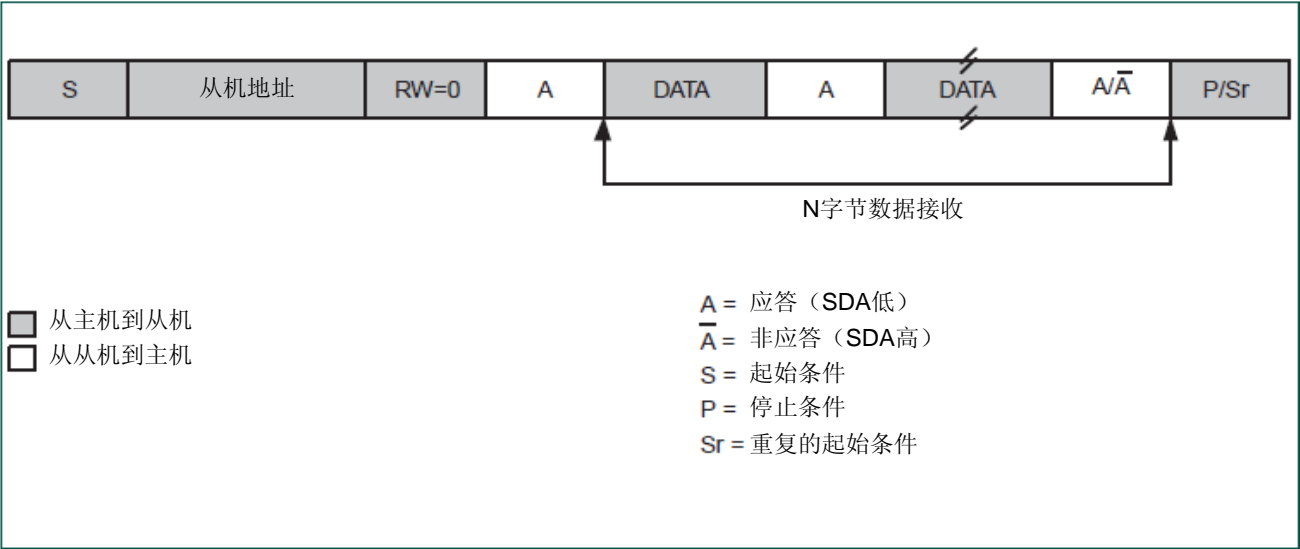
表495. 用于配置从机模式的 I2C0CONSET 和 I2C1CONSET

位	7	6	5	4	3	2	1	0
符号	-	I2EN	STA	STO	SI	AA	-	-
值	-	1	0	0	0	1	-	-

I2EN 必须设置为 1 以使能 I²C 功能。AA 位必须设置为 1，以使 I²C 应答自身的从机地址或通用调用地址。STA、STO 和 SI 位设置为 0。

在 I2ADR 和 I2CONSET 初始化以后，I²C 接口会一直等待直到它被自身的从机地址或通用调用地址（后面都紧跟数据方向位）寻址为止。如果方向位为 0（W），则 I²C 将进入从接收器模式。如果方向位为 1（R），则 I²C 将进入从发送器模式。在接收到地址和方向位后，SI 置位并可从状态寄存器（I2STAT）中读出一个有效的状态码。有关状态码及操作，请参见表 513。

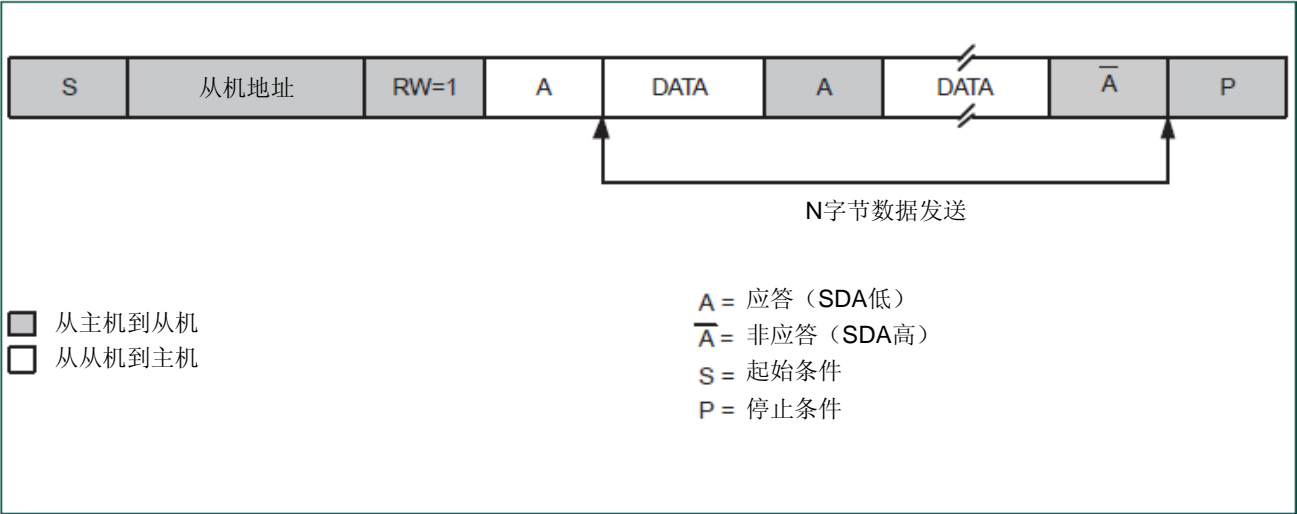
图110. 从接收器模式的格式



22.6.4 从发送器模式

第一个字节的接收和处理与从接收器模式中的相同。但在该模式下，方向位为 1，表示执行读操作。串行数据通过 SDA 发送，而串行时钟通过 SCL 输入。起始和停止条件用来指示串行传输的起始和结束。在一个指定的应用中，I²C 既可以作为主机也可以作为从机。在从机模式下，I²C 硬件寻找其自身的从机地址和通用调用地址。如果检测到其中一个地址，将产生中断请求。当微控制器希望成为总线主机时，硬件要等到总线空闲才能进入主机模式，以保证不会中断可能的从机操作。如果在主机模式下总线仲裁丢失，则 I²C 接口会立刻切换到从机模式，并可以在同一个串行传输中检测自身的从地址。

图111. 从发送器模式的格式



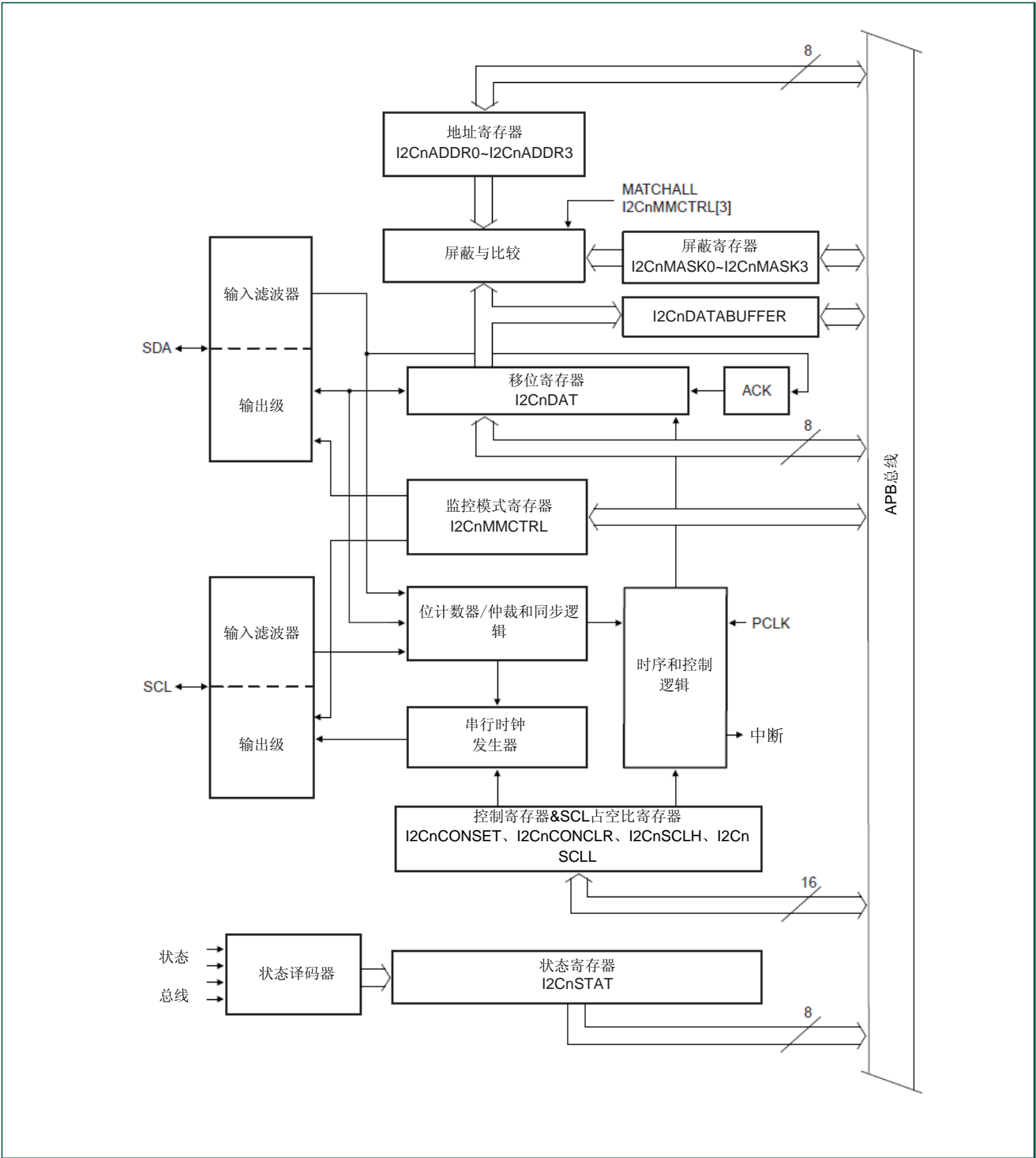
22.7 I²C 的实现和操作

图 112 所示为片上 I²C 总线接口的实现方式，之后对图中各个模块进行描述。

22.7.1 输入滤波器和输出级

输入信号与内部时钟同步，不到三个时钟的尖峰信号将被滤除。

图112. I²C 串行接口模块框图



22.7.2 地址寄存器 (I2ADR0~I2ADR3)

当设备被编程为从发送器或接收器时，这些寄存器可装入 I²C 模块即将响应的 7 位从机地址（7 个最高有效位）。LSB（GC）用于使能通用调用地址（0x00）的识别。当使能了多个从机地址时，在接收到设备自身从机地址后，实际接收到的地址可从 I2DAT 寄存器中读出。

注：在本章剩余的篇幅中，“自身从机地址”这一短语指在地址屏蔽后，四个已配置的从机地址中的任意一个。

22.7.3 地址屏蔽寄存器 (I2MASK0~I2MASK3)

每个屏蔽寄存器（共四个）都包含七个有效位（7:1）。这些寄存器中任何位被置位为“1”都会引起接收地址与屏蔽寄存器相关联的 I2ADRN 寄存器中的相应位做自动比较。换句话说，在确定一个地址匹配时不会考虑 I2ADRN 寄存器中已被屏蔽的位。

当出现一个地址匹配中断时，处理器将必须读取数据寄存器（I2DAT）来确定实际引起匹配的接收地址。

22.7.4 比较器

屏蔽后，比较器将接收到的 7 位从机地址与其自身的从机地址（I2ADR0~I2ADR3）进行比较。它还将首次接收到的 8 位字节与通用调用地址（0x00）相比较。如果发现了一个匹配，则相应的状态位置位并产生一个中断请求。

22.7.5 移位寄存器 I2DAT

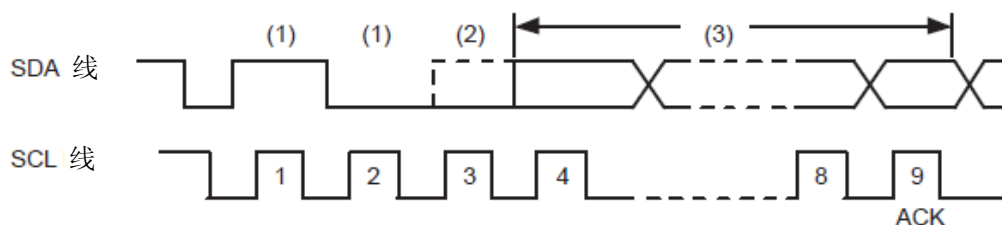
该 8 位寄存器包含要发送的一个字节的串行数据或刚接收到的一个字节。I2DAT 中的数据总是从右向左移动；最先发送的位是 MSB（位 7），接收完一个字节后，接收到的数据的第一位放置到 I2DAT 的 MSB。数据移出时，总线上的数据同时移入；I2DAT 通常保存的是总线上的最后一个字节。因此，当仲裁丢失时，主发送器到从接收器的变换和 I2DAT 中的数据更新同时进行。

22.7.6 仲裁和同步逻辑

在主发送器模式下，仲裁逻辑会检查每个发送的逻辑 1，看它是否真正出现在 I²C 总线上。如果总线上另一个设备撤消了一个逻辑 1 并拉低 SDA 线，仲裁丢失，I²C 模块立即从主发送器变为从接收器。I²C 模块将继续输出时钟脉冲（在 SCL 上），直至当前的串行字节发送完毕。

在主接收器模式下，仲裁也可能丢失。在该模式下，只有当 I²C 模块正在向总线返回一个“非应答（逻辑 1）”时才可能发生仲裁丢失。当总线上另一个设备将信号拉低时仲裁丢失。由于它只在一个串行字节结束时出现，因此 I²C 模块不会再产生时钟脉冲。[图 113](#) 所示为仲裁过程。

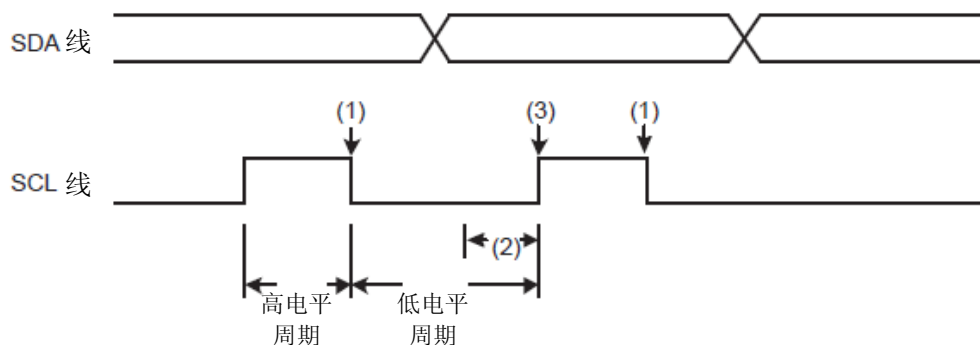
图113. 仲裁过程



- (1) 另一设备发送串行数据;
 - (2) 另一设备通过拉低SDA线先撤消了该I²C主机发送的一个逻辑（虚线）。仲裁丢失，I²C进入从接收模式;
 - (3) 此时I²C处于从接收模式，但仍产生时钟脉冲，直至发送完当前的字节。
- I²C将不为下个字节的传输产生时钟脉冲。一旦赢得仲裁，SDA上的数据传输由新的主机来启动。

同步逻辑使串行时钟发生器与另一个设备 SCL 线上的时钟脉冲同步。如果两个或更多主设备产生时钟脉冲，则“mark”持续时间取决于产生最短“mark”时间的设备；“space”持续时间则取决于产生最长“space”时间的设备。图 114 所示为同步过程。

图114. 串行时钟同步



- (1) 另一个设备在 I^2C 计时完一个完整的高电平时间之前拉低SCL线。其他设备决定了（较短）高电平的时间；
- (2) 另一个设备在 I^2C 计时完一个完整低电平时间并释放SCL后继续拉低SCL线。
 I^2C 时钟发生器被迫等待，直到SCL变高。其他设备决定了（较长）低电平的时间；
- (3) SCL线被释放，时钟发生器开始计时高电平时间。

从机可以延长 **space** 持续时间来使总线主机减速。也可通过延长 **space** 持续时间来实现握手。这一操作在每位或一个完整字节传输之后进行。I²C 模块将在发送或接收完一个字节且传输完应答位后延长 **SCL space** 时间。设置串行中断标志(SI)，继续延长 **space** 时间，直至串

行中断标志清除。

22.7.7 串行时钟发生器

当 I²C 模块处于主发送器模式或主接收器模式时，这个可编程时钟脉冲发生器可产生 SCL 时钟脉冲。当 I²C 模块处于从机模式时，该发生器关闭。I²C 输出时钟频率和占空比可通过 I²C 时钟控制寄存器编程。

有关详细信息，参见 I2CSCLL 和 I2CSCLH 寄存器的描述。除非总线与前述的其他 SCL 时钟源同步，否则输出时钟脉冲使用设定好的占空比。

22.7.8 时序和控制

时序和控制逻辑会产生处理串行字节的时序和控制信号。该逻辑模块为 I2DAT 中的数据提供移位脉冲，可启用比较器、生成和检测起始和停止条件以及接收和发送应答位、控制主机模式和从机模式，该模块还具有中断请求逻辑以及监控 I²C 总线状态等功能。

22.7.9 控制寄存器，I2CONSET 和 I2CONCLR

I²C 控制寄存器包含用于控制以下 I²C 模块功能的位：串行传输的启动和重启、串行传输的终止、位速率、地址识别和应答。

I²C 控制寄存器的内容可以读取为 I2CONSET。写 I2CONSET 会置位 I²C 控制寄存器中相应的位为 1；相反，写 I2CONCLR 会清除 I²C 控制寄存器中被写入 1 的位。

22.7.10 状态译码器和状态寄存器

状态译码器读取所有内部状态位并将它们压缩成一个 5 位代码。该代码与每个 I²C 总线状态位一一对应。该 5 位代码可用于产生向量地址，以便快速处理不同的服务程序。每个服务程序处理一个特定的总线状态。如果 I²C 模块的全部四种模式都被使用，则有 26 种可能的总线状态。当串行中断标志置位（通过硬件）并一直保持时，5 位状态代码锁存到状态寄存器的五个最高有效位，直至中断标志被软件清除。状态寄存器的三个最低有效位始终为零。如果状态代码用作一个服务程序的向量，则程序转移到 8 位地址指向的空间。大多数的服务程序不会超过 8 字节（请参见本节中的软件示例）。

22.8 寄存器描述

每个 I²C 接口包含 16 个寄存器，如表 496 中所示。

注：LPC178x/177x 中新增了下列寄存器，以支持对从机模式下多个地址和新监控模式的响应：I2ADR1~3、I2MASK0~3、MMCTRL 以及 I2DATA_BUFFER。

表496. I²C 寄存器映射

通用名称	描述	访问	复位值 ^[1]	I ² Cn 的名称和地址	表
I2CONSET	I²C 控制置位寄存器。 当向该寄存器写入 1 时，I ² C 控制寄存器中相应位置位。写 0 到 I ² C 控制寄存器的相应位没有影响。	R/W	0x00	I2C0CONSET—0x4001 C000 I2C1CONSET—0x4005 C000 I2C2CONSET—0x400A 0000	表 497
I2STAT	I²C 状态寄存器。 在 I ² C 操作中，该寄存器提供详细的状态码，使软件确定所需的下一步操作。	RO	0xF8	I2C0STAT—0x4001 C004 I2C1STAT—0x4005 C004 I2C2STAT—0x400A 0004	表 499
I2DAT	I²C 数据寄存器。 在主/从机发送模式下，要发送的数据被写入该寄存器。在主/从机接收模式下，接收到的数据可从该寄存器中读取。	R/W	0x00	I2C0DAT—0x4001 C008 I2C1DAT—0x4005 C008 I2C2DAT—0x400A 0008	表 500
I2ADR0	I²C 从地址寄存器 0。 包含从机模式下 I ² C 接口操作的 7 位从地址，不在主机模式下使用。最低位决定从机是否响应通用调用地址。	R/W	0x00	I2C0ADR0—0x4001 C00C I2C1ADR0—0x4005 C00C I2C2ADR0—0x400A 000C	表 503
I2SCLH	SCH 占空比寄存器高半字。 确定 I ² C 时钟的高电平时间。	R/W	0x04	I2C0SCLH—0x4001 C010 I2C1SCLH—0x4005 C010 I2C2SCLH—0x400A 0010	表 505
I2SCLL	SCH 占空比寄存器低半字。 确定 I ² C 时钟的低电平时间。I2nSCLL 和 I2nSCLH 一起确定 I ² C 主机产生的时钟频率和从机模式下使用的特定时间。	R/W	0x04	I2C0SCLL—0x4001 C014 I2C1SCLL—0x4005 C014 I2C2SCLL—0x400A 0014	表 506
I2CONCLR	I²C 控制清零寄存器。 当向该寄存器写入 1 时，I ² C 控制寄存器中相应位被清零。写 0 到 I ² C 控制寄存器的相应位没有影响。	WO	无	I2C0CONCLR—0x4001 C018 I2C1CONCLR—0x4005 C018 I2C2CONCLR—0x400A 0018	表 498
MMCTRL	监控模式控制寄存器。	R/W	0x00	I2C0MMCTRL—0x4001 C01C I2C1MMCTRL—0x4005 C01C I2C2MMCTRL—0x400A 001C	表 501
I2ADR1	I²C 从地址寄存器 1。 包含从机模式下 I ² C 接口操作的 7 位从地址，不在主机模式下使用。最低位决定从机是否响应通用调用地址。	R/W	0x00	I2C0ADR1—0x4001 C020 I2C1ADR1—0x4005 C020 I2C2ADR1—0x400A 0020	表 503
I2ADR2	I²C 从地址寄存器 2。 包含从机模式下 I ² C 接口操作的 7 位从地址，不在主机模式下使用。最低位决定从机是否响应通用调用地址。	R/W	0x00	I2C0ADR2—0x4001 C024 I2C1ADR2—0x4005 C024 I2C2ADR2—0x400A 0024	表 503
I2ADR3	I²C 从地址寄存器 3。 包含从机模式下 I ² C 接口操作的 7 位从地址，不在主机模式下使用。最低位决定从机是否响应通用调用地址。	R/W	0x00	I2C0ADR3—0x4001 C028 I2C1ADR3—0x4005 C028 I2C2ADR3—0x400A 0028	表 503
I2DATA_BUFFER	数据缓冲寄存器。 每次从总线接收到 9 个位（8 个数据位和一个应答位或一个非应答位）之后，移位寄存器 I2DAT 中的最高 8 位就会自动发送到 I2DATA_BUFFER 中。	RO	0x00	I2C0DATA_BUFFER—0x4001 C02C I2C1DATA_BUFFER—0x4005 C02C I2C2DATA_BUFFER—0x400A 002C	表 502
I2MASK0	I²C 从地址屏蔽寄存器 0。 该屏蔽寄存器与地址寄存器 I2ADR0 相关联，可确定地址匹配。当与通用调用地址（“0000000”）比较时，屏蔽寄存器不起作用。	R/W	0x00	I2C0MASK0—0x4001 C030 I2C1MASK0—0x4005 C030 I2C2MASK0—0x400A 0030	表 504

通用名称	描述	访问	复位值 ^[1]	I ² Cn 的名称和地址	表
I2MASK1	I²C 从地址屏蔽寄存器 1。 该屏蔽寄存器与地址寄存器 I2ADR0 相关联，可确定地址匹配。当与通用调用地址（“0000000”）比较时，屏蔽寄存器不起作用。	R/W	0x00	I2C0MASK1—0x4001 C034 I2C1MASK1—0x4005 C034 I2C2MASK1—0x400A 0034	表 504
I2MASK2	I²C 从地址屏蔽寄存器 2。 该屏蔽寄存器与地址寄存器 I2ADR0 相关联，可确定地址匹配。当与通用调用地址（“0000000”）比较时，屏蔽寄存器不起作用。	R/W	0x00	I2C0MASK2—0x4001 C038 I2C1MASK2—0x4005 C038 I2C2MASK2—0x400A 0038	表 504
I2MASK3	I²C 从地址屏蔽寄存器 3。 该屏蔽寄存器与地址寄存器 I2ADR0 相关联，可确定地址匹配。当与通用调用地址（“0000000”）比较时，屏蔽寄存器不起作用。	R/W	0x00	I2C0MASK3—0x4001 C03C I2C1MASK3—0x4005 C03C I2C2MASK3—0x400A 003C	表 504

[1] 复位值仅反映已使用位中保存的数据，不包括保留位的内容。

22.8.1 I²C 控制置位寄存器（I2CONSET: I²C0, I2C0CONSET—0x4001 C000; I²C1, I2C1CONSET—0x4005 C000; I²C2, I2C2CONSET—0x400A 0000）

I2CONSET 寄存器控制 I2CON 寄存器中各个位的设置，从而控制 I²C 接口的操作。向该寄存器中的一个位写入 1 会置位 I²C 控制寄存器中的相应位。写入 0 没有影响。读取该寄存器会得到控制位和标志位的当前值。

表497. I²C 控制置位寄存器位描述（I2CONSET: I²C0, I2C0CONSET—地址 0x4001 C000, I²C1, I2C1CONSET—地址 0x4005 C000, I²C2, I2C2CONSET—地址 0x400A 0000）

位	符号	描述	复位值
1:0	-	保留。读取值未定义，只写入 0。	无
2	AA	应答标志。	0
3	SI	I ² C 中断标志。	0
4	STO	停止标志。	0
5	STA	起始标志。	0
6	I2EN	I ² C 接口使能。	0
31:7	-	保留。读取值未定义，只写入 0。	无

I2EN: I²C 接口使能。当 I2EN 为 1 时，使能 I²C 接口。向 I2CONCLR 寄存器中的 I2ENC 位写入 1 将使 I2EN 位清零。当 I2EN 为 0 时，I²C 接口功能被禁止。

当 I2EN 为 0 时，SDA 和 SCL 输入信号被忽略，I²C 模块在“不可寻址的”从机状态中，且 STO 位被强制为“0”。

I2EN 不应用于暂时释放 I²C 总线，当 I2EN 复位时，I²C 总线状态丢失。应使用 AA 标志代替。

STA: 起始标志。当 STA=1 时，I²C 接口进入主模式并发送一个起始条件，如果已经处于主模式，则发送一个重复起始条件。

当 STA=1 并且 I²C 接口还没进入主模式时，I²C 接口进入主模式，检测总线并在总线空闲时产生一个起始条件。如果总线忙，则等待一个停止条件（释放总线）并在延迟半个内部时钟发生器周期后发送一个起始条件。当 I²C 接口已经处于主模式中并发送或接收了数据时，I²C 接口会发送一个重复的起始条件。STA 可在任何时候置位，当 I²C 接口处于可寻址的从模式时，STA 也可以置位。

向 I2CONCLR 寄存器中的 STAC 位写入 1 可使 STA 位清零。当 STA=0 时,不会产生起始或重复起始条件。

当 STA 和 STO 都置位时,如果 I²C 接口处于主模式,I²C 接口将向总线发送一个停止条件,然后发送一个起始条件。如果 I²C 接口处于从模式,则产生一个内部停止条件,但不发送到总线上。

STO: 停止标志。在主模式中,当 STO 为 1 时,会使 I²C 接口发送一个停止条件或在从模式中从错误状态中恢复。当主模式中 STO=1 时,向 I²C 总线发送停止条件。当总线检测到停止条件时,STO 自动清零。

在从模式中,置位 STO 位可从错误状态中恢复。这种情况下不向总线发送停止条件。硬件的表现就好像是接收到一个停止条件并切换到不可寻址的从接收器模式。STO 标志由硬件自动清零。

SI: I²C 中断标志。当 I²C 状态改变时 SI 置位。但是,进入状态 F8 不会使 SI 置位,因为在那种情况下中断服务程序不起作用。

当 SI 置位时,SCL 线上的串行时钟低电平周期扩展,且串行传输被中止。当 SCL 为高时,SI 标志的状态不受影响。SI 必须通过软件复位,通过向 I2CONCLR 寄存器的 SIC 位写入 1 来实现。只有当所需的位已经置位且在 I2DAT 中的值已被载入或读取,SI 位才应被清零。

AA: 应答标志位。当 AA 置为 1 时,在 SCL 线的应答时钟脉冲内,出现下面的任意条件之一将返回一个应答信号(SDA 线为低电平):

- 接收到一个匹配地址,该地址被寄存器 I2ADR0~I2ADR3 所定义,被寄存器 I2MASK0~I2MASK3 所屏蔽;
- 当 I2ADR 中的通用调用位(GC)置位时,接收到通用调用地址;
- 当 I²C 接口处于主接收器模式时,接收到一个数据字节;
- 当 I²C 接口处于可寻址的从接收器模式时,接收到一个数据字节。

向 I2CONCLR 寄存器中的 AAC 位写入 1 会使 AA 位清零。当 AA 为零时,SCL 线的应答时钟脉冲内出现下列情况将返回一个非应答信号(SDA 线为高电平):

- 当 I²C 接口处于主接收器模式时,接收到一个数据字节;
- 当 I²C 接口处于可寻址的从接收器模式时,接收到一个数据字节。

22.8.2 I2C 控制清零寄存器 (I2CONCLR: I2C0, I2C0CONCLR—0x4001 C018; I2C1, I2C1CONCLR—0x4005 C018; I2C2, I2C2CONCLR—0x400A 0018)

I2CONCLR 寄存器控制 I2CON 寄存器中各个位的清零，从而控制 I2C 接口的操作。向该寄存器中的一个位写入 1 会清零 I2C 控制寄存器中的相应位。写入 0 没有影响。I2CONCLR 是一个只写寄存器。从 I2CONSET 寄存器可以读取相关位的值。

表498. I2C 控制清零寄存器位描述 (I2CONCLR: I2C0, I2C0CONCLR—0x4001 C018, I2C1, I2C1CONCLR—0x4005 C018, I2C2, I2C2CONCLR—0x400A 0018)

位	符号	描述
1:0	-	保留。读取值未定义，只写入 0。
2	AAC	应答清零位。
3	SIC	I2C 中断清零位。
4	-	保留。读取值未定义，只写入 0。
5	STAC	起始标志清零位。
6	I2ENC	I2C 接口禁止位。
31:7	-	保留。读取值未定义，只写入 0。

AAC: 应答标志清零位。向该位写入 1 会清零 I2CONSET 寄存器中的 AA 位。写入 0 无效。
SIC: I2C 中断标志清零位。向该位写入 1 会清零 I2CONSET 寄存器中的 SI 位。写入 0 无效。
STAC: 起始标志清零位。向该位写入 1 会清零 I2CONSET 寄存器中的 STA 位。写入 0 无效。
I2ENC: I2C 接口禁止。向该位写入 1 会清零 I2CONSET 寄存器中的 I2EN 位。写入 0 无效。

22.8.3 I2C 状态寄存器 (I2STAT: I2C0, I2C0STAT—0x4001 C004; I2C1, I2C1STAT—0x4005 C004; I2C2, I2C2STAT—0x400A 0004)

每个 I2C 状态寄存器都反映了对应 I2C 接口的状况。I2C 状态寄存器是只读寄存器。

表499. I2C 状态寄存器 (I2STAT: I2C0, I2C0STAT—0x4001 C004; I2C1, I2C1STAT—0x4005 C004; I2C2, I2C2STAT—0x400A 0004)

位	符号	描述	复位值
2:0	-	不使用这些位，它们总是为 0。	0
7:3	Status	这些位提供 I2C 接口的实际状态信息。	0x1F
31:8	-	保留。从保留位读出的值未定义。	无

三个最低有效位始终为 0。当作为一个字节时，状态寄存器代表一个状态代码。一共有 26 种可能存在的状态代码。当状态代码为 0xF8 时，没有可用的相关信息，SI 位不会置位。所有其他 25 种状态代码都对应一个已定义的 I2C 状态。当进入其中一种状态时，SI 位将置位。有关状态代码的完整列表，请参见[表 511~表 514](#)。

22.8.4 I2C 数据寄存器 (I2DAT: I2C0, I2C0DAT—0x4001 C008; I2C1, I2C1DAT—0x4005 C008; I2C2, I2C2DAT—0x400A 0008)

该寄存器包含要发送或刚接收的数据。只有当该寄存器没有执行字节移位的操作时，CPU 才可以对该寄存器进行读写。该寄存器只能在 SI 位置位时访问。在 SI 置位期间，I2DAT 中的数据保持稳定。I2DAT 中的数据移位总是从右到左进行：要发送的第一个位是 MSB（位 7），在接收到一个字节后，所接收数据的第一位存放到 I2DAT 的 MSB。

表500. I2C 数据寄存器位描述(I2DAT: I2C0, I2C0DAT—0x4001 C008; I2C1, I2C1DAT—0x4005 C008; I2C2, I2C2DAT—0x400A 0008)

位	符号	描述	复位值
7:0	Data	该寄存器保存已接收的数据或要发送的数据。	0
31:8	-	保留。读取值未定义，只写入 0。	无

22.8.5 I2C 监控模式控制寄存器(I2MMCTRL: I2C0, I2C0MMCTRL—0x4001 C01C; I2C1, I2C1MMCTRL—0x4005 C01C; I2C2, I2C2MMCTRL—0x400A 001C)

该寄存器控制监控模式的使能，它可以使 I2C 模块监控 I2C 总线的流量，而不需要实际参与通信，也不会干扰 I2C 总线。

表501. I2C 监控模式控制寄存器位描述 (I2MMCTRL: I2C0, I2C0MMCTRL—0x4001 C01C; I2C1, I2C1MMCTRL—0x4005 C01C; I2C2, I2C2MMCTRL—0x400A 001C)

位	符号	值	描述	复位值
0	MM_ENA		监控模式使能。	0
		0	监控模式禁止。	
		1	I2C 模块将进入监控模式。在该模式下，SDA 输出将被置于高阻抗模式。避免了 I2C 模块向 I2C 数据总线输出任何类型的数据（包括应答）。根据 ENA_SCL 位的状态，输出也可以被强制为高电平，避免模块控制 I2C 时钟线。	
1	ENA_SCL		SCL 输出使能位。	0
		0	当模块在监控模式下该位被清零时，SCL 输出将强制为高电平。如上所述，可避免模块控制 I2C 时钟线。	
		1	当该位置位时，I2C 模块可对时钟线进行相同的操作，即正常操作。这意味着，作为一个从机设备，I2C 模块可以扩展时钟线（将它保持为低电平），直到它有时间响应 I2C 中断为止。 [1]	
2	MATCH_ALL		选择中断寄存器匹配。	0
		0	当该位清零时，中断只会在（多达 4 个地址寄存器：I2ADR0~I2ADR3）中的一个地址寄存器出现匹配时产生。这就是说，模块会作为一个普通的从机响应，直到有地址识别。	
		1	当该位置为“1”且 I2C 在监控模式下时，只要接收到任意一个地址，中断就会产生。这将使器件监控总线上的所有通信数据。	
31:3	-		保留。读取值未定义，只写入 0。	无

[1] 当 ENA_SCL 位被清零且 I2C 不能再延迟时钟时，中断响应时间就变得很重要。为了在这些情况下能给予器件更多响应 I2C 中断的时间，需使用 I2DATA_BUFFER 寄存器来保存接收到的数据(参见 [22.8.6 节](#))，保存时间为一个完整的 9 位字传输时间。

注：当 MM_ENA 为 0 时（即模块未处于监控模式），ENA_SCL 和 MATCH_ALL 位无效。

22.8.5.1 监控模式下的中断

模块处于监控模式时，所有中断都照常发生。这意味着：首次中断将出现在检测到一个地址匹配时（如果 MATCH_ALL 置位，则接收任意一个地址就会产生中断，否则中断只会在 4 个地址寄存器的其中一个出现匹配时产生）。

在检测到一个地址匹配后，对于从写传输，每接收到一个数据字节就产生一个中断；对于从读传输，在每发送完模块认为是要发送的字节以后就会产生中断。在第二种情况下，数据寄存器实际上保存了总线上其他从机发送的数据，这个从机实际上是被主机寻址的。

所有中断产生完以后，处理器会读取数据寄存器来查看总线上实际发送的数据。

22.8.5.2 监控模式下的仲裁丢失

在监控模式下，I²C 模块将不能响应总线主机的信息请求或发送应答，而是由总线上的其他从机代之做响应。

软件应当意识到：当模块处于监控模式下时，不对任何检测到的仲裁状态丢失做出响应。

22.8.6 I²C 数据缓冲寄存器 (I2DATA_BUFFER: I²C0, I2CDATA_BUFFER—0x4001 C02C ; I²C1 , I2C1DATA_BUFFER- 0x4005 C02C ; I²C2 , I2C2DATA_BUFFER- 0x400A 002C)

在监控模式下，如果 ENA_SCL 位没有置位，则 I²C 模块可能无法延长时钟。也就是说，处理器读取总线上已接收数据内容的时间有限。如果处理器以正常速度读取 I2DAT 移位寄存器，那么在已接收数据被新数据覆写之前，它可能只有 1 位时间对中断做出响应。

为了使处理器有更多响应时间，新增了一个 8 位只读寄存器 I2DATA_BUFFER。总线上每接收到 9 个位（8 个数据位加上应答位或非应答位）后，I2DAT 移位寄存器的 8 个 MSB 将自动发送到 I2DATA_BUFFER。这表示在数据被覆写之前，处理器有 9 个位的传输时间来响应中断和读取数据。

处理器仍可直接读取 I2DAT，I2DAT 无论如何是不会改变的。

即便 I2DATA_BUFFER 寄存器主要用于监控模式（ENA_SCL=0），但在任何一种操作模式下它可以随时读取。

表502. I²C 数据缓冲寄存器位描述 (I2DATA_BUFFER: I²C0, I2CDATA_BUFFER—0x4001 C02C; I²C1, I2C1DATA_BUFFER—0x4005 C02C; I²C2, I2C2DATA_BUFFER—0x400A 002C)

位	符号	描述	复位值
7:0	Data	该寄存器保存 I2DAT 移位寄存器的最高 8 位。	0
31:8	-	保留。从保留位读取的值未定义。	无

22.8.7 I²C 从地址寄存器 (I2ADR0~3: I²C0, I2C0ADR[0、1、2、3]—0x4001 C0[0C、20、24、28]; I²C1, I2C1ADR[0、1、2、3]—地址 0x4005 C0[0C、20、24、28]; I²C2, I2C2ADR[0、1、2、3]—地址 0x400A 00[0C、20、24、28])

这些寄存器可读可写,只用于 I²C 接口被设置为从机模式时。在主机模式下,该寄存器无效。I2ADR 的 LSB 是通用调用位。当该位置位时,通用调用地址 (0x00) 被识别。

如果这些寄存器包含 0x00,则 I²C 将不会应答总线上的任何地址。复位时,全部四个寄存器都会被清零为这个禁止状态。

表503. I²C 从地址寄存器位描述 (I2ADR0~3: I²C0, I2C0ADR[0, 1, 2, 3]—0x4001 C0[0C, 20, 24, 28]; I²C1, I2C1ADR[0, 1, 2, 3]—地址 0x4005 C0[0C, 20, 24, 28]; I²C2, I2C2ADR[0, 1, 2, 3]—地址 0x400A 00[0C, 20, 24, 28])

位	符号	描述	复位值
0	GC	通用调用使能位。	0
7:1	Address	从模式下的 I ² C 设备地址。	0
31:8	-	保留。读取值未定义,只写入 0。	无

22.8.8 I²C 屏蔽寄存器 (I2MASK0~3: I²C0, I2C0MASK[0、1、2、3]—0x4001 C0[30、34、38、3C]; I²C1, I2C1MASK[0、1、2、3]—地址 0x4005 C0[30、34、38、3C]; I²C2, I2C2MASK[0、1、2、3]—地址 0x400A 00[30、34、38、3C])

四个屏蔽寄存器各包含七个有效位 (7:1)。这些寄存器中的任一个位置位为 1 都会引起接收地址相应位的自动比较 (当它与屏蔽寄存器关联的 I2ADRN 寄存器比较时)。也就是说,在确定一个地址匹配时,不用考虑 I2ADRN 寄存器中已被屏蔽的位。

与通用调用地址 (“0000000”) 比较时,屏蔽寄存器无效。

当发生一个地址匹配中断时,处理器必须读取数据寄存器 (I2DAT) 来确定实际引起匹配的接收地址。

表504. I²C 屏蔽寄存器位描述 (I2MASK0~3: I²C0, I2C0MASK[0, 1, 2, 3]—0x4001 C0[30, 34, 38, 3C]; I²C1, I2C1MASK[0, 1, 2, 3]—地址 0x4005 C0[30, 34, 38, 3C]; I²C2, I2C2MASK[0, 1, 2, 3]—地址 0x400A 00[30, 34, 38, 3C])

位	符号	描述	复位值
0	-	保留,只写入 0。读取这些位总是返回 0。	0
7:1	MASK	屏蔽位。	0
31:8	-	保留,只写入 0。读取这些位总是返回 0。	0

22.8.9 I²C SCL 高电平占空比寄存器 (I2SCLH: I²C0, I2C0SCLH—0x4001 C010; I²C1, I2C1SCLH—0x4005 C010; I²C2, I2C2SCLH—0x400A 0010)

表505. I²C SCL 高电平占空比寄存器位描述 (I2SCLH: I²C0, I2C0SCLH—地址 0x4001 C010; I²C1, I2C1SCLH—地址 0x4005 C010; I²C2, I2C2SCLH—0x400A 0010)

位	符号	描述	复位值
15:0	SCLH	SCL 高电平时间周期选择的计数。	0x0004
31:16	-	保留。读取值未定义,只写入 0。	无

22.8.10 I²C SCL 低电平占空比寄存器 (I2SCLL: I²C0—I2C0SCLL: 0x4001 C014; I²C1—I2C1SCLL: 0x4005 C014; I²C2—I2C2SCLL: 0x400A 0014)

表506. I²C SCL 低电平占空比寄存器位描述 (I2SCLL: I²C0—I2C0SCLL: 0x4001 C014; I²C1—I2C1SCLL: 0x4005 C014; I²C2—I2C2SCLL: 0x400A 0014)

位	符号	描述	复位值
15:0	SCLL	SCL 低电平时间周期选择的计数。	0x0004
31:16	-	保留。读取值未定义，只写入 0。	无

22.8.11 选择合适的 I²C 数据速率和占空比

软件必须通过对 I2SCLH 和 I2SCLL 寄存器进行设置来选择合适的数据速率和占空比。I2SCLH 定义 SCL 高电平所保持的 PCLK 周期数，I2SCLL 定义 SCL 低电平的 PCLK 周期数。使用以下公式确定位频率（PCLK 是外设总线 APB 的频率）：

(13)

$$I^2C_{bitfrequency} = \frac{PCLKI^2C}{I2CSCLH + I2CSCLL}$$

I2SCLL 和 I2SCLH 的值必须确保得出的数据速率在 I²C 数据速率的范围之内。每个寄存器的值必须大于或等于 4。[表 507](#) 给出了一些根据 PCLK 频率和 I2SCLL 和 I2SCLH 值计算出来的 I²C 总线速率的示例。

表507. I²C 时钟速率的实例

I ² C 速率	PCLK 下 I2SCLL+I2SCLH 的值 (MHz)													
	6	8	10	12	16	20	30	40	50	60	70	80	90	100
100kHz (标准)	60	80	100	120	160	200	300	400	500	600	700	800	900	1000
400kHz (快速模式)	15	20	25	30	40	50	75	100	125	150	175	200	225	250
1MHz (增强型快速模式)	-	8	10	12	16	20	30	40	50	60	70	80	90	100

I2SCLL 和 I2SCLH 的值不一定要相同。通过设置这两个寄存器，软件可以为 SCL 设置不同的占空比。例如，I2C 总线规范定义在快速模式和增强型快速模式下的 SCL 低电平时间和高电平时间是不同的。

22.9 I²C 操作模式的详细信息

I²C 接口有 4 种操作模式：

- 主发送器模式
- 主接收器模式
- 从接收器模式
- 从发送器模式

每种操作模式的数据传输如[图 115](#)、[图 116](#)、[图 117](#)、[图 118](#)和[图 119](#)所示。[表 508](#)说明了在描述 I²C 操作模式时图中缩写的含义。

表508. 描述 I²C 操作的缩写

缩写	说明
S	起始条件
SLA	7 位从机地址
R	读数据位（SDA 为高电平）
W	写数据位（SDA 为低电平）
A	应答位（SDA 为低电平）
A	非应答位（SDA 为高电平）
Data	8 位数据字节
P	停止条件
Sr	重复起始条件

在[图 115](#)到[图 119](#)中，圆圈用于指示何时设置串行中断标志。圆圈中的数字表示 I2STAT 寄存器中的状态代码。每当出现这些代码时，必须执行服务程序来继续或结束串行传输。若串行传输被挂起，就不要考虑服务程序，直至串行中断标志被软件清除。

当进入串行中断程序时，I2STAT 的状态代码用于指向跳转到的相应的服务程序。对于每个状态代码所需的软件操作和后续串行传输的详细信息，参见[表 511~表 515](#)。

22.9.1 主发送器模式

在主发送器模式下，数据字节被发送到一个从接收器（参见图 115）。在进入主发送器模式之前，必须对 I2CON 进行如下初始化：

表509. 用于初始化主发送器模式的 I2CONSET

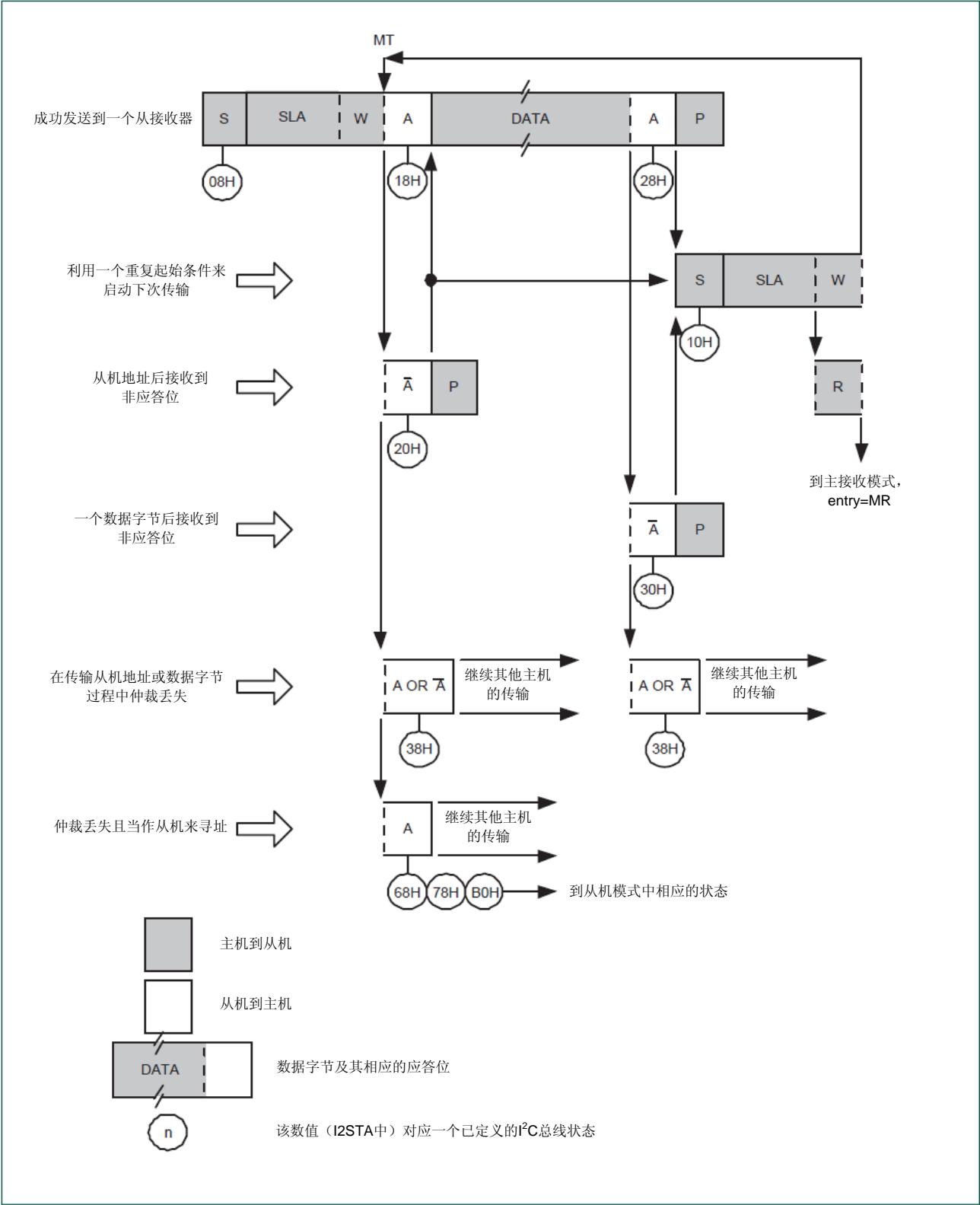
位	7	6	5	4	3	2	1	0
符号	-	I2EN	STA	STO	SI	AA	-	-
值	-	1	0	0	0	x	-	-

I²C 速率也必须在 I2SCLL 和 I2SCLH 寄存器中配置。I2EN 必须设置为逻辑 1 以使能 I²C 模块。如果 AA 位复位，当另一个设备正变成总线主机时，I²C 模块将不会应答其自身的从机地址或通用调用地址。换句话说，如果 AA 位复位，I²C 接口无法进入从机模式。STA、STO 和 SI 必须复位。

此时，可通过置位 STA 进入主发送器模式。一旦总线空闲，I²C 逻辑会马上测试 I²C 总线并产生一个起始条件。当发送一个起始条件时，串行中断标志（SI）置位，状态寄存器（I2STAT）中的状态代码为 0x08。中断服务程序利用该状态代码进入相应的状态服务程序，将从机地址和数据方向位（SLA+W）装入 I2DAT。I2CON 中的 SI 位必须在串行传输继续之前复位。

当发送完一个从机地址和方向位且接收到一个应答位时，串行中断标志（SI）再次置位，I2STAT 中可能是一系列不同的状态码。对于主机模式，状态代码是 0x18、0x20 或 0x38，如果已使能从机模式（AA=逻辑 1），状态代码则是 0x68、0x78 或 0xB0。每个状态代码执行的相应操作详见表 511。在发送完一个重复的起始条件（状态 0x10）后，I²C 模块通过将 SLA+R 装入 I2DAT 切换到主接收器模式。

图115. 主发送器模式中的格式和状态

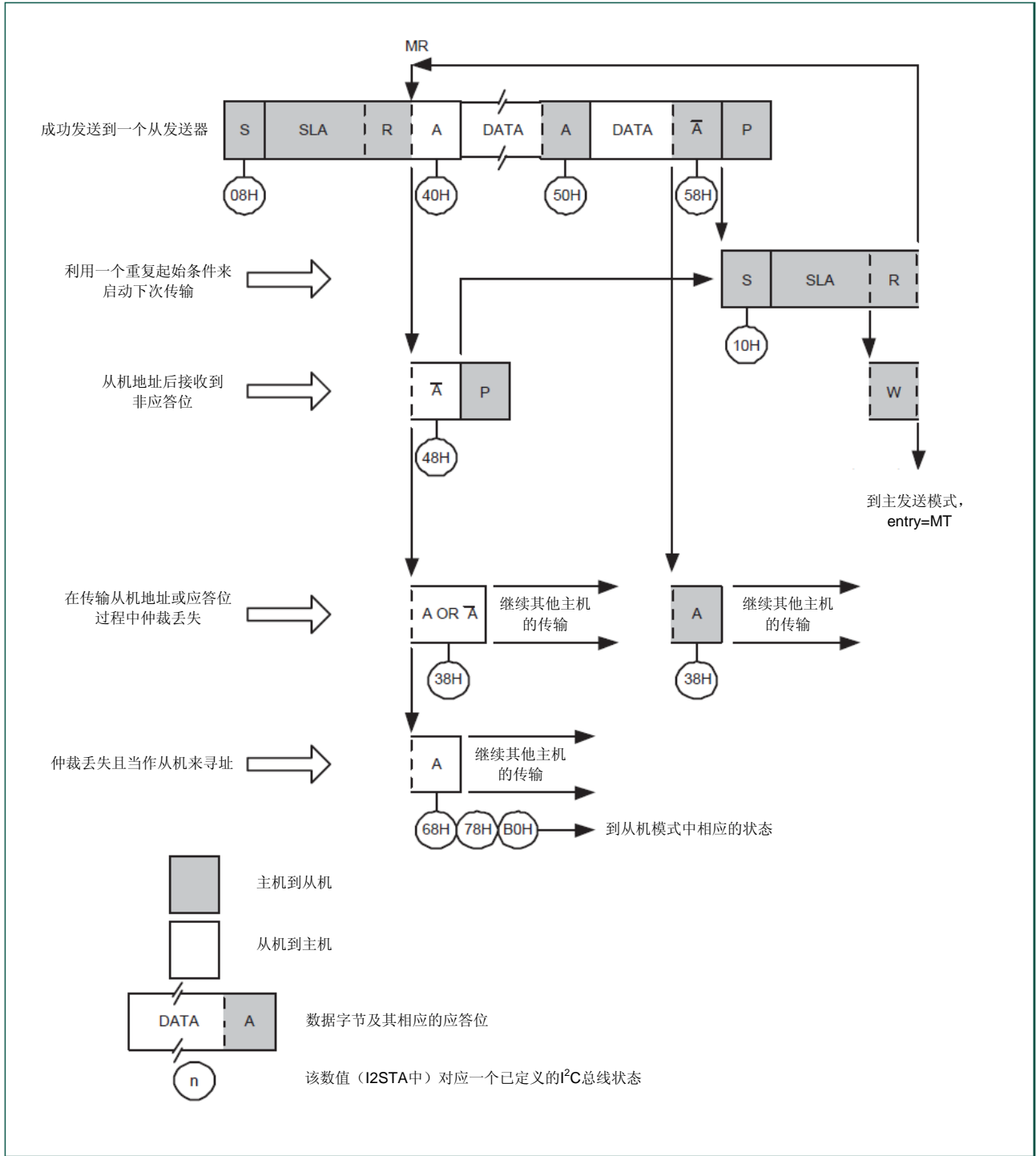


22.9.2 主接收器模式

在主接收器模式下，主机所接收的数据字节来自从发送器（见[图 116](#)）。传输的初始化过程与主发送器模式中一样。发送完起始条件后，中断服务程序必须把 7 位从机地址和数据方向位（SLA+R）装入 I2DAT。必须先清零 I2CON 中的 SI 位，再继续执行串行传输。

当发送完从机地址和数据方向位且接收到一个应答位时，串行中断标志（SI）再次置位，I2STAT 中可能是一系列不同的状态代码。对于主机模式，状态代码是 0x40、0x48 或 00x38，如果已使能从机模式（AA=1），则状态代码是 0x68、0x78 或 0xB0。每个状态代码执行的相应操作详见[表 512](#)。在发送完一个重复的起始条件（状态 0x10）后，I2C 模块通过将 SLA+W 装入 I2DAT 切换到主发送器模式。

图116. 主接收器模式中的格式和状态



22.9.3 从接收器模式

在从接收器模式下，从机接收的数据字节来自主发送器（见图 117）。要初始化从接收器模式，必须配置 I2CON 寄存器、I2ADR 寄存器和 I2MASK 寄存器。

四个 I2ADR 寄存器的值以及四个 I2MASK 寄存器的值决定了当使能了从机功能时，I²C 模块会响应哪一个或哪些地址。有关详细信息，参见第 22.7.2、22.7.3、22.8.7 和 22.8.8 节。

表510. 用于初始化从接收器模式的 I2CONSET

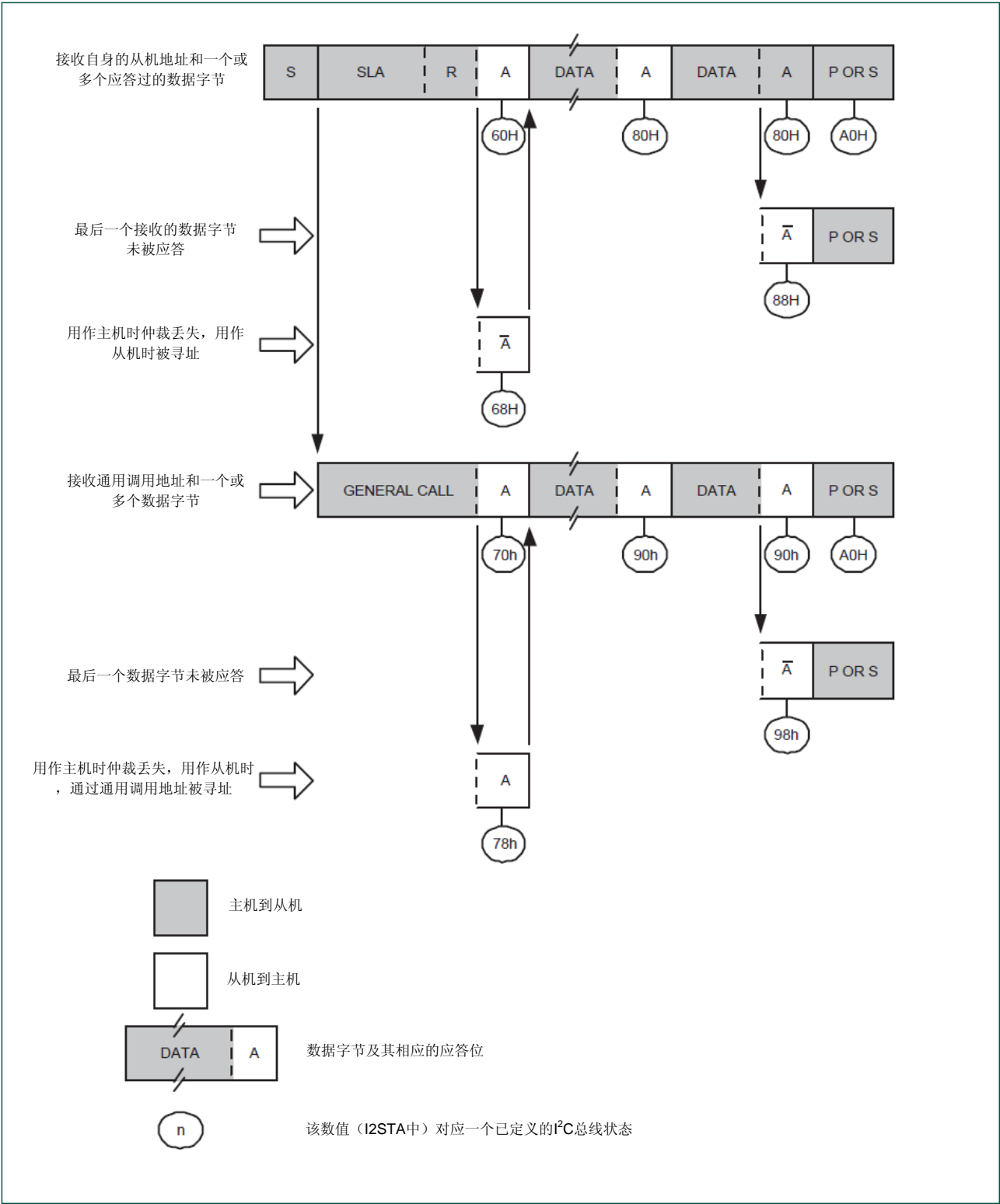
位	7	6	5	4	3	2	1	0
符号	-	I2EN	STA	STO	SI	AA	-	-
值	-	0	0	0	1		-	-

I²C 总线速率设置不影响从机模式下的 I²C 模块。I2EN 必须设置为逻辑 1 以使能 I²C 模块。AA 位必须置位以使能 I²C 模块来应答其自身的从机地址或通用调用地址。STA、STO 和 SI 必须复位。

当 I2ADR、I2MASK 和 I2CON 寄存器完成初始化之后，I²C 模块一直等待，直至被从机地址寻址，之后是数据方向位寻址，为了工作在从接收器模式中，数据方向位必须为“0”(W)。接收完其自身的从机地址和 W 位之后，串行中断标志（SI）置位，可以从 I2STAT 中读出一个有效的状态代码。该状态代码用作一个状态服务程序的向量。每个状态代码执行的相应操作详见表 513。如果 I²C 模块在主机模式中仲裁丢失，也可以进入从接收器模式（请参见状态 0x68 和 0x78 的描述）。

如果 AA 位在传输过程中复位，则在接收完下一个数据字节后，I²C 模块将向 SDA 返回一个非应答（逻辑 1）。当 AA 复位时，I²C 模块不响应其自身的从机地址或通用调用地址。但是，I²C 总线仍被监控，而且可随时通过置位 AA 来恢复地址识别。这意味着 AA 位可临时将 I²C 模块从 I²C 总线分离出来。

图117. 从接收器模式中的格式和状态

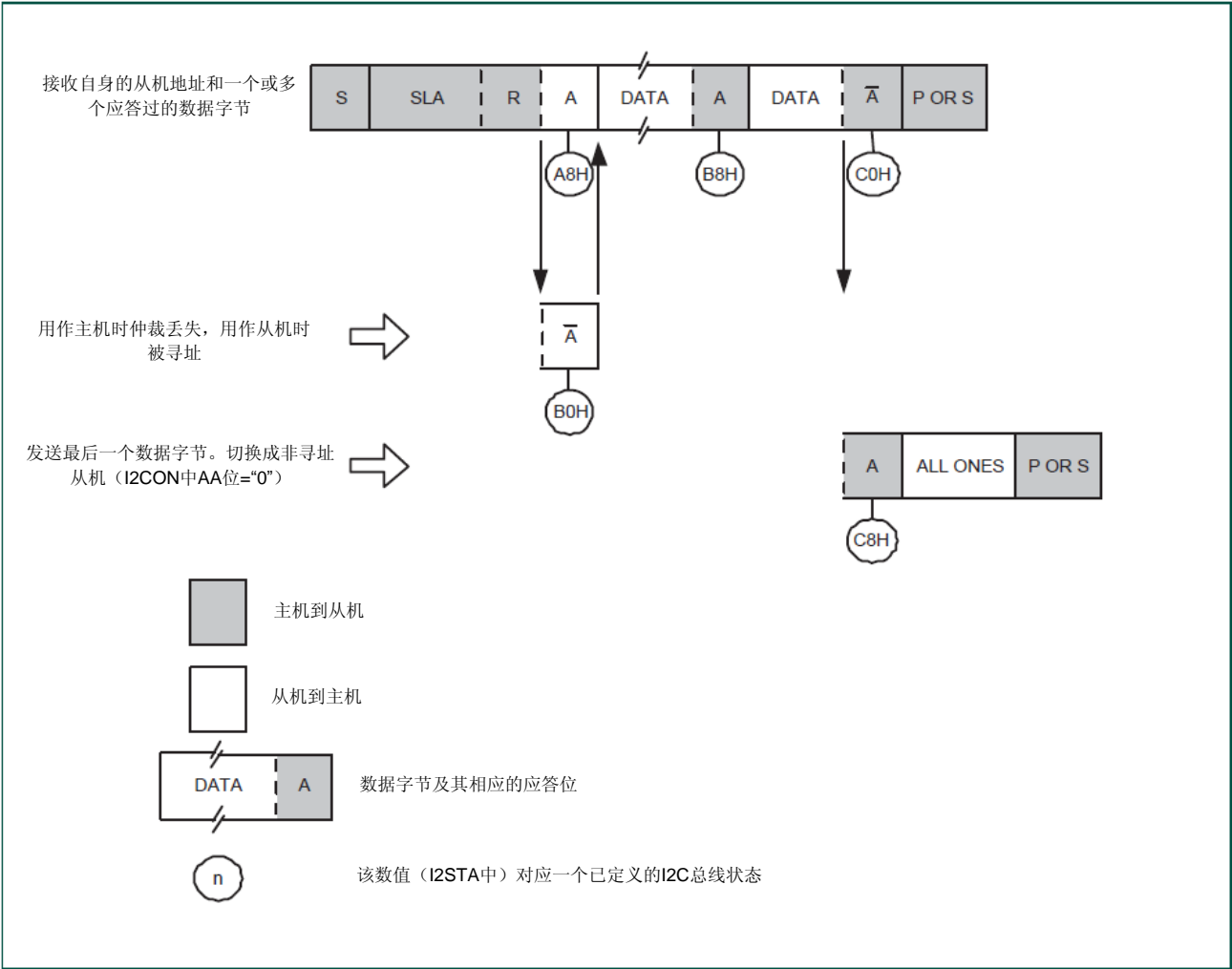


22.9.4 从发送器模式

在从发送器模式下，数据字节被发送到一个主接收器（见图 118）。数据传输的初始化过程与从接收器模式中一样。当 I2ADR 和 I2CON 寄存器完成初始化之后，I²C 模块一直等待，直到被其自身从机地址寻址，之后是数据方向位，该方向位必须为“1”(R)，以便 I²C 模块工作在从发送器模式下。接收完其自身的从机地址和 R 位之后，串行中断标志（SI）置位，并可从 I2STAT 中读取一个有效的状态代码。该状态代码用作一个状态服务程序的向量，每个状态代码执行的相应操作详见表 514。如果 I²C 模块在主机模式下时仲裁丢失，也可以进入从发送器模式（见状态 0xB0）。

如果 AA 位在一次传输过程中复位，I²C 模块将在发送完传输的最后一个字节后，进入状态 0xC0 或 0xC8。I²C 模块切换到非寻址的从机模式，如果继续传输，它将忽略主接收器。因此主接收器接收所有 1 作为串行数据。当 AA 复位时，I²C 模块不响应其自身的从机地址或通用调用地址。但是，I²C 总线仍被监控，而且可随时通过置位 AA 来恢复地址识别。这意味着 AA 位可用来暂时将 I²C 模块从 I²C 总线上分离出来。

图118. 从发送器模式中的格式和状态



22.9.5 详细的状态表

下面的表格给出了 I²C 四种操作模式的详细状态信息。

表511. 主发送器模式

状态代码 (I2CSTAT)	I ² C 总线和硬件的状态	应用软件的响应		I ² C 硬件执行的下一个动作				
		读/写 I2DAT	写 I2CON				AA	
			STA	STO	SI	AA		
0x08	已发送一个起始条件。	装入 SLA+W; 清除	STA	X	0	0	X	将发送 SLA+W; 接收 ACK 位。
0x10	已发送一个重复的起始条件。	装入 SLA+W 或		X	0	0	X	同上。
		装入 SLA+R; 清除	STA	X	0	0	X	将发送 SLA+W; I ² C 模块将切换为 MST/REC 模式。
0x18	已发送 SLA+W; 已接收 ACK。	装入数据字节或		0	0	0	X	将发送数据字节, 接收 ACK 位。
		无 I2DAT 动作或		1	0	0	X	将发送重复的起始条件。
		无 I2DAT 动作或		0	1	0	X	将发送停止条件; STO 标志将复位。
		无 I2DAT 动作		1	1	0	X	将发送停止条件, 然后发送起始条件; STO 标志将复位。
0x20	已发送 SLA+W; 已接收非 ACK。	装入数据字节或		0	0	0	X	将发送数据字节, 接收 ACK 位。
		无 I2DAT 动作或		1	0	0	X	将发送重复的起始条件。
		无 I2DAT 动作或		0	1	0	X	将发送停止条件; STO 标志将复位。
		无 I2DAT 动作		1	1	0	X	将发送停止条件, 然后发送起始条件; STO 标志将复位。
0x28	已发送 I2DAT 中的的数据字节; 已接收 ACK。	装入数据字节或		0	0	0	X	将发送数据字节, 接收 ACK 位。
		无 I2DAT 动作或		1	0	0	X	将发送重复的起始条件。
		无 I2DAT 动作或		0	1	0	X	将发送停止条件; STO 标志将复位。
		无 I2DAT 动作		1	1	0	X	将发送停止条件, 然后发送起始条件; STO 标志将复位。
0x30	已发送 I2DAT 中的的数据字节; 已接收非 ACK。	装入数据字节或		0	0	0	X	将发送数据字节, 接收 ACK 位。
		无 I2DAT 动作或		1	0	0	X	将发送重复的起始条件。
		无 I2DAT 动作或		0	1	0	X	将发送停止条件; STO 标志将复位。
		无 I2DAT 动作		1	1	0	X	将发送停止条件, 然后发送起始条件; STO 标志将复位。
0x38	在 SLA+R/W 或数据字节中丢失仲裁	无 I2DAT 动作或		0	0	0	X	I ² C 总线将被释放; 进入不可寻址从模式。
		无 I2DAT 动作		1	0	0	X	当总线空闲时发送起始条件。

表512. 主接收器模式

状态代码 (I2CSTAT)	I ² C 总线和硬件的状态	应用软件的响应		I ² C 硬件执行的下一个动作			
		读/写 I2DAT	写 I2CON				
			STA	STO	SI	AA	
0x08	已发送一个起始条件。	装入 SLA+R	X	0	0	X	将发送 SLA+R；接收 ACK 位。
0x10	已发送一个重复的起始条件。	装入 SLA+R 或	X	0	0	X	同上。
		装入 SLA+W	X	0	0	X	将发送 SLA+W；I2C 模块将切换为 MST/TRX 模式。
0x38	在非 ACK 位中丢失仲裁。	无 I2DAT 动作或	0	0	0	X	I ² C 总线将被释放；I ² C 模块将进入从模式。
		无 I2DAT 动作	1	0	0	X	当总线空闲时发送起始条件。
0x40	已发送 SLA+R；已接收 ACK。	无 I2DAT 动作或	0	0	0	0	将接收数据字节，返回非 ACK 位。
		无 I2DAT 动作	0	0	0	1	将接收数据字节，返回 ACK 位。
0x48	已发送 SLA+R；已接收非 ACK。	无 I2DAT 动作或	1	0	0	X	将发送重复的起始条件。
		无 I2DAT 动作或	0	1	0	X	将发送停止条件；STO 标志将复位。
		无 I2DAT 动作	1	1	0	X	将发送停止条件，然后发送起始条件；STO 标志将复位。
0x50	已接收数据字节；ACK 已返回。	读取数据字节或	0	0	0	0	将接收数据字节，返回非 ACK 位。
		读取数据字节	0	0	0	1	将接收数据字节，返回 ACK 位。
0x58	已接收数据字节，非 ACK 已返回。	读取数据字节或	1	0	0	X	将发送重复的起始条件。
		读取数据字节或	0	1	0	X	将发送停止条件；STO 标志将复位。
		读取数据字节	1	1	0	X	将发送停止条件，然后发送起始条件；STO 标志将复位。

表513. 从接收器模式

状态代码 (I2CSTAT)	I ² C 总线和硬件的状态	应用软件的响应		I ² C 硬件执行的下一个动作			
		读/写 I2DAT	写 I2CON	STA	STO	SI	AA
0x60	已接收自身的 SLA+W, 已返回 ACK。	无 I2DAT 动作或	X	0	0	0	将接收数据字节, 返回非 ACK 位。
		无 I2DAT 动作	X	0	0	1	将接收数据字节, 返回 ACK 位。
0x68	用作主机时在 SLA+R/W 中丢失仲裁; 已接收自身的 SLA+W, 已返回 ACK。	无 I2DAT 动作或	X	0	0	0	将接收数据字节, 返回非 ACK 位。
		无 I2DAT 动作	X	0	0	1	将接收数据字节, 返回 ACK 位。
0x70	已接收通用调用地址 (0x00); 已返回 ACK。	无 I2DAT 动作或	X	0	0	0	将接收数据字节, 返回非 ACK 位。
		无 I2DAT 动作	X	0	0	1	将接收数据字节, 返回 ACK 位。
0x78	用作主机时在 SLA+R/W 中丢失仲裁; 已接收通用调用地址, 已返回 ACK。	无 I2DAT 动作或	X	0	0	0	将接收数据字节, 返回非 ACK 位。
		无 I2DAT 动作	X	0	0	1	将接收数据字节, 返回 ACK 位。
0x80	前一次寻址使用自身 SLA 地址; 已接收数据字节, 已返回 ACK。	读取数据字节或	X	0	0	0	将接收数据字节, 返回非 ACK 位。
		读取数据字节	X	0	0	1	将接收数据字节, 返回 ACK 位。
0x88	前一次寻址使用自身 SLA; 已接收数据字节; 已返回非 ACK。	读取数据字节或	0	0	0	0	切换到不可寻址 SLV 模式; 不识别自身 SLA 以及通用调用地址。
		读取数据字节或	0	0	0	1	切换到不可寻址 SLV 模式; 识别自身 SLA; 如果 I2ADR[0]=逻辑 1, 将识别通用调用地址。
		读取数据字节或	1	0	0	0	切换到不可寻址 SLV 模式; 不识别自身 SLA 以及通用调用地址。当总线空闲时发送一个起始条件。
		读取数据字节	1	0	0	1	切换到不可寻址 SLV 模式; 识别自身 SLA; 如果 I2ADR[0]=逻辑 1, 将识别通用调用地址。当总线空闲时发送一个起始条件。
0x90	前一次寻址使用通用调用; 已接收数据字节, 已返回 ACK。	读取数据字节或	X	0	0	0	将接收数据字节, 返回非 ACK 位。
		读取数据字节	X	0	0	1	将接收数据字节, 返回 ACK 位。
0x98	前一次寻址使用通用调用; 已接收数据字节; 已返回非 ACK。	读取数据字节或	0	0	0	0	切换到不可寻址 SLV 模式; 不识别自身 SLA 以及通用调用地址。
		读取数据字节或	0	0	0	1	切换到不可寻址 SLV 模式; 识别自身 SLA; 如果 I2ADR[0]=逻辑 1, 将识别通用调用地址。
		读取数据字节或	1	0	0	0	切换到不可寻址 SLV 模式; 不识别自身 SLA 以及通用调用地址。当总线空闲时发送一个起始条件。
		读取数据字节	1	0	0	1	切换到不可寻址 SLV 模式; 识别自身 SLA; 如果 I2ADR[0]=逻辑 1, 将识别通用调用地址。当总线空闲时发送一个起始条件。
0xA0	当使用从接收器或从发送器进行静态寻址时, 接收到一个停止条件或重复的起	无 STDAT 动作或	0	0	0	0	切换到不可寻址 SLV 模式; 不识别自身 SLA 以及通用调用地址。
		无 STDAT 动作或	0	0	0	1	切换到不可寻址 SLV 模式; 识别自身 SLA; 如果 I2ADR[0]=逻辑 1,

状态代码 (I2CSTAT)	I ² C 总线和硬件的状态	应用软件的响应 读/写 I2DAT	写 I2CON				I ² C 硬件执行的下一个动作
			STA	STO	SI	AA	
	始条件。						将识别通用调用地址。
		无 STDAT 动作或	1	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 以及通用调用地址。当总线空闲时发送一个起始条件。
		无 STDAT 动作或	1	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=逻辑 1，将识别通用调用地址。当总线空闲时发送一个起始条件。

表514. 从发送器模式

状态代码 (I2CSTAT)	I ² C 总线和硬件的状态	应用软件的响应 读/写 I2DAT	写 I2CON				I ² C 硬件执行的下一个动作
			STA	STO	SI	AA	
0xA8	已接收自身的 SLA+R，已返回 ACK。	装入数据字节或	X	0	0	0	将发送最后一个数据字节，接收 ACK 位。
		装入数据字节	X	0	0	1	将发送数据字节，接收 ACK 位。
0xB0	用作主机时在 SLA+R/W 中丢失仲裁；已接收自身的 SLA+R，已返回 ACK。	装入数据字节或	X	0	0	0	将发送最后一个数据字节，接收 ACK 位。
		装入数据字节	X	0	0	1	将发送数据字节，接收 ACK 位。
0xB8	已发送 I2DAT 中的数据字节；已接收 ACK。	装入数据字节或	X	0	0	0	将发送最后一个数据字节，接收 ACK 位。
		装入数据字节	X	0	0	1	将发送数据字节，接收 ACK 位。
0xC0	已发送 I2DAT 中的数据字节；已接收非 ACK。	无 I2DAT 动作或	0	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 以及通用调用地址。
		无 I2DAT 动作或	0	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=逻辑 1，将识别通用调用地址。
		无 I2DAT 动作或	1	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 以及通用调用地址。当总线空闲时发送一个起始条件。
		无 I2DAT 动作	1	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=逻辑 1，将识别通用调用地址。当总线空闲时发送一个起始条件。
0xC8	最后一个 I2DAT 中的数据字节已被发送 (AA=0)；已接收 ACK。	无 I2DAT 动作或	0	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 以及通用调用地址。
		无 I2DAT 动作或	0	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=逻辑 1，将识别通用调用地址。
		无 I2DAT 动作或	1	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 以及通用调用地址。当总线空闲时发送一个起始条件。
		无 I2DAT 动作	1	0	0	01	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=逻辑 1，将识别通用调用地址。当总线空闲时发送一个起始条件。

22.9.6 其他状态

还有两种不对应已定义的 I²C 硬件状态的 I2STAT 代码（见[表 515](#)）。下面将对这两种代码进行讨论。

22.9.6.1 I2STAT=0xF8

该状态代码指示没有任何可用的相关信息，因为串行中断标志 SI 还没有置位。这种情况在其它状态和 I²C 模块还未开始执行串行传输之间出现。

22.9.6.2 I2STAT=0x00

该状态代码表示在一次 I²C 串行传输过程中发生了一个总线错误。如果在格式帧的非法位置上出现起始或停止条件，就会引起总线错误。这些非法位置是指在串行传输过程中的一个地址字节、数据字节或应答位。当外部干扰影响到 I²C 模块内部信号时也会引起总线错误。发生总线错误时，SI 置位。要从总线错误中恢复，STO 标志必须置位，SI 必须被清除。这会使 I²C 模块进入“非寻址的”从机模式（一个已定义的状态）并清除 STO 标志（不影响 I2CON 中的其他位）。SDA 和 SCL 线被释放（不发送停止条件）。

表515. 其他状态

状态代码 (I2CSTAT)	I ² C 总线和硬件的状态	应用软件的响应		I ² C 硬件执行的下一个动作			
		读/写 I2DAT	写 I2CON STA STO SI AA				
0xF8	无可用的相关状态信息；SI=0。	无 I2DAT 动作	无 I2DAT 动作				等待或执行当前传输。
0x00	由于非法起始或停止条件的出现，在 MST 或选择的从机模式中将出现总线错误。当外部干扰使 I ² C 模块进入未定义的状态时也出现 0x00 状态。	无 I2DAT 动作	0	1	0	X	只有 MST 或寻址的 SLV 模式中的内部硬件受影响。一般情况下，总线被释放、I ² C 模块切换到非寻址的 SLV 模式。STO 复位。

22.9.7 一些特殊情况

I²C 硬件可以处理以下几种串行传输过程中可能出现的特殊情况：

22.9.7.1 两个主机同时启动重复起始条件

在主发送器或主接收器模式下都可以生成重复的起始条件。如果此时另一个主机同时生成重复的起始条件，则将出现一种特殊情况（见[图 119](#)）。在这种情况下出现以前，任一主机都不会丢失仲裁，因为它们发送的数据相同。

如果 I²C 硬件在生成重复起始条件之前在 I²C 总线上检测到一个重复起始条件，它将释放总线，而且不生成中断请求。如果另一个主机通过生成一个停止条件来释放总线，则 I²C 模块将发送一个正常的起始条件（状态 0x08），并开始重新尝试一个完整的串行数据传输。

22.9.7.2 仲裁丢失后的数据传输

在主发送和主接收器模式下仲裁都有可能会丢失（见[图 113](#)）。I2STAT 中的下列状态代码

可指示仲裁丢失：0x38、0x68、0x78 和 0xB0（见[图 115](#)和[图 116](#)）。

如果 I2CON 中的 STA 标志被这些状态的服务程序置位，则当总线再次空闲时，会发起一个起始条件（状态 0x08），并且不受 CPU 的影响，开始重新尝试完整的串行数据传输。

22.9.7.3 强制访问 I2C 总线

在某些应用中，非控制源可能会导致总线挂起。在这种情况下，干扰、总线的暂时中断或 SDA 与 SCL 之间的暂时短路都会导致总线挂起。

如果非控制源产生了一个多余的起始条件或屏蔽了一个停止条件，则 I2C 总线一直保持忙碌状态。如果 STA 标志置位且在相应的时间内未访问总线，则 I2C 总线有可能会被强制访问。这可通过在 STA 标志仍被设置时置位 STO 标志来实现。不发送停止条件。I2C 的硬件操作就好像是接收到一个停止条件一样，可以发送起始条件。STO 标志由硬件清除（见[图 120](#)）。

22.9.7.4 SCL 或 SDA 低电平妨碍 I2C 总线的操作

如果 SDA 或 SCL 线被总线上任意一个设备拉低，I2C 总线就会挂起。如果 SCL 线被总线上的一个设备阻塞（拉低），不能继续串行传输，这时可通过拉低 SCL 线的设备来处理。

通常，SDA 线可能会被总线上另一个不和当前总线主机同步的设备拉低，不同步的原因可能是丢失了一个时钟周期或是以一个噪声脉冲作为时钟信号。在这种情况下，可通过向 SCL 线发送另外的时钟脉冲来处理（见[图 121](#)）。虽然 I2C 接口没有专门用来检测妨碍总线的超时定时器，但可以用系统中的其他定时器来完成。检测到总线挂起时，软件会强制给 SCL 最多九个时钟信号，直至 SDA 被设备释放。此时，从机仍可能不同步，所以还要发送一个起始条件以确保所有 I2C 外设都同步。

22.9.7.5 总线错误

如果在格式帧的某个非法位置上检测到了一个起始或停止条件，就会发生总线错误。非法位置是指串行传输过程中的一个地址字节、数据位或应答位。

仅当 I2C 硬件作为主机或被寻址的从机进行串行传输时，它才对总线错误有响应。检测到总线错误时，I2C 模块会立即切换到非寻址的从机模式，并释放 SDA 和 SCL 线，设置中断标志，并将 0x00 装入状态寄存器。这个状态代码可以用作指向一个状态服务程序的向量，尝试终止串行传输或从错误状况中恢复，如[表 515](#)所示。

图119. 两个主机同时发送重复起始条件

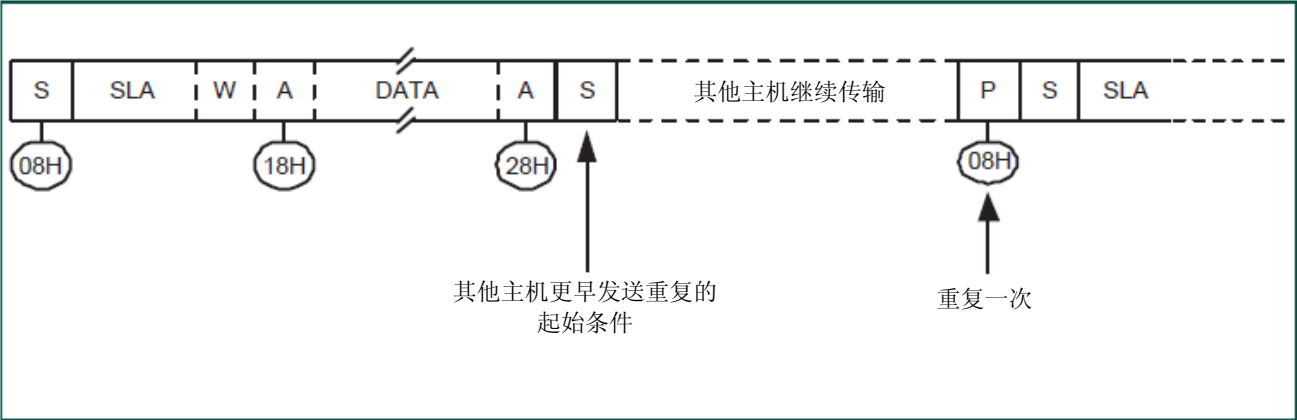


图120. 强制访问忙碌的 I²C 总线

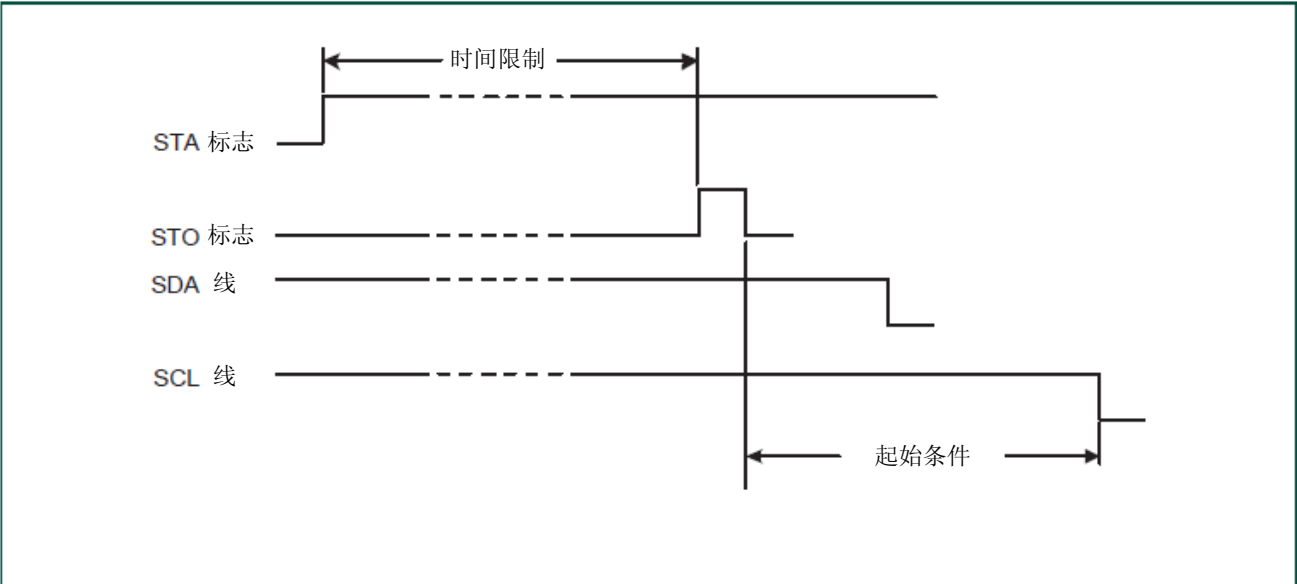
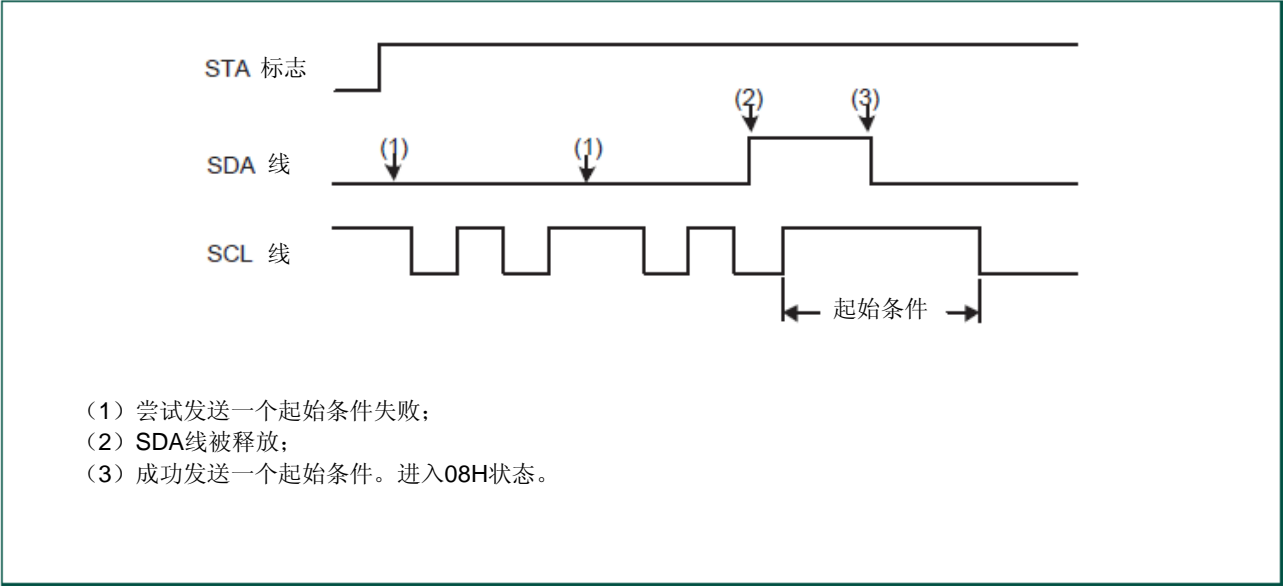


图121. 从总线干扰中恢复（由 SDA 低电平引起的）



22.9.8 I2C 状态服务程序

本节将举例说明不同 I2C 状态服务程序必须执行的操作。它们包括：

- 复位后 I2C 模块的初始化；
- I2C 中断服务；
- 支持四种 I2C 操作模式的 26 种状态服务程序。

22.9.8.1 初始化

在初始化示例中，I2C 模块可在主机模式和从机模式中使能。对于每种模式，缓冲区都可以用于发送和接收数据。初始化程序将执行以下操作：

- 向 I2ADR 和 I2MASK 寄存器中装入器件自身的从机地址和通用调用位（GC）；
- 置位 I2C 中断使能位和中断优先级位；
- 设置 I2CON 中的 I2EN 和 AA 位使能从机模式，向 I2SCLH 和 I2SCLL 寄存器中装入数据来定义串行时钟频率（主机模式）。主机程序必须从主程序中启动。

这时，I2C 硬件开始在 I2C 总线上检查其自身的从机地址和通用调用位。一旦检测到通用调用位或自身从机地址，则请求一个中断，并向 I2STAT 中装入相应的状态信息。

22.9.8.2 I2C 中断服务

当进入 I2C 中断时，I2STAT 存入一个状态代码，可识别要执行的 26 个状态服务中的其中一个。

22.9.8.3 状态服务程序

每个状态程序都是 I²C 中断程序的组成部分，分别用来处理 26 种状态。

22.9.8.4 实际应用中的状态服务

状态服务示例演示了响应 26 个 I²C 状态代码时必须执行的典型操作。如果未用到四种 I²C 操作模式中的一个或多个，则模式的相关的状态服务可被忽略，只要小心处理，那些状态就不会出现。

在应用中，可能需要在 I²C 操作过程中执行一些超时处理，来限制无效的总线或丢失的服务程序。

22.10 软件示例

22.10.1 初始化程序

将 I²C 接口初始化用作从机和/或主机的示例。

1. 将自身的从机地址装入 I2ADR 和 I2MASK 寄存器中，使能通用调用识别（如果需要的话）；
2. 使能 I²C 中断；
3. 将 0x44 写入 I2CONSET 以置位 I2EN 和 AA 位，并使能从机功能。对于主机功能，将 0x40 写入 I2CONSET。

22.10.2 启动主机发送功能

通过建立缓冲区、指针和数据计数然后发起一个起始条件便可执行主发送操作。

1. 初始化主机数据计数器。
2. 建立数据将被发送到的从机地址，并且添加写操作位。
3. 将 0x20 写入 I2CONSET 以置位 STA 位。
4. 在主发送缓冲区内建立要发送的数据。
5. 初始化主机数据计数器来匹配正在发送消息的长度。
6. 退出

22.10.3 启动主机接收功能

通过建立缓冲区、指针和数据计数然后发起一个起始条件便可执行主接收操作。

1. 初始化主机数据计数器。
2. 建立数据将被发送到的从机地址，并且添加读操作位。
3. 将 0x20 写入 I2CONSET 来置位 STA 位。
4. 建立主接收缓冲区。
5. 初始化主机数据计数器来匹配要接收消息的长度。
6. 退出

22.10.4 I²C 中断程序

确定 I²C 的状态和处理该状态的状态程序。

1. 从 I2STA 中读出 I²C 的状态。
2. 使用状态值跳转到 26 个可能状态程序中的一个。

22.10.5 无指定模式的状态

22.10.5.1 状态： 0x00

总线错误。进入非寻址的从机模式并释放总线。

1. 将 0x14 写入 I2CONSET 以置位 STO 和 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 退出

22.10.5.2 主机状态

状态 0x08 和状态 0x10 都用于主发送模式和主接收模式。 R/W 位决定了下一个状态是主发送模式还是主接收模式。

22.10.5.3 状态： 0x08

已发送了一个起始条件。 此时将发送从机地址和 R/W 位。

1. 将从机地址和 R/W 位写入 I2DAT。
2. 将 0x04 写入 I2CONSET 以置位 AA 位。
3. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
4. 建立主发送模式数据缓冲区。
5. 建立主接收模式数据缓冲区。
6. 初始化主机数据计数器。
7. 退出

22.10.5.4 状态： 0x10

已发送了一个重复的起始条件。 此时将发送从机地址和 R/W 位。

1. 将从机地址和 R/W 位写入 I2DAT。
2. 将 0x04 写入 I2CONSET 以置位 AA 位。
3. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
4. 建立主发送模式数据缓冲区。
5. 建立主接收模式数据缓冲区。
6. 初始化主机数据计数器。
7. 退出

22.10.6 主发送器状态

22.10.6.1 状态：0x18

前面的状态是状态 0x08 或状态 0x10 表示已发送从机地址和写操作位，已接收应答位。将发送第一个数据字节。

1. 将主机发送缓冲区的第一个数据字节装入 I2DAT。
2. 将 0x04 写入 I2CONSET 以置位 AA 位。
3. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
4. 主机发送缓冲区指针递增。
5. 退出

22.10.6.2 状态：0x20

已发送从机地址和写操作位并接收到非应答位。将发送一个停止条件。

1. 将 0x14 写入 I2CONSET 以置位 STO 和 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 退出

22.10.6.3 状态：0x28

已发送数据并接收到应答位。如果发送的数据是最后一个数据字节，则发送一个停止条件，否则发送下一个数据字节。

1. “主机数据计数器递减，如果发送的不是最后一个数据字节，则跳至到第 5 步。
2. 将 0x14 写入 I2CONSET 以置位 STO 和 AA 位。
3. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
4. 退出
5. 将主机发送缓冲区中的下一个数据字节装入 I2DAT 中。
6. 将 0x04 写入 I2CONSET 以置位 AA 位。
7. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
8. 主机发送缓冲区指针递增。
9. 退出。

22.10.6.4 状态：0x30

已发送数据并接收到非应答位。将发送一个停止条件。

1. 将 0x14 写入 I2CONSET 以置位 STO 和 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 退出。

22.10.6.5 状态： 0x38

仲裁已在从机地址和写操作或数据的过程中丢失。总线已被释放并进入非寻址的从机模式。当总线再次空闲时将发送一个新的起始条件。

1. 将 0x24 写入 I2CONSET 以置位 STA 和 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 退出。

22.10.7 主接收状态**22.10.7.1 状态： 0x40**

前面的状态是状态 08 或状态 10 表示已发送从机地址和读操作位，并接收到应答位。将接收数据并返回应答位。

1. 将 0x04 写入 I2CONSET 以置位 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 退出。

22.10.7.2 状态： 0x48

已发送从机地址和读操作位，并接收到非应答。将发送一个停止条件。

1. 将 0x14 写入 I2CONSET 以置位 STO 和 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 退出。

22.10.7.3 状态： 0x50

已接收数据，并返回应答位。将从 I2DAT 中读取数据。将接收其他的数据。如果这是最后一个数据字节，将返回非应答，否则将返回应答。

1. 读取 I2DAT 中的数据字节，存放到主机接收缓冲区。
2. 主机数据计数器递减，如果不是最后一个数据字节，则跳到第 5 步。
3. 将 0x0C 写入 I2CONCLR 以清除 SI 标志和 AA 位。
4. 退出。
5. 将 0x04 写入 I2CONSET 以置位 AA 位。
6. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
7. 主机接收缓冲区指针递增。
8. 退出。

22.10.7.4 状态： 0x58

已接收到数据，并已返回非应答。将从 I2DAT 中读取数据和发送一个停止条件。

1. 读取 I2DAT 中的数据字节，存放到主机接收缓冲区。
2. 将 0x14 写入 I2CONSET 以置位 STO 和 AA 位。
3. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
4. 退出。

22.10.8 从接收器状态**22.10.8.1 状态： 0x60**

已接收自身的从机地址和写操作位，并已返回应答。 将接收数据并返回应答。

1. 将 0x04 写入 I2CONSET 以置位 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 建立从接收模式数据缓冲区。
4. 初始化从机数据计数器。
5. 退出。

22.10.8.2 状态： 0x68

用作总线主机时，仲裁已在传输从机地址和 R/W 位时丢失。已接收到自身从机地址和写操作位，并已返回应答。将接收数据并返回应答。当总线再次空闲后，置位 STA 以重新启动主机模式。

1. 将 0x24 写入 I2CONSET 以置位 STA 和 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 建立从机接收模式数据缓冲区。
4. 初始化从机数据计数器。
5. 退出。

22.10.8.3 状态： 0x70

已接收通用调用并返回应答。将接收数据并返回应答。

1. 将 0x04 写入 I2CONSET 以置位 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 建立从机接收模式数据缓冲区。
4. 初始化从机数据计数器。
5. 退出。

22.10.8.4 状态： 0x78

用作总线主机时，仲裁已在传输从机地址和 R/W 位时丢失。已接收通用调用并返回应答。将接收数据并返回应答。当总线再次空闲后，置位 STA 以重新启动主机模式。

1. 将 0x24 写入 I2CONSET 以设置 STA 和 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 建立从机接收模式数据缓冲区。
4. 初始化从机数据计数器。
5. 退出。

22.10.8.5 状态： 0x80

之前寻址自身从机地址。已接收数据并返回应答。将读取其它数据。

1. 读取 I2DAT 中的数据字节，存放到从机接收缓冲区。
2. 从机数据计数器递减，如果不是最后一个数据字节，则跳至第 5 步。
3. 将 0x0C 写入 I2CONCLR 以清除 SI 标志和 AA 位。
4. 退出。
5. 将 0x04 写入 I2CONSET 以置位 AA 位。
6. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
7. 从机接收缓冲区指针递增。
8. 退出。

22.10.8.6 状态： 0x88

之前寻址自身从机地址。已接收数据并返回非应答。将不保存接收到的数据。进入非寻址的从机模式。

1. 将 0x04 写入 I2CONSET 以置位 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 退出。

22.10.8.7 状态： 0x90

之前寻址通用调用地址。已接收数据并返回应答。将保存接收到的数据。只接收第一个数据字节并返回应答。接收其它数据后返回非应答。

1. 读取 I2DAT 中的数据字节，存放到从机接收缓冲区。
2. 将 0x0C 写入 I2CONCLR 以清除 SI 标志和 AA 位。
3. 退出。

22.10.8.8 状态： 0x98

之前寻址通用调用地址。已接收数据并返回非应答。将不保存接收到的数据。进入非寻址的从机模式。

1. 将 0x04 写入 I2CONSET 以置位 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 退出。

22.10.8.9 状态： 0xA0

接收到一个停止条件或重复的起始条件，但仍然作为一个从机寻址。将不保存数据。进入非寻址的从机模式。

1. 将 0x04 写入 I2CONSET 以置位 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 退出。

22.10.9 从发送器状态**22.10.9.1 状态： 0xA8**

已接收自身从机地址和读操作位并返回应答。将发送数据和接收应答位。

1. 将从机发送缓冲区中的第一个数据字节装入 I2DAT。
2. 将 0x04 写入 I2CONSET 以置位 AA 位。
3. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
4. 建立从机发送模式数据缓冲区。
5. 从机发送缓冲区指针递增。
6. 退出。

22.10.9.2 状态： 0xB0

用作总线主机时，仲裁在传输从机地址和 R/W 位时丢失。已接收自身的从机地址和读操作位并返回应答。将发送数据和接收应答位。当总线再次空闲后，置位 STA 以重新启动主机模式。

1. 将从机发送缓冲区中的第一个数据字节装入 I2DAT 中。
2. 将 0x24 写入 I2CONSET 以置位 STA 和 AA 位。
3. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
4. 建立从机发送模式数据缓冲区。
5. 从机发送缓冲区指针递增。
6. 退出。

22.10.9.3 状态： 0xB8

已发送数据并接收到应答。将发送数据和接收应答位。

1. 将从机发送缓冲区中的数据字节装入 I2DAT。
2. 将 0x04 写入 I2CONSET 以置位 AA 位。
3. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
4. 从机发送缓冲区指针递增。
5. 退出

22.10.9.4 状态： 0xC0

已发送数据并接收到非应答。 进入非寻址的从机模式。

1. 将 0x04 写入 I2CONSET 以置位 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 退出。

22.10.9.5 状态： 0xC8

已发送最后一个数据字节并接收到应答。 进入非寻址的从机模式。

1. 将 0x04 写入 I2CONSET 以置位 AA 位。
2. 将 0x08 写入 I2CONCLR 以清除 SI 标志。
3. 退出。

23.1 基本配置

使用下列寄存器来配置 I²S 接口：

1. 功率： 在 PCONP 寄存器（[表 37](#)）中置位 PCI2S。
注： 复位时，I²S 接口会被禁能（PCI2S=0）。
2. 外设时钟： I²S 接口的功能部分使用 CPU 时钟（CCLK）来运行，而非 PCLK。总线接口使用 APB 外设的通用 PCLK 来运行。 参见 [4.6.4](#) 节。
3. 管脚： 在相关的 IOCON 寄存器（参见 [8.4.1](#) 节）中选择 I²S 管脚及其模式。
4. 利用相应的中断置位使能寄存器使能 NVIC 中的中断。
5. DMA： I²S 接口支持两种 DMA 请求，参见 [表 523](#)、[表 524](#) 和 [表 664](#)。

23.2 特性

I²S 总线为数字音频应用提供了一个标准的通信接口。 I²S 总线规范定义 I²S 接口为一条 3 线串行总线，包含一根数据线、一根时钟线和一根字选择信号线。 I²S 连接包括一个主机（始终作为主机）和一个从机。 LPC178x/177x 系列的 I²S 接口提供了彼此独立的发送和接收通道，每个通道都可以作为主机或从机，还提供了可选的过采样主机时钟输出（MCLK）。

- I²S 输入通道可在主机和从机模式下工作。
- I²S 输出通道可在主机和从机模式下工作而与输入通道无关。
- 可处理 8、16 和 32 位大小的数据字。
- 支持单声道和立体声道音频数据的传输。
- 多种时钟，包括独立的发送和接收分数速率发生器，而且可以为 4 线模式使用单独时钟输入或输出。
- 支持采样频率（fs）范围为 16kHz~96kHz（16、22.05、32、44.1、48 或 96kHz）或更高的音频应用，采样频率范围取决于时钟频率。
- 发送通道和接收通道具有独立的主机时钟输出，支持高达 I²S 采样频率 512 倍的时钟。
- 在主机模式下，字选择周期可配置（I²S 输入和输出各自独立配置）。
- 提供两个 8 字（32 字节）的 FIFO 数据缓冲区，一个用于发送，另一个用于接收。
- 当缓冲区深度超过预设触发深度（可编程的边界）时产生中断请求。
- 两个 DMA 请求由可编程的缓冲区深度控制。这两个 DMA 请求被连接到通用 DMA 模块。
- 可对 I²S 输入和 I²S 输出通道分别执行复位、停止和静音等控制操作。
- 可选的 MCLK（过采样）输出。

23.3 描述

I²S 通过发送通道执行串行数据输出，通过接收通道执行串行数据输入。在单声道和立体声道音频数据传输中，这两个通道都支持 8、16 和 32 位的 NXP Inter IC 格式的音频数据。配置、数据访问和控制由 APB 寄存器集来执行。数据流通过 8 个字深度的 FIFO 进行缓冲。

在从机模式或主机模式下，I²S 接收级和发送级可单独操作。在 I²S 模块中，这些模式间的差别在于决定数据发送时序的字选择 (WS) 信号。在 WS 改变后，数据字在发送时钟的下一个下降沿上开始。在立体声道模式下，当 WS 为低电平时发送左声道数据，当 WS 为高电平时发送右声道数据。在单声道模式下，相同的数据会被发送两次，WS 为低电平和高电平时各发送一次。

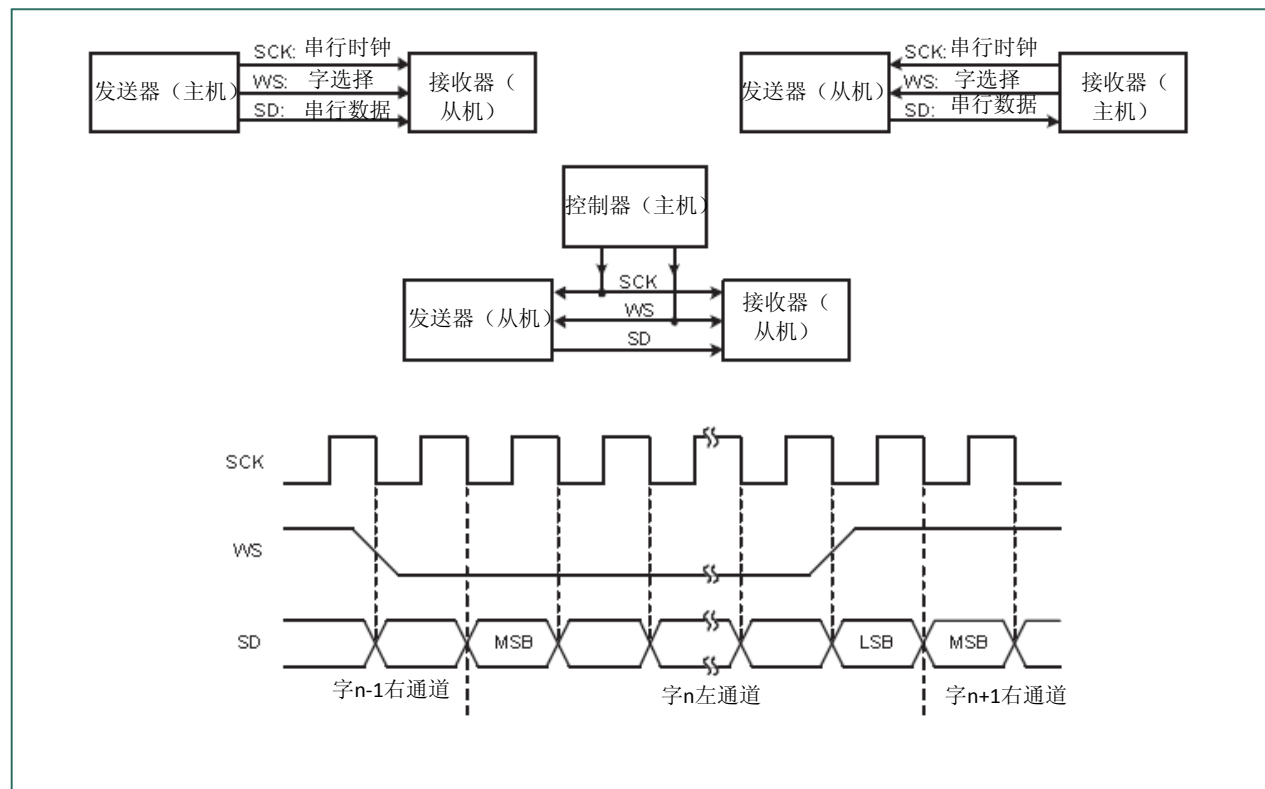
- 在主机模式下，字选择通过 9 位计数器在内部执行。可以在控制寄存器中设置该计数器的半周期计数值。
- 在从机模式下，字选择从相关的总线管脚输入。
- 当 I²S 总线有效时，总线主机连续发送字选择信号、接收时钟和发送时钟信号，而发送器会连续发送数据。
- 可分别通过发送和接收通道的停止或静音控制位来禁能 I²S。
- 停止位将禁止通过发送通道或接收通道访问 FIFO，并把发送通道置于静音模式下。
- 静音控制位将发送通道置于静音模式下。在静音模式下，发送通道 FIFO 正常操作，但输出被废弃并由 0 替代。该位不影响接收通道，可以正常进行数据接收。

23.4 管脚描述

表516. 管脚描述

管脚名称	类型	描述
I2S_RX_CLK	输入/输出	接收时钟。用于同步接收通道上的数据传输。由主机驱动从机接收。与 I ² S 总线规范中的信号 SCK 对应。
I2S_RX_WS	输入/输出	接收字选择。选择接收数据的通道。由主机驱动从机接收。与 I ² S 总线规范中的信号 WS 对应。 WS=0 表示通道 1（左通道）正在接收数据。 WS=1 表示通道 2（右通道）正在接收数据。
I2S_RX_SDA	输入/输出	接收数据。串行数据，先接收 MSB。由发送器驱动接收器读取。与 I ² S 总线规范中的信号 SD 对应。
I2S_RX_MCLK	输出	用于 I ² S 接收功能的可选主机时钟输出。
I2S_TX_CLK	输入/输出	发送时钟。用于同步发送通道上的数据传输。由主机驱动从机接收。与 I ² S 总线规范中的信号 SCK 对应。
I2S_TX_WS	输入/输出	发送字选择。选择发送数据的通道。由主机驱动从机接收。与 I ² S 总线规范中的信号 WS 对应。 WS=0 表示数据正被发送到通道 1（左通道）。 WS=1 表示数据正被发送到通道 2（右通道）。
I2S_TX_SDA	输入/输出	发送数据。串行数据，先发送 MSB。由发送器驱动接收器读取。与 I ² S 总线规范中的信号 SD 对应。
I2S_TX_MCLK	输出	用于 I ² S 发送功能的可选主机时钟输出。

图122. 简单的 I²S 配置和总线时序



23.5 寄存器描述

表 517 所示为与 I²S 接口相关的寄存器及其功能汇总。之后会对每个寄存器进行详细描述。

表 517. I²S 寄存器映射

名称	描述	访问	复位值 ^[1]	地址	表
I2SDAO	数字音频输出寄存器。含有 I ² S 发送通道的控制位。	R/W	0x87E1	0x400A 8000	表 518
I2SDAI	数字音频输入寄存器。含有 I ² S 接收通道的控制位。	R/W	0x07E1	0x400A 8004	表 519
I2STXFIFO	发送 FIFO。访问 8x32 位发送 FIFO 的寄存器。	WO	0	0x400A 8008	表 520
I2SRXFIFO	接收 FIFO。访问 8x32 位接收 FIFO 的寄存器。	RO	0	0x400A 800C	表 521
I2SSTATE	状态反馈寄存器。含有 I ² S 接口的状态信息。	RO	0x7	0x400A 8010	表 522
I2SDMA1	DMA 配置寄存器 1。含有 DMA 请求 1 的控制信息。	R/W	0	0x400A 8014	表 523
I2SDMA2	DMA 配置寄存器 2。含有 DMA 请求 2 的控制信息。	R/W	0	0x400A 8018	表 524
I2SIRQ	中断请求控制寄存器。含有控制如何产生 I ² S 中断请求的位。	R/W	0	0x400A 801C	表 525
I2STXRATE	发送参考时钟分频器。该寄存器确定 I ² S TX_REF 速率，通过指定 CCLK 的分频值来实现（以便产生 TX_REF）。	R/W	0	0x400A 8020	表 526
I2SRXRATE	接收参考时钟分频器。该寄存器确定 I ² S RX_REF 速率，通过指定 CCLK 的分频值来实现（以便产生 RX_REF）。	R/W	0	0x400A 8024	表 527
I2STXBITRATE	发送位速率分频器。该寄存器确定 I ² S 发送位速率，通过指定 TX_REF 的分频值来实现（以便产生发送位时钟）。	R/W	0	0x400A 8028	表 528
I2SRXBITRATE	接收位速率分频器。该寄存器确定 I ² S 接收位速率，通过指定 RX_REF 的分频值来实现（以便产生接收位时钟）。	R/W	0	0x400A 802C	表 529
I2STXMODE	发送模式控制。	R/W	0	0x400A 8030	表 530
I2SRXMODE	接收模式控制。	R/W	0	0x400A 8034	表 531

[1] 复位值仅反映已使用位中保存的数据， 不包括保留位的内容。

23.5.1 数字音频输出寄存器（I2SDAO—0x400A 8000）

I2SDAO 寄存器控制 I²S 发送通道的操作。 DAO 中各个位的功能如表 518 所示。

表 518. 数字音频输出寄存器位描述（I2SDAO—地址 0x400A 8000）

位	符号	值	描述	复位值
1:0	wordwidth		选择需发送数据字的宽度：	01
		00	8 位数据	
		01	16 位数据	
		10	保留，不使用该设置。	
		11	32 位数据	
2	mono		1：单声道模式； 0：立体声模式	0
3	stop		1：禁止访问 FIFO，发送通道被静音	0
4	reset		1：异步复位发送通道和发送 FIFO	0
5	ws_sel		0：接口处于主机模式 1：接口处于从机模式 关于 I2STXMODE 与该位的有用的组合1 的汇总，参见 23.7 节。	
14:6	ws_halfperiod		其值为采样周期的一半再减 1，即，WS 64clk period -> ws_halfperiod = 31。	0x1F
15	mute		当该位为 1 时，发送通道仅发送 0	1
31:16	-		保留。 读取值未定义，只写入 0。	无

23.5.2 数字音频输入寄存器（I2SDAI—0x400A 8004）

I2SDAI 寄存器控制 I²S 接收通道的操作。 DAI 中各个位的功能如[表 519](#)所示。

表519. 数字音频输入寄存器位描述（I2SDAI—地址 0x400A 8004）

位	符号	值	描述	复位值
1:0	wordwidth		选择需接收数据字的宽度：	01
		00	8 位数据	
		01	16 位数据	
		10	保留，不使用该设置。	
		11	32 位数据	
2	mono		1：单声道模式； 0：立体声模式	0
3	stop		1：禁止访问 FIFO，发送通道被静音	0
4	reset		1：异步复位发送通道和发送 FIFO	0
5	ws_sel		0：接口处于主机模式 1：接口处于从机模式 关于 I2SRXMODE 与该位的有用的组合1 的汇总，参见 23.7 节。	
14:6	ws_halfperiod		其值为采样周期的一半再减 1，即，WS 64clk period -> ws_halfperiod = 31。	0x1F
31:15	-		保留。 读取值未定义，只写入 0。	无

23.5.3 发送FIFO 寄存器（I2STXFIFO—0x400A 8008）

通过 I2STXFIFO 寄存器访问发送 FIFO。 I2STXFIFO 中各个位的功能如[表 520](#)所示。

表520. 发送 FIFO 寄存器位描述（I2STXFIFO—地址 0x400A 8008）

位	符号	描述	复位值
31:0	I2STXFIFO	8x32 位发送 FIFO。	Level = 0

23.5.4 接收FIFO 寄存器（I2SRXFIFO—0x400A 800C）

通过 I2SRXFIFO 寄存器访问接收 FIFO。 I2SRXFIFO 中各个位的功能如[表 521](#)所示。

表521. 接收 FIFO 寄存器位描述（I2RXFIFO—地址 0x400A 800C）

位	符号	描述	复位值
31:0	I2SRXFIFO	8x32 位接收 FIFO。	level = 0

23.5.5 状态反馈寄存器（I2SSSTATE—0x400A 8010）

I2SSSTATE 寄存器提供有关 I²S 接口的状态信息。 I2SSSTATE 中各个位的描述如[表 522](#)所示。

表522. 状态反馈寄存器位描述（I2SSSTATE—地址 0x400A 8010）

位	符号	描述	复位值
0	irq	该位反映了出现接收中断或发送中断。 该位的值由当前 FIFO 的深度与 I2SIRQ 寄存器中 rx_depth_irq 和 tx_depth_irq 字段的值进行比较来确定。	1
1	dmareq1	该位反映了出现接收或发送 DMA 请求 1。该位通过比较当前 FIFO 深度与 I2SDMA1 寄存器中的 rx_depth_dma1 和 tx_depth_dma1 字段来确定。	1
2	dmareq2	该位反映了出现接收或发送 DMA 请求 2。该位通过比较当前 FIFO 深度与 I2SDMA2 寄存器中的 rx_depth_dma2 和 tx_depth_dma2 字段来确定。	1

位	符号	描述	复位值
7:3	Unused	未使用。	0
11:8	rx_level	反映了接收 FIFO 的当前深度。	0
15:12	-	保留。读取值未定义，只写入 0。	无
19:16	tx_level	反映了发送 FIFO 的当前深度。	0
31:20	-	保留。读取值未定义，只写入 0。	无

23.5.6 DMA 配置寄存器 1（I2SDMA1—0x400A 8014）

I2SDMA1 寄存器控制 DMA 请求 1 的操作。I2SDMA1 中各个位的功能如表 523 中所示。有关 DMA 操作的详细信息，参见“通用 DMA 控制器”一章。

表523. DMA 配置寄存器 1 位描述(I2SDMA1—地址 0x400A 8014)

位	符号	描述	复位值
0	rx_dma1_enable	该位为 1 时，使能 DMA1 用于 I ² S 接收。	0
1	tx_dma1_enable	该位为 1 时，使能 DMA1 用于 I ² S 发送。	0
7:2	-	保留。读取值未定义，只写入 0。	0
11:8	rx_depth_dma1	设置在 DMA1 上触发一个接收 DMA 请求的 FIFO 深度。	0
15:12	-	保留。读取值未定义，只写入 0。	无
19:16	tx_depth_dma1	设置在 DMA1 上触发一个发送 DMA 请求的 FIFO 深度。	0
31:20	-	保留。读取值未定义，只写入 0。	无

23.5.7 DMA 配置寄存器 2（I2SDMA2—0x400A 8018）

I2SDMA2 寄存器控制 DMA 请求 2 的操作。I2SDMA2 中各个位的功能如表 518 中所示。

表524. DMA 配置寄存器 2 位描述(I2SDMA2—地址 0x400A 8018)

位	符号	描述	复位值
0	rx_dma2_enable	该位为 1 时，使能 DMA2 用于 I ² S 接收。	0
1	tx_dma2_enable	该位为 1 时，使能 DMA2 用于 I ² S 发送。	0
7:2	Unused	未使用。	0
11:8	rx_depth_dma2	设置在 DMA2 上触发一个接收 DMA 请求的 FIFO 深度。	0
15:12	-	保留。读取值未定义，只写入 0。	无
19:16	tx_depth_dma2	设置在 DMA2 上触发一个发送 DMA 请求的 FIFO 深度。	0
31:20	-	保留。读取值未定义，只写入 0。	无

23.5.8 中断请求控制寄存器（I2SIRQ—0x400A 801C）

I2SIRQ 寄存器控制 I²S 中断请求的操作。I2SIRQ 各个位的功能如表 518 中所示。

表525. 中断请求控制寄存器位描述（I2SIRQ—地址 0x400A 801C）

位	符号	描述	复位值
0	rx_irq_enable	该位为 0 时，使能 I ² S 接收中断。	0
1	tx_irq_enable	该位为 1 时，使能 I ² S 发送中断。	0
7:2	Unused	未使用。	0
11:8	rx_depth_irq	设置产生一个中断请求的接收 FIFO 深度。	0
15:12	-	保留。读取值未定义，只写入 0。	无
19:16	tx_depth_irq	设置产生一个中断请求的发送 FIFO 深度。	0
31:20	-	保留。读取值未定义，只写入 0。	无

23.5.9 发送时钟速率寄存器（I2STXRATE—0x400A 8020）

I²S 发送器的 TX_REF 速率由 I2STXRATE 寄存器的值决定。所需的 I2STXRATE 的设置取决于所需的音频采样率以及使用的数据格式（立体/单声道）和数据大小。当为发送功能使能 MCLK 输出时，会将 TX_REF 发送到 I2S_TX_MCLK 管脚。

TX_REF 速率是使用分数速率发生器除以 CCLK 频率生成的。必须选择分子(X)和分母(Y)的值，以便生成一个为 TX_REF 所需频率两倍的频率，该频率必须是发送器位时钟率的整数倍数。在选择设置时，用户应了解有关分数速率发生器的一些信息，参见 23.5.9.1 节。分数速率发生器使用的等式如下：

I2S TX_REF=CCLKx(X/Y)/2

注：如果 X 或 Y 的值为 0，则时钟分频器会被旁路。另外，Y 值必须大于或等于 X 值。

表526. 发送时钟速率寄存器位描述（I2TXRATE—地址 0x400A 8020）

位	符号	描述	复位值
7:0	Y_divider	I ² S 发送 TX_REF 速率的分母。CCLK 除以该值来获取 TX_REF。8 位分频支持较大的速率范围。该值为 0 时时钟分频器被旁路。	0
15:8	X_divider	I ² S 发送 TX_REF 速率的分子。CCLK 乘以该值来获取 TX_REF。该值为 0 时时钟分频器被旁路。8 位分频支持较大的速率范围。注：计算速率时 X/Y 必须除以 2。	0
31:16	-	保留。读取值未定义，只写入 0。	无

23.5.9.1 有关分数速率发生器的说明

分数速率发生器的性质决定了使用某些分频设置时会存在一定的输出抖动。这是因为分数速率发生器是一个全数字功能发生器，所以输出时钟的变换与源时钟保持同步，而理论上的理想分数速率可能存在与源时钟无关的沿。因此，在连续时钟沿之间，输出抖动不会大于加上或减去一个源时钟的值。

例如，如果 X=0x07 而 Y=0x11，则分数速率发生器每隔 17（十六进制数 11）个输入时钟会输出 7 个时钟，并尽可能地进行平均分布。在该示例中，无法做到绝对平均地分配输出时钟，所以一些时钟会比其他时钟长。将输出时钟除以 2 来均分，这也有助于减少抖动。频率平均值可精确到 (7/17) /2，但一些时钟的长度会比其相邻时钟略长。在选择分数时，如果使 Y 能够被 X 整除，例如 2/4、2/6、3/9、1/N 等等，则可以完全避免抖动。

23.5.10 接收时钟速率寄存器（I2SRXRATE—0x400A 8024）

I²S 接收器的 RX_REF 速率由 I2SRXRATE 寄存器的值决定。所需的 I2SRXRATE 设置取决于 CPU 时钟速率（CCLK）和需要的 RX_REF 速率（如 256fs）。当为接收功能使能 MCLK 输出时，会将 RX_REF 发送到 I2S_RX_MCLK 管脚。

使用分数速率发生器除以 CCLK 频率来生成 RX_REF 速率。必须选择分子（X）和分母（Y）的值，以便生成一个为 RX_REF 所需频率两倍的频率，该频率必须是接收器位时钟速率的整数倍数。在选择设置时，用户应了解有关分数速率发生器的一些信息，参见 23.5.9.1 节。分数速率发生器的等式如下：I2S RX_REF=CCLK×(X/Y)/2

注：如果 X 或 Y 的值为 0，则时钟分频器被旁路。另外，Y 的值必须大于或等于 X 值。

表527. 接收时钟速率寄存器位描述（I2SRXRATE—地址 0x400A 8024）

位	符号	描述	复位值
7:0	Y_divider	I ² S 接收 RX_REF 速率的分母。CCLK 除以该值来获取 RX_REF。8 位分频支持较大的速率范围。该值为 0 时时钟分频器被旁路。	0
15:8	X_divider	I ² S 接收 RX_REF 速率的分子。该值乘以 CCLK 来获取 RX_REF。该值为 0 时时钟分频器被旁路。8 位分频支持较大的速率范围。注：计算速率时 X/Y 必须除以 2。	0
31:16	-	保留。读取值未定义，只写入 0。	无

23.5.11 发送时钟位速率寄存器 (I2STXBITRATE—0x400A 8028)

I²S 发送器的位速率由 I2STXBITRATE 寄存器的值决定。该值取决于所需的音频采样速率以及使用的数据格式（立体/单声道）和数据大小。例如，48kHz 采样率的 16 位数据立体声道需要的位速率为：48,000×16×2=1.536MHz。

表528. 发送时钟位速率寄存器位描述 (I2TXBITRATE—地址 0x400A 8028)

位	符号	描述	复位值
5:0	tx_bitrate	I ² S 发送位速率。该值加 1 用于分频 TX_REF 来产生发送位时钟。	0
31:6	-	保留。读取值未定义，只写入 0。	无

23.5.12 接收时钟位速率寄存器 (I2SRXBITRATE—0x400A 802C)

I²S 接收器的位速率由 I2SRXBITRATE 寄存器的值决定。该值取决于音频采样速率、数据大小和使用的格式。其计算与 I2SRXBITRATE 寄存器中的计算相同。

表529. 接收时钟位速率寄存器位描述 (I2SRXBITRATE—地址 0x400A 802C)

位	符号	描述	复位值
5:0	rx_bitrate	I ² S 接收位速率。该值加 1 用于分频 RX_REF 来产生接收位时钟。	0
31:6	-	保留。读取值未定义，只写入 0。	无

23.5.13 发送模式控制寄存器 (I2STXMODE—0x400A 8030)

发送模式控制寄存器可控制发送时钟源、4 管脚模式的使能、TX_REF 使用方式以及是否使能 MCLK 输出。有关有用的模式组合的汇总，参见 23.7 节。

表530. 发送模式控制寄存器位描述 (I2STXMODE—0x400A 8030)

位	符号	值	描述	复位值
1:0	TXCLKSEL		选择发送位时钟分频器的时钟源。	0
		00	选择 TX 分数速率分频器时钟输出作为时钟源。	
		01	保留。	
		10	选择 RX_REF 信号作为 TX_REF 时钟源。	
		11	保留。	
2	TX4PIN		选择 4 管脚发送模式，该位为 1 时，使能 4 管脚发送模式。	0
3	TXMCENA		TX_MCLK 输出的使能位。	0
		0	禁能 TX_MCLK 至一个管脚的输出。	
		1	使能 TX_MCLK 至一个管脚的输出。	
31:4	-		保留。读取值未定义，只写入 0	无

23.5.14 接收模式控制寄存器（I2SRXMODE—0x400A 8034）

接收模式控制寄存器可控制接收时钟源、4 管脚模式的使能以及 RX_REF 使用方式。有关有用的模式组合的汇总，参见 [23.7](#) 节。

表531. 接收模式控制寄存器位描述（I2SRXMODE—0x400A 8034）

位	符号	值	描述	复位值
1:0	RXCLKSEL		选择接收位时钟分频器的时钟源。	0
		00	选择 RX 分数速率分频器时钟输出作为时钟源。	
		01	保留。	
		10	选择 TX_REF 信号作为 RX_REF 时钟源。	
		11	保留。	
2	RX4PIN		选择 4 管脚接收模式。该位为 1 时，使能 4 管脚接收模式。	0
3	RXMCENA		RX_MCLK 输出的使能位。	0
		0	禁能 RX_MCLK 至一个管脚的输出。	
		1	使能 RX_MCLK 至一个管脚的输出。	
31:4	-		保留。读取值未定义，只写入 0	无

23.6 I²S 的发送和接收接口

I²S 接口可发送和接收 8、16 或 32 位的立体声道或单声道音频信息。I²S 接口传输数据时的细节如下：

- 当 FIFO 为空时，发送通道将重复发送相同的数据，直到有新数据写入 FIFO。
- 静音时，发送数据值 0。
- 当禁能单声道时，两个连续的数据字分别是左声道和右声道的数据。
- 数据字长度由配置寄存器中字宽度的值决定。
接收通道和发送通道有单独的字宽度值。
 - 0：认为字包含四个 8 位的数据字。
 - 1：认为字包含两个 16 位的数据字。
 - 3：认为字包含一个 32 位的数据字。
- 当发送 FIFO 数据不足时，发送通道将重复发送最后一个数据，直至有新数据可用。这种情况会在微处理器或 DMA 无法尽快提供新数据时出现。由于在新数据中存在上述延时，所以需要填充间隙（延时），可通过继续发送最后的采样来实现填充。数据不会被静音，因为这将会在声音上产生明显而不合乎需要的效果。
- 发送通道和接收通道仅处理 32 位对齐的字，较长的数据块必须要进行截取而较短的数据则必须扩展为 32 位长的倍数才能发送和接收。

在切换数据宽度或模式时，I²S 必须通过控制寄存器中的复位位进行复位，以确保能够正确地进行同步操作。建议同时置位停止位，直到有足够的数写入发送 FIFO。请注意，在停止时，数据输出会被静音。

所有访问 FIFO 的数据为 32 位。[图 136](#) 所示为可能的数据序列。一个 FIFO 数据采样包括：

- 1 个 8 位或 16 位立体声道模式的 32 位数据。
- 1 个单声道模式的 32 位数据。
- 2 个 32 位数据，第一个为左声道数据，第二个为右声道数据，32 位立体声道模式。

数据在 WS 下降沿后从发送 FIFO 中读出，在 WS 上升沿后传输到发送时钟域。在 WS 的下一个下降沿，左声道数据将被载入到移位寄存器中并被发送，在 WS 的下一个上升沿，右声道数据会被载入并发送。

接收通道将在 WS 改变后开始接收数据。字选择（WS）变低时，期望接收数据是左声道数据，当 WS 为高时，希望接收数据是右声道数据。当位计数器达到字宽度值设置的极限时停止接收数据。在 WS 的下次改变时，接收到的数据将保存到相应的保持寄存器中。当接收完所有数据时，它将被写入到接收 FIFO 中。

23.7 I²S 操作模式

I²S 接口的时钟源和 WS 使用是可以配置的。除了主机和从机模式（可独立配置用于发送器和接收器）以外，还可以选择多个不同的时钟源，包括在发送器和接收器之间共用时钟和/或 WS。最后一种配置使用的 I²S 管脚较少，通常只有 4 个。

存在多种配置，但并非每种都有用，下面的表格和图格将对最有用的配置进行详细描述。

23.7.1 I²S 发送模式

表532. I²S 发送模式

I2SDAO [5]	I2STXMODE [3:0]	描述
0	0 0 0 0	典型的发送主机模式参见图 123。 I ² S 发送功能用作主机。 发送时钟源为分数速率分频器。 使用的 WS 为内部产生的 TX_WS。 TX_MCLK 不是 I2S_TX_MCLK 管脚输出。
0	0 0 1 0	发送主机模式共用接收器的参考时钟，参见图 124。 I ² S 发送功能用作主机。 发送时钟源为 RX_REF。 使用的 WS 为内部产生的 TX_WS。 TX_MCLK 不是 I2S_TX_MCLK 管脚输出。
0	0 1 0 0	4 线发送主机模式共用接收器的位时钟和 WS，参见图 125。 I ² S 发送功能用作主机。 发送时钟源为 RX 位时钟。 使用的 WS 为内部产生的 RX_WS。 TX_MCLK 不是 I2S_TX_MCLK 管脚输出。
0	1 0 0 0	发送主机模式，TC_MCLK 输出，参见图 123。 I ² S 发送功能用作主机。 发送时钟源为分数速率分频器。 使用的 WS 为内部产生的 TX_WS。 TX_MCLK 是 I2S_TX_MCLK 管脚输出。
1	0 0 0 0	典型的发送从机模式，参见图 126。 I ² S 发送功能用作从机。 发送时钟源来自 TX_CLK 管脚。 使用的 WS 为 TX_WS 管脚。
1	0 0 1 0	发送从机模式共用接收器的参考时钟，参见图 127。 I ² S 发送功能用作从机。 发送时钟源为 RX_REF。 使用的 WS 为 TX_WS 管脚。
1	0 1 0 0	4 线发送从机模式共用接收器的位时钟和 WS，参见图 128。 I ² S 发送功能用作从机。 发送时钟源为 RX 位时钟。 使用的 WS 为 RX_WS ref。

图123. 典型的发送主机模式（有或没有 MCLK 输出）

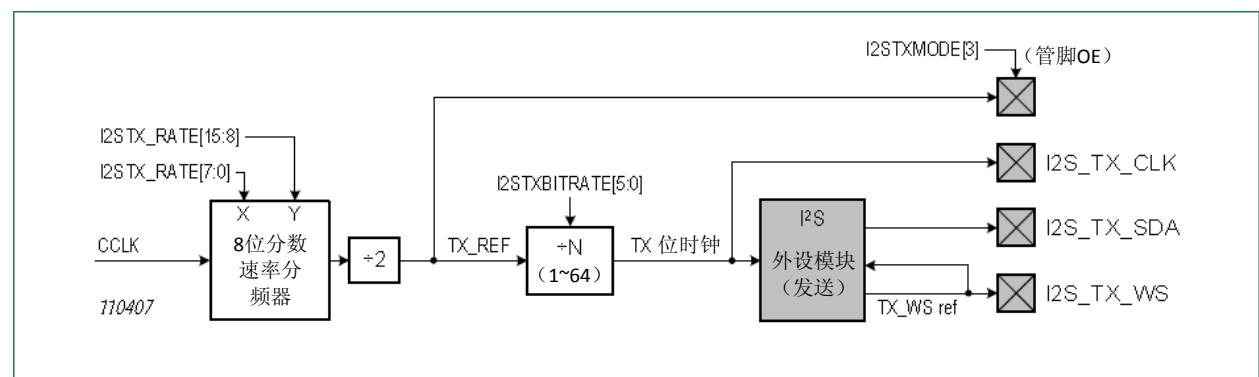


图124. 发送主机模式共用接收器的参考时钟

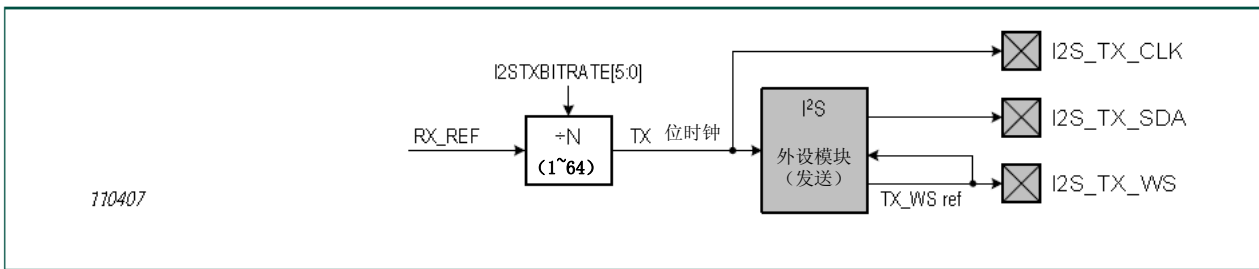


图125. 4 线发送主机模式共用接收器的位时钟和 WS

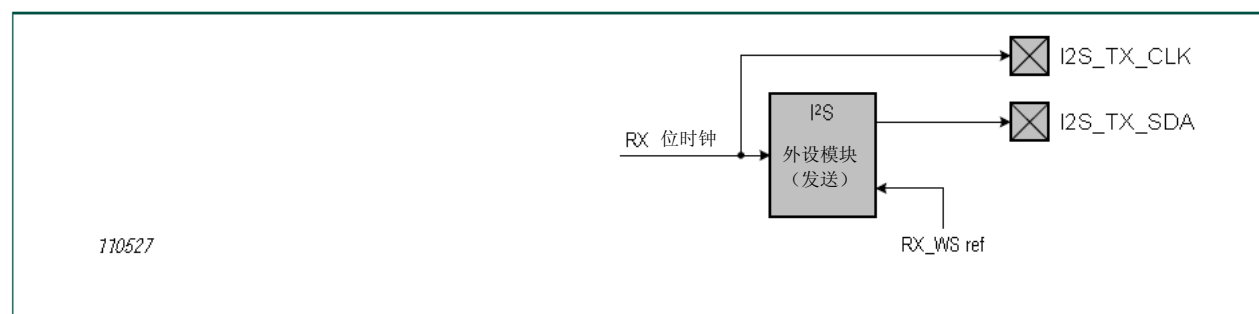


图126. 典型的发送从机模式

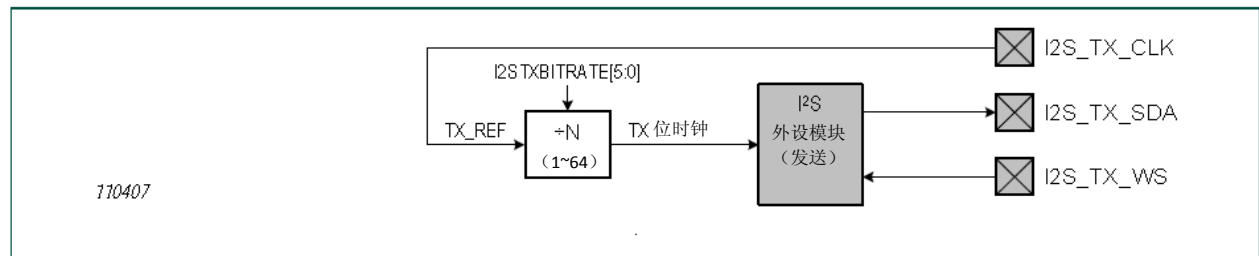


图127. 发送从机模式共用接收器的参考时钟

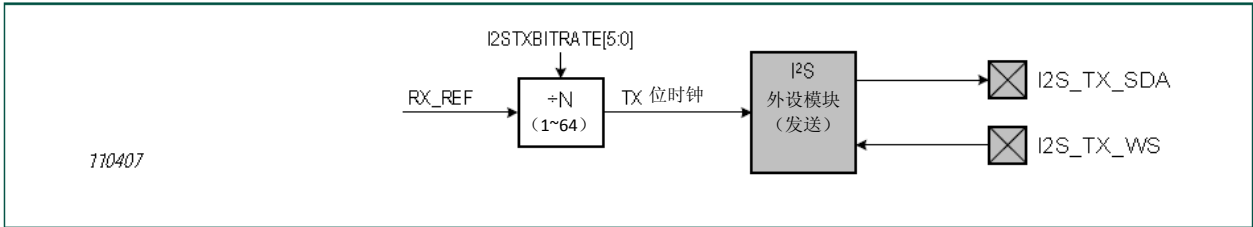
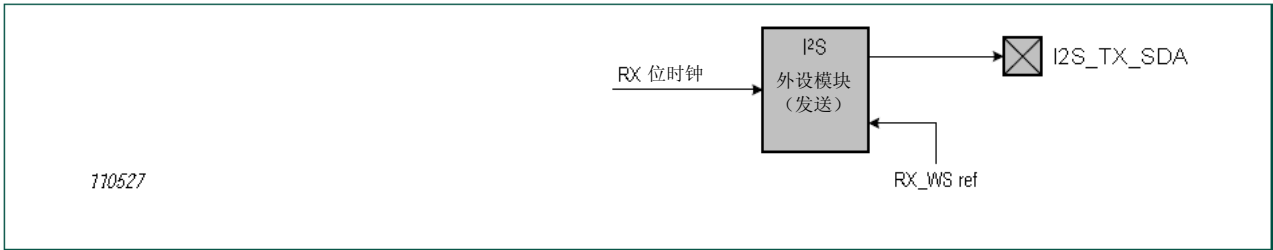


图128. 4 线发送从机模式共用接收器的位时钟和 WS



23.7.2 I²S 接收模式

表533. I²S 接收模式

I2SDAI [5]	I2SRXMODE [3:0]	描述
0	0 0 0 0	典型的接收主机模式，参见图 129。 I²S 接收功能用作主机。 接收时钟源为分数速率分频器。 使用的 WS 为内部产生的 RX_WS。 RX_MCLK 不是 I2S_RX_MCLK 管脚输出。
0	0 0 1 0	接收主机模式共用发送器的参考时钟，参见图 130。 I²S 接收功能用作主机。 接收时钟源为 TX_REF。 使用的 WS 为内部产生的 RX_WS。 RX_MCLK 不是 I2S_RX_MCLK 管脚输出。
0	0 1 0 0	4 线接收主机模式共用发送器的位时钟和 WS，参见图 131。 I²S 接收功能用作主机。 接收时钟源为 TX 位时钟。 使用的 WS 为内部产生的 TX_WS。 RX_MCLK 不是 I2S_RX_MCLK 管脚输出。
0	1 0 0 0	接收主机模式，RX_MCLK 输出，参见图 129。 I²S 接收功能用作主机。 接收时钟源为分数速率分频器。 使用的 WS 为内部产生的 RX_WS。 RX_MCLK 为 I2S_RX_MCLK 管脚输出。
1	0 0 0 0	典型的接收从机模式，参见图 132。 I²S 接收功能用作从机。 接收时钟源来自 RX_CLK 管脚。 使用的 WS 为 RX_WS 管脚。
1	0 0 1 0	接收从机模式共用发送器的参考时钟，参见图 133。 I²S 接收功能用作从机。 接收时钟源为 TX_REF。 使用的 WS 为 RX_WS 管脚。
1	0 1 0 0	4 线接收从机模式共用发送器的位时钟和 WS，参见图 134。 I²S 接收功能用作从机。 接收时钟源为 TX 位时钟。 使用的 WS 为 TX_WS ref。

图129. 典型的接收主机模式（有或没有 MCLK 输出）

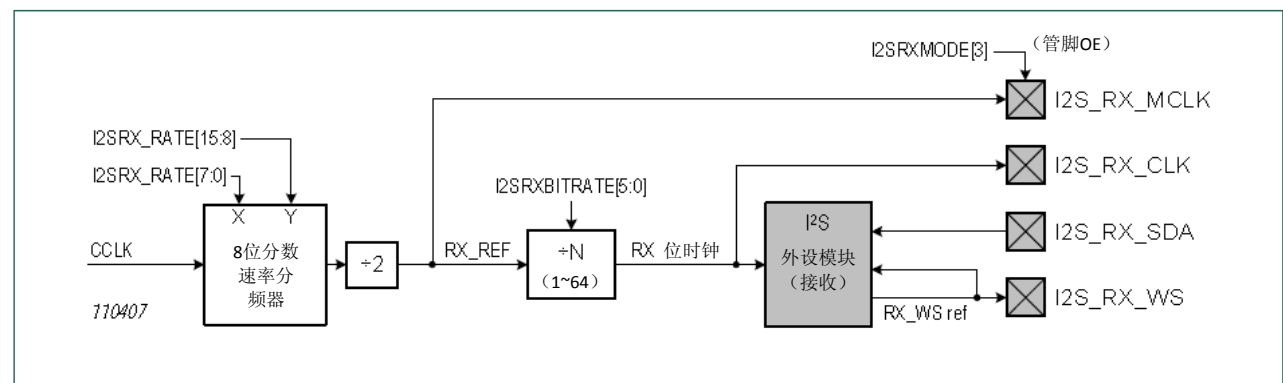


图130. 接收主机模式共用发送器的参考时钟

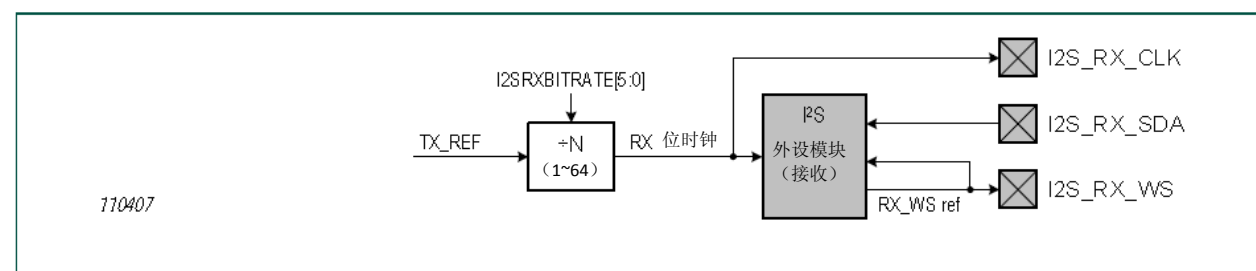


图131. 4 线接收主机模式共用发送器位时钟和 WS

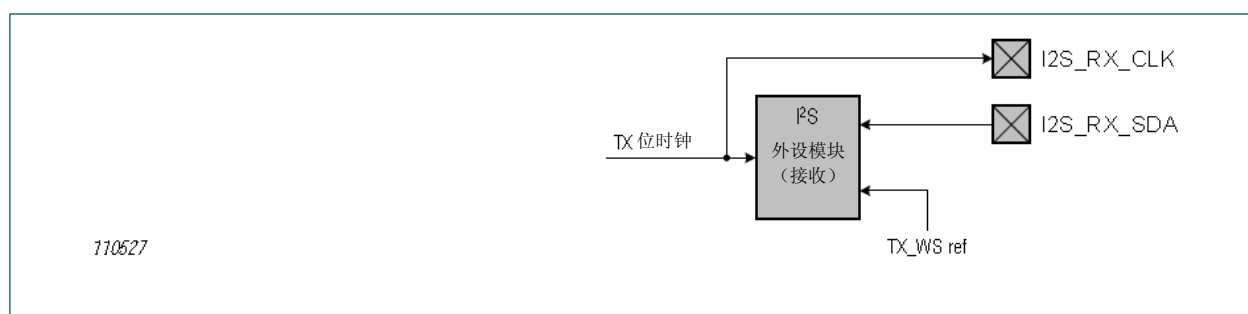


图132. 典型的接收从机模式

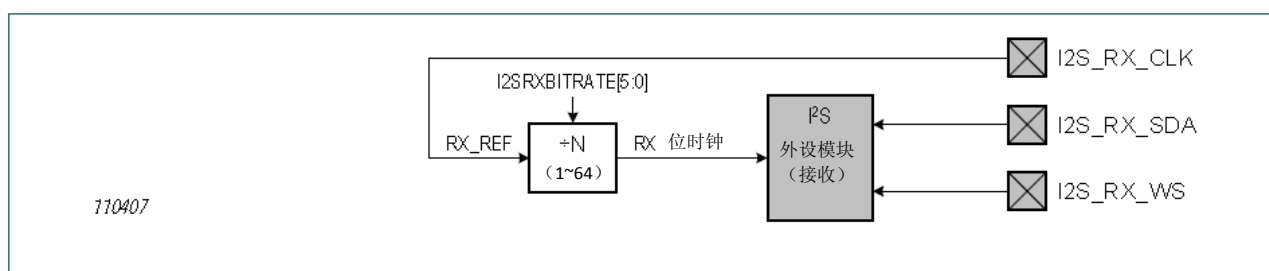


图133. 接收从机模式共用发送器参考时钟

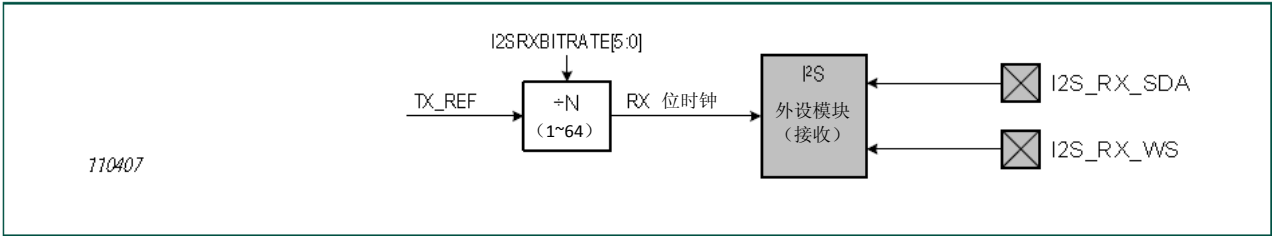
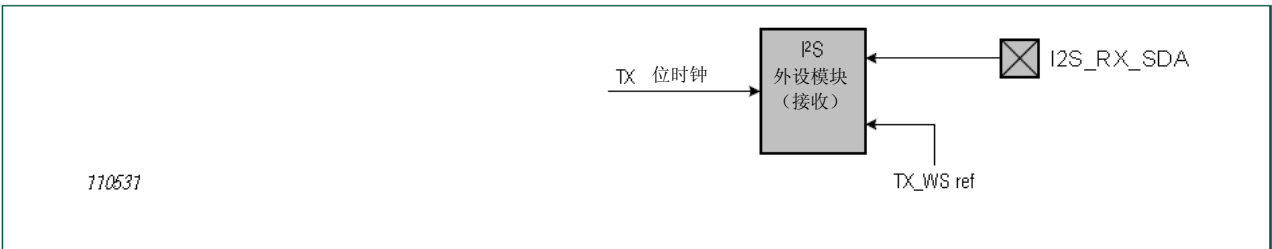


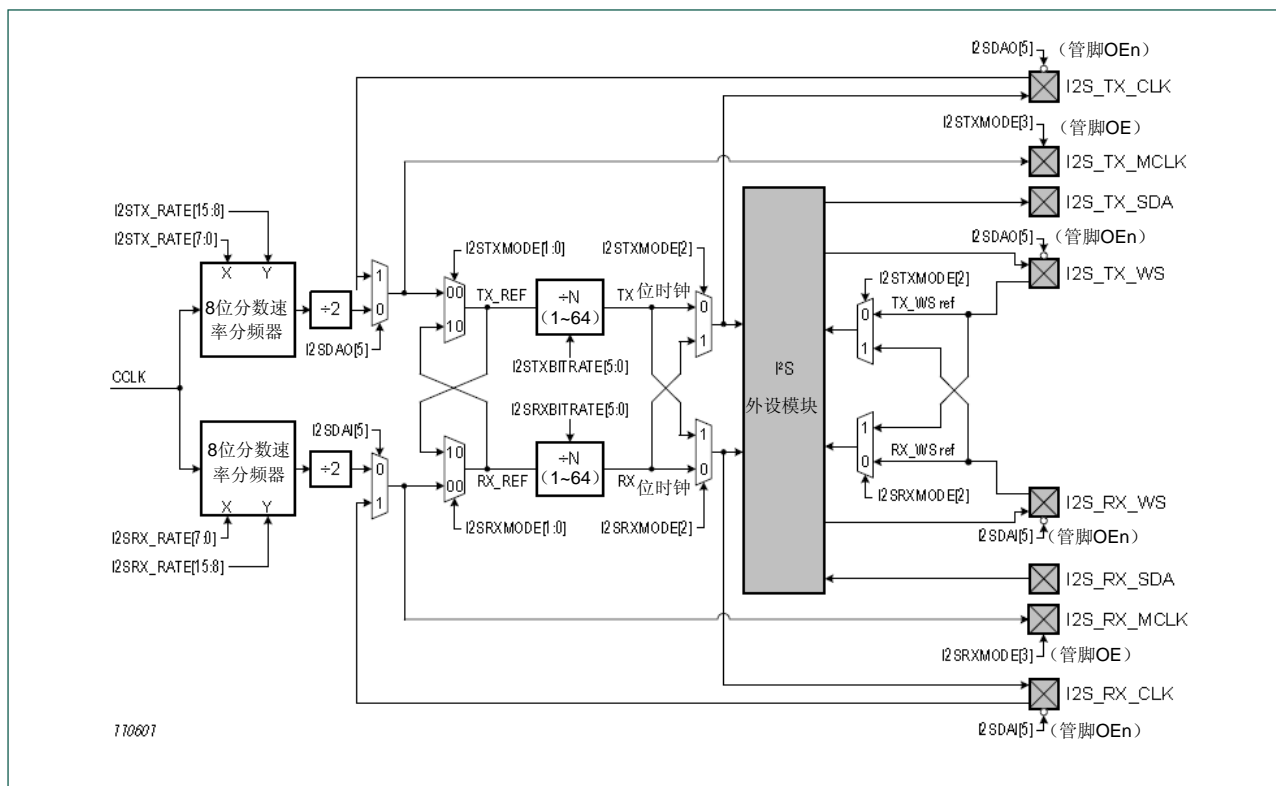
图134. 4 线接收从机模式共用发送器位时钟和 WS



23.7.2.1 全部时钟和管脚连接

图 135 中给出了 I²S 模块的所有时钟连接和管脚连接。

图135.I²S 时钟和管脚连接



23.8 FIFO 控制器

发送和接收数据的处理都是通过 FIFO 控制器来执行的，它可以产生两个 DMA 请求和一个中断请求。该控制器包括一组比较器，它们将 FIFO 的深度与寄存器中包含的设置深度进行比较。深度比较器的当前状态可在 APB 状态寄存器中查看。

表534. FIFO 深度比较的条件

深度比较	条件
dmareq_tx_1	tx_depth_dma1 >= tx_level
dmareq_rx_1	rx_depth_dma1 <= rx_level
dmareq_tx_2	tx_depth_dma2 >= tx_level
dmareq_rx_2	rx_depth_dma2 <= rx_level
irq_tx	tx_depth_irq >= tx_level
irq_rx	rx_depth_irq <= rx_level

当深度检测结果为真且使能时，系统会发出信号示意。

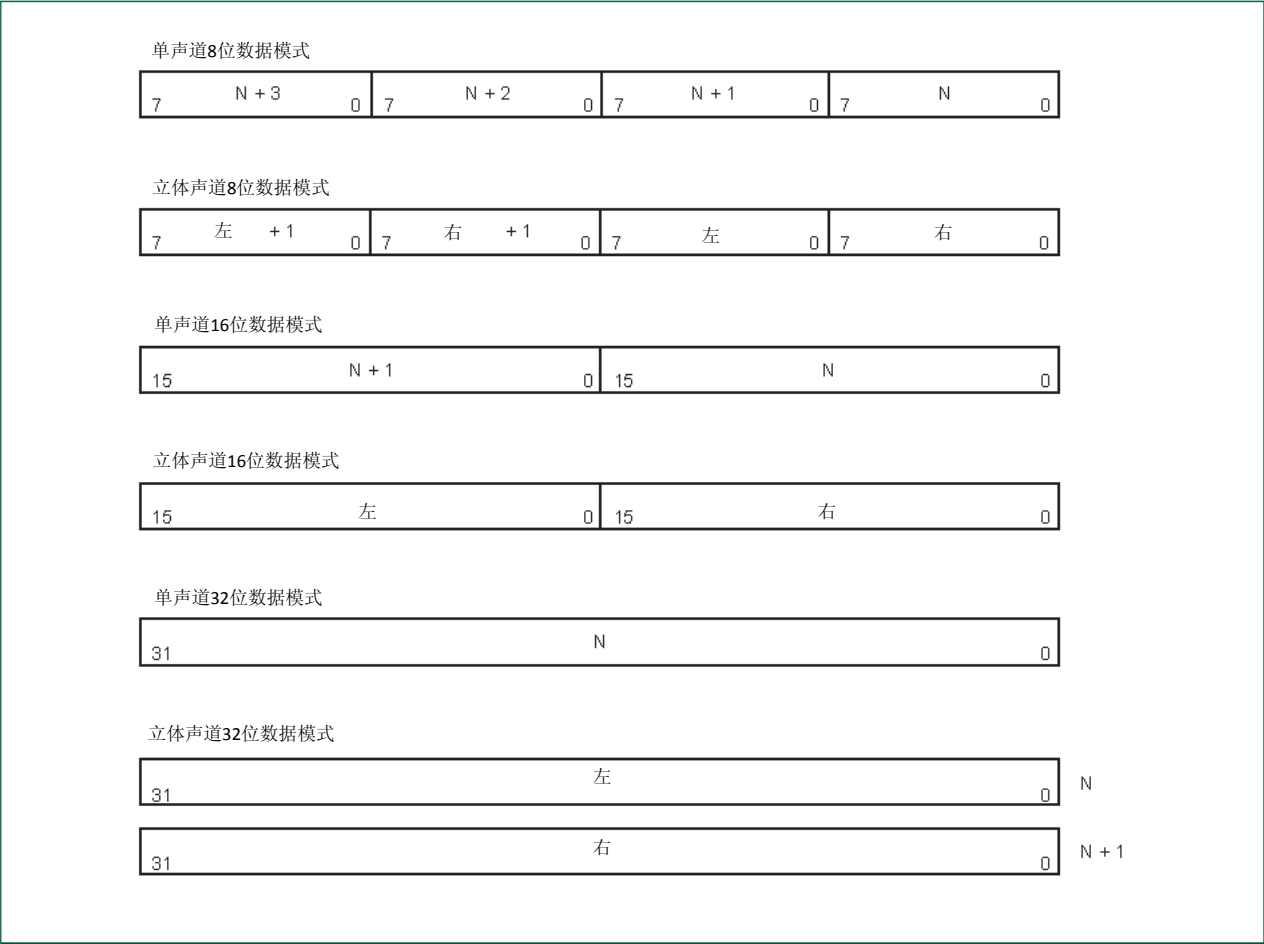
表535. DMA 和中断请求产生

系统发出信号	条件
irq	(irq_rx & rx_irq_enable) (irq_tx & tx_irq_enable)
dmareq[0]	(dmareq_tx_1 & tx_dma1_enable) (dmareq_rx_1 & rx_dma1_enable)
dmareq[1]	(dmareq_tx_2 & tx_dma2_enable) (dmareq_rx_2 & rx_dma2_enable)

表536. I2SSTATE 寄存器中的状态反馈

状态反馈	状态
irq	irq_rx irq_tx
dmareq1	(dmareq_tx_1 dmareq_rx_1)
dmareq2	(dmareq_rx_2 dmareq_tx_2)

图136. 不同 I²S 模式的 FIFO



24.1 基本配置

使用下列寄存器来配置定时器 0/1/2/3 外设：

1. 功率：在 PCONP 寄存器（[表 37](#)）中置位 PCTIM0/1/2/3。
注：复位时，定时器 0/1 使能（PCTIM0/1=1），而定时器 2/3 禁能（PCTIM2/3=0）。
2. 外设时钟：定时器使用公共 PCLK，它既用于总线接口，也用于大多数 APB 外设的功能部分，参见 [4.6.4](#) 节。
3. 管脚：在相关的 IOCON 寄存器（[8.4.1](#) 节）中选择定时器管脚和管脚模式。
4. 中断：有关匹配和捕获事件的内容，参见寄存器 T0/1/2/3MCR（[表 542](#)）和 T0/1/2/3CCR（[表 543](#)）。利用相应的中断置位使能寄存器使能 NVIC 中的中断。
5. DMA：在最多 2 种匹配情况下可产生定时的 DMA 请求，参见[表 664](#)。

24.2 特性

注：除了外设基地址以外，四个定时器/计数器完全相同。四个定时器至少有两个捕获输入和两个匹配输出，并且有多个管脚可以选择。定时器 2 引出了全部四个匹配输出。

- 32 位的定时器/计数器，带有一个可编程的 32 位预分频器。
- 计数器或定时器操作。
- 每个定时器包含多达两个 32 位的捕获通道，当输入信号变换时可捕获定时器的瞬时值。也可以选择使捕获事件生成一个中断。
- 4 个 32 位匹配寄存器，允许执行以下操作：
 - 在匹配时继续工作，在匹配时可选择产生中断；
 - 在匹配时停止定时器运行，可选择产生中断；
 - 在匹配时复位定时器，可选择产生中断。
- 有多达四个匹配寄存器相对应的外部输出，这些输出具有以下功能：
 - 匹配时设为低电平；
 - 匹配时设为高电平；
 - 匹配时翻转电平；
 - 匹配时不执行任何操作。

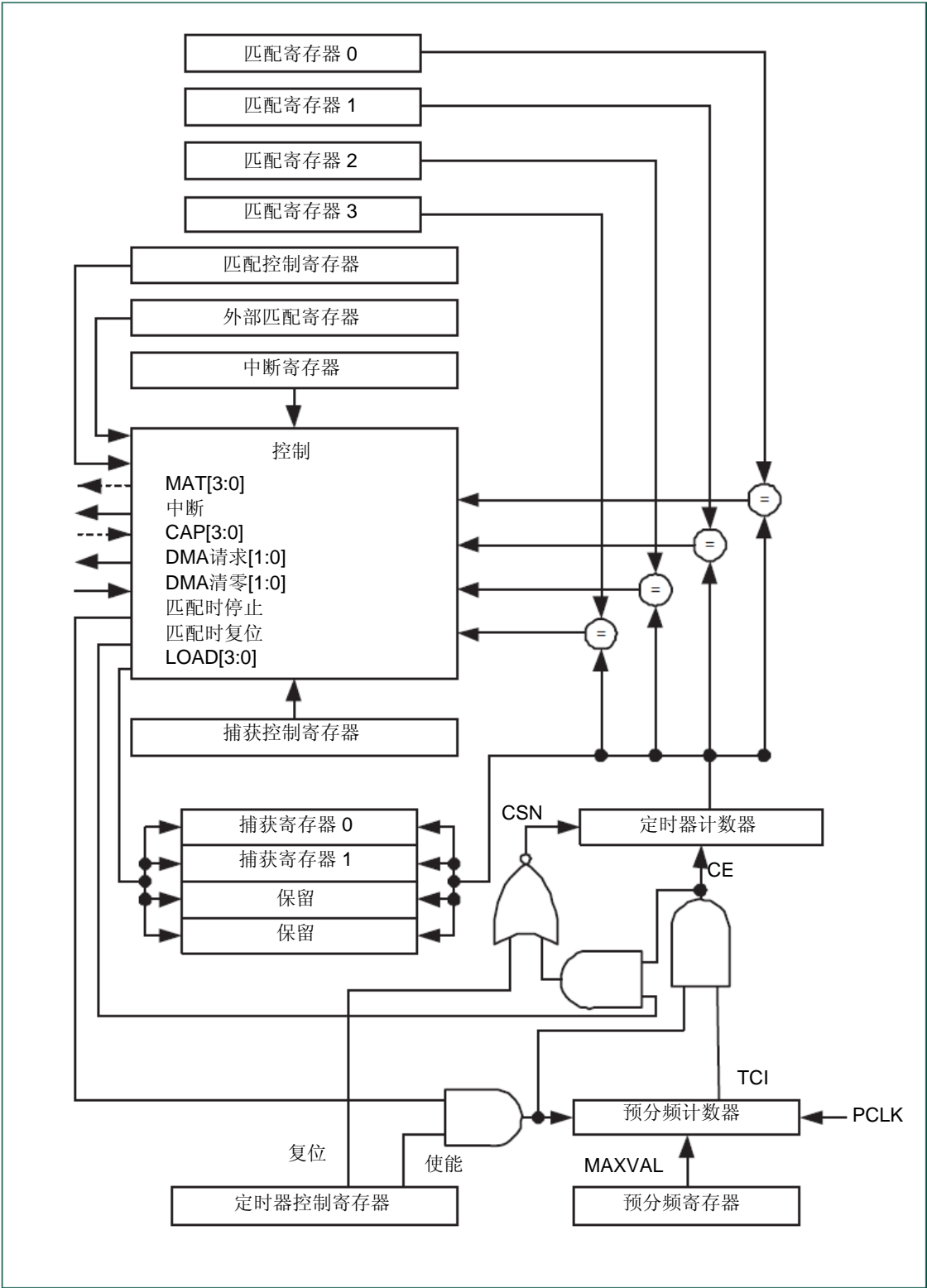
24.3 应用

- 间隔定时器，用来计数内部事件。
- 脉宽解调器（经由捕获输入）。
- 自由运行的定时器。

24.4 描述

定时器/计数器用来对外设时钟（PCLK）或外部提供的时钟周期进行计数，可选择在规定的时间处产生中断或执行其他操作，这都由四个匹配寄存器的值决定。它还包括四个捕获输入，用来在输入信号变换时捕获定时器的瞬时值，也可以选择产生中断。

图137. 定时器模块框图



24.5 管脚描述

表 537 对每个定时器/计数器的相关管脚进行了简要描述。

表537. 定时器/计数器管脚描述

管脚	类型	描述
T0_CAP1:0	输入	捕获信号—当捕获管脚出现特定的捕获事件时，可以将当前的定时器计数器值装入一个捕获寄存器中，也可以选择产生一个中断。定时器/计数器的捕获管脚有多个。当有多个管脚被同时选择作 TIMER0/1 通道的捕获输入时，端口编号最小的管脚被使用。
T1_CAP1:0		
T2_CAP1:0		
T3_CAP1:0		
T0_MAT1:0	输出	外部匹配输出—当匹配寄存器（ MR3:0 ）的值与定时器计数器（ TC ）的值相等时，相应的输出可以翻转、变低、变高或不执行任何操作。外部匹配寄存器（ EMR ）控制着输出的功能。并行的多个管脚可以被选择用作匹配输出功能。
T1_MAT1:0		
T2_MAT3:0		
T3_MAT1:0		

24.5.1 多个 CAP 和 MAT 管脚

软件可以在 **IOCON** 寄存器（见 8.4.1 节）中选择多个管脚用作 **CAP** 或 **MAT** 功能。当有多个管脚用作 **MAT** 输出时，所有这些管脚都统一驱动。当有多个管脚用作 **CAP** 输入时，使用端口编号最低的管脚。请注意，在不使用设备管脚的情况下，匹配功能也可以在内部使用。

24.6 寄存器描述

每个定时器/计数器包含的寄存器如表 538 中所示（“复位值”仅指已使用位中存储的数据，不包括保留位的内容）。详见下文中的描述。

表538. 定时器/计数器 0~3 的寄存器映射

通用寄存器名称	描述	访问	复位值 ^[1]	TIMERN 寄存器的名称&地址	参考
IR	中断寄存器。可向 IR 写入相应的值来清除中断。也可读 IR 来确定中断源中哪个中断源被挂起。	R/W	0	T0IR-0x4000 4000 T1IR-0x4000 8000 T2IR-0x4009 0000 T3IR-0x4009 4000	24.6.1
TCR	定时器控制寄存器。 TCR 用来控制定时器计数器的功能。定时器计数器可以通过 TCR 来禁能或复位。	R/W	0	T0TCR-0x4000 4004 T1TCR-0x4000 8004 T2TCR-0x4009 0004 T3TCR-0x4009 4004	24.6.2
TC	定时器计数器。32 位的 TC 每隔（ PR +1）个 PCLK 周期递增一次。 TC 通过 TCR 来控制。	R/W	0	T0TC-0x4000 4008 T1TC-0x4000 8008 T2TC-0x4009 0008 T3TC-0x4009 4008	24.6.4
PR	预分频寄存器。当预分频计数器（如下）的值与这个寄存器的值相等时，下个时钟 TC 加这个 TC ， PC 清零。	R/W	0	T0PR-0x4000 400C T1PR-0x4000 800C T2PR-0x4009 000C T3PR-0x4009 400C	24.6.5
PC	预分频计数器。32 位的 PC 是一个计数器，该计数器的值会递增到与 PR 中存放的值相等。当达到了 PR 的值时，加 TC ， PC 清零。可以通过总线接口来观察	R/W	0	T0PC-0x4000 4010 T1PC-0x4000 8010 T2PC-0x4009 0010	24.6.6

通用寄存器名称	描述	访问	复位值 ^[1]	TIMERn 寄存器的名称&地址	参考
	和控制 PC。			T3PC-0x4009 4010	
MCR	匹配控制寄存器。MCR 用来控制在匹配出现时是否产生中断和是否复位 TC。	R/W	0	T0MCR-0x4000 4014 T1MCR-0x4000 8014 T2MCR-0x4009 0014 T3MCR-0x4009 4014	24.6.8
MR0	匹配寄存器 0。MR0 可通过 MCR 做如下设置：当 MR0 的值与 TC 值匹配时，复位 TC、停止 TC 和 PC 和/或产生中断。	R/W	0	T0MR0-0x4000 4018 T1MR0-0x4000 8018 T2MR0-0x4009 0018 T3MR0-0x4009 4018	24.6.7
MR1	匹配寄存器 1。请参见 MR0 的描述。	R/W	0	T0MR1-0x4000 401C T1MR1-0x4000 801C T2MR1-0x4009 001C T3MR1-0x4009 401C	24.6.7
MR2	匹配寄存器 2。请参见 MR0 的描述。	R/W	0	T0MR2-0x4000 4020 T1MR2-0x4000 8020 T2MR2-0x4009 0020 T3MR2-0x4009 4020	24.6.7
MR3	匹配寄存器 3。请参见 MR0 的描述。	R/W	0	T0MR3-0x4000 4024 T1MR3-0x4000 8024 T2MR3-0x4009 0024 T3MR3-0x4009 4024	24.6.7
CCR	捕获控制寄存器。CCR 控制捕获输入的哪个沿用于装入捕获寄存器的值以及当捕获发生时是否生成中断。	R/W	0	T0CCR-0x4000 4028 T1CCR-0x4000 8028 T2CCR-0x4009 0028 T3CCR-0x4009 4028	24.6.10
CR0	捕获寄存器 0。当 Tn_CAP0 输入上发生一个事件时，CR0 装载 TC 的值。	RO	0	T0CR0-0x4000 402C T1CR0-0x4000 802C T2CR0-0x4009 002C T3CR0-0x4009 402C	24.6.9
CR1	捕获寄存器 1。当 Tn_CAP1 输入上发生一个事件时，CR1 装载 TC 的值。	RO	0	T0CR1-0x4000 4030 T1CR1-0x4000 8030 T2CR1-0x4009 0030 T3CR1-0x4009 4030	24.6.9
EMR	外部匹配寄存器。EMR 控制外部匹配管脚 Tn_MAT0-3。	R/W	0	T0EMR-0x4000 403C T1EMR-0x4000 803C T2EMR-0x4009 003C T3EMR-0x4009 403C	24.6.11
CTCR	计数控制寄存器。CTCR 选择在定时器模式还是计数器模式下工作，在计数器模式中用于选择要计数的信号和沿。	R/W	0	T0CTCR-0x4000 4070 T1CTCR-0x4000 8070 T2CTCR-0x4009 0070 T3CTCR-0x4009 4070	24.6.3

[1] 复位值只反映已使用位的数据，不包括保留位的内容。

24.6.1 中断寄存器（T[0/1/2/3]IR—0x4000 4000, 0x4000 8000, 0x4009 0000, 0x4009 4000）

中断寄存器包含 4 个用于匹配中断的位，4 个用于捕获中断的位。如果产生了一个中断，则 IR 中的对应位将会置位为高，否则，该位为低。向对应的 IR 位写入逻辑 1 会复位中断。写入 0 无效。清除定时器匹配中断的操作也会清除对应的 DMA 请求。

表539. 中断寄存器位描述（T[0/1/2/3]IR—地址 0x4000 4000, 0x4000 8000, 0x4009 0000, 0x4009 4000）

位	符号	描述	复位值
0	MR0 Interrupt	匹配通道 0 的中断标志	0
1	MR1 Interrupt	匹配通道 1 的中断标志	0
2	MR2 Interrupt	匹配通道 2 的中断标志	0
3	MR3 Interrupt	匹配通道 3 的中断标志	0
4	CR0 Interrupt	捕获通道 0 事件的中断标志	0
5	CR1 Interrupt	捕获通道 1 事件的中断标志	0
31:6	-	保留。读取值未定义，只写入 0。	-

24.6.2 定时器控制寄存器（T[0/1/2/3]CR—0x4000 4004, 0x4000 8004, 0x4009 0004, 0x4009 4004）

定时器控制寄存器（TCR）用于控制定时器/计数器的操作。

表540. 定时器控制寄存器位描述（TCR, TIMERN: TnTCR—地址 0x4000 4004, 0x4000 8004, 0x4009 0004, 0x4009 4004）

位	符号	描述	复位值
0	Counter Enable	为 1 时，定时器计数器和预分频计数器使能计数。为 0 时，计数器被禁能。	0
1	Counter Reset	为 1 时，定时器计数器和预分频计数器在 PCLK 的下一个正相沿同步复位。计数器在 TCR[1]恢复为 0 之前保持复位状态。	0
31:2	-	保留。读取值未定义，只写入 0。	无

24.6.3 计数控制寄存器（T[0/1/2/3]CTCR—0x4000 4070，0x4000 8070，0x4009 0070，0x4009 4070）

计数控制寄存器（CTCR）用于在定时器模式和计数器模式之间进行选择，在计数器模式中用于选择要计数的管脚和沿。

如果选择在计数器模式下操作，在每个 PCLK 时钟的上升沿对 CAP 输入（使用 CTCR 的位 3:2 来选择）进行采样。在对这个 CAP 输入的连续两次采样值进行比较之后，可以识别出下面其中一种事件：上升沿、下降沿、上升/下降沿或所选 CAP 输入的电平不变。如果识别出的事件与 CTCR 寄存器的位 1:0 选择的一个事件相对应，定时器计数器寄存器的值将增加 1。

当计数器计数外部供应时钟时，处理的效率会受到一些限制。由于识别 CAP 所选输入的一个沿需要使用 PCLK 时钟两个连续的上升沿，因此 CAP 输入的频率不能大于 PCLK 时钟频率的四分之一。因此，在这种情况下，同一个 CAP 输入管脚的高/低电平持续时间不能小于 1/（2PCLK）。

表541. 计数控制寄存器位描述（T[0/1/2/3]CTCR—地址 0x4000 4070，0x4000 8070，0x4009 0070，0x4009 4070）

位	符号	值	描述	复位值
1:0	Counter/ Timer Mode		这个字段选择在哪些条件下定时器的预分频计数值（PC）递增、PC 清零以及定时器计数器值（TC）递增。计数器模式下预分频器被旁路。	00
		00	定时器模式：当预分频计数器与预分频寄存器的值匹配时 TC 增加。预分频计数器在每个 PCLK 的上升沿增加。	
		01	计数器模式：TC 在位 3:2 选择的 CAP 输入的上升沿出现时增加。	
		10	计数器模式：TC 在位 3:2 选择的 CAP 输入的下降沿出现时增加。	
		11	计数器模式：TC 在位 3:2 选择的 CAP 输入的两个边沿出现时增加。	
3:2	Count Input Select		当寄存器的位 1:0 不为 00 时，这两位选择采样用于计时的 CAP 管脚。	00
		00	CAPn.0 用于 TIMERN。	
		01	CAPn.1 用于 TIMERN。	
		10	保留。	
		11	保留。	
注：如果 TnCTCR 中的特定 CAPn 输入选择在计数器模式下操作，则输入时时钟源 CAPn 在捕获控制寄存器（TnCCR）中对应的 3 个位必须被编程为 000。但是，同一个定时器的其他 3 个 CAPn 输入管脚仍可以被选择用作捕获和/或中断功能。				
31:4			保留。读取值未定义，只写入 0。	无

24.6.4 定时器计数器寄存器（T0TC—T3TC，0x4000 4008，0x4000 8008，0x4009 0008，0x4009 4008）

当预分频计数器到达其计数上限时，32 位的定时器计数器寄存器值会加 1。如果定时器计数器在达到其上限之前没有复位，则它将一直计数到 0xFFFF FFFF，然后翻转到 0x0000 0000。该事件不会产生中断，但在需要时，可以使用匹配寄存器检测溢出。

24.6.5 预分频寄存器（T0PR—T3PR，0x4000 400C，0x4000 800C，0x4009 000C，0x4009 400C）

32 位预分频寄存器指定了预分频计数器的最大值。

24.6.6 预分频计数器寄存器（T0PC—T3PC，0x4000 4010，0x4000 8010，0x4009 0010，0x4009 4010）

32 位预分频计数器使用某个常量值来控制 PCLK 的分频，再使之输入定时器计数器。由此可以控制定时器分辨率与定时器溢出之前的最大时间值之间的关系。预分频计数器每个 PCLK 周期加 1。当其值到达预分频寄存器中保存的值时，定时器计数器加 1，预分频计数器在下一个 PCLK 周期复位。例如，当 PR=0 时，定时器计数器每个 PCLK 周期加 1，当 PR=1 时，定时器计数器每两个 PCLK 周期加 1，以此类推。

24.6.7 匹配寄存器（MR0—MR3）

匹配寄存器值连续与定时器计数器值进行比较。当两个值相等时，会自动触发相应操作。可能的操作包括产生中断、复位定时器计数器或停止定时器。所执行的操作由 MCR 寄存器控制。

24.6.8 匹配控制寄存器（T[0/1/2/3]MCR—0x4000 4014，0x4000 8014，0x4009 0014，0x4009 4014）

匹配控制寄存器用于控制当某个匹配寄存器与定时器计数器匹配时所执行的操作。每个位的功能如表 542 所示。

表542. 匹配控制寄存器位描述（T[0/1/2/3]MCR—地址 0x4000 4014，0x4000 8014，0x4009 0014，0x4009 4014）

位	符号	值	描述	复位值
0	MR0I	1	MR0 引发的中断：MR0 与 TC 值匹配时将产生中断。	0
		0	中断被禁止。	0
1	MR0R	1	MR0 引发的复位：MR0 与 TC 值匹配时 TC 复位。	0
		0	该特性被禁止。	0
2	MR0S	1	MR0 引发的停止：MR0 与 TC 值匹配时 TC 和 PC 停止，TCR[0]清零。	0
		0	该特性被禁止。	0
3	MR1I	1	MR1 引发的中断：MR1 与 TC 值匹配时将产生中断。	0
		0	中断被禁止。	0
4	MR1R	1	MR1 引发的复位：MR1 与 TC 值匹配时 TC 复位。	0
		0	该特性被禁止。	0

位	符号	值	描述	复位值
5	MR1S	1	MR1 引发的停止：MR1 与 TC 值匹配时 TC 和 PC 停止，TCR[0]清零。	0
		0	该特性被禁止。	0
6	MR2I	1	MR2 引发的中断：MR2 与 TC 值匹配时将产生中断。	0
		0	中断被禁止。	0
7	MR2R	1	MR2 引发的复位：MR2 与 TC 值匹配时 TC 复位。	0
		0	该特性被禁止。	0
8	MR2S	1	MR2 引发的停止：MR2 与 TC 值匹配时 TC 和 PC 停止，TCR[0]清零。	0
		0	该特性被禁止。	0
9	MR3I	1	MR3 引发的中断：MR3 与 TC 值匹配时将产生中断。	0
		0	中断被禁止。	0
10	MR3R	1	MR3 引发的复位：MR3 与 TC 值匹配时 TC 复位。	0
		0	该特性被禁止。	0
11	MR3S	1	MR3 引发的停止：MR3 与 TC 值匹配时 TC 和 PC 停止，TCR[0]清零。	0
		0	该特性被禁止。	0
31:12	-		保留。读取值未定义，只写入 0。	无

24.6.9 捕获寄存器（CR0—CR1）

每个捕获寄存器都与一个设备管脚相关联。当捕获管脚上发生特定的事件时，可将定时器计数器的值装入该捕获寄存器。通过捕获控制寄存器的设定来决定是否使能捕获功能，以及捕获事件的类型——关联管脚的上升沿、下降沿还是双边沿。

24.6.10 捕获控制寄存器（T[0/1/2/3]CCR—0x4000 4028，0x4000 8028，0x4009 0028，0x4009 4028）

捕获控制寄存器用于控制：在发生捕获事件时，是否将定时器计数器的值装入其中一个捕获寄存器，以及是否产生中断。同时设置捕获事件——上升沿和下降沿，这样会在两个沿都触发捕获事件。在下面的描述中，“n”代表定时器的编号 0 或 1。

注：如果在 CTCR 中选择某个特定的 CAP 输入作为计数器的输入时，这个输入管脚对应在这个寄存器中的 3 个位必须被编程为 000。但是，其它 3 个 CAP 输入可以被选择用作捕获和/或中断功能。

表543. 捕获控制寄存器位描述（T[0/1/2/3]CCR—地址 0x4000 4028，0x4000 8020，0x4009 0028，0x4009 4028）

位	符号	值	描述	复位值
0	CAP0RE	1	CAPn.0 上升沿捕获：CAPn.0 上 0 到 1 的跳变将导致 TC 的内容装入 CR0。	0
		0	该特性被禁止。	
1	CAP0FE	1	CAPn.0 下降沿捕获：CAPn.0 上 1 到 0 的跳变将导致 TC 的内容装入 CR0。	0
		0	该特性被禁止。	
2	CAP0I	1	CAPn.0 事件中断：CAPn.0 的捕获事件所导致的 CR0 装载将产生一个中断。	0
		0	该特性被禁止。	
3	CAP1RE	1	CAPn.1 上升沿捕获：CAPn.1 上 0 到 1 的跳变将导致 TC 的内容装入 CR1。	0
		0	该特性被禁止。	
4	CAP1FE	1	CAPn.1 下降沿捕获：CAPn.1 上 1 到 0 的跳变将导致 TC 的内容装入 CR1。	0
		0	该特性被禁止。	

位	符号	值	描述	复位值
5	CAP1I	1	CAPn.1 事件中断：CAPn.1 的捕获事件所导致的 CR1 装载将产生一个中断。	0
		0	该特性被禁止	
31:6	-		保留。读取值未定义，只写入 0。	无

24.6.11 外部匹配寄存器（T[0/1/2/3]EMR—0x4000 403C, 0x4000 803C, 0x4009 003C, 0x4009 403C）

外部匹配寄存器提供对外部匹配管脚的控制及其状态。在下文的描述中，“n”代表定时器编号 0 或 1，“m”代表匹配编号 0 到 3。

每个定时器中的匹配 0 和 1 的匹配事件都可以引起一个 DMA 请求，参见 24.6.12 节。

表544. 外部匹配寄存器位描述（T[0/1/2/3]EMR—地址 0x4000 403C, 0x4000 803C, 0x4009 003C,0x4009 403C）

位	符号	描述	复位值
0	EM0	外部匹配 0。当 TC 和 MR0 匹配时，该位可翻转、变为低电平、变为高电平或不执行任何操作，这取决于该寄存器的 5:4 位。该位的值会被驱动到 MATn.0 管脚上，采用正逻辑方式（0=低电平，1=高电平）。	0
1	EM1	外部匹配 1。当 TC 和 MR1 匹配时，该位可翻转、变为低电平、变为高电平或不执行任何操作，这取决于该寄存器的 7:6 位。该位的值会被驱动到 MATn.1 管脚上，采用正逻辑方式（0=低电平，1=高电平）。	0
2	EM2	外部匹配 2。当 TC 和 MR2 匹配时，该位可翻转、变为低电平、变为高电平或不执行任何操作，这取决于该寄存器的 9:8 位。该位的值会被驱动到 MATn.2 管脚上，采用正逻辑方式（0=低电平，1=高电平）。	0
3	EM3	外部匹配 3。当 TC 和 MR3 匹配时，该位可翻转、变为低电平、变为高电平或不执行任何操作，这取决于该寄存器的 11:10 位。该位的值会被驱动到 MATn.3 管脚上，采用正逻辑方式（0=低电平，1=高电平）。	0
5:4	EMC0	外部匹配控制 0。决定外部匹配 0 的功能。关于这些位的编码，参见表 545。	00
7:6	EMC1	外部匹配控制 1。决定外部匹配 1 的功能。关于这些位的编码，参见表 545。	00
9:8	EMC2	外部匹配控制 2。决定外部匹配 2 的功能。关于这些位的编码，参见表 545。	00
11:10	EMC3	外部匹配控制 3。决定外部匹配 3 的功能。关于这些位的编码，参见表 545。	00
15:12	-	保留。 读取值未定义，只写入 0。	无

表545. 外部匹配控制

EMR[11:10], EMR[9:8],功能 EMR[7:6]或 EMR[5:4]	
00	不执行任何操作。
01	将对应的外部匹配位/输出清零（如果将 MATn.m 管脚引出来，则输出低电平）。
10	将对应的外部匹配位/输出设置为 1（如果将 MATn.m 管脚引出来，则输出高电平）。
11	使对应的外部匹配位/输出翻转。

24.6.12 DMA 操作

如果定时器计数器（TC）寄存器的值与匹配寄存器 0（MR0）或匹配寄存器 1（MR1）的值匹配，则会生成 DMA 请求。这与 EMR 寄存器控制的匹配输出的操作无关。每个匹配都会设置一个 DMA 请求标志，这一标志与 DMA 控制器相连。要想执行 DMA 请求，必须配置好 GPDMA，并通过 DMAREQSEL 寄存器将相关的定时器 DMA 请求选择成为一个 DMA 请求源，参见 34.5.15 节。

如果定时器一开始就被设置成产生一个 DMA 请求，那么在匹配条件发生以前，该请求可能已被申请了（有效）。软件向中断标志位写 1 可防止产生一个初始 DMA 请求，如同清除一个定时器中断一样。参见 24.6.1 节。当 GPDMA 控制器对其进行操作时，DMA 请求将会自动清除。

注：只要定时器值等于相关的匹配寄存器值，就会生成定时器 DMA 请求。因此，在定时器运行时将会始终生成 DMA 请求，除非匹配寄存器的值超过了定时器的计数上限值。必须注意，除非已将定时器正确配置为生成有效的 DMA 请求，否则，不要在 GPDMA 模块中选择并启用定时器 DMA 请求。

24.7 定时器操作示例

图 138 所示为定时器在发生匹配时复位并产生中断。预分频器设置为 2，匹配寄存器设置为 6。在发生匹配的定时器周期结束时，定时器计数值复位。这样就使匹配值具有一个完整长度的周期。指示匹配发生的中断在定时器到达匹配值的下一个时钟产生。

图 139 所示为定时器配置为在发生匹配时停止并产生中断。预分频器同样设置为 2，匹配寄存器设置为 6。在定时器到达匹配值后的下一个时钟周期，TCR 中的定时器使能位被清零并产生指示匹配发生的中断。

图138. 定时器周期设置为 PR=2，MRx=6，匹配时使能中断和复位

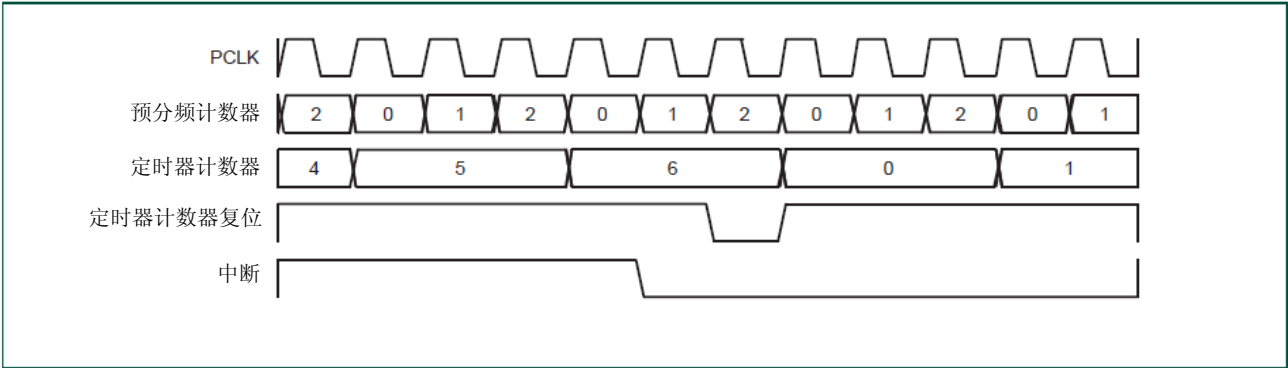
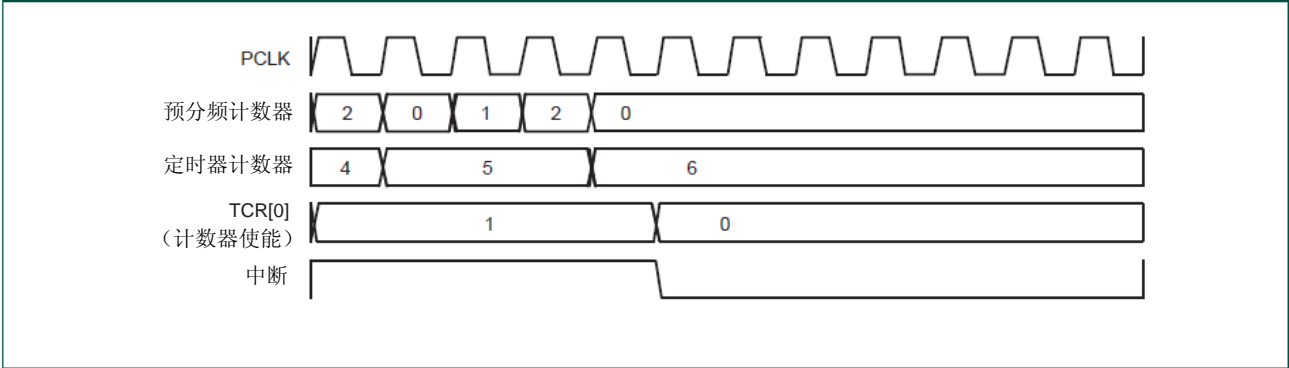


图139. 定时器周期设置为 PR=2，MRx=6，匹配时使能中断和停止



25.1 基本配置

使用下列寄存器来配置系统节拍定时器：

1. 时钟源： 在 **STCTRL** 寄存器中，选择内部 **CCLK** 或外部 **STCLK**（管脚 **P3[26]**）时钟作为时钟源。
2. 管脚： 如果选择了 **STCLK**（管脚 **P3[26]**）作为时钟源，在相关的 **IOCON** 寄存器（[8.4.1](#) 节）中使能 **STCLK** 管脚功能。
3. 中断： 利用相应的中断置位使能寄存器在 **NVIC** 中使能系统节拍定时器中断。**Systick** 中断在 **Cortex-M3** 中使用硬线连接作为系统节拍异常（系统处理程序 15）。请参阅 **NVIC** 章节（[6.1](#) 节）和本手册中的 **Cortex-M3** 用户指南（[39.4.2](#) 节）。

25.2 特性

- 10 毫秒的时间间隔。
- 专用异常向量。
- 可由 **CPU** 提供内部时钟信号或由管脚（**STCLK**）输入时钟

25.3 描述

系统节拍定时器是 **Cortex-M3** 的主要组成部分。系统节拍定时器专为操作系统或其他的系统管理软件提供 10 毫秒的间隔中断。

因为系统节拍定时器是 **Cortex-M3** 的一部分，所以提供一个可用在基于 **Cortex-M3** 的标准定时器就很容易进行软件移植。

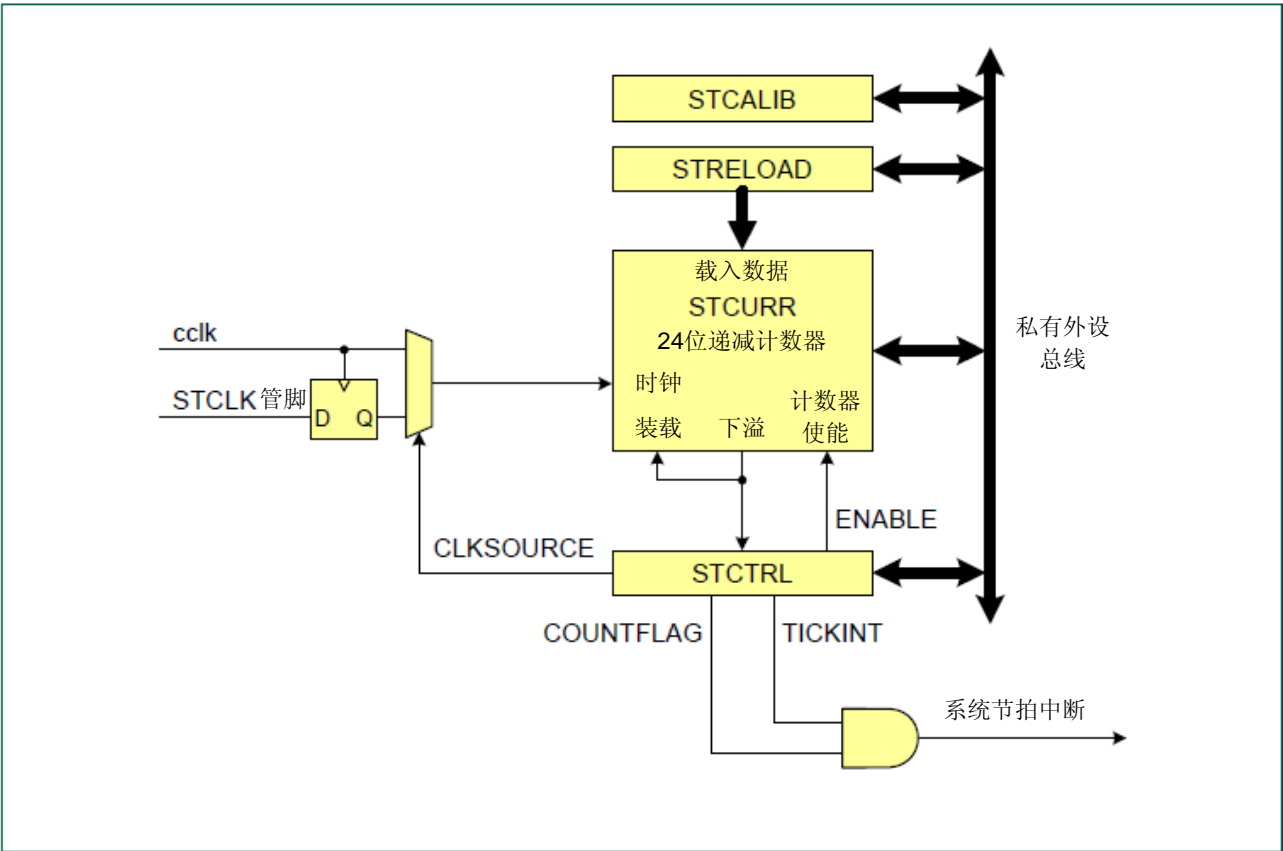
有关系统节拍定时器操作的详细信息，请参考本手册中的 **Cortex-M3** 用户指南（[39.4.4](#) 节）。

25.4 操作

系统节拍定时器是一个 24 位定时器，当计数值达到 0 时产生中断。系统节拍定时器的作用是在下次中断前提供 10 毫秒的固定时间间隔。系统节拍定时器的时钟信号可以由 **CPU** 时钟提供，也可以由外部管脚 **STCLK** 提供。**STCLK** 功能与其他功能共用管脚 **P3[26]**，而且必须选择管脚 **P3[26]** 来为系统节拍定时器提供时钟信号。要想在规定的间隔上产生中断（循环产生），必须先将指定的正确时间间隔值装入 **STRELOAD** 寄存器进行初始化。默认的时间间隔值保存在 **STCALIB** 寄存器中，软件可修改该值。如果 **CPU** 的时钟频率设置为 100MHz，那么默认的中断速率就是 10 毫秒。

系统节拍定时器框图如[图 140](#) 所示。

图140. 系统节拍定时器框图



25.5 寄存器描述

表546. 系统节拍定时器寄存器的映射

名称	描述	访问	复位值 ^[1]	地址	表
STCTRL	系统定时器控制和状态寄存器	R/W	0x4	0xE000 E010	表 547
STRELOAD	系统定时器重载值寄存器	R/W	0	0xE000 E014	表 548
STCURRE	系统定时器当前值寄存器	R/W	0	0xE000 E018	表 549
STCALIB	系统定时器校准值寄存器	R/W	0x000F 423F	0xE000 E01C	表 550

[1] 复位值仅反映已使用位中保存的数据， 不包括保留位的内容。

25.5.1 系统定时器控制和状态寄存器（STCTRL—0xE000 E010）

STCTRL 寄存器包含系统节拍定时器的控制信息，而且还可以提供状态标志。

表547. 系统定时器控制和状态寄存器的位描述（STCTRL—0xE000 E010）

位	符号	描述	复位值
0	ENABLE	系统节拍计数器使能。 为 1 时，计数器使能；为 0 时，计数器禁能。	0
1	TICKINT	系统节拍中断使能。为 1 时，系统节拍中断使能。为 0 时，系统节拍中断禁能。使能	0

位	符号	描述	复位值
		时，中断在系统节拍计数器递减至 0 时产生。	
2	CLKSOURCE	系统节拍时钟源的选择。为 1 时，CPU 时钟被选定。为 0 时，外部时钟管脚（STCLK）1 被选定。	1
15:3	-	保留。读取值未定义，只写入 0。	无
16	COUNTFLAG	系统节拍计数器标志。当系统节拍计数器值递减至 0 时该标志置位，读取该寄存器 0 时该标志被清除。	0
31:17	-	保留。读取值未定义，只写入 0。	无

25.5.2 系统定时器重载值寄存器（STRELOAD—0xE000 E014）

STRELOAD 寄存器设置了系统节拍定时器的计数值递减至 0 后要重新载入的值。在定时器进行初始化时，该寄存器由软件加载。如果 CPU 或外设时钟在某频率下很适合使用 STCALIB 的值，那么可读取 STCALIB 寄存器的值并用作 STRELOAD 的值。

表548. 系统定时器重载值寄存器的位描述（STRELOAD—0xE000 E014）

位	符号	描述	复位值
23:0	RELOAD	该值在系统节拍计数器值递减至 0 时装入该计数器。	0
31:24	-	保留。读取值未定义，只写入 0。	无

25.5.3 系统定时器当前值寄存器（STCURR—0xE000 E018）

当软件读取系统节拍计数器时，STCURR 寄存器就会返回计数器的当前计数值。

表549. 系统定时器当前值寄存器的位描述（STCURR—0xE000 E018）

位	符号	描述	复位值
23:0	CURRENT	读该寄存器会返回系统节拍计数器的当前值。写任意位会清零系统节拍计数器和 STCTRL 中的 COUNTFLAG 位。	0
31:24	-	保留。读取值未定义，只写入 0。	无

25.5.4 系统定时器校准值寄存器（STCALIB—0xE000 E01C）

STCALIB 寄存器中包含一个厂商编程的值，由引导代码（boot code）初始化而来，用于在系统节拍定时器的时钟频率为 100MHz 时每隔 10 毫秒产生一个中断。这就是 ARM 的系统节拍定时器的设计用途。通过选择正确的重装值，利用系统节拍定时器还可以在其他频率下产生中断。

表550. 系统定时器校准值寄存器的位描述（STCALIB—0xE000 E01C）

位	符号	描述	复位值
23:0	TENMS	当系统频率为 100MHz 时重载可产生 10 毫秒的系统节拍溢出率的值。该值在复位时会被初始化成厂商为 LPC178x/177x 系列选定的值。 TENMS、SKEW、和 NOREF 仅在时钟源为 CPU 或外部 STCLK（100MHz）时才使用。	0x0F423F
29:24	-	保留。 读取值未定义，只写入 0。	无
30	SKEW	显示 TENMS 是否会准确地产生 10 毫秒时间间隔的中断或接近 10 毫秒间隔的中断。该位在复位时会被初始化成厂商为 LPC178x/177x 系列选定的值。 请参见上文中 TENMS 的描述。 为 0 时，TENMS 的值准确。 为 1 时，TENMS 的值不准确。	0
31	NOREF	显示是否有可用的外部参考时钟。 该位在复位时被初始化成厂商为 LPC178x/177x 系列选定的值。 请参考上文中 TENMS 的描述。 为 0 时，有一个单独的参考时钟可用。 为 1 时，没有可用的单独的参考时钟。	0

25.6 定时器计算示例

下列示例说明如何根据不同的系统配置选择系统节拍定时器的值。 在所有示例中，都使用 10 毫秒的中断间隔进行计算，就像系统节拍定时器的设计用法一样。

示例 1)

该示例适用于使用 100MHz 的 CPU 时钟（CCLK）运行的系统节拍定时器。

STCTRL=7。当 STCTRL 为 7 时会使能定时器及其中断，并选择 CCLK 作为时钟源。
 $RELOAD = (cclk / 100) - 1 = 1,000,000 - 1 = 999,999 = 0xF423F$
在该例中，不存在舍入误差，所以结果与 CCLK 一样精确。

示例 2)

该示例适用于使用 80MHz 的 CPU 时钟（CCLK）运行的系统节拍定时器。

STCTRL=7。当 STCTRL 为 7 时会使能定时器及其中断，并选择 CCLK 作为时钟源。
 $RELOAD = (cclk / 100) - 1 = 800,000 - 1 = 799,999 = 0xC34FF$
在该例中，不存在舍入误差，所以结果与 CCLK 一样精确。

示例 3)

该示例适用于来自内部 RC 振荡器（IRC）的 CPU 时钟（CCLK），工厂校准频率为 4MHz。

STCTRL=7。当 STCTRL 为 7 时会使能定时器及其中断，并选择 CCLK 作为时钟源。
 $RELOAD = (F_{IRC} / 100) - 1 = 40,000 - 1 = 39,999 = 0x9C3F$
在该例中，不存在舍入误差，所以结果与 IRC 一样精确。

示例 4)

该示例适用于使用外部时钟源（STCLK 管脚）运行的系统节拍定时器，在该例中时钟频率为 32.768kHz。

STCTRL=3。当 STCTRL 为 3 时会使能定时器及其中断，并选择 STCLK 管脚作为时钟源。

必须选择 STCLK 提供相关管脚的功能， 参见 [8.4.1](#) 节。

$$\text{RELOAD} = (\text{cclk} / 100) - 1 = 327.6 - 1 = 327 \text{ (已舍入)} = 0x0147$$

该例中存在舍入误差，所以中断速率与输入频率存在较小的差值。

26.1 基本配置

使用下列寄存器来配置 PWM：

1. 功率：在 PCONP 寄存器 ([表 37](#)) 中置位 PCPWM1。
注：复位时，PWM1 被使能 (PCPWM1=1)，而 PWM0 被禁能 (PCPWM1=0)。
2. 外设时钟：PWM 使用公共 PCLK，它既用于总线接口，也用于大多数 APB 外设的功能部分，参见 [4.6.4](#) 节。
3. 管脚：通过相关的 IOCON 寄存器 ([8.4.1](#) 节) 来选择 PWM 管脚并为具有 PWM 功能的端口管脚选择管脚模式。
4. 中断：有关匹配和捕获事件，参见寄存器 PWMMCR ([表 557](#)) 和 PWMCCR ([表 558](#))。利用相应的中断置位使能寄存器来使能 NVIC 中的中断。

26.2 特性

- 两个 PWM 具有相同的操作特性。通过将 PWM 设置为以相同速率运行，然后同时使能这两个 PWM 可以使二者的操作同步。用于上述用途时，PWM0 作为主机，而 PWM1 作为从机。
- 计数器或定时器操作（可使用外设时钟或其中一个捕获输入作为时钟源）。
- 7 个匹配寄存器，可实现多达 6 个单沿控制或 3 个双沿控制的 PWM 输出，或两种类型的混合输出。匹配寄存器还允许：
 - 在匹配时继续操作，可选择产生中断。
 - 在匹配时停止定时器，可选择产生中断。
 - 在匹配时复位定时器，可选择产生中断。
- 支持单沿控制和/或双沿控制的 PWM 输出。单沿控制 PWM 输出在每个周期开始时总是为高电平，除非输出保持为恒定低电平。双沿控制 PWM 输出可在一个周期内的任何位置产生沿，这样可产生正或负脉冲。
- 脉冲周期和宽度可以是定时器计数的任何值，这就允许在分辨率和重复速率之间灵活地权衡。所有 PWM 输出都以相同的重复速率发生。
- 双沿控制的 PWM 输出可以编程为正脉冲或负脉冲。
- 匹配寄存器的更新与脉冲的输出同步，以便防止错误脉冲的产生。软件必须新的匹配值生效之前将它们释放。
- 在 PWM 模式没有使能时，PWM 定时器可以用作标准定时器。
- 带可编程 32 位预分频器的 32 位定时器/计数器。
- 一个捕获输入信号的跳变会触发 32 位捕获通道来取得 32 位定时器的瞬时值。捕获事件也可以选择产生中断。

26.3 描述

PWM 功能基于标准的定时器模块并继承了定时器的所有特性，但是定时器的多项功能并未输出到封装管脚。定时器可以对外设时钟（PCLK）或一个捕获输入进行周期计数，可选择产生中断或在出现指定的计数值时执行其他操作（参考七个匹配寄存器）。它还包括捕获输入，用于在输入信号跳变时保存定时器值，还可以选择在发生事件时产生中断。PWM 功能是一个附加特性，建立在匹配寄存器事件的基础之上。

PWM 可以分别控制上升沿和下降沿的位置使得它可以用于更多的应用中。例如，多相电机控制，通常需要 3 个非重叠的 PWM 输出，可单独控制 3 个输出的脉冲宽度和位置。

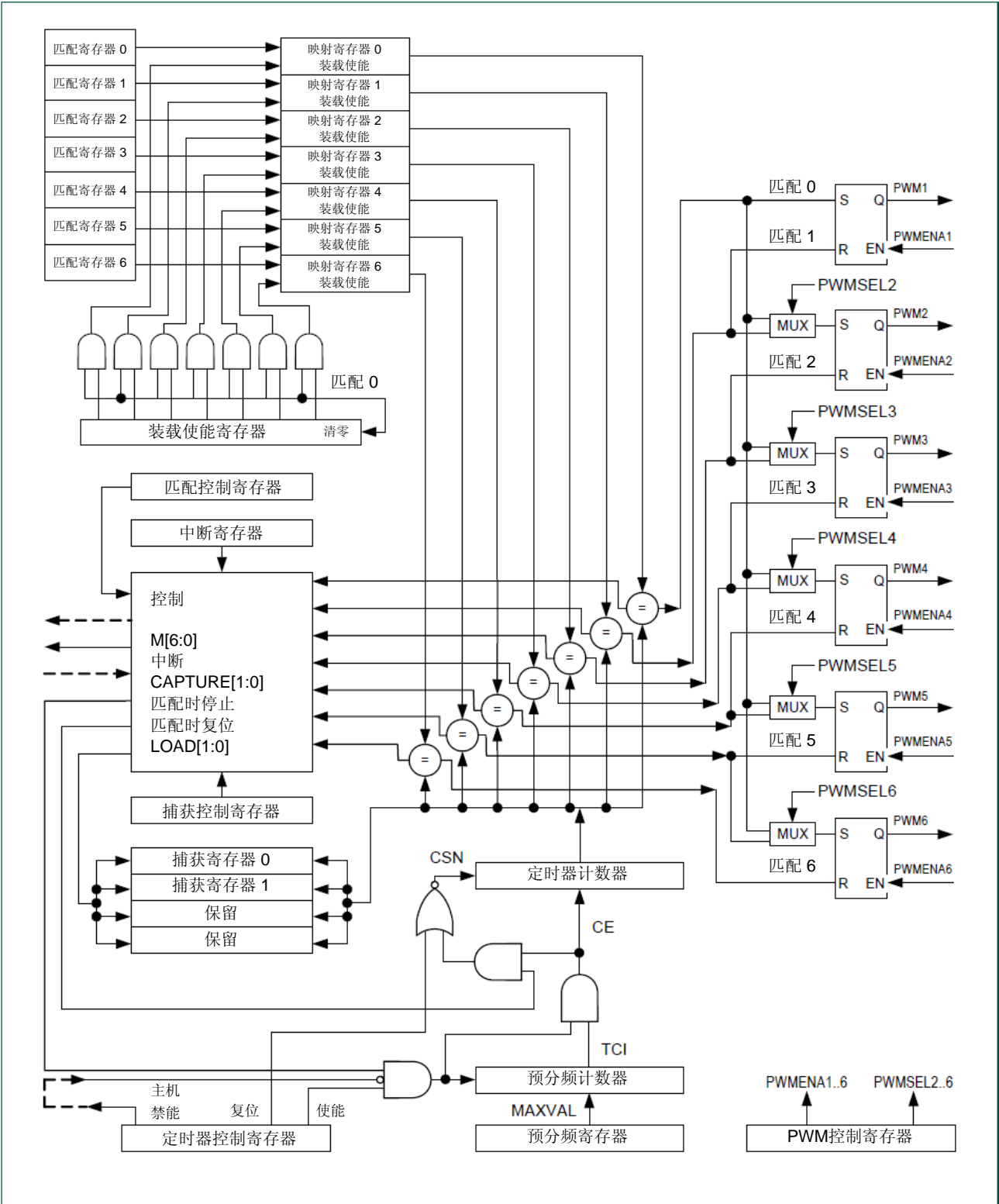
两个匹配寄存器可用于提供单沿控制的 PWM 输出。其中一个匹配寄存器（MR0）通过在匹配时将计数器复位来控制 PWM 的周期频率。另一个匹配寄存器控制 PWM 沿的位置。此外，每增加一个单沿控制的 PWM 输出，需要增加一个匹配寄存器，因为所有 PWM 输出的重复频率都相同。当 MR0 出现匹配时，几个单沿控制的 PWM 输出都会在每个 PWM 周期的开始出现上升沿。

三个匹配寄存器可用于提供双沿控制的 PWM 输出。其中，MR0 匹配寄存器控制 PWM 的周期频率。其他匹配寄存器控制两个 PWM 沿的位置。每增加一个双沿控制的 PWM 输出，需要增加两个匹配寄存器，因为所有 PWM 输出的重复频率都相同。

使用双沿控制的 PWM 输出时，指定的匹配寄存器控制输出的上升沿和下降沿。这样会产生正 PWM 脉冲（上升沿出现在下降沿之前）和负 PWM 脉冲（下降沿出现在上升沿之前）。

图 141 所示为 PWM 的框图。图的右边和图的顶端是附加到标准定时器模块的部分。图的左下方是来自定时器控制寄存器的主机使能输出，定时器控制寄存器允许主机 PWM（PWM0）在需要时同时使能自身和从机 PWM（PWM1）。来自 PWM0 的主机使能输出与两个 PWM 模块的外部使能输入相连接。

图141. PWM 模块框图



26.4 单沿和双沿控制规则的采样波形

图 142 中给出有关 PWM 值与波形输出之间关系的示例。图 141 中给出了 PWM 输出的逻辑，可利用多路复用管脚（由位 PWMSELn 控制）选择单沿控制的 PWM 输出或双沿控制的 PWM 输出。表 551 所示为不同 PWM 输出的匹配寄存器选项。支持 N-1 个单沿 PWM 输出或(N-1)/2 个双沿 PWM 输出，其中 N 为匹配寄存器以及已实现输出的数量。需要时，PWM 也可以是混合边沿类型的输出。

图142. PWM 波形样例

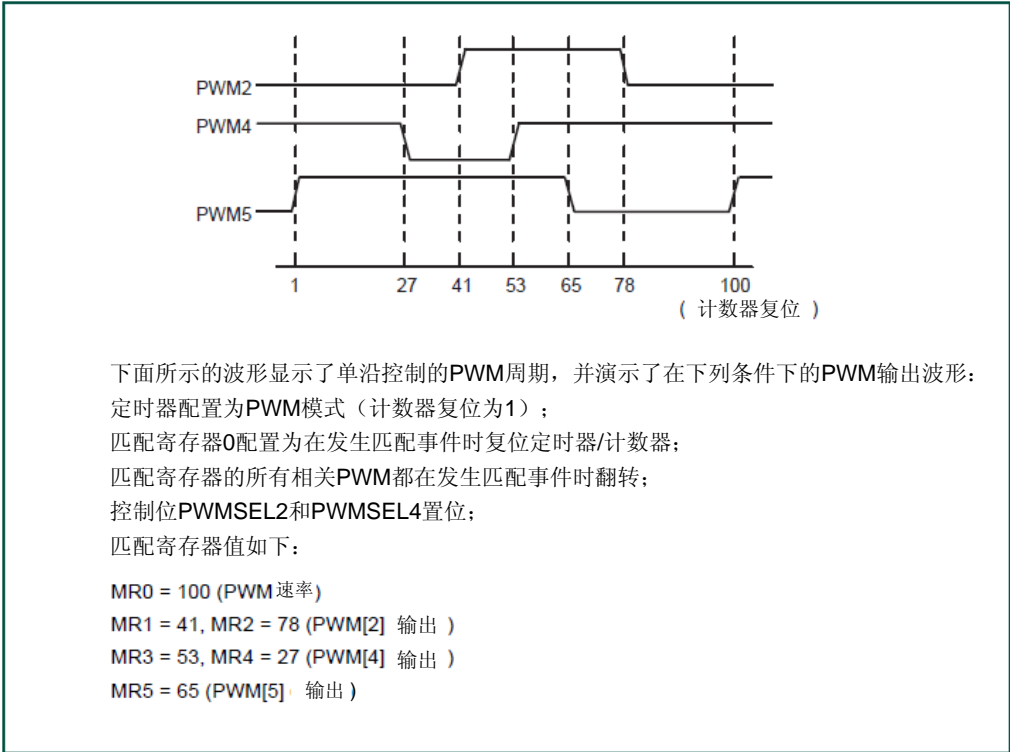


表551. PWM 触发器的置位和复位输入

PWM 通道	单沿 PWM（PWMSELn=0）		双沿 PWM （PWMSELn=1）	
	置位	复位	置位	复位
1	匹配 0	匹配 1	匹配 0 ^[1]	匹配 1 ^[1]
2	匹配 0	匹配 2	匹配 1	匹配 2
3	匹配 0	匹配 3	匹配 2 ^[2]	匹配 3 ^[2]
4	匹配 0	匹配 4	匹配 3	匹配 4
5	匹配 0	匹配 5	匹配 4 ^[2]	匹配 5 ^[2]
6	匹配 0	匹配 6	匹配 5	匹配 6

- [1] 这种情况下与单沿模式相同，因为匹配 0 是相邻的匹配寄存器。基本上 PWM1 不能用作双沿输出。
- [2] 通常不建议使用 PWM 通道 3 和 5 作为双沿 PWM 输出，因为这样会减少可用的双沿 PWM 输出的个数。使用 PWM[2]、PWM[4]和 PWM[6]可得到最多对数的双沿 PWM 输出。

26.4.1 单沿控制的 PWM 输出规则

- 1. 所有单沿控制的 PWM 输出在 PWM 周期开始时都为高电平，除非其匹配值等于 0。
- 2. 每个 PWM 输出在达到其匹配值时都会变为低电平。 如果没有发生匹配（即匹配值大于 PWM 速率），则 PWM 输出将一直保持高电平。

26.4.2 双沿控制的 PWM 输出规则

当一个新的周期要开始时，使用以下 5 个规则来决定下一个 PWM 输出的值：

- 1. 下一个 PWM 周期的匹配值在一个 PWM 周期结束时（下一个 PWM 周期开始的重合时间点）使用，例外见第 3 点规则。
- 2. 匹配值等于 0 与等于当前 PWM 速率（与匹配通道 0 的值相同）是等效的，例外见第 3 点规则。例如，在 PWM 周期开始时的下降沿请求与 PWM 周期结束时的下降沿请求等效。
- 3. 在修改匹配值时，如果“旧”匹配值中有一个等于 PWM 速率且不等于 0，并且新的匹配值不等于 0 或 PWM 速率，那么这个“旧”的匹配值将被使用一次。
- 4. 如果同时请求了 PWM 输出置位和清零，清零优先。在置位和清零匹配值相同或者当它们都等于 0 且其他值都等于 PWM 速率时这种情况才发生。
- 5. 如果匹配值超出范围（即大于 PWM 速率值），将不会发生匹配事件，匹配通道对输出不起作用。也就是说，PWM 输出将一直保持一种状态，可以为低电平、高电平或保持输出“无变化”。

26.5 管脚描述

[表 552](#) 对每个 PWM 相关的管脚进行了简要汇总。

表552. 管脚汇总

管脚	类型	描述
PWM0[6:1]	输出	PWM0 通道 6 至 1 的输出。
PWM0_CAP0	输入	PWM0 捕获输入。 当捕获管脚出现跳变时，可以将当前的定时器计数器值装入相应的捕获寄存器中，也可以选择产生一个中断。 此管脚的每个电平必须保持至少 1 个 PCLK 周期的时间以确保能被 PWM 使用。 因此，管脚的最大可用频率为 PCLK/2。
PWM1[6:1]	输出	PWM1 通道 6 至 1 的输出。
PWM1_CAP1:0	输入	PWM1 捕获输入。当捕获管脚出现跳变时，可以将当前的定时器计数器值装入相应的捕获寄存器中，也可以选择产生一个中断。

26.6 寄存器描述

表553. PWM0 和 PWM1 寄存器映射

通用名称	描述	访问	复位值 ^[1]	PWMn 寄存器的名称&地址	参考
IR	中断寄存器。可以通过写 IR 来清除中断，也可以通过读取 IR 来识别出哪个 PWM 中断源被挂起。	R/W	0	PWM0IR-0x4001 4000 PWM1IR-0x4001 8000	26.6.1 节
TCR	定时器控制寄存器。TCR 用于控制定时器计数器的功能。	R/W	0	PWM0TCR—0x4001 4004 PWM1TCR 0x4001 8004	26.6.2 节
CTCR	计数控制寄存器。CTCR 用来在定时器模式和计数器模式之间进行选择并在计数器模式中选择要计数的信号和沿。	R/W	0	PWM0CTCR-0x4001 4070 PWM1CTCR 0x4001 8070	26.6.3 节
TC	定时器计数器。32 位的 TC 每隔 PR+1 个 PCLK 周期递增一次。TC 通过 TCR 来控制。	R/W	0	PWM0TC 0x4001 4008 PWM1TC 0x4001 8008	26.6.4 节
PR	预分频寄存器。决定 PWM 计数器递增的频率。	R/W	0	PWM0PR 0x4001 400C PWM1PR 0x4001 800C	26.6.5 节
PC	预分频计数器。主 PWM 计数器的预分频器。	R/W	0	PWM0PC 0x4001 4010 PWM1PC 0x4001 8010	26.6.6 节
MCR	匹配控制寄存器。MCR 用来控制在匹配出现时是否产生中断以及 PWM 计数器是否复位。	R/W	0	PWM0MCR 0x4001 4014 PWM1MCR 0x4001 8014	26.6.7 节
MR0	匹配寄存器 0。匹配寄存器不断与 PWM 计数器的值进行比较以控制 PWM 输出沿。	R/W	0	PWM0MR0 0x4001 4018 PWM1MR0 0x4001 8018	26.6.8 节
MR1	匹配寄存器 1。参见 MR0 的描述。	R/W	0	PWM0MR1 0x4001 401C PWM1MR1 0x4001 801C	26.6.8 节
MR2	匹配寄存器 2。参见 MR0 的描述。	R/W	0	PWM0MR2 0x4001 4020 PWM1MR2 0x4001 8020	26.6.8 节
MR3	匹配寄存器 3。参见 MR0 的描述。	R/W	0	PWM0MR3 0x4001 4024 PWM1MR3 0x4001 8024	26.6.8 节
MR4	匹配寄存器 4。参见 MR0 的描述。	R/W	0	PWM0MR 0x4001 4040 PWM1MR 0x4001 8040	26.6.8 节
MR5	匹配寄存器 5。参见 MR0 的描述。	R/W	0	PWM0MR 0x4001 4044 PWM1MR 0x4001 8044	26.6.8 节
MR6	匹配寄存器 6。参见 MR0 的描述。	R/W	0	PWM0MR 0x4001 4048 PWM1MR 0x4001 8048	26.6.8 节
CCR	捕获控制寄存器。CCR 控制捕获输入的哪个沿用于装入捕获寄存器以及当捕获出现时是否生成中断。	R/W	0	PWM0CCR 0x4001 4028 PWM1CCR 0x4001 8028	26.6.9 节
CR0	捕获寄存器 0。当 PWMn_CAP0 输入上发生一个事件时，TC 值被载入 PWMn 的 CR0 中。	RO	0	PWM0CR0 0x4001 402C PWM1CR0 0x4001 802C	26.6.10 节
CR1	捕获寄存器 1。参见 CR0 的描述。	RO	0	PWM1CR1 0x4001 8030	26.6.10 节
PCR	PWM 控制寄存器。它使能 PWM 输出并将 PWM 输出通道类型选择为单沿或双沿控制。	R/W	0	PWM0PCR 0x4001 404C PWM1PCR 0x4001 804C	26.6.11 节
LER	装载使能寄存器。使能更新后的 PWM 匹配值的使用。	R/W	0	PWM0LER 0x4001 4050 PWM1LER 0x4001 8050	26.6.12 节

[1] 复位值仅反映使用位中保存的数据，不包括保留位的内容。

26.6.1 PWM 中断寄存器（PWM0IR—0x4001 4000 和 PWM1IR—0x4001 8000）

PWM 中断寄存器包含 11 个位（表 554），其中 7 个位用于匹配中断，4 个位是保留位。如果产生了中断，则 PWMIR 中的对应位将置位为高电平。否则，该位为低电平。相对应的 IR 位写入逻辑 1 会复位中断。写入 0 无效。

表554. PWM 中断寄存器位描述 (PWM0IR—0x4001 4000 和 PWM1IR—0x4001 8000)

通用名称	符号	描述	复位值
0	PWMMR0 Interrupt	PWM 匹配通道 0 的中断标志。	0
1	PWMMR1 Interrupt	PWM 匹配通道 1 的中断标志。	0
2	PWMMR2 Interrupt	PWM 匹配通道 2 的中断标志。	0
3	PWMMR3 Interrupt	PWM 匹配通道 3 的中断标志。	0
4	PWMCAP0 Interrupt	捕获输入 0 的中断标志。	0
5	PWMCAP1 Interrupt	捕获输入 1 的中断标志 (仅 PWM1IR 可用, 此位在 PWM0IR 中为保留位)。	0
7:6	-	保留。 读取值未定义, 只写入 0。	-
8	PWMMR4 Interrupt	PWM 匹配通道 4 的中断标志。	0
9	PWMMR5 Interrupt	PWM 匹配通道 5 的中断标志。	0
10	PWMMR6 Interrupt	PWM 匹配通道 6 的中断标志。	0
15:11	-	保留。 读取值未定义, 只写入 0。	无

26.6.2 PWM 定时器控制寄存器 (PWM0TCR—0x4001 4004 和 PWM1TCR—0x4001 8004)

PWM 定时器控制寄存器 (PWMTCR) 用于控制 PWM 定时器计数器的操作。 每个位的功能如表 555 所示。

表555. PWM 定时器控制寄存器位描述 (PWM0TCR—0x4001 4004 和 PWM1TCR—0x4001 8004)

位	符号	值	描述	复位值
0	Counter Enable	1	PWM 定时器计数器和 PWM 预分频计数器使能计数。	0
		0	计数器被禁止。	
1	Counter Reset	1	PWM 定时器计数器和 PWM 预分频计数器在 PCLK 的下一个正相沿上同步复位。 0 计数器在该位恢复为 0 之前保持复位状态。	0
		0	清零复位。	
2	-		保留。 读取值未定义, 只写入 0。	无
3	PWM Enable	1	PWM 模式使能 (计数器复位为 1)。 PWM 模式将映像寄存器连接到匹配寄存器。 0 只有在 PWMLER 中的相应位置位后发生的 PWM 匹配 0 事件才会使程序写入匹配寄存器的值生效。需要注意的是, 决定 PWM 速率 (PWM 匹配寄存器 0-MR0) 的 PWM 匹配寄存器必须在使能 PWM 之前设定。否则不会出现匹配事件来使映像寄存器的内容生效。	0
		0	定时器模式使能 (计数器复位为 0)。	
4	Master Disable (PWM0 only)		2 个 PWM 可能同步使用主机禁能控制位。主 PWM (PWM0 模型) 的主机禁能 0 位控制 PWM 的二级使能输入, 如图 141 所示。	0
		1	此位在从 PWM (PWM1) 中无效。	
		0	PWM0 是主机且两个 PWM 都被使能计数。	
			0 PWM 被分开使用。单独的计数器使能位用来控制 PWM。	
			保留。 读取值未定义, 只写入 0。	
7:5	-		保留。 读取值未定义, 只写入 0。	无

26.6.3 PWM 计数控制寄存器 (PWM0CTCR—0x4001 4070 和 PWM1CTCR—0x4001 8070)

计数控制寄存器 (CTCR) 用来在定时器模式和计数器模式之间进行选择, 并在计数器模式下选择要计数的管脚和沿。 每个位的功能如表 556 所示。

表556. PWM 计数控制寄存器位描述 (PWM0CTCR—0x4001 4070 和 PWM1CTCR—0x4001 8070)

位	符号	描述	复位值
1:0	Counter/Timer Mode	00: 定时器模式： 当预分频计数器与预分频寄存器值匹配时 TC 值递增。 01: 计数器模式： 通过设置位 3:2 使 TC 在 PWM_CAP 输入信号的上升沿递增。 10: 计数器模式： 通过设置位 3:2 使 TC 在 PWM_CAP 输入信号的下降沿递增。 11: 计数器模式： 通过设置位 3:2 使 TC 在 PWM_CAP 输入信号的上升沿和下降沿都递增。	00
3:2	Count Input Select	当寄存器的位 1:0 位不为 00 时，这些位选择哪个携带信号的 PWM_CAP 管脚用来使 TC 值 00 递增。 PWM0: 00=PWM0_CAP0 (其他组合保留) PWM1: 00=PWM1_CAP0, 01=PWM1_CAP1 (其他组合保留)	
7:4	-	保留。 读取值未定义，只写入 0。	无

[1] PWM_CAP 输入信号频率不得超过 PCLK/4。在通过 PWM_CAP 管脚提供 PWM 时钟时，该管脚上的信号高电平或低电平的持续时间始终不得小于 1/(2×PCLK)。

26.6.4 PWM 定时器计数器(PWM0TC—0x4001 4008 和 PWM1TC—0x4001 8008)

当预分频计数器达到其计数上限时，32 位的 PWM 定时器计数器会递增。 如果 PWMTC 在达到其上限前没有复位，则它将一直计数至 0xFFFF FFFF，然后翻转到 0x0000 0000。该事件不会导致中断产生，但在需要时，可以使用一个匹配寄存器来检测溢出。

26.6.5 PWM 预分频寄存器(PWM0PR—0x4001 400C 和 PWM1PR—0x4001 800C)

32 位 PWM 预分频寄存器规定了 PWM 预分频计数器的最大值。

26.6.6 PWM 预分频计数器寄存器(PWM0PC—0x4001 4010 和 PWM1PC—0x4001 8010)

32 位 PWM 预分频计数器在 PCLK 应用于 PWM 定时器计数器之前，使用某个常量值来控制其分频。 由此可以控制定时器分辨率与定时器溢出前的最大时间值之间的关系。 PWM 预分频计数器值在每个 PCLK 上递增一次。 当 PWM 预分频计数器值达到 PWM 预分频寄存器中存储的值时，PWM 定时器计数器会递增，而 PWM 预分频计数器在下一个 PCLK 周期被复位。 这样，当 PWMPR=0 时，PWM 定时器计数器值 (TC) 会在每个 PCLK 上递增，而当 PWMPR=1 时，在每两个 PCLK 上递增。

26.6.7 PWM 匹配控制寄存器(PWM0MCR—0x4001 4014 和 PWM1MCR—0x4001 8014)

PWM 匹配控制寄存器用于控制在某个 PWM 匹配寄存器与 PWM 定时器计数器匹配时所执行的操作。每个位的功能如表 557 所示。

表557. 匹配控制寄存器位描述 (PWM0MCR—0x4001 4014 和 PWM1MCR—0x4001 8014)

位	符号	值	描述	复位值
0	PWMMR0I	1	PWMMR0 中断： PWMMR0 与 PWMTC 的值匹配时产生一个中断。	0
		0	中断被禁止。	
1	PWMMR0R	1	PWMMR0 复位： PWMTC 与 PWMMR0 的值匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
2	PWMMR0S	1	PWMMR0 停止： PWMMR0 和 PWMTC 的值匹配时停止 PWMTC 和 PWMPC 且 PWMTCR 的位 0 置位为 0。	0
		0	该特性被禁止。	
3	PWMMR1I	1	PWMMR1 中断： PWMMR1 与 PWMTC 的值匹配时产生一个中断。	0
		0	中断被禁止。	
4	PWMMR1R	1	PWMMR1 复位： PWMTC 与 PWMMR1 的值匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
5	PWMMR1S	1	PWMMR1 停止： PWMMR1 和 PWMTC 的值匹配时停止 PWMTC 和 PWMPC 且 PWMTCR 的位 0 置位为 0。	0
		0	该特性被禁止。	
6	PWMMR2I	1	PWMMR2 中断： PWMMR2 与 PWMTC 的值匹配时产生一个中断。	0
		0	中断被禁止。	
7	PWMMR2R	1	PWMMR2 复位： PWMTC 与 PWMMR2 的值匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
8	PWMMR2S	1	PWMMR2 停止： PWMMR2 和 PWMTC 的值匹配时停止 PWMTC 和 PWMPC 且 PWMTCR 的位 0 置位为 0。	0
		0	该特性被禁止。	
9	PWMMR3I	1	PWMMR3 中断： PWMMR3 与 PWMTC 的值匹配时产生一个中断。	0
		0	中断被禁止。	
10	PWMMR3R	1	PWMMR3 复位： PWMTC 与 PWMMR3 的值匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
11	PWMMR3S	1	PWMMR3 停止： PWMMR3 和 PWMTC 的值匹配时停止 PWMTC 和 PWMPC 且 PWMTCR 的位 0 置位为 0。	0
		0	该特性被禁止。	
12	PWMMR4I	1	PWMMR4 中断： PWMMR4 与 PWMTC 的值匹配时产生一个中断。	0
		0	中断被禁止。	
13	PWMMR4R	1	PWMMR4 复位： PWMTC 与 PWMMR4 的值匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
14	PWMMR4S	1	PWMMR4 停止： PWMMR4 和 PWMTC 的值匹配时停止 PWMTC 和 PWMPC 且 PWMTCR[0]置位为 0。	0
		0	该特性被禁止。	
15	PWMMR5I	1	PWMMR5 中断： PWMMR5 与 PWMTC 的值匹配时产生一个中断。	0
		0	中断被禁止。	
16	PWMMR5R	1	PWMMR5 复位： PWMTC 与 PWMMR5 的值匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
17	PWMMR5S	1	PWMMR5 停止： PWMMR5 和 PWMTC 的值匹配时停止 PWMTC 和 PWMPC 且 PWMTCR[0]置位为 0。	0
		0	该特性被禁止。	

位	符号	值	描述	复位值
		0	该特性被禁止。	
18	PWMMR6I	1	PWMMR6 中断： PWMMR6 与 PWMTC 的值匹配时产生一个中断。	0
		0	中断被禁止。	
19	PWMMR6R	1	PWMMR6 复位： PWMTC 与 PWMMR6 的值匹配时 PWMTC 复位。	0
		0	该特性被禁止。	
20	PWMMR6S	1	PWMMR6 停止： PWMMR6 和 PWMTC 的值匹配时停止 PWMTC 和 PWMPC 且 PWMTCR[0]置位为 0。	0
		0	该特性被禁止。	
31:21	-		保留。 读取值未定义，只写入 0。	无

26.6.8 PWM 匹配寄存器（PWM0MR0~PWM0MR6， PWM1MR0~PWM1MR6）

32 位 PWM 匹配寄存器的值不断与 PWM 定时器计数器的值进行比较。 当两个值相等时，会自动触发操作。可能的操作包括产生一个中断、复位 PWM 定时器计数器或停止定时器。所执行的操作通过 PWMMCR 寄存器中的设置来控制。

26.6.9 PWM 捕获控制寄存器（PWM0CCR—0x4001 4028 和 PWM1CCR—0x4001 8028）

捕获控制寄存器用来控制在 PWM0_CAP0 或 PWM1_CAP1:0 上发生捕获事件时，是否向任意一个捕获寄存器中加载定时器计数器中的值，以及决定捕获事件是否产生中断。 上升沿和下降沿可以同时设置，这将导致在两个沿上都会发生捕获事件。 在下文的描述中，“n”代表定时器的编号 0 或 1。

注： 如果在 CTCR 中选择一个特定的 PWM_CAP 输入用于计数器模式，则该输入在本寄存器中的 3 个位都应该被编程为 000，而其他 2 个 PWM_CAP 输入可以选择进行捕获和/或产生中断。

表558. PWM 捕获控制寄存器位描述 (PWM0CCR—0x4001 4028 和 PWM1CCR—0x4001 8028)

位	符号	值	描述	复位值
0	Capture on PWMn_CAP0 rising edge	0	该特性被禁止。	0
		1	PWMn_CAP0 上同步采样到的上升沿将使 TC 的内容载入 CR0。	
1	Capture on PWMn_CAP0 falling edge	0	该特性被禁止。	0
		1	PWMn_CAP0 上同步采样到的下降沿将使 TC 的内容载入 CR0。	
2	Interrupt on PWMn_CAP0 event	0	该特性被禁止。	0
		1	PWMn_CAP0 事件所导致的 CR0 装载将产生一个中断。	
3	Capture on PWMn_CAP1 rising edge ^[1]	0	该特性被禁止。	0
		1	PWMn_CAP1 上同步采样到的上升沿将使 TC 的内容载入 CR1。	
4	Capture on PWMn_CAP1 falling edge ^[1]	0	该特性被禁止。	0
		1	PWMn_CAP1 上同步采样到的下降沿将使 TC 的内容载入 CR1。	
5	Interrupt on PWMn_CAP1 event ^[1]	0	该特性被禁止。	0
		1	PWMn_CAP1 事件所导致的 CR1 装载将产生一个中断。	
31:6	-		保留。 读取值未定义，只写入 0。	无

[1] 为 PWM0 的保留位。

26.6.10 PWM 捕获寄存器(PWM0CR0—0x4001 402C 和 PWM1CR0/CR1—0x4001 802C/30)

每个 32 位捕获寄存器都与一个设备管脚相关联，且当该管脚上发生指定事件时，可以将 PWM 定时器计数器的值加载到该寄存器中。 PWM 捕获控制寄存器中的设置决定是否使能捕获功能，以及捕获事件是发生在关联管脚的上升沿、下降沿还是同时在两个沿上发生。

26.6.11 PWM 控制寄存器(PWM0PCR—0x4001 404C 和 PWM1PCR 0x4001 804C)

PWM 控制寄存器用来使能和选择每个 PWM 通道的类型。每个位的功能如表 559 所示。

表559. PWM 控制寄存器位描述 (PWMPCR—0x4001 404C 和 PWM1PCR—0x4001 804C)

位	符号	值	描述	复位值
1:0	Unused		未使用，始终为 0。	无
2	PWMSEL2		PWM[2]输出单/双沿控制模式。	0
		1	选择双沿控制模式。	
		0	选择单沿控制模式。	
3	PWMSEL3	1	PWM[3]输出沿控制。 详见 PWMSEL2。	0
4	PWMSEL4	1	PWM[4]输出沿控制。 详见 PWMSEL2。	0
5	PWMSEL5	1	PWM[5]输出沿控制。 详见 PWMSEL2。	0
6	PWMSEL6	1	PWM[6]输出沿控制。 详见 PWMSEL2。	0
8:7	-		保留。 读取值未定义，只写入 0。	无
9	PWМЕНА1		PWM[1]输出使能控制。	0
		0	PWM 输出禁能。	

位	符号	值	描述	复位值
		1	PWM 输出使能。	
10	PWMENA2		PWM[2]输出使能控制。 详见 PWMENA1。	0
11	PWMENA3		PWM[3]输出使能控制。 详见 PWMENA1。	0
12	PWMENA4		PWM[4]输出使能控制。 详见 PWMENA1。	0
13	PWMENA5		PWM[5]输出使能控制。 详见 PWMENA1。	0
14	PWMENA6		PWM[6]输出使能控制。 详见 PWMENA1。	0
31:15	Unused		未使用，始终为 0。	无

26.6.12 PWM 锁存使能寄存器（PWM0LER—0x4001 4050 和 PWM1LER 0x4001 8050）

当用于 PWM 生成时，PWM 锁存使能寄存器用来控制 PWM 匹配寄存器的更新。 当定时器处于 PWM 模式时，如果软件对 PWM 匹配寄存器地址执行写操作，写入值实际上被保存在一个映像寄存器中，不会立即使用。

在 PWM 匹配 0 事件发生时（在 PWM 模式下，通常也会复位定时器），如果对应的锁存使能寄存器位已经置位，那么映像寄存器的内容将传送到实际的匹配寄存器中。 此时，新的值会生效并决定下一个 PWM 周期。 在发生新值传送时，LER 中的所有位会被自动清零。在 PWMLER 中相应位置位和 PWM 匹配 0 事件发生之前，任何写入 PWM 匹配寄存器的值都不会影响 PWM 操作。

例如，当 PWM 配置为双沿操作并处于运行中时，改变时序的事件的典型序列如下：

- 将新值写入 PWM 匹配 1 寄存器。
- 将新值写入 PWM 匹配 2 寄存器。
- 写 PWMLER，同时置位位 1 和位 2。
- 更改的值将在下一次定时器复位时（当 PWM 匹配 0 事件发生时）生效。

写两个 PWM 匹配寄存器的顺序并不重要，因为写入的值在写 PWMLER 之前都是无效的。 由此可确保两个值可以同时生效（如果要求）。 如果需要，可以通过同样的方法来修改单个值。

PWMLER 中每个位的功能如[表 560](#) 所示。

表560. PWM 锁存使能寄存器位描述（PWM0LER—0x4001 4050 和 PWM1LER—0x4001 8050）

位	符号	描述	复位值
0	Enable PWM Match 0 Latch	PWM MR0 寄存器更新控制。向该位写 1 允许最后写入 PWM 匹配寄存器 0 的值在由 PWM 0 匹配事件引起的下次定时器复位时生效。 详见 26.6.7 节。	
1	Enable PWM Match 1 Latch	PWM MR1 寄存器更新控制。 详见位 0。	0
2	Enable PWM Match 2 Latch	PWM MR2 寄存器更新控制。 详见位 0。	0
3	Enable PWM Match 3 Latch	PWM MR3 寄存器更新控制。 详见位 0。	0
4	Enable PWM Match 4 Latch	PWM MR4 寄存器更新控制。 详见位 0。	0
5	Enable PWM Match 5 Latch	PWM MR5 寄存器更新控制。 详见位 0。	0
6	Enable PWM Match 6 Latch	PWM MR6 寄存器更新控制。 详见位 0。	0
7		保留。读取值未定义，只写入 0。	无

27.1 基本配置

使用下列寄存器来配置电机控制 PWM：

1. 功率：在 PCONP 寄存器（[表 37](#)）中置位 PCMCPWM。
注：复位时，MCPWM 被禁能（PCMCPWM=0）。
2. 外设时钟：MCPWM 使用公共 PCLK，它既用于总线接口，也用于大多数 APB 外设的功能部分，参见 [4.6.4](#) 节。
3. 管脚：通过相关的 IOCON 寄存器（[8.4.1](#) 节）选择 MCPWM 管脚并为具有 MCPWM 功能的端口管脚选择管脚模式。
4. 中断：参见 [27.9.8](#) 节。利用相应的中断置位使能寄存器来使能 NVIC 中的中断。

27.2 简介

电机控制 PWM（MCPWM）非常适用于三相交流和直流电机控制应用，但它同样可用于需要定时、计数、捕获和比较的其他多种应用中。

27.3 描述

MCPWM 包含三个独立的通道，每个通道包括：

- 1 个 32 位定时器/计数器（TC）
- 1 个 32 位界限寄存器（LIM）
- 1 个 32 位匹配寄存器（MAT）
- 1 个 10 位死区时间寄存器（DT）和相应的 10 位死区时间计数器
- 1 个 32 位捕获寄存器（CAP）
- 2 个极性相反的已调制输出（MC_A 和 MC_B）
- 1 个周期中断、1 个脉宽中断和 1 个捕获中断

输入管脚 MC_FB0-2 可以触发 TC 捕获或使通道的 TC 值递增。一个全局中止输入会强制所有通道进入“A 无效”状态并产生一个中断。

27.4 管脚描述

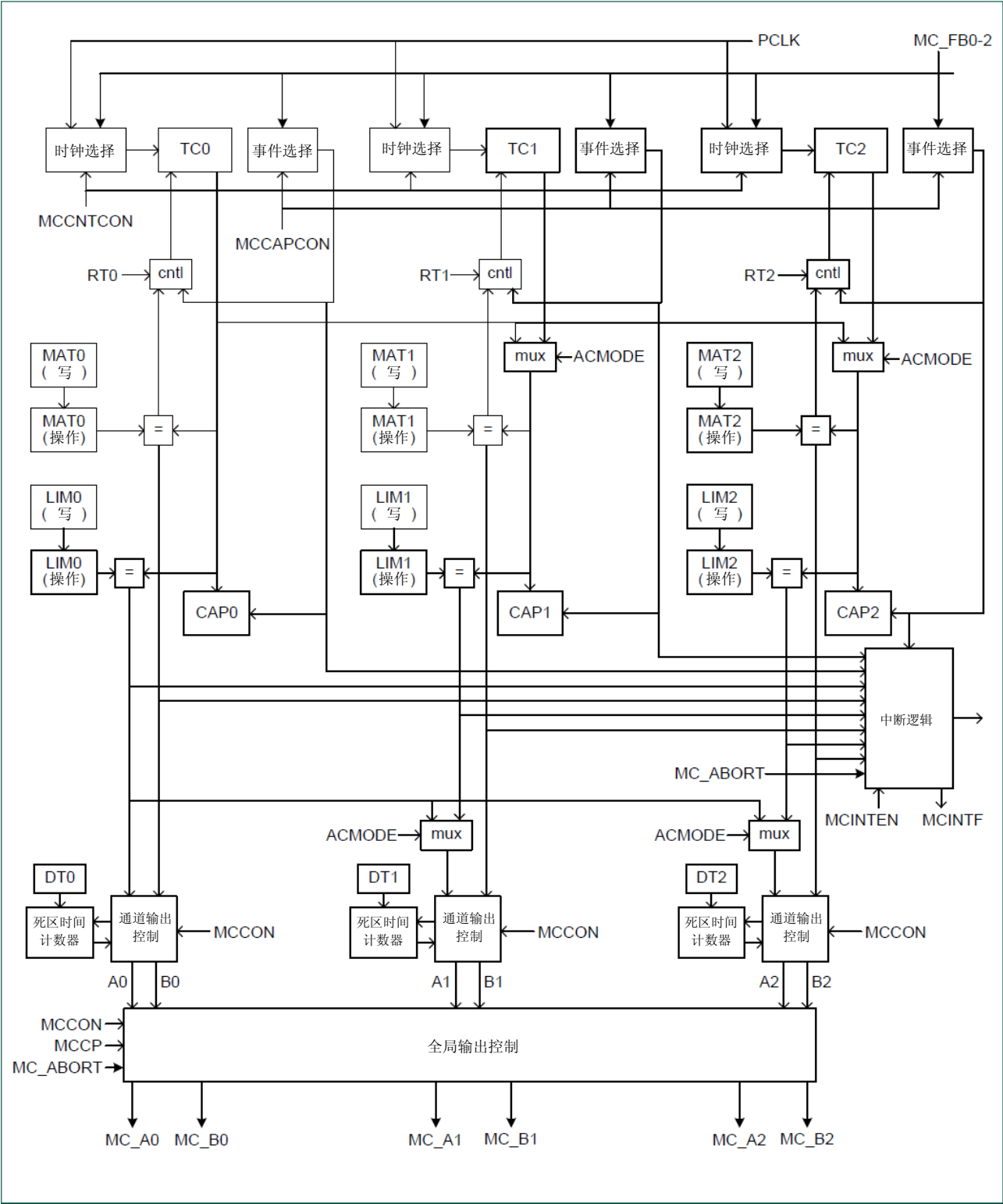
[表 561](#) 所示为 MCPWM 管脚。

表561. 管脚汇总

管脚	类型	描述
MC_0A, MC_0B	O	通道 0 的输出 A 和输出 B。
MC_1A, MC_1B	O	通道 1 的输出 A 和输出 B。
MC_2A, MC_2B	O	通道 2 的输出 A 和输出 B。
MC_ABORT	I	低电平有效的快速中止。
MC_FB0, MC_FB1, MC_FB2	I	通道 0、1、2 的输入。

27.5 模块框图

图143. MCPWM 模块框图



27.6 配置其它模块用于 MCPWM

在使用电机控制 PWM 之前配置其他模块中的寄存器，寄存器如下：

1. 功率：在 PCONP 寄存器（[表 37](#)）中置位 PCMCPWM。
注：复位时，MCPWM 被禁能（PCMCPWM=0）。
2. 外设时钟：MCPWM 使用公共 PCLK，它既用于总线接口，也用于大多数 APB 外设的功能部分，参见 [4.6.4](#) 节。
3. 管脚：通过相关的 IOCON 寄存器（[8.4.1](#) 节）选择 MCPWM 功能管脚并为这些管脚选择管脚模式。
4. 中断：有关电机控制 PWM 的相关中断，参见 [27.8.3](#) 节。利用相应的中断置位使能寄存器来使能 NVIC 中的中断。

27.7 一般操作

[27.9](#) 节包含了各种 MCPWM 操作模式的详细描述，但在此先给出一个初步的概述，作为以下寄存器描述的背景资料。

MCPWM 包括 3 个通道，每个通道均控制一对输出，接着这些输出可控制某些片外操作，例如控制电机中的一组线圈。每个通道都包括一个通过处理器时钟（定时器模式）或输入管脚（计数器模式）使之递增的定时器/计数器（TC）寄存器。

每个通道都有一个与 TC 值进行比较的界限寄存器。当出现匹配时，TC 通过两种方式之一进行“复位”。在“沿对齐模式”下，TC 复位为 0，而在“中心对齐模式”下，发生匹配时 TC 会切换到另一种状态，该状态下每经过一个处理器时钟 TC 值递减一次，直至为 0，此时它再次开始递增计数。

此外，每个通道还包括一个匹配寄存器，其值比界限寄存器的值小。在沿对齐模式下，通道的输出在 TC 与匹配寄存器或界限寄存器的值匹配时进行切换，而在中心对齐模式下，只有在 TC 与匹配寄存器的值匹配时才切换。

因此，界限寄存器控制输出的周期，而匹配寄存器控制每个输出周期内各种状态所占用的时间。如果输出叠加到电压，则在界限寄存器中保存一个小的值会最大限度地减少“波纹”，并允许 MCPWM 控制高速运行的设备。

界限寄存器中保存小值的“缺点”是它们会降低由匹配寄存器控制的占空比分辨率。如果界限寄存器中的值为 8，那么匹配寄存器只能选择 0%、12.5%、25%、...、87.5%或 100% 中的一个占空比。

一般来说，匹配值的每级分辨率是界限寄存器值除以 1。

分辨率与周期/频率之间的平衡是脉宽调制器设计始终存在的问题。

27.8 寄存器描述

MCPWM 中的多个寄存器具有单独的读、置位和清零地址。读这类寄存器的读地址（例如 MCCON）会得到每个寄存器位的状态。向置位地址（例如 MCCON_SET）写入 1 会置位该寄存器的位，而向清零地址（例如 MCCON_CLR）写入 1 会清零该寄存器的位。其他寄存器是标准的读写寄存器。

表562. 电机控制脉宽调制器（MCPWM）寄存器映射

名称	描述	访问	复位值	地址	表
MCCON	PWM 控制寄存器读地址	RO	0	0x400B 8000	表 563
MCCON_SET	PWM 控制寄存器置位地址	WO	-	0x400B 8004	表 564
MCCON_CLR	PWM 控制寄存器清零地址	WO	-	0x400B 8008	表 565
MCCAPCON	捕获控制寄存器读地址	RO	0	0x400B 800C	表 566
MCCAPCON_SET	捕获控制寄存器置位地址	WO	-	0x400B 8010	表 567
MCCAPCON_CLR	事件控制寄存器清零地址	WO	-	0x400B 8014	表 568
MCTC0	定时器计数器寄存器，通道 0。	R/W	0	0x400B 8018	表 580
MCTC1	定时器计数器寄存器，通道 1。	R/W	0	0x400B 801C	表 580
MCTC2	定时器计数器寄存器，通道 2。	R/W	0	0x400B 8020	表 580
MCLIM0	界限寄存器，通道 0。	R/W	0xFFFF FFFF	0x400B 8024	表 581
MCLIM1	界限寄存器，通道 1。	R/W	0xFFFF FFFF	0x400B 8028	表 581
MCLIM2	界限寄存器，通道 2。	R/W	0xFFFF FFFF	0x400B 802C	表 581
MCMAT0	匹配寄存器，通道 0。	R/W	0xFFFF FFFF	0x400B 8030	表 582
MCMAT1	匹配寄存器，通道 1。	R/W	0xFFFF FFFF	0x400B 8034	表 582
MCMAT2	匹配寄存器，通道 2。	R/W	0xFFFF FFFF	0x400B 8038	表 582
MCDT	死区时间寄存器。	R/W	0x3FFF FFFF	0x400B 803C	表 583
MCCP	通信格式寄存器。	R/W	0	0x400B 8040	表 584
MCCAP0	捕获寄存器，通道 0。	RO	0	0x400B 8044	表 585
MCCAP1	捕获寄存器，通道 1。	RO	0	0x400B 8048	表 585
MCCAP2	捕获寄存器，通道 2。	RO	0	0x400B 804C	表 585
MCINTEN	中断使能寄存器读地址。	RO	0	0x400B 8050	表 571
MCINTEN_SET	中断使能寄存器置位地址。	WO	-	0x400B 8054	表 572
MCINTEN_CLR	中断使能寄存器清零地址。	WO	-	0x400B 8058	表 573
MCCNTCON	计数控制寄存器读地址。	RO	0	0x400B 805C	表 577
MCCNTCON_SET	计数控制寄存器置位地址。	WO	-	0x400B 8060	表 578
MCCNTCON_CLR	计数控制寄存器清零地址。	WO	-	0x400B 8064	表 579
MCINTF	中断标志寄存器读地址。	RO	0	0x400B 8068	表 574
MCINTF_SET	中断标志寄存器置位地址。	WO	-	0x400B 806C	表 575
MCINTF_CLR	中断标志寄存器清零地址。	WO	-	0x400B 8070	表 576
MCCAP_CLR	捕获寄存器清零地址。	WO	-	0x400B 8074	表 586

27.8.1 MCPWM 控制寄存器

27.8.1.1 MCPWM 控制寄存器读地址（MCCON—0x400B 8000）

MCCON 寄存器控制所有 PWM 通道的操作。该地址是只读的，但可以通过写地址 MCCON_SET 和 MCCON_CLR 来修改下面的寄存器。

表563. MCPWM 控制寄存器读地址位描述（MCCON—0x400B 8000）

位	符号	值	描述	复位值
0	RUN0		停止/启动定时器通道 0。	0
		0	停止。	
		1	运行。	
1	CENTER0		通道 0 的沿/中心对齐操作。	0
		0	沿对齐。	
		1	中心对齐。	
2	POLA0		选择 MC_0A 和 MC_0B 管脚的极性。	0
		0	无效的状态为低电平，有效的状态为高电平。	
		1	无效的状态为高电平，有效的状态为低电平。	
3	DTE0		控制通道 0 的死区时间特性。	0
		0	死区时间被禁止。	
		1	死区时间使能。	
4	DISUP0		使能/禁止通道 0 的功能寄存器更新（参见 27.9.2 节）。	0
		0	在每个 PWM 周期结束时用写寄存器的值更新功能寄存器。	
		1	只要定时器在运行，功能寄存器保持原样。	
7:5	-		保留。 读取值未定义，只写入 0。	
8	RUN1		停止/启动定时器通道 1。	0
		0	停止。	
		1	运行。	
9	CENTER1		通道 1 的沿/中心对齐。	0
		0	沿对齐。	
		1	中心对齐。	
10	POLA1		选择 MC_1A 和 MC_1B 管脚的极性。	0
		0	无效的状态为低电平，有效的状态为高电平。	
		1	无效的状态为高电平，有效的状态为低电平。	
11	DTE1		控制通道 1 的死区时间特性。	0
		0	死区时间被禁止。	
		1	死区时间使能。	
12	DISUP1		使能/禁止通道 1 的功能寄存器更新（参见 27.9.2 节）。	0
		0	在每个 PWM 周期结束时用写寄存器的值更新功能寄存器。	
		1	只要定时器在运行，功能寄存器保持原样。	
15:13	-		保留。 从保留位读取的值未定义。	0
16	RUN2		停止/启动定时器通道 2。	0
		0	停止。	
		1	运行。	
17	CENTER2		通道 2 的沿/中心对齐。	0
		0	沿对齐。	
		1	中心对齐。	

位	符号	值	描述	复位值
18	POLA2		选择 MC_2A 和 MC_2B 管脚的极性。	0
		0	无效的状态为低电平，有效的状态为高电平。	
		1	无效的状态为高电平，有效的状态为低电平。	
19	DTE2		控制通道 1 的死区时间特性。	0
		0	死区时间被禁止。	
		1	死区时间使能。	
20	DISUP2		使能/禁止通道 2 的功能寄存器更新（参见 27.9.2 节）。	0
		0	在每个 PWM 周期结束时用写寄存器的值更新功能寄存器。	
		1	只要定时器在运行，功能寄存器保持原样。	
28:21	-		保留。 读取值未定义，只写入 0。	
29	INVBDC		控制全部 3 个通道的 MC_B 输出的极性。该位仅在三相 DC 模式中才置位为 1。	
		0	MC_B 输出与 MC_A 输出的极性相反（除死区时间）。	
		1	MC_B 输出与 MC_A 输出的基本极性相同（参见 27.9.6 节）。	
30	ACMODE		三相 AC 模式选择（参见 27.9.7 节）。	0
		0	三相 AC 模式关闭： 每个 PWM 通道使用其自身的定时器计数器和周期寄存器。	
		1	三相 AC 模式打开： 所有 PWM 通道都使用通道 0 的定时器计数器和周期寄存器。	
31	DCMODE		三相 DC 模式选择（参见 27.9.6 节）。	0
		0	三相 DC 模式关闭： PWM 通道独立（除非位 ACMODE=1）。	
		1	三相 DC 模式打开： 内部 MC_0A 输出通过 M CCP（也就是屏蔽）寄存器连接到所有 6 个 PWM 输出。	

27.8.1.2 MCPWM 控制寄存器置位地址（MCCON_SET—0x400B 8004）

向该只写地址写入 1 会置位 MCCON 中的相应位。

表564. MCPWM 控制寄存器置位地址位描述（MCCON_SET—0x400B 8004）

位	描述
31:0	向该地址写入 1 会置位 MCCON 寄存器中的相应位， 参见表 563。

27.8.1.3 MCPWM 控制寄存器清零地址（MCCON_CLR—0x400B 8008）

向该只写地址写入 1 会清零 MCCON 中的相应位。

表565. MCPWM 控制寄存器清零地址位描述（MCCON_CLR—0x400B 8008）

位	描述
31:0	向该地址写入 1 会清零 MCCON 寄存器中的相应位， 参见表 563。

27.8.2 MCPWM 捕获控制寄存器

27.8.2.1 MCPWM 捕获控制寄存器读地址（MCCAPCON—0x400B 800C）

MCCAPCON 寄存器控制所有 MCPWM 通道上 MC_FB0-2 输入的事件检测。三个 MC_FB 输入中的任意一个都可以用来触发任意一个或所有三个通道上的捕获事件。该地址是只读的，但可以通过写地址 MCCAPCON_SET 和 MCCAPCON_CLR 来修改下面的寄存器。

表566. MCPWM 捕获控制寄存器读地址位描述（MCCAPCON—0x400B 800C）

位	符号	描述	复位值
0	CAP0MCFB0_RE	为 1 时，在 MC_FB0 的上升沿上使能一个通道 0 的捕获事件。	0
1	CAP0MCFB0_FE	为 1 时，在 MC_FB0 的下降沿上使能一个通道 0 的捕获事件。	0
2	CAP0MCFB1_RE	为 1 时，在 MC_FB1 的上升沿上使能一个通道 0 的捕获事件。	0
3	CAP0MCFB1_FE	为 1 时，在 MC_FB1 的下降沿上使能一个通道 0 的捕获事件。	0
4	CAP0MCFB2_RE	为 1 时，在 MC_FB2 的上升沿上使能一个通道 0 的捕获事件。	0
5	CAP0MCFB2_FE	为 1 时，在 MC_FB2 的下降沿上使能一个通道 0 的捕获事件。	0
6	CAP1MCFB0_RE	为 1 时，在 MC_FB0 的上升沿上使能一个通道 1 的捕获事件。	0
7	CAP1MCFB0_FE	为 1 时，在 MC_FB0 的下降沿上使能一个通道 1 的捕获事件。	0
8	CAP1MCFB1_RE	为 1 时，在 MC_FB1 的上升沿上使能一个通道 1 的捕获事件。	0
9	CAP1MCFB1_FE	为 1 时，在 MC_FB1 的下降沿上使能一个通道 1 的捕获事件。	0
10	CAP1MCFB2_RE	为 1 时，在 MC_FB2 的上升沿上使能一个通道 1 的捕获事件。	0
11	CAP1MCFB2_FE	为 1 时，在 MC_FB2 的下降沿上使能一个通道 1 的捕获事件。	0
12	CAP2MCFB0_RE	为 1 时，在 MC_FB0 的上升沿上使能一个通道 2 的捕获事件。	0
13	CAP2MCFB0_FE	为 1 时，在 MC_FB0 的下降沿上使能一个通道 2 的捕获事件。	0
14	CAP2MCFB1_RE	为 1 时，在 MC_FB1 的上升沿上使能一个通道 2 的捕获事件。	0
15	CAP2MCFB1_FE	为 1 时，在 MC_FB1 的下降沿上使能一个通道 2 的捕获事件。	0
16	CAP2MCFB2_RE	为 1 时，在 MC_FB2 的上升沿上使能一个通道 2 的捕获事件。	0
17	CAP2MCFB2_FE	为 1 时，在 MC_FB2 的下降沿上使能一个通道 2 的捕获事件。	0
18	RT0	若该位为 1，TC0 在出现通道 0 捕获事件时复位。	0
19	RT1	若该位为 1，TC1 在出现通道 1 捕获事件时复位。	0
20	RT2	若该位为 1，TC2 在出现通道 2 捕获事件时复位。	0
21	HNFCAP0	硬件噪声滤波器： 若该位为 1，通道 0 上的捕获事件延迟，参见 27.9.4 节。	0
22	HNFCAP1	硬件噪声滤波器： 若该位为 1，通道 1 上的捕获事件延迟，参见 27.9.4 节。	0
23	HNFCAP2	硬件噪声滤波器： 若该位为 1，通道 2 上的捕获事件延迟，参见 27.9.4 节。	0
31: 24	-	保留。 从保留位读取的值未定义。	-

27.8.2.2 MCPWM 捕获控制寄存器置位地址（MCCAPCON_SET—0x400B 8010）

向该只写地址写入 1 会置位 MCCAPCON 中的相应位。

表567. MCPWM 捕获控制寄存器置位地址位描述（MCCAPCON_SET—0x400B 8010）

位	描述
31:0	向该地址写入 1 会置位 MCCAPCON 寄存器中的相应位， 参见表 566。

27.8.2.3 MCPWM 捕获控制寄存器清零地址（MCCAPCON_CLR—0x400B 8014）

向该只写地址写入 1 会清零 MCCAPCON 中的相应位。

表568. MCPWM 捕获控制寄存器清零地址位描述（MCCAPCON_CLR—地址 0x400B 8014）

位	描述
31:0	向该地址写入 1 会清零 MCCAPCON 寄存器中的相应位，参见表 566。

27.8.3 MCPWM 中断寄存器

电机控制 PWM 模块包括下列中断源：

表569. 电机控制 PWM 中断

符号	描述
ILIM0/1/2	通道 0、1、2 的界限中断。
IMAT0/1/2	通道 0、1、2 的匹配中断。
ICAP0/1/2	通道 0、1、2 的捕获中断。
ABORT	快速中止中断。

所有 MCPWM 中断寄存器都有一个位对应于各个中断源，如表 570 所示。

表570. 中断源位分配表

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	-	-	-	-
位	23	22	21	20	19	18	17	16
符号	-	-	-	-	-	-	-	-
位	15	14	13	12	11	10	9	8
符号	ABORT	-	-	-	-	ICAP2	IMAT2	ILIM2
位	7	6	5	4	3	2	1	0
符号	-	ICAP1	IMAT1	ILIM1	-	ICAP0	IMAT0	ILIM0

27.8.3.1 MCPWM 中断使能寄存器读地址（MCINTEN—0x400B 8050）

MCINTEN 寄存器控制要使能哪一个 MCPWM 中断。该地址是只读的，但可以通过写地址 MCINTEN_SET 和 MCINTEN_CLR 来修改下面的寄存器。

表571. MCPWM 中断使能寄存器读地址位描述（MCINTEN—0x400B 8050）

位	值	描述	复位值
31:0		关于各位的分配情况，参见表 570。	0
	0	中断被禁能。	
	1	中断被使能。	

27.8.3.2 MCPWM 中断使能寄存器置位地址（MCINTEN_SET—0x400B 8054）

向该只写地址写入 1 会置位 MCINTEN 中的相应位，从而使能中断。

表572. PWM 中断使能寄存器置位地址位描述（MCINTEN_SET—地址 0x400B 8054）

位	描述
31:0	向该地址写入 1 会置位 MCINTEN 寄存器中的相应位，从而使能中断， 参见表 570。

27.8.3.3 MCPWM 中断使能寄存器清零地址（MCINTEN_CLR—0x400B 8058）

向该只写地址写入 1 会清零 MCINTEN 中的相应位，从而禁用中断。

表573. PWM 中断使能寄存器清零地址位描述（MCINTEN_CLR—地址 0x400B 8058）

位	描述
31:0	向该地址写入 1 会清零 MCINTEN 寄存器中的相应位，从而禁用中断，参见表 570。

27.8.3.4 MCPWM 中断标志寄存器读地址（MCINTF—0x400B 8068）

MCINTF 寄存器包含所有 MCPWM 中断标志，这些标志在发生相应的硬件事件时置位，或在写 1 到 MCINTF_SET 地址时被置位。当 MCINTEN 和该寄存器的相应位都为 1 时，MCPWM 向中断控制器模块提交其中断请求。该地址是只读的，但可以通过写 1 到地址 MCINTF_SET 和 MCINTF_CLR 来修改下面的寄存器中的位。

表574. MCPWM 中断标志寄存器读地址位描述（MCINTF—0x400B 8068）

位	值	描述	复位值
31:0		关于各位的分配情况，参见表 570。	0
	0	该中断源不用于 MCPWM 中断请求。	
	1	如果 MCINTEN 中的相应位为 1，MCPWM 模块就会将其中断请求提交到中断控制器。	

27.8.3.5 MCPWM 中断标志寄存器置位地址（MCINTF_SET—0x400B 806C）

向该只写地址写入 1 会置位 MCINTF 中的相应位，从而有可能模拟硬件中断。

表575. MCPWM 中断标志寄存器置位地址位描述（MCINTF_SET—0x400B 806C）

位	描述
31:0	向这个只写地址写入 1 可置位 MCINTF 寄存器中的相应位，这样的话可能会模拟硬件中断，参见表 570。

27.8.3.6 MCPWM 中断标志寄存器清零地址（MCINTF_CLR—0x400B 8070）

向该只写地址写入 1 会置位 MCINTF 中的相应位，从而清除相应的中断请求。该操作通常在中断服务程序中进行。

表576. MCPWM 中断标志寄存器清零地址位描述（MCINTF_CLR—0x400B 8070）

位	描述
31:0	向这个只写地址写入 1 可置位 MCINTF 寄存器中的相应位，从而清除相应的中断请求， 参见表 570。

27.8.4 MCPWM 计数控制寄存器

27.8.4.1 MCPWM 计数控制寄存器读地址（MCCNTCON—0x400B 805C）

MCCNTCON 寄存器控制 MCPWM 通道是处于定时器模式还是计数器模式，以及在计数器模式下计数器是在其中一个或全部三个 MC_FB 输入的上升沿和/或下降沿递增计数。如果选择了定时器模式，则计数器值随 PCLK 时钟递增。

该地址是只读的。要置位或清零寄存器的位，可向 MCCNTCON_SET 或 MCCNTCON_CLR 地址写入 1。

表577. MCPWM 计数控制寄存器读地址位描述（MCCNTCON—0x400B 805C）

位	符号	值	描述	复位值
0	TC0MCFB0_RE	0	MC_FB0 的上升沿不影响计数器 0。	0
		1	如果 MODE0 为 1，计数器 0 在 MC_FB0 的上升沿上递增计数。	
1	TC0MCFB0_FE	0	MC_FB0 的下降沿不影响计数器 0。	0
		1	如果 MODE0 为 1，计数器 0 在 MC_FB0 的下降沿上递增计数。	0
2	TC0MCFB1_RE	0	MC_FB1 的上升沿不影响计数器 0。	0
		1	如果 MODE0 为 1，计数器 0 在 MC_FB1 的上升沿上递增计数。	
3	TC0MCFB1_FE	0	MC_FB1 的下降沿不影响计数器 0。	0
		1	如果 MODE0 为 1，计数器 0 在 MC_FB1 的下降沿上递增计数。	
4	TC0MCFB2_RE	0	MC_FB2 的上升沿不影响计数器 0。	0
		1	如果 MODE0 为 1，计数器 0 在 MC_FB2 的上升沿上递增计数。	
5	TC0MCFB2_FE	0	MC_FB2 的下降沿不影响计数器 0。	0
		1	如果 MODE0 为 1，计数器 0 在 MC_FB2 的下降沿上递增计数。	
6	TC1MCFB0_RE	0	MC_FB0 的上升沿不影响计数器 1。	0
		1	如果 MODE1 为 1，计数器 1 在 MC_FB0 的上升沿上递增计数。	
7	TC1MCFB0_FE	0	MC_FB0 的下降沿不影响计数器 1。	0
		1	如果 MODE1 为 1，计数器 1 在 MC_FB0 的下降沿上递增计数。	
8	TC1MCFB1_RE	0	MC_FB1 的上升沿不影响计数器 1。	0
		1	如果 MODE1 为 1，计数器 1 在 MC_FB1 的上升沿上递增计数。	
9	TC1MCFB1_FE	0	MC_FB1 的下降沿不影响计数器 1。	0
		1	如果 MODE1 为 1，计数器 1 在 MC_FB1 的下降沿上递增计数。	
10	TC1MCFB2_RE	0	MC_FB2 的上升沿不影响计数器 1。	0
		1	如果 MODE1 为 1，计数器 1 在 MC_FB2 的上升沿上递增计数。	
11	TC1MCFB2_FE	0	MC_FB2 的下降沿不影响计数器 1。	0
		1	如果 MODE1 为 1，计数器 1 在 MC_FB2 的下降沿上递增计数。	
12	TC2MCFB0_RE	0	MC_FB0 的上升沿不影响计数器 2。	0
		1	如果 MODE2 为 1，计数器 2 在 MC_FB0 的上升沿上递增计数。	
13	TC2MCFB0_FE	0	MC_FB0 的下降沿不影响计数器 2。	0
		1	如果 MODE2 为 1，计数器 2 在 MC_FB0 的下降沿上递增计数。	
14	TC2MCFB1_RE	0	MC_FB1 的上升沿不影响计数器 2。	0
		1	如果 MODE2 为 1，计数器 2 在 MC_FB1 的上升沿上递增计数。	
15	TC2MCFB1_FE	0	MC_FB1 的下降沿不影响计数器 2。	0
		1	如果 MODE2 为 1，计数器 2 在 MC_FB1 的下降沿上递增计数。	
16	TC2MCFB2_RE	0	MC_FB2 的上升沿不影响计数器 2。	0
		1	如果 MODE2 为 1，计数器 2 在 MC_FB2 的上升沿上递增计数。	

位	符号	值	描述	复位值
17	TC2MCFB2_FE	0	MC_FB2 的下降沿不影响计数器 2。	0
		1	如果 MODE2 为 1，计数器 2 在 MC_FB2 的下降沿上递增计数。	
28:18	-		保留。读取值未定义，只写入 0。	-
29	CNTR0	0	通道 0 为定时器模式。	0
		1	通道 0 为计数器模式。	
30	CNTR1	0	通道 1 为定时器模式。	0
		1	通道 1 为计数器模式。	
31	CNTR2	0	通道 2 为定时器模式。	0
		1	通道 2 为计数器模式。	

27.8.4.2 MCPWM 计数控制寄存器置位地址（MCCNTCON_SET—0x400B 8060）

向该只写地址写入 1 会置位 MCCNTCON 中的相应位。

表578. MCPWM 计数控制寄存器置位地址位描述（MCCNTCON_SET—0x400B 8060）

位	描述
31:0	向该只写地址写入 1 会置位 MCCNTCON 寄存器中的相应位，参见表 577。

27.8.4.3 MCPWM 计数控制寄存器清零地址（MCCNTCON_CLR—0x400B 8064）

向该只写地址写入 1 会清零 MCCNTCON 中的相应位。

表579. MCPWM 计数控制寄存器清零地址位描述（MCCAPCON_CLR—0x400B 8064）

位	描述
31:0	向该只写地址写入 1 会清零 MCCNTCON 寄存器中的相应位，参见表 577。

27.8.5 MCPWM 定时器/计数器 0~2 寄存器（MCTC0~2—0x400B 8018, 0x400B 801C, 0x400B 8020）

这些寄存器包含通道 0~2 的 32 位计数器/定时器的当前值。根据 MCCNTCON 的选择，每个值会在每个 PCLK 上递增，或在 MC_FB0-2 管脚的沿上递增。定时器/计数器从 0 开始递增计数直至它达到其相应的 MCLIM 寄存器的值为止(或通过写 MCON_CLR 来停止计数)。

TC 寄存器可以随时读取，但仅当其通道停止时才可以被写入。否则，写操作不会执行，也不会生成异常。

表580. MCPWM 定时器/计数器 0~2 寄存器位描述（MCTC0~2—0x400B 8018, 0x400B 801C, 0x400B 8020）

位	符号	描述	复位值
31:0	MCTC0/1/2	通道 0、1、2 的定时器/计数器值	0

27.8.6 MCPWM 界限 0~2 寄存器（MCLIM0~2—0x400B 8024, 0x400B 8028, 0x400B 802C）

这些寄存器保存了定时器/计数器 0~2 的界限值。当定时器/计数器达到其相应界限值时：1）在沿对齐模式下，TC 复位，然后从 0 开始计数；2）在中心对齐模式下，TC 开始从该值向 0 递减计数，然后再从 0 开始递增计数。

如果 MCON 中通道的 CENTER 位为 0，则选择沿对齐模式，那么当 TC 与 LIM 匹配时通道的 A 输出从“有效”切换到“无效”状态。如果 MCON 中通道的 CENTER 和 DTE 位都为 0，那么上述匹配同时会将通道 B 输出从“无效”切换到“有效”状态。

如果通道的 CENTER 位为 0，但 DTE 位为 1，则上述匹配会触发通道死区时间计数器开始计数—当死区时间计数器终止时，通道的 B 输出从“无效”切换到“有效”状态。

在中心对齐模式下，通道的 TC 和 LIM 寄存器之间的匹配对其 A 和 B 输出没有影响。

写界限寄存器或匹配（见 27.8.7 节）寄存器都会将写入值装载到“写”寄存器，对于停止的通道，它还会装载到与 TC 比较的“操作”寄存器。对于运行通道，如果 MCON 中的“禁止更新”位为 0，那么操作寄存器载入写寄存器的值，如下所示：1）在沿对齐模式下，当 TC 与操作界限寄存器匹配时；2）在中心对齐模式下，当 TC 递减计数到 0 时。如果通道正在运行而且“禁止更新”位为 1，那么操作寄存器不会从写寄存器中载入，直到软件停止该通道。

读取 MCLIM 地址总是返回操作值。

表581. MCPWM 界限 0~2 寄存器位描述（MCLIM0~2—0x400B 8024, 0x400B 8028, 0x400B 802C）

位	符号	描述	复位值
31:0	MCLIM0/1/2	TC0、1、2 的界限值	0xFFFF FFFF

注：在定时器模式下，界限寄存器决定通道的已调制 MC_A 和 MC_B 输出的周期，匹配寄存器决定周期开始处的脉宽。您可以将界限寄存器和匹配寄存器分别当作“周期寄存器”和“脉宽寄存器”。

27.8.7 MCPWM 匹配 0~2 寄存器（MCMAT0~2—0x400B 8030, 0x400B 8034, 0x400B 8038）

这些寄存器和上述界限寄存器一样，也具有“写”和“操作”模式。操作寄存器同样也可以与通道的 TC 相比较。有关读和写界限寄存器和匹配寄存器的详细信息，参见 27.8.6 节。

匹配和界限寄存器控制着 MC_A 和 MC_B 输出。要使匹配寄存器在其通道上有效，其包含的值必须比相应的界限寄存器的值小。

表582. MCPWM 匹配 0~2 寄存器位描述（MCMAT0~2—地址 0x400B 8030, 0x400B 8034, 0x400B 8038）

位	符号	描述	复位值
31:0	MCMAT0/1/2	TC0、1、2 的匹配值。	0xFFFF FFFF

27.8.7.1 沿对齐模式下的匹配寄存器

如果 MCCON 中通道的 CENTER 位为 0，则选择沿对齐模式，那么 TC 与 MAT 之间的匹配将使通道的 B 输出从“有效”切换到“无效”状态。如果 MCCON 中通道的 CENTER 和 DTE 位都为 0，那么上述匹配同时会使通道的 A 输出从“无效”切换到“有效”状态。

如果通道的 CENTER 位为 0，但 DTE 位为 1，则上述匹配会触发通道的死区时间计数器开始计数—当死区时间计数器终止时，通道的 A 输出从“无效”切换到“有效”状态。

27.8.7.2 中心对齐模式下的匹配寄存器

如果 MCCON 中通道的 CENTER 位为 1，会选择中心对齐模式，那么当 TC 递增时 TC 与 MAT 之间的匹配将使通道的 B 输出从“有效”切换到“无效”状态，而当 TC 递减时匹配将使通道的 A 输出从“有效”切换到“无效”状态。如果 MCCON 中通道的 CENTER 位为 1，但 DTE 位为 0，那么上述匹配将同时反方向切换通道的其他输出。

如果通道的 CENTER 和 DTE 位都为 1，那么 TC 和 MAT 匹配会触发通道的死区时间计数器开始计数—当死区时间计数器终止时，如果 TC 在匹配时递增计数，则通道的 B 输出从“无效”切换到“有效”状态；如果 TC 在匹配时递减计数，则通道的 A 输出从“无效”切换到“有效”状态。

27.8.7.3 0 和 100%占空比

要使通道的 MC_A 和 MC_B 输出锁定在“B 有效，A 无效”这个状态，只需要写入较大的值到它的匹配寄存器，且该值必须大于写入界限寄存器的值。这样匹配就不会发生。

要将通道的 MC_A 和 MC_B 输出锁定在相反的“A 有效，B 无效”状态，只需写 0 到其匹配寄存器。

27.8.8 MCPWM 死区时间寄存器（MCDT—0x400B 803C）

该寄存器保存三个通道的死区时间值。如果 MCON 中通道的 DTE 位为 1 以使能它的死区时间计数器，那么在其通道输出从“有效”状态变为“无效”状态时，计数器从该值开始递减计数。当死区时间计数器到达 0 时，通道的其他输出从“无效”状态变为“有效”状态。

死区时间的操作特性是功率晶体管（例如，在电机控制应用中由 A 和 B 输出驱动的功率晶体管）完全断开所需的时间比导通的时间更长。如果 A 和 B 晶体管同时打开，那么浪费且具有破坏性的电流将通过晶体管在电源导轨之间流动。在这些应用中，用 PCLK 周期的数量来编程死区时间寄存器，PCLK 周期数大于或等于晶体管的最大关断时间减去其最小的导通时间。

表583. MCPWM 死区时间寄存器位描述（MCDT—地址 0x400B 803C）

位	符号	描述	复位值
9:0	DT0	通道 0 的死区时间 ^[1] 。	0x3FF
19:10	DT1	通道 1 的死区时间 ^[2] 。	0x3FF
29:20	DT2	通道 2 的死区时间 ^[2] 。	0x3FF
31:30	-	保留。 读取值未定义，只写入 0。	

- [1] 当 ACMODE=1 时（选择 AC 模式），该域控制所有 3 个通道的死区时间。
- [2] 当 ACMODE=0 的情况下执行。

27.8.9 MCPWM 通信格式寄存器（MCCP—0x400B 8040）

该寄存器仅在直流模式下使用。在该寄存器中各位的控制下，内部 MC_0A 信号将与 6 个输出管脚中的任意一个或全部管脚连接。与匹配寄存器和界限寄存器一样，该寄存器具有“写”和“操作”版本。其他细节详见 27.8.6 和 27.9.2。

表584. MCPWM 通信格式寄存器位描述（MCCP—地址 0x400B 8040）

位	符号	描述	复位值
0	CCPA0	0=MC_0A 无效。1=内部 MC_0A。	0
1	CCPB0	0 = MC_0B 无效。1=MC_0B 跟踪内部 MC_0A。	0
2	CCPA1	0 = MC_1A 无效。1 = MC_1A 跟踪内部 MC_0A。	0
3	CCPB1	0 = MC_1B 无效。1 = MC_1B 跟踪内部 MC_0A。	0
4	CCPA2	0 = MC_2A 无效。1 = MC_2A 跟踪内部 MC_0A。	0
5	CCPB2	0 = MC_2B 无效。1=MC_2B 跟踪内部 MC_0A。	0
31:6	-	保留。 读取值未定义，只写入 0。	

27.8.10MCPWM 捕获寄存器

27.8.10.1 MCPWM 捕获寄存器读地址(MCCAP0~2—0x400B 8044, 0x400B 8048, 0x400B 804C)

MCCAPCON 寄存器（表 566）允许软件选择 MC_FB0-2 输入上的任意沿作为每个通道的捕获事件。当通道上发生捕获事件时，该通道的当前 TC 值就保存到它的只读捕获寄存器中。这些地址是只读的，但可以通过写 CAP_CLR 地址来清零下面的寄存器。

表585. MCPWM 捕获寄存器读地址位描述 (MCCAP0/1/2—0x400B 8044, 0x400B 8048, 0x400B 804C)

位	符号	描述	复位值
31:0	CAP0/1/2	出现捕获事件时通道 0、1、2 上的 TC 值。	0x0000 0000

27.8.10.2 MCPWM 捕获寄存器清零地址（MCCAP_CLR—0x400B 8074）

向该只写地址写入 1 会清零所选的 CAP 寄存器。

表586. MCPWM 捕获寄存器清零地址位描述（CAP_CLR—0x400B 8074）

位	符号	描述
0	CAP_CLR0	向该位写入 1 会清零 MCCAP0 寄存器。
1	CAP_CLR1	向该位写入 1 会清零 MCCAP1 寄存器。
2	CAP_CLR2	向该位写入 1 会清零 MCCAP2 寄存器。
31:3	-	保留。读取值未定义，只写入 0。

27.9 PWM 操作

27.9.1 脉宽调制

MCPWM 的每个通道都有两个输出 A 和 B，它们可以驱动一对晶体管来切换两个电源导轨之间的一个受控点。大多数情况下两个输出极性相反，但可以使能死区时间特性（以每个通道为基础）来延迟信号从“无效”到“有效”状态的跳变，这样晶体管永远都不会同时导通。更普遍的说法是，每个输出对的状态可认为是“high”、“low”、“floating”或“up”、“down”和“center-off”。

每个通道的映射从“有效”和“无效”到“高电平”和“低电平”的过程是可编程的。复位后，三个 A 输出都为无效状态或为低电平，而 B 输出都是有效状态或为高电平。

MCPWM 可执行沿对齐和中心对齐的脉宽调制。

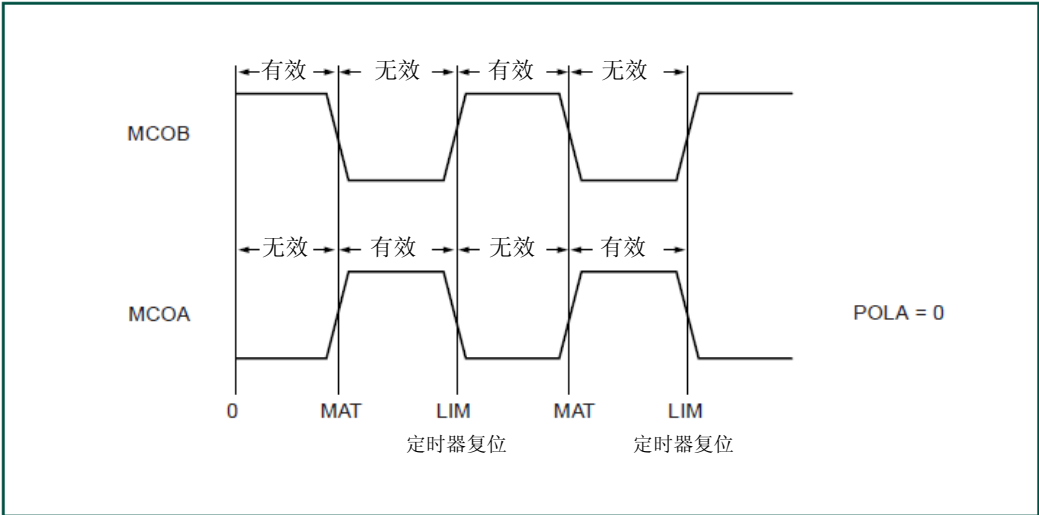
*注：*在定时器模式下，界限寄存器决定通道的已调制 MC_A 和 MC_B 输出的周期，匹配寄存器决定周期开始处的脉宽。您可以将界限寄存器和匹配寄存器分别看作是“周期寄存器”和“脉宽寄存器”。

不带死区时间的沿对齐 PWM

在该模式下，定时器 TC 从 0 开始递增计数到 LIM 寄存器中的值。如图 144 所示，在 TC 与匹配寄存器值相匹配之前，输出管脚一直保持“A 无效”状态，匹配时，它的状态变为“A 有

效”。当 TC 与界限寄存器的值匹配时，输出管脚状态变回“A 无效”，TC 复位且再次开始递增计数。

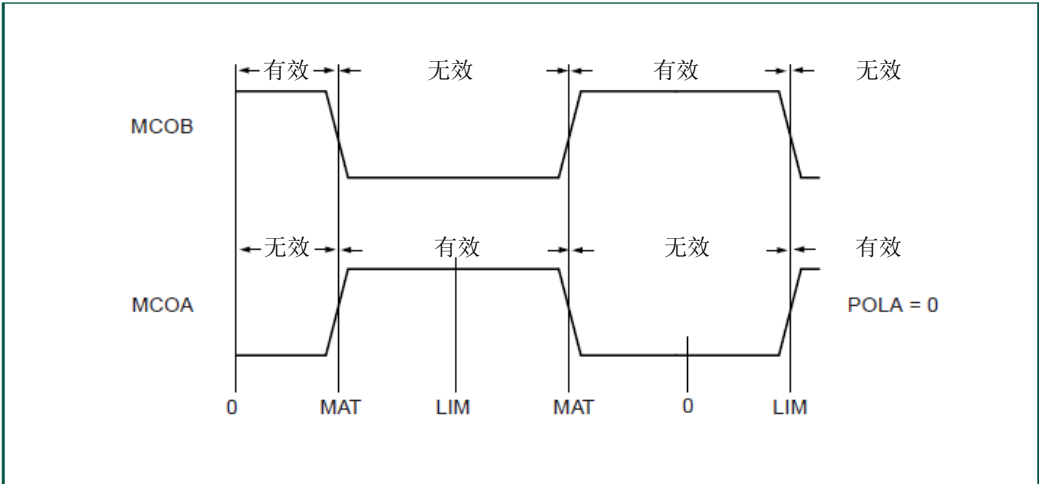
图144. 不带死区时间的沿对齐 PWM 的波形（POLA=0）



不带死区时间的中心对齐 PWM

在该模式下，定时器 TC 从 0 开始递增计数直到 LIM 寄存器中的值，然后递减计数到 0 并重复操作。如图 145 所示，当定时器递增计数时，输出管脚状态为“A 无效”直至 TC 与匹配寄存器的值匹配，此时状态变为“A 有效”。当 TC 与界限寄存器的值匹配时，它开始递减计数。当 TC 在递减计数过程中与匹配寄存器的值匹配时，输出管脚状态变回“A 无效”。

图145. 不带死区时间的中心对齐 PWM 的波形（POLA=0）



死区时间计数器

当通道 DTE 位已在 MCCON 中置位时，死区时间计数器会延迟两个输出管脚的“无效至有效状态”的跳变。只要通道的 A 输出或 B 输出从有效状态变为无效状态，死区时间计数器就开始递减计数，从通道的 DT 值（在 MCDT 寄存器）到 0。其他输出从无效状态到有效状态的跳变会被延迟，直至死区时间计数器到达 0。在死区内，MC_A 和 MC_B 输出电平都无效。图 146 所示为带死区时间的沿对齐模式，图 147 所示为带死区时间的中心对齐模式的操作。

图146. 带死区时间的沿对齐 PWM 的波形（POLA=0）

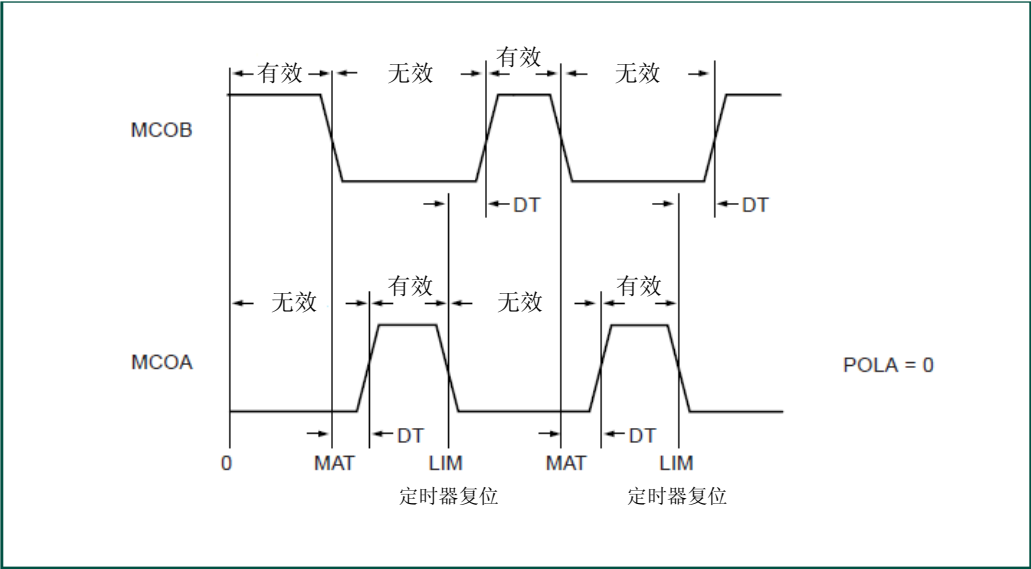
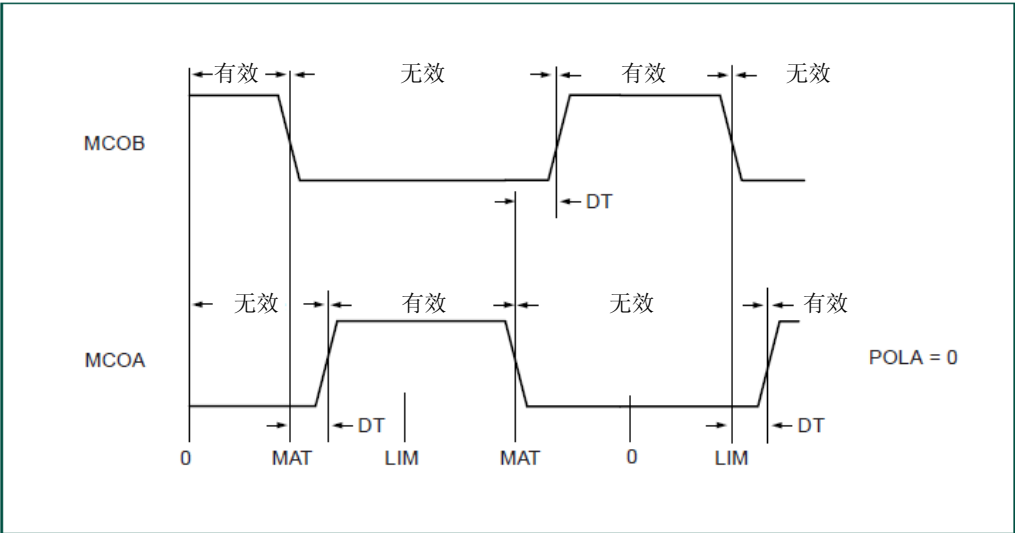


图147. 带死区时间的中心对齐的波形（POLA=0）



27.9.2 映射寄存器和同时更新

界限、匹配和通信格式寄存器（MCLIM、MCMAT 和 MCCP）作为寄存器对来执行，每对中包含一个写寄存器和一个操作寄存器。软件可写入写寄存器。操作寄存器控制每个通道的实际操作，并且在 TC 从 0 开始递增计数时将写寄存器的当前值载入到操作寄存器中。

置位 MCCON 寄存器中通道的 DISUP 位可禁止功能寄存器的更新。如果 DISUP 位置位，那么功能寄存器不更新，直至软件停止通道。

如果通道在软件写入其 LIM 或 MAT 寄存器时不运行，那么功能寄存器立即更新。

只有在通道已停止时，软件才能写 TC 寄存器。

27.9.3 快速中止（ABORT）

MCPWM 有一个外部输入管脚 $\overline{\text{MC_ABORT}}$ 。当该输入变为低电平时，全部六个输出管脚都处于“A 无效”状态，且在中断使能的情况下产生 Abort 中断。输出在“A 无效”状态下保持锁定直至 ABORT 中断标志被清除或 Abort 中断被禁用。ABORT 标志在 $\overline{\text{MC_ABORT}}$ 输入变为高电平以前可能不会被清除。

要清除 ABORT 标志，必须将 1 写入 MCINTF_CLR 寄存器的位 15。由此可删除中断请求。同样，将 1 写入 MCINTEN_CLR 寄存器的位 15 也会禁用中断。

27.9.4 捕获事件

当输入信号发生跳变时，每个 PWM 通道可捕获 TC 的瞬间值。在 MCCAPCON 寄存器的控制下，任意通道都可把任意或所有 MC_FB0-2 输入的上升沿和/或下降沿的任意组合作为捕获事件。输入上升沿或下降沿的检测与 PCLK 同步。

如果通道 MCCAPCON 寄存器的 HNF 位置位以使能“噪声滤波”，那么 MC_FB 管脚上选择的沿将启动该通道的死区时间计数器，而下文所述的捕获事件操作会被延迟，直至死区时间计数器到达 0。该功能特别适用于执行具有霍尔（Hall）传感器的三相无刷直流电机控制。

通道上的捕获事件（可能通过 HNF 延迟）会触发下列操作：

- TC 的当前值保存在捕获寄存器（CAP）中。
- 如果通道的捕获事件中断被使能（参见表 571），那么捕获事件中断标志置位。
- 如果通道的 RT 位在 MCCAPCON 寄存器中被置位，使能捕获事件上的复位，那么输入事件等效于通道的 TC 与其 LIM 寄存器相匹配。这包括复位 TC 以及在沿对齐模式下切换输出管脚，如 27.8.6 节和 27.9.1 节所述。

27.9.5 外部事件计数（计数器模式）

如通道 MODE 位在 MCCNTCON 中置位为 1，那么通道的 TC 将在 MC_FB0-2 输入的上升沿和/或下降沿（同时被检测）上递增，而不需要 PCLK。PWM 的功能和捕获功能不受影响。

27.9.6 三相直流模式

三相直流模式通过置位 **MCCON** 寄存器中的 **DCMODE** 位来选择。

在该模式下，内部 **MC_0A** 信号可以被连接到任意或全部的输出管脚。每个输出管脚可使用当前通信格式寄存器 **MCCP** 中的一个位来屏蔽。如果 **MCCP** 寄存器中的一个位为 0，则对应的输出管脚具有输出 **MC_0A** 的无效状态的逻辑电平。断开状态的极性由 **POLA0** 位来决定。

在 **MCCP** 寄存器中所有含有 1 的输出管脚由内部 **MC_0A** 信号来控制。

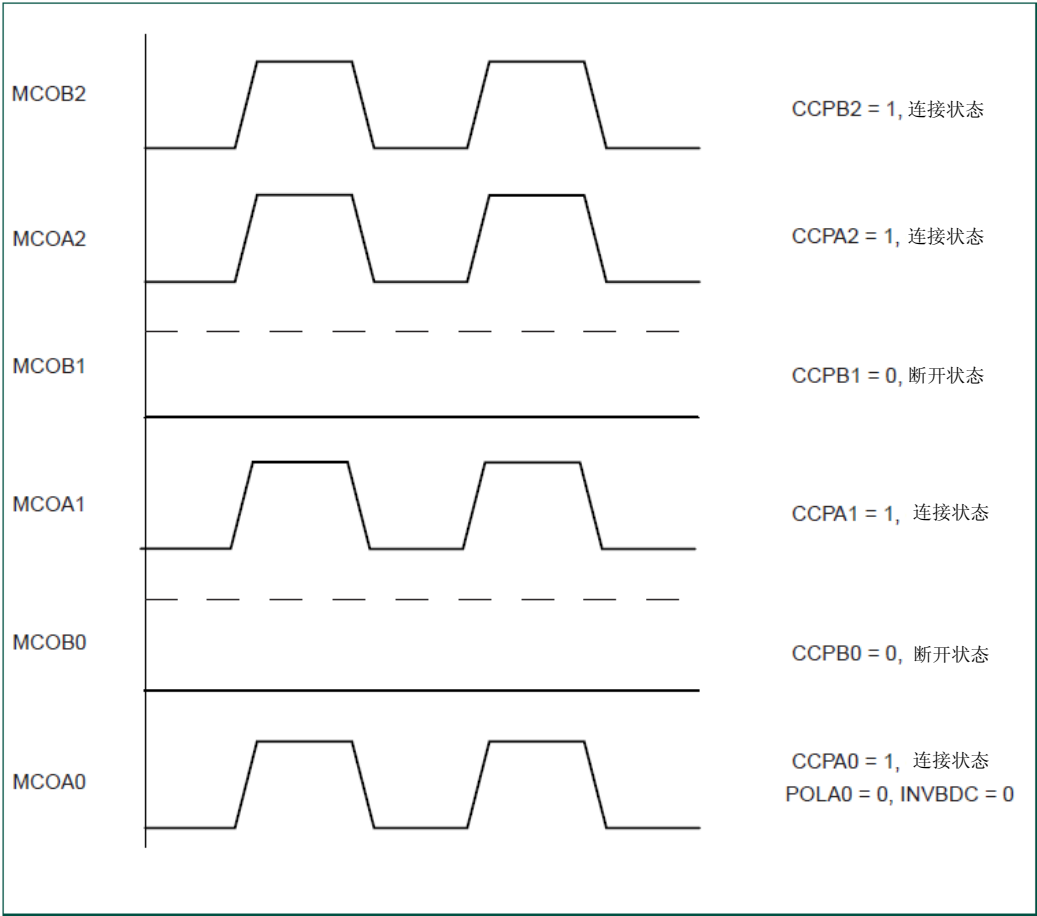
当 **MCCON** 寄存器中的位 **INVBDC** 为 1 时，三个 **MC_B** 输出管脚会被反相。该特性可用来调节桥驱动器（桥驱动器的低端开关为低电平有效输入）。

MCCP 寄存器作为一对映像寄存器来操作，因此有效通信格式的变化在新 PWM 周期的起始处出现。有关写和读这类寄存器的内容请参见 [27.8.6](#) 节和 [27.9.2](#) 节。

[图 148](#) 所示为三相直流模式下输出管脚的示例波形。**MCCP** 寄存器中的位 1 和位 3（对应于输出 **MC_1B** 和 **MC_0B**）为 0，所以这些输出会被屏蔽并处于断开状态。它们的逻辑电平由 **POLA0** 位来决定（此处，**POLA0=0** 使得无效状态为逻辑低电平）。**INVBDC** 位被设为 0（逻辑电平未反相），所以 B 输出与 A 输出的极性相同。请注意，该模式与其他模式不同，因为其他模式的 **MC_B** 输出不是 **MC_A** 输出的反相。

如[图 148](#)所示，**MCCP** 寄存器中的位 0、2、4 和 5 被设为 1，表示 **MC_1A** 以及通道 2 的两个输出管脚跟随 **MC_0A** 信号。

图148. 三相 DC 模式的示例波形



27.9.7 三相交流模式

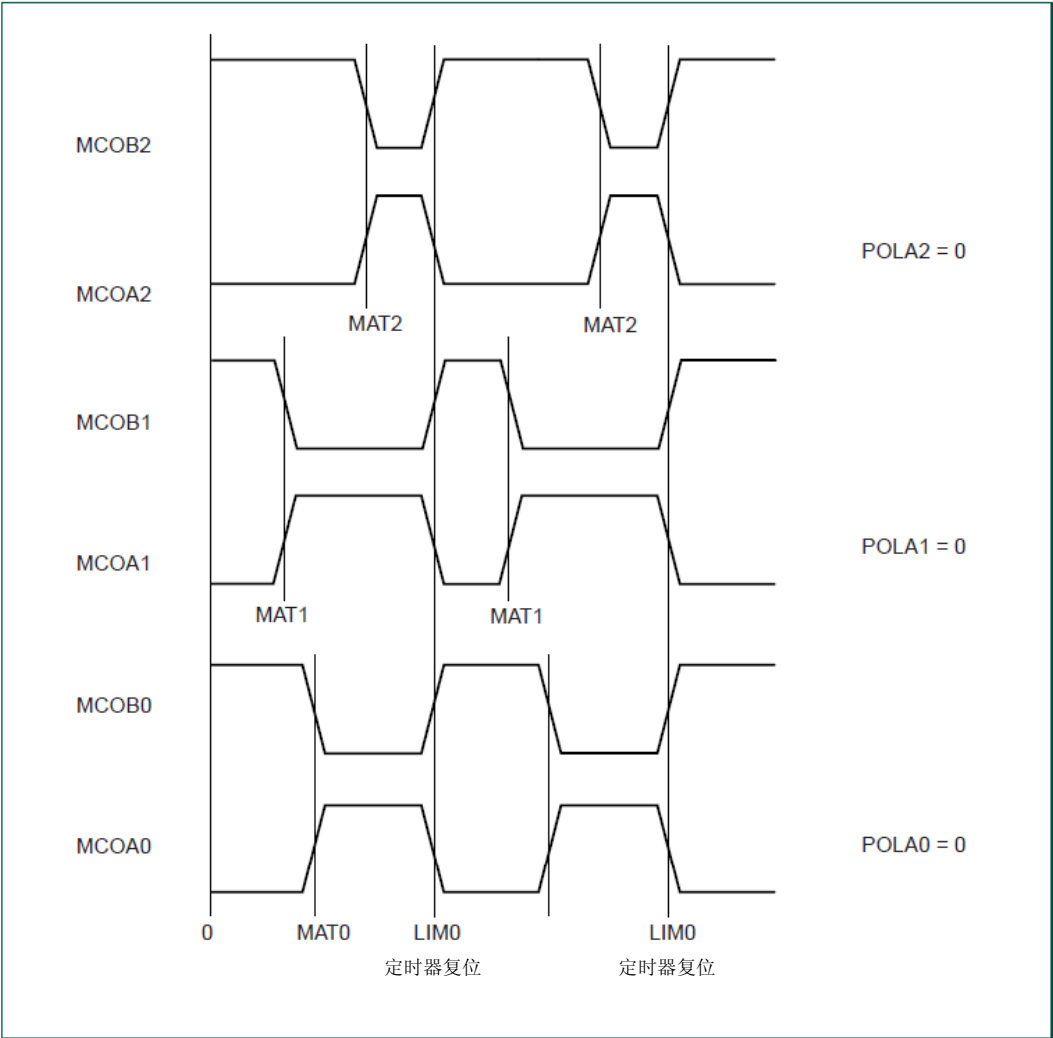
三相交流模式通过置位 **MCCON** 寄存器中的 **ACMODE** 位来选择。

在该模式下，通道 0 的 **TC** 值可用于与所有通道的 **MAT** 寄存器进行比较（不使用 **LIM1-2** 寄存器）。

每个通道通过比较其 **MAT** 值与 **TC0** 来控制其输出管脚。

图 149 所示为三相交流模式下六个输出管脚的示例波形。 **POLA** 位设为 0 用于所有的 3 个通道，因此对于所有的输出管脚来说有效状态下的电平为高电平，无效状态下的电平为低电平。每个通道具有不同的 **MAT** 值，可以与 **MCTC0** 值进行比较。在这种模式下，全部三个通道的周期值都相同且由 **MCLIM0** 来决定。死区时间模式被禁能。

图149. 三相 AC 模式的示例波形（沿对齐 PWM 模式）



27.9.8 中断

MCPWM 有 10 个可能的中断源：

- 当任意通道的 TC 与其匹配寄存器匹配时。
- 当任意通道的 TC 与其界限寄存器匹配时。
- 当任意通道捕获其 TC 值并保存到捕获寄存器时（因为选定沿在任意 MC_FB0-2 上出现）。
- 当全部三个通道的输出被强制为“A 无效”状态时（因为 $\overline{\text{MC_ABORT}}$ 管脚变为低电平）。

[27.8.3](#) 节解释了怎样使能这些中断，[27.8.2](#) 介绍了怎样映射 MC_FB0-2 输入的沿到三个通道的“捕获事件”。

28.1 基本配置

使用以下寄存器来配置 QEI：

1. 功率：在功率寄存器 PCONP（参见[表 37](#)）中置位 PCQEI。
注：复位后，QEI 被禁止（PCQEI=0）。
2. 外设时钟：使用公共 PCLK，它既用于总线接口，也用于大多数 APB 外设的功能部分。
参见 [4.6.4](#) 节。
3. 管脚：通过相关的 IOCON 寄存器（参见 [8.4.1](#) 节）来选择 QEI 管脚并选择有 QEI 功能的管脚的模式。
4. 中断：参见 [28.6.4](#) 节。利用相应的中断置位使能寄存器来使能 NVIC 中的 QEI 中断。

28.2 特性

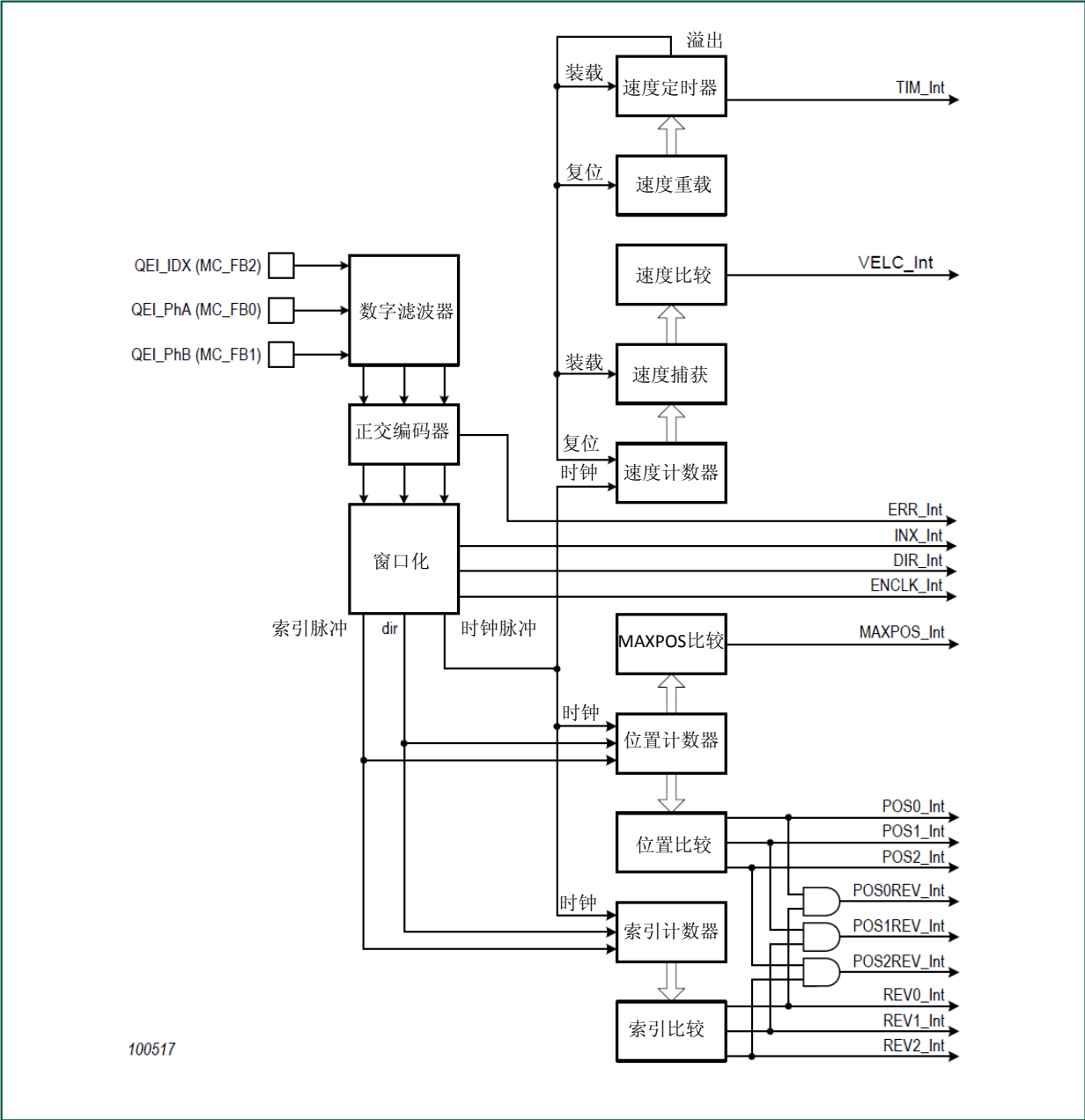
正交编码接口（QEI）具有以下特性：

- 跟踪编码器的位置。
- 根据方向进行递增/递减计数。
- 可编程,可选择 2X 模式或 4X 模式位置计数。
- 使用内置定时器来捕获速度。
- 速度比较功能，当捕获速度小于比较速度时产生中断。
- 使用 32 位寄存器来保存位置和速度。
- 3 个位置比较寄存器，可产生中断。
- 用于旋转计数的索引计数器。
- 索引比较寄存器，可产生中断。
- 可结合索引和位置中断来产生整个位移或局部旋转位移的中断。
- 带可编程编码器输入信号延迟的数字滤波器。
- 可接收已解码的输入信号（时钟和方向）。

28.3 简介

正交编码器（又名双通道增量式编码器），用于将线性位移转换成 2 个脉冲信号。通过监控脉冲的数目和这 2 个脉冲信号的相对相位，用户可以跟踪旋转的位置、方向和速度。此外，还有第三个通道，即索引信号，可用于对位置计数器进行复位。正交编码器接口模块对正交编码器轮产生的代码进行解码，将它们解释成位置对时间的积分，并确定旋转的方向。另外，它还能够捕获编码器轮运转时的速度。

图150. 编码器接口方框图



28.4 功能描述

QEI 模块对正交编码器轮产生的 2 位格雷码进行解码，将它们解释成位置对时间的积分，并确定旋转的方向。此外，它还可以捕获编码器轮运转时的速度。

28.4.1 输入信号

QEI 模块支持 2 种信号操作模式：正交相位模式和时钟/方向模式。在正交相位模式中，编码器产生 2 个相位差为 90°的时钟信号；它们的边沿关系被用来确定旋转方向。在时钟/方向模式中，编码器产生一个时钟信号和一个方向信号，分别表示步长和旋转方向。

这两种模式的选择由 QEI 控制寄存器（QEICON）中的 SigMode 位确定（见表 593）。当 SigMode=1 时，正交编码器被旁路，相 A 管脚为方向信号，相 B 管脚为计数器的时钟信号。当 SigMode=0 时，正交编码器对相 A 和相 B 进行解码。在此模式中，正交编码器会产生旋转方向和计数器的时钟信号。两种模式中的方向信号都受方向反转位（DIRINV）的影响。

28.4.1.1 正交输入信号

当相 A 的边沿超前于相 B 的边沿时，位置计数器递增。当相 B 的边沿超前于相 A 的边沿时，位置计数器递减。当一对上升沿和下降沿出现在其中一个相位上，而在另一个相位上没有任何边沿时，这表示旋转方向已经发生了改变。

表587. 编码器状态

相 A	相 B	状态
1	0	1
1	1	2
0	1	3
0	0	4

表588. 编码器状态的转换^[1]

从状态	到状态	方向
1	2	正向
2	3	
3	4	
4	1	
4	3	反向
3	2	
2	1	
1	4	

[1] 除此表外其他状态转换都是非法的，应置位 ERR。

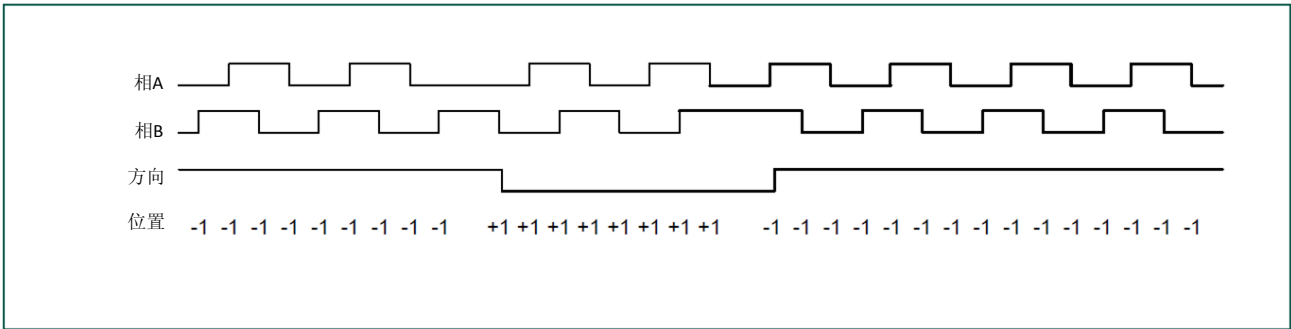
相 A 和相 B 输入信号的互换是通过求位 DIR 的补码来实现的。当 DIR=1 时，方向翻转位（DIRINV）与位 DIR 互补。

表589. 编码器方向

位 DIR	位 DIRINV	方向
0	0	向前
1	0	向后
0	1	向后
1	1	向前

图 151 中给出了正交编码信号与方向和计数的对等关系。

图151. 正交编码器基本操作



28.4.1.2 数字输入滤波

3 路编码器输入信号（相 A、相 B 和索引脉冲）都需要进行数字滤波。用户可编辑 1~4，294，967，295（0xFFFF FFFF）个采样时钟。为了能接受信号转变，输入信号必须在设定的采样时钟数期间保持新的状态。

28.4.2 位置捕获

位置积分器的捕获模式可设成在相 A 信号的上升沿和下降沿或是在相 A 和相 B 的上升沿和下降沿对位置计数器进行更新。在相 A 和相 B 的上升和下降沿上更新位置计数器可提供更高精度的数据（更多位置计数），但位置计数器的计数范围却相对变少了。

可以单独使能位置积分器和速度捕获。另外，相位信号也可以解释为时钟信号和方向信号，将它们作为某些编码器的输出。

位置计数器遇到下列其中一种情况时将自动复位：**1.**计数值超过最大位置值(QEIMAXPOS)时，位置计数器复位为 0；**2.**如果索引位（RESPI）被设置成复位，在下一次位置分辨率提高（校准）后第一次检测到索引脉冲时就会将位置计数器复位为 0；**3.**若是索引位被设置成连续复位（CRESPI），在下一次位置分辨率提高（再校准）后，一旦索引脉冲出现就会连续复位位置计数器为 0。

28.4.3 速度捕获

速度捕获包含一个可编程定时器和一个捕获寄存器。定时器在给定的时间周期内对相位边沿进行计数（使用与位置积分器相同的配置）。当速度定时器（QEITIME）溢出时，速度计数器（QEIVEL）的值将被存入捕获寄存器中（QEICAP），这时速度计数器清零。速度定时器会加载速度重载寄存器（QEILOAD）中的值。最后产生速度中断（TIM_Int）。在给定的时间内所计得的边沿数目与编码器的速度直接成正比。请注意，不管旋转方向如何，也不管旋转方向是否发生变化，速度计数器一样累加计数。

置位复位速度位（RESV）将会清零速度计数器，将速度捕获寄存器复位成 0xFFFF FFFF，速度定时器会加载速度重载寄存器（QEILOAD）中的值。

可以使用下面的等式将速度计数值转换为 RPM 值：

$$\text{RPM} = (\text{PCLK} \times \text{QEICAP} \times 60) \div (\text{QEILOAD} \times \text{PPR} \times \text{Edges})$$

其中：

- **PCLK** 表示 QEI 模块的外设时钟速率。有关 PCLK 的更多详情，参见 4.6.4 节。
- **QEICAP** 表示最后速度定时器期间所捕获到的速度计数器值。
- **QEILOAD** 表示速度定时器重装值。
- **PPR** 表示实际用到的物理编码器旋转一圈的脉冲数。
- **Edges** 为 2 或 4，这是根据 QEICON 寄存器中设置的捕获模式来决定（当 CapMode 被设置为 0 时，Edges 取 2；当 CapMode 被设置为 1 时，Edges 取 4）。

例如：有一个运行速率为 600rpm 的电机。在电机上连接一个每旋转 1 次可产生 2048 个脉冲的正交编码器，这样每转可获得 8192 个相位沿（PPR x Edges）。这样就是每秒可获得 81,920 个脉冲（在速率为 600rpm 的情况下，每旋转一次可获得 8192 个相位沿，而电机每秒转动 10 次）。如果定时器的时钟频率为 10,000Hz，装载值（QEILOAD）为 2500（相当于 1/4 秒），则每次更新定时器时，可计得 20,480 个脉冲。使用上述等式：

$$\text{RPM} = (10000 \times 1 \times 20480 \times 60) \div (2500 \times 2048 \times 4) = 600 \text{ RPM}$$

现在，假设电机的转速要达到 3000rpm。这时正交编码器每秒必须产生 409,600 个脉冲，即 1/4 秒可产生 102,400 个脉冲，再次使用上述等式：

$$\text{RPM} = (10000 \times 1 \times 102400 \times 60) \div (2500 \times 2048 \times 4) = 3000 \text{ RPM}$$

这些都是简单的例子，在实际使用时，PCLK 的速度要更高，并且 QEILOAD 的值也会更大。

28.4.4 速度比较

除了速度捕获之外，速度测量系统还有一个可编程速度比较寄存器。每出现一次速度捕获事件后，速度捕获寄存器（QEICAP）的值就会和速度比较寄存器（VELCOMP）的值做比较。如果捕获到的速度小于比较值，则会产生一个中断，速度比较中断使能位置位。这可以用来检测电动机的转动轴是否停止或转速太慢。

28.5 管脚描述

表590. QEI 的管脚描述

管脚名称	I/O	描述
MC_FB0 ^[1]	I	作为正交编码器接口的相 A 输入（PHA）
MC_FB1 ^[1]	I	作为正交编码器接口的相 B 输入（PHB）
MC_FB2 ^[1]	I	作为正交编码器接口的索引脉冲输入（IDX）

[1] 正交解码器接口利用相同的管脚作为机电控制 PWM 的反馈输入管脚类似使用。
如果作为机电控制器的部分使用，正交解码器接口就是可直接反馈给 MCPWM 的备用接口。

28.6 寄存器描述

28.6.1 寄存器汇总

表591. 寄存器汇总

符号	描述	R/W	复位值	地址	表
控制寄存器					
QEICON	控制寄存器	WO	无	0x400B C000	表 592
QEICONF	配置寄存器	R/W	0	0x400B C008	表 593
QEISTAT	状态寄存器	RO	0	0x400B C004	表 594
位置、索引和定时器寄存器					
QEIP0S	位置寄存器	RO	0	0x400B C00C	表 595
QEIMAXPOS	最大位置值寄存器	R/W	0	0x400B C010	表 596
CMPOS0	位置比较寄存器 0	R/W	0xFFFF FFFF	0x400B C014	表 597
CMPOS1	位置比较寄存器 1	R/W	0xFFFF FFFF	0x400B C018	表 598
CMPOS2	位置比较寄存器 2	R/W	0xFFFF FFFF	0x400B C01C	表 599
INXCNT	索引计数寄存器 0	RO	0	0x400B C020	表 600
INXCMP0	索引比较寄存器 0	R/W	0xFFFF FFFF	0x400B C024	表 601
INXCMP1	索引比较寄存器 1	R/W	0xFFFF FFFF	0x400B C04C	表 602
INXCMP2	索引比较寄存器 2	R/W	0xFFFF FFFF	0x400B C050	表 603
QEILOAD	速度定时器重载寄存器	R/W	0	0x400B C028	表 604
QEITIME	速度定时器寄存器	RO	0	0x400B C02C	表 605
QEIVEL	速度计数器寄存器	RO	0	0x400B C030	表 606
QEICAP	速度捕获寄存器	RO	0xFFFF FFFF	0x400B C034	表 607
VELCOMP	速度比较寄存器	R/W	0	0x400B C038	表 608
FILTERPHA	用于相 A 的数字滤波器寄存器	R/W	0	0x400B C03C	表 609
FILTERPHB	用于相 B 的数字滤波器寄存器	R/W	0	0x400B C040	表 610
FILTERINX	用于 INX 的数字滤波器寄存器	R/W	0	0x400B C044	表 611
WINDOW	索引验收窗口寄存器	R/W	0xF	0x400B C048	表 612
中断寄存器					
QEINTSTAT	中断状态寄存器	RO	0	0x400B CFE0	表 613
QEISET	中断状态设置寄存器	WO	无	0x400B CFEC	表 614
QEICLR	中断状态清除寄存器	WO	无	0x400B CFE8	表 615
QEIE	中断使能寄存器	RO	0	0x400B CFE4	表 616
QEIES	中断使能置位寄存器	WO	无	0x400B CFDC	表 617
QEIEC	中断使能清除寄存器	WO	无	0x400B CFD8	表 618

28.6.2 控制寄存器

28.6.2.1 QEI 控制寄存器（QEICON—0x400B C000）

该寄存器中包含的位，可控制 QEI 模块中的位置计数器和速度计数器的操作。

表592. QEI 控制寄存器位描述（QEICON—地址 0x400B C000）

位	符号	描述	复位值
0	RESP	复位位置计数器。当为 1 时，位置计数器被复位为 0。RESP 在位置计数器被清零时自动清零。	0
1	RESPI	索引脉冲出现时复位位置计时器。当为 1 时，位置计数器在出现索引脉冲时被复位为 0。RESPI 在位置计数器被清零时自动清零。	0
2	RESV	复位速度。当为 1 时，速度计数器被复位为 0，速度定时器被重载，速度比较寄存器被预设。RESV 在速度计数器被清零时自动清零。	0
3	RESI	复位索引计数器。当为 1 时，索引计数器被复位为 0。RESI 在索引计数器被清零时自动清零。	0
31:4	-	保留。读取值未定义，只写入 0。	无

28.6.2.2 QEI 配置寄存器（QEICONF—0x400B C008）

该寄存器包含了 QEI 模块的配置信息。

表593. QEI 配置寄存器位描述（QEICONF—地址 0x400B C008）

位	符号	描述	复位值
0	DIRINV	方向反相。为 1 时，求位 DIR 位补码。	0
1	SIGMODE	信号模式。为 0 时，相 A 和相 B 作为作为解码器的正交相位信号。为 1 时，相 A 为方向信号，相 B 为时钟信号。	
2	CAPMODE	捕获模式。为 0 时，只对相 A 边沿进行计数（2X）。为 1 时，同时对相 A 和相 B 边沿进行计数（4X），这样位置分辨率增加但计数范围会减少。	
3	INVINX	索引脉冲反相。为 1 时，将索引脉冲输入的侦听反相。	0
4	CRESPI	索引脉冲出现时连续复位位置计数器。为 1 时，在下一个位置分辨率提高（再校准）后，一旦索引脉冲出现，位置计数器就会被复位为 0。	
15:5	-	保留。读取值未定义，只写入 0。	无
19:16	INXGATE	索引脉冲选通配置： 当 INXGATE[16]=1 且 PHA=1 和 PHB=0 时，允许索引脉冲通过，否则阻止索引脉冲。 当 INXGATE[17]=1 且 PHA=1 和 PHB=1 时，允许索引脉冲通过，否则阻止索引脉冲。 当 INXGATE[18]=1 且 PHA=0 和 PHB=1 时，允许索引脉冲通过，否则阻止索引脉冲。 当 INXGATE[19]=1 且 PHA=0 和 PHB=0 时，允许索引脉冲通过，否则阻止索引脉冲。	0xF
31:20	-	保留。读取值未定义，只写入 0。	无

28.6.2.3 QEI 状态寄存器（QEISTAT—0x400B C004）

该寄存器提供了编码器接口的状态。

表594. QEI 状态寄存器位描述（QEISTAT—地址 0x400B C004）

位	符号	描述	复位值
0	DIR	方向位。该位与位 DIRINV 一起来表示旋转方向是正向还是反向。 详见表 589。	
31:1	-	保留。读取值未定义，只写入 0。	无

28.6.3 位置、索引和定时器寄存器

28.6.3.1 QEI 位置寄存器（QEIPOS—0x400B C00C）

该寄存器包含了编码器位置的当前值。编码器会根据旋转的方向进行递增或递减计数。

表595. QEI 位置寄存器位描述（QEIPOS—地址 0x400B C00C）

位	符号	描述	复位值
31:0	-	当前的位置值。	0

28.6.3.2 QEI 最大位置值寄存器（QEIMAXPOS—0x400B C010）

该寄存器包含编码器位置的最大值。当正向旋转时，如果位置寄存器的值大于该寄存器的值，那么位置寄存器复位为 0。当反向旋转时，如果执行减 1 操作的位置寄存器，其值已达到 0，则位置寄存器复位为该寄存器中的值。

表596. QEI 最大位置值寄存器位描述（QEIMAXPOS—地址 0x400B C010）

位	符号	描述	复位值
31:0	-	当前最大位置值。	0

28.6.3.3 QEI P 位置比较寄存器 0（CMPOS0—0x400B C014）

该寄存器含有一个位置比较值。该值会与位置寄存器中的当前值做比较。若比较值等于位置寄存器的当前值时，都会产生中断。（设置这个）比较值时必须要考虑一点：开始位置值为 0。

表597. QEI 位置比较寄存器 0 位描述（CMPOS0—地址 0x400B C014）

位	符号	描述	复位值
31:0	-	位置比较值 0。	0

28.6.3.4 QEI 位置比较寄存器 1（CMPOS1—0x400B C018）

该寄存器含有一个位置比较值。该值会与位置寄存器中的当前值做比较。若比较值等于位置寄存器的当前值时，都会产生中断。（设置这个）比较值时必须要考虑一点：开始位置为 0。

表598. QEI 位置比较寄存器 1 位描述（CMPOS1—地址 0x400B C018）

位	符号	描述	复位值
31:0	-	位置比较值 1。	0

28.6.3.5 QEI 位置比较寄存器 2（CMPOS1—0x400B C018）

该寄存器包含有一个位置比较值。将该值与位置寄存器中的当前值进行比较。若比较值等于位置寄存器的当前值时，都会产生中断。（设置这个）比较值时必须要考虑一点：开始位置值为 0。

表599. QEI 位置比较寄存器 2 位描述（CMPOS2 地址 0x400B C01C）

位	符号	描述	复位值
31:0	-	位置比较值 2。	0

28.6.3.6 QEI 索引计数寄存器（INXCNT—0x400B C020）

该寄存器含有索引计数器的当前值。在进行索引计数时,寄存器会更新其内容。当位置计数器的值超出 MAXPOS 的值时，编码器会根据旋转的方向进行递增计数，当位置计数器的值低于 0 时，编码器会根据旋转的方向进行递减计数。当检测到索引脉冲而进行（再）校准时，内部出现强制上溢出或下溢出。

表600. QEI 索引计数寄存器位描述（INXCNT—地址 0x400B C020）

位	符号	描述	复位值
31:0	-	当前的索引计数器值。	0

28.6.3.7 QEI 索引比较寄存器 0（INXCMP0—0x400B C024）

该寄存器含有一个索引比较值。该值会与索引计数寄存器的当前值做比较。当比较值小于、等于或大于索引计数寄存器的当前值时，都会产生中断。

表601. QEI 索引比较寄存器 0 位描述（INXCMP0—地址 0x400B C024）

位	符号	描述	复位值
31:0	-	索引比较值 0。	0

28.6.3.8 QEI 索引比较寄存器 1（INXCMP1—0x400B C04C）

该寄存器含有一个索引比较值。该值会与索引计数寄存器的当前值做比较。当比较值小于、等于或大于索引计数寄存器的当前值时，都会产生中断。

表602. QEI 索引比较寄存器 1 位描述（INXCMP1—地址 0x400B C04C）

位	符号	描述	复位值
31:0	-	索引比较值 1。	0

28.6.3.9 QEI 索引比较寄存器 2（INXCMP2—0x400B C050）

该寄存器含有一个索引比较值。该值会与索引计数寄存器的当前值做比较。当比较值小于、等于或大于索引计数寄存器的当前值时，都会产生中断。

表603. QEI 索引比较寄存器 2 位描述（INXCMP2—地址 0x400B C050）

位	符号	描述	复位值
31:0	-	索引比较值 2。	0

28.6.3.10 QEI 速度定时器重载寄存器（QEILOAD—0x400B C028）

该寄存器包含速度定时器的重载值。当定时器（QEITIME）溢出或位 RESV 被置位时，该值就装入定时器（QEITIME）。

表604. QEI 速度定时器重载寄存器位描述（QEILOAD—地址 0x400B C028）

位	符号	描述	复位值
31:0	-	当前速度定时器的装载值。	0

28.6.3.11 QEI 速度定时器寄存器（QEITIME—0x400B C02C）

该寄存器包含速度定时器的当前值。当速度定时器（QEITIME）溢出时，速度计数器（QEIVEL）中的值就存入到速度捕获寄存器（QEICAP）。然后该速度计数器被复位为 0，定时器重新装入速度重载寄存器（QEILOAD）中存储的值，并产生速度中断（TIM_Int）。

表605. QEI 定时器寄存器位描述（QEITIME—地址 0x400B C02C）

位	符号	描述	复位值
31:0	-	当前的速度定时器值。	0

28.6.3.12 QEI 速度寄存器（QEIVEL—0x400B C030）

该寄存器包含在当前时间周期内正在计数的速度脉冲的个数。当速度定时器（QEITIME）溢出时，该寄存器的值会存入速度捕获寄存器中（QEICAP）。然后，寄存器复位为 0。该寄存器在速度复位位（RESV）发出信号的时候，也会复位。

表606. QEI 速度寄存器位描述（QEIVEL—地址 0x400B C030）

位	符号	描述	复位值
31:0	-	当前速度脉冲的计数。	0

28.6.3.13 QEI 速度捕获寄存器（QEICAP—0x400B C034）

该寄存器包含最近测得的正交编码器的速度。它与上一个速度定时器周期内计数所得的脉冲数相对应。当速度定时器溢出时，当前速度计数会锁存到该寄存器中。

表607. QEI 速度捕获寄存器位描述（QEICAP—地址 0x400B C034）

位	符号	描述	复位值
31:0	-	最后速度捕获值。	0

28.6.3.14 QEI 速度比较寄存器（VELCOMP—0x400B C038）

该寄存器含有一个速度比较值。该值会与速度捕获寄存器所捕获的速度做比较。如果捕获速度小于该比较寄存器的值，则会产生速度比较中断（VELC_Int）（如果使能）。

表608. QEI 速度比较寄存器位描述（VELCOMP—地址 0x400B C038）

位	符号	描述	复位值
31:0	-	速度脉冲比较计数。	0

28.6.3.15 用于相 A 的 QEI 数字滤波器（FILTERPHA—0x400B C03C）

该寄存器包含数字滤波器的采样个数。如果采样个数为 0，滤波器被旁路。

表609. 用于相 A 的 QEI 数字滤波器位描述（FILTERPHA—地址 0x400B C03C）

位	符号	描述	复位值
31:0	-	相 A 的数字滤波器采样延时。	0

28.6.3.16 用于相 B 的 QEI 数字滤波器（FILTERPHB—0x400B C040）

该寄存器包含数字滤波器的采样个数。如果采样个数为 0，滤波器被旁路。

表610. 用于相 B 的 QEI 数字滤波器位描述（FILTERPHB—地址 0x400B C040）

位	符号	描述	复位值
31:0	-	相 B 的数字滤波器采样延时。	0

28.6.3.17 用于 INX 的 QEI 数字滤波器（FILTERINX—0x400B C044）

该寄存器包含数字滤波器的采样个数。如果采样个数为 0，滤波器被旁路。

表611. 用于 INX 的数字滤波器位描述（FILTERINX—地址 0x400B C044）

位	符号	描述	复位值
31:0	-	索引脉冲的数字滤波器采样延时。	0

28.6.3.18 QEI 索引验收窗口（WINDOW—0x400B C048）

当索引脉冲和相位信号/时钟信号边沿几乎汇聚在一起的时候，该寄存器包括有索引验收窗口的宽度。若是激活的相位信号/时钟信号边沿比索引脉冲先降落在窗口范围内，则将在这个相位信号/时钟信号边沿启动（再）校准。

表612. QEI 索引验收窗口位描述（WINDOW—地址 0x400B C048）

位	符号	描述	复位值
31:0	-	索引验收窗口宽度。	0xF

28.6.4 中断寄存器

28.6.4.1 QEI 中断状态寄存器（QEINTSTAT—0x400B CFE0）

该寄存器提供了正交编码接口的状态信息和控制器对应中断源的当前设置。寄存器中的位被置位为 1,表示已出现了锁存事件；置位为 0 表示还未出现事件。

表613. QEI 中断状态寄存器位描述（QEINSTAT—地址 0x400B CFE0）

位	符号	描述	复位值
0	INX_Int	表示检测到一个索引脉冲。	0
1	TIM_Int	表示速度定时器溢出。	0
2	VELC_Int	表示捕获的速度小于比较速度。	0
3	DIR_Int	表示检测到方向改变。	0
4	ERR_Int	表示检测到编码器相位错误。	0
5	ENCLK_Int	表示检测到编码器时钟脉冲。	0
6	POS0_Int	表示位置 0 的比较值与当前位置值相等。	0
7	POS1_Int	表示位置 1 的比较值与当前位置值相等。	0
8	POS2_Int	表示位置 2 的比较值与当前位置值相等。	0
9	REV0_Int	表示索引比较 0 的值与当前索引计数值相等。	0
10	POS0REV_Int	位置 0 和旋转计数中断的组合。当 POS0_Int 和 REV0_Int 同时置位时该位置位。	0
11	POS1REV_Int	位置 1 和旋转计数中断的组合。当 POS1_Int 和 REV1_Int 同时置位时该位置位。	0
12	POS2REV_Int	位置 2 和旋转计数中断的组合。当 POS2_Int 和 REV2_Int 同时置位时该位置位。	0
13	REV1_Int	表示索引比较 1 的值与当前索引计数值相等。	0
14	REV2_Int	表示索引比较 2 的值与当前索引计数值相等。	0
15	MAXPOS_Int	表示当前位置计数在正向从 MAXPOS 值变化到 0，在反向从 0 变化到 MAXPOS 值。	0
31:16	-	保留。读取值未定义，只写入 0。	无

28.6.4.2 QEI 中断设置寄存器（QEISET—0x400B CFEC）

向该寄存器中某位写入 1 会设置 QEI 中断状态寄存器（QEISTAT）中对应的位。

表614. QEI 中断设置寄存器位描述（QEISET—地址 0x400B CFEC）

位	符号	描述
0	INX_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 INX_Int。
1	TIM_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 TIN_Int。
2	VELC_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 VELC_Int。
3	DIR_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 DIR_Int。
4	ERR_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 ERR_Int。
5	ENCLK_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 ENCLK_Int。
6	POS0_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 POS0_Int。
7	POS1_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 POS1_Int。
8	POS2_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 POS2_Int。
9	REV0_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 REV0_Int。
10	POS0REV_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 POS0REV_Int。
11	POS1REV_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 POS1REV_Int。
12	POS2REV_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 POS2REV_Int。
13	REV1_Int	向该寄存器中的位写入 1 会设置 QEIINTSTAT 中的位 REV1_Int。
14	REV2_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 REV2_Int。
15	MAXPOS_Int	向该寄存器中的某位写入 1 会设置 QEIINTSTAT 中的位 MAXPOS_Int。
31:16	-	保留。读取值未定义，只写入 0。

28.6.4.3 QEI 中断状态清除寄存器（QEICLR—0x400B CFE8）

向该寄存器中的某位写入 1 会清除 QEI 中断状态寄存器（QEISTAT）中对应的位。

表615. QEI 中断清除寄存器位描述（QEICLR—0x400B CFE8）

位	符号	描述
0	INX_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 INX_Int。
1	TIM_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 TIN_Int。
2	VELC_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 VELC_Int。
3	DIR_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 DIR_Int。
4	ERR_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 ERR_Int。
5	ENCLK_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 ENCLK_Int。
6	POS0_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 POS0_Int。
7	POS1_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 POS1_Int。
8	POS2_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 POS2_Int。
9	REV0_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 REV0_Int。
10	POS0REV_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 POS0REV_Int。
11	POS1REV_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 POS1REV_Int。
12	POS2REV_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 POS2REV_Int。
13	REV1_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 REV1_Int。
14	REV2_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 REV2_Int。
15	MAXPOS_Int	向该寄存器中的某位写入 1 会清除 QEIINTSTAT 中的位 MAXPOS_Int。
31:16	-	保留。读取值未定义，只写入 0。

28.6.4.4 QEI 中断使能寄存器（QEIE—0x400B CFE4）

该寄存器可使能中断源.将该寄存器中的某位设为 1 会使能对应的中断；设为 0 会禁止对应的中断。

表616. QEI 中断使能寄存器位描述（QEIE—地址 0x400B CFE4）

位	符号	描述	复位值
0	INX_Int	将该寄存器中的某位设为 1 会使能 INX_Int 中断。	0
1	TIM_Int	将该寄存器中的某位设为 1 会使能 TIN_Int 中断。	0
2	VELC_Int	将该寄存器中的某位设为 1 会使能 VELC_Int 中断。	0
3	DIR_Int	将该寄存器中的某位设为 1 会使能 DIR_Int 中断。	0
4	ERR_Int	将该寄存器中的某位设为 1 会使能 ERR_Int 中断。	0
5	ENCLK_Int	将该寄存器中的某位设为 1 会使能 ENCLK_Int 中断。	0
6	POS0_Int	将该寄存器中的某位设为 1 时会使能 POS0_Int 中断。	0
7	POS1_Int	将该寄存器中的某位设为 1 时会使能 POS1_Int 中断。	0
8	POS2_Int	将该寄存器中的某位设为 1 时会使能 POS2_Int 中断。	0
9	REV0_Int	将该寄存器中的某位设为 1 时会使能 REV0_Int 中断。	0
10	POS0REV_Int	将该寄存器中的某位设为 1 时会使能 POS0REV_Int 中断。	0
11	POS1REV_Int	将该寄存器中的某位设为 1 时会使能 POS1REV_Int 中断。	0
12	POS2REV_Int	将该寄存器中的某位设为 1 时会使能 POS2REV_Int 中断。	0
13	REV1_Int	将该寄存器中的某位设为 1 时会使能 REV1_Int 中断。	0
14	REV2_Int	将该寄存器中的某位设为 1 时会使能 REV2_Int 中断。	0
15	MAXPOS_Int	将该寄存器中的某位设为 1 时会使能 MAXPOS_Int 中断。	0
31:16	-	保留。读取值未定义，只写入 0。	无

28.6.4.5 QEI 中断使能置位寄存器（QEIES—0x400B CFDC）

向该寄存器中的某位写入 1 会设置 QEI 中断使能寄存器（QEIE）中对应的位。

表617. QEI 中断使能置位寄存器位描述（QEIES—地址 0x400B CFDC）

位	符号	描述
0	INX_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 INX_Int 中断。
1	TIM_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 TIN_Int 中断。
2	VELC_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 VELC_Int 中断。
3	DIR_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 DIR_Int 中断。
4	ERR_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 ERR_Int 中断。
5	ENCLK_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 ENCLK_Int 中断。
6	POS0_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 POS0_Int 中断。
7	POS1_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 POS1_Int 中断。
8	POS2_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 POS2_Int 中断。
9	REV0_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 REV0_Int 中断。
10	POS0REV_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 POS0REV_Int 中断。
11	POS1REV_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 POS1REV_Int 中断。
12	POS2REV_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 POS2REV_Int 中断。
13	REV1_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 REV1_Int 中断。
14	REV2_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 REV2_Int 中断。
15	MAXPOS_Int	向该寄存器中的某位写入 1 会使能 QEIE 寄存器中的 MAXPOS_Int 中断。
31:16	-	保留。读取值未定义，只写入 0。

28.6.4.6 QEI 中断使能清除寄存器（QEIEC—0x400B CFD8）

向该寄存器中的某位写入 1 会清除 QEI 中断使能寄存器（QEIE）中对应的位。

表618. QEI 中断使能清除寄存器位描述（QEIEC—地址 0x400B CFD8）

位	符号	描述
0	INX_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 INX_Int 中断。
1	TIM_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 TIM_Int 中断。
2	VELC_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 VELC_Int 中断。
3	DIR_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 DIR_Int 中断。
4	ERR_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 ERR_Int 中断。
5	ENCLK_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 ENCLK_Int 中断。
6	POS0_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 POS0_Int 中断。
7	POS1_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 POS1_Int 中断。
8	POS2_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 POS2_Int 中断。
9	REV0_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 REV0_Int 中断。
10	POS0REV_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 POS0REV_Int 中断。
11	POS1REV_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 POS1REV_Int 中断。
12	POS2REV_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 POS2REV_Int 中断。
13	REV1_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 REV1_Int 中断。
14	REV2_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 REV2_Int 中断。
15	MAXPOS_Int	向该寄存器中的某位写入 1 会禁止 QEIE 寄存器中的 MAXPOS_Int 中断。
31:16	-	保留。读取值未定义，只写入 0。

29.1 基本配置

使用下列寄存器来配置 RTC：

1. 功率：在寄存器 PCONP 中置位 PCRTC（见[表 37](#)），适用于 RTC 和事件记录器。
注：复位后，RTC 被使能。有关降低功耗的内容参见[29.7](#)节。
2. 时钟：RTC 将 RTC 振荡器输出的 1Hz 时钟信号作为内部时钟源。外设时钟被用于访问 RTC 寄存器。
3. 中断：有关 RTC 中断处理的详细内容请参见 [29.6.1](#) 节。利用相应的中断置位使能寄存器来使能 NVIC 中的中断。

29.2 特性

- 带有日历和时钟功能，提供秒、分、小时、日期（月）、月、年、星期和日期（年）的信息
- 超低功耗设计，可支持电池供电系统。电池供电操作所需的电流不到 1 微安。还可使用 CPU 电源（如果有的话）
- 容量为 20 字节的电池供电存储器，当 CPU 电源被移除时，RTC 可继续运行
- 专用的 32KHz 超低功耗振荡器
- 专用的电池电源管脚
- RTC 电源与芯片的其它部分分离
- 校准计数器可对时间进行校准（分辨率大于±1 秒/天）
- 周期性中断在时间寄存器任意字段的值递增时产生
- 在出现特殊的日期和时间比时产生报警中断

29.3 概述

实时时钟（RTC）是一组用于测量时间的计数器，在系统掉电时也可以继续运行。在其寄存器未接入 CPU 的情况下，RTC 所的功耗极低，尤其是在省电模式下的功耗更低。LPC178x/177x 系列中的 RTC 时钟源可以是单独的 32KHz 的振荡器，它可以产生一个 1Hz 内部时基。RTC 由自带的电源管脚（VBAT）供电，VBAT 可以与电池相连，与一个外部的 3V 电源相连或保持悬浮状态。

[图 152](#) 中所示为 RTC 电源域的总体设计。[图 153](#) 是维持 RTC 时钟信号部分的详解图。

29.4 结构

图152. RTC 域概念图

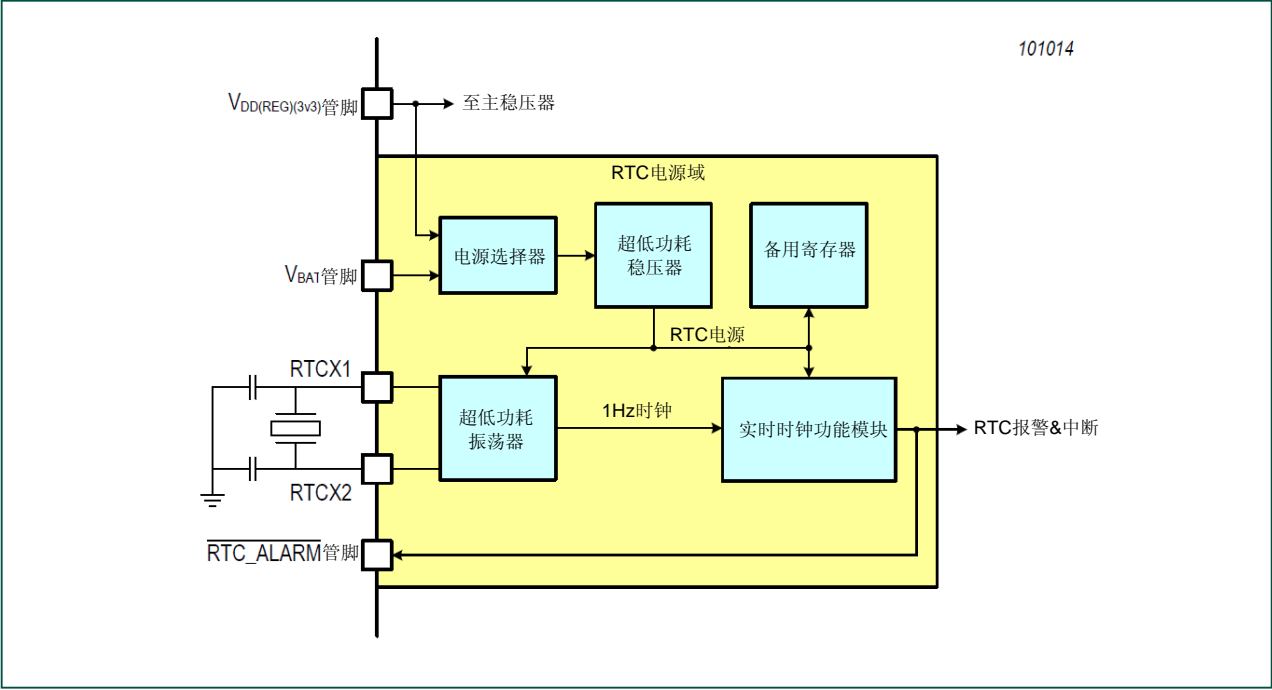
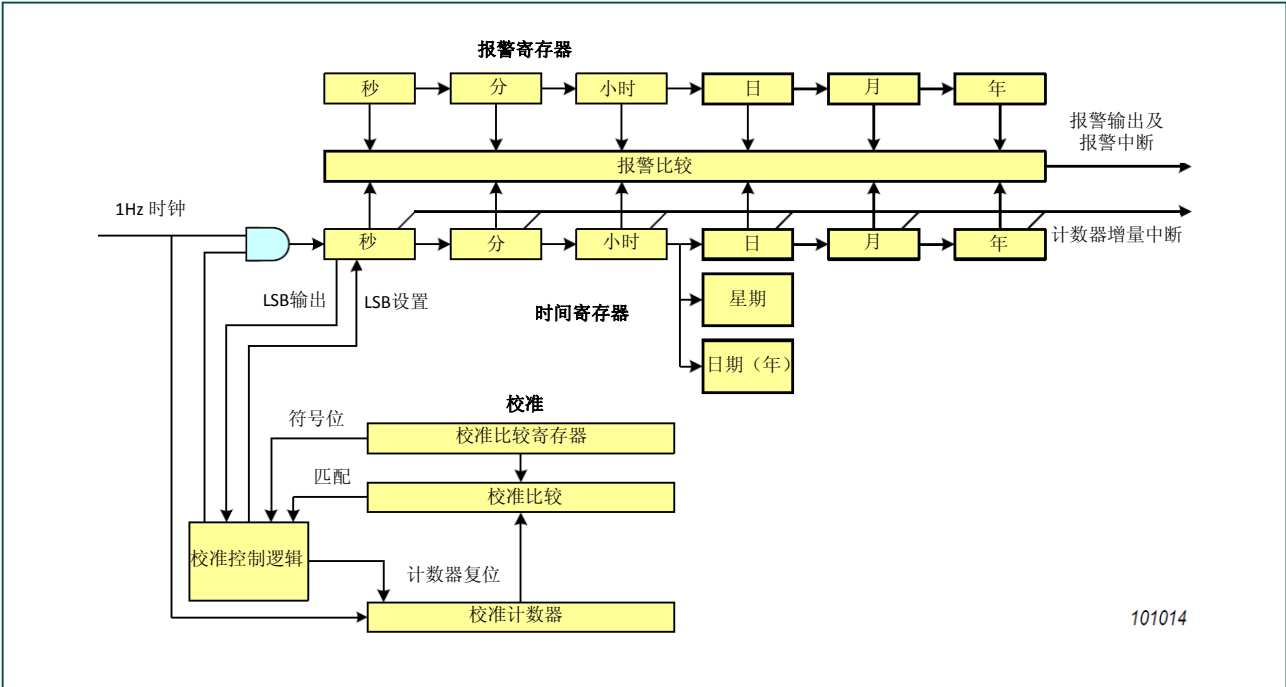


图153. RTC 功能方框图



29.5 管脚描述

表619. RTC 管脚描述

名称	类型	描述
RTC_ALARM	输出	RTC 的报警输出管脚。当内部 RTC 报警产生时，这是一个低有效的开漏管脚。向中断位置寄存器中的位 RTCALF 写入 1 可清除该输出管脚。该管脚由 V _{BAT} 供电。
RTCX1	输入	RTC 振荡器电路的输入管脚。
RTCX2	输出	RTC 振荡器电路的输出管脚。 备注： 如果不使用 RTC，RTCX1/2 管脚可被置为悬浮状态。
V _{BAT}	输入	RTC 电源： 通常与一个外部的 3V 电池相连。如果该管脚没有供电，则 RTC 仍由内部供电（如果有 V _{DD(REG)(3V3)} ）。

29.6 寄存器描述

RTC 包含了许多寄存器，如[表 620](#)所示。具体的描述见下文。在这些描述中，大多数寄存器的“复位值”一栏显示的都是“NC”，表示这些寄存器的值不因复位而改变。在从上电到 RTC 运行这段时间内，软件必须将这些寄存器初始化。这些寄存器依据功能被划分成五个部分。

表620. 实时时钟寄存器映射

名称	描述	访问	复位值 ^[1]	地址	表
混合寄存器（详见 29.6.2 节）					
ILR	中断位置寄存器	R/W	0	0x4002 4000	表 621
CCR	时钟控制寄存器	R/W	NC	0x4002 4008	表 622
CIIR	计数器增量中断寄存器	R/W	0	0x4002 400C	表 623
AMR	报警屏蔽寄存器	R/W	0	0x4002 4010	表 624
RTC_AUX	RTC 辅助控制寄存器	R/W	0x8	0x4002 405C	表 625
RTC_AUXEN	RTC 辅助使能寄存器	R/W	0	0x4002 4058	表 626
完整时间寄存器（详见 29.6.3 节）					
CTIME0	完整时间寄存器 0	RO	NC	0x4002 4014	表 627
CTIME1	完整时间寄存器 1	RO	NC	0x4002 4018	表 628
CTIME2	完整时间寄存器 2	RO	NC	0x4002 401C	表 629
时间计数器寄存器（详见 29.6.4 节）					
SEC	秒计数器	R/W	NC	0x4002 4020	表 631
MIN	分寄存器	R/W	NC	0x4002 4024	表 631
HOURL	小时寄存器	R/W	NC	0x4002 4028	表 631
DOM	日期（月）寄存器	R/W	NC	0x4002 402C	表 631
DOW	星期寄存器	R/W	NC	0x4002 4030	表 631
DOY	日期（年）寄存器	R/W	NC	0x4002 4034	表 631
MONTH	月寄存器	R/W	NC	0x4002 4038	表 631
YEAR	年寄存器	R/W	NC	0x4002 403C	表 631
CALIBRATION	校准值寄存器	R/W	NC	0x4002 4040	表 632
通用寄存器（详见 29.6.6 节）					
GPREG0	通用寄存器 0	R/W	NC	0x4002 4044	表 633
GPREG1	通用寄存器 1	R/W	NC	0x4002 4048	表 633
GPREG2	通用寄存器 2	R/W	NC	0x4002 404C	表 633
GPREG3	通用寄存器 3	R/W	NC	0x4002 4050	表 633
GPREG4	通用寄存器 4	R/W	NC	0x4002 4054	表 633
报警寄存器组（详见 29.6.7 节）					
ALSEC	秒报警值	R/W	NC	0x4002 4060	表 634
ALMIN	分报警值	R/W	NC	0x4002 4064	表 634
ALHOUR	小时报警值	R/W	NC	0x4002 4068	表 634
ALDOM	日期（月）报警值	R/W	NC	0x4002 406C	表 634
ALDOW	星期报警值	R/W	NC	0x4002 4070	表 634
ALDOY	日期（年）报警值	R/W	NC	0x4002 4074	表 634
ALMON	月报警值	R/W	NC	0x4002 4078	表 634
ALYEAR	年报警值	R/W	NC	0x4002 407C	表 634

[1] 复位值只用来上电 RTC 模块，而其他类型的复位对 RTC 模块都没有影响。
 由于 RTC 可以由 V_{DD(REG)(3V3)}或 V_{BAT} 供电（只要存在的话），所以上电复位只在两种电源都不存在时产生,然后其中一种电源开启。大部分寄存器都不受 RTC 上电的影响，如果 RTC 使能，这些寄存器必须通过软件进行初始化。复位值仅反映已使用位中保存的数据，不包括保留位的内容。

29.6.1 RTC 中断

中断的产生由中断位置寄存器（ILR）、计数器增量中断寄存器（CIIR）、报警寄存器和报警屏蔽寄存器（AMR）控制。只有切换到中断状态才能产生中断。ILR 分别使能 CIIR 和 AMR 中断。CIIR 中的每一位都对应一个时间计数器。如果使能其中某一位，那么该位对应的计数器每增加一次就产生一次中断。报警寄存器允许用户设定产生中断的日期和时间。AMR 提供一个屏蔽报警比较的机制。当所有非屏蔽报警寄存器均与它们对应的时间计数器的值匹配时，就会产生中断。

如果将 RTC 中断和事件记录器中断相结合，在运行 RTC 且 NVIC 中的 RTC/事件记录器中断被使能的情况下，就可以让微控制器退出睡眠、深度睡眠或掉电模式。要详细了解基于 RTC 的唤醒过程，请参见 4.7.9 节和 4.8 节。

29.6.2 混合寄存器组

29.6.2.1 中断位置寄存器（ILR—0x4002 4000）

中断位置寄存器是一个 2 位寄存器，用于指定哪些模块产生了中断（见表 621）。向该寄存器的对应位写入 1 可清除相应的中断。写入 0 无效。程序员可读取该寄存器并将读取出的值回写到寄存器中以清除检测到的中断。

表621. 中断位置寄存器位描述（ILR—地址 0x4002 4000）

位	符号	描述	复位值
0	RTCCIF	为 1 时，计数器增量中断模块产生中断。向该位写入 1 清除计数器增量中断。	0
1	RTCALF	为 1 时，报警寄存器产生中断。向该位写入 1 清除报警中断。	0
31:21	-	保留。读取值未定义，只写入 0。	无

29.6.2.2 时钟控制寄存器（CCR—0x4002 4008）

时钟控制寄存器是一个 4 位寄存器，用于控制时钟分频电路的操作。每位的功能详见表 622。在 RTC 首次开启的时候，该寄存器中的所有 NC 位都应被初始化。

表622. 时钟控制寄存器位描述（CCR—地址 0x4002 4008）

位	符号	值	描述	复位值
0	CLKEN		时钟使能。	NC
		1	时间计数器被使能。	
		0	时间计数器被禁能，所以它们可能被初始化。	
1	CTCRST		CTC 复位。	0
		1	为 1 时，内部振荡器分频器的元件全部复位，直至 CCR[1]变为 0。该分频器从 32.768 kHz 晶振中产生 1Hz 时钟。分频器的状态对软件是不可见的。	
		0	没有影响。	
3:2	-		内部测试模式控制。RTC 要正常运作，这些位必须为 0。	NC
4	CCALEN		校准计数器使能。	NC
		1	校准计数器被禁能并复位为 0。	
		0	校准计数器被使能并开始计数，频率为 1Hz。当校准计数值等于校准寄存器中的值时，计数器复位并重新向校准寄存器的值开始递增计数。详见 29.6.4.2 节和 29.6.5 节。	
31:5	-		保留。读取值未定义，只写入 0。	无

29.6.2.3 计数器增量中断寄存器（CIIR—0x4002 400C）

计数器增量中断寄存器（CIIR）可使计数器每增加 1 就产生一次中断。在清除增量中断之前，该中断一直保持有效。清除增量中断的方法：向中断位置寄存器（ILR[0]）的位 0 写入 1。

表623. 计数器增量中断寄存器位描述（CIIR—地址 0x4002 400C）

位	符号	描述	复位值
0	IMSEC	为 1 时，秒值的增加产生一次中断。	0
1	IMMIN	为 1 时，分值的增加产生一次中断。	0
2	IMHOUR	为 1 时，小时值的增加产生一次中断。	0
3	IMDOM	为 1 时，日期（月）值的增加产生一次中断。	0
4	IMDOW	为 1 时，星期值的增加产生一次中断。	0
5	IMDOY	为 1 时，日期（年）值的增加产生一次中断。	0
6	IMMON	为 1 时，月值的增加产生一次中断。	0
7	IMYEAR	为 1 时，年值的增加产生一次中断。	0
31:8	-	保留。读取值未定义，只写入 0。	无

29.6.2.4 报警屏蔽寄存器（AMR—0x4002 4010）

报警屏蔽寄存器（AMR）允许用户屏蔽所有的报警寄存器。[表 624](#) 所示为 AMR 中的各位与报警寄存器之间的关系。对于报警功能来说，若要产生中断，未被屏蔽的报警寄存器必须与对应的时间计数器相匹配。且只在第一次从不匹配到匹配时才会产生中断。若向中断位置寄存器（ILR）的相关位写入 1，则相应的中断就会被清除。如果所有屏蔽位都置位，报警将被禁止。

表624. 报警屏蔽寄存器位描述（AMR—地址 0x4002 4010）

位	符号	描述	复位值
0	AMRSEC	为 1 时，秒计数值不与报警寄存器比较。	0
1	AMRMIN	为 1 时，分计数值不与报警寄存器比较。	0
2	AMRHOUR	为 1 时，小时计数值不与报警寄存器比较。	0
3	AMRDOM	为 1 时，日期（月）计数值不与报警寄存器比较。	0
4	AMRDOW	为 1 时，星期计数值不与报警寄存器比较。	0
5	AMRDOY	为 1 时，日期（年）计数值不与报警寄存器比较。	0
6	AMRMON	为 1 时，月计数值不与报警寄存器比较。	0
7	AMRYEAR	为 1 时，年计数值不与报警寄存器比较。	0
31:8	-	保留。读取值未定义，只写入 0。	无

29.6.2.5 RTC 辅助控制寄存器（RTC_AUX—0x4002 405C）

RTC 辅助控制寄存器保存了一些附加的中断标志，这些标志用于那些不属于实时时钟本身的部分功能（即记录时间经过和产生与功能相关的时间的部分）。在 LPC178x/177x 系列微控制器中，只有附加的中断标志对 RTC 振荡器无效。

表625. RTC 辅助控制寄存器（RTC_AUX—地址 0x4002 405C）

位	符号	描述	复位值
3:0	-	保留。读取值未定义，只写入 0。	无
4	RTC_OSCF	RTC 振荡器失效探测标志。 读：该位在 RTC 振荡器停止时置位，或在 RTC 电源首次启动时置位。该位置位时，中断产生，RTC_AUXEN 中的位 RTC_OSCFEN 被置位为 1，且 NVIC 中的 RTC 中断被使能。 写：向该位写入 1 会清除这个标志。	1
5	-	保留。读取值未定义，只写入 0。	无
6	RTC_PDOUT	为 0 时：RTC_ALARM 管脚反映了 RTC 报警状态。 为 1 时：RTC_ALARM 管脚表示深度掉电模式。	0
31:7	-	保留。读取值未定义，只写入 0。	无

29.6.2.6 RTC 辅助使能寄存器（RTC_AUXEN—0x4002 4058）

RTC 辅助使能寄存器控制着是否有其他 RTC 辅助控制寄存器的中断源被使能。

表626. RTC 辅助使能寄存器位描述（RTC_AUXEN—地址 0x4002 4058）

位	符号	描述	复位值
3:0	-	保留。读取值未定义，只写入 0。	无
4	RTC_OSCFEN	振荡器失效探测中断使能。 为 0 时：RTC 振荡器失效探测中断被禁止。 为 1 时：RTC 振荡器失效探测中断被使能。详见 29.6.2.5 节。	0
31:5	-	保留。读取值未定义，只写入 0。	无

29.6.3 完整时间寄存器

时间计数器的值可选择以一个完整格式读出，这样程序员只需执行 3 次读操作即可读出所有时间计数器的值。完整时间寄存器都为 32 位，见表 627、表 628 和表 629。每个寄存器的最低位分别位于寄存器的位 0、位 8、位 16 或位 24。

完整时间寄存器为只读寄存器。要向时间计数器写入新的值，必须使用时间计数器地址

29.6.3.1 完整时间寄存器 0（CTIME0—0x4002 4014）

完整时间寄存器 0 包含时间值中的低位：秒、分、小时和星期。

表627. 完整时间寄存器 0 位描述（CTIME0—地址 0x4002 4014）

位	符号	描述	复位值
5:0	秒	秒值，该值范围为 0-59	NC
7:6	-	保留。从保留位读出的值未定义。	无
13:8	分	分值，该值范围为 0-59	NC
15:14	-	保留。从保留位读出的值未定义。	无
20:16	小时	小时值，该值范围为 0-23	NC
23:21	-	保留。从保留位读出的值未定义。	NC
26:24	星期	星期值，该值范围为 0-6	无
31:27	-	保留。从保留位读出的值未定义。	NC

29.6.3.2 完整时间寄存器 1（CTIME1—0x4002 4018）

完整时间寄存器 1 包含的时间值为：日期（月）、月和年。

表628. 完整时间寄存器 1 位描述（CTIME1—地址 0x4002 4018）

位	符号	描述	复位值
4:0	日期（月）	日期（月）值，该值范围为 1-28、29、30 或 31（取决于月份以及是否为闰年）。	NC
7:5	-	保留。从保留位读出的值未定义。	无
11:8	月	月值，该值范围为 1-12。	NC
15:12	-	保留。从保留位读出的值未定义。	无
27:16	年	年值，该值范围为 0-4095。	NC
31:28	-	保留。从保留位读出的值未定义。	无

29.6.3.3 完整时间寄存器 2（CTIME2—0x4002 401C）

完整时间寄存器 2 仅包含的值为：日期（年）。

表629. 完整时间寄存器 2 位描述（CTIME2—地址 0x4002 401C）

位	符号	描述	复位值
11:0	日期（年）	日期（年）值，该值范围为 1-365（闰年为 366）。	NC
31:12	-	保留。从保留位读出的值未定义。	无

29.6.4 时间计数器组

时间值由 8 个计数器值组成，见表 630 和表 631。这些计数器可对表 631 中所示的单元进行读/写。

表630. 时间计数器的关系和值

计数器	规格	计数器驱动源	最小值	最大值
秒	6	1Hz 时钟	0	59
分	6	秒	0	59
小时	5	分	0	23
日期（月）	5	小时	1	28、29、30 或 31
星期	3	小时	0	6
日期（年）	9	小时	1	365 或 366（闰年）
月	4	日期（月）	1	12
年	12	月或日期（年）	0	4095

表631. 时间计数器寄存器

名称	规格	描述	访问	地址
SEC	6	秒值，该值范围为 0-59	R/W	0x4002 4020
MIN	6	分值，该值范围为 0-59	R/W	0x4002 4024
HOURL	5	小时值，该值范围为 0-23	R/W	0x4002 4028
DOM	5	日期（月）值，该值范围为 1-28、29、30 或 31（取决于月份以及是否为闰年）。[1]	R/W	0x4002 402C
DOW	3	星期值，该值范围为 0-6[1]	R/W	0x4002 4030
DOY	9	日期（年）值，该值范围为 1-365（闰年为 366）[1]	R/W	0x4002 4034
MONTH	4	月值，该值范围为 1-12	R/W	0x4002 4038
YEAR	12	年值，该值范围为 0-4095	R/W	0x4002 403C

[1] 这些值在经过了适当的时间间隔后递增，在发生定义的溢出时复位。
不建议通过计算获得这些值，为了让这些值更准确，应当进行正确的初始化。

29.6.4.1 闰年计算

RTC 执行一个简单的位比较，看年计数器的最低两位是否为 0。如果为 0，那么 RTC 认为这一年为闰年。RTC 认为所有能被 4 整除的年份都为闰年。这个算法从 1901 年到 2099 年都是准确的，但会在 2100 年出错，2100 年并不是闰年。闰年对 RTC 的影响只是改变 2 月份的天数、日期（月）和年的计数值。

29.6.4.2 校准寄存器（CALIBRATION–地址 0x4002 4040）

下列寄存器可用于时间计数器的校准。

表632. 校准寄存器位描述（CALIBRATION—地址 0x4002 4040）

位	符号	值	描述	复位值
16:0	CALVAL	-	如果校准使能，校准计数器会向该值递增计数。最大值为 131,072，对应的计数时间大约是 36.4 小时。 如果 CALVAL=0，校准功能禁止。	NC
17	CALDIR		校准方向	NC
		1	逆向校准。当 CALVAL 等于校准计数值时，RTC 定时器会停止 1 秒再递增。	
		0	正向校准。当 CALVAL 等于校准计数值时，RTC 定时器会跳进 2 秒。	
31:12			保留。读取值未定义，只写入 0。	无

29.6.5 校准过程

校准逻辑会定时通过使计数器的值增加 2（而非 1）来对时间计数器进行调整。这样就可以在典型电压和适当的温度下对 RTC 振荡器直接进行校准，无需通过外部仪器来调节 RTC 振荡器。

建议用来确定校准值的方法：利用 CLKOUT 特性，在对 RTC 进行调节的情况下观察 RTC 振荡器的频率，计算在时间结束之前所观察到的时钟数，再用这个值来确定 CALVAL。

如果 RTC 振荡器需要通过外部调节，那么观察 RTC 振荡器频率的这种方法有助于外部调节过程。

向后校准

使能 RTC 定时器，在寄存器 CCR 中进行校准（置位 CLKEN=1、CCALEN=0）。把校准寄存器中的校准值 CALVAL 设置成大于等于 1 的值，并将位 CALDIR 设为 1。

- 每隔一个时钟周期（1Hz），SEC 定时器和校准计数器加 1；
- 在校准计数值达到 CALVAL 时，出现校准匹配，所有 RTC 定时器将停止运行一个时钟周期，这样定时器就不会在下一个周期后加 1；
- 若在出现校准匹配的同时也出现报警匹配，则报警中断会被延迟一个周期以免产生两次报警中断。

向前校准

使能 RTC 定时器，并寄存器 CCR 中进行校准（置位 CLKEN=1、CALEN = 0）。把校准寄存器中的校准值 CALVAL 设置成大于等于 1 的值，并将位 CALDIR 设为 0。

- 每隔一个时钟周期（1Hz），SEC 定时器和校准计数器加 1；
- 在校准计数值达到 CALVAL 时，校准匹配出现，RTC 定时器加 2；
- 当出现校准事件时，寄存器 ALSEC 的 LSB 值会强制变为 1，这样报警中断就不会在秒值跳跃时丢失。

29.6.6 通用寄存器

29.6.6.1 通用寄存器 0~4（GPREG0~GPREG4—地址 0x4002 4044~0x4002 4054）

这类寄存器可在主电源断开时保存重要的信息。芯片复位时，不会影响到这些寄存器中的值。

表633. 通用寄存器 0~4 位描述（GPREG0~GPREG4—地址 0x4002 4044~0x4002 4054）

位	符号	描述	复位值
31:0	GP0-GP4	通用存储器。	无

29.6.7 报警寄存器组

报警寄存器见表 634。将这些寄存器中的值与时间计数器进行比较。如果所有未屏蔽（请参见 29.6.2.4 节）的报警寄存器都与它们对应的时间计数器相匹配，那么将产生一次中断。向中断位置寄存器（ILR[1]）的位 1 写入 1，就可清除中断。

表634. 报警寄存器

名称	规格	描述	访问	地址
ALSEC	6	秒报警值	R/W	0x4002 4060
ALMIN	6	分报警值	R/W	0x4002 4064
ALHOUR	5	小时报警值	R/W	0x4002 4068
ALDOM	5	日期（月）报警值	R/W	0x4002 406C
ALDOW	3	星期报警值	R/W	0x4002 4070
ALDOY	9	日期（年）报警值	R/W	0x4002 4074
ALMON	4	月报警值	R/W	0x4002 4078
ALYEAR	12	年报警值	R/W	0x4002 407C

29.7 使用注意事项

若使用了 RTC，就必须将 V_{BAT} 与独立的电源相连（通常是外部蓄电池），或将 V_{BAT} 悬空。如果有 V_{DD(3V3)}，即便 V_{BAT} 没有与电源连接，内部也会一直为 RTC 域供电。只要 V_{DD(REG)(3V3)} 或 V_{BAT} 可用，RTC 就不会丢失其它的时间值和备份寄存器中的内容。如果 V_{DD(3V3)} 和 V_{BAT} 都不可用，所有的 RTC 信息都将会丢失。如果时钟源丢失、被中断或改变，RTC 也会停止递增或不可预测。事件记录器的供电方式与 RTC 相同。

30.1 基本配置

使用下列寄存器来配置事件监控器/记录器：

1. 功率：在寄存器 PCONP（见[表 37](#)）中置位 PCRTC（适用于 RTC 和事件监测器/记录器）。

注：复位后，事件监测器/记录器被使能。请参考《LPC178x/7x 用户手册》中 RTC 相关章节。

2. 时钟：事件监测器/记录器将 RTC 振荡器提供的 1Hz 时钟信号做为内部功能时钟。外设时钟被用于访问 RTC 寄存器。
3. 中断：有关 RTC/事件监测器/记录器的中断处理，请参考《LPC178x/7x 用户手册》中 RTC 相关章节。利用相应的中断置位使能寄存器来使能 NVIC 中的中断。

30.2 特性

- V_{BAT} 电源域中有三路数字事件输入。
- 事件就是数字事件输入发生了电平变化。
- 每个事件通道中都有两个时间戳来标识事件的开始和结束。每个通道中还有专门的计数器用来跟踪事件的总数。从 RTC 中获取时间戳值（请参考《LPC178x/7x 用户手册》中 RTC 相关章节）。
- 在 V_{BAT} 电源域中运行，与系统电源无关。因此可以在深度掉电模式下运行。
- 超低功耗设计。
- 在系统上电之后，可以产生中断。
- 合格事件可以被用作唤醒触发器。
- 可以通过软件经通用 I/O 获取事件输入的状态。

30.3 应用

记录对密封产品外壳的侵入事件。当有人试图打开产品外壳，或以任何方式损害器件的时候，传感器会将这些情报上报。事件监测器/记录器主要用来在设备仅由备用电池供电的情况下，存储这类事件的记录。

30.4 描述

在典型应用中，将两路事件输入连接到保护计量器盖和端子盖的开关。第三路事件输入可监测磁场传感器的输出实例。

事件监测器/记录器一直依赖于 V_{BAT} 工作。 V_{BAT} 电压的丢失或骤降都会导致实时时钟掉电检测器将事件记录复位。因此，需要有一个 V_{BAT} 电源，在所能预期的最长市电中断期间也能作为事件检测器/记录器供电。

在系统电源恢复之后，CPU 可以检测是否有侵入事件的记录。若有侵入事件，则首个和末尾事件的时间戳寄存器将标示这些事件发生的持续时间。

可用 1kHz 时钟、64Hz 时钟或 16Hz 时钟对事件输入的边沿进行采样。这个时钟的两个连续边沿需要捕捉正/反方向的跳变，便于确定是否为有效跳变。1kHz 采样时钟会提供一个 1-2ms 的带阻滤波器。64Hz 采样时钟会提供一个 15.6-31.2ms 的带阻滤波器。16Hz 采样时钟会提供一个 62.5-125ms 的带阻滤波器。这样的事件会将 ERSTATUS 寄存器中的 EVx 位在 1Hz 时钟的下一个上升沿上置位。

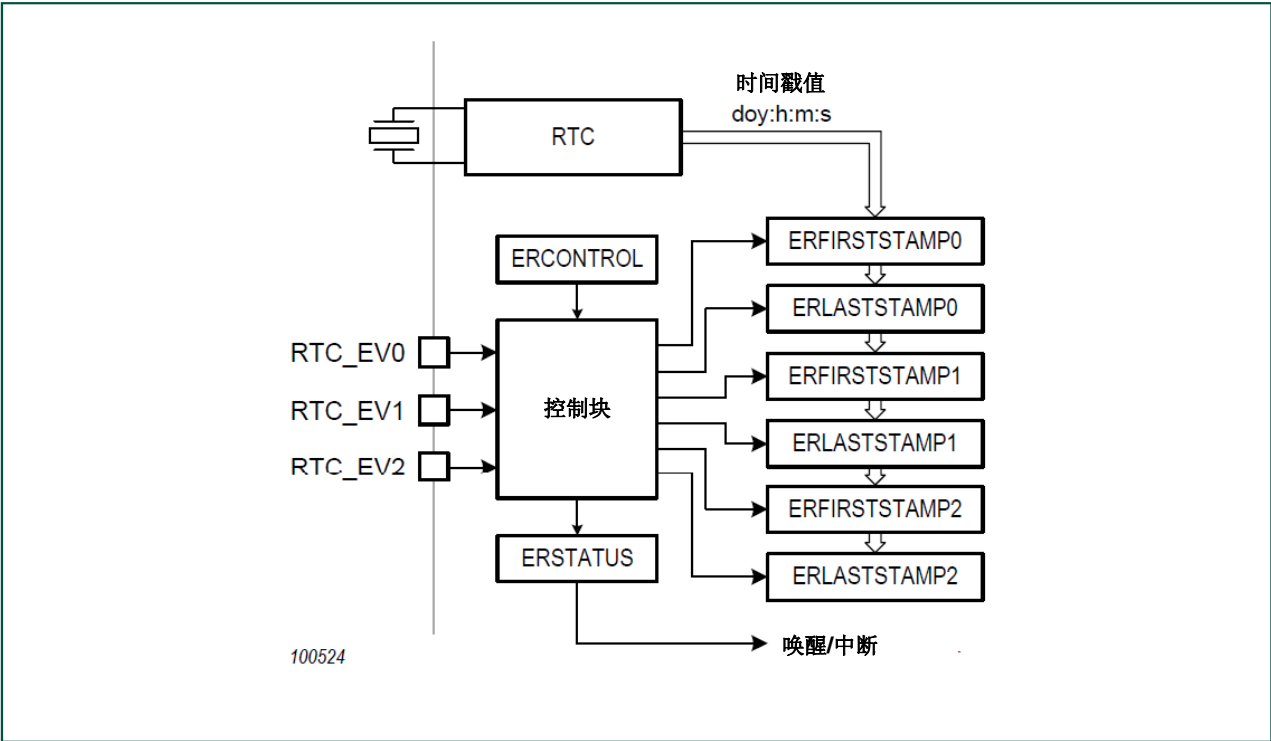
若有新事件发生，则将从 RTC 中获取一个时间戳并存储在 ERLASTSTAMPx 寄存器中，此后，每有新事件发生，这个时间戳就会被更新。如果该事件是状态寄存器中的 EVx 位自上一次被清除之后发生的首个事件，那么 ERFIRSTSTAMPx 寄存器也会被更新。

除了取走时间戳之外，一个 3 位的计数器（ERCOUNTERx）会在 1Hz 时钟的上升沿进行递增计数（即，与 ERLASTSTAMPx 寄存器的更新一致）。这个计数器在计数到 7 时会停止计数并处于保持状态。在软件清除状态寄存器中的 EVx 位时，这个计数器也会自动清零。

一个事件可以被使能用来在 RTC 模块中异步清除备份寄存器。即便在 32kHz 振荡器没有运行或事件监测器/记录器被禁止的情况下，这一功能也可行。

下图所示为事件监测器/记录器的框图。

图154. 事件监控器/记录器框图



CPU 可能随时会检查 ERSTATUS 寄存器，查看是否有事件发生。若是位 EV0 被置位，那么对应的 ERFIRSTSTAMP0 和 ERLASTSTAMP0 寄存器会包含有效时间戳。ERCOUNTER0 寄存器还会包含通道 0 中发生的所有事件数的有效计数（最大值为 7）。

一旦读取到（不公开的）时间戳，CPU 会通过向 ERSTATUS.EVx 位写入 1 来清除该位。

在 ERSTATUS.EVx 位被清除后，CPU 将忽略时间戳寄存器。在状态寄存器中的事件标志被清除之后，就没有清除或让时间戳失效的方法了。在新的合格事件发生之前，时间戳寄存器将一直保持旧有的值。这类合格事件会对 ERSTATUS.EVx 位置位，并告知 CPU 时间戳寄存器中包含有了新的值。

通过设置 ERCONTROL 寄存器中的 INTWAKE_ENAx 位，某个事件通道就可以做为一个唤醒触发器信号。因此，该事件通道中发生的事件会将设备从省电模式中唤醒。

30.5 管脚描述

表635. 事件监测器/记录器管脚描述

名称	类型	描述
RTC_EV0	输入	事件监测器/记录器通道 0 的事件输入管脚。
RTC_EV1	输入	事件监测器/记录器通道 1 的事件输入管脚。
RTC_EV2	输入	事件监测器/记录器通道 2 的事件输入管脚。

30.6 寄存器描述

事件监测器/记录器包含若干寄存器，如表 636 所示。下面将对这些寄存器进行详细描述。在这些描述中，表中所示的复位值仅适用于 RTC 域上电模式，也就是说，这些寄存器不会受设备复位的影响。

表636. 事件监测器/记录器寄存器映射

名称	描述	访问	复位值 ^[1]	地址
ERCONTROL	事件监测器/记录器控制寄存器。含有对事件通道以及事件监测器/记录器设置行为起控制作用的位。	R/W	0	0x4002 4084
ERSTATUS	事件监测器/记录器状态寄存器。含有事件通道以及其他事件监测器/记录器情况的状态标志。	R/W	0	0x4002 4080
ERCOUNTERS	事件监测器/记录器计数器寄存器。允许读取与事件通道关联的计数器。	RO	0	0x4002 4088
ERFIRSTSTAMP0	通道 0 的事件监测器/记录器首个时间戳寄存器。保留通道 0 上首个事件的时间戳。	RO	无	0x4002 4090
ERFIRSTSTAMP1	通道 1 的事件监测器/记录器首个时间戳寄存器（详见 ERFIRSTSTAMP0 的描述）。	RO	无	0x4002 4094
ERFIRSTSTAMP2	通道 2 的事件监测器/记录器首个时间戳寄存器（详见 ERFIRSTSTAMP0 的描述）。	RO	无	0x4002 4098
ERLASTSTAMP0	通道 0 的事件监测器/记录器末尾时间戳寄存器。保留通道 0 上末次事件（即最新发生的事件）的时间戳。	RO	无	0x4002 40A0
ERLASTSTAMP1	通道 1 的事件监测器/记录器末尾时间戳寄存器（详见 ERLASTSTAMP0 的描述）。	RO	无	0x4002 40A4
ERLASTSTAMP2	通道 2 的事件监测器/记录器末尾时间戳寄存器（详见 ERLASTSTAMP0 的描述）。	RO	无	0x4002 40A8

[1] 复位值仅适用于处于上电模式的事件监测器/记录器模块，而其他类型的复位对该模块都没有影响。由于监控器/记录器可以由 V_{DD(REG)(3V3)} 或 V_{BAT} 供电（只要存在的话），所以上电复位只在两种电源都不存在时产生，然后其中一种电源开启。复位值仅反映已使用位中保存的数据，不包括保留位的内容。

30.6.1 事件监测器/记录器控制寄存器（ERCONTROL—0x4002 4084）

事件监测器/记录器控制寄存器允许对事件监测器/记录器进行设置，并允许对每一通道操作内容进行单独控制。

表637. 事件监测器/记录器控制寄存器位描述（ERCONTROL—0x4002 4084）

位	符号	值	描述	复位值
0	INTWAKE_EN0		通道 0 的中断和唤醒使能。	0
		0	事件通道 0 不会触发中断或唤醒的产生。	
		1	通道 0 中的事件将触发一个（RTC）中断及唤醒请求。	
1	GPCLEAR_EN0		当通道 0 中发生事件时，该寄存器自动使能清除 RTC 通用寄存器。	0
		0	通道 0 对通用寄存器没有影响。	
		1	通道 0 中发生的一个事件会异步清除通用寄存器。	
2	POL0		选择输入管脚 RTC_EV0 上的一个事件的极性。	0
		0	一个通道 0 事件被定义为 RTC_EV0 上的负相沿。	
		1	一个通道 0 事件被定义为 RTC_EV0 上的正相沿。	
3	EV0_INPUT_EN		通道 0 的事件使能控制。 [1]	0
		0	事件 0 输入被禁止且在内部被强迫为高电平。	
		1	事件 0 输入被使能。	
9:4	-		保留。读取值未定义，只写入 0。	无
10	INTWAKE_EN1		通道 1 的中断和唤醒使能。	0
		0	事件通道 1 不会触发中断或唤醒的产生。	
		1	通道 1 中的事件将触发一个（RTC）中断及唤醒请求。	
11	GPCLEAR_EN1		当通道 1 中发生事件时，该寄存器自动使能清除 RTC 通用寄存器。	0
		0	通道 1 对通用寄存器没有影响。	
		1	通道 1 中发生的一个事件会异步清除通用寄存器。	
12	POL1		选择输入管脚 RTC_EV1 上一个事件的极性。	0
		0	一个通道 1 事件被定义为输入管脚 RTC_EV1 上的负相沿。	
		1	一个通道 1 事件被定义为输入管脚 RTC_EV1 上的正相沿。	
13	EV1_INPUT_EN		通道 1 的事件使能控制。 [1]	0
		0	事件 1 输入被禁止且在内部被强迫为高电平。	
		1	事件 1 输入被使能。	
19:14	-		保留。读取值未定义，只写入 0。	无
20	INTWAKE_EN2		通道 2 的中断和唤醒使能。	0
		0	事件通道 2 不会触发中断或唤醒的产生。	
		1	通道 2 中的事件将触发一个（RTC）中断及唤醒请求。	
21	GPCLEAR_EN2		当通道 2 中发生事件时，该寄存器自动使能清除 RTC 通用寄存器。	0
		0	通道 2 对通用寄存器没有影响。	
		1	通道 2 中发生的一个事件会异步清除通用寄存器。	

位	符号	值	描述	复位值
22	POL2		选择输入管脚 RTC_EV2 上一个事件的极性。	0
		0	一个通道 2 事件被定义为输入管脚 RTC_EV2 上的负相沿。	
		1	一个通道 2 事件被定义为 RTC_EV2 上的正相沿。	
23	EV2_INPUT_EN		通道 2 的事件使能控制。 ^[1]	0
		0	事件 2 输入被禁止且在内部被强迫为高电平。	
		1	事件 2 输入被使能。	
29:24	-		保留。读取值未定义，只写入 0。	无
31:30	ERMODE		通过控制能够使能事件监测器/记录器并选择它的工作频率。 ^[2]	0
		00	事件监测/记录器时钟被禁止。 除了异步清除 GP 寄存器（如果已选），事件监控器/记录器的其他操作被禁止。	
		01	使能事件监测器/记录器并且为事件输入沿检测和毛刺抑制选择一个 16Hz 的样本时钟。 任一方向上低于 62.5ms 至 125ms 的脉冲将被过滤掉。	
		10	使能事件监测器/记录器并且为事件输入沿检测和干扰抑制选择一个 64Hz 的样本时钟。 任一方向上低于 15.6ms 至 31.2ms 的脉冲将被过滤掉。	
		11	使能事件监测器/记录器并且为事件输入沿检测和干扰抑制选择一个 1kHz 的样本时钟。 任一方向上低于 1ms 至 2ms 的脉冲将被过滤掉。	

[1] 当事件输入未被用做事件检测时，应处于禁止状态，尤其是当相关的管脚被用在其他功能上时。

[2] 事件监测器/记录器通常能被写入这些位，无论这些位处于什么状态。时钟使能后的一秒钟间隔内发生的事件可能无法被识别。

30.6.2 事件监测器/记录器状态寄存器（ERSTATUS—0x4002 4080）

事件监测器/记录器状态寄存器包含 3 个事件通道的标志、通用寄存器清除标志和中断/唤醒标志。

表638. 事件监测器/记录器状态寄存器位描述（ERSTATUS—0x4002 4080）

位	符号	值	描述	复位值
0	EV0		通道 0 的事件标志（RTC_EV0 管脚）。如果在前一秒有事件发生，该位在接下来任一秒末置位。向该位写入 1 即可将其清除。向该位写入 0 没有任何影响。	0
		0	在通道 0 中没有事件变化。	
		1	通道 0 中至少发生了一个事件。	
1	EV1		通道 1 的事件标志（RTC_EV1 管脚）。如果在前一秒有事件发生，该位在接下来任一秒末置位。向该位写入 1 即可将其清除。写 0 没有影响。	0
		0	通道 1 中没有事件变化。	
		1	通道 1 至少有一个事件发生。	
2	EV2		通道 2（RTC_EV2 管脚）的事件标志。如果在前一秒有事件发生，该位在接下来任一秒末置位。向该位写入 1 即可将其清除。向该位写入 0 没有任何影响。	0
		0	通道 2 中没有事件变化。	
		1	通道 2 中至少发生了一个事件。	
3	GP_CLEARED		通用寄存器异步清除标志。向该位写入 1 即可将其清除。向该位写入 0 没有任何影响。	0
		0	通用寄存器没有被异步清除。	
		1	通用寄存器已被异步清除。	
30:4 -			保留。读取值未确定，只写入 0。	无
31	WAKEUP		中断/唤醒请求标志（只读）。向该位写入 1 即可将其清除。向该位写入 0 没有任何影响。	0
		0	没有中断/唤醒请求处于挂起状态。	
		1	一个中断/唤醒请求处于挂起状态。	

30.6.3 事件监测器/记录器计数器寄存器（ERCOUNTERS—0x4002 4088）

事件监测器/记录器计数器寄存器是一个只读寄存器，允许读取计数器。计数器中记录了每个事件监测器/记录器通道中发生的事件数。

表639. 事件监测器/记录器计数器寄存器位描述（ERCOUNTERS—0x4002 4088）

位	符号	描述	复位值
2:0	COUNTER0	事件 0 的计数器值。如果计数器计到满数（值 7），当另外有事件发生时，该计数器值仍然不变。当 ERSTATUS 寄存器中相应的 Evx 位被软件清零时，该计数器也被清零。	0
7:3	-	保留。从保留位读出的值未定义。	无
10:8	COUNTER1	事件 1 的计数器值。详见计数器 0 的描述。	0
15:11	-	保留。从保留位读出的值未定义。	无
18:16	COUNTER2	事件 2 的计数器值。详见计数器 0 的描述。	0
31:19	-	保留。从保留位读出的值未定义。	无

30.6.4 事件监测器/记录器首个时间戳寄存器
（ERFIRSTSTAMP0—0x0x4002 4090，ERFIRSTSTAMP1—0x0x4002 4094，ERFIRSTSTAMP2—0x4002 4098）

只读事件监测器/记录器首个时间戳寄存器记录了每个事件监测器/记录器通道中发生的首个事件的时间戳（来自 RTC），前提是 ERSTATUS 寄存器中相应的 EVx 位被置位。一旦记录下来，这些寄存器中的值将保持不变，直到软件清除 ERSTATUS 寄存器中相对应的 EVx 位。

只有在 ERSTATUS 寄存器中对应的 EVx 位等于 1 的情况下，这些寄存器中的内容才有效。

表640. 事件监测器/记录器首个时间戳寄存器位描述（ERFIRSTSTAMP0—0x0x4002 4090，ERFIRSTSTAMP1—0x0x4002 4094，ERFIRSTSTAMP2—0x4002 4098）

位	符号	描述	复位值
5:0	SEC	秒值，该值范围为 0-59。	无
11:6	MIN	分值，该值范围为 0-59。	无
16:12	HOURL	小时值，该值范围为 0-23。	无
25:17	DOY	日期（年）值，该值范围为 1-366。	无
31:26	-	保留。从保留位读出的值未定义。	无

30.6.5 事件监测器/记录器末尾时间戳寄存器
(ERLASTSTAMP0—0x0x4002 40A0 , ERLASTSTAMP1—0x0x4002 40A4 ,
ERLASTSTAMP2—0x4002 40A8)

当事件监测器/记录器各个通道中发生事件的时候，只读事件监测器/记录器末尾时间戳寄存器会记录一个时间戳。

只有在 ERSTATUS 寄存器中对应的 EVx 位等于 1 的情况下,这些寄存器中的内容才有效。

注意，每个通道发生首个事件之后，相对应的 ERFIRSTSTAMP 和 ERLASTSTAMP 寄存器中的内容是一样的（首个事件和最近发生的事件是相同的）。若在同一个通道发生第二个事件，则这两个寄存器中的值就会不同了。

表641. 事件监测器/记录器末尾时间戳寄存器位描述（ERLASTSTAMP0—0x0x4002 40A0，
ERLASTSTAMP1—0x0x4002 40A4，ERLASTSTAMP2—0x4002 40A8）

位	符号	描述	复位值
5:0	SEC	秒值，该值范围为 0-59。	无
11:6	MIN	分值，该值范围为 0-59。	无
16:12	HOURL	小时值，该值范围为 0-23。	无
25:17	DOY	日期（年）值，该值范围为 1-366。	无
31:26	-	保留。从保留位读出的值未定义。	无

31.1 特性

- 如果在可编程超时期间内没有被重载，可内部复位芯片。
- 可选的窗口式操作要求在最小和最大超时期间（两者都可通过编程设定）之间被重载。
- 在看门狗超时之前一段时间（可通过编程设定）可选择生成报警中断。
- 可编程的 24 位定时器，带内部固定的预分频器。
- 时间周期的范围在 1,024 个（ $TWDCLK \times 256 \times 4$ ）看门狗时钟至 6700 万个（ $TWDCLK \times 224 \times 4$ ）看门狗时钟之间，每次递增 4 个看门狗时钟。
- “安全的”看门狗操作模式。一旦被使能，要求通过硬件复位或看门狗复位来禁止。
- 有专用的片上看门狗振荡器，用于提供可靠时钟源。这一时钟源即使在运行看门狗定时器的时候也不会被关掉。
- 在看门狗被使能的情况下，若有错误的喂狗序列会马上导致看门狗复位。
- 可以有选择地保护看门狗的重载值，这样这个值只有在达到“报警中断”时间之后才会发生改变。
- 具有指示看门狗发生复位的标识。

31.2 应用

看门狗定时器的用途是在微控制器进入错误状态后的一段合理的时间内将其复位。看门狗被使能后，如果用户程序没有在预先设定的时间内喂狗（即给看门狗定时器重载定时值），那么将会产生一个看门狗事件。看门狗事件会导致芯片复位（如果事先已做好相关设置）。

在对看门狗窗口编程结束后，早期“喂狗”也会被作为看门狗事件处理。这样就可以避免在系统故障的情况下仍继续“喂狗”。例如，在包含有喂狗操作的一个中断服务中，应用程序代码可能被卡住。通过设置窗口就可发生早期喂狗，进而生成一个看门狗事件，为系统恢复做准备。

有关片上看门狗和其它外设之间的相互作用（尤其是复位和启动程序），请参考 [3.4](#) 节。

31.3 描述

看门狗定时器包括一个 4 分频的预分频器和一个 24 位的计数器（递减计数）。计数器递减的最小值为 0xFF。如果设置一个小于 0xFF 的值，系统会将 0xFF 装入计数器。因此看门狗定时器的最小定时间隔为 ($T_{WDCLK} \times 256 \times 4$)，最大定时间隔为 ($T_{WDCLK} \times 2^{24} \times 4$)，两者都是 $T_{WDCLK} \times 4$ 的倍数。看门狗应按照下面的方法来使用：

- 在 WDTCR 寄存器中设置看门狗定时器常数重载值。
- 在 WDMOD 寄存器中设置看门狗定时器的工作模式。
- 若要求窗口式操作，应在 WDWINDOW 寄存器中设置看门狗窗口时间值。
- 若要求报警中断，应在 WDWARNINT 寄存器中设置一个用于产生看门狗报警中断的值。
- 通过向 WDFEED 寄存器依序写入 0xAA 和 0x55 来启动看门狗。
- 为了避免出现看门狗事件，在看门狗计数器递减至 0 之前，应再次“喂狗”。如果已经通过程序设置了一个窗口值，那么在看门狗计数器超过这个值以后也应再次“喂狗”。

若看门狗定时器被设置为当看门狗事件发生时会导致复位且计数器为 0，那么当看门狗事件发生时，会引发 CPU 复位，并从向量表中加载栈指示器和编程计数器（与外部复位时的情况一样）。通过检查看门狗超时标志 (WDTOF) 可以判断系统复位是否由看门狗引发。WDTOF 标志必须由软件清零。

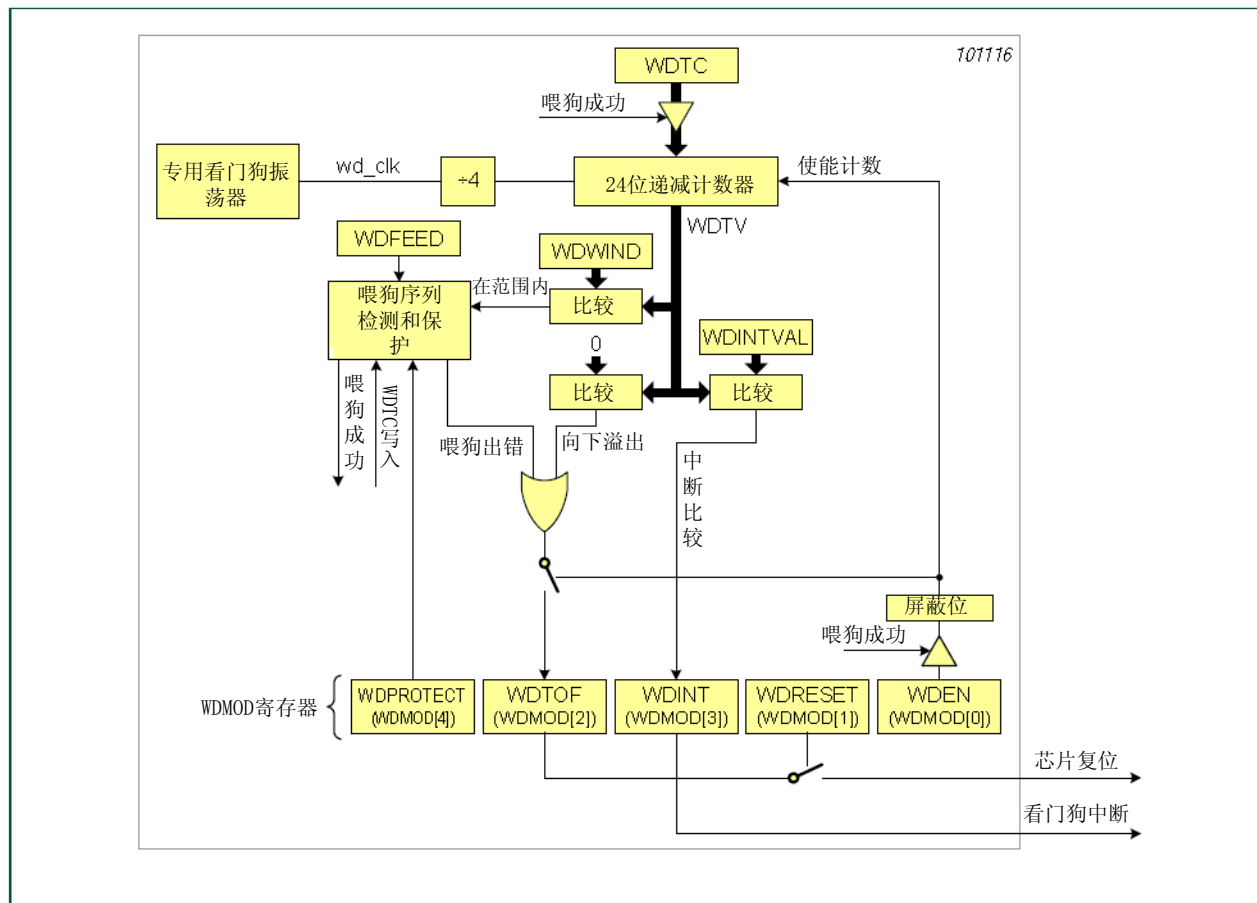
若看门狗定时器被设置为产生报警中断，那么当计数器中的值与 WDWARNINT 寄存器中所定义的值相匹配的时候，就会产生中断。

看门狗定时器模块使用了 2 个时钟：PCLK 和 WDCLK。PCLK 供 APB 访问看门狗寄存器使用。WDCLK 供看门狗定时器计数使用。这一时钟来自专用的振荡器，该振荡器在看门狗定时器被使能之后一直处于启动状态。该振荡器的典型频率为 500kHz（见 4.3.4 节）。

两个时钟域之间有同步逻辑。当 WDMOD 和 WDTCR 通过 APB 操作更新时，新的值在 3 个时钟周期后将会在 WDCLK 时钟域逻辑上生效。当看门狗定时器运行时，同步逻辑会先锁存运行频率为 WDCLK 的计数器值，然后将计数器与 PCLK 同步，再作为 WDTV 寄存器的值供 CPU 读取。

图 155 中所示为看门狗框图。这个框图中没有给出同步逻辑 (PCLK-WDCLK)。

图155.看门狗定时器框图



31.4 寄存器描述

看门狗包含 6 个寄存器，如[表 642](#)所示。

表642. 看门狗寄存器映射

名称	描述	访问	复位值 ^[1]	地址	表
WDMOD	看门狗模式寄存器。该寄存器决定看门狗定时器的基本模式和状态。	R/W	0	0x4000 0000	表 643
WDTC	看门狗定时器常数寄存器。该寄存器中的值决定超时周期（超时值）。	R/W	0xFF	0x4000 0004	表 645
WDFEED	看门狗喂狗寄存器。向该寄存器顺序写入 0xAA 和 0x55 使看门狗定时器重新装入 WDTC 的值。	WO	无	0x4000 0008	表 646
WDTV	看门狗定时器值寄存器。该寄存器读出看门狗定时器的当前值。	RO	0xFF	0x4000 000C	表 647
WDWARNINT	看门狗报警中断比较值。	R/W	0	0x4000 0014	表 648
WDWINDOW	看门狗窗口比较值。	R/W	0xFF FFFF	0x4000 0018	表 649

[1] 复位值仅反映已使用位中保存的数据，不包括保留位的内容。

31.4.1 看门狗模式寄存器（WDMOD—0x4000 0000）

WDMOD 寄存器通过 WDEN 位和 RESET 位的组合来控制看门狗的操作。需要注意的是，必须在任何 WDMOD 寄存器改变生效之前喂狗。

表643. 看门狗模式寄存器位描述（WDMOD—0x4000 0000）

位	符号	值	描述	复位值
0	WDEN		看门狗使能位。只能置位。	0
		0	看门狗定时器停止运行。	
		1	看门狗定时器运行。	
1	WDRESET		看门狗复位使能位。只能置位。	0
		0	看门狗超时不会引起芯片复位。	
		1	看门狗超时会引起芯片复位。	
2	WDTOF		看门狗超时标志。在看门狗定时器超时、喂狗错误或发生与 WDPROTECT 相关的事件时，该位由软件清零。WDRESET 为 1 时会引起芯片复位。	0
3	WDINT		看门狗中断标志。该位在定时器的值达到 WDWARNINT 的值时置位。由软件清零。	0
4	WDPROTECT		看门狗更新模式。只能置位。	0
		0	看门狗重载值（WDTC）可在任何时候变更。	
		1	看门狗重载值（WDTC）只有在计数器值低于 WDWARNINT 和 WDWINDOW 值的时候才可被改变。注：只有当 WDRESET=1 时该模式才适用。	
7:5	-		保留。读取值未定义，只写入 0。	无

一旦 **WDEN**、**WDPROTECT** 或 **WDRESET** 位置位，就无法使用软件将其清零。这些标志都由外部复位或看门狗定时器复位来清除。

WDTOF：若看门狗定时器溢出、喂狗出错或 **WDPROTECT** 位为 1 时，看门狗超时标志被置位，并试图写入 **WDTC** 寄存器。通过软件将该位写入 0，即可清除这个标志。

WDINT：在看门狗计数器达到 **WDWARNINT** 设置的值时，看门狗中断标志被置位。当有复位发生的时候，这个标志会被清除，也可以通过软件向该位写入 1 来清除这个标志。

在看门狗运行期间，随时可能会发生复位或中断。如果在睡眠或深度睡眠模式中出现看门狗中断，那么设备会被唤醒。

表644. 看门狗工作模式选择

WDEN	WDRESET	工作模式
0	X (0 or 1)	调试/操作模式，看门狗关闭时。
1	0	看门狗中断模式：看门狗报警中断使能但看门狗复位不使能。当这种模式被选择时，看门狗计数器值为 WDWARNINT 的特定值时将置位 WDINT 标志，并产生看门狗中断请求。
1	1	看门狗复位模式：看门狗中断和复位都使能。当这种模式被选择时，看门狗计数器值为 WDWARNINT 的特定值时将置位 WDINT 标志，并产生看门狗中断请求。此外，看门狗计数器值为 0 时将复位微控制器。在计数器值达到 WDWINDOW 设定值之前喂狗也会引起看门狗复位。

31.4.2 看门狗定时器常数寄存器（WDTC—base + -x04）

WDTC 寄存器决定看门狗定时器的超时值。每当喂狗序列产生时，**WDTC** 中的常量就会被重装入看门狗定时器。一旦发生复位，就会预先装入 0x00 00FF。若写入一个小于 0xFF 的值，系统会将 0x0000 00FF 装入 **WDTC** 寄存器中。因此最小超时间隔为 **TWDCLK**×256×4。

若 **WDMOD** 寄存器中 **WDPROTECT** 位为 1，那么如果在看门狗计数器小于 **WDWARNIN** 和 **WDWINDOW** 之前试图改变 **WDTC** 的值就会导致看门狗复位和 **WDTOF** 标志被置位。

表645. 看门狗定时器常数寄存器位描述（WDTC—0x4000 0004）

位	符号	描述	复位值
23:0	Count	看门狗超时间隔。	0x00 00FF
31:24	-	保留。读取值未定义，只写入 0。	无

31.4.3 看门狗喂狗寄存器（WDFEED—0x4000 0008）

向该寄存器写入 0xAA，然后写入 0x55 将会使看门狗定时器重新载入 **WDTC** 的值。如果通过 **WDMOD** 寄存器使能了看门狗，该操作还将启动看门狗运行。仅将 **WDMOD** 寄存器中的 **WDEN** 位置位还不足以使能看门狗。在设置 **WDEN** 之后，还必须完成一次有效的喂狗序列，看门狗才能复位。在真正使能之前，看门狗将忽略喂狗错误。在看门狗使能后，如果向 **WDFEED** 寄存器写入 0xAA 之后的下一个操作不是向 **WDFEED** 寄存器写入 0x55，而是访问任一看门狗寄存器，那么会立刻产生复位/中断并置位 **WDTOF** 标志。在喂狗序列中，在不正确访问看门狗寄存器之后的第二个 **PCLK** 周期将产生复位。

表646. 看门狗喂狗寄存器位描述（WDFEED—0x4000 0008）

位	符号	描述	复位值
7:0	Feed	喂狗值应当是 0xAA，然后是 0x55。	无

31.4.4 看门狗定时器值寄存器（WDTV—0x4000 000C）

WDTV 寄存器被用于读取看门狗定时器计数器的当前值。

当读取 24 位计数器的值时，锁定和同步过程要占用 6 个 WDCLK 时钟周期和 6 个 PCLK 时钟周期，因此，WDTV 的值比 CPU 正在读取到的定时器的实际值要长。

表647. 看门狗定时器值寄存器位描述（WDTV—0x4000 000C）

位	符号	描述	复位值
23:0	Count	计数器定时器值。	0x00 00FF
31:24	-	保留。读取值未定义，只写入 0。	无

31.4.5 看门狗定时器报警中断寄存器（WDWARNINT—0x4000 0014）

WDWARNINT 寄存器决定产生看门狗中断的看门狗定时器计数值。当看门狗定时器计数器的值与 WDWARNINT 所设定的值相匹配的时候，就会在接下来的 WDCLK 时钟周期之后产生中断。

当计数器的低 10 位与 WARNINT 的低 10 位值相等且计数器余下的高位全部为 0 的时候，我们认为看门狗定时器计数器的值与 WDWARNIN 所设定的值相匹配。这样，在看门狗事件发生之前有多达 1023 个看门狗定时器计数（4096 个看门狗时钟）可以用于形成中断。若 WARNINT 位为 0，则在发生看门狗事件的同时产生中断。

表648. 看门狗定时器报警中断寄存器位描述（WDWINDOW—0x4000 0014）

位	符号	描述	复位值
9:0	WARNINT	看门狗报警中断比较值。	0
31:10	-	保留。读取值未定义，只写入 0。	无

31.4.6 看门狗定时器窗口寄存器（WDWINDOW—0x4000 0018）

WDWINDOW 寄存器决定在喂狗过程中允许的最大 WDTV 值。如果在 WDTV 的值没有达到这个最大值之前已完成了一个有效的喂狗序列，那么将发生看门狗事件。

WDWINDOW 复位为这个最大可能的 WDTV 值，因此窗口不受此影响。

表649. 看门狗定时器窗口寄存器位描述（WDWINDOW—0x4000 0018）

位	符号	描述	复位值
23:0	WINDOW	看门狗窗口值。	0xFF FFFF
31:24	-	保留。读取值未定义，只写入 0。	无

31.5 看门狗序列示例

图 156 从多个方面解释了看门狗定时器的操作。

图156. 窗口模式使能状态下的早期喂狗

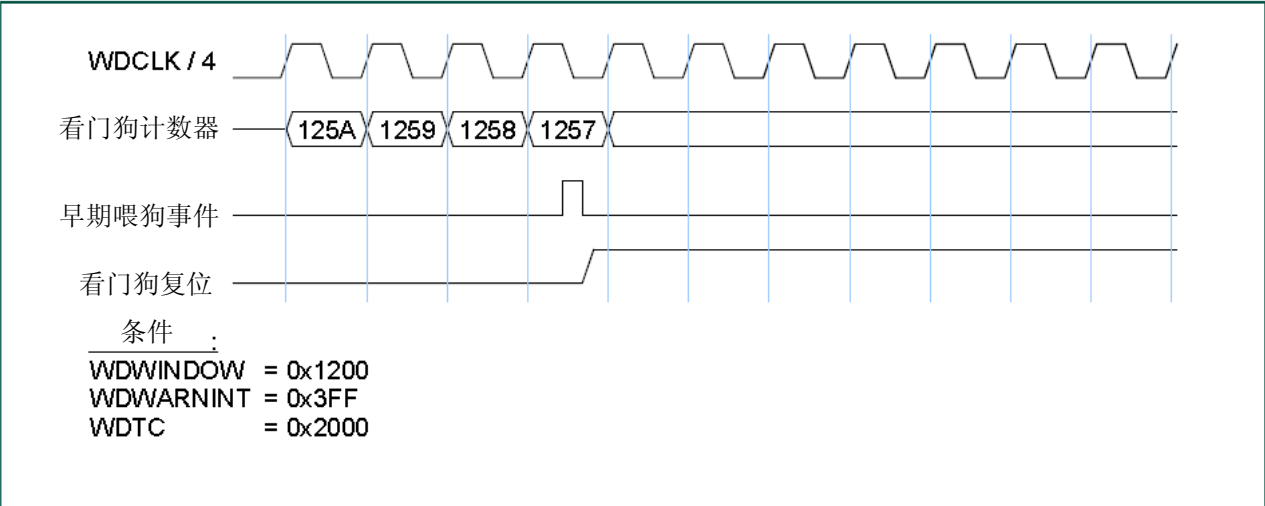


图157. 窗口模式使能状态下的正确喂狗

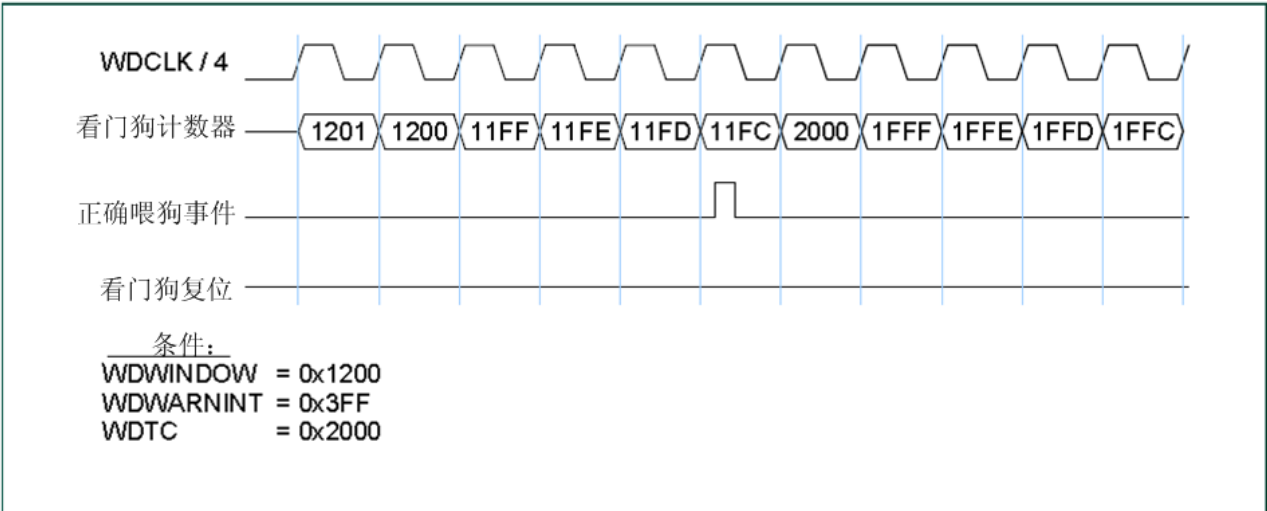
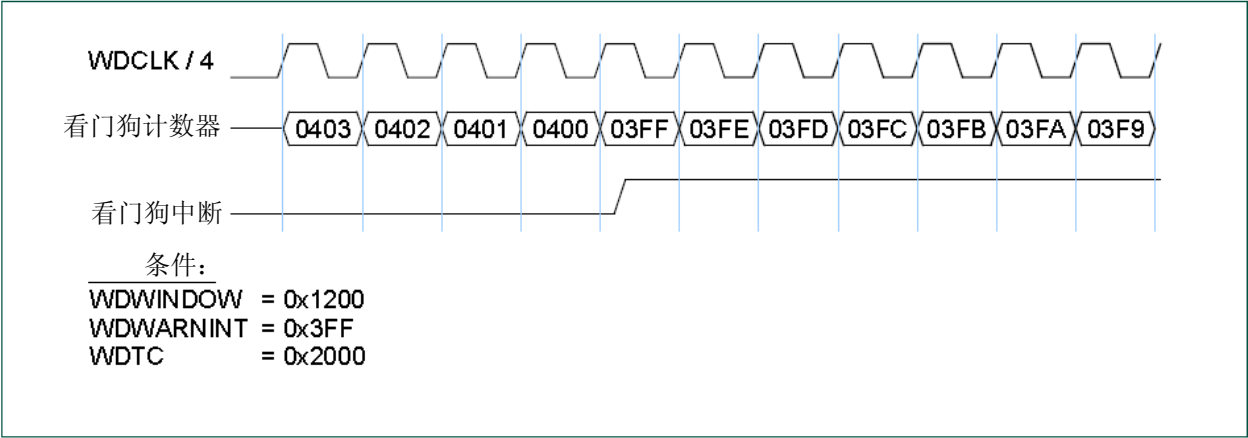


图158. 看门狗报警中断



32.1 基本配置

使用以下寄存器来配置 ADC：

1. 功率：在PCONP寄存器（见[表37](#)）中置位PCADC。
注：复位后，ADC被禁止。若要使能ADC，首先将位PCADC置位，然后使能AD0CR寄存器中的ADC（位PDN，见[表652](#)）。若要禁止ADC，必须先将位PDN清零，再将位PCADC清零。
2. 外设时钟：ADC使用公共PCLK，它既用于总线接口，也用于大多数APB外设的功能部分。参见[4.6.4](#)节。若要调节ADC时钟，见[表652](#)中的CLKDIV位。
3. 管脚：通过相关的IOCON寄存器（见[8.4.1](#)节）来使能ADC0管脚并为带ADC0功能的端口管脚选择管脚模式。
4. 中断：若要使能ADC中断，参见[表656](#)。通过适当的中断设置使能寄存器可以使能NVIC中的中断；也可以通过适当的中断设置使能寄存器来禁止NVIC中的ADC中断。
5. DMA：参见[32.6.4](#)节。有关GPDMA系统连接的详细内容，参见[表664](#)。

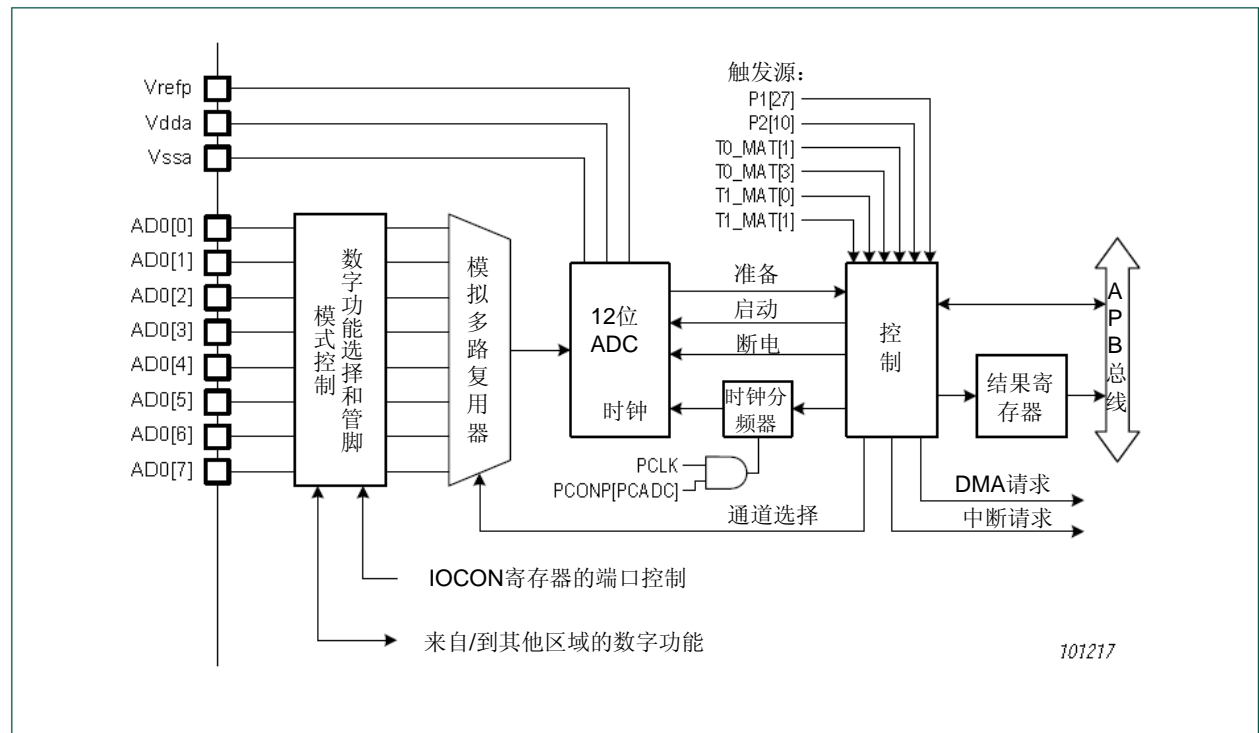
32.2 特性

- 12位逐次逼近式模数转换器。
- 8个管脚复用为输入脚。
- 具有掉电模式。
- 测量范围： V_{SS} 至 V_{REFP} （通常为3V；不超过 V_{DDA} ）。
- 12位转换速率，400kHz。
- 一个或多个输入的突发（Burst）转换模式。
- 可选择由输入管脚跳变或定时器匹配信号触发转换。

32.3 描述

A/D 转换器的基本时钟由 APB 时钟提供。每个转换器包含有一个可编程的分频器，用于将 APB 时钟调整为逐次逼近转换所需的时钟（最大可达 12.4MHz）。并且，完全满足精度要求的转换需要 31 个这样的时钟。

图159. ADC 框图



32.4 管脚描述

表 650 对 ADC 的每个相关管脚进行了简要总结。

表650. ADC 管脚描述

管脚	类型	描述
AD0[7]-AD0[0]	输入	模拟输入。 ADC 单元可以测量这些输入信号的电压。当管脚选择寄存器中该管脚上的 ADC 功能被选用时，电子信号与 ADC 输入管脚之间的连接断开。 警告： 如果使用 ADC，模拟输入管脚上的信号电平在任何时候都不能超过 V _{DDA} 。否则会导致 A/D 转换器的读数无效。如果在应用中未使用 A/D 转换器，那么与 A/D 输入相关的管脚就可用作最大可承受 5V 电压的数字 I/O 管脚。
V _{REFP}	参考	参考电压。 该管脚为 A/D 转换器和 D/A 转换器提供参考电压。 注： 如果不使用数模转换器和模数转换器，该管脚应与 V _{DD(3V3)} 相连。
V _{DDA} , V _{SSA}	电源	模拟电源和地。它们分别和 V _{DD} 和 V _{SS} 的电压相同。但为了降低噪声和出错几率，两者应当隔离。 注： 如果不使用数模转换器和模数转换器，V _{DDA} 应与 V _{DD(3V3)} 相连，V _{SSA} 应与 V _{SS} 相连。

32.5 寄存器描述

A/D 转换器包含的寄存器如[表 651](#)所示。

表651. ADC 寄存器

通用名称	描述	访问	复位值 ^[1]	D0 名称&地址	表
ADCR	A/D 控制寄存器。A/D 转换开始前，必须设置 ADCR 寄存器来选择工作模式。	R/W	1	AD0CR – 0x4003 4000	表 652
ADGDR	A/D 全局数据寄存器。该寄存器包含 ADC 的位 DONE 以及最近一次 A/D 转换的结果。	R/W	无	AD0GDR – 0x4003 4004	表 653
ADINTEN	A/D 中断使能寄存器。该寄存器包含的使能位控制每条 A/D 通道的 DONE 标志是否用来产生 A/D 中断。	R/W	0x100	AD0INTEN – 0x4003 400C	表 654
ADDR0	A/D 通道 0 数据寄存器。该寄存器包含在通道 0 上完成的最近一次转换的结果。	RO	无	AD0DR0 – 0x4003 4010	表 655
ADDR1	A/D 通道 1 数据寄存器。该寄存器包含在通道 1 上完成的最近一次转换的结果。	RO	无	AD0DR1 – 0x4003 4014	表 655
ADDR2	A/D 通道 2 数据寄存器。该寄存器包含在通道 2 上完成的最近一次转换的结果。	RO	无	AD0DR2 – 0x4003 4018	表 655
ADDR3	A/D 通道 3 数据寄存器。该寄存器包含在通道 3 上完成的最近一次转换的结果。	RO	无	AD0DR3 – 0x4003 401C	表 655
ADDR4	A/D 通道 4 数据寄存器。该寄存器包含在通道 4 上完成的最近一次转换的结果。	RO	无	AD0DR4 – 0x4003 4020	表 655
ADDR5	A/D 通道 5 数据寄存器。该寄存器包含在通道 5 上完成的最近一次转换的结果。	RO	无	AD0DR5 – 0x4003 4024	表 655
ADDR6	A/D 通道 6 数据寄存器。该寄存器包含在通道 6 上完成的最近一次转换的结果。	RO	无	AD0DR6 – 0x4003 4028	表 655
ADDR7	A/D 通道 7 数据寄存器。该寄存器包含在通道 7 上完成的最近一次转换的结果。	RO	无	AD0DR7 – 0x4003 402C	表 655
ADSTAT	A/D 状态寄存器。该寄存器包含所有 A/D 通道的 DONE 标志和 OVERRUN 标志，以及 A/D 中断/DMA 标志。	RO	0	AD0STAT – 0x4003 4030	表 656
ADTRM	ADC 调节（trim）寄存器。	R/W	0	AD0TRM - 0x4003 4034	表 657

[1] 复位值仅反映已使用位中保存的数据，不包括保留位的内容。

32.5.1 A/D 控制寄存器（AD0CR—0x4003 4000）

表652. A/D 控制寄存器位描述（AD0CR—地址 0x4003 4000）

位	符号	值	描述	复位值
7:0	SEL		从 AD0[7:0]中选择采样和转换的输入脚。对于 ADO,位 0 选择管脚 AD0[0], 位 7 选择管脚 AD0[7]。在软件控制模式下, 只有一位可被置位为 1。在硬件扫描模式下, 任何包含 1-8 的值都是可以写进该位的。全零等效于 0x01。	0x01
15:8	CLKDIV		将 APB 时钟（PCLK）进行分频（CLKDIV 值+1）得到 A/D 转换器的时钟。该时钟应小于或等于 12.4MHz。典型地, 软件将 CLKDIV 编程为最小值来得到 12.4MHz 或稍低于 12.4MHz 的时钟。但某些情况下（比如高阻抗模拟电源）可能需要一个更低的时钟。	0
16	BURST	1	A/D 转换器以高达 400kHz 的速率重复执行转换, 并且（如果必要）扫描 SEL 字段中为 1 的位选择的管脚。A/D 转换器启动后, 首先转换 SEL 字段中编号最低的为 1 的位的管脚, 如果适用, 接着变换稍高的为 1 的位的管脚。清零该位可停止重复转换, 但该位被清零时并不会中止正在进行的转换。 注： 当 BURST=1 时, 起始位必须是 000, 否则转换不会开始。	0
		0	转换由软件控制且需要 31 个时钟才能完成。	
20:17	-		保留。读取值未定义, 只写入 0。	无
21	PDN	1	A/D 转换器处于正常工作模式。	0
		0	A/D 转换器处于掉电模式。	
23:22	-		保留。读取值未定义, 只写入 0。	无
26:24	START		当 BURST=0 时, 这些位控制着 A/D 转换是否启动以及何时启动:	0
		000	不启动（当 PDN 清零时使用该值）。	
		001	立即启动转换。	
		010	当位 27 选择的边沿出现在 P2[10]管脚上时启动转换。	
		011	当位 27 选择的边沿出现在 P1[27]管脚上时启动转换。	
		100	当位 27 选择的边沿出现在 MAT0.1 上时启动转换。注意, MAT0.1 功能不一定要输出到器件管脚上。	
		1	当位 27 选择的边沿出现在 MAT0.3 上时启动转换。注意, MAT0.3	
		0	功能不一定要输出到器件管脚上。	
		1		
		110	当位 27 选择的边沿出现在 MAT1.0 上时启动转换。注意, MAT1.0 功能不一定要输出到器件管脚上。	
		111	当位 27 选择的边沿出现在 MAT1.1 上时启动转换。注意, MAT1.1 功能不一定要输出到器件管脚上。	
27	EDGE		该位只有在 START 字段为 010-111 时有效, 在这种情况下:	0
		1	在所选 CAP/MAT 信号的下降沿启动转换。	
		0	在所选 CAP/MAT 信号的上升沿启动转换。	
31:28	-		保留。读取值未定义, 只写入 0。	无

32.5.2 A/D 全局数据寄存器（AD0GDR—0x4003 4004）

A/D 全局数据寄存器包含最近一次已经完成的 A/D 转换的结果，还包含在此转换过程中出现的状态标志的附件。

有两种方法可以读取 ADC 的转换结果。一种方法就是利用 A/D 全局数据寄存器来读取 ADC 的全部数据，另一种方法是读取 A/D 通道数据寄存器。统一使用一种方法非常关键，否则 DONE 和 OVERRUN 标志在 AD0GDR 和 A/D 通道数据寄存器之间就不会同步，还可能会引起错误中断或 DMA 操作。

表653. A/D 全局数据寄存器位描述（AD0GDR—地址 0x4003 4004）

位	符号	描述	复位值
3:0	-	保留。读取值未定义，只写入 0。	无
15:4	RESULT	当 DONE 为 1 时，该字段为二进制形式，表示 SEL 字段在 AD0[n]管脚上选中的电压，电压可选范围在 V _{REFP} 和 V _{SS} 之间。 该字段为 0 表示输入管脚上的电压小于、等于或接近于 V _{SS} ；该字段为 0xFFFF 表示输入管脚上的电压接近于、等于或者大于 V _{REFP} 。	无
23:16	-	保留。读取值未定义，只写入 0。	无
26:24	CHN	这些位包含的是通道，RESULT 位通过该通道进行转换（例如，000 代表通道 0，001 代表通道 1，以此类推）。	无
29:27	-	保留。读取值未定义，只写入 0。	无
30	OVERRUN	突发（Burst）模式下，如果在产生 RESULT 位结果的转换前一个或多个转换结果丢失或覆盖，该位为 1。读取该寄存器可清零该位。	0
31	DONE	A/D 转换结束时该位设置为 1。该位在该寄存器被读取和 ADCR 被写入时清零。如果 ADCR 在转换过程中被写入，那么该位置位并启动一次新的转换。	0

32.5.3 A/D 中断使能寄存器（AD0INTEN—0x4003 400C）

该寄存器用来控制转换完成时哪个 A/D 通道产生中断。例如，当需要通过对 A/D 通道连续转换来监控传感器时，最近一次的转换结果可根据需要随时由应用程序读出。在这种情况下，A/D 通道转换结束时都不使用中断方式。

表654. A/D 中断使能寄存器位描述（AD0INTEN—地址 0x4003 400C）

位	符号	值	描述	复位值
0	ADINTEN0	0	ADC 通道 0 转换结束时不会产生中断。	0
		1	ADC 通道 0 转换结束时产生中断。	
1	ADINTEN1	0	ADC 通道 1 转换结束时不会产生中断。	0
		1	ADC 通道 1 转换结束时产生中断。	
2	ADINTEN2	0	ADC 通道 2 转换结束时不会产生中断。	0
		1	ADC 通道 2 转换结束时产生中断。	
3	ADINTEN3	0	ADC 通道 3 转换结束时不会产生中断。	0
		1	ADC 通道 3 转换结束时产生中断。	
4	ADINTEN4	0	ADC 通道 4 转换结束时不会产生中断。	0
		1	ADC 通道 4 转换结束时产生中断。	
5	ADINTEN5	0	ADC 通道 5 转换结束时不会产生中断。	0
		1	ADC 通道 5 转换结束时产生中断。	
6	ADINTEN6	0	ADC 通道 6 转换结束时不会产生中断。	0
		1	ADC 通道 6 转换结束时产生中断。	
7	ADINTEN7	0	ADC 通道 7 转换结束时不会产生中断。	0
		1	ADC 通道 7 转换结束时产生中断。	
8	ADGINTEN	0	只有个别由 ADINTEN7:0 使能的通道才产生中断。	1
		1	除个别 ADC 通道被使能可以产生中断外，使能 ADDR 的全局 DONE 标志也可产生中断。	
31:17	-		保留。读取值未定义，只写入 0。	无

32.5.4 A/D 数据寄存器（AD0DR0~AD0DR7—0x4003 4010~0x4003 402C）

在完成 A/D 转换后，A/D 数据寄存器保存着每个 A/D 通道最后一次转换的结果，同时包含指示转换结束以及转换溢出的标志。

有两种方法可以读取 ADC 的转换结果。一种方法是利用 A/D 全局数据寄存器来读取 ADC 的全部数据，另一种方法是读取 A/D 通道数据寄存器。统一使用一种方法非常关键，否则 DONE 和 OVERRUN 标志在 AD0GDR 和 A/D 通道数据寄存器之间就不会同步，还可能会引起错误中断或 DMA 操作。

表655. A/D 数据寄存器位描述（AD0DR0~AD0DR7—0x4003 4010~0x4003 402C）

位	符号	描述	复位值
3:0	-	保留。读取值未定义，只写入 0。	无
15:4	RESULT	当 DONE 为 1 时，该字段为二进制形式，表示 SEL 字段在 AD0[n]管脚上选中的电压，电压可选范围在 V _{REFP} 和 V _{SS} 之间。 该字段为 0 表示输入管脚上的电压小于、等于或接近于 V _{SS} ；该字段为 0xFFFF 表示输入管脚上的电压接近于、等于或者大于 V _{REFP} 。	无
29:16	-	保留。读取值未定义，只写入 0。	无
30	OVERRUN	突发（Burst）模式下，如果在产生 RESULT 位结果的转换前一个或多个转换结果丢失或覆盖，该位为 1。读取该寄存器可清零该位。	无
30	OVERRUN	突发（Burst）模式下，如果在产生 RESULT 位结果的转换前一个或多个转换结果丢失或覆盖，该位为 1。读取该寄存器可清零该位。	无

32.5.5 A/D 状态寄存器（ADSTAT—0x4003 4030）

A/D 状态寄存器允许同时检查所有 A/D 通道的状态。每个 A/D 通道的 ADDRn 寄存器的 DONE 和 OVERRUN 标志都反映在 ADSTAT 中。在 ADSTAT 中同样可以找到中断标记（所有 DONE 标志逻辑相或的结果）。

表656. A/D 状态寄存器位描述（AD0STAT—地址 0x4003 4030）

位	符号	描述	复位值
0	DONE0	该位反映了 A/D 通道 0 的结果寄存器中的 DONE 状态标志。	0
1	DONE1	该位反映了 A/D 通道 1 的结果寄存器中的 DONE 状态标志。	0
2	DONE2	该位反映了 A/D 通道 2 的结果寄存器中的 DONE 状态标志。	0
3	DONE3	该位反映了 A/D 通道 3 的结果寄存器中的 DONE 状态标志。	0
4	DONE4	该位反映了 A/D 通道 4 的结果寄存器中的 DONE 状态标志。	0
5	DONE5	该位反映了 A/D 通道 5 的结果寄存器中的 DONE 状态标志。	0
6	DONE6	该位反映了 A/D 通道 6 的结果寄存器中的 DONE 状态标志。	0
7	DONE7	该位反映了 A/D 通道 7 的结果寄存器中的 DONE 状态标志。	0
8	OVERRUN0	该位反映了 A/D 通道 0 的结果寄存器中的 OVERRRUN 状态标志。	0
9	OVERRUN1	该位反映了 A/D 通道 1 的结果寄存器中的 OVERRRUN 状态标志。	0
10	OVERRUN2	该位反映了 A/D 通道 2 的结果寄存器中的 OVERRRUN 状态标志。	0
11	OVERRUN3	该位反映了 A/D 通道 3 的结果寄存器中的 OVERRRUN 状态标志。	0
12	OVERRUN4	该位反映了 A/D 通道 4 的结果寄存器中的 OVERRRUN 状态标志。	0
13	OVERRUN5	该位反映了 A/D 通道 5 的结果寄存器中的 OVERRRUN 状态标志。	0
14	OVERRUN6	该位反映了 A/D 通道 6 的结果寄存器中的 OVERRRUN 状态标志。	0
15	OVERRUN7	该位反映了 A/D 通道 7 的结果寄存器中的 OVERRRUN 状态标志。	0
16	ADINT	该位为 A/D 中断标志。当使能 A/D 产生中断时（通过 ADINTEN 寄存器）任何一条 A/D 通道 Done 标志被设置，该位为 1。	0
31:17	-	保留。读取值未定义，只写入 0。	无

32.5.6 A/D 调节寄存器（ADTRIM—0x4003 4034）

启动时该寄存器通过引导代码来设置。它包含 DAC 和 ADC 转换的调整值。用户可修改 ADC 的偏移调节值。在读取该寄存器的时候，全部的 12 位都可见。

表657. A/D 调节寄存器位描述（ADTRM—地址 0x4003 4034）

位	符号	描述	复位值
3:0	-	保留。读取值未定义，只写入 0。	无
7:4	ADCOFFS	ADC 操作的偏移调节位。通过引导代码进行初始化。用户可修改。	0
11:8	TRIM	通过引导代码写入。用户不能修改这些位。引导代码写入后这些位被锁定。	0
31:12	-	保留。读取值未定义，只写入 0。	无

32.6 操作

ADC 转换一旦开始，就不能被中断。若前一个转换正在进行中，软件新写入就不能发起新的转换，新的边沿触发事件也会被忽略。

32.6.1 硬件触发的转换

如果 ADCR 中的 BURST 位为 0 且 START 字段的值包含在 010-111 之内，当所选管脚上或定时器匹配的信号发生跳变时，A/D 转换器启动一次转换。也可选择在 4 个匹配信号中任何一个的指定边沿转换，或者在 2 个捕获/匹配管脚中任何一个的指定边沿转换。将所选端口的管脚状态或所选的匹配信号与 ADCR 的位 27 异或（XOR）来作为边沿检测逻辑。

32.6.2 中断

当 DONE 标志位为 1 时，中断请求会被提交到 NVIC。软件通过 NVIC 中的 A/D 中断使能位来控制是否产生中断。在读取 ADDR 时，DONE 标志被否决。

32.6.3 精度和数字接收器

必须通过相关 IOCON 寄存器中的 ADMODE 位来选择 ADC 功能，从而获取监控管脚上的电压读数。此外，这个 IOCON 寄存器还应被设置为禁用上拉或下拉电阻的模式。对于 ADC 输入管脚，不需要选择数字功能也可以读取有效的 ADC 值。只要管脚上选定了数字功能，模拟输入就会被禁止。

32.6.4 DMA 控制

DMA 传输请求产生于 ADC 中断请求线。发起 DMA 传输的情况与产生中断的情况相同（参见 [32.6.2](#) 节和 [32.5.3](#) 节）。

注：如果使用 DMA，必须禁止 NVIC 中的 ADC 中断。

对于 DMA 传输，只支持突发传输请求。可将 DMA 通道控制寄存器中的突发（Burst）大小设为 1（参见 [34.5.20](#) 节）。若 ADC 通道个数不等于任意一个支持 DMA 的突发大小（可用的 DMA 突发大小有 1、4 和 8，见 [34.5.20](#) 节），突发大小被设为 1。

DMA 传输大小决定 DMA 中断产生的时间。传输大小可以设置成 ADC 通道的个数（参见 [34.5.20](#) 节）。不相邻的通道可通过 DMA 使用分散/聚集链表来传输（参见 [34.5.19](#) 节）。

33.1 基本配置

使用以下寄存器来配置 DAC：

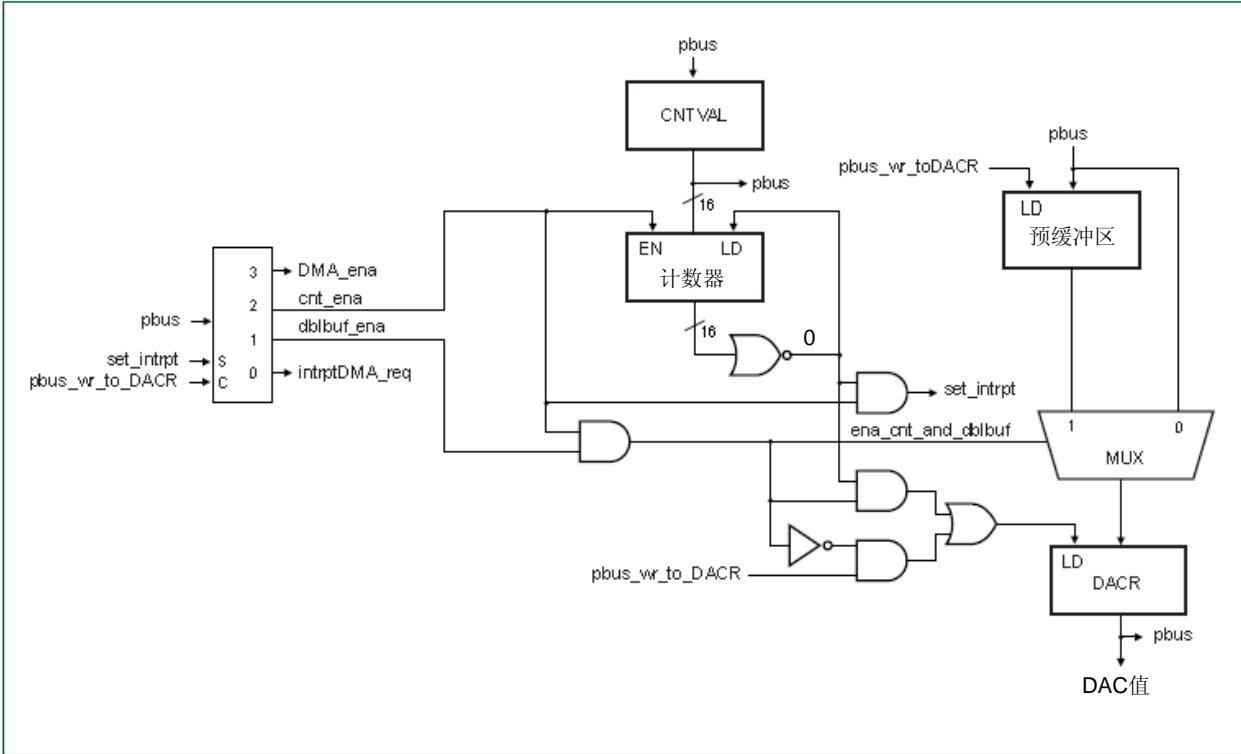
1. 功率：DAC 总是被连接到 V_{DDA} 。通过设置 IOCON 寄存器（参见下文）来决定是否访问寄存器。
2. 外设时钟：DAC 使用公共 PCLK，它既用于总线接口，也用于大多数 APB 外设的功能部分。参见 [4.6.4](#) 节。
3. 管脚：通过相关的 IOCON 寄存器（参见 [8.4.1](#) 节）来使能 DAC 管脚并选择 DACOUT 的管脚模式。该操作必须在访问 DAC 寄存器之前完成。
4. DMA：DAC 可以连接到 GPDMA 控制器（参见 [33.5.2](#) 节）。有关 GPDMA 的连接，参见 [表 664](#)。

33.2 特性

- 10 位数模转换器
- 电阻串结构
- 缓冲输出
- 具有掉电模式
- 可选速度和功耗
- 最大更新速率为 1MHz

33.3 结构

图160. 通过 DMA 中断和定时器实现 DAC 控制



33.4 管脚描述

表 658 对每个 DAC 相关的管脚进行了简要总结。

表658. D/A 管脚描述

管脚	类型	描述
DAC_OUT	输出	模拟输出。当 DACR 写入新的值后，经过所选的设定时间，该管脚的电压为 $VALUE \times (V_{REFP} - V_{REFN}) / 1024 + V_{REFN}$ （相对于 V_{SSA} ）。
V_{REFP}	参考	参考电压。该管脚为 ADC 和 DAC 提供参考电压。 注：如果不使用 ADC 和 DAC，该管脚应与 $V_{DD(3V3)}$ 相连。
V_{DDA} , V_{SSA}	电源	模拟电源和地。它们应当分别与 V_{DD} 和 V_{SS} 的电压相同，但为了降低噪声和出错几率，两者应当隔离。 注：如果不使用 ADC 和 DAC，V_{DDA} 应与 $V_{DD(3V3)}$ 相连，V_{SSA} 应与 V_{SS} 相连。

33.5 寄存器描述

DAC 寄存器如表 659 中示。需要注意的是 PCONP 中没有 DAC 控制位。若要使能 DAC，必须通过配置相关的 IOCON 寄存器(参见 8.4.1 节)来选择将其输出到相关的管脚 P0[26]。在访问 DAC 寄存器之前须用该方法将 DAC 使能。

表659. DAC 寄存器

名称	描述	访问	复位值 ^[1]	地址	表
DACR	D/A 转换寄存器。该寄存器包含转换成模拟值的数字设置值和功耗控制位。	R/W	0	0x4008 C000	表 660
DACCTRL	DAC 控制寄存器。该寄存器控制 DMA 和定时器的操作。	R/W	0	0x4008 C004	表 661
DACCNTVAL	DAC 计数器值寄存器。该寄存器包含 DAC DMA/中断定时器的重载值。	R/W	0	0x4008 C008	表 662

[1] 复位值仅反映已使用位中保存的数据，不包括保留位的内容。

33.5.1 D/A 转换寄存器（DACR—0x4008 C000）

这个读/写寄存器包含待转换成模拟值的数字设定值，以及用来平衡性能和功耗的位。位 5.0 保留用于之后分辨率更高的 D/A 转换器。

表660. D/A 转换器寄存器位描述（DACR—地址 0x4008 C000）

位	符号	值	描述	复位值
5:0	-		保留。读取值未定义，只写入 0。	无
15:6	VALUE		当该字段被写入新值后，经过所选的设定时间，管脚 DAC_OUT 上的电压为 VALUE x ((V _{REFP} -V _{REFN})/1024) + V _{REFN} （相对于 V _{SSA} ）。	0
16	BIAS ^[1]	0	DAC 的最大设定时间为 1μs，最大电流为 700μA。由此可支持的最高更新速度为 1MHz。	0
		1	DAC 的最大设定时间为 2.5μs，最大电流为 350μA。由此可支持的最高更新速度为 400kHz。	
31:17	-		保留。读取值未定义，只写入 0。	无

[1] 在 DAC_OUT 管脚上接有一个不超过 100pF 的电容的情况下，BIAS 位的描述中提到的设定时间才是有效的。如果负载阻抗值大于该值，将会导致设定时间比规定时间长。最终版本的数据手册将会给出有关负载阻抗和设定时间的图表。

33.5.2 D/A 转换器控制寄存器（DACCTRL—0x4008 C004）

该读/写寄存器用于使能 DMA 操作和控制 DMA 定时器。

表661. D/A 控制寄存器位描述（DACCTRL—地址 0x4008 C004）

位	符号	值	描述	复位值
0	INT_DMA_REQ	0	该位在写 DACR 寄存器时清零。	0
		1	定时器超时该位由硬件置位。	
1	DBLBUF_ENA	0	DACR 双缓冲被禁能。	0
		1	当该位和 CNT_ENA 都置位时，DACR 寄存器中的双缓冲功能被使能。向 DACR 寄存器写数据会先将数据写入一个预缓冲区，数据在下次计数器超时时被发送到 DACR。	
2	CNT_ENA	0	超时计数器操作被禁能。	0
		1	超时计数器操作被使能。	
3	DMA_ENA	0	DMA 访问被禁止。	0
		1	DMA 突发请求输入 7 被使能用于 DAC 转换（详见 表 664 ）。	
31:4	-		保留。读取值未定义，只写入 0。	无

33.5.3 D/A 转换器计数器值寄存器（DACCNTVAL—0x4008 C008）

该读/写寄存器包含了中断/DMA 计数器的重载值。

表662. D/A 转换器计数器值寄存器位描述（DACCNTVAL—地址 0x4008 C008）

位	符号	描述	复位值
15:0	VALUE	DAC 中断/DMA 定时器的 16 位重载值。	0

33.6 操作

33.6.1 DMA 计数器

当 DACCTRL 中的计数器使能位 CNT_ENA 被置位时，16 位计数器就从 DACCNTVAL 寄存器中已编程设定的值开始递减计数，递减速度由 PCLK（见[表 33](#)）选定。计数器每次递减至零之后都会重新加载 DACCNTVAL 的值，DMA 请求位 INT_DMA_REQ 也会通过硬件置位。

需要注意的是 DACCTRL 和 DACCNTVAL 中的内容都是可读/写的，但是定时器本身是不可读/写的。

若 DACCTRL 中的 DMA_ENA 位置位，DAC DMA 请求将提交给 GPDMA。当 DMA_ENA 位被清除，复位后，DAC DMA 请求被禁止（默认）。

33.6.2 双缓冲

仅当 DACCTRL 中的 CNT_ENA 位和 DBLBUF_ENA 位都被置位时双缓冲才会使能。双缓冲使能后，写入 DACR 寄存器的数据只是先装入预缓存区中，该预缓存区和 DACR 寄存器共享一个寄存器地址。当计数器达到 0 且 DMA 请求被设置时，预缓冲区中的数据就会被装入 DACR 寄存器中。同时，计数器再重新装入 COUNTVAL 寄存器的值。

读取 DACR 寄存器只会返回 DACR 中的内容，不包含预缓冲区中的内容。

若 CNT_ENA 位和 DBLBUF_ENA 位中有一位为 0，写入 DACR 地址的数据就会直接写到 DACR 中。

34.1 基本配置

使用以下寄存器来配置 GPDMA:

1. 功率: 在 PCONP 寄存器 ([表 37](#)) 中,对其 PCGPDMA 位进行设置。
注: 复位后, GPDMA被禁止 (PCGPDMA=0)。
2. 时钟: GPDMA 的运行速率为 AHB 总线速率,和 CPU 时钟速率 (CCLK) 是一样的。
3. 中断: 利用相应的中断设置使能寄存器使能 NVIC 中的中断。
4. 编程: 参见 [34.6](#)。

34.2 简介

DMA 控制器允许外设到存储器、存储器到外设和存储器到存储器之间的的传输。每个 DMA 流都可以为单个源和目标提供单向串行 DMA 传输。例如, 一个双向端口就需要一个专门的发送流和一个专门的接收流。源和目标可以是一个存储区或外设。

34.3 特性

- 8 路 DMA 通道。每个通道可支持一路单向传输。
- 提供 16 根 DMA 请求线。
- 支持存储器到存储器、存储器到外设和外设到存储器的传输。
- GPDMA 支持的外设: SD 卡接口、SSP、I²S、UART、A/D 转换器和 D/A 转换器。DMA 还可以通过选定的定时器匹配条件来触发。此外,GPDMA 还支持存储器到存储器的传输和源自或目标是 GPIO 的传输。
- 通过使用链表来支持分散/聚集的 DMA, 这就意味着源区和目标区不一定要占用连续的存储区。
- 硬件 DMA 通道优先级。
- AHB 从机 DMA 编程接口。可以通过 AHB 从机接口对 DMA 控制寄存器进行写入操作, 从而实现对 DMA 控制器的编程操作。
- 具有一个用于传输数据的 AHB 总线主机。这个接口在 DMA 请求有效时传输数据。
- 32 位的 AHB 主机总线宽度。
- 源和目标区可设置为递增寻址或非递增寻址。
- DMA 突发 (burst) 大小可编程。编程 DMA 突发大小可以提高传输数据的效率。
- 每个通道内部包含有一个 4 字大小的 FIFO。

- 支持 8、16 和 32 位宽的传输。
- 支持大端和小端模式。复位后，DMA 控制器默认为小端模式。
- 在 DMA 操作完成或出现 DMA 错误时，可以向处理器发出中断请求信号。
- 原始中断状态。在中断被屏蔽之前，都可以读出 DMA 错误和 DMA 计数的原始中断状态。
- DMA 支持在睡眠模式下运行。（值得注意的是,在睡眠模式下,GPDMA 无法访问 Flash 存储器或主 SRAM）。

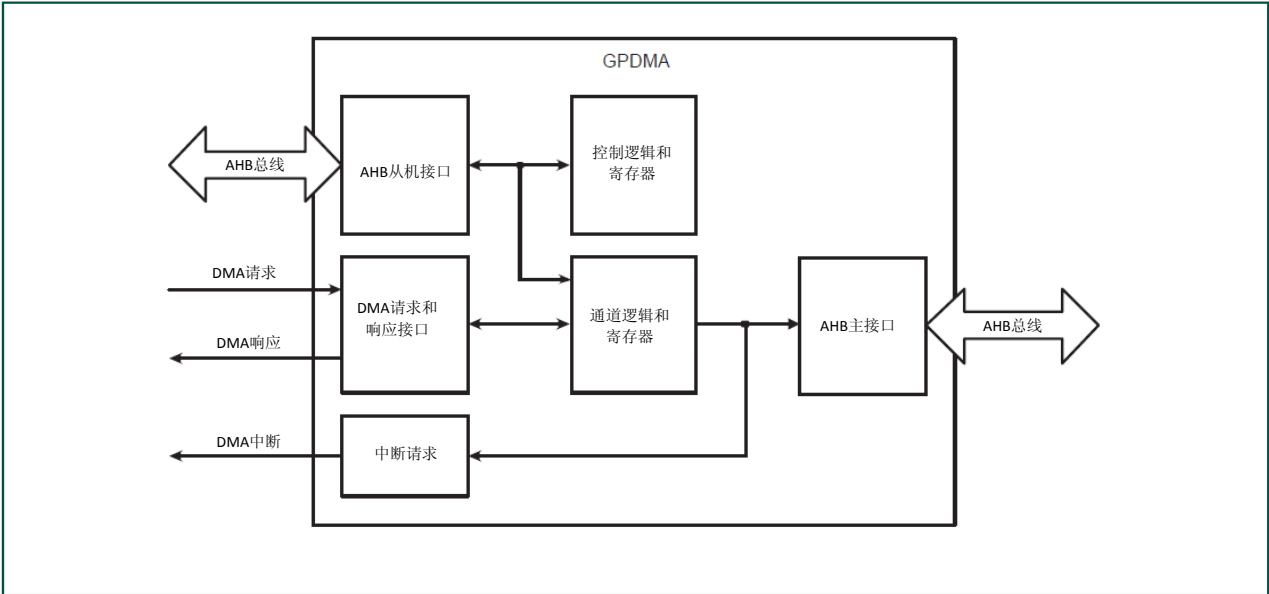
34.4 功能描述

本小节将介绍 DMA 控制器的几个主要的功能模块。

34.4.1 DMA 控制器的功能描述

DMA 控制器使能外设到存储器、存储器到外设、外设到外设和存储器到存储器的传输。每个 DMA 流都可以为单个源和目标提供单向串行 DMA 传输。例如，一个双向端口就需要一个专门的发送流和一个专门的接收流。源和目标可以是一个存储区或外设，并且可以通过 AHB 主机进行访问。图 161 所示为 DAM 控制器的框图。

图161.DMA 控制器框图



我们将在接下来的小节中对 DMA 控制器的功能进行介绍。

34.4.1.1 AHB 从机接口

通过 AHB 从机接口对 DMA 控制器的所有传输都是 32 位宽度。不支持 8 位和 16 位访问，因为这会导致异常出现。

34.4.1.2 控制逻辑和寄存器组

寄存器模块存储通过 AHB 接口写入或读出的数据。

34.4.1.3 DMA 请求和响应接口

有关 DMA 请求和响应接口的信息,请参见 [34.4.2](#) 节。

34.4.1.4 通道逻辑和通道寄存器组

通道逻辑和通道寄存器组包含了每个 DMA 通道所需的寄存器和逻辑。

34.4.1.5 中断请求

中断请求向 ARM 处理器产生中断。

34.4.1.6 AHB 主机接口

DMA 控制器包含 1 个 AHB 主机接口。AHB 主机能够处理下列类型的 AHB 传输:

- 分离、重试和从机的错误响应。如果一个外设执行一次分离或重试传输, DMA 控制器停止并等待传输结束。
- 锁定每个流的源和目标传输。
- 为每个流的传输设置保护位。

34.4.1.6.1 总线和传输宽度

AHB 总线的物理宽度是 32 位。源和目标的传输宽度可以不同,也可以相同,也可以比物理总线宽度更窄。DMA 控制器会根据需要对数据进行打包或拆分。

34.4.1.6.2 字节顺序 (Endian) 特性

DMA 控制器可处理小端和大端寻址。

在内部, DMA 控制器将所有数据当作一个字节流来处理,而不是 16 位或 32 位的数据量。这就意味着在执行源和目标区的传输字节顺序不同的混合字节时,可以观察到 32 位数据总线内的字节交换。

注: 如果不需要字节交换, 请避免在源和目标地址之间使用不同的字节顺序。[表 663](#) 所示为不同的源和目标组合的字节顺序特性。

表663. 字节顺序特性

源字节顺序	目标字节顺序	源宽度	目标宽度	源传输编号 / 源数据 字节通道		目标传输编号 / 字节通道	目标数据
小端	小端	8	8	1/[7:0]	21	1/[7:0]	21212121
				2/[15:8]	43	2/[15:8]	43434343
				3/[23:16]	65	3/[23:16]	65656565
				4/[31:24]	87	4/[31:24]	87878787
小端	小端	8	16	1/[7:0]	21	1/[15:0]	43214321
				2/[15:8]	43	2/[31:16]	87658765
				3/[23:16]	65		
				4/[31:24]	87		
小端	小端	8	32	1/[7:0]	21	1/[31:0]	87654321
				2/[15:8]	43		
				3/[23:16]	65		
				4/[31:24]	87		
小端	小端	16	8	1/[7:0]	21	1/[7:0]	21212121
				1/[15:8]	43	2/[15:8]	43434343
				2/[23:16]	65	3/[23:16]	65656565
				2/[31:24]	87	4/[31:24]	87878787
小端	小端	16	16	1/[7:0]	21	1/[15:0]	43214321
				1/[15:8]	43	2/[31:16]	87658765
				2/[23:16]	65		
				2/[31:24]	87		
小端	小端	16	32	1/[7:0]	21	1/[31:0]	87654321
				1/[15:8]	43		
				2/[23:16]	65		
				2/[31:24]	87		
小端	小端	32	8	1/[7:0]	21	1/[7:0]	21212121
				1/[15:8]	43	2/[15:8]	43434343
				1/[23:16]	65	3/[23:16]	65656565
				1/[31:24]	87	4/[31:24]	87878787
小端	小端	32	16	1/[7:0]	21	1/[15:0]	43214321
				1/[15:8]	43	2/[31:16]	87658765
				1/[23:16]	65		
				1/[31:24]	87		
小端	小端	32	32	1/[7:0]	21	1/[31:0]	87654321
				1/[15:8]	43		
				1/[23:16]	65		
				1/[31:24]	87		
大端	大端	8	8	1/[31:24]	12	1/[31:24]	12121212
				2/[23:16]	34	2/[23:16]	34343434
				3/[15:8]	56	3/[15:8]	56565656
				4/[7:0]	78	4/[7:0]	78787878

源字节顺序	目标字节顺序	源宽度	目标宽度	源 传 输 编 号 / 源数据 字 节 通 道		目标传输编号 / 字节通道	目标数据
大端	大端	8	16	1/[31:24]	12	1/[15:0]	12341234
				2/[23:16]	34	2/[31:16]	56785678
				3/[15:8]	56		
				4/[7:0]	78		
大端	大端	8	32	1/[31:24]	12	1/[31:0]	12345678
				2/[23:16]	34		
				3/[15:8]	56		
				4/[7:0]	78		
大端	大端	16	8	1/[31:24]	12	1/[31:24]	12121212
				1/[23:16]	34	2/[23:16]	34343434
				2/[15:8]	56	3/[15:8]	56565656
				2/[7:0]	78	4/[7:0]	78787878
大端	大端	16	16	1/[31:24]	12	1/[15:0]	12341234
				1/[23:16]	34	2/[31:16]	56785678
				2/[15:8]	56		
				2/[7:0]	78		
大端	大端	16	32	1/[31:24]	12	1/[31:0]	12345678
				1/[23:16]	34		
				2/[15:8]	56		
				2/[7:0]	78		
大端	大端	32	8	1/[31:24]	12	1/[31:24]	12121212
				1/[23:16]	34	2/[23:16]	34343434
				1/[15:8]	56	3/[15:8]	56565656
				1/[7:0]	78	4/[7:0]	78787878
大端	大端	32	16	1/[31:24]	12	1/[15:0]	12341234
				1/[23:16]	34	2/[31:16]	56785678
				1/[15:8]	56		
				1/[7:0]	78		
大端	大端	32	32	1/[31:24]	12	1/[31:0]	12345678
				1/[23:16]	34		
				1/[15:8]	56		
				1/[7:0]	78		

34.4.1.6.3 错误条件

DMA 传输过程中的错误标志是由外设标记的。在传输过程中，外设会在 AHB 总线上产生一个错误响应,并直接将错误标志标记出来。在当前的传输结束后，DMA 控制器会自动禁止 DMA 流，并且可以选择产生一个错误中断发送给 CPU。这个错误中断可以被屏蔽。

34.4.1.7 通道硬件

每个流都有专门的硬件通道,包括独立的源和目标控制器、FIFO。这就比只带有一个硬件信道(并且该信道还被其它几个 DMA 流共享)的 DMA 控制器具有更快的响应速度,而且简化了控制逻辑。

34.4.1.8 DMA 请求优先级

DMA 通道的优先级固定。DMA 通道 0 的优先级最高, DMA 通道 7 的优先级最低。

若在 DMA 控制器传送优先级较低的通道的数据时,有优先级较高的通道变得有效,则 DMA 控制器会先传输完优先级较低的通道交付给主机接口的数据,然后再传输优先级较高的通道的数据。交付给主机接口的传输数据放在 DMA 通道 FIFO 中,因此,需要传输的数据量最大可能达到 4 个字长。

建议: 存储器到存储器的传输使用优先级最低的通道。

34.4.1.9 中断的产生

DMA 将个别中断请求相"或"后得到一个复合中断输出,再连接到中断控制器。

34.4.2 DMA 系统连接

34.4.2.1 DMA 请求信号

外设利用 DMA 请求信号来请求数据传输。DMA 请求信号指示需要的是一个单次数据传输还是突发数据传输。可用的 DMA 请求信号有:

DMACBREQ[15:0]—突发请求信号。这些信号使能已编程的突发长度的数据的传输。

DMACSREQ[15:0]—单次传输请求信号。这些信号使能一个单次数据传输。DMA 控制器实现与外设之间的单次传输。

DMACLBREQ[15:0]—最后一个突发请求信号。

DMACLSREQ[15:0]—最后一个单次传输请求信号。

需要注意的是,该器件的大多数外设都不支持“最后”类型的请求,并且很多外设不支持单次请求和突发请求,详见 [34.4.2.3](#) 节。

34.4.2.2 DMA 响应信号

DMA 响应信号指示 DMA 请求信号启动的传输是否已经结束。响应信号也可以用来指示一个完整的数据包是否已经完成传输。DMA 控制器的响应信号有:

DMACCLR[15:0]—DMA 清除或应答信号。DMA 控制器利用 DMACCLR 信号来响应外设的 DMA 请求。

DMACTC[15:0]—DMA 终端计数信号。DMA 控制器使用 DMACTC 信号告知外设 DMA 传输已经结束。

34.4.2.3 DMA 请求连接

GPDMA 与支持外设的连接取决于这些外设的 DMA 功能。[表 664](#)所示为支持 DMA 的外设所使用的 DMA 请求编号。可通过 DMAReqSel 寄存器来选择通道 0~7 和 10~15 的请求。参见 [34.5.15](#) 节。

表664. DMA 连接

DMA 请求	突发请求	单次请求	最后一个突发请求	最后一个单次请求	注释
0	(unused) / T0_MAT[0]	-	-	-	专用 DMA 请求
1	SD card / T0_MAT[1]	仅限于 SD 卡	仅限于 SD 卡	仅限于 SD 卡	专用 DMA 请求
2	SSP0 Tx / T1_MAT[0]	仅限于 SSP0 Tx	-	-	专用 DMA 请求
3	SSP0 Rx / T1_MAT[1]	仅限于 SSP0 Rx	-	-	专用 DMA 请求
4	SSP1 Tx / T2_MAT[0]	仅限于 SSP1 Tx	-	-	专用 DMA 请求
5	SSP1 Rx / T2_MAT[1]	仅限于 SSP1 Rx	-	-	专用 DMA 请求
6	SSP2 TX / I ² S channel 0	仅限于 SSP2 Tx	-	-	专用 DMA 请求
7	SSP2 Rx / I ² S channel 1	仅限于 SSP2 Rx	-	-	专用 DMA 请求
8	ADC	-	-	-	ADC 中断请求 ^[1]
9	DAC	-	-	-	专用 DMA 请求
10	UART0 Tx / UART3 Tx	-	-	-	专用 DMA 请求
11	UART0 Rx / UART3 Rx	-	-	-	专用 DMA 请求
12	UART1 Tx / UART4 Tx	-	-	-	专用 DMA 请求
13	UART1 Rx / UART4 Rx	-	-	-	专用 DMA 请求
14	UART2 Tx / T3_MAT[0]	-	-	-	专用 DMA 请求
15	UART2 Rx / T3_MAT[1]	-	-	-	专用 DMA 请求

[1] 产生中断和/或 DMA 请求，取决于软件设置。

34.5 寄存器描述

DMA 控制器支持 8 个通道。每个信道都专门对应该通道操作的寄存器。其它的寄存器控制源外设与 DMA 控制器的关联。还有全局 DMA 控制寄存器和状态寄存器。

DMA 控制器的寄存器如[表 665](#)所示。

表665. GPDMA 寄存器映射

名称	描述	访问	复位值	地址	表
通用寄存器					
DMACIntStat	DMA 中断状态寄存器	RO	0	0x2008 0000	表 666
DMACIntTCStat	DMA 中断终端计数请求状态寄存器	RO	0	0x2008 0004	表 667
DMACIntTCClear	DMA 中断终端计数请求清除寄存器	WO	-	0x2008 0008	表 668
DMACIntErrStat	DMA 中断错误状态寄存器	RO	0	0x2008 000C	表 669
DMACIntErrClr	DMA 中断错误清除寄存器	WO	-	0x2008 0010	表 670
DMACRawIntTCStat	DMA 原始中断终端计数状态寄存器	RO	0	0x2008 0014	表 671
DMACRawIntErrStat	DMA 原始错误中断状态寄存器	RO	0	0x2008 0018	表 672
DMACEnbldChns	DMA 使能通道寄存器	RO	0	0x2008 001C	表 673
DMACSoftBReq	DMA 软件突发请求寄存器	R/W	0	0x2008 0020	表 674
DMACSoftSReq	DMA 软件单次请求寄存器	R/W	0	0x2008 0024	表 675
DMACSoftLBReq	DMA 软件最后一个突发请求寄存器	R/W	0	0x2008 0028	表 676
DMACSoftLSReq	DMA 软件最后一个单次请求寄存器	R/W	0	0x2008 002C	表 677
DMACConfig	DMA 配置寄存器	R/W	0	0x2008 0030	表 678
DMACSync	DMA 同步寄存器	R/W	0	0x2008 0034	表 679
DMACReqSel	选择通道 0-7 和 10-15 的请求。	R/W	0	0x400F C1C4	表 680
通道 0 寄存器					
DMACC0SrcAddr	DMA 通道 0 源地址寄存器	R/W	0	0x2008 0100	表 681
DMACC0DestAddr	DMA 通道 0 目标地址寄存器	R/W	0	0x2008 0104	表 682
DMACC0LLI	DMA 通道 0 链表项寄存器	R/W	0	0x2008 0108	表 683
DMACC0Control	DMA 通道 0 控制寄存器	R/W	0	0x2008 010C	表 684
DMACC0Config	DMA 通道 0 配置寄存器	R/W	0 ^[1]	0x2008 0110	表 685
通道 1 寄存器					
DMACC1SrcAddr	DMA 通道 1 源地址寄存器	R/W	0	0x2008 0120	表 681
DMACC1DestAddr	DMA 通道 1 目标地址寄存器	R/W	0	0x2008 0124	表 682
DMACC1LLI	DMA 通道 1 链表项寄存器	R/W	0	0x2008 0128	表 683
DMACC1Control	DMA 通道 1 控制寄存器	R/W	0	0x2008 012C	表 684
DMACC1Config	DMA 通道 1 配置寄存器	R/W	0 ^[1]	0x2008 0130	表 685
通道 2 寄存器					
DMACC2SrcAddr	DMA 通道 2 源地址寄存器	R/W	0	0x2008 0140	表 681
DMACC2DestAddr	DMA 通道 2 目标地址寄存器	R/W	0	0x2008 0144	表 682
DMACC2LLI	DMA 通道 2 链表项寄存器	R/W	0	0x2008 0148	表 683
DMACC2Control	DMA 通道 2 控制寄存器	R/W	0	0x2008 014C	表 684

名称	描述	访问	复位值	地址	表
DMACC2Config	DMA 通道 2 配置寄存器	R/W	0 ^[1]	0x2008 0150	表 685
通道 3 寄存器					
DMACC3SrcAddr	DMA 通道 3 源地址寄存器	R/W	0	0x2008 0160	表 681
DMACC3DestAddr	DMA 通道 3 目标地址寄存器	R/W	0	0x2008 0164	表 682
DMACC3LLI	DMA 通道 3 链表项寄存器	R/W	0	0x2008 0168	表 683
DMACC3Control	DMA 通道 3 控制寄存器	R/W	0	0x2008 016C	表 684
DMACC3Config	DMA 通道 3 配置寄存器	R/W	0 ^[1]	0x2008 0170	表 685
通道 4 寄存器					
DMACC4SrcAddr	DMA 通道 4 源地址寄存器	R/W	0	0x2008 0180	表 681
DMACC4DestAddr	DMA 通道 4 目标地址寄存器	R/W	0	0x2008 0184	表 682
DMACC4LLI	DMA 通道 4 链表项寄存器	R/W	0	0x2008 0188	表 683
DMACC4Control	DMA 通道 4 控制寄存器	R/W	0	0x2008 018C	表 684
DMACC4Config	DMA 通道 4 配置寄存器	R/W	0 ^[1]	0x2008 0190	表 685
通道 5 寄存器					
DMACC5SrcAddr	DMA 通道 5 源地址寄存器	R/W	0	0x2008 01A0	表 681
DMACC5DestAddr	DMA 通道 5 目标地址寄存器	R/W	0	0x2008 01A4	表 682
DMACC5LLI	DMA 通道 5 链表项寄存器	R/W	0	0x2008 01A8	表 683
DMACC5Control	DMA 通道 5 控制寄存器	R/W	0	0x2008 01AC	表 684
DMACC5Config	DMA 通道 5 配置寄存器	R/W	0 ^[1]	0x2008 01B0	表 685
通道 6 寄存器					
DMACC6SrcAddr	DMA 通道 6 源地址寄存器	R/W	0	0x2008 01C0	表 681
DMACC6DestAddr	DMA 通道 6 目标地址寄存器	R/W	0	0x2008 01C4	表 682
DMACC6LLI	DMA 通道 6 链表项寄存器	R/W	0	0x2008 01C8	表 683
DMACC6Control	DMA 通道 6 控制寄存器	R/W	0	0x2008 01CC	表 684
DMACC6Config	DMA 通道 6 配置寄存器	R/W	0 ^[1]	0x2008 01D0	表 685
通道 7 寄存器					
DMACC7SrcAddr	DMA 通道 7 源地址寄存器	R/W	0	0x2008 01E0	表 681
DMACC7DestAddr	DMA 通道 7 目标地址寄存器	R/W	0	0x2008 01E4	表 682
DMACC7LLI	DMA 通道 7 链表项寄存器	R/W	0	0x2008 01E8	表 683
DMACC7Control	DMA 通道 7 控制寄存器	R/W	0	0x2008 01EC	表 684
DMACC7Config	DMA 通道 7 配置寄存器	R/W	0 ^[1]	0x2008 01F0	表 685

[1] 该寄存器的 bit17 是一个只读状态标志位。

34.5.1 DMA 中断状态寄存器（DMACIntStat—0x2008 0000）

DMACIntStat 是一个只读寄存器，它显示屏蔽后中断的状态。值为 1 的位表明某个特定的 DMA 通道中断请求有效。中断请求可以由错误产生，也可以由终端计数中断产生。[表 666](#) 所示为 DMACIntStat 寄存器的位分配。

表666. DMA 中断状态寄存器（DMACIntStat—0x2008 0000）

位	名称	功能
7:0	IntStat	屏蔽后 DMA 通道中断的状态。每一位代表一条通道： 0—相应的通道没有有效的中断请求。 1—相应的通道有一个有效的中断请求。
31:8	-	保留。从保留位读出的值未定义。

34.5.2 DMA 中断终端计数请求状态寄存器（DMACIntTCStat—0x2008 0004）

DMACIntTCStat 是一个只读寄存器，指示了屏蔽后终端计数的状态。[表 667](#) 所示为 DMACIntTCStat 寄存器的位分配。

表667. DMA 中断终端计数请求状态寄存器（DMACIntTCStat—0x2008 0004）

位	名称	功能
7:0	IntTCStat	DMA 通道终端计数中断请求的状态。每一位代表一条通道： 0—相应的通道没有有效的终端计数中断请求。 1—相应的通道有一个有效的中断请求。
31:8	-	保留。从保留位读出的值未定义。

34.5.3 DMA 中断终端计数请求清除寄存器（DMACIntTCClear—0x2008 0008）

DMACIntTCClear 寄存器是一个只写寄存器，它清除一个或多个终端计数中断请求。向该寄存器中的某一位写入 1 会清除对应的状态寄存器（DMACIntTCStat）中的相应位。写入 0 无效。[表 668](#) 所示为 DMACIntTCClear 寄存器的位分配。

表668. DMA 中断终端计数请求清除寄存器（DMACIntTCClear—0x2008 0008）

位	名称	功能
7:0	IntTCClear	允许清除 DMA 通道的终端计数中断请求（IntTCStat）。每一位代表一条通道： 0—写入 0 无效。 1—清除相应的通道终端计数中断。
31:8	-	保留。读取值未定义，只写入 0。

34.5.4 DMA 中断错误状态寄存器（DMACIntErrStat—0x2008 000C）

DMACIntErrStat 是一个只读寄存器，它指示了屏蔽后的错误请求的状态。[表 669](#) 所示为 DMACIntErrStat 寄存器的位分配。

表669. DMA 中断错误状态寄存器（DMACIntErrStat—0x2008 000C）

位	名称	功能
7:0	IntErrStat	DMA 通道中断错误状态。每一位代表一条通道： 0—相应的通道没有有效的错误中断请求。 1—相应的通道有一个有效的错误中断请求。
31:8	-	保留。从保留位读出的值未定义。

34.5.5 DMA 中断错误清除寄存器（DMACIntErrClr—0x2008 0010）

DMACIntErrClr 寄存器是一个只写寄存器，它清除错误中断请求。向该寄存器的某一位写入 1 会清除状态寄存器的相应位。写入 0 无效。[表 670](#) 所示为 DMACIntErrClr 寄存器的位分配。

表670. DMA 中断错误清除寄存器（DMACIntErrClr—0x2008 0010）

位	名称	功能
7:0	IntErrClr	写入 1 来清除 DMA 通道的错误中断请求（IntErrStat）。每一位代表一条通道： 0—写入 0 无效。 1—清除相应的通道错误中断。
31:8	-	保留。读取值未定义，只写入 0。

34.5.6 DMA 原始中断终端计数状态寄存器（DMACRawIntTCStat—0x2008 0014）

DMACRawIntTCStat 是一个只读寄存器，它指示在屏蔽之前哪个 DMA 通道正在请求完成传输（终端计数中断）。（注：DMACIntTCStat 寄存器在屏蔽之后仍包含相同的信息。）值为 1 的位表明屏蔽前终端计数中断请求有效。[表 671](#) 所示为 DMACRawIntTCStat 寄存器的位分配。

表671. DMA 原始中断终端计数状态寄存器（DMACRawIntTCStat—0x2008 0014）

位	名称	功能
7:0	RawIntTCStat	屏蔽前 DMA 通道终端计数中断的状态。每一位代表一条通道： 0—相应的通道没有有效的终端计数中断请求。 1—相应的通道有一个有效的终端计数中断请求。
31:8	-	保留。从保留位读出的值未定义。

34.5.7 DMA 原始错误中断状态寄存器（DMACRawIntErrStat -0x2008 0018）

DMACRawIntErrStat 是一个只读寄存器，它指示屏蔽前哪个 DMA 通道正在请求一个错误中断。（注：屏蔽后 DMACIntErrStat 寄存器仍包含相同的信息。）值为 1 的位表明屏蔽前错误中断请求有效。[表 672](#)所示为 DMACRawIntErrStat 寄存器的位分配。

表672. DMA 原始错误中断状态寄存器（DMACRawIntErrStat—0x2008 0018）

位	名称	功能
7:0	RawIntErrStat	屏蔽前 DMA 通道错误中断的状态。每一位代表一条通道： 0—相应的通道没有有效的错误中断请求。 1—相应的通道有一个有效的错误中断请求。
31:8	-	保留。从保留位读出的值未定义。

34.5.8 DMA 使能通道寄存器（DMACEnbldChns—0x2008 001C）

DMACEnbldChns 是一个只读寄存器，它指示了哪个 DMA 通道被使能（与 DMACCxConfig 寄存器中的使能位所指示的一样。）值为 1 的位表明一个 DMA 通道被使能。在 DMA 通道的传输结束时相应的位被清零。[表 673](#)为 DMACEnbldChns 寄存器的位分配。

表673. DMA 使能通道寄存器（DMACEnbldChns—0x2008 001C）

位	名称	功能
7:0	EnabledChannels	DMA 通道的使能状态。每一位代表一条通道： 0—DMA 通道被禁能。 1—DMA 通道被使能。
31:8	-	保留。从保留位读出的值未定义。

34.5.9 DMA 软件突发请求寄存器（DMACSoftBReq—0x2008 0020）

DMACSoftBReq 是一个可读/写的寄存器，它使能软件产生 DMA 突发请求。每个源的 DMA 请求可以通过向相应的寄存器位写入 1 来产生。在传输结束时一个寄存器位被清零。读取寄存器来指示哪个源正在请求 DMA 突发传输。请求可以由外设或软件请求寄存器来产生。每个位在相关的传输结束时被清零。[表 674](#)所示为 DMACSoftBReq 寄存器的位分配。

表674. DMA 软件突发请求寄存器（DMACSoftBReq—0x2008 0020）

位	名称	功能
15:0	SoftBReq	16 个可能的源的软件突发请求标志。每个位代表一条 DMA 请求线或一个外设功能（有关到 DMA 控制器的外设硬件连接，参见 表 664 ）： 0—写入 0 无效。 1—写入 1 产生相应请求线的 DMA 突发请求。
31:16	-	保留。读取值未定义，只写入 0。

注：建议不要同时使用软件和硬件外设请求。

34.5.10 DMA 软件单次请求寄存器（DMACSoftSReq—0x2008 0024）

DMACSoftSReq 是一个可读/写的寄存器，它使能软件产生 DMA 单次请求。每个源的 DMA 请求可以通过向相应的寄存器位写 1 来产生。在传输结束时一个寄存器位被清零。读取寄存器来指示哪个源正在请求单次 DMA 传输。请求可以由外设或软件请求寄存器产生。[表 675](#)所示为 DMACSoftSReq 寄存器的位分配。

表675. DMA 软件单次请求寄存器（DMACSoftSReq—0x2008 0024）

位	名称	功能
5:0	SoftSReq	16 个可能的源的软件单次传输请求标志。每个位代表一条 DMA 请求线或一个外设功能： 0—写入 0 无效。 1—写入 1 产生相应请求线的 DMA 单次传输请求。
31:16	-	保留。读取值未定义，只写入 0。

34.5.11 DMA 软件最后一个突发请求寄存器（DMACSoftLBReq—0x2008 0028）

DMACSoftLBReq 是一个可读/写的寄存器，它使能软件产生 DMA 最后一个突发请求。每个源的 DMA 请求可以通过向相应的寄存器位写 1 来产生。传输结束时一个寄存器位被清零。读取寄存器来指示哪个源正在请求最后一个突发 DMA 传输。请求可以由外设或软件请求寄存器产生。[表 676](#)所示为 DMACSoftLBReq 寄存器的位分配。

表676. DMA 软件最后一个突发请求寄存器（DMACSoftLBReq—0x2008 0028）

位	名称	功能
15:0	SoftLBReq	16 个可能的源的软件最后一个突发请求标志。每个位代表一条 DMA 请求线或一个外设功能： 0—写入 0 无效。 1—写入 1 产生相应请求线的 DMA 最后一个突发请求。
31:16	-	保留。读取值未定义，只写入 0。

34.5.12 DMA 软件最后一个单次请求寄存器（DMACSoftLSReq—0x2008 002C）

DMACSoftLSReq 是一个可读/写寄存器，它使能软件产生 DMA 最后一个单次请求。每个源的 DMA 请求可以通过向相应的寄存器位写入 1 来产生。在传输结束时一个寄存器位被清零。读取寄存器来指示哪个源正在请求最后一个单次 DMA 传输。请求可以由外设或软件请求寄存器产生。[表 677](#)所示为 DMACSoftLSReq 寄存器的位分配。

表677. DMA 软件最后一个单次请求寄存器（DMACSoftLSReq—0x2008 002C）

位	名称	功能
15:0	SoftLSReq	16 个可能的源的软件最后一个单次传输请求标志。每个位代表一条 DMA 请求线或一个外设功能： 0—写入 0 无效。 1—写入 1 产生相应请求线的 DMA 最后一个单次传输请求。
31:16	-	保留。读取值未定义，只写入 0。

34.5.13 DMA 配置寄存器（DMACConfig—0x2008 0030）

DMACConfig 是一个可读/写的寄存器，它配置 DMA 控制器的操作。AHB 主机接口的字节顺序通过写这个寄存器的 M 位来改变。复位时，AHB 主机接口被设置成小端模式。[表 678](#)所示为 DMACConfig 寄存器的位分配。

表678. DMA 配置寄存器（DMACConfig—0x2008 0030）

位	名称	功能
0	E	DMA 控制器使能： 0=禁能（默认）。禁能 DMA 控制器可降低功耗。 1=使能。
1	M	AHB 主机字节顺序配置： 0=小端模式（默认）。 1=大端模式。
31:2	-	保留。读取值未定义，只写入 0。

34.5.14 DMA 同步寄存器（DMACSync—0x2008 0034）

DAMCSync 是一个可读/写的寄存器，它使能或禁能 DMA 请求信号的同步逻辑。DMA 请求信号由 DMACBREQ[15:0]、DMACSREQ[15:0]、DMALBREQ[15:0]和 DMALSREQ[15:0]组成。将一个位设为 0 可使能某组 DMA 请求的同步逻辑。将一个位设为 1 可禁能某组 DMA 请求的同步逻辑。若将这个寄存器复位为 0，则默认使能同步逻辑。[表 679](#)所示为 DMACSync 寄存器的位分配。

表679. DMA 同步寄存器（DMACSync—0x2008 0034）

位	名称	功能
15:0	DMACSync	控制 DMA 请求信号的同步逻辑。每个位代表一组 DMA 请求线（如前文所述）： 0—对应 DMA 请求信号的同步逻辑被使能。 1—对应 DMA 请求信号的同步逻辑被禁能。
31:16	-	保留。读取值未定义，只写入 0。

34.5.15 DMA 请求选择寄存器（DMACReqSel—0x400F C1C4）

DMACReqSel 是一个可读/写的寄存器,它可以为 DMA 输入 0~7 和 10~15 选择 DMA 请求。
[表 680](#) 所示为 DMACReqSel 寄存器的位分配。

表680. DMA 请求选择寄存器（DMACReqSel—0x400F C1C4）

位	名称	功能
0	DMASEL00	为 GPDMA 输入 0 选择 DMA 请求： 0—（不使用） 1—定时器 0 匹配 0 被选中。
1	DMASEL01	为 GPDMA 输入 1 选择 DMA 请求： 0—SD 卡接口被选中。 1—定时器 0 匹配 1 被选中。
2	DMASEL02	为 GPDMA 输入 2 选择 DMA 请求： 0—SSP0 发送被选中。 1—定时器 1 匹配 0 被选中。
3	DMASEL03	为 GPDMA 输入 3 选择 DMA 请求： 0—SSP0 接收被选中。 1—定时器 1 匹配 1 被选中。
4	DMASEL04	为 GPDMA 输入 4 选择 DMA 请求： 0—SSP1 发送被选中。 1—定时器 2 匹配 0 被选中。
5	DMASEL05	为 GPDMA 输入 5 选择 DMA 请求： 0—SSP1 接收被选中。 1—定时器 2 匹配 1 被选中。
6	DMASEL06	为 GPDMA 输入 6 选择 DMA 请求： 0—SSP2 发送被选中。 1—I ² S 通道 0 被选中。
7	DMASEL07	为 GPDMA 输入 7 选择 DMA 请求： 0—SSP2 接收被选中。 1—I ² S 通道 1 被选中。
9:8	-	保留。读取值未定义，只写入 0。
10	DMASEL10	为 GPDMA 输入 10 选择 DMA 请求： 0—UART0 发送被选中。 1—UART3 发送被选中。
11	DMASEL11	为 GPDMA 输入 11 选择 DMA 请求： 0—UART0 接收被选中。 1—UART3 接收被选中。
12	DMASEL12	为 GPDMA 输入 12 选择 DMA 请求： 0—UART1 发送被选中。 1—UART4 发送被选中。
13	DMASEL13	为 GPDMA 输入 13 选择 DMA 请求： 0—UART1 接收被选中。 1—UART4 接收被选中。
14	DMASEL14	为 GPDMA 输入 14 选择 DMA 请求： 0—UART2 发送被选中。 1—定时器 3 匹配 0 被选中。
15	DMASEL15	为 GPDMA 输入 15 选择 DMA 请求： 0—UART2 接收被选中。 1—定时器 3 匹配 1 被选中。
31:16	-	保留。读取值未定义，只写入 0。

34.5.15.1 定时器 DMA 请求

当定时器的值与相关的匹配寄存器相匹配时，定时器就会产生定时器 DMA 请求（参见 [24.6.12](#) 节）。如果 DMA 控制器已被设置为选择一个定时器 DMA 请求作为某个 DMA 通道的输入并且该 DMA 通道被使能，那么 DMA 控制器将会按照这个请求进行操作。

34.5.16DMA 通道寄存器

通道寄存器被用来对 8 个 DMA 通道进行编程。这类寄存器包括：

- 8 个 DMACCxSrcAddr 寄存器
- 8 个 DMACCxDesAddr 寄存器
- 8 个 DMACCxLLI 寄存器
- 8 个 DMACCxControl 寄存器
- 8 个 DMACCxConfig 寄存器

在执行分散/聚集 DMA 传输时，这些寄存器中的前四个寄存器会自动更新。

34.5.17DMA 通道源地址寄存器（DMACCxSrcAddr—0x2008 01x0）

8 个可读/写的 DMACCxSrcAddr 寄存器（DMACC0SrcAddr~DMACC7SrcAddr）包含当前需要传输数据的源地址（字节对齐）。每个寄存器在相应的通道使能前由软件直接编程。当 DMA 通道被使能后，这个寄存器会自动更新：

- 跟随源地址的增加而更新；
- 当一个完整的数据包传输结束时源地址紧跟在链表之后。

在通道有效时读取这个寄存器不能提供有用的信息，这是因为软件已经处理了读取值，地址已自动更新。只有在停止通道后读取这个寄存器，寄存器显示的才是读出的最后一项的源地址。

注：源和目的地址必须与源和目标宽度对齐。

[表 681](#) 所示为 DMACCxSrcAddr 寄存器的位分配。

表681. DMA 通道源地址寄存器（DMACCxSrcAddr—0x2008 01x0）

位	名称	功能
31:0	SrcAddr	DMA 源地址。读取该寄存器会返回当前的源地址。

34.5.18DMA 通道目标地址寄存器（DMACCxDestAddr—0x2008 01x4）

8 个可读/写的 DMACCxDestAddr 寄存器（DMACC0DestAddr~DMACC7DestAddr）包含当前要传输数据的目标地址（字节对齐）。每个寄存器在相应的通道使能之前由软件直接编程。在 DMA 通道使能后，寄存器的内容也会跟随目标地址的增加而更新，并在一个完整的数据包传输结束后紧跟在链表之后。在通道有效时读取这个寄存器不能提供有用的信息，因为软件已经处理了读取值，地址已自动更新。只有在停止通道后读取这个寄存器，寄存器显示的才是读出的最后一项的目标地址。[表 682](#) 所示为 DMACCxDestAddr 寄存器的位分配。

表682. DMA 通道目标地址寄存器（DMACCxDestAddr—0x2008 01x4）

位	名称	功能
31:0	DestAddr	DMA 目标地址。读取该寄存器会返回当前的目标地址。

34.5.19DMA 通道链表项寄存器（DMACCxLLI—0x2008 01x8）

8 个可读/写的 DMACCxLLI 寄存器(DMACC0LLI~DMACC7LLI)包含下一个链表项(LLI)的字对齐的地址。如果 LLI 为 0，则当前的 LLI 是链中的最后一项，当与之相关的所有 DMA 传输结束后，DMA 通道被禁止。在 DMA 通道使能时对这个寄存器编程可能会有不可预知的副作用。[表 683](#)所示为 DMACCxLLI 寄存器的位分配。

表683. DMA 通道链表项寄存器（DMACCxLLI—0x2008 01x8）

位	名称	功能
1:0	-	保留，必须为 0。
31:2	LLI	链表项。下个 LLI 的地址位[31:2]。地址位[1:0]为 0。

34.5.20DMA 通道控制寄存器（DMACCxControl—0x2008 01xC）

8 个可读/写 DMACCxControl 寄存器(DMACC0Control~DMACC7Control)包含传输大小、突发大小和传输宽度等 DMA 通道的控制信息。每个寄存器在相应的通道使能之前由软件直接编程。当通道被使能时，在一个完整的数据包传输结束后这个寄存器的值代表的地址紧跟在链表之后。在通道有效时读取这个寄存器不能提供有用的信息，因为软件已经处理了读取值，通道会自动更新。只有在停止通道后才能读取这个寄存器。[表 684](#)所示为 DMACCxControl 寄存器的位分配。

34.5.20.1 保护和访问信息

在传输时，会向源外设和/或目标外设提供 AHB 访问信息，尽管这一操作在 LPC178x/177x 上是无效的。编程 DMA 通道（DMACCxControl 寄存器的 Prot 位以及 DMACCxConfig 寄存器的 Lock 位）来提供传输信息。这些位由软件来编程，并且可以供外设使用。[表 684](#)提供了 3 个位的信息。

表684. DMA 通道控制寄存器 (DMACCxControl—0x2008 01xC)

位	名称	功能
11:0	TransferSize	传输大小。当 DMA 控制器为流控制器时，这个字段可设置传输的大小。传输大小值必须在通道使能前设置。当数据传输结束时，传输大小及时更新。 读取该字段来表示在目标总线上已完成的传输数量。在通道有效时读取该寄存器不能提供有用的信息，因为此时软件已经处理了读取值，通道也可能已经更新。只有当通道被使能然后再被禁能时，才能通过读取该寄存器获得有效值。如果外设为流控制器，传输大小值无法使用。
14:12	SBSize	源突发大小。该字段表示一个源突发的传输数量。这个值必须设置成源外设的突发大小，如果源是存储器，这个值就设置成存储器边界大小。在源外设突发请求信号 (DMACBREQ) 有效时，突发大小为传输的数据量。 000—1 001—4 010—8 011—16 100—32 101—64 110—128 111—256
17:15	DBSize	目标突发大小。该字段表示一个目标突发传输请求的传输数量。这个值必须设置成目标外设的突发大小，如果目标是存储器，这个值就设置成存储器边界大小。在目标外设突发请求信号 (DMACBREQ) 有效时，突发大小为传输的数据量。 000—1 001—4 010—8 011—16 100—32 101—64 110—128 111—256
20:18	SWidth	源传输宽度。源和目标宽度可以不同。硬件会根据需要自动打包和拆分数据。 000—字节 (8 位) 001—半字 (16 位) 010—字 (32 位) 011~111—保留
23:21	DWidth	目标传输宽度。源和目标宽度可以不同。硬件会根据需要自动打包和拆分数据。 000—字节 (8 位) 001—半字 (16 位) 010—字 (32 位) 011~111—保留
25:24	-	保留，必须为 0。

位	名称	功能
26	SI	源增加： 0—每次传输后源地址不增加。 1—每次传输后源地址增加。
27	DI	目标增加： 0—每次传输后目标地址不增加。 1—每次传输后目标地址增加。
28	Prot1	在 DMA 总线访问期间供外设使用，表明是在用户模式还是特权模式下访问总线。该信息不在 LPC178x/177x 中使用。 0—在用户模式下访问。 1—在特权模式下访问。
29	Prot2	在 DMA 总线访问期间供外设使用，表明对于外设来说，访问是否可缓冲。该信息不在 LPC178x/177x 中使用。 0—访问不可缓冲。 1—访问可缓冲。
30	Prot3	在 DMA 总线访问期间供外设使用，表明对于外设来说，访问是否可缓存。该信息不在 LPC178x/177x 中使用。 0—访问不可缓存。 1—访问可缓存。
31	I	终端计数中断使能位。 0—禁止终端计数中断。 1—允许终端计数中断。

34.5.21 DMA 通道配置寄存器（DMACCxConfig—0x2008 01x0）

这 8 个 DMACCxConfig 寄存器(DMACC0Config~DMACC7Config)是可读/写的,但位[17]除外，它是只读的。这些寄存器用来配置 DMA 信道。当请求一个新的 LLI 时寄存器不会更新。[表 685](#)所示为 DMACCxConfig 寄存器的位分配。

表685. DMA 通道配置寄存器（DMACCxConfig—0x2008 01x0）

位	名称	功能
0	E	通道使能。读该位可以获得当前通道的使能状态： 0=通道禁能。 1=通道使能。 通道使能位状态也可以通过读取 DMACEnbldChns 寄存器获得。置位使能位可以使能一个通道。 清零使能位可以禁止一个通道。会使当前 AHB 传输（如果有一个传输正在进行）结束，通道禁止。相应通道 FIFO 里的所有数据会丢失。如果通过置位通道使能位来重启通道，就会产生不可预知的影响。因此，必须对通道重新进行初始化。 当最后 LLI 到达时、DMA 传输完成时，或者碰到一个通道错误时，通道也可以被禁能，且通道使能位清零。 如果某个通道必须被禁能而又不丢失 FIFO 中的数据，则 Halt 位必须被置位，使得后面的 DMA 请求被忽略。然后轮询 Active 位直到它为 0，表明 FIFO 中没有数据了。最后，清零通道使能位。
5:1	SrcPeripheral	源外设。该值选择 DMA 源请求外设。如果源传输是存储器，忽略该字段。有关外设识别的内容，参见表 664。
10:6	DestPeripheral	目标外设。该值选择 DMA 目标请求外设。如果目标传输是存储器，忽略该字段。有关外设识别的内容，参见表 664。
13:11	TransferType	该值表明传输类型并指定流控制器。传输类型可以是存储器到存储器、存储器到外设、外设到存储器或外设到外设。流量可以通过 DMA 控制器、源外设或目标外设控制。 有关该字段编码的内容，参见表 686。
14	IE	中断错误屏蔽。IE 被清零时会屏蔽相关通道的错误中断。
15	ITC	终端计数中断屏蔽。ITC 被清零时会屏蔽相关通道的终端计数中断。
16	L	锁定。L 置位时会使能被锁定的传输。该信息不在 LPC178x/177x 中使用。
18	H	Halt: 0=使能 DMA 请求。 1=忽略后面的源 DMA 请求。 通道 FIFO 中的数据被传输。 该位可以和 Active 以及通道使能位一起用来彻底禁能一个 DMA 通道。
31:19	-	保留。读取值未定义，只写入 0。

34.5.21.1 锁定控制

向 DMACCxConfig 寄存器的位 16 写入 1 就可以设置锁定位。产生突发传输时，AHB 仲裁器不允许在突发传输过程中剥夺主机的授权，直至锁定被取消。一个单次突发传输（例如：一个长源读取突发传输或一个长目标输出数据突发传输）也可以锁定 DMA 控制器。一般情况下，DMA 控制器不会连续锁定在源读取突发传输和目标输出数据突发传输中。

当 DMA 的内部条件允许它在源读取数后紧接着执行目标输出数据时，DMA 才可能在先是源传输、后面紧跟着目标传输的情况下被锁定。

34.5.21.2 传输类型

表 686 列出了表 685 所要识别的传输类型的位值。

表686. 传输类型位

位值	传输类型	流控制器
000	存储器到存储器	DMA 控制器
001	存储器到外设	DMA 控制器
010	外设到存储器	DMA 控制器
011	源外设到目标外设	DMA 控制器
100	源外设到目标外设	目标外设
101	存储器到外设	目标外设
110	外设到存储器	源外设
111	源外设到目标外设	源外设

34.6 使用 DMA 控制器

34.6.1 编程 DMA 控制器

DMA 控制器的所有寄存器只能使用字读取和字写入来访问（即 32 位访问）。

34.6.1.1 使能 DMA 控制器

置位 DMACConfig 寄存器的使能位来使能 DMA 控制器。

34.6.1.2 禁能 DMA 控制器

执行以下操作来禁能 DMA 控制器：

- 读取DMACEnbldChns寄存器，确保所有DMA通道已被禁能。如果还有通道有效，请参考有关禁能DMA通道的方法。
- 通过清零 DMACConfig 寄存器中的 DMA 使能位来禁能 DMA 控制器。

34.6.1.3 使能 DMA 通道

置位相应 DMA 信道配置寄存器的信道使能位可使能 DMA 通道。注意：通道在使能前必须完全地初始化。

34.6.1.4 禁能 DMA 通道

可通过以下 3 种方法来禁能 DMA 通道：

- 直接写通道使能位。此时，通道FIFO中所有未处理的数据都会丢失。
- 将Active位、Halt位与通道使能位一起使用来禁能DMA通道。
- 等到传输结束，通道会自动清除。

禁能 DMA 通道且 FIFO 中的数据被丢失

若清除相应通道配置寄存器中的相关通道使能位，当前 AHB 传输会结束（如果有一个传输正在进行），该通道也会被禁能。FIFO 中的所有数据都会丢失。因此，通道在再次被使能之前应完全地初始化。

禁止 DMA 通道且不会丢失 FIFO 中的数据

- 置位相应信道配置寄存器的Halt位，后面的DMA请求将被忽略。
- 轮询相应信道配置寄存器的Active位，直至它的值到达0。这一位表示通道中是否有要传输的数据。
- 清除相应通道配置寄存器中的通道使能位。

34.6.1.5 设置一个新的 DMA 传输

设置一个新的 DMA 传输：

如果某个通道不能留出来供 DMA 传输使用，则：

1. 读取 DMACEndChns 寄存器并找出空闲的信道。
2. 选择一个具有所需优先级的空闲通道。
3. 编程 DMA 控制器。

34.6.1.6 中止某个 DMA 通道

置位相关通道配置寄存器的 Halt 位会响应当前的源请求。后来的源 DMA 请求都会被忽略直至 Halt 位被清零。

34.6.1.7 编程 DMA 通道

1. 选择一个具有所需优先级的空闲信道，DMA通道0的优先级最高，DMA通道7的优先级最低。
2. 通过写DMACIntTCClear和DMACIntErrClear寄存器清除所使用通道上的挂起中断标志。之前的通道操作可能会使中断继续有效。
3. 向 DMACCxSrcAddr 寄存器写入源地址。
4. 向 DMACCxDestAddr 寄存器写入目标地址。
5. 向DMACCxLLI寄存器写入下一个LLI的地址。如果只传输一个数据包，则必须向这个寄存器写入0。
6. 向DMACCxControl寄存器写入控制信息。
7. 向DMACCxConfig寄存器写入通道配置信息。如果使能位被置位，则DMA通道自动使能。

34.6.2 流控制

控制数据包长度的设备就是流控制器。流控制器通常指的是 DMA 控制器，在 DMA 通道使能前，软件必须编程好数据包的长度。大多数外设不能用作流控制器，但如果支持这一特性，就可以被用作源外设或目标外设。

当 DMA 传输结束时：

1. DMA 控制器向外设产生一个应答来指示传输已经结束。
2. 产生一个 TC（终端计数）中断（如果这个中断被使能）。
3. DMA 控制器转向下一个 LLI。

接下来我们将描述 4 种被允许的 DMA 传输类型的 DMA 数据流：

- 存储器到外设
- 外设到存储器
- 存储器到存储器
- 外设到外设

除了存储器对存储器传输之外，其它传输都将外设或 DMA 控制器作为流控制器，进而得到 8 种可能的控制场景。

表 687 所示为每种类型的传输所使用的请求信号。

表687. DMA 请求信号的使用

传输方向	请求发生器	流控制器
存储器到外设	目标外设	DMA 控制器
外设到存储器	源外设	DMA 控制器
存储器到存储器	DMA 控制器	DMA 控制器
源外设到目标外设	源外设和目标外设	DMA 控制器
存储器到外设	目标外设	目标外设
外设到存储器	源外设	源外设
源外设到目标外设	源外设和目标外设	源外设
源外设到目标外设	源外设和目标外设	目标外设

34.6.2.1 外设到存储器或存储器到外设 DMA 流

对于外设到存储器或存储器到外设的 DMA 流，会出现以下序列：

1. 编程和使能 DMA 通道。
2. 等待一个 DMA 请求。
3. 当下列情况出现时，DMA 开始传输数据：
 - DMA 请求变有效
 - DMA 流具有最高挂起优先级
 - DMA 控制器是 AHB 总线的总线主机
4. 如果在传输数据时出错，则会产生一个错误中断并禁能 DMA 流，数据流传输终止。
5. 传输计数递减。
6. 如果传输已经结束（如果 DMA 控制器正在执行流控制，就以传输计数为 0 来指示；或如果外设正在执行流控制时，就以外设发送 DMA 请求来指示）：
 - DMA 控制器用一个 DMA 应答来响应；
 - 产生终端计数中断（这个中断可以被屏蔽）；
 - 如果 DMACCxLLI 寄存器不为 0，则重装 DMACCxSrcAddr、DMACCxDestAddr、DMACCLLI 和 DMACCxControl 寄存器，并返回到步骤 2）。但是，如果 DMACCxLLI 寄存器为 0，则 DMA 流被禁能，流序列终止。

34.6.2.2 外设到外设 DMA 流

对于外设到外设的 DMA 流，会出现以下序列：

1. 编程和使能 DMA 通道。
2. 等待一个源 DMA 请求。
3. 当下列情况出现时，DMA 开始传输数据：
 - DMA 请求变有效
 - DMA 流具有最高挂起优先级
 - DMA 控制器是 AHB 总线的总线主机
4. 如果在传输数据时出错，则产生一个错误中断并禁能 DMA 流，流序列终止。
5. 传输计数递减。

6. 如果传输已经结束（如果 DMA 控制器正在执行流控制，就以传输计数为 0 来指示；或如果外设正在执行流控制时，就以外设发送 DMA 请求来指示）：
 - DMA 控制器用一个对源外设的 DMA 应答来响应
 - 后面的源 DMA 请求被忽略
7. 当目标 DMA 请求变得有效且 DMA 控制器通道 FIFO 中有数据时，将数据传输到目标外设。
8. 如果传输数据时出错，则产生一个错误中断并禁能 DMA 流，流序列终止。
9. 如果传输已经结束（如果 DMA 控制器正在执行流控制，就以传输计数为 0 来指示；或如果外设正在执行流控制时，就以外设发送 DMA 请求来指示）：
 - DMA 控制器用一个对目标外设的 DMA 应答来响应
 - 产生终端计数中断（这个中断可以被屏蔽）
 - 如果 DMACCxLLI 寄存器不为 0，则重装 DMACCxSrcAddr、DMACCxDestAddr、DMACCLLI 和 DMACCxControl 寄存器，并返回到步骤 2。但是，如果 DMACCxLLI 为 0，则 DMA 流被禁能，流序列终止。

34.6.2.3 存储器到存储器的 DMA 流

对于存储器到存储器的 DMA 流，会出现以下流程：

1. 编程和使能 DMA 通道。
2. 只要对应的 DMA 通道具有最高挂起优先级并且 DMA 控制器获得 AHB 总线的控制权，就开始传输数据。
3. 如果在传输数据时出错，则产生一个错误中断并禁能 DMA 流。
4. 传输计数递减。
5. 如果计数已经达到 0：
 - 产生一个终端计数中断（这个中断可以被屏蔽）
 - 如果 DMACCxLLI 寄存器不为 0，则重装 DMACCxSrcAddr、DMACCxDestAddr、DMACCLLI 和 DMACCxControl 寄存器，并返回到步骤 2。但是，如果 DMACCxLLI 为 0，则 DMA 流被禁能，流序列终止。

注：存储器到存储器的传输应该使用低优先级通道，否则在存储器到存储器传输结束之前，其它 DMA 通道都不能访问总线，或者其它 AHB 主机都不能执行任何传输。

34.6.3 中断请求

当遇到一个 AHB 错误或者在当前 LLI 对应的数据被传输到目标之后（终端计数），可以产生中断请求。可以通过编程 DMACCxControl 寄存器和 DMACCxConfig 通道寄存器中的相应位来屏蔽中断。在 DMACRawIntTCStat 和 DMACRawIntErrStat 寄存器中可以找到所有 DMA 通道的中断请求。DMACIntTCStat 和 DMACIntErrStat 寄存器中包含有 DMA 中断数据的屏蔽版本。然后，DMACIntStat 寄存器将 DMACIntTCStat 和 DMACIntErrStat 请求组合到一个寄存器中，这样中断源就可以很快被找到。通过写 DMACIntTCClear 或者置位 DMACIntErrClr 寄存器中的某位为 1 可以有选择性地清除中断。

34.6.3.1 硬件中断序列

当一个 DMA 中断请求出现时，中断服务程序需要：

1. 读取DMACIntTCStatus寄存器来判断中断是否因为传输的结束而产生（终端计数）。
2. 为1时表明传输结束。如果有1个以上的请求有效，建议先检查最高优先级的通道。
3. 读取DMACIntErrStatus寄存器来判断中断是否是因为错误的出现而导致的。为1时表明出现了错误。
4. 处理中断请求。可以通过读取DMACIntStat寄存器来确定导致中断产生的通道。如果有一个以上的请求有效，建议优先处理最高优先级的请求。
5. 对于终端计数中断，可以向DMACIntTCClr寄存器的相关位写入1；对于一个错误引起的中断，可以向DMACIntErrClr寄存器的相关位写入1来清除中断请求。

34.6.4 地址的产生

执行 DMA 传输时，地址可以是递增的，也可以是非递增的（不支持地址回环）。

有些设备，特别是存储器，在突发访问时不允许跨过指定的地址边界。DMA 控制器假设这是在任何源或目标区都是递增寻址的情况下。而假设地址是以指定的突发大小为边界的。例如：如果设置一个进行 16 次突发传输（到 32 位宽的设备），那么地址边界就是 64 字节（也就是地址位[5:0]=0）。如果某个 DMA 突发超出了其中一个边界，那么突发传输就会分成几个独立的 AHB 传输。

34.6.4.1 跨边界的字对齐传输

配置一个用于 16 次突发传输的通道，每次向目标（递增寻址的）传输 32 位数据。当前突发的起始地址为 0x0C000024，下一个边界为 0x0C000040（由突发大小和传输宽度计算而来）。

传输将分成 2 个 AHB 传输：

- 7 次突发传输，从地址 0x0C000024 开始；
- 9 次突发传输，从地址 0x0C000040 开始。

34.6.5 分散/聚集

DMA 使用链表来支持分散/聚集。这就意味着源和目标区不需要占用连续的存储器空间。在不需要分散/聚集时，DMACCxLLI 寄存器必须被设置成 0。

源和目标数据区由一连串的链表来定义。每个 LLI 控制着一个数据块的传输，将这个数据块传输完毕后，选择并装载另一个 LLI 来继续 DMA 操作或停止 DMA 流。第一个 LLI 需要被编程到 DMA 控制器中。

LLI 描述的传输数据（指的是数据包）通常需要进行一次或多次 DMA 突发传输（到每个源和目标）。

34.6.5.1 链表项

一个链表项 (LLI) 由 4 个字组成。这些字按照以下顺序来排列:

1. DMACCxSrcAddr
2. DMACCxDestAddr
3. DMACCxLLI
4. DMACCxControl

注: DMACCxConfig DMA 通道配置寄存器并不是链表项的一部分。

34.6.5.1.1 编程 DMA 控制器用于分散/聚集 DMA

执行下列操作来编程 DMA 控制器, 使之用于分散/聚集 DMA:

1. 在存储器中创建整个 DMA 传输的链表项。每个链表项包含 4 个字:
 - 源地址;
 - 目标地址;
 - 指向下个 LLI 的指针;
 - 控制字。

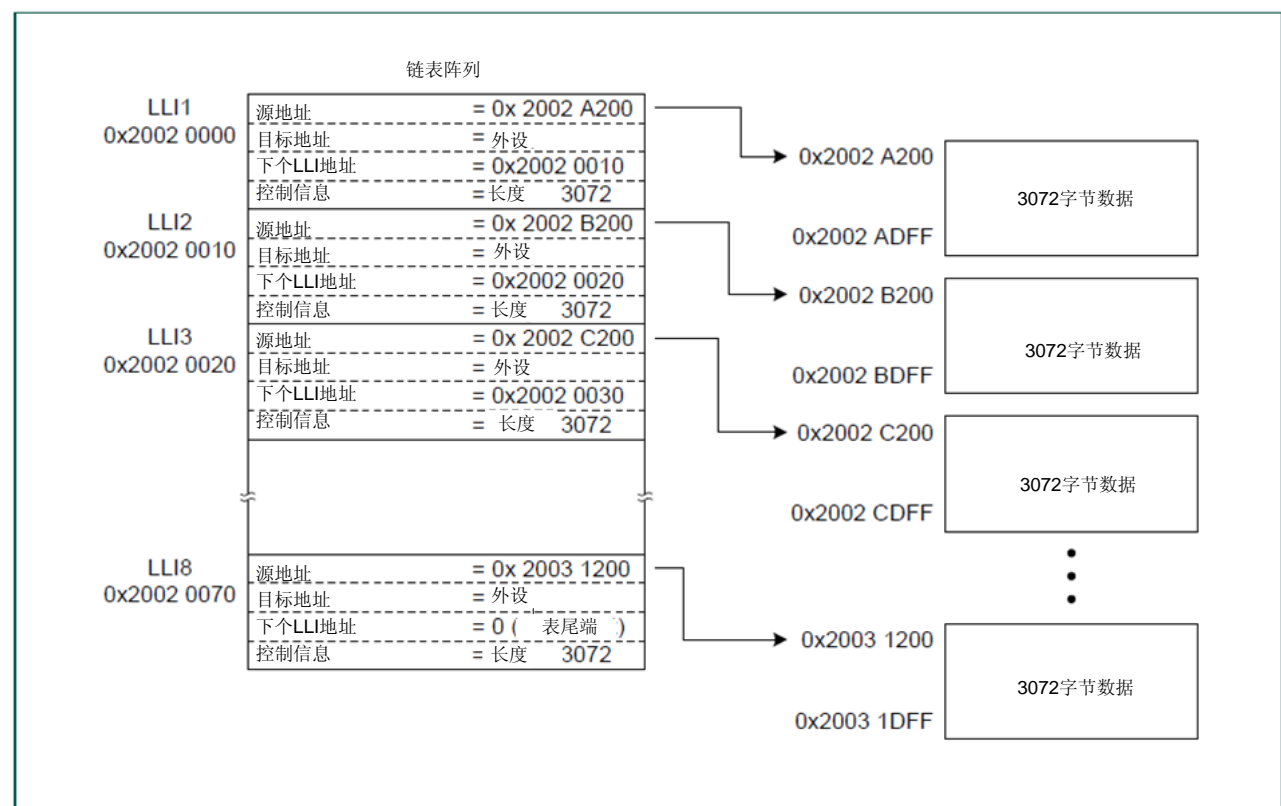
最后一个 LLI 的链表项指针需设置为 0。

2. 选择一个具有所需优先级的空闲 DMA 通道。DMA 通道 0 的优先级最高, DMA 通道 7 的优先级最低。
3. 将第一个先前写入存储器的链表项写入 DMA 控制器的相关通道。
4. 向通道配置寄存器写入通道配置信息, 并置位通道使能位。接下来 DMA 控制器会开始传输第一个数据包。每个数据包传输完毕后, 都会自动装载下一个链表项。
5. 在每个 LLI 传输完毕时都可以产生中断, 由 DMACCxControl 寄存器的终端计数位决定。如果这个位被置位, 则传输完对应的 LLI 后会产生中断。然后, 执行中断请求服务, 通过置位 DMACIntTCClear 寄存器中的相关位来清除该中断。

34.6.5.1.2 分散/聚集 DMA 的例子

[图 162](#) 所示为一个 LLI 示例。某个存储区的内容将被传输到一个外设。图的左边是每个 LLI 表项的地址 (十六进制)。描述传输的链表项将从地址 0x2002 0000 开始保存在一个连续区域内。图的右边显示了存储器所包含的待传输的数据。

图162. LLI 示例



第一个链表项存放在 0x20000, 定义了要传输的第一个数据块, 这个数据块是存放在 0x2002 A200 至 0x2002 ADFF 之间的数据。

- 源起始地址 0x2002 A200
- 目标地址设为目标外设地址
- 传输数据宽度: 1 个字 (32 位)
- 传输大小: 3072 个字节 (0xC00)
- 源和目标突发大小: 16 个传输
- 下个 LLI 地址: 0x2002 0010

第二个链表项存放在 0x2002 0010, 描述了要传输的下个数据块:

- 源起始地址 0x2002 B200
- 目标地址设为目标外设地址
- 传输数据宽度: 1 个字 (32 位)
- 传输大小: 3072 个字节 (0xC00)
- 源和目标突发大小: 16 个传输;
- 下个 LLI 地址: 0x2002 0020

这样, 一个链表就形成了, 每个链表项都指向链表中的下一个链表项。为了初始化 DMA 流, 需要将第一个链表项 (0x2002 0000) 编程到 DMA 控制器中。当第一个数据包被传输后,

下个 LLI 就自动被装载。

最后一个 LLI 存放在 0x2002 0070，它包含：

- 源起始地址 0x2003 1200
- 目标地址设为目标外设地址
- 传输数据宽度：1 个字（32 位）
- 传输大小：3072 个字节（0xC00）
- 源和目标突发大小：16 个传输
- 下个 LLI 地址：0x0

由于下个 LLI 地址被设为 0，因此，这是最后一个描述符，而且，DMA 通道在传输完最后这个数据项后被禁能。当然，也可以将通道设置成在这一点上产生中断来向 ARM 处理器指明该通道可以被重新编程了。

35.1 简介

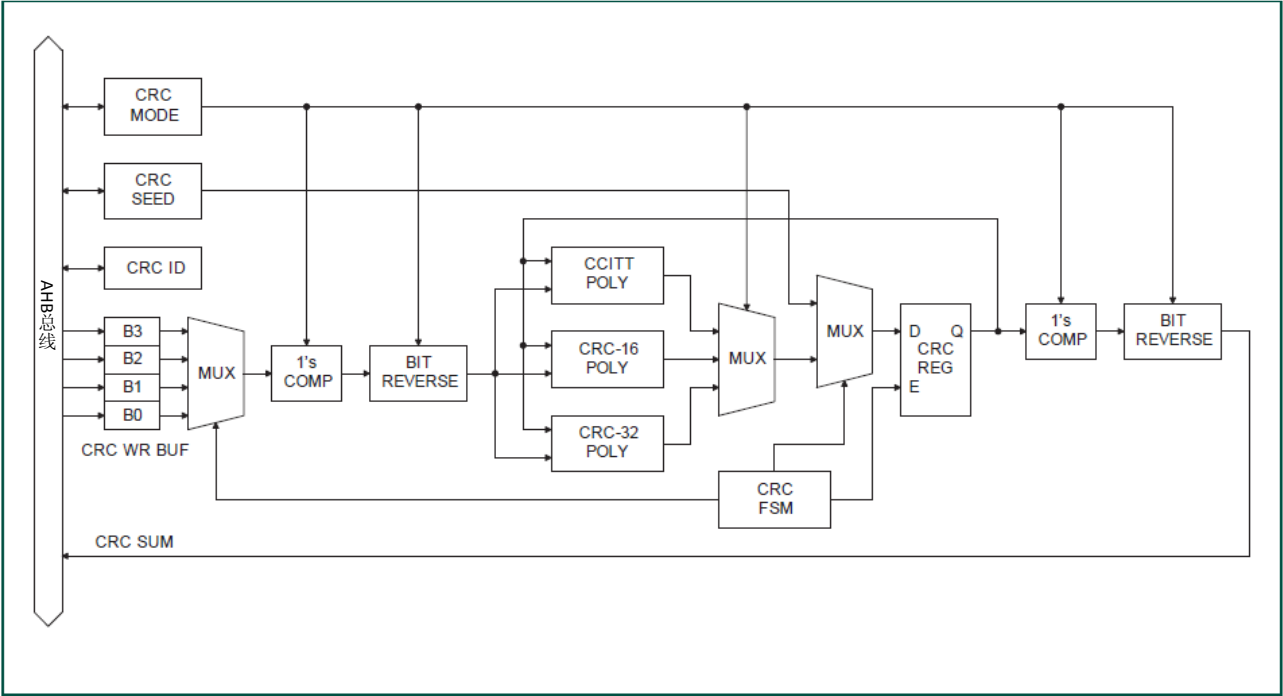
带有可编程多项式设置的循环冗余校验（CRC）发生器支持常用的多种 CRC 标准。为了节省系统功耗和总线带宽，除了使用 CPU 进行软件 PIO 操作之外，CRC 引擎还支持 DMA 传输。

35.2 特性

- 支持三种常见的多项式：CRC-CCITT、CRC-16 和 CRC-32。
 - CRC-CCITT: $x^{16}+x^{12}+x^5+1$
 - CRC-16: $x^{16}+x^{15}+x^2+1$
 - CRC-32: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$
- 位顺序反转和 1 的补码可编程设置，用于输入数据和 CRC 求和。
- 可编程的种子数设置。
- 支持 CPU PIO 或 DMA 连续性传输。
- 每次写操作接受任何大小的数据宽度：8、16 或 32 位。
 - 写入 8 位：需要 1 个操作周期
 - 写入 16 位：需要 2 个操作周期（8 位 x2 周期）
 - 写入 32 位：需要 4 个操作周期（8 位 x4 周期）

35.3 描述

图163.CRC 框图



35.4 寄存器描述

表688. 寄存器概述：CRC 引擎

名称	描述	访问	复位值 ^[1]	地址	表
CRC_MODE	CRC 模式寄存器	R/W	0	0x2009 0000	表 689
CRC_SEED	CRC 种子寄存器	R/W	0xFFFF	0x2009 0004	表 690
CRC_SUM	CRC 校验和寄存器	RO	0xFFFF	0x2009 0008	表 691
CRC_DATA	CRC 数据寄存器	WO	-	0x2009 0008	表 692

[1] 复位值仅反映使用位中保存的数据，不包括保留位的内容。

35.4.1 CRC 模式寄存器（CRC_MODE—0x2009 0000）

表689. CRC 模式寄存器位描述（CRC_MODE—0x2009 0000）

位	符号	描述	复位值
1:0	CRC_POLY	1X: CRC-32 多项式 01: CRC-16 多项式 00: CRC-CCITT 多项式	0
2	BIT_RVS_WR	1: CRC_WR_DATA 位顺序反转（每字节） 0: CRC_WR_DATA 位顺序未反转（每字节）	0
3	CMPL_WR	1: CRC_WR_DATA1 的补码 0: CRC_WR_DATA 没有 1 的补码	0
4	BIT_RVS_SUM	1: CRC_SUM 位顺序反转 0: CRC_SUM 位顺序没有反转	0
5	CMPL_SUM	1: CRC_SUM1 的补码 0: CRC_SUM 没有 1 的补码	0
31:6	Reserved	读取时总是 0	0

35.4.2 CRC 种子寄存器（CRC_SEED—0x2009 0004）

表690. CRC 种子寄存器位描述（CRC_SEED—0x2009 0004）

位	符号	描述	复位值
31:0	CRC_SEED	该寄存器的写访问会通过已选择的位顺序及 1 的补码预处理向 CRC_SUM 寄存器中载入 CRC 种子值。 注：该寄存器的写访问会对正在进行的 CRC 计算予以否决。	0xFFFF

35.4.3 CRC 校验和寄存器（CRC_SUM – 0x2009 0008）

表691. CRC 校验和寄存器位描述（CRC_SUM—0x2009 0008）

位	符号	描述	复位值
31:0	CRC_SUM	通过已选择的位顺序及 1 的补码预处理，可以从该寄存器读取到最近的 CRC 总和。	0xFFFF

35.4.4 CRC 数据寄存器（CRC_DATA—0x2009 0008）

该寄存器是一个只写寄存器，包含有助于计算 CRC 总和的数据块。

表692. CRC 数据寄存器位描述（CRC_DATA—0x2009 0008）

位	符号	描述	复位值
31:0	CRC_WR_DATA	通过已选择的位顺序及 1 的补码预处理，写入该寄存器的数据会被用来执行 CRC 计 - 算。8、16 或 32 位数据宽度都可以被写入并接受连续性传输。	

35.5 功能描述

以下章节描述了用于支持不同 CRC 标准的寄存器设置情况：

CRC-CCITT 设定（编程设定示例）

多项式= $x^{16}+x^{12}+x^5+1$
种子值=0xFFFF
数据输入是否进行位顺序反转：NO
数据输入是否进行 1 的补码：NO
CRC 总和是否进行位顺序反转：NO
CRC 总和是否进行 1 的补码：NO
CRC_MODE=0x0000 0000
CRC_SEED=0x0000 FFFF

CRC-16 设定

多项式= $x^{16}+x^{15}+x^2+1$
种子值= 0x0000
数据输入是否进行位顺序反转：YES
数据输入是否进行 1 的补码：NO
CRC 总和是否进行位顺序反转：YES
CRC 总和是否进行 1 的补码：NO
CRC_MODE=0x0000 0015
CRC_SEED=0x0000 0000

CRC-32 设定

多项式= $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$
种子值=0xFFFF FFFF
数据输入是否进行位顺序反转：YES
数据输入是否进行 1 的补码：NO
CRC 总和是否进行位顺序反转：YES
CRC 总和是否进行 1 的补码：YES
CRC_MODE=0x0000 0036
CRC_SEED=0xFFFF FFFF

36.1 基本配置

采用以下寄存器对 EEPROM 进行配置：

1. 功率：复位时，EEPROM 被使能，但在不需要的时候可以关闭，见 [36.4.1.7](#) 节。
2. 时钟：EEPROM 必须在使用之前先设置好时序，见 [36.4.1.5](#) 节和 [36.4.1.6](#) 节。
3. 中断：使用一组寄存器来控制中断，见 [36.4.2](#) 节。利用相应的中断置位使能寄存器来使能 NVIC 中的中断，见 [表 43](#)。

36.2 描述

EEPROM 是一类非易失性存储器，主要用于存储相对少量的数据，如存储设置值。通过地址或数据寄存器可间接地访问 EEPROM，因此 CPU 不能执行 EEPROM 存储器中的代码。

36.3 特性

- 4kB EEPROM
- 在 AHB 总线通过地址和数据寄存器进行访问
- 擦除/编程时间小于 3ms
- 耐用性：可擦除/编程 10 万次以上

36.4 寄存器描述

表 693 给出了与 EEPROM 操作有关的寄存器。每个寄存器的详细描述如下。

表693. 寄存器概述

名称	描述	访问	复位值 ^[1]	地址	表
EEPROM 寄存器					
EECMD	EEPROM 命令寄存器	R/W	0	0x0020 0080	表 694
EEADDR	EEPROM 地址寄存器	R/W	0	0x0020 0084	表 695
EEWDATA	EEPROM 写数据寄存器	WO	无	0x0020 0088	表 696
EERDATA	EEPROM 读数据寄存器	RO	无	0x0020 008C	表 697
EEWSTATE	EEPROM 等待状态寄存器	R/W	0	0x0020 0090	表 698
EECLKDIV	EEPROM 时钟分频器寄存器	R/W	0	0x0020 0094	表 699
EPPWRDWN	EEPROM 掉电寄存器	R/W	0	0x0020 0098	表 700
EEPROM 中断寄存器：					
EEINTEN	EEPROM 中断使能	RO	0	0x0020 0FE4	表 701
EEINTENCLR	EEPROM 中断使能清零	WO	0	0x0020 0FD8	表 702
EEINTENSET	EEPROM 中断使能置位	WO	0	0x0020 0FDC	表 703
EEINTSTAT	EEPROM 中断状态	RO	0	0x0020 0FE0	表 704
EEINTSTATCLR	EEPROM 中断状态清零	WO	0	0x0020 0FE8	表 705
EEINTSTATSET	EEPROM 中断状态置位	WO	0	0x0020 0FEC	表 706

[1] 复位值仅反映已使用位中保存的数据，不包括保留位的内容。

36.4.1 EEPROM 控制寄存器

36.4.1.1 EEPROM 命令寄存器（EECMD—0x0020 0080）

EEPROM 命令寄存器用于选择和启动一次读、写或擦除/编程操作。写寄存器的副作用就是会启动对 EEPROM 设备的读操作和擦除/编程操作（对写数据寄存器写入数据的副作用就是启动写操作）。

表694. EEPROM 命令寄存器位描述（EECMD—地址 0x0020 0080）

位	符号	描述	复位值
2:0	CMD	命令 000: 8 位读取 001: 16 位读取 010: 32 位读取 011: 8 位写入 100: 16 位写入 101: 32 位写入 110: 擦除/编程页 111: 保留	0
3	RDPREFETCH	读取数据预取位 0: 没有预取下一个读数据作为从读数据寄存器读出的结果。 1: 预取下一个读数据作为从读数据寄存器读出的结果。 当该位置位时，多个连续数据元可被读取，无需向地址寄存器中编程新地址。地址后增及自动读数据预取（如已使能）只允许读取读数据寄存器来获得数据。	0

位	符号	描述	复位值
31:4	-	保留。读取值未定义，只写入 0。	无

36.4.1.2 EEPROM 地址寄存器（EEADDR—0x0020 0084）

EEPROM 地址寄存器是用于对读、写或擦除/编程操作的地址进行编程。地址字段的宽度取决于 EEPROM 设备的数量（数量可以通过一个配置参数进行选择）。

表695. EEPROM 地址寄存器位描述（EEADDR—地址 0x0020 0084）

位	符号	描述	复位值
11:0	ADDR	地址 地址字段含： 1 EEPROM 设备 => x = 13 (8 MSB、 0 CS 和 6 LSB 位)	0
31:12	-	保留。读取值未定义，只写入 0。	无

36.4.1.3 EEPROM 写数据寄存器（EEWDATA—0x0020 0088）

EEPROM 写数据寄存器用于将数据写入页寄存器（写操作）。

写这个寄存器的副作用是启动写操作。地址将后递增。因此，可以通过对该寄存器进行连续写操作来写入突发数据。地址会依据写操作数据的大小自动递增。

若向该寄存器写入数据时，前一个操作（对同一个设备的读、写或擦除/编程操作）处于挂起状态，那么通过使就绪信号失效来暂停 AHB 上的写命令，直到前面的操作完成为止。为了避免系统总线的暂停，可以用中断状态寄存器轮询挂起操作的状态。

表696. EEPROM 写数据寄存器位描述（EEWDATA—地址 0x0020 0088）

位	符号	描述	复位值
31:0	WDATA	写数据 如果出现以下情况： 8 位写操作：位[7:0]必须包含有效写数据。 16 位写操作：位[15:0]必须包含有效写数据。 32 位写操作：位[31:0]必须包含有效写数据。	-

36.4.1.4 EEPROM 读数据寄存器（EERDATA—0x0020 008C）

EEPROM 读数据寄存器用于从存储器中读取数据。

读这个寄存器将启动下一个读操作，并且地址将后递增。可以通过对该寄存器进行连续读操作来读取突发数据。地址会依据读操作数据的大小自动递增。

如果从寄存器读取数据时，仍有处于挂起状态的读操作，那么通过使就绪信号失效来暂停 AHB 总线上的读命令，直到挂起的操作完成为止。为了避免系统总线的暂停，可以用中断状态寄存器轮询挂起操作的状态。

表697. EEPROM 读数据寄存器位描述（EERDATA—地址 0x0020 008C）

位	符号	描述	复位值
31:0	RDATA	读数据 如果出现以下情况： 8 位读操作：位[7:0]包含读数据，其他位为 0。 16 位读操作：位[15:0]包含读数据，其他位为 0。 32 位读操作：位[31:0]包含读数据。	-

36.4.1.5 EEPROM 等待状态寄存器（EEWSTATE—0x0020 0090）

EEPROM 控制器利用该寄存器中规定的时间来完成各种内部时序功能。用户必须给这个等待状态寄存器中的等待状态字段编写相应的值。这些字段采用-1 编码方式，因此编写 0 将生成一个周期的等待状态。

表698. EEPROM 等待状态寄存器位描述（EEESTATE—地址 0x0020 0090）

位	符号	描述	复位值
7:0	PHASE3	等待状态 3（减 1 编码） 系统时钟周期的数量，要求提供至少 15ns 的时间。	0
15:8	PHASE2	等待状态 2（减 1 编码） 系统时钟周期的数量，要求提供至少 55ns 的时间。	0
23:16	PHASE1	等待状态 1（减 1 编码） 系统时钟周期的数量，要求提供至少 35ns 的时间。	0
31:24	-	保留。读取值未定义，只写入 0。	无

36.4.1.6 EEPROM 时钟分频器寄存器（EECLKDIV—0x0020 0094）

EEPROM 设备需要一个 375kHz 的时钟。对系统总线时钟分频可得到这个时钟。时钟分频器寄存器包含有分频系数。

如果分频系数为 0，则时钟进入 IDLE 状态以节省功率。

$$\frac{cclk}{CLKDIV+1} \approx 375kHz \pm 6.67\%$$

例如, 如果 CPU 时钟为 80MHz, 则 CLKDIV 可以设置为 212(十进制)。80 MHz/213 (CLKDIV + 1) = 375.6kHz.

表699. EEPROM 时钟分频器寄存器位描述（EECLKDIV—地址 0x0020 0094）

位	符号	描述	复位值
15:0	CLKDIV	分频系数（减 1 编码）	0
31:16	-	保留。读取值未定义，只写入 0。	无

36.4.1.7 EEPROM 掉电寄存器（EEPWRDWN—0x0020 0098）

EEPROM 掉电寄存器可以用于将 EEPROM 设备设置为掉电模式。

在一个挂起的 EEPROM 操作过程中，EEPROM 不会进入掉电模式。在清零这个位后，所有 EEPROM 操作都必须挂起 100µs 的时间，等待 EEPROM 唤醒。

表700. EEPROM 掉电/DCM 寄存器位描述（EEPWRDWN—地址 0x0020 0098）

位	符号	描述	复位值
0	PWRDWN	掉电模式位。 0: 不处于掉电模式。 1: 处于掉电模式（会使所有 EEPROM 设备进入掉电模式）。	0
31:1	-	保留。读取值未定义，只写入 0。	无

36.4.2 中断寄存器

这些寄存器控制来自 EEPROM 的中断。

36.4.2.1 中断使能寄存器（EEINTEN—0x0020 0FE4）

表701. 中断使能寄存器位描述（EEINTEN—地址 0x0020 0FE4）

位	符号	描述	复位值
25:0	-	保留。从保留位读取的值未定义。	无
26	EE_RW_DONE	EEPROM 读/写操作结束中断使能位。 该位： —在写 1 到 EEINTENSET 寄存器中的相应位时置位。 —在写 1 到 EEINTENCLR 寄存器中的相应位时清零。	0
27	-	保留。从保留位读取的值未定义。	无
28	EE_PROG_DONE	EEPROM 编程操作已完成中断使能位。 该位： —在写 1 到 EEINTENSET 寄存器的相应位时置位。 —在写 1 到 EEINTENCLR 寄存器的相应位时清零。	0
31:29	-	保留。从保留位读取的值未定义。	无

当对 EEINTSTAT 和 EEINTEN 逐位相与（AND）操作的结果非零时，中断请求输出就会生效。对于 EEPROM 读/写操作已完成中断，最好不要使能这个中断，而仅仅去轮询 EEINSTAT 寄存器中的位。这是因为这些操作的速度相对较快，来不及调用软件中的中断服务子程序。

36.4.2.2 中断使能清零寄存器（EEINTENCLR, 0x0020 0FD8）

表702. 中断使能清零寄存器位描述（EEINTENCLR—地址 0x0020 0FD8）

位	符号	描述	复位值
25:0	-	保留。读取值未定义，只写入 0。	无
26	RDWR_CLR_EN	清零读/写操作已完成中断使能位（EEPROM） 0: 不改变相应的位。 1: 清零相应的位。	0
27	-	保留。 读取值未定义，只写入 0。	无
28	PROG1_CLR_EN	清零 EEPROM 设备 1 的编程操作已完成中断使能位。 0: 不改变相应的位。 1: 清零相应的位。	0
31:29	-	保留。读取值未定义，只写入 0。	无

36.4.2.3 中断使能置位寄存器（EEINTENSET—0x0020 0FDC）

表703. 中断使能置位寄存器位描述（EEINTENSET—地址 0x0020 0FDC）

位	符号	描述	复位值
25:0	-	保留。读取值未定义，只写入 0。	无
26	RDWR_SET_EN	置位读/写操作已完成中断使能位（EEPROM）。 0: 不改变相应的位。 1: 置位相应的位。	0
27	-	保留。读取值未定义，只写入 0。	无
28	PROG1_SET_EN	置位 EEPROM 设备 1 的编程操作已完成中断使能位。 0: 不改变相应的位。 1: 置位相应的位。	0
31:29	-	保留。读取值未定义，只写入 0。	无

36.4.2.4 中断状态寄存器（EEINTSTAT—0x0020 0FE0）

表704. 中断状态寄存器位描述（EEINTSTAT—地址 0x0020 0FE0）

位	符号	描述	复位值
25:0	-	保留。从保留位读取的值未定义。	无
26	END_OF_RDWR	EEPROM 读/写操作已完成中断状态位。 该位： —在写 1 到 EEINTSTATSET 寄存器的相应位且该操作已完成相或(OR)时置位。 —在写 1 到 EEINTSTATCLR 寄存器的相应位时清零。	0
27	-	保留。从保留位读取的值未定义。	无
28	END_OF_PROG1	EEPROM 编程操作已完成中断状态位。 该位： —在写 1 到 EEINTSTATSET 寄存器的相应位且该操作已完成相或(OR)时置位。 —在写 1 到 EEINTSTATCLR 寄存器的相应位时清零。	0
31:29	-	保留。从保留位读取的值未定义。	无

当对 EEINTSTAT 和 EEINTEN 逐位相与 (AND) 操作的结果为非零时，中断请求输出有效。对于 EEPROM 读/写操作已完成中断，最好不要使能这个中断，而仅仅去轮询 EEINSTAT 寄存器中的位。这是因为，这些操作的速度相对较快，来不及调用软件中的中断服务子程序。

36.4.2.5 中断状态清零寄存器（0x0020 0FE8）

表705. 中断状态清零寄存器位描述（EEINTSTATCLR—地址 0x0020 0FE8）

位	符号	描述	复位值
25:0	-	保留。读取值未定义，只写入 0。	无
26	RDWR_CLR_ST	清零读/写操作已完成中断状态位（EEPROM）。 0：不改变相应的位。 1：清零相应的位。	0
27	-	保留。读取值未定义，只写入 0。	无
28	PROG1_CLR_ST	清零 EEPROM 设备 1 的编程操作已完成中断状态位。 0：不改变相应的位。 1：清零相应的位。	0
31:29	-	保留。读取值未定义，只写入 0。	无

36.4.2.6 中断状态置位（EEINTSTATSET—0x0020 0FEC）

表706. 中断状态置位寄存器（EEINTSTATSET—地址 0x0020 0FEC）

位	符号	描述	复位值
25:0	-	保留。读取值未定义，只写入 0。	无
26	RDWR_SET_ST	设置读/写操作已完成中断状态位（EEPROM）。 0：不改变相应的位。 1：置位相应的位。	0
27	-	保留。读取值未定义，只写入 0。	无
28	PROG1_SET_ST	置位 EEPROM 设备 1 的编程操作已完成中断状态位。 0：不改变相应的位。 1：置位相应的位。	0
31:29	-	保留。读取值未定义，只写入 0。	无

36.5 EEPROM 操作

36.5.1 EEPROM 设备描述

EEPROM 是一类非易失性存储器，主要用于存储相对较少量的数据，如存储设置值。

EEPROM 存储器的访问有三种操作方式：读、写、擦除/编程。EEPROM 存储器的“写”操作又可以分成两个单独的操作：写和擦除/编程。本文将第一个操作称为“写”，“写”操作不会真的更新 EEPROM 存储器，而只是更新被称之为“页寄存器”的临时数据寄存器。本文中所说的第二个操作—“擦除/编程”才会真正地更新非易失性存储器，但在此操作之前，需要对页寄存器写入至少 1 个字节和至多 8 个字节。请注意，写入页寄存器的数据没有被“缓存”，因此，在真正地被编入非易失性存储器之前，页寄存器数据是无法读取的。

64 字节的页寄存器的大小等于 EEPROM 存储器中的一页。4kB 的 EEPROM 可以保存 64 个页。

36.5.2 EEPROM 操作

EEPROM 设备无法直接被编程。写数据到存储器和存储器的擦除/编程是两个分开的步骤。页寄存器（64 字节）临时保管写入的数据。一旦需要从 EEPROM 中读取这些数据，或需
要将数据写入另一页，则首先需要将页寄存器的内容编入 EEPROM 存储器。

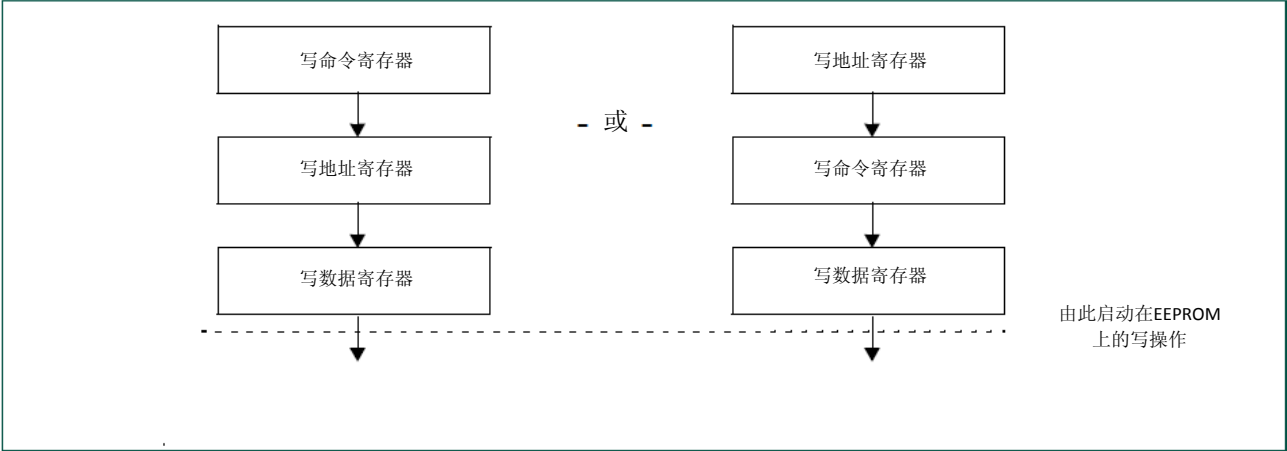
以下小节将对 EEPROM 的操作（读、写和编程）进行更详细的描述。

36.5.2.1 写

EEPROM 控制器支持 8、16 或 32 位元的写入。EEPROM 设备不支持 32 位操作，因此，
控制器将 32 位操作分解成两个 16 位操作。

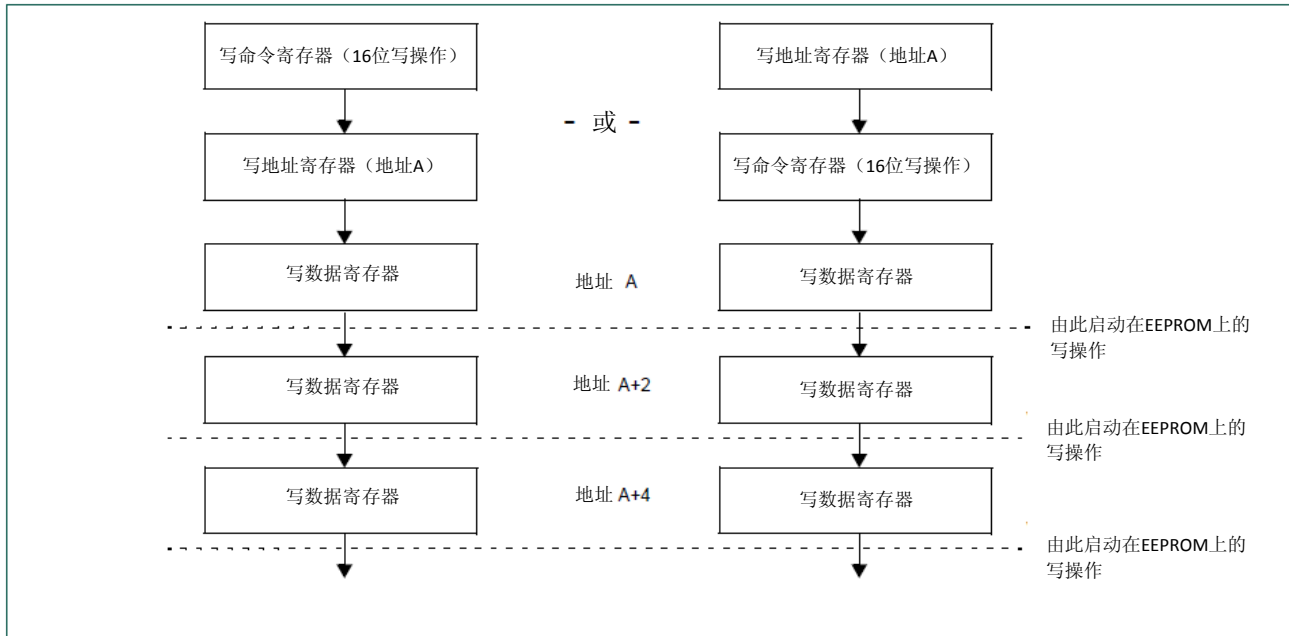
对于写操作，首先要将一个地址写入地址寄存器，并在命令寄存器中选择写操作的类型（可
以按任意顺序完成）。将数据写入到写数据寄存器之后，将自动启动对 EEPROM 设备的写
操作。

图164. 启动一个写操作



写操作会导致一个地址的自动后增，从而允许对页寄存器进行连续写操作而不需要为每个写操作写入一个新地址。当然，地址寄存器可以用其它地址值写入到其它位置。

图165. (16 位) 后递增地址写操作



在写数据寄存器时，如果前一个 **EEPROM** 操作仍处于挂起状态，则通过使就绪信号失效暂停系统总线上的写传输，直到之前的操作完成为止。为避免出现总线上的暂停，在启动写操作之前可以轮询中断状态寄存器，看是否有操作仍处于挂起状态。轮询一般只对以高频运行 (>200MHz) 的系统才有作用。

软件必须确保遵守以下原则：

- 不允许覆写（在擦除/编程操作之前写两次）64 字节页寄存器的某个位置，因为这将导致之前写入的数据丢失（这是 **EEPROM** 模块的特性的必然结果，参见 **EEPROM** 规范）。
- 若使用默认的地址后递增（post-incrementing），则不得跨越页寄存器的上边界。
- 在读取刚写入的数据之前，需将页寄存器的内容编程到非易失性存储器中。
- 对未对齐地址的写操作会对写数据寄存器的写传输产生一个出错响应（例如，对一个非 0x4 倍数地址的 32 位写操作）。该操作将不会被执行。

36.5.2.2 编程

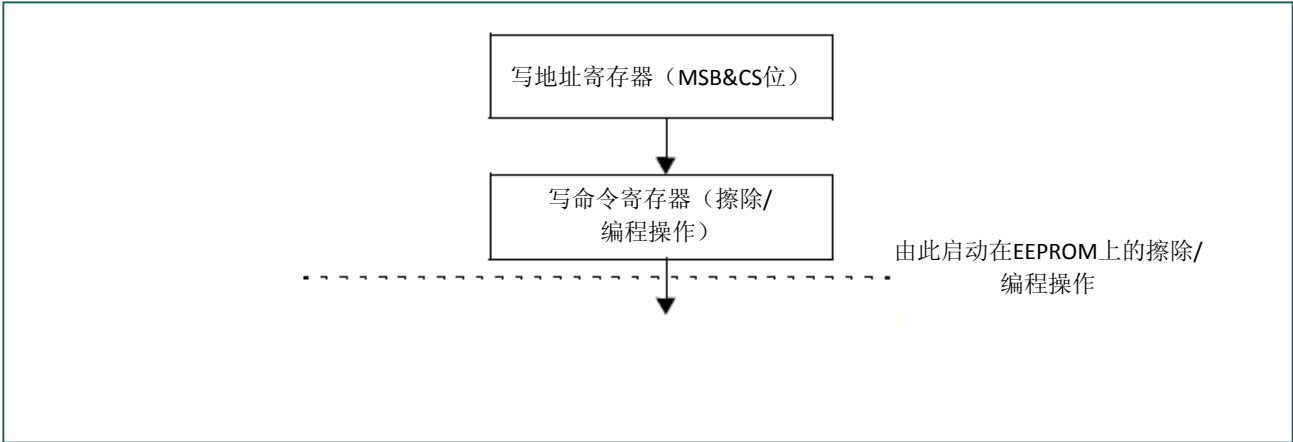
当已经将数据写入页寄存器时，必须将数据编程到非易失性存储器中。这是一个单独的步骤，只对页寄存器执行写操作不会写 **EEPROM** 存储器。

将页寄存器中的内容编程到存储器需要很长的时间，因此可以使能相应的中断或轮询中断状态位，以避免系统总线的暂停。

提供某个地址的 **MSB**（用于选择一个设备存储器的页）和 **CS** 位（用于选择设备），就可以

启动擦除/编程操作。用户“不必理会”6 个 LSB。通过写命令寄存器（选择擦除/编程操作）就可以开始操作。在开始一个编程操作之前，应轮询 EEPROM 状态，以确保前一个写操作已经完成。

图166. 编程一个页寄存器的内容到存储器

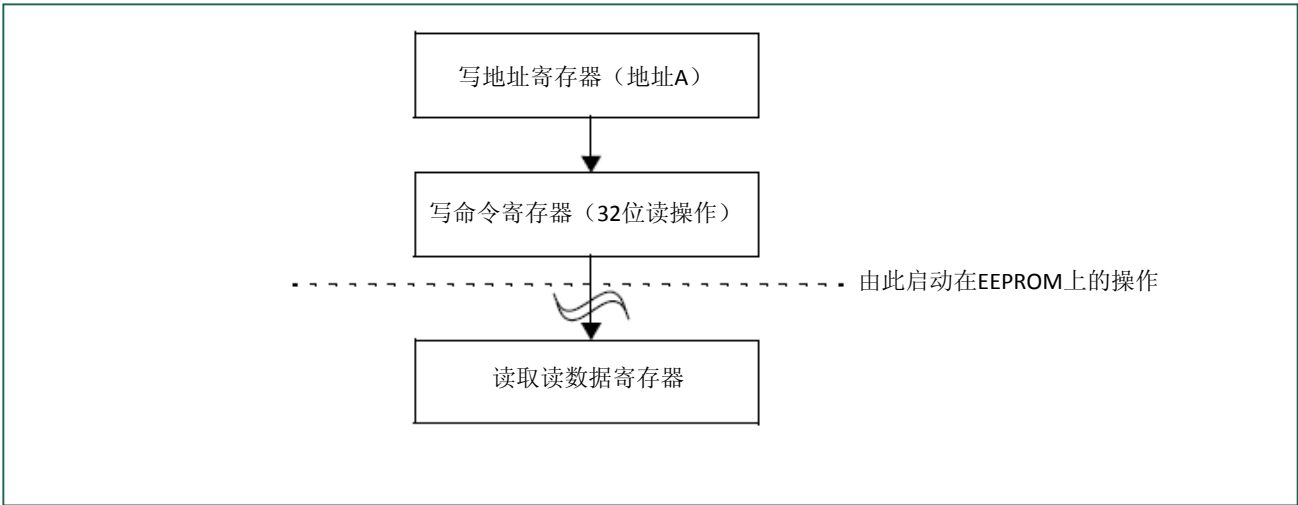


36.5.2.3 读

EEPROM 控制器支持读取 8、16 或 32 位元。EEPROM 设备不支持 32 位操作，因此，控制器将该操作分解成两个 16 位操作。

对于读操作，首先要向地址寄存器写入一个地址。然后要在命令寄存器中选择操作类型。写命令寄存器将自动启动对 EEPROM 设备的读操作。

图167. 启动一个读操作（从地址 A 的 32 位读操作）



如果在某个读操作处于挂起状态时就读取读数据寄存器，则通过使就绪信号失效来暂停系统总线上的读传输，直到之前的读操作完成。为了避免总线上的操作暂停，可以在读取读数据寄存器之前轮询中断状态寄存器来了解是否有处于挂起状态的操作。

读操作会自动地对地址寄存器做后递增,这样就可以对 EEPROM 存储器进行连续的读操作而无需为每一次读操作都写入一个新地址。置位命令寄存器中的读数据预取位后,读取读数据寄存器将自动启动从下一地址(递增的)处开始的一个读操作。当以这种方式做连续的读操作时,第一个读操作的开始就是写命令寄存器的结果。其后读操作的启动都源于对读数据寄存器的读取,以获取之前读操作的结果。

对未对齐地址的读操作会对命令寄存器的写传输产生一个出错响应(例如,对一个非 0x4 倍数地址的 32 位读操作)。该操作将不会被执行。

36.5.2.4 出错响应

控制器可以在下述情形中,生成以下与 EEPROM 有关的出错响应:

- 对受保护页的擦除/编程操作会导致对命令寄存器写传输的一个出错响应。通过设置 OVP 输入信号来撤销对受保护页的保护可以避免这一出错响应的产生。
- 写一个只读寄存器或读一个只写寄存器都将产生一个出错响应。
- 向不存在的寄存器位置发起的传输将会产生一个出错响应。

37.1 简介

引导加载器（boot loader）控制芯片复位后的初始化操作，并提供对 Flash 存储器进行编程的工具。引导代码可以对空片进行初始编程、对事先已编程的芯片进行擦除和再编程，或者通过运行系统中的应用程序对 Flash 存储器进行编程。

37.2 特性

- 在系统编程：在系统编程（ISP）是通过引导加载器软件和 UART0 串口对片上 Flash 存储器进行编程或再编程的方法。这种方法也可以在芯片位于终端用户板的时候使用。
- 在应用编程：在应用编程（IAP）是依据终端用户的应用代码对片上 Flash 存储器进行擦除/编程操作。
- Flash 符号差生成：内置硬件可以为一段 Flash 地址或整个 Flash 存储器生成符号差。

37.3 描述

Flash 引导加载器代码在芯片每次上电或复位后最先执行。加载器可以执行 ISP 命令处理器或用户的应用代码。复位之后，P2[10]引脚的低电平可被视作启动使用 UART0 管脚 P0[2]（U0_TXD）和 P0[3]（U0_RXD）的 ISP 命令处理器的外部硬件请求。假定在 $\overline{\text{RESET}}$ 管脚出现上升沿时，电源管脚出现标称电平，那么在采样 P2[10]之前有 3ms 的时间来决定是执行用户代码还是 ISP 处理器。如果 P2[10]取样得到低电平且看门狗溢出标志置位，那么启动 ISP 命令处理器的外部硬件请求将被忽略。在没有 ISP 命令处理器执行的请求（P2[10]管脚在复位后采样为高电平）时，将搜索有效的用户程序。若发现有效的用户程序，执行控制权就被转移给该用户程序。若没有找到有效的用户程序，就将调用自动波特率程序。

管脚 P2[10]被用作 ISP 的硬件请求信号，因此需要特别注意：由于 P2[10]在复位后处于高阻抗模式，用户需要提供外部硬件（上拉电阻或其它器件）来保持该管脚的状态稳定。否则可能会误进入 ISP 模式。

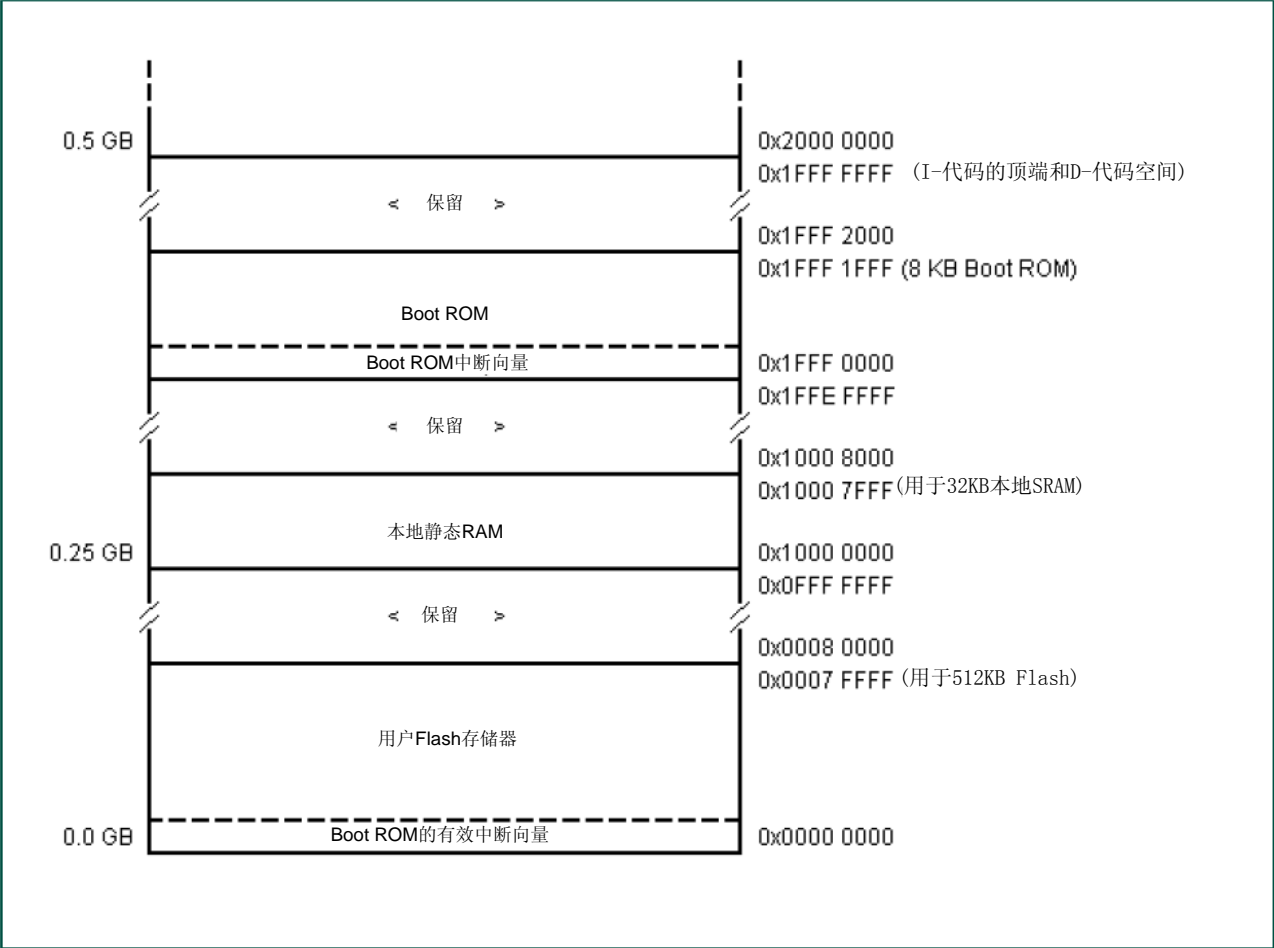
上电复位后，一旦处理器进入 ISP 模式，CPU 和外设就使用 12MHz 的 IRC 频率。在这种情况下就可以轻松获得以下波特率：9600 波特、19200 波特、38400 波特、57600 波特、115200 波特和 230400 波特。当 ISP 被用户应用程序调用的时候，情况可能就不一样了（参见 [37.8.9 节](#)）。

Flash 存储器内置有硬件 Flash 符号差生成功能，可使用这一特性来生成符号差，用来核实 Flash 内容。有关 Flash 符号差生成的更多内容，参见 37.10 节。

37.3.1 复位后的存储器映射

复位后，在用户程序开始运行时，中断向量被设置成指向 Flash 存储器的开头。

图168. 低存储器影射



37.3.1.1 有效用户代码的判定标准

保留的 Cortex-M3 向量单元（除向量单元 7 以外，向量表中的偏移量为 0x001C）应包含表项 0~6 的校验和的 2 的补码，这样就使前 8 个表项的校验和为 0。引导加载器代码首先计算 Flash 扇区 0 中前 8 个向量单元的校验和。如果结果为 0，将控制权转移给用户代码。

如果符号差无效，那么自动波特率程序通过串口 0 与主机进行同步。主机应当发送一个同步字符“?” (0x3F) 并等待响应。主机的串口应设定为 8 个数据位、1 个停止位和无奇偶校验。自动波特率程序根据自身的频率测量接收到的同步字符的位时间并对串口波特率发生器进行编程。它还向主机发送一个 ASCII 字符串 (“Synchronized<CR><LF>”)。作为响应，主机应当发送与接收到的字符串相同的字符串 (“Synchronized<CR><LF>”)。自动波特率程序通过观察接收到的字符来验证是否同步。如果通过同步验证，则向主机发送字符串 “OK<CR><LF>”。主机应当通过发送正在运行部分的晶振频率（单位：KHz）作为响应。例如，如果运行部分的晶振频率为 10MHz，主机的响应应当为“10000<CR><LF>”。在接收到晶振频率后再向主机发送字符串“OK<CR><LF>”。如果同步验证没有通过，那么自动波特率程序再次等待一个同步字符。在用户调用 ISP 的情况下要使自动波特率正确工作，CCLK 频率应当大于或等于 10MHz。

有关复位、PLL 和启动/引导代码之间相互作用的详细信息，参见 [4.5.1](#) 节。

在接收到晶振频率后，执行初始化并调用 ISP 命令处理器。出于安全方面的考虑，在执行 Flash 编程/擦除操作命令和“Go”命令之前必须执行“Unlock（解锁）”命令。其它命令不需要执行解锁命令。每次 ISP 命令处理都要执行一次“Unlock（解锁）”命令。有关解锁命令的详细信息，参见 [37.7](#) 节。

37.3.2 通信协议

所有 ISP 命令都以单个 ASCII 字符串形式发送。字符串应当以回车 (CR) 和/或换行 (LF) 控制字符作为结束符。多余的<CR>和<LF>将被忽略。所有 ISP 响应都是以<CR><LF>结束的 ASCII 字符串形式发送。数据是以 UU 编码格式发送和接收。

37.3.2.1 ISP 命令格式

“命令参数_0 参数_1...参数_n<CR><LF>”“数据”（只适用于写命令）

37.3.2.2 ISP 响应格式

“返回_代码<CR><LF>响应_0<CR><LF>响应_1<CR><LF>...响应_n<CR><LF>”“数据”（只适用于读命令）

37.3.2.3 ISP 数据格式

数据流采用 UU 编码格式。UU 编码算法将 3 字节二进制数据转换成 4 字节可打印的 ASCII 字符集，该编码的效率高于 Hex 格式 (Hex 格式将 1 字节二进制数据转换成 2 字节 ASCII Hex 数据)。发送器在发送 20 个 UU 编码行之后发送校验和。任何 UU 编码行的长度都不应超过 61 个字符（字节），也就是说它可以保持 45 个数据字节。接收器应当将该校验和与接收数据的校验和进行比较。如果校验和匹配，接收器响应“OK<CR><LF>”，并等待下一次发送。如果校验和不匹配，接收器响应“RESEND<CR><LF>”。作为响应，发送器还应当将字节重新发送。

37.3.2.4 ISP 流程控制

软件 XON/XOFF 流控制机制可防止缓冲区溢出时的数据丢失。当数据快速到达时，发送 ASCII 控制字符 DC3 (0x13) 停止数据流。发送 ASCII 控制字符 DC1 (0x11) 恢复数据流。主机也应支持相同的流控制机制。

37.3.2.5 ISP 命令中止

命令可通过发送 ASCII 控制字符“ESC” (0x1B) 来中止。该特性在“ISP 命令”一节中并没有作为一个命令。一旦接收到“escape”代码，ISP 命令处理器就会等待一个新的命令。

37.3.2.6 IAP 过程中的中断

在擦除/编程操作过程中，片上 Flash 存储器不可访问。只有当用户应用代码启动执行时，用户 Flash 区中的中断向量才有效。在调用 Flash 擦除/写 IAP 之前，用户应当禁止中断或确保用户中断向量在 RAM 中有效和中断处理程序位于 RAM 中。IAP 代码不能使用或禁止中断。

37.3.2.7 ISP 命令处理器使用的 RAM

ISP 命令使用片上地址 0x1000 0118 到 0x1000 01FF 范围内的 RAM。用户可以使用该区域，但是在复位时内容可能会丢失。Flash 编程命令使用片上 RAM 最顶端的 32 字节。堆栈位于 RAM 顶端－32。可使用的最大堆栈为 256 字节，堆栈是向下递加的。

37.3.2.8 进入用户程序之前引导过程所使用的 RAM

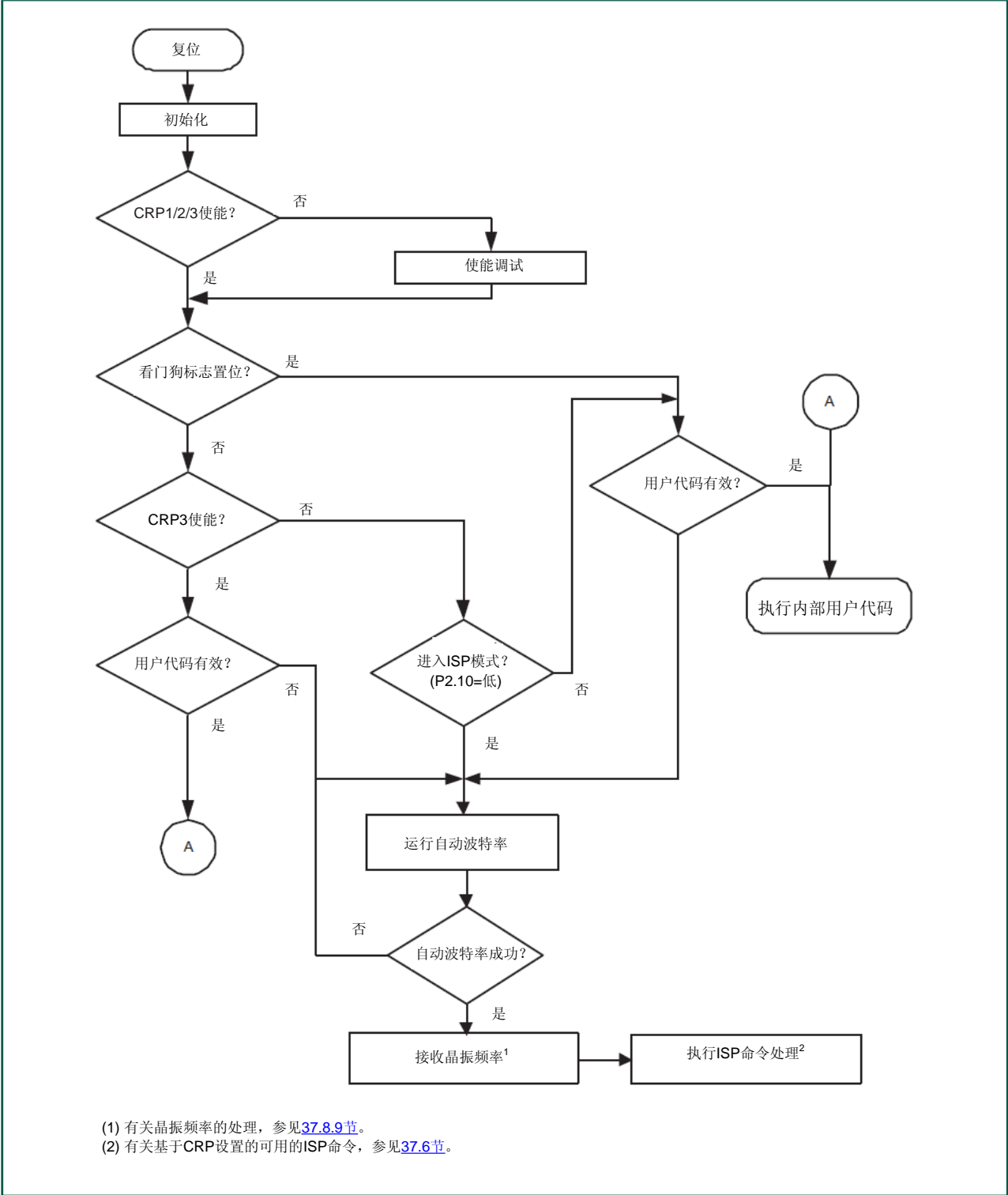
在芯片复位之后，引导程序使用的是用于 ISP 命令处理程序的 RAM 的一个子集。包括位置向量 0x1000 0120 以及片上 RAM 顶部的 32 字节的一部分。堆栈位于 RAM 顶端－32。可使用的最大堆栈为 32 字节。如果用户程序假定 RAM 在设备电源没有拆除而进行复位的情况下保持不变，那么意识到这些异常情况是很重要的。

37.3.2.9 IAP 命令处理器使用的 RAM

Flash 编程命令使用片上 RAM 最顶端的 32 字节。在用户分配的堆栈空间内可使用的最大堆栈为 128 字节，堆栈是向下递加的。

37.4 引导过程流程图

图169. Boot 过程流程图



37.5 扇区号

有些 IAP 和 ISP 命令以“扇区”为单位进行操作，同时还指定扇区号。下表列出了 178x/177x 系列微控制器设备的扇区分布图（扇区号和存储器地址之间的对应关系），这些设备分别包含 32、64、128、256 和 512KB Flash。IAP 和 ISP 程序都位于 Boot ROM。

表707. LPC178x/177x 系列微控制器设备中的扇区

扇区号	扇区规格 (KB)	起始地址	结束地址	32KB 器件	64KB 器件	128KB 器件	256KB 器件	512KB 器件
0	4	0X0000 0000	0X0000 0FFF	x	x	x	x	x
1	4	0X0000 1000	0X0000 1FFF	x	x	x	x	x
2	4	0X0000 2000	0X0000 2FFF	x	x	x	x	x
3	4	0X0000 3000	0X0000 3FFF	x	x	x	x	x
4	4	0X0000 4000	0X0000 4FFF	x	x	x	x	x
5	4	0X0000 5000	0X0000 5FFF	x	x	x	x	x
6	4	0X0000 6000	0X0000 6FFF	x	x	x	x	x
7	4	0X0000 7000	0X0000 7FFF	x	x	x	x	x
8	4	0X0000 8000	0X0000 8FFF		x	x	x	x
9	4	0X0000 9000	0X0000 9FFF		x	x	x	x
10 (0x0A)	4	0X0000 A000	0X0000 AFFF		x	x	x	x
11 (0x0B)	4	0X0000 B000	0X0000 BFFF		x	x	x	x
12 (0x0C)	4	0X0000 C000	0X0000 CFFF		x	x	x	x
13 (0x0D)	4	0X0000 D000	0X0000 DFFF		x	x	x	x
14 (0X0E)	4	0X0000 E000	0X0000 EFFF		x	x	x	x
15 (0x0F)	4	0X0000 F000	0X0000 FFFF		x	x	x	x
16 (0x10)	32	0x0001 0000	0x0001 7FFF			x	x	x
17 (0x11)	32	0x0001 8000	0x0001 FFFF			x	x	x
18 (0x12)	32	0x0002 0000	0x0002 7FFF				x	x
19 (0x13)	32	0x0002 8000	0x0002 FFFF				x	x
20 (0x14)	32	0x0003 0000	0x0003 7FFF				x	x
21 (0x15)	32	0x0003 8000	0x0003 FFFF				x	x
22 (0x16)	32	0x0004 0000	0x0004 7FFF					x
23 (0x17)	32	0x0004 8000	0x0004 FFFF					x
24 (0x18)	32	0x0005 0000	0x0005 7FFF					x
25 (0x19)	32	0x0005 8000	0x0005 FFFF					x
26 (0x1A)	32	0x0006 0000	0x0006 7FFF					x
27 (0x1B)	32	0x0006 8000	0x0006 FFFF					x
28 (0x1C)	32	0x0007 0000	0x0007 7FFF					x
29 (0x1D)	32	0x0007 8000	0x0007 FFFF					x

37.6 代码读保护（CRP）

代码读保护机制允许用户使能系统中的不同安全级别以便访问片上 Flash 和限制 ISP 的使用。需要时，可通过在 Flash 地址单元 0x0000 02FC 编程特定的格式来调用 CRP。IAP 命令不受代码读保护的影响。

重要提示：仅当设备经过一个电源周期后，CPR 的改变才有效。

表708. 代码读保护选项

名称	在 0x0000 02FC 处编程的格式	描述
CRP1	0x1234 5678	禁止通过 JTAG 管脚访问芯片。该模式允许使用下面 ISP 命令和约束来进行部分 Flash 更新： 写 RAM 命令不能访问在 0x1000 0200 以下的 RAM。这是由于 ISP 代码使用了 RAM 的缘故，参见 37.3.2.7 节。 <ul style="list-style-type: none">读存储器命令：禁止。将 RAM 内容复制到 Flash 命令：不能写扇区 0。运行命令：禁止。擦除扇区命令：能擦除扇区 0 以外的任一扇区，或能马上擦除所有扇区。比较命令：禁止。 当需要 CRP 且更新 Flash 字段时可使用该模式，但是不能在擦除所有扇区时使用该模式。由于在 Flash 部分更新的情况下比较命令被禁止，因此第二装载程序应执行校验和机制来验证 Flash 的完整性。
CRP2	0x8765 4321	除了以下附加内容外与 CRP1 相似： <ul style="list-style-type: none">写 RAM 命令：禁止。将 RAM 内容复制到 Flash：禁止。擦除命令：只允许擦除所有扇区。
CRP3	0x4321 8765	与 CRP2 相似，但如果 Flash 扇区 0 中存在有效的用户代码，ISP 入口通过拉低 P2[10]禁能。该模式通过 P2[10]管脚有效禁止了 ISP 重写。用户的应用程序可决定是调用 IAP 来进行 Flash 更新还是通过 UART0 重新调用 ISP 命令来进行 Flash 更新。 注：如果选择了 CRP3,则不再对该设备执行更多的工厂测试。

表709. 代码读保护硬件/软件的相互作用

CRP	用户代码是否有效	复位时 P2[10] 管脚的电平	JTAG 是否使能	LPC178x/177 系列微控制器是否进入 ISP 模式	在 ISP 模式中的部分 Flash 更新
None	否	X	是	是	是
	是	高	是	否	无
	是	低	是	是	是
CRP1	否	x	否	是	是
	是	高	否	否	无
	是	低	否	是	是
CRP2	否	x	否	是	否
	是	高	否	否	无
	是	低	否	是	否
CRP3	否	x	否	是	否
	是	x	否	否	无

如果使能了任意一个 CRP 模式并允许通过 ISP 访问芯片，那么不支持的或受到限定的 ISP 命令将通过返回代码 CODE_READ_PROTECTION_ENABLED 终止。

37.7 ISP 命令

下面的命令是 ISP 命令处理器所接受的命令。每个命令都支持具体的状态代码。当接收到未定义命令时，命令处理器会发送返回代码 `INVALID_COMMAND`。命令和返回代码为 ASCII 格式。

只有当接收到的 ISP 命令执行完毕时，ISP 命令处理器才会发送 `CMD_SUCCESS`，这时主机才能发送新的 ISP 命令。但“设置波特率”、“写 RAM”、“读存储器”和“运行”命令除外。

表710. ISP 命令汇总

ISP 命令	用法	描述
解锁	U <解锁代码>	见表 711
设置波特率	B <波特率> <停止位>	见表 712
回应	A <设定>	见表 714
写 RAM	W <起始地址> <字节数>	见表 715
读存储器	R <地址> <字节数>	见表 716
准备写操作的扇区	P <起始扇区号> <结束扇区号>	见表 717
复制 RAM 内容到 Flash	C <Flash 地址> <RAM 地址> <字节数>	见表 718
运行	G <地址> <模式>	见表 719
擦除扇区	E <起始扇区号> <结束扇区号>	见表 720
扇区查空	I <起始扇区号> <结束扇区号>	见表 721
读器件 ID	J	见表 722
读引导代码版本	K	见表 724
读序列号	N	见表 725
比较	M <地址 1> <地址 2> <字节数>	见表 726

37.7.1 解锁<解锁代码>

表711. ISP 解锁命令

命令	U
输入	解锁代码：23130 ₁₀
返回代码	CMD_SUCCESS INVALID_CODE PARAM_ERROR
描述	该命令用来解锁 Flash 写、擦除及运行命令。
举例	“U 23130<CR><LF>”解锁 Flash 写/擦除&运行命令。

37.7.2 设置波特率<波特率><停止位>

表712. ISP 设置波特率命令

命令	B
输入	波特率：9600 19200 38400 57600 115200 230400 停止位：1 2
返回代码	CMD_SUCCESS INVALID_BAUD_RATE INVALID_STOP_BIT PARAM_ERROR
描述	该命令用来改变波特率。新的波特率在命令处理器发送 CMD_SUCCESS 返回代码之后生效。
举例	“B 57600 1<CR><LF>”设置串口波特率 57600bps 和 1 个停止位。

当 ISP 被用户程序重新调用后使用 Set Baud Rate（设置波特率）命令时（使用“Re-invoke ISP（重新调用 ISP）”IAP 命令，见 37.8.9 节），波特率将基于当前外设时钟频率来设置。表 713 给出了频率和其所支持波特率的例子。表中给出了波特率误差小于±1.5%的例子。这为通过 IRC 生成的频率（其本身就有±1%的偏差）提供了富余空间，可以在所有情况下实现总波特率误差小于±2.5%。

表713. 支持的 ISP 波特率和 PCLK 频率示例（单位：MHz）

ISP 波特率 和 PCLK 频率	9600	19200	38400	57600	115200	230400
10.0000	+	+		+		
11.0592	+	+	+	+	+	+
12.0000 ^[1]	+	+		+		
12.2880	+	+	+			
14.7456	+	+	+	+	+	+
15.3600	+	+	+			
18.4320	+	+	+	+	+	+
19.6608	+	+	+			
24.5760	+	+	+	+		
25.0000	+	+	+	+		

[1] 芯片复位后 ISP 入口用片上 IRC 在 PCLK=12MHz 时运行设备。

37.7.3 回应<设定>

表714. ISP 回应命令

命令	A
输入	设定：打开=1 关闭=0
返回代码	CMD_SUCCESS PARAM_ERROR
描述	回应命令的默认设定是打开。当打开时,ISP 命令处理器将收到的串行数据发送回主机。
举例	“A 0<CR><LF>”回应关闭。

37.7.4 写 RAM<起始地址><字节数>

主机应当仅在接收到 CMD_SUCCESS 返回代码后才发送数据并在发送完 20 个 UU 编码行之后发送校验和。校验和在增加新数据（在 UU 编码前）字节时产生并在发送完 20 个 UU 编码行后重新设置。任何 UU 编码行的长度不应超过 61 个字符（字节），即可以保持 45 个数据字节。当数据少于 20 个 UU 编码行时，校验和按照实际发送的字节数进行计算。ISP 命令处理器将它与接收字节的校验和进行比较。如果校验和匹配，那么 ISP 命令处理器响应“OK<CR><LF>”，并等待下一次发送。如果校验和不匹配，接收器响应“RESEND<CR><LF>”。作为响应，主机应当将字节重新发送。

表715. ISP 写 RAM 命令

命令	W
输入	起始地址： 被写入数据字节的 RAM 地址。该地址应以字为边界。 字节数： 写入的字节数。计数值应当为 4 的倍数。
返回代码	CMD_SUCCESS ADDR_ERROR （地址不是以字为边界） ADDR_NOT_MAPPED COUNT_ERROR （字节计数值不是 4 的倍数） PARAM_ERROR CODE_READ_PROTECTION_ENABLED
描述	该命令用于将数据下载到 RAM。数据应当为 UU 编码格式。当 CRP2 或 CRP3 被使能时该命令被禁止。
举例	“W 268435968 4<CR><LF>”向地址 0x1000 0200 写入 4 个字节数据。

37.7.5 读存储器<地址><字节数>

数据流之后是命令成功返回代码。校验和在发送完 20 个 UU 编码行之后发送。校验和在增加新数据（UU 编码前）字节时产生，发送完 20 个编码行后重新设置。任何 UU 编码行的长度都不应超过 61 个字符（字节），即它可以保持 45 个数据字节。当数据少于 20 个 UU 编码行时，校验和按照实际发送的字节数进行计算。主机将它与接收字节的校验和进行比较。如果校验和匹配，那么主机响应“OK<CR><LF>”，并等待下一次发送。如果校验和不匹配，主机响应“RESEND<CR><LF>”。作为响应，ISP 命令处理器应当重新发送字节。

表716. ISP 读存储器命令

命令	R
输入	起始地址：待读出数据字节的地址。该地址应以字为边界。 字节数：待读出的字节数。计数值应当为 4 的倍数。
返回代码	CMD_SUCCESS，后面是<实际数据（UU 编码）> ADDR_ERROR （地址不是以字为边界） ADDR_NOT_MAPPED COUNT_ERROR （字节计数值不是 4 的倍数） PARAM_ERROR CODE_READ_PROTECTION_ENABLED
描述	该命令用于读出 RAM 或 Flash 存储器的数据。当 CRP 被使能时该命令被禁止。
举例	“R 268435968 4<CR><LF>”从地址 0x1000 0200 读出 4 个字节数据。

37.7.6 准备写操作的扇区<起始扇区号> <结束扇区号>

该命令使 Flash 写/擦除操作分成两个步骤处理。

表717. ISP 准备写操作的扇区命令

命令	P
输入	起始扇区号 结束扇区号：应大于或等于起始扇区号。
返回代码	CMD_SUCCESS BUSY INVALID_SECTOR PARAM_ERROR
描述	该命令必须在执行“复制 RAM 内容到 Flash”或“擦除扇区”命令前执行。这两个命令的成功执行使得相关扇区再次被保护。要准备单个扇区，可将起始和结束扇区号设置为相同值。
举例	“P 0 0<CR><LF>”准备 Flash 扇区 0。

37.7.7 将 RAM 内容复制到 Flash<Flash 地址><RAM 地址><字节数>

表718. ISP 复制命令

命令	C
输入	Flash 地址（DST） ：待写入数据字节的目标 Flash 地址。目标地址的边界应为 256 字节。 RAM 地址（SRC） ：待读出数据字节的源 RAM 地址。 字节数 ：待写入的字节数。应当为 256 512 1024 4096。
返回代码	CMD_SUCCESS SRC_ADDR_ERROR（地址不是以字为边界） DST_ADDR_ERROR（地址边界错误） SRC_ADDR_NOT_MAPPED DST_ADDR_NOT_MAPPED COUNT_ERROR（字节计数值不是 256 512 1024 4096） SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION BUSY CMD_LOCKED PARAM_ERROR CODE_READ_PROTECTION_ENABLED
描述	该命令用来编程 Flash 存储器。“准备写操作的扇区”命令应当在该命令前被执行。当成功执行复制命令后，受影响的扇区将自动再次受到保护。当 CRP2 或 CRP3 被使能时该命令被禁止。当 CRRP1 使能时，扇区 0 以外的扇区能被写入。
举例	“C 0 268468224 512<CR><LF>”将 RAM 地址 0x1000 8000 开始的 512 字节复制到 Flash 地址 0。

37.7.8 运行<地址><模式>

表719. ISP 运行命令

命令	G
输入	地址 ：代码执行起始的 Flash 或 RAM 地址。该地址应以字为边界。 模式（为向后兼容性保留） ：T（执行 Thumb 模式下的程序） A（不允许）。
返回代码	CMD_SUCCESS ADDR_ERROR ADDR_NOT_MAPPED CMD_LOCKED PARAM_ERROR CODE_READ_PROTECTION_ENABLED
描述	该命令用来执行位于 RAM 或 Flash 存储器中的程序。一旦该命令成功执行，就有可能不再返回 ISP 命令处理器。当代码读保护使能时该命令被禁止。
举例	“G 0 T<CR><LF>”跳转到地址 0x0000 0000。

在使用 GO 命令时，从指定地址（假定它是一个可执行地址）开始执行且设备处于之前针对 ISP 代码进行的配置模式。也就是说，这和在芯片复位后直接输入用户代码是有所不同的。在 PLL 关闭的情况下，CPU 将使用 12MHz IRC 运行。

37.7.9 擦除扇区<起始扇区号><结束扇区号>

表720. ISP 擦除扇区命令

命令	E
输入	起始扇区号 结束扇区号：应大于或等于起始扇区号。
返回代码	CMD_SUCCESS BUSY INVALID_SECTOR SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION CMD_LOCKED PARAM_ERROR CODE_READ_PROTECTION_ENABLED
描述	该命令用来擦除片上 Flash 存储器的一个或多个扇区。当 CRP3 使能时该命令被禁止。 当 CRP1 使能时，扇区 0 以外的扇区能被擦除。在 CRP1 和 CRP2 中，所有片区可马上被擦除。
举例	"E 2 3<CR><LF>"擦除 Flash 扇区 2 和 3。

37.7.10 扇区查空<扇区号><结束扇区号>

表721. ISP 扇区查空命令

命令	I
输入	起始扇区号： 结束扇区号：应大于或等于起始扇区号。
返回代码	CMD_SUCCESS SECTOR_NOT_BLANK（后跟<第一个非空字的偏移量> <非空字的内容>） INVALID_SECTOR PARAM_ERROR
描述	该命令用于对片上 Flash 存储器的一个或多个扇区进行查空。
举例	"I 2 3<CR><LF>"对 Flash 扇区 2 和 3 进行查空。

37.7.11 读器件标识号

表722. ISP 读器件 ID 命令

命令	J
输入	无。
返回代码	CMD_SUCCESS 后跟 ASCII 格式的器件标识号（详见表 723）。
描述	该命令用来读取器件的标识号。标识号映射了设备族内的特征子集。标识号通常不会因为技术上的修改而改变。

表723. LPC178x/177x 系列微控制器器件标识号

器件	ASCII/dec 编码	Hex 编码
LPC1788	673005383	0x281D 3F47
LPC1787	673003335	0x281D 3747
LPC1786	672997187	0x281D 1F43
LPC1785	672995139	0x281D 1743
LPC1778	655966023	0x2719 3F47
LPC1777	655963975	0x2719 3747
LPC1776	655957827	0x2719 1F43
LPC1774	654381362	0x2701 1132

37.7.12 读引导代码版本号

表724. ISP 读取引导代码版本号命令

命令	K
输入	无
返回代码	CMD_SUCCESS 后跟 2 字节 ASCII 格式的引导代码版本号。将其解释为<字节 1（主）>.<字节 0（次）>。
描述	该命令用来读取器件的引导代码版本号。

37.7.13 读设备序列号

表725. ISP 读器件序列号命令

命令	N
输入	无
返回代码	CMD_SUCCESS 后跟 4 个十进制的 ASCII 组的设备序列号，每个代表一个 32 位值。
描述	该命令用来读取设备序列号。该序列号可用来在所有 LPC178x/177x 系列微控制器中识别一个设备。

37.7.14 比较<地址 1> <地址 2> <字节数>

表726. ISP 比较命令

命令	M
输入	<p>地址 1（DST）：要比较的数据字节的起始 Flash 或 RAM 地址。该地址应以字为边界。</p> <p>地址 2（SRC）：要比较的数据字节的起始 Flash 或 RAM 地址。该地址应以字为边界。</p> <p>字节数：待比较的字节数；应为 4 的倍数。</p>
返回代码	CMD_SUCCESS （源和目标数据相同） COMPARE_ERROR （后跟第一个不匹配字节的偏移量） COUNT_ERROR （字节计数值不是 4 的倍数） ADDR_ERROR ADDR_NOT_MAPPED PARAM_ERROR
描述	该命令用来比较两个地址的存储器内容。当 CRP 使能时该命令被禁止。
举例	“M 8192 268435968 4<CR><LF>”将 RAM 地址 0x1000 0200 开始的 4 个字节与 Flash 地址 0x2000 开始的 4 个字节进行比较。

37.7.15ISP 返回代码

表727. ISP 返回代码汇总

返回代码	符号	描述
0	CMD_SUCCESS	命令被成功执行。只有当由主机发出的命令被成功执行结束后，才会由 ISP 处理器发送。
1	INVALID_COMMAND	无效命令。
2	SRC_ADDR_ERROR	源地址没有以字为边界。
3	DST_ADDR_ERROR	目标地址的边界错误。
4	SRC_ADDR_NOT_MAPPED	源地址没有在存储器映射中被映射。计数值必须考虑到可用性。
5	DST_ADDR_NOT_MAPPED	目标地址没有在存储器映射中被映射。计数值必须考虑到可用性。
6	COUNT_ERROR	字节计数值不是 4 的倍数或是一个非法值。
7	INVALID_SECTOR	扇区号无效或结束扇区号小于起始扇区号。
8	SECTOR_NOT_BLANK	扇区非空。
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	为写操作准备扇区的命令未执行。
10	COMPARE_ERROR	源和目标数据不相等。
11	BUSY	Flash 编程硬件接口忙。
12	PARAM_ERROR	参数不足或无效参数。
13	ADDR_ERROR	地址没有以字为边界。
14	ADDR_NOT_MAPPED	地址没有在存储器映射中被映射。计数值必须考虑到可用性。
15	CMD_LOCKED	命令被锁定。
16	INVALID_CODE	解锁代码无效。
17	INVALID_BAUD_RATE	无效波特率设置。
18	INVALID_STOP_BIT	无效停止位设置。
19	CODE_READ_PROTECTION_ENABLED	代码读保护使能。

37.8 IAP 命令

对于在应用编程来说,应当通过寄存器 R0 中的字指针指向存储器 (RAM) 包含的命令代码和参数来调用 IAP 程序。IAP 命令的结果返回到寄存器 R1 所指向的返回表。用户可通过传递寄存器 R0 和 R1 中的相同指针重用命令表来得到结果。参数表应当大到足够保存所有的结果以防结果的数目大于参数的数目。参数传递见 [图 170](#)。参数和结果的数目根据 IAP 命令而有所不同。参数的最大数目为 5,由“将 RAM 内容复制到 Flash(Copy RAM to Flash)”命令传递。结果的最大数目为 4,由“读取设备序列号 (Read device serial numbe)”命令返回。命令处理器在接收到一个未定义的命令时,发送状态代码 INVALID_COMMAND。IAP 程序位于地址 0x1FFF 1FF0。

可用下面的 C 代码来调用 IAP 功能。

定义 IAP 程序的入口地址。由于 IAP 地址的位 0 被置位,因为 Cortex-M3 只使用 Thumb 模式。

```
#define IAP_LOCATION 0x1FFF1FF1
```

定义数据结构或指针,将 IAP 命令表和结果表传递给 IAP 函数:

```
unsigned long command[5];  
unsigned long output[5];
```

或

```
unsigned long * command;  
unsigned long * output;  
command= (unsigned long *) 0x... output= (unsigned long *) 0x...
```

定义函数类型指针,函数包含 2 个参数,无返回值。注意,IAP 返回结果和表格在 R1 中的基地址。

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

设置函数指针

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

使用下面的语句来调用 IAP:

```
iap_entry (command, output);
```

使用 ADS (ARM 开发套件) 中 ARM 连接器所支持的符号定义文件特性,可以进一步简化 IAP 的调用。用户还可使用汇编代码来调用 IAP 程序。

注: 命令表中的第一个表项是 IAP 命令,后面跟着所要求的命令参数,从 Param0 开始。输入表的第一个表项是返回代码,后面跟着其它结果,从 Result0 开始。

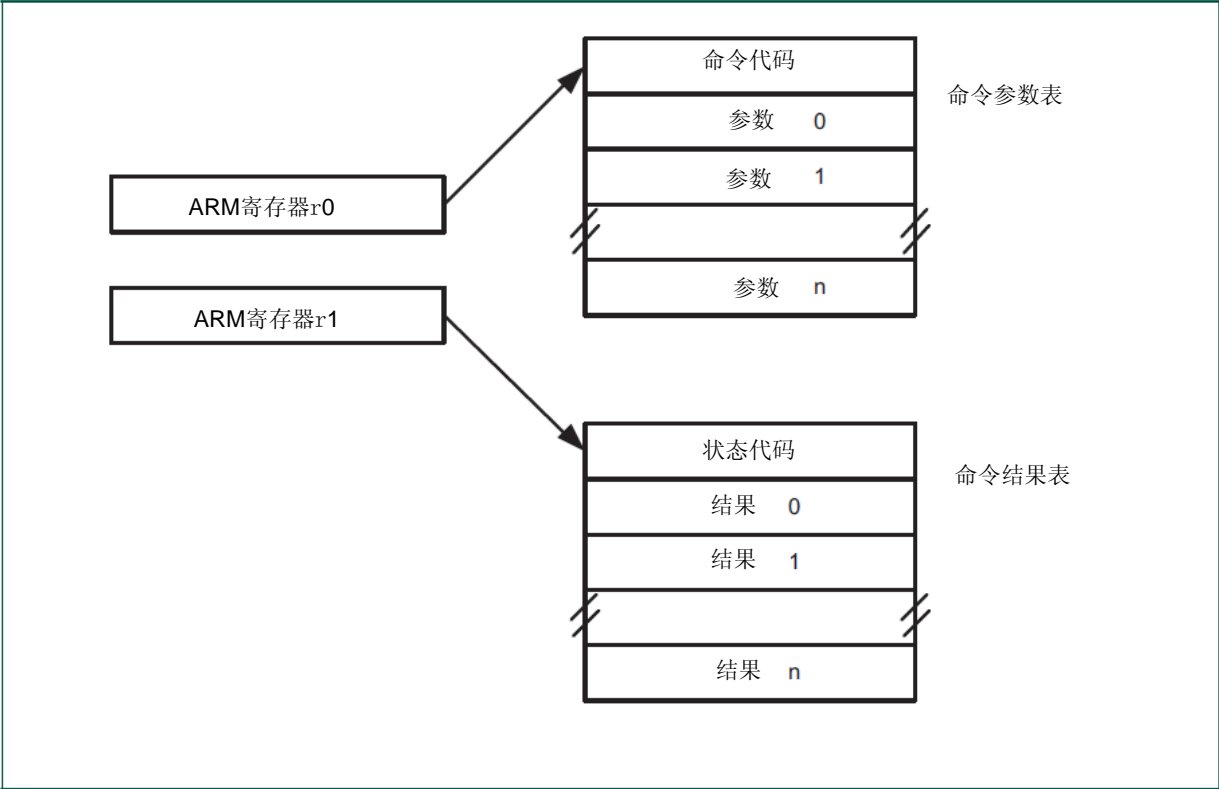
根据 ARM 规范（ARM Thumb 过程调用标准 SWS ESPC 0002 A-05），R0、R1、R2 和 R3 寄存器能够分别传递最多 4 个参数。其他参数通过堆栈传递。最多有 4 个参数可以返回 R0、R1、R2 和 R3 寄存器。其他的参数间接通过存储器返回。有些 IAP 调用需要的参数多于 4 个。如果使用 ARM 建议的机制来传递/返回参数，则有可能因为不同厂商所提供的 C 编译器不同而产生问题。使用建议的参数传递机制便可降低这种风险。

Flash 存储器在写或擦除操作过程中不可被访问。执行 Flash 写/擦除操作的 IAP 命令使用片上 RAM 顶端的 32 个字节空间。如果应用程序中允许 IAP Flash 编程，那么用户程序不应使用该空间。

表728. IAP 命令汇总

IAP 命令	命令代码	描述
准备写操作扇区	50 ₁₀	见表 729
将 RAM 内容复制到 Flash	51 ₁₀	见表 730
擦除扇区	52 ₁₀	见表 731
扇区查空	53 ₁₀	见表 732
读器件 ID	54 ₁₀	见表 733
读引导代码版本	55 ₁₀	见表 734
读设备序列号	58 ₁₀	见表 735
比较	56 ₁₀	见表 736
重新调用 ISP	57 ₁₀	见表 737

图170. IAP 参数传递



37.8.1 准备写操作扇区

该命令使 Flash 写/擦除操作分两步执行。

表729. IAP 准备写操作扇区命令

命令	准备写操作扇区
输入	命令代码：50 ₁₀ 参数 0：起始扇区号 参数 1：结束扇区号（应大于或等于起始扇区号）。
返回代码	CMD_SUCCESS BUSY INVALID_SECTOR
结果	无
描述	该命令必须在执行“将 RAM 内容复制到 Flash”或“擦除扇区”命令前执行。 这两个命令的成功执行使得相关扇区被再次保护。要准备单个扇区，可将起始和结束扇区号设置为相同值。

37.8.2 将 RAM 内容复制到 Flash

表730. IAP 复制 RAM 内容到 Flash 命令

命令	将 RAM 内容复制到 Flash
输入	命令代码：51 ₁₀ 参数 0（DST）：待写入数据字节的目标 Flash 地址。该地址的边界应为 256 字节。 参数 1（SRC）：待读出数据字节的源 RAM 地址。该地址应以字为边界。 参数 2：待写入的字节数。应当为 256 512 1024 4096。 参数 3：CPU 时钟频率（CCLK）（单位：kHz）。
返回代码	CMD_SUCCESS SRC_ADDR_ERROR （地址不是以字为边界） DST_ADDR_ERROR （地址边界错误） SRC_ADDR_NOT_MAPPED DST_ADDR_NOT_MAPPED COUNT_ERROR （字节计数值不是 256 512 1024 4096） SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION BUSY
结果	无
描述	该命令用来编程 Flash 存储器。受影响的扇区应先通过调用“准备写操作扇区”命令准备好。当成功执行复制命令后，受影响的扇区将自动再次受到保护。

37.8.3 擦除扇区

表731. IAP 擦除扇区命令

命令	擦除扇区
输入	命令代码: 52 ₁₀ 参数 0: 起始扇区号 参数 1: 结束扇区号 (应大于或等于起始扇区号)。 参数 2: CPU 时钟频率 (CCLK) (单位: kHz)。
返回代码	CMD_SUCCESS BUSY SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION INVALID_SECTOR
结果	无
描述	该命令用来擦除片上 Flash 存储器的一个或多个扇区。要擦除单个扇区, 可将起始和结束扇区号设置为相同值。

37.8.4 扇区查空

表732. IAP 扇区查空命令

命令	扇区查空
输入	命令代码: 53 ₁₀ 参数 0: 起始扇区号 参数 1: 结束扇区号 (应大于或等于起始扇区号)。
返回代码	CMD_SUCCESS BUSY SECTOR_NOT_BLANK INVALID_SECTOR
结果	结果 0: 状态代码为 SECTOR_NOT_BLANK 时第一个非空字位置的偏移量。 结果 1: 非空字位置的内容。
描述	该命令用于对片上 Flash 存储器的一个或多个扇区进行查空。要对单个扇区进行查空, 可将“起始”和“结束”扇区号设置为相同值。

37.8.5 读器件标识号

表733. IAP 读器件标识号命令

命令	读器件标识号
输入	命令代码: 54 ₁₀ 参数: 无
返回代码	CMD_SUCCESS
结果	结果 0: 器件标识号。
描述	该命令用来读取器件的标识号。返回的值是十六进制形式的器件 ID 版本号。详见 表 723 。

37.8.6 读取引导代码版本号

表734. IAP 读取引导代码版本号命令

命令	读取引导代码版本号
输入	命令代码：55 ₁₀ 参数：无
返回代码	CMD_SUCCESS
结果	结果 0：2 字节引导代码版本号（ASCII 格式）。将其解释为<字节 1（主）>.<字节 0（次）>。
描述	该命令用来读取引导代码版本号。

37.8.7 读设备序列号

表735. IAP 读设备序列号命令

命令	读设备序列号
输入	命令代码：58 ₁₀ 参数：无
返回代码	CMD_SUCCESS
结果	结果 0：第一个 32 位字的设备标识号（位于最低地址） 结果 1：第二个 32 位字的设备标识号 结果 2：第三个 32 位字的设备标识号 结果 3：第四个 32 位字的设备标识号
描述	该命令用来读取设备标识号。该序列号可用来在所有 LPC178x/177x 系列微控制器中识别一个设备。

37.8.8 比较<地址 1> <地址 2> <字节数>

表736. IAP 比较命令

命令	比较
输入	命令代码：56 ₁₀ 参数 0（DST）：待比较的数据字节的起始 Flash 或 RAM 地址。该地址应以字为边界。 参数 1（SRC）：待比较的数据字节的起始 Flash 或 RAM 地址。该地址应以字为边界。 参数 2：待比较的字节数；应为 4 的倍数。
返回代码	CMD_SUCCESS COMPARE_ERROR COUNT_ERROR （字节计数值不是 4 的倍数） ADDR_ERROR ADDR_NOT_MAPPED
结果	结果 0：当状态代码为 COMPARE_ERROR 时，第一个不匹配字节的偏移量。
描述	该命令用来比较两个地址的存储器内容。 当源或目标地址包含从地址 0 开始的前 64 个字节中的任意一个地址时，比较的结果可能不准确。因为前 64 个字节可被重新映射到 RAM。

37.8.9 重新调用 ISP

表737. 重新调用 ISP 命令

命令	比较
输入	命令代码：57 ₁₀
返回代码	无
结果	无。
描述	该命令用来调用 ISP 模式下的引导加载器程序。该命令映射引导向量，设置 PCLK=CCLK/4,配置 UART0 的发送和接收管脚，复位 TIMER1 及 U0FDR 寄存器（参见 18.6.11 节）。当内部 Flash 存储器中存在有效的用户程序并且 P2[10]管脚无法进入 ISP 模式时，可以使用该命令。 该命令不禁止 PLL，所以当该器件不在 PLL 下运行时可调用引导加载器程序。在这种情况下，ISP 程序应当在自动波特率握手后传递 CCLK 频率（晶体或 PLL 输出取决于时钟源的选择，参见 4.6 节）。该命令还可用来禁能 PLL 以及在调用 IAP 之前选择 IRC 作为时钟源。 在这种情况下，ISP 发送的频率被忽略，且 12MHz 的 IRC 直接供 CPU 时钟使用。

37.8.10 IAP 状态代码

表738. IAP 状态代码汇总

返回代码	符号	描述
0	CMD_SUCCESS	命令被成功执行。
1	INVALID_COMMAND	无效命令。
2	SRC_ADDR_ERROR	源地址没有以字为边界。
3	DST_ADDR_ERROR	目标地址的边界错误。
4	SRC_ADDR_NOT_MAPPED	源地址没有在存储器映射中被映射。计数值必须考虑到可用性。
5	DST_ADDR_NOT_MAPPED	目标地址没有在存储器映射中被映射。计数值必须考虑到可用性。
6	COUNT_ERROR	字节计数值不是 4 的倍数或是一个非法值。
7	INVALID_SECTOR	扇区号无效。
8	SECTOR_NOT_BLANK	扇区非空。
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	为写操作准备扇区的命令未执行。
10	COMPARE_ERROR	源和目标数据不相同。
11	BUSY	Flash 编程硬件接口忙。

37.9 JTA Flash 编程接口

调试工具可将一部分 Flash 映像写入 RAM，然后按照正确的偏移地址重复调用命令“将 RAM 内容复制到 Flash（Copy RAM to Flash）”。

37.10 Flash 符号差生成

Flash 模块包括有一个内置符号差发生器。该发生器可以为一段 Flash 存储器产生一个 128 位的符号差。一个典型的用法就是验证对比闪存内容和一个计算出的符号差（例如在编程期间）。

用于生成符号差的地址范围必须对齐 Flash-字边界，例如 128 位边界。在开始之后，符号差生成独立完成。在生成符号差的过程中，Flash 存储器不能用于其他用途，如果在生成符号差的过程中读取 Flash 存储器会使能等待状态，直至符号差生成结束。在符号差生成过程中可以执行 Flash 外的代码（例如：内部 RAM）。如果中断向量表被重新映射至 Flash 存储器以外的存储器中，中断服务也可以被包含在内。启动符号差生成的代码必须位于 Flash 存储器外部。

37.10.1 用于生成符号差的寄存器描述

表739. 寄存器概述：FMC（基地址 0x0020 0000）

名称	描述	访问	复位值	地址	参考
FMSSTART	符号差起始地址寄存器	R/W	0	0x0020 0020	见 表 740
FMSSTOP	符号差结束地址寄存器	R/W	0	0x0020 0024	见 表 741
FMSW0	128 位符号差字 0	RO	-	0x0020 002C	见 表 742
FMSW1	128 位符号差字 1	RO	-	0x0020 0030	见 表 743
FMSW2	128 位符号差字 2	RO	-	0x0020 0034	见 表 744
FMSW3	128 位符号差字 3	RO	-	0x0020 0038	见 表 745
FMSTAT	符号差生成状态寄存器	RO	0	0x0020 0FE0	见 37.10.1.3 节
FMSTATCLR	符号差生成状态清除寄存器	WO	-	0x0020 0FE8	见 37.10.1.4 节

37.10.1.1 符号差生成地址和控制寄存器

这些寄存器控制符号差的自动生成。可以为 Flash 存储器内容的任何部分生成符号差。可以通过向符号差启动地址寄存器（FMSSTART）中写入起始地址以及向符号差结束地址寄存器（FMSSTOP）中写入结束地址来对定义用于符号差生成的地址范围。起始和结束地址必须对齐 128 位边界，并且可以通过将字节地址除以 16 来得到。

通过置位 FMSSTOP 寄存器中的 SIG_START 位就可以启动符号差生成。在一次单独写操作过程中，对 SIG_START 位进行置位与符号差结束地址是结合在一起的。

[表 740](#) 和 [表 741](#) 中所示为 FMSSTART 和 FMSSTOP 寄存器中的位分配。

表740. Flash 模块符号差起始寄存器位描述（FMSSTART—0x0020 0020）

位	符号	描述	复位值
16: 0	START	符号差生成起始地址（相当于 AHB 字节地址位[20: 4]）。	0
31: 17	-	保留。读取值未定义，只写入 0。	无

表741. Flash 模块符号差结束寄存器位描述（FMSSTOP— 0x0020 0024）

位	符号	值	描述	复位值
16: 0	STOP		BIST 结束地址除以 16（相当于 AHB 字节地址[20: 4]）。	0
17	SIG_START		符号差生成的起始控制位。	0
		0	符号差生成停止。	
		1	发起符号差生成。	
31: 18	-		保留。读取值未定义，只写入 0。	无

37.10.1.2 符号差生成结果寄存器

符号差生成结果寄存器返回由嵌入式符号差发生器生成的 Flash 符号差。这个 128 位的符号差由 FMSW0、FMSW1、FMSW2 和 FMSW3 这四个寄存器来体现。

生成的 Flash 符号差可以用来验证 Flash 存储器的内容。将生成的符号差与预期的符号差进行对比可以节省时间和代码空间。有关生成符号差的方法，参见 [37.10.2](#) 节。

[表 745](#) 所示为 FMSW0 和 FMSW1、FMSW2、FMSW3 寄存器的位分配。

表742. FMSW0 寄存器位描述（FMSW0，地址：0x0020 002C）

位	符号	描述	复位值
31: 0	SW0[31: 0]	128 位符号差的字 0（位 31: 0）。	-

表743. FMSW1 寄存器位描述（FMSW1，地址：0x0020 0030）

位	符号	描述	复位值
31: 0	SW1[63: 32]	128 位符号差的字 1（位 63: 32）。	-

表744. FMSW2 寄存器位描述（FMSW2，地址：0x0020 0034）

位	符号	描述	复位值
31: 0	SW2[95: 64]	128 位符号差的字 2（位 95: 64）。	-

表745. FMSW3 寄存器位描述（FMSW3，地址：0x0020 0038）

位	符号	描述	复位值
31: 0	SW3[127: 96]	128 位符号差的字 3（位 127: 96）。	-

37.10.1.3 Flash 模块状态寄存器（FMSTAT—0x0x0020 0FE0）

只读 FMSTAT 寄存器提供一种确定符号差生成何时完成的方法。通过轮询 FMSTAT 中的 SIG_DONE 位可以得知符号差是否已经生成。应通过 FMSTATCLR 寄存器来清除 SIG_DONE，否则该状态可能表示前一个操作的完成。

表746. Flash 模块状态寄存器位描述（FMSTAT—0x0020 0FE0）

位	符号	描述	复位值
1: 0	-	保留。从保留位读取的值未定义。	无
2	SIG_DONE	为 1 时，之前启动的符号差生成已完成。要清除该标志，参见 FMSTATCLR 寄存器的描述。	0
31: 2	-	保留。从保留位读取的值未定义。	无

37.10.1.4 Flash 模块状态清除寄存器（FMSTATCLR—0x0x0020 0FE8）

FMSTATCLR 寄存器用于清除符号差生成完成标志。

表747. Flash 模块状态清除寄存器位描述（FMSTATCLR—0x0x0020 0FE8）

位	符号	描述	复位值
1: 0	-	保留。读取值未定义，只写入 0。	无
2	SIG_DONE_CLR	向该位写入 1 即可清除 FMSTAT 寄存器中的符号差生成完成标志（SIG_DONE）。	0
31: 2	-	保留。读取值未定义，只写入 0。	无

37.10.2 生成符号差的算法和步骤

生成符号差

可以就 Flash 存储器内容的任何部分生成符号差。通过向 FMSSTART 寄存器中写入起始地址以及向 FMSSTOP 寄存器中写入结束地址来定义用于生成符号差的地址范围。

向 FMSSTOPMISR_START 写 1 就可以启动符号差生成。通常在启动符号差生成的同时也会在同一寄存器中的 FMSSTOP.FMSSTOP 字段来定义结束地址。

生成符号差所用的时间与生成符号差的地址范围成正比。读取 Flash 存储器用于生成符号差时使用自定时读取机制，与 Flash 的可配置定时设定无关。生成符号差所用时间的安全预估公式是：

$$\text{持续时间} = \text{int}((60/\text{tcy}) + 3) \times (\text{FMSSTOP} - \text{FMSSTART} + 1)$$

若是符号差生成由软件来触发，持续时间为 AHB 时钟周期内，tcy（单位：ns）为一个 AHB 时钟的时间。可以通过软件轮询 FMSTAT 中的 SIG_DONE 位来确定符号差生成是否已完成。

若是符号差生成由 JTAG 来触发，则持续时间为 JTAG tck 周期内，tcy（单位：ns）是一个 JTAG 时钟的时间。在这种情况下，无法轮询 FMSTAT 中的 SIG_DONE 位。

在生成符号差之后，可以从 FMSW0~FMSW3 寄存器读取到一个 128 位的符号差。这个 128 位的签名反映了从 Flash 中读取到的正确数据（Flash 奇偶位和校验位的值）。

内容验证

从 FMSW0 到 FMSW3 寄存器读取到的符号差必须等于参考符号差。[图 171](#) 所示为生成参考符号差的算法。

图171. 生成一个 128-位符号差的算法

```
sign = 0
FOR address = FMSTART.FMSTART TO FMSTOP.FMSTOP
{
    FOR i = 0 TO 126
        nextSign[i] = f_Q[address][i] XOR sign[i+1]
        nextSign[127] = f_Q[address][127] XOR sign[0] XOR sign[2] XOR
            sign[27] XOR sign[29]
    sign = nextSign
}
signature128 = sign
```

38.1 特性

- 支持标准的 JTAG 和 ARM 串行调试模式。
- 可直接对所有存储器、寄存器和外设进行调试。
- 调试阶段不需要目标资源。
- 可跟踪端口，使得 CPU 可以跟踪指令。
- 8 个断点。其中 6 个指令断点也可以用来重新映射修补代码的指令地址。另外两个数据比较器可用于重新映射修补文字值的地址。
- 四个数据观察点，也可用作跟踪触发器。
- 指令跟踪宏单元支持对其它软件控制的跟踪。

38.2 简介

ARM Cortex-M3 集成了调试和跟踪功能。除了标准 JTAG 调试和并行跟踪 (parallel trace) 功能之外，设备还支持串行调试和跟踪功能。ARM Cortex-M3 被配置为支持多达 8 个断点和 4 个观察点。

38.3 描述

LPC178x/177x 的默认调试方式为 JTAG。若使用了 JTAG 调试模式，调试工具就可以切换到串行调试模式。

利用一个使用 5 个管脚的 4 位并行接口来进行指令跟踪。请注意，Cortex-M3 的跟踪功能与之前的基于 ARM7 的设备有很大区别。Cortex-M3 在跟踪时仅占用 5 个管脚，而基于 ARM7 的设备要占用 10 个管脚。

38.4 管脚描述

以下表格列出了与调试和跟踪有关的各类管脚功能。有些管脚具备几种不同的功能，而某一种功能可能其他管脚也具备，因此管脚功能不会同时执行。当管脚作为 JTAG 端口时就不能用作串行调试和串行输出。使用并行跟踪需要 5 个管脚（可能是用户程序需要的），从而限制了这些特性的调试。

表748. JTAG 管脚描述

管脚名称	类型	描述
JTAG_TCK	输入	JTAG 测试时钟。 在 JTAG 调试模式下，该管脚为调试逻辑提供时钟信号。
JTAG_TMS	输入	JTAG 测试模式选择。 TMS 管脚可选择 TAP 状态机的下一个状态。该管脚含一个与 IEEE1149.1 相符的内部上拉电阻。
JTAG_TDI	输入	JTAG 测试数据输入。 这是移位寄存器的串行数据输入管脚。该管脚含一个与 IEEE1149.1 相符的内部上拉电阻。
JTAG_TDO	输出	JTAG 测试数据输出。 这是移位寄存器的串行数据输出管脚。数据在 TCK 信号的负相沿从寄存器输出。
JTAG_TRST	输入	JTAG 测试复位。 该管脚可用来复位调试逻辑中的测试逻辑。该管脚含一个与 IEEE1149.1 相符的内部上拉电阻。

表749. 串行调试管脚描述

管脚名称	类型	描述
SWDCLK	输入	串行时钟。 在串行调试模式下，该管脚为调试逻辑提供时钟信号。这是为 JTAG_TCK 管脚选择的内部交变函数。
SWDIO	输入 / 输出	串行调试数据 I/O。 外部调试工具可通过 SWDIO 管脚与 Cortex-M3 CPU 通信并控制 Cortex-M3 CPU。这是为 JTAG_TMS 管脚选择的内部交变函数。
SWO	输出	串行输出。 SWO 管脚可向外部调试工具随意提供 ITM 和/或 ETM 的数据进行评估。这是为 JTAG_TDO 管脚选择的内部交变函数。

表750. 并行跟踪管脚描述

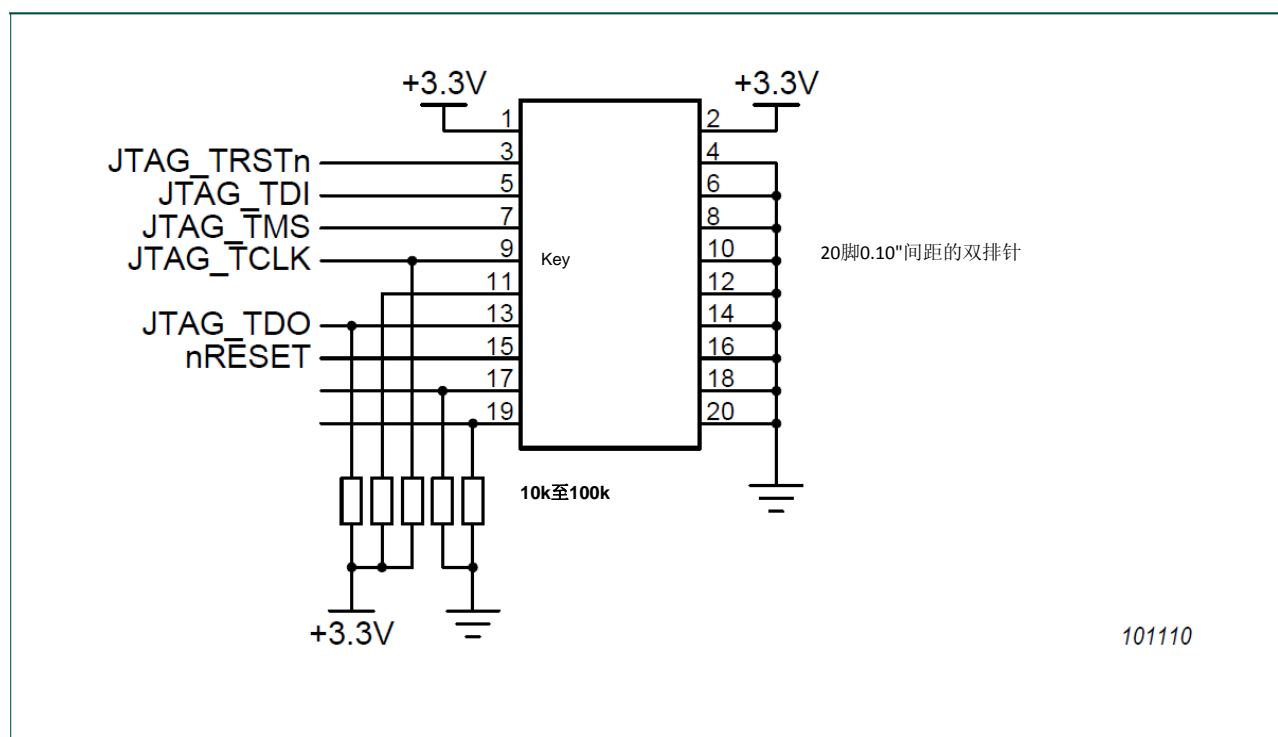
管脚名称	类型	描述
TRACECLK	输入	跟踪时钟。 当外部调试工具使能跟踪功能时，该管脚可为 TRACEDATA 管脚（用来跟踪数据）提供采样时钟信号。
TRACEDATA[3: 0]	输出	跟踪数据位 3: 0。 当外部调试工具使能跟踪功能时，这些管脚提供 ETM 跟踪数据。调试工具可解密压缩信息供用户读取。

38.5 调试连接

LPC178x/7x 提供用于 JTAG 和串行调试 (SWD) 的专用管脚。在开始调试之后, 器件将处于 JTAG 模式, 这也是 ARM 公司在设计器件时所推荐采用的模式。一旦进入调试模式, 调试器就可以将设备切换到 SWD 模式。

目标板与调试器的连接有多种方法。新板设计选用什么样的调试连接器取决于所使用的调试工具。例如，在过去，基于 ARM 的设备所使用的调试工具采用的是[图 172](#)所示的标准连接。在考虑那些有内置上拉（电阻）管脚的情况下，将该连接图表经修改之后也适用于LPC178x/7x。

图172. ARM 标准 JTAG 连接器



更新款的工具可能会使用如图 173 所示的仅用于调试的小型连接器。若用到调试跟踪特性，还可以使用如图 174 所示的调试-跟踪连接器。这两个连接器引出线在 ARM 公司的《CoreSight™ 元器件技术参考手册》中有定义。请注意，在设计应用板之前，必须先与工具厂商核对所要采用的调试连接方案。

图173. Cortex 调试连接器

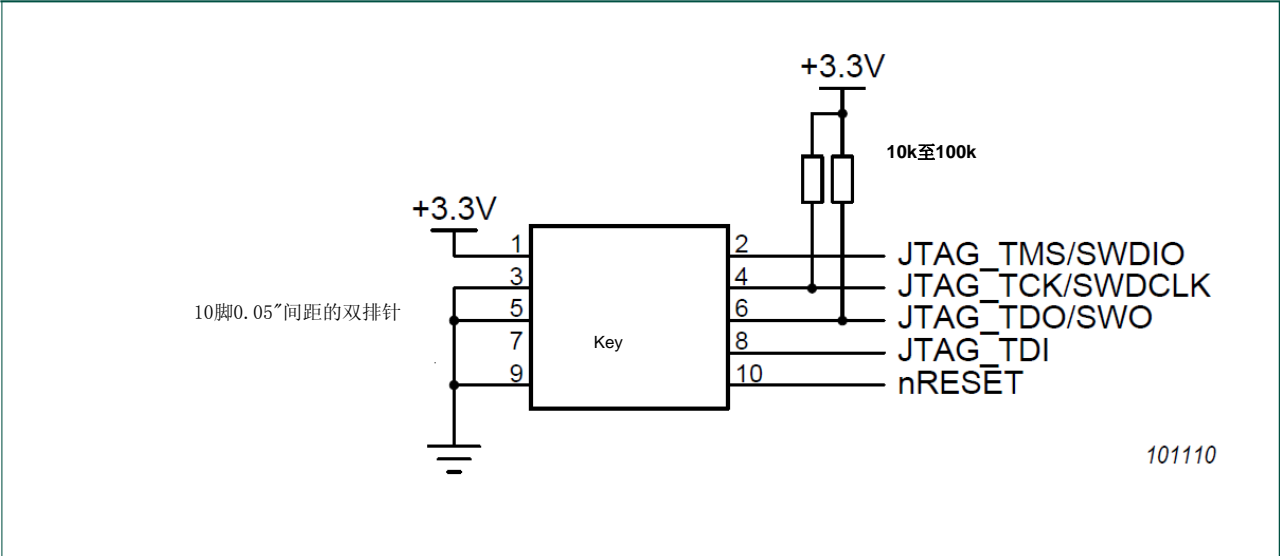
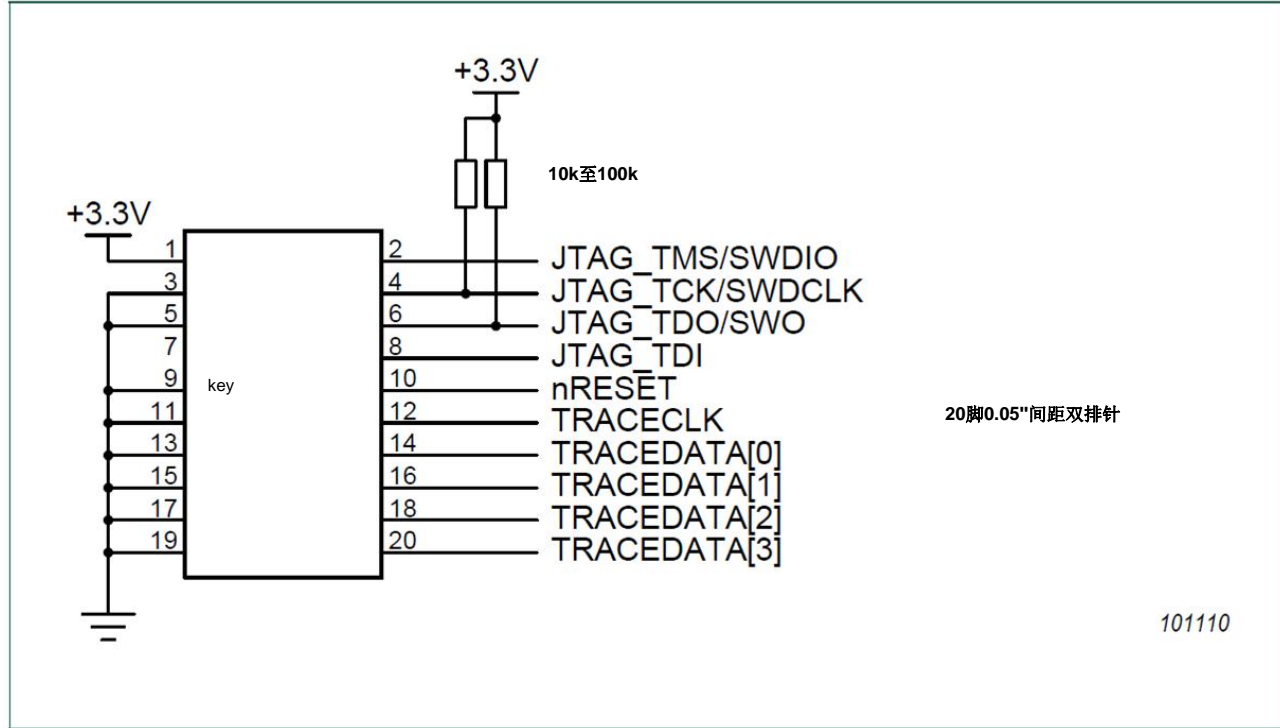


图174. Cortex 调试&ETM 连接器



38.6 JTAG TAP 标识

JTAG TAP 控制器包含有器件 ID，可供调试软件用于识别设备的一般类型。有关设备信息的详细内容，参见 ISP/IAP 命令（参见 [37.7](#) 节和 [37.8](#) 节）。对于 LPC178x/177x 系列，这一 ID 值为 0x4BA0 0477。

38.7 调试注释

重要提示： 用户必须意识到调试过程中有很多限制。最重要的是，出于 Cortex-M3 整体性方面的考虑，LPC178x/177x 不能通过常规方法从深度睡眠和掉电模式中唤醒。因此，建议在调试期间不要使用这些模式。

在通过 JTAG/SWD 接口下载完应用程序之后，应将目标板上的 USB-SWD/JTAG 调试适配器移除。在这之后，若对 LPC178x/177x 做动力循环(power cycle)，就可以使 LPC178x/177x 从深度睡眠和掉电模式中唤醒。

另外一个问题是，调试模式改变了 Cortex-M3 CPU 的低功耗工作模式。这使得在调试阶段的功耗模式有别于常规模式下的功耗模式。这些差别意味着在调试期间不能进行功率测量，因为调试阶段下所测量得到的功率会比在普通操作期间测量的值高。

在调试期间，只要 CPU 停止，系统节拍定时器就会自动停止。其它外设则不受此影响。

若使能代码读保护，则不能对设备进行调试。

38.8 调试存储器重新映射

在芯片复位之后，有一部分 Boot ROM 被映射到地址 0，这样它就能被自动执行。在执行用户代码之前，Boot ROM 再将映射转换为指向 Flash 存储器。这样，用户通常无需知道重新映射的发生。

然而，若调试器在复位之后马上终止 CPU 的执行，Boot ROM 仍会被映射到地址 0，可能导致混乱。在理想情况下，调试器在这种情况下要能自动地修正映射地址，这样就不需要用户再处理了。

38.8.1 存储器映射控制寄存器（MEMMAP—0x400F C040）

MEMMAP 寄存器允许转换存储器底部的映射，包括 Boot ROM 和片上 Flash 存储器底部之间的默认复位和中断向量。

表751. 存储器映射控制寄存器位描述（MEMMAP—0x400F C040）

位	符号	值	描述	复位值
0	MAP		存储器映射控制。	0
		0	启动模式。部分 Boot ROM 被映射到地址 0。	
		1	用户模式。片上 Flash 存储器被映射到地址 0。	
31: 1	-		保留。读取值未定义，只写入 0。	无

39.1 ARM Cortex-M3 用户指南：简介

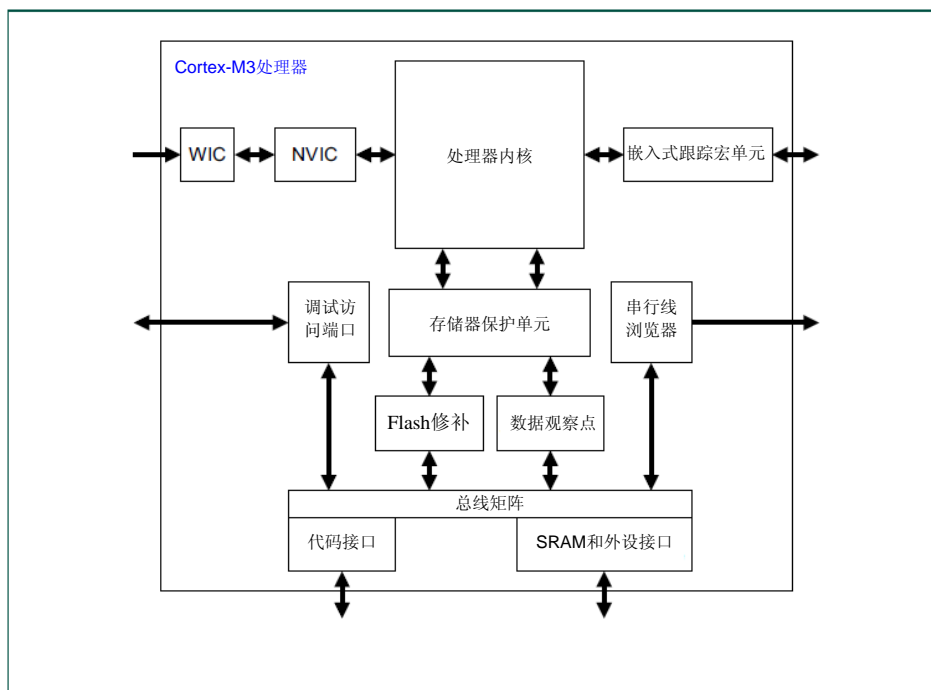
ARM Limited 提供本附录中的资料，并包括在含 Cortex-M3 CPU 的设备的用户手册中。为了反映只针对 LPC178x/177x 设备的实现选项以及其他的差异，内容有轻微的改动。

39.1.1 关于处理器和内核外设

Cortex-M3 处理器是一种为微控制器市场设计的高性能 32 位处理器。它为开发人员提供了显著的益处，包括：

- 杰出的处理性能，以及快速的中断处理能力；
- 大量断点和跟踪能力，增强了系统调试能力；
- 高效的处理器内核、系统和存储器；
- 集成睡眠模式，具有超低功耗；
- 有可选的集成存储器保护单元（MPU），保证平台的安全性。

图175. Cortex-M3 的典型应用



Cortex-M3 处理器建立在一个高性能处理器内核上，采用一个 3 级流水线哈佛架构，非常适合于高要求的嵌入式应用。该处理器使用了一个高效指令集和广泛优化的设计，具备了优异的能效，提供高端的处理硬件，包括单周期 32x32 乘法器和专用的硬件除法器。

为了降低设备的设计成本，Cortex-M3 处理器实现了紧密耦合的系统部件，在降低处理器面积的同时，还显著提高了中断处理和系统调试能力。Cortex-M3 处理器实现了一个 Thumb 指令集版本，从而确保了高代码密度，降低了对程序存储器的要求。Cortex-M3 指令集具有 8 位和 16 位微控制器的高代码密度，提供了现代 32 位架构理想的优异性能。

Cortex-M3 处理器紧密地集成了一个可配置的可嵌套向量中断控制器 (NVIC)，提供了业界领先的中断性能。NVIC 包括一个不可屏蔽中断 (NMI)，并提供高达 256 个中断优先级。处理器内核和 NVIC 的紧密耦合实现了中断服务程序 (ISR) 的快速执行，大大缩短了中断延迟。其实现方式是通过寄存器的硬件堆栈，以及挂起多个加载 (load-multiple) 和多个存储 (store-multiple) 操作的能力。中断处理程序不需要任何的汇编程序插桩 (stub)，消除了所有来自 ISR 的代码开销。末尾连锁 (Tail-chaining) 优化也显著降低了从一个 ISR 切换到另一个 ISR 时所产生的开销。

为了优化低功耗设计，NVIC 集成了睡眠模式，包括一个深度睡眠功能，能够使整个设备快速进入掉电模式。

LPC178x/177x 设备还支持其他的降功耗模式，详细内容参见 [4.7 “功率控制”](#)。

39.1.1.1 系统级接口

Cortex-M3 处理器采用 AMBA 技术提供了多个接口，以实现高速、低等待时间的存储器访问。它支持非对齐的数据访问，实现了原子位操作 (atomic bit manipulation)，能够实现更快的外设控制、系统自旋锁 (spinlock)，以及线程安全的 Boolean 数据处理。

Cortex-M3 处理器有一个可选的存储器保护单元 (MPU)，提供细粒度的存储器控制，使应用程序能够实现安全特权级别，在不同任务基础上划分代码、数据和堆栈。在很多嵌入式应用 (如汽车) 中，这类需求正变得越来越重要。LPC178x/177x 设备中包括了 MPU。

39.1.1.2 集成的可配置调试功能

Cortex-M3 处理器实现了完全的硬件调试解决方案。通过传统的 JTAG 端口或一个双管脚的串行线调试 (SWD) 端口提供了对处理器和存储器的系统高可视性，SWD 非常适用于微控制器和其它小封装设备。MCU 厂商决定了调试特性的配置，因此它不同设备和产品系列上可能存在差异。

在系统跟踪方面，除数据观察点和一个分析单元外，处理器中还集成了一个指令跟踪宏单元 (ITM)。为了能够简单而高效地分析这些生成的系统事件，串行线浏览器 (SWV) 可通过一个管脚输出一个软件生成消息、数据跟踪和数据分析信息流。

可选的嵌入式跟踪宏单元 (ETM) 的面积远小于传统跟踪单元的面积，但提供了无与伦比的指令跟踪捕获能力，从而使很多低成本 MCU 第一次实现了完全的指令跟踪。

LPC178x/177x 设备支持 JTAG 和串行线调试、串行线浏览器，并包括了嵌入式跟踪宏单元。更多信息参见 [38.1](#) 节。

39.1.1.3 Cortex-M3 处理器特性和优点汇总

- 系统外设的紧集成不但节约了空间还降低了开发成本；
- Thumb 指令集兼备了高代码密度和 32 位性能；
- 具备代码修补能力，支持 ROM 系统更新；
- 系统部件的功率控制优化；
- 集成睡眠模式，实现低功耗；
- 快速的代码执行允许使用较慢的处理器时钟或增加睡眠模式的时间；
- 硬件除法器 and 快速乘法器；
- 确定性的高性能中断处理，可用于对时间要求严格的应用中；
- 可选的**存储器保护单元**（MPU），用于对安全性要求严格的应用中；
- 全面的调试和跟踪能力：
 - 串行线调试和串行线跟踪，减少了调试和跟踪所需的管脚数。

39.1.1.4 Cortex-M3 内核外设

Cortex-M3 内核外设：

- **可嵌套向量中断控制器**
可嵌套向量中断控制器（NVIC）是一种支持低等待时间中断处理的嵌入式中断控制器。
- **系统控制模块**
系统控制模块（SCB）是编程模型与处理器之间的接口。它提供了系统实现信息和系统控制，包括配置、控制以及系统异常的报告。
- **系统定时器**
系统定时器 SysTick 是一个 24 位的递减计数定时器。它可以用作实时操作系统（RTOS）的节拍定时器或一个普通计数器。
- **存储器保护单元**
存储器保护单元（MPU）通过为不同存储区定义存储器属性，提高了系统的可靠性。它最多可以提供 8 个不同的区域，还有一个可选的预定义背景区。

39.2 ARM Cortex-M3 用户指南：指令集

39.2.1 指令集汇总

该处理器实现了一个 Thumb 指令集版本。[表 752](#) 列出了支持的指令。

注

[表 752](#) 中：

- 尖括号<>中为操作数的代替形式
- 花括号{}中为可选操作数
- “操作数” 栏并未列出全部
- Op2是一个灵活的第二操作数，可以是一个寄存器或是一个常数
- 多数指令可使用一个可选的条件代码后缀。

有关指令和操作数的更多信息，请参见指令描述部分。

表752. Cortex-M3 指令

助记符	操作数	简要描述	标志	章节
ADC, ADCS	{Rd,} Rn, Op2	带进位加法	N,Z,C,V	39.2.5.1
ADD, ADDS	{Rd,} Rn, Op2	加法	N,Z,C,V	39.2.5.1
ADD, ADDW	{Rd,} Rn, #imm12	加法	N,Z,C,V	39.2.5.1
ADR	Rd, label	加载相对 PC 地址	-	39.2.4.1
AND, ANDS	{Rd,} Rn, Op2	逻辑 “与”	N,Z,C	39.2.5.2
ASR, ASRS	Rd, Rm, <Rs #n>	算术右移	N,Z,C	39.2.5.3
B	label	跳转	-	39.2.9.1
BFC	Rd, #lsb, #width	位域清零	-	39.2.8.1
BFI	Rd, Rn, #lsb, #width	位域插入	-	39.2.8.1
BIC, BICS	{Rd,} Rn, Op2	位清零	N,Z,C	39.2.5.2
BKPT	#imm	断点	-	39.2.10.1
BL	label	带链接的跳转	-	39.2.9.1
BLX	Rm	带链接的跳转并切换指令集	-	39.2.9.1
BX	Rm	跳转并切换指令集	-	39.2.9.1
CBNZ	Rn, label	比较，结果非零则跳转	-	39.2.9.2
CBZ	Rn, label	比较，结果为零则跳转	-	39.2.9.2
CLREX	-	独占清零	-	39.2.4.9
CLZ	Rd, Rm	计算前导零数目	-	39.2.5.4
CMN, CMNS	Rn, Op2	与负值比较	N,Z,C,V	39.2.5.5
CMP, CMPS	Rn, Op2	比较	N,Z,C,V	39.2.5.5
CPSID	iflags	更改处理器状态，禁能中断	-	39.2.10.2
CPSIE	iflags	更改处理器状态，使能中断	-	39.2.10.2
DMB	-	数据内存屏障	-	39.2.10.3
DSB	-	数据同步屏障	-	39.2.10.4
EOR, EORS	{Rd,} Rn, Op2	异或	N,Z,C	39.2.5.2
ISB	-	指令同步屏障	-	39.2.10.5
IT	-	If-Then 条件模块	-	39.2.9.3
LDM	Rn{!}, reglist	加载多个寄存器，之后递增	-	39.2.4.6

助记符	操作数	简要描述	标志	章节
LDMDB, LDMEA	Rn{!}, reglist	加载多个寄存器，之前递减	-	39.2.4.6
LDMFD, LDMIA	Rn{!}, reglist	加载多个寄存器，之后递增	-	39.2.4.6
LDR	Rt, [Rn, #offset]	加载字至寄存器	-	39.2.4
LDRB, LDRBT	Rt, [Rn, #offset]	加载字节至寄存器	-	39.2.4
LDRD	Rt, Rt2, [Rn, #offset]	加载双字节至寄存器	-	39.2.4.2
LDREX	Rt, [Rn, #offset]	独占加载寄存器	-	39.2.4.8
LDREXB	Rt, [Rn]	独占加载字节至寄存器	-	39.2.4.8
LDREXH	Rt, [Rn]	独占加载半字至寄存器	-	39.2.4.8
LDRH, LDRHT	Rt, [Rn, #offset]	加载半字至寄存器	-	39.2.4
LDRSB, LDRSBT	Rt, [Rn, #offset]	加载经有符号扩展后的字节至寄存器	-	39.2.4
LDRSH, LDRSHT	Rt, [Rn, #offset]	加载经有符号扩展后的半字至寄存器	-	39.2.4
LDRT	Rt, [Rn, #offset]	加载字至寄存器	-	39.2.4
LSL, LSLS	Rd, Rm, <Rs n>	逻辑左移	N,Z,C	39.2.5.3
LSR, LSRS	Rd, Rm, <Rs n>	逻辑右移	N,Z,C	39.2.5.3
MLA	Rd, Rn, Rm, Ra	乘加，结果为 32 位	-	39.2.6.1
MLS	Rd, Rn, Rm, Ra	乘减，结果为 32 位	-	39.2.6.1
MOV, MOVs	Rd, Op2	移动	N,Z,C	39.2.5.6
MOVT	Rd, #imm16	移动到顶部	-	39.2.5.7
MOVW, MOV	Rd, #imm16	移动 16 位常数	N,Z,C	39.2.5.6
MRS	Rd, spec_reg	将专用寄存器的内容移到通用寄存器中	-	39.2.10.6
MSR	spec_reg, Rm	将通用寄存器的内容移到专用寄存器中	N,Z,C,V	39.2.10.7
MUL, MULS	{Rd,} Rn, Rm	乘法，结果为 32 位	N,Z	39.2.6.1
MVN, MVNS	Rd, Op2	取反移动	N,Z,C	39.2.5.6
NOP	-	不执行任何操作	-	39.2.10.8
ORN, ORNS	{Rd,} Rn, Op2	逻辑“或非”	N,Z,C	39.2.5.2
ORR, ORRS	{Rd,} Rn, Op2	逻辑“或”	N,Z,C	39.2.5.2
POP	reglist	从堆栈弹出寄存器	-	39.2.4.7
PUSH	reglist	将寄存器推入堆栈	-	39.2.4.7
RBIT	Rd, Rn	反转位	-	39.2.5.8
REV	Rd, Rn	反转字中的字节顺序	-	39.2.5.8
REV16	Rd, Rn	反转每个半字中的字节顺序	-	39.2.5.8
REVSH	Rd, Rn	反转低半字中的字节顺序，并将符号扩展	-	39.2.5.8
ROR, RORS	Rd, Rm, <Rs n>	向右循环移	N,Z,C	39.2.5.3
RRX, RRXS	Rd, Rm	带扩展向右循环移	N,Z,C	39.2.5.3
RSB, RSBS	{Rd,} Rn, Op2	反向减法	N,Z,C,V	39.2.5.1
SBC, SBcs	{Rd,} Rn, Op2	带进位减法	N,Z,C,V	39.2.5.1
SBFX	Rd, Rn, #lsb, #width	有符号位域提取	-	39.2.8.2
SDIV	{Rd,} Rn, Rm	有符号除法	-	39.2.6.3
SEV	-	发送事件	-	39.2.10.9
SMLAL	RdLo, RdHi, Rn, Rm	有符号乘加 (32 x 32 + 64)， 结果为 64 位	-	39.2.6.2
SMULL	RdLo, RdHi, Rn, Rm	有符号乘法 (32 x 32)，结果为 64 位	-	39.2.6.2
SSAT	Rd, #n, Rm {,shift #s}	有符号饱和	Q	39.2.7.1
STM	Rn{!}, reglist	存储多个寄存器，之后递增	-	39.2.4.6
STMDB, STMEA	Rn{!}, reglist	存储多个寄存器，之前递减	-	39.2.4.6
STMFD, STMIA	Rn{!}, reglist	存储多个寄存器，之后递增	-	39.2.4.6
STR	Rt, [Rn, #offset]	存储寄存器字	-	39.2.4
STRB, STRBT	Rt, [Rn, #offset]	存储寄存器字节	-	39.2.4

助记符	操作数	简要描述	标志	章节
STRD	Rt, Rt2, [Rn, #offset]	存储寄存器双字	-	39.2.4.2
STREX	Rd, Rt, [Rn, #offset]	独占存储寄存器	-	39.2.4.8
STREXB	Rd, Rt, [Rn]	独占存储寄存器, 字节	-	39.2.4.8
STREXH	Rd, Rt, [Rn]	独占存储寄存器, 半字	-	39.2.4.8
STRH, STRHT	Rt, [Rn, #offset]	存储寄存器, 半字	-	39.2.4
STRT	Rt, [Rn, #offset]	存储寄存器, 字	-	39.2.4
SUB, SUBS	{Rd,} Rn, Op2	减法	N,Z,C,V	39.2.5.1
SUB, SUBW	{Rd,} Rn, #imm12	减法	N,Z,C,V	39.2.5.1
SVC	#imm	超级用户调用	-	39.2.10.10
SXTB	{Rd,} Rm {,ROR #n}	有符号扩展一个字节	-	39.2.8.3
SXTH	{Rd,} Rm {,ROR #n}	有符号扩展一个半字	-	39.2.8.3
TBB	[Rn, Rm]	表跳转字节	-	39.2.9.4
TBH	[Rn, Rm, LSL #1]	表跳转半字	-	39.2.9.4
TEQ	Rn, Op2	相等测试	N,Z,C	39.2.5.9
TST	Rn, Op2	测试	N,Z,C	39.2.5.9
UBFX	Rd, Rn, #lsb, #width	无符号位域提取	-	39.2.8.2
UDIV	{Rd,} Rn, Rm	无符号除法	-	39.2.6.3
UMLAL	RdLo, RdHi, Rn, Rm	无符号乘加 (32 x 32 + 64), 结果为 64 位	-	39.2.6.2
UMULL	RdLo, RdHi, Rn, Rm	无符号乘法 (32 x 32), 结果为 64 位	-	39.2.6.2
USAT	Rd, #n, Rm {,shift #s}	无符号饱和	Q	39.2.7.1
UXTB	{Rd,} Rm {,ROR #n}	用零扩展一个字节	-	39.2.8.3
UXTH	{Rd,} Rm {,ROR #n}	用零扩展一个半字	-	39.2.8.3
WFE	-	等待事件	-	39.2.10.11
WFI	-	等待中断	-	39.2.10.12

39.2.2 内在函数

ANSI 不能直接访问某些 Cortex-M3 指令。本节描述了可生成这些指令的内在函数，这些函数由 CMSIS 提供，也可以由 C 编译器提供。如果 C 编译器不支持合适的内在函数，则可能必须要用内联汇编器来访问这些指令。

CMSIS 提供以下内在函数，用以生成 ANSI 不能直接访问的指令：

表753. 用来生成一些 Cortex-M3 指令的 CMSIS 内在函数

指令	CMSIS 内在函数
CPSIE I	void __enable_irq(void)
CPSID I	void __disable_irq(void)
CPSIE F	void __enable_fault_irq(void)
CPSID F	void __disable_fault_irq(void)
ISB	void __ISB(void)
DSB	void __DSB(void)
DMB	void __DMB(void)
REV	uint32_t __REV(uint32_t int value)
REV16	uint32_t __REV16(uint32_t int value)
REVSH	uint32_t __REVSH(uint32_t int value)
RBIT	uint32_t __RBIT(uint32_t int value)
SEV	void __SEV(void)
WFE	void __WFE(void)
WFI	void __WFI(void)

CMSIS 还提供了一些函数，用于访问使用 MRS 和 MSR 指令的专用寄存器：

表754. 用来访问专用寄存器的 CMSIS 内在函数

专用寄存器	访问	CMSIS 函数
PRIMASK	读	uint32_t __get_PRIMASK (void)
	写	void __set_PRIMASK (uint32_t value)
FAULTMASK	读	uint32_t __get_FAULTMASK (void)
	写	void __set_FAULTMASK (uint32_t value)
BASEPRI	读	uint32_t __get_BASEPRI (void)
	写	void __set_BASEPRI (uint32_t value)
CONTROL	读	uint32_t __get_CONTROL (void)
	写	void __set_CONTROL (uint32_t value)
MSP	读	uint32_t __get_MSP (void)
	写	void __set_MSP (uint32_t TopOfMainStack)
PSP	读	uint32_t __get_PSP (void)
	写	void __set_PSP (uint32_t TopOfProcStack)

39.2.3 关于指令描述

以下章节提供了与使用指令相关的更多信息：

- [39.2.3.1 “操作数”](#)
- [39.2.3.2 “使用 PC 或 SP 时的限制”](#)
- [39.2.3.3 “灵活的第二操作数”](#)
- [39.2.3.4 “移位操作”](#)
- [39.2.3.5 “地址对齐”](#)
- [39.2.3.6 “相对 PC 的表达式”](#)
- [39.2.3.7 “条件执行”](#)
- [39.2.3.8 “指令宽度选择”](#)

39.2.3.1 操作数

指令操作数可以是一个 ARM 寄存器、一个常数或是其他指令特有的参数。指令按操作数执行，并通常将结果保存在一个目标寄存器中。当指令中有一个目标寄存器时，它通常在操作数前指定。

某些指令中的操作数比较灵活，因为它即可以是一个寄存器，也可以是一个常数，参见 [39.2.3.3](#) 节。

39.2.3.2 使用 PC 或 SP 时的限制

很多指令对操作数或目标寄存器是否可使用**程序计数器（PC）**或**堆栈指针（SP）**有限制。更多信息请参见指令描述。

注：当使用 BX、BLX、LDM、LDR 或 POP 指令写 PC 时，任意写入地址的位[0]必须为 1 才能正确执行指令，因为此位表示所需的指令集，而 Cortex-M3 处理器只支持 Thumb 指令。

39.2.3.3 灵活的第二操作数

很多通用数据处理指令都有一个灵活的第二操作数，在各个指令的语法描述中表示为**操作数 2（Operand2）**。

Operand2 可以是：

- [39.2.3.3.1 “常数”](#)
- [39.2.3.3.2 “带可选移位的寄存器”](#)

39.2.3.3.1 常数

以下列形式指定一个 *Operand2* 常数：

#constant

其中“constant”可以是：

- 在一个 32 位的字内，将一个 8 位值左移任意位数可以得到的任何常数；

- 任何 $0x00XY00XY$ 形式的常数；
- 任何 $0xXY00XY00$ 形式的常数；
- 任何 $0xXYXYXYXY$ 形式的常数。

注：在上述常数中，X 和 Y 均为 16 进制数字。

此外，在少数指令中，“constant”可以是更大范围的值。这在单个的指令描述中说明。

当将一个 **Operand2** 常数与指令 MOVs、MVNS、ANDS、ORRS、ORNS、EORS、BICS、TEQ 或 TST 一起使用时，如果常数大于 255 且可通过移位一个 8 位值产生，则进位标志被更新为常数的位[31]。如果 **Operand2** 为其他任何常数，则这些指令不影响进位标志。

指令替代：当指定了一个不被允许的常数时，所使用的汇编器可能会产生一条等效指令。例如，某个汇编器可能将指令 `CMP Rd, #0xFFFFFFFF` 汇编为等效指令 `CMN Rd, #0x2`。

39.2.3.3.2 带可选移位的寄存器

以下列形式指定 **Operand2** 寄存器：

Rm {, *shift*}

其中：

Rm 为保存第二操作数数据的寄存器。

shift 为用于 *Rm* 的一个可选移位。它可以是以下内容之一：

ASR#n：算术右移 *n* 位， $1 \leq n \leq 32$ 。

LSL#n：逻辑左移 *n* 位， $1 \leq n \leq 31$ 。

LSR#n：逻辑右移 *n* 位， $1 \leq n \leq 32$ 。

ROR#n：向右循环移 *n* 位， $1 \leq n \leq 31$ 。

RRX：带扩展向右循环移 1 位。

—：如省略，则不发生移位，等效于 **LSL#0**。

如果省略移位或指定 **LSL #0**，则指令使用 *Rm* 中的值。

如果指定了一个移位，则将移位用于 *Rm* 中的值，得到的 32 位值由指令使用。但寄存器 *Rm* 中的内容保持不变。指定一个带移位的寄存器还会更新与某些指令一起使用的进位标志。有关移位操作及其对进位标志影响方式的信息，请参见 [39.2.3.4 “移位操作”](#)

39.2.3.4 移位操作

寄存器移位操作是根据一个规定的位数（**移位长度**）将一个寄存器内的位向左移或向右移。寄存器移位可以按照以下方式进行：

- 直接使用指令 ASR、LSR、LSL、ROR 和 RRX 进行寄存器移位，移位结果被写入目标寄存器。
- 当指令（指定第二操作数作为一个带移位的寄存器的指令）计算 **Operand2** 时进行寄存器移位，见 [39.2.3.3](#) 节。移位结果被指令所使用。

允许的移位长度取决于移位类型和指令，请参见各指令的描述或 [39.2.3.3](#) 节。如果移位长度为 0，则不发生移位。除了指定的移位长度为 0 的情况之外，寄存器移位操作都会更新进位标志。以下各小节描述了各种移位操作，以及它们对进位标志的影响方式。在这些描述中，*Rm* 为包含待移位值的寄存器，*n* 为移位长度。

39.2.3.4.1 ASR

算术右移 *n* 位，是将寄存器 *Rm* 左边 32-*n* 位向右移 *n* 个位置，成为结果中的右边 32-*n* 位。并且，它还将寄存器原来的位[31]复制到结果左边的 *n* 个位上。参见图 176。

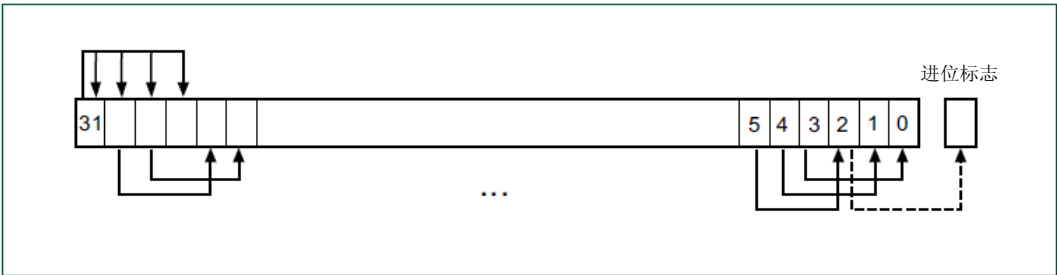
可使用 ASR #*n* 操作来将寄存器 *Rm* 中的值除以 2^{*n*}，得到的结果向负无穷方向舍入。

当指令为 ASRS，或指令 MOV_S、MVNS、AND_S、ORR_S、ORNS、EOR_S、BIC_S、TEQ 或 TST 中将 ASR #*n* 用于 *Operand2* 时，进位标志更新为寄存器 *Rm* 的最后一个移出位，即位[*n*-1]。

注

- 如果 *n* 为 32 或更大，则结果中的所有位都设置为 *Rm* 位[31]的值。
- 如果 *n* 为 32 或更大且进位标志被更新，则其更新为 *Rm* 位[31]的值。

图176. ASR #3



39.2.3.4.2 LSR

逻辑右移 *n* 位，是将寄存器 *Rm* 左边 32-*n* 位向右移 *n* 个位置，成为结果中的右边 32-*n* 位。并且，将结果的左边 *n* 位设置为 0。参见图 177。

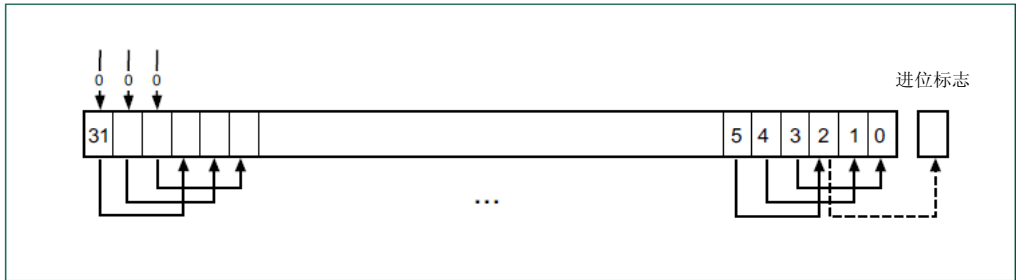
如果 *Rm* 中的值被视为一个无符号整数，则可以使用 LSR #*n* 操作将寄存器 *Rm* 中的值除以 2^{*n*}。

当指令为 LSRS，或将 LSR #*n* 用于指令 MOV_S、MVNS、AND_S、ORR_S、ORNS、EOR_S、BIC_S、TEQ 或 TST 的 *Operand2* 时，进位标志更新为寄存器 *Rm* 的最后一个移出位，即位[*n*-1]。

注

- 如果 *n* 为 32 或更大，则结果中的全部位都清零。
- 如果 *n* 为 33 或更大且进位标志被更新，则其更新为 0。

图177. LSR #3



39.2.3.4.3 LSL

逻辑左移 n 位，是将寄存器 Rm 右边 $32-n$ 位向左移 n 个位置，成为结果的左边 $32-n$ 位。并且将结果的右边 n 位设置为 0。参见图 178。

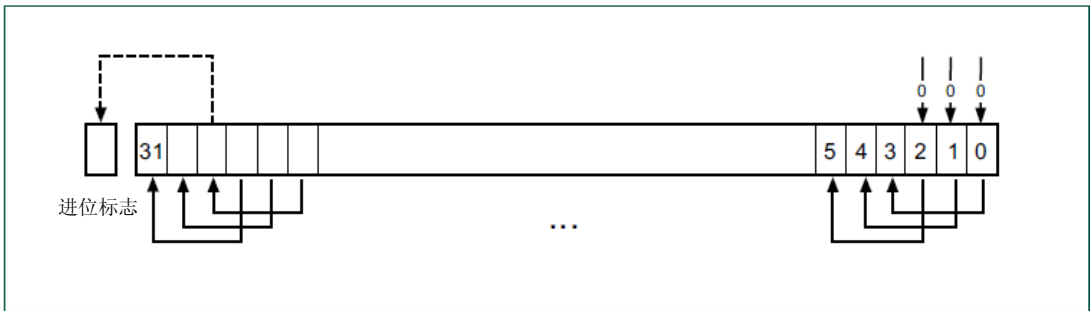
如果寄存器 Rm 中的值被视为一个无符号整数或一个 2 的补码有符号整数，则可使用 $LSL \#n$ 操作将寄存器 Rm 中的值乘以 2^n 。这可能会引发不带报警的溢出。

当指令为 $LSLS$ ，或将 $LSL \#n$ (n 非零) 用于指令 $MOV\ S$ 、 $MV\ NS$ 、 $AND\ S$ 、 $ORR\ S$ 、 $ORN\ S$ 、 $EOR\ S$ 、 $BIC\ S$ 、 TEQ 或 TST 的 $Operand2$ 时，进位标志更新为寄存器 Rm 的最后一个移出位，即位 $[32-n]$ 。当使用 $LSL \#0$ 时，这些指令不影响进位标志。

注

- 如果 n 为 32 或更大，则结果中全部位都清零。
- 如果 n 为 33 或更大且进位标志被更新，则其更新为 0。

图178. LSL#3



39.2.3.4.4 ROR

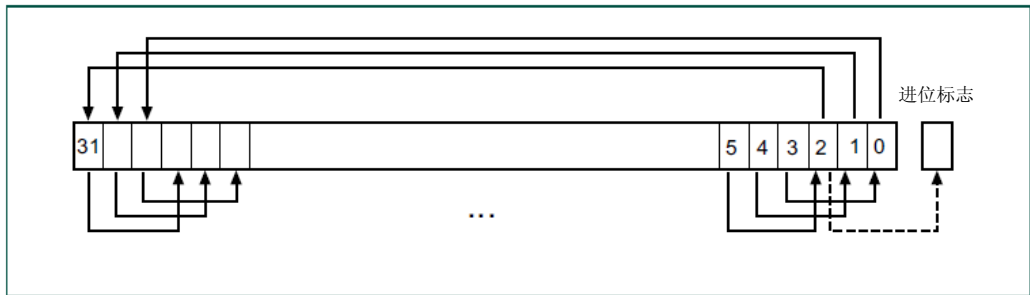
向右循环移 n 位，是将寄存器 Rm 左边 $32-n$ 位向右移 n 个位置，成为结果的右边 $32-n$ 位。并且将寄存器的右边 n 位移到结果的左边 n 位。参见图 179。

当指令为 $RORS$ ，或将 $ROR \#n$ 用于指令 $MOV\ S$ 、 $MV\ NS$ 、 $AND\ S$ 、 $ORR\ S$ 、 $ORN\ S$ 、 $EOR\ S$ 、 $BIC\ S$ 、 TEQ 或 TST 的 $Operand2$ 时，进位标志更新为寄存器 Rm 的最后一个循环位，即位 $[n-1]$ 。

注

- 如果 n 为 32，则结果的值与 Rm 中的值相同，并且如果进位标志被更新，则它更新为 Rm 的位[31]。
- 移位长度 n 大于 32 的 ROR 与移位长度为 $n-32$ 的 ROR 相同。

图179. ROR#3

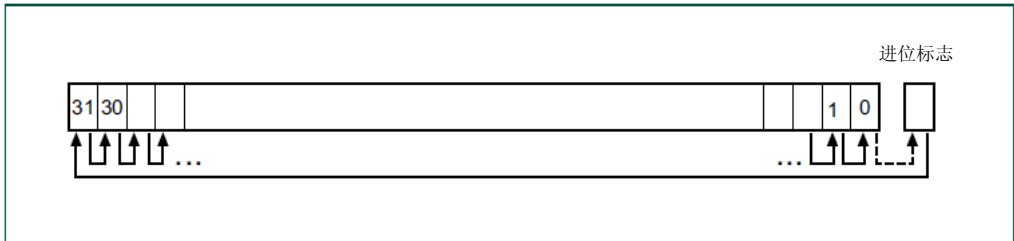


39.2.3.4.5 RRX

带扩展的向右循环移是将寄存器 Rm 中的位向右移一位。它还将进位标志复制到结果的位 [31]中。参见图 180。

当指令为 RRXS，或将 RRX 用于指令 MOV_S、MVN_S、AND_S、ORR_S、ORN_S、EOR_S、BIC_S、TEQ 或 TST 的 *Operand2* 时，进位标志更新为寄存器 Rm 的位[0]。

图180. RRX



39.2.3.5 地址对齐

对齐访问是将一个按字对齐的地址用于一个字、双字和多字的访问中，或将一个按半字对齐的地址用于半字访问中。字节访问总是对齐的。

Cortex-M3 处理器仅对下列指令支持非对齐的访问：

- LDR, LDRT
- LDRH, LDRHT
- LDRSH, LDRSHT
- STR, STRT
- STRH, STRHT

如果执行非对齐访问，则所有其他的加载和存储指令都会产生使用故障异常，因此，这些指令的访问都必须是地址对齐访问。有关使用故障的更多信息，请参见 39.3.4 “故障处理”。

非对齐访问通常慢于对齐访问。另外，某些存储区可能不支持非对齐访问。因此，ARM 建议，程序员应确保每次访问都是对齐访问。为了避免非对齐访问的偶然发生，应在配置和控制寄存器中使用 UNALIGN_TRP 位来捕获所有的非对齐访问，见 [39.4.3.8 “配置和控制寄存器”](#)。

39.2.3.6 相对 PC 的表达式

相对 PC 的表达式或**标签**是一个代表指令地址或立即数（literal data）的符号。在指令中，它表示为：PC 值加上或减去一个数值偏移量。汇编器根据标签和当前指令的地址计算出所需的偏移量。如果偏移量过大，则汇编器产生一个错误。

注

- 对于 B、BL、CBNZ 和 CBZ 指令，PC 值为当前指令地址加上 4 字节。
- 对于使用标签的所有其他指令，PC 值为当前指令地址加上 4 字节，结果的位[1]清零，使之按字对齐。
- 所用汇编器可能允许相对 PC 表达式的其他语法，例如一个标签加上或减去一个数字，或一个[PC, #number]形式的表达式。

39.2.3.7 条件执行

大多数数据处理指令都具有一个选项，该选项可以根据操作结果来更新**应用程序状态寄存器**（SPAR）中的条件标志，见 [39.3.1.3.5 “程序状态寄存器”](#)。有些指令会更新全部标志，有些指令则只更新一个子集。如果某个标志没有更新，则会保留其原始值。受影响的标志参见指令描述部分。

可以根据其他指令中设置的条件标志有条件地执行一条指令，执行时间为：

- 在更新标志的指令后立即执行；
- 在没有更新标志的任意数目的插入指令之后执行。

当使用条件跳转或为指令添加条件代码后缀时，可以实现有条件地执行。[表 755](#) 是一份后缀清单，这些后缀都可以添加到指令中使其可以有条件地执行。条件代码后缀使处理器能够根据标志来测试一个条件。如果某个条件指令的条件测试失败，则指令会：

- 不执行；
- 不向其目标寄存器写入任何值；
- 不影响任何标志；
- 不产生任何异常。

除了条件跳转外，条件指令必须处于一个 If-Then 指令模块中。有关使用 IT 指令的更多信息和限制，请参见 [39.2.9.3](#) 节。如果在 IT 块外存在条件指令，汇编器可能自动插入一条 IT 指令（取决于厂商）。

使用 CBZ 和 CBNZ 指令将寄存器值与零比较，并根据结果执行跳转（branch）。

本节介绍以下内容：

- [39.2.3.7.1 “条件标志”](#)
- [39.2.3.7.2 “条件代码后缀”](#)。

39.2.3.7.1 条件标志

APSR 包括以下条件标志：

N—操作结果为负值时设置为 1，否则清零。

Z—操作结果为零时设置为 1，否则清零。

C—操作导致一个进位出现时设置为 1，否则清零。

V—操作引起溢出时设置为 1，否则清零。

有关 APSR 的更多信息，请参见 [39.3.1.3.5 “程序状态寄存器”](#)。以下情况中出现进位：

- 如果加法结果大于或等于 2^{32} ；
- 如果减法结果为正值或零；
- 在一个移动或逻辑指令中，内联桶式（inline barrel）移位器操作的结果。

如果一个加法、减法或比较的结果大于或等于 2^{31} ，或小于 -2^{31} ，则发生溢出。

注：多数指令仅在指定后缀 S 的情况下才更新状态标志。更多信息请参见指令描述。

39.2.3.7.2 条件代码后缀

可有条件地执行的指令具有一个可选条件代码，如{cond}中的语法描述所示。条件执行时要求之前有一条 IT 指令。仅当 APSR 中的条件代码标志满足指定的条件时，才会执行带有条件代码的指令。[表 755](#) 显示了可使用的条件代码。

可以通过 IT 指令使用有条件地执行来减少代码中跳转指令的数量。

[表 755](#) 还显示了条件代码后缀和 N、Z、C 和 V 标志之间的关系。

表755. 条件代码后缀

后缀	标志	含义
EQ	Z = 1	等于
NE	Z = 0	不等于
CS 或 HS	C = 1	大于或等于（无符号≥）
CC 或 LO	C = 0	小于（无符号<）
MI	N = 1	负数

后缀	标志	含义
PL	$N = 0$	正数或零
VS	$V = 1$	溢出
VC	$V = 0$	无溢出
HI	$C = 1$ 且 $Z = 0$	大于（无符号 $>$ ）
LS	$C = 0$ 或 $Z = 1$	小于或等于（无符号 \leq ）
GE	$N = V$	大于或等于（有符号 \geq ）
LT	$N \neq V$	小于（有符号 $<$ ）
GT	$Z = 0$ 且 $N = V$	大于（有符号 $>$ ）
LE	$Z = 1$ 且 $N \neq V$	小于或等于（有符号 \leq ）
AL	任意值	始终。无指定后缀时的缺省值

示例：绝对值展示了使用一个条件指令来得到某个数的绝对值。`R0 = ABS (R1)`。

示例：比较和更新值展示了当符号值 R0 大于 R1 且 R2 大于 R3 时，使用条件指令来更新 R4 的值。

```

示例：绝对值：      MOVSB R0, R1      ; R0 = R1, setting flags
IT      MI            ; IT instruction for the negative condition
RSBMB   R0, R1, #0    ; If negative, R0 = -R1

```

```

示例：比较和更新值：    CMP  R0, R1  ; Compare R0 and R1,
setting flags
ITT  GT  ; IT instruction for the two GT conditions
CMPGT R2, R3 ; If 'greater than', compare R2 and R3, setting flags
MOVGT R4, R5 ; If still 'greater than', do R4 = R5

```

39.2.3.8 指令宽度选择

很多指令既可产生 16 位编码,也可产生 32 位编码,这取决于指定的操作数和目标寄存器。对其中的某些指令,可使用一个指令宽度后缀来强制设置一个特定的指令尺寸。“W”后缀会强制使用 32 位指令编码。“N”后缀会强制使用 16 位指令编码。

如果指定了一个指令宽度后缀,但汇编器无法生成所要求宽度的指令编码,则产生一个错误。

注:某些情况下指定.W后缀是非常有必要的,例如,当操作数为一个指令的标签或立即数时,如跳转指令的情况。这是因为汇编器可能无法自动生成大小正确的编码。

当使用指令宽度后缀时，将其紧跟在指令助记符和条件代码（如果有）后面。[39.2.3.8.1](#) 节列出了带指令宽度后缀的指令。

39.2.3.8.1 示例：指令宽度选择

```
BCS.W Label ; creates a 32-bit instruction even for a short branch
```

```
ADDS.W R0, R0, R1 ; creates a 32-bit instruction even though the same
                  ; operation can be done by a 16-bit instruction
```

39.2.4 内存访问指令

表 756 中列出了内存访问指令：

表756. 内存访问指令

助记符	简要描述	参见
ADR	加载相对 PC 地址	39.2.4.1
CLREX	独占清零	39.2.4.9
LDM{mode}	加载多个寄存器	39.2.4.6
LDR{type}	利用直接偏移量加载寄存器	39.2.4.2
LDR{type}	利用寄存器偏移量加载寄存器	39.2.4.3
LDR{type}T	通过非特权访问加载寄存器	39.2.4.4
LDR	利用相对 PC 地址加载寄存器	39.2.4.5
LDREX{type}	独占加载寄存器	39.2.4.8
POP	从堆栈弹出寄存器	39.2.4.7
PUSH	将寄存器推入堆栈	39.2.4.7
STM{mode}	存储多个寄存器	39.2.4.6
STR{type}	利用直接偏移量存储寄存器	39.2.4.2
STR{type}	利用寄存器偏移量存储寄存器	39.2.4.3
STR{type}T	通过非特权访问存储寄存器	39.2.4.4
STREX{type}	独占存储寄存器	39.2.4.8

39.2.4.1 ADR

加载相对 PC 地址。

39.2.4.1.1 语法

`ADR{cond} Rd, label`

其中：

cond 为一个可选的条件代码，见 [39.2.3.7 “条件执行”](#)。

Rd 为目标寄存器。

label 为一个相对 PC 表达式。参见[39.2.3.6 “相对 PC 的表达式”](#)。

39.2.4.1.2 操作

ADR 通过将一个立即值与 PC 值相加来确定地址，并将结果写入目标寄存器。

ADR 生成与位置无关的代码，因为地址相对于 PC。

如果使用 ADR 为 BX 或 BLX 指令生成目标地址，则必须确保将所生成地址的位[0]设为 1，才能保证正确执行。

label 的值必须在 PC 中地址的-4095 到+4095 范围内。

注：要得到最大偏移范围或生成未按字对齐的地址时可能必须使用到“.w”后缀。参见 [39.2.3.8 “指令宽度选择”](#)。

39.2.4.1.3 限制

Rd 不得为 SP 且不得为 PC。

39.2.4.1.4 条件标志

此指令不改变标志。

39.2.4.1.5 示例

```
ADR    R1, TextMessage    ; Write address value of a location labelled as  
                        ; TextMessage to R1
```


39.2.4.2 LDR 和 STR（直接偏移量）

带有直接偏移量、前变址直接偏移量或后变址直接偏移量的加载和存储。

39.2.4.2.1 语法

```
op{type}{cond} Rt, [Rn {, #offset}] ; 直接偏移量
op{type}{cond} Rt, [Rn, #offset]!   ; 前变址
op{type}{cond} Rt, [Rn], #offset     ; 后变址
opD{cond} Rt, Rt2, [Rn {, #offset}]  ; 直接偏移量, 双字
opD{cond} Rt, Rt2, [Rn, #offset]!    ; 前变址, 双字
opD{cond} Rt, Rt2, [Rn], #offset;    ; 后变址, 双字
```

其中:

Op 可为以下指令之一:

LDR: 加载寄存器。

STR: 存储寄存器。

Type 可以是下列项之一:

B: 无符号字节, 加载时零扩展到 32 位。

SB: 有符号字节, 符号扩展到 32 位 (仅 LDR)。

H: 无符号半字, 加载时零扩展到 32 位。

SH: 有符号半字, 符号扩展到 32 位 (仅 LDR)。

—: 如果是字, 则省略。

cond 为一个可选的条件代码, 见 [39.2.3.7 “条件执行”](#)。

Rt 为要加载或存储的寄存器。

Rn 为存储器地址所基于的寄存器。

offset 为一个 *Rn* 的偏移量。如果省略 *offset*, 则该地址为 *Rn* 的内容。

Rt2 为附加寄存器, 在双字操作时使用, 用于加载或存储。

39.2.4.2.2 操作

LDR 指令将存储器中的一个值加载到一个或两个寄存器中。

STR 指令将一个或两个寄存器值存储到存储器中。

带直接偏移量的加载和存储指令可用于以下寻址模式:

• 偏移量寻址

在寄存器 *Rn* 获得的地址上加上或减去一个偏移量。结果用于存储器的访问地址。寄存器 *Rn* 不改变。此模式的汇编语言语法为：

```
[Rn, #offset]
```

• 前变址寻址

在寄存器 *Rn* 获得的地址上加上或减去一个偏移量。结果用于存储器的访问地址，并写回寄存器 *Rn* 中。此模式的汇编语言语法为：

```
[Rn, #offset]!
```

• 后变址寻址

使用来自寄存器 *Rn* 的地址作为存储器的访问地址。给地址加上或减去一个偏移量，并写回寄存器 *Rn* 中。此模式的汇编语言语法为：

```
[Rn], #offset
```

要加载或存储的值可以是一个字节、半字、字或双字。字节和半字可以是有符号的，也可以是无符号的。参见 [39.2.3.5 “地址对齐”](#)。

[表 757](#) 显示了直接、前变址和后变址形式的偏移量范围。

表757. 偏移量范围

指令类型	直接偏移量	前变址偏移量	后变址偏移量
字、半字、有符号半字、字节或有符号字节	-255~4095	-255~255	-255~255
双字	-1020~1020 范围内 4 的倍数	-1020~1020 范围内 4 的倍数	-1020~1020 范围内 4 的倍数

39.2.4.2.3 限制

对于加载指令：

- 仅对字加载，*Rt* 可以是 SP 或 PC；
- 对双字加载，*Rt* 必须与 *Rt2* 不同；
- 前变址或后变址形式中，*Rn* 必须与 *Rt* 和 *Rt2* 不同。

当字加载指令中的 *Rt* 为 PC 时：

- 加载值的位[0]必须为 1 才能正确执行；
- 发生到当被加载值位[0]变为 0 时生成的一个地址的跳转；
- 如果指令是有条件的，则它必须是 IT 块中的最后一条指令。

对于存储指令：

- 仅对字存储，*Rt* 可以是 SP 或 PC；
- *Rt* 不得为 PC；
- *Rn* 不得为 PC；
- 前变址或后变址形式中，*Rn* 必须与 *Rt* 和 *Rt2* 不同。

39.2.4.2.4 条件标志

这些指令不改变标志。

39.2.4.2.5 示例

```
LDR      R8, [R10]           ; Loads R8 from the address in R10.
LDRNE    R2, [R5, #960]!     ; Loads (conditionally) R2 from a word
                               ; 960 bytes above the address in R5, and
                               ; increments R5 by 960.
STR       R2, [R9, #const-struct] ; const-struct is an expression evaluating
                               ; to a constant in the range 0-4095.
STRH      R3, [R4], #4        ; Store R3 as halfword data into address in
                               ; R4, then increment R4 by 4
LDRD      R8, R9, [R3, #0x20] ; Load R8 from a word 32 bytes above the
                               ; address in R3, and load R9 from a word 36
                               ; bytes above the address in R3
STRD      R0, R1, [R8], #-16   ; Store R0 to address in R8, and store R1 to
                               ; a word 4 bytes above the address in R8,
                               ; and then decrement R8 by 16.
```

39.2.4.3 LDR 和 STR（寄存器偏移量）

带有寄存器偏移量的加载与存储。

39.2.4.3.1 语法

$op\{type\}\{cond\} Rt, [Rn, Rm\{, LSL \#n\}]$

其中：

op 为以下指令之一：

LDR：加载寄存器。

STR：存储寄存器。

$type$ 为下列项之一：

B：无符号字节，加载时零扩展到 32 位。

SB：有符号字节，符号扩展到 32 位（仅 LDR）。

H：无符号半字，加载时零扩展到 32 位。

SH：有符号半字，符号扩展到 32 位（仅 LDR）。

—：如果是字，则省略。

$cond$ 为一个可选的条件代码，见 [39.2.3.7 “条件执行”](#)。

Rt 为要加载或存储的寄存器。

Rn 为存储器地址所基于的寄存器。

Rm 为一个寄存器，包含要用作偏移量的值。

$LSL \#n$ 为一个可选的移位， n 的范围为 0~3。

39.2.4.3.2 操作

LDR 指令将存储器的一个值加载到寄存器。

STR 指令将一个寄存器值存入存储器。

加载或存入的存储器地址为来自寄存器 Rn 的一个偏移量。偏移量由寄存器 Rm 指定，并可使用 LSL 左移最多 3 位。

要加载或存储的值可以是一个字节、半字，或字。对于加载指令，字节和半字可以是有符号的，也可以是无符号的。参见 [39.2.3.5 “地址对齐”](#)。

39.2.4.3.3 限制

在这些指令中：

- *Rn* 不得为 PC;
- *Rm* 不得为 SP 且不得为 PC;
- *Rt* 仅在字加载或字存储时才可以是 SP;
- *Rt* 仅在字加载时才可以是 PC。

当字加载指令中的 *Rt* 为 PC 时:

- 被加载值位[0]必须为 1 才能正确执行，并且发生到这个以半字对齐的地址的跳转;
- 如果指令是有条件的，它必须是 IT 块中的最后一条指令。

39.2.4.3.4 条件标志

这些指令不改变标志。

39.2.4.3.5 示例

```
STR    R0, [R5, R1]           ; Store value of R0 into an address equal to
                                ; sum of R5 and R1
LDRSB  R0, [R5, R1, LSL #1]    ; Read byte value from an address equal to
                                ; sum of R5 and two times R1, sign extended it
                                ; to a word value and put it in R0
STR    R0, [R1, R2, LSL #2]    ; Stores R0 to an address equal to sum of R1
                                ; and four times R2
```

39.2.4.4 LDR 和 STR（非特权）

带非特权访问的加载和存储。

39.2.4.4.1 语法

$op\{type\}T\{cond\} Rt, [Rn\{, \#offset\}]$; 直接偏移量

其中:

op 为下列指令之一:

LDR: 加载寄存器。

STR: 存储寄存器。

$type$ 为下列项之一:

B: 无符号字节, 加载时零扩展到 32 位。

SB: 有符号字节, 符号扩展到 32 位 (仅 LDR)。

H: 无符号半字, 加载时零扩展到 32 位。

SH: 有符号半字, 符号扩展到 32 位 (仅 LDR)。

—: 如果是字, 则省略。

$cond$ 为一个可选条件代码, 见 [39.2.3.7 “条件执行”](#)。

Rt 为要加载或存储的寄存器。

Rn 为存储器地址所基于的寄存器。

$offset$ 为 Rn 的偏移量, 取值范围为 0~255。如果省略 $offset$, 则该地址为 Rn 中的值。

39.2.4.4.2 操作

这些加载和存储指令完成与带直接偏移量的存储器访问指令相同的功能, 见 [39.2.4.2](#) 节。差别是, 即使在特权软件中, 这些指令也只能进行非特权访问。

当在非特权软件中使用时, 这些指令与带直接偏移量的正常存储器访问指令完全一样。

39.2.4.4.3 限制

在这些指令中:

- Rn 不得为 PC;
- Rt 不得为 SP 且不得为 PC。

39.2.4.4.4 条件标志

这些指令不改变标志。

39.2.4.4.5 示例

```
STRBTEQ    R4, [R7]           ; Conditionally store least significant byte in
                                ; R4 to an address in R7, with unprivileged access
LDRHT      R2, [R2, #8]       ; Load halfword value from an address equal to
                                ; sum of R2 and 8 into R2, with unprivileged access
```

39.2.4.5 LDR（相对 PC）

从存储器加载寄存器。

39.2.4.5.1 语法

LDR{type}{cond} Rt, label

LDRD{cond} Rt, Rt2, label ; 加载双字

type 为下列项之一：

- B：无符号字节，加载时零扩展到 32 位。
- SB：有符号字节，符号扩展到 32 位（仅 LDR）。
- H：无符号半字，加载时零扩展到 32 位。
- SH：有符号半字，符号扩展到 32 位（仅 LDR）。
- ：如果是字，则省略。

cond 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

Rt 为要加载或存储的寄存器。

Rt2 为第二个要加载或存储的寄存器。

label 为一个相对 PC 表达式。参见 [39.2.3.6 “相对 PC 的表达式”](#)。

39.2.4.5.2 操作

LDR 用一个相对 PC 存储器地址的值加载寄存器。存储器地址由标签指定，或由一个 PC 偏移量指定。

加载或存储的值可以是一个字节、半字或字。对于加载指令，字节和半字可以是有符号的，也可以是无符号的。参见 [39.2.3.5 “地址对齐”](#)。

label 必须在当前指令的限制范围内。[表 758](#) 显示了标签和 PC 之间的可能的偏移量。

表758. 偏移量范围

指令类型	偏移量范围
字、半字、有符号半字、字节、有符号字节	-4095~4095
双字	-1020~1020

注：要获得最大偏移量范围可能需要使用 “.w” 后缀。见 [39.2.3.8 “指令宽度选择”](#)。

39.2.4.5.3 限制

在这些指令中：

- 仅对字加载，*Rt* 可以是 SP 或 PC；
- *Rt2* 不得为 SP 且不得为 PC；
- *Rt* 必须与 *Rt2* 不同。

当一个字加载指令中的 *Rt* 为 PC 时：

- 被加载值位[0]必须为 1 才能正确执行，并且发生到这个以半字对齐的地址上的跳转。
- 如果指令是有条件的，它必须是 IT 块中的最后一条指令。

39.2.4.5.4 条件标志

这些指令不改变标志。

39.2.4.5.5 示例

```
LDR      R0, LookUpTable    ; Load R0 with a word of data from an address
                                ; labelled as LookUpTable
LDRSB    R7, localdata      ; Load a byte value from an address labelled
                                ; as localdata, sign extend it to a word
                                ; value, and put it in R7
```

39.2.4.6 LDM 和 STM

加载和存储多个寄存器。

39.2.4.6.1 语法

$op\{addr_mode\}\{cond\} Rn\{!\}, reglist$

其中：

op 为下列指令之一：

LDM：加载多个寄存器。

STM：存储多个寄存器。

$addr_mode$ 为下列项之一：

IA：每次访问后递增地址。这是默认设置。

DB：每次访问前递减地址。

$cond$ 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

Rn 为存储器地址所基于的寄存器。

“!” 为一个可选的写回后缀。如果有 “!”，则要加载或存储到的最终地址被写回到 Rn 。

$reglist$ 为要加载或存储的一个或多个寄存器列表，括在大括号内。它可以包含寄存器范围。如果包含一个以上的寄存器或寄存器范围，则必须以逗号隔开，见 [39.2.4.6.5](#) 节。

LDM 和 LDMFD 与 LDMIA 同义。LDMFD 指它用于从满降序堆栈（full descending stack）弹出（pop）数据。

LDMEA 与 LDMDB 同义，表示它用于从空升序堆栈（empty ascending stack）弹出数据。

STM 和 STMIA 与 STMIA 同义，STMEA 表示它用于向空升序堆栈推入（push）数据。

STMFD 与 STMDB 同义，STMFD 表示它用于向满降序堆栈推入数据。

39.2.4.6.2 操作

LDM 指令将基于 Rn 的存储器地址的字值加载到 $reglist$ 中的寄存器。

STM 指令将 $reglist$ 中寄存器的字值存储到基于 Rn 的存储器地址。

对于 LDM、LDMIA、LDMFD、STM、STMIA 和 STMEA，用于访问的存储器地址以 4 个字节为间隔，范围从 Rn 到 $Rn + 4 \times (n-1)$ ，其中 n 为 $reglist$ 中的寄存器序号。访问按照寄存器序号递增的顺序进行，最低序号的寄存器使用最低的存储器地址，最高序号的寄存器使用最高的存储器地址。如果指定了写回后缀，则 $Rn + 4 \times (n-1)$ 的值被写回到 Rn 。

对于 LDMDB、LDMEA、STMDB 和 STMTD，用于访问的存储器地址以 4 个字节为间隔，范围为从 Rn 到 $Rn - 4 \times (n-1)$ ，其中 n 为 *reglist* 中的寄存器序号。访问是按照寄存器序号递减的顺序进行，最高序号的寄存器使用最高的存储器地址，最低序号的寄存器使用最低的存储器地址。如果指定了写回后缀，则 $Rn - 4 \times (n-1)$ 的值被写回到 Rn 。

PUSH 和 POP 指令可以用此形式表示。详情参见 [39.2.4.7](#) 节。

39.2.4.6.3 限制

在这些指令中：

- Rn 不得为 PC；
- *reglist* 不得包括 SP；
- 在任何 STM 指令中，*reglist* 不得包括 PC；
- 在任何 LDM 指令中，如果 *reglist* 包括 LR，则它不得包括 PC；
- 如果指定了写回后缀，则 *reglist* 不得包括 Rn 。

STM 指令中，当 PC 包括在 *reglist* 中时：

- 加载到 PC 值的位[0]必须为 1 才能正确执行，并且发生到此半字对齐地址的跳转。
- 如果指令是有条件的，则它必须是 IT 块中的最后一条指令。

39.2.4.6.4 条件标志

这些指令不改变标志。

39.2.4.6.5 示例

```
LDM    R8,{R0,R2,R9}      ; LDMIA is a synonym for LDM
STMDB  R1!,{R3-R6,R11,R12}
```

39.2.4.6.6 不正确示例

```
STM    R5!,{R5,R4,R9} ; Value stored for R5 is unpredictable
LDM    R2, {}          ; There must be at least one register in the list
```

39.2.4.7 PUSH 和 POP

向一个满降序堆栈推入和弹出寄存器。

39.2.4.7.1 语法

`PUSH{cond} reglist POP{cond} reglist`

其中：

`cond` 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

`reglist` 为一个非空的寄存器列表，括在大括号内。它可以包括寄存器范围。如果包含了一个以上寄存器或寄存器范围，则必须以逗号隔开。

`PUSH` 和 `POP` 与 `STMDB` 和 `LDM`（或 `LDMIA`）同义，以存储器地址进行基于 `SP` 的访问，访问的最终地址被写回 `SP`。在这些情况下，`PUSH` 和 `POP` 是首选助记符。

39.2.4.7.2 操作

`PUSH` 按照寄存器序号递减的顺序将寄存器存储到堆栈上，最高序号的寄存器使用最高的存储器地址，最低序号的寄存器使用最低的存储器地址。

`POP` 按照寄存器序号递增的顺序从堆栈加载寄存器，最低序号的寄存器使用最低的存储器地址，最高序号的寄存器使用最高的存储器地址。

更多信息请参见 [39.2.4.6](#) 节。

39.2.4.7.3 限制

在这些指令中：

- `reglist` 不得包括 `SP`；
- 对于 `PUSH` 指令，`reglist` 不得包括 `PC`；
- 对于 `POP` 指令，如果 `reglist` 包括 `LR`，则它不得包括 `PC`。

在 `POP` 指令中，当 `PC` 包括在 `reglist` 中时：

- 被加载到 `PC` 值的位[0]必须为 1 才能正确执行，并且发生到此半字对齐地址的跳转。
- 如果指令是有条件的，它必须是 `IT` 块中的最后一条指令。

39.2.4.7.4 条件标志

这些指令不改变标志。

39.2.4.7.5 示例

```
PUSH    {R0,R4-R7}
PUSH    {R2,LR}
POP     {R0,R10,PC}
```

39.2.4.8 LDREX 和 STREX

独占加载和存储寄存器。

39.2.4.8.1 语法

LDREX{*cond*} *Rt*, [*Rn*{, #*offset*}]

STREX{*cond*} *Rd*, *Rt*, [*Rn*{, #*offset*}]

LDREXB{*cond*} *Rt*, [*Rn*]

STREXB{*cond*} *Rd*, *Rt*, [*Rn*]

LDREXH{*cond*} *Rt*, [*Rn*]

STREXH{*cond*} *Rd*, *Rt*, [*Rn*]

其中：

cond 为一个可选的条件代码，见 [39.2.3.7 “条件执行”](#)。

Rd 为存放返回状态的目标寄存器。

Rt 为要加载或存储的寄存器。

Rn 为存储器地址所基于的寄存器。

offset 为应用于 *Rn* 中的值的可选偏移量。如果省略 *offset*，则该地址为 *Rn* 中的值。

39.2.4.8.2 操作

LDREX、LDREXB 和 LDREXH 分别从存储器地址加载一个字、字节和半字。

STREX、STREXB 和 STREXH 分别尝试将一个字、字节和半字存储到某个存储器地址。独占存储指令中所用的地址必须与刚刚执行的独占加载指令中的地址相同。独占存储指令存储值的数据大小也必须与之前独占加载指令加载值的数据大小相同。这就是说，软件一定要用一条独占加载指令和一条匹配的独占存储指令来执行同步操作，见 [39.3.2.7 “同步原语”](#)。

如果一条独占存储指令执行存储，则它将 0 写入其目标寄存器。如果它未执行存储，则将 1 写入其目标寄存器。如果独占存储指令将 0 写入目标寄存器，则可保证在独占加载和独占存储指令之间，系统中没有其他进程访问过此存储单元。

出于性能考虑，尽量将相应独占加载和独占存储指令之间的指令数量降到最低。

注：如果执行一个独占存储指令的地址不同于之前独占加载指令的地址，则结果不可预知。

39.2.4.8.3 限制

在这些指令中：

- 不要使用 PC；
- *Rd* 和 *Rt* 不要使用 SP；
- 对于 STREX，*Rd* 必须与 *Rt* 和 *Rn* 都不同；
- *offset* 的值必须为 0~1020 范围内 4 的倍数。

39.2.4.8.4 条件标志

这些指令不改变标志。

39.2.4.8.5 示例

```
MOV      R1, #0x1           ; Initialize the 'lock taken' value
try
LDREX    R0, [LockAddr]     ; Load the lock value
CMP      R0, #0             ; Is the lock free?
ITT      EQ                 ; IT instruction for STREXEQ and CMPEQ
STREXEQ  R0, R1, [LockAddr  ; Try and claim the lock
CMPEQ    R0, #0             ; Did this succeed?
BNE      try                ; No - try again
....      ; Yes - we have the lock
```

39.2.4.9 CLREX

独占清零。

39.2.4.9.1 语法

CLREX{*cond*}

其中：

cond 为一个可选的条件代码，见 [39.2.3.7 “条件执行”](#)。

39.2.4.9.2 操作

使用 CLREX 使下一条 STREX、STREXB 或 STREXH 指令把 1 写入其目标寄存器且不执行存储。在异常处理程序代码中，当一个同步操作中独占加载指令和相应的独占存储指令之间发生异常时，它可用于强制独占存储失败。

更多信息请参见 [39.3.2.7 “同步原语”](#)。

39.2.4.9.3 条件标志

这些指令不改变标志。

39.2.4.9.4 示例

CLREX

39.2.5 通用数据处理指令

表 759 列出了数据处理指令：

表759. 数据处理指令

助记符	简要描述	参见
ADC	带进位加法	39.2.5.1
ADD	加法	39.2.5.1
ADDW	加法	39.2.5.1
AND	逻辑“与”	39.2.5.2
ASR	算术右移	39.2.5.3
BIC	位清零	39.2.5.2
CLZ	计算前导零数目	39.2.5.4
CMN	与负值比较	39.2.5.5
CMP	比较	39.2.5.5
EOR	异或	39.2.5.2
LSL	逻辑左移	39.2.5.3
LSR	逻辑右移	39.2.5.3
MOV	移动	39.2.5.6
MOVT	移动到顶部	39.2.5.7
MOVW	移动 16 位常数	39.2.5.6
MVN	取反移动	39.2.5.6
ORN	逻辑“或非”	39.2.5.2
ORR	逻辑“或”	39.2.5.2
RBIT	反转位	39.2.5.8
REV	反转字中的字节顺序	39.2.5.8
REV16	反转每个半字中的字节顺序	39.2.5.8
REVSH	反转低半字中的字节顺序，并将符号扩展	39.2.5.8
ROR	向右循环移	39.2.5.3
RRX	带扩展向右循环移	39.2.5.3
RSB	反向减法	39.2.5.1
SBC	带进位减法	39.2.5.1
SUB	减法	39.2.5.1
SUBW	减法	39.2.5.1
TEQ	相等测试	39.2.5.9
TST	测试	39.2.5.9

39.2.5.1 ADD、ADC、SUB、SBC 和 RSB

加法、带进位加法、减法、带进位减法和反向减法。

39.2.5.1.1 语法

$op\{S\}\{cond\}\{Rd,\} Rn, Operand\ 2$

$op\{cond\}\{Rd,\} Rn, \#imm12$; 仅 ADD 和 SUB

其中:

op 为下列项之一:

ADD: 加法。

ADC: 带进位加法。

SUB: 减法。

RSB: 反向减法。

S: 是一个可选后缀。如果指定了 *S*, 则会更新操作结果的条件代码标志, 见 [39.2.3.7 “条件执行”](#)。

cond: 是一个可选条件代码, 见 [39.2.3.7 “条件执行”](#)。

Rd 为目标寄存器。如果 *Rd* 被省略, 则目标寄存器为 *Rn*。

Rn 为存放第一个操作数的寄存器。

Operand2 为一个灵活的第二操作数。此选项详细说明请参见[39.2.3.3](#) 节。

imm12 可为 0~4095 范围内的任意值。

39.2.5.1.2 操作

ADD 指令将 *Rn* 中的值与 *Operand2* 或 *imm12* 中的值相加。

ADC 指令将 *Rn* 中的值与 *Operand2* 相加 (带进位标志)。

SUB 指令从 *Rn* 中的值减去 *Operand2* 或 *imm12* 的值。

SBC 指令从 *Rn* 中的值减去 *Operand2* 的值。如果进位标志被清除, 则结果减 1。

RSB 指令将 *Operand2* 的值减去 *Rn* 中的值。这是很有用的, 因为有了该指令, *Operand2* 的选项范围就会更大。

可以使用 ADC 和 SBC 进行合成多字运算, 请参见 [39.2.5.1.6](#) 节。

亦可参见 [39.2.4.1](#) 节。

注: ADDW 与使用 *imm12* 操作数的 ADD 语法等效。SUBW 与使用 *imm12* 操作数的 SUB 语法

等效。

39.2.5.1.3 限制

- *Operand2* 不得为 *SP* 且不得为 *PC*;
- 仅在 *ADD* 和 *SUB* 中, *Rd* 可以是 *SP*, 并有附加的限制:
 - *Rn* 必须也是 *SP*;
 - *Operand2* 中的任何移位都限于用 *LSL* 最多移 3 位;
- 仅在 *ADD* 和 *SUB* 中, *Rn* 可以是 *SP*;
- 仅在 *cond* 指令中, *Rd* 可以是 *PC*, 且其中:
 - 不得指定 *S* 后缀;
 - *Rm* 不得为 *PC* 且不得为 *SP*;
 - 如果指令是有条件的, 则必须是 *IT* 块中的最后一条指令。
- *cond* 指令有异常时, 仅在 *ADD* 和 *SUB* 中, *Rn* 可以是 *SP*, 且仅在下列附加限制下:
 - 不得指定 *S* 后缀;
 - 第二操作数必须是 0~4095 范围内的一个常数。

注

- 当使用 *PC* 进行加法或减法时, 计算前要将 *PC* 的位[1:0]舍入到 *b00*, 从而让计算的基址按字对齐。
- 如果想产生某条指令的地址, 则必须根据 *PC* 的值调整常数。*ARM* 建议使用 *ADR* 指令代替 *Rn* 等于 *PC* 的 *ADD* 或 *SUB* 指令, 因为汇编器会自动为 *ADR* 指令计算正确的常数。

当 *cond* 指令中的 *Rd* 为 *PC* 时:

- 写入 *PC* 的值的位[0]被忽略;
- 发生到将该值的位[0]强制为 0 生成的地址的跳转。

39.2.5.1.4 条件标志

如果指定了 *S*, 这些指令根据结果更新 *N*、*Z*、*C* 和 *V* 标志。

39.2.5.1.5 示例

```
ADD    R2, R1, R3
SUBS   R8, R6, #240      ; Sets the flags on the result
RSB    R4, R4, #1280     ; Subtracts contents of R4 from 1280
ADCHI  R11, R0, R3       ; Only executed if C flag set and Z
                        ; flag clear
```

39.2.5.1.6 多字算术操作示例

[64 位加法](#)中的两条指令将 *R2* 和 *R3* 中包含的一个 64 位整数与 *R0* 和 *R1* 中包含的另一个 64 位整数相加, 并将结果存入 *R4* 和 *R5*。

多字值不一定非要使用连续的寄存器。[96 位减法](#)中的指令从 R6、R2 和 R8 中包含的一个 96 位整数中减去 R9、R1 和 R11 中包含的一个 96 位整数。示例将结果存储在 R6、R9 和 R2 中。

64 位加法：

```
ADDS    R4, R0, R2    ; add the least significant words
ADC      R5, R1, R3    ; add the most significant words with carry
```

96 位减法：

```
SUBS    R6, R6, R9    ; subtract the least significant words
SBCS    R9, R2, R1    ; subtract the middle words with carry
SBC      R2, R8, R11   ; subtract the most significant words with carry
```

39.2.5.2 AND, ORR, EOR, BIC 和 ORN

逻辑“与”、“或”、“异或”、位清零和“或非”。

39.2.5.2.1 语法

op{*S*}{*cond*}{*Rd*,} *Rn*, *Operand2*

其中：

op 为下列项之一：

AND: 逻辑“与”。

ORR: 逻辑“或”，或置位。

EOR: 逻辑“异或”。

BIC: 逻辑“与非”，或位清零。

ORN: 逻辑“或非”。

S：是一个可选后缀。如果指定了 *S*，则会更新操作结果的条件代码标志，见 [39.2.3.7 “条件执行”](#)。

cond 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

Rd 为目标寄存器。

Rn 为保存第一个操作数的寄存器。

Operand2 是灵活的第二操作数。各选项详细说明请参见 [39.2.3.3](#) 节。

39.2.5.2.2 操作

AND、EOR 和 ORR 指令可对 *Rn* 和 *Operand2* 中的值按位进行“与”、“异或”和“或”操作。

BIC 指令对 *Rn* 中的位和 *Operand2* 值中相应位的补码进行“与”操作。

ORN 指令对 *Rn* 中的位和 *Operand2* 值中相应位的补码进行“或”操作。

39.2.5.2.3 限制

不使用 SP 且不使用 PC。

39.2.5.2.4 条件标志

如果指定了 S，则这些指令：

- 根据结果更新 N 和 Z 标志；
- 可以在 *Operand2* 运算期间更新 C 标志，见 [39.2.3.3](#) 节；
- 不影响 V 标志。

39.2.5.2.5 示例

```
AND R9, R2, #0xFF00
ORREQ R2, R0, R5
ANDS R9, R8, #0x19
EORS R7, R11, #0x18181818
BIC R0, R1, #0xab
ORN R7, R11, R14, ROR #4
ORNS R7, R11, R14, ASR #32
```

39.2.5.3 ASR, LSL, LSR, ROR 和 RRX

算术右移、逻辑左移、逻辑右移、向右循环移和带扩展向右循环移。

39.2.5.3.1 语法

$op\{S\}\{cond\} Rd, Rm, Rs$

$op\{S\}\{cond\} Rd, Rm, \#n$

$RRX\{S\}\{cond\} Rd, Rm$

其中：

op 为下列项之一：

ASR: 算术右移。

LSL: 逻辑左移。

LSR: 逻辑右移。

ROR: 向右循环移。

S ：是一个可选后缀。如果指定了 S ，则会更新操作结果的条件代码标志，见 [39.2.3.7 “条件执行”](#)。

Rd 为目标寄存器。

Rm 为存储将被移位数值的寄存器。

Rs 为存放移位长度的寄存器，所存放的移位长度应用于 Rm 中的值。仅使用最低有效字节，范围为 0~255。

n 为移位长度。不同指令的移位长度的范围为：

ASR: 移位长度为 1~32；

LSL: 移位长度为 0~31；

LSR: 移位长度为 1~32；

ROR: 移位长度为 1~31。

注：对于 $LSL\{S\}\{cond\} Rd, Rm, \#0$ ，首选语法为 $MOV\{S\}\{cond\} Rd, Rm$ 。

39.2.5.3.2 操作

ASR、LSL、LSR 和 ROR 将寄存器 Rm 中的位左移或右移，移位的数量由常数 n 或寄存器 Rs 指定。

RRX 将寄存器 Rm 中的位向右移 1 位。

在所有这些指令中，结果写到 *Rd*，但寄存器 *Rm* 中的值保持不变。欲详细了解不同指令产生的结果，请参见 [39.2.3.4 “移位操作”](#)。

39.2.5.3.3 限制

不使用 SP 且不使用 PC。

39.2.5.3.4 条件标志

如果指定了 S，则：

- 这些指令根据结果更新 N 和 Z 标志；
- 如果移位长度为 0，则不会影响 C 标志。否则，C 标志会更新为移出的最后一位见 [39.2.3.4 “移位操作”](#)。

39.2.5.3.5 示例

```
ASR    R7, R8, #9 ; Arithmetic shift right by 9 bits
LSLS   R1, R2, #3 ; Logical shift left by 3 bits with flag update
LSR    R4, R5, #6 ; Logical shift right by 6 bits
ROR    R4, R5, R6 ; Rotate right by the value in the bottom byte of R6
RRX    R4, R5     ; Rotate right with extend
```

39.2.5.4 CLZ

计算前导零数目。

39.2.5.4.1 语法

CLZ{cond} Rd, Rm

其中：

cond 为一个可选条件代码，见 39.2.3.7 节。

Rd 为目标寄存器。

Rm 为操作数寄存器。

39.2.5.4.2 操作

CLZ 指令对 Rm 中值中的前导零数目进行计算，并将结果返回到 Rd 中。如果未在源寄存器中设置任何位，则该结果值为 32，如果位[31]被置位，则结果值为零。

39.2.5.4.3 限制

不使用 SP 且不使用 PC。

39.2.5.4.4 条件标志

此指令不改变标志。

39.2.5.4.5 示例

```
CLZ      R4,R9
CLZNE    R2,R3
```

39.2.5.5 CMP 和 CMN

比较和与负值比较。

39.2.5.5.1 语法

CMP{cond} Rn, Operand2

CMN{cond} Rn, Operand2

其中：

cond 为一个可选条件代码，见 [39.2.3.7](#) 节。

Rn 为存放第一个操作数的寄存器。

Operand2 是灵活的第二操作数。各选项详细说明请参见 3-10 页关于灵活的第二操作数的内容。

39.2.5.5.2 操作

这些指令可比较寄存器中的值与 *Operand2* 比较。它们更新结果的条件标志，但不将结果写到寄存器。

CMP 指令从 *Rn* 中的值减去 *Operand2* 的值。这与 SUBS 指令功能相同，只是结果被丢弃。

CMN 指令给 *Rn* 中的值加上 *Operand2* 的值。这与 ADDS 指令功能相同，只是结果被丢弃。

39.2.5.5.3 限制

在这些指令中：

- 不使用 PC；
- *Operand2* 不得为 SP。

39.2.5.5.4 条件标志

这些指令根据结果更新 N、Z、C 和 V 标志。

39.2.5.5.5 示例

```
CMP      R2, R9
CMN      R0, #6400
CMPGT    SP, R7, LSL #2
```


39.2.5.6 MOV 和 MVN

移动和取反移动。

39.2.5.6.1 语法

MOV{S}{cond} *Rd*, *Operand2*

MOV{cond} *Rd*, #*imm16*

MVN{S}{cond} *Rd*, *Operand2*

其中：

S 为一个可选后缀。如果指定了 *S*，则会更新操作结果的条件代码标志，见 [39.2.3.7](#) 节。

cond 为一个可选条件代码，见 [39.2.3.7](#) 节。

Rd 为目标寄存器。

Operand2 是一个灵活的第二操作数。各选项详细说明请参见 3-10 页关于灵活的第二操作数的内容。

imm16 可为 0~65535 范围内的任意值。

39.2.5.6.2 操作

MOV 指令可将 *Operand2* 的值复制到 *Rd* 中。

当 MOV 指令中的 *Operand2* 为一个带移位而非 LSL #0 的寄存器时，首选语法为相应的移位指令：

- 对于 MOV{S}{cond} *Rd*, *Rm*, ASR #*n*, ASR{S}{cond} *Rd*, *Rm*, #*n* 为首选语法
- 对于 MOV{S}{cond} *Rd*, *Rm*, LSL #*n*, LSL{S}{cond} *Rd*, *Rm*, #*n* 为首选语法 (*n* if !=0)
- 对于 MOV{S}{cond} *Rd*, *Rm*, LSR #*n*, LSR{S}{cond} *Rd*, *Rm*, #*n* 为首选语法
- 对于 MOV{S}{cond} *Rd*, *Rm*, ROR #*n*, ROR{S}{cond} *Rd*, *Rm*, #*n* 为首选语法
- 对于 MOV{S}{cond} *Rd*, *Rm*, RRX, RRX{S}{cond} *Rd*, *Rm* 为首选语法。

另外，作为移位指令的同义词，MOV 指令还允许其他形式的 *Operand2*：

- MOV{S}{cond} *Rd*, *Rm*, ASR *Rs* 与 SR{S}{cond} *Rd*, *Rm*, *Rs* 同义
- MOV{S}{cond} *Rd*, *Rm*, LSL *Rs* 与 LSL{S}{cond} *Rd*, *Rm*, *Rs* 同义
- MOV{S}{cond} *Rd*, *Rm*, LSR *Rs* 与 LSR{S}{cond} *Rd*, *Rm*, *Rs* 同义
- MOV{S}{cond} *Rd*, *Rm*, ROR *Rs* 与 ROR{S}{cond} *Rd*, *Rm*, *Rs* 同义

见 [39.2.5.3](#) 节。

MOVN 指令先获取 *Operand2* 的值，然后对该值按位进行逻辑“非”操作，并将结果存入 *Rd*。

注：MOVW 指令提供与 MOV 相同的功能，但仅限于使用 *imm16* 操作数。

39.2.5.6.3 限制

仅在MOV 指令中可使用SP 和 PC，并有下列限制：

- 第二操作数必须为一个未带移位的寄存器；
- 不得指定 S 后缀。

当 MOV 指令中 *Rd* 为 PC 时：

- 写入 PC 的值的位[0]被忽略；
- 发生到强制使该值的位[0]为零产生的地址的跳转。

注：尽管可以将 MOV 作为一条跳转指令，但 ARM 强烈建议使用 BX 或 BLX 指令进行跳转，以保证软件对 ARM 指令集的可移植性。

39.2.5.6.4 条件标志

如果指定了 S，则这些指令：

- 根据结果更新 N 和 Z 标志；
- *Operand2* 运算过程中可以更新 C 标志，见 [39.2.3.3](#) 节；
- 不影响 V 标志。

39.2.5.6.5 示例

```
MOVS  R11, #0x000B    ; Write value of 0x000B to R11, flags get updated
MOV   R1, #0xFA05     ; Write value of 0xFA05 to R1, flags are not updated
MOVS  R10, R12         ; Write value in R12 to R10, flags get updated
MOV   R3, #23         ; Write value of 23 to R3
MOV   R8, SP          ; Write value of stack pointer to R8
MVNS  R2, #0xF        ; Write value of 0xFFFFFFFF (bitwise inverse of 0xF)
                        ; to the R2 and update flags
```

39.2.5.7 MOVT

移动到顶部。

39.2.5.7.1 语法

MOVT{*cond*} *Rd*, #*imm16*

其中：

cond 为一个可选条件代码，见 [39.2.3.7](#) 节。

Rd 为目标寄存器。

imm16 为一个 16 位立即常数。

39.2.5.7.2 操作

MOVT 可将一个 16 位立即值 *imm16* 写入其目标寄存器的高半字，*Rd*[31:16]中。该写操作不会影响 *Rd*[15:0]。

可使用 *MOV*、*MOVT* 指令对生成任意的 32 位常数。

39.2.5.7.3 限制

Rd 不得为 *SP* 且不得为 *PC*。

39.2.5.7.4 条件标志

此指令不改变标志。

39.2.5.7.5 示例

```
MOVT    R3, #0xF123 ; Write 0xF123 to upper halfword of R3, lower halfword
                        ; and APSR are unchanged
```

39.2.5.8 REV, REV16, REVSH 和 RBIT

在字或半字内反转字节或位的顺序。

39.2.5.8.1 语法

op{cond} Rd, Rn

其中：

op 为下列项之一：

- REV 反转字中的字节顺序。
- REV16 独立地反转每个半字中的字节顺序。
- REVSH 反转低半字中的字节顺序，并将符号扩展
- RBIT 反转 32 位字中的位的顺序。

cond 为一个可选条件代码，见 [39.2.3.7](#) 节。

Rd 为目标寄存器。

Rn 为存放操作数的寄存器。

39.2.5.8.2 操作

可使用这些指令来更改数据的字节序（endianness）：

- REV：将 32 位大端数据转换为小端数据，或将 32 位小端数据转换为大端数据。
- REV16 将 16 位大端数据转换为小端数据，或将 16 位小端数据转换为大端数据。
- REVSH 可完成以下任一转换：
 - 将 16 位带符号大端数据转换为 32 位带符号小端数据；
 - 将 16 位带符号小端数据转换为 32 位带符号大端数据。

39.2.5.8.3 限制

不使用 SP 且不使用 PC。

39.2.5.8.4 条件标志

这些指令不改变标志。

39.2.5.8.5 示例

```
REV    R3, R7 ; Reverse byte order of value in R7 and write it to R3
REV16  R0, R0 ; Reverse byte order of each 16-bit halfword in R0
REVSH  R0, R5 ; Reverse Signed Halfword
REVSH  R3, R7 ; Reverse with Higher or Same condition
RBIT   R7, R8 ; Reverse bit order of value in R8 and write the result to R7
```

39.2.5.9 TST 和 TEQ

位测试和相等测试。

39.2.5.9.1 语法

`TST{cond} Rn, Operand2`

`TEQ{cond} Rn, Operand2`

其中：

cond 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

Rn 为存放第一个操作数的寄存器。

Operand2 是一个灵活的第二操作数。各选项详细说明请参见 [39.2.3.3](#) 节。

39.2.5.9.2 操作

这些指令可利用 *Operand2* 来测试寄存器中的值。它们会更新结果的条件标志，但不会将结果写入任何寄存器中。

TST 指令对 *Rn* 中的值和 *Operand2* 的值按位进行“与”操作。除了结果会被丢弃以外，这与 ANDS 指令功能相同。

若要测试 *Rn* 的某个位是否为 0 或 1，使用带一个 *Operand2* 常数的 TST 指令，会使该位置位为 1，并清零所有其他位。

TEQ 指令对 *Rn* 中的值和 *Operand2* 的值按位进行“异或”操作。除了结果会被丢弃以外，这与 EORS 指令功能相同。

使用 TEQ 指令可在不影响 V 或 C 标志的情况下，测试两个值是否相等。

TEQ 也可用来测试值的符号。比较完毕后，两个操作数的符号位的逻辑“异或”运算结果将成为 N 标志。

39.2.5.9.3 限制

不得使用 SP 且不得使用 PC。

39.2.5.9.4 条件标志

这些指令：

- 根据结果更新 N 和 Z 标志；
- 在 *Operand2* 运算过程中能够更新 C 标志，见 [39.2.3.3](#) 节；
- 不影响 V 标志。

39.2.5.9.5 示例

```
TST      R0, #0x3F8    ; Perform bitwise AND of R0 value to 0x3F8,  
                        ; APSR is updated but result is discarded  
TEQEQ    R10, R9        ; Conditionally test if value in R10 is equal to  
                        ; value in R9, APSR is updated but result is discarded
```

39.2.6 乘法和除法指令

表 760 列出了乘法和除法指令：

表760. 乘法和除法指令

助记符	简要描述	参见
MLA	乘加，结果为 32 位	39.2.6.1
MLS	乘减，结果为 32 位	39.2.6.1
MUL	乘法，结果为 32 位	39.2.6.1
SDIV	有符号除法	39.2.6.3
SMLAL	有符号乘加（32x32+64），结果为 64 位	39.2.6.2
SMULL	有符号乘法（32 x 32），结果为 64 位	39.2.6.2
UDIV	无符号除法	39.2.6.3
UMLAL	无符号乘加（32x32+64），结果为 64 位	39.2.6.2
UMULL	无符号乘法（32 x 32），结果为 64 位	39.2.6.2

39.2.6.1 MUL、MLA 和 MLS

使用 32 位操作数的乘法、乘加和乘减，并产生一个 32 位的结果。

39.2.6.1.1 语法

$MUL\{S\}\{cond\}\{Rd,\} Rn, Rm$; 乘法

$MLA\{cond\} Rd, Rn, Rm, Ra$; 乘加

$MLS\{cond\} Rd, Rn, Rm, Ra$; 乘减

其中：

cond 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

S 是一个可选后缀。如果指定了 *S*，则会更新操作结果的条件代码标志，见 [39.2.3.7 “条件执行”](#)。

Rd 为目标寄存器。如果 *Rd* 被省略，则目标寄存器为 *Rn*。

Rn, *Rm* 为存放乘数的寄存器。

Ra 为存放被加数或被减数的寄存器。

39.2.6.1.2 操作

MUL 指令可将 *Rn* 和 *Rm* 中的值相乘，并将所得结果的低 32 位存入 *Rd*。

MLA 指令可将 *Rn* 和 *Rm* 中的值相乘，然后再将乘积与 *Ra* 的值相加，最后将所得结果的低 32 位存入 *Rd*。

MLS 指令可将 *Rn* 和 *Rm* 中的值相乘，然后再从 *Ra* 中的值减去乘积，最后将所得结果的低 32 位存入 *Rd*。

这些指令的结果与操作数是有符号还是无符号无关。

39.2.6.1.3 限制

在这些指令中，不得使用 *SP* 且不得使用 *PC*。

如果 *MUL* 指令使用 *S* 后缀：

- *Rd*、*Rn* 和 *Rm* 必须全部都在 R0~R7 范围内；
- *Rd* 必须与 *Rm* 相同；
- 不得使用 *cond* 后缀。

39.2.6.1.4 条件标志

如果指定了 S，则 MUL 指令：

- 根据结果更新 N 和 Z 标志；
- 不影响 C 和 V 标志。

39.2.6.1.5 示例

```
MUL    R10, R2, R5      ; Multiply, R10 = R2 x R5
MLA    R10, R2, R1, R5  ; Multiply with accumulate, R10 = (R2 x R1) + R5
MULS   R0, R2, R2       ; Multiply with flag update, R0 = R2 x R2
MULLT  R2, R3, R2       ; Conditionally multiply, R2 = R3 x R2
MLS    R4, R5, R6, R7   ; Multiply with subtract, R4 = R7 - (R5 x R6)
```

39.2.6.2 UMULL, UMLAL, SMULL 和 SMLAL

使用 32 位操作数的有符号长乘法和无符号长乘法，产生一个 64 位结果。

39.2.6.2.1 语法

$op\{cond\} RdLo, RdHi, Rn, Rm$

其中：

op 为下列项之一：

UMULL: 无符号长乘法。

UMLAL: 无符号长乘法，带累加。

SMULL: 有符号长乘法。

SMLAL: 有符号长乘法，带累加。

$cond$ 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

$RdHi$ 、 $RdLo$ 为目标寄存器。对于 UMLAL 和 SMLAL，它们还用于存放累加值。

Rn 、 Rm 为存放操作数的寄存器。

39.2.6.2.2 操作

UMULL 指令会将 Rn 和 Rm 中的值解释为无符号整数。它会先求这两个整数的乘积，然后将结果的低 32 位存入 $RdLo$ ，高 32 位存入 $RdHi$ 。

UMLAL 指令会将 Rn 和 Rm 中的值解释为无符号整数。它会先求这两个整数的乘积，然后将 64 位结果与 $RdHi$ 和 $RdLo$ 中的 64 位无符号整数相加，并将结果写回到 $RdHi$ 和 $RdLo$ 。

SMULL 指令会将 Rn 和 Rm 中的值解释为有符号整数的二进制补码。它会先求这两个整数的乘积，然后将结果的低 32 位存入 $RdLo$ ，高 32 位存入 $RdHi$ 。

SMLAL 指令会将 Rn 和 Rm 中的值解释为有符号整数的二进制补码。它会先求这两个整数的乘积，然后将 64 位结果与 $RdHi$ 和 $RdLo$ 中的 64 位有符号整数相加，并将结果写回到 $RdHi$ 和 $RdLo$ 。

39.2.6.2.3 限制

在这些指令中：

- 不得使用 SP 且不得使用 PC；
- $RdHi$ 和 $RdLo$ 必须为不同的寄存器。

39.2.6.2.4 条件标志

这些指令不影响条件代码标志。

39.2.6.2.5 示例

```
UMULL    R0, R4, R5, R6 ; Unsigned (R4,R0) = R5 x R6
SMLAL    R4, R5, R3, R8 ; Signed (R5,R4) = (R5,R4) + R3 x R8
```

39.2.6.3 SDIV 和 UDIV

有符号除法和无符号除法。

39.2.6.3.1 语法

SDIV{cond} {Rd}, Rn, Rm

UDIV{cond} {Rd}, Rn, Rm

其中：

cond 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

Rd 为目标寄存器。如果 Rd 被省略，则目标寄存器为 Rn。

Rn 为存放被除数的寄存器。

Rm 为存放除数的寄存器。

39.2.6.3.2 操作

SDIV对 Rn 中的值用 Rm 中的值进行有符号的整数除法。

UDIV 对 Rn 中的值用 Rm 中的值进行无符号的整数除法。

对于这两条指令来说，如果 Rn 中的值不能被 Rm 中的值整除，则结果向零舍入。

39.2.6.3.3 限制

不得使用 SP 且不得使用 PC。

39.2.6.3.4 条件标志

这些指令不改变标志。

39.2.6.3.5 示例

```
SDIV R0, R2, R4 ; Signed divide, R0 = R2/R4
UDIV R8, R8, R1 ; Unsigned divide, R8 = R8/R1
```

39.2.7 饱和指令

本节介绍饱和指令，SSAT 和 USAT。

39.2.7.1 SSAT 和 USAT

有符号饱和到任何位位置和无符号饱和到任何位位置，可选择在饱和前进行移位。

39.2.7.1.1 语法

$op\{cond\} Rd, \#n, Rm\{, shift\#s\}$

其中：

op 为下列项之一：

SSAT 可将一个有符号值饱和到一个有符号范围内。

USAT 可将一个有符号值饱和到一个无符号范围内。

$cond$ 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

Rd 为目标寄存器。

n 指定要饱和到的位位置：

- 对于 SSAT， n 的范围为 1~32。
- 对于 USAT， n 的范围为 0~31。

Rm 为存放待饱和值的寄存器。

$shift\#s$ 为一个可选的饱和前的移位，应用于 Rm 。它必须为下列项之一：

ASR $\#s$ ：其中， s 的范围为 1~31；

LSL $\#s$ ：其中， s 的范围为 0~31。

39.2.7.1.2 操作

这些指令饱和到一个有符号或无符号的 n 位值。

SSAT 指令会先进行指定的移位，然后将结果饱和到有符号范围 $-2^{n-1} \leq x \leq 2^{n-1}-1$ 。

USAT 指令会先进行指定的移位，然后将结果饱和到无符号范围 $0 \leq x \leq 2^n-1$ 。

对于使用 SSAT 的有符号 n 位饱和，这意味着：

- 如果待饱和的值小于 -2^{n-1} ，则返回结果为 -2^{n-1} ；
- 如果待饱和的值大于 $2^{n-1}-1$ ，则返回结果为 $2^{n-1}-1$ ；

- 否则，返回结果与待饱和的值相同。

对于使用 USAT 的无符号 n 位饱和，这意味着：

- 如果待饱和的值小于 0，则返回结果为 0；
- 如果待饱和的值大于 2^n-1 ，则返回结果为 2^n-1 ；
- 否则，返回结果与待饱和的值相同。

如果返回结果与待饱和的值不同，则称为**饱和**。如果发生了饱和，该指令在 APSR 中将 Q 标志设为 1。否则，它将保持 Q 标志不变。如要清零 Q 标志，必须使用 MSR 指令，见 [39.2.10.7](#) 节。

如要读取 Q 标志的状态，可使用 MRS 指令，见 [39.2.10.6](#) 节。

39.2.7.1.3 限制

不得使用 SP 且不得使用 PC。

39.2.7.1.4 条件标志

这些指令不影响条件代码标志。

如果发生饱和，这些指令将 Q 标志设置为 1。

39.2.7.1.5 示例

```
SSAT    R7, #16, R7, LSL #4 ; Logical shift left value in R7 by 4, then
                                ; saturate it as a signed 16-bit value and
                                ; write it back to R7
USATNE  R0, #7, R5           ; Conditionally saturate value in R5 as an
                                ; unsigned 7 bit value and write it to R0
```

39.2.8 位域指令

表 761 列出了对寄存器中位的相邻集（或称位域）操作的指令：

表761. 组合和分离指令

助记符	简要描述	参见
BFC	位域清零	39.2.8.1
BFI	位域插入	39.2.8.1
SBFX	有符号位域提取	39.2.8.2
SXTB	有符号扩展一个字节	39.2.8.3
SXTH	有符号扩展一个半字	39.2.8.3
UBFX	无符号位域提取	39.2.8.2
UXTB	用零扩展一个字节	39.2.8.3
UXTH	用零扩展一个半字	39.2.8.3

39.2.8.1 BFC 和 BFI

位域清零和位域插入。

39.2.8.1.1 语法

BFC{cond} Rd, #lsb, #width
BFI{cond} Rd, Rn, #lsb, #width

其中：

cond 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

Rd 为目标寄存器。

Rn 为源寄存器。

lsb 为位域最低有效位的位置。lsb 必须在 0~31 范围之内。

width 为位域宽度，必须在 1~32-lsb 范围内。

39.2.8.1.2 操作

BFC 清零寄存器中的一个位域。它从低位位置 lsb 开始，清零 Rd 中的 width 位。Rd 中的其他位不变。

BFI 将一个位域从一个寄存器复制到另一个寄存器。它用 Rn 从位[0]开始的 width 位替换 Rd 中从低位位置 lsb 开始的 width 位。Rd 中的其他位不变。

39.2.8.1.3 限制

不得使用 SP 且不得使用 PC。

39.2.8.1.4 条件标志

这些指令不影响标志。

39.2.8.1.5 示例

```
BFC  R4, #8, #12      ; Clear bit 8 to bit 19 (12 bits) of R4 to 0
BFI  R9, R2, #8, #12  ; Replace bit 8 to bit 19 (12 bits) of R9 with
                        ; bit 0 to bit 11 from R2
```


39.2.8.2 SBFX 和 UBFX

有符号位域提取和无符号位域提取。

39.2.8.2.1 语法

`SBFX{cond} Rd, Rn, #lsb, #width`

`UBFX{cond} Rd, Rn, #lsb, #width`

其中：

cond 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

Rd 为目标寄存器。

Rn 为源寄存器。

lsb 为位域中的最低有效位的位置。*lsb* 必须在 0~31 范围内。

width 为位域宽度，必须在 1~32-*lsb* 范围内。

39.2.8.2.2 操作

SBFX 从一个寄存器中提取一个位域，用符号将其扩展到 32 位，并将结果写到目标寄存器。

UBFX 从一个寄存器中提取一个位域，用零将其扩展到 32 位，并将结果写到目标寄存器。

39.2.8.2.3 限制

不得使用 SP 且不得使用 PC。

39.2.8.2.4 条件标志

这些指令不影响标志。

39.2.8.2.5 示例

```
SBFX R0, R1, #20, #4 ; Extract bit 20 to bit 23 (4 bits) from R1 and sign
                        ; extend to 32 bits and then write the result to R0.
UBFX R8, R11, #9, #10 ; Extract bit 9 to bit 18 (10 bits) from R11 and zero
                        ; extend to 32 bits and then write the result to R8
```

39.2.8.3 SXT 和 UXT

符号扩展和零扩展。

39.2.8.3.1 语法

$SXTextend\{cond\}\{Rd\}, Rm\{, ROR\#n\}$

$UXTextend\{cond\}\{Rd\}, Rm\{, ROR\#n\}$

其中：

extend 为下列项之一：

B：将一个 8 位值扩展为 32 位值。

H：将一个 16 位值扩展为 32 位值。

cond 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

Rd 为目标寄存器。

Rm 为存放待扩展值的寄存器。

ROR #*n* 为下列项之一：

ROR #8：将 *Rm* 中的值向右循环移 8 位。

ROR #16：将 *Rm* 中的值向右循环移 16 位。

ROR #24：将 *Rm* 中的值向右循环移 24 位。

如果 ROR #*n* 被省略，则不执行循环移位。

39.2.8.3.2 操作

这些指令执行以下操作：

1. 将 *Rm* 中的值向右循环移 0、8、16 或 24 位。
2. 从结果值中提取位：
 - SXTB 提取位[7:0]，并用符号扩展到 32 位。
 - UXTB 提取位[7:0]，并用零扩展到 32 位。
 - SXTH 提取位[15:0]，并用符号扩展到 32 位。
 - UXTH 提取位[15:0]，并用零扩展到 32 位。

39.2.8.3.3 限制

不得使用 SP 且不得使用 PC。

39.2.8.3.4 条件标志

这些指令不影响标志。

39.2.8.3.5 示例

```
SXTH  R4, R6, ROR #16 ; Rotate R6 right by 16 bits, then obtain the lower
                        ; halfword of the result and then sign extend to
                        ; 32 bits and write the result to R4.
UXTB  R3, R10          ; Extract lowest byte of the value in R10 and zero
                        ; extend it, and write the result to R3.
```

39.2.9 跳转和控制指令

表 762 列出了跳转和控制指令：

表762. 跳转和控制指令

助记符	简要描述	参见
B	跳转	39.2.9.1
BL	带链接的跳转	39.2.9.1
BLX	带链接的跳转并切换指令集	39.2.9.1
BX	跳转并切换指令集	39.2.9.1
CBNZ	比较，结果非零则跳转	39.2.9.2
CBZ	比较，结果为零则跳转	39.2.9.2
IT	If-Then	39.2.9.3
TBB	表跳转字节	39.2.9.4
TBH	表跳转半字	39.2.9.4

39.2.9.1 B、BL、BX 和 BLX

跳转指令。

39.2.9.1.1 语法

B{cond} lable

BL{cond} lable

BX{cond} Rm

BLX{cond} Rm

其中：
B 为跳转（立即）。

BL 为带链接跳转（立即）。

BX 为跳转并切换指令集（寄存器）。

BLX 为带链接的跳转并切换指令集（寄存器）。

cond 为一个可选条件代码，见 39.2.3.7 “条件执行”。

label 为一个相对 PC 表达式。见39.2.3.6 “相对 PC 的表达式”。

Rm 为一个寄存器，指示要跳转到的目标地址。Rm 中值的位[0]必须为 1，但跳转到的目标地址是通过将位[0]变为 0 生成的。

39.2.9.1.2 操作

所有这些指令都会引发跳转，或跳转到 lable，或跳转到 Rm 所指示的地址处。此外：

- BL 和 BLX 指令可将下一条指令的地址复制到链接寄存器（LR）R14 中。
- 如果 Rm 位[0]为 0，BX 和 BLX 指令会导致 UsageFault 异常。

Bcond 标签是唯一一个既可在 IT 块内又可在 IT 块外的条件指令。所有其他跳转指令在 IT 块内都必须是有条件的，在 IT 块外必须是无条件的，见 39.2.9.3 节。

表 763 列出了各种跳转指令的范围。

表763. 跳转范围

指令	跳转范围
B lable	-16MB~+16MB
Bcond lable (IT 块外)	-1MB~+1MB
Bcond lable (IT 块内)	-16MB~+16MB
BL{cond} lable	-16MB~+16MB
BX{cond} Rm	寄存器中的任意值
BLX{cond} Rm	寄存器中的任意值

注：要获得最大跳转范围，可能需要使用.w 后缀。参见 [39.2.3.8 “指令宽度选择”](#)。

39.2.9.1.3 限制

限制如下：

- BLX 指令中不使用 PC；
- 对于 BX 和 BLX，*Rm* 的位[0]必须为 1 才能正确执行指令，但会发生到将位[0]变为 0 所生成的目标地址的跳转。
- 如果这些指令中的任意一个处于一个 IT 块内，则它必须是 IT 块中的最后一条指令。

Bcond 是唯一一条不需要在 IT 块内的条件指令。但是，当它在 IT 块内时，具有较大的跳转范围。

39.2.9.1.4 条件标志

这些指令不改变标志。

39.2.9.1.5 示例

```

B      loopA    ; Branch to loopA
BLE    ng       ; Conditionally branch to label ng
B.W    target   ; Branch to target within 16MB range
BEQ    target   ; Conditionally branch to target
BEQ.W  target   ; Conditionally branch to target within 1MB
BL     funC     ; Branch with link (Call) to function funC, return address
                ; stored in LR
BX     LR       ; Return from function call
BXNE   R0       ; Conditionally branch to address stored in R0
BLX    R0       ; Branch with link and exchange (Call) to a address stored
                ; in R0
    
```

39.2.9.2 CBZ 和 CBNZ

比较，结果为零则跳转；比较，结果非零则跳转。

39.2.9.2.1 语法

CBZ *Rn*, *lable*

CBNZ *Rn*, *lable*

其中：

Rn 为存放操作数的寄存器。

lable 为跳转目标。

39.2.9.2.2 操作

可以使用 CBZ 或 CBNZ 指令避免改变条件代码标志并减少指令数目。

除了不改变条件标志外，CBZ *Rn*, *lable* 与下列指令序列功能相同：

```
CMP      Rn, #0
BEQ      lable
```

除了不改变条件标志外，CBNZ *Rn*, *lable* 与下列指令序列功能相同：

```
CMP      Rn, #0
BNE      lable
```

39.2.9.2.3 限制

限制如下：

- *Rn* 必须在 R0~R7 范围内；
- 跳转目标必须在指令之后的 4 到 130 个字节之内；
- 这些指令一定不能在一个 IT 块内使用。

39.2.9.2.4 条件标志

这些指令不改变标志。

39.2.9.2.5 示例

```
CBZ      R5, target ; Forward branch if R5 is zero
CBNZ     R0, target ; Forward branch if R0 is not zero
```

39.2.9.3 IT

If-Then (IT) 条件指令。

39.2.9.3.1 语法

$IT\{x\{y\{z\}\}\} \text{ cond}$

其中：

x 指定 IT 块中第二条指令的条件开关。

y 指定 IT 块中第三条指令的条件开关。

z 指定 IT 块中第四条指令的条件开关。

cond 指定 IT 块中第一条指令的条件。

IT 块中第二、第三和第四条指令的条件开关可以是下列项之一：

T: Then。将条件 **cond** 应用于指令。

E: Else。将 **cond** 的相反条件应用于指令。

注：在 IT 指令中，**cond** 可以使用 **AL** (“总是 (满足)” 条件)。如果这样，则 IT 块中所有指令都必须是无条件的，且每个 **x**、**y** 和 **z** 都必须为 **T** (不能是 **E**) 或省略。

39.2.9.3.2 操作

IT 指令最多可以构成 4 条后续条件指令。条件可以全部相同，或一部分条件可以是其他条件的逻辑反相。IT 指令之后的条件指令构成了 **IT 块**。

IT 块中的指令，包括所有跳转，都必须在其语法的{**cond**}部分指定条件。

注：汇编器也可以自动地为条件指令生成所需 IT 指令，从而不需要自己编写。详细情况请参见所用汇编器的文档。

IT 块中的 **BKPT** 指令总会得到执行，即使无法满足其条件也是如此。

在一条 IT 指令和相应的 IT 块之间 (或一个 IT 块内) 都会发生异常。如果发生此异常，则会进入到相应的异常处理程序，同时将适用的返回信息存入 **LR** 和压入堆栈的 **PSR** 中。

可像平常一样利用异常返回指令从异常中返回，IT 块将会继续正常执行。这是 **PC** 修改指令跳转到一个 IT 块中的指令的唯一方法。

39.2.9.3.3 限制

在 IT 块中不允许使用下列指令：

- **IT**
- **CBZ** 和 **CBNZ**

- CPSID 和 CPSIE。

使用 IT 块的其他限制有：

- 跳转指令或修改 PC 的任何指令或者必须处于 IT 块外，或者必须是 IT 块中的最后一条指令。这些指令有：
 - ADD PC、PC、Rm
 - MOV PC、Rm
 - B、BL、BX、BLX
 - LDM、LDR 或任何写入 PC 的 POP 指令
 - TBB 和 TBH
- 无法跳转到 IT 块内的任何指令，除非是从一个异常处理程序返回时。
- 除 **Bcond** 外，所有其他条件指令都必须在 IT 块内。**Bcond** 可以在 IT 块外，也可以在 IT 块内，但它在 IT 块内时具有较大的跳转范围。
- IT 块内每条指令都必须指定一个条件代码后缀。这些条件代码后缀可以是相同的，也可以是与块中其他指令的条件的逻辑反相。

注：所用的汇编器可能对 IT 块的使用有额外的限制，例如禁止在块内使用汇编器指令。

39.2.9.3.4 条件标志

此指令不改变标志。

39.2.9.3.5 示例

```
ITTE  NE      ; Next 3 instructions are conditional
ANDNE R0, R0, R1 ; ANDNE does not update condition flags
ADDSNE R2, R2, #1 ; ADDSNE updates condition flags
MOVEQ  R2, R3   ; Conditional move
CMP    R0, #9   ; Convert R0 hex value (0 to 15) into ASCII
                ; ('0'-'9', 'A'-'F')
ITE    GT      ; Next 2 instructions are conditional
ADDGT  R1, R0, #55 ; Convert 0xA -> 'A'
ADDLE  R1, R0, #48 ; Convert 0x0 -> '0'

IT     GT      ; IT block with only one conditional instruction
ADDGT  R1, R1, #1 ; Increment R1 conditionally

ITTEE  EQ      ; Next 4 instructions are conditional
MOVEQ  R0, R1   ; Conditional move
ADDEQ  R2, R2, #10 ; Conditional add
ANDNE  R3, R3, #1 ; Conditional AND
BNE.W  dloop    ; Branch instruction can only be used in the last
                ; instruction of an IT block
IT     NE      ; Next instruction is conditional
ADD    R0, R0, R1 ; Syntax error: no condition code used in IT block
```

39.2.9.4 TBB 和 TBH

表跳转字节和表跳转半字。

39.2.9.4.1 语法

TBB [*Rn*, *Rm*]

TBH [*Rn*, *Rm*, LSL #1]

其中：

Rn 为寄存器，包含跳转长度表的地址。

如果 *Rn* 为 PC，则表格地址为紧跟在 TBB 或 TBH 指令后的字节的地址。

Rm 为索引寄存器。用于存放到表格的索引。对于半字表，LSL #1 将 *Rm* 中的值加倍，以形成表中的正确偏移量。

39.2.9.4.2 操作

这些指令可使用一个单字节偏移表（TBB）或半字偏移表（TBH）来产生一个相对 PC 的向前跳转。*Rn* 可提供一个表的指针，而 *Rm* 可提供表的索引。对于 TBB，跳转偏移量是从表返回的无符号字节值的两倍；对于 TBH，跳转偏移量是从表返回的无符号半字值的两倍。跳转到的地址是紧跟 TBB 或 TBH 指令后字节地址偏移量的地址。

39.2.9.4.3 限制

限制如下：

- *Rn* 不得为 SP；
- *Rm* 不得为 SP 且不得为 PC；
- 这些指令中的任意一个在 IT 块中使用时，它必须是 IT 块中的最后一条指令。

39.2.9.4.4 条件标志

这些指令不改变标志。

39.2.9.4.5 示例

```

ADR.W  R0, BranchTable_Byte
TBB    [R0, R1]           ; R1 is the index, R0 is the base address of the
                           ; branch table

Case1
; an instruction sequence follows
Case2
; an instruction sequence follows
Case3
; an instruction sequence follows
BranchTable_Byte

```

```
DCB      0                ; Case1 offset calculation
DCB      ((Case2-Case1)/2) ; Case2 offset calculation
DCB      ((Case3-Case1)/2) ; Case3 offset calculation
TBH      [PC, R1, LSL #1]  ; R1 is the index, PC is used as base of the
                           ; branch table

BranchTable_H
DCI      ((CaseA - BranchTable_H)/2)  ; CaseA offset calculation
DCI      ((CaseB - BranchTable_H)/2)  ; CaseB offset calculation
DCI      ((CaseC - BranchTable_H)/2)  ; CaseC offset calculation

CaseA
; an instruction sequence follows
CaseB
; an instruction sequence follows
CaseC
; an instruction sequence follows
```

39.2.10 其他指令

[表 764](#) 列出了 Cortex-M3 的其他指令：

表764. 其他指令

助记符	简要描述	参见
BKPT	断点	39.2.10.1
CPSID	更改处理器状态，禁能中断	39.2.10.2
CPSIE	更改处理器状态，使能中断	39.2.10.2
DMB	数据内存屏障	39.2.10.3
DSB	数据同步屏障	39.2.10.4
ISB	指令同步屏障	39.2.10.5
MRS	将专用寄存器的内容移到通用寄存器中	39.2.10.6
MSR	将通用寄存器的内容移到专用寄存器中	39.2.10.7
NOP	不执行任何操作	39.2.10.8
SEV	发送事件	39.2.10.9
SVC	超级用户调用	39.2.10.10
WFE	等待事件	39.2.10.11
WFI	等待中断	39.2.10.12

39.2.10.1 BKPT

断点。

39.2.10.1.1 语法

`BKPT #imm`

其中：

imm 为一个表达式，其值为 0~255 范围内的一个整数（8 位值）。

39.2.10.1.2 操作

BKPT 指令可使处理器进入调试状态。当指令到达某个特定地址时，调试工具可以使用此指令来检查系统的状态。

imm 被处理器忽略。如果需要，调试器可用它来存储关于断点的附加信息。

BKPT 指令可放置在一个 IT 块内，但它会无条件执行，不受 IT 指令指定条件的影响。

39.2.10.1.3 条件标志

此指令不改变标志。

39.2.10.1.4 示例

```
BKPT 0xAB ; Breakpoint with immediate value set to 0xAB (debugger can
           ; extract the immediate value by locating it using the PC)
```

39.2.10.2 CPS

更改处理器状态。

39.2.10.2.1 语法

CPS *effect iflags*

其中：

effect 为下列项之一：

IE 清零专用寄存器。

ID 置位专用寄存器。

iflags 为一个或多个标志组成的序列：

i 置位或清零 PRIMASK。

f 置位或清零 FAULTMASK。

39.2.10.2.2 操作

CPS 会更改 PRIMASK 和 FAULTMASK 专用寄存器值。有关这些寄存器的更多信息，请参见 [39.3.1.3.6 “异常屏蔽寄存器”](#)。

39.2.10.2.3 限制

限制如下：

- 仅可通过特权软件使用 CPS，在非特权软件中使用该指令无效。
- CPS 不能是有条件指令，因此无法用于 IT 块内。

39.2.10.2.4 条件标志

此指令不改变条件标志。

39.2.10.2.5 示例

```
CPSID i ; Disable interrupts and configurable fault handlers (set PRIMASK)
CPSID f ; Disable interrupts and all fault handlers (set FAULTMASK)
CPSIE i ; Enable interrupts and configurable fault handlers (clear PRIMASK)
CPSIE f ; Enable interrupts and fault handlers (clear FAULTMASK)
```

39.2.10.3 DMB

数据内存屏障。

39.2.10.3.1 语法

DMB{*cond*}

其中：

cond 为一个可选的条件代码，见 [39.2.3.7 “条件执行”](#)。

39.2.10.3.2 操作

DMB 用作一种数据内存屏障。它可确保会先检测到程序中位于 DMB 指令前的所有显式内存访问指令，然后再检测到程序中位于 DMB 指令后的显式内存访问指令。DMB 不影响其他不访问存储器的指令的执行顺序。

39.2.10.3.3 条件标志

此指令不改变标志。

39.2.10.3.4 示例

```
DMB ; Data Memory Barrier
```

39.2.10.4 DSB

数据同步屏障。

39.2.10.4.1 语法

`DSB{cond}`

其中：

cond 为一个可选的条件代码，见 [39.2.3.7 “条件执行”](#)。

39.2.10.4.2 操作

DSB 是一种特殊的数据同步内存屏障。只有当 DSB 指令执行完毕后，才会执行程序中位于此指令后的指令。当 DSB 指令前的所有显式内存访问完成时，DSB 指令才会完成。

39.2.10.4.3 条件标志

此指令不改变标志。

39.2.10.4.4 示例

```
DSB ; Data Synchronisation Barrier
```


39.2.10.5 ISB

指令同步屏障。

39.2.10.5.1 语法

ISB{*cond*}

其中：

cond 为一个可选的条件代码，见 [39.2.3.7 “条件执行”](#)。

39.2.10.5.2 操作

ISB 用作指令同步屏障。它可刷新处理器中的流水线，以便在 ISB 指令完成后，才从缓存或存储器中提取位于该指令后的所有其他指令。

39.2.10.5.3 条件标志

此指令不改变标志。

39.2.10.5.4 示例

```
ISB ; Instruction Synchronisation Barrier
```

39.2.10.6 MRS

将一个专用寄存器的内容移到通用寄存器。

39.2.10.6.1 语法

`MRS{cond} Rd, spec_reg`

其中：

cond 为一个可选的条件代码，见 [39.2.3.7 “条件执行”](#)。

Rd 为目标寄存器。

spec_reg 可以是以下任意一个：APSR、IPSR、EPSR、IEPSR、IAPSR、EAPSR、PSR、MSP、PSP、PRIMASK、BASEPRI、BASEPRI_MAX、FAULTMASK 或 CONTROL。

39.2.10.6.2 操作

可将 MRS 与 MSR 配合使用作为一个更新 PSR 的读取-修改-写入序列的一部分，例如清除 Q 标志。

在进程交换代码中，必须保存换出进程的编程模型状态，包括 PSR 的相关内容。同样，也必须恢复换入进程的状态。这些操作将 MRS 用于状态保存指令序列，将 MSR 用于状态恢复指令序列。

注：与 MRS 指令共同使用时，BASEPRI_MAX 为 BASEPRI 的一个别名。

参见 [39.2.10.7](#) 节。

39.2.10.6.3 限制

Rd 不得为 SP 且不得为 PC。

39.2.10.6.4 条件标志

此指令不改变标志。

39.2.10.6.5 示例

```
MRS R0, PRIMASK ; Read PRIMASK value and write it to R0
```

39.2.10.7 MSR

将一个通用寄存器的内容移到指定的专用寄存器。

39.2.10.7.1 语法

`MSR{cond} spec_reg, Rn`

其中：

cond 为一个可选的条件代码，见 [39.2.3.7 “条件执行”](#)。

Rn 为源寄存器。

spec_reg 可以是以下任意一个：APSR、IPSR、EPSR、IEPSR、IAPSR、EAPSR、PSR、MSP、PSP、PRIMASK、BASEPRI、BASEPRI_MAX、FAULTMASK 或 CONTROL。

39.2.10.7.2 操作

MSR 中寄存器访问操作取决于特权级别。非特权软件只能访问 APSR，见 [表 768 “APSR 位分配”](#)。特权软件可访问所有专用寄存器。

在非特权软件中，对 PSR 中未分配状态位或执行状态位的写入被忽略。

注

当写入 BASEPRI_MAX 时，指令仅在以下情况之一时才写入 BASEPRI：

- *Rn* 为非零值且当前 BASEPRI 值为 0；
- *Rn* 为非零值且小于当前 BASEPRI 值。

参见 [39.2.10.6](#) 节。

39.2.10.7.3 限制

Rn 不得为 SP 且不得为 PC。

39.2.10.7.4 条件标志

此指令根据 *Rn* 中的值显式更新标志。

39.2.10.7.5 示例

`MSR CONTROL, R1 ; Read R1 value and write it to the CONTROL register`

39.2.10.8 NOP

不执行任何操作。

39.2.10.8.1 语法

`NOP{cond}`

其中：

cond 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

39.2.10.8.2 操作

NOP 不执行任何操作。NOP 未必一定是消耗时间的 NOP。也许在该指令执行前，处理器就会将其从流水线中删除。

可以使用 NOP 进行填充（padding），例如将后续指令置于 64 位边界上。

39.2.10.8.3 条件标志

此指令不改变标志。

39.2.10.8.4 示例

```
NOP ; No operation
```

39.2.10.9 SEV

发送事件。

39.2.10.9.1 语法

`SEV{cond}`

其中：

cond 为一个可选的条件代码，见 [39.2.3.7 “条件执行”](#)。

39.2.10.9.2 操作

SEV 是一条提示指令，向一个多处理器系统中的所有处理器发送事件信号。它还会将本地事件寄存器设置为 1，见 [39.3.5 “电源管理”](#)。

39.2.10.9.3 条件标志

此指令不改变标志。

39.2.10.9.4 示例

```
SEV ; Send Event
```

39.2.10.10 SVC

超级用户调用。

39.2.10.10.1 语法

`SVC{cond} #imm`

其中：

cond 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

imm 为一个表达式，其值为 0~255 范围内的一个整数（8 位值）。

39.2.10.10.2 操作

SVC 指令会引发一个 SVC 异常。

Imm 会被处理器忽略。如果需要，异常处理程序会将其恢复，借以确定所请求的服务。

39.2.10.10.3 条件标志

此指令不改变标志。

39.2.10.10.4 示例

```
SVC 0x32 ; Supervisor Call (SVC handler can extract the immediate value  
; by locating it via the stacked PC)
```

39.2.10.11 WFE

等待事件。

39.2.10.11.1 语法

WFE{*cond*}

其中：

cond 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

39.2.10.11.2 操作

WFE 为一条提示指令。

如果事件寄存器为 0，WFE 会挂起程序执行，直到发生任一以下事件后再恢复执行：

- 一个异常，除非被异常屏蔽寄存器或当前优先级所屏蔽；
- 一个异常进入挂起状态，前提是系统控制寄存器中的 SEVONPEND 位置位；
- 一个调试进入请求，前提是调试已使能。
- 一个外设或多处理器系统中的另一个处理器使用 SEV 指令发出了一个事件信号。

如果事件寄存器为 1，WFE 将其清零并立即返回。

更多信息参见[39.3.5 “电源管理”](#)。

39.2.10.11.3 条件标志

此指令不改变标志。

39.2.10.11.4 示例

```
WFE ; Wait for event
```

39.2.10.12 WFI

等待中断。

39.2.10.12.1 语法

WFI{*cond*}

其中：

cond 为一个可选条件代码，见 [39.2.3.7 “条件执行”](#)。

39.2.10.12.2 操作

WFI 是一条提示指令，它会挂起程序执行，直到发生以下事件之一：

- 一个异常；
- 一个调试进入请求，无论是否使能了调试。

39.2.10.12.3 条件标志

此指令不改变标志。

39.2.10.12.4 示例

```
WFI ; Wait for interrupt
```


39.3 ARM Cortex-M3 用户指南：处理器

39.3.1 编程模型

本节描述了 Cortex-M3 的编程模型。除了对各个内核寄存器的描述外，它还包括关于处理器模式以及软件执行和堆栈的特权等级的信息。

39.3.1.1 处理器模式和软件执行的特权等级

处理器模式包括：

- 线程模式
用于执行应用软件。处理器在退出复位时进入线程模式。
- 处理模式
用于处理异常。处理器在完成异常处理后返回线程模式。

软件执行的特权等级有：

- 非特权
软件：
 - 对 MSR 和 MRS 指令的有限访问权限，且不能使用 CPS 指令；
 - 不能访问系统定时器、NVIC 或系统控制模块；
 - 可能限制对存储器或外设的访问。

非特权软件在非特权等级执行。

- 特权
软件可使用所有指令，并可访问全部资源。

特权软件在特权等级执行。

在线程模式中，控制寄存器控制着软件执行是有特权的还是非特权的，见[表 774](#)。在处理模式中，软件执行总是有特权的。

只有特权软件才可写入控制寄存器，以改变线程模式下软件执行的特权等级。非特权软件可使用 SVC 指令执行[超级用户调用](#)，以将控制权转移给特权软件。

39.3.1.2 堆栈

处理器使用满降序堆栈。也就是说，堆栈指针指示最后推入堆栈存储器的项。当处理器将一个新项推入堆栈时，它会递减堆栈指针，然后将该项写入新的存储单元。处理器实现了两个堆栈：**主堆栈**和**进程堆栈**，它们有独立的堆栈指针副本，见[39.3.1.3.2](#)节。

在线程模式下，控制寄存器控制着处理器是使用主堆栈还是进程堆栈，见[表 774](#)。在处理模式下，处理器总是使用主堆栈。处理器工作的选项有：

表765. 处理器模式、执行特权级别和堆栈使用选择汇总

处理器模式	用于执行	软件执行的特权级别	使用的堆栈
线程	应用程序	特权或非特权 ^[1]	主堆栈或进程堆栈 ^[1]
处理	异常处理程序	总是特权	主堆栈

[1] 参见表 774。

39.3.1.3 内核寄存器

处理器的内核寄存器有：

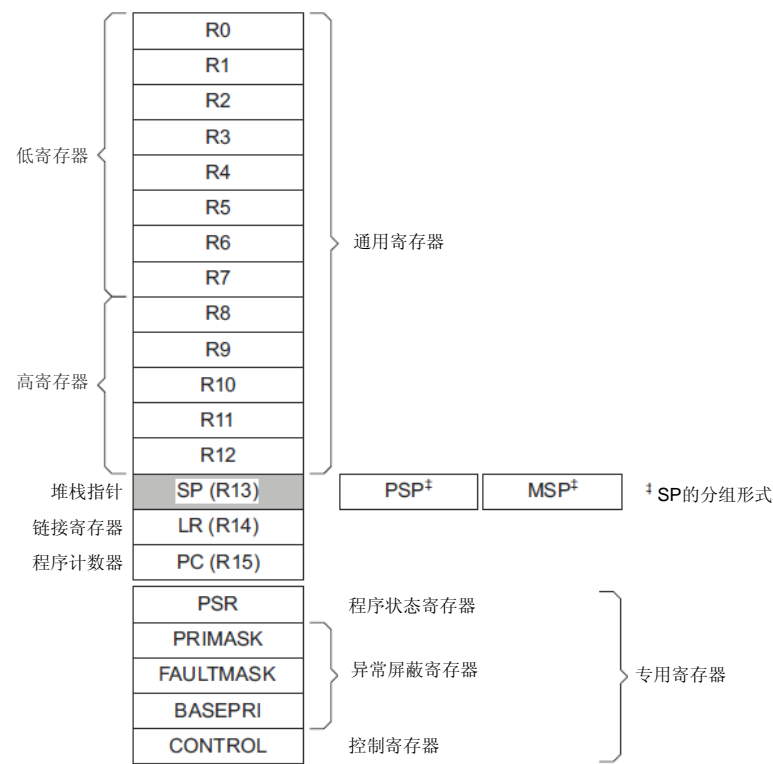


表766. 内核寄存器集汇总

名称	类型 ^[1]	所需特权 ^[2]	复位值	描述
R0-R12	RW	两者皆可	未定义	39.3.1.3.1
MSP	RW	特权	参见描述章节	39.3.1.3.2
PSP	RW	两者皆可	未定义	39.3.1.3.2
LR	RW	两者皆可	0xFFFFFFFF	39.3.1.3.3
PC	RW	两者皆可	参见描述章节	39.3.1.3.4
PSR	RW	特权	0x01000000	39.3.1.3.5
ASPR	RW	两者皆可	0x00000000	表 768
IPSR	RO	特权	0x00000000	表 769

名称	类型 ^[1]	所需特权 ^[2]	复位值	描述
EPSR	RO	特权	0x01000000	表 770
PRIMASK	RW	特权	0x00000000	表 771
FAULTMASK	RW	特权	0x00000000	表 772
BASEPRI	RW	特权	0x00000000	表 773
CONTROL	RW	特权	0x00000000	表 774

- [1] 描述线程模式和处理模式下程序执行过程中的访问类型。调试访问可以不同。
- [2] “两者皆可”意味着特权或非特权软件都可以访问该寄存器。

39.3.1.3.1 通用寄存器

R0-R12 为用于数据操作的 32 位通用寄存器。

39.3.1.3.2 堆栈指针

堆栈指针（SP）为寄存器 R13。在线程模式下，控制寄存器中的位[1]表示所使用的堆栈指针：

复位时，处理器将地址 0x00000000 的值加载到 MSP。

- 0 = **主堆栈指针**（MSP）。这是复位值。
- 1 = **进程堆栈指针**（PSP）。

39.3.1.3.3 链接寄存器

链接寄存器（LR）为寄存器 R14。它存储子程序、函数调用和异常的返回信息。复位时，处理器加载 LR 值 0xFFFFFFFF。

39.3.1.3.4 程序计数器

程序计数器（PC）为寄存器 R15。它包含当前的程序地址。位[0]总是为 0，因为指令的取指必须按半字对齐。复位时，处理器用复位向量的值加载 PC，复位向量在地址 0x00000004。

39.3.1.3.5 程序状态寄存器

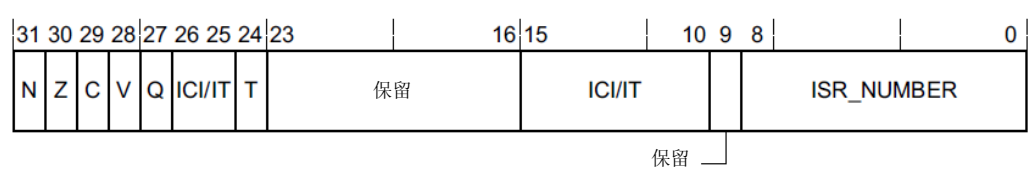
程序状态寄存器（PSR）组合了：

- **应用程序状态寄存器**（APSR）
- **中断程序状态寄存器**（IPSR）
- **执行程序状态寄存器**（EPSR）。

这些寄存器在 32 位 PSR 中为互斥的位域。位的分配如下：



PSR 位分配如下：



单独地访问这些寄存器，或访问任意两或三个寄存器的组合，可用寄存器名作为 MSR 或 MRS 指令的一个参数。例如：

- 使用带有 MRS 指令的 PSR 读取全部寄存器；
- 使用带有 MSR 指令的 APSR 写入 APSR。

PSR 组合和属性如下：

表767. PSR 寄存器组合

寄存器	类型	组合
PSR	RW ^[1] [2]	APSR、EPSR 与 IPSR
IEPSR	RO	EPSR 与 IPSR
IAPSR	RW ^[1]	APSR 与 IPSR
EAPSR	RW ^[2]	APSR 与 EPSR

- [1] 处理器忽略 IPSR 位的写入值。
- [2] 对 EPSR 位进行读操作返回结果为零且处理器忽略这些位的写入值。

关于如何访问程序状态寄存器的更多信息，请参见指令描述 [39.2.10.6 “MRS”](#) 和 [39.2.10.7 “MSR”](#)。

应用程序状态寄存器：APSR 包含了前面指令执行的条件标志的当前状态。其属性请参见[表 766](#) 中寄存器汇总。位分配如下：

表768. APSR 位分配

位	名称	功能
[31]	N	负数或小于标志： 0：操作结果为正数、零、大于或等于 1：操作结果为负数或小于
[30]	Z	零标志： 0：操作结果非零 1：操作结果为零
[29]	C	进位或借位标志： 0：加法操作未产生进位或减法操作产生了一个借位 1：加法操作产生了一个进位或减法操作未产生借位
[28]	V	溢出标志 0：操作未导致溢出发生 1：操作导致溢出发生
[27]	Q	粘着饱和（sticky saturation）标志： 0：复位或该位被最终清零后没有出现饱和 1：SSAT 或 USAT 指令导致了饱和的出现 该位由软件使用 MRS 指令清零。
[26:0]	-	保留

中断程序状态寄存器：IPSR 包含当前中断服务程序（ISR）的异常类型号。其属性请参见[表 766](#) 中寄存器汇总。位分配如下：

表769. IPSR 位分配

位	名称	功能
[31:9]	-	保留
[8:0]	ISR_NUMBER	该位为当前异常的序号。 0 = 线程模式 1 = 保留 2 = NMI 3 = 硬故障 4 = 存储器管理故障 5 = 总线故障 6 = 使用故障 7~10 = 保留 11 = SVCall 12 = 保留供调试使用 13 = 保留 14 = PendSV 15 = SysTick 16 = IRQ0 17 = IRQ1（第一个设备专用中断） . . 255 = IRQ243（最后执行的中断，取决于设备） 更多信息请参见 39.3.3.2 节。

执行程序状态寄存器：EPSR 包含以下所列项之一的 Thumb 状态位和执行状态位：

- **If-Then (IT)** 指令
- 用于一条中断多个加载或多个存储指令的**可中断-可继续指令 (ICI)** 域。

EPSR 属性，请参见表 766 中寄存器汇总。位分配为：

表770. EPSR 位分配

位	名称	功能
[31:27]	-	保留。
[26:25], [15:10]	ICI	可中断-可继续的指令位，参见 39.3.1.3.6。
[26:25], [15:10]	IT	IT 指令的执行状态位，参见 39.2.9.3 “IT”。
[24]	T	一直设置为 1。
[23:16]	-	保留。
[9:0]	-	保留。

如试图用 MSR 指令，直接通过应用软件读取 EPSR，则永远返回零。试图在应用软件中使用 MSR 指令写 EPSR，则会被忽略。故障处理程序可检查推入堆栈的 PSR 中 EPSR 的值，以指示出有故障的操作。参见 39.3.3.7 节。

可中断-可继续指令：当执行 LDM 或 STM 指令期间发生中断时，处理器执行下列操作：

处理中断以后，处理器：

- 暂时停止多个加载或多个存储指令的运行。
- 将多个操作中的下一个寄存器操作数保存到 EPSR 的位[15:12]。
- 返回位[15:12]指向的寄存器。
- 恢复执行多个加载或存储指令。

当 EPSR 保存有 ICI 执行状态时，位[26:25， 11:10]为零。

If-Then 块： If-Then 块在一个 16 位 IT 指令后可跟随最多 4 条指令。块中的每条指令都是有条件的。指令条件既可以全部相同，也可以是其它条件的逻辑反。更多信息请参见 [39.2.9.3 “IT”](#)。

39.3.1.3.6 异常屏蔽寄存器

异常屏蔽寄存器禁止由处理器处理异常。当异常可能影响到时序关键任务时禁用异常。

用 use MSR 和 MRS 指令访问异常屏蔽寄存器，或使用 CPS 指令来改变 PRIMASK 或 FAULTMASK 的值。更多信息请参见 [39.2.10.6 “MRS”](#)、[39.2.10.7 “MSR”](#) 和 [39.2.10.2 “CPS”](#)。

优先级屏蔽寄存器： PRIMASK 寄存器能阻止所有可配置优先级的异常的激活。其属性见 [表 766](#) 中寄存器汇总。位分配见 [表 771](#)。

表771. PRIMASK 寄存器的位分配

位	名称	功能
[31:1]	-	保留
[0]	PRIMASK	0：无影响 1：阻止所有可配置优先级的异常的激活

故障屏蔽寄存器： 故障屏蔽寄存器阻止除不可屏蔽中断（NMI）外所有异常的激活。其属性见 [表 766](#) 中寄存器汇总。位分配见 [表 772](#)。

表772. FAULTMASK 寄存器的位分配

位	名称	功能
[31:1]	-	保留
[0]	FAULTMASK	0：无影响 1：阻止除 NMI 外其他异常的激活

当从任意异常处理程序退出时（NMI 处理程序除外），处理器将 FAULTMASK 位清零。

基本优先级屏蔽寄存器： BASEPRI 寄存器定义了针对异常处理的最低优先级。当 BASEPRI 设置为一个非零值时，它阻止激活所有与 BASEPRI 值相同或更低优先级的异常。其属性见 [表 766](#) 中寄存器汇总。位分配见 [表 773](#)。

表773. BASEPRI 寄存器的位分配

位	名称	功能
[31:8]	-	保留
[7:0]	BASEPRI ^[1]	优先级屏蔽位： 0x0000：无影响 非零值：定义了异常处理的基本优先级 处理器不处理那些优先级值大于或等于 BASEPRI 值的异常。

[1] 该域与中断优先级寄存器中的优先级域相似。处理器仅执行该域的位[7:M]，位[M-1:0]被读为 0 且忽略写入值。M 值取决于所选设备。更多信息参见 [39.4.2.7“中断优先级寄存器”](#)。请记住，较高优先级域值对应较低异常优先级。

39.3.1.3.7 控制寄存器

当处理器处于线程模式时，控制寄存器控制所用堆栈和软件执行的特权等级。其属性见[表 766](#) 中寄存器汇总。位分配见[表 774](#)。

表774. 控制寄存器的位分配

位	名称	功能
[31:2]	-	保留
[1]	有效堆栈指针	定义了当前堆栈： 0: MSP 是当前堆栈指针 1: PSP 是当前堆栈指针 在处理模式下，该位读取为 0 且忽略写入值。
[0]	线程模式特权级别	定义了线程模式下的特权级别： 0: 特权 1: 非特权

处理模式始终使用 MSP，因此，当在处理模式下时，处理器会忽略对控制寄存器有效堆栈指针位的显式写入。异常进入和返回机制会更新控制寄存器。

在一个 OS 环境中，ARM 建议线程模式下运行的线程使用进程堆栈与内核，而异常处理程序使用主堆栈。

默认情况下，线程模式使用 MSP。如要将线程模式下使用的堆栈指针切换到 PSP，使用 MSR 指令将“有效”堆栈指针位设置为 1，见 [39.2.10.7 “MSR”](#)。

注：改变堆栈指针时，软件必须在 MSR 指令后立即使用一条 ISB 指令。这样可确保 ISB 后的指令使用新的堆栈指针执行操作。参见 [39.2.10.5 “ISB”](#)。

39.3.1.4 异常和中断

Cortex-M3 处理器支持中断和系统异常。处理器和可嵌套向量中断控制器（NVIC）确定所有异常的优先级并进行处理。一个异常会改变软件控制的正常流程。处理器使用处理模式来处理除复位外的所有异常。更多信息参见 [39.3.3.7.1](#) 节和 [39.3.3.7.2](#) 节。

NVIC 寄存器控制中断处理。更多信息参见 [39.4.2 “可嵌套向量中断控制器”](#)。

39.3.1.5 数据类型

处理器：

- 支持以下数据类型：
 - 32 位字
 - 16 位半字
 - 8 位字节
- 支持 64 位数据传输指令。
- 以小端形式管理所有的数据内存访问。更多信息参见[39.3.2.1](#) 节。

39.3.1.6 Cortex 微控制器软件接口标准

对于 Cortex-M3 微控制器系统，**Cortex 微控制器软件接口标准**（CMSIS）定义了：

- 一种通用方式，用来：
 - 访问外设寄存器；
 - 定义异常向量。
- 以下项的名称：
 - 内核外设的寄存器；
 - 内核异常向量。
- 用于 RTOS 内核的设备独立接口，包括一个调试通道。

CMSIS 包括 Cortex-M3 处理器中内核外设的地址定义和数据结构。它还包括可选的中间件组件接口，中间件组件包括 TCP/IP 堆栈和 Flash 文件系统。

CMSIS 能够重用各个中间件厂商的模板代码，以及符合 CMSIS 要求的软件组件组合，从而简化了软件开发工作。软件厂商可扩充 CMSIS，包括加入其外设定义，以及这些外设的访问函数。

本文件包括了 CMSIS 中定义的寄存器名称，并简要描述了用于访问处理器内核和内核外设的 CMSIS 函数。

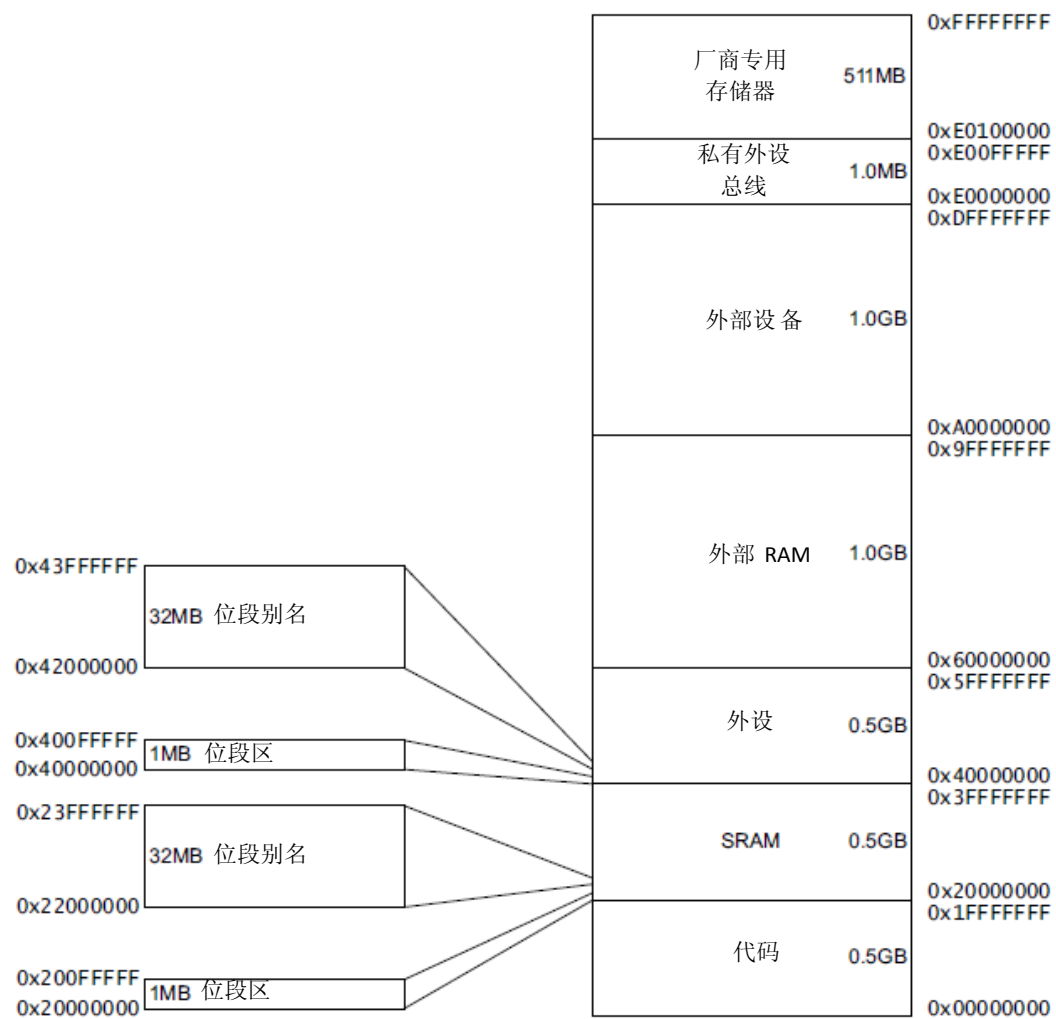
注：本文件使用了 CMSIS 定义的寄存器的简称。在个别情况下，这些名称可能不同于其他文件中使用的结构上的简称。

以下各节给出了关于 CMSIS 的更多信息：

- [39.3.5.4](#) 节
- [39.2.2 “内在函数”](#)
- [39.4.2.1 “Cortex-M3 NVIC 寄存器的 CMSIS 映射”](#)
- [39.4.2.10.1 “NVIC 编程提示”](#)。

39.3.2 存储器模型

本节描述了处理器存储器映射、存储器访问行为，以及位段（bit-banding）特性。处理器具有固定存储器映射，可提供高达 4GB 可寻址存储器。存储器映射为：



SRAM 和外设区包括了位段区。位段提供了对位数据的原子操作，见 [39.3.2.5](#) 节。

处理器将私有外设总线（PPB）区保留用于内核外设寄存器，见 [39.4.1 “关于 Cortex-M3 外设”](#)。

39.3.2.1 存储区、类型和属性

MPU 的存储器映射和编程将存储器映射分隔为区。每个区都有一个确定的存储器类型，有些区有附加的存储器属性。存储器类型和属性决定了对该区访问的行为。

存储器类型有：

- 正常型：处理器可将事务重新排序以改善效率，或执行猜读（speculative read）。
- 设备型：处理器保持相对于其他设备或非常有序型存储器的顺序。
- 非常有序型：处理器保持相对于全部其他事务的顺序。

设备型和非常有序型存储器有不同的排序要求，这意味着存储器系统可以缓冲对设备型存储器的写入，但不得缓冲对非常有序型存储器的写入。

其他存储器属性包括：

- **可共享**
对于一个可共享存储区，存储器系统可在一个系统的总线主机与多个总线主机之间（例如处理器与 DMA 控制器之间）提供数据同步。
非常有序型存储器总是可共享的。
如果多个总线主机可访问一个不可共享的存储区，则软件必须保证多个总线主机之间的数据一致性。
- **从不执行（XN）**
表示处理器阻止指令的访问。任何从 XN 区获取指令的尝试都会导致一个存储器管理故障异常。

39.3.2.2 存储器系统访问的排序

对于多数显式存储器访问指令引起的存储器访问，只要不影响指令序列的行为，存储器系统就不会保证访问完成的顺序与指令的程序顺序相匹配。通常，如果程序正确执行依赖于两个存储器访问按程序的顺序完成，则软件必须在存储器访问指令之间插入一条存储器屏障指令，见 39.3.2.4 节。

但是，存储器系统一定要保证对设备型和非常有序型存储器的一定访问次序。对于两条存储器访问指令 A1 和 A2，如果在程序顺序中，A1 在 A2 之前发生，则两条指令产生的存储器访问次序为：

A1 \ A2	正常访问	设备访问		非常有序访问
		非共享	可共享	
正常访问	-	-	-	-
设备访问，非共享	-	<	-	<
设备访问，可共享	-	-	<	<
非常有序访问	-	<	<	<

其中：

- “-”表示存储器系统不保证访问次序。
- “<”表示访问依照了程序顺序，即，A1 总在 A2 之前执行。

39.3.2.3 存储器访问行为

存储器映射中各区的访问行为如下：

表775. 存储器访问行为

地址范围	存储区	存储器类型	XN	描述
0x00000000~ 0x1FFFFFFF	代码	正常 ^[1]	-	编程代码的可执行区。也可在此放置数据。
0x20000000~ 0x3FFFFFFF	SRAM	正常 ^[1]	-	数据的可执行区。也可在此放置代码。 该区包含位段和位段别名区，参见表 776。
0x40000000~ 0x5FFFFFFF	外设	设备 ^[1]	XN ^[1]	该区包含位段区和位段别名区，参见表 777。
0x60000000~ 0x9FFFFFFF	外部 RAM	正常 ^[1]		数据的可执行区。
0xA0000000~ 0xDFFFFFFF	外部设备	设备 ^[1]	XN ^[1]	外部设备存储器。
0xE0000000~ 0xE0FFFFFF	私有外设总线	非常有序 ^[1]	XN ^[1]	该区包含 NVIC、系统定时器和系统控制模块。
0xE0100000~ 0xFFFFFFFF	厂商专用设备	设备 ^[1]	XN ^[1]	不用于 NXP 设备。

[1] 详细信息请参见 39.3.2.1 节。

代码、SRAM 和外部 RAM 区可保存程序。然而，最有效的程序访问是从代码区。这是因为处理器有单独的总线，能够同时进行指令获取和数据访问。

MPU 可以覆盖本节所述的默认存储器访问行为。更多信息见 39.4.5 “存储器保护单元”。

39.3.2.4 存储器访问的软件排序

程序流中指令顺序并不是总能保证对应存储器事务的顺序。这是因为：

- 只要不影响指令序列的行为，处理器可将一些存储器访问重新排序来提高效率。
- 处理器有多个总线接口。
- 存储器映射中的存储器或设备具有不同的等待状态。
- 有些存储器访问是缓冲的或者是猜测的。

39.3.2.2 节描述了存储器系统保证存储器访问顺序的情况。否则，如果存储器访问顺序很关键，软件必须包括存储器屏障指令以强制排序。处理器提供以下存储器屏障指令：

- DMB
数据内存屏障（DMB）指令确保未完成的存储器事务先于后续存储器事务完成。见 39.2.10.3 “DMB”。
- DSB
数据同步屏障（DSB）指令确保未完成的存储器事务在执行后续指令前完成。请参见 39.2.10.4 “DSB”。
- ISB

指令同步屏障 (ISB) 确保全部完成的存储器事务的效果可以被后续指令识别。请参见 [39.2.10.5 “ISB”](#)。

在下列示例情况下可使用存储器屏障指令：

对非常有序型存储器（例如系统控制模块）的存储器访问不需要使用 DMB 指令。

- MPU 编程：**
 - 使用一个 DSB 指令，确保在上下文切换结束时 MPU 立即生效。
 - 如果用一个跳转或调用访问了 MPU 配置代码，则使用一个 ISB 指令，以确保在一个或多个 MPU 区编程后，新的 MPU 设置立即生效。如果是通过异常机制输入 MPU 配置代码，则不需要 ISB 指令。
- 向量表。**如果程序改变了一个向量表项，然后使能了对应的异常，则要在两个操作间使用 DMB 指令。这样可以确保如果异常在使能后被立即获取，处理器使用新的异常向量。
- 自修改代码。**如果程序包含自修改代码，则在程序中的代码修改之后要立即使用一条 ISB 指令。这样可确保后续指令执行时使用更新后的程序。
- 存储器映射切换。**如果系统包含一个存储器映射切换机制，则在切换程序中的存储器映射后使用 DSB 指令。这样可确保后续指令执行时使用更新后的存储器映射。
- 动态异常优先级更改。**如果在异常挂起或有效期间，必须更改一个异常的优先级，则在更改后使用 DSB 指令。这样可确保更改在 DSB 指令完成时生效。
- 在多主机系统中使用一个信号量。**如果系统包含一个以上的总线主机（例如系统中有另一个处理器），则每个处理器都必须在任何信号量指令后使用一个 DMB 指令，才能确保其他总线主机能按照执行的顺序看到存储器事务。

39.3.2.5 位段

位段区将一个**位段别名区**中的每个字映射到**位段区**中的单个位。位段区占用 SRAM 和外设存储区的最低 1MB。

存储器映射中有两个 32MB 的别名区，它们映射到两个 1MB 的位段区：

- 对 32MB SRAM 别名区的访问映射到 1MB SRAM 位段区，如[表 776](#)所示；
- 对 32MB 外设别名的访问映射到 1MB 外设位段区，如[表 777](#)所示。

表776. SRAM 存储器位段区

地址范围	存储区	指令与数据访问
0x20000000 - 0x200FFFFFF	SRAM 位段区	直接访问此存储区与访问 SRAM 存储器一样，但是也可以通过位段别名区对此区进行位寻址。
0x22000000 - 0x23FFFFFF0	SRAM 位段别名区	对此区的数据访问被重映射到位段区。写操作的效果相当于读取-修改-写入。指令访问不能重映射。

表777. 外设存储器位段区

地址范围	存储区	指令与数据访问
0x40000000~ 0x400FFFFFF	外设位段别名区	对此存储区的直接访问与访问外设存储器一样，但是也可以通过位段别名区对此区进行位寻址。
0x42000000~ 0x44FFFFFF	外设位段区	对此区的数据访问被重映射到位段区。写操作的效果相当于读取-修改-写入。不允许指令访问。

注：对 SRAM 或外设位段别名区的字访问映射到 SRAM 或外设位段区中的单个位。

以下公式给出了别名区映射到位段区的方式：

`bit_word_offset = (byte_offset x 32) + (bit_number x 4)`

`bit_word_addr = bit_band_base + bit_word_offset`

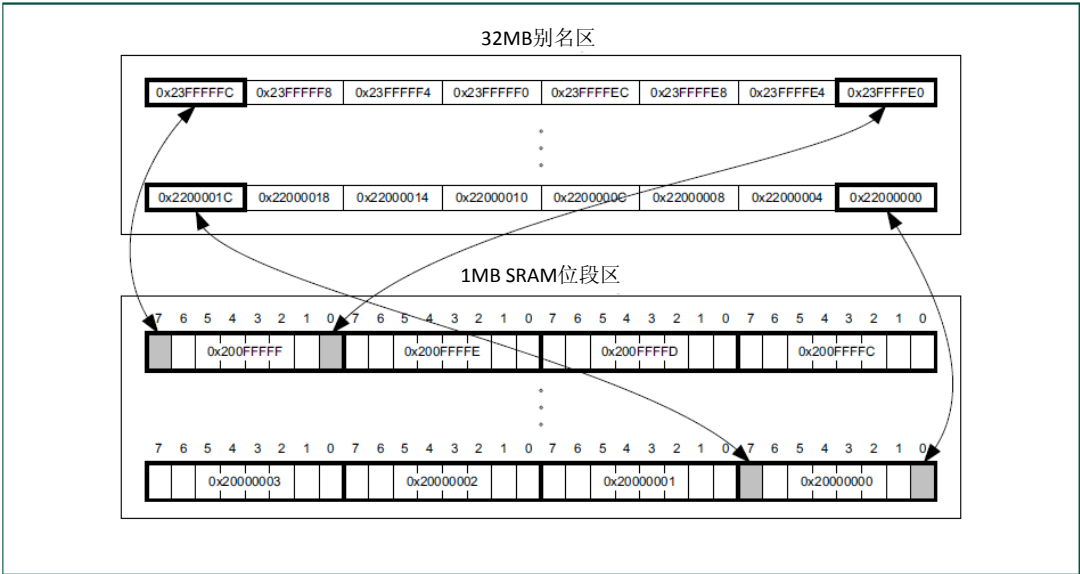
其中：

- `bit_word_offset` 是位段存储区中目标位的位置。
- `bit_word_addr` 是映射到目标位的字在别名存储区中的地址。
- `bit_band_base` 是别名区的起始地址。
- `byte_offset` 是包含目标位的位段区的字节号。
- `bit_number` 是目标位的位置 0-7。

图 181 给出了 SRAM 位段别名区和 SRAM 位段区之间的位段映射示例：

- 0x23FFFFE0 处的别名字映射到 0x200FFFFFF 处位段字节的位[0]：
 $0x23FFFFE0 = 0x22000000 + (0xFFFFF \times 32) + (0 \times 4)$ 。
- 0x23FFFFFC 处的别名字映射到 0x200FFFFFF 处位段字节的位[7]：
 $0x23FFFFFC = 0x22000000 + (0xFFFFF \times 32) + (7 \times 4)$ 。
- 0x22000000 处的别名字映射到 0x20000000 处位段字节的位[0]：
 $0x22000000 = 0x22000000 + (0 \times 32) + (0 \times 4)$ 。
- 0x2200001C 处的别名字映射到 0x20000000 处位段字节的位[7]：
 $0x2200001C = 0x22000000 + (0 \times 32) + (7 \times 4)$ 。

图181. 位段映射



39.3.2.5.1 直接访问一个别名区

向别名区中一个字执行写操作会更新位段区中的单个位。

向别名区中一个字写入值的位[0]决定了写入位段区中目标位的值。写入一个位[0]设为 1 的值，则会向位段位写入一个 1，写入一个位[0]设为 0 的值，则会向位段位写入一个 0。

别名字的位[31:1]对位段位无影响。写入 0x01 与写入 0xFF 效果相同。写入 0x00 与写入 0x0E 效果相同。

读取别名区中一个字：

- 0x00000000 表示位段区中目标位被设为 0；
- 0x00000001 表示位段区中目标位被设为 1。

39.3.2.5.2 直接访问一个位段区

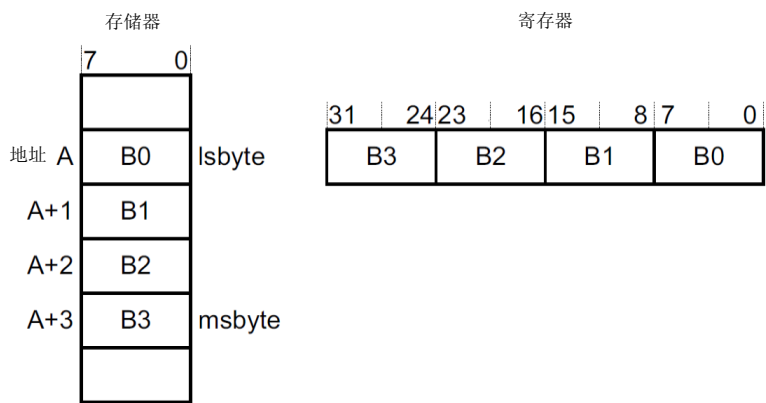
[39.3.2.3](#) 节描述了对位段区的直接字节、半字或字访问的行为。

39.3.2.6 存储器字节序

处理器将存储器视为一个线性的字节集合，从零开始递增序号。例如，字节 0-3 保存最先存储的字,字节 4-7 保存第二个存储的字。[39.3.2.6.1](#) 节描述了数据字如何存储在存储器中。

39.3.2.6.1 小端（Little-endian）格式

在小端格式中，处理器将一个字的最低（有效）位字节存储在序号最小的字节，最高位（有效）位字节存储在序号最大的字节。例如：



39.3.2.7 同步原语

Cortex-M3 指令集包括多对**同步原语**。这些原语提供了一种可以为线程或进程使用的非阻塞机制，从而获得某个存储单元的独占访问权限。软件可用它们来执行有保障的读取-修改-写入存储器更新序列，或实现一个信号量（**semaphore**）机制。一个同步原语对包括：

- 一条“独占加载”指令
用于读取某个存储单元的值，申请对该单元的独占访问。
- 一条“独占存储”指令
用于尝试写入相同的存储单元，返回一个状态位到寄存器。如果此位为：
 - 0：表示线程或进程获得对存储器的独占访问权限，写入成功；
 - 1：表示线程或进程未获得对存储器的独占访问权限，未进行写入。

“独占加载”和“独占存储”指令对有：

- 字指令 LDREX 和 STREX
- 半字指令 LDREXH 和 STREXH
- 字节指令 LDREXB 和 STREXB。

对于“独占存储”指令，软件必须使用相应的“独占加载”指令。

为了对某个存储单元进行有保障的读取-修改-写入，软件必须：

1. 使用一条“独占加载”指令，读取某个单元的值。
2. 按需要更新值。
3. 使用一条“独占存储”指令，尝试将新的值写回存储单元，并测试返回的状态位。如果此位为：
 - 0：读取-修改-写入成功完成；
 - 1：未完成读取-修改-写入。这表示步骤 1 中返回的值可能已过期。软件必须重试读取-修改-写入序列。

软件可使用同步原语来实现一个信号量，如下：

- 使用一条“独占加载”指令从信号量地址读取，检查信号量是否空闲。
- 如果信号量空闲，则使用“独占存储”将声明值（**claim value**）写入信号量地址。
- 如果步骤 2 返回的状态位指示“独占存储”成功，则软件已获得了信号量。但是，如果“独占存储”失败，则可能在软件执行步骤 1 后，有其他进程获得了信号量。

Cortex-M3 包括一个独占访问监控器，它会标注处理器已执行“独占加载”指令的情况。

如有以下情况，处理器会删除独占访问标签：

- 执行了一条 CLREX 指令。
- 无论写入是否成功，执行了一条“独占存储”指令。
- 发生一个异常。这表示处理器可解决不同线程间的信号量冲突。

有关同步原语指令的更多信息，请参见 [39.2.4.8 “LDREX 和 STREX”](#) 和 [39.2.4.9](#) 节。

39.3.2.8
同步原语的编程提示

ANSI C 不能直接产生独占访问指令。有些 C 编译器提供了可产生这些指令的内在函数：

表778.
独占访问指令的 C 编译器内在函数

指令	内在函数
LDREX,LDREXH 或 LDREXB	unsigned int __ldrex(volatile void *ptr)
STREX,STREXH 或 STREXB	int __strex(unsigned int val, volatile void *ptr)
CLREX	void __clrex(void)

实际产生的独占访问指令取决于传递给内在函数的指针数据类型。例如，以下 C 代码产生所需的 LDREXB操作：

```
__ldrex((volatile char *) 0xFF);
```

39.3.3
异常模型

本节描述了异常模型。

39.3.3.1
异常状态

每个异常都是以下几个状态之一：

- 无效
 - 异常无效且未被挂起。
- 挂起
 - 异常等待处理器的处理。
 - 来自外设或软件的中断请求可将相应中断的状态改为挂起。
- 有效
 - 异常正在由处理器处理，但尚未完成。

注：一个异常处理程序可以中断其他异常处理程序的执行。在此情况下，两个异常都处于有效状态。

- 有效和挂起

处理器正在处理一个异常，且有一个相同来源的处于挂起状态的异常。

39.3.3.2 异常类型

异常类型如下：

- 复位

上电或热复位时，复位启动。异常模型将复位作为一种特殊的异常形式。当复位使能时，处理器可能在一条指令的任何位置停止操作。当复位禁能时，由向量表中的复位表项提供重新开始执行的地址。在线程模式下，重新执行是特权执行。

- NMI

不可屏蔽中断（NMI）可由一个外设发出信号或由软件触发。这是除复位外的最高优先级异常。它被永久性使能，具有固定的优先级-2。NMI 不能：

- 因任何其他异常的激活而被屏蔽或阻止
- 被除复位外的其他任何异常抢占。

- 硬故障

硬故障是一种异常，发生原因是异常处理期间出错，或异常无法被任何其他异常机制管理。硬故障具有固定优先级-1，意味着它们的优先级高于任何具有可配置优先级的异常。

- 存储器管理故障

存储器管理故障是一种由存储器保护相关故障而引发的异常。对于指令和数据存储事务，MPU 或固定的存储器保护约束条件决定此故障。此故障用于中止对**从不执行（XN）**存储区的指令访问，即使 MPU 被禁能也是如此。

- 总线故障

总线故障是一种异常，由一个指令或数据存储事务的存储器相关故障引发。这可能源于从一个存储器系统总线上检测出的错误。

- 使用故障

使用故障是指令执行相关故障导致的一种异常。这包括：

- 未定义的指令
- 非法的非对齐访问
- 指令执行时的无效状态
- 异常返回时的错误。

当内核被配置为报告使用故障时，以下情况可以导致一个使用故障：

- 字和半字存储器访问时的非对齐地址；
- 除以零。

- SVCall

一次**超级用户调用（SVC）**是由 svc 指令触发的异常。在 OS 环境中，应用程序可使用 svc 指令来访问 OS 内核函数和设备驱动程序。

- PendSV
PendSV 是一个对系统级服务的中断驱动请求。在 OS 环境中，当无其他有效的异常时，使用 PendSV 进行上下文切换。
- SysTick
SysTick 异常是当系统定时器达到零时产生的异常。软件也可以产生一个 SysTick 异常。在 OS 环境中，处理器可将此异常用作系统节拍。
- 中断（IRQ）
中断（IRQ）是由一个外设发出信号，或由一个软件请求而产生的异常。所有中断都与指令执行异步。在系统中，外设使用中断与处理器进行通信。

表779. 不同异常类型的属性

异常序号 ^[1]	IRQ 序号 ^[1]	异常类型	优先级	向量地址或偏移量 ^[2]	激活
1	-	复位	-3, 最高	0x00000004	异步
2	-14	NMI	-2	0x00000008	异步
3	-13	硬故障	-1	0x0000000C	-
4	-12	存储器管理故障	可配置 ^[3]	0x00000010	同步
5	-11	总线故障	可配置 ^[3]	0x00000014	精确时是同步，不精确时是异步
6	-10	使用故障	可配置 ^[3]	0x00000018	同步
7-10	-	-	-	保留	-
11	-5	SVCall	可配置 ^[3]	0x0000002C	同步
12-13	-	-	-	保留	-
14	-2	PendSV	可配置 ^[3]	0x00000038	异步
15	-1	SysTick	可配置 ^[3]	0x0000003C	异步
16 及以上	0 及以上	中断（IRQ）	可配置 ^[4]	0x00000040 及以上 ^[5]	异步

[1] 为简化软件层级，CMSIS 仅使用 IRQ 序号，因此针对异常（而非中断）使用负值。IPSR 返回异常序号，参见表 769。

[2] 详细信息请参见 39.3.3.4 节。

[3] 参见 39.4.3.9 “系统处理程序优先级寄存器”。

[4] 参见 39.4.2.7 “中断优先级寄存器”。

[5] 以 4 为单位递增。

对于异步的异常，除复位外，在异常被触发和处理器进入异常处理程序之间，处理器也可以执行其他指令。

特权软件可以禁能表 779 中列出的具有可配置优先级的异常，见：

- 39.4.3.10 “系统处理程序控制和状态寄存器”
- 39.4.2.3 “中断清零使能寄存器”。

有关硬故障、存储器管理故障、总线故障和使用故障的更多信息，请参见 39.3.4 节。

39.3.3.3 异常处理程序

处理器使用以下程序来处理异常：

- 中断服务程序（ISR）
IRQ0 和更高级别的中断是由 ISR 处理的异常。
- 故障处理程序
硬故障、存储器管理故障、使用故障、总线故障是由故障处理程序处理的异常。
- 系统处理程序
NMI、PendSV、SVCall SysTick 和故障异常都是系统异常，由系统处理程序处理。

39.3.3.4 向量表

向量表包含了堆栈指针复位值和全部异常处理程序的起始地址，又称异常向量。[图 182](#) 列出了向量表中异常向量的顺序。每个向量的最低有效位必须为 1，这表示异常处理程序为 Thumb 代码。请注意 IRQ 序号的上限因设备而异。

图182. 向量表

异常序号	IRQ序号	偏移量	向量
127	111	0x1FC	IRQ111
.	.	.	.
.	.	.	.
.	.	.	.
18		0x004C	IRQ2
17	2	0x0048	IRQ1
16	1	0x0044	IRQ0
15	0	0x0040	Systick
14	-1	0x003C	PendSV
13	-2	0x0038	保留
12			保留供调试使用
11			SVCall
10	-5	0x002C	保留
9			
8			
7			
6	-10	0x0018	使用故障
5	-11	0x0014	总线故障
4	-12	0x0010	存储器管理故障
3	-13	0x000C	硬故障
2	-14	0x0008	NMI
1		0x0004	复位
		0x0000	初始SP值

系统复位时，向量表固定在地址 0x00000000。特权软件可通过写入 VTOR，将向量表起始地址重新定位到不同的存储单元，范围为 0x00000080 至 0x3FFFFFF80，见 [39.4.3.5 “向量表偏移量寄存器”](#)。

39.3.3.5 异常优先级

如[表 779](#)所示，所有异常都有一个相关的优先级，且：

- 用一个较低的优先级值表示一个较高的优先级；
- 除复位、硬故障和 NMI 外的其他异常都有可配置的优先级。

如果软件未配置任何优先级，则所有具有可配置优先级的异常都具有 0 优先级。有关异常优先级配置的信息，请参见：

- [39.4.3.9 “系统处理程序优先级寄存器”](#)
- [39.4.2.7 “中断优先级寄存器”](#)。

注：可配置的优先级值范围为 0 到 31。这就是说，具有固定的负数优先级值的复位、硬故障和 NMI 异常总是具有比其他异常更高的优先级。

例如，如果给 IRQ[0]赋以较高的优先级值并给 IRQ[1]赋以较低的优先级值，那么这就意味着 IRQ[1]的优先级高于 IRQ[0]。如果 IRQ[1]和 IRQ[0]都有效，则 IRQ[1]先于 IRQ[0]处理。

如果多个挂起的异常具有相同的优先级，则最先处理具有最低异常序号的异常。例如，如果 IRQ[0]和 IRQ[1]都挂起并具有相同的优先级，则 IRQ[0]先于 IRQ[1]被处理。

当处理器正在执行一个异常处理程序时，如果发生更高优先级的异常，则异常处理程序被抢占。如果发生了与正在处理的异常优先级相同的异常，无论异常序号如何，处理程序不会被抢占。但是，新中断的状态变为挂起。

39.3.3.6 中断优先级分组

为了对具有中断的系统加强优先级控制，NVIC 支持优先级分组。这将每个中断优先级寄存器项划分为两个域：

- 上区域定义了**组优先级**；
- 下区域定义了组内的**子优先级**。

只有组优先级才能决定中断异常的抢占。当处理器正在执行一个中断异常处理程序时，另一个与正在处理中的中断具有相同组优先级的中断不会抢占处理程序。

如果多个挂起的中断具有相同的组优先级，子优先级域决定了它们的处理顺序。如果多个挂起的中断具有相同的组优先级和子优先级，则具有最低 IRQ 序号的中断最先被处理。

有关中断优先级域拆分为组优先级和子优先级的相关信息，见 [39.4.3.6 “应用中断和复位控制寄存器”](#)。

39.3.3.7 异常进入和返回

使用以下术语来描述异常处理：

- 抢占

当处理器正在执行一个异常处理程序时，如果某个异常的优先级高于正在处理中异常的优先级，则这个异常可抢占异常处理程序。有关中断抢占的更多信息，请参见 [39.3.3.6](#) 节。

当某个异常抢占另一个异常的处理程序时，这些异常被称为嵌套异常。更多信息请参见 [39.3.3.7.1](#) 节。

- 返回

当异常处理程序完成且满足下列条件时，发生返回：

- 不存在挂起的异常，并且它们有被处理的足够优先级；
- 完成的异常处理程序并未在处理一个迟来（late-arriving）的异常。

处理器弹出堆栈，并将处理器状态恢复为中断发生前的状态。更多信息，请参见 [39.3.3.7.2](#) 节。

- 末尾连锁（Tail-chaining）

此机制可提高异常服务的速度。当一个异常处理程序完成时，如果有一个挂起的异常符合异常进入要求，则跳过堆栈弹出将控制权转移给新的异常处理程序。

- 迟来（late-arriving）

此机制可提高抢占速度。如果在前一异常状态保存期间发生了较高优先级的异常，处理器就会切换去处理较高优先级的异常，并启动对此异常的向量获取。状态保存不受迟来异常的影响，因为对这两个异常来说，保存的状态都是一样的。因此，状态保存可以不中断地继续进行。处理器可以接受迟来异常，直到初始异常的异常处理程序中的第一条指令进入处理器的执行阶段。在迟来异常的异常处理程序返回时，采用正常的末尾连锁规则。

39.3.3.7.1 异常进入

当存在一个有足够优先级的挂起异常，且满足以下条件之一时，发生异常进入：

- 处理器处于线程模式；
- 新的异常比正在处理中的异常具有更高优先级，在此情况下新的异常抢占原来的异常。

当一个异常抢占另一个异常时，异常被嵌套。

足够优先级指异常具有高于屏蔽寄存器所设置限值的优先级，见 [39.3.1.3.6](#) 节。优先级低于它的异常被挂起，但不被处理器处理。

当处理器取得一个异常时，除非异常为末尾连锁或迟来异常，否则，处理器会将信息推入当前堆栈。此操作被称为**入栈（stacking）**，而 8 个数据字的结构被称为**堆栈帧**。堆栈帧包含以下信息：

- R0-R3, R12;
- 返回地址;
- PSR;
- LR。

紧接着入栈之后，堆栈指针表示堆栈帧中的最低地址。除非禁用了堆栈对齐，否则，堆栈帧按双字地址对齐。如果配置控制寄存器（CCR）的 STKALIGN 位被设为 1，则在入栈期间要进行堆栈对齐调节。

堆栈帧包括返回地址。这是中断的程序中下一条指令的地址。在异常返回时，此值被恢复到 PC，以便让中断的程序继续运行。

入栈操作的同时，处理器执行向量取指，即从向量表读取异常处理程序的起始地址。当入栈完成时，处理器开始执行异常处理程序。同时，处理器将一个 EXC_RETURN 值写入 LR。用以指示哪个堆栈指针对应于堆栈帧，以及在异常进入发生前，处理器处于何种操作模式。

如果异常进入期间没有发生更高优先级的异常，则处理器开始执行异常处理程序，并自动地将相应挂起中断的状态改变为有效。

如果异常进入期间发生了更高优先级的异常，则处理器开始执行此异常的异常处理程序，且不改变前一异常的挂起状态。这就是迟来异常的情况。

39.3.3.7.2 异常返回

当处理器处于处理模式，并执行下列指令之一将 EXC_RETURN 值加载到 PC 时，发生异常返回：

- 一条包括 PC 的 POP 指令；
- 一条带任意寄存器的 BX 指令；
- 一条以 PC 为目标的 LDR 或 LDM 指令。

EXC_RETURN 为异常进入时加载到 LR 中的值。异常机制靠此值来检测处理器何时完成一个异常处理程序。此值的最低 4 位提供关于返回堆栈和处理器模式的信息。[表 780](#) 列出了 EXC_RETURN[3:0] 的值以及异常返回行为的描述。

处理器将 EXC_RETURN 的位[31:4]设为 0xFFFFFFFF。当此值被加载到 PC 中时，它向处理器指示异常已完成，处理器就会启动异常返回序列操作。

表780. 异常返回行为

EXC_RETURN[3:0]	描述
bXXX0	保留。
b0001	返回处理模式。 异常返回，获得来自 MSP 的状态。 返回后指令执行使用 MSP。
b0011	保留。
b01X1	保留。
b1001	返回线程模式。 异常返回，获得来自 MSP 的状态。 返回后指令执行使用 MSP。
b1101	返回线程模式。 异常返回，获得来自 PSP 的状态。 返回后指令执行使用 PSP。
b1X11	保留。

39.3.4 故障处理

故障为异常的一个子集，见 [39.3.3](#) 节。以下情况下会产生一个故障：

- 以下情况中的一个总线错误：
 - 指令取指或向量表加载
 - 数据访问
- 一个内部检测到的错误，例如未定义的指令，或试图用 BX 指令改变状态；
- 试图从标记为不可执行（XN）的存储区执行一个指令；
- 因特权违犯或试图访问一个未管理（unmanaged）区所引起的 MPU 故障。

39.3.4.1 故障类型

[表 781](#) 列出了故障类型、故障的处理程序、相应的故障状态寄存器和指示发生故障的寄存器位。关于故障状态寄存器的更多信息，请参见 [39.4.3.11 “可配置故障状态寄存器”](#)。

表781. 故障

故障	处理程序	位名称	故障状态寄存器
读向量总线错误	硬故障	VECTTBL	39.4.3.12 “硬故障状态寄存器”
升级为硬故障的故障		FORCED	
MPU 不匹配：	存储器管理故障	-	-
由于指令访问		IACCVIOL ^[1]	39.4.3.13 “存储器管理故障地址寄存器”
由于数据访问		DACCVIOL	
异常入栈期间		MSTKERR	
异常出栈期间		MUNSKERR	
总线错误：	总线故障	-	-
异常入栈期间		STKERR	39.4.3.14 “总线故障地址寄存器”
异常出栈期间		UNSTKERR	
指令预取期间		IBUSERR	
精确数据总线错误		PRECISERR	
非精确数据总线错误		IMPRECISERR	
试图访问一个协处理器	使用故障	NOCP	39.4.3.11.3 “存储器管理故障地址寄存器”
未定义指令		UNDEFINSTR	
试图进入一个无效的指令集状态 ^[2]		INVSTATE	
无效的 EXC_RETURN 值		INVPC	
非法的未对齐加载或存储		UNALIGNED	
除以 0		DIVBYZERO	

[1] 即使 MPU 被禁能，该操作也会在每次访问 XN 区时发生。
[2] 试图使用一个非 Thumb 指令集。

39.3.4.2 故障升级和硬故障

除了硬故障，所有故障异常都具有可配置的异常优先级，见 [39.4.3.9 “系统处理程序优先级寄存器”](#)。软件可禁止执行这些故障的处理程序，见 [39.4.3.10 “系统处理程序控制和状态寄存器”](#)。

寄存器”。

通常，异常优先级与异常屏蔽寄存器的值共同决定了处理器是否进入故障处理程序，以及一个故障处理程序是否可抢占另一个故障处理程序。如 39.3.3 节所述。

在某些情况下，有可配置优先级的故障可作为一个硬故障。这被称为**优先级升级**，该故障被描述为**升级到硬故障**。出现以下情况时发生升级到硬故障：

- 故障处理程序会产生与它正在处理的故障相同类型的故障。升级到硬故障的原因是故障处理程序不能抢占自身，因为它必须与当前优先级具有相同的优先级。
- 故障处理程序导致一个与它正在处理的故障具有相同或更低优先级的故障。这是因为新故障的处理程序不能抢占当前正在执行的故障处理程序。
- 异常处理程序导致一个与正在执行的异常具有相同或更低优先级的故障。
- 发生了一个故障，但此故障的处理程序未被使能。

如果在进入总线故障处理程序时，堆栈推入期间发生了一个总线故障，则总线故障不会升级到硬故障。这就是说，如果被破坏的堆栈导致一个故障发生，则即使处理程序的堆栈推入失败，故障处理程序也会执行。此故障处理程序可以工作，但堆栈内容被破坏。

注：只有复位和 NMI 可抢占固定优先级的硬故障。硬故障可抢占除复位、NMI 或另一硬故障外的任何异常。

39.3.4.3 故障状态寄存器和故障地址寄存器

故障状态寄存器指示故障的起因。对于总线故障和存储器管理故障来说，故障地址寄存器指示导致故障发生的操作所访问的地址，如表 782 所示。

表782. 故障状态和故障地址寄存器

处理程序	状态寄存器名称	地址寄存器名称	寄存器描述
硬故障	HFSR	-	39.4.3.12 “硬故障状态寄存器”
存储器管理故障	MMFSR	MMFAR	39.4.3.13 “存储器管理故障地址寄存器” 39.4.3.11.1 “存储器管理故障状态寄存器”
总线故障	BFSR	BFAR	39.4.3.11.2 “总线故障状态寄存器” 39.4.3.14 “总线故障地址寄存器”
使用故障	UFSR	-	39.4.3.11.3 “使用故障状态寄存器”

39.3.4.4 锁定

执行 NMI 或硬故障处理程序时，如果发生硬故障，则处理器进入一个锁定状态。当处理器处于锁定状态时，它不执行任何指令。处理器保持在锁定状态，直到发生以下情况之一：

- 处理器被复位；
- 发生一个 NMI 故障。

注：如果 NMI 处理程序使处理器进入锁定状态，则后续 NMI 也不会改变处理器的锁定状态。

39.3.5 电源管理

注：基于 Cortex-M3 处理器的 NXP 设备（包括 LPC178x/177x）都支持额外的降功率模式。有关全部可用的低功率模式的信息，请参见 [4.7 “功率控制”](#)。

Cortex-M3 处理器的睡眠模式可降低功耗：

- 睡眠模式会停止处理器时钟；
- 深度睡眠模式会停止系统时钟，并关闭 PLL 和 Flash 存储器的电源。

SCR 的 SLEEPDEEP 位用来选择使用哪种睡眠模式，见 [39.4.3.7 “系统控制寄存器”](#)。更多有关睡眠模式行为的信息，请参见 [4.7 “功率控制”](#)。

本节叙述了进入睡眠模式的机制和从睡眠模式唤醒的条件。

39.3.5.1 进入睡眠模式

本节叙述了软件用于将处理器置于睡眠模式的机制。

系统可能发生伪唤醒事件，例如，一个调试操作唤醒处理器。因此，软件必须能够在该类事件发生后，将处理器返回睡眠模式。程序可以有一个空闲循环（idle loop）来将处理器置回睡眠模式。

39.3.5.1.1 等待中断

等待中断指令 WFI 会使处理器立即进入睡眠模式。当处理器执行 WFI 指令时，处理器会停止执行指令并进入睡眠模式。更多信息，请参见 [39.2.10.12 “WFI”](#)。

39.3.5.1.2 等待事件

注：基于 Cortex-M3 处理器的 LPC178x/177x 设备不执行外部事件。

等待事件指令 WFE 使处理器根据一个单一位事件寄存器的值，有条件地进入睡眠模式。当处理器执行 WFE 指令时，它会检查该寄存器的下列内容：

- 如果该寄存器为 0，处理器停止执行指令并进入睡眠模式；
- 如果该寄存器为 1，处理器将该寄存器清零，不进入睡眠模式并继续执行指令。

更多信息，请参见 [39.2.10.11 “WFE”](#)。

如果事件寄存器为 1，就表示当执行 WFE 指令时，处理器不得进入睡眠模式。通常，这是因为使能了一个外部事件信号，或系统中的一个处理器执行了一条 SEV 指令，请参见 [39.2.10.9 “SEV”](#)。软件不能直接访问此寄存器。

39.3.5.1.3 退出时睡眠 (Sleep-on-exit)

如果 SCR 的 SLEEPONEXIT 位设为 1，则当处理器执行完一个异常处理程序时会返回到线程模式，并立即进入睡眠模式。在发生异常时，只在那些需要处理器运行的应用中使用此机制。

39.3.5.2 从睡眠模式唤醒

处理器唤醒的条件取决于导致处理器进入睡眠模式的机制。

39.3.5.2.1 从“WFI”或“退出时睡眠”唤醒

正常情况下，只有在处理器检测到一个具有足够优先级且可造成异常进入的异常时，处理器才会唤醒。

某些嵌入式系统可能必须在处理器唤醒后和执行一个中断处理程序前，先执行系统恢复任务。为了实现该操作，将 PRIMASK 位设为 1，并将 FAULTMASK 位设为 0。如果有一个中断到达，该中断被使能且具有比当前异常优先级更高的优先级，则处理器唤醒，但不执行中断处理程序，直到处理器将 PRIMASK 设为零。有关 PRIMASK 和 FAULTMASK 的更多信息，请参见 [39.3.1.3.6](#) 节。

39.3.5.2.2 从“WFE”唤醒

如有以下情况，处理器被唤醒：

- 处理器探测到一个有足够优先级的异常，可产生异常进入。

另外，如果 SCR 中 SEVONPEND 位被设为 1，则任何新挂起的中断都会触发一个事件并唤醒处理器，哪怕该中断被禁止或没有足够高的优先级来产生异常进入。更多关于 SCR 的信息，请参见 [39.4.3.7 “系统控制寄存器”](#)。

39.3.5.3 唤醒中断控制器

唤醒中断控制器 (WIC) 是一个外设，它可以检测一个中断并将处理器从深度睡眠模式、掉电或深度掉电模式唤醒。只有在 SCR 中 DEEPSLEEP 位被设为 1 时，WIC 才被使能。见 [39.4.3.7 “系统控制寄存器”](#)。

注：NXP 微控制器扩展了降功率模式的数量，使之超出了 Cortex-M3 直接支持的模式数量。关于全部可用降功率模式和对 LPC178x/177x 唤醒可能性的详细说明，请参见 [4.7 “功率控制”](#)。

WIC 不可编程，且没有任何寄存器或用户接口。它完全依靠来自硬件的信号工作。

当 WIC 被使能，且处理器进入深度睡眠模式或掉电模式时，系统中电源管理单元可能会关闭 Cortex-M3 处理器的大部分电源。产生的副作用是会停止 SysTick 定时器。当 WIC 收到一个中断时，它要消耗一些时钟周期来唤醒处理器并恢复其状态，然后才能处理中断。这就意味着，在深度睡眠模式下，中断等待时间变长。从掉电模式唤醒需要启动设备的很多其他部分，需要消耗更长的时间。从深度掉电模式唤醒还要加上重新建立片上稳压器电压的时间。

注：如果处理器检测到一个与调试器的连接，它会禁能 WIC。

39.3.5.4 电源管理编程提示

ANSI C 不能直接产生 WFI 和 WFE 指令。CMSIS 为这些指令提供了以下内在函数：

```
void __WFE(void) // Wait for Event  
  
void __WFI(void) // Wait for Interrupt
```

39.4 ARM Cortex-M3 用户指南：外设

39.4.1 关于 Cortex-M3 外设

私有外设总线（PPB）的地址映射如下：

表783. 内核外设寄存器区

地址	内核外设	描述
0xE000E008~0xE000E00F	系统控制模块	表 794 “系统控制模块寄存器汇总”
0xE000E010~0xE000E01F	系统定时器	表 814 “系统定时器寄存器汇总”
0xE000E100~0xE000E4EF	可嵌套向量中断控制器	表 784 “NVIC 寄存器汇总”
0xE000ED00~0xE000ED3F	系统控制模块	表 794 “系统控制模块寄存器汇总”
0xE000ED90~0xE000EDB8	存储器保护单元	表 820 “MPU 寄存器汇总”
0xE000EF00~0xE000EF03	可嵌套向量中断控制器	表 784 “NVIC 寄存器汇总”

在寄存器描述中：

- 寄存器**类型**描述如下：
 - RW**：读写。
 - RO**：只读。
 - WO**：只写。
- 所需特权**给出访问寄存器所需的下列特权等级：
 - 特权**：仅特权软件可访问寄存器。
 - 非特权**：非特权和特权软件均可访问寄存器。

39.4.2 可嵌套向量中断控制器

本节叙述了可嵌套向量中断控制器（NVIC）和它所使用的寄存器。NVIC 支持：

- 多达 112 个中断。实现的中断数量因设备而异。
- 每个中断有可编程的 0 到 31 优先等级。较高等级对应于较低的优先级，因此 0 级为最高的中断优先级。
- 中断信号的电平和脉冲检波。
- 中断优先级的动态重设。
- 优先级值分组为组优先级和子优先级域。
- 中断末尾连锁。
- 一个外部**不可屏蔽中断**（NMI）。

处理器在异常进入时将其状态自动入栈，并在异常退出时将其状态自动出栈，没有指令开销。这样就提供了低延迟的异常处理。NVIC 寄存器的硬件实现为：

表784. NVIC 寄存器汇总

地址	名称	类型	所需特权	复位值	描述
0xE000E100~0xE000E10C	ISER0 - ISER3	RW	特权	0x00000000	表 786
0xE000E180~0xE000E18C	ICER0 - ICER3	RW	特权	0x00000000	表 787
0xE000E200~0xE000E20C	ISPR0 - ISPR3	RW	特权	0x00000000	表 788
0xE000E280~0xE000E28C	ICPR0 - ICPR3	RW	特权	0x00000000	表 789
0xE000E300~0xE000E30C	IABR0 - IABR3	RO	特权	0x00000000	表 790
0xE000E400~0xE000E46C	IPR0 - IPR27	RW	特权	0x00000000	表 791
0xE000EF00	STIR	WO	可配置 ^[1]	0x00000000	表 792

[1] 每个数组元对应一个 NVIC 寄存器。例如，数组元 ICER[1]对应 ICER1 寄存器。

39.4.2.1 Cortex-M3 NVIC 寄存器的 CMSIS 映射

为了提高软件效率，CMSIS 简化了 NVIC 寄存器的表达。在 CMSIS 中：

- 置位使能、清零使能、置位挂起、清零挂起和有效位寄存器映射到 32 位整数的数组，从而使：
 - 数组 ISER[0]到 ISER[3]对应于寄存器 ISER0 - ISER3;
 - 数组 ICER[0]到 ICER[3]对应于寄存器 ICER0 - ICER3;
 - 数组 ISPR[0]到 ISPR[3]对应于寄存器 ISPR0 - ISPR3;
 - 数组 ICPR[0]到 ICPR[3]对应于寄存器 ICPR0 - ICPR3;
 - 数组 IABR[0]到 IABR[3]对应于寄存器 IABR0 - IABR3。
- 8 位的“中断优先级寄存器”域映射到一个 8 位整数数组，从而使数组 IP[0]到 IP[112]对应于寄存器 IPR0 - IPR59，数组表项 IP[n]保存中断 n 的中断优先级。

CMSIS 提供线程安全代码，它实现了对“中断优先级寄存器”的原子访问。更多信息，请参见 [39.4.2.10.1 “NVIC 编程提示”](#) 中 NVIC_SetPriority 函数的描述。[表 785](#) 列出了中断、或 IRQ 序号是如何映射到中断寄存器和对相应的 CMSIS 变量，这些变量对每个中断有一个位。

表785. 中断到中断变量的映射

中断	CMSIS 数组元 ^[1]				
	置位-使能	清零-使能	置位-挂起	清零-挂起	有效位
0-31	ISER[0]	ICER[0]	ISPR[0]	ICPR[0]	IABR[0]
32-63	ISER[1]	ICER[1]	ISPR[1]	ICPR[1]	IABR[1]
64-95	ISER[2]	ICER[2]	ISPR[2]	ICPR[2]	IABR[2]
96-127	ISER[3]	ICER[3]	ISPR[3]	ICPR[3]	IABR[3]

[1] 每个数组元对应一个 NVIC 寄存器。例如，数组元 ICER[1]对应 ICER1 寄存器。

39.4.2.2 中断置位使能寄存器

ISER0-ISER3 寄存器使能中断，并显示哪些中断被使能。请参见：

- [表 784](#) 中寄存器汇总的寄存器属性；
- [表 785](#) 了解每个寄存器都控制哪些中断。

位分配见[表 786](#)。

表786. ISER 位分配

位	名称	功能
[31:0]	SETENA	中断置位-使能位。 写： 0：无影响 1：使能中断。 读： 0：中断被禁能 1：中断被使能。

如果某个挂起中断被使能，NVIC 根据其优先级激活中断。如果一个中断没有被使能，则这个中断信号的发出会将中断状态变为挂起，但无论其优先级如何，NVIC 永远不会激活此中断。

39.4.2.3 中断清零使能寄存器

ICER0-ICER3 寄存器禁止中断，并显示哪些中断被使能。请参见：

- [表 784](#) 中寄存器汇总的寄存器属性；
- [表 785](#) 了解每个寄存器都控制哪些中断。

位分配见[表 787](#)。

表787. ICER 位分配

位	名称	功能
[31:0]	CLRENA	中断清零-使能位。 写： 0：无影响 1：禁能中断。 读： 0：中断被禁能 1：中断被使能。

39.4.2.4 中断置位挂起寄存器

ISPR0-ISPR3 寄存器强制中断进入挂起状态，并显示哪些中断正在挂起。请参见：

- [表 784](#) 中寄存器汇总的寄存器属性；
- [表 785](#) 了解每个寄存器都控制哪些中断。

位分配见[表 788](#)。

表788. ISPR 位分配

位	名称	功能
[31:0]	SETPEND	中断置位-挂起位。 写： 0：无影响 1：改变中断状态为挂起状态 读： 0：中断未挂起 1：中断挂起

注：将 1 写到对应于以下项的 ISPR 位：

- 一个挂起的中断，无效果；
- 一个被禁用的中断，设置该中断的状态为挂起。

39.4.2.5 中断清零挂起寄存器

ICPR0-ICPR3 寄存器移除中断的挂起状态，并显示有哪些中断在挂起。请参见：

- [表 784](#) 中寄存器汇总的寄存器属性；
- [表 785](#) 了解每个寄存器都控制哪些中断。

位分配见[表 789](#)。

表789. ICPR 位分配

位	名称	功能
[31:0]	CLRPEND	中断清零-挂起位。 写： 0：无影响 1：清除挂起状态的一个中断 读： 0：中断未挂起 1：中断挂起

注：将 1 写到一个 ICPR 位不影响相应中断的行为状态。

39.4.2.6 中断有效位寄存器

IABR0-IABR3 寄存器指示哪些中断是有效的。请参见：

- [表 784](#) 中寄存器汇总的寄存器属性；
- [表 785](#) 了解每个寄存器都控制哪些中断。

位分配见[表 790](#)。

表790. IABR 位分配

位	名称	功能
[31:0]	ACTIVE	中断有效标志： 0：中断无效 1：中断有效。

如果相应中断的状态为有效或有效并挂起，则该位读取值为 1。

39.4.2.7 中断优先级寄存器

IPR0-IPR27 寄存器为每个中断提供一个 5 位的优先级域。这些寄存器可按字节访问。其属性请参见[表 784](#)中寄存器汇总。每个寄存器保存 4 个优先级域，它映射到 CMSIS 中断优先级数组 IP[0]到 IP[111]中的 4 个元素，如下所示：

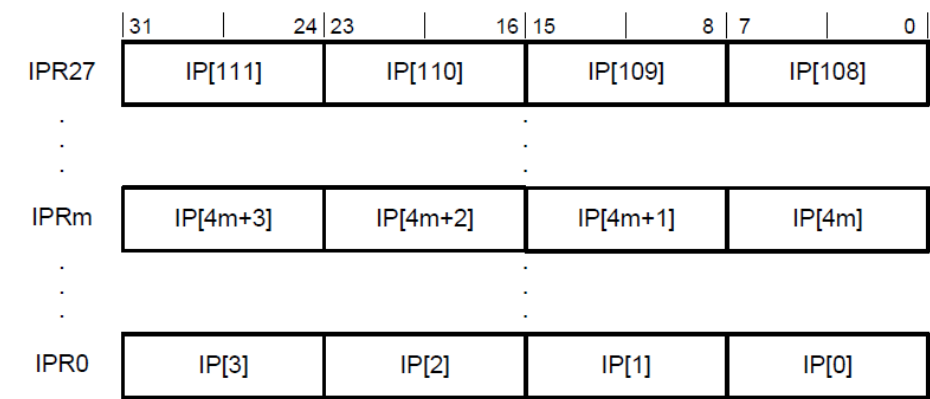


表791. IPR 位分配

位	名称	功能
[31:24]	优先级，字节偏移量 3	每个优先级域承载一个优先级值，0 至 31。这个值越低，相应中断的优先级就越高。处理器仅执行每个域的位 [7:3]，位[2:0]读为零且忽略写入值。
[23:16]	优先级，字节偏移量 2	
[15:8]	优先级，字节偏移量 1	
[7:0]	优先级，字节偏移量 0	

更多关于 IP[0]到 IP[111]中断优先级数组的信息，请参见[表 785](#)，该数组提供了中断优先级的软件视图。

按以下方式查找中断 **N** 的 IPR 号和字节偏移量：

- 相应的 IPR 号 **M**，用 $M = N$ 除以 4 得到；
- 此寄存器中所需“优先级”域的字节偏移量为 $N \bmod 4$ ，其中：
 - 字节偏移量 0 引用寄存器位[7:0]；
 - 字节偏移量 1 引用寄存器位[15:8]；
 - 字节偏移量 2 引用寄存器位[23:16]；
 - 字节偏移量 3 引用寄存器位[31:24]。

39.4.2.8 软件触发中断寄存器

写入 STIR 产生一个软件产生的中断(SGI)。请参见[表 784](#)中寄存器汇总了解 STIR 属性。

当 CCR 中 USERSETMPEND 位设置为 1 时，非特权软件可访问 STIR，见 [39.4.3.8 “配置和控制寄存器”](#)。

注：只有特权软件才能使能对 STIR 的非特权访问。

位分配见[表 792](#)。

表792. STIR 位分配

位	域	功能
[31:9]	-	保留。
[8:0]	INTID	所需 SGI 的中断 ID 范围（0~111）。例如，b0000000011 指定的是中断 IRQ3。

39.4.2.9 电平触发和脉冲中断

处理器同时支持电平触发中断和脉冲中断。脉冲中断也称为边沿触发中断。

电平触发中断会保持有效状态，直到外设撤销中断信号。通常这是因为 ISR 访问外设，导致它清除中断请求。脉冲中断是一个在处理器时钟上升沿同步采样的中断信号。为了确保 NVIC 检测到中断，外设必须使中断信号保持有效至少一个时钟周期，在此期间 NVIC 可检测到脉冲并锁存中断。

当处理器进入 ISR 时，它自动从中断移除挂起状态，见 [39.4.2.9.1](#) 节。对于电平触发中断，如果在处理器从 ISR 返回前信号没有撤销，中断重新变为挂起状态，则处理器必须再次执行它的 ISR。这就是说，外设可以保持中断一直有效，直到不再需要处理为止。

39.4.2.9.1 中断的软/硬件控制

Cortex-M3 锁存全部中断。一个外设中断会因以下原因之一而变为挂起状态：

- NVIC 检测到中断信号为高电平且中断为无效状态；
- NVIC 检测到一个中断信号的上升沿；
- 软件写入相应的中断置位挂起寄存器位，见[表 788](#)，或写入 STIR，造成一个 SGI 挂

起，见[表 792](#)。

一个挂起中断会保持在挂起状态，直到发生以下情况之一：

- 处理器进入该中断的 **ISR**。使中断的状态从挂起变为有效。然后：
 - 对于电平触发中断，当处理器从 **ISR** 返回时，**NVIC** 采样中断信号。如果信号被使能，则中断变为挂起状态，这可能使处理器立即重新进入 **ISR**。否则，中断状态变为无效。
 - 对于脉冲中断，**NVIC** 持续监测中断信号，如果有信号脉冲，则中断状态变为挂起和有效。在此情况下，当处理器从 **ISR** 返回时，中断状态变为挂起，这可能使处理器立即重新进入 **ISR**。如果处理器在 **ISR** 中时没有中断信号脉冲，则当处理器从 **ISR** 返回时，中断状态变为无效。
- 软件写入相应的中断清零挂起寄存器位。
 - 对于电平触发中断，如果中断信号仍然有效，则中断状态不变。否则中断状态变为无效。
 - 对于脉冲中断，中断状态变为：
 - 无效，如果原状态为挂起；
 - 有效，如果原状态为有效和挂起。

39.4.2.10 NVIC 设计提示和建议

确保软件使用了正确对齐的寄存器访问。处理器不支持对 **NVIC** 寄存器的非对齐访问。请参见各个寄存器的描述了解它所支持的访问大小。

即使某个中断被禁用，它也可以进入挂起状态。

在对 **VTOR** 编程以重新定位向量表以前，要确保对故障处理程序、**NMI** 和所有使能的异常（如中断）都建立了新向量表的向量表项。更多信息，请参见[表 798](#)。

39.4.2.10.1 NVIC 编程提示

软件使用 CPSIE I 和 CPSID I 指令来使能和禁能中断。CMSIS 提供为这些指令提供了以下内在函数：

```
void __disable_irq(void) // Disable Interrupts

void __enable_irq(void) // Enable Interrupts
```

另外，CMSIS 提供了一些 NVIC 控制函数，包括：

表793. 用于 NVIC 控制的 CMSIS 函数

CMSIS 中断控制函数	描述
void NVIC_SetPriorityGrouping(uint32_t priority_grouping)	设置优先级分组
void NVIC_EnableIRQ(IRQn_t IRQn)	使能 IRQn
void NVIC_DisableIRQ(IRQn_t IRQn)	禁能 IRQn
uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn)	如果 IRQn 挂起，返回结果为真（IRQ-Number 即 IRQ 序号）
void NVIC_SetPendingIRQ (IRQn_t IRQn)	设置 IRQn 挂起
void NVIC_ClearPendingIRQ (IRQn_t IRQn)	清除 IRQn 挂起状态
uint32_t NVIC_GetActive (IRQn_t IRQn)	返回有效中断的 IRQ 序号
void NVIC_SetPriority (IRQn_t IRQn, uint32_t priority)	设置 IRQn 优先级
uint32_t NVIC_GetPriority (IRQn_t IRQn)	读取 IRQn 优先级
void NVIC_SystemReset (void)	复位系统

有关这些函数的更多信息，请参见 CMSIS 文档。

39.4.3 系统控制模块

系统控制模块（SCB）提供系统执行信息和系统控制。这包括对系统异常的配置、控制和报告。系统控制模块寄存器有：

表794. 系统控制模块寄存器汇总

地址	名称	类型	所需特权	复位值	描述
0xE000E008	ACTLR	RW	特权	0x00000000	表 795
0xE000ED00	CPUID	RO	特权	0x412FC230	表 796
0xE000ED04	ICSR	RW ^[1]	特权	0x00000000	表 797
0xE000ED08	VTOR	RW	特权	0x00000000	表 798
0xE000ED0C	AIRCR	RW ^[1]	特权	0xFA050000	表 799
0xE000ED10	SCR	RW	特权	0x00000000	表 801
0xE000ED14	CCR	RW	特权	0x00000200	表 802
0xE000ED18	SHPR1	RW	特权	0x00000000	表 804
0xE000ED1C	SHPR2	RW	特权	0x00000000	表 805
0xE000ED20	SHPR3	RW	特权	0x00000000	表 806
0xE000ED24	SHCRS	RW	特权	0x00000000	表 807
0xE000ED28	CFSR	RW	特权	0x00000000	39.4.3.11 节
0xE000ED28	MMSR ^[2]	RW	特权	0x00	表 808
0xE000ED29	BFSR ^[2]	RW	特权	0x00	表 809
0xE000ED2A	UFSR ^[2]	RW	特权	0x0000	表 810
0xE000ED2C	HFSR	RW	特权	0x00000000	表 811
0xE000ED34	MMFAR	RW	特权	未定义	表 812
0xE000ED38	BFAR	RW	特权	未定义	表 813

[1] 更多信息，参见寄存器描述。
[2] CFSR 的一个子寄存器。

39.4.3.1 Cortex-M3 SCB 寄存器的 CMSIS 映射

为了提高软件效率，CMSIS 简化了 SCB 寄存器的表述。在 CMSIS 中，字节数组 SHP[0] 到 SHP[12]对应于寄存器 SHPR1 至 SHPR3。

39.4.3.2 辅助控制寄存器（ACTLR）

ACTLR 提供了以下处理器功能的禁用位：

- IT 折叠（folding）；
- 用于访问默认存储器映射的写缓冲；
- 多周期指令的中断。

请参见[表 794](#) 中寄存器汇总了解 ACTLR 属性。位分配见[表 795](#)。

表795. ACTLR 位分配

位	名称	功能
[31:3]	-	保留
[2]	DISFOLD	该位为 1 时，禁能 IT 折叠。更多信息请参见 39.4.3.2.1 节。
[1]	DISDEFWBUF	该位为 1 时，在默认存储器映射访问期间禁能写缓冲使用。这使得所有的总线故障都是精确总线故障，但却降低了性能，因为存储器的所有存储操作都必须在处理器能够执行下一指令前完成。 注： 该位仅影响在 Cortex-M3 处理器中执行的写缓冲。
[0]	DISMCYCINT	该位为 1 时，禁能多个加载和多个存储指令的中断。这增加了处理器的中断延迟，因为所有的 LDM 或 STM 都必须在处理器能够堆栈当前状态并进入中断处理程序前完成。

39.4.3.2.1 关于 IT 折叠

在某些情况下，处理器可以在执行 IT 指令时，开始执行某个 IT 块中的第一条指令。此行为称为 IT 折叠，可提高性能，然而，IT 折叠可能导致循环中的抖动（jitter）。如果某个任务必须避免抖动，则要在执行任务前将 DISFOLD 位设为 1，禁用 IT 折叠。

39.4.3.3 CPUID 基址寄存器

CPUID 寄存器包括处理器部件号、版本和执行信息。其属性请参见[表 794](#) 中的寄存器汇总。位分配见[表 796](#)。

表796. CPUID 寄存器的位分配

位	名称	功能
[31:24]	Implementer	实现者代码：0x41 = ARM
[23:20]	Variant	变量号， rn timer 产品修订标识符中的 r 值：0x2 = r2p0
[19:16]	Constant	读为 0xF
[15:4]	PartNo	处理器的部件序号：0xC23 = Cortex-M3
[3:0]	Revision	修订号， rn timer 产品修订标识符中的 p 值：0x0 = r2p0

39.4.3.4 中断控制和状态寄存器

ICSR:

- 提供：
 - 用于不可屏蔽中断（NMI）异常的置位挂起位；
 - 用于 PendSV 和 SysTick 异常的置位挂起位和清零挂起位。
- 指示：
 - 正在被处理的异常的异常号；
 - 有无被抢占的有效异常；
 - 具有最高优先级的挂起异常的异常号；
 - 是否有挂起中断。

ICSR 属性请参见[表 794](#) 中寄存器汇总和[表 797](#) 中类型描述。位分配见[表 797](#)。

表797. ICSR 位分配

位	名称	类型	功能
[31]	NMIPENDSET	RW	NMI 置位-挂起位。 写： 0: 无影响 1: 改变 NMI 异常状态为挂起。 读： 0: NMI 异常未挂起 1: NMI 异常挂起。 因为 NMI 是最高优先级的异常，通常情况下处理器在该位被写入 1 后马上进入 NMI 异常处理程序，进入处理程序将该位清零。当处理器执行某个 NMI 异常处理程序时，只有当 NMI 信号再次有效时，该处理程序对该位的读取结果才返回为 1。
[30:29]	-	-	保留。

位	名称	类型	功能
[28]	PENDSVSET	RW	PendSV 置位-挂起位。 写： 0：无影响 1：改变 PendSV 异常状态为挂起。 读： 0：PendSV 异常未挂起 1：PendSV 异常挂起。 向该位写入 1 是将 PendSV 异常状态设为挂起的唯一途径。
[27]	PENDSVCLR	WO	PendSV 清零-挂起位。 写： 0：无影响 1：移除 PendSV 异常中的挂起状态
[26]	PENDSTSET	RW	SysTick 异常置位-挂起位。 写： 0：无影响 1：改变 SysTick 异常状态为挂起。 读： 0：SysTick 异常未挂起 1：SysTick 异常挂起。
[25]	PENDSTCLR	WO	SysTick 异常清零-挂起位。 写： 0：无影响 1：移除 SysTick 异常中的挂起状态。 该位为只写位。该位在寄存器上读取值未知。
[24]	-	-	保留。
[23]	Reserved for Debug use	RO	该位保留用于调试，当处理器不在调试模式时该位读取为零。
[22]	ISR_PENDING	RO	中断挂起标志，不包括 NMI 和故障： 0：中断未挂起 1：中断挂起。
[21:18]	-	-	保留。
[17:12]	VECT_PENDING	RO	该位表示最高优先级挂起使能异常的异常序号： 0：无挂起异常 非零值：最高优先级挂起使能异常的异常序号。 该域值包括 BASEPRI 和 FAULTMASK 寄存器的效果，但不包括 PRIMASK 寄存器的任何效果。
[11]	RETTOBASE	RO	该位表示有无被抢占的有效异常： 0：有待执行的被抢占的有效异常 1：无有效异常，或当前执行的异常是唯一有效异常。
[10:9]	-	-	保留。
[8:0]	VECTACTIVE ^[1]	RO	包含有效异常序号： 0：线程模式 非零值：当前有效异常的异常序号 ^[1] 。 注：该值减 16 可以得到用来索引到中断清零使能、置位使能、清零挂起、置位挂起或优先级寄存器所需的 IRQ 序号，见 表 769 “IPSR 位分配” 。

[1] 这与 IPSR 位[8:0]的值相同，见[表 769 “IPSR 位分配”](#)

当写入 ICSR 时，如有以下情况，则后果不可预知：

- 写入 1 到 PENDSVSET 位并写入 1 到 PENDSVCLR 位；
- 写入 1 到 PENDSTSET 位并写入 1 到 PENDSTCLR 位。

39.4.3.5 向量表偏移量寄存器

VTOR 表示从存储器地址 0x00000000 开始的向量表基址偏移量。其属性请参见表 794 中寄存器汇总。

位分配见表 798。

表798. VTOR 位分配

位	名称	功能
[31:30]	-	保留。
[29:8]	TBLOFF	向量表的基址偏移域。它包含向量表基址与存储器映射底部的偏移量的位 [29:8]。 注：位[29]决定向量表是否在编码或 SRAM 存储区： 位[29]有时被称为位 TBLBASE。 <ul style="list-style-type: none">• 0：编码• 1：SRAM。
[7:0]	-	保留。

当置位 TBLOFF 时，必须使偏移对准向量表中异常进入的序号。建议的对齐方式为 256 个字，可用于 128 个中断。

注：表对齐要求的含义是表偏移的位[7:0]始终为零。

39.4.3.6 应用中断和复位控制寄存器

AIRCR 提供了异常模型的优先级分组控制、数据访问的字节序状态，以及系统的复位控制。其属性请参见[表 794](#) 中寄存器汇总和[表 799](#)。

如要对此寄存器执行写操作，必须将 0x5VA 写入 VECTKEY 域，否则处理器会忽略写入值。

位分配见[表 799](#)。

表799. AIRCR 位分配

位	名称	类型	功能
[31:16]	写：VECTKEYSTAT 读：VECTKEY	RW	寄存器密钥（register key）： 读为 0x05FA 执行写操作时要求向 VECTKEY 写入 0x5FA，否则写入值被忽略。
[15]	ENDIANESS	RO	数据字节序位： 0：小端。
[14:11]	-	-	保留
[10:8]	PRIGROUP	R/W	中断优先级分组域。该域决定从子优先级中拆分组优先级，见 39.4.3.6.1 节。
[7:3]	-	-	保留。
[2]	SYSRESETREQ	WO	系统复位请求： 0：无系统复位请求 1：让信号在外部系统有效，表示请求一个复位。 该位用于强制对调试设备之外的所有主要设备进行一次大的系统复位。 注：LPC178x/177x 设备支持 SYSRESETREQ。 该位读为 0。
[1]	VECTCLRACTIVE	WO	该位保留供调试使用。该位读为 0。当向寄存器执行写操作时，必须向该位写入 0，否则该行为不可预知。
[0]	VECTRESET	WO	该位保留供调试使用。该位读为 0。当向寄存器执行写操作时，必须向该位写入 0，否则该行为不可预知。

39.4.3.6.1 二进制点

PRIGROUP 域指示了二进制点的位置，该点将“中断优先级寄存器”中 PRI_n 域分割为独立的组优先级和子优先级域。[表 800](#) 列出了 PRIGROUP 值如何控制此分割。

表800. 优先级分组

中断优先级级别值，PRI_N[7:0]			数目		
	二进制点 [1]	组优先级位	子优先级位	组优先级	子优先级
b010	bxxxxx.000	[7:3]	无	32	1
b011	bxxx.y000	[7:4]	[3]	16	2
b100	bxxx.yy000	[7:5]	[4:3]	8	4
b101	bxx.yyy000	[7:6]	[5:3]	4	8
b110	bx.yyyy000	[7]	[6:3]	2	16
b111	b.yyyyy000	无	[7:3]	1	32

[1] PRI_n[7:0]域显示二进制点。X 指示一个组优先级域位，y 指示一个子优先级域位。位[2:0]不在 LPC178x/177x 设备中使用。

备注：确定一个异常是否抢占时，只使用组优先级域，见 [39.3.3.6 “中断优先级分组”](#)。

39.4.3.7 系统控制寄存器

SCR 控制着进入和离开低功耗状态的特性。其属性请参见[表 794](#) 中寄存器汇总。位分配见[表 801](#)。

表801. SCR 位分配

位	名称	功能
[31:5]	-	保留。
[4]	SEVONPEND	在挂起位上发送事件： 0：仅使能的中断或事件可以唤醒处理器，禁能的中断不能唤醒处理器。 1：使能的事件和所有中断，包括禁能的中断，都可以唤醒处理器。 当一个事件或中断进入挂起状态，事件信号将处理器从 WFE 中唤醒。 如果处理器没有在等待一个事件，该事件被记录并影响下一 WFE。 该处理器也可以在执行 SEV 指令或一个外部事件时被唤醒。
[3]	-	保留。
[2]	SLEEPDEEP	控制处理器是否使用睡眠或深度睡眠作为其低功耗模式： 0：睡眠 1：深度睡眠。
[1]	SLEEPONEXIT	表示当从处理模式返回到线程模式时开始“退出时睡眠”： 0：返回到线程模式时不进入到睡眠模式 1：从 ISR 返回时进入睡眠或深度睡眠模式。 将该位设为 1 会使能一个中断驱动应用程序以避免返回到空的主应用程序中。
[0]	-	保留。

39.4.3.8 配置和控制寄存器

CCR 控制着线程模式的进入，并使能：

- 通过 FAULTMASK 升级的 NMI、硬故障和故障处理程序，以忽略总线故障；
- 被零除和非对齐访问的捕获；
- 非特权软件对 STIR 的访问，见[表 792](#)。

CCR 属性请参见[表 794](#) 中寄存器汇总。

位分配见[表 802](#)。

表802. CCR 位分配

位	名称	功能
[31:10]	-	保留。
[9]	STKALIGN	表示进入异常时的堆栈对齐。 0: 4 字节对齐 1: 8 字节对齐 进入异常时，处理器使用压入堆栈的 PSR 位[9]来指示堆栈对齐。从异常返回时，这个堆栈位被用来恢复正确的堆栈对齐。
[8]	BFHFNMI	使能时，使得以优先级位-1 或-2 运行的处理程序忽略加载和存储指令引起的数据总线故障。它用于硬故障、NMI 和 FAULTMASK 升级处理程序中： 0: 加载和存储指令引起的数据总线故障会引起锁定。 1: 以优先级-1 或-2 运行的处理程序忽略加载和存储指令引起的数据总线故障。 仅在处理程序和其数据处于绝对安全的存储器时将该位设为 1。一般将该位用于探测系统设备和桥接器以检测并纠正控制路径问题。
[7:5]	-	保留。
[4]	DIV_0_TRP	当处理器进行除 0 操作（SDIV 或 UDIV 指令）时，会导致故障或停止。 0: 不捕获除以零故障 1: 捕获除以零故障。 当该位设为 0 时，除以零返回的商数为 0。
[3]	UNALIGN_TRP	使能非对齐访问捕获： 0: 不捕获非对齐半字和字访问 1: 捕获非对齐半字和字访问。 如果该位设为 1，非对齐访问产生一个使用故障。无论 UNALIGN_TRP 是否设为 1，非对齐的 LDM、STM、LDRD 和 STRD 指令总是出错。
[2]	-	保留。
[1]	USERSETM PEND	使能对 STIR 的无特权软件访问，参见 表 792 。 0: 禁能 1: 使能
[0]	NONEBASE THRDENA	指示处理器如何进入线程模式： 0: 处理器仅在没有有效异常时才能够进入线程模式。 1: 处理器可以从 EXC_RETURN 值控制下的任何级别进入线程模式，参见 39.3.3.7.2 “异常返回” 。

39.4.3.9 系统处理程序优先级寄存器

SHPR1-SHPR3 寄存器用于设置有可配置优先级异常处理程序的优先级 0 到 31。

SHPR1-SHPR3 可按字节访问。这些寄存器的属性请参见[表 794](#) 中寄存器汇总。

系统故障处理程序，以及各处理程序的优先级域和寄存器如下：

表803. 系统故障处理程序优先级域

处理程序	域	寄存器描述
存储器管理故障	PRI_4	表 804
总线故障	PRI_5	
使用故障	PRI_6	
SVCall	PRI_11	表 805
PendSV	PRI_14	表 806
SysTick	PRI_15	

每个 PRI_N 域为 8 位宽，但是处理器仅实现每个域的位[7:3]，位[2:0]读取值为零并忽略写入值。

39.4.3.9.1 系统处理程序优先级寄存器 1

位分配见[表 804](#)。

表804. SHPR1 寄存器的位分配

位	名称	功能
[31:24]	PRI_7	保留
[23:16]	PRI_6	系统处理程序 6 的优先级，使用故障
[15:8]	PRI_5	系统处理程序 5 的优先级，总线故障
[7:0]	PRI_4	系统处理程序 4 的优先级，存储器管理故障

39.4.3.9.2 系统处理程序优先级寄存器 2

位分配见[表 805](#)。

表805. SHPR2 寄存器的位分配

位	名称	功能
[31:24]	PRI_11	系统处理程序 11 的优先级，SVCall
[23:0]	-	保留

39.4.3.9.3 系统处理程序优先级寄存器 3

位分配见[表 806](#)。

表806. SHPR3 寄存器的位分配

位	名称	功能
[31:24]	PRI_15	系统处理程序 15 的优先级，SysTick 异常
[23:16]	PRI_14	系统处理程序 14 的优先级，PendSV
[15:0]	-	保留

39.4.3.10 系统处理程序控制和状态寄存器

SHCSR 使能系统处理程序，并指示：

- 总线故障、存储器管理故障和 SVC 异常的挂起状态；
- 系统处理程序的有效状态。

SHCSR 属性请参见[表 794](#)中寄存器汇总。位分配见[表 807](#)。

表807. SHCSR 位分配

位	名称	功能
[31:19]	-	保留
[18]	USGFAULTENA	使用故障使能位，设为 1 时使能 ^[1]
[17]	BUSFAULTENA	总线故障使能位，设为 1 时使能 ^[1]
[16]	MEMFAULTENA	存储器管理故障使能位，设为 1 时使能 ^[1]
[15]	SVCALLPENDED	SVC 调用挂起位，如果异常挂起，该位读为 1 ^[2]
[14]	BUSFAULTPENDED	总线故障异常挂起位，如果异常挂起，该位读为 1 ^[2]
[13]	MEMFAULTPENDED	存储器管理故障异常挂起位，如果异常挂起，该位读为 1 ^[2]
[12]	USGFAULTPENDED	使用故障异常挂起位，如果异常挂起，该位读为 1 ^[2]
[11]	SYSTICKACT	SysTick 异常有效位，如果异常有效，该位读为 1 ^[3]
[10]	PENDSVACT	PendSV 异常有效位，如果异常有效，该位读为 1
[9]	-	保留
[8]	MONITORACT	调试监控有效位，如果调试监控有效，该位读为 1
[7]	SVCALLACT	SVC 调用有效位，如果 SVC 调用有效，该位读为 1
[6:4]	-	保留
[3]	USGFAULTACT	使用故障异常有效位，如果异常有效，该位读为 1
[2]	-	保留
[1]	BUSFAULTACT	总线故障异常有效位，如果异常有效，该位读为 1
[0]	MEMFAULTACT	存储器管理故障异常有效位，如果异常有效，该位读为 1

- [1] 使能位，设为 1 使能异常，或设为 0 禁能异常。
- [2] 挂起位，如果异常挂起，该位读为 1，或如果异常未挂起，该位读为 0。可以向这些位写入值以改变异常的挂起状态。
- [3] 有效位，如果异常有效，该位读为 1，或如果异常无效，该位读为 0。可以向这些位写入值以改变异常的有效状态，但需关注本章节中的“注意”部分。

如果禁用某个系统处理程序，并发生了对应的故障，则处理器将该故障作为硬故障。

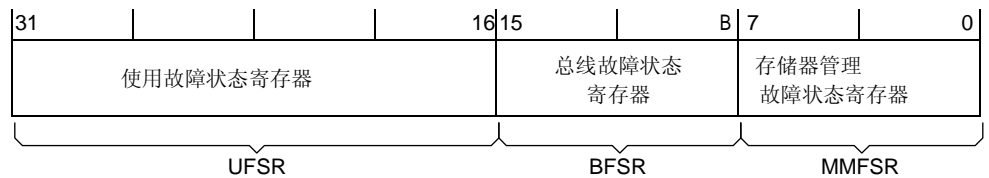
可写入此寄存器来改变系统异常的挂起或有效状态。OS 内核可以写入有效位进行上下文切换，从而改变当前异常的类型。

注意

- 如果软件改变了此寄存器的某个有效位的值，而未正确调整入栈的内容，则可能会使处理器产生一个故障异常。要确保写入此寄存器的软件继续并在随后恢复当前的有效状态。
- 使能了系统处理程序后，如果必须改变此寄存器中某个位的值，则必须采用读取-修改-写入步骤，以确保只改变需要的位。

39.4.3.11 可配置故障状态寄存器

CFSR 指示存储器管理故障、总线故障或使用故障的原因。CFSR 属性请参见表 794 中寄存器汇总。位分配为：



以下各小节描述了构成 CFSR 的子寄存器：

- [表 808 “MMFSR 位分配”](#)；
- [表 813 “BFAR 位分配”](#)；
- [表 810 “UFSR 位分配”](#)。

CFSR 可按字节访问。可如下访问 CFSR 或其子寄存器：

- 通过到 0xE000ED28 的字访问来访问整个 CFSR；
- 通过到 0xE000ED28 的字节访问来访问 MMFSR；
- 通过到 0xE000ED28 的半字访问来访问 MMFSR 和 BFSR；
- 通过到 0xE000ED29 的字节访问来访问 BFSR；
- 通过到 0xE000ED2A 的半字访问来访问 UFSR。

39.4.3.11.1 存储器管理故障状态寄存器

MMFSR 中标志指示存储器访问故障的原因。位分配见表 808。

表808. MMFSR 位分配

位	名称	功能
[7]	MMARVALID	存储器管理故障地址寄存器（MMAR）有效标志： 0：MMAR 中的值不是一个有效故障地址 1：MMAR 中保留一个有效故障地址。 如果发生了一个存储器管理故障，并由于优先级的原因升级成一个硬故障，那么硬故障处理程序必须将该位设为 0。 这样可以避免在返回到 MMAR 值已被覆写的压入堆栈的有效存储器管理故障处理程序时出现问题。
[6:5]	-	保留。
[4]	MSTKERR	进入异常时的入栈操作引起的存储器管理故障： 0：无入栈故障 1：进入异常时的入栈操作引起了一个或一个以上的访问违犯。 当该位设为 1 时，依然要对 SP 进行调节，并且堆栈的上下文区域的值可能不正确。处理器没有向 MMAR 中写入故障地址。

位	名称	功能
[3]	MUNSTKERR	异常返回时的出栈操作引起的存储器管理故障： 0：无出栈故障 1：异常返回时的出栈操作已引起一个或一个以上的访问违犯 该故障与处理程序相连，这意味着当该位为 1 时，原始的返回堆栈仍然存在。 处理器不能对返回失败的 SP 进行调节，并且不会执行新的存储操作。 处理器没有向 MMAR 中写入故障地址。
[2]	-	保留
[1]	DACCVIOL	数据访问违犯标志： 0：无数据访问违犯故障 1：处理器试图在不允许执行操作的位置上进行加载和存储。 当该位为 1 时，异常返回的压入堆栈的 PC 值指向出错指令。处理器已在 MMAR 中加载了目标访问的地址。
[0]	IACCVIOL	指令访问违犯标志： 0：无指令访问违犯错误 1：处理器试图从不允许执行操作的位置上进行指令获取。 即使 MPU 被禁能，这一故障也会在每次访问 XN 区时发生。 当该位为 1 时，异常返回的压入堆栈的 PC 值指向出错指令。处理器没有向 MMAR 中写入故障地址。

39.4.3.11.2 总线故障状态寄存器

BFSR 中标志指示总线访问故障原因。位分配见[表 809](#)。

表809. BFSR 位分配

位	名称	功能
[7]	BFARVALID	总线故障地址寄存器（BFAR）有效标志： 0：BFAR 中的值不是有效故障地址 1：BFAR 中保留一个有效故障地址。 在地址已知的总线故障发生后处理器将该位设为 1。该位可以被其他故障清零，例如之后发生的存储器管理故障。 如果发生总线故障，并由于优先级原因升级为一个硬错误，那么硬故障处理程序必须将该位设为 0。当返回至一个 BFAR 值已被覆写的压入堆栈的有效总线故障处理程序时，这样做可以防止问题的出现。
[6:5]	-	保留。

位	名称	功能
[4]	STKERR	<p>进入异常时的入栈操作引起的总线故障：</p> <p>0：无入栈故障</p> <p>1：进入异常时的入栈操作已引起一个或一个以上的总线故障。</p> <p>当处理器将该位设为 1 时，依然要对 SP 进行调节，并且堆栈的上下文区域的值可能不正确。处理器没有向 BFAR 中写入故障地址。</p>
[3]	UNSTKERR	<p>异常返回时的出栈操作引起的总线故障：</p> <p>0：无出栈故障</p> <p>1：异常返回时的出栈操作已引起一个或一个以上的总线故障。</p> <p>该故障与处理程序相连， 这意味着当处理器将该位设为 1 时，原始的返回堆栈仍然存在。处理器不能对返回失败的 SP 进行调节，并且不会执行新的存储操作，也未向 BFAR 中写入故障地址。</p>
[2]	IMPRECISERR	<p>非精确数据总线错误：</p> <p>0：无非精确数据总线错误</p> <p>1：已发生一个数据总线错误，但是堆栈帧中的返回地址与引起错误的指令无关。</p> <p>当处理器将该位设为 1 时，不向 BFAR 中写入故障地址。</p> <p>这是一个异步故障。因此，如果在当前进程的优先级高于总线故障优先级时检测到该故障，总线故障被挂起并仅在处理器从所有更高优先级进程中返回时开始变为有效。如果在处理器进入非精确总线故障的处理程序前发生一个精确故障，那么处理程序同时对 IMPRECISERR 和其中一个精确故障状态位进行检测，判断它们是否置位为 1。</p>
[1]	PRECISERR	<p>精确数据总线错误：</p> <p>0：非精确数据总线错误</p> <p>1：已发生一个数据总线错误，且异常返回的压入堆栈的 PC 值指向引起故障的指令。</p> <p>当处理器将该位设为 1 时，向 BFAR 中写入故障地址。</p>
[0]	IBUSERR	<p>指令总线错误：</p> <p>0：无指令总线错误</p> <p>1：指令总线错误。</p> <p>处理器检测到预取指令时的指令总线错误，但仅在其试图签发故障指令时才将 IBUSERR 标志设为 1。</p> <p>当处理器将该位设为 1 时，不向 BFAR 中写入故障地址。</p>

39.4.3.11.3 使用故障状态寄存器

UFSR 指示使用故障的原因。位分配见[表 810](#)。

表810. UFSR 位分配

位	名称	功能
[15:10]	-	保留。
[9]	DIVBYZERO	除以零使用故障： 0：无除以零故障或除以零捕获未使能 1：处理器已执行 SDIV 或 UDIV 指令（除以零）。 当处理器将该位设为 1 时，异常返回的压入堆栈的 PC 值指向执行除以零的指令。 通过将 CCR 中的 DIV_0_TRP 位设为 1 使能除以零捕获，参见 表 802 。
[8]	UNALIGNED	非对齐访问使用故障： 0：无非对齐访问故障，或非对齐访问捕获未使能 1：处理器已进行了一次非对齐的存储器访问。 通过将 CCR 中的 UNALIGN_TRP 位设为 1 来使能非对齐访问捕获，参见 表 802 。 非对齐的 LDM、STM、LDRD 和 STRD 指令总是出错，与 UNALIGN_TRP 的设置无关。
[7:4]	-	保留。
[3]	NOCP	无协处理器使用故障。处理器不支持协处理器指令： 0：试图访问一个协处理器未引起使用故障 1：处理器已试图访问一个协处理器。
[2]	INVPC	EXC_RETURN 的无效 PC 加载引起的无效 PC 加载使用故障： 0：没有发生无效 PC 加载使用故障 1：处理器已试图将 EXC_RETURN 非法载入 PC，作为一个无效的上下文或一个无效的 EXC_RETURN 值。 当该位被设为 1 时，异常返回的压入堆栈的 PC 值指向尝试执行非法 PC 加载的指令。
[1]	INVSTATE	无效状态使用故障： 0：未发生无效状态使用故障 1：处理器已试图执行一个非法使用 EPSR 的指令。 当该位设为 1 时，异常返回的压入堆栈的 PC 值指向一个尝试非法使用 EPSR 的指令。 如果一个未定义的指令使用了 EPSR，则该位不被置位为 1。
[0]	UNDEFINSTR	未定义的指令使用故障： 0：无未定义的指令使用故障 1：处理器已试图执行一个未定义的指令。当该位设为 1 时，异常返回的压入堆栈的 PC 值指向未定义的指令。 未定义的指令是一条不能被处理器译码的指令。

注：UFSR 位是粘性（sticky）位。这就是说，当一个或多个故障发生时，相关的位被设为 1。被设为 1 的位只能通过向该位写入 1 或复位才能清零。

39.4.3.12 硬故障状态寄存器

HFSR 提供关于激活硬故障处理程序的事件的信息。其属性请参见表 794 中寄存器汇总。

此寄存器为可读并通过写入清零。也就是说，此寄存器中的位可正常读取，但向任何位写入 1 会将该位清零。位分配见表 811。

表811. HFSR 位分配

位	名称	功能
[31]	DEBUGEVT	保留供调试使用。对寄存器执行写操作时，必须向该位写入 0；否则，该行为不可预知。
[30]	FORCED	指示一个强制的硬故障，该硬故障由一个优先级可配置且无法处理的故障升级而来，这一故障无法处理是由于优先级或可配置故障被禁能的原因： 0：无强制的硬故障 1：强制的硬故障。 当该位设为 1 时，硬故障处理程序必须读其他故障状态寄存器以找出故障原因。
[29:2]	-	保留。
[1]	VECTTBL	指示一个在异常处理过程中读向量表而引起的总线故障： 0：读向量表未引起总线故障 1：读向量表引起了总线故障。 这一错误通常情况下都由硬故障处理程序来处理。 当该位设为 1 时，异常返回的压入堆栈的 PC 值指向被异常抢占的指令。
[0]	-	保留。

注：HFSR 位是粘性 (sticky) 位。这就是说，当一个或多个故障发生时，相关的位被设为 1。被设为 1 的位，只能通过向该位写入 1 或复位才能清零。

39.4.3.13 存储器管理故障地址寄存器

MMFAR 包括产生存储器管理故障的位置的地址。其属性请参见表 794 中寄存器汇总。位分配为：

表812. MMFAR 位分配

位	名称	功能
[31:0]	ADDRESS	当 MMFSR 的位 MMARVALID 设为 1 时，该域中保存的是产生存储器管理故障的单元地址。

当出现非对齐访问故障时，地址就是实际出错的地址。由于单个读或写指令可能被分割为多个对齐的访问，故障地址可能是请求访问尺寸范围内的任何地址。

MMFSR 中的标志指示故障原因以及 MMFAR 中的值是否有效。请参见表 808。

39.4.3.14 总线故障地址寄存器

BFAR 包括产生总线故障的位置的地址。其属性请参见[表 794](#) 中寄存器汇总。位分配为：

表813. BFAR 位分配

位	名称	功能
[31:0]	ADDRESS	当 BFSR 的位 BFARVALID 设为 1 时，该域中保存的是产生总线故障的单元地址。

当发生非对齐访问故障时，BFAR 中地址为指令所请求的地址，即使该地址不是故障的地址。

BFSR 中的标志用于指示故障原因以及 BFAR 中的值是否有效。请参见[表 809](#)。

39.4.3.15 系统控制模块设计提示和建议

要确保软件使用正确尺寸的对齐访问来访问系统控制模块寄存器：

- 除了 CFSR 和 SHPR1-SHPR3，必须使用对齐的字访问；
- 对于 CFSR 和 SHPR1-SHPR3，可使用字节或对齐的半字或字访问。

处理器不支持对系统控制模块寄存器的非对齐访问。

在故障处理程序中，为了确定真实的出错地址：

1. 读取并保存 MMFAR 或 BFAR 值。
2. 读取 MMFSR 中 MMARVALID 位，或 BFSR 中 BFARVALID 位。仅在此位为 1 时，MMFAR 或 BFAR 地址才有效。

软件必须按照此顺序，因为可能会有另一个更高优先级的异常改变 MMFAR 或 BFAR 的值。例如，如果更高优先级的处理程序抢占了当前故障处理程序，则其他故障可能会改变 MMFAR 或 BFAR 值。

39.4.4 系统定时器 SysTick

处理器有一个 24 位系统定时器 SysTick，它从重载值递减计数至零，在下一时钟边沿重载（绕回）加载寄存器中的值，然后在后续的时钟中递减计数。

注：请参阅 LPC178x/177x 用户手册中的单独章节 [25.1](#) 节，了解具体设备的系统定时器信息。

注：当处理器停止调试时，计数器不会递减计数。

系统定时器寄存器有：

表814. 系统定时器寄存器汇总

地址	名称	类型	所需特权	复位值	描述
0xE000E010	CTRL	RW	特权	0x00000004	表 815
0xE000E014	LOAD	RW	特权	0x00000000	表 816
0xE000E018	VAL	RW	特权	0x00000000	表 817
0xE000E01C	CALIB	RO	特权	0x000F423F ^[1]	表 818

[1] SysTick 校准值。该值特用于 LPC178x/177x 设备。

39.4.4.1 SysTick 控制和状态寄存器

SysTick CTRL 寄存器用于使能 SysTick 特性。其属性请参见[表 814](#)中寄存器汇总。位分配见[表 815](#)。

表815. SysTick 控制寄存器的位分配

位	名称	功能
[31:17]	-	保留。
[16]	COUNTFLAG	从上次读取定时器开始，如果定时器计数到 0，则返回 1。
[15:3]	-	保留。
[2]	CLKSOURCE	指示时钟源： 0：外部时钟 1：处理器时钟。
[1]	TICKINT	使能 SysTick 异常请求： 0：向下计数至 0 不会使能 SysTick 异常请求 1：向下计数至 0 会使能 SysTick 异常请求。软件可以使用 COUNTFLAG 来判断 SysTick 是否曾计数到 0。
[0]	ENABLE	使能计数器： 0：计数器禁能 1：计数器使能。

当 ENABLE 设为 1 时，计数器从加载寄存器加载重载（RELOAD）值，然后递减计数。达到 0 时，它将 COUNTFLAG 设为 1，并可选择根据 TICKINT 的值使能 SysTick。然后它再次加载重载值并开始计数。

39.4.4.2 SysTick 重载值寄存器

加载寄存器规定了载入 VAL 寄存器的起始值。其属性请参见表 814 中寄存器汇总。位分配见表 816。

表816. 加载寄存器的位分配

位	名称	功能
[31:24]	-	保留。
[23:0]	RELOAD	当计数器使能且计数器值到0时要载入 VAL 寄存器的值，参见 39.4.4.2.1 节。

39.4.4.2.1 计算重载值

重载值可以是 0x00000001 至 0x00FFFFFF 范围内的任意值。可以以 0 为起始值，但这样没有效果，因为 SysTick 异常请求和 COUNTFLAG 在从 1 计数到 0 时激活。

根据其用途，计算得到重载值：

- 如要产生一个具有 N 个处理器时钟周期的连拍式（multi-shot）定时器，使用重载值 N-1。例如，如果每 100 时钟脉冲需要有一个 SysTick 中断，则将重载设到 99。
- 如要提供一个经过了 N 个处理器时钟周期延迟后的单次触发，则使用重载值 N。例如，如果要求 400 时钟脉冲后发生 SysTick 中断，则将重载设到 400。

39.4.4.3 SysTick 当前值寄存器

VAL 寄存器包含 SysTick 计数器的当前值。其属性请参见表 814 中寄存器汇总。位分配见表 817。

表817. VAL 寄存器的位分配

位	名称	功能
[31:24]	-	保留。
[23:0]	CURRENT	读取返回 SysTick 计数器的当前值。 向该寄存器写入任意值都可以将该域清零，还会导致 SysTick CTRL.COUNTFLAG 位清零。

39.4.4.4 SysTick 校准值寄存器

CALIB 寄存器指示 SysTick 的校准性质。其属性请参见表 814 中寄存器汇总。位分配见表 818。

表818. CALIB 寄存器的位分配

位	名称	功能
[31]	NOREF	是否有一个可用的单独参考时钟。该位的值在出厂时已预设，参见 25.1 节。
[30]	SKEW	TENMS 的值是否精确。这会影响 SysTick 作为一个软件实时时钟的适合性。该位的值在出厂时已预设，参见 25.1 节。
[29:24]	-	保留。
[23:0]	TENMS	该位的值在出厂时已预设，参见 25.1 节。

如果所用频率与工厂预设值的频率不同，则需要根据处理器时钟或外部时钟频率计算校准值。

39.4.4.5 SysTick 设计提示和建议

SysTick 计数器使用处理器时钟运行。如果 SysTick 计数器靠其运行时，为了低功耗模式而停止了此时钟信号，则 SysTick 计数器将停止。

要确保软件使用对齐的字访问来访问 SysTick 寄存器。

39.4.5 存储器保护单元

本节介绍存储器保护单元（MPU）。

MPU 将存储器映射划分为多个区，并定义了每个区的位置、大小、访问权限和存储器属性。它支持：

- 每个区独立的属性设置；
- 重叠区；
- 存储器属性导出到系统。

存储器属性影响对区的存储器访问的行为。Cortex-M3 MPU 定义了：

- 8 个单独的存储区 0-7；
- 1 个背景区。

当存储区重叠时，存储器访问受序号最大的区的属性影响。例如，区 7 的属性要优先于任何与区 7 重叠的区。

背景区具有与默认存储器映射相同的存储器访问属性，但只能从特权软件访问。

Cortex-M3 MPU 存储器映射是统一的。这就是说指令访问和数据访问具有相同的区设置。

如果某个程序访问被 MPU 禁止的存储单元，则处理器产生一个存储器管理故障。这会导致故障异常，并可能导致 OS 环境中的进程终止。

在 OS 环境中，内核可根据要执行的进程，动态地更新 MPU 区设置。通常，嵌入式 OS 使用 MPU 进行存储器保护。

MPU 区的配置基于存储器的类型，见 [39.3.2.1 “存储区、类型和属性”](#)。

[表 819](#) 列出了可能的 MPU 区属性。它们包括与多数微控制器实现无关的“可共享性”和缓存行为属性。该类实现的编程指南见[表 830](#)。

表819. 存储器属性汇总

存储器类型	可共享性	其他属性	描述
非常有序	-	-	所有对非常有序存储器的访问都按照程序顺序进行。所有非常有序区都假设是可共享的。
设备	可共享	-	一些处理器共享的存储器映射的外设。
	非共享	-	仅供一个处理器使用的存储器映射的外设。

存储器类型	可共享性	其他属性	描述
正常	可共享	不可缓存 直写 可缓存 写回 可缓存	几个处理器共享的正常存储器。
	非共享	不可缓存 直写 可缓存 写回 可缓存	仅供一个处理器使用的正常存储器。

使用 MPU 寄存器来定义 MPU 区及其属性。MPU 寄存器有：

表820. MPU 寄存器汇总

地址	名称	类型	所需特权	复位值	描述
0xE000ED90	TYPE	RO	特权	0x00000800	表 821
0xE000ED94	CTRL	RW	特权	0x00000000	表 822
0xE000ED98	RNR	RW	特权	0x00000000	表 823
0xE000ED9C	RBAR	RW	特权	0x00000000	表 824
0xE000EDA0	RASR	RW	特权	0x00000000	表 825
0xE00EDA4	RBAR_A1	RW	特权	0x00000000	RBAR 的别名，参见 表 824
0xE00EDA8	RASR_A1	RW	特权	0x00000000	RASR 的别名，参见 表 825
0xE00EDAC	RBAR_A2	RW	特权	0x00000000	RBAR 的别名，参见 表 824
0xE00EDB0	RASR_A2	RW	特权	0x00000000	RASR 的别名，参见 表 825
0xE00EDB4	RBAR_A3	RW	特权	0x00000000	RBAR 的别名，参见 表 824
0xE00EDB8	RASR_A3	RW	特权	0x00000000	RASR 的别名，参见 表 825

39.4.5.1 MPU 类型寄存器

类型（TYPE）寄存器指示是否存在 MPU，如果存在，它支持多少个区。该寄存器属性请参见[表 820](#)中的寄存器汇总。位分配见[表 821](#)。

表821. 类型寄存器的位分配

位	名称	功能
[31:24]	-	保留。
[23:16]	IREGION	指示支持的 MPU 指令区数目。 总是包含 0x00。MPU 存储器映射是统一的，描述参见 DREGION 域。

位	名称	功能
[15:8]	DREGION	指示支持的 MPU 数据区数目： 0x08：8 个 MPU 区。
[7:0]	-	保留。
[0]	SEPARATE	指示支持统一的或单独的指令和数据存储器映射： 0：统一的。

39.4.5.2 MPU 控制寄存器

MPU CTRL 寄存器用于：

- 使能 MPU；
- 使能默认存储器映射的背景区；
- 在硬故障、不可屏蔽中断（NMI）和 FAULTMASK 升级的处理程序中时，使能 MPU 的使用。

MPU CTRL 的属性请参见[表 820](#)中的寄存器汇总。位分配见[表 822](#)。

表822. MPU 控制寄存器的位分配

位	名称	功能
[31:3]	-	保留。
[2]	PRIVDEFENA	使能供特权软件访问使用的默认存储器映射： 0：如果 MPU 使能，禁能默认存储器映射的使用。当任意存储器访问一个未被任何使能区覆盖的单元时会导致故障产生。 1：如果 MPU 使能，使能默认存储器映射的使用，将其作为背景区供特权软件访问使用。 使能时，背景区充当区号-1。所有已定义且使能的区的优先级都高于这个缺省映射。 如果 MPU 禁能，处理器忽略该位的写入值。
[1]	HFNMENA	在处于硬故障、NMI 和 FAULTMASK 处理程序时，该位使能 MPU。 当 MPU 使能时： 0：在处于硬故障、NMI 和 FAULTMASK 处理程序时，MPU 被禁能，与 ENABLE 位的值无关。 1：在处于硬故障、NMI 和 FAULTMASK 处理程序时，MPU 被使能。 当 MPU 禁能时，如果该位设为 1，那么行为不可预知。
[0]	ENABLE	使能 MPU： 0：MPU 禁能 1：MPU 使能。

当 ENABLE 和 PRIVDEFENA 都设置为 1 时：

- 对于特权访问，默认存储器映射如[39.3.2 “存储器模型”](#)所述。特权软件对任何不以使能的存储区为地址的访问，其行为都如默认存储器映射的定义。
- 非特权软件对任何不以使能的存储区为地址的访问都会导致一个存储器管理故障。

无论 ENABLE 位的值如何，XN 和非常有序规则总是适用于“系统控制空间”。

当 ENABLE 位设为 1 时，存储器映射的至少一个区必须被使能，以便让系统能工作，除非 PRIVDEFENA 位设为 1。如果 PRIVDEFENA 位设为 1，且没有区被使能，则只有特权软件可工作。

当 ENABLE 位设为 0 时，系统使用默认存储器映射。这与没有实施 MPU 的情况具有相同的存储器属性，见[表 775 “存储器访问行为”](#)。默认存储器映射适用于特权软件和非特权软件的访问。

当使能 MPU 时，对“系统控制空间”和向量表的访问总是被允许。其他区域的访问许可要看存储区的情况，以及 PRIVDEFENA 是否设为 1。

除非 HFNMIENA 设为 1，否则当处理器正为一个优先级为-1 或-2 的异常执行处理程序时，MPU 不被使能。只有在处理一个硬故障或 NMI 异常，或当 FAULTMASK 被使能时，才可能有这样的优先级。设置 HFNMIENA 位到 1，会使能运行在这两个优先级的 MPU。

39.4.5.3 MPU 区号寄存器

RNR 选择哪个存储区被 RBAR 和 RASR 寄存器引用。其属性请参见[表 820](#)中的寄存器汇总。位分配见[表 823](#)。

表823. RNR 位分配

位	名称	功能
[31:8]	-	保留。
[7:0]	REGION	指示被 RBAR 和 RASR 寄存器引用的 MPU 区。 MPU 支持 8 个存储区，因此，该域的允许值范围为 0~7。

通常，在访问 RBAR 或 RASR 前，要将所需的区号写入该寄存器。但是，可通过写入 RBAR 将 VALID 置位为 1 而改变区号，见[表 824](#)。此写入更新 REGION 域的值。

39.4.5.4 MPU 区基址寄存器

RBAR 定义了 RNR 所选择 MPU 区的基址，并可以更新 RNR 的值。其属性请参见[表 820](#)中的寄存器汇总。

将 VALID 置位为 1 写入 RBAR，可改变当前区号和更新 RNR。位分配见[表 824](#)。

表824. RBAR 位分配

位	名称	功能
[31:N]	ADDR	区基址域。N 值取决于区域的大小。更多详情请参见 39.4.5.4.1 节。
[(N-1):5]	-	保留。
[4]	VALID	MPU 区号有效位： 写： 0：RNR 保持不变且处理器： <ul style="list-style-type: none">更新 RNR 中规定区域的基址忽略 REGION 域的值 1：处理器： <ul style="list-style-type: none">将 RNR 的值更新为 REGION 域的值更新 REGION 域规定区域的基址。 读取值总是为 0。
[3:0]	REGION	MPU 区的域： 有关写操作行为，参见 VALID 域的描述。 读取该位返回由 RNR 设定的当前区号。

39.4.5.4.1 ADDR 域

ADDR 域为 RBAR 的位[31:N]。这个区域的大小由 RASR 中的 SIZE 域指定，它定义了 N 的值：

$$N = \text{Log}_2 \text{（单位为字节的区大小）,}$$

如果 RASR 中区的大小被配置为 4GB，则没有有效的 ADDR 域。在此情况下，区占用了全部的存储器映射，基址为 0x00000000。

基址按区的大小调整。例如一个 64 kB 的区必须对齐 64 kB 的倍数，例如，在 0x00010000 或 0x00020000。

39.4.5.5 MPU 区的属性和大小寄存器

RASR 定义了区的大小和 RNR 所指定的 MPU 区的存储器属性，并使能该区及其子区。此寄存器属性请参见[表 820](#)中的寄存器汇总。

RASR 可使用字或半字访问来访问：

位分配见[表 825](#)。

- 高位半字保存区的属性；
- 低位半字保存区的大小以及区和子区的使能位。

表825. RASR 位分配

位	名称	功能
[31:29]	-	保留。
[28]	XN	指令访问禁能位： 0: 指令获取使能 1: 指令获取禁能。
[27]	-	保留。
[26:24]	AP	访问许可域，参见表 829。
[23:22]	-	保留。
[21:19, 17, 16]	TEX, C, B	存储器访问属性，参见表 827。
[18]	S	可共享位，参见表 827。
[15:8]	SRD	子区禁能位。该域的每个位： 0: 相应子区被使能 1: 相应子区被禁能 更多详情请参见 39.4.5.8.3 节。 不支持 128 字节及更小的子区。对该域写属性时，向 SRD 域写入 0x00。
[7:6]	-	保留。
[5:1]	SIZE	指示 MPU 保护区的大小。最小值为 3 (b00010)，更多详情参见 39.4.5.5.1 节。
[0]	ENABLE	区域使能位。

有关访问权限的信息，见 39.4.5.6 节。

39.4.5.5.1 SIZE 域的值

SIZE 域定义了由 RNR 指定的 MPU 存储区的大小。定义如下：

$$(\text{单位为字节的小区的大小}) = 2^{(\text{SIZE}+1)}$$

允许的最小区为 32B，对应的 SIZE 值是 4。表 826 给出了 SIZE 值示例，并列出了相应的小区的大小和 RBAR 中 N 的值。

表826. SIZE 域值示例

SIZE 值	区域大小	N 值 ^[1]	注
b00100 (4)	32 B	5	最小区域值
b01001 (9)	1 kB	10	-
b10011 (19)	1 MB	20	-
b11101 (29)	1 GB	30	-
b11111 (31)	4 GB	b01100	最大区域值

[1] RBAR 中，参见表 824。

39.4.5.6 MPU 访问权限属性

本节描述了 MPU 访问权限属性。RASR 的访问权限位 TEX、C、B、S、AP 和 XN 控制对相应存储区的访问。如果对没有所需权限的存储区域进行访问，则 MPU 产生一个许可故障。

表 827 列出了 TEX、C、B、S 访问权限位的编码。

表827. TEX、C、B 和 S 的编码

TEX	C	B	S	存储器类型	可共享性	其他属性
b000	0	0	x ^[1]	非常有序	可共享	-
		1	x ^[1]	设备	可共享	-
	1	0	0	正常	非共享	外部和内部直写，无写分配。
			1		可共享	
b001	0	0	0	正常	非共享	外部和内部不可缓存。
			1		可共享	
		1	x ^[1]	保留编码		
	1	0	x ^[1]	实现定义的属性。		
		1	0	正常	非共享	外部和内部写回，写和读分配。
			1		可共享	
b010	0	0	x ^[1]	设备	非共享	非共享设备。
		1	x ^[1]	保留编码		-
	1	x ^[1]	x ^[1]	保留编码		-
b1BB	A	A	0	正常	非共享	缓存的存储器 ^[2] ，BB=外部策略，AA=
			1		可共享	内部策略。

[1] MPU 忽略该位的值。
[2] 有关 AA 和 BB 位的编码，参见表 828。

表 828 列出了存储器属性编码的缓存策略，TEX 值在 4~7 范围内。

表828. 存储器属性编码的缓存策略

编码，AA 或 BB	相应缓存策略
00	不可缓存
01	写回，写和读分配
10	直写，无写分配
11	写回，无写分配

表 829 列出了定义特权和非特权软件访问权限的 AP 编码。

表829. AP 编码

AP[2:0]	特权许可	非特权许可	描述
000	不可访问	不可访问	所有的访问都会产生一个许可故障
001	RW	不可访问	只可进行特权软件访问
010	RW	RO	非特权软件进行写操作时会产生一个许可故障
011	RW	RW	完全访问
100	不可预知	不可预知	保留
101	RO	不可访问	只能进行特权软件读操作
110	RO	RO	只能进行特权或非特权软件读操作
111	RO	RO	只能进行特权或非特权软件读操作

39.4.5.7 MPU 不匹配

当一个访问违反 MPU 权限时,处理器产生一个存储器管理故障,见 [39.3.1.4“异常和中断”](#)。MMFSR 指示故障的原因。更多信息见[表 808](#)。

39.4.5.8 更新一个 MPU 区

如要更新一个 MPU 区的属性,需更新 RNR、RBAR 和 RASR 寄存器。可以对每个寄存器单独编程,也可使用一个多字写入,对全部这些寄存器编程。可使用 RBAR 和 RASR 别名利用一条 STM 指令同时对 4 个区编程。

39.4.5.8.1 使用单独的字更新一个 MPU 区

配置一个区的简单代码:

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address

LDR R0,=MPU_RNR      ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]    ; Region Number
STR R4, [R0, #0x4]    ; Region Base Address
STRH R2, [R0, #0x8]   ; Region Size and Enable
STRH R3, [R0, #0xA]   ; Region Attribute
```

如果已使能了要更改的区,则在写入新的区设置到 MPU 前要先禁用该区。例如:

```
; R1 = region number
; R2 = size/enable
```

```
; R3 = attributes

; R4 = address

LDR R0,=MPU_RNR      ; 0xE000ED98, MPU region number register

STR R1, [R0, #0x0]    ; Region Number

BIC R2, R2, #1        ; Disable

STRH R2, [R0, #0x8]   ; Region Size and Enable

STR R4, [R0, #0x4]    ; Region Base Address

STRH R3, [R0, #0xA]   ; Region Attribute

ORR R2, #1            ; Enable

STRH R2, [R0, #0x8]   ; Region Size and Enable
```

以下情况下，软件必须使用存储器屏障指令：

- 如果可能有未完成的存储器传输（如缓冲写入），可能受到 MPU 设置更改的影响，则要在 MPU 设置前使用；
- 如果存储器传输必须使用新的 MPU 设置，则要在 MPU 设置后使用。

但是，如果 MPU 设置进程通过进入一个异常处理程序启动，或后边跟一个异常返回，则不需要存储器屏障指令，因为异常进入和异常返回机制会引起存储器屏障行为。

MPU 设置期间，软件不需要任何存储器屏障指令，因为它通过 PPB 访问 MPU，而 PPB 为一个非常有序型存储区。

例如，如果需要在编程序列后，让全部存储器访问行为立即生效，可以使用一条 DSB 指令和一条 ISB 指令。改变 MPU 设置后，例如上下文切换后，需要使用一个 DSB。如果利用跳转或调用进入一个或多个 MPU 区的编程代码，则需要使用 ISB。如果利用异常返回或是获取异常进入一个编程序列，则不需要 ISB。

39.4.5.8.2 使用多字写入更新一个 MPU 区

根据信息的分割方式，可使用多字写入直接编程。考虑以下再编程（reprogramming）：

```
; R1 = region number

; R2 = address

; R3 = size, attributes in one

LDR R0, =MPU_RNR      ; 0xE000ED98, MPU region number register

STR R1, [R0, #0x0]    ; Region Number

STR R2, [R0, #0x4]    ; Region Base Address

STR R3, [R0, #0x8]    ; Region Attribute, Size and Enable
```

使用一条 STM 指令来优化它：

```
; R1 = region number

; R2 = address

; R3 = size, attributes in one

LDR R0, =MPU_RNR      ; 0xE000ED98, MPU region number register

STM R0, {R1-R3}       ; Region Number, address, attribute, size and enable
```

对于预封装的信息，可用双字实现此操作。这就是说，RBAR 包含所需的区号，并将 VALID 位设为 1，见[表 824](#)。当数据是静态封装时，例如在一个启动加载程序中时，使用此方法：

```
; R1 = address and region number in one

; R2 = size and attributes in one

LDR R0, =MPU_RBAR     ; 0xE000ED9C, MPU Region Base register

STR R1, [R0, #0x0]    ; Region base address and

                        ; region number combined with VALID (bit 4) set to 1

STR R2, [R0, #0x4]    ; Region Attribute, Size and Enable
```

使用一条 STM 指令来优化它：

```
; R1 = address and region number in one

; R2 = size and attributes in one

LDR R0,=MPU_RBAR      ; 0xE000ED9C, MPU Region Base register

STM R0, {R1-R2}       ; Region base address, region number and VALID bit,

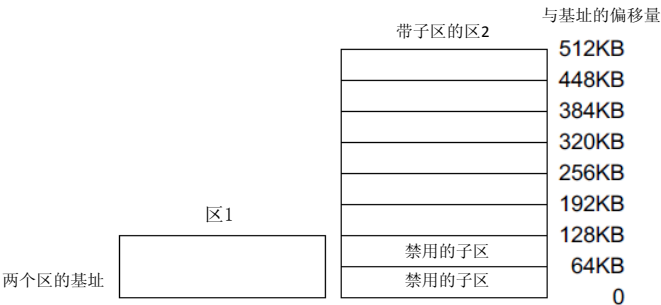
                        ; and Region Attribute, Size and Enable
```

39.4.5.8.3 子区

256 或更多字节的区可划分为 8 个相同大小的子区。可设置 RASR 中 SRD 域内相应的位来禁用某个子区，见[表 825](#)。SRD 的最低位控制第一个子区，最高位控制最后一个子区。禁用某个子区就意味着，另一个与禁用区重叠的区取代了它实现了匹配。如果没有其他使能的区与禁用的子区重叠，MPU 发出一个故障。

32、64 和 128 个字节的区不支持子区，使用这些大小的区时，必须将 SRD 域设置为 0x00，否则 MPU 行为不可预知。

SRD 使用示例：两个具有相同基址的区重叠。区 1 为 128 kB，区 2 为 512 kB。为了保证区 1 的属性应用到第一个 128 kB 区，将区 2 的 SRD 域设置为 b00000011，禁用前两个子区，如图所示。



39.4.5.9 MPU 设计提示和建议

为了避免不期望的行为，在更新某个中断处理程序可能访问的区属性前，禁用中断。

确保软件使用大小正确的对齐访问来访问 MPU 寄存器：

- 除 RASR 外，必须使用对齐字访问；
- 对于 RASR，可使用字节、对齐半字或字访问。

处理器不支持对 MPU 寄存器的非对齐访问。

当设置 MPU 时，如果 MPU 已经编程，则禁用未使用的区，以防止之前任何区的设置影响到新的 MPU 设置。

39.4.5.9.1 微控制器的 MPU 配置

通常，一个微控制器系统只有单个处理器且没有缓存。在该类系统中，MPU 编程如下：

表830. 微控制器的存储区属性

存储区	TEX	C	B	S	存储器类型和属性
Flash 存储器	b000	1	0	0	正常存储器，非共享，直写
内部 SRAM	b000	1	0	1	正常存储器，可共享，直写
外部 SRAM	b000	1	1	1	正常存储器，可共享，写回，写分配
外设	b000	0	1	1	设备存储器，可共享

在大多数微控制器执行中，可共享性和缓存策略的属性都不影响系统行为。然而，如果 MPU 区使用了这些设置，可使应用代码具有更好的可移植性。所列出的值是针对典型情况。在特殊系统中，例如多处理器设计或具有单独 DMA 引擎的设计中，可共享性属性可能会很重要。在这些情况下，请参考存储设备制造商的建议。

39.5 ARM Cortex-M3 用户指南：术语表

中止 — 向处理器表示一个存储器访问相关的值无效的一种机制。中止可能由外部或内部存储器系统引起，作为试图访问无效指令或数据存储器的结果。

对齐 — 如果保存在一个地址的数据项能够被定义数据大小的字节数整除，则此数据项被称为是对齐的。对齐的字和半字地址分别可被 4 和 2 整除。因此，术语字对齐和半字对齐规定，其地址能够分别被 4 和 2 整除。

分组寄存器 — 一个寄存器有多个物理副本，处理器的状态决定了使用哪个副本。堆栈指针 SP (R13) 就是一个分组寄存器。

基址寄存器 — 在指令描述中，加载或存储指令指定一个寄存器，用于保持指令地址计算的基值。根据指令及其寻址模式，可在基址寄存器值上加或减一个偏移量，就形成了要发送到存储器的地址。亦见 *索引寄存器*。

大端 (big-endian) — 字节定序方案，其中递减有效字节保存在存储器的递增地址中。亦见 *字节不变*、*字节序*、*小端*。

大端存储器 — 具有以下特点的存储器：

- 字对齐地址上的字节或半字为此地址的字中的最高有效字节或半字。
- 半字对齐地址的字节为此地址的半字中的最高有效字节。

断点 — 断点是调试器提供的一个机制，用来识别程序执行被停止的指令。断点由程序员插入，以便能够检查程序执行中的寄存器内容，存储单元和某个固定点的变量值，从而检查程序工作是否正常。成功完成程序测试后，断点被移除。

字节不变 (byte-invariant) — 在字节不变系统中，当在小端和大端操作间切换时，存储器每个字节的地址保持不变。当把大于 1 个字节的数据项从存储器加载或存储到存储器时，根据存储器访问的字节序，将构成该数据项的字节安排为正确的顺序。

ARM 的字节不变实现还支持非对齐的半字和字存储器访问。多字访问按字对齐。

缓存 — 片上或片外快速访问存储单元模块，位于处理器和主存储器之间，用于存储和检索常用指令、数据或同时包括指令和数据的副本。这样可大大增加存储器访问的平均速度，改善处理器性能。

条件域 — 指令中一个 4 位的域，它指定了指令可以执行的条件。

上下文 — 对于一个多任务操作系统，各进程运行的环境。在 ARM 处理器中，它仅限于表示可访问的存储器物理地址范围，以及相关的存储器访问权限。

协处理器 — 协助主处理器的一种处理器。Cortex-M3 不支持任何协处理器。

调试器 — 一种包含某个程序的调试系统，用于检测、定位和纠正软件故障，还包括支持软件调试的定制硬件。

直接存储器访问 (DMA) — 无需处理器对相关数据执行任何访问，而直接访问主存的操作。

双字 — 一个 64 位数据项。除非另外说明，其内容被视为一个无符号整数。

双字对齐 — 存储器地址可被 8 整除的数据项。

字节序 (endianness) — 字节定序。该机制用来确定存储在存储器中的一个数据字的连续字节的顺序。是系统的存储器映射的一个方面。亦见小端和大端。

异常 — 中断程序执行的一个事件。当发生异常时，处理器挂起正常的程序流，并开始从相应异常向量所指示的地址执行。指示的地址存有异常处理程序的第一条指令。

一个异常可以是一个中断请求、一个故障，或软件产生的系统异常。故障包括试图进行非法的存储器访问、试图在非法的处理器状态下执行某条指令，以及试图执行一条未定义的指令。

异常服务程序 — 亦见 *中断处理程序*。

异常向量 — 亦见 *中断向量*。

平板地址映射 (flat address mapping) — 一种存储器组织体系，其中存储器空间中每个物理地址都与对应的虚拟地址相同。

半字 — 一个 16 位的数据项。

非法指令 — 结构上未定义的指令。

实现定义的 (Implementation-defined) — 结构上未定义，但在单独实现中定义并证明的。

实现特定的 (Implementation-specific) — 结构上未定义，且不必在单独实现中证明。当有多个实现选项可供选择，且所选选项不影响软件兼容性时，使用该“实现特定的”。

索引寄存器 — 在某些加载和存储指令描述中，此寄存器的值被作为一个偏移量，将其加入基址寄存器值或从基址寄存器中减去，以形成发送到存储器的地址。某些寻址模式可以选择在加法或减法前，使能被移位的索引寄存器值。亦见 *基址寄存器*。

指令周期数 — 指令占用流水线执行阶段所消耗的周期数。

中断处理程序 — 发生中断时，处理器的控制权传递到这个程序。

中断向量 — 存储器低地址空间或存储器高地址空间（如果配置了高位向量）内一些固定地址，存有相应中断处理程序的第一条指令。

小端 (little-endian) — 字节定序机制，其中一个数据字中的递增有效的字节保存在存储器的递增地址中。亦见大端、*字节不变*、*字节序*。

小端存储器 — 具有以下特点的存储器：

- 字对齐地址中的字节或半字为此地址字中的最低有效字节或半字。
- 半字对齐地址中的字节为此地址半字中的最低有效字节。

亦见 *大端存储器*。

加载/存储架构 — 一种处理器架构，数据处理操作只针对寄存器的内容，而不直接对存储器内容。

存储器保护单元 (MPU) — 控制对存储器模块访问权限的硬件。MPU 不进行任何地址转换。

预取 — 在流水线处理器中，当前面的指令执行完成前，从存储器获取指令以填充流水线的过程。预取一条指令并不意味着该指令一定会执行。

读 — 读定义为具有一种加载语义的存储器操作。读包括 Thumb 指令 LDM、LDR、LDRSH、LDRH、LDRSB、LDRB 和 POP。

区 — 存储器空间的分区。

保留 — 在控制寄存器或指令格式中，如果某个域被实现定义了，或者在该域不为零时会产生不可预知的结果，则该域是保留域。这些域保留用于将来的结构扩展，或是由实现特定使用。所有未被实现使用的保留位都必须写为 0，且读取为 0。

应为 1 (Should Be One, SBO) — 软件写 1 或位域全写 1。写入 0 会产生不可预知的结果。

应为 0 (Should Be Zero, SBZ) — 软件写 0 或位域全写 0。写入 1 会产生不可预知的结果。

应为 0 或保留 (Should Be Zero or Preserved, SBZP) — 软件写 0 或位域全写 0，或通过将前面读出的相同处理器上相同域的值写回来保留。

线程安全 (Thread-safe) — 在多任务环境中，线程安全功能在访问共享资源时使用保护机制，以确保操作正确，而不会有共享访问冲突的风险。

Thumb 指令 — 指定处理器完成一个操作的一个或两个半字。Thumb 指令必须按半字对齐。

非对齐 — 存储在不能被定义的数据大小的字节数整除的地址的数据项被称为“非对齐”。例如，存储在不能被 4 整除的地址的一个字。

不可预知 (UNP) — 不能确知的行为。不可预知的行为不代表安全漏洞。不可预知的行为不得让处理器或系统的任何部分停止或挂起。

热复位 — 又称内核复位。它对处理器的大部分元件初始化，调试控制器和调试逻辑除外。这种类型的复位在使用处理器的调试功能时非常有用。

WA — 见 *写分配*。

WB — 见 *写回*。

字 — 一个 32 位的数据项。

写 — 写定义为具有加载语义的存储器操作。写包括 Thumb 指令 STM、STR、STRH、STRB 和 PUSH。

写分配 (write-allocate) — 在一个写分配缓存中，存储数据的缓存命中会使一个缓存线被分配到缓存中。

写回 (write-back) — 在写回缓存中，仅在缓存命中后行替换时，数据被强制清出缓存，才将数据写入主存储器。否则，处理器的写入只更新缓存。这又称回拷 (copyback)。

写缓冲 — 位于数据缓存和主存之间的高速存储器模块，用作一个 FIFO 缓冲区，目的是优化到存储器的存储。

直写 (write-through) — 在直写缓存中，数据在缓存被更新的同时写入主存储器。

40.1 缩写

表831. 缩写

缩写	描述
ADC	模数转换器
AHB	先进的高性能总线
AMBA	先进的微控制器总线架构
APB	先进的外设总线
BOD	掉电检测
CAN	控制器局域网络
DAC	数模转换器
DMA	直接存储器访问
EOP	包结束
ETM	嵌入式跟踪宏单元
GPIO	通用输入/输出
I2C	IC 间控制总线
I2S	IC 间音频总线
IrDA	红外数据协会
JTAG	联合测试行动组
MII	媒体独立接口（以太网相关）
MIIM	媒体独立接口管理（以太网相关）
PHY	物理层接口
PLL	锁相环
PWM	脉宽调制器
QEI	正交编码器接口
RMI	简化媒体独立接口（以太网相关）
SE0	单端零（USB 相关）
SPI	串行外设接口
SSI	串行同步接口
SSP	同步串口
UART	通用异步接收器/发送器
USB	通用串行总线

40.2 法律信息

40.2.1 定义

草案——本文件只是草案。对于文件内容，仍在进行内部审查，并有待内部审批，可能修订或添加新的内容。恩智浦半导体公司不会担保或保证文件中所包含内容的准确性或完整性，不应由该等信息引起的后果承担任何责任。

40.2.2 免责条款

有限保证与责任——本文件中的信息被视为准确可靠的。然而，恩智浦半导体公司不提供任何陈述或担保，包括明示或暗示，保障该等信息的准确性或完整性，并且不应由使用该等信息引起的后果承担任何责任。

无论在任何情况下，恩智浦半导体公司均不应由任何间接、偶然、特殊或间接损害（包括但不限于利润损失、储蓄损失、业务中断、拆除或更换任何产品成本或者返工费用等）承担任何责任，无论该等损害是基于侵权（包括疏忽）、保证、违约还是任何其他法律理论。

尽管客户可能由于任何原因而遭受任何损害，恩智浦半导体公司对于客户就本文所述产品的总体和累计责任应按照恩智浦半导体公司的商业销售条款和条件而有所限制。

变更权利——恩智浦半导体公司保留随时更改本文件中所发布文件的权利，包括但不限于规范和产品描述，而无需提前给出任何通知。本文件代替和更换文件发布之前提供的所有信息。

适用性——恩智浦半导体公司的产品并非设计、授权或保证适用于维持生命、生命攸关或者对安全至关重要的系统或设备，也不用于由于恩质朴半导体电子的

产品故障而会可能合理导致人员伤亡或者严重财产或环境损害的应用。恩智浦半导体公司不接受因为在该等设备或应用中包含或使用其产品用于而承担任何责任，包含或使用其产品所造成的风险应由客户自行承担。

应用——本文中针对这些产品所述应用仅用于说明目的。恩智浦半导体公司不保证该等应用不经进一步测试或修订而适用于特定用途。客户负责使用恩智浦半导体公司的产品设计和运行其应用及产品，并且恩智浦半导体公司无义务对应用或客户产品设计提供任何援助。客户应承担全部责任，确定恩智浦半导体公司的产品是否适用于其所规划的应用和产品，以及所规划的应用及使用客户的第三方客户。客户应当提供使用的设计和运行保障，以便最大限度减少与其应用和产品有关的风险。

对于由客户的应用或产品中的任何缺点或违约或者应用客户的第三方客户所引起的任何违约、损害或成本，恩智浦半导体公司不承担任何责任。客户负责进行所有必要测试，检测其使用恩智浦半导体公司产品的应用及产品，以便避免应用和产品的违约，或者避免客户的第三方客户的应用或使用所造成的违约。对于该等违约事宜，恩智浦半导体公司不承担任何责任。

出口控制——本文件及本文件所述条目受出口控制规定制约。出口可能需要事先获得国家权威部门的授权。

40.2.3 商标

注：本文件所提及品牌、产品名称、服务名称和商标属于其各自所有者。

l²C-bus 标识是 NXP B.V 的商标。

40.3 表目录

表 1.	订购信息	9	表 30.	CPU 时钟选择寄存器 (CCLKSEL—0x400F C104) 位描述	53
表 2.	LPC178x/177x 器件订购选项	10	表 31.	USB 时钟选择寄存器 (USBCLKSEL—0x400F C108) 位描述	54
表 3.	LPC178x/177x 存储器使用及明细	15	表 32.	EMC 时钟选择寄存器 (EMCCLKSEL—0x400F C100) 位描述	54
表 4.	AHB 外设及基址	18	表 33.	外设时钟选择寄存器 (PCLKSEL—0x400F C1A8) 位描述	55
表 5.	APB0 外设及基址	18	表 34.	功率控制寄存器	58
表 6.	APB1 外设及基址	19	表 35.	功率模式控制寄存器 (PCON—0x400F C0C0) 位描述	59
表 7.	矩阵仲裁寄存器 (矩阵_仲裁-0x400F C188) 位描述	20	表 36.	低功耗模式的编码	60
表 8.	管脚汇总	22	表 37.	外设功率控制寄存器 (PCONP—0x400F C0C4) 位描述	61
表 9.	系统控制模块寄存器汇总	23	表 38.	功率提升控制寄存器 (PBOOST—0x400F C1B0) 位描述	62
表 10.	复位源标识寄存器 (RSID—0x400F C180) 位描述	26	表 39.	低功耗模式的编码	63
表 11.	复位控制寄存器 0 (RSTCON0—0x400F C1CC) 位描述	27	表 40.	时钟输出配置寄存器 (CLKOUTCFG—0x400F C1C8) 位描述	66
表 12.	复位控制寄存器 1 (RSTCON1—0x400F C1D0) 位描述	28	表 41.	存储器加速模块寄存器汇总	68
表 13.	外部中断寄存器	31	表 42.	存储器加速模块配置寄存器 (FLASHCFG—0x400F C000) 位描述	69
表 14.	外部中断标志寄存器 (EXTINT—0x400F C140) 位描述	32	表 43.	连接到向量中断控制器的中断源	72
表 15.	外部中断模式寄存器 (EXTMODE—0x400F C148) 位描述	33	表 44.	NVIC 寄存器映射	75
表 16.	外部中断极性寄存器 (EXTPOLAR—0x400F C14C) 位描述	34	表 45.	中断使能设置寄存器 0 寄存器 (ISER0—0xE000 E100)	76
表 17.	系统控制和状态寄存器 (SCS—0x400F C1A0) 位描述	35	表 46.	中断使能设置寄存器 1 寄存器 (ISER1—0xE000 E104)	77
表 18.	系统控制寄存器汇总	39	表 47.	中断使能清除寄存器 0 (ICER0—0xE000 E180)	78
表 19.	振荡模式下 C _{X1/X2} 的建议取值 (晶体和外部元件参数) 低频模式 (OSCRANGE = 0, 参见表 17)	41	表 48.	中断使能清除寄存器 1 寄存器 (ICER1—0xE000 E184)	79
表 20.	振荡模式下 C _{X1/X2} 的建议取值 (晶体和外部元件参数) 高频模式 (OSCRANGE = 1, 参见表 17)	41	表 49.	中断暂挂设置寄存器 0 寄存器 (ISPR0—0xE000 E200)	80
表 21.	时钟源选择寄存器位描述	43	表 50.	中断暂挂设置寄存器 1 寄存器 (ISPR1—0xE000 E204)	81
表 22.	PLL1 寄存器	45	表 51.	中断暂挂清除寄存器 0 寄存器 (ICPR0—0xE000 E280)	82
表 23.	PLL 控制寄存器 (PLL0CON—0x400F C080, PLL1CON—0x400F C0A0) 位描述	46	表 52.	中断清除暂挂寄存器 1 寄存器 (ICPR1—0xE000 E284)	83
表 24.	PLL 配置寄存器 (PLL0CFG—0x400F C084, PLL1CFG—0x400F C0A4) 位描述	46	表 53.	中断活动位寄存器 0 (IABR0—0xE000 E300)	84
表 25.	PLL 状态寄存器 (PLL0STAT—0x400F C088, PLL1STAT—0x400F C0A8) 位描述	46	表 54.	中断活动位寄存器 1 (IABR1—0xE000 E304)	85
表 26.	PLL 馈送寄存器 (PLL0FEED—0x400F C08C, PLL1FEED—0x400F C0AC) 位描述	47	表 55.	中断优先级寄存器 0 (IPR0—0xE000 E400)	86
表 27.	PLL 频率参数	49	表 56.	中断优先级寄存器 1 (IPR1—0xE000 E404)	
表 28.	PLL 倍频器值	50			
表 29.	PLL 分频器值	51			

表 57.	中断优先级寄存器 2 (IPR2—0xE000 E408)	86	表 90.	高速 GPIO 端口方向控制字节和半字访问寄存器描述	134
表 58.	中断优先级寄存器 3 (IPR3—0xE000 E40C)	87	表 91.	高速 GPIO 端口输出设置寄存器 (FIO0SET~FIO4SET—0x2009 8018~0x2009 8098) 位描述	135
表 59.	中断优先级寄存器 4 (IPR4—0xE000 E410)	87	表 92.	高速 GPIO 端口输出设置字节和半字访问寄存器描述	136
表 60.	中断优先级寄存器 5 (IPR5—0xE000 E414)	88	表 93.	高速 GPIO 端口输出清零寄存器 (FIO0CLR~FIO4CLR—0x2009 801C~0x2009 809C) 位描述	137
表 61.	中断优先级寄存器 6 (IPR6—0xE000 E418)	88	表 94.	高速 GPIO 端口输出清零字节和半字访问寄存器描述	138
表 62.	中断优先级寄存器 7 (IPR7—0xE000 E41C)	89	表 95.	高速 GPIO 端口管脚值寄存器 (FIO0PIN~FIO4PIN—0x2009 8014~0x2009 8094) 位描述	139
表 63.	中断优先级寄存器 8 (IPR8—0xE000 E420)	89	表 96.	高速 GPIO 端口管脚值字节和半字访问寄存器描述	139
表 64.	中断优先级寄存器 9 (IPR9—0xE000 E424)	90	表 97.	高速 GPIO 端口屏蔽寄存器 (FIO0MASK~FIO4MASK—0x2009 8010~0x2009 8090) 位描述	140
表 65.	中断优先级寄存器 10 (IPR10—0xE000 E428)	90	表 98.	高速 GPIO 端口屏蔽字节和半字访问寄存器描述	141
表 66.	软件触发中断寄存器 (STIR—0xE000 EF00)	91	表 99.	GPIO 整体中断状态寄存器 (IOIntStatus—0x4002 8080) 位描述	142
表 67.	LPC178x/177x 管脚描述	91	表 100.	端口 0 上升沿的 GPIO 中断使能 (IO0IntEnR—0x4002 8090) 位描述	143
表 68.	P3[15]/D15 和 P3[14]/D14 管脚的引导控制	93	表 101.	端口 2 上升沿的 GPIO 中断使能 (IO2IntEnR—0x4002 80B0) 位描述	144
表 69.	I/O 管脚配置寄存器汇总	108	表 102.	端口 0 下降沿的 GPIO 中断使能 (IO0IntEnF—0x4002 8094) 位描述	145
表 70.	端口 0 的 I/O 控制寄存器	109	表 103.	端口 2 下降沿的 GPIO 中断使能 (IO2IntEnF—0x4002 80B4) 位描述	146
表 71.	端口 1 的 I/O 控制寄存器	113	表 104.	端口 0 上升沿中断的 GPIO 中断状态 (IO0IntStatR—0x4002 8084) 位描述	147
表 72.	端口 2 的 I/O 控制寄存器	114	表 105.	端口 2 上升沿中断的 GPIO 中断状态 (IO2IntStatR—0x4002 80A4) 位描述	148
表 73.	端口 3 的 I/O 控制寄存器	115	表 106.	端口 0 下降沿中断的 GPIO 中断状态 (IO0IntStatF—0x4002 8088) 位描述	149
表 74.	端口 4 的 I/O 控制寄存器	116	表 107.	端口 2 下降沿中断的 GPIO 中断状态 (IO2IntStatF—0x4002 80A8) 位描述	150
表 75.	端口 5 的 I/O 控制寄存器	117	表 108.	端口 0 的 GPIO 中断清零寄存器 (IO0IntClr—0x4002 808C) 位描述	151
表 76.	D 类型 IOCON 寄存器位描述	118	表 109.	端口 2 的 GPIO 中断清零寄存器 (IO2IntClr—0x4002 80AC) 位描述	152
表 77.	D 型 I/O 控制寄存器: FUNC 值和管脚功能	120	表 110.	EMC 配置	154
表 78.	A 型 IOCON 寄存器位描述	121	表 111.	存储器组的选择	161
表 79.	A 型 I/O 控制寄存器: FUNC 值和管脚功能	122	表 112.	焊盘接口和控制信号描述	162
表 80.	U 型 IOCON 寄存器位描述	127			
表 81.	U 型 I/O 控制寄存器: FUNC 值和管脚功能	128			
表 82.	I 型 IOCON 寄存器位描述	128			
表 83.	I 型 I/O 控制寄存器: FUNC 值和管脚功能	129			
表 84.	W 型 IOCON 寄存器位描述	129			
表 85.	W 型 I/O 控制寄存器: FUNC 值和管脚功能	130			
表 86.	GPIO 管脚描述	130			
表 87.	GPIO 寄存器映射 (局部总线可访问寄存器—增强型 GPIO 特性)	132			
表 88.	GPIO 中断映射寄存器	133			
表 89.	高速 GPIO 端口方向寄存器 (FIO0DIR~FIO5DIR—0x2009 8000~0x2009 80A0) 位描述	134			

表 113. EMC 寄存器汇总	163	0x2009 C120,0x2009 C140, 0x2009 C160)	174
表 114. EMC 控制寄存器位描述 (EMCControl-0x2009 C000)	165	表 133. 地址映射	175
表 115. EMC 状态寄存器位描述 (EMCStatus—0x2009 C008)	166	表 134. 动态存储器 RAS & CAS 延迟寄存器位描述 (EMCDynamicRasCas0~3—0x2009 C104,0x2009 C124, 0x2009 C144, 0x2009 C164)	177
表 116. EMC 配置寄存器位描述 (EMCConfig—0x2009 C008)	166	表 135. 静态存储器配置寄存器位描述 (EMCStaticConfig0~3—0x2009 C200, 0x2009 C220,0x2009 C240, 0x2009 C260)	177
表 117. 动态控制寄存器位描述 (EMCDynamicControl-0x2009 C020)	167	表 136. 静态存储器写使能延迟寄存器位描述 (EMCStaticWaitWen0~3—0x2009 C204,0x2009 C224,0x2009 C244, 0x2009 C264)	179
表 118. 动态存储器刷新定时器寄存器位描述 (EMCDynamicRefresh—0x2009 C024)	168	表 137. 静态存储器输出使能延迟寄存器位描述 (EMCStaticWaitOen0~3—0x2009 C208, 0x2009 C228,0x2009 C248, 0x2009 C268)	179
表 119. 动态存储器读取配置寄存器位描述 (EMCDynamicReadConfig—0x2009 C028)	168	表 138. 静态存储器读取延迟寄存器位描述 (EMCStaticWaitRd0~3—0x2009 C20C, 0x2009 C22C,0x2009 C24C, 0x2009 C26C)	180
表 120. 动态存储器预充电命令周期寄存器位描述 (EMCDynamicRP—0x2009 C030) ...	169	表 139. 静态存储器页读模式读取延迟寄存器 0~3 位 描述 (EMCStaticWaitPage0~3—0x2009 C210,0x2009 C230, 0x2009 C250, 0x2009 C270)	180
表 121. 动态存储器有效到预充电命令周期寄存器位 描述 (EMCDynamicRAS – 0x2009 C034)	169	表 140. 静态存储器写延迟寄存器 0~3 位描述 (EMCStaticWaitWr—0x2009 C214, 0x2009 C234,0x2009 C254, 0x2009 C274)	181
表 122. 动态存储器自刷新退出时间寄存器位描述 (EMCDynamicSREX—0x2009 C038)	170	表 141. 静态存储器周转延迟寄存器 0~3 位描述 (EMCStaticWaitTurn0~3—0x2009 C218,0x2009 C238, 0x2009 C258, 0x2009 C278)	181
表 123. 动态存储器最后数据输出至有效时间寄存器 的位描述 (EMCDynamicAPR— 0x2009 C03C)	170	表 142. 延迟控制寄存器位描述 (EMCDLYCTL—0x400F C1DC)	182
表 124. 动态存储器数据输入至有效命令时间寄存器 位描述 (EMCDynamicDAL—0x2009 C040)	170	表 143. EMC 校准寄存器位描述 (EMCCAL—0x400F C1E0)	183
表 125. 动态存储器写入恢复时间寄存器 (EMCDynamicWR— 0x2009 C044) 位描 述	171	表 144. 以太网的缩写词与定义	189
表 126. 动态存储器有效至有效命令周期寄存器位描 述 (EMCDynamicRC—0x2009 C048)	171	表 145. PHY 设备实例	194
表 127. 动态存储器自刷新周期寄存器 (EMCDynamicRFC—0x2009 C04C) 位描 述	172	表 146. 以太网 MII 的管脚描述	194
表 128. 动态存储器退出自刷新寄存器位描述 (EMCDynamicXSR—0x2009 C050)	172	表 147. 以太网 RMII 的管脚描述	194
表 129. 动态存储器有效组 A 至有效组 B 时间寄存器 位描述 (EMCDynamicRRD—0x2009 C054)	173	表 148. 以太网 MIIM 的管脚描述	194
表 130. 动态存储器装载模式寄存器至有效命令时间 位描述 (EMCDynamicMRD—0x2009 C058)	173	表 149. 以太网寄存器描述	195
表 131. 静态存储器延长等待寄存器位描述 (EMCStaticExtendedWait—0x2009 C080)	174	表 150. MAC 配置寄存器 1 位描述 (MAC1—0x2008 4000)	197
表 132. 动态存储器配置寄存器位描述 (EMCDynamicConfig0~3—0x2009 C100,		表 151. MAC 配置寄存器 2 位描述 (MAC2—0x2008 4004)	197
		表 152. 填充操作	198

表 153. 连续两包的内部包间隔寄存器位描述 (IPGT—0x2008 4008)	199	表 178. 发送描述符数目寄存器位描述 (TxDescriptorNumber—0x2008 4124)	208
表 154. 非连续两包的内部包间隔寄存器位描述 (IPGR—0x2008 400C)	199	表 179. 发送产生索引寄存器 (TxProduceIndex—0x2008 4128) 位描述	208
表 155. 冲突窗口/重试寄存器位描述 (CLRT—0x2008 4010)	199	表 180. 发送消耗索引寄存器位描述 (TxConsumeIndex—0x2008 412C) ...	209
表 156. 最大帧寄存器位描述 (MAXF—0x2008 4014)	200	表 181. 发送状态向量 0 寄存器位描述 (TSV0—0x2008 4158)	209
表 157. PHY 支持寄存器位描述 (SUPP—0x2008 4018)	200	表 182. 发送状态向量 1 寄存器位描述 (TSV1—0x2008 415C)	210
表 158. 测试寄存器位描述 (TEST—0x2008 401C)	200	表 183. 接收状态向量寄存器位描述 (RSV—address 0x2008 4160)	210
表 159. MII Mgmt 配置寄存器位描述 (MCFG—0x2008 4020)	201	表 184. 流控制计数器寄存器位描述 (FlowControlCounter—0x2008 4170)	211
表 160. 时钟选择的编码	201	表 185. 流控制状态寄存器位描述 (FlowControlStatus—0x2008 4174) ..	211
表 161. MII Mgmt 命令寄存器位描述 (MCMD—0x2008 4024)	202	表 186. 接收滤波器控制寄存器位描述 (RxFilterCtrl—0x2008 4200)	212
表 162. MII Mgmt 地址寄存器位描述 (MADR—0x2008 4028)	202	表 187. 接收滤波器 WoL 状态寄存器位描述 (RxFilterWoLStatus—0x2008 4204) ..	212
表 163. MII Mgmt 写数据寄存器位描述 (MWTD—0x2008 402C)	202	表 188. 接收滤波器 WoL 清零寄存器位描述 (RxFilterWoLClear—0x2008 4208) ...	213
表 164. MII Mgmt 读数据寄存器 (MRDD—0x2008 4030) 位描述	203	表 189. Hash 滤波器表 LSB 寄存器位描述 (HashFilterL—0x2008 4210)	213
表 165. MII Mgmt 指示寄存器位描述 (MIND—0x2008 4034)	203	表 190. Hash 滤波器表 MSB 寄存器位描述 (HashFilterH—0x2008 4214)	213
表 166. 站地址寄存器位描述 (SA0—0x2008 4040)	204	表 191. 中断状态寄存器位描述 (IntStatus—0x2008 4FE0)	214
表 167. 站地址寄存器位描述 (SA1—0x2008 4044)	204	表 192. 中断使能寄存器位描述 (intEnable—0x2008 4FE4)	214
表 168. 站地址寄存器位描述 (SA2—0x2008 4048)	204	表 193. 中断清零寄存器位描述 (IntClear—0x2008 4FE8)	215
表 169. 命令寄存器位描述 (Command—0x2008 4100)	205	表 194. 中断置位寄存器位描述 (IntSet—0x2008 4FEC)	215
表 170. 状态寄存器位描述 (Status—0x2008 4104)	205	表 195. 掉电寄存器位描述 (PowerDown—0x2008 4FF4)	216
表 171. 接收描述符基址寄存器位描述 (RxDescriptor—0x2008 4108)	206	表 196. 接收描述符的区域	218
表 172. 接收状态基址寄存器位描述 (RxStatus—0x2008 410C)	206	表 197. 接收描述符控制字	218
表 173. 接收描述符数目寄存器位描述 (RxDescriptor—0x2008 4110)	206	表 198. 接收状态的区域	218
表 174. 接收产生索引寄存器位描述 (RxProduceIndex—0x2008 4114)	207	表 199. 接收状态 Hash CRC 字	218
表 175. 接收消耗索引寄存器位描述 (RxConsumeIndex—0x2008 4118) ...	207	表 200. 接收状态信息字	219
表 176. 发送描述符基址寄存器位描述 (TxDescriptor—0x2008 411C)	207	表 201. 发送描述符的区域	221
表 177. 发送状态基址寄存器位描述 (TxStatus—0x2008 4120)	208	表 202. 发送描述符的控制字	221
		表 203. 发送状态的区域	221
		表 204. 发送状态信息字	221
		表 205. LCD 控制器管脚	258
		表 206. 单面板 STN 显示器使用的管脚	259

表 207.	双面板 STN 显示器使用的管脚	259	表 238.	光标图像寄存器 (CRSR_IMG, RW — 0x2008 8800 到 0x2008 8BFC)	291
表 208.	TFT 显示器使用的管脚	260	表 239.	光标控制寄存器 (CRSR_CTRL, RW — 0x2008 8C00)	291
表 209.	与小端字节和小端像素次序相对应的 FIFO 位	264	表 240.	光标配置寄存器 (CRSR_CFG, RW — 0x2008 8C04)	292
表 210.	与大端字节和大端像素次序相对应的 FIFO 位	265	表 241.	光标调色板寄存器 0 (CRSR_PAL0, RW — 0x2008 8C08)	292
表 211.	与小端字节和大端像素次序相对应的 FIFO 位	266	表 242.	光标调色板寄存器 1 (CRSR_PAL1, RW — 0x2008 8C0C)	293
表 212.	RGB 模式下的数据格式	267	表 243.	光标 XY 位置寄存器 (CRSR_XY, RW — 0x2008 8C10)	293
表 213.	TFT 模式下的调色板数据存储	268	表 244.	光标剪裁位置寄存器 (CRSR_CLIP, RW — 0x2008 8C14)	294
表 214.	STN 彩色模式下的调色板数据存储	268	表 245.	光标中断屏蔽寄存器 (CRSR_INTMSK, RW — 0x2008 8C20)	294
表 215.	STN 单色模式下的调色板数据存储	269	表 246.	光标中断清零寄存器 (CRSR_INTCLR, RW — 0x2008 8C24)	294
表 216.	STN 单色模式下的调色板数据存储	269	表 247.	光标原始中断状态寄存器 (CRSR_INTRAW, RW — 0x2008 8C28)	295
表 217.	32 x 32 光标地址	273	表 248.	光标屏蔽中断状态寄存器 (CRSR_INTSTAT, RW — 0x2008 8C2C)	295
表 218.	32 x 32 像素光标格式下缓冲到像素映射	273	表 249.	STN 单面板模式下 LCD 面板连接	300
表 219.	64x 64 像素光标格式下缓冲到像素映射	274	表 250.	TN 双面板模式下 LCD 面板连接	301
表 220.	像素编码	274	表 251.	TFT 面板所使用的 LCD 面板连接	302
表 221.	2 2/3 像素数据驱动的彩色显示器	275	表 252.	本章用到的与 USB 相关的首字母缩写词、简写以及定义	304
表 222.	LCD 控制器寄存器汇总	279	表 253.	固定的端点配置	305
表 223.	LCD 配置寄存器 (LCD_CFG, RW — 0x400F C1B8)	280	表 254.	USB 外部接口	308
表 224.	水平时序寄存器 (LCD_TIMH, RW — 0x2008 8000)	280	表 255.	USB 设备控制器时钟源	309
表 225.	垂直时序寄存器 (LCD_TIMV, RW — 0x2008 8004)	281	表 256.	USB 设备寄存器映射	311
表 226.	时钟和信号极性寄存器 (LCD_POL, RW — 0x2008 8008)	282	表 257.	USB 端口选择寄存器 (USBPortSel—0x2008 C110) 位描述	312
表 227.	线端控制寄存器 (LCD_LE, RW — 0x2008 800C)	283	表 258.	USBClkCtrl 寄存器 (USBClkCtrl—0x2008 CFF4) 位描述	313
表 228.	上面板帧基址寄存器 (LCD_UPBASE, RW — 0x2008 8010)	283	表 259.	USB 时钟状态寄存器 (USBClkSt—0x2008 CFF8) 位描述	313
表 229.	下面板帧基址寄存器 (LCD_LPBASE, RW — 0x2008 8014)	284	表 260.	USB 中断状态寄存器 (USBIntSt—0x2008 C1C0) 位描述	314
表 230.	LCD 控制寄存器 (LCD_CTRL, RW — 0x2008 8018)	285	表 261.	USB 设备中断状态寄存器 (USBDevIntSt—0x2008 C200) 位分配	314
表 231.	中断屏蔽寄存器 (LCD_INTMSK, RW — 0x2008 801C)	287	表 262.	USB 设备中断状态寄存器 (USBDevIntSt—0x2008 C200) 位描述	315
表 232.	原始中断状态寄存器 (LCD_INTRAW, RW — 0x2008 8020)	288	表 263.	USB 设备中断使能寄存器 (USBDevIntEn—0x2008 C204) 位分配	315
表 233.	被屏蔽中断状态寄存器 (LCD_INTSTAT, RW — 0x2008 8024)	288	表 264.	USB 设备中断使能寄存器	
表 234.	中断清零寄存器 (LCD_INTCLR, RW — 0x2008 8028)	289			
表 235.	上面板当前地址寄存器 (LCD_UPCURR, RW — 0x2008 802C)	289			
表 236.	下面板当前地址寄存器 (LCD_LPCURR, RW — 0x2008 8030)	290			
表 237.	彩色调色板寄存器 (LCD_PAL, RW — 0x2008 8200 到 0x2008 83FC)	290			

	(USBDevIntEn—0x2008 C204) 位描述	316
表 265.	USB 设备中断清除寄存器 (USBDevIntClr—0x2008 C208) 位分配	316
表 266.	USB 设备中断清零寄存器 (USBDevIntClr—0x2008 C208) 位描述	316
表 267.	USB 设备中断设置寄存器 (USBDevIntSet—0x2008 C20C) 位分配	317
表 268.	USB 设备中断设置寄存器 (USBDevIntSet—0x2008 C20C) 位描述	317
表 269.	USB 设备中断优先级寄存器 (USBDevIntPri—0x2008 C22C) 位描述	317
表 270.	USB 端点中断状态寄存器 (USBEPIntSt—0x2008 C230) 位分配	318
表 271.	USB 端点中断状态寄存器 (USBEPIntSt—0x2008 C230) 位描述	318
表 272.	USB 端点中断使能寄存器 (USBEPIntEn—0x2008 C234) 位分配	319
表 273.	USB 端点中断使能寄存器 (USBEPIntEn—0x2008 C234) 位描述	319
表 274.	USB 端点中断清除寄存器 (USBEPIntClr—0x2008 C238) 位分配	320
表 275.	USB 端点中断清除寄存器 (USBEPIntClr—0x2008 C238) 位描述	320
表 276.	USB 端点中断设置寄存器 (USBEPIntSet—0x2008 C23C) 位分配	321
表 277.	USB 端点中断设置寄存器 (USBEPIntSet—0x2008 C23C) 位描述	321
表 278.	USB 端点中断优先级寄存器 (USBEPIntPri—0x2008 C240) 位分配	321
表 279.	USB 端点中断优先级寄存器 (USBEPIntPri—0x2008 C240) 位描述	322
表 280.	USB 使用端点寄存器 (USBReEp—0x2008 C244) 位分配	323
表 281.	USB 使用端点寄存器 (USBReEp—0x2008 C244) 位描述	323
表 282.	USB 端点索引寄存器 (USBEPIn—0x2008 C248) 位描述	324
表 283.	USB MaxPacketSize 寄存器 (USBMaxPSize—0x2008 C24C) 位描述	324
表 284.	USB 接收数据寄存器 (USBRxData—0x2008 C218) 位描述	325
表 285.	USB 接收包长度寄存器 (USBRxPlen—0x2008 C220) 位描述	325
表 286.	USB 发送数据寄存器 (USBTxData—0x2008 C21C) 位描述	325
表 287.	USB 发送包长度寄存器 (USBTxPLen—0x2008 C224) 位描述	326
表 288.	USB 控制寄存器 (USBCtrl—0x2008 C228) 位描述	326
表 289.	USB 命令代码寄存器 (USBCmdCode—0x2008 C210) 位描述	327
表 290.	USB 命令数据寄存器 (USBCmdData—0x2008 C214) 位描述	327
表 291.	USB DMA 请求状态寄存器 (USBDMARSt—0x2008 C250) 位分配	328
表 292.	USB DMA 请求状态寄存器 (USBDMARSt—0x2008 C250) 位描述	328
表 293.	USB DMA 请求清除寄存器 (USBDMARClr—0x2008 C254) 位描述	329
表 294.	USB DMA 请求设置寄存器 (USBDMARSet—0x2008 C258) 位描述	329
表 295.	USB UDCA Head 寄存器 (USBUDCAH—0x2008 C280) 位描述	329
表 296.	USB EP DMA 状态寄存器 (USBEPDMASt—0x2008 C284) 位描述	330
表 297.	USB EP DMA 使能寄存器 (USBEPDMAEn—0x2008 C288) 位描述	330
表 298.	USB EP DMA 禁能寄存器 (USBEPDMADis—0x2008 C28C) 位描述	331
表 299.	USB DMA 中断状态寄存器 (USBDMAIntSt—0x2008 C290) 位描述	331
表 300.	USB DMA 中断使能寄存器 (USBDMAIntEn—0x2008 C294) 位描述	331
表 301.	USB 传输结束中断状态寄存器 (USBEoTIntSt—0x2008 C2A0) 位描述	332
表 302.	USB 传输结束中断清零寄存器 (USBEoTIntClr—0x2008 C2A4) 位描述	332
表 303.	USB 传输结束中断置位寄存器	

	(USBEoTIntSet—0x2008 C2A8) 位描述	332
表 304.	USB 新 DD 请求中断状态寄存器 (USBNDDRIntSt—0x2008 C2AC) 位描述	332
表 305.	USB 新 DD 请求中断清零寄存器 (USBNDDRIntClr—0x2008 C2B0) 位描述	333
表 306.	USB 新 DD 请求中断置位寄存器 (USBNDDRIntSet—0x2008 C2B4) 位描述	333
表 307.	USB 系统错误中断状态寄存器 (USBSysErrIntSt—0x2008 C2B8) 位描述	333
表 308.	USB 系统错误中断清零寄存器 (USBSysErrIntClr—0x2008 C2BC) 位描述	333
表 309.	USB 系统错误中断置位寄存器 (USBSysErrIntSet—0x2008 C2C0) 位描述	334
表 310.	SIE 命令代码表	338
表 311.	设置地址命令位描述	338
表 312.	配置设备命令位描述	339
表 313.	设置模式命令位描述	339
表 314.	设置设备状态命令位描述	340
表 315.	获得错误代码命令位描述	342
表 316.	读错误状态命令位描述	342
表 317.	选择端点命令位描述	343
表 318.	设置端点状态命令位描述	344
表 319.	清空缓冲区命令位描述	345
表 320.	DMA 描述符	351
表 321.	本章用到的与 USB (OHCI) 相关的首字母缩 写词/简写	364
表 322.	USB OTG 端口管脚	366
表 323.	USB 主机寄存器地址定义	367
表 324.	USB OTG 端口 1 的管脚	371
表 325.	USB OTG 和 I ² C 寄存器地址定义	376
表 326.	USB 中断状态寄存器 (USBIntSt—0x2008 C1C0) 位描述	377
表 327.	OTG 中断状态寄存器 (OTGIntSt—0x2008 C100) 位描述	377
表 328.	OTG 状态控制寄存器 (OTGStCtrl—0x2008 C110) 位描述	378
表 329.	OTG 定时器寄存器 (OTGTmr—0x2008 C114) 位描述	379
表 330.	OTG 时钟控制寄存器 (OTGClkCtrl—0x2008 CFF4) 位描述	379
表 331.	OTG 时钟状态寄存器 (OTGClkSt—0x2008 CFF8) 位描述	380
表 332.	I ² C 接收寄存器 (I2C_RX—address 0x2008 C300) 位描述	380
表 333.	I ² C 发送寄存器 (I2C_TX—0x2008 C300) 位描述	381
表 334.	I ² C 状态寄存器 (I2C_STS—0x2008 C304) 位描述	381
表 335.	I ² C 控制寄存器 (I2C_CTL—0x2008 C308) 位描述	382
表 336.	I ² C_CLKHI 寄存器 (I2C_CLKHI—address 0x2008 C30C) 位描述	383
表 337.	I ² C_CLKLO 寄存器 (I2C_CLKLO—address 0x2008 C310) 位描述	383
表 338.	SD/MMC 卡接口管脚描述	398
表 339.	命令格式	402
表 340.	简单响应格式	402
表 341.	长响应格式	403
表 342.	命令路径状态标志	403
表 343.	CRC 令牌状态	407
表 344.	数据路径状态标志	407
表 345.	发送 FIFO 状态标志	408
表 346.	接收 FIFO 状态标志	408
表 347.	卡接口寄存器汇总	410
表 348.	电源控制寄存器 (MCIPWR—0x400C 0000) 位描述	410
表 349.	时钟控制寄存器 (MCIClock—0x400C 0004) 位描述	411
表 350.	参数寄存器 (MCIArgument—0x400C 0008) 位描述	412
表 351.	命令寄存器 (MCICommand—0x400C 000C) 位描述	412
表 352.	命令响应类型	412
表 353.	命令响应寄存器 (MCIRspCommand—0x400C 0010) 位描 述	413
表 354.	响应寄存器 (MCIResponse0-3—0x400C 0014, 0x400C 0018, 0x400C 001C 和 0x400C 0020) 位描述	413
表 355.	响应寄存器类型	413
表 356.	数据定时器寄存器 (MCIDataTimer—0x400C 0024) 位描述	413
表 357.	数据长度寄存器 (MCIDataLength—0x400C 0028) 位描述	414
表 358.	数据控制寄存器 (MCIDataCtrl—0x400C 002C) 位描述	414
表 359.	数据块长度	415
表 360.	数据计数器寄存器 (MCIDataCnt—0x400C 0030) 位描述	415
表 361.	状态寄存器 (MCIStatus—0x400C 0034) 位	

描述.....	415	表 388. UART1 RS-485 延时值寄存器位描述 (U1RS485DLY—0x4001 0054)	441
表 362. 清零寄存器 (MCIClear—0x400C 0038) 位 描述.....	416	表 389. UARTn 管脚描述	446
表 363. 中断屏蔽寄存器 (MCIMask0—0x400C 003C) 位描述.....	416	表 390. UART0/2/3 寄存器映射	446
表 364. FIFO 计数器寄存器 (MCIFifoCnt—0x400C 0048) 位描述	417	表 391. UARTn 接收器缓冲寄存器 (U0RBR—0x4000 C000, U2RBR—0x4009 8000, U3RBR—0x4009 C000, DLAB = 0) 位描述	447
表 365. 数据 FIFO 寄存器 (MCIFIFO—0x400C 0080 至 0x400C 00BC) 位描述	417	表 392. UARTn 发送保持寄存器位描述 (U0THR—0x4000 C000, U2THR—0x4009 8000, U3THR—0x4009 C000, DLAB = 0)	448
表 366. UART1 管脚描述.....	421	表 393. UARTn 除数锁存器 LSB 寄存器位描述 (U0DLL—0x4000 C000, U2DLL—0x4009 8000, U3DLL—0x4009 C000, DLAB = 1)	448
表 367. UART1 寄存器映射	422	表 394. UARTn 除数锁存器 MSB 寄存器位描述 (U0DLM—0x4000 C004, U2DLL—0x4009 8004, U3DLL—0x4009 C004, DLAB = 1)	448
表 368. UART1 接收器缓冲寄存器位描述 (U1RBR— 0x4001 0000, DLAB = 0)	423	表 395. UARTn 中断使能寄存器位描述 (U0IER—0x4000 C004, U2IER—0x4009 8004, U3IER—0x4009 C004, DLAB = 0)	448
表 369. UART1 发送保持寄存器位描述 (U1THR— 0x4001 0000, DLAB = 0)	423	表 396. UARTn 中断标识寄存器位描述 (U0IIR —0x4000 C008, U2IIR—0x4009 8008, U3IIR—0x4009 C008)	449
表 370. UART1 除数锁存器 LSB 寄存器位描述 (U1DLL—0x4001 0000, DLAB = 1)	424	表 397. UARTn 中断处理	450
表 371. UART1 除数锁存器 MSB 寄存器位描述 (U1DLM—0x4001 0004, DLAB = 1)	424	表 398. UARTn FIFO 控制寄存器位描述 (U0FCR—0x4000 C008, U2FCR—0x4009 8008, U3FCR—0x4007 C008)	451
表 372. UART1 中断使能寄存器位描述 (U1IER— 0x4001 0004, DLAB = 0)	424	表 399. UARTn 线控制寄存器位描述 (U0LCR—0x4000 C00C, U2LCR—0x4009 800C, U3LCR—0x4009 C00C)	452
表 373. UART1 中断标识寄存器位描述 (U1IIR— 0x4001 0008)	425	表 400. UARTn 线状态寄存器位描述 (U0LSR—0x4000 C014, U2LSR—0x4009 8014, U3LSR—0x4009 C014)	452
表 374. UART1 中断处理.....	426	表 401. UARTn 高速缓存寄存器位描述 (U0SCR—0x4000 C01C, U2SCR—0x4009 801C, U3SCR—0x4009 C01C)	453
表 375. UART1 FIFO 控制寄存器位描述 (U1FCR— 0x4001 0008)	427	表 402. UARTn 自动波特率控制寄存器位描述 (U0ACR—0x4000 C020, U2ACR—0x4009 8020, U3ACR—0x4009 C020)	454
表 376. UART1 线控制寄存器位描述 (U1LCR—0x4001 000C)	428	表 403. UARTn 分数分频器寄存器位描述 (U0FDR—0x4000 C028, U2FDR—0x4009 8028, U3FDR—0x4009 C028)	457
表 377. UART1 Modem 控制寄存器位描述 (U1MCR—0x4001 0010)	429	表 404. 分数分频器设置查找表	459
表 378. Modem 状态中断产生	430	表 405. UARTn 发送使能寄存器位描述	
表 379. UART1 线状态寄存器位描述 (U1LSR—0x4001 0014)	431		
表 380. UART1 Modem 状态寄存器位描述 (U1MSR —0x4001 0018)	432		
表 381. UART1 高速缓存寄存器位描述 (U1SCR— 0x4001 0014)	433		
表 382. 自动波特率控制寄存器位描述 (U1ACR—0x4001 0020)	433		
表 383. UART1 分数分频器寄存器位描述 (U1FDR— 0x4001 0028)	437		
表 384. 分数分频器设置查找表.....	439		
表 385. UART1 发送使能寄存器位描述 (U1TER— 0x4001 0030)	440		
表 386. UART1 RS485 控制寄存器位描述 (U1RS485CTRL—0x4001 004C)	440		
表 387. UART1 RS-485 地址匹配寄存器位描述 (U1RS485ADRMATCH—0x4001 0050)	441		

	(U0TER—0x4000 C030, U2TER—0x4009 8030, U3TER—0x4009 C030)	460
表 406.	UARTn RS485 控制寄存器位描述 (U0RS485CTRL—0x4000 C04C, U2RS485CTRL—0x4009 804C, U3RS485CTRL—0x4009 C04C)	460
表 407.	UARTn RS-485 地址匹配寄存器位描述 (U0RS485ADRMATCH—0x4000 C050, U2RS485ADRMATCH—0x4009 8050, U3RS485ADRMATCH—0x4009 C050)	461
表 408.	UARTn RS-485 延时值寄存器位描述 (U0RS485DLY—0x4000 0054U2RS485DLY—0x4009 8054, U3RS485DLY—0x4009 C054)	461
表 409.	UART4 管脚描述	466
表 410.	UART4 寄存器映射	466
表 411.	UART4 接收器缓冲寄存器位描述 (U4RBR—0x400A 4000, DLAB = 0)	467
表 412.	UART4 发送保持寄存器位描述 (U4THR—0x400A 4000, DLAB = 0)	467
表 413.	UART1 除数锁存器 LSB 寄存器位描述 (U4DLL—0x400A 4000, DLAB = 1)	468
表 414.	UART4 除数锁存器 MSB 寄存器位描述 (U4DLL—0x400A 4004, DLAB = 1)	468
表 415.	UART4 中断使能寄存器位描述 (U4IER—0x400A 4004, DLAB = 0)	468
表 416.	UART4 中断标识寄存器位描述 (U4IIR—0x400A 4008)	469
表 417.	UART4 中断处理	470
表 418.	UART4 FIFO 控制寄存器位描述 (U4FCR—0x400A 4008)	470
表 419.	UART4 线控制寄存器位描述 (U4LCR—0x400A 400C)	471
表 420.	UART4 线状态寄存器位描述 (U4LSR—0x400A 4014)	472
表 421.	UART4 高速缓存寄存器位描述 (U4SCR—0x400A 401C)	473
表 422.	UART4 自动波特率控制寄存器位描述 (U4ACR—0x400A 4020)	473
表 423.	UART4 IrDA 控制寄存器位描述 (U4ICR—0x400A 4024)	476
表 424.	IrDA 脉冲宽度	477
表 425.	UART4 分数分频器寄存器位描述 (U4FDR—0x400A 4028)	477
表 426.	分数分频器设置查找表	480
表 427.	UART4 过采样寄存器位描述 (U4OSR—0x400A 402C)	481
表 428.	UART4 智能卡接口控制寄存器位描述 (U4SCICTRL—0x400A 4048)	481
表 429.	UART4 RS485 控制寄存器位描述 (U4RS485CTRL—0x400A 404C)	482
表 430.	UART4 RS-485 地址匹配寄存器位描述 (U4RS485ADRMATCH—0x400A 4050)	483
表 431.	UART4 RS-485 延时值寄存器位描述 (U4RS485DLY—0x400A 4054)	483
表 432.	UART4 同步模式控制寄存器位描述 (U4SYNCCTRL—0x400A 4058)	485
表 433.	UART4 发送使能寄存器位描述 (U4TER—0x400A 4030)	486
表 434.	CAN 管脚描述	488
表 435.	CAN 模块的内存映像	494
表 436.	CAN 验收滤波器和集中 CAN 寄存器	494
表 437.	CAN1 和 CAN2 控制器寄存器映射	494
表 438.	CAN1 和 CAN2 控制器寄存器概括	496
表 439.	CAN 唤醒和睡眠寄存器	496
表 440.	CAN 模式寄存器位描述 (CAN1MOD—地址 0x4004 4000, CAN2MOD—地址 0x4004 8000)	496
表 441.	CAN 命令寄存器位描述 (CAN1CMR—地址 0x4004 4004, CAN2CMR—地址 0x4004 8004)	498
表 442.	CAN 全局状态寄存器位描述 (CAN1GSR— 地址 0x4004 4008, CAN2GSR—地址 0x4004 8008)	500
表 443.	CAN 中断和捕获寄存器位描述 (CAN1ICR— 地址 0x4004 400C, CAN2ICR—地址 0x4004 800C)	502
表 444.	CAN 中断使能寄存器的位描述 (CAN1IER— 地址 0x4004 4010, CAN2IER—地址 0x4004 8010)	505
表 445.	CAN 总线时序寄存器的位描述 (CAN1BTR— 地址 0x4004 4014, CAN2BTR—地址 0x4004 8014)	506
表 446.	CAN 错误报警界限寄存器的位描述 (CAN1EWL—地址 0x4004 4018, CAN2EWL—地址 0x4004 8018)	507
表 447.	CAN 状态寄存器的位描述 (CAN1SR—地址 0x4004 401C, CAN2SR—地址 0x4004 801C)	508
表 448.	CAN 接收帧状态寄存器的位描述 (CAN1RFS—地址 0x4004 4020, CAN2RFS—地址 0x4004 8020)	510
表 449.	CAN 接收标识符寄存器的位描述 (CAN1RID —地址 0x4004 4024, CAN2RID—地址 0x4004 8024)	511
表 450.	FF=1 时的 RX 标识符寄存器	511
表 451.	CAN 接收数据寄存器 A 的位描述	

	(CAN1RDA—地址 0x4004 4028, CAN2RDA—地址 0x4004 8028)512				地址 0x4003 C018)525
表 452.	CAN 接收数据寄存器 B 的位描述 (CAN1RDB—地址 0x4004 402C, CAN2RDB—地址 0x4004 802C)512	表 472.	LUT 错误寄存器的位描述 (LUTerr—地址 0x4003 C01C)525	表 473.	全局 FullCAN 中断使能寄存器位描述 (FCANIE—地址 0x4003 C020)526
表 453.	CAN 发送帧信息寄存器位描述 (CAN1TFI[1/2/3]—地址 0x4004 40[30/40/50], CAN2TFI[1/2/3]—地址 0x400480[30/40/50])513	表 474.	FullCAN 中断和捕获寄存器 0 的位描述 (FCANIC0—地址 0x4003 C024)526	表 475.	FullCAN 中断和捕获寄存器 1 的位描述 (FCANIC1—地址 0x4003 C028)526
表 454.	CAN 发送标识符寄存器位描述 (CAN1TID[1/2/3]—地址 0x4004 40[34/44/54], CAN2TID[1/2/3]—地址 0x400480[34/44/54])514	表 476.	自动保存的 Rx 报文的格式529	表 477.	FullCAN 信号量操作529
表 455.	FF=1 时的发送标识符寄存器514	表 478.	验收滤波器表格和 ID 索引值的示例.....540	表 479.	使用的 ID 查找表区543
表 456.	CAN 发送数据寄存器 A 位描述 (CAN1TDA[1/2/3]—地址 0x4004 40[38/48/58], CAN2TDA[1/2/3]—地址 0x400480[38/48/58])515	表 480.	使用的 ID 查找表区545	表 481.	SSP 管脚描述549
表 457.	CAN 发送数据寄存器 B 位描述 (CAN1TDB[1/2/3]—地址 0x4004 40[3C/4C/5C], CAN2TDB[1/2/3]—地址 0x400480[3C/4C/5C])515	表 482.	SSP 寄存器映射557	表 483.	SSPn 控制寄存器 0 位描述 (SSP0CR0—地 址 0x4008 8000, SSP1CR0—地址 0x4003 0000, SSP2CR0—0x400A C000)558
表 458.	CAN 睡眠清零寄存器的位描述 (CANSLEEPCLR—地址 0x400F C110)515	表 484.	SSPn 控制寄存器 1 位描述 (SSP0CR1—地 址 0x4008 8004, SSP1CR1—地址 0x4003 0004, SSP2CR1—0x400A C004)559	表 485.	SSPn 数据寄存器位描述 (SSP0DR—地址 0x4008 8008, SSP1DR—0x4003 0008, SSP2DR—0x400A C008)559
表 459.	CAN 唤醒标志寄存器的位描述 (CANWAKEFLAGS—地址 0x400F C114)516	表 486.	SSPn 状态寄存器位描述 (SSP0SR—地址 0x4008 800C, SSP1SR—0x4003 000C, SSP2SR—0x400A C00C)560	表 487.	SSPn 时钟预分频寄存器位描述 (SSP0CPSR—地址 0x4008 8010, SSP1CPSR—0x4003 0010, SSP2CPSR—0x400A C010)560
表 460.	集中发送状态寄存器的位描述 (CANTxSR— 地址 0x4004 0000)518	表 488.	SSPn 中断使能置位/清零寄存器位描述 (SSP0IMSC—地址 0x4008 8014, SSP1IMSC—0x4003 0014, SSP2IMSC—0x400A C014)560	表 489.	SSPn 原始中断状态寄存器位描述 (SSP0RIS—地址 0x4008 8018, SSP1RIS—0x4003 0018, SSP2RIS—0x400A C018)561
表 461.	集中接收状态寄存器的位描述 (CANRxSR— 地址 0x4004 0004)518	表 490.	SSPn 使能中断状态寄存器位描述 (SSP0MIS—地址 0x4008 801C, SSP1MIS—0x4003 001C, SSP2MIS—0x400A C01C)561	表 491.	SSPn 中断清零寄存器位描述 (SSP0ICR— 地址 0x4008 8020, SSP1ICR—0x4003 0020, SSP2ICR—0x400A C020)562
表 462.	集中其它状态寄存器的位描述 (CANMSR— 地址 0x4004 0008)518	表 492.	SSPn DMA 控制寄存器位描述 (SSP0DMACR—地址 0x4008 8024,		
表 463.	验收滤波器模式和访问控制519				
表 464.	区配置寄存器设置520				
表 465.	验收滤波器模式寄存器的位描述 (AFMR—地 址 0x4003 C000)523				
表 466.	标准帧单个起始地址寄存器位描述 (SFF_sa—地址 0x4003 C004)523				
表 467.	标准帧组起始地址寄存器的位描述 (SFF_GRP_sa—地址 0x4003 C008)524				
表 468.	扩展帧起始地址寄存器的位描述 (EFF_sa— 地址 0x4003 C00C)524				
表 469.	扩展帧组起始地址寄存器的位描述 (EFF_GRP_sa—地址 0x4003 C010)524				
表 470.	AF 表结束寄存器的位描述 (ENDofTable—地 址 0x4003 C014)525				
表 471.	LUT 错误地址寄存器的位描述 (LUTerrAd—				

SSP1DMACR—0x4003 0024, SSP2DMACR—0x400A C024)562	
表 493. I ² C 管脚描述566	
表 494. 用于配置主机模式的 I2C0CONSET 和 I2C1CONSET567	
表 495. 用于配置从机模式的 I2C0CONSET 和 I2C1CONSET570	
表 496. I ² C 寄存器映射576	
表 497. I ² C 控制置位寄存器位描述(I2CONSET:I ² C0, I2C0CONSET—地址 0x4001 C000, I ² C1, I2C1CONSET—地址 0x4005 C000, I ² C2, I2C2CONSET—地址 0x400A 0000)577	
表 498. I ² C 控制清零寄存器位描述(I2CONCLR:I ² C0, I2C0CONCLR—0x4001 C018, I ² C1, I2C1CONCLR—0x4005 C018, I ² C2, I2C2CONCLR—0x400A 0018)579	
表 499. I ² C 状态寄存器 (I2STAT: I ² C0, I2C0STAT—0x4001 C004; I ² C1, I2C1STAT—0x4005 C004; I ² C2, I2C2STAT—0x400A 0004)579	
表 500. I ² C 数据寄存器位描述 (I2DAT: I ² C0, I2C0DAT—0x4001 C008; I ² C1, I2C1DAT—0x4005 C008; I ² C2, I2C2DAT—0x400A 0008)580	
表 501. I ² C 监控模式控制寄存器位描述 (I2MMCTRL: I ² C0, I2C0MMCTRL—0x4001 C01C; I ² C1, I2C1MMCTRL—0x4005 C01C; I ² C2, I2C2MMCTRL—0x400A 001C)580	
表 502. I ² C 数据缓冲寄存器位描述 (I2DATA_BUFFER: I ² C0, I2CDATA_BUFFER—0x4001 C02C; I ² C1, I2C1DATA_BUFFER—0x4005 C02C; I ² C2, I2C2DATA_BUFFER—0x400A 002C)581	
表 503. I ² C 从地址寄存器位描述 (I2ADR0-3: I ² C0, I2C0ADR[0, 1, 2, 3]—0x4001 C0[0C, 20, 24, 28]; I ² C1, I2C1ADR[0, 1, 2, 3]—地 址 0x4005 C0[0C, 20, 24, 28]; I ² C2, I2C2ADR[0, 1, 2, 3]—地址 0x400A 00[0C, 20, 24, 28])582	
表 504. I ² C 屏蔽寄存器位描述 (I2MASK0~3: I ² C0, I2C0MASK[0, 1, 2, 3]—0x4001 C0[30, 34, 38, 3C]; I ² C1, I2C1MASK[0, 1, 2, 3]—地址 0x4005 C0[30, 34, 38,3C]; I ² C2, I2C2MASK[0, 1, 2, 3]—地址 0x400A 00[30, 34, 38,3C])582	
表 505. I ² C SCL 高电平占空比寄存器位描述 (I2SCLH: I ² C0, I2C0SCLH—地址 0x4001 C010; I ² C1, I2C1SCLH—地址 0x4005 C010; I ² C2, I2C2SCLH—0x400A 0010)582	
表 506. I ² C SCL 低电平占空比寄存器位描述 (I2SCLL: I ² C0—I2C0SCLL: 0x4001 C014; I ² C1—I2C1SCLL: 0x4005 C014; I ² C2—I2C2SCLL: 0x400A 0014)583	
表 507. I ² C 时钟速率的实例583	
表 508. 描述 I ² C 操作的缩写584	
表 509. 用于初始化主发送器模式的 I2CONSET585	
表 510. 用于初始化从接收器模式的 I2CONSET589	
表 511. 主发送器模式592	
表 512. 主接收器模式593	
表 513. 从接收器模式594	
表 514. 从发送器模式595	
表 515. 其他状态596	
表 516. 管脚描述610	
表 517. I ² S 寄存器映射611	
表 518. 数字音频输出寄存器位描述 (I2SDAO—地址 0x400A 8000)611	
表 519. 数字音频输入寄存器位描述 (I2SDAI—地址 0x400A 8004)612	
表 520. 发送 FIFO 寄存器位描述 (I2STXFIFO—地址 0x400A 8008)612	
表 521. 接收 FIFO 寄存器位描述 (I2RXFIFO—地址 0x400A 800C)612	
表 522. 状态反馈寄存器位描述 (I2SSTATE—地址 0x400A 8010)612	
表 523. DMA 配置寄存器 1 位描述(I2SDMA1—地址 0x400A 8014)613	
表 524. DMA 配置寄存器 2 位描述(I2SDMA2—地址 0x400A 8018)613	
表 525. 中断请求控制寄存器位描述 (I2SIRQ—地址 0x400A 801C)614	
表 526. 发送时钟速率寄存器位描述 (I2TXRATE—地 址 0x400A 8020)614	
表 527. 接收时钟速率寄存器位描述 (I2SRXRATE— 地址 0x400A 8024)615	
表 528. 发送时钟位速率寄存器位描述 (I2TXBITRATE—地址 0x400A 8028)616	
表 529. 接收时钟位速率寄存器位描述 (I2SRXBITRATE—地址 0x400A 802C)616	
表 530. 发送模式控制寄存器位描述 (I2STXMODE—0x400A 8030)616	
表 531. 接收模式控制寄存器位描述 (I2SRXMODE—0x400A 8034)617	
表 532. I ² S 发送模式620	
表 533. I ² S 接收模式623	
表 534. FIFO 深度比较的条件627	
表 535. DMA 和中断请求产生627	
表 536. I2SSTATE 寄存器中的状态反馈627	

表 537. 定时器/计数器管脚描述.....	632	PWM1PCR—0x4001 804C)	657
表 538. 定时器/计数器 0~3 的寄存器映射	632	表 560. PWM 锁存使能寄存器位描述	
表 539. 中断寄存器位描述 (T[0/1/2/3]IR—地址		(PWM0LER—0x4001 4050 和	
0x4000 4000, 0x4000 8000, 0x4009 0000,		PWM1LER—0x4001 8050)	659
0x4009 4000)	634	表 561. 管脚汇总	660
表 540. 定时器控制寄存器位描述 (TCR, TIMERn:		表 562. 电机控制脉宽调制器 (MCPWM) 寄存器映射	
TnTCR—地址 0x4000 4004, 0x4000 8004,		663
0x4009 0004, 0x4009 4004)	634	表 563. MCPWM 控制寄存器读地址位描述	
表 541. 计数控制寄存器位描述 (T[0/1/2/3]CTCR—		(MCCON—0x400B 8000)	664
地址 0x4000 4070, 0x4000 8070, 0x4009		表 564. MCPWM 控制寄存器置位地址位描述	
0070, 0x4009 4070)	635	(MCCON_SET—0x400B 8004)	665
表 542. 匹配控制寄存器位描述 (T[0/1/2/3]MCR—地		表 565. MCPWM 控制寄存器清零地址位描述	
址 0x4000 4014, 0x4000 8014, 0x4009 0014,		(MCCON_CLR—0x400B 8008)	665
0x4009 4014)	636	表 566. MCPWM 捕获控制寄存器读地址位描述	
表 543. 捕获控制寄存器位描述 (T[0/1/2/3]CCR—地		(MCCAPCON—0x400B 800C)	666
址 0x4000 4028, 0x4000 8020, 0x4009 0028,		表 567. MCPWM 捕获控制寄存器置位地址位描述	
0x4009 4028)	637	(MCCAPCON_SET—0x400B 8010)	666
表 544. 外部匹配寄存器位描述 (T[0/1/2/3]EMR—地		表 568. MCPWM 捕获控制寄存器清零地址位描述	
址 0x4000 403C, 0x4000 803C, 0x4009		(MCCAPCON_CLR—地址 0x400B 8014)	
003C, 0x4009 403C)	638	667
表 545. 外部匹配控制	638	表 569. 电机控制 PWM 中断	667
表 546. 系统节拍定时器寄存器的映射	642	表 570. 中断源位分配表	667
表 547. 系统定时器控制和状态寄存器的位描述		表 571. MCPWM 中断使能寄存器读地址位描述	
(STCTRL—0xE000 E010)	642	(MCINTEN—0x400B 8050)	667
表 548. 系统定时器重载值寄存器的位描述		表 572. PWM 中断使能寄存器置位地址位描述	
(STRELOAD—0xE000 E014)	643	(MCINTEN_SET—地址 0x400B 8054)	
表 549. 系统定时器当前值寄存器的位描述		668
(STCURRE—0xE000 E018)	643	表 573. PWM 中断使能寄存器清零地址位描述	
表 550. 系统定时器校准值寄存器的位描述		(MCINTEN_CLR—地址 0x400B 8058)	
(STCALIB—0xE000 E01C)	644	668
表 551. PWM 触发器的置位和复位输入	650	表 574. MCPWM 中断标志寄存器读地址位描述	
表 552. 管脚汇总	651	(MCINTF—0x400B 8068)	668
表 553. PWM0 和 PWM1 寄存器映射	652	表 575. MCPWM 中断标志寄存器置位地址位描述	
表 554. PWM 中断寄存器位描述 (PWM0IR—0x4001		(PWMINTF_SET—0x400B 806C)	668
4000 和 PWM1IR—0x4001 8000)	653	表 576. MCPWM 中断标志寄存器清零地址位描述	
表 555. PWM 定时器控制寄存器位描述		(PWMINTF_CLR—0x400B 8070)	668
(PWM0TCR—0x4001 4004 和		表 577. MCPWM 计数控制寄存器读地址位描述	
PWM1TCR—0x4001 8004)	653	(MCCNTCON—0x400B 805C)	669
表 556. PWM 计数控制寄存器位描述		表 578. MCPWM 计数控制寄存器置位地址位描述	
(PWM0CTCR—0x4001 4070 和		(MCCNTCON_SET—0x400B 8060)	670
PWM1CTCR—0x4001 8070)	654	表 579. MCPWM 计数控制寄存器清零地址位描述	
表 557. 匹配控制寄存器位描述		(MCCAPCON_CLR—0x400B 8064)	670
(PWM0MCR—0x4001 4014 和		表 580. MCPWM 定时器/计数器 0~2 寄存器位描述	
PWM1MCR—0x4001 8014)	655	(MCTC0~2—0x400B 8018, 0x400B 801C,	
表 558. PWM 捕获控制寄存器位描述		0x400B 8020)	670
(PWM0CCR—0x4001 4028 和		表 581. MCPWM 界限 0~2 寄存器位描述	
PWM1CCR—0x4001 8028)	657	(MCLIM0~2—0x400B 8024, 0x400B 8028,	
表 559. PWM 控制寄存器位描述		0x400B 802C)	671
(PWMPER—0x4001 404C 和		表 582. MCPWM 匹配 0~2 寄存器位描述	

(MCMAT0~2—地址 0x400B 8030, 0x400B 8034, 0x400B 8038)	672	表 609. 用于相 A 的 QEI 数字滤波器位描述 (FILTERPHA—地址 0x400B C03C) ...	692
表 583. MCPWM 死区时间寄存器位描述 (MCDT— 地址 0x400B 803C)	673	表 610. 用于相 B 的 QEI 数字滤波器位描述 (FILTERPHB—地址 0x400B C040) ...	692
表 584. MCPWM 通信格式寄存器位描述 ((MCCP— 地址 0x400B 8040)	673	表 611. 用于 INX 的数字滤波器位描述 (FILTERINX— 地址 0x400B C044)	692
表 585. MCPWM 捕获寄存器读地址位描述 (MCCAP0/1/2—0x400B 8044, 0x400B 8048, 0x400B 804C)	674	表 612. QEI 索引验收窗口位描述 (WINDOW—地址 0x400B C048)	692
表 586. MCPWM 捕获寄存器清零地址位描述 (CAP_CLR—0x400B 8074)	674	表 613. QEI 中断状态寄存器位描述 (QEINSTAT— 地址 0x400B CFE0)	693
表 587. 编码器状态	683	表 614. QEI 中断设置寄存器位描述 (QEISET—地址 0x400B CFEC)	694
表 588. 编码器状态的转换 ^[1]	683	表 615. QEI 中断清除寄存器位描述 (QEICLR—0x400B CFE8)	695
表 589. 编码器方向	684	表 616. QEI 中断使能寄存器位描述 (QEIIE—地址 0x400B CFE4)	696
表 590. QEI 的管脚描述	686	表 617. QEI 中断使能置位寄存器位描述 (QEIIES— 地址 0x400B CFDC)	697
表 591. 寄存器汇总	687	表 618. QEI 中断使能清除寄存器位描述 (QEIEC— 地址 0x400B CFD8)	698
表 592. QEI 控制寄存器位描述 (QEICON—地址 0x400B C000)	688	表 619. RTC 管脚描述	701
表 593. QEI 配置寄存器位描述 (QEICONF—地址 0x400B C008)	688	表 620. 实时时钟寄存器映射	702
表 594. QEI 状态寄存器位描述 (QEISTAT—地址 0x400B C004)	689	表 621. 中断位置寄存器位描述 (ILR—地址 0x4002 4000)	703
表 595. QEI 位置寄存器位描述 (QEIP0S—地址 0x400B C00C)	689	表 622. 时钟控制寄存器位描述 (CCR—地址 0x4002 4008)	703
表 596. QEI 最大位置值寄存器位描述 (QEIMAXPOS—地址 0x400B C010)	689	表 623. 计数器增量中断寄存器位描述 (CIIR—地址 0x4002 400C)	704
表 597. QEI 位置比较寄存器 0 位描述 (CMPOS0— 地址 0x400B C014)	689	表 624. 报警屏蔽寄存器位描述 (AMR—地址 0x4002 4010)	705
表 598. QEI 位置比较寄存器 1 位描述 (CMPOS1— 地址 0x400B C018)	690	表 625. RTC 辅助控制寄存器 (RTC_AUX—地址 0x4002 405C)	705
表 599. QEI 位置比较寄存器 2 位描述 (CMPOS2 地 址 0x400B C01C)	690	表 626. RTC 辅助使能寄存器位描述 (RTC_AUXEN—地址 0x4002 4058)	705
表 600. QEI 索引计数寄存器位描述 (INXCNT—地址 0x400B C020)	690	表 627. 完整时间寄存器 0 位描述 (CTIME0—地址 0x4002 4014)	706
表 601. QEI 索引比较寄存器 0 位描述 (INXCMP0— 地址 0x400B C024)	690	表 628. 完整时间寄存器 1 位描述 (CTIME1—地址 0x4002 4018)	706
表 602. QEI 索引比较寄存器 1 位描述 (INXCMP1— 地址 0x400B C04C)	690	表 629. 完整时间寄存器 2 位描述 (CTIME2—地址 0x4002 401C)	706
表 603. QEI 索引比较寄存器 2 位描述 (INXCMP2— 地址 0x400B C050)	691	表 630. 时间计数器的关系和值	707
表 604. QEI 速度定时器重载寄存器位描述 (QEILOAD—地址 0x400B C028)	691	表 631. 时间计数器寄存器	707
表 605. QEI 定时器寄存器位描述 (QEITIME—地址 0x400B C02C)	691	表 632. 校准寄存器位描述 (CALIBRATION—地址 0x4002 4040)	708
表 606. QEI 速度寄存器位描述 (QEIVEL—地址 0x400B C030)	691	表 633. 通用寄存器 0~4 位描述 (GPREG0~GPREG4—地址 0x4002 4044~0x4002 4054)	709
表 607. QEI 速度捕获寄存器位描述 (QEICAP—地址 0x400B C034)	691	表 634. 报警寄存器	709
表 608. QEI 速度比较寄存器位描述 (VELCOMP—地 址 0x400B C038)	692		

表 635. 事件监测器/记录器管脚描述	713	表 661. D/A 控制寄存器位描述 (DACCTRL—地址 0x4008 C004)	739
表 636. 事件监测器/记录器寄存器映射	713	表 662. D/A 转换器计数器值寄存器位描述 (DACNTVAL—地址 0x4008 C008) ..	739
表 637. 事件监测器/记录器控制寄存器位描述 (ERCONTROL—0x4002 4084)	714	表 663. 字节顺序特性	744
表 638. 事件监测器/记录器状态寄存器位描述 (ERSTATUS—0x4002 4080)	716	表 664. DMA 连接	747
表 639. 事件监测器/记录器计数器寄存器位描述 (ERCOUNTERS—0x4002 4088)	717	表 665. GPDMA 寄存器映射	748
表 640. 事件监测器/记录器首个时间戳寄存器位描述 (ERFIRSTSTAMP0—0x0x4002 4090, ERFIRSTSTAMP1—0x0x4002 4094, ERFIRSTSTAMP2—0x4002 4098)	717	表 666. DMA 中断状态寄存器 (DMACIntStat—0x2008 0000)	750
表 641. 事件监测器/记录器末尾时间戳寄存器位描述 (ERLASTSTAMP0—0x0x4002 40A0, ERLASTSTAMP1—0x0x4002 40A4, ERLASTSTAMP2—0x4002 40A8)	718	表 667. DMA 中断终端计数请求状态寄存器 (DMACIntTCStat—0x2008 0004)	750
表 642. 看门狗寄存器映射	722	表 668. DMA 中断终端计数请求清除寄存器 (DMACIntTCClear—0x2008 0008)	750
表 643. 看门狗模式寄存器位描述 (WDMOD—0x4000 0000)	722	表 669. DMA 中断错误状态寄存器 (DMACIntErrStat—0x2008 000C)	751
表 644. 看门狗工作模式选择	723	表 670. DMA 中断错误清除寄存器 (DMACIntErrClr—0x2008 0010)	751
表 645. 看门狗定时器常数寄存器位描述 (WDTC—0x4000 0004)	723	表 671. DMA 原始中断终端计数状态寄存器 (DMACRawIntTCStat—0x2008 0014)	751
表 646. 看门狗喂狗寄存器位描述 (WDFEED—0x4000 0008)	724	表 672. DMA 原始错误中断状态寄存器 (DMACRawIntErrStat—0x2008 0018)	752
表 647. 看门狗定时器值寄存器位描述 (WDTV—0x4000 000C)	724	表 673. DMA 使能通道寄存器 (DMACEnbldChns—0x2008 001C) ...	752
表 648. 看门狗定时器报警中断寄存器位描述 (WDWINDOW—0x4000 0014)	724	表 674. DMA 软件突发请求寄存器 (DMACSoftBReq—0x2008 0020)	752
表 649. 看门狗定时器窗口寄存器位描述 (WDWINDOW—0x4000 0018)	724	表 675. DMA 软件单次请求寄存器 (DMACSoftSReq—0x2008 0024)	753
表 650. ADC 管脚描述	728	表 676. DMA 软件最后一个突发请求寄存器 (DMACSoftLBReq—0x2008 0028)	753
表 651. ADC 寄存器	729	表 677. DMA 软件最后一个单次请求寄存器 (DMACSoftLSReq—0x2008 002C) ...	753
表 652. A/D 控制寄存器位描述 (AD0CR—地址 0x4003 4000)	730	表 678. DMA 配置寄存器 (DMACConfig—0x2008 0030)	754
表 653. A/D 全局数据寄存器位描述 (AD0GDR—地址 0x4003 4004)	731	表 679. DMA 同步寄存器 (DMACSync—0x2008 0034)	754
表 654. A/D 中断使能寄存器位描述 (AD0INTEN—地址 0x4003 400C)	732	表 680. DMA 请求选择寄存器 (DMACReqSel—0x400F C1C4)	755
表 655. A/D 数据寄存器位描述 (AD0DR0~AD0DR7—0x4003 4010~0x4003 402C)	733	表 681. DMA 通道源地址寄存器 (DMACCxSrcAddr—0x2008 01x0)	756
表 656. A/D 状态寄存器位描述 (AD0STAT—地址 0x4003 4030)	734	表 682. DMA 通道目标地址寄存器 (DMACCxDestAddr—0x2008 01x4) ..	757
表 657. A/D 调节寄存器位描述 (ADTRM—地址 0x4003 4034)	734	表 683. DMA 通道链表项寄存器 (DMACCxLLI—0x2008 01x8)	757
表 658. D/A 管脚描述	737	表 684. DMA 通道控制寄存器 (DMACCxControl—0x2008 01xC)	758
表 659. DAC 寄存器	738	表 685. DMA 通道配置寄存器 (DMACCxConfig—0x2008 01x0)	760
表 660. D/A 转换器寄存器位描述 (DACR—地址 0x4008 C000)	738		

表 686. 传输类型位	761	表 715. ISP 写 RAM 命令	795
表 687. DMA 请求信号的使用	764	表 716. ISP 读存储器命令	796
表 688. 寄存器概述: CRC 引擎	772	表 717. ISP 准备写操作的扇区命令	796
表 689. CRC 模式寄存器位描述 (CRC_MODE—0x2009 0000)	772	表 718. ISP 复制命令	797
表 690. CRC 种子寄存器位描述 (CRC_SEED—0x2009 0004)	772	表 719. ISP 运行命令	797
表 691. CRC 校验和寄存器位描述 (CRC_SUM—0x2009 0008)	772	表 720. ISP 擦除扇区命令	798
表 692. CRC 数据寄存器位描述 (CRC_DATA—0x2009 0008)	773	表 721. ISP 扇区查空命令	798
表 693. 寄存器概述	775	表 722. ISP 读器件 ID 命令	798
表 694. EEPROM 命令寄存器位描述 (EECMD—地 址 0x0020 0080)	775	表 723. LPC178x/177x 系列微控制器器件标识号	798
表 695. EEPROM 地址寄存器位描述 (EEADDR—地 址 0x0020 0084)	776	表 724. ISP 读取引导代码版本号命令	799
表 696. EEPROM 写数据寄存器位描述 (EEWDATA—地址 0x0020 0088)	776	表 725. ISP 读器件序列号命令	799
表 697. EEPROM 读数据寄存器位描述 (EERDATA—地址 0x0020 008C)	777	表 726. ISP 比较命令	799
表 698. EEPROM 等待状态寄存器位描述 (EEESTATE—地址 0x0020 0090)	777	表 727. ISP 返回代码汇总	800
表 699. EEPROM 时钟分频器寄存器位描述 (EECLKDIV—地址 0x0020 0094)	778	表 728. IAP 命令汇总	802
表 700. EEPROM 掉电/DCM 寄存器位描述 (EEPWRDWN—地址 0x0020 0098)	778	表 729. IAP 准备写操作扇区命令	803
表 701. 中断使能寄存器位描述 (EEINTEN—地址 0x0020 0FE4)	779	表 730. IAP 复制 RAM 内容到 Flash 命令	803
表 702. 中断使能清零寄存器位描述 (EEINTENCLR—地址 0x0020 0FD8)	779	表 731. IAP 擦除扇区命令	804
表 703. 中断使能置位寄存器位描述 (EEINTENSET—地址 0x0020 0FDC)	780	表 732. IAP 扇区查空命令	804
表 704. 中断状态寄存器位描述 (EEINTSTAT—地址 0x0020 0FE0)	780	表 733. IAP 读器件标识号命令	804
表 705. 中断状态清零寄存器位描述 (EEINTSTATCLR—地址 0x0020 0FE8)	781	表 734. IAP 读取引导代码版本号命令	805
表 706. 中断状态置位寄存器 (EEINTSTATSET—地 址 0x0020 0FEC)	781	表 735. IAP 读设备序列号命令	805
表 707. LPC178x/177x 系列微控制器设备中的扇区	791	表 736. IAP 比较命令	805
表 708. 代码读保护选项	792	表 737. 重新调用 ISP 命令	806
表 709. 代码读保护硬件/软件的相互作用	792	表 738. IAP 状态代码汇总	806
表 710. ISP 命令汇总	793	表 739. 寄存器概述: FMC (基地址 0x0020 0000)	807
表 711. ISP 解锁命令	793	表 740. Flash 模块符号差起始寄存器位描述 (FMSSTART—0x0020 0020)	808
表 712. ISP 设置波特率命令	794	表 741. Flash 模块符号差结束寄存器位描述 (FMSSTOP—0x0020 0024)	808
表 713. 支持的 ISP 波特率和 PCLK 频率示例(单位: MHz)	794	表 742. FMSW0 寄存器位描述 (FMSW0, 地址: 0x0020 002C)	809
表 714. ISP 回应命令	795	表 743. FMSW1 寄存器位描述 (FMSW1, 地址: 0x0020 0030)	809
		表 744. FMSW2 寄存器位描述 (FMSW2, 地址: 0x0020 0034)	809
		表 745. FMSW3 寄存器位描述 (FMSW3, 地址: 0x0020 0038)	809
		表 746. Flash 模块状态寄存器位描述 (FMSTAT—0x0020 0FE0)	809
		表 747. Flash 模块状态清除寄存器位描述 (FMSTATCLR—0x0020 0FE8)	809
		表 748. JTAG 管脚描述	812
		表 749. 串行调试管脚描述	812
		表 750. 并行跟踪管脚描述	812
		表 751. 存储器映射控制寄存器位描述 (MEMMAP—0x400F C040)	815
		表 752. Cortex-M3 指令	819
		表 753. 用来生成一些 Cortex-M3 指令的 CMSIS 内在	

函数.....	822	表 801. SCR 位分配	940
表 754. 用来访问专用寄存器的 CMSIS 内在函数	822	表 802. CCR 位分配	941
表 755. 条件代码后缀	829	表 803. 系统故障处理程序优先级域	942
表 756. 内存访问指令	831	表 804. SHPR1 寄存器的位分配	942
表 757. 偏移量范围	834	表 805. SHPR2 寄存器的位分配	942
表 758. 偏移量范围	840	表 806. SHPR3 寄存器的位分配	942
表 759. 数据处理指令	848	表 807. SHCSR 位分配	943
表 760. 乘法和除法指令	863	表 808. MMFSR 位分配	944
表 761. 组合和分离指令	871	表 809. BFSR 位分配	945
表 762. 跳转和控制指令	876	表 810. UFSR 位分配	947
表 763. 跳转范围	877	表 811. HFSR 位分配	948
表 764. 其他指令	884	表 812. MMFAR 位分配	948
表 765. 处理器模式、执行特权级别和堆栈使用选择汇 总	898	表 813. BFAR 位分配	949
表 766. 内核寄存器集汇总	898	表 814. 系统定时器寄存器汇总	950
表 767. PSR 寄存器组合	901	表 815. SysTick 控制寄存器的位分配	950
表 768. APSR 位分配	901	表 816. 加载寄存器的位分配	951
表 769. IPSR 位分配	902	表 817. VAL 寄存器的位分配	951
表 770. EPSR 位分配	902	表 818. CALIB 寄存器的位分配	951
表 771. PRIMASK 寄存器的位分配	903	表 819. 存储器属性汇总	953
表 772. FAULTMASK 寄存器的位分配	903	表 820. MPU 寄存器汇总	954
表 773. BASEPRI 寄存器的位分配	904	表 821. 类型寄存器的位分配	954
表 774. 控制寄存器的位分配	904	表 822. MPU 控制寄存器的位分配	956
表 775. 存储器访问行为	908	表 823. RNR 位分配	957
表 776. SRAM 存储器位段区	909	表 824. RBAR 位分配	958
表 777. 外设存储器位段区	910	表 825. RASR 位分配	959
表 778. 独占访问指令的 C 编译器内在函数	913	表 826. SIZE 域值示例	959
表 779. 不同异常类型的属性	915	表 827. TEX、C、B 和 S 的编码	960
表 780. 异常返回行为	920	表 828. 存储器属性编码的缓存策略	960
表 781. 故障	921	表 829. AP 编码	961
表 782. 故障状态和故障地址寄存器	922	表 830. 微控制器的存储区属性	964
表 783. 内核外设寄存器区	926	表 831. 缩写	969
表 784. NVIC 寄存器汇总	927		
表 785. 中断到中断变量的映射	927		
表 786. ISER 位分配	928		
表 787. ICER 位分配	928		
表 788. ISPR 位分配	929		
表 789. ICPR 位分配	929		
表 790. IABR 位分配	930		
表 791. IPR 位分配	930		
表 792. STIR 位分配	931		
表 793. 用于 NVIC 控制的 CMSIS 函数	933		
表 794. 系统控制模块寄存器汇总	934		
表 795. ACTLR 位分配	935		
表 796. CPUID 寄存器的位分配	936		
表 797. ICSR 位分配	936		
表 798. VTOR 位分配	938		
表 799. AIRCR 位分配	939		
表 800. 优先级分组	939		

40.4 图目录

图 1.	LPC178x/177x 简化方框图	11	图 41.	中断事件处理	336
图 2.	LPC178x/177x 方框图, CPU 和总线	14	图 42.	UDCA Head 寄存器和 DMA 描述符	349
图 3.	LPC 1788 系统存储器映射	17	图 43.	同步 OUT 端点操作举例	357
图 4.	复位模块方框图 (包括唤醒定时器)	24	图 44.	ATLE 模式中的数据传输	358
图 5.	复位后启动示例	25	图 45.	USB 主机控制器方框图	365
图 6.	外部中断逻辑图	30	图 46.	USB OTG 控制器方框图	370
图 7.	LPC178x/177x 时钟生成	38	图 47.	USB OTG 端口配置: 端口 U1 为 OTG 双角 色设备, 端口 U2 为主机	372
图 8.	振荡器模式和模型: (a) 从属模式, (b) 振 荡模式, (c) 外部晶振模型 (用来评估 $C_{x1/x2}$ 的值)	41	图 48.	USB OTG 端口配置: VP_VM 模式	373
图 9.	PLL1 模块方框图	45	图 49.	USB 主机端口配置: 端口 U1、U2 作为主机	374
图 10.	CLKOUT 选择	65	图 50.	USB 设备端口配置: 端口 U1 为主机和端口 U2 为设备	375
图 11.	Flash 加速器简化模块方框图 (显示潜在总线 连接)	67	图 51.	PORT_FUNC bit 0 = 0 和 PORT_FUNC bit 1 = 0 端口选择	379
图 12.	I/O 配置	110	图 52.	USB OTG 中断处理	384
图 13.	EMC 方框图	157	图 53.	有软件栈的 USB OTG 控制器	385
图 14.	EMC 可编程延迟	182	图 54.	B-设备从外设状态切换到主机状态的硬件支 持	387
图 15.	EMC 延迟校验	183	图 55.	B-设备从外设状态切换到主机状态时软件的 状态转变	388
图 16.	32 位存储器组的外部存储器接口 (bits MW=10)	184	图 56.	A-设备从主机状态切换到外设状态的硬件支 持	390
图 17.	位存储器组的外部存储器接口 (bits MW = 01)	186	图 57.	A-设备从主机状态切换到外设状态时软件的 状态转变	391
图 18.	位存储器组的外部存储器接口 (bits MW = 00)	186	图 58.	时钟和功率的控制	394
图 19.	典型存储器配置图	187	图 59.	SD 存储卡连接	398
图 20.	以太网方框图	190	图 60.	多媒体卡系统	399
图 21.	以太网包字段	192	图 61.	SD 卡接口	400
图 22.	接收描述符在存储器中的规划	217	图 62.	命令路径状态机	401
图 23.	发送描述符在存储器中的规划	220	图 63.	命令传输	402
图 24.	发送示例的存储器和寄存器	230	图 64.	数据路径状态机	404
图 25.	接收示例的存储器和寄存器	235	图 65.	挂起命令启动	406
图 26.	发送流控制	240	图 66.	UART1 框图	420
图 27.	接收滤波器方框图	242	图 67.	Auto-RTS 功能时序	430
图 28.	接收有效/无效的状态机	246	图 68.	Auto-CTS 功能时序	431
图 29.	发送有效/无效的状态机	247	图 69.	自动波特率 a) 模式 0 和 b) 模式 1 波形图	436
图 30.	LCD 控制器方框图	262	图 70.	设置 UART 分频器的算法	438
图 31.	光标移动	270	图 71.	UART0、2、3 框图	445
图 32.	光标剪裁	271	图 72.	自动波特率 a) 模式 0 和 b) 模式 1 波形图	456
图 33.	光标图像格式	272	图 73.	设置 UART 分频器的算法	458
图 34.	上电和掉电时序	278	图 74.	UART4 框图	465
图 35.	STN 显示器的水平时序	296	图 75.	Auto-baud a) 模式 0 和 b) 模式 1 波形图	476
图 36.	STN 显示器的垂直时序	297	图 76.	设置 UART 分频器的算法	479
图 37.	TFT 显示器的水平时序	298	图 77.	CAN 控制器模块框图	489
图 38.	TFT 显示器的垂直时序	299			
图 39.	USB 设备控制器方框图	306			
图 40.	USB MaxPacketSize 寄存器数组索引	324			

图 78.	标准和扩展帧格式配置的发送缓冲器的分布	490
图 79.	标准和扩展帧格式配置的接收缓冲器的分布	491
图 80.	全局自测试（以高速 CAN 总线为例）	492
图 81.	本地自测试（以高速 CAN 总线为例）	493
图 82.	一行 FullCAN 模式下单个标准标识符表格	521
图 83.	一行标准标识符范围表格	521
图 84.	一行扩展标识符表格	521
图 85.	说明查找算法的 ID 查找表示例	527
图 86.	读取一个自动存储报文的信号量流程	530
图 87.	ID 查找表中 FullCAN 区的示例	532
图 88.	FullCAN 报文对象分布图	532
图 89.	正常情况（无报文丢失）	534
图 90.	报文丢失	535
图 91.	报文被改写	536
图 92.	报文的改写由信号量位和报文丢失来指示	537
图 93.	报文的改写由报文丢失来指示	538
图 94.	清零报文丢失位	539
图 95.	验收滤波器表格和 ID 索引值的详细示例	542
图 96.	ID 查找表配置示例（无 FullCAN）	544
图 97.	ID 查找表配置示例（FullCAN 被激活并启用）	546
图 98.	TI 同步串行数据帧的格式：a）单帧传输 b）连续/顺序 2 帧传输	550
图 99.	CPOL=0 和 CPHA=0 时的 SPI 帧格式（a）单帧传输和 b）连续帧传输	551
图 100.	CPOL=0 和 CPHA=1 时的 SPI 帧格式	552
图 101.	CPOL=1 和 CPHA=0 时的 SPI 帧格式（a）单帧传输（b）连续帧传输	553
图 102.	CPOL=1 和 CPHA=1 时的 SPI 帧格式	554
图 103.	Microwire 帧格式（单帧传输）	555
图 104.	Microwire 帧格式（连续传输）	556
图 105.	Microwire 帧格式的建立和保持时间	556
图 106.	I ² C 总线配置	565
图 107.	主发送器模式的格式	568
图 108.	主接收器模式的格式	569
图 109.	在发送重复的起始条件后，主接收器切换为主发送器	569
图 110.	从接收器模式的格式	570
图 111.	从发送器模式的格式	571
图 112.	I ² C 串行接口模块框图	572
图 113.	仲裁过程	574
图 114.	串行时钟同步	574
图 115.	主发送器模式中的格式和状态	586
图 116.	主接收器模式中的格式和状态	588
图 117.	从接收器模式中的格式和状态	590
图 118.	从发送器模式中的格式和状态	591
图 119.	两个主机同时发送重复起始条件	598
图 120.	强制访问忙碌的 I ² C 总线	598
图 121.	从总线干扰中恢复（由 SDA 低电平引起的）	599
图 122.	简单的 I ² S 配置和总线时序	610
图 123.	典型的发送主机模式（有或没有 MCLK 输出）	621
图 124.	发送主机模式共用接收器的参考时钟	621
图 125.	4 线发送主机模式共用接收器的位时钟和 WS	621
图 126.	典型的发送从机模式	621
图 127.	发送从机模式共用接收器的参考时钟	622
图 128.	4 线发送从机模式共用接收器的位时钟和 WS	622
图 129.	典型的接收主机模式（有或没有 MCLK 输出）	624
图 130.	接收主机模式共用发送器的参考时钟	624
图 131.	4 线接收主机模式共用发送器位时钟和 WS	624
图 132.	典型的接收从机模式	624
图 133.	接收从机模式共用发送器参考时钟	625
图 134.	4 线接收从机模式共用发送器位时钟和 WS	625
图 135.	I ² S 时钟和管脚连接	626
图 136.	不同 I ² S 模式的 FIFO	628
图 137.	定时器模块框图	631
图 138.	定时器周期设置为 PR=2, MRx=6, 匹配时使能中断和复位	639
图 139.	定时器周期设置为 PR=2, MRx=6, 匹配时使能中断和停止	640
图 140.	系统节拍定时器框图	642
图 141.	PWM 模块框图	649
图 142.	PWM 波形样例	650
图 143.	MCPWM 模块框图	661
图 144.	不带死区时间的沿对齐 PWM 的波形（POLA=0）	675
图 145.	不带死区时间的中心对齐 PWM 的波形（POLA=0）	675
图 146.	带死区时间的沿对齐 PWM 的波形（POLA=0）	676
图 147.	带死区时间的中心对齐的波形（POLA=0）	676
图 148.	三相 DC 模式的示例波形	679
图 149.	三相 AC 模式的示例波形（沿对齐 PWM 模式）	680
图 150.	编码器接口方框图	682
图 151.	正交编码器基本操作	684
图 152.	RTC 域概念图	700
图 153.	RTC 功能方框图	700

图 154. 事件监控器/记录器框图.....712

图 155. 看门狗定时器框图721

图 156. 窗口模式使能状态下的早期喂狗725

图 157. 窗口模式使能状态下的正确喂狗725

图 158. 看门狗报警中断726

图 159. ADC 框图728

图 160. 通过 DMA 中断和定时器实现 DAC 控制 .737

图 161. DMA 控制器框图742

图 162. LLI 示例.....768

图 163. CRC 框图771

图 164. 启动一个写操作782

图 165. （16 位）后递增地址写操作783

图 166. 编程一个页寄存器的内容到存储器784

图 167. 启动一个读操作（从地址 A 的 32 位读操作）
.....784

图 168. 低存储器影射787

图 169. Boot 过程流程图790

图 170. IAP 参数传递.....802

图 171. 生成一个 128-位符号差的算法.....810

图 172. ARM 标准 JTAG 连接器813

图 173. Cortex 调试连接器814

图 174. Cortex 调试&ETM 连接器814

图 175. Cortex-M3 的典型应用816

图 176. ASR #3825

图 177. LSR #3.....826

图 178. LSL#3826

图 179. ROR#3.....827

图 180. RRX827

图 181. 位段映射911

图 182. 向量表916

40.5 目录

第 1 章： 概述	3
1.1 简介	3
1.2 特性	4
1.3 应用	8
1.4 订购信息	9
1.4.1 器件选项汇总	10
1.5 简化方框图	11
1.6 结构概述	12
1.7 ARM Cortex-M3 处理器	12
1.7.1 Cortex-M3 配置选项	12
1.8 片上 Flash 存储器系统	13
1.9 片上静态 RAM	13
1.10 片上 EEPROM	13
1.11 详细方框图	14
第 2 章： LPC178x/7x 存储器映射	15
2.1 存储器映射与外设寻址	15
2.2 存储器映射	16
2.3 片上外设	18
2.3.1 AHB 外设	18
2.3.2 APB 外设地址	18
2.4 存储器重新映射	20
2.5 AHB 仲裁	20
2.5.1 矩阵仲裁寄存器（Matrix_Arb—0x400F C188）	20
第 3 章： LPC178x/7x 系统控制	22
3.1 简介	22
3.2 管脚描述	22
3.3 寄存器描述	23
3.4 芯片复位	24
3.4.1 复位源标识寄存器（RSID—0x400F C180）	26
3.5 外设复位控制	27
3.5.1 复位控制寄存器 0（RSTCON0—0x400F C1CC）	27
3.5.2 复位控制寄存器 1（RSTCON1—0x400F C1D0）	28
3.6 掉电检测	29

3.7	外部中断输入	30
3.7.1	寄存器描述	31
3.7.2	外部中断标志寄存器 (EXTINT—0x400F C140)	31
3.7.3	外部中断模式寄存器 (EXTMODE—0x400F C148)	32
3.7.4	外部中断极性寄存器 (EXTPOLAR—0x400F C14C)	34
3.8	其它系统控制与状态标志	35
3.8.1	系统控制与状态寄存器 (SCS—0x400F C1A0)	35
第 4 章：	LPC178x/7x 的计时与功率控制.....	37
4.1	计时与功率控制功能汇总	37
4.2	寄存器描述	39
4.3	振荡器	40
4.3.1	内部 RC 振荡器	40
4.3.2	主振荡器	40
4.3.2.1	主振荡器的起振	42
4.3.3	RTC 振荡器	42
4.3.4	看门狗振荡器	42
4.4	时钟源选择多路复用.....	43
4.4.1	时钟源选择寄存器 (CLKSRCSEL—0x400F C10C)	43
4.5	PLL0 与 PLL1 (锁相环)	44
4.5.1	PLL 与启动/引导代码的相互关系.....	44
4.5.2	PLL 寄存器描述	45
4.5.3	PLL 控制寄存器 (PLL0CON—0x400F C080, PLL1CON—0x400F C0A0)	46
4.5.4	PLL 配置寄存器 (PLL0CFG—0x400F C084, PLL1CFG—0x400F C0A4)	46
4.5.5	PLL 状态寄存器 (PLL0STAT—0x400F C088, PLL1STAT—0x400F C0A8)	46
4.5.6	PLL 中断: PLOCK0 与 PLOCK1	47
4.5.7	PLL 馈送寄存器 (PLL0FEED—0x400F C08C, PLL1FEED—0x400F C0AC)	47
4.5.8	PLL 与掉电模式	48
4.5.9	PLL 频率计算.....	49
4.5.10	确定 PLL 设置的步骤.....	49
4.5.11	PLL 配置序列.....	51
4.5.12	PLL 配置示例.....	51
4.6	时钟选择与分频.....	53
4.6.1	CPU 时钟选择寄存器 (CCLKSEL—0x400F C104)	53
4.6.2	USB 时钟选择寄存器 (USBCLKSEL—0x400F C108)	54
4.6.3	EMC 时钟选择寄存器 (EMCCLKSEL—0x400F C100)	54
4.6.4	外设时钟选择寄存器 (PCLKSEL—0x400F C1A8)	55
4.7	功率控制.....	56
4.7.1	睡眠模式	56
4.7.2	深度睡眠模式.....	56
4.7.3	掉电模式	57
4.7.4	深度掉电模式.....	57
4.7.5	外设功率控制.....	58
4.7.6	功率提升	58
4.7.7	寄存器描述	58

4.7.8	功率模式控制寄存器 (PCON—0x400F C0C0)	59
4.7.8.1	低功耗模式的编码	59
4.7.9	从低功耗模式中唤醒	60
4.7.10	外设功率控制寄存器 (PCONP—0x400F C0C4)	60
4.7.11	功率控制注意事项	62
4.7.12	功率提升控制寄存器 (PBOOST—0x400F C1B0)	62
4.7.12.1	低功耗模式的编码	63
4.7.13	电源域	63
4.8	唤醒定时器	64
4.9	外部时钟输出管脚	65
4.9.1	时钟输出配置寄存器 CLKOUTCFG—0x400F C1C8)	65
第 5 章：LPC178x/7x Flash 加速器		67
5.1	简介	67
5.2	Flash 加速器模块	67
5.2.1	Flash 存储器组	67
5.2.2	Flash 编程问题	68
5.3	寄存器描述	68
5.4	Flash 加速器配置寄存器 (FLASHCFG—0x400F C000)	69
5.5	运行	69
第 6 章：LPC178x/7x 可嵌套向量中断控制器		71
6.1	特性	71
6.2	描述	71
6.3	中断源	71
6.4	向量表的重新映射	74
6.5	寄存器描述	75
6.5.1	中断使能设置寄存器 0 (ISER0—0xE000 E100)	76
6.5.2	中断使能设置寄存器 1 (ISER1—0xE000 E104)	77
6.5.3	中断使能清除寄存器 0 (ICER0—0xE000 E180)	78
6.5.4	中断使能清除寄存器 1 (ICER1—0xE000 E184)	79
6.5.5	中断暂挂设置寄存器 0 (ISPR0—0xE000 E200)	80
6.5.6	中断暂挂设置寄存器 1 (ISPR1—0xE000 E204)	81
6.5.7	中断暂挂清除寄存器 0 (ICPR0—0xE000 E280)	82
6.5.8	中断暂挂清除寄存器 1 (ICPR1—0xE000 E284)	83
6.5.9	中断活动位寄存器 0 (IABR0—0xE000 E300)	84
6.5.10	中断活动位寄存器 1 (IABR1—0xE000 E304)	85
6.5.11	中断优先级寄存器 0 (IPR0—0xE000 E400)	86
6.5.12	中断优先级寄存器 1 (IPR1—0xE000 E404)	86
6.5.13	中断优先级寄存器 2 (IPR2—0xE000 E408)	87
6.5.14	中断优先级寄存器 3 (IPR3—0xE000 E40C)	87
6.5.15	中断优先级寄存器 4 (IPR4—0xE000 E410)	88
6.5.16	中断优先级寄存器 5 (IPR5—0xE000 E414)	88

6.5.17	中断优先级寄存器 6 (IPR6—0xE000 E418)	89
6.5.18	中断优先级寄存器 7 (IPR7—0xE000 E41C)	89
6.5.19	中断优先级寄存器 8 (IPR8—0xE000 E420)	90
6.5.20	中断优先级寄存器 9 (IPR9—0xE000 E424)	90
6.5.21	中断优先级寄存器 10 (IPR10—0xE000 E428)	91
6.5.22	软件触发中断寄存器 (STIR—0xE000 EF00)	91

第 7 章：LPC178x/7x 管脚配置 92

7.1	LPC178x/177x 管脚配置	92
7.2	引导控制.....	108

第 8 章：LPC178x/7x I/O 配置 109

8.1	本章阅读方法	109
8.2	描述	109
8.3	IOCON 寄存器.....	109
8.3.1	管脚功能	110
8.3.2	管脚模式	111
8.3.3	迟滞	111
8.3.4	输入反相	111
8.3.5	模拟/数字模式.....	111
8.3.6	输入滤波器	111
8.3.7	输出转换速率	111
8.3.8	I ² C 模式.....	112
8.3.9	开漏极模式	112
8.3.10	DAC 使能.....	112
8.4	寄存器描述	113
8.4.1	I/O 配置寄存器内容 (IOCON)	119
8.4.1.1	D 型 IOCON 寄存器 (适用于大多数 GPIO 端口管脚)	120
8.4.1.2	A 型 IOCON 寄存器 (适用于包含模拟功能的管脚)	126
8.4.1.3	U 型 IOCON 寄存器 (适用于包含 USB D+或 D-功能的管脚)	128
8.4.1.4	I 型 IOCON 寄存器 (适用于包含专用 I ² C 功能的管脚)	129
8.4.1.5	W 型 IOCON 寄存器 (这些管脚其它方面与 D 型相同, 但包含一个可选输入毛刺滤波器, 默认其上拉/下拉被禁用)	130

第 9 章：LPC178x/7x 通用输入/输出 (GPIO) 131

9.1	基本配置.....	131
9.2	特性	131
9.2.1	数字 I/O 端口	131
9.2.2	可产生中断的数字端口	131
9.3	应用	132
9.4	管脚描述.....	132
9.5	寄存器描述	133
9.5.1	GPIO 端口方向寄存器 FIOxDIR (FIO0DIR~FIO5DIR—0x2009 8000~0x2009 80A0)	134

9.5.2	GPIO 端口输出设置寄存器 FIOxSET (FIO0SET~FIO5SET—0x2009 8018~0x2009 80B8)	135
9.5.3	GPIO 端口输出清零寄存器 FIOxCLR (FIO0CLR~FIO5CLR—0x2009 801C~0x2009 80BC)	137
9.5.4	9.5.4 GPIO 端口管脚值寄存器 FIOxPIN (FIO0PIN~FIO5PIN—0x2009 8014~0x2009 80B4)	138
9.5.5	高速 GPIO 端口屏蔽寄存器 FIOxMASK (FIO0MASK~FIO5MASK—0x2009 8010~0x2009 80B0) ..	140
9.5.6	GPIO 中断寄存器	142
9.5.6.1	GPIO 整体中断状态寄存器 (IOIntStatus—0x4002 8080)	142
9.5.6.2	端口 0 上升沿的 GPIO 中断使能 (IO0IntEnR—0x4002 8090)	143
9.5.6.3	端口 2 上升沿的 GPIO 中断使能 (IO2IntEnR—0x4002 80B0)	144
9.5.6.4	端口 0 下降沿的 GPIO 中断使能 (IO0IntEnF—0x4002 8094)	145
9.5.6.5	端口 2 下降沿的 GPIO 中断使能 (IO2IntEnF—0x4002 80B4)	146
9.5.6.6	端口 0 上升沿中断的 GPIO 中断状态 (IO0IntStatR—0x4002 8084)	147
9.5.6.7	端口 2 上升沿中断的 GPIO 中断状态 (IO2IntStatR—0x4002 80A4)	148
9.5.6.8	端口 0 下降沿中断的 GPIO 中断状态 (IO0IntStatF—0x4002 8088)	149
9.5.6.9	端口 2 下降沿中断的 GPIO 中断状态 (IO2IntStatF—0x4002 80A8)	150
9.5.6.10	端口 0 的 GPIO 中断清零寄存器 (IO0IntClr—0x4002 808C)	151
9.5.6.11	端口 2 的 GPIO 中断清零寄存器 (IO2IntClr—0x4002 80AC)	152
9.6	GPIO 使用注意事项	153
9.6.1	实例：GPIO 端口上 0s 和 1s 的瞬时输出	153
9.6.2	写入 FIOSET/FIOCLR 与 FIOPIN 的比较	153

第 10 章： LPC178x/7x 外部存储控制器 (EMC) 154

10.1	如何阅读本章	154
10.2	基本配置	155
10.3	简介	155
10.4	特性	156
10.5	EMC 功能描述	156
10.5.1	AHB 从寄存器接口	158
10.5.2	AHB 从存储器接口	158
10.5.2.1	存储器传输的字节序	158
10.5.2.2	存储器传输的大小	158
10.5.2.3	写保护的存储区	158
10.5.3	焊盘接口	158
10.5.4	数据缓冲区	159
10.5.4.1	写缓冲区	159
10.5.4.2	读缓冲区	159
10.5.5	存储控制器状态机	160
10.5.6	通过可编程延时元素的时序控制	160
10.6	低功率操作	160
10.6.1	低功率 SDRAM 深度睡眠模式	160
10.6.2	低功率 SDRAM 部分数组刷新	161
10.7	存储器组选择	161
10.8	EMC 复位	161
10.9	地址移位模式	162

10.10	存储器映射 I/O 与突发禁能	162
10.11	管脚描述	162
10.12	寄存器描述	163
10.12.1	EMC 控制寄存器 (EMCControl—0x2009 C000)	165
10.12.2	EMC 状态寄存器 (EMCStatus—0x2009 C004)	166
10.12.3	EMC 配置寄存器 (EMCConfig—0x2009 C008)	166
10.12.4	动态存储器控制寄存器 (EMCDynamicControl—0x2009 C020)	166
10.12.5	动态存储器刷新定时器寄存器 (EMCDynamicRefresh—0x2009 C024)	167
10.12.6	动态存储器读取配置寄存器 (EMCDynamicReadConfig—0x2009 C028)	168
10.12.7	动态存储器预充电命令周期寄存器 (EMCDynamictRP—0x2009 C030)	168
10.12.8	动态存储器有效至预充电命令周期寄存器 (EMCDynamictRAS—0x2009 C034)	169
10.12.9	动态存储器自刷新退出时间寄存器 (EMCDynamictSREX—0x2009 C038)	169
10.12.10	动态存储器最后数据输出至有效时间寄存器 (EMCDynamictAPR—0x2009 C03C)	170
10.12.11	动态存储器数据输入至有效命令时间寄存器 (EMCDynamictDAL—0x2009 C040)	170
10.12.12	动态存储器写入恢复时间寄存器 (EMCDynamictWR—0x2009 C044)	171
10.12.13	动态存储器有效至有效命令周期寄存器 (EMCDynamictRC—0x2009 C048)	171
10.12.14	动态存储器自动刷新周期寄存器 (EMCDynamictRFC—0x2009 C04C)	172
10.12.15	动态存储器退出自刷新寄存器 (EMCDynamictXSR—0x2009 C050)	172
10.12.16	动态存储器有效组 A 至有效组 B 的时间寄存器 (EMCDynamictRRD—0x2009 C054)	173
10.12.17	动态存储器装载模式寄存器至有效命令时间寄存器 (EMCDynamictMRD—0x2009 C058)	173
10.12.18	静态存储器延长等待寄存器 (EMCStaticExtendedWait—0x2009 C080)	174
10.12.19	动态存储器配置寄存器 (EMCDynamicConfig0~3—0x2009 C100, 120, 140, 160)	174
10.12.20	动态存储器 RAS & CAS 延时寄存器 (EMCDynamicRASCAS0~3—0x2009 C104, 124, 144, 164) ..	176
10.12.21	静态存储器配置寄存器 (EMCStaticConfig0~3—0x2009 C200, 220, 240, 260)	177
10.12.22	静态存储器写使能延迟寄存器 (EMCStaticWaitWen0~3—0x2009 C204, 224, 244, 264)	178
10.12.23	静态存储器输出使能延迟寄存器 (EMCStaticWaitOen0~3—0x2009 C208, 228, 248, 268)	179
10.12.24	静态存储器读取延迟寄存器 (EMCStaticWaitRd0~3—0x2009 C20C, 22C, 24C, 26C)	179
10.12.25	静态存储器页读模式读取延迟寄存器 (EMCStaticwaitPage0~3—0x2009 C210, 230, 250, 270)	180
10.12.26	静态存储器写入延迟寄存器 (EMCStaticWaitWr0~3—0x2009 C214, 234, 254, 274)	180
10.12.27	静态存储器周转延迟寄存器 (EMCStaticWaitTurn0~3—0x2009 C218, 238, 258, 278)	181
10.12.28	延迟控制寄存器 (EMCDLYCTL—0x400F C1DC)	181
10.12.29	EMC 校准寄存器 (EMCCAL—0x400F C1E0)	182
10.13	外部存储器接口	183
10.13.1	32 位宽存储器组连接	184
10.13.2	16 位宽存储器组连接	185
10.13.3	位宽存储器组连接	186
10.13.4	存储器配置实例	187

第 11 章：LPC178x/7x 以太网

11.1	基本配置	188
11.2	简介	188
11.3	特性	189
11.4	结构与操作	190
11.5	DMA 引擎功能	191

11.6	DMA 操作概述.....	191
11.7	以太网包.....	192
11.8	综述.....	192
11.8.1	分区	192
11.8.2	PHY 设备实例	193
11.9	管脚描述.....	194
11.10	寄存器与软件接口	195
11.10.1	寄存器映射	195
11.11	以太网 MAC 寄存器定义	197
11.11.1	MAC 配置寄存器 1 (MAC1—0x2008 4000)	197
11.11.2	MAC 配置寄存器 2 (MAC2—0x2008 4004)	197
11.11.3	连续两包的内部包间隔寄存器 (IPGT—0x2008 4008)	199
11.11.4	非连续两包的内部间隔寄存器 (IPGR—0x2008 400C)	199
11.11.5	冲突窗口/重试寄存器 (CLRT—0x2008 4010)	199
11.11.6	最大帧寄存器 (MAXF—0x2008 4014)	200
11.11.7	PHY 支持寄存器 (SUPP—0x2008 4018)	200
11.11.8	测试寄存器 (TEST—0x2008 401C)	200
11.11.9	MII Mgmt 配置寄存器 (MCFG—0x2008 4020)	201
11.11.10	MII Mgmt 命令寄存器 (MCMD—0x2008 4024)	202
11.11.11	MII Mgmt 地址寄存器 (MADR—0x2008 4028)	202
11.11.12	MII Mgmt 写数据寄存器 (MWTD—0x2008 402C)	202
11.11.13	MII Mgmt 读数据寄存器 (MRDD—0x2008 4030)	203
11.11.14	MII Mgmt 指示寄存器 (MIND—0x2008 4034)	203
11.11.15	站地址 0 寄存器 (SA0—0x2008 4040)	204
11.11.16	站地址 1 寄存器 (SA1—0x2008 4044)	204
11.11.17	站地址 2 寄存器 (SA2—0x2008 4048)	204
11.12	控制寄存器定义.....	205
11.12.1	命令寄存器 (Command—0x2008 4100)	205
11.12.2	状态寄存器 (Status—0x2008 4104)	205
11.12.3	接收描述符基址寄存器 (RxDescriptor—0x2008 4108)	205
11.12.4	接收状态基址寄存器 (RxStatus—0x2008 410C)	206
11.12.5	接收描述符数目寄存器 (RxDescriptor—0x2008 4110)	206
11.12.6	接收产生索引寄存器 (RxProduceIndex—0x2008 4114)	207
11.12.7	接收消耗索引寄存器 (RxConsumeIndex—0x2008 4118)	207
11.12.8	发送描述符基址寄存器 (TxDescriptor—0x2008 411C)	207
11.12.9	发送状态基址寄存器 (TxStatus—0x2008 4120)	208
11.12.10	发送描述符数目寄存器 (TxDescriptorNumber—0x2008 4124)	208
11.12.11	发送产生索引寄存器 (TxProduceIndex—0x2008 4128)	208
11.12.12	发送消耗索引寄存器 (TxConsumeIndex—0x2008 412C)	208
11.12.13	发送状态向量 0 寄存器 (TSV0—0x2008 4158)	209
11.12.14	发送状态向量 1 寄存器 (TSV1—0x2008 415C).....	210
11.12.15	接收状态向量寄存器 (RSV—0x2008 4160)	210
11.12.16	流控制计数器寄存器 (FlowControlCounter—0x2008 4170)	211
11.12.17	流控制状态寄存器 (FlowControlStatus—0x2008 4174)	211
11.13	接收过滤寄存器定义.....	212
11.13.1	接收滤波器控制寄存器 (RxFilterCtrl—0x2008 4200)	212

11.13.2	接收滤波器 WoL 状态寄存器 (RxFilterWoLStatus—0x2008 4204)	212
11.13.3	接收滤波器 WoL 清零寄存器 (RxFilterWoLClear—0x2008 4208)	213
11.13.4	Hash 滤波器表 LSB 寄存器 (HashFilterL—0x2008 4210)	213
11.13.5	Hash 滤波器表 MSB 寄存器 (HashFilterH—0x2008 4214)	213
11.14	模块控制寄存器定义	214
11.14.1	中断状态寄存器 (IntStatus—0x2008 4FE0)	214
11.14.2	中断使能寄存器 (IntEnable—0x2008 4FE4)	214
11.14.3	中断清零寄存器 (IntClear—0x2008 4FE8)	215
11.14.4	中断置位寄存器 (IntSet—0x2008 4FEC)	215
11.14.5	掉电寄存器 (PowerDown—0x2008 4FF4)	216
11.15	描述符与状态格式	216
11.15.1	接收描述符与状态	216
11.15.2	发送描述符与状态	219
11.16	以太网模块功能描述	222
11.16.1	概述	222
11.16.2	AHB 接口	222
11.17	中断	223
11.17.1	直接存储器访问 (DMA)	223
11.17.2	初始化	225
11.17.3	发送过程	227
11.17.4	接收过程	232
11.17.5	发送重试	237
11.17.6	状态 Hash CRC 计算	237
11.17.7	双工模式	238
11.17.8	IEEE 802.3/条款 31 流控制	238
11.17.9	半双工模式背压	240
11.17.10	接收过滤	241
11.17.11	功率管理	243
11.17.12	LAN 上唤醒	244
11.17.13	接收与发送的使能与禁能	245
11.17.14	发送填充与 CRC	247
11.17.15	超长帧与帧长度检验	248
11.17.16	统计计数器	248
11.17.17	MAC 状态向量	248
11.17.18	复位	248
11.17.19	以太网错误	250
11.18	AHB 带宽	251
11.18.1	DMA 访问	251
11.18.2	CPU 访问的类型	252
11.18.3	总带宽	252
11.19	CRC 计算	253
第 12 章：LPC178x/7x LCD 控制器		255
12.1	如何阅读本章	255
12.2	基本配置	255

12.3	简介	255
12.4	特性	256
12.4.1	可编程参数	256
12.4.2	硬件光标支持	257
12.4.3	支持的 LCD 面板类型	257
12.4.4	TFT 面板	257
12.4.5	彩色 STN 面板	258
12.4.6	单色 STN 面板	258
12.5	管脚描述	258
12.5.1	信号使用	259
12.5.1.1	单面板 STN 显示器使用的信号	259
12.5.1.2	双面板 STN 显示器使用的信号	259
12.5.1.3	TFT 显示器使用的信号	260
12.6	LCD 控制器功能描述	260
12.6.1	AHB 接口	262
12.6.1.1	AMBA AHB 从接口	262
12.6.1.2	AMBA AHB 主接口	262
12.6.2	双 DMA FIFO 与相关控制逻辑	263
12.6.3	像素串行器	263
12.6.4	RAM 调色板	267
12.6.5	硬件光标	269
12.6.5.1	光标的操作	269
12.6.5.2	光标尺寸	269
12.6.5.3	光标的移动	269
12.6.5.4	光标的 XY 定位	270
12.6.5.5	光标的剪裁	270
12.6.5.6	光标图像格式	271
12.6.6	灰度计	274
12.6.7	上、下面板格式器	275
12.6.8	面板时钟发生器	275
12.6.9	时序控制器	275
12.6.10	STN 与 TFT 的数据选择	275
12.6.10.1	STN 显示器	275
12.6.10.2	TFT 显示器	275
12.6.11	中断产生	276
12.6.11.1	主机总线错误中断	276
12.6.11.2	垂直比较中断	276
12.6.11.2.1	下一个基址更新中断	276
12.6.11.2.2	FIFO 下溢中断	276
12.6.12	LCD 上电与掉电顺序	277
12.7	寄存器描述	279
12.7.1	LCD 配置寄存器 (LCD_CFG, RW—0x400F C1B8)	280
12.7.2	水平时序寄存器 (LCD_TIMH, RW — 0x2008 8000)	280
12.7.2.1	水平时序限制	280
12.7.3	垂直时序寄存器 (LCD_TIMV, RW — 0x2008 8004)	281
12.7.4	时钟与信号极性寄存器 (LCD_POL, RW — 0x2008 8008)	282
12.7.5	线端控制寄存器 (LCD_LE, RW — 0x2008 800C)	283

12.7.6	上面板帧基址寄存器 (LCD_UPBASE, RW — 0x2008 8010)	283
12.7.7	下面板帧基址寄存器 (LCD_LPBASE, RW — 0x2008 8014)	284
12.7.8	LCD 控制寄存器 (LCD_CTRL, RW — 0x2008 8018)	285
12.7.9	中断屏蔽寄存器 (LCD_INTMSK, RW — 0x2008 801C)	287
12.7.10	原始中断状态寄存器 (LCD_INTRAW, RW — 0x2008 8020)	288
12.7.11	被屏蔽中断状态寄存器 (LCD_INTSTAT, RW — 0x2008 8024)	288
12.7.12	中断清零寄存器 (LCD_INTCLR, RW — 0x2008 8028)	289
12.7.13	上面板当前地址寄存器 (LCD_UPCURR, RW — 0x2008 802C)	289
12.7.14	下面板当前地址寄存器 (LCD_LPCURR, RW - 0x2008 8030)	289
12.7.15	彩色调色板寄存器 (LCD_PAL, RW — 0x2008 8200 to 0x2008 83FC)	290
12.7.16	光标图像寄存器 (CRSR_IMG, RW — 0x2008 8800 至 0x2008 8BFC)	291
12.7.17	光标控制寄存器 (CRSR_CTRL, RW — 0x2008 8C00)	291
12.7.18	光标配置寄存器 (CRSR_CFG, RW — 0x2008 8C04)	292
12.7.19	光标调色板寄存器 0 (CRSR_PAL0, RW— 0x2008 8C08)	292
12.7.20	光标调色板寄存器 1 (CRSR_PAL1, RW — 0x2008 8C0C)	292
12.7.21	光标 XY 位置寄存器 (CRSR_XY, RW — 0x2008 8C10)	293
12.7.22	光标剪裁位置寄存器 (CRSR_CLIP, RW — 0x2008 8C14)	293
12.7.23	光标中断屏蔽寄存器 (CRSR_INTMSK, RW — 0x2008 8C20)	294
12.7.24	光标中断清零寄存器 (CRSR_INTCLR, RW — 0x2008 8C24)	294
12.7.25	光标原始中断状态寄存器 (CRSR_INTRAW, RW — 0x2008 8C28)	294
12.7.26	光标屏蔽中断状态寄存器 (CRSR_INTSTAT, RW — 0x2008 8C2C)	295
12.8	LCD 时序图	296
12.9	LCD 面板的信号使用	300
第 13 章：LPC178x/7x USB 设备控制器		303

13.1	如何阅读本章	303
13.2	基本配置	303
13.3	简介	303
13.4	特性	304
13.5	固定端点配置	305
13.6	功能描述	306
13.6.1	模拟收发器	307
13.6.2	串行接口引擎 (SIE)	307
13.6.3	端点 RAM (EP_RAM)	307
13.6.4	EP_RAM 访问控制	307
13.6.5	DMA 引擎与总线主接口	307
13.6.6	寄存器接口	307
13.6.7	SoftConnect	307
13.6.8	GoodLink	308
13.7	操作概述	308
13.8	管脚描述	308
13.9	时钟与功率管理	309
13.9.1	功率要求	309
13.9.2	时钟	309

13.9.3	功率管的支持.....	309
13.9.4	远程唤醒.....	310
13.10	寄存器描述.....	311
13.10.1	端口选择寄存器.....	312
13.10.1.1	USB 端口选择寄存器 (USBPortSel—0x2008 C110)	312
13.10.2	时钟控制寄存器.....	313
13.10.2.1	USB 时钟控制寄存器 (USBClkCtrl—0x2008 CFF4)	313
13.10.2.2	USB 时钟状态寄存器 (USBClkSt—0x2008 CFF8)	313
13.10.3	设备中断寄存器.....	314
13.10.3.1	USB 中断状态寄存器 (USBIntSt—0x2008 C1C0)	314
13.10.3.2	USB 设备中断状态寄存器 (USBDevIntSt—0x2008 C200)	314
13.10.3.3	USB 设备中断使能寄存器 (USBDevIntEn—0x2008 C204)	315
13.10.3.4	USB 设备中断清除寄存器 (USBDevIntClr—0x2008 C208)	316
13.10.3.5	USB 设备中断设置寄存器 (USBDevIntSet—0x2008 C20C)	316
13.10.3.6	USB 设备中断优先级寄存器 (USBDevIntPri—0x2008 C22C)	317
13.10.4	端点中断寄存器.....	318
13.10.4.1	USB 端点中断状态寄存器 (USBEPIntSt—0x2008 C230)	318
13.10.4.2	USB 端点中断使能寄存器 (USBEPIntEn—0x2008 C234)	319
13.10.4.3	USB 端点中断清除寄存器 (USBEPIntClr—0x2008 C238)	320
13.10.4.4	USB 端点中断设置寄存器 (USBEPIntSet—0x2008 C23C)	320
13.10.4.5	USB 端点中断优先级寄存器 (USBEPIntPri—0x2008 C240)	321
13.10.5	端点使用寄存器.....	322
13.10.5.1	EP RAM 要求.....	322
13.10.5.2	USB 使用端点寄存器 (USBReEp—0x2008 C244)	323
13.10.5.3	USB 端点索引寄存器 (USBEPIn—0x2008 C248)	324
13.10.5.4	USB MaxPacketSize 寄存器 (USBMaxPSize—0x2008 C24C)	324
13.10.6	USB 传输寄存器.....	325
13.10.6.1	USB 接收数据寄存器 (USBRxData—0x2008 C218)	325
13.10.6.2	USB 接收包长度寄存器 (USBRxPLen—0x2008 C220)	325
13.10.6.3	USB 发送数据寄存器 (USBTxData—0x2008 C21C)	325
13.10.6.4	USB 发送包长度寄存器 (USBTxPLen—0x2008 C224)	326
13.10.6.5	USB 控制寄存器 (USBCtrl—0x2008 C228)	326
13.10.7	SIE 命令代码寄存器.....	327
13.10.7.1	USB 命令代码寄存器 (USBCmdCode—0x2008 C210)	327
13.10.7.2	USB 命令数据寄存器 (USBCmdData—0x2008 C214)	327
13.10.8	DMA 寄存器.....	328
13.10.8.1	USB DMA 请求状态寄存器 (USBDMARSt—0x2008 C250)	328
13.10.8.2	USB DMA 请求清除寄存器 (USBDMARClr—0x2008 C254)	328
13.10.8.3	USB DMA 请求设置寄存器 (USBDMARSet—0x2008 C258)	329
13.10.8.4	USB UDCA Head 寄存器 (USBUDCAH—0x2008 C280)	329
13.10.8.5	USB EP DMA 状态寄存器 (USBEPDMASt—0x2008 C284)	330
13.10.8.6	USB EP DMA 使能寄存器 (USBEPDMAEn—0x2008 C288)	330
13.10.8.7	USB EP DMA 禁能寄存器 (USBEPDMADis—0x2008 C28C)	330
13.10.8.8	USB DMA 中断状态寄存器 (USBDMAIntSt—0x2008 C290)	331
13.10.8.9	USB DMA 中断使能寄存器 (USBDMAIntEn—0x2008 C294)	331
13.10.8.10	USB 传输结束中断状态寄存器 (USBEoTIntSt—0x2008 C2A0)	332
13.10.8.11	USB 传输结束中断清零寄存器 (USBEoTIntClr—0x2008 C2A4)	332
13.10.8.12	USB 传输结束中断置位寄存器 (USBEoTIntSet—0x2008 C2A8)	332

13.10.8.13	USB 新 DD 请求中断状态寄存器 (USBNDDRIntSt—0x2008 C2AC)	332
13.10.8.14	USB 新 DD 请求中断清零寄存器 (USBNDDRIntClr—0x2008 C2B0)	333
13.10.8.15	USB 新 DD 请求中断置位寄存器 (USBNDDRIntSet—0x2008 C2B4)	333
13.10.8.16	USB 系统错误中断状态寄存器 (USBSysErrIntSt—0x2008 C2B8)	333
13.10.8.17	USB 系统错误中断清零寄存器 (USBSysErrIntClr—0x2008 C2BC)	333
13.10.8.18	USB 系统错误中断置位寄存器 (USBSysErrIntSet—0x2008 C2C0)	334
13.11	中断处理	335
13.12	串行接口引擎命令描述	337
13.12.1	设置地址 (命令: 0xD0, 数据: 写 1 个字节)	338
13.12.2	配置设备 (命令: 0xD8, 数据: 写 1 个字节)	339
13.12.3	设置模式 (命令: 0xF3, 数据: 写 1 个字节)	339
13.12.4	读当前帧编号 (命令: 0xF5, 数据: 读 1 或 2 个字节)	340
13.12.5	读测试寄存器 (命令: 0xFD, 数据: 读 2 个字节)	340
13.12.6	设置设备状态 (命令: 0xFE, 数据: 写 1 个字节)	340
13.12.7	获得设备状态 (命令: 0xFE, 数据: 读 1 个字节)	341
13.12.8	获得错误代码 (命令: 0xFF, 数据: 读 1 个字节)	341
13.12.9	读错误状态 (命令: 0xFB, 数据: 读 1 个字节)	342
13.12.10	选择端点 (命令: 0x00—0x1F, 数据: 读 1 个字节 (可选))	342
13.12.11	选择端点/清除中断 (命令: 0x40—0x5F, 数据: 读 1 个字节)	343
13.12.12	设置端点状态 (命令: 0x40—0x55, 数据: 写 1 个字节 (可选))	343
13.12.13	清空缓冲区 (命令: 0xF2, 数据: 读 1 个字节 (可选))	344
13.12.14	确认缓冲区 (命令: 0xFA, 数据: 无)	345
13.13	USB 设备控制器的初始化	345
13.14	从模式操作	347
13.14.1	中断的产生	347
13.14.2	OUT 端点的数据传输	347
13.14.3	IN 端点的数据传输	347
13.15	DMA 操作	348
13.15.1	传输术语	348
13.15.2	USB 设备通信区域	348
13.15.3	触发 DMA 引擎	350
13.15.4	DMA 描述符	350
13.15.4.1	Next_DD_pointer	351
13.15.4.2	DMA_mode	351
13.15.4.3	Next_DD_valid	351
13.15.4.4	Isochronous_endpoint	352
13.15.4.5	Max_packet_size	352
13.15.4.6	DMA_buffer_length	352
13.15.4.7	DMA_buffer_start_addr	352
13.15.4.8	DD_retired	352
13.15.4.9	DD_status	352
13.15.4.10	Packet_valid	353
13.15.4.11	LS_byte_extracted	353
13.15.4.12	MS_byte_extracted	353
13.15.4.13	Present_DMA_count	353
13.15.4.14	Message_length_position	353
13.15.4.15	Isochronous_packetsize_memory_address	353
13.15.5	非同步端点操作	354

13.15.5.1	设置 DMA 传输	354
13.15.5.2	查找 DMA 描述符	354
13.15.5.3	传输数据	354
13.15.5.4	优化的描述符读取操作	354
13.15.5.5	结束数据包传输	355
13.15.5.6	No_Packet DD	355
13.15.6	同步端点操作	355
13.15.6.1	设置 DMA 传输	355
13.15.6.2	查找 DMA 描述符	356
13.15.6.3	传输数据	356
13.15.6.4	DMA 描述符结束	356
13.15.6.5	同步 OUT 端点操作举例	356
13.15.7	自动长度传输提取 (ATLE) 模式操作	358
13.15.7.1	设置 DMA 传输	359
13.15.7.2	查找 DMA 描述符	359
13.15.7.3	传输数据	359
13.15.7.4	结束包传输	360
13.16	双缓冲的端点操作	360
13.16.1	批量端点	360
13.16.2	同步端点	361

第 14 章：LPC178x/7xUSB 主机控制器.....363

14.1	如何阅读本章	363
14.2	基本配置.....	363
14.3	简介	364
14.3.1	特性	364
14.3.2	结构	365
14.4	接口	366
14.4.1	管脚描述	366
14.4.1.1	USB 主机使用注意事项	367
14.4.2	软件接口	367
14.4.2.1	寄存器映射	367
14.4.2.2	USB 主机寄存器定义	368

第 15 章：LPC178x/7x USB OTG 控制器.....369

15.1	如何阅读本章	369
15.2	基本配置.....	369
15.3	简介	369
15.4	特性	369
15.5	结构	370
15.6	操作模式.....	370
15.7	管脚配置.....	371
15.7.1	连接端口 U1 到 OTG	372

15.7.2	将端口 U1 和 U2 作为主机的连接.....	374
15.7.3	将端口 U1 用作主机、端口 U2 用作设备的连接.....	375
15.8	寄存器描述.....	376
15.8.1	USB 中断状态寄存器 (USBIntSt—0x2008 C1C0)	376
15.8.2	OTG 中断状态寄存器 (OTGIntSt—0x2008 C100)	377
15.8.3	OTG 中断使能寄存器 (OTGIntEn—0x2008 C104)	377
15.8.4	OTG 中断设置寄存器 (OTGIntSet—0x2008 C20C)	377
15.8.5	OTG 中断清除寄存器 (OTGIntClr—0x2008 C10C)	378
15.8.6	OTG 状态与控制寄存器 (OTGStCtrl—0x2008 C110)	378
15.8.7	OTG 定时器寄存器 (OTGTmr—0x2008 C114)	379
15.8.8	OTG 时钟控制寄存器 (OTGClkCtrl—0x2008 CFF4)	379
15.8.9	OTG 时钟状态寄存器 (OTGClkSt—0x2008 CFF8)	380
15.8.10	I ² C 接收寄存器 (I2C_RX—0x2008 C300)	380
15.8.11	I ² C 发送寄存器 (I2C_TX—0x2008 C300)	380
15.8.12	I ² C 状态寄存器 (I2C_STS—0x2008 C304)	381
15.8.13	I ² C 控制寄存器 (I2C_CTL—0x2008 C308)	382
15.8.14	I ² C 时钟高电平寄存器 (I2C_CLKHI—0x2008 C30C)	383
15.8.15	I ² C 时钟低电平寄存器 (I2C_CLKLO—0x2008 C310)	383
15.8.16	中断处理.....	383
15.9	支持 HNP	385
15.9.1	B 设备：外设到主机的切换	386
15.9.2	A 设备：主机到外设的 HNP 切换.....	389
15.10	时钟与功率管理.....	393
15.10.1	设备时钟请求信号	395
15.10.1.1	主机时钟请求信号	395
15.10.2	掉电模式支持.....	395
15.11	USB OTG 控制器初始化	396
第 16 章： LPC178x/7x SD 卡接口		397
16.1	基本配置.....	397
16.2	简介	397
16.3	特性	397
16.4	管脚描述.....	398
16.5	功能概述	398
16.5.1	SD 存储卡.....	398
16.5.1.1	SD 存储卡总线信号	398
16.5.2	多媒体卡	398
16.5.3	SD 卡接口详细信息	399
16.5.3.1	适配器寄存器模块	400
16.5.3.2	控制单元.....	400
16.5.3.3	命令路径.....	401
16.5.3.4	命令路径状态机	401
16.5.3.5	命令格式.....	402

16.5.3.6	数据路径	403
16.5.3.7	数据路径状态机	404
16.5.3.8	数据计数器	405
16.5.3.9	总线模式	406
16.5.3.10	CRC 令牌状态	406
16.5.3.11	状态标志	407
16.5.3.12	CRC 发生器	407
16.5.3.13	数据 FIFO	407
16.5.3.14	发送 FIFO	408
16.5.3.15	接收 FIFO	408
16.5.3.16	APB 接口	409
16.5.3.17	中断逻辑	409
16.6	寄存器说明	410
16.6.1	电源控制寄存器 (MCIPWR—0x400C 0000)	410
16.6.2	时钟控制寄存器 (MCIClock—0x400C 0004)	411
16.6.3	参数寄存器 (MCIArgument—0x400C 0008)	412
16.6.4	命令寄存器 (MCICommand—0x400C 000C)	412
16.6.5	命令响应寄存器 (MCIRespCommand—0x400C 0010)	413
16.6.6	响应寄存器 (MCIResponse0-3—0x400C 0014、0x400C 0018、.....)	413
16.6.7	数据定时器寄存器 (MCIDataTimer—0x400C 0024)	413
16.6.8	数据长度寄存器 (MCIDataLength—0x400C 0028)	414
16.6.9	数据控制寄存器 (MCIDataCtrl—0x400C 002C)	414
16.6.10	数据计数器寄存器 (MCIDataCnt—0x400C 0030)	415
16.6.11	状态寄存器 (MCIStatus—0x400C 0034)	415
16.6.12	清零寄存器 (MCIClear—0x400C 0038)	416
16.6.13	中断屏蔽寄存器 (MCIMask0—0x400C 003C)	416
16.6.14	FIFO 计数器寄存器 (MCIFifoCnt—0x400C 0048)	417
16.6.15	数据 FIFO 寄存器 (MCIFIFO—0x400C 0080 至 0x400C 00BC)	417

第 17 章：LPC178x/7x UART1 418

17.1	基本配置	418
17.2	特性	418
17.3	结构	419
17.4	管脚描述	421
17.5	寄存器描述	422
17.5.1	UART1 接收器缓冲寄存器 (U1RBR—0x4001 0000, DLAB = 0)	423
17.5.2	UART1 发送保持寄存器 (U1THR—0x4001 0000, DLAB = 0)	423
17.5.3	UART1 除数锁存器 LSB 和 MSB 寄存器 (U1DLL—0x4001 0000 和 U1DLM—0x4001 0004, DLAB = 1)	423
17.5.4	UART1 中断使能寄存器 (U1IER—0x4001 0004, DLAB = 0)	424
17.5.5	UART1 中断标识寄存器 (U1IIR—0x4001 0008)	425
17.5.6	UART1 FIFO 控制寄存器 (U1FCR—0x4001 0008)	427
17.5.6.1	DMA 操作	427
17.5.7	UART1 线控制寄存器 (U1LCR—0x4001 000C)	428
17.5.8	UART1 Modem 控制寄存器 (U1MCR—0x4001 0010)	428
17.5.9	自动流控制	429

17.5.9.1	Auto-RTS	429
17.5.9.2	Auto-CTS	430
17.5.10	UART1 线状态寄存器 (U1LSR—0x4001 0014)	431
17.5.11	UART1 Modem 状态寄存器 (U1MSR—0x4001 0018)	432
17.5.12	UART1 高速缓存寄存器 (U1SCR—0x4001 001C)	433
17.5.13	UART1 自动波特率控制寄存器 (U1ACR—0x4001 0020)	433
17.5.14	自动波特率	434
17.5.15	自动波特率模式	435
17.5.16	UART1 分数分频器寄存器 (U1FDR—0x4001 0028)	436
17.5.16.1	波特率计算	437
17.5.16.1.1	示例 1: PCLK=14.7456 MHz, BR=9600	439
17.5.16.1.2	示例 2: PCLK=12 MHz, BR=115200	439
17.5.17	UART1 发送使能寄存器 (U1TER—0x4001 0030)	439
17.5.18	UART1 RS485 控制寄存器 (U1RS485CTRL—0x4001 004C)	440
17.5.19	UART1 RS-485 地址匹配寄存器 (U1RS485ADRMATCH -0x4001 0050)	441
17.5.20	UART1 RS-485 延时值寄存器 (U1RS485DLY—0x4001 0054)	441
17.5.21	RS-485/EIA-485 操作模式	441

第 18 章： LPC178x/7x UART0/2/3 443

18.1	如何阅读本章	443
18.2	基本配置	443
18.3	特性	444
18.4	结构	444
18.5	管脚描述	446
18.6	寄存器描述	446
18.6.1	UARTn 接收器缓冲寄存器 (U0RBR—0x4000 C000, U2RBR—0x4009 8000, U3RBR—0x4009 C000, DLAB = 0)	447
18.6.2	UARTn 发送保持寄存器 (U0THR—0x4000 C000, U2THR—0x4009 8000, U3THR—0x4009 C000, DLAB = 0)	447
18.6.3	UARTn 除数锁存器 LSB 寄存器 (U0DLL—0x4000 C000, U2DLL—0x4009 8000, U3DLL—0x4009 C000, DLAB = 1) 和 UARTn 除数锁存器 MSB 寄存器 (U0DLM—0x4000 C004, U2DLM—0x4009 8004, U3DLM—0x4009 C004, DLAB = 1)	448
18.6.4	UARTn 中断使能寄存器 (U0IER—0x4000 C004, U2IER—0x4009 8004, U3IER—0x4009 C004, DLAB = 0)	448
18.6.5	UARTn 中断标识寄存器 (U0IIR—0x4000 C008, U2IIR—0x4009 8008, U3IIR—0x4009 C008) ...	449
18.6.6	UARTn FIFO 控制寄存器 (U0FCR—0x4000 C008, U2FCR—0x4009 8008, U3FCR—0x4009 C008)	451
18.6.6.1	DMA 操作	451
18.6.7	UARTn 线控制寄存器 (U0LCR—0x4000 C00C, U2LCR—0x4009 800C, U3LCR—0x4009 C00C)	452
18.6.8	UARTn 线状态寄存器 (U0LSR—0x4000 C014, U2LSR—0x4009 8014, U3LSR—0x4009 C014)	452
18.6.9	UARTn 高速缓存寄存器 (U0SCR—0x4000 C01C, U2SCR—0x4009 801C U3SCR—0x4009 C01C)	453
18.6.10	UARTn 自动波特率控制寄存器 (U0ACR—0x4000 C020, U2ACR—0x4009 8020, U3ACR—0x4009 C020)	454
18.6.10.1	自动波特率	454

18.6.10.2	自动波特率模式	455
18.6.11	UARTn 分数分频器寄存器 (U0FDR—0x4000 C028, U2FDR—0x4009 8028, U3FDR—0x4009 C028)	456
18.6.11.1	波特率计算	457
18.6.11.1.1	示例 1: PCLK=14.7456 MHz, BR=9600	459
18.6.11.1.2	示例 2: PCLK=12 MHz, BR=115200	459
18.6.12	UARTn 发送使能寄存器 (U0TER—0x4000 C030, U2TER—0x4009 8030, U3TER—0x4009 C030)	459
18.6.13	UARTn RS485 控制寄存器 (U0RS485CTRL—0x4000 C04C, U2RS485CTRL—0x4009 804C, U3RS485CTRL—0x4009 C04C)	460
18.6.14	UARTn RS-485 地址匹配寄存器 (U0RS485ADRMATCH—0x4000 C050, U2RS485ADRMATCH—0x4009 8050, U3RS485ADRMATCH—0x4009 C050)	460
18.6.15	UARTn RS-485 延时值寄存器 (U0RS485DLY—0x4000 C054, U2RS485DLY—0x4009 8054, U3RS485DLY—0x4009 C054)	461
18.6.16	RS-485/EIA-485 操作模式	461

第 19 章：LPC178x/7x UART4 463

19.1	如何阅读本章	463
19.2	基本配置	463
19.3	特性	464
19.4	结构	464
19.5	管脚描述	466
19.6	寄存器描述	466
19.6.1	UART4 接收器缓冲寄存器 (U4RBR—0x400A C000, DLAB = 0)	467
19.6.2	UART4 发送保持寄存器 (U4THR—0x400A C000, DLAB = 0)	467
19.6.3	UART4 除数锁存器 LSB 寄存器 (U4DLL—0x400A 4000, DLAB = 1) 和 UART4 除数锁存器 MSB 寄存器 (U4DLM—0x400A C004, DLAB = 1)	467
19.6.4	UART4 中断使能寄存器 (U4IER—0x400A C004, DLAB = 0)	468
19.6.5	UART4 中断标识寄存器 (U4IIR—0x400A C008)	468
19.6.6	UART4 FIFO 控制寄存器 (U4FCR—0x400A C008)	470
19.6.6.1	DMA 操作	471
19.6.7	UART4 线控制寄存器 (U4LCR—0x400A C00C)	471
19.6.8	UART4 线状态寄存器 (U4LSR—0x400A C014)	472
19.6.9	UART4 高速缓存寄存器 (U4SCR—0x400A C01C)	473
19.6.10	UART4 自动波特率控制寄存器 (U4ACR—0x400A C020)	473
19.6.10.1	自动波特率	474
19.6.10.2	自动波特率模式	475
19.6.11	UART4 IrDA 控制寄存器 (U4ICR—0x400A C024)	476
19.6.12	UART4 分数分频器寄存器 (U4FDR—0x400A C028)	477
19.6.12.1	波特率计算	478
19.6.12.1.1	示例 1: PCLK = 14.7456 MHz, BR = 9600	480
19.6.12.1.2	示例 2: PCLK = 12 MHz, BR = 115200	480
19.6.13	UART4 过采样寄存器 (U4OSR—0x400A 402C)	481
19.6.14	UART4 智能卡接口控制寄存器 (U4SCICTRL—0x400A 4048)	481
19.6.14.1	智能卡连接	482
19.6.14.2	智能卡设置	482

19.6.15	UART4 RS485 控制寄存器 (U4RS485CTRL—0x400A 404C)	482
19.6.16	UART4 RS-485 地址匹配寄存器 (U4RS485ADRMATCH—0x400A 4050)	483
19.6.17	UART4 RS-485 延时值寄存器 (U4RS485DLY—0x400A 4054)	483
19.6.18	RS-485/EIA-485 操作模式.....	483
19.6.19	UART4 同步模式控制寄存器 (U4SYNCTRL—0x400A 4058)	485
19.6.20	UART4 发送使能寄存器 (U4TER—0x400A C05C)	486

第 20 章：LPC178x/7x CAN 控制器.....487

20.1	基本配置.....	487
20.2	CAN 控制器.....	487
20.3	特性	488
20.3.1	通用 CAN 特性	488
20.3.2	CAN 控制器特性.....	488
20.3.3	验收滤波器特性	488
20.4	管脚描述.....	488
20.5	CAN 控制器结构	489
20.5.1	APB 接口模块 (AIB)	490
20.5.2	接口管理逻辑 (IML)	490
20.5.3	发送缓冲器 (TXB)	490
20.5.4	接收缓冲器 (RXB)	491
20.5.5	错误管理逻辑 (EML)	492
20.5.6	位时序逻辑 (BTL)	492
20.5.7	比特流处理器 (BSP)	492
20.5.8	CAN 控制器自测试	492
20.6	CAN 模块的内存映射	494
20.7	CAN 控制器寄存器.....	494
20.7.1	CAN 模式寄存器 (CAN1MOD—0x4004 4000, CAN2MOD—0x4004 8000)	496
20.7.2	CAN 命令寄存器 (CAN1CMR—0x4004 x004, CAN2CMR—0x4004 8004)	498
20.7.3	CAN 全局状态寄存器 (CAN1GSR—0x4004 x008, CAN2GSR—0x4004 8008)	500
20.7.4	CAN 中断和捕获寄存器 (CAN1ICR—0x4004 400C, CAN2ICR—0x4004 800C)	502
20.7.5	CAN 中断使能寄存器 (CAN1IER—0x4004 4010, CAN2IER—0x4004 8010)	505
20.7.6	CAN 总线时序寄存器 (CAN1BTR—0x4004 4014, CAN2BTR—0x4004 8014)	506
20.7.7	CAN 错误报警界限寄存器 (CAN1EWL—0x4004 4018, CAN2EWL—0x4004 8018)	507
20.7.8	CAN 状态寄存器 (CAN1SR—0x4004 401C, CAN2SR—0x4004 801C)	508
20.7.9	CAN 接收帧状态寄存器 (CAN1RFS—0x4004 4020, CAN2RFS—0x4004 8020)	510
20.7.9.1	ID 索引字段.....	511
20.7.10	CAN 接收标识符寄存器 (CAN1RID—0x4004 4024, CAN2RID -0x4004 8024)	511
20.7.11	CAN 接收数据寄存器 A (CAN1RDA—0x4004 4028, CAN2RDA -0x4004 8028)	512
20.7.12	CAN 接收数据寄存器 B (CAN1RDB—0x4004 402C, CAN2RDB—0x4004 802C)	512
20.7.13	CAN 发送帧信息寄存器 (CAN1TFI[1/2/3]—0x4004 40[30/ 40/50], CAN2TFI[1/2/3]—0x4004 80[30/40/50])	513
20.7.14	CAN 发送标识符寄存器 (CAN1TID[1/2/3]—0x4004 40[34/44/54], CAN2TID[1/2/3]—0x4004 80[34/44/54])	514
20.7.15	CAN 发送数据寄存器 A (CAN1TDA[1/2/3]—0x4004 40[38/48/58], CAN2TDA[1/2/3]—0x4004 80[38/48/58])	514
20.7.16	CAN 发送数据寄存器 B (CAN1TDB[1/2/3]—0x4004 40[3C/4C/5C], CAN2TDB[1/2/3]—0x4004	

80[3C/4C/5C])	515
20.7.17 CAN 睡眠清零寄存器 (CANSLEEPCLR—0x400F C110)	515
20.7.18 CAN 唤醒标志寄存器 (CANWAKEFLAGS—0x400F C114)	516
20.8 CAN 控制器操作	516
20.8.1 错误处理	516
20.8.2 睡眠模式	516
20.8.3 中断	517
20.8.4 发送优先级	517
20.9 集中 CAN 寄存器	517
20.9.1 集中发送状态寄存器 (CANTxSR—0x4004 0000)	518
20.9.2 集中接收状态寄存器 (CANRxSR—0x4004 0004)	518
20.9.3 集中其他状态寄存器 (CANMSR—0x4004 0008)	518
20.10 全局验收滤波器	519
20.11 验收滤波器模式	519
20.11.1 验收滤波器关闭模式	519
20.11.2 验收滤波器旁路模式	519
20.11.3 验收滤波器工作模式	520
20.11.4 FullCAN 模式	520
20.12 ID 查找表 RAM 的各个区	520
20.13 ID 查找表 RAM	520
20.14 验收滤波器寄存器	522
20.14.1 验收滤波器模式寄存器 (AFMR—0x4003 C000)	522
20.14.2 区配置寄存器	523
20.14.3 标准帧单个起始地址寄存器 (SFF_sa—0x4003 C004)	523
20.14.4 标准帧组起始地址寄存器 (SFF_GRP_sa—0x4003 C008)	524
20.14.5 扩展帧起始地址寄存器 (EFF_sa—0x4003 C00C)	524
20.14.6 扩展帧组起始地址寄存器 (EFF_GRP_sa—0x4003 C010)	524
20.14.7 AF 表结束寄存器 (ENDofTable—0x4003 C014)	525
20.14.8 状态寄存器	525
20.14.9 LUT 错误地址寄存器 (LUTerrAd—0x4003 C018)	525
20.14.10 LUT 错误寄存器 (LUTerr—0x4003 C01C)	525
20.14.11 全局 FullCAN 中断使能寄存器 (FCANIE—0x4003 C020)	526
20.14.12 FullCAN 中断和捕获寄存器 (FCANIC0—0x4003 C024 与 FCANIC1—0x4003 C028)	526
20.15 配置和搜索算法	526
20.15.1 验收滤波器搜索算法	526
20.16 FullCAN 模式	528
20.16.1 FullCAN 报文的分布	529
20.16.2 FullCAN 中断	531
20.16.2.1 FullCAN 报文中断使能位	531
20.16.2.2 报文丢失位和 CAN 通道编号	532
20.16.2.3 置位中断挂起位 (IntPnd 63~0)	533
20.16.2.4 清零中断挂起位 (IntPnd 63~0)	533
20.16.2.5 置位一个 FullCAN 报文对象的报文丢失位 (MsgLost 63~0)	533
20.16.2.6 清零一个 FullCAN 报文对象的报文丢失位 (MsgLost 63~0)	533
20.16.3 FullCAN 中断的置位和清零机制	533

20.16.3.1	状况 1: 正常情况, 无报文丢失	533
20.16.3.2	状况 2: 报文丢失	535
20.16.3.3	状况 3: 报文被改写, 由信号量位来指示	536
20.16.3.4	状况 3.1: 报文被改写, 由信号量位和报文丢失来指示	537
20.16.3.5	状况 3.2: 报文被改写, 由报文丢失来指示	538
20.16.3.6	状况 4: 清零报文丢失位	539
20.17	验收滤波器表格和 ID 索引值示例	540
20.17.1	示例 1: 仅使用一个区	540
20.17.2	示例 2: 所有的区都被使用	540
20.17.3	示例 3: 多个区 (但并非所有区) 被使用	540
20.17.4	配置示例 4	540
20.17.5	配置示例 5	541
20.17.6	配置示例 6	543
20.17.7	配置示例 7	545
20.17.8	查找表编程准则	547

第 21 章: LPC178x/7x SSP 接口 548

21.1	基本配置	548
21.2	特性	548
21.3	描述	549
21.4	管脚描述	549
21.5	总线描述	550
21.5.1	TI 同步串行数据帧格式	550
21.5.2	SPI 帧格式	550
21.5.2.1	时钟极性 (CPOL) 和相位 (CPHA) 控制	550
21.5.2.2	CPOL=0, CPHA=0 时的 SPI 格式	551
21.5.2.3	CPOL=0, CPHA=1 时的 SPI 格式	552
21.5.2.4	CPOL = 1, CPHA = 0 时的 SPI 格式	553
21.5.2.5	CPOL = 1, CPHA = 1 时的 SPI 格式	554
21.5.3	National 半导体 Microwire 帧格式	555
21.5.3.1	Microwire 模式下 CS 相对于 SK 的建立和保持时间要求	556
21.6	寄存器描述	556
21.6.1	SSPn 控制寄存器 0 (SSP0CR0—0x4008 8000, SSP1CR0—0x4003 0000, SSP2CR0—0x400A C000)	558
21.6.2	SSPn 控制器寄存器 1 (SSP0CR1—0x4008 8004, SSP1CR1—0x4003 0004, SSP2CR1—0x400A C004)	558
21.6.3	SSPn 数据寄存器 (SSP0DR—0x4008 8008, SSP1DR—0x4003 0008, SSP2DR—0x400A C008)	559
21.6.4	SSPn 状态寄存器 (SSP0SR—0x4008 800C, SSP1SR—0x4003 000C, SSP2SR—0x400A C) ...	560
21.6.5	SSPn 时钟预分频寄存器 (SSP0CPSR—0x4008 8010, SSP1CPSR—0x4003 0010, SSP2CPSR—0x400A C010)	560
21.6.6	SSPn 中断使能置位/清零寄存器 (SSP0IMSC—0x4008 8014, SSP1IMSC—0x4003 0014, SSP2IMSC—0x400A C014)	560
21.6.7	SSPn 原始中断状态寄存器 (SSP0RIS—0x4008 8018, SSP1RIS—0x4003 0018, SSP2RIS—0x400A C018)	561
21.6.8	SSPn 使能中断状态寄存器 (SSP0MIS—0x4008 801C, SSP1MIS—0x4003 001C, SSP2MIS—0x400A C01C)	561

21.6.9	C01C)561	561
21.6.9	SSPn 中断清零寄存器 (SSP0ICR—0x4008 8020, SSP1ICR—0x4003 0020, SSP2ICR—0x400A C020)562	562
21.6.10	SSPn DMA 控制寄存器 (SSP0DMACR—0x4008 8024, SSP1DMACR—0x4003 0024, SSP2DMACR—0x400A C024)562	562
第 22 章：LPC178x/7x I²C 总线接口563		563
22.1	基本配置.....563	563
22.2	特性564	564
22.3	应用564	564
22.4	描述564	564
22.4.1	I ² C 增强型快速模式566	566
22.5	管脚描述566	566
22.6	I ² C 操作模式.....567	567
22.6.1	主发送器模式.....567	567
22.6.2	主接收器模式.....568	568
22.6.3	从接收器模式.....570	570
22.6.4	从发送器模式.....571	571
22.7	I ² C 的实现和操作.....571	571
22.7.1	输入滤波器和输出级571	571
22.7.2	地址寄存器 (I2ADR0~I2ADR3)573	573
22.7.3	地址屏蔽寄存器 (I2MASK0~I2MASK3)573	573
22.7.4	比较器573	573
22.7.5	移位寄存器 I2DAT573	573
22.7.6	仲裁和同步逻辑573	573
22.7.7	串行时钟发生器575	575
22.7.8	时序和控制575	575
22.7.9	控制寄存器, I2CONSET 和 I2CONCLR.....575	575
22.7.10	状态译码器和状态寄存器575	575
22.8	寄存器描述576	576
22.8.1	I ² C 控制置位寄存器 (I2CONSET: I ² C0, I2C0CONSET—0x4001 C000; I ² C1, I2C1CONSET—0x4005 C000; I ² C2, I2C2CONSET—0x400A 0000)577	577
22.8.2	I ² C 控制清零寄存器 (I2CONCLR: I ² C0, I2C0CONCLR—0x4001 C018; I ² C1, I2C1CONCLR—0x4005 C018; I ² C2, I2C2CONCLR—0x400A 0018)579	579
22.8.3	I ² C 状态寄存器 (I2STAT: I ² C0, I2C0STAT—0x4001 C004; I ² C1, I2C1STAT—0x4005 C004; I ² C2, I2C2STAT—0x400A 0004)579	579
22.8.4	I ² C 数据寄存器 (I2DAT: I ² C0, I2C0DAT—0x4001 C008; I ² C1, I2C1DAT—0x4005 C008; I ² C2, I2C2DAT—0x400A 0008)580	580
22.8.5	I ² C 监控模式控制寄存器 (I2MMCTRL: I ² C0, I2C0MMCTRL—0x4001 C01C; I ² C1, I2C1MMCTRL—0x4005 C01C; I ² C2, I2C2MMCTRL—0x400A 001C)580	580
22.8.5.1	监控模式下的中断581	581
22.8.5.2	监控模式下的仲裁丢失581	581

22.8.6	I ² C 数据缓冲寄存器 (I2DATA_BUFFER: I ² C0, I2CDATA_BUFFER—0x4001 C02C; I ² C1, I2C1DATA_BUFFER—0x4005 C02C; I ² C2, I2C2DATA_BUFFER—0x400A 002C)	581
22.8.7	I ² C 从地址寄存器 (I2ADR0~3: I ² C0, I2C0ADR[0、1、2、3]—0x4001 C0[0C、20、24、28]; I ² C1, I2C1ADR[0、1、2、3]—地址 0x4005 C0[0C、20、24、28]; I ² C2, I2C2ADR[0、1、2、3]—地址 0x400A 00[0C、20、24、28])	582
22.8.8	I ² C 屏蔽寄存器 (I2MASK0~3: I ² C0, I2C0MASK[0、1、2、3]—0x4001 C0[30、34、38、3C]; I ² C1, I2C1MASK[0、1、2、3]—地址 0x4005 C0[30、34、38、3C]; I ² C2, I2C2MASK[0、1、2、3]—地址 0x400A 00[30、34、38、3C])	582
22.8.9	I ² C SCL 高电平占空比寄存器 (I2SCLH: I ² C0, I2C0SCLH—0x4001 C010; I ² C1, I2C1SCLH—0x4005 C010; I ² C2, I2C2SCLH—0x400A 0010)	582
22.8.10	I ² C SCL 低电平占空比寄存器 (I2SCLL: I ² C0—I2C0SCLL: 0x4001 C014; I ² C1—I2C1SCLL: 0x4005 C014; I ² C2—I2C2SCLL: 0x400A 0014)	583
22.8.11	选择合适的 I ² C 数据速率和占空比	583
22.9	I²C 操作模式的详细信息	584
22.9.1	主发送器模式	585
22.9.2	主接收器模式	587
22.9.3	从接收器模式	589
22.9.4	从发送器模式	591
22.9.5	详细的状态表	592
22.9.6	其他状态	596
22.9.6.1	I2STAT=0xF8	596
22.9.6.2	I2STAT=0x00	596
22.9.7	一些特殊情况	596
22.9.7.1	两个主机同时启动重复起始条件	596
22.9.7.2	仲裁丢失后的数据传输	596
22.9.7.3	强制访问 I ² C 总线	597
22.9.7.4	SCL 或 SDA 低电平妨碍 I ² C 总线的操作	597
22.9.7.5	总线错误	597
22.9.8	I ² C 状态服务程序	599
22.9.8.1	初始化	599
22.9.8.2	I ² C 中断服务	599
22.9.8.3	状态服务程序	600
22.9.8.4	实际应用中的状态服务	600
22.10	软件示例	600
22.10.1	初始化程序	600
22.10.2	启动主机发送功能	600
22.10.3	启动主机接收功能	600
22.10.4	I ² C 中断程序	601
22.10.5	无指定模式的状态	601
22.10.5.1	状态: 0x00	601
22.10.5.2	主机状态	601
22.10.5.3	状态: 0x08	601
22.10.5.4	状态: 0x10	601
22.10.6	主发送器状态	602

22.10.6.1	状态: 0x18	602
22.10.6.2	状态: 0x20	602
22.10.6.3	状态: 0x28	602
22.10.6.4	状态: 0x30	602
22.10.6.5	状态: 0x38	603
22.10.7	主接收状态	603
22.10.7.1	状态: 0x40	603
22.10.7.2	状态: 0x48	603
22.10.7.3	状态: 0x50	603
22.10.7.4	状态: 0x58	604
22.10.8	从接收器状态	604
22.10.8.1	状态: 0x60	604
22.10.8.2	状态: 0x68	604
22.10.8.3	状态: 0x70	604
22.10.8.4	状态: 0x78	605
22.10.8.5	状态: 0x80	605
22.10.8.6	状态: 0x88	605
22.10.8.7	状态: 0x90	605
22.10.8.8	状态: 0x98	606
22.10.8.9	状态: 0xA0	606
22.10.9	从发送器状态	606
22.10.9.1	状态: 0xA8	606
22.10.9.2	状态: 0xB0	606
22.10.9.3	状态: 0xB8	607
22.10.9.4	状态: 0xC0	607
22.10.9.5	状态: 0xC8	607

第 23 章：LPC178x/7x I²S 接口 608

23.1	基本配置	608
23.2	特性	608
23.3	描述	609
23.4	管脚描述	610
23.5	寄存器描述	611
23.5.1	数字音频输出寄存器 (I2SDAO—0x400A 8000)	611
23.5.2	数字音频输入寄存器 (I2SDAI—0x400A 8004)	612
23.5.3	发送FIFO 寄存器 (I2STXFIFO—0x400A 8008)	612
23.5.4	接收FIFO 寄存器 (I2SRXFIFO—0x400A 800C)	612
23.5.5	状态反馈寄存器 (I2SSTATE—0x400A 8010)	612
23.5.6	DMA 配置寄存器 1 (I2SDMA1—0x400A 8014)	613
23.5.7	DMA 配置寄存器 2 (I2SDMA2—0x400A 8018)	613
23.5.8	中断请求控制寄存器 (I2SIRQ—0x400A 801C)	614
23.5.9	发送时钟速率寄存器 (I2STXRATE—0x400A 8020)	614
23.5.9.1	有关分数速率发生器的说明	615
23.5.10	接收时钟速率寄存器 (I2SRXRATE—0x400A 8024)	615
23.5.11	发送时钟位速率寄存器 (I2STXBITRATE—0x400A 8028)	616
23.5.12	接收时钟位速率寄存器 (I2SRXBITRATE—0x400A 802C)	616

23.5.13	发送模式控制寄存器 (I2STXMODE—0x400A 8030)	616
23.5.14	接收模式控制寄存器 (I2SRXMODE—0x400A 8034)	617
23.6	I²S 的发送和接收接口.....	618
23.7	I²S 操作模式.....	619
23.7.1	I ² S 发送模式.....	620
23.7.2	I ² S 接收模式.....	623
23.7.2.1	全部时钟和管脚连接.....	626
23.8	FIFO 控制器.....	627
第 24 章： LPC178x/7x 定时器 0/1/2/3		629
24.1	基本配置.....	629
24.2	特性	629
24.3	应用	630
24.4	描述	630
24.5	管脚描述.....	632
24.5.1	多个 CAP 和 MAT 管脚	632
24.6	寄存器描述	632
24.6.1	中断寄存器 (T[0/1/2/3]IR—0x4000 4000, 0x4000 8000,	633
24.6.2	定时器控制寄存器 (T[0/1/2/3]CR—0x4000 4004, 0x4000 8004,	634
24.6.3	计数控制寄存器 (T[0/1/2/3]CTCR—0x4000 4070, 0x4000 8070,	635
24.6.4	定时器计数器寄存器 (T0TC—T3TC, 0x4000 4008, 0x4000 8008,	636
24.6.5	预分频寄存器 (T0PR—T3PR, 0x4000 400C, 0x4000 800C,	636
24.6.6	预分频计数器寄存器 (T0PC—T3PC, 0x4000 4010, 0x4000 8010,	636
24.6.7	匹配寄存器 (MR0—MR3)	636
24.6.8	匹配控制寄存器 (T[0/1/2/3]MCR—0x4000 4014, 0x4000 8014,	636
24.6.9	捕获寄存器 (CR0—CR1)	637
24.6.10	捕获控制寄存器 (T[0/1/2/3]CCR—0x4000 4028, 0x4000 8028,	637
24.6.11	外部匹配寄存器 (T[0/1/2/3]EMR—0x4000 403C, 0x4000 803C,	638
24.6.12	DMA 操作.....	639
24.7	定时器操作示例.....	639
第 25 章： LPC178x/7x 系统节拍定时器		641
25.1	基本配置.....	641
25.2	特性	641
25.3	描述	641
25.4	操作	641
25.5	寄存器描述	642
25.5.1	系统定时器控制和状态寄存器 (STCTRL—0xE000 E010)	642
25.5.2	系统定时器重载值寄存器 (STRELOAD—0xE000 E014)	643
25.5.3	系统定时器当前值寄存器 (STCURR—0xE000 E018)	643

25.5.4	系统定时器校准值寄存器（STCALIB—0xE000 E01C）	643
25.6	定时器计算示例	644
第 26 章：LPC178x/7x 脉宽调制器（PWM0/1）		646
26.1	基本配置	646
26.2	特性	647
26.3	描述	648
26.4	单沿和双沿控制规则的采样波形	650
26.4.1	单沿控制的 PWM 输出规则	651
26.4.2	双沿控制的 PWM 输出规则	651
26.5	管脚描述	651
26.6	寄存器描述	652
26.6.1	PWM 中断寄存器（PWM0IR—0x4001 4000 和 PWM1IR—0x4001 8000）	652
26.6.2	PWM 定时器控制寄存器（PWM0TCR—0x4001 4004 和 PWM1TCR—0x4001 8004）	653
26.6.3	PWM 计数控制寄存器（PWM0CTCR—0x4001 4070 和 PWM1CTCR—0x4001 8070）	653
26.6.4	PWM 定时器计数器（PWM0TC—0x4001 4008 和 PWM1TC—0x4001 8008）	654
26.6.5	PWM 预分频寄存器（PWM0PR—0x4001 400C 和 PWM1PR—0x4001 800C）	654
26.6.6	PWM 预分频计数器寄存器（PWM0PC—0x4001 4010 和 PWM1PC—0x4001 8010）	654
26.6.7	PWM 匹配控制寄存器（PWM0MCR—0x4001 4014 和 PWM1MCR—0x4001 8014）	654
26.6.8	PWM 匹配寄存器（PWM0MR0~PWM0MR6, PWM1MR0~PWM1MR6）	656
26.6.9	PWM 捕获控制寄存器（PWM0CCR—0x4001 4028 和 PWM1CCR—0x4001 8028）	656
26.6.10	PWM 捕获寄存器（PWM0CR0—0x4001 402C 和 PWM1CR0/CR1—0x4001 802C/30）	657
26.6.11	PWM 控制寄存器（PWM0PCR—0x4001 404C 和 PWM1PCR 0x4001 804C）	657
26.6.12	PWM 锁存使能寄存器（PWM0LER—0x4001 4050 和 PWM1LER 0x4001 8050）	658
第 27 章：LPC178x/7x 电机控制 PWM		660
27.1	基本配置	660
27.2	简介	660
27.3	描述	660
27.4	管脚描述	660
27.5	模块框图	661
27.6	配置其它模块用于 MCPWM	662
27.7	一般操作	662
27.8	寄存器描述	663
27.8.1	MCPWM 控制寄存器	664
27.8.1.1	MCPWM 控制寄存器读地址（MCCON—0x400B 8000）	664
27.8.1.2	MCPWM 控制寄存器置位地址（MCCON_SET—0x400B 8004）	665
27.8.1.3	MCPWM 控制寄存器清零地址（MCCON_CLR—0x400B 8008）	665
27.8.2	MCPWM 捕获控制寄存器	666
27.8.2.1	MCPWM 捕获控制寄存器读地址（MCCAPCON—0x400B 800C）	666
27.8.2.2	MCPWM 捕获控制寄存器置位地址（MCCAPCON_SET—0x400B 8010）	666

27.8.2.3	MCPWM 捕获控制寄存器清零地址 (MCCAPCON_CLR—0x400B 8014)	667
27.8.3	MCPWM 中断寄存器	667
27.8.3.1	MCPWM 中断使能寄存器读地址 (MCINTEN—0x400B 8050)	667
27.8.3.2	MCPWM 中断使能寄存器置位地址 (MCINTEN_SET—0x400B 8054)	667
27.8.3.3	MCPWM 中断使能寄存器清零地址 (MCINTEN_CLR—0x400B 8058)	668
27.8.3.4	MCPWM 中断标志寄存器读地址 (MCINTF—0x400B 8068)	668
27.8.3.5	MCPWM 中断标志寄存器置位地址 (MCINTF_SET—0x400B 806C)	668
27.8.3.6	MCPWM 中断标志寄存器清零地址 (MCINTF_CLR—0x400B 8070)	668
27.8.4	MCPWM 计数控制寄存器	669
27.8.4.1	MCPWM 计数控制寄存器读地址 (MCCNTCON—0x400B 805C)	669
27.8.4.2	MCPWM 计数控制寄存器置位地址 (MCCNTCON_SET—0x400B 8060)	670
27.8.4.3	MCPWM 计数控制寄存器清零地址 (MCCNTCON_CLR—0x400B 8064)	670
27.8.5	MCPWM 定时器/计数器 0~2 寄存器 (MCTC0~2—0x400B 8018, 0x400B 801C, 0x400B 8020)	670
27.8.6	MCPWM 界限 0~2 寄存器 (MCLIM0~2—0x400B 8024, 0x400B 8028, 0x400B 802C)	671
27.8.7	MCPWM 匹配 0~2 寄存器 (MCMAT0~2—0x400B 8030, 0x400B 8034, 0x400B 8038)	672
27.8.7.1	沿对齐模式下的匹配寄存器	672
27.8.7.2	中心对齐模式下的匹配寄存器	672
27.8.7.3	0 和 100% 占空比	672
27.8.8	MCPWM 死区时间寄存器 (MCDT—0x400B 803C)	673
27.8.9	MCPWM 通信格式寄存器 (MCCP—0x400B 8040)	673
27.8.10	MCPWM 捕获寄存器	674
27.8.10.1	MCPWM 捕获寄存器读地址 (MCCAP0~2—0x400B 8044, 0x400B 8048, 0x400B 804C)	674
27.8.10.2	MCPWM 捕获寄存器清零地址 (MCCAP_CLR—0x400B 8074)	674
27.9	PWM 操作	674
27.9.1	脉宽调制	674
27.9.2	映射寄存器和同时更新	677
27.9.3	快速中止 (ABORT)	677
27.9.4	捕获事件	677
27.9.5	外部事件计数 (计数器模式)	677
27.9.6	三相直流模式	678
27.9.7	三相交流模式	679
27.9.8	中断	680

第 28 章：LPC178x/7x 正交编码器接口 (QEI) 681

28.1	基本配置	681
28.2	特性	681
28.3	简介	682
28.4	功能描述	683
28.4.1	输入信号	683
28.4.1.1	正交输入信号	683
28.4.1.2	数字输入滤波	684
28.4.2	位置捕获	684
28.4.3	速度捕获	685
28.4.4	速度比较	685
28.5	管脚描述	686

28.6	寄存器描述	687
28.6.1	寄存器汇总	687
28.6.2	控制寄存器	688
28.6.2.1	QEI 控制寄存器 (QEICON—0x400B C000)	688
28.6.2.2	QEI 配置寄存器 (QEICONF—0x400B C008)	688
28.6.2.3	QEI 状态寄存器 (QEISTAT—0x400B C004)	689
28.6.3	位置、索引和定时器寄存器	689
28.6.3.1	QEI 位置寄存器 (QEIPOS—0x400B C00C)	689
28.6.3.2	QEI 最大位置值寄存器 (QEIMAXPOS—0x400B C010)	689
28.6.3.3	QEI P 位置比较寄存器 0 (CMPOS0—0x400B C014)	689
28.6.3.4	QEI 位置比较寄存器 1 (CMPOS1—0x400B C018)	690
28.6.3.5	QEI 位置比较寄存器 2 (CMPOS1—0x400B C018)	690
28.6.3.6	QEI 索引计数寄存器 (INXCNT—0x400B C020)	690
28.6.3.7	QEI 索引比较寄存器 0 (INXCMP0—0x400B C024)	690
28.6.3.8	QEI 索引比较寄存器 1 (INXCMP1—0x400B C04C)	690
28.6.3.9	QEI 索引比较寄存器 2 (INXCMP2—0x400B C050)	691
28.6.3.10	QEI 速度定时器重载寄存器 (QEILOAD—0x400B C028)	691
28.6.3.11	QEI 速度定时器寄存器 (QEITIME—0x400B C02C)	691
28.6.3.12	QEI 速度寄存器 (QEIVEL—0x400B C030)	691
28.6.3.13	QEI 速度捕获寄存器 (QEICAP—0x400B C034)	691
28.6.3.14	QEI 速度比较寄存器 (VELCOMP—0x400B C038)	692
28.6.3.15	用于相 A 的 QEI 数字滤波器 (FILTERPHA—0x400B C03C)	692
28.6.3.16	用于相 B 的 QEI 数字滤波器 (FILTERPHB—0x400B C040)	692
28.6.3.17	用于 INX 的 QEI 数字滤波器 (FILTERINX—0x400B C044)	692
28.6.3.18	QEI 索引验收窗口 (WINDOW—0x400B C048)	692
28.6.4	中断寄存器	693
28.6.4.1	QEI 中断状态寄存器 (QEIINTSTAT—0x400B CFE0)	693
28.6.4.2	QEI 中断设置寄存器 (QEISET—0x400B CFEC)	694
28.6.4.3	QEI 中断状态清除寄存器 (QEICLR—0x400B CFE8)	695
28.6.4.4	QEI 中断使能寄存器 (QEIIE—0x400B CFE4)	696
28.6.4.5	QEI 中断使能置位寄存器 (QEIIES—0x400B CFDC)	697
28.6.4.6	QEI 中断使能清除寄存器 (QEIEEC—0x400B CFD8)	698

第 29 章：LPC178x/177x 实时时钟 (RTC) 699

29.1	基本配置.....	699
29.2	特性.....	699
29.3	概述.....	699
29.4	结构.....	700
29.5	管脚描述.....	701
29.6	寄存器描述.....	701
29.6.1	RTC 中断.....	703
29.6.2	混合寄存器组.....	703
29.6.2.1	中断位置寄存器 (ILR—0x4002 4000)	703
29.6.2.2	时钟控制寄存器 (CCR—0x4002 4008)	703
29.6.2.3	计数器增量中断寄存器 (CIIR—0x4002 400C)	704
29.6.2.4	报警屏蔽寄存器 (AMR—0x4002 4010)	704

29.6.2.5	RTC 辅助控制寄存器 (RTC_AUX—0x4002 405C)	705
29.6.2.6	RTC 辅助使能寄存器 (RTC_AUXEN—0x4002 4058)	705
29.6.3	完整时间寄存器	706
29.6.3.1	完整时间寄存器 0 (CTIME0—0x4002 4014)	706
29.6.3.2	完整时间寄存器 1 (CTIME1—0x4002 4018)	706
29.6.3.3	完整时间寄存器 2 (CTIME2—0x4002 401C)	706
29.6.4	时间计数器组	707
29.6.4.1	闰年计算	707
29.6.4.2	校准寄存器 (CALIBRATION—地址 0x4002 4040)	707
29.6.5	校准过程	708
29.6.6	通用寄存器	709
29.6.6.1	通用寄存器 0~4 (GPREG0~GPREG4—地址 0x4002 4044~0x4002 4054)	709
29.6.7	报警寄存器组	709
29.7	使用注意事项	709

第 30 章：LPC178x/177x 事件监控器/记录器 710

30.1	基本配置	710
30.2	特性	710
30.3	应用	710
30.4	描述	711
30.5	管脚描述	713
30.6	寄存器描述	713
30.6.1	事件监测器/记录器控制寄存器 (ERCONTROL—0x4002 4084)	714
30.6.2	事件监测器/记录器状态寄存器 (ERSTATUS—0x4002 4080)	716
30.6.3	事件监测器/记录器计数器寄存器 (ERCOUNTERS—0x4002 4088)	717
30.6.4	事件监测器/记录器首个时间戳寄存器	717
30.6.5	事件监测器/记录器末尾时间戳寄存器	718

第 31 章：LPC178x/7x 窗口式看门狗定时器 (WWDT) 719

31.1	特性	719
31.2	应用	719
31.3	描述	720
31.4	寄存器描述	722
31.4.1	看门狗模式寄存器 (WDMOD—0x4000 0000)	722
31.4.2	看门狗定时器常数寄存器 (WDTC—base + -x04)	723
31.4.3	看门狗喂狗寄存器 (WDFEED—0x4000 0008)	723
31.4.4	看门狗定时器值寄存器 (WDTV—0x4000 000C)	724
31.4.5	看门狗定时器报警中断寄存器 (WDWARNINT—0x4000 0014)	724
31.4.6	看门狗定时器窗口寄存器 (WDWINDOW—0x4000 0018)	724
31.5	看门狗序列示例	725

第 32 章：LPC178x/7x 模数转换器 (ADC) 727

32.1	基本配置.....	727
32.2	特性.....	727
32.3	描述.....	728
32.4	管脚描述.....	728
32.5	寄存器描述.....	729
32.5.1	A/D 控制寄存器 (AD0CR—0x4003 4000)	730
32.5.2	A/D 全局数据寄存器 (AD0GDR—0x4003 4004)	731
32.5.3	A/D 中断使能寄存器 (AD0INTEN—0x4003 400C)	732
32.5.4	A/D 数据寄存器 (AD0DR0~AD0DR7—0x4003 4010~0x4003 402C)	733
32.5.5	A/D 状态寄存器 (ADSTAT—0x4003 4030)	734
32.5.6	A/D 调节寄存器 (ADTRIM—0x4003 4034)	734
32.6	操作.....	735
32.6.1	硬件触发的转换	735
32.6.2	中断.....	735
32.6.3	精度和数字接收器	735
32.6.4	DMA 控制	735

第 33 章： LPC178x/7x 数模转换器 (DAC) 736

33.1	基本配置.....	736
33.2	特性.....	736
33.3	结构.....	737
33.4	管脚描述.....	737
33.5	寄存器描述.....	738
33.5.1	D/A 转换寄存器 (DACR—0x4008 C000)	738
33.5.2	D/A 转换器控制寄存器 (DACCTRL—0x4008 C004)	739
33.5.3	D/A 转换器计数器值寄存器 (DACCNTVAL—0x4008 C008)	739
33.6	操作.....	740
33.6.1	DMA 计数器.....	740
33.6.2	双缓冲	740

第 34 章： LPC178x/7x 通用 DMA 控制器..... 741

34.1	基本配置.....	741
34.2	简介.....	741
34.3	特性.....	741
34.4	功能描述.....	742
34.4.1	DMA 控制器的功能描述	742
34.4.1.1	AHB 从机接口	742
34.4.1.2	控制逻辑和寄存器组.....	743
34.4.1.3	DMA 请求和响应接口	743
34.4.1.4	通道逻辑和通道寄存器组	743
34.4.1.5	中断请求.....	743

34.4.1.6	AHB 主机接口	743
34.4.1.6.1	总线和传输宽度	743
34.4.1.6.2	字节顺序 (Endian) 特性	743
34.4.1.6.3	错误条件	745
34.4.1.7	通道硬件	746
34.4.1.8	DMA 请求优先级	746
34.4.1.9	中断的产生	746
34.4.2	DMA 系统连接	746
34.4.2.1	DMA 请求信号	746
34.4.2.2	DMA 响应信号	746
34.4.2.3	DMA 请求连接	747
34.5	寄存器描述	748
34.5.1	DMA 中断状态寄存器 (DMACIntStat—0x2008 0000)	750
34.5.2	DMA 中断终端计数请求状态寄存器 (DMACIntTCStat—0x2008 0004)	750
34.5.3	DMA 中断终端计数请求清除寄存器 (DMACIntTCClear—0x2008 0008)	750
34.5.4	DMA 中断错误状态寄存器 (DMACIntErrStat—0x2008 000C)	750
34.5.5	DMA 中断错误清除寄存器 (DMACIntErrClr—0x2008 0010)	751
34.5.6	DMA 原始中断终端计数状态寄存器 (DMACRawIntTCStat—0x2008 0014)	751
34.5.7	DMA 原始错误中断状态寄存器 (DMACRawIntErrStat—0x2008 0018)	752
34.5.8	DMA 使能通道寄存器 (DMACEnbldChns—0x2008 001C)	752
34.5.9	DMA 软件突发请求寄存器 (DMACSoftBReq—0x2008 0020)	752
34.5.10	DMA 软件单次请求寄存器 (DMACSoftSReq—0x2008 0024)	753
34.5.11	DMA 软件最后一个突发请求寄存器 (DMACSoftLBReq—0x2008 0028)	753
34.5.12	DMA 软件最后一个单次请求寄存器 (DMACSoftLSReq—0x2008 002C)	753
34.5.13	DMA 配置寄存器 (DMACConfig—0x2008 0030)	754
34.5.14	DMA 同步寄存器 (DMACSync—0x2008 0034)	754
34.5.15	DMA 请求选择寄存器 (DMACReqSel—0x400F C1C4)	755
34.5.15.1	定时器 DMA 请求	755
34.5.16	DMA 通道寄存器	756
34.5.17	DMA 通道源地址寄存器 (DMACCxSrcAddr—0x2008 01x0)	756
34.5.18	DMA 通道目标地址寄存器 (DMACCxDestAddr—0x2008 01x4)	756
34.5.19	DMA 通道链表项寄存器 (DMACCxLLI—0x2008 01x8)	757
34.5.20	DMA 通道控制寄存器 (DMACCxControl—0x2008 01xC)	757
34.5.20.1	保护和访问信息	757
34.5.21	DMA 通道配置寄存器 (DMACCxConfig—0x2008 01x0)	759
34.5.21.1	锁定控制	760
34.5.21.2	传输类型	761
34.6	使用 DMA 控制器	762
34.6.1	编程 DMA 控制器	762
34.6.1.1	使能 DMA 控制器	762
34.6.1.2	禁能 DMA 控制器	762
34.6.1.3	使能 DMA 通道	762
34.6.1.4	禁能 DMA 通道	762
34.6.1.5	设置一个新的 DMA 传输	762
34.6.1.6	中止某个 DMA 通道	763
34.6.1.7	编程 DMA 通道	763
34.6.2	流控制	763
34.6.2.1	外设到存储器或存储器到外设 DMA 流	764

34.6.2.2	外设到外设 DMA 流	764
34.6.2.3	存储器到存储器的 DMA 流	765
34.6.3	中断请求	765
34.6.3.1	硬件中断序列	766
34.6.4	地址的产生	766
34.6.4.1	跨边界的字对齐传输	766
34.6.5	分散/聚集	766
34.6.5.1	链表项	767
34.6.5.1.1	编程 DMA 控制器用于分散/聚集 DMA	767
34.6.5.1.2	分散/聚集 DMA 的例子	767

第 35 章：LPC178x/7x CRC 引擎..... 770

35.1	简介	770
35.2	特性	770
35.3	描述	771
35.4	寄存器描述	772
35.4.1	CRC 模式寄存器 (CRC_MODE—0x2009 0000)	772
35.4.2	CRC 种子寄存器 (CRC_SEED—0x2009 0004)	772
35.4.3	CRC 校验和寄存器 (CRC_SUM – 0x2009 0008)	772
35.4.4	CRC 数据寄存器 (CRC_DATA—0x2009 0008)	773
35.5	功能描述	773

第 36 章：LPC178x/7x EEPROM 存储器 774

36.1	基本配置	774
36.2	描述	774
36.3	特性	774
36.4	寄存器描述	775
36.4.1	EEPROM 控制寄存器	775
36.4.1.1	EEPROM 命令寄存器 (EECMD—0x0020 0080)	775
36.4.1.2	EEPROM 地址寄存器 (EEADDR—0x0020 0084)	776
36.4.1.3	EEPROM 写数据寄存器 (EEWDATA—0x0020 0088)	776
36.4.1.4	EEPROM 读数据寄存器 (EERDATA—0x0020 008C)	777
36.4.1.5	EEPROM 等待状态寄存器 (EEWSTATE—0x0020 0090)	777
36.4.1.6	EEPROM 时钟分频器寄存器 (EECLKDIV—0x0020 0094)	778
36.4.1.7	EEPROM 掉电寄存器 (EEPWRDWN—0x0020 0098)	778
36.4.2	中断寄存器	779
36.4.2.1	中断使能寄存器 (EEINTEN—0x0020 0FE4)	779
36.4.2.2	中断使能清零寄存器 (EEINTENCLR, 0x0020 0FD8)	779
36.4.2.3	中断使能置位寄存器 (EEINTENSET—0x0020 0FDC)	780
36.4.2.4	中断状态寄存器 (EEINTSTAT—0x0020 0FE0)	780
36.4.2.5	中断状态清零寄存器 (0x0020 0FE8)	781
36.4.2.6	中断状态置位 (EEINTSTATSET—0x0020 0FEC)	781
36.5	EEPROM 操作	781
36.5.1	EEPROM 设备描述	781

36.5.2	EEPROM 操作	782
36.5.2.1	写	782
36.5.2.2	编程	783
36.5.2.3	读	784
36.5.2.4	出错响应	785
第 37 章： LPC178x/7x Flash 存储器		786
37.1	简介	786
37.2	特性	786
37.3	描述	786
37.3.1	复位后的存储器映射	787
37.3.1.1	有效用户代码的判定标准	787
37.3.2	通信协议	788
37.3.2.1	ISP 命令格式	788
37.3.2.2	ISP 响应格式	788
37.3.2.3	ISP 数据格式	788
37.3.2.4	ISP 流程控制	788
37.3.2.5	ISP 命令中止	789
37.3.2.6	IAP 过程中的中断	789
37.3.2.7	ISP 命令处理器使用的 RAM	789
37.3.2.8	进入用户程序之前引导过程所使用的 RAM	789
37.3.2.9	IAP 命令处理器使用的 RAM	789
37.4	引导过程流程图	790
37.5	扇区号	791
37.6	代码读保护 (CRP)	792
37.7	ISP 命令	793
37.7.1	解锁<解锁代码>	793
37.7.2	设置波特率<波特率><停止位>	794
37.7.3	回应<设定>	795
37.7.4	写 RAM<起始地址><字节数>	795
37.7.5	读存储器<地址><字节数>	795
37.7.6	准备写操作的扇区<起始扇区号> <结束扇区号>	796
37.7.7	将 RAM 内容复制到 Flash<Flash 地址><RAM 地址><字节数>	797
37.7.8	运行<地址><模式>	797
37.7.9	擦除扇区<起始扇区号><结束扇区号>	798
37.7.10	37.7.10 扇区查空<扇区号><结束扇区号>	798
37.7.11	读器件标识号	798
37.7.12	读引导代码版本号	799
37.7.13	读设备序列号	799
37.7.14	37.7.14 比较<地址 1> <地址 2> <字节数>	799
37.7.15	ISP 返回代码	800
37.8	IAP 命令	801
37.8.1	准备写操作扇区	803
37.8.2	将 RAM 内容复制到 Flash	803
37.8.3	擦除扇区	804

37.8.4	扇区查空	804
37.8.5	读器件标识号	804
37.8.6	读取引导代码版本号	805
37.8.7	读设备序列号	805
37.8.8	比较<地址 1> <地址 2> <字节数>	805
37.8.9	重新调用 ISP	806
37.8.10	AP 状态代码	806
37.9	JTA Flash 编程接口	806
37.10	Flash 符号差生成	807
37.10.1	用于生成符号差的寄存器描述	807
37.10.1.1	符号差生成地址和控制寄存器	808
37.10.1.2	符号差生成结果寄存器	808
37.10.1.3	Flash 模块状态寄存器 (FMSTAT—0x0x0020 0FE0)	809
37.10.1.4	Flash 模块状态清除寄存器 (FMSTATCLR—0x0x0020 0FE8)	809
37.10.2	生成符号差的算法和步骤	810

第 38 章：LPC178x/7x 的 JTAG、串行调试和跟踪..... 811

38.1	特性	811
38.2	简介	811
38.3	描述	811
38.4	管脚描述	812
38.5	调试连接	813
38.6	JTAG TAP 标识	815
38.7	调试注释	815
38.8	调试存储器重新映射	815
38.8.1	存储器映射控制寄存器 (MEMMAP—0x400F C040)	815

第 39 章：ARM Cortex-M3 附录..... 816

39.1	ARM Cortex-M3 用户指南：简介	816
39.1.1	关于处理器和内核外设	816
39.1.1.1	系统级接口	817
39.1.1.2	集成的可配置调试功能	817
39.1.1.3	Cortex-M3 处理器特性和优点汇总	818
39.1.1.4	Cortex-M3 内核外设	818
39.2	ARM Cortex-M3 用户指南：指令集	819
39.2.1	指令集汇总	819
39.2.2	内在函数	822
39.2.3	关于指令描述	822
39.2.3.1	操作数	823
39.2.3.2	使用 PC 或 SP 时的限制	823
39.2.3.3	灵活的第二操作数	823
39.2.3.3.1	常数	823
39.2.3.3.2	带可选移位的寄存器	824

39.2.3.4	移位操作	824
39.2.3.4.1	ASR	825
39.2.3.4.2	LSR	825
39.2.3.4.3	LSL	826
39.2.3.4.4	ROR	826
39.2.3.4.5	RRX	827
39.2.3.5	地址对齐	827
39.2.3.6	相对 PC 的表达式	828
39.2.3.7	条件执行	828
39.2.3.7.1	条件标志	829
39.2.3.7.2	条件代码后缀	829
39.2.3.8	指令宽度选择	830
39.2.3.8.1	示例：指令宽度选择	830
39.2.4	内存访问指令	831
39.2.4.1	ADR	832
39.2.4.1.1	语法	832
39.2.4.1.2	操作	832
39.2.4.1.3	限制	832
39.2.4.1.4	条件标志	832
39.2.4.1.5	示例	832
39.2.4.2	LDR 和 STR（直接偏移量）	833
39.2.4.2.1	语法	833
39.2.4.2.2	操作	833
39.2.4.2.3	限制	834
39.2.4.2.4	条件标志	835
39.2.4.2.5	示例	835
39.2.4.3	LDR 和 STR（寄存器偏移量）	836
39.2.4.3.1	语法	836
39.2.4.3.2	操作	836
39.2.4.3.3	限制	836
39.2.4.3.4	条件标志	837
39.2.4.3.5	示例	837
39.2.4.4	LDR 和 STR（非特权）	838
39.2.4.4.1	语法	838
39.2.4.4.2	操作	838
39.2.4.4.3	限制	838
39.2.4.4.4	条件标志	838
39.2.4.4.5	示例	839
39.2.4.5	LDR（相对 PC）	840
39.2.4.5.1	语法	840
39.2.4.5.2	操作	840
39.2.4.5.3	限制	840
39.2.4.5.4	条件标志	841
39.2.4.5.5	示例	841
39.2.4.6	LDM 和 STM	842
39.2.4.6.1	语法	842
39.2.4.6.2	操作	842
39.2.4.6.3	限制	843
39.2.4.6.4	条件标志	843

39.2.4.6.5	示例	843
39.2.4.6.6	不正确示例	843
39.2.4.7	PUSH 和 POP	844
39.2.4.7.1	语法	844
39.2.4.7.2	操作	844
39.2.4.7.3	限制	844
39.2.4.7.4	条件标志	844
39.2.4.7.5	示例	844
39.2.4.8	LDREX 和 STREX	845
39.2.4.8.1	语法	845
39.2.4.8.2	操作	845
39.2.4.8.3	限制	846
39.2.4.8.4	条件标志	846
39.2.4.8.5	示例	846
39.2.4.9	CLREX	847
39.2.4.9.1	语法	847
39.2.4.9.2	操作	847
39.2.4.9.3	条件标志	847
39.2.4.9.4	示例	847
39.2.5	通用数据处理指令	848
39.2.5.1	ADD、ADC、SUB、SBC 和 RSB	849
39.2.5.1.1	语法	849
39.2.5.1.2	操作	849
39.2.5.1.3	限制	850
39.2.5.1.4	条件标志	850
39.2.5.1.5	示例	850
39.2.5.1.6	多字算术操作示例	850
39.2.5.2	AND, ORR, EOR, BIC 和 ORN	851
39.2.5.2.1	语法	851
39.2.5.2.2	操作	851
39.2.5.2.3	限制	852
39.2.5.2.4	条件标志	852
39.2.5.2.5	示例	852
39.2.5.3	ASR, LSL, LSR, ROR 和 RRX	853
39.2.5.3.1	语法	853
39.2.5.3.2	操作	853
39.2.5.3.3	限制	854
39.2.5.3.4	条件标志	854
39.2.5.3.5	示例	854
39.2.5.4	CLZ	855
39.2.5.4.1	语法	855
39.2.5.4.2	操作	855
39.2.5.4.3	限制	855
39.2.5.4.4	条件标志	855
39.2.5.4.5	示例	855
39.2.5.5	CMP 和 CMN	856
39.2.5.5.1	语法	856
39.2.5.5.2	操作	856
39.2.5.5.3	限制	856

39.2.5.5.4	条件标志	856
39.2.5.5.5	示例	856
39.2.5.6	MOV 和 MVN	857
39.2.5.6.1	语法	857
39.2.5.6.2	操作	857
39.2.5.6.3	限制	858
39.2.5.6.4	条件标志	858
39.2.5.6.5	示例	858
39.2.5.7	MOVT	859
39.2.5.7.1	语法	859
39.2.5.7.2	操作	859
39.2.5.7.3	限制	859
39.2.5.7.4	条件标志	859
39.2.5.7.5	示例	859
39.2.5.8	REV, REV16, REVSH 和 RBIT	860
39.2.5.8.1	语法	860
39.2.5.8.2	操作	860
39.2.5.8.3	限制	860
39.2.5.8.4	条件标志	860
39.2.5.8.5	示例	860
39.2.5.9	TST 和 TEQ	861
39.2.5.9.1	语法	861
39.2.5.9.2	操作	861
39.2.5.9.3	限制	861
39.2.5.9.4	条件标志	861
39.2.5.9.5	示例	862
39.2.6	乘法和除法指令	863
39.2.6.1	MUL、MLA 和 MLS	864
39.2.6.1.1	语法	864
39.2.6.1.2	操作	864
39.2.6.1.3	限制	864
39.2.6.1.4	条件标志	865
39.2.6.1.5	示例	865
39.2.6.2	UMULL, UMLAL, SMULL 和 SMLAL	866
39.2.6.2.1	语法	866
39.2.6.2.2	操作	866
39.2.6.2.3	限制	866
39.2.6.2.4	条件标志	866
39.2.6.2.5	示例	867
39.2.6.3	SDIV 和 UDIV	868
39.2.6.3.1	语法	868
39.2.6.3.2	操作	868
39.2.6.3.3	限制	868
39.2.6.3.4	条件标志	868
39.2.6.3.5	示例	868
39.2.7	饱和指令	869
39.2.7.1	SSAT 和 USAT	869
39.2.7.1.1	语法	869
39.2.7.1.2	操作	869

39.2.7.1.3	限制	870
39.2.7.1.4	条件标志	870
39.2.7.1.5	示例	870
39.2.8	位域指令	871
39.2.8.1	BFC 和 BFI	872
39.2.8.1.1	语法	872
39.2.8.1.2	操作	872
39.2.8.1.3	限制	872
39.2.8.1.4	条件标志	872
39.2.8.1.5	示例	872
39.2.8.2	SBFX 和 UBFX	873
39.2.8.2.1	语法	873
39.2.8.2.2	操作	873
39.2.8.2.3	限制	873
39.2.8.2.4	条件标志	873
39.2.8.2.5	示例	873
39.2.8.3	SXT 和 UXT	874
39.2.8.3.1	语法	874
39.2.8.3.2	操作	874
39.2.8.3.3	限制	874
39.2.8.3.4	条件标志	874
39.2.8.3.5	示例	875
39.2.9	跳转和控制指令	876
39.2.9.1	B、BL、BX 和 BLX	877
39.2.9.1.1	语法	877
39.2.9.1.2	操作	877
39.2.9.1.3	限制	878
39.2.9.1.4	条件标志	878
39.2.9.1.5	示例	878
39.2.9.2	CBZ 和 CBNZ	879
39.2.9.2.1	语法	879
39.2.9.2.2	操作	879
39.2.9.2.3	限制	879
39.2.9.2.4	条件标志	879
39.2.9.2.5	示例	879
39.2.9.3	IT	880
39.2.9.3.1	语法	880
39.2.9.3.2	操作	880
39.2.9.3.3	限制	880
39.2.9.3.4	条件标志	881
39.2.9.3.5	示例	881
39.2.9.4	TBB 和 TBH	882
39.2.9.4.1	语法	882
39.2.9.4.2	操作	882
39.2.9.4.3	限制	882
39.2.9.4.4	条件标志	882
39.2.9.4.5	示例	882
39.2.10	其他指令	884
39.2.10.1	BKPT	885

39.2.10.1.1	语法	885
39.2.10.1.2	操作	885
39.2.10.1.3	条件标志	885
39.2.10.1.4	示例	885
39.2.10.2	CPS	886
39.2.10.2.1	语法	886
39.2.10.2.2	操作	886
39.2.10.2.3	限制	886
39.2.10.2.4	条件标志	886
39.2.10.2.5	示例	886
39.2.10.3	DMB	887
39.2.10.3.1	语法	887
39.2.10.3.2	操作	887
39.2.10.3.3	条件标志	887
39.2.10.3.4	示例	887
39.2.10.4	DSB	888
39.2.10.4.1	语法	888
39.2.10.4.2	操作	888
39.2.10.4.3	条件标志	888
39.2.10.4.4	示例	888
39.2.10.5	ISB	889
39.2.10.5.1	语法	889
39.2.10.5.2	操作	889
39.2.10.5.3	条件标志	889
39.2.10.5.4	示例	889
39.2.10.6	MRS	890
39.2.10.6.1	语法	890
39.2.10.6.2	操作	890
39.2.10.6.3	限制	890
39.2.10.6.4	条件标志	890
39.2.10.6.5	示例	890
39.2.10.7	MSR	891
39.2.10.7.1	语法	891
39.2.10.7.2	操作	891
39.2.10.7.3	限制	891
39.2.10.7.4	条件标志	891
39.2.10.7.5	示例	891
39.2.10.8	NOP	892
39.2.10.8.1	语法	892
39.2.10.8.2	操作	892
39.2.10.8.3	条件标志	892
39.2.10.8.4	示例	892
39.2.10.9	SEV	893
39.2.10.9.1	语法	893
39.2.10.9.2	操作	893
39.2.10.9.3	条件标志	893
39.2.10.9.4	示例	893
39.2.10.10	SVC	894
39.2.10.10.1	语法	894
39.2.10.10.2	操作	894

39.2.10.10.3	条件标志	894
39.2.10.10.4	示例	894
39.2.10.11	WFE	895
39.2.10.11.1	语法	895
39.2.10.11.2	操作	895
39.2.10.11.3	条件标志	895
39.2.10.11.4	示例	895
39.2.10.12	WFI	896
39.2.10.12.1	语法	896
39.2.10.12.2	操作	896
39.2.10.12.3	条件标志	896
39.2.10.12.4	示例	896
39.3	ARM Cortex-M3 用户指南：处理器	897
39.3.1	编程模型	897
39.3.1.1	处理器模式和软件执行的特权等级	897
39.3.1.2	堆栈	897
39.3.1.3	内核寄存器	898
39.3.1.3.1	通用寄存器	899
39.3.1.3.2	堆栈指针	900
39.3.1.3.3	链接寄存器	900
39.3.1.3.4	程序计数器	900
39.3.1.3.5	程序状态寄存器	900
39.3.1.3.6	异常屏蔽寄存器	903
39.3.1.3.7	控制寄存器	904
39.3.1.4	异常和中断	904
39.3.1.5	数据类型	905
39.3.1.6	Cortex 微控制器软件接口标准	905
39.3.2	存储器模型	906
39.3.2.1	存储区、类型和属性	906
39.3.2.2	存储器系统访问的排序	907
39.3.2.3	存储器访问行为	908
39.3.2.4	存储器访问的软件排序	908
39.3.2.5	位段	909
39.3.2.5.1	直接访问一个别名区	911
39.3.2.5.2	直接访问一个位段区	911
39.3.2.6	存储器字节序	911
39.3.2.6.1	小端（Little-endian）格式	911
39.3.2.7	同步原语	912
39.3.2.8	同步原语的编程提示	913
39.3.3	异常模型	913
39.3.3.1	异常状态	913
39.3.3.2	异常类型	914
39.3.3.3	异常处理程序	916
39.3.3.4	向量表	916
39.3.3.5	异常优先级	917
39.3.3.6	中断优先级分组	917
39.3.3.7	异常进入和返回	918
39.3.3.7.1	异常进入	918
39.3.3.7.2	异常返回	919

39.3.4	故障处理	921
39.3.4.1	故障类型	921
39.3.4.2	故障升级和硬故障	921
39.3.4.3	故障状态寄存器和故障地址寄存器	922
39.3.4.4	锁定	922
39.3.5	电源管理	923
39.3.5.1	进入睡眠模式	923
39.3.5.1.1	等待中断	923
39.3.5.1.2	等待事件	923
39.3.5.1.3	退出时睡眠（Sleep-on-exit）	924
39.3.5.2	从睡眠模式唤醒	924
39.3.5.2.1	从“WFI”或“退出时睡眠”唤醒	924
39.3.5.2.2	从“WFE”唤醒	924
39.3.5.3	唤醒中断控制器	924
39.3.5.4	电源管理编程提示	925
39.4	ARM Cortex-M3 用户指南：外设	926
39.4.1	关于 Cortex-M3 外设	926
39.4.2	可嵌套向量中断控制器	926
39.4.2.1	Cortex-M3 NVIC 寄存器的 CMSIS 映射	927
39.4.2.2	中断置位使能寄存器	927
39.4.2.3	中断清零使能寄存器	928
39.4.2.4	中断置位挂起寄存器	929
39.4.2.5	中断清零挂起寄存器	929
39.4.2.6	中断有效位寄存器	930
39.4.2.7	中断优先级寄存器	930
39.4.2.8	软件触发中断寄存器	931
39.4.2.9	电平触发和脉冲中断	931
39.4.2.9.1	中断的软/硬件控制	931
39.4.2.10	NVIC 设计提示和建议	932
39.4.2.10.1	NVIC 编程提示	933
39.4.3	系统控制模块	934
39.4.3.1	Cortex-M3 SCB 寄存器的 CMSIS 映射	934
39.4.3.2	辅助控制寄存器（ACTLR）	934
39.4.3.2.1	关于 IT 折叠	935
39.4.3.3	CPUID 基址寄存器	936
39.4.3.4	中断控制和状态寄存器	936
39.4.3.5	向量表偏移量寄存器	938
39.4.3.6	应用中断和复位控制寄存器	939
39.4.3.6.1	二进制点	939
39.4.3.7	系统控制寄存器	940
39.4.3.8	配置和控制寄存器	941
39.4.3.9	系统处理程序优先级寄存器	941
39.4.3.9.1	系统处理程序优先级寄存器 1	942
39.4.3.9.2	系统处理程序优先级寄存器 2	942
39.4.3.9.3	系统处理程序优先级寄存器 3	942
39.4.3.10	系统处理程序控制和状态寄存器	942
39.4.3.11	可配置故障状态寄存器	944
39.4.3.11.1	存储器管理故障状态寄存器	944

39.4.3.11.2	总线故障状态寄存器	945
39.4.3.11.3	使用故障状态寄存器	947
39.4.3.12	硬故障状态寄存器	948
39.4.3.13	存储器管理故障地址寄存器	948
39.4.3.14	总线故障地址寄存器	949
39.4.3.15	系统控制模块设计提示和建议	949
39.4.4	系统定时器 SysTick	950
39.4.4.1	SysTick 控制和状态寄存器	950
39.4.4.2	SysTick 重载值寄存器	951
39.4.4.2.1	计算重载值	951
39.4.4.3	SysTick 当前值寄存器	951
39.4.4.4	SysTick 校准值寄存器	951
39.4.4.5	SysTick 设计提示和建议	952
39.4.5	存储器保护单元	953
39.4.5.1	MPU 类型寄存器	954
39.4.5.2	MPU 控制寄存器	956
39.4.5.3	MPU 区号寄存器	957
39.4.5.4	MPU 区基址寄存器	958
39.4.5.4.1	ADDR 域	958
39.4.5.5	MPU 区的属性和大小寄存器	958
39.4.5.5.1	SIZE 域的值	959
39.4.5.6	MPU 访问权限属性	960
39.4.5.7	MPU 不匹配	961
39.4.5.8	更新一个 MPU 区	961
39.4.5.8.1	使用单独的字更新一个 MPU 区	961
39.4.5.8.2	使用多字写入更新一个 MPU 区	962
39.4.5.8.3	子区	963
39.4.5.9	MPU 设计提示和建议	964
39.4.5.9.1	微控制器的 MPU 配置	964
39.5	ARM Cortex-M3 用户指南：术语表	965
第 40 章：	补充信息	969
40.1	缩写	969
40.2	法律信息	970
40.2.1	定义	970
40.2.2	免责条款	970
40.2.3	商标	970
40.3	表目录	971
40.4	图目录	987
40.5	目录	990