LPC2101/02/03 使用指南

目录

第1章 概述

1.1 简介

LPC2101/02/03 是基于一个支持实时仿真的 16/32 位 ARM7 TDMI-S CPU 的微控制器，并带有 8kB,16kB 或 32kB 嵌入的高速 Flash 存储器。128 位宽度的存储器接口和独特的加速结构使 32 位代码能够在最大时钟速率下运行。对中断服务程序和 DSP 算法中性能要求严格的应用，这增加的性能比在 Thumb 模式下的性能超出多达 30%。对代码规模有严格控制的应用，使用 16 位 Thumb 模式将代码规模降低超过 30%，而性能的损失却很小。

较小的封装和很低的功耗使 LPC2101/02/03 特别适用于访问控制和 POS 机等小型应用中；由于内置了宽范围的串行通信接口（范围从多个 UART、SPI 和 SSP 到两条 I^2C 总线）和 2kB/4kB/8kB 的片内 SRAM，它们也非常适合于通信网关和协议转换器。高级性能还使这些器件适合用作数学协处理器。多个 32 位和 16 位定时器、1 个改良的 10 位 ADC、所有定时器上输出匹配的 PWM 特性、以及具有多达 13 个边沿或电平触发的外部中断管脚的 32 条高速 GPIO 线，使这些微控制器特别适用于工业控制和医疗系统中。

1.2 特性

- 16/32 位 ARM7 TDMI-S 微控制器，超小 LQFP48 封装。
- 2kB/4kB/8kB 的片内静态 RAM 和 8kB/16kB/32kB 的片内 Flash 程序存储器。128 位宽度接口/加速器可实现高达 70 MHz 工作频率。
- 通过片内 boot 装载程序实现在系统/在应用编程（ISP/IAP）。单个 Flash 扇区或整片擦除时间为 100ms。256 字节编程时间为 1ms。
- 嵌入式 ICE RT 通过片内 RealMonitor 软件提供实时调试。
- 10 位 A/D 转换器提供 8 路模拟输入（每个通道的转换时间低至 2.44us），以及特定的结果寄存器来最大限度地减少中断开销。
- 2 个 32 位定时器/外部事件计数器（带 7 路捕获和 7 路比较通道）。
- 2 个 16 位定时器/外部事件计数器（带 3 路捕获和 7 路比较通道）。
- 低功耗实时时钟(RTC)具有独立的电源和特定的 32kHz 时钟输入。
- 多个串行接口，包括 2 个 UART(16C550)、2 个高速 I^2C 总线(400 kbit/s)、SPI 和具有缓冲作用和数据长度可变功能的 SSP。
- 向量中断控制器(VIC)，可配置优先级和向量地址。
- 多达 32 个通用 I/O 口（可承受 5V 电压）。
- 多达 13 个边沿或电平触发的外部中断管脚。
- 通过一个可编程的片内 PLL（100us 的设置时间）可实现最大为 70MHz 的 CPU 操作频率，其具有 10MHz～25MHz 的输入频率。
- 片内集成振荡器与外部晶体的操作频率范围为 1～25MHz。
- 低功耗模式包括空闲模式、掉电模式和带有效 RTC 的掉电模式。

- 可通过个别使能/禁止外围功能和外围时钟分频来优化额外功耗。
- 通过外部中断或 RTC 将处理器从掉电模式中唤醒。

1.3 应用

- 工业控制
- 医疗系统
- 访问控制
- POS 机
- 通信网关
- 嵌入式软 modem
- 一般性应用

1.4 器件信息

表 1 LPC2101/02/03 器件信息

器件	管脚数	片内 SRAM	片内 FLASH	ADC 通道	注释
LPC2101	48	2kB	8kB	8 输入	-
LPC2102	48	4 kB	16 kB	8 输入	-
LPC2103	48	8 kB	32 kB	8 输入	带完整 modem 接口的 UART1

1.5 结构概述

LPC2101/02/03 包含一个支持仿真的 ARM7TDMI-S CPU，片内存储器控制器接口的 ARM7 局部总线，中断控制器接口的 AMBA 先进高性能总线（AHB）和连接片内外设功能的 ARM 外设总线（APB，ARM AMBA 先进外设总线的兼容超集）。LPC2101/02/03 将 ARM7 TDMI-S 处理器配置为小端字节顺序。

AHB 外设分配了 2M 字节的地址范围，它位于 4G 字节 ARM 存储器空间的最顶端。每个 AHB 外设都在 AHB 地址空间内分配了 16k 字节的地址空间。LPC2101/02/03 的外设功能（中断控制器除外）都连接到 APB 总线。AHB 到 APB 的桥将 APB 总线与 AHB 总线相连。APB 外设也分配了 2M 字节的地址范围，从 3.5G 字节地址点开始。每个 APB 外设在 APB 地址空间内都分配了 16k 字节地址空间。

片内外设与器件管脚的连接由管脚连接模块控制（见 7.4 节）。该模块必须由软件进行配置以符合外设功能与管脚在特定应用中的需求。

1.6 ARM7TDMI-S 处理器

ARM7TDMI-S 是通用的 32 位微处理器，它具有高性能和低功耗的特性。ARM 结构是基于精简指令集计算机(RISC)原理而设计的，指令集和相关的译码机制比微程序的复杂指令集计算机要简单得多。这样使用一个小的、廉价的处理器核，就可实现很高的指令吞吐量和

实时的中断响应。

由于使用了流水线技术，处理和存储系统的所有部分都可连续工作。通常，在执行一条指令的同时对下一条指令进行译码，并将第三条指令从存储器中取出。

ARM7TDMI-S 处理器也使用了一个被称为 THUMB 的独特结构化策略，它非常适用于那些对存储器有限制或者需要较高代码密度的大批量产品的应用。

在 THUMB 后面一个关键的概念是“超精简指令集”。基本上，ARM7TDMI-S 处理器具有两个指令集：

- 标准 32 位 ARM 指令集
- 16 位 THUMB 指令集

THUMB 指令集的 16 位指令长度使其可以达到标准 ARM 代码两倍的密度，却仍然保持 ARM 的大多数性能上的优势，这些优势是使用 16 位寄存器的传统 16 位处理器所不具备的。因为 THUMB 代码和 ARM 代码一样，在相同的 32 位寄存器集上进行操作。

THUMB 代码仅为 ARM 代码规模的 65%，但其性能却相当于连接到 16 位存储器系统的相同 ARM 处理器性能的 160%。

关于 ARM7TDMI-S 处理器的详细内容请参阅 ARM 官方网站上的 ARM7TDMI-S 数据手册。

1.7 片内 Flash 存储器系统

LPC2101/02/03 分别含有 8kB、16kB 和 32kB Flash 存储器系统。该存储器可用作代码和数据的存储。对 FLASH 存储器的编程可通过几种方法来实现：

- 使用内置的串行 JTAG 接口
- 使用在系统编程（ISP）和 UART
- 使用在应用编程（IAP）功能

使用 IAP 功能的应用程序也可以在应用程序运行时对 Flash 进行擦除和/或编程，这样为数据存储和现场固件升级等都带来了极大的灵活性。整个 Flash 存储器都可用来存放用户代码，因为引导装载程序位于一个独立的存储器单元中。

LPC2101/02/03 的 Flash 存储器至少可擦除/编程 100,000 次，保存数据的时间长达 20 年。

1.8 片内静态 RAM(SRAM)

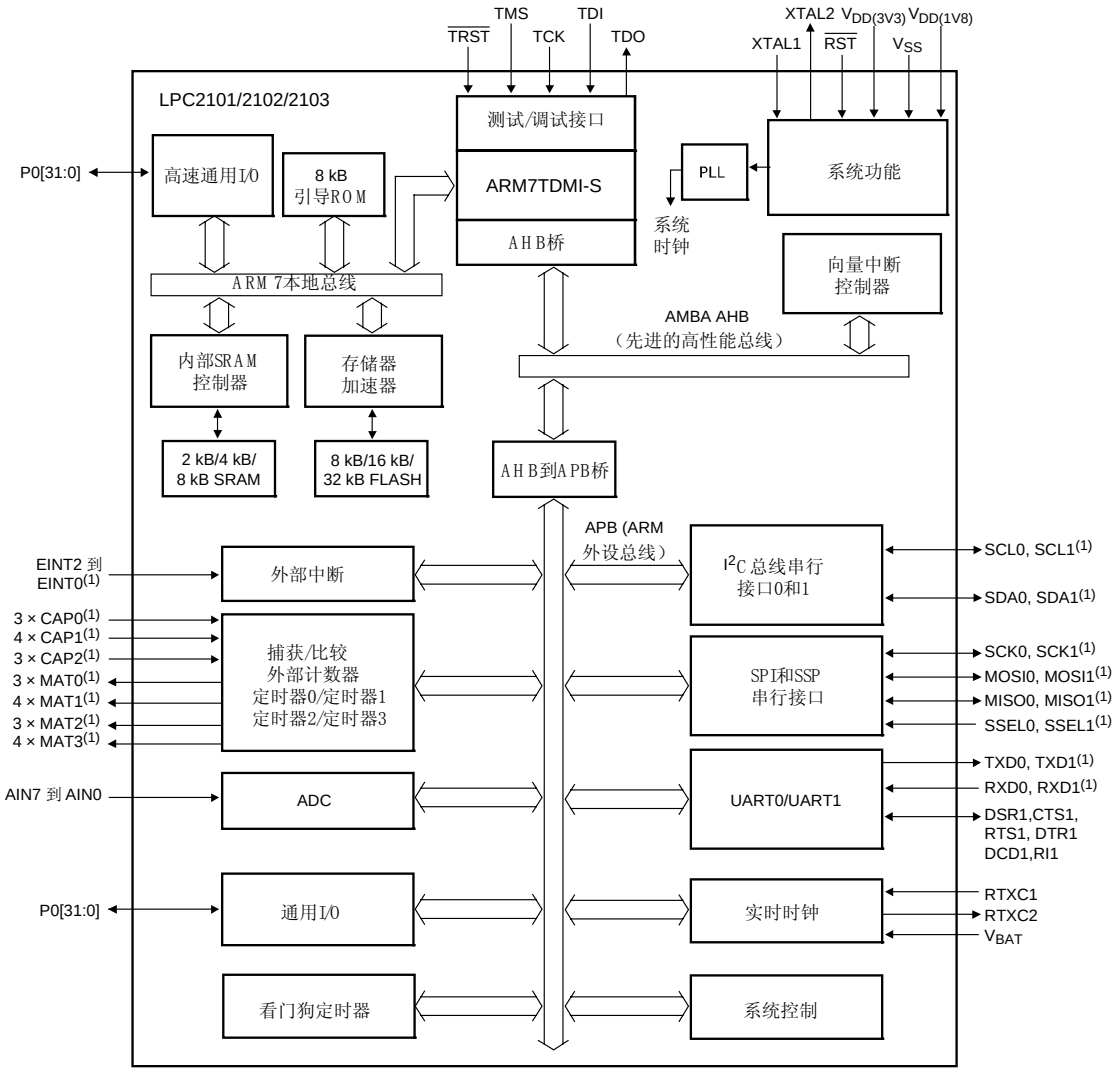
片内静态 RAM（SRAM）可用作代码和/或数据的存储，它支持 8 位、16 位和 32 位的访问。LPC2101/02/03 分别含有 2/4/8kB 的静态 RAM。

LPC2101/02/03 SRAM 可作为一个字节寻址的存储器访问。对存储器进行字和半字访问时将忽略地址对准，并访问被寻址的自然对准值（因此，对存储器进行字访问时将忽略地址位 0 和 1，半字访问时将忽略地址位 0）。因此，有效的读写操作要求半字数据访问的地址线 0 为 0（地址以 0、2、4、6、8、A、C 和 E 结尾），字数据访问的地址线 0 和 1 都为 0（地址以 0、4、8 和 C 结尾）。该原则同样用于片外和片内存储器。

SRAM 控制器包含一个回写缓冲区，它用于防止 CPU 在连续的写操作时停止运行。回写缓冲区总是保存着软件发送到 SRAM 的最后一个字节。该数据只有在软件请求下一次写

操作时才写入 SRAM（数据只有在软件执行另外一次写操作时被写入 SRAM）。如果发生芯片复位，实际的 SRAM 内容将不会反映最近一次的写请求（即：在一次“热”芯片复位后，SRAM 不会反映最后一次写入的内容）。任何在复位后检查 SRAM 内容的程序都必须注意这一点。通过对一个单元执行两次相同的写操作可保证复位后数据的写入。或者，也可通过在进入空闲或掉电模式前执行虚写（dummy write）操作来保证最后的数据在复位后被真正写入到 SRAM。

1.9 方框图



(1) 管脚与 GPIO 共用。

图 1 LPC2101/02/03 方框图

第2章 LPC2101/02/03 存储器寻址

2.1 存储器映射

LPC2101/02/03 包含几个不同的存储器组，见以下各图。图 2 所示为复位后从用户编程角度所看到的整个地址空间映射。中断向量区域支持地址的重新映射，详见这部分后面的描述。

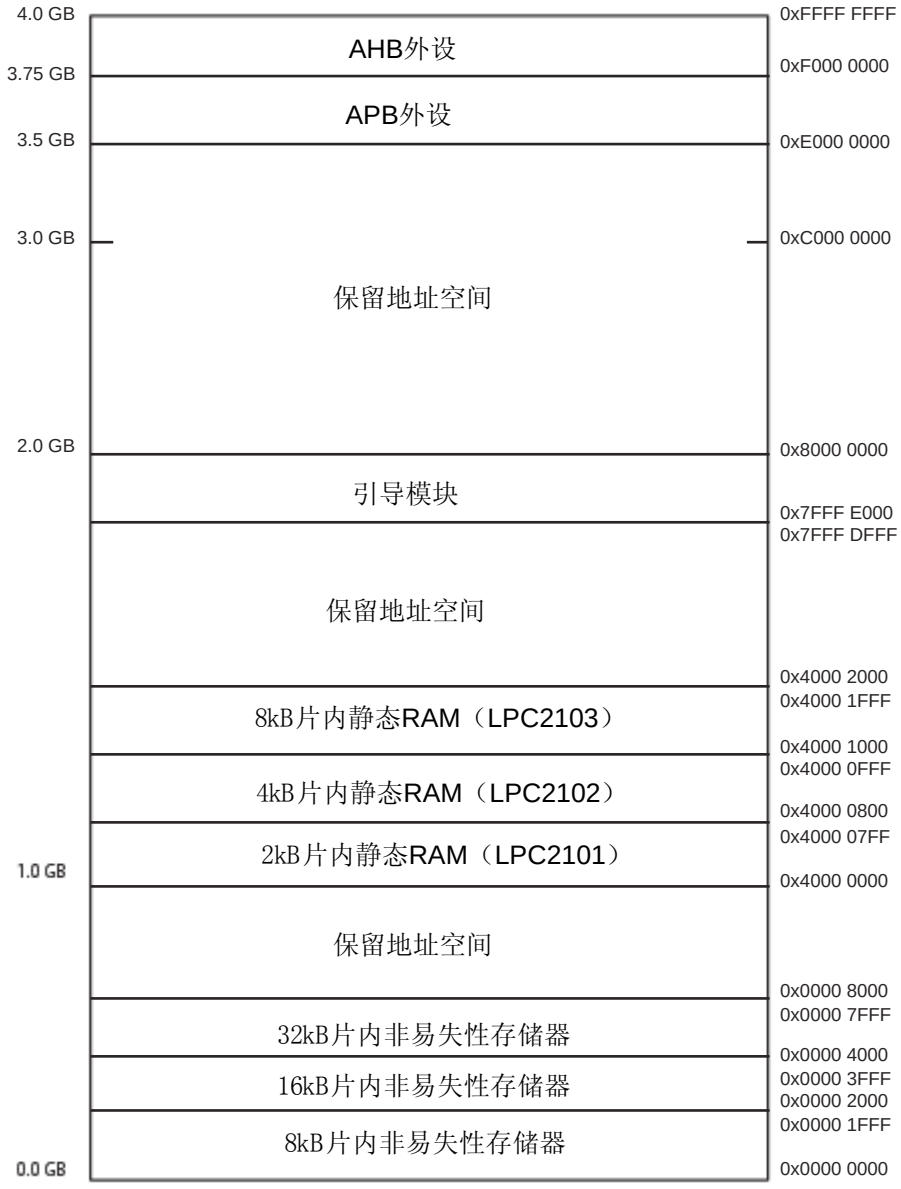
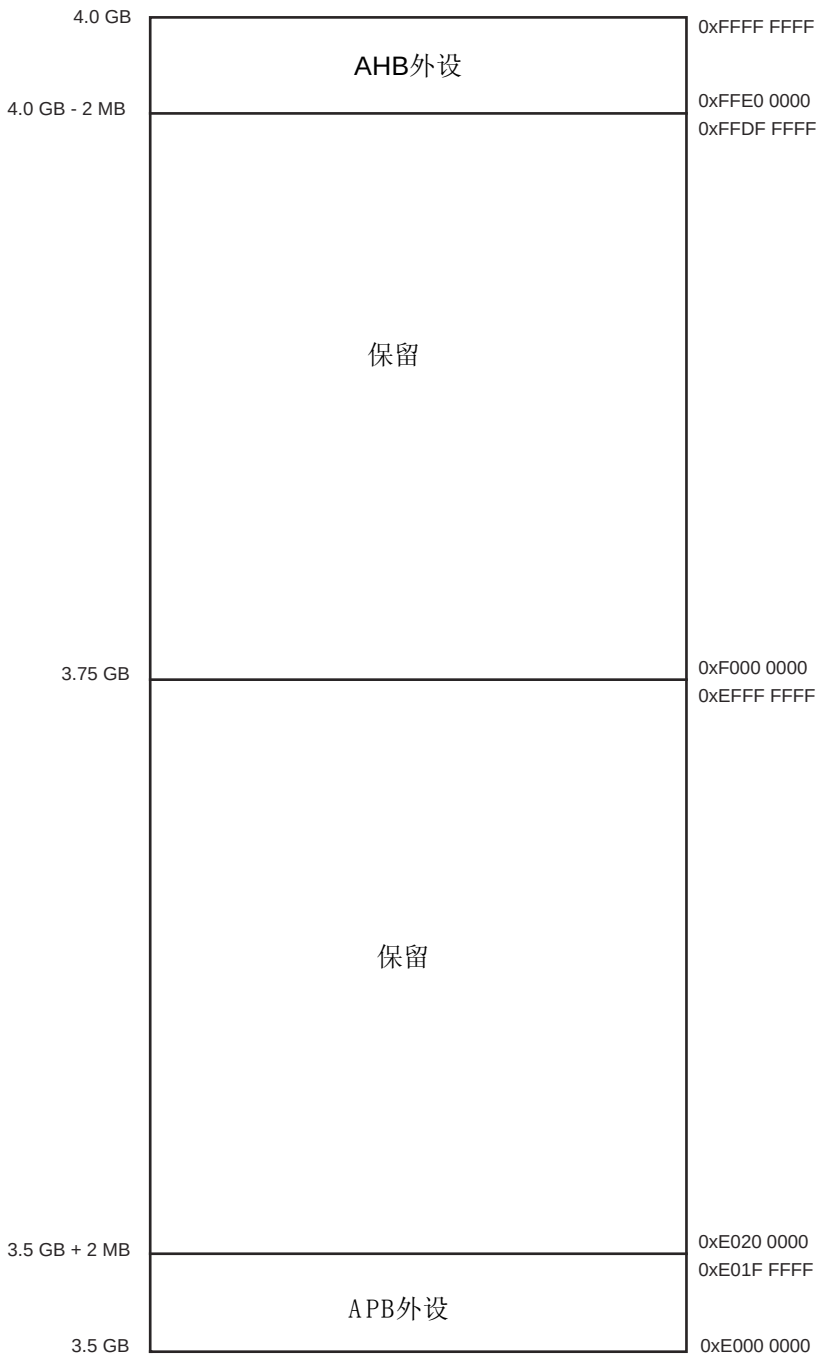


图 2　系统存储器映射



AHB 部分是 128×16kB 的范围（共 2MB）

APB 部分是 128×16kB 的范围（共 2MB）

图 3　外设存储器映射

图 3 到 4 和表 2 显示了从不同角度所观察到的外设地址空间。AHB 和 APB 外设区域都为 2M 字节，可各自分配最多 128 个外设。每个外设空间的规格都为 16k 字节。这样可简化每个外设的地址译码。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的硬件来进行小边界的字节（8 位）或半字（16 位）访问。不管字还是半字寄存器都是一次性访问。例如，不可能对一个字寄存器的最高字节执行单独的读或写操作。

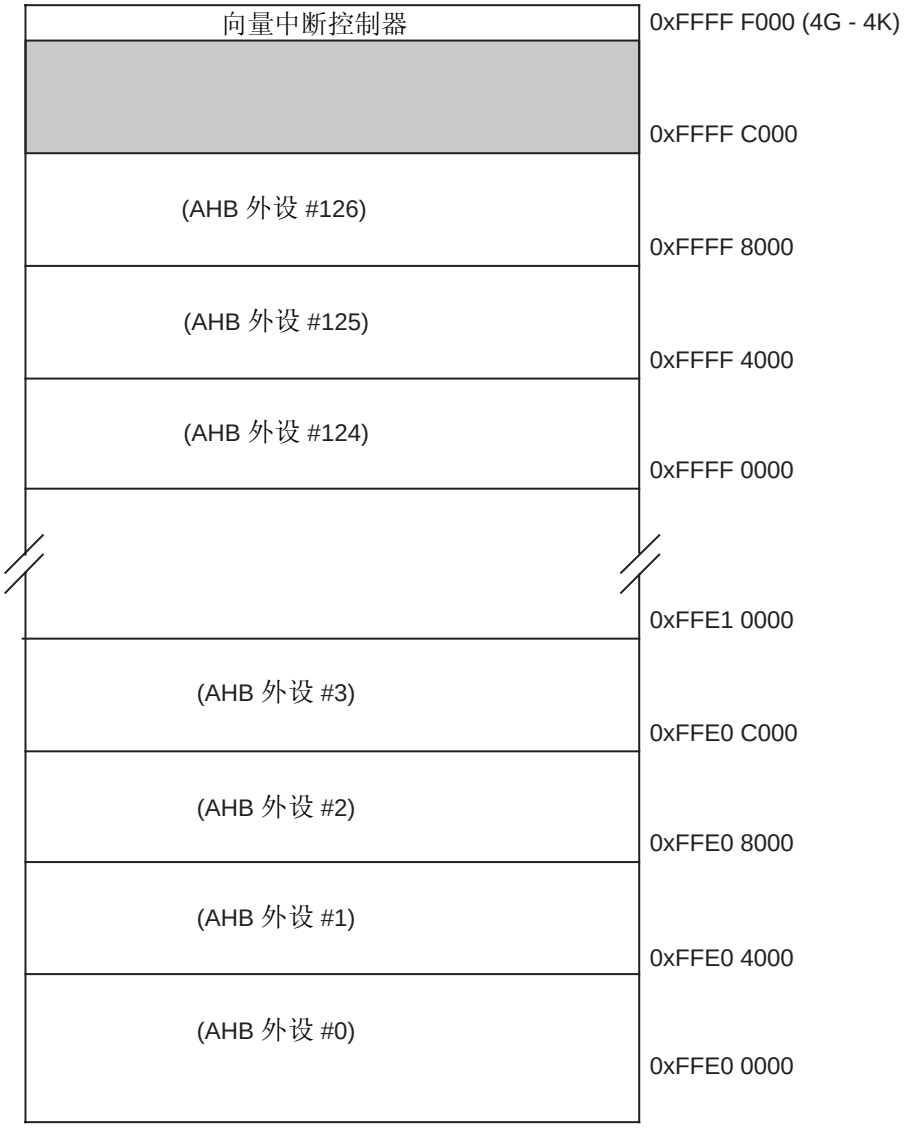


图 4　AHB 外设映射

表 2　APB 外设和基址

APB 外设	基址	外设名称
0	0xE000 0000	看门狗定时器
1	0xE000 4000	定时器 0
2	0xE000 8000	定时器 1
3	0xE000 C000	UART0
4	0xE001 0000	UART1
5	0xE001 4000	不使用
6	0xE001 8000	不使用
7	0xE001 C000	I^2C0
8	0xE002 0000	SPI0
9	0xE002 4000	RTC
10	0xE002 8000	GPIO

续上表

APB 外设	基址	外设名称
11	0xE002 C000	管脚连接模块
12	0xE003 0000	不使用
13	0xE003 4000	ADC
14－22	0xE003 8000 0xE005 8000	不使用
23	0xE005 C000	I^2C1
24	0xE006 0000	不使用
25	0xE006 4000	不使用
26	0xE006 8000	SSP
27	0xE006 C000	
28	0xE007 0000	定时器 3
29	0xE007 4000	定时器 4
30－126	0xE007 8000 0xE01F 8000	不使用
127	0xE01F C000	系统控制模块

2.2 LPC2101/02/03 存储器重新映射和引导模块

2.2.1 存储器映射概念和操作模式

LPC2101/02/03 的基本概念是：每个存储器组在存储器映射中都有一个“物理的”位置。它是一个地址范围，该范围内可写入程序代码。每一个存储器空间的容量都永久固定在同一个位置，这样就不需要将代码设计成在不同地址范围内运行。

由于 ARM7 处理器上的中断向量位置（地址 0x0000 0000~0x0000 001C，见下面的表 3），Boot Block 和 SRAM 空间的一小部分需要重新映射来实现在不同操作模式下对中断的使用，见表 4。中断的重新映射通过存储器映射控制特性来实现（详见 3.7 节“存储器映射控制”）。

表 3 ARM 异常向量位置

地址	异常
0x0000 0000	复位
0x0000 0004	未定义指令
0x0000 0008	软件中断
0x0000 000C	预取指中止（从存储器取指出错）
0x0000 0010	数据中止（数据访问存储器出错）
0x0000 0014	保留 **注：**在 ARM 文档中标识为保留，该位置被 Boot 装载程序用作有效的用户程序关键字。详见 19.4.2 节“有效用户代码的标准”。
0x0000 0018	IRQ
0x0000 001C	FIQ

表 4 LPC2101/02/03 存储器映射模式

模式	激活	用途
Boot 装载程序模式	硬件通过任何复位激活	Boot 装载程序**总是**在任何复位后执行。Boot Block 中断向量映射到存储器的底部以允许处理异常并在 Boot 装载过程中使用中断。
用户 Flash 模式	软件通过 Boot 代码激活	当在存储器中识别了一个有效的用户程序标识并且 Boot 装载程序操作未被执行时，由 Boot 装载程序启动。中断向量没有重新映射，它位于 Flash 存储器的底部。
用户 RAM 模式	软件通过用户程序激活	需要时由用户程序激活。中断向量重新映射到静态 RAM 的底部。

2.2.2 存储器的重新映射

为了与将来器件相兼容，整个 Boot Block 都被映射到片内存储器空间的顶端。在这种方式下，使用较大或较小的 Flash 模块都不需要改变 Boot Block（需要改变 Boot 装载程序自身的代码）的位置或改变 Boot Block 中断向量的映射。除了中断向量之外的存储器空间都保持固定的位置。图 5 所示为使用上述定义的模式映射的片内存储器。

存储器重新映射的部分允许在不同模式下处理中断，它包括中断向量区（32 字节）和额外的 32 字节，一共是 64 字节。重新映射的代码位置与地址 0x0000 0000~0x0000 003F 重叠。一个位于 Flash 存储器中的典型用户程序可以将整个 FIQ 处理程序放置在地址 0x0000 001C 而不需要考虑存储器的边界。包含在 SRAM、外部存储器和 Boot Block 中的向量必须包含跳转到实际中断处理程序的分支或者其它执行跳转到中断处理程序分支的指令。

选择这种配置有三个原因：

1. 使 Flash 存储器的 FIQ 处理程序不必考虑因为重新映射所导致的存储器边界问题。
2. 用来处理代码空间中段边界仲裁的 SRAM 和 Boot Block 向量的使用大大减少。
3. 为了提供空间来保存常量以得到在单字分支指令范围以外的跳转。

重新映射的存储器组（包括中断向量），除了重新映射的地址外，仍然继续出现在它们最初的位置。

有关重新映射及其举例详见 3.7 节“存储器映射控制”。

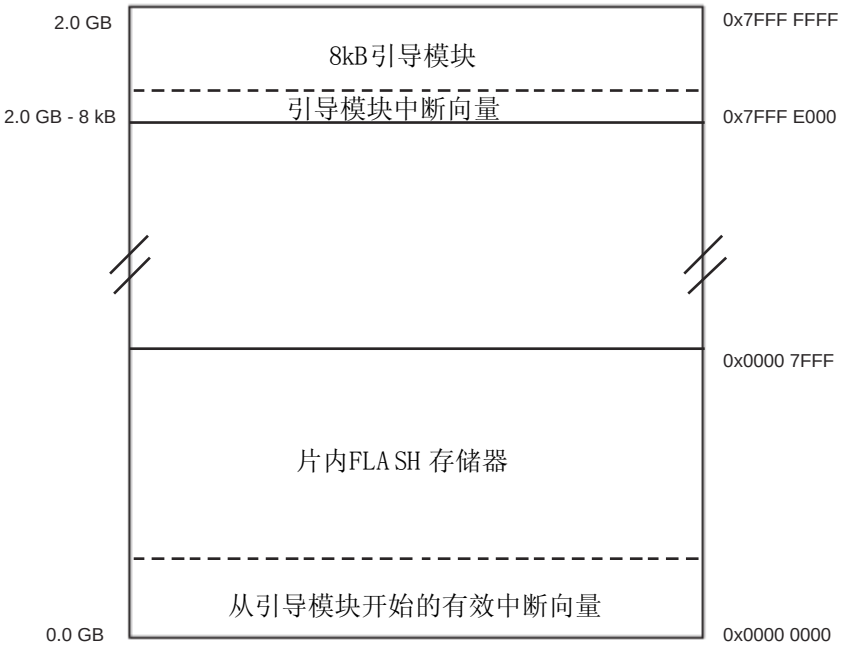


图 5　显示已重新映射和可重新映射区域的低字节存储器映射（带 32kB Flash 的 LPC2103）

2.3 预取指中止和数据中止异常

如果试图访问一个保留地址或未分配区域的地址，则 LPC2101/02/03 将产生适当的总线周期中止异常。这些区域包括：

- 特定的 ARM 器件所没有执行的存储器映射区域。对于 LPC2101/02/03，它们是：
  - －片内非易失性存储器与片内 SRAM 之间的地址空间，在图 2 中标为“保留地址空间”。对于 32kB Flash 器件来说，它们是 0x0000 8000 到 0x3FFF FFFF 的存储器地址空间；对于 16kB Flash 器件来说，它们是 0x0000 4000 到 0x3FFF FFFF 的存储器地址空间；对于 8kB Flash 器件来说，它们是 0x0000 2000 到 0x3FFF FFFF 的存储器地址空间。
  - －片内静态 RAM 与 Boot Block 之间的地址空间，在图 2 中标为“保留地址空间”。对于 8kB SRAM 器件，地址范围从 0x4000 2000 到 0x7FFF DFFF；对于 4kB SRAM 器件，地址范围从 0x4000 1000 到 0x7FFF DFFF；而对于 2kB SRAM 器件，地址范围从 0x4000 0800 到 0x7FFF DFFF。
  - －0x8000 0000 和 0xDFFF FFFF 之间的地址范围，标有“保留地址空间”。
  - －AHB 和 APB 空间的保留区域，见图 3。
- 未分配的 AHB 外设空间，见图 4。
- 未分配的 APB 外设空间，见表 2。

对于这些区域，对数据的访问和对指令的取指都会产生异常。另外，产生预取指中止异常用于映射到 AHB 或 APB 外设地址的任何指令取指。

在现有的 APB 外设地址空间内，对未定义地址的访问不会产生数据中止异常。每个外设内的地址译码被限制为外设内部需要区分的已定义寄存器。例如，对地址 0xE000 D000（UART0 空间内一个未定义的地址）的访问可能导致对定义在地址 0xE000 C000 处的寄存器进行访问。外设空间内的这种地址混淆在 LPC2101/02/03 文档中没有定义，并且它也不是

一个被 LPC2101/02/03 支持的特性。

需要注意的是，只有在试图执行从非法地址取指的指令时，ARM 才会将预取指中止标志与相关的指令（没有意义的指令）一起保存到流水线并对中止进行处理。当代码在非常靠近存储器边界执行时，这样防止了由预取指所导致的意外中止。

第3章 系统控制模块

3.1 系统控制模块功能汇总

系统控制模块包括几个系统特性和控制寄存器，这些寄存器具有众多与特定外设器件无关的功能。它们包括：

- 晶体振荡器
- 外部中断输入
- 各种系统控制和状态
- 存储器映射控制
- PLL
- 功率控制
- 复位
- APB 分频器
- 唤醒定时器

为了满足将来扩展的需要，每种类型的功能都有其自身的寄存器且不需要的位则定义为保留位。无关的功能不共用相同的寄存器地址。

3.2 管脚描述

表 5 所示为系统控制模块功能相关的管脚。

表 5 管脚汇总

管脚名称	管脚方向	管脚描述
X1	输入	**晶振输入**－振荡器和内部时钟发生器电路的输入
X2	输出	**晶振输出**－振荡器放大器的输出
EINT0	输入	**外部中断输入 0**－低电平/高电平有效或下降/上升沿的通用中断输入。该管脚可用于将处理器从空闲或掉电模式中唤醒。 可选择 P0.16 来执行 EINT0 功能。
EINT1	输入	**外部中断输入 1**－见上面的 EINT0 描述。 可选择 P0.14 来执行 EINT1 功能。 **注意：**复位后管脚 P0.14 上立即出现的低电平被看作是一个启动 ISP 命令处理器的外部硬件请求。有关 ISP 和串行 Boot Loader 的详细情况见 19.4 节。
EINT2	输入	**外部中断输入 2**－见上面的 EINT0 描述。 可选择 P0.7 和 P0.15 来执行 EINT2 功能。
$\overline{\text{RESET}}$	输入	**外部复位输入**－该管脚上的低电平将芯片复位，使 I/O 口和外设恢复其默认状态，并使处理器从地址 0x0000 0000 开始执行程序。

3.3 寄存器描述

所有寄存器不管规格大小都以字地址作为边界。这些寄存器的详细信息见相关功能的描述。

表 6 系统控制寄存器汇总

名称	描述	访问	复位值[1]	地址
外部中断				
EXTINT	外部中断标志寄存器	R/W	0	0xE01FC140
INTWAKE	中断唤醒寄存器	R/W	0	0xE01FC144
EXTMODE	外部中断方式寄存器	R/W	0	0xE01FC148
EXTPOLAR	外部中断极性寄存器	R/W	0	0xE01FC14C
存储器映射控制				
MEMMAP	存储器映射控制	R/W	0	0xE01FC040
锁相环				
PLLCON	PLL 控制寄存器	R/W	0	0xE01FC080
PLLCFG	PLL 配置寄存器	R/W	0	0xE01FC084
PLLSTAT	PLL 状态寄存器	RO	0	0xE01FC088
PLLFEED	PLL 馈送寄存器	WO	NA	0xE01FC08C
功率控制				
PCON	功率控制寄存器	R/W	0	0xE01FC0C0
PCONP	外设的功率控制	R/W	0x03BE	0xE01FC0C4
APB 分频器				
APBDIV	APB 分频器控制	R/W	0	0xE01FC100
复位				
RSIR	复位源识别寄存器	R/W	0	0xE01FC180
代码安全/调试				
CSPR	代码安全保护寄存器	RO	0	0xE01FC184
系统控制各种的寄存器				
SCS	系统控制和状态	R/W	0	0xE01FC1A0

[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。

3.4 晶体振荡器

LPC2101/02/03 板上振荡器电路仅支持 1MHz～25MHz 的外部晶振。如果片内 PLL 系统或引导装载程序被使用，那么输入时钟频率将被限制到 10MHz～25MHz。

振荡器输出频率称为 F_{OSC}，为了便于频率等式的书写及本文档的描述，ARM 处理器时钟频率称为 CCLK。F_{OSC} 和 CCLK 的值相同，除非 PLL 正在运行且被连接。详细内容和频率限制见 3.8 节“锁相环(PLL)”。

LPC2101/02/03 的振荡器可工作在两种模式下：从属模式和振荡模式。

从属模式下，输入时钟信号与一个 100pF 的电容（图 6 的 Cc，a 图）相连，其幅值至少为 200mVrms。该配置下的 X2 管脚不连接。如果选择从属模式，Fosc 信号（占空比为 50-50）

的频率被限制在 1MHz～50MHz。

振荡模式中使用的外部元件和模型见图 6 中的 b 和 c 图以及表 7。由于片内集成了反馈电阻，只需在外部连接一个晶体和电容 Cx_1、Cx_2 就可形成基本模式的振荡（基本频率用 L、C_L 和 Rs 来表示）。图 6 中 c 图的电容 Cp 是并联封装电容，其值不能大于 7pF。参数 Fc、C_L、Rs 和 Cp 都由晶体制造商提供。

如果选择器件的振荡器模式为板上振荡模式，那么 Fosc 时钟限制在 1MHz～30MHz。

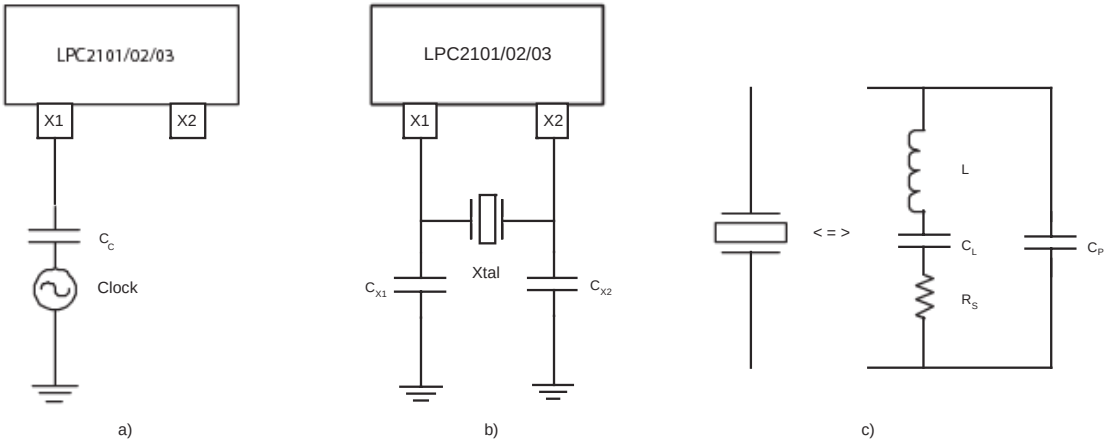


图 6　振荡器模式和模型：a)从属模式，b)振荡模式，c）外部晶体模型（用来评估 $C_{X1/X2}$ 的值）

表 7　振荡模式下 $C_{X1/X2}$ 的建议取值（晶体和外部元件参数）

基本振荡频率 F_{OSC}	晶体负载电容 C_L	最大晶体串联电阻 Rs	外部负载电容 Cx_1，Cx_2
1 MHz～5MHz	10pF	NA	NA
	20pF	NA	NA
	30pF	<300Ω	58pF，58pF
5 MHz～10MHz	10pF	<300Ω	18pF，18pF
	20pF	<300Ω	38pF，38pF
	30pF	<300Ω	58pF，58pF
10 MHz～15MHz	10pF	<300Ω	18pF，18pF
	20pF	<220Ω	38pF，38pF
	30pF	<140Ω	58pF，58pF
15 MHz～20MHz	10pF	<220Ω	18pF，18pF
	20pF	<140Ω	38pF，38pF
	30pF	<80Ω	58pF，58pF
20 MHz～25MHz	10pF	<160Ω	18pF，18pF
	20pF	<90Ω	38pF，38pF
	30pF	<50Ω	58pF，58pF
25 MHz～30MHz	10pF	<130Ω	18pF，18pF
	20pF	<50Ω	38pF，38pF
	30pF	NA	NA

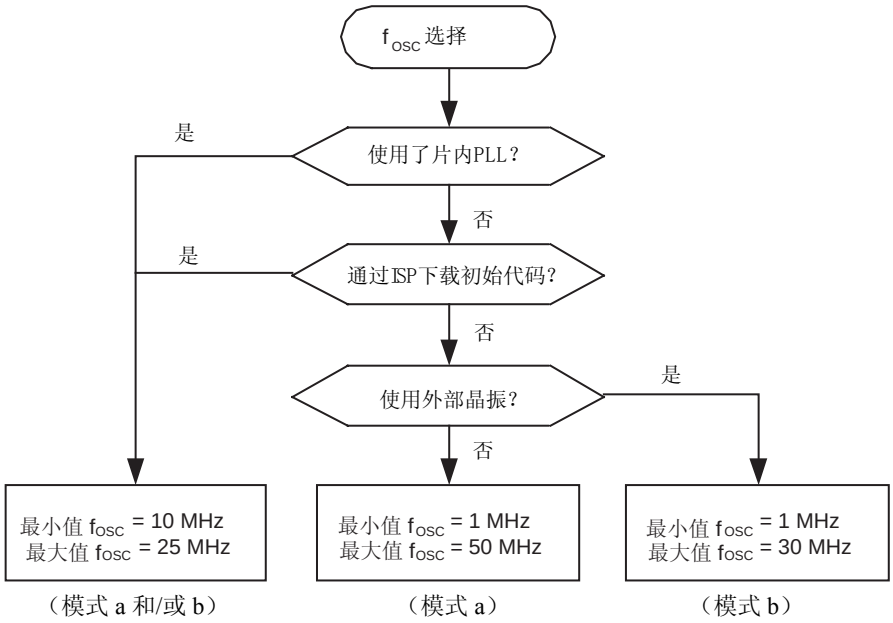


图 7 Fosc 的选择算法

3.5 外部中断输入

LPC2101/02/03 含有 3 个外部中断输入（作为可选的管脚功能）。当组合管脚时，外部事件可作为 3 个独立的中断信号处理。外部中断输入可用于将处理器从掉电模式中唤醒。

此外，所有 10 个捕获输入还可以用作外部中断而无需将器件从掉电模式中唤醒。

3.5.1 寄存器描述

外部中断功能具有 4 个相关的寄存器。EXTINT 寄存器包含中断标志且 EXTWAKE 寄存器包含使能唤醒位，可使能独立的外部中断输入将微控制器从掉电模式唤醒。EXTMODE 和 EXTPOLAR 寄存器指定电平和边沿激活参数。

表 8 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。见表 9。	R/W	0	0xE01F C140
INTWAKE	中断唤醒寄存器包含 4 个用于控制外部中断是否将处理器从掉电模式唤醒的使能位，见表 10。	R/W	0	0xE01F C144
EXTMODE	外部中断方式寄存器控制每个管脚的边沿或电平激活。	R/W	0	0xE01F C148
EXTPOLAR	外部中断极性寄存器控制由每个管脚的哪种电平或边沿来产生中断。	R/W	0	0xE01F 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。

3.5.2 外部中断标志寄存器（EXTINT - 0xE01F C140）

当一个管脚选择使用外部中断功能时，对应在 EXTPOLAR 和 EXTMODE 寄存器中的位选择的电平或边沿将置位 EXTINT 寄存器的中断标志。这样来向 VIC 提出中断请求，如

果管脚中断使能，则产生中断。

通过向 EXTINT 寄存器的位 EINT0～位 EINT2 写入 1 来将其清零。电平激活方式下，该操作只有在管脚处于无效状态时才有效。

一旦 EINT0～EINT2 中的一位被置位并开始执行相应的代码（处理唤醒和/或外部中断），该位必须清零。否则 EINT 管脚刚触发的事件以后将不会被识别。

重要：只要执行外部中断操作模式（也就是，激活电平/边沿）的变化（包括外部中断的初始化），那么 EXTINT 寄存器中相应的位必须被清零！详细内容请见 3.5.4 节“外部中断方式寄存器(EXTMODE – 0xE01F C148)”**和** 3.5.5 节“外部中断极性寄存器(EXTPOLAR – 0xE01F C14C)”。

例如，如果外部中断 0 管脚的低电平将系统从掉电模式唤醒，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 位仍保持置位状态，则后来唤醒掉电模式的任何操作都将失败。外部中断处理也不例外。

有关掉电模式的详细信息见后面章节的描述。

表 9 外部中断标志寄存器（EXTINT- 地址 0xE01F C140）位描述

位	名称	描述	复位值
0	EINT0	电平激活方式下，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；边沿激活方式下，如果管脚的 EINT0 功能被选用且管脚上出现所选边沿时，该位置位。 该位通过写入 1 清除，但电平激活方式下管脚处于有效状态的情况除外（例如，如果选择 EINT0 为低电平激活且低电平在相应的管脚上出现时，该位不能被清除；只有该管脚上的信号变为高电平时该位才能被清除）。	0
1	EINT1	电平激活方式下，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；边沿激活方式下，如果管脚的 EINT1 功能被选用且管脚上出现所选边沿时，该位置位。 该位通过写入 1 清除，但电平激活方式下管脚处于有效状态的情况除外。（例如，如果选择 EINT1 为低电平激活且低电平在相应的管脚上出现时，该位不能被清除；只有该管脚上的信号变为高电平时该位才能被清除）。	0
2	EINT2	电平激活方式下，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；边沿激活方式下，如果管脚的 EINT2 功能被选用且管脚上出现所选边沿时，该位置位。 该位通过写入 1 清除，但电平激活方式下管脚处于有效状态的情况除外。（例如，如果选择 EINT2 为低电平激活且低电平在相应的管脚上出现时，该位不能被清除；只有该管脚上的信号变为高电平时该位才能被清除）。	0
7:3	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

3.5.3 中断唤醒寄存器（INTWAKE - 0xE01F C144）

INTWAKE 寄存器中的使能位允许外部中断和其它源将处理器从掉电模式中唤醒。相关的 EINTn 功能必须映射到管脚才能实现掉电唤醒。但中断并不必要为了实现唤醒操作而在向量中断控制器中被使能。这样做的好处是允许外部中断输入将处理器从掉电模式唤醒，但不产生中断（只是简单地恢复操作），或者在掉电模式下使能中断而不会将处理器唤醒（这样，当应用中并不需要唤醒特性时，也不必关闭中断）。

为了使外部中断管脚成为将微控制器从掉电模式中唤醒的一种源，有必要清除外部中断标志寄存器中的相应位（见 3.5.2 节）。

表 10 中断唤醒寄存器（INTWAKE – 地址 0xE01F C144）位描述

位	名称	描述	复位值
0	EXTWAKE0	该位为 1 时，使能 EINT0 将处理器从掉电模式唤醒。	0
1	EXTWAKE1	该位为 1 时，使能 EINT1 将处理器从掉电模式唤醒。	0
2	EXTWAKE2	该位为 1 时，使能 EINT2 将处理器从掉电模式唤醒。	0
14:3	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
15	RTCWAKE	该位为 1 时，RTC 中断可将处理器从掉电模式唤醒。	0

3.5.4 外部中断方式寄存器（EXTMODE – 0xE01F C148）

EXTMODE 寄存器中的位用来选择每个 EINT 脚是电平或边沿激活。只有选择用作 EINT 功能（见 7.4 节）并已通过 VICIntEnable 寄存器（见 5.4.4 节）使能的管脚才能产生外部中断功能的中断（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：当某个中断在 VICIntEnable 寄存器中被禁能时，软件应该只改变 EXTMODE 寄存器中相应位的值，并且在中断使能（初始化）或重新使能前，软件向 EXTINT 写入 1 来清除 EXTINT 位，EXTINT 位可通过改变激活方式来置位。

表 11 外部中断方式寄存器（EXTMODE – 地址 0xE01F C148）位描述

位	名称	值	描述	复位值
0	EXTMODE0	0 1	EINT0 使用电平激活。 EINT0 使用边沿激活。	0
1	EXTMODE1	0 1	EINT1 使用电平激活。 EINT1 使用边沿激活。	0
2	EXTMODE2	0 1	EINT2 使用电平激活。 EINT2 使用边沿激活。	0
7:3	-	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

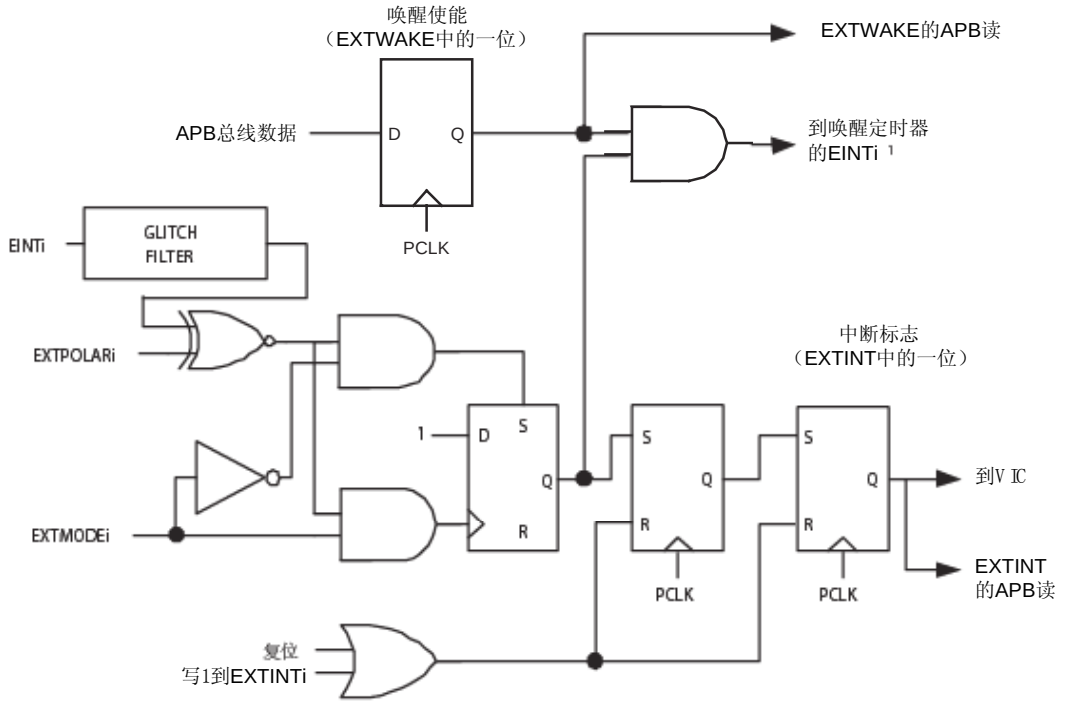
3.5.5 外部中断极性寄存器（EXTPOLAR – 0xE01F C14C）

在电平激活方式中，EXTPOLAR 寄存器的位选择相应管脚是高电平或低电平有效。在边沿激活方式中，EXTPOLAR 寄存器的位选择管脚上升沿或下降沿有效。只有选择用作 EINT 功能（见 7.4 节）并已在 VICIntEnable 寄存器中（见 5.4.4 节）使能的管脚才能产生外部中断功能的中断（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：当某个中断在 VICIntEnable 寄存器中禁能时，软件应该只改变 EXTPOLAR 寄存器中相应位的值。并且在中断使能（初始化）或重新使能前，软件向 EXTINT 写入 1 来清除 EXTINT 位，EXTINT 位可通过改变极性来置位。

表 12 外部中断极性寄存器（EXTPOLAR – 地址 0xE01F C14C）位描述

位	名称	值	描述	复位值
0	EXTPOLAR0	0 1	EINT0 为低电平或下降沿有效（由 EXTMODE0 决定）。 EINT0 为高电平或上升沿有效（由 EXTMODE0 决定）	0
1	EXTPOLAR1	0 1	EINT1 为低电平或下降沿有效（由 EXTMODE1 决定）。 EINT1 为高电平或上升沿有效（由 EXTMODE1 决定）。	0
2	EXTPOLAR2	0 1	EINT2 为低电平或下降沿有效（由 EXTMODE2 决定）。 EINT2 为高电平或上升沿有效（由 EXTMODE2 决定）。	0
7:3	—	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA



(1) 见图 10 “含有唤醒定时器的复位模块框图”。

图 8 外部中断逻辑

3.6 其它系统控制

不适合外设或其它寄存器的控制 LPC2101/02/03 操作的某些部分在这里分组。

3.6.1 系统控制和状态标志寄存器 (SCS – 0xE01F C1A0)

表 13 系统控制和状态标志寄存器 (SCS – 地址 0xE01F C1A0)位描述

位	名称	值	描述	复位值
0	GPIO0M	0	GPIO 端口 0 模式选择。 通过 APB 地址访问 GPIO 端口 0，与先前的 LCP2000 器件兼容。	0
		1	高速 GPIO 在 GPIO 端口 0 上使能，通过片内存储器范围的地址来访问。该模式包括端口屏蔽特性，该特性在 GPIO 一章中描述。	
31:1	-		保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

3.7 存储器映射控制

存储器映射控制用于改变从地址 0x0000 0000 开始的中断向量的映射。这允许运行在不同存储器空间中的代码对中断进行控制。

3.7.1 存储器映射控制寄存器（MEMMAP – 0xE01F C040）

只要需要异常处理，微控制器就会取异常对应地址上的指令，异常对应地址在表 3“ARM 异常向量位置”中描述。MEMMAP 寄存器确定填写该表的数据源。

表 14 存储器映射控制寄存器（MEMMAP – 地址 0xE01F C040）位描述

位	名称	值	描述	复位值
1:0	MAP	00 01 10 11	Boot 装载程序模式。中断向量被重新映射到 Boot Block。 用户 Flash 模式。中断向量不重新映射，它位于 Flash 中。 用户 RAM 模式。中断向量被重新映射到静态 RAM。 保留。不使用该选项。 **警告**：该值不正确的设定会导致器件的错误操作。	00
7:2	—	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

3.7.2 存储器映射控制的使用注意事项

存储器映射控制只从处理 ARM 异常（中断）必需的 3 个数据源（每个 64 字节）中选择一个使用。

例如，每当产生一个软件中断请求，ARM 内核就从 0x0000 0008 处取出 32 位数据（见表 3“ARM 异常向量位置”）。这就意味着当 MEMMAP[1:0]＝10（用户 RAM 模式）时，从 0x0000 0008 的读数/取指是对 0x4000 0008 单元进行操作。当 MEMMAP[1:0]=00（Boot 装载程序模式）时，从 0x0000 0008 的读数/取指是对 0x7FFF E008 单元的数据进行操作（Boot Block 从片内 Boot 装载程序重新映射）。

3.8 锁相环（PLL）

PLL 接受的输入时钟频率范围仅为 10MHz~25MHz。输入频率通过使用一个电流控制振荡器（CCO）倍增到范围 10MHz~60MHz。倍频器可以是从 1 到 32 的整数（实际上，由于 CPU 最高频率的限制，倍频值不能高于 7）。CCO 的操作频率范围为 156MHz~320MHz，因此在环中有一个额外的分频器在 PLL 提供所需要的输出频率时使 CCO 保持在其频率范围内。输出分频器可设置为 2，4，8 或 16 分频来产生输出时钟。由于输出分频器的最小值为 2，它保证了 PLL 输出有 50%的占空比。图 9 为 PLL 的方框图。

PLL 的激活由 PLLCON 寄存器控制。PLL 倍频器和分频器的值由 PLLCFG 寄存器控制。为了防止 PLL 参数发生意外改变或 PLL 失效，对这两个寄存器进行了保护。当提供芯片时钟时，由于芯片的所有操作（包括看门狗定时器）都依赖于 PLL，因此 PLL 设置的意外改变将导致微控制器执行不期望的动作。对它们的保护由一个类似于操作看门狗定时器的馈送(feed)序列来实现。详情请参阅 PLLFEED 寄存器的描述。

PLL 在芯片复位和进入掉电模式时被关闭并旁路。PLL 只能通过软件使能。程序必须在配置并激活 PLL 后等待其锁定，然后作为时钟源连接到 PLL。

3.8.1 寄存器描述

PLL 由表 15 所示的寄存器进行控制。以下有更详细的描述。

警告：PLL 值的不正确设定会导致器件的错误操作！

表 15 PLL 寄存器

通用名称	描述	访问	复位值[1]	地址
PLLCON	PLL 控制寄存器。保持寄存器用于更新 PLL 控制位。写入该寄存器的值在有效的 PLL 馈送序列执行之前不起作用。	R/W	0	0xE01F C080
PLLCFG	PLL 配置寄存器。保持寄存器用于更新 PLL 配置值。写入该寄存器的值在有效的 PLL 馈送序列执行之前不起作用。	R/W	0	0xE01F C084
PLLSTAT	PLL 状态寄存器。PLL 控制和配置信息的读回寄存器。如果曾对 PLLCON 或 PLLCFG 执行了写操作，但没有产生 PLL 馈送序列，这些值将不会反映 PLL 的当前状态。读取该寄存器提供了控制 PLL 和 PLL 状态的实际值。	RO	0	0xE01F C088
PLLFEED	PLL 馈送寄存器。该寄存器使能装载 PLL 控制和配置信息，该配置信息从 PLLCON 和 PLLCFG 寄存器装入实际影响 PLL 操作的映像寄存器中。	WO	NA	0xE01F C08C

[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。

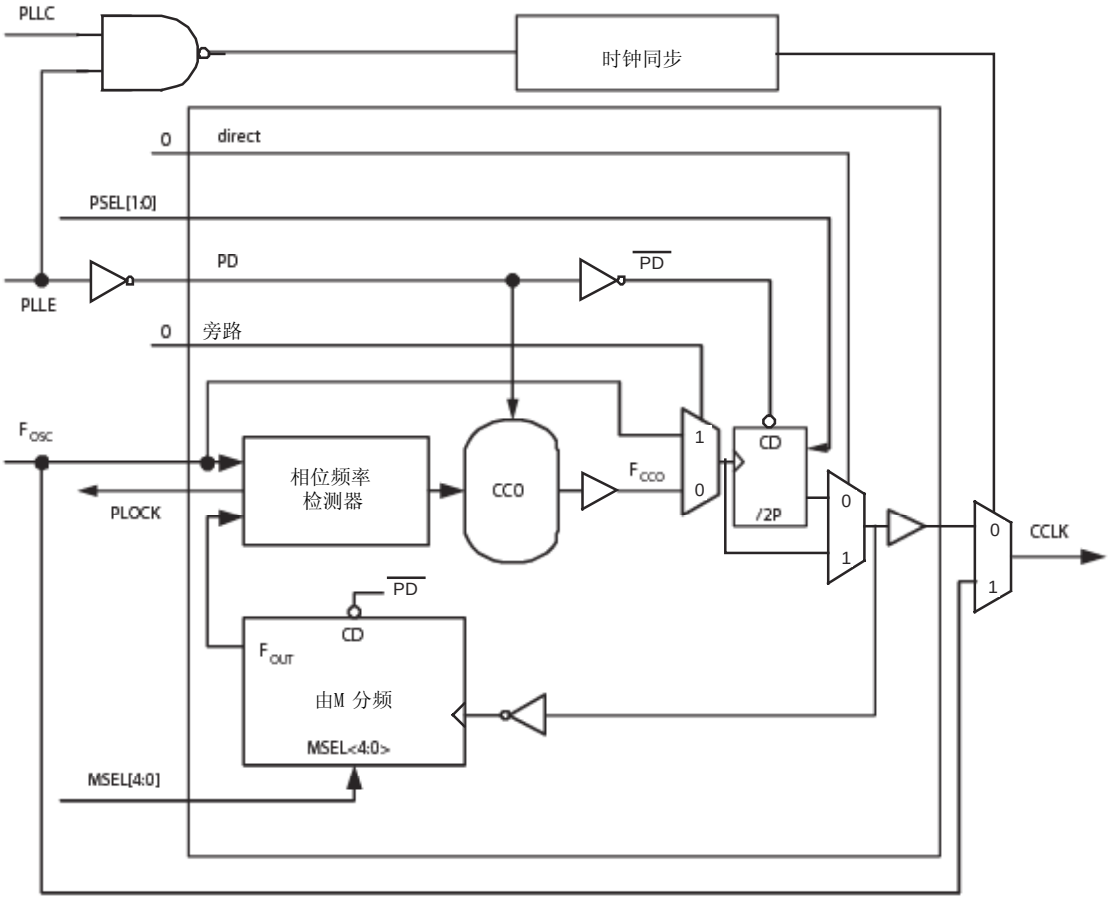


图 9 PLL 方框图

3.8.2 PLL 控制寄存器（PLLCON – 0xE01F C080）

PLLCON 寄存器包含使能和连接 PLL 的位。使能 PLL 锁定到当前倍频器和分频器值的设定频率上。连接 PLL 将使处理器和所有片内功能都根据 PLL 输出时钟来运行。对 PLLCON 寄存器的更改只有在对 PLLFEED 寄存器执行了正确的 PLL 馈送序列后才生效（见 3.8.7 节“PLL 馈送寄存器（PLLFEED - 0xE01F C08C）”和 3.8.3 节“PLL 配置寄存器(PLLCFG – 0xE01F C084)”的描述）。

表 16 PLL 控制寄存器（PLLCON – 地址 0xE01F C080）位描述

位	名称	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。见 PLLSTAT 寄存器，表 18。	0
1	PLLC	**PLL 连接**。当 PLLC 和 PLLE 都为 1 并且在有效的 PLL 馈送后，连接 PLL 作为微控制器的时钟源。否则，微控制器直接使用振荡器时钟。见 PLLSTAT 寄存器的描述，表 18。	0
7:2	–	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

PLL 在作为时钟源之前必须进行设置、使能并锁定。将振荡器时钟切换到 PLL 输出或反过来操作时，内部电路对操作进行同步以确保不会产生干扰。硬件不能确保 PLL 在连接之前锁定或在 PLL 失去锁定时自动断开连接。在 PLL 失去锁定的情况下，振荡器很可能已经变得不稳定，这样断开 PLL 也挽救不了这种状况。

3.8.3 PLL 配置寄存器（PLLCFG – 0xE01F C084）

PLLCFG 寄存器包含 PLL 倍频器和分频器值。在执行正确的 PLL 馈送序列之前改变 PLLCFG 寄存器的值不会生效（见 3.8.7 节“PLL 馈送寄存器(PLLFEED – 0xE01F C08C)”的描述）。PLL 频率和倍频器以及分频器值的计算，详见 3.8.9 节“PLL 频率计算”。

表 17 PLL 配置寄存器（PLLCFG – 地址 0xE01F C084）位描述

位	名称	描述	复位值
4:0	MSEL	PLL 倍频器值。在 PLL 频率计算中其值为 M。 **注**：有关 MSEL 正确值的选取，见 3.8.9 节“PLL 频率计算”。	0
6:5	PSEL	PLL 分频器值。在 PLL 频率计算中其值为 P。 **注**：有关 PSEL 正确值的选取，见 3.8.9 节“PLL 频率计算”。	0
7	–	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

3.8.4 PLL 状态寄存器（PLLSTAT - 0xE01F C088）

从只读 PLLSTAT 寄存器读出的是正在使用的真实 PLL 参数和状态。PLLSTAT 可能与 PLLCON 和 PLLCFG 中的值不同，这是因为没有执行正确的 PLL 馈送序列，这两个寄存器中的值并未生效(详见 3.8.7 节“PLL 馈送寄存器（PLLFEED – 0xE01F C08C）”的描述)。

表 18 PLL 状态寄存器（PLLSTAT – 地址 0xE01F C088）位描述

位	名称	描述	复位值
4:0	MSEL	读出的 PLL 倍频器值。这是 PLL 当前使用的值。	0
6:5	PSEL	读出的 PLL 分频器值。这是 PLL 当前使用的值。	0
7	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
8	PLLE	读出的 PLL 使能位。当该位为 1 时，PLL 处于激活状态；为 0 时，PLL 关闭。当进入掉电模式时，该位自动清零。	0
9	PLLC	读出的 PLL 连接位。当 PLLC 和 PLLE 都为 1 时，PLL 作为微控制器的时钟源被连接；当 PLLC 或 PLLE 位为 0 时，PLL 被旁路，微控制器直接使用振荡器时钟。当激活掉电模式时，该位自动清零。	0
10	PLOCK	反映 PLL 的锁定状态。为 0 时，PLL 未锁定；为 1 时，PLL 锁定到指定的频率。	0
15:11	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

3.8.5 PLL 中断

PLLSTAT 寄存器中的 PLOCK 位连接到中断控制器。这使软件可以打开 PLL 并继续执行其它功能，不需要等待 PLL 锁定。当发生中断时(PLOCK=1)，可以连接 PLL 并禁止中断。关于如何使能和禁能 PLL 中断的细节，请见 5.4.4 节“中断使能寄存器(VICIntEnable – 0xFFFF F010)”和 5.4.5 节 “中断使能清除寄存器(VICIntEnClear – 0xFFFF F014)”。

3.8.6 PLL 模式

PLLE 和 PLLC 的组合见表 19。

表 19 PLL 控制位组合

PLLC	PLLE	PLL 功能
0	0	PLL 被关闭并断开连接。CCLK 等于未更改的时钟输入。
0	1	PLL 被激活但是尚未连接。PLL 可在 PLOCK 有效后连接。
1	0	与 00 组合相同。这样消除了 PLL 已连接但没有使能的可能性。
1	1	PLL 已激活并已连接。CCLK/系统时钟的时钟源是 PLL。

3.8.7 PLL 馈送寄存器（PLLFEED – 0xE01F C08C）

必须将正确的馈送序列写入 PLLFEED 寄存器才能使 PLLCON 和 PLLCFG 寄存器的更改生效。馈送序列如下：

1. 将值 0xAA 写入 PLLFEED
2. 将值 0x55 写入 PLLFEED。

这两个写操作的顺序必须正确，而且必须是连续的 APB 总线周期。后面一个要求表明在执行 PLL 馈送操作时必须禁止中断。如果写入的值不正确或没有满足前两个条件，对 PLLCON 或 PLLCFG 寄存器的更改都不会生效。

表 20 PLL 馈送寄存器（PLLFEED – 地址 0xE01F C08C）位描述

位	名称	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

3.8.8 PLL 和掉电模式

掉电模式会自动关闭并断开 PLL。从掉电模式唤醒不会自动恢复 PLL 的设定，PLL 的恢复必须由软件来完成。通常，一个将 PLL 激活并等待锁定，然后将 PLL 连接的子程序可以在任何中断服务程序的开始调用，该中断服务程序也可以由于唤醒而被调用。有一点非常重要，那就是不要试图在掉电唤醒之后简单地执行馈送序列来重新启动 PLL。这会在 PLL 锁定建立之前同时使能并连接 PLL。

3.8.9 PLL 频率计算

PLL 等式使用下列参数：

表 21 确定 PLL 频率的要素

要素	描述
F_{OSC}	晶体振荡器/外部振荡器的频率
F_{CCO}	PLL 电流控制振荡器的频率
CCLK	PLL 输出频率（也是处理器的时钟频率）
M	PLLCFG 寄存器中 MSEL 位的 PLL 倍频器值
P	PLLCFG 寄存器中 PSEL 位的 PLL 分频器值

PLL 输出频率（当 PLL 激活并连接时）由下式得到：

$CCLK = M \times F_{OSC}$ 或 $CCLK = F_{CCO} / (2 \times P)$

CCO 频率可由下式得到：

$F_{CCO} = CCLK \times 2 \times P$ 或 $F_{CCO} = F_{OSC} \times M \times 2 \times P$

PLL 输入和设定必须满足下面的条件：

F_{OSC} 的范围：10MHz~25MHz

CCLK 的范围：10MHz~F_{max}（微控制器的最大允许频率－由内置的系统微控制器决定）

F_{CCO} 的范围：156MHz~320MHz

3.8.10 确定 PLL 设定的过程

如果一个特定的应用使用 PLL，它的配置必须依照下面的原则：

1. 选择需要的处理器操作频率（CCLK）。这可以根据处理器的整体要求、UART 波特率特定设置的支持等因素来决定。记住外围器件的时钟频率可以低于处理器频率（见 3.11 节“APB 分频器”描述）。
2. 选择振荡器频率（F_{OSC}）。CCLK 一定是 F_{OSC} 的整数（非小数）倍。
3. 计算 M 值以配置 MSEL 位。$M = CCLK/F_{OSC}$，M 的取值范围为 1~32。在 PLLCFG 中，写入 MSEL 位的值为 M-1(见表 23)。

4. 选择 P 值以配置 PSEL 位，使 F_{CCO} 在定义的频率限制范围内，F_{CCO} 可通过前面的等式计算。P 必须是 1, 2, 4 或 8 其中的一个。写入 PLLCFG 中 PSEL 位的值为：00 表示 P=1；01 表示 P=2；10 表示 P=4；11 表示 P=8 (见表 22)。

表 22 PLL 分频器值

PSEL 位（PLLCFG 位[6:5]）	P 值
00	1
01	2
10	4
11	8

表 23 PLL 倍频器值

MSEL 位（PLLCFG 位[4:0]）	M 值
00000	1
00001	2
00010	3
00011	4
...	...
11110	31
11111	32

3.8.11 PLL 配置举例

例：配置 PLL 的应用程序

系统设计要求 Fosc＝10MHz 和要求 CCLK＝60MHz。

根据这些要求，可得出 M=CCLK / Fosc＝60MHz/10MHz＝6。因此，M-1＝5 将写入 PLLCFG[4:0]。

P 值可由 P=Fcco/(CCLK×2)得出，Fcco 必须在 156MHz~320MHz 的范围内。假设 Fcco 取最低频率 156MHz，则 P=156MHz/(2×60MHz)＝1.3。Fcco 取最高频率可得出 P=2.67。因此，P 的唯一解决方案是同时满足 Fcco 最低和最高频率的值，见表 22 中列出的 P=2。所以，将使用 PLLCFG[6:5]=1。

3.9 功率控制

LPC2101/02/03 支持两种节电模式：空闲模式和掉电模式。在空闲模式下，指令的执行被挂起直到发生复位或中断为止。外设功能在空闲模式下继续操作并可产生中断使处理器恢复运行。空闲模式使处理器、存储器系统和相关控制器以及内部总线不再消耗功率。

在掉电模式下，振荡器关闭，这样芯片没有任何内部时钟。处理器状态和寄存器、外设寄存器以及内部 SRAM 值在掉电模式下保持且芯片管脚的逻辑电平保持静态。复位或特定的不需要时钟仍能工作的中断可终止掉电模式并使芯片恢复正常运行。由于掉电模式使芯片所有的动态操作都挂起，因此芯片的功耗降低到几乎为零。

如果 RTC 在进入掉电模式时与其外部的 32KHz 振荡器一起运行，则操作可使用 RTC 的中断恢复（见 17.4.1 节“RTC 中断”）。

掉电和空闲模式的进入必须与程序的执行同步进行。通过中断唤醒掉电或空闲模式恢复程序的执行，这种执行的方式不会使指令丢失、不完整或重复。从掉电模式唤醒将在 3.12 节“唤醒定时器”中作进一步讨论。

外设的功率控制特性允许独立关闭应用中不需要的外设，这样进一步降低了功耗。

3.9.1 寄存器描述

功率控制功能包含两个寄存器，如表 24 所示。更详细的内容见后面的描述。

表 24 功率控制寄存器

名称	描述	访问	复位值[1]	地址
PCON	功率控制寄存器。该寄存器包含微控制器两种节电模式的控制位。见表 25。	R/W	0x00	0xE01F C0C0
PCONP	外设寄存器的功率控制。该寄存器包含使能和禁止单个外设功能的控制位。该寄存器可使未被使用的外设不消耗功率。	R/W	0x0018 17BE	0xE01F C0C4

[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。

3.9.2 功率控制寄存器（PCON – 0xE01F COCO）

PCON 寄存器包含两个位。置位其中一个位，将会进入掉电或空闲模式。如果两位都置位，则进入掉电模式。

表 25 功率控制寄存器（PCON – 地址 0xE01F COCO）位描述

位	名称	描述	复位值
0	IDL	空闲模式－当该位置位时，处理器时钟停止，但片内外围功能保持工作状态。外设或外部中断源所产生的任何中断都会使处理器恢复运行。	0
1	PD	掉电模式－当该位置位时，振荡器和所有片内时钟都停止。外部中断所产生的唤醒条件可使振荡器重新启动并使 PD 位清零，处理器恢复运行。	0
7:2	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

3.9.3 外设功率控制寄存器（PCONP – 0xE01F COC4）

PCONP 寄存器允许将所选的外设功能关闭以实现节电的目的。这可通过关断特定外围模块的时钟源来实现。有少数外设功能不能被关闭（看门狗定时器、GPIO、管脚连接模块和系统控制模块）。某些外设，特别是包含模拟功能的外设可消耗功率，它们的操作无需时钟。这些外设包含独立的禁能控制位，通过它们来关闭电路以降低功耗。PCONP 中的每个位都控制一个外设。每个位所对应的外设编号请见 2.1 节“存储器映射”部分中 APB 外设映射的表 2“APB 外设和基址”。

如果外设控制位为 1，则该外设被使能。如果外设位为 0，则该外设禁能以保存功率。例如，如果位 19 为 1，则 I^2C1 接口被使能。如果位 19 为 0，则 I^2C1 接口被禁能。

重要：只要该外设在 PCONP 寄存器中被使能，就有可能对外设寄存器进行有效的读取和写入！

表 26 外设功率控制寄存器（PCONP – 地址 0xE01F C0C4）位描述

位	名称	描述	复位值
0	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
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
6:5	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
7	PCI2C0	I^2C0 接口功率/时钟控制位。	1
8	PCSPI	SPI 接口功率/时钟控制位。	1
9	PCRTC	RTC 功率/时钟控制位。	1
10	PCSPI	SSP 接口功率/时钟控制位。	1
11	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
12	PCAD	A/D 转换器 0 (ADC0) 功率/时钟控制位。 **注：**清零该位前先清零 ADCR 寄存器的 PDN 位，且该位应当在置位 PDN 前被置位。	1
18:13	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
19	PCI2C1	I^2C1 接口功率/时钟控制位。	1
27:20	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
28	PCTIM2	定时器/计数器 2 功率/时钟控制位	1
29	PCTIM3	定时器/计数器 3 功率/时钟控制位	1
31:30	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

3.9.4 功率控制注意事项

每次复位后，PCONP 寄存器的值设置成使能所有接口和外围功能（受 PCONP 控制的）。因此，除了对外围功能相关的寄存器进行配置外，用户应用程序不需要访问 PCONP 寄存器以便使用片内的任何外围功能启动。

对于以节电为目的的系统，只有在应用中用到的外围功能对应在 PCONP 寄存器中的位为 1。寄存器的其它“保留”位或当前无需使用的外围功能对应在寄存器中的位都必须清零。

3.10 复位

LPC2101/02/03 有两个复位源：$\overline{\text{RESET}}$ 管脚和看门狗复位。$\overline{\text{RESET}}$ 管脚为施密特触发输入管脚，带有一个额外的干扰滤波器。任何复位源提供的芯片复位都会启动唤醒定时器（详见 3.12 节“唤醒定时器”的描述），复位将保持有效直至外部复位撤除，振荡器开始运行。当计数达到一个固定个数的时钟时，片内电路完成其初始化。复位、振荡器以及唤醒定时器之间的关系见图 10。

复位干扰滤波器使处理器可以忽略非常短的外部复位脉冲并决定 $\overline{\text{RESET}}$ 保证芯片复位所必须保持的最短时间。$\overline{\text{RESET}}$ 一旦有效，它就只有当晶振运行稳定并且 LPC2101/02/03 的 X1 脚上出现适当的信号时才能撤除。如果晶振子系统使用的是外部晶体，上电后 $\overline{\text{RESET}}$ 脚的信号必须保持 10ms。对于晶振已经稳定运行且 X1 脚上已出现稳定信号时出现的复位，$\overline{\text{RESET}}$ 脚的信号只需保持 300ns。

当内部复位撤除时，处理器从地址 0 开始运行，此处为从 Boot Block 映射的复位向量。此时所有的处理器和外设寄存器都被初始化为预先确定的值。

外部复位和内部复位有一些小的区别。外部复位使特定管脚的值被锁存以实现配置。外部电路无法确定内部复位什么时候发生进而对特定管脚的值进行配置，因此那些锁存在内部复位过程中不会重新装载。在外部复位时对管脚 26(RTCK)进行检测（见 6.2 节和 7.4 节）。当复位后执行引导装载程序时，片内引导装载程序将对 P0.14 进行检测（见 19.4 节）。

芯片复位可以发生在 Flash 编程或擦除操作过程中。Flash 存储器会中断正在进行的操作并使 CPU 复位延迟到内部 Flash 高电压降低后才完成。

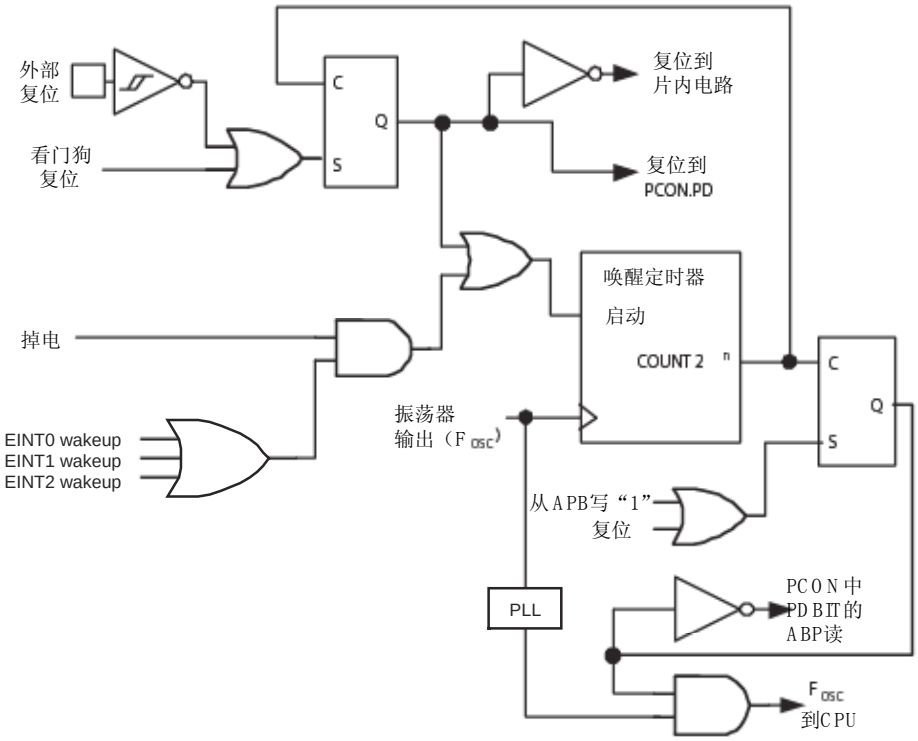


图 10 包括唤醒定时器的复位方框图

3.10.1 复位源识别寄存器（RSIR – 0xE01F C180）

每个复位源都在该寄存器中对应一位。置位这些位中的任何一个将使其读出时为 0。4 个复位源相互之间的关系见下表的描述。

表 27 复位源识别寄存器（RSIR – 地址 0xE01F C180）位描述

位	名称	描述	复位值
0	POR	上电复位(POR)事件设置该位，并清除 RSIR 寄存器中的其它位。但是如果 POR 信号撤销后另一个复位信号（如外部复位）仍然保持有效时，则这个复位信号对应的位置位。POR 位不受其它任何其它复位源的影响。	见文中描述
1	EXTR	$\overline{\text{RESET}}$ 信号有效时该位置位。该位由 POR 来清除，但不受 WDT 复位的影响。	见文中描述
2	WDTR	当看门狗定时器溢出和看门狗方式寄存器的 WDTRESET 位为 1 时，该位置位。该位可被其它任何一个复位源清除。	见文中描述
7:3	—	保留。用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

3.11 APB 分频器

APB 分频器决定处理器时钟（CCLK）与外设器件所使用的时钟（PCLK）之间的关系。APB 分频器有两个用途。

第一个是通过 APB 总线为外设提供所需的 PCLK 时钟以便外设可以在为 ARM 处理器而选择的速度下工作。为了实现此目的，APB 总线可以降低到 1/2 或 1/4 处理器时钟速率。由于 APB 总线必须在上电后正常工作（并且如果由于 APB 分频器控制器位于 APB 总线上而使上电时 APB 总线不工作，其时序就不能改变），APB 总线在复位时默认的状态是以 1/4 速度运行。

APB 分频器的第二个用途是在应用不需要任何外设全速运行时使功耗降低。

APB 分频器与振荡器和处理器时钟的连接见图 11。由于 APB 分频器连接到 PLL 输出，PLL 在空闲模式下保持有效（如果 PLL 处于运行状态）。

3.11.1 寄存器描述

只有一个寄存器用于控制 APB 分频器。

表 28 APB 分频器寄存器映射

名称	描述	访问	复位值[1]	地址
APBDIV	控制 APB 时钟速率与处理器时钟之间的关系	R/W	0x00	0xE01F C100

[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。

3.11.2 APBDIV 寄存器（APBDIV - 0xE01F C100）

APB 分频器寄存器包含两个位，可以设定 3 个分频值，详见表 29。

表 29 APB 分频器寄存器（APBDIV – 地址 0xE01F C100）位描述

位	名称	值	描述	复位值
1:0	APBDIV	00	APB 总线时钟为处理器时钟的 1/4。	00
		01	APB 总线时钟与处理器时钟相同。	
		10	APB 总线时钟为处理器时钟的 1/2。	
		11	保留。将该值写入 APBDIV 寄存器无效（保留原来的设定）。	
7:2	—	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

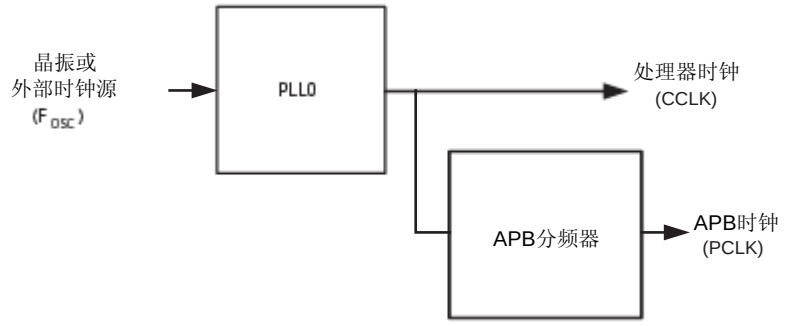


图 11 APB 分频器连接

3.12 唤醒定时器

唤醒定时器的用途是确保振荡器和芯片所需要的其它模拟功能在处理器开始执行指令之前能够正确工作。这在上电、所有类型的复位以及任何原因所导致上述功能关闭时非常重要。由于振荡器和其它功能在掉电模式下关闭，使处理器从掉电模式中唤醒都必须使用唤醒定时器。

唤醒定时器通过检测晶振是否能可靠地开始代码的执行来对其进行监视。当给芯片加电或某个事件使芯片退出掉电模式时，振荡器需要一段时间来产生足够振幅的信号驱动时钟逻辑。时间的长度取决于许多因素，包括 V_{DD} 的上升速率（上电时）、晶振的类型及其电气特性（如果使用石英晶振）、任何其它外部电路（例如电容）和振荡器在现有环境下自身的特性。

一旦检测到一个时钟，唤醒定时器则对 4096 个时钟计数，这段时间可使片内电路进行初始化。当片内电路初始化完毕时，如果外部复位已撤除，处理器开始执行指令。当系统使用外部时钟源（与晶振连接的管脚相反）时，需要考虑振荡器的启动延时可能很短甚至没有。唤醒定时器的设计确保了芯片所需要的任何其它功能在程序运行之前都能够进行操作。

任何复位、外部中断 EINT2:0 和 RTC 中断（如果 RTC 工作在其自身的 RTCX1-2 管脚的振荡器下）都可使微控制器退出掉电模式。如果一个中断使能产生唤醒并且所选中断事件出现，那么必须启动振荡器唤醒。实际的中断（如果有）在唤醒定时器停止后产生，由向量中断控制器进行处理。

但是，LPC2101/02/03 的复用管脚（见 6.2 节和 7.4 节）允许其它外设起作用，使器件退出掉电模式。下面的管脚功能对允许 RI1/EINT2 相关事件来产生中断。

要使器件进入掉电模式并允许总线或管脚上的一个或多个事件能使其恢复正常操作，软件应该对管脚的外部中断功能重新编程，选择中断合适的方式和极性以及掉电模式。唤醒时软件应恢复管脚复用的外围功能。

总的来说，在 LPC2101/02/03 上，唤醒定时器根据晶振执行最短时间的复位，并且只要定时器从掉电模式中唤醒或任何类型的复位产生时激活。

3.13 代码安全 vs.调试

开发应用中非常需要 LPC2101/02/03 的调试和跟踪功能。在一个应用后面的使用过程中，保护应用代码以防对手或竞争者盗用变得更加重要。LPC2101/02/03 的下述特性允许应用控制代码是否被调试或被保护以防盗用。

有关代码读保护的详细信息请参考 19.7 节“代码读保护(CRP)”。

第4章 存储器加速模块（MAM）

4.1 简介

当 LPC2101/02/03 运行 Flash 存储器的代码时，器件内部的 MAM 模块极大地提高了 ARM 处理器的性能，只需要使用一个简单的 Flash 组就可实现。

4.2 操作

存储器加速模块（MAM）将需要的下一个 ARM 指令锁存以防止 CPU 取指暂停。与以前的其它器件使用 2 个 Flash 组相比，LPC2101/02/03 只使用一组 Flash 存储器。Flash 组包含 3 个 128 位的缓冲区：预取指缓冲区、分支跟踪缓冲区和数据缓冲区。当预取指缓冲区或分支跟踪缓冲区不能满足一次指令取指的需要，并且此行的预取指还没有启动时，ARM 在启动 128 位行的取指时暂停。如果预取指已经启动但还未完成，则 ARM 暂停的时间会更短一些。预取指在 Flash 结束前面的访问后立即启动，除非被数据访问终止。预取指行被 Flash 模块锁存，但 MAM 不能在预取指缓冲区中捕获该行，直到 ARM 内核给出预取指开始的地址。如果内核给出的地址与预取指地址不同，则预取指行丢弃。

每个预取指缓冲区和分支跟踪缓冲区包含 4 个 32 位 ARM 指令或 8 个 16 位 Thumb 指令。在连续执行代码时，通常预取指缓冲区包含当前指令和含有该指令的整个 Flash 行。

MAM 利用 LPROT[0]行来区分指令和数据访问。代码和数据访问分别使用独立的 128 位缓冲区。每 4 个连续 32 位代码中的 3 个代码或数据访问都在缓冲区中完成，无需访问 Flash（每 8 个连续 16 位代码中的 7 个，每 16 个连续字节访问中的 15 个）。第 4 个（第 8 个，第 16 个）连续数据的访问必须访问 Flash，终止正在执行的任何预取指。Flash 数据访问结束后，重新启动已执行的预取指。

Flash 读操作的时序可编程，其描述见本节后面的内容。

这样，当 CPU 时钟周期大于或等于 1/4 的 Flash 访问时间时，连续指令的执行不会影响代码取指。其中用于程序分支的平均时间相对较短（少于 25％），在 ARM 代码中，利用 ARM 指令（不是 Thumb）的条件执行特性，这个时间会变得最短。这种条件执行通常用来防止不必要的较短的前向分支。

分支和其它程序流的变化导致前面所讲述的连续指令取指出现中断。分支跟踪缓冲区捕获发生非连续中断的行。如果相同的分支再次出现，则从分支跟踪缓冲区内取出下条指令。当分支超出了预取指和分支跟踪缓冲区内容时，需要终止几个时钟来装载分支跟踪缓冲区。这样，不会再出现指令取指延时，直到一个新的和不同的分支出现。

4.3 MAM 模块

存储器加速器模块(MAM)分成以下几个功能块：

- 1 个 Flash 地址锁存和 1 个增量器功能用于形成预取指地址。
- 1 个 128 位的预取指缓冲区及其相关的地址锁存和比较器。
- 1 个 128 位的分支跟踪缓冲区及其相关的地址锁存和比较器。
- 1 个 128 位的数据缓冲区及其相关的地址锁存和比较器。

- 控制逻辑
- 等待逻辑

图 12 所示为存储器加速器模块数据通路的一个简化框图。

在下面的描述中，“取指”一词表示 ARM 发出的一个直接的 Flash 读请求。“预取指”一词表示对当前处理器取指地址之外的地址执行 Flash 读操作。

4.3.1 Flash 存储器组

LPC2101/02/03 MAM 含有一个 Flash 存储器组。

Flash 编程操作不受 MAM 的控制，而是作为一个独立的功能进行处理。“boot block”扇区包含作为应用程序的一部分调用的 Flash 编程算法和一个可对 Flash 存储器进行串行编程的装载程序。

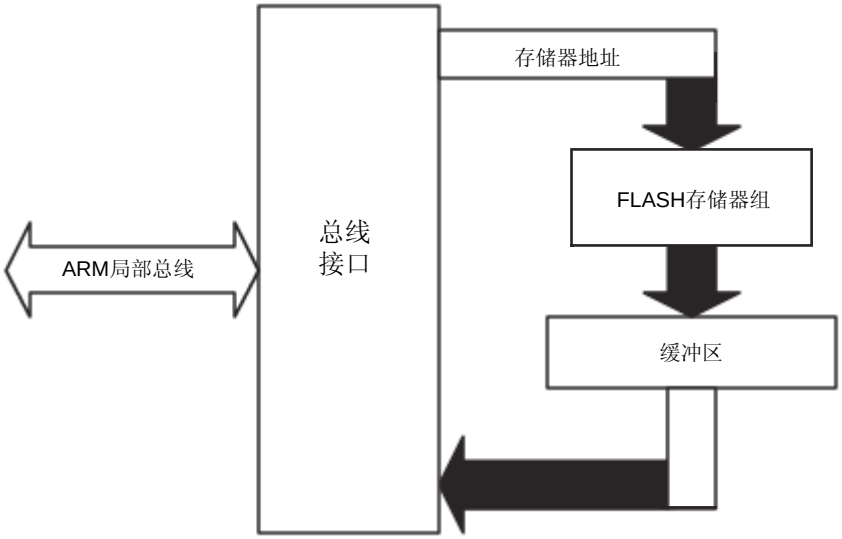


图 12 存储器加速器模块(MAM)的简化框图

4.3.2 指令锁存和数据锁存

代码和数据的访问由存储器加速器模块分别进行处理。每个缓冲区（预取指、分支跟踪和数据缓冲区）都有 1 套相关的 128 位锁存、15 位地址锁存和 15 位比较器。每个 128 位锁存保存 4 个代码字（4 条 ARM 指令或 8 条 Thumb 指令）。每个缓冲区还有 32 个 4:1 多路复用器，用来从 128 位行中选择所需的字。

对数据锁存中没有的数据进行访问会导致读取 Flash 的 4 个数据字，它们由数据锁存所捕获。使用数据锁存加速了连续数据的访问，但对于随机数据访问几乎没什么影响。

4.3.3 Flash 编程问题

由于在编程和擦除操作过程中不允许访问 Flash 存储器，那么如果在 Flash 模块忙时存储器请求访问 Flash 地址，MAM 就必须强制 CPU 等待（这通过声明 ARM7TDMI-S 局部总线信号 CLKEN 来实现）。在某些情况下，代码执行的延迟会导致看门狗超时。用户必须注意到这种可能性并采取措施来确保在编程或擦除 Flash 存储器时不会出现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，LPC2101/02/03 MAM 使锁存在 Flash 编程

或擦除操作开始时自动失效。在 Flash 操作结束后，任何对 Flash 地址的读操作将启动新的取指。

4.4 MAM 的操作模式

MAM 定义了 3 种操作模式，可以在性能和可预测性之间进行选择：

模式 0：MAM 关闭。所有存储器请求都会导致 Flash 的读操作（见下面的注 2）。无指令预取指。

模式 1：MAM 部分使能。如果数据可用，则从保持锁存区执行连续的指令访问。指令预取指使能。非连续的指令访问启动 Flash 读操作（见下面的注 2）。这意味着所有的转移指令都会导致对存储器的取指。由于缓冲的数据访问时序很难预测并且非常依赖于所处的状况，因此所有数据操作都会导致 Flash 读操作。

模式 2：MAM 完全使能。任何存储器请求（代码或数据），如果其值已经包含在其中一个保持锁存当中，那么从缓冲区执行该代码或数据的访问。指令预取指使能。Flash 读操作用于启动指令的预取指和相应保持锁存中所没有的代码或数据值。

表 30　MAM 响应的不同类型的程序访问

程序存储器请求类型	MAM 模式		
	0	1	2
连续访问，数据位于 MAM 锁存当中	启动取指[2]	使用锁存的数据[1]	使用锁存的数据[1]
连续访问，数据不在 MAM 锁存当中	启动取指	启动取指[1]	启动取指[1]
非连续访问，数据位于 MAM 锁存当中	启动取指[2]	启动取指[1],[2]	使用锁存的数据[1]
非连续访问，数据不在 MAM 锁存当中	启动取指	启动取指[1]	启动取指[1]

[1] 指令预取指在模式 1 和 2 中使能。

[2] 只要锁存的数据可用，MAM 则使用锁存的数据，但模仿 Flash 读操作的时序。这样虽然使用相同的执行时序，但却降低了功耗。将 MAMTIM 中的取指时间设置为 1 个时钟可关闭 MAM。

表 31　MAM 响应的不同类型的数据和 DMA 访问

数据存储器请求类型	MAM 模式		
	0	1	2
连续访问，数据位于 MAM 锁存当中	启动取指[1]	启动取指[1]	使用锁存的数据
连续访问，数据不在 MAM 锁存当中	启动取指	启动取指	启动取指
非连续访问，数据位于 MAM 锁存当中	启动取指[1]	启动取指[1]	使用锁存的数据
非连续访问，数据不在 MAM 锁存当中	启动取指	启动取指	启动取指

[1] 只要锁存的数据可用，MAM 则使用锁存的数据，但模仿 Flash 读操作的时序。这样虽然使用相同的执行时序，但却降低了功耗。将 MAMTIM 中的取指时间设置为 1 个时钟可关闭 MAM。

4.5 MAM 配置

在复位后，MAM 默认为禁止状态。软件可以随时将存储器访问加速打开或关闭。这样就可使大多数应用程序以最高速度运行，而当要求更精确时序时某些功能可以在较慢但更可预测的速度下运行。

4.6 寄存器描述

所有寄存器不管规格如何，都以字地址为边界。详细的寄存器内容见各个功能的描述。

表 32 MAM 寄存器汇总

名称	描述	访问	复位值[1]	地址
MAMCR	存储器加速器模块控制寄存器。决定 MAM 的操作模式，也就是，使能 MAM 性能增强的程度，见表 33。	R/W	0x0	0xE01F C000
MAMTIM	存储器加速器模块定时控制。决定 Flash 存储器取指所使用的时钟个数(1 到 7 个处理器时钟)。	R/W	0x07	0xE01F C004

[1] 复位值仅指已使用位中保存的数据，不包括保留位的内容。

4.7 MAM 控制寄存器（MAMCR – 0xE01F C000）

两个配置位选择 MAM 的 3 种操作模式，见表 33。在复位后，MAM 功能被禁止。改变 MAM 操作模式会导致 MAM 所有的保持锁存内容无效，因此需要执行新的 Flash 读操作。

表 33 MAM 控制寄存器（MAMCR – 地址 0xE01F C000）位描述

位	名称	值	描述	复位值
1:0	MAM_mode_control	00 01 10 11	MAM 功能被禁止 MAM 功能部分使能 MAM 功能完全使能 保留。不在应用中使用。	0
7:2	—	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

4.8 MAM 定时寄存器（MAMTIM – 0xE01F C004）

MAM 定时寄存器决定使用多少个 CCLK 周期访问 Flash 存储器。这样可调整 MAM 时序使其匹配处理器操作频率。Flash 访问时间可以从 1 到 7 个时钟。单个时钟的 Flash 访问实际上将 MAM 从定时计算中移除。这种情况下可以选择 MAM 模式对功耗进行优化。

表 34 MAM 定时寄存器（MAMTIM – 地址 0xE01F C004）位描述

位	名称	值	描述	复位值
2:0	MAM_fetch_cycle_timing	000 001 010 011 100 101 110 111	0－保留 1－MAM 取指周期为 1 个处理器时钟（CCLK）。 2－MAM 取指周期为 2 个处理器时钟（CCLK）。 3－MAM 取指周期为 3 个处理器时钟（CCLK）。 4－MAM 取指周期为 4 个处理器时钟（CCLK）。 5－MAM 取指周期为 5 个处理器时钟（CCLK）。 6－MAM 取指周期为 6 个处理器时钟（CCLK）。 7－MAM 取指周期为 7 个处理器时钟（CCLK）。 **警告**：这里列出的是这些位设置 MAM Flash 取指操作的过程。不正确的设定会导致器件的错误操作。	07
7:3	-	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

4.9 MAM 使用注意事项

当改变 MAM 定时值时，必须先通过向 MAMCR 写入 0 来关闭 MAM，然后将新值写入 MAMTIM。最后，将需要的操作模式的对应值（1 或 2）写入 MAMCR，再次打开 MAM。

对于低于 20MHz 的系统时钟，MAMTIM 设定为 001。对于 20MHz 到 40MHz 之间的系统时钟，建议将 Flash 访问时间设定为 2 CCLK，而在高于 40MHz 的系统时钟下，建议使用 3 CCLK。

第5章 向量中断控制器（VIC）

5.1 特性

- ARM PrimeCell ™ 向量中断控制器
- 32 个中断请求输入
- 16 个向量 IRQ 中断
- 16 个优先级，可动态分配给中断请求
- 软件中断产生

5.2 描述

向量中断控制器（VIC）具有 32 个中断请求输入，可将其编程分为 3 类：FIQ、向量 IRQ 和非向量 IRQ。可编程分配机制意味着不同外设的中断优先级可以动态分配并调整。

快速中断请求（FIQ）要求具有最高优先级。如果分配给 FIQ 的请求多于 1 个，VIC 将中断请求“相或”后向 ARM 处理器产生 FIQ 信号。当只有一个中断被分配为 FIQ 时可实现最短的 FIQ 等待时间，因为 FIQ 服务程序只要简单地启动器件的处理就可以了。但如果分配给 FIQ 级的中断多于 1 个，FIQ 服务程序从 VIC 中读出一个字来识别产生中断请求的 FIQ 中断源是哪一个。

向量 IRQ 具有中等优先级。该级别可分配 32 个请求中的 16 个。32 个请求中的任意一个都可分配到 16 个向量 IRQ slot 中的任意一个，其中 slot0 具有最高优先级，而 slot15 则为最低优先级。

非向量 IRQ 的优先级最低。

VIC 将所有向量和非向量 IRQ“相或”向 ARM 处理器产生 IRQ 信号。IRQ 服务程序可通过读取 VIC 的一个寄存器立即启动并跳转到相应地址。如果有任意一个向量 IRQ 发出请求，VIC 则提供最高优先级请求 IRQ 服务程序的地址，否则提供所默认程序的地址。该默认程序由所有非向量 IRQ 共用。默认程序可读取另一个 VIC 寄存器以确定哪个 IRQ 被激活。

VIC 中所有的寄存器都为字寄存器。不支持字节和半字的读和写操作。

关于向量中断控制器的其它信息请参阅 ARM PrimeCell ™ 向量中断控制器（PL190）的相关文档。

5.3 寄存器描述

VIC 所包含的寄存器如表 35 所示。详细内容见后面的描述。

表 35 VIC 寄存器映射

名称	描述	访问	复位值[1]	地址
VICIRQStatus	IRQ 状态寄存器。该寄存器读出定义为 IRQ 并使能的中断请求的状态。	RO	0	0xFFFF F000
VICFIQStatus	FIQ 状态请求。该寄存器读出定义为 FIQ 并使能的中断请求的状态。	RO	0	0xFFFF F004
VICRawIntr	原始中断状态寄存器。该寄存器读出 32 个中断请求/软件中断的状态，不管中断是否使能或分类。	RO	0	0xFFFF F008
VICIntSelect	中断选择寄存器。该寄存器将 32 个中断请求的每个都分配为 FIQ 或 IRQ。	R/W	0	0xFFFF F00C
VICIntEnable	中断使能寄存器。该寄存器控制将 32 个中断请求和软件中断中的哪些使能为 FIQ 或 IRQ。	R/W	0	0xFFFF F010
VICIntEnClr	中断使能清零寄存器。该寄存器允许软件将中断使能寄存器中的一个或多个位清零。	WO	0	0xFFFF F014
VICSoftInt	软件中断寄存器。该寄存器的内容与 32 个不同外设的中断请求“相或”。	R/W	0	0xFFFF F018
VICSoftIntClear	软件中断清零寄存器。该寄存器允许软件将软件中断寄存器中的一个或多个位清零。	WO	0	0xFFFF F01C
VICProtection	保护使能寄存器。该寄存器允许特权模式下运行的软件对 VIC 寄存器进行有限的访问。	R/W	0	0xFFFF F020
VICVectAddr	向量地址寄存器。当发生一个 IRQ 中断时，IRQ 服务程序可读出该寄存器并跳转到读出的地址。	R/W	0	0xFFFF F030
VICDefVectAddr	默认向量地址寄存器。该寄存器保存了非向量 IRQ 的中断服务程序（ISR）地址。	R/W	0	0xFFFF F034
VICVectAddr0	向量地址 0 寄存器。向量地址寄存器 0-15 保存了 16 个向量 IRQ slot 的中断服务程序(ISR)地址。	R/W	0	0xFFFF F100
VICVectAddr1	向量地址 1 寄存器	R/W	0	0xFFFF F104
VICVectAddr2	向量地址 2 寄存器	R/W	0	0xFFFF F108
VICVectAddr3	向量地址 3 寄存器	R/W	0	0xFFFF F10C
VICVectAddr4	向量地址 4 寄存器	R/W	0	0xFFFF F110
VICVectAddr5	向量地址 5 寄存器	R/W	0	0xFFFF F114
VICVectAddr6	向量地址 6 寄存器	R/W	0	0xFFFF F118
VICVectAddr7	向量地址 7 寄存器	R/W	0	0xFFFF F11C
VICVectAddr8	向量地址 8 寄存器	R/W	0	0xFFFF F120
VICVectAddr9	向量地址 9 寄存器	R/W	0	0xFFFF F124
VICVectAddr10	向量地址 10 寄存器	R/W	0	0xFFFF F128
VICVectAddr11	向量地址 11 寄存器	R/W	0	0xFFFF F12C
VICVectAddr12	向量地址 12 寄存器	R/W	0	0xFFFF F130
VICVectAddr13	向量地址 13 寄存器	R/W	0	0xFFFF F134
VICVectAddr14	向量地址 14 寄存器	R/W	0	0xFFFF F138
VICVectAddr15	向量地址 15 寄存器	R/W	0	0xFFFF F13C

续上表

名称	描述	访问	复位值[1]	地址
VICVectCntl0	向量控制 0 寄存器。向量控制寄存器 0-15 分别控制 16 个向量 IRQ slot 中的一个。Slot0 优先级最高，而 Slot15 优先级最低。	R/W	0	0xFFFF F200
VICVectCntl1	向量控制 1 寄存器	R/W	0	0xFFFF F204
VICVectCntl2	向量控制 2 寄存器	R/W	0	0xFFFF F208
VICVectCntl3	向量控制 3 寄存器	R/W	0	0xFFFF F20C
VICVectCntl4	向量控制 4 寄存器	R/W	0	0xFFFF F210
VICVectCntl5	向量控制 5 寄存器	R/W	0	0xFFFF F214
VICVectCntl6	向量控制 6 寄存器	R/W	0	0xFFFF F218
VICVectCntl7	向量控制 7 寄存器	R/W	0	0xFFFF F21C
VICVectCntl8	向量控制 8 寄存器	R/W	0	0xFFFF F220
VICVectCntl9	向量控制 9 寄存器	R/W	0	0xFFFF F224
VICVectCntl10	向量控制 10 寄存器	R/W	0	0xFFFF F228
VICVectCntl11	向量控制 11 寄存器	R/W	0	0xFFFF F22C
VICVectCntl12	向量控制 12 寄存器	R/W	0	0xFFFF F230
VICVectCntl13	向量控制 13 寄存器	R/W	0	0xFFFF F234
VICVectCntl14	向量控制 14 寄存器	R/W	0	0xFFFF F238
VICVectCntl15	向量控制 15 寄存器	R/W	0	0xFFFF F23C

[1] 复位值仅指已使用位中保存的数据，不包括保留位的内容。

5.4 VIC 寄存器

这一节按照 VIC 逻辑中的使用顺序对 VIC 寄存器进行描述，该顺序为从那些与中断请求输入最密切的寄存器到那些由软件所使用的最抽象的寄存器。对大多数人来说，这也是在学习 VIC 时读取寄存器的最佳顺序。

5.4.1 软件中断寄存器（VICSoftInt - 0xFFFF F018）

在执行任何逻辑之前，将该寄存器的内容与 32 个不同外设的中断请求相或。

表 36 软件中断寄存器（VICSoftInt – 地址 0xFFFF F018）位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	TIMER3	TIMER2	-	-
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
位	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
符号	-	-	-	-	I2C1	AD0	-	EINT2
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

续上表

位	15	14	13	12	11	10	9	8
符号	EINT1	EINT0	RTC	PLL	SSP/SPI1	SPI0	I2C0	-
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
位	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
符号	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

表 37 软件中断寄存器（VICSoftInt－地址 0xFFFF F018）位描述

位	名称	值	描述	复位值
31:0	见 VICSoftInt 位分配表	0	不强制产生中断请求。向 VICSoftInt 写入 0 无效，见 VICSoftIntClear。（5.4.2 节）	0
		1	强制产生与该位相关的中断请求。	

5.4.2 软件中断清零寄存器（VICSoftIntClear - 0xFFFF F01C）

该寄存器在不需读取软件中断寄存器的情况下，可用软件清零软件中断寄存器中的一个或多个位。

表 38 软件中断清零寄存器（VICSoftIntClear – 地址 0xFFFF F01C）位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	TIMER3	TIMER2	-	-
访问	WO	WO	WO	WO	WO	WO	WO	WO
位	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
符号	-	-	-	-	I2C1	AD0	-	EINT2
访问	WO	WO	WO	WO	WO	WO	WO	WO
位	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
符号	EINT1	EINT0	RTC	PLL	SPI1/SSP	SPI0	I2C0	-
访问	WO	WO	WO	WO	WO	WO	WO	WO
位	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
符号	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT
访问	WO	WO	WO	WO	WO	WO	WO	WO

表 39 软件中断清零寄存器（VICSoftIntClear – 地址 0xFFFF F01C）位描述

位	名称	值	描述	复位值
31:0	见 VICSoftIntClear 位分配表	0	写入 0 不会影响 VICSoftInt 中的相应位。	0
		1	写入 1 清零软件中断寄存器的相应位，并解除强制的中断请求。	

5.4.3 原始中断状态寄存器（VICRawIntr - 0xFFFF F008）

这是一个只读寄存器。该寄存器读取所有 32 个中断请求和软件中断的状态，不管中断是否使能或分类。

表 40 原始中断状态寄存器（VICRawIntr – 地址 0xFFFF F008）位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	TIMER3	TIMER2	-	-
访问	RO	RO	RO	RO	RO	RO	RO	RO
位	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
符号	-	-	-	-	I2C1	AD0	-	EINT2
访问	RO	RO	RO	RO	RO	RO	RO	RO
位	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
符号	EINT1	EINT0	RTC	PLL	SSP/SPI1	SPI0	I2C0	-
访问	RO	RO	RO	RO	RO	RO	RO	RO
位	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
符号	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT
访问	RO	RO	RO	RO	RO	RO	RO	RO

表 41 原始中断状态寄存器（VICRawIntr – 地址 0xFFFF F008）位描述

位	名称	值	描述	复位值
31:0	见 VICRawIntr 位分配表	0 1	对应位的中断请求或软件中断未声明。 对应位的中断请求或软件中断声明。	0

5.4.4 中断使能寄存器（VICIntEnable - 0xFFFF F010）

这是可进行读/写访问的寄存器。该寄存器控制 32 个中断请求或软件中断的哪些分配为 FIQ 或 IRQ。

表 42 中断使能寄存器（VICIntEnable – 地址 0xFFFF F010）位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	TIMER3	TIMER2	-	-
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
位	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
符号	-	-	-	-	I2C1	AD0	-	EINT2
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
位	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
符号	EINT1	EINT0	RTC	PLL	SSP/SPI1	SPI0	I2C0	-
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
位	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
符号	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

表 43 中断使能寄存器（VICIntEnable – 地址 0xFFFF F010）位描述

位	名称	功能	复位值
31:0	见 VICIntEnable 位分配表	当读取该寄存器时，1 表示中断请求或软件中断使能为 FIQ 或 IRQ。当写该寄存器时，1 使能中断请求或软件中断分配为 FIQ 或 IRQ，0 无效。见 5.4.5 节“中断使能清零寄存器(VICIntEnClear - 0xFFFF F014)”以及下面的表 45 给出了中断禁止的方法。	0

5.4.5 中断使能清零寄存器（VICIntEnClear - 0xFFFF F014）

这是一个只写寄存器。该寄存器在不需要读取中断使能寄存器（见 5.4.4 节“中断使能寄存器(VICIntEnable – 0xFFFF F010)）的情况下，可用软件清零其中的一个或多个位。

表 44 软件中断清零寄存器（VICIntEnClear – 地址 0xFFFF F014）位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	TIMER3	TIMER2	-	-
访问	WO	WO	WO	WO	WO	WO	WO	WO
位	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
符号	-	-	-	-	I2C1	AD0	-	EINT2
访问	WO	WO	WO	WO	WO	WO	WO	WO
位	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
符号	EINT1	EINT0	RTC	PLL	SSP/SPI1	SPI0	I2C0	-
访问	WO	WO	WO	WO	WO	WO	WO	WO
位	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
符号	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT
访问	WO	WO	WO	WO	WO	WO	WO	WO

表 45 软件中断清零寄存器（VICIntEnClear – 地址 0xFFFF F014）位描述

位	名称	值	描述	复位值
31:0	见 VICIntEnClear 位分配表	0 1	写入 0 不影响中断使能寄存器中的位。 写入 1 清零中断使能寄存器中的对应位并禁止对应的中断请求。	0

5.4.6 中断选择寄存器（VICIntSelect - 0xFFFF F00C）

这是可进行读/写访问的寄存器。该寄存器将 32 个中断请求分别分配为 FIQ 或 IRQ。

表 46 中断选择寄存器（VICIntSelect – 地址 0xFFFF F00C）位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	TIMER3	TIMER2	-	-
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
位	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
符号	-	-	-	-	I2C1	AD0	-	EINT2
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

续上表

位	15	14	13	12	11	10	9	8
符号	EINT1	EINT0	RTC	PLL	SSP/SPI1	SPI0	I2C0	-
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
位	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
符号	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

表 47 中断选择寄存器（VICIntSelect – 地址 0xFFFF F00C）位描述

位	名称	值	描述	复位值
31:0	见 VICIntSelect 位分配表	0 1	对应的中断请求分配为 IRQ。 对应的中断请求分配为 FIQ。	0

5.4.7 IRQ 状态寄存器（VICIRQStatus - 0xFFFF F000）

这是一个只读寄存器。该寄存器读取使能并分配为 IRQ 的中断请求的状态，它不对向量和非向量 IRQ 进行区分。

表 48 IRQ 状态寄存器（VICIRQStatus – 地址 0xFFFF F000）位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	TIMER3	TIMER2	-	-
访问	RO	RO	RO	RO	RO	RO	RO	RO
位	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
符号	-	-	-	-	I2C1	AD0	-	EINT2
访问	RO	RO	RO	RO	RO	RO	RO	RO
位	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
符号	EINT1	EINT0	RTC	PLL	SSP/SPI1	SPI0	I2C0	-
访问	RO	RO	RO	RO	RO	RO	RO	RO
位	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
符号	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT
访问	RO	RO	RO	RO	RO	RO	RO	RO

表 49 IRQ 状态寄存器（VICIRQStatus – 地址 0xFFFF F000）位描述

位	名称	功能	复位值
31:0	见 VICIRQStatus 位分配表	读出为 1 的位表示对应位的中断请求使能，分配为 IRQ，并且声明。	0

5.4.8 FIQ 状态寄存器（VICFIQStatus - 0xFFFF F004）

这是一个只读寄存器。该寄存器读取使能并分配为 FIQ 的中断请求的状态。如果有超过一个请求分配为 FIQ，FIQ 服务程序可读取该寄存器来确定是哪一个（几个）请求被激活。

表 50 FIQ 状态寄存器（VICFIQStatus – 地址 0xFFFF F004）位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	TIMER3	TIMER2	-	-
访问	RO	RO	RO	RO	RO	RO	RO	RO
位	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
符号	-	-	-	-	I2C1	AD0	-	EINT2
访问	RO	RO	RO	RO	RO	RO	RO	RO
位	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
符号	EINT1	EINT0	RTC	PLL	SSP/SPI1	SPI0	I2C0	-
访问	RO	RO	RO	RO	RO	RO	RO	RO
位	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
符号	UART1	UART0	TIMER1	TIMER0	ARMCore1	ARMCore0	-	WDT
访问	RO	RO	RO	RO	RO	RO	RO	RO

表 51 FIQ 状态寄存器（VICFIQStatus – 地址 0xFFFF F004）位描述

位	名称	功能	复位值
31:0	见 VICFIQStatus 位分配表	该位读出为 1 表示对应位的中断请求使能，分配为 FIQ，并且声明。	0

5.4.9 向量控制寄存器 0-15（VICVectCntl 0-15 - 0xFFFF F200-23C）

这是可进行读/写访问的寄存器。每一个寄存器控制 16 个向量 IRQ slot 中的一个。Slot0 优先级最高，Slot15 优先级最低。注意在 VICVectCntl 寄存器中禁止一个向量 IRQ slot 不会禁止中断本身，中断只是变为非向量的形式。

表 52 向量控制寄存器（VICVectCntl 0-15 - 0xFFFF F200-23C）位描述

位	名称	描述	复位值
4:0	int_request/ sw_int_assig	分配给此向量 IRQ slot 的中断请求或软件中断的编号。作为一个良好的编程习惯，不要将把相同的中断编号分配给多于一个使能的向量 IRQ slot。但如果这样做了，当中断请求或软件中断使能，被分配为 IRQ 并声明时，会使用最低编号的 slot。	0
5	IRQslot_en	当该位为 1 时，向量 IRQ slot 使能，当分配的中断请求或软件中断使能，被分配为 IRQ 并声明时，可产生一个唯一的 ISR 地址。	0
31:6	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

5.4.10 向量地址寄存器 0-15（VICVectAddr0-15 - 0xFFFF F100-13C）

这是可进行读/写访问的寄存器。这些寄存器保存 16 个向量 IRQ slot 中断服务程序(ISR)的地址。

表 53 向量地址寄存器(VICVectAddr0-15 – 地址 0xFFFF F100-13C)位描述

位	名称	描述	复位值
31:0	IRQ_vector	当一个或多个中断请求或软件中断使能，分配为 IRQ 并声明，并且分配为向量 IRQ slot 的中断请求使能时，IRQ 服务程序读取向量地址寄存器－VICVectAddr（5.4.10 节）时会得到最高优先级 slot 寄存器的值。	0x0000 0000

5.4.11 默认向量地址寄存器（VICDefVectAddr - 0xFFFF F034）

这是可进行读/写访问的寄存器。这些寄存器保存非向量 IRQ 中断服务程序（ISR）的地址。

表 54 默认向量地址寄存器（VICDefVectAddr – 地址 0xFFFF F034）位描述

位	名称	描述	复位值
31:0	IRQ_vector	当一个 IRQ 服务程序读取向量地址寄存器（VICVectAddr），并且没有 IRQ slot 响应时，则返回该寄存器中的地址。	0x0000 0000

5.4.12 向量地址寄存器（VICVectAddr - 0xFFFF F030）

这是可进行读/写访问的寄存器。当发生一个 IRQ 中断时，IRQ 服务程序可读取该寄存器并跳转到读出的地址。

表 55 向量地址寄存器（VICVectAddr – 地址 0xFFFF F030）位描述

位	名称	描述	复位值
31:0	IRQ_vector	当任何分配给向量 IRQ slot 的中断请求或软件中断使能，分配为 IRQ 并声明时，读取该寄存器将返回最高优先级 slot（最低编号）在向量地址寄存器中的地址。否则返回默认向量地址寄存器中的地址。 由于写入该寄存器的设置值并不能在将来读出，因此，该寄存器应该在 ISR 快结束时写入，以便更新优先级硬件。	0x0000 0000

5.4.13 保护使能寄存器（VICProtection - 0xFFFF F020）

这是可进行读/写访问的寄存器。它通过运行在用户模式下的软件访问 VIC 寄存器。

表 56 保护使能寄存器（VICProtection – 地址 0xFFFF F020）位描述

位	名称	值	描述	复位值
0	VIC_access	0 1	VIC 寄存器可在用户模式或特权模式下访问。 VIC 寄存器只能在特权模式下访问。	0
31:1	—	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

5.5 中断源

表 57 列出了每一个外设功能的中断源。每个外围设备都有一条中断线连接到向量中断控制器，但有些可能拥有几个内部中断标志。单个中断标志也有可能代表一个以上的中断源。

表 57 连接到向量中断控制器(VIC)的中断源

模块	标志	VIC 通道#和 Hex 掩码
WDT	看门狗中断（WDINT）	0 0x0000 0001
-	仅保留给软件中断	1 0x0000 0002
ARM 内核	Embedded ICE, DbgCommRx	2 0x0000 0004
ARM 内核	Embedded ICE, DbgCommTx	3 0x0000 0008
定时器 0	匹配 0-2（MR0, MR1, MR2） 捕获 0-2（CR0, CR1, CR2）	4 0x0000 0010
定时器 1	匹配 0-3（MR0, MR1, MR2, MR3） 捕获 0-3（CR0, CR1, CR2, CR3）	5 0x0000 0020
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）	6 0x0000 0040
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） Modem 状态中断（MSI）	7 0x0000 0080
-	保留	8 0x0000 0100
I^2C0	SI（状态改变）	9 0x0000 0200
SPI0	SPI0 中断标志（SPI0F） 模式错误（MODF）	10 0x0000 0400
SPI1 （SSP）	Tx FIFO 至少一半为空（TXRIS） Rx FIFO 至少一半为满（RXRIS） 接收超时条件（RTRIS） 接收溢出（RORRIS）	11 0x0000 0800
PLL	PLL 锁定（PLOCK）	12 0x0000 1000
RTC	计数器增加（RTCCIF） 报警（RTCALF）	13 0x0000 2000
系统控制	外部中断 0（EINT0）	14 0x0000 4000
	外部中断 1（EINT1）	15 0x0000 8000
	外部中断 2（EINT2）	16 0x0001 0000
	保留	17 0x0002 0000
ADC	A/D 转换器 0 转换结束	18 0x0004 0000
I^2C1	SI（状态改变）	19 0x0008 0000
-	保留	20 0x0010 0000 25 0x0200 0000
定时器 2	匹配 0-2（MR0, MR1, MR2） 捕获 0-2（CR0, CR1, CR2）	26 0x0400 0000
定时器 3	匹配 0-3（MR0, MR1, MR2, MR3）	27 0x0800 0000

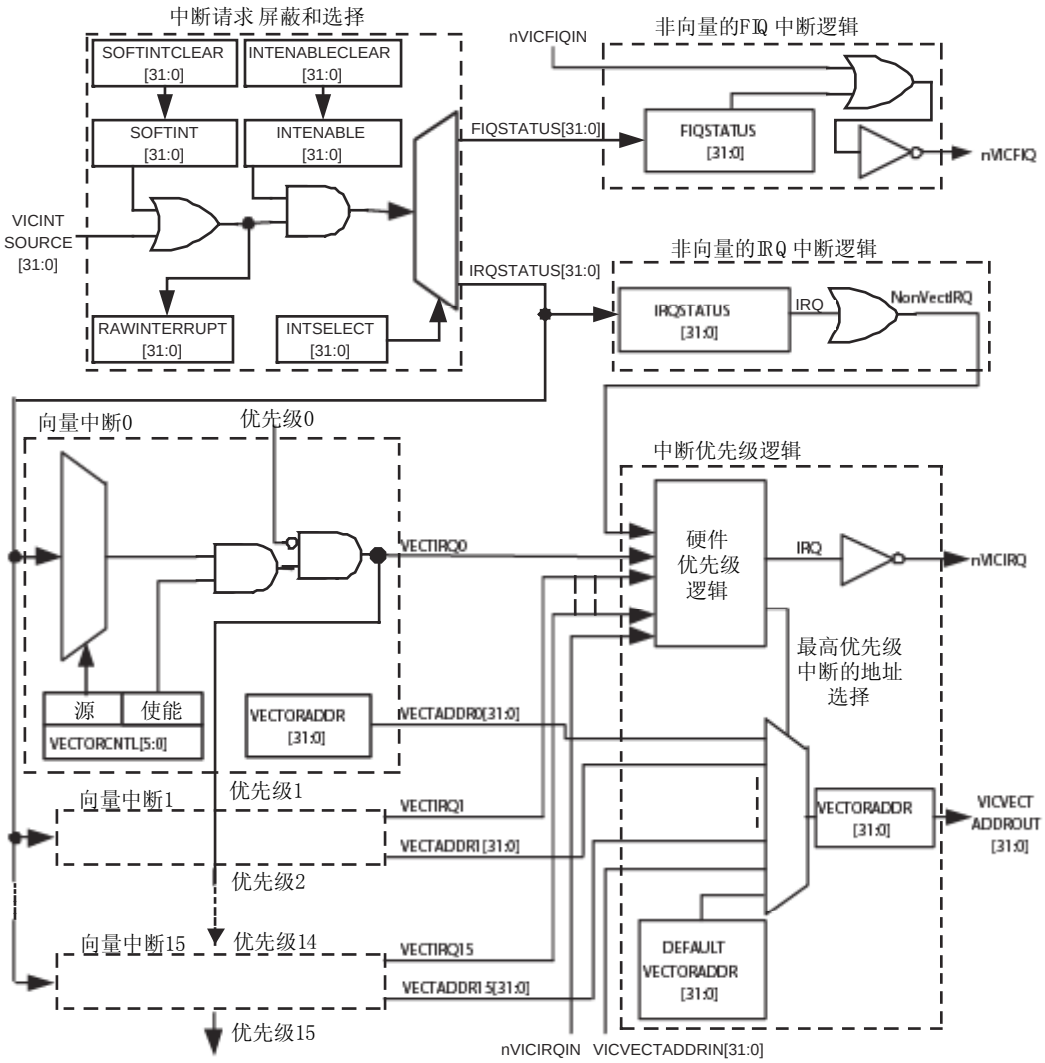


图 13 向量中断控制器(VIC)的方框图

5.6 伪中断

由于异步中断处理的原因，伪中断可能出现在基于 ARM7TDMI 的微控制器（例如 LPC2101/02/03）中。中断处理的异步特性来源于内核和 VIC 的相互作用。如果在内核检测到中断和内核真正开始处理中断的过程中 VIC 的状态发生改变，则产生中断的异步特性。

应用中可能经过以下步骤：

1. VIC 判断是否有 IRQ 中断并向内核发送 IRQ 信号。
2. 内核保存 IRQ 状态。
3. 执行流水线的多个周期的处理。
4. 内核从 VIC 中装入 IRQ 地址。

另外，VIC 的状态可能在第 3 步就发生了变化。例如，VIC 的变化使得触发从第 1 步开始的时序中断不再是挂起中断，在运行的代码中禁止。此时，VIC 不能清楚地识别产生中断请求的中断，最后只能返回到 VicDefVectAddr (0xFFFF F034)默认中断。

这一系列的情况可通过下面两种方法来防止：

1. 编写应用程序来防止伪中断的产生。仅仅简单地对 VIC 状态进行监控还不够，因为当干扰出现在电平激活方式的中断上也可能产生伪中断。
2. 应当正确设置和检测 VIC 默认处理器。

5.6.1 伪中断的详述和个案

本节的详细内容请登陆ARM官方网站（http://www.arm.com）。FAQ的“技术支持”链接为：http://www.arm.com/support/faqip/3677.html 。

当被禁能的中断产生时将出现怎样的情况？

应用范围：ARM7TDMI

如果在执行禁能中断指令时内核接收到中断请求，ARM7 系列器件仍然响应中断。这种情况出现在 IRQ 和 FIQ 中断中。

例如，假设正在执行下面的代码：

```
MRS     r0, cpsr
ORR     r0, r0, # I_Bit : OR : F_Bit    ；禁止 IRQ 和 FIQ 中断
MSR     cpsr_c, r0
```

如果在执行 MSR 指令时接收到 IRQ 中断，则执行以下操作：

- 保存 IRQ 中断
- 执行 MSR cpsr , r0 指令来完成 CPSR 中 I 位和 F 位的设置
- 响应 IRQ 中断，因为 CPSR 的 I 位被置位前内核将对中断异常进行处理
- CPSR（I 位和 F 位已被置位）的内容移入 SPSR_IRQ

这就意味着，在 IRQ 中断服务程序的入口，当 SPSR 的 I 位被置位时，你可以看到 IRQ 中断被处理的异常效果。在上面的例子中，CPSR 和 SPSR 的 F 位都置位。表明在 IRQ 服务程序的入口处 FIQ 被禁止，FIQ 将一直被禁止直到重新使能。FIQ 不能通过 IRQ 返回序列来自动重新使能。

尽管上例中的 IRQ 和 FIQ 中断都被禁止，但是如果两个中断类型中只有一种被禁止，出现的情况也与之类似。执行完禁止 IRQ 的 MSR 指令后内核才对 IRQ 进行处理，这通常不会产生任何问题，因为只有中断正好在一个周期之前到达才会被处理。当中断程序通过下面的指令返回时：

```
SUBS    pc , lr , #4
```

SPSR_IRQ 的值恢复到 CPSR。这时，CPSR 的 I 位和 F 位将被置位，从而将继续执行操作，所有中断禁止。但是，这样会造成以下情况：

情况 1：调用一个特定程序，该程序可以是 IRQ 处理程序或普通子程序。之后，系统必须保证 IRQ 在调用程序之前被禁能。程序利用这个限制条件来决定调用方式（通过检测 SPSR 的 I 位状态），并使用合适的指令返回。如果程序在执行禁能 IRQ 的 MSR 指令的过程中接收到 IRQ 时进入，那么置位 SPSR 的 I 位。因此，程序不可能通过 IRQ 来进入。

情况 2：FIQ 和 IRQ 通过同一条写 CPSR 的指令来禁止。这时，如果在写 CPSR 过程中接收到 IRQ，FIQ 将在执行 IRQ 处理程序时被禁止。这种情况不会出现在不允许 FIQ 禁能多个周期的系统中。

5.6.2 解决方案

这里推荐了 3 种解决方案。哪一种方案最合适取决于特定的系统要求。

5.6.3 方案 1：在写 CPSR 来禁止 IRQ 过程中检测接收到的 IRQ

在中断服务程序开始时增加类似于下面的代码。

```
SUB         lr , lr , #4                ；调整 LR，指向返回
STMFD       sp! , {… , lr}              ；获取一些空闲的寄存器
MRS         lr , SPSR                   ；当中断禁止时，判断是否有中断产生
TST         lr , # I_Bit                ；
LDMNEFD     sp! , {… , pc}^             ；如果有中断产生，立即返回。
                                        ；由于该中断并未被响应，因此仍保持待处理状态，
                                        ；它将在下次使能时再被重新处理。
                                        ；以下为中断程序
```

这部分代码用于在写 CPSR 来禁止 IRQ 过程中检测 IRQ 中断的产生。如果检测到 IRQ，程序立即返回，从而使得 IRQ 不被响应（清除），也禁止后面的 IRQ。

FIQ 处理程序也可使用类似的代码来解决情况 1。

由于该方案解决了上述的两种情况，因此推荐用户使用。但是，对于情况 2 的处理，该方案确实将 FIQ 禁止的最大时间延长了几个周期。

5.6.4 方案 2：分别使用两条写 CPSR 的指令来禁能 IRQ 和 FIQ

```
MRS     r0 , cpsr
ORR     r0 , r0 , # I_Bit       ；禁止 IRQ
MSR     cpsr_c , r0
ORR     r0 , r0 , # F_Bit       ；禁止 FIQ
MSR     cpsr_c , r0
```

当 FIQ 的禁能最长时间受到严格限制时，这是最好的一个方案（它根本不会延长 FIQ 的禁能时间）。但是，该方案并不能解决情况 1，要解决情况 1，还必须在每条禁能 IRQ 和 FIQ 的指令前同时添加一些其它指令。

5.6.5 方案 3：在 IRQ 处理程序的开始重新使能 FIQ

CPSR 的 C 字段的所有的位必须是可知的，最有效的方法就是向 CPSR_C 写入一个立即数，例如：

```
MSR     cpsr_c , # I_Bit : OR : irq_MODE     ；IRQ 应该禁能
                                             ；FIQ 使能
                                             ；ARM 状态，IRQ 模式
```

本方案只需要修改 IRQ 处理程序便可实现，它使得 FIQ 的重新使能比方案 1 更快。但是，这仅当系统可以保证在 IRQ 使能时 FIQ 不会被禁能的情况下使用本方案。本方案不能解决情况 1。

5.7 VIC 使用事项

如果在片内 RAM 当中运行代码并且应用程序需要调用中断，那么必须将中断向量重新映射到片内地址 0x0。这样做是因为所有的异常向量都位于地址 0x0 及以上。通过将寄存器 MEMMAP（见 3.7.1 节“存储器映射控制寄存器(MEMMAP – 0xE01F C040)”）配置为用户 RAM 模式来实现这一点。用户代码被连接以便使中断向量表 （IVT）装载到 0x4000 0000。

虽然可以选择多个中断源（通过 VICIntSelect）来产生 FIQ 请求，但是只有一个专门的中断服务程序来服务响应所有可用/出现的 FIQ 请求。因此，如果分配为 FIQ 的中断多于一个，FIQ 中断服务程序就必须读取 VICFIQStatus 的内容来决定如何处理中断请求。不过我们还是建议只将一个中断分配为 FIQ。多个 FIQ 中断源会增加中断延迟。

在中断服务程序执行完毕后，对外设中断标志的清零将会对 VIC 寄存器（VICRawlntr, VICFIQStatus 和 VICIRQStatus）当中的对应位产生影响。同时，在服务下次中断前，必须在中断返回之前对 VICVectAddr 寄存器执行写操作。该写操作将清零内部中断优先级硬件当中对应的中断标志。

通常要禁止 VIC 中断，必须清零 VICIntEnClr 寄存器中的对应位，该操作使 VICIntEnable 寄存器的对应位清零。这同样应用于 VICSoftInt 和 VICSoftIntClear，VICSoftIntClear 将会使 VICSoftInt 中的对应位清零。例如，如果 VICSoftInt=0x0000 0005 并且 bit0 必须清零，那么 VICSoftIntClear=0x0000 0001 可实现该操作。通过写 VICSoftIntClear 来执行对 VICSoftInt 当中相同位的新的清零操作之前，必须执行 VICSoftIntClear=0x0000 0000。因此向 VICSoftIntClear 寄存器任何位写入 1 对目标寄存器都是一次有效。

如果看门狗只在溢出或无效喂狗时产生中断，那么无法清除中断。唯一的方法是通过 VICIntEnClr 禁止 VIC 中断来实现中断返回。

举例：

假设 UART0 和 SPI0 产生中断请求，它们被分配为向量 IRQ（UART0 的优先级高于 SPI0），而 UART1 和 I^2C 产生非向量 IRQ，下面就是 VIC 一种可能的设定：

```
VICIntSelect = 0x0000 0000      (SPI0, I2C0, UART1 和 UART0 为 IRQ => bit10, bit9, bit7 和 bit6=0)
VICIntEnable = 0x0000 06C0      (SPI0, I2C0, UART1 和 UART0 中断使能 => bit10, bit9, bit 7 和 bit6=1)
VICDefVectAddr = 0x…            (保存服务非向量 IRQ 的程序地址 (即，UART1 和 I2C 的起始地址))
VICVectAddr0 = 0x…              (保存 UART0 IRQ 服务程序的起始地址)
VICVectAddr1 = 0x…              (保存 SPI0 IRQ 服务程序的起始地址)
VICVectCntl0 = 0x0000 0026      (VIC 通道号为 6（UART0）的中断源使能为优先级 0（最高级）)
VICVectCntl1 = 0x0000 002A      (VIC 通道号为 10（SPI0）中断源使能为优先级 1)
```

在任何 IRQ 请求（SPI0, I^2C, UART0 或 UART1）产生之后，微控制器跳转到地址 0x0000 0018 执行代码。对于向量和非向量 IRQ，可在地址 0x0000 0018 放入下面指令：

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

该指令将 VICVectAddr 寄存器中保存的地址装入 PC。

一旦产生 UART0 请求，VICVectAddr 和 VICVectAddr0 相同。如果产生 SPI0 请求，中断服务程序的地址保存在 VICVectAddr1 中。如果 UART0 和 SPI0 都没有产生 IRQ 请求，而 UART1 和/或 I^2C 产生请求，那么 VICVectAddr 的内容与 VICDefVectAddr 相同。

第6章 管脚配置

6.1 LPC2101/2102/2103 的管脚配置

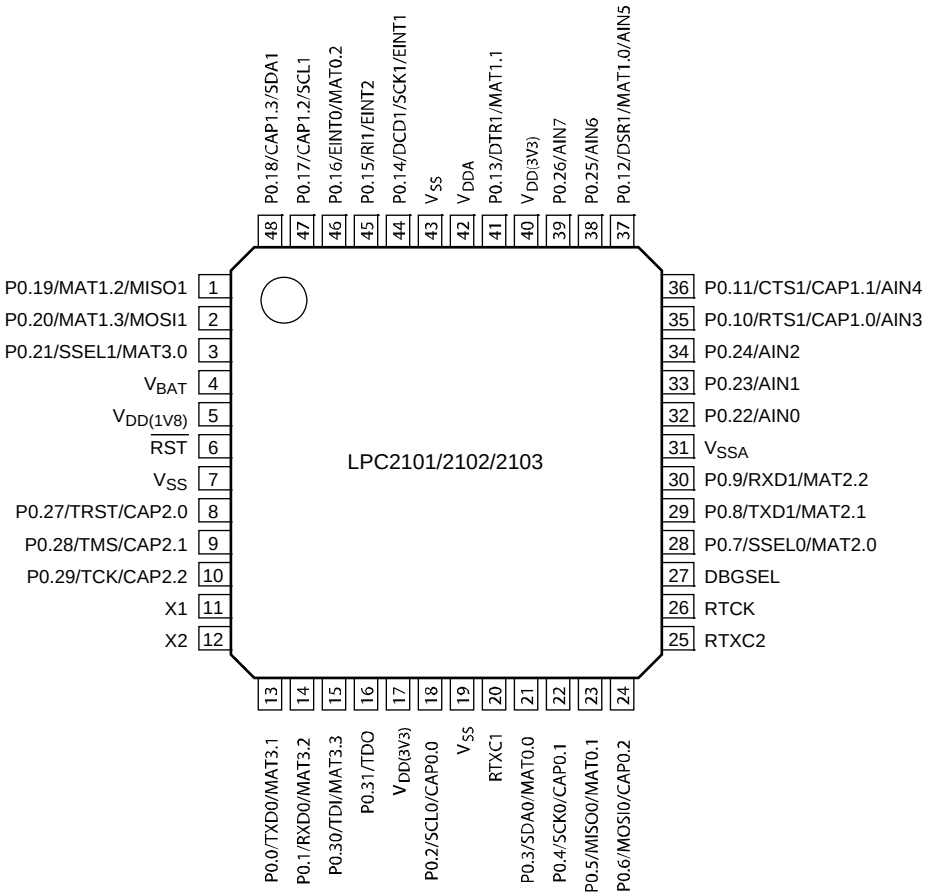


图 14 LQFP48 管脚配置

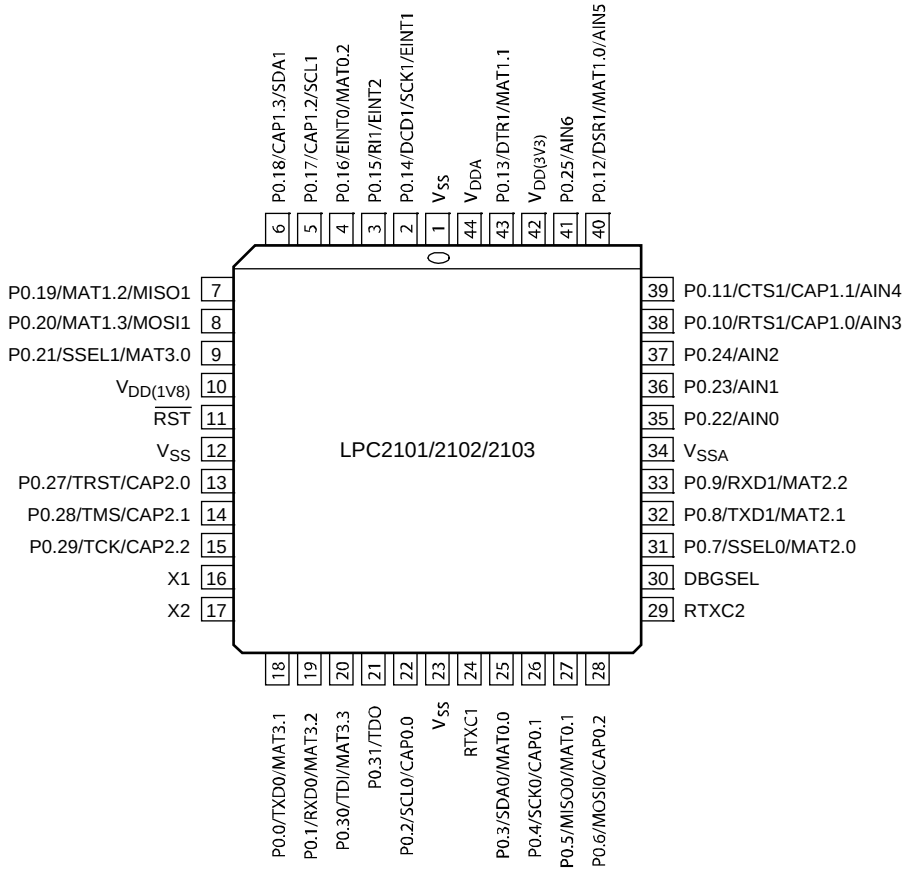


图 15 PLCC44 管脚配置

6.2 LPC2101/02/03 的管脚描述

LPC2101/02/03 的管脚描述及其主要功能见下表。

表 58 管脚描述

管脚名称	LQFP48	PLCC44	类型	描 述
P0.0~P0.31			I/O	**P0 口：**P0 口是一个 32 位 I/O 口。每个位都有独立的方向控制。有 31 个 P0 口可用作通用双向数字 I/O 口，P0.31 只用作输出口。P0 口管脚的操作取决于管脚连接模块所选择的功能。
P0.0/TXD0 /MAT3.1	13 [1]	18 [1]	I/O O O	**P0.0**－通用输入/输出数字管脚(GPIO) **TXD0**－UART0 的发送器输出 **MAT3.1**－定时器 3 的 PWM 输出 1
P0.1/RXD0 /MAT3.2	14 [2]	19 [2]	I/O I O	**P0.1**－通用输入/输出数字管脚(GPIO) **RXD0**－UART0 的接收器输入 **MAT3.2**－定时器 3 的 PWM 输出 2
P0.2/SCL0/ CAP0.0	18 [3]	22 [3]	I/O I/O I	**P0.2**－通用输入/输出数字管脚(GPIO) **SCL0**－I^2C0 时钟输入/输出。开漏输出(符合 I^2C 规范) **CAP0.0**－定时器 0 捕获输入 0
P0.3/SDA0/ MAT0.0	21 [3]	25 [3]	I/O I/O O	**P0.3**－通用输入/输出数字管脚(GPIO) **SDA0**－I^2C0 数据输入/输出。开漏输出(符合 I^2C 规范) **MAT0.0**－定时器 0 的 PWM 输出 0
P0.4/SCK0/ CAP0.1	22 [4]	26 [4]	I/O I/O I	**P0.4**－通用输入/输出数字管脚(GPIO) **SCK0**－SPI0 串行时钟，主机输出或从机输入的时钟 **CAP0.1**－定时器 0 捕获输入 1
P0.5/ MISO0/ MAT0.1	23 [4]	27 [4]	I/O I/O O	**P0.5**－通用输入/输出数字管脚(GPIO) **MISO0**－SPI0 主机输入/从机输出，从机到主机的数据传输 **MAT0.1**－定时器 0 的 PWM 输出 1
P0.6/ MOSI0/ CAP0.2	24 [4]	28 [4]	I/O I/O I	**P0.6**－通用输入/输出数字管脚(GPIO) **MOSI0**－SPI0 主机输出/从机输入，主机到从机的数据传输 **CAP0.2**－定时器 0 捕获输入 2
P0.7/ SSEL0/ MAT2.0	28 [2]	31 [2]	I/O I O	**P0.7**－通用输入/输出数字管脚(GPIO) **SSEL0**－SPI0 从机选择，选择 SPI 接口用作从机。 **MAT2.0**－定时器 2 的 PWM 输出，通道 0
P0.8/ TXD1/ MAT2.1	29 [4]	32 [4]	I/O O O	**P0.8**－通用输入/输出数字管脚(GPIO) **TXD1**－UART1 的发送器输出 **MAT2.1**－定时器 2 的 PWM 输出，通道 1
P0.9/ RXD1/ MAT2.2	30 [2]	33 [2]	I/O I O	**P0.9**－通用输入/输出数字管脚(GPIO) **RxD1**－UART1 的接收器输入 **MAT2.2**－定时器 2 的 PWM 输出，通道 2
P0.10/ RTS1/ CAP1.0/ AIN3	35 [4]	38 [4]	I/O O I I	**P0.10**－通用输入/输出数字管脚(GPIO) **RTS1**－UART1 请求发送输出。 **CAP1.0**－定时器 1 捕获输入 0 **AIN3**－模拟输入 3

续上表

管脚名称	LQFP48	PLCC44	类型	描 述
P0.11/	36 [3]	39 [3]	I/O	**P0.11**－通用输入/输出数字管脚(GPIO)
CTS1/			I	**CTS1**－UART1 的清零发送输入。
CAP1.1/			I	**CAP1.1**－定时器 1 捕获输入 1
AIN4			I	**AIN4**－模拟输入 4
P0.12/	37 [4]	40 [4]	I/O	**P0.12**－通用输入/输出数字管脚(GPIO)
DSR1/			I	**DSR1**－UART1 的数据设置就绪输入。
MAT1.0/			O	**MAT1.0**－定时器 1 的 PWM 输出 0
AIN5			I	**AIN5**－模拟输入 5
P0.13/	41 [4]	43 [4]	I/O	**P0.13**－通用输入/输出数字管脚(GPIO)
DTR1/			O	**DTR1**－UART1 的数据终端就绪输出。
MAT1.1			O	**MAT1.1**－定时器 1 的 PWM 输出 1
P0.14/	44 [3]	2 [3]	I/O	**P0.14**－通用输入/输出数字管脚(GPIO)
DCD1/			I	**DCD1**－UART1 数据载波检测输入。
SCK1/			I/O	**SCK1**－SPI1 的串行时钟。SPI 时钟的主机输出或从机输入。
EINT1			I	**EINT1**－外部中断 1 输入
P0.15/RI1/	45 [4]	3 [4]	I/O	**P0.15**－通用输入/输出数字管脚(GPIO)
EINT2			I	**RI1**－UART1 铃声指示输入。
			I	**EINT2**－外部中断 2 输入
P0.16/	46 [2]	4 [2]	I/O	**P0.16**－通用输入/输出数字管脚(GPIO)
EINT0/			I	**EINT0**－外部中断 0 输入
MAT0.2			O	**MAT0.2**－定时器 0 的 PWM 输出 2
P0.17/	47 [1]	5 [1]	I/O	**P0.17**－通用输入/输出数字管脚(GPIO)
CAP1.2/			I	**CAP1.2**－定时器 1 捕获输入 2
SCL1			I/O	**SCL1**－I^2C1 时钟输入/输出。开漏输出（遵循 I^2C 总线规范）
P0.18/CAP	48 [1]	6 [1]	I/O	**P0.18**－通用输入/输出数字管脚(GPIO)
1.3/SDA1			I	**CAP1.3**－定时器 1 捕获输入 3
			I/O	**SDA1**－I^2C1 数据输入/输出。开漏输出（遵循 I^2C 总线规范）
P0.19/	1 [1]	7 [1]	I/O	**P0.19**－通用输入/输出数字管脚(GPIO)
MAT1.2/			O	**MAT1.2**－定时器 1 的 PWM 输出 2
MISO1			I/O	**MISO1**－SSP 主机输出/从机输入，主机到从机的数据传输
P0.20/	2 [2]	8 [2]	I/O	**P0.20**－通用输入/输出数字管脚(GPIO)
MAT1.3/			O	**MAT1.3**－定时器 1 的 PWM 输出 3
MOSI1			I/O	**MOSI1**－SSP 主机输出/从机输入，主机到从机的数据传输
P0.21/	3 [4]	9 [4]	I/O	**P0.21**－通用输入/输出数字管脚(GPIO)
SSEL1/			I	**SSEL1**－SPI1 的从机选择。选择 SPI 接口作为从机。
MAT3.0			O	**MAT3.0**－定时器 3 的 PWM 输出，通道 0
P0.22/AIN0	32 [4]	35 [4]	I/O	**P0.22**－通用输入/输出数字管脚(GPIO)
			I	**AIN0**－模拟输入 0
P0.23/AIN1	33 [1]	36 [1]	I/O	**P0.23**－通用输入/输出数字管脚(GPIO)
			I	**AIN1**－模拟输入 1
P0.24/AIN2	34 [1]	37 [1]	I/O	**P0.24**－通用输入/输出数字管脚(GPIO)
			I	**AIN2**－模拟输入 2

续上表

管脚名称	LQFP48	PLCC44	类型	描 述
P0.25/AIN6	38 [1]	41 [1]	I/O I	**P0.25**－通用输入/输出数字管脚(GPIO) **AIN6**－模拟输入 6
P0.26/AIN7	39 [1]	n.c	I/O I	**P0.26**－通用输入/输出数字管脚(GPIO) **AIN7**－模拟输入 7
P0.27/ TRST/ CAP2.0	8 [4]	13 [4]	I/O I I	**P0.27**－通用输入/输出数字管脚(GPIO) **TRST**－JTAG 接口的测试复位 **CAP2.0**－定时器 2 捕获输入，通道 0
P0.28/TMS /CAP2.1	9 [4]	14 [4]	I/O I I	**P0.28**－通用输入/输出数字管脚(GPIO) **TMS**－JTAG 接口的测试模式选择 **CAP2.1**－定时器 2 捕获输入，通道 1
P0.29/TCK/ CAP2.2	10 [4]	15 [4]	I/O I I	**P0.29**－通用输入/输出数字管脚(GPIO) **TCK**－JTAG 接口的测试时钟 **CAP2.2**－定时器 2 捕获输入，通道 2
P0.30/TDI/ MAT3.3	15 [4]	20 [4]	I/O I O	**P0.30**－通用输入/输出数字管脚(GPIO) **TDI**－JTAG 接口的测试数据输入 **MAT3.3**－定时器 3 的 PWM 输出 3
P0.31/TDO	16 [4]	21 [4]	O O	**P0.31**－通用仅为输出的数字管脚(GPO) **TDO**－JTAG 接口的测试数据输出
RTXC1	20 [5]	24 [5]	I	RTC 振荡器电路的输入
RTXC2	25 [5]	29 [5]	O	RTC 振荡器电路的输出
RTCK	26 [5]	n.c	I/O	**返回的测试时钟输出**：JTAG 端口的额外信号。当处理器频率变化时帮助调试器保持同步。带内部上拉的双向口。
X1	11	16	I	振荡器电路和内部时钟发生器电路的输入。
X2	12	17	O	振荡放大器的输出。
DBGSEL	27	30	I	**调试选择**：当该管脚为低电平时，器件正常操作。当该管脚为高电平时，进入调试模式。输入内部拉低。
$\overline{RST}$	6	11	I	**外部复位输入**：该管脚的低电平将器件复位，并使 I/O 口和外围功能恢复默认状态，处理器从地址 0 开始执行。带迟滞的 TTL 电平，管脚可承受 5V 电压。
V_{SS}	7,19,43	1,12,23	I	**地**：0V 参考点。
V_{SSA}	31	34	I	**模拟地**：0V 参考点。标称电压与 V_{SS} 相同，但应当互相隔离以减少噪声和故障。
V_{DDA}	42	44	I	**模拟 3.3V 端口电源**：标称电压与 $V_{DD(3V3)}$相同，但应当互相隔离以减少噪声和故障。该电压也用来向片内 PLL 供电。
$V_{DD(1V8)}$	5	10	I	**1.8V 内核电源**：这是内部电路的电源电压。
$V_{DD(3V3)}$	17,40	42	I	**3.3V 端口电源**：这是 I/O 口的电源电压。
V_{BAT}	4	n.c	I	**RTC 电源**：RTC 的 3.3V 电源端。

[1] 5V 电压容限端口提供带 TTL 电平、滞后和 10ns 转换速率控制的数字 I/O 功能。

[2] 5V 电压容限端口提供带 TTL 电平、滞后和 10ns 转换速率控制的数字 I/O 功能。如果配置为输入功能，该端口利用内置的干扰滤波器阻止短于 3ns 的脉冲。

[3] 可承受开漏 5V 电压的数字 I/O I^2C 总线 400kHz 规格的兼容端口。它需要外部上拉提供多种用途的输出。

[4] 5V 电压容限端口提供数字 I/O（带 TTL 电平、滞后和 10ns 转换速率控制）和模拟输入功能。如果配置为输入功能，该端口利用内置的干扰滤波器阻止短于 3ns 的脉冲。当配置为 ADC 输入时，端口的数字部分禁能。

[5] 端口提供特殊的模拟功能。

第7章 管脚连接模块

7.1 特性

管脚连接模块可实现独立的管脚配置。

7.2 应用

管脚连接模块的用途是将微控制器管脚配置为需要的功能。

7.3 描述

管脚连接模块可以使微控制器的所选管脚具有 1 个以上的功能。配置寄存器控制多路开关来连接管脚与片内外设。

外设在激活和任何相关中断使能之前必须连接到适当的管脚。任何使能的外设功能如果没有映射到相关的管脚，则被认为是无效的。

当管脚只选择一个功能时，其它功能无效。

上述规则唯一的例外是管脚用作 A/D 转换器输入时。不论管脚选择何种功能，它都仍可用作 A/D 输入，A/D 输入可随时被读取，管脚的电压变化都从 A/D 的读取值中反映出来。但是，只有选择模拟输入功能，才能读出有效的模拟值。也只有这种情况下，管脚和 A/D 模块之间的接口电路才有效。其它情况下，执行数字功能所必需的数字逻辑部分将有效，从而影响 A/D 转换器的正确操作。

7.4 寄存器描述

管脚控制模块包含 2 个寄存器，见表 59。

表 59 管脚控制模块寄存器映射

名称	描述	访问	复位值[1]	地址
PINSEL0	管脚功能选择寄存器 0	读/写	0x0000 0000	0xE002 C000
PINSEL1	管脚功能选择寄存器 1	读/写	0x0000 0000	0xE002 C004

[1] 复位值仅指已使用位中保存的数据，不包括保留位的内容。

7.4.1 管脚功能选择寄存器 0（PINSEL0 - 0xE002 C000）

PINSEL0 寄存器按照表 62 当中的设定来控制管脚的功能。IO0DIR 寄存器中的方向控制位只有在管脚选择 GPIO 功能时才有效。对于其它功能，方向是自动控制的。

表 60 管脚功能选择寄存器 0（PINSEL0 – 地址 0xE002 C000）

PINSEL0	管脚名称	值	功能	复位值
1:0	P0.0	0 0	GPIO P0.0	0
		0 1	TxD0 (UART0)	
		1 0	MAT3.1（定时器 3）	
		1 1	保留	
3:2	P0.1	0 0	GPIO P0.1	0
		0 1	RxD0 (UART0)	
		1 0	MAT3.2（定时器 3）	
		1 1	保留	
5:4	P0.2	0 0	GPIO P0.2	0
		0 1	SCL0 (I^2C0)	
		1 0	CAP0.0 (定时器 0)	
		1 1	保留	
7:6	P0.3	0 0	GPIO P0.3	0
		0 1	SDA0 (I^2C0)	
		1 0	MAT0.0 (定时器 0)	
		1 1	保留	
9:8	P0.4	0 0	GPIO P0.4	0
		0 1	SCK0(SPI0)	
		1 0	CAP0.1 (定时器 0)	
		1 1	保留	
11:10	P0.5	0 0	GPIO P0.5	0
		0 1	MISO0(SPI0)	
		1 0	MAT0.1 (定时器 0)	
		1 1	保留	
13:12	P0.6	0 0	GPIO P0.6	0
		0 1	MOSI0(SPI0)	
		1 0	CAP0.2 (定时器 0)	
		1 1	保留	
15:14	P0.7	0 0	GPIO P0.7	0
		0 1	SSEL0(SPI0)	
		1 0	MAT2.0（定时器 2）	
		1 1	保留	
17:16	P0.8	0 0	GPIO P0.8	0
		0 1	TxD1（UART1）	
		1 0	MAT2.1（定时器 2）	
		1 1	保留	
19:18	P0.9	0 0	GPIO P0.9	0
		0 1	RxD(UART1)	
		1 0	MAT2.2（定时器 2）	
		1 1	保留	

续上表

PINSEL0	管脚名称	值	功能	复位值
21:20	P0.10	0 0	GPIO P0.10	0
		0 1	RTS1 (UART1)	
		1 0	CAP1.0 (定时器 1)	
		1 1	AIN3	
23:22	P0.11	0 0	GPIO P0.11	0
		0 1	CTS1 (UART1)	
		1 0	CAP1.1 (定时器 1)	
		1 1	AIN4	
25:24	P0.12	0 0	GPIO P0.12	0
		0 1	DSR1 (UART1)	
		1 0	MAT1.0 (定时器 1)	
		1 1	AIN5	
27:26	P0.13	0 0	GPIO P0.13	0
		0 1	保留	
		1 0	MAT1.1 (定时器 1)	
		1 1	DTR1 (UART1)	
29:28	P0.14	0 0	GPIO P0.14	0
		0 1	EINT1	
		1 0	SCK1 (SSP1)	
		1 1	DCD1 (UART1)	
31:30	P0.15	0 0	GPIO P0.15	0
		0 1	EINT2	
		1 0	保留	
		1 1	RI1 (UART1)	

7.4.2 管脚功能选择寄存器 1（PINSEL1 - 0xE002 C004）

PINSEL1 寄存器按照下表中的设定来控制管脚的功能。IO0DIR 寄存器中的方向控制位只有在管脚选择 GPIO 功能时才有效。对于其它功能，方向是自动控制的。

表 61 管脚功能选择寄存器 1（PINSEL1 – 地址 0xE002 C004）位描述

PINSEL1	管脚名称	值	功能	复位值
1:0	P0.16	0 0	GPIO P0.16	0
		0 1	EINT0	
		1 0	MAT0.2（定时器 0)	
		1 1	保留	
3:2	P0.17	0 0	GPIO P0.17	0
		0 1	SCL1 (I^2C1)	
		1 0	CAP1.2 (定时器 1)	
		1 1	保留	

续上表

位	符号	值	功能	复位值
5:4	P0.18	0 0	GPIO P0.18	0
		0 1	SDA1 (I^2C1)	
		1 0	CAP1.3 (定时器 1)	
		1 1	保留	
7:6	P0.19	0 0	GPIO P0.19	0
		0 1	MISO1 (SPI1)	
		1 0	MAT1.2（定时器 1）	
		1 1	保留	
9:8	P0.20	0 0	GPIO P0.20	0
		0 1	MOSI1 (SPI1)	
		1 0	MAT1.3（定时器 1）	
		1 1	保留	
11:10	P0.21	0 0	GPIO P0.21	0
		0 1	SSEL1 (SPI1)	
		1 0	MAT3.0（定时器 3）	
		1 1	保留	
13:12	P0.22	0 0	GPIO P0.22	0
		0 1	保留	
		1 0	保留	
		1 1	AIN0	
15:14	P0.23	0 0	GPIO P0.23	0
		0 1	保留	
		1 0	保留	
		1 1	AIN1	
17:16	P0.24	0 0	GPIO P0.24	0
		0 1	保留	
		1 0	保留	
		1 1	AIN2	
19:18	P0.25	0 0	GPIO P0.25	0
		0 1	保留	
		1 0	保留	
		1 1	AIN6	
21:20	P0.26	0 0	GPIO P0.26	0
		0 1	保留	
		1 0	保留	
		1 1	AIN7	
23:22	P0.27	0 0	GPIO P0.27	0
		0 1	TRST (JTAG)	
		1 0	CAP2.0（定时器 2）	
		1 1	保留	

续上表

位	符号	值	功能	复位值
25:24	P0.28	0 0	GPIO P0.28	0
		0 1	TMS (JTAG)	
		1 0	CAP2.1 (定时器 2)	
		1 1	保留	
27:26	P0.29	0 0	GPIO P0.29	0
		0 1	TCK (JTAG)	
		1 0	CAP2.2 (定时器 2)	
		1 1	保留	
29:28	P0.30	0 0	GPIO P0.30	0
		0 1	TDI (JTAG)	
		1 0	MAT3.3（定时器 3）	
		1 1	保留	
31:30	P0.31	0 0	GPIO P0.31	0
		0 1	TDO (JTAG)	
		1 0	保留	
		1 1	保留	

7.4.3 管脚功能选择寄存器值

PINSEL 寄存器控制下表中器件管脚的功能。每一对寄存器位对应一个特定的器件管脚。

表 62 管脚功能选择寄存器位

PINSEL0 和 PINSEL1 的值	功能	复位值
00	首选（默认）功能，通常为 GPIO 口	00
01	第一可选功能	
10	第二可选功能	
11	保留	

只有当管脚选择 GPIO 功能时，IO0DIR 寄存器的方向控制位才有效。其它功能的方向是自动控制的。每个派生器件通常具有不同的管脚配置，因此每个管脚可能有不同的功能。详见对应的器件手册。

第8章 通用输入/输出口(GPIO)

8.1 特性

- 每一个物理的 GPIO 端口都可通过寄存器组（提供增强型特性和加速的端口访问）或遗留的寄存器组来访问。
- 加速的 GPIO 功能：
  - －GPIO 寄存器被转移到 ARM 局部总线，以实现高速 I/O 时序。
  - －标志寄存器允许将端口位集作为一组，而其它位不变。
  - －所有寄存器为字节和半字可寻址。
  - －整个端口值可用一条指令写入。
- 位电平置位和清零寄存器允许一条指令置位或清零一个端口的任何位数。
- 单个位的方向控制
- 所有 I/O 口在复位后默认为输入
- 向后兼容其它较早的器件，保留 APB 总线上原有地址处出现的遗留寄存器。

8.2 应用

- 通用 I/O 口
- 驱动 LED 或其它指示器
- 控制片外器件
- 检测数字输入

8.3 管脚描述

表 63 GPIO 管脚描述

管脚名称	类型	描述
P0.0 – P0.31	输入/输出	通用输入/输出口。实际可用的 GPIO 数量取决于可选功能的使用。

8.4 寄存器描述

LPC2101/02/03 有 1 个 32 位的通用 I/O 口。PORT0 使用总共 32 脚的输入/输出管脚，PORT0 由表 64 和表 65 中列出的的寄存器控制。

表 64 所示的遗留寄存器允许向后兼容较早系列的器件，使用现有的代码。保留原 GPIO 执行的功能和相关时序。

表 65 中的寄存器反映 LPC2101/02/03 可用的增强型 GPIO 特性。所有的这些寄存器直接位于 CPU 的局部总线上，以便于实现可能的最高速读写时序。添加了额外的特性为所有 GPIO 寄存器提供字节可寻址性。屏蔽寄存器允许将单个 GPIO 端口的一组位与该端口上的其它位分开处理。

用户必须选择 GPIO 是通过提供增强型特性的寄存器来访问还是通过寄存器的遗留集

合来访问（见 3.6.1 节“系统控制和状态标志寄存器(SCS – 0xE01F C1A0)”）。当两个端口的高速和遗留的 GPIO 寄存器正在控制相同的物理管脚时，这两个端口控制分支相互排斥且独立操作。例如，通过高速寄存器改变管脚的输出将不可通过相关的遗留寄存器来观察到。

下文中将把遗留 GPIO 看成为“低速”GPIO，而把具有增强型特性的 GPIO 看成为“高速”GPIO。

表 64 GPIO 寄存器映射（遗留 APB 可访问的寄存器）

通用名称	描述	访问	复位值[1]	PORT0 地址&名称
IOPIN	GPIO 端口管脚值寄存器。不管管脚的方向如何设定，GPIO 配置端口管脚的当前状态都可从该寄存器中读出。	R/W	NA	0xE002 8000 IO0PIN
IOSET	GPIO 端口输出置位寄存器。该寄存器和 IOCLR 寄存器一起控制输出管脚的状态。写入 1 使对应管脚输出高电平。写入 0 无效。	R/W	0x0000 0000	0xE002 8004 IO0SET
IODIR	GPIO 端口方向控制寄存器。该寄存器单独控制每个 I/O 口的方向。	R/W	0x0000 0000	0xE002 8008 IO0DIR
IOCLR	GPIO 端口输出清零寄存器。该寄存器控制输出管脚的状态。写入 1 使对应管脚输出低电平并清零 IOSET 寄存器中的对应位。写入 0 无效。	WO	0x0000 0000	0xE002 800C IO0CLR

[1] 复位值仅指已使用位中保存的数据。它不包括保留位的内容。

表 65 GPIO 寄存器映射（局部总线可访问的寄存器 – 增强型的 GPIO 特性）

通用名称	描述	访问	复位值[1]	PORT0 地址&名称
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	0x0000 0000	0x3FFF C000 FIO0DIR
FIOMASK	端口的高速屏蔽寄存器。写，设置，清除和读端口（通过写入 FIOPIN, FIOSET 和 FIOCLR，以及读 FIOPIN 完成）改变或返回仅在寄存器中被 0 使能的位。	R/W	0x0000 0000	0x3FFF C010 FIO0MASK
FIOPIN	使用 FIOMASK 的高速端口管脚值寄存器。不管管脚方向或交替的功能选择如何，可从该寄存器中读取数字端口管脚的当前状态（只要管脚不配置为 ADC 的输入）。读出的值通过和 FIOMASK 相与来屏蔽。写该寄存器，在 FIOMASK 中由 0 使能的所有位中放置对应的值。	R/W	0x0000 0000	0x3FFF C014 FIO0PIN
FIOSET	使用 FIOMASK 的高速端口输出设置寄存器。该寄存器控制输出管脚的状态。写 1 在相应的端口管脚产生高电平。写 0 无效。读该寄存器返回端口输出寄存器的当前内容。仅可改变在 FIOMASK 中被 1 使能的位。	R/W	0x0000 0000	0x3FFF C018 FIO0SET
FIOCLR	使用 FIOMASK 的高速端口输出清零寄存器。该寄存器控制输出管脚的状态。写 1 在相应的端口管脚产生低电平。写 0 无效。仅可改变在 FIOMASK 中被 1 使能的位。	WO	0x0000 0000	0x3FFF C01C FIO0CLR

[1] 复位值仅指已使用位中保存的数据。它不包括保留位的内容。

8.4.1 GPIO 端口 0 方向寄存器（IODIR，端口 0：IO0DIR - 0xE002 8008；FIODIR，端口 0：FIO0DIR – 0x3FFF C000）

当管脚配置为 GPIO 模式时，使用可字访问的寄存器控制管脚的方向。必须根据管脚的功能设置任意管脚的方向位。

当通过 FIO0DIR 寄存器支持增强型的 GPIO 功能时，遗留的寄存器为 IO0DIR。

表 66 GPIO 端口 0 方向寄存器（IO0DIR – 地址 0xE002 8008）位描述

位	符号	值	描述	复位值
31:0	P0xDIR		低速 GPIO 方向控制位。位 0 控制 P0.0 ... 位 30 控制 P0.30。	0x0000 0000
		0	控制的管脚为输入。	
		1	控制的管脚为输出。	

表 67 高速 GPIO 端口 0 方向寄存器（FIO0DIR –地址 0x3FFF C000）位描述

位	符号	值	描述	复位值
31:0	FP0xDIR		高速 GPIO 方向控制位。FIO0DIR 中的位 0 控制 P0.0 ... FIO0DIR 中的位 30 控制 P0.30。	0x0000 0000
		0	控制的管脚为输入。	
		1	控制的管脚为输出。	

除了 32 位长和仅为字访问的 FIODIR 寄存器外，可通过表 68 中列出的几个字节和可半字访问的寄存器来控制每个高速的 GPIO 端口。除了提供与 FIODIR 寄存器相同的功能，这些额外的寄存器允许对物理端口管脚提供更方便和更快速的访问。

表 68 高速 GPIO 端口 0 方向控制字节和可半字访问的寄存器描述

寄存器名称	寄存器长度(位)&访问	地址	描述	复位值
FIO0DIR0	8（字节）	0x3FFF C000	高速 GPIO 端口 0 方向控制寄存器 0。FIO0DIR0 寄存器中的位 0 对应 P0.0…位 7 对应 P0.7。	0x00
FIO0DIR1	8（字节）	0x3FFF C001	高速 GPIO 端口 0 方向控制寄存器 1。FIO0DIR1 寄存器中的位 0 对应 P0.8…位 7 对应 P0.15。	0x00
FIO0DIR2	8（字节）	0x3FFF C002	高速 GPIO 端口 0 方向控制寄存器 2。FIO0DIR2 寄存器中的位 0 对应 P0.16…位 7 对应 P0.23。	0x00
FIO0DIR3	8（字节）	0x3FFF C003	高速 GPIO 端口 0 方向控制寄存器 3。FIO0DIR3 寄存器中的位 0 对应 P0.24…位 7 对应 P0.31。	0x00
FIO0DIRL	16（半字）	0x3FFF C000	高速 GPIO 端口 0 方向控制低半字寄存器。FIO0DIRL 寄存器中的位 0 对应 P0.0…位 15 对应 P0.15。	0x0000
FIO0DIRU	16（半字）	0x3FFF C002	高速 GPIO 端口 0 方向控制高半字寄存器。FIO0DIRU 寄存器中的位 0 对应 P0.16…位 15 对应 P0.31。	0x0000

8.4.2 高速 GPIO 端口 0 屏蔽寄存器 (FIOMASK, 端口 0：FIO0MASK – 0x3FFF C010)

该寄存器只能从增强型的寄存器组中得到。它用于选择端口的管脚会不会受到写访问 FIOPIN, FIOSET 或 FIOCLR 寄存器的影响。当读 FIOPIN 寄存器时，屏蔽寄存器也去掉端口的内容。

寄存器的位为 0 通过读或写访问使能相应的物理管脚。如果该寄存器中的位为 1，相应的管脚将不会通过写访问改变，且如果进行读操作，将不会在更新的 FIOPIN 寄存器中反映。关于软件的例子，见 8.5 节“GPIO 使用注意事项”。

表 69 高速 GPIO 端口 0 屏蔽寄存器(FIO0MASK –地址 0x3FFF C010)位描述

位	符号	值	描述	复位值
31:0	FP0xMASK	0	高速 GPIO 物理管脚访问控制。 通过写 FIOSET, FIOCLR 和 FIOPIN 寄存器影响管脚。 将可以在 FIOPIN 寄存器中看到管脚的当前状态。	0x0000 0000
		1	通过写 FIOSET, FIOCLR 和 FIOPIN 寄存器不影响物理管脚。当读取 FIOPIN 寄存器时，该位不能通过物理管脚的状态更新。	

除了 32 位长和仅为字访问的 FIOMASK 寄存器外，可通过表 70 中列出的几个字节和半字访问的寄存器来控制每个高速的 GPIO 端口。其次是提供如 FIOMASK 寄存器相同的功能，这些附加的寄存器允许更方便和更快地访问物理端口管脚。

表 70 高速 GPIO 端口 0 屏蔽字节和可半字访问的寄存器描述

寄存器名称	寄存器长度(位)&访问	地址	描述	复位值
FIO0MASK0	8（字节）	0x3FFF C010	高速 GPIO 端口 0 屏蔽寄存器 0。FIO0MASK0 寄存器中的位 0 对应 P0.0…位 7 对应 P0.7。	0x00
FIO0MASK1	8（字节）	0x3FFF C011	高速 GPIO 端口 0 屏蔽寄存器 1。FIO0MASK1 寄存器中的位 0 对应 P0.8…位 7 对应 P0.15。	0x00
FIO0MASK2	8（字节）	0x3FFF C012	高速 GPIO 端口 0 屏蔽寄存器 2。FIO0MASK2 寄存器中的位 0 对应 P0.16…位 7 对应 P0.23。	0x00
FIO0MASK3	8（字节）	0x3FFF C013	高速 GPIO 端口 0 屏蔽寄存器 3。FIO0MASK3 寄存器中的位 0 对应 P0.24…位 7 对应 P0.31。	0x00
FIO0MASKL	16（半字）	0x3FFF C010	高速 GPIO 端口 0 屏蔽低半字寄存器。FIO0MASKL 寄存器中的位 0 对应 P0.0…位 15 对应 P0.15。	0x0000
FIO0MASKU	16（半字）	0x3FFF C012	高速 GPIO 端口 0 屏蔽高半字寄存器。FIO0MASKU 寄存器中的位 0 对应 P0.16…位 15 对应 P0.31。	0x0000

8.4.3 GPIO 端口 0 管脚值寄存器（IOPIN, 端口 0：IO0PIN - 0xE002 8000；FIOPIN, 端口 0：FIO0PIN - 0x3FFF C014）

该寄存器提供端口管脚的值，此管脚被配置为仅实现数字功能。不管管脚是否配置为输入/输出，或作为 GPIO 或可选的数字功能，该寄存器将给出管脚的逻辑值。例如，一个特定的端口管脚可具有 GPIO 输入，GPIO 输出，UART 接收和 PWM 输出作为可选的功能。该管脚的任何配置将允许从 IOPIN 寄存器中读取其当前的逻辑状态。

如果管脚有一个模拟功能作为其选项之一，那么若选择了模拟配置，管脚的状态就不能读。选择管脚作为 A/D 输入，则断开管脚的数字特性。在这种情况下，在 IOPIN 寄存器中读出的管脚值无效。

写 IOPIN 寄存器存储端口输出寄存器中的值，忽略使用 IOSET 和 IOCLR 寄存器来得到整个写入的值。在应用中要小心使用该特性，因为它影响整个端口。

当通过 FIO0PIN 寄存器支持增强型的 GPIO 时，遗留寄存器为 IO0PIN。通过 FIOPIN 寄存器访问端口管脚的状况受到相应的 FIOMASK 寄存器限制（见 8.4.2 节“高速 GPIO 端口 0 屏蔽寄存器(FIOMASK，端口 0：FIO0MASK– 0x3FFF C010)”）。

在屏蔽寄存器中仅用 0 屏蔽的管脚（见 8.4.2 节“高速 GPIO 端口 0 屏蔽寄存器(FIOMASK，端口 0：FIO0MASK – 0x3FFF C010)”）将与高速 GPIO 端口管脚值寄存器的当前内容相关。

表 71 GPIO 端口 0 管脚值寄存器（IO0PIN – 地址 0xE002 8000）位描述

位	符号	描述	复位值
31:0	P0xVAL	低速 GPIO 管脚值位。IO0PIN 的位 0 对应于 P0.0 ... 位 31 对应于 P0.31	NA

表 72 高速 GPIO 端口 0 管脚值寄存器(FIO0PIN –地址 0x3FFF C014)位描述

位	符号	描述	复位值
31:0	FP0xVAL	高速 GPIO 管脚值位。FIO0PIN 的位 0 对应于 P0.0 ... 位 31 对应于 P0.31	NA

除了 32 位长和仅为字访问的 FIOPIN 寄存器外，可通过表 73 中列出的几个字节和半字访问寄存器来控制每个高速的 GPIO 端口。其次是提供如 FIOPIN 寄存器相同的功能，这些附加的寄存器允许更方便和更快地访问物理端口管脚。

表 73 高速 GPIO 端口 0 管脚值字节和可半字访问的寄存器描述

寄存器名称	寄存器长度(位)&访问	地址	描述	复位值
FIO0PIN0	8（字节）	0x3FFF C014	高速 GPIO 端口 0 管脚值寄存器 0。FIO0PIN0 寄存器中的位 0 对应 P0.0…位 7 对应 P0.7。	0x00
FIO0PIN1	8（字节）	0x3FFF C015	高速 GPIO 端口 0 管脚值寄存器 1。FIO0PIN1 寄存器中的位 0 对应 P0.8…位 7 对应 P0.15。	0x00
FIO0PIN2	8（字节）	0x3FFF C016	高速 GPIO 端口 0 管脚值寄存器 2。FIO0PIN2 寄存器中的位 0 对应 P0.16…位 7 对应 P0.23。	0x00
FIO0PIN3	8（字节）	0x3FFF C017	高速 GPIO 端口 0 管脚值寄存器 3。FIO0PIN3 寄存器中的位 0 对应 P0.24…位 7 对应 P0.31。	0x00
FIO0PINL	16（半字）	0x3FFF C014	高速 GPIO 端口 0 管脚值低半字寄存器。FIO0PINL 寄存器中的位 0 对应 P0.0…位 15 对应 P0.15。	0x0000
FIO0PINU	16（半字）	0x3FFF C016	高速 GPIO 端口 0 管脚值高半字寄存器。FIO0PINU 寄存器中的位 0 对应 P0.16…位 15 对应 P0.31。	0x0000

8.4.4 GPIO 端口 0 输出置位寄存器（IOSET,端口 0：IO0SET - 0xE002 8004；FIOSET,端口 0：FIO0SET - 0x3FFF C018）

当管脚配置为 GPIO 输出模式时，可使用该寄存器从管脚输出高电平。写入 1 使对应管脚输出高电平。写入 0 无效。如果一个管脚被配置为输入或第二功能，那么在 IOSET 中的相应位写入 1 无效。

读 IOSET 寄存器返回 GPIO 输出寄存器中的值。该值由前一次对 IOSET 和 IOCLR（或前面提到的 IOPIN）的写操作决定。该值不反映任何外部环境对 I/O 管脚的影响。

当通过 FIO0SET 寄存器支持增强型的 GPIO 时，遗留寄存器为 IO0SET。通过 FIOSET 寄存器访问端口管脚的状况受到相应的 FIOMASK 寄存器控制（见 8.4.2 节“高速 GPIO 端口 0 屏蔽寄存器(FIOMASK，端口 0：FIO0MASK – 0x3FFF C010)”）。

表 74 GPIO 端口 0 输出置位寄存器（IO0SET – 地址 0xE002 8004）位描述

位	符号	描述	复位值
31:0	P0xSET	低速 GPIO 输出值 SET 位。IO0SET 的位 0 对应于 P0.0 ... 位 31 对应于 P0.31。	0x0000 0000

表 75 高速 GPIO 端口 0 输出置位寄存器（FIO0SET – 地址 0x3FFF C018）位描述

位	符号	描述	复位值
31:0	FP0xSET	高速 GPIO 输出值 SET 位。FIO0SET 的位 0 对应于 P0.0 ... 位 31 对应于 P0.31。	0x0000 0000

除了 32 位长和仅为字访问的 FIOSET 寄存器外，可通过表 76 中列出的几个字节和半字访问的寄存器来控制每个高速的 GPIO 端口。其次是提供如 FIOSET 寄存器相同的功能，这些附加的寄存器允许更方便和更快地访问物理端口管脚。

表 76 高速 GPIO 端口 0 输出置位字节和可半字访问的寄存器描述

寄存器名称	寄存器长度(位)&访问	地址	描述	复位值
FIO0SET0	8（字节）	0x3FFF C018	高速 GPIO 端口 0 输出置位寄存器 0。FIO0SET0 寄存器中的位 0 对应 P0.0…位 7 对应 P0.7。	0x00
FIO0SET1	8（字节）	0x3FFF C019	高速 GPIO 端口 0 输出置位寄存器 1。FIO0SET1 寄存器中的位 0 对应 P0.8…位 7 对应 P0.15。	0x00
FIO0SET2	8（字节）	0x3FFF C01A	高速 GPIO 端口 0 输出置位寄存器 2。FIO0SET2 寄存器中的位 0 对应 P0.16…位 7 对应 P0.23。	0x00
FIO0SET3	8（字节）	0x3FFF C01B	高速 GPIO 端口 0 输出置位寄存器 3。FIO0SET3 寄存器中的位 0 对应 P0.24…位 7 对应 P0.31。	0x00

续上表

寄存器名称	寄存器长度(位)&访问	地址	描述	复位值
FIO0SETL	16（半字）	0x3FFF C018	高速 GPIO 端口 0 输出置位低半字寄存器。FIO0SETL 寄存器中的位 0 对应 P0.0…位 15 对应 P0.15。	0x0000
FIO0SETU	16（半字）	0x3FFF C01A	高速 GPIO 端口 0 输出置位高半字寄存器。FIO0SETU 寄存器中的位 0 对应 P0.16…位 15 对应 P0.31。	0x0000

8.4.5 GPIO 端口 0 输出清零寄存器(IOCLR,端口 0:IO0CLR – 0xE002 800C；FIOCLR,端口 0:FIO0CLR – 0x3FFF C01C）

当管脚配置为 GPIO 输出模式时，可使用该寄存器从管脚输出低电平。写入 1 使对应管脚输出低电平并清零 IOSET 寄存器中相应的位。写入 0 无效。如果一个管脚被配置为输入或第二功能，那么写 IOCLR 无效。

当通过 FIO0CLR 寄存器支持增强型的 GPIO 时，遗留寄存器为 IO0CLR。通过 FIOCLR 寄存器访问端口管脚的状况受到相应的 FIOMASK 寄存器控制（见 8.4.2 节“高速 GPIO 端口 0 屏蔽寄存器(FIOMASK，端口 0：FIO0MASK – 0x3FFF C010）。

表 77 GPIO 端口 0 输出清零寄存器 0(IO0CLR – 地址 0xE002 800C)位描述

位	符号	描述	复位值
31:0	P0xCLR	低速 GPIO 输出值 CLEAR 位。IO0CLR 的位 0 对应于 P0.0 … 位 31 对应于 P0.31。	0x0000 0000

表 78 高速 GPIO 端口 0 输出清零寄存器 0（FIO0CLR – 地址 0x3FFF C01C）位描述

位	符号	描述	复位值
31:0	FP0xCLR	高速 GPIO 输出值 CLEAR 位。FIO0CLR 的位 0 对应于 P0.0 … 位 31 对应于 P0.31。	0x0000 0000

除了 32 位长和仅为字访问的 FIOCLR 寄存器外，可通过表 79 中列出的几个字节和半字访问寄存器来控制每个高速的 GPIO 端口。其次是提供如 FIOCLR 寄存器相同的功能，这些附加的寄存器允许更方便和更快地访问物理端口管脚。

表 79 高速 GPIO 端口 0 输出清零字节和可半字访问的寄存器描述

寄存器名称	寄存器长度(位)&访问	地址	描述	复位值
FIO0CLR0	8（字节）	0x3FFF C01C	高速GPIO端口0输出清零寄存器0。FIO0CLR0寄存器中的位 0 对应 P0.0…位 7 对应 P0.7。	0x00
FIO0CLR1	8（字节）	0x3FFF C01D	高速GPIO端口0输出清零寄存器1。FIO0CLR1寄存器中的位 0 对应 P0.8…位 7 对应 P0.15。	0x00
FIO0CLR2	8（字节）	0x3FFF C01E	高速GPIO端口0输出清零寄存器2。FIO0CLR2寄存器中的位 0 对应 P0.16…位 7 对应 P0.23。	0x00

续上表

寄存器名称	寄存器长度(位)&访问	地址	描述	复位值
FIO0CLR3	8（字节）	0x3FFF C01F	高速 GPIO 端口 0 输出清零寄存器 3。FIO0CLR3 寄存器中的位 0 对应 P0.24…位 7 对应 P0.31。	0x00
FIO0CLRL	16（半字）	0x3FFF C01C	高速 GPIO 端口 0 输出清零低半字寄存器。FIO0CLRL 寄存器中的位 0 对应 P0.0…位 15 对应 P0.15。	0x0000
FIO0CLRU	16（半字）	0x3FFF C01E	高速 GPIO 端口 0 输出清零高半字寄存器。FIO0CLRU 寄存器中的位 0 对应 P0.16…位 15 对应 P0.31。	0x0000

8.5 GPIO 使用注意事项

8.5.1 例 1：顺序访问 IOSET 和 IOCLR 寄存器对同一个 GPIO 管脚/位的影响

GPIO 管脚配置的输出状态由写入 IOSET 和 IOCLR 寄存器的值决定。IOSET/IOCLR 两者中后访问的寄存器将决定管脚的最终输出状态。

代码如下：

```
IO0DIR = 0x0000 0080          ; P0.7 配置用作输出
IO0CLR = 0x0000 0080          ; P0.7 输出为低
IO0SET = 0x0000 0080          ; P0.7 输出为高
IO0CLR = 0x0000 0080          ; P0.7 输出为低
```

先将 P0.7 设置成输出（写 IO0DIR 寄存器）；然后，P0.7 输出设为低电平（先写 IO0CLR 寄存器）；接着，P0.7 管脚上出现短高电平脉冲（写 IO0SET）；最后，写 IO0CLR 寄存器又将 P0.7 输出设置成低电平。

8.5.2 例 2：使 GPIO 管脚上输出瞬时的 0 和 1

先写 IOSET 再写 IOCLR 寄存器可使管脚先输出 1 再输出 0。有的系统允许有效输出的这段延时时间。但某些应用要求一个 GPIO 口的一组管脚同时输出一个二进制数（0 和 1 混合）。这可通过写端口的 IOPIN 寄存器来实现。

下面的代码所实现的功能是：P0.[31:16]和 P0.[7:0]输出保持不变的同时将 P0.[15:8]设置成 0xA5，不管 P0.[15:8]之前是何值：

```
IO0PIN＝(IO0PIN && 0xFFFF00FF)‖0x0000A500
```

使用高速端口访问可以得到相同的结果。

方法 1：使用 32 位（字）可访问高速 GPIO 寄存器

```
FIO0MASK = 0xFFFF00FF;
FIO0PIN = 0x0000A500;
```

方法 2：使用 16 位（半字）可访问高速 GPIO 寄存器

```
FIO0MASKL = 0x00FF;
FIO0PINL = 0xA500;
```

方法 3：使用 8 位（字节）可访问高速 GPIO 寄存器

```
FIO0PIN1 = 0xA5;
```

8.5.3 写 IOSET/IOCLR .vs. IOPIN

写 IOSET/IOCLR 寄存器很容易使所选输出管脚的状态同时变为高/低电平。只有 IOSET/IOCLR 中被写入 1 的位对应的管脚才能设置为高/低电平，写入 0 的位对应的管脚的状态不发生改变。但是，仅通过写 IOSET 或 IOCLR 寄存器是不可能使一个 GPIO 口同时输出包含 0 和 1 的任意二进制数。

写 IOPIN 寄存器使能并行 GPIO 输出所需的瞬时值。写入 IOPIN 寄存器的二进制数据将影响整个并行口的所有输出配置管脚：IOPIN 的位为 0 时使管脚输出低电平，IOPIN 的位为 1 时使管脚输出高电平。为了改变一组端口管脚的输出，必须将 IOPIN 读出的内容和一个值相与（该值使要改变的管脚对应的位用 0 来屏蔽，其它管脚对应的位为 1）。最后，再将相与的结果和期望得到的管脚输出对应的二进制数相或并将相或的结果存回 IOPIN 寄存器。上面例 2 的等式中实现的功能是 P0.15～P0.8 输出 0xA5，P0 口其它位的状态保持不变。

8.5.4 当使用遗留和增强型 GPIO 寄存器时输出信号频率需要考虑的事项

在该微控制器上可用的高速 GPIO 端口的增强型特性使 GPIO 脚更好地响应能控制它们的代码。特别地，软件通过高速 GPIO 寄存器访问 GPIO 脚的时间比软件使用遗留寄存器集的时间快了 3.5 倍。从而随着访问速度的增加，数字管脚最大的输出频率也增加了 3.5 倍。使用无格式的 C 代码时输出频率的这种极大增加不会总是能看得到的，并且处理高速端口输出的应用部分将需要用汇编代码来写且在 ARM 模式中执行。

以下是在管脚控制部分用 ARM 的汇编语言写的代码。首先，端口 0 配置为低速端口，且程序在 P0.20 上产生两个脉冲。接着端口 0 配置为高速端口，且在 P0.16 上产生两个脉冲。这描述了高速和低速 GPIO 端口输出能力的差别。一旦该代码在 ARM 模式中编译，其片内 Flash 的执行将在配置 MAM 模块时（在 4.9 节“MAM 使用注意事项”中描述）产生最好的结果。片内 SRAM 的执行与 MAM 的设置相独立。

```
        /*设置端口 0 为低速 GPIO */
    ldr r0,=0xe01fc1a0     /*寄存器地址—SCS 寄存器*/
    mov r1,#0x0            /*设置位 0 为 0*/
    str r1,[r0]             /*使能低速端口*/
    ldr r1,=0xffffffff       /* */
    ldr r0,=0xe0028008     /*寄存器地址--IODIR*/
    str r1,[r0]             /*设置端口 0 为输出*/
    ldr r2,=0xe0010000      /*选择 P0.20*/
    ldr r0,=0xe0028004      /*寄存器地址--IOSET*/
    ldr r1,=0xe002800C      /*寄存器地址--IOCLR*/
        /*在 P0.20 上使用低速 GPIO 产生两个脉冲*/
    str r2,[r0]              /*高电平*/
    ldr r0,[r1]              /*低电平*/
    str r2,[r0]              /*高电平*/
    ldr r0,[r1]              /*低电平*/
        /*设置 P0 口为高速 GPIO */
```

```
    ldr r0,=0xe01fc1a0        /*寄存器地址—使能高速端口*/
    mov r1,#0x1
    str r1,[r0]               /*使能高速端口 0*/
    ldr r1,=0xffffffff
    ldr r0,=0x3fffc000        /*高速端口 0 的方向*/
    str r1,[r0]
    ldr r0,=0x3fffc018        /*FIO0SET – 高速端口 0 寄存器*/
    ldr r1,=0x3fffc01c        /*FIO0CLR0 – 高速端口 0 寄存器*/
    ldr r2,=0xC0010000        /*选择高速端口 0.16 用于触发*/
        /*在高速端口上产生两个脉冲*/
    str r2,[r0]
    str r2,[r1]
    str r2,[r0]
    str r2,[r1]
loop: b loop
```

图 16 描绘了上述从 LPC2101/02/03 Flash 存储器开始执行的代码。PLL 产生的 F_{CCLK}=60MHz 超出了外部 F_{OSC}=12MHz。MAM 通过 MEMCR=2 和 MEMTIM=3 完全使能，且 APBDIV=1(PCLK=CCLK)。

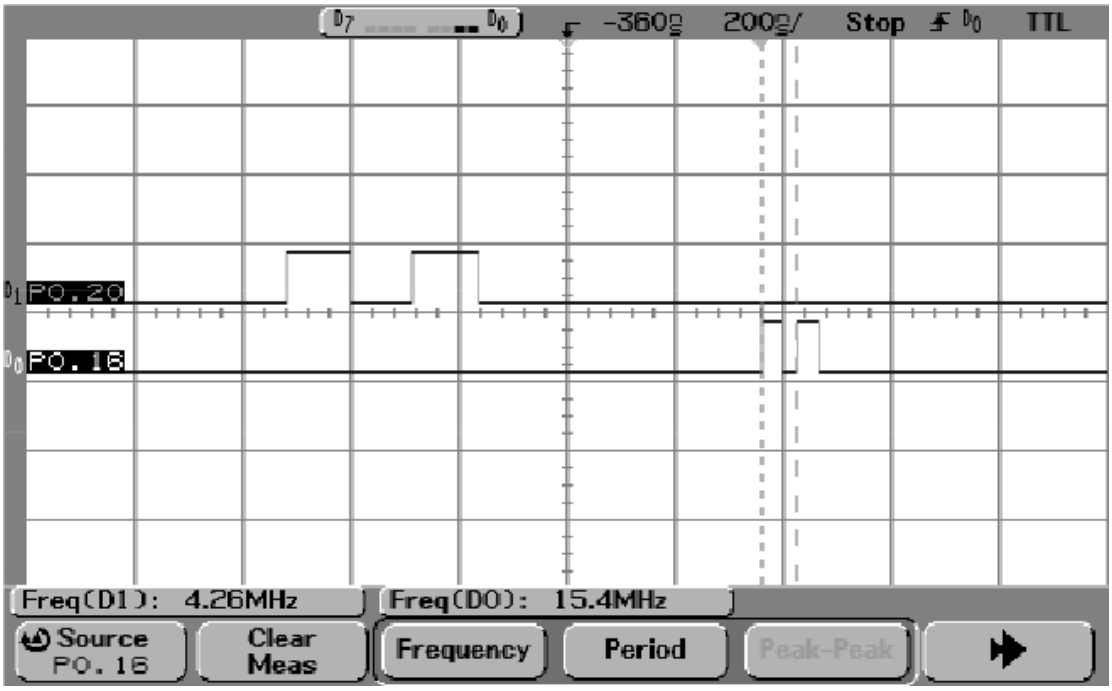


图 16　高速和低速 GPIO 访问和输出显示管脚输出频率的 3.5x 增量

第9章 通用异步接收器/发送器 0 (UART0)

9.1 特性

- 16 字节收发 FIFO
- 寄存器位置符合’550 工业标准
- 接收器 FIFO 触发点可为 1, 4, 8 和 14 字节
- 内置带波特率功能的波特率发生器
- 包含使能实现软件和硬件流控制的机制

9.2 管脚描述

表 80　UART0 管脚描述

管脚名称	类型	描述
RxD0	输入	**串行输入。** 串行接收数据。
TxD0	输出	**串行输出。** 串行发送数据。

9.3 寄存器描述

UART0 包含表 81 中所示的寄存器。除数锁存访问位（DLAB）位于 U0LCR[7]，它使能对除数锁存的访问。

表 81　UART0 寄存器映射

名称	描述	位功能和地址								访问	复位值 [1]	地址
		MSB							LSB			
		BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0			
U0RBR	接收缓冲寄存器	8 位读数据								RO	NA	0xE000C000 DLAB=0
U0THR	发送保持寄存器	8 位写数据								WO	NA	0xE000C000 DLAB=0
U0DLL	除数锁存 LSB	8 位数据								R/W	0x01	0xE000C000 DLAB=1
U0DLM	除数锁存 MSB	8 位数据								R/W	0x00	0xE000C004 DLAB=1
U0IER	中断使能寄存器	-	-	-	-	-	-	En.ABTO	En.ABEO	R / W	0 x 0 0	0xE000C004 DLAB=0
		-	-	-	-	-	使能 Rx 线状态中断	使能 THRE 中断	使用 RX 数据可用中断			

续上表

名称	描述	位功能和地址								访问	复位值[1]	地址
		MSB							LSB			
		BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0			
U0IIR	中断 ID 寄存器	-	-	-	-	-	-	ABTO 中断	ABEO 中断	RO	0x01	0xE000C008
		FIFO 使能		-	-	IIR3	IIR2	IIR1	IIR0			
U0FCR	FIFO 控制寄存器	Rx 触发		-	-	-	Tx FIFO 复位	Rx FIFO 复位	FIFO 使能	WO	0x00	0xE000C008
U0LCR	线控制寄存器	DLAB	设置间隔	奇偶固定	偶选择	奇偶使能	停止位个数	字长度选择		R/W	0x00	0xE000C00C
U0LSR	线状态寄存器	Rx FIFO 错误	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	0xE000C014
U0SCR	高速缓存	8 位数据								R/W	0x00	0xE000C01C
U0ACR	自动波特率控制寄存器	-	-	-	-	-	-	ABTO Int.Clr	ABEO Int.Clr	R/W	0x00	0xE000C020
		-	-	-	-	-	Aut.Rstrt.	模式	启动			
U0FDR	小数分频寄存器	保留[31:8]									0x10	0xE000C028
		MulVal				DivAddVal						
U0TER	发送使能寄存器	TxEn	-	-	-	-	-	-	-	R/W	0x80	0xE000C030

[1] 复位值仅指已使用位中保存的数据。它不包括保留位的内容。

9.3.1 UART0 接收器缓存寄存器（U0RBR - 0xE000 C000，DLAB=0，只读）

U0RBR 是 UART0 Rx FIFO 的最高字节。它包含了最早接收到的字符，可通过总线接口读出。LSB（bit0）代表最早接收到的数据位。如果接收到的字符小于 8 位，未使用的 MSB 填充为 0。

如果要访问 U0RBR，U0LCR 的除数锁存访问位（DLAB）必须为 0。U0RBR 为只读寄存器。

由于 PE、FE 和 BI 位与 RBR FIFO 顶端的字节相对应（即下次读 RBR 时读出的字节），因此，将接收的字节及其状态位成对读出的正确方法是先读 U0LSR 寄存器的内容，再读 U0RBR 的字节。

表 82　UART0 接收器缓存寄存器（U0RBR – 地址 0xE000 C000，DLAB=0，只读）位描述

位	符号	描述	复位值
7:0	RBR	UART0 接收器缓存寄存器包含 UART0 Rx FIFO 中最早接收到的字节	未定义

9.3.2 UART0 发送器保持寄存器（U0THR - 0xE000C000，DLAB=0，只写）

U0THR 是 UART0 Tx FIFO 的最高字节。它包含了 Tx FIFO 中最新的字符，可通过总线接口写入。LSB 代表最先发送的位。

如果要访问 U0THR，U0LCR 的除数锁存访问位（DLAB）必须为 0。U0THR 为只写寄

存器。

表 83　UART0 发送器保持寄存器（U0THR – 地址 0xE000 C000，DLAB=0，只写）位描述

位	符号	描述	复位值
7:0	THR	写 UART0 发送器保持寄存器使数据保存到 UART0 发送 FIFO 当中。当字节到达 FIFO 的最底部并且发送器就绪时，该字节将被发送。	NA

9.3.3 UART0 除数锁存寄存器（U0DLL - 0xE000 C000 和 U0DLM - 0xE000 C004，当 DLAB=1 时）

UART0 除数锁存是 UART0 波特率发生器的一部分，它保存了用于产生波特率时钟的 APB 时钟（PCLK）分频值，波特率时钟必须是波特率的 16 倍（等式 1）。U0DLL 和 U0DLM 寄存器一起构成一个 16 位除数，U0DLL 包含除数的低 8 位，U0DLM 包含除数的高 8 位。值 0x0000 被看作是 0x0001，因为除数是不允许为 0 的。当访问 UART0 除数锁存寄存器时，U0LCR 中的除数锁存访问位（DLAB）必须为 1。

如何选择 U0DLL 和 U0DLM 正确值的细节在本章后面的部分描述。

表 84　UART0 除数锁存 LSB 寄存器（U0DLL – 地址 0xE000 C000，DLAB=1）位描述

位	符号	描述	复位值
7:0	DLL	UART0 除数锁存 LSB 寄存器与 U0DLM 寄存器一起决定 UART0 的波特率。	0x01

表 85　UART0 除数锁存 MSB 寄存器（U0DLM – 地址 0xE000C004，DLAB=1）位描述

位	符号	描述	复位值
7:0	DLM	UART0 除数锁存 MSB 寄存器与 U0DLL 寄存器一起决定 UART0 的波特率。	0x00

9.3.4 UART0 小数分频寄存器（U0FDR – 0xE000 C028）

UART0 小数分频寄存器(U0FDR)控制产生波特率的时钟预分频器并可在用户的判断下被读写。该预分频器在每次指定的小数要求中使用 APB 时钟和产生一个输出时钟。

表 86　UART0 小数分频寄存器(U0FDR – 地址 0xE000C028)位描述

位	功能	描述	复位值
3:0	DIVADDVAL	波特率生成预分频除数值。如果该字段为 0，小数波特率发生器将不会影响 UART0 波特率。	0
7:4	MULVAL	波特率预分频乘数值。不管小数波特率发生器是否使用，该字段必须大于或等于 1 以使 UART0 正常操作。	1
31:8	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

该寄存器控制波特率生成的时钟预分频器。该寄存器的复位值保持 UART0 禁能时的小数功能，确保 UART0 的软件和硬件与没有该特性的 UART 完全兼容。

可用下面的等式计算 UART0 波特率：

$$UART0_{baudrate}=\frac{PCLK}{16x(16xU0DLM+U0DLL)X(1+\frac{DivAddVal}{MulVal})} \quad (1)$$

其中 PCLK 为外围时钟，U0DLM 和 U0DLL 为标准的 UART0 波特率除数寄存器，DIVADDVAL 和 MULVAL 为 UART0 小数波特率发生器特定的参数。

MULVAL 和 DIVADDVAL 的值应遵循以下的条件：

1. 0＜MULVAL≤15
2. 0≤DIVADDVAL≤15

如果 U0FDR 寄存器值不遵循这两个请求，那么小数分频输出未定义。如果 DIVADDVAL 为 0，那么小数分频禁止且时钟将不会被分频。

当正在发送/接收数据或数据可能丢失或破坏时，不应修改 U0FDR 的值。

使用注释：在实际使用中，UART0 波特率公式可用下面的形式表示，这种形式识别没有小数波特率发生器时生成的 UART 波特率部分，以及该模块添加的校准系数：

$$UART0_{baudrate}=\frac{PCLK}{16x(16xU0DLM+U0DLL)}X\frac{MulVal}{MulVal+DivAddVal} \quad (2)$$

根据这种表示，小数波特率发生器也可描述为具有 MULVAL/(MULVAL+DIVADDVAL) 系数的预分频。

9.3.5 UART0 波特率计算

例 1：使用上面的 $UART0_{baudrate}$ 公式，假设系统的 PCLK＝20MHz，如果 U0DL=130（U0DLM=0x00 和 U0DLL=0x82）、DIVADDVAL＝0 和 MULVAL＝1，则可得出 UART0 的 $UART0_{baudrate}$=9615 bauds。

例 2：使用上面的 $UART0_{baudrate}$ 公式，假设系统的 PCLK＝20MHz，如果 U0DL=93（U0DLM=0x00 和 U0DLL=0x5D）、DIVADDVAL＝2 和 MULVAL＝5，则可得出 UART0 的 $UART0_{baudrate}$=9600 bauds。

表 87　外围时钟为 20MHz 时的波特率(PCLK=20MHz)

期望的波特率	MULVAL=0　DIVADDVAL=0			可选的 MULVAL & DIVADDVAL		
	U0DLM:U0DLL hex[2]	dec[1]	%error[3]	U0DLM:U0DLL dec[1]	小数的预分频值 $\frac{MULDIV}{MULDIV+DIVADDVAL}$	%error[3]
50	61A8	25000	0.0000	25000	1/(1+0)	0.0000
75	411B	16667	0.0020	12500	3/(3+1)	0.0000
110	2C64	11364	0.0032	6250	11/(11+9)	0.0000
134.5	244E	9294	0.0034	3983	3/(3+4)	0.0001
150	208D	8333	0.0040	6250	3/(3+1)	0.0000
300	1047	4167	0.0080	3125	3/(3+1)	0.0000
600	0823	2083	0.0160	1250	3/(3+2)	0.0000
1200	0412	1042	0.0320	625	3/(3+2)	0.0000

续上表

期望的波特率	MULVAL=0　DIVADDVAL=0			可选的 MULVAL & DIVADDVAL		
	U0DLM:U0DLL hex[2]	dec[1]	%error[3]	U0DLM:U0DLL dec[1]	小数的预分频值 $\frac{MULDIV}{MULDIV+DIVADDVAL}$	%error[3]
1800	02B6	694	0.0640	625	9/(9+1)	0.0000
2000	0271	625	0.0000	625	1/(1+0)	0.0000
2400	0209	521	0.0320	250	12/(12+13)	0.0000
3600	015B	347	0.0640	248	5/(5+2)	0.0064
4800	0104	260	0.1600	125	12/(12+13)	0.0000
7200	00AE	174	0.2240	124	5/(5+2)	0.0064
9600	0082	130	0.1600	93	5/(5+2)	0.0064
19200	0041	65	0.1600	31	10/(10+11)	0.0064
38400	0021	33	1.3760	12	7/(7+12)	0.0594
56000	0021	22	1.4400	13	7/(7+5)	0.0160
57600	0016	22	1.3760	19	7/(7+1)	0.0594
112000	000B	11	1.4400	6	7/(7+6)	0.1600
115200	000B	11	1.3760	4	7/(7+12)	0.0594
224000	0006	6	7.5200	3	7/(7+6)	0.1600
448000	0003	3	7.5200	2	5/(5+2)	0.3520

[1] 这一行的值是 16 位（DLM:DLL）的等效十进制值。

[2] 这一行的值是 16 位（DLM:DLL）的等效十六进制值。

[3] 期望波特率和实际波特率的百分比误差。

9.3.6 UART0 中断使能寄存器（U0IER - 0xE000 C004，DLAB=0）

U0IER 用于使能 UART0 中断源。

表 88　UART0 中断使能寄存器(U0IER–地址 0xE000 C004，DLAB=0)位描述

位	符号	值	描述	复位值
0	RBR 中断使能		U0IER[0]使能 UART0 接收数据可用中断。它还控制字符接收超时中断。	0
		0	禁止 RDA 中断	
		1	使能 RDA 中断	
1	THRE 中断使能		U0IER[1]使能 UART0 THRE 中断。该中断的状态可从 U0LSR[5]读出。	0
		0	禁止 THRE 中断	
		1	使能 THRE 中断	
2	Rx 线状态中断使能		U0IER[2]使能 UART0 Rx 线状态中断。该中断的状态可从 U0LSR[4:1]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，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA
8	ABTO IntEn		U0IER8 使能自动波特率超时中断。	0
		0	禁止自动波特率超时中断。	
		1	使能自动波特率超时中断。	

续上表

位	符号	值	描述	复位值
9	ABEO IntEn	0 1	U0IER9 使能自动波特率中断结束。 禁止自动波特率中断结束。 使能自动波特率中断结束。	0
31:10	-	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

9.3.7 UART0 中断标识寄存器（U0IIR - 0xE000 C008，只读）

U0IIR 提供状态代码用于指示一个挂起中断的中断源和优先级。在访问 U0IIR 过程中，中断被冻结。如果在访问 U0IIR 时产生了中断，该中断被记录，下次 U0IIR 访问可读出。

表 89　UART0 中断标识寄存器（U0IIR – 地址 0xE000 C008，只读）位描述

位	符号	值	描述	复位值
0	中断挂起	0 1	U0IIR[0]为低有效。挂起的中断可通过 U0IIR[3:1]确定。 至少有 1 个中断被挂起 没有挂起的中断	1
3:1	中断标识	011 010 110 001	U0IER[3:1]指示对应于 UART0 Rx FIFO 的中断。上面未列出的 U0IER[3:1]的其它组合都为保留值（000, 100, 101, 111） 1. 接收线状态（RLS） 2a. 接收数据可用（RDA） 2b. 字符超时指示（CTI） 3. THRE 中断	0
5:4	-	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA
7:6	FIFO 使能		这些位等效于 U0FCR[0]	0
8	ABEOInt		自动波特率中断的结束。如果成功完成波特率且中断被使能则 ABEOInt 为 1。	0
9	ABTOInt		自动波特率超时中断。如果波特率超时且中断被使能，则 ABTOInt 为 1。	0
31:10	-	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

中断的处理见表 90。给定了 U0IIR[3:0]的状态，中断处理程序就能确定中断源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 U0IIR 来清除中断。

UART0 RLS 中断（U0IIR[3:1]=011）是最高优先级的中断。只要 UART0 Rx 输入产生 4 个错误条件（溢出错误（OE）、奇偶错误（PE）、帧错误（FE）和间隔中断（BI））中的任意一个，该中断标志将置位。产生该中断的 UART0 Rx 错误条件可通过查看 U0LSR[4:1]得到。当读取 U0LSR 时清除中断。

UART0 RDA 中断（U0IIR[3:1]=010）与 CTI 中断（U0IIR[3:1]=110）共用第二优先级。当 UART0 Rx FIFO 到达 U0FCR[7:6]所定义的触发点时，RDA 被激活。当 UART0 Rx FIFO 的深度低于触发点时，RDA 复位。当 RDA 中断激活时，CPU 可读出由触发点所定义的数据块。

CTI 中断（U0IIR[3:1]=110）为第二优先级中断。当 UART0 Rx FIFO 包含至少 1 个字符

并且在接收 3.5 到 4.5 字符的时间内没有发生 UART0 Rx FIFO 动作时，该中断置位。UART0 Rx FIFO 的任何动作（读或写 UART0 RSR）都将清除该中断。在接收到的信息不是触发值的倍数后，CTI 中断将会清空 UART0 RBR。例如，如果一个外设想要发送一个 105 个字符的信息，而触发值为 10 个字符，那么 CPU 接收 10 个 RDA 中断将导致前 100 个字符的传输，而 CPU 接收 1 到 5 个 CTI 中断（取决于服务程序）将导致剩下 5 个字符的传输。

表 90　UART0 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]	U0LSR 读操作[2]
0100	第二	Rx 数据可用	Rx 数据可用或 FIFO 模式下（U0FCR0=1）到达触发点	U0RBR 读[3]或 UART0 FIFO 低于触发值
1100	第二	字符超时指示	Rx FIFO 包含至少 1 个字符并且在一段时间内无字符输入或移出，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）设置的触发值。 实际的时间为： [(字长度) × 7-2] × 8+[(触发值 - 字符数) × 8+1]RCLK	U0RBR 读操作[3]
0010	第三	THRE	THRE[2]	U0IIR 读（如果是中断源）或 THR 写操作[4]

[1] “0000”, “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”, “1101”, “1110”, “1111”为保留值。

[2] 详见 9.3.10 节“UART0 线状态寄存器(U0LSR – 0xE000 C014，只读)”02

[3] 详见 9.3.1 节“UART0 接收器缓存寄存器(U0RBR – 0xE000 C000, DLAB=0，只读)”

[4] 详见 9.3.7 节“UART0 中断识别寄存器(U0IIR - 0xE000 C008，只读)”和 9.3.2 节“UART0 发送保持寄存器(U0THR – 0xE000 C000, DLAB=0，只写)”。

UART0 THRE 中断(U0IIR[3:1]=001)为第三优先级中断。当 UART0 THR FIFO 为空并且满足特定的初始化条件时，该中断激活。这些初始化条件将使 UART0 THR FIFO 被数据填充，以免在系统启动时产生许多 THRE 中断。初始化条件在 THRE=1 时实现了一个字符的延时减去停止位并在上一次 THRE=1 事件之后在 U0THR 中没有存在至少 2 个字符。在没有译码和服务 THRE 中断时，该延迟为 CPU 提供了将数据写入 U0THR 的时间。如果 UART0 THR FIFO 中曾经有两个或更多字符，而当前 U0THR 为空时，THRE 中断立即设置。当发生 U0THR 写操作或 U0IIR 读操作并且 THRE 为最高优先级中断（U0IIR[3:1]=001）时，THRE 中断复位。

9.3.8 UART0 FIFO 控制寄存器（U0FCR - 0xE000 C008）

U0FCR 控制 UART0 Rx 和 Tx FIFO 的操作。

表 91　UART0 FIFO 控制寄存器（U0FCR – 地址 0xE000 C008）位描述

位	符号	值	描述	复位值
0	FIFO 使能	0 1	UART0 FIFO 被禁止。在应用中必须不能使用。 高电平使能对 UART0 Rx 和 Tx FIFO 以及 U0FCR[7:1]的访问。该位必须置位以实现正确的 UART0 操作。该位的任何变化都将使 UART0 FIFO 自动清空。	0
1	Rx FIFO 复位	0 1	对任意的 UART0 FIFO 无影响。 写 1 到 U0FCR[1]将清零 UART0 Rx FIFO 中的所有字节并复位指针逻辑。该位自动清零。	0
2	Tx FIFO 复位	0 1	对任意的 UART0 FIFO 无影响。 写 1 到 U0FCR[2]将清零 UART0 Tx FIFO 中的所有字节并复位指针逻辑。该位自动清零。	0
5:3	—	0	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA
7:6	Rx 触发选择	 00 01 10 11	这两个位决定在激活中断之前，接收器 UART0 FIFO 必须写入多少个字符。 触发点 0（1 个字符或 0x01） 触发点 1（4 个字符或 0x04） 触发点 2（8 个字符或 0x08） 触发点 3（14 个字符或 0x0E）	0

9.3.9 UART0 线控制寄存器（U0LCR - 0xE000 C00C）

U0LCR 决定发送或接收数据字符的格式。

表 92　UART0 线控制寄存器（U0LCR – 地址 0xE000 C00C）位描述

位	符号	值	描述	复位值
1:0	字长度选择	00 01 10 11	5 位字符长度 6 位字符长度 7 位字符长度 8 位字符长度	0
2	停止位选择	0 1	1 个停止位 2 个停止位（如果 U0LCR[1:0]=00 则为 1.5）	0
3	奇偶使能	0 1	禁止奇偶产生和校验 使能奇偶产生和校验	0
5:4	奇偶选择	00 01 10 11	奇数。发送字符中 1 的个数和附带的校验位为奇数。 偶数。发送字符中 1 的个数和附带的校验位为偶数。 强制为“1”，奇偶固定 强制为“0”，奇偶固定	0
6	间隔控制	0 1	禁止间隔发送。 使能间隔发送。当 U0LCR[6]为高电平有效时，输出管脚 UART0 TxD 强制为逻辑 0。	0
7	除数锁存访问位(DLAB)	0 1	禁止访问除数锁存 使能访问除数锁存	0

9.3.10 UART0 线状态寄存器（U0LSR - 0xE000 C014，只读）

U0LSR 为只读寄存器，它提供 UART0 Tx 和 Rx 模块的状态信息。

表 93　UART0 线状态寄存器（U0LSR – 地址 0xE000 C014，只读）位描述

位	符号	值	描述	复位值
0	接收器数据就绪（RDR）	0 1	当 U0RBR 包含未读取的字符时，U0LSR0 置位；当 UART0 RBR FIFO 为空时，U0LSR0 清零。 U0RBR 为空 U0RBR 包含有效数据	0
1	溢出错误（OE）	0 1	溢出错误条件在错误发生后立即设置。U0LSR 读操作清零 U0LSR1。当 UART0 RSR 已经有新的字符组合而 UART0 RBR FIFO 已满时，U0LSR1 置位。此时 UART0 RBR FIFO 不会被覆盖，UART0 RSR 中的字符将丢失。 溢出错误状态未激活 溢出错误状态激活	0
2	奇偶错误（PE）	0 1	当接收字符的校验位处于错误状态时产生一个奇偶错误。U0LSR 读操作清零 U0LSR[2]位。奇偶错误检测时间取决于 U0FCR[0]。 **注：**奇偶错误与 UART0 RBR FIFO 中顶部的字符相关。 奇偶错误状态未激活 奇偶错误状态激活	0
3	帧错误（FE）	0 1	当接收字符的停止位为 0 时，产生帧错误。U0LSR 读操作清零 U0LSR[3]。帧错误检测时间取决于 U0FCR0。当检测到一个帧错误时，Rx 将尝试与数据重新同步并假设错误的停止位实际是一个超前的起始位。但即使没有出现帧错误，它也不能假设下一个接收到的字节是正确的。 **注：**帧错误与 UART0 RBR FIFO 中顶部的字符相关。 帧错误状态未激活 帧错误状态激活	0
4	间隔中断（BI）	0 1	在发送整个字符（起始位、数据、校验位和停止位）过程中 RxD0 如果都保持逻辑 0，则产生间隔中断。当检测到间隔条件时，接收器立即进入空闲状态直到 RxD0 变为全 1 状态。U0LSR 读操作清零该状态位。间隔检测的时间取决于 U0FCR[0]。 **注：**间隔中断与 UART0 RBR FIFO 中顶部的字符相关。 间隔中断状态未激活 间隔中断状态激活	0
5	发送器保持寄存器空（THRE）	0 1	当检测到 UART0 THR 空时，THRE 置位，U0THR 写操作清零该位。 U0THR 包含有效数据 U0THR 为空	1

续上表

位	符号	值	描述	复位值
6	发送器空（TEMT）	0 1	当 U0THR 和 U0TSR 都为空时，TEMT 置位。当 U0TSR 或 U0THR 包含有效数据时，TEMT 清零。 U0THR 和/或 U0TSR 包含有效数据 U0THR 和 U0TSR 为空	1
7	Rx FIFO 错误（RXFE）	0 1	当一个带有 Rx 错误（例如帧错误、奇偶错误或间隔中断）的字符装入 U0RBR 时，U0LSR[7]置位。当读取 U0LSR 寄存器并且 UART0 FIFO 中不再有错误时，U0LSR[7]清零。 U0RBR 中没有 UART0 Rx 错误，或 U0FCR[0]=0 UART0 RBR 包含至少一个 UART0 Rx 错误	0

9.3.11 UART0 高速缓存寄存器（U0SCR – 0xE000 C01C）

在 UART0 操作时 U0SCR 无效。用户可自由对该寄存器进行读或写。不提供中断接口向主机指示 U0SCR 所发生的读或写操作。

表 94　UART0 高速缓存寄存器（U0SCR – 地址 0xE000 C01C）位描述

位	符号	描述	复位值
7:0	Pad	一个可读可写的字节	0x00

9.3.12 UART0 自动波特率控制寄存器(U0ACR – 0xE000 C020)

UART0 自动波特率控制寄存器(U0ACR)控制测量波特率生成的输入时钟/数据率的过程，用户可自由对该寄存器进行读或写。

表 95　自动波特率控制寄存器（U0ACR – 0xE000 C020）位描述

位	符号	值	描述	复位值
0	Start	0 1	自动波特率结束后该位自动清零。 自动波特率停止（自动波特率不运行）。 自动波特率启动（自动波特率正在运行）。自动波特率运行位。该位在自动波特率结束后自动清零。	0
1	Mode	0 1	自动波特率模式选择位。 模式 0 模式 1	0
2	AutoRestart	0 1	不重新启动 如果超时则重新启动（计数器在下一个 UART0 Rx 下降沿重新启动）	0
7:3	—	NA	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	0
8	ABEOIntClr		自动波特率中断结束清零位（仅可写访问）。写 1 将在 U0IIR 中清除相应的中断。写 0 无影响。	0
9	ABTOIntClr		自动波特率超时中断清零位（仅可写访问）。写 1 将在 U0IIR 中清除相应的中断。写 0 无影响。	0
31:10	—	NA	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	0

9.3.13 自动波特率

UART0 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。如果使能，那么自动波特率特性将测量接收数据流的位时并因此设置除数锁存寄存器 U0DLM 和 U0DLL。

自动波特率通过设置 U0ACR 起始位来启动。自动波特率可通过清零 U0ACR 起始位来停止。一旦自动波特率结束起始位将清零，并且读该位将返回自动波特率的状态（等待/完成）。

可通过选择 U0ACR 模式位来使用两种自动速率测量模式。在模式 0 中，波特率在 UART0 Rx 管脚的两个连续的下降沿上测量（起始位的下降沿和最低位的下降沿）。在模式 1 中，波特率在 UART0 Rx 管脚的下降沿和后续的上升沿之间测量（起始位的长度）。

如果超时出现（速率测量计数器溢出），那么 U0ACR AutoRestart 位可用于自动重新启动波特率测量。如果该位置位，速率测量将在 UART0 Rx 管脚的下一个下降沿重新启动。

自动波特率功能可产生两个中断。

如果中断使能，那么将设置 U0IIR ABTOInt 中断（U0IER ABToIntEn 置位且自动波特率测量计数器溢出）。

如果中断使能，那么将设置 U0IIR ABEOInt 中断（U0IER ABEoIntEn 置位且自动波特率成功完成）。

通过置位相应的 U0ACR ABTOIntClr 和 ABEOIntEn 位来清零自动波特率中断。

在自动波特率期间，小数波特率发生器被禁能(DIVADDVAL=0)。然而，如果小数波特率发生器被使能(DIVADDVAL＞0)，那么它将影响 UART0 Rx 管脚波特率的测量，但 U0FDR 寄存器的值在速率测量后不会被修改。同时，当使用波特率时，任何写 U0DLM 和 U0DLL 寄存器的操作应在写 U0ACR 寄存器前完成。UART0 支持的波特率最小和最大值受 PCLK, 数据位的个数、停止位和校验位影响。

$$\text{ratemin}=\frac{2\times \text{PCLK}}{16\times 2^{15}} \leqslant \text{UART0}_{\text{baudrate}} \leqslant \frac{\text{PCLK}}{16\times(2+\text{数据位}+\text{奇偶位}+\text{停止位})}=\text{ratemax} \qquad (3)$$

9.3.14 UART0 发送使能寄存器（U0TER – 0xE000 C030）

LPC2101/02/03 的 U0TER 使能实现软件流控制。当 TxEn＝1 时，只要数据可用，UART0 发送器就将持续发送数据。一旦 TxEn 变为 0，UART0 发送器立刻停止工作。

表 96 描述了如何利用 TxEn 位来实现软件流控制。

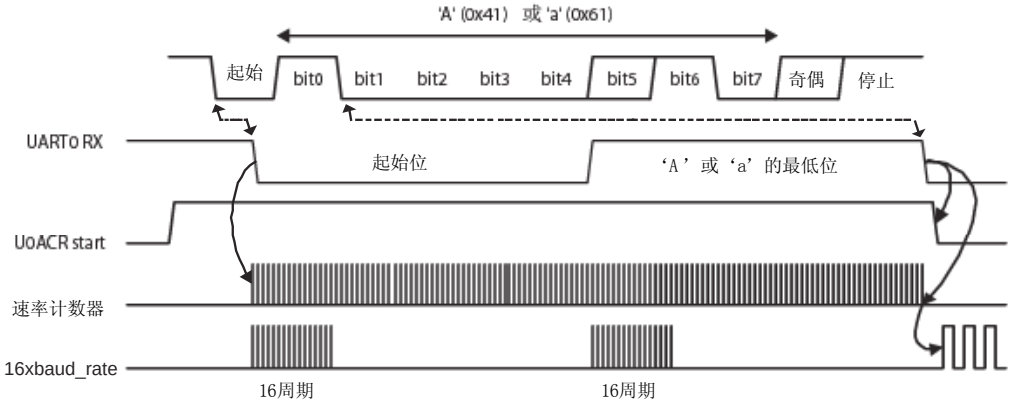
表 96　UART0 发送使能寄存器（U0TER – 地址 0xE000 C030）位描述

位	符号	描述	复位值
6:0	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
7	TxEn	该位为 1 时（复位后），一旦以前的数据都被发送后，写入 THR 的数据就输出到 TxD 管脚。如果一个字符正在发送时该位清零，则结束这个字符的发送，不发送更多的字符直至该位重新置位。换言之，该位为 0 将终止从 THR 或 Tx FIFO 到发送移位寄存器的字符传输。当接收到 XOFF 字符（DC3）时，软件（可实现软件握手）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

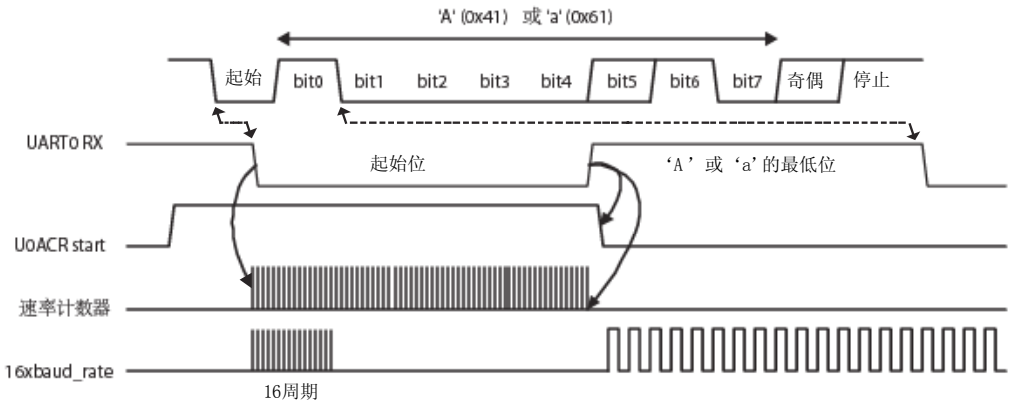
9.3.15 自动波特率模式

当软件正在期望“AT”命令时，它用期望的字符格式配置 UART0 并设置 U0ACR 起始位。可以不用关心除数锁存器 U0DLL 和 U0DLM 的初始值。由于“A”或“a”ASCII 编码（“A”=0x41, “a”=0x61），因此 UART0 Rx 管脚检测起始位且所期望字符的 LSB 由两个下降沿限定。当 U0ACR 起始位置位时，自动波特率协议将执行以下过程：

1. 一旦 U0ACR 起始位置位，波特率测量计数器复位，同时 UART0 U0RSR 复位。U0RSR 波特率切换为最高的速率。
2. UART0 Rx 管脚下降沿触发起始位的开始。速率测量计数器将开始对 PCLK（可选择被小数波特率发生器预分频）进行计数。
3. 在起始位的接收过程中，RSR 的波特率输入端产生 16 个具有 UART0 输入时钟频率（被小数波特率发生器预分频）的脉冲（脉冲频率和 UART0 输入时钟频率一致），保证起始位存储在 U0RSR 中。
4. 在接收起始位（以及模式 0 下字符的 LSB）的过程中，速率计数器将随着经过预分频的 UART0 输入时钟（PCLK）继续增加。
5. 如果 Mode=0，那么速率计数器将在 UART0 Rx 管脚的下一个下降沿停止。如果 Mode=1，那么速率计数器将在 UART0 Rx 管脚的下一个上升沿停止。
6. 速率计数器的值被装入到 U0DLM/U0DLL，并且波特率将自动切换为正常操作模式。U0DLM/U0DLL 设置完成后，如果自动波特率结束中断被使能的话，U0IIR ABEOInt 将被置位。接下来 U0RSR 将继续接收“A/a”字符剩下的位。



a. 模式0（起始位和LSB用于自动波特率）



b. 模式1（仅起始位用于自动波特率）

图 17　自动波特率 a) 模式 0 和 b) 模式 1 波形

9.4 结构

UART0 的结构如下方框图所示。

APB 接口提供 CPU 或主机与 UART0 之间的通信连接。

UART0 接收器模块 U0Rx 监视串行输入线 RxD0 的有效输入。UART0 Rx 移位寄存器（U0RSR）通过 RxD0 接收有效的字符。当 U0RSR 接收到一个有效字符后，它将该字符传送到 UART0 Rx 缓存寄存器 FIFO 中，等待 CPU 或主机通过主机接口进行访问。

UART0 发送器模块 U0Tx 接收 CPU 或主机写入的数据并将数据缓存到 UART0 Tx 保持寄存器 FIFO（U0THR）中。UART0 Tx 移位寄存器（U0TSR）读取 U0THR 中的数据并将数据通过串行输出管脚 TxD0 发送。

UART0 波特率发生器模块 U0BRG 产生 UART0 Tx 模块所使用的定时。U0BRG 模块时钟源为 APB 时钟（PCLK）。主时钟与 U0DLL 和 U0DLM 寄存器所定义的除数相除得到 UART0 Tx 模块使用的时钟。该时钟为 16 倍过采样时钟 NBAUDOUT。

中断接口包含寄存器 U0IER 和 U0IIR。中断接口接收几个由 U0Tx 和 U0Rx 模块发出的单时钟宽度的使能信号。

U0Tx 和 U0Rx 的状态信息保存在 U0LSR 中。U0Tx 和 U0Rx 的控制信息保存在 U0LCR 中。

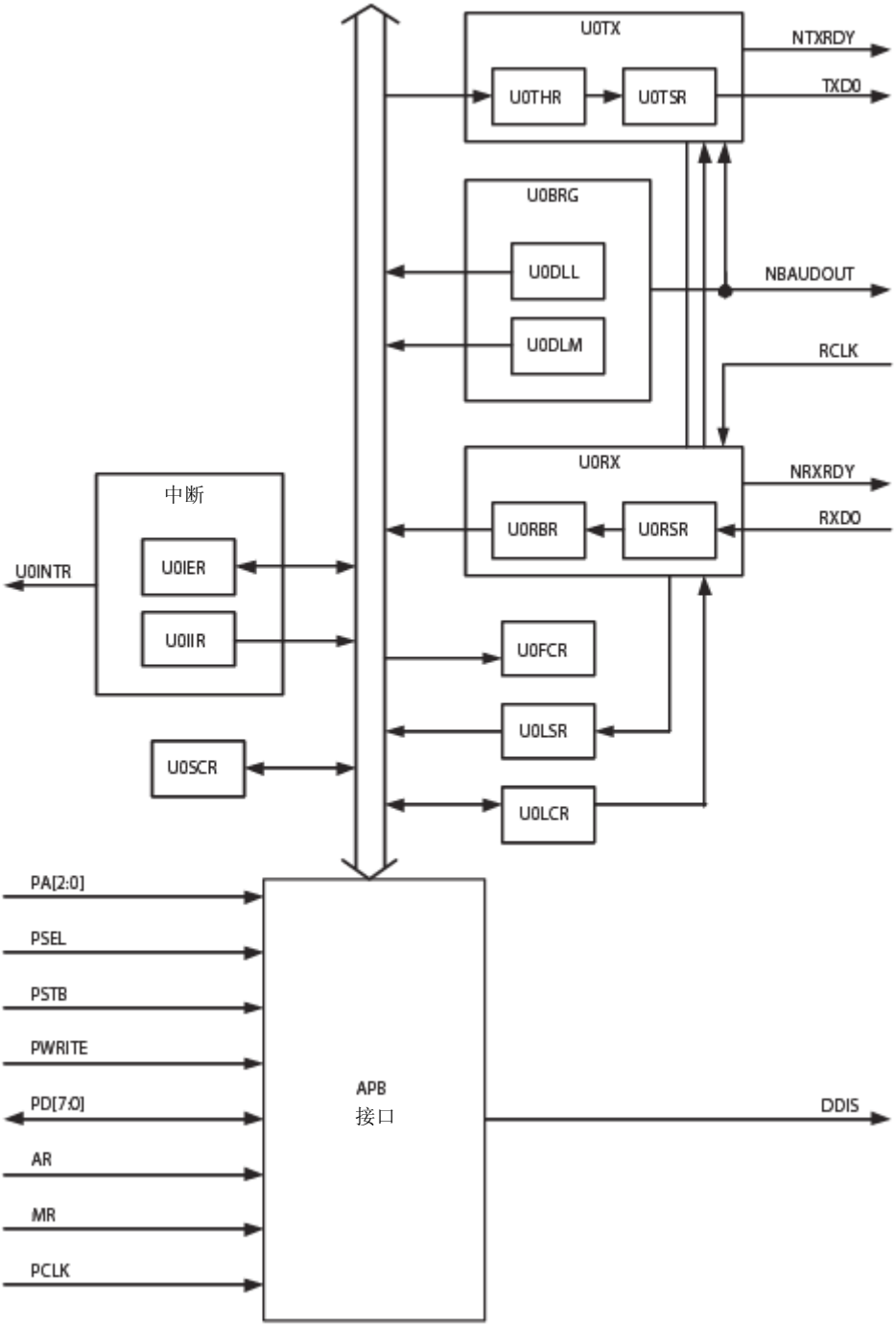


图 18 UART0 方框图

第10章 通用异步接收器/发送器 1（UART1）

10.1 特性

UART1 与 UART0 相同，只是增加了一个调制解调器（modem）接口

UART1 包含 16 字节接收和发送 FIFO

寄存器位置符合'550 工业标准

接收器 FIFO 触发点可为 1, 4, 8 和 14 字节

内置波特率发生器

使能实现软件和硬件流控制的机制

含有标准调制解调器接口信号，且硬件完全支持流控制(auto-CTS/RTS)

10.2 管脚描述

表 97 UART1 管脚描述

管脚名称	类型	描述
RxD1	输入	**串行输入。**串行接收数据。
TxD1	输出	**串行输出。**串行发送数据。
CTS1	输入	**清零发送。**低电平有效信号指示外部 modem 的接收是否已经准备就绪，UART1 数据可通过 TxD1 发送。在 modem 接口的正常操作中(U1MCR[4]=0)，该信号的补码保存在 U1MSR[4]中。状态改变信息保存在 U1MSR[0]中，如果第 4 优先级中断使能(U1IER[3]=1)，该信息将作为中断源。
DCD1	输入	**数据载波检测**。低有效信号指示外部 modem 是否已经与 UART1 建立了通信连接，可以进行数据交换。在 modem 接口的正常操作中(U1MCR[4]=0)，该信号的补码保存在 U1MSR[7]中。状态改变信息保存在 U1MSR[3]中，如果第 4 优先级中断使能(U1IER[3]=1)，该信息将作为中断源。
DSR1	输入	**数据设备就绪。**低电平有效指示外部 modem 是否准备建立与 UART1 的通信连接。在 modem 接口的正常操作中(U1MCR[4]=0)，该信号的补码保存在 U1MSR[5]中。状态改变信息保存在 U1MSR[1]中，如果第 4 优先级中断使能(U1IER[3]=1)，该信息将作为中断源。
DTR1	输出	**数据终端就绪。**低电平有效指示 UART1 准备建立与外部 modem 的连接。该信号的补码保存在 U1MCR[0]中。
RI1	输入	**铃响指示。**有效低电平指示 modem 检测到电话的响铃信号。在 modem 接口的正常操作中(U1MCR[4]=0)，该信号的补码保存在 U1MSR[6]中。状态改变信息保存在 U1MSR[2]中，如果第 4 优先级中断使能(U1IER[3]=1)，该信息将作为中断源。
RTS1	输出	**请求发送。**低电平有效指示 UART1 打算向外部 modem 发送数据。该信号的补码保存在 U1MCR[1]中。

10.3 寄存器描述

UART1 包含表 76 中所示的寄存器。除数锁存访问位（DLAB）位于 U1LCR[7]，它使能对除数锁存的访问。

表 98　UART1 寄存器映射

名称	描述	位功能和地址								访问	复位值[1]	地址
		MSB							LSB			
		BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0			
U1RBR	接收缓存寄存器	8 位读数据								RO	NA	0xE0010000 DLAB=0
U1THR	发送保持寄存器	8 位写数据								WO	NA	0xE0010000 DLAB=0
U1DLL	除数锁存 LSB	8 位数据								R/W	0x01	0xE0010000 DLAB=1
U1DLM	除数锁存 MSB	8 位数据								R/W	0x00	0xE0010004 DLAB=1
U1IER	中断使能寄存器	-	-	-	-	-	-	En.ABTO	En.ABEO	R/W	0x00	0xE0010004 DLAB=0
		En.CTS Int	-	-	-	E. Modem St. Int	En.Rx Lin.St.Int	使能 THRE Int	En. Rx Dat. Av. Int			
U1IIR	中断 ID 寄存器	-	-	-	-	-	-	ABTO.Int	ABEO.Int	RO	0x01	0xE0010008
		FIFO 使能		-	-	IIR3	IIR2	IIR1	IIR0			
U1FCR	FIFO 控制寄存器	Rx 触发		-	-	--	Tx FIFO 复位	Rx FIFO 复位	FIFO 使能	WO	0x00	0xE0010008
U1LCR	线控制寄存器	DLAB	设置间隔	奇偶固定	偶选择	奇偶使能	停止位个数	字长度选择		R/W	0x00	0xE001000C
U1MCR	Modem Ctrl 寄存器	CTSen	RTSen	-	环路回送	-	-	RTS	DTR	R/W	0x00	0xE0010010
U1LSR	线状态寄存器	Rx FIFO 错误	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	0xE0010014
U1MSR	Modem 状态	DCD	RI	DSR	CTS	Delta DCD	后沿 RI	Delta DSR	Delta CTS	RO	0x00	0xE0010018
U1SCR	高速缓存	8 位数据								R/W	0x00	0xE001001C
U1ACR	自动波特率控制寄存器	-	-	-	-	-	-	ABTO IntClr	ABEO IntClr	R/W	0x00	0xE0010020
		-	-	-	-	-	Aut.Rstrt.	模式	启动			
U1FDR	小数分频	保留[31:8]								R/W	0x10	0xE0010028
		MulVal				DivAddVal						
U1TER	发送使能	TxEn	-	-	-	-	-	-	-	R/W	0x80	0xE0010030

[1] 复位值仅指已使用位中保存的数据，不包括保留位的内容。

10.3.1 UART1 接收器缓存寄存器（U1RBR - 0xE0010000，DLAB=0，只读）

U1RBR 是 UART1 Rx FIFO 的最高字节。它包含了最早接收到的字符，可通过总线接口读出。LSB（bit0）代表最早接收到的数据位。如果接收到的字符小于 8 位，未使用的 MSB 填充为 0。

如果要访问 U1RBR，U1LCR 的除数锁存访问位（DLAB）必须为 0。U1RBR 为只读寄存器。

由于 PE、FE 和 BI 位与 RBR FIFO 顶端的字节相对应（即下次读 RBR 时读出的字节），因此，将接收的字节及其状态位成对读出的正确方法是先读 U1LSR，再读 U1RBR。

表 99 UART1 接收器缓存寄存器（U1RBR – 地址 0xE0010000，DLAB=0，只读）位描述

位	符号	描述	复位值
7:0	RBR	UART1 接收器缓存寄存器包含 UART1 Rx FIFO 当中最早接收到的字节	未定义

10.3.2 UART1 发送器保持寄存器（U1THR - 0xE0010000，DLAB=0，只写）

U1THR 是 UART1 Tx FIFO 的最高字节。该最高字节是 Tx FIFO 中最新的字符，可通过总线接口写入。LSB 代表最先发送的位。

如果要访问 U1THR，U1LCR 的除数锁存访问位（DLAB）必须为 0。U1THR 为只写寄存器。

表 100 UART1 发送器保持寄存器（U1THR – 地址 0xE0010000，DLAB=0，只写）位描述

位	符号	描述	复位值
7:0	THR	写 UART1 发送器保持寄存器使数据保存到 UART1 发送 FIFO 当中。当字节到达 FIFO 的最底部并且发送器就绪时，该字节将被发送。	NA

10.3.3 UART1 除数锁存寄存器 0 和 1（U1DLL - 0xE001 0000 和 U1DLM - 0xE001 0004，当 DLAB=1 时）

UART1 除数锁存是 UART1 波特率发生器的一部分，它保存了用于产生波特率时钟的 APB 时钟（PCLK）分频值，波特率时钟必须是波特率的 16 倍（等式 4）。U1DLL 和 U1DLM 寄存器一起构成一个 16 位除数，U1DLL 包含除数的低 8 位，U1DLM 包含除数的高 8 位。值 0x0000 被看作是 0x0001，因为除数是不允许为 0 的。当访问 UART1 除数锁存寄存器时，U1LCR 中的除数锁存访问位（DLAB）必须为 1。

如何选择 U1DLL 和 U1DLM 正确值的细节在本章后面的部分描述。

表 101 UART1 除数锁存 LSB 寄存器（U1DLL – 地址 0xE001 0000，DLAB=1）位描述

位	符号	描述	复位值
7:0	DLLSB	UART1 除数锁存 LSB 寄存器与 U1DLM 寄存器一起决定 UART1 的波特率。	0x01

表 102　UART1 除数锁存 MSB 寄存器（U1DLM – 地址 0xE001 0004，DLAB=1）位描述

位	符号	描述	复位值
7:0	DLMSB	UART1 除数锁存 MSB 寄存器与 U1DLL 寄存器一起决定 UART1 的波特率。	0x00

10.3.4 UART1 小数分频寄存器（U1FDR – 0xE001 0028）

UART1 小数分频寄存器(U1FDR)控制产生波特率的时钟预分频器并可在用户的判断下被读写。该预分频器在每次指定的小数要求中使用 APB 时钟和产生一个输出时钟。

表 103　UART1 小数分频寄存器(U1FDR – 地址 0xE001 0028)位描述

位	功能	描述	复位值
3:0	DIVADDVAL	波特率生成预分频除数值。如果该字段为 0，小数波特率发生器将不会影响 UART1 波特率。	0
7:4	MULVAL	波特率预分频乘数值。不管小数波特率发生器是否使用，该字段必须大于或等于 1 以使 UART1 正常操作。	1
31:8	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

该寄存器控制波特率生成的时钟预分频器。该寄存器的复位值保持 UART1 禁能时的小数功能，确保 UART1 的软件和硬件与没有该特性的 UART 完全兼容。

可用下面的等式计算 UART1 波特率：

$$UART1_{baudrate}=\frac{PCLK}{16x(16xU1DLM+U1DLL)X(1+\frac{DivAddVal}{MulVal})} \qquad (4)$$

其中 PCLK 为外围时钟，U1DLM 和 U1DLL 为标准的 UART1 波特率除数寄存器，DIVADDVAL 和 MULVAL 为 UART1 小数波特率发生器特定的参数。

MULVAL 和 DIVADDVAL 的值应遵循以下的条件：

1. 0＜ MULVAL ≤15
2. 0≤ DIVADDVAL ≤15

如果 U1FDR 寄存器值不遵循这两个请求，那么小数分频输出未定义。如果 DIVADDVAL 为 0，那么小数分频禁止且时钟将不会被分频。

当正在发送/接收数据或数据可能丢失或破坏时，不应修改 U1FDR 的值。

使用注释：在实际使用中，UART1 波特率公式可用下面的形式表示，这种形式识别没有小数波特率发生器时生成的 UART 波特率部分，以及该模块添加的校准系数：

$$UART1_{baudrate}=\frac{PCLK}{16x(16xU1DLM+U1DLL)}X\frac{MulVal}{MulVal+DivAddVal} \qquad (5)$$

根据这种表示，小数波特率发生器也可描述为具有 MULVAL / (MULVAL+DIVADDVAL) 系数的预分频。

10.3.5 UART1 波特率计算

例 1：使用上面的 $UART1_{baudrate}$ 公式，假设系统的 PCLK＝20MHz，如果 U1DL=130（U1DLM=0x00 和 U1DLL=0x82）、DIVADDVAL＝0 和 MULVAL＝1，则可得出 UART1 的

$UART1_{baudrate}$=9615 bauds。

例 2：使用上面的 $UART1_{baudrate}$ 公式，假设系统的 PCLK＝20MHz，如果 U1DL=93（U1DLM=0x00 和 U1DLL=0x5D）、DIVADDVAL＝2 和 MULVAL＝5，则可得出 UART1 的 $UART1_{baudrate}$=9600 bauds。

表 104　外围时钟为 20MHz 时的波特率(PCLK=20MHz)

期望的波特率	MULVAL=0　DIVADDVAL=0			可选的 MULVAL & DIVADDVAL		
	U1DLM:U1DLL hex[2]	dec[1]	%error[3]	U1DLM:U1DLL dec[1]	小数的预分频值 $\frac{MULDIV}{MULDIV+DIVADDVAL}$	%error[3]
50	61A8	25000	0.0000	25000	1/(1+0)	0.0000
75	411B	16667	0.0020	12500	3/(3+1)	0.0000
110	2C64	11364	0.0032	6250	11/(11+9)	0.0000
134.5	244E	9294	0.0034	3983	3/(3+4)	0.0001
150	208D	8333	0.0040	6250	3/(3+1)	0.0000
300	1047	4167	0.0080	3125	3/(3+1)	0.0000
600	0823	2083	0.0160	1250	3/(3+2)	0.0000
1200	0412	1042	0.0320	625	3/(3+2)	0.0000
1800	02B6	694	0.0640	625	9/(9+1)	0.0000
2000	0271	625	0.0000	625	1/(1+0)	0.0000
2400	0209	521	0.0320	250	12/(12+13)	0.0000
3600	015B	347	0.0640	248	5/(5+2)	0.0064
4800	0104	260	0.1600	125	12/(12+13)	0.0000
7200	00AE	174	0.2240	124	5/(5+2)	0.0064
9600	0082	130	0.1600	93	5/(5+2)	0.0064
19200	0041	65	0.1600	31	10/(10+11)	0.0064
38400	0021	33	1.3760	12	7/(7+12)	0.0594
56000	0021	22	1.4400	13	7/(7+5)	0.0160
57600	0016	22	1.3760	19	7/(7+1)	0.0594
112000	000B	11	1.4400	6	7/(7+6)	0.1600
115200	000B	11	1.3760	4	7/(7+12)	0.0594
224000	0006	6	7.5200	3	7/(7+6)	0.1600
448000	0003	3	7.5200	2	5/(5+2)	0.3520

[1] 这一行的值是 16 位（DLM:DLL）的等效十进制值。

[2] 这一行的值是 16 位（DLM:DLL）的等效十六进制值。

[3] 期望波特率和实际波特率的百分比误差。

10.3.6 UART1 中断使能寄存器（U1IER - 0xE001 0004，DLAB=0）

U1IER 用于使能 UART1 中断源。

表 105　UART1 中断使能寄存器（U1IER – 地址 0xE001 0004，DLAB=0）位描述

位	符号	值	描述	复位值
0	RBR 中断使能	0 1	U1IER[0]使能 UART1 接收数据可用中断。它还控制字符接收超时中断。 禁止 RDA 中断 使能 RDA 中断	0
1	THRE 中断使能	0 1	U1IER[1]使能 UART1 THRE 中断。该中断的状态可从 U1LSR[5]读出。 禁止 THRE 中断 使能 THRE 中断	0
2	Rx 线中断使能	0 1	U1IER[2]使能 UART1 Rx 线状态中断。该中断的状态可从 U1LSR[4:1]读出。 禁止 Rx 线状态中断 使能 Rx 线状态中断	0
3	Modem 状态中断使能[1]	0 1	U1IER[3]使能 modem 中断。该中断的状态可从 U1MSR[3:0]中读出。 禁止 modem 中断。 使能 modem 中断。	0
6:4	–	–	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA
7	CTS 中断使能	0 1	如果自动 CTS 模式被使能，那么该位使能/禁能 CTS1 信号跳变上产生的 modem 状态中断。如果自动 CTS 模式被禁止，若 modem 状态中断使能（U1IER[3]）置位，那么 CTS1 跳变将产生一个中断。在正常操作下，CTS1 信号跳变将产生 modem 状态中断除非中断已被禁止（通过清零 U1IER 寄存器中的位 U1IER[3]）。在自动 CTS 模式中，只要 U1IER[3]和 U1IER[7]位置位，则 CTS1 位上的跳变将触发一个中断。 禁止 CTS 中断。 使能 CTS 中断。	0
8	ABTOIntEn	0 1	U1IER8 使能自动波特率超时中断。 禁止自动波特率超时中断。 使能自动波特率超时中断。	0
9	ABEOIntEn	0 1	U1IER9 使能自动波特率结束中断。 禁止自动波特率结束中断。 使能自动波特率结束中断。	0
31:10	-	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

10.3.7 UART1 中断标识寄存器（U1IIR - 0xE001 0008，只读）

U1IIR 提供状态代码用于指示一个挂起中断的中断源和优先级。在访问 U1IIR 过程中，中断被冻结。如果在访问 U1IIR 时产生了中断，该中断被记录，下次 U1IIR 访问可读出。

表 106　UART1 中断标识寄存器（U1IIR – 地址 0xE001 0008，只读）位描述

位	符号	值	描述	复位值
0	中断挂起	0 1	U1IIR[0]为低有效。挂起的中断可通过 U1IIR[3:1]确定。 至少有 1 个中断被挂起 没有挂起的中断	1
3:1	中断标识	011 010 110 001 000	U1IER[3:1]指示对应于 UART1 Rx FIFO 的中断。上面未列出的 U1IER[3:1]的其它组合都为保留值（100, 101, 111） 1. 接收线状态（RLS） 2a. 接收数据可用（RDA） 2b. 字符超时指示（CTI） 3. THRE 中断 4. modem 中断	0
5:4	-	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA
7:6	FIFO 使能		这些位等效于 U1FCR[0]	0
8	ABEOInt		自动波特率结束中断。如果成功完成波特率且中断被使能则 ABEOInt 为 1。	0
9	ABTOInt		自动波特率超时中断。如果波特率超时且中断被使能，则 ABTOInt 为 1。	0
31:10	-	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

中断的处理见表 83。给定了 U1IIR[3:0]的状态，中断处理程序就能确定中断源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 UART1 Rx 输入产生 4 个错误条件（溢出错误（OE）、奇偶错误（PE）、帧错误（FE）和间隔中断（BI））中的任意一个，该中断标志将置位。产生该中断的 UART1 Rx 错误条件可通过查看 U1LSR[4:1]得到。当读取 U1LSR 时清除中断。

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 到达 U1FCR[7:6]所定义的触发点时，RDA 被激活。当 UART1 Rx FIFO 的深度低于触发点时，RDA 复位。当 RDA 中断激活时，CPU 可读出由触发点所定义的数据块。

CTI 中断（U1IIR[3:1]=110）为第二优先级中断。当 UART1 Rx FIFO 包含至少 1 个字符并且在接收 3.5 到 4.5 字符的时间内没有发生 UART1 Rx FIFO 动作时，该中断置位。UART1 Rx FIFO 的任何动作（读或写 UART1 RSR）都将清除该中断。在接收到的信息不是触发值的倍数后，CTI 中断将会清空 UART1 RBR。例如，如果一个外设想要发送一个 105 个字符的信息，而触发值为 10 个字符，那么 CPU 接收 10 个 RDA 中断将导致前 100 个字符的传输，而 CPU 接收 1 到 5 个 CTI 中断（取决于服务程序）将导致剩下 5 个字符的传输。

表 107　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 FIFO 模式下（U1FCR0=1）到达触发点	U1RBR 读[3]或 UART1 FIFO 低于触发值
1100	第二	字符超时指示	Rx FIFO 包含至少 1 个字符并且在一段时间内无字符输入或移出，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）设置的触发值。 实际的时间为： [(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[4]（如果是中断源）或 THR 写操作
0000	第四	Modem 状态	CTS 或 DSR 或 RI 或 DCD	MSR 读

[1] “0000”（见表注释 2）, “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”, “1101”, “1110”, “1111”为保留值。

[2] 详见 10.3.11 节“UART1 线状态寄存器(U1LSR – 0xE001 0014，只读)”

[3] 详见 10.3.1 节“UART1 接收器缓存寄存器(U1RBR – 0xE001 0000, DLAB=0, 只读)”

[4] 详见 10.3.7 节“UART1 中断识别寄存器(U1IIR - 0xE001 0008, 只读)”和 10.3.2 节“UART1 发送保持寄存器(U1THR – 0xE001 0000, DLAB=0、只写)”。

UART1 THRE 中断(U1IIR[3:1]=001)为第三优先级中断。当 UART1 THR FIFO 为空并且满足特定的初始化条件时，该中断激活。这些初始化条件将使 UART1 THR FIFO 被数据填充，以免在系统启动时产生许多 THRE 中断。初始化条件在 THRE=1 时实现了一个字符的延时减去停止位并在上一次 THRE=1 事件之后在 U1THR 中没有存在至少 2 个字符。在没有译码和服务 THRE 中断时，该延迟为 CPU 提供了将数据写入 U1THR 的时间。如果 UART1 THR FIFO 中曾经有两个或更多字符，而当前 U1THR 为空时，THRE 中断立即设置。当发生 U1THR 写操作或 U1IIR 读操作并且 THRE 为最高优先级中断（U1IIR[3:1]=001）时，THRE 中断复位。

Modem 中断（U1IIR[3:1]=000）仅可用于 LPC2104/05/06。它是最低优先级中断，只要在 modem 输入管脚 DCD, DSR 或 CTS 上有任何状态变化，modem 中断就被激活。此外，在 modem 输入 RI 上从低到高的跳变将产生 modem 中断。Modem 中断源可通过检测 U1MSR[3:0]来确定。U1MSR 读操作将清除 modem 中断。

10.3.8 UART1 FIFO 控制寄存器（U1FCR - 0xE001 0008）

U1FCR 控制 UART1 Rx 和 Tx FIFO 的操作。

表 108　UART1 FIFO 控制寄存器（U1FCR – 地址 0xE001 0008）位描述

位	符号	值	描述	复位值
0	FIFO 使能	0 1	UART1 FIFO 被禁止。在应用中必须不能使用。 高电平使能对 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。	0
1	Rx FIFO 复位	0 1	对任意的 UART1 FIFO 无影响。 写 1 到 U1FCR[1]将清零 UART1 Rx FIFO 中的所有字节并复位指针逻辑。该位自动清零。	0
2	Tx FIFO 复位	0 1	对任意的 UART1 FIFO 无影响。 写 1 到 U1FCR[2]将清零 UART1 Tx FIFO 中的所有字节并复位指针逻辑。该位自动清零。	0
5:3	—	0	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA
7:6	Rx 触发选择	 00 01 10 11	这两个位决定在激活中断之前，接收器 UART1 FIFO 必须写入多少个字符。 触发点 0（1 个字符或 0x01） 触发点 1（4 个字符或 0x04） 触发点 2（8 个字符或 0x08） 触发点 3（14 个字符或 0x0E）	0

10.3.9 UART1 线控制寄存器（U1LCR - 0xE001 000C）

U1LCR 决定发送和接收数据字符的格式。

表 109　UART1 线控制寄存器（U1LCR – 地址 0xE001 000C）位描述

位	符号	值	描述	复位值
1:0	字长度选择	00 01 10 11	5 位字符长度 6 位字符长度 7 位字符长度 8 位字符长度	0
2	停止位选择	0 1	1 个停止位 2 个停止位（如果 U1LCR[1:0]=00 则为 1.5）	0
3	奇偶使能	0 1	禁止奇偶产生和校验 使能奇偶产生和校验	0
5:4	奇偶选择	00 01 10 11	奇数。发送字符中 1 的个数和附带的校验位为奇数。 偶数。发送字符中 1 的个数和附带的校验位为偶数。 强制为“1”，奇偶固定 强制为“0”，奇偶固定	0
6	间隔控制	0 1	禁止间隔发送。 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。	0
7	除数锁存访问位(DLAB)	0 1	禁止访问除数锁存 使能访问除数锁存	0

10.3.10 UART1 Modem 控制寄存器（U1MCR - 0xE001 0010）

U1MCR 使能 modem 的回写模式并控制 modem 的输出信号。

表 110　UART1 Modem 控制寄存器（U1MCR – 地址 0xE001 0010）位描述

位	符号	值	描述	复位值
0	DTR 控制		选择 modem 输出管脚 DTR。该位在回写模式激活时读出为 0。	0
1	RTS 控制		选择 modem 输出管脚 RTS。该位在回写模式激活时读出为 0。	0
3:2	-		保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
4	回写模式选择		modem 回写模式提供了一个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。输入脚 RxD1 对回写模式无影响，输出脚 TxD1 保持总为 1 的状态。4 个 modem 输入（CTS, DSR, RI 和 DCD）与外部断开。从外部来看，modem 的输出端（RTS, DTR）无效。在内部，4 个 modem 输出连接到 4 个 modem 输入。这样连接的结果是 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 位就可产生 modem 状态中断。	0
		0	禁止 modem 回写模式	
		1	使能 modem 回写模式	
5:3	-		保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
6	RTSen		自动 RTS 控制位。	0
		0	禁止自动 RTS 流控制。	
		1	使能自动 RTS 流控制。	
7	CTSen		自动 CTS 控制位。	0
		0	禁止自动 CTS 流控制。	
		1	使能自动 CTS 流控制。	

自动流控制

如果自动 RTS 模式被使能，那么 UART1 的接收器 FIFO 硬件控制 UART1 的 RTS1 输出。如果自动 CTS 模式被使能，若 CTS1 输入信号有效，那么 UART1 的 U1TSR 硬件将只启动发送。

自动 RTS

自动 RTS 功能通过设置 CTSen 位使能。自动 RTS 数据流控制在 U1RBR 模块中产生并链接到已编程的接收 FIFO 触发电平。如果自动 RTS 被使能，当接收 FIFO 电平到达已编程的触发电平时，RTS1 无效（到一个高值）。发送的 UART 可在到达触发电平后发送一个额外的字节（假设发送的 UART 有另一个字节要发送），因为它可能不知道 RTS1 的无效直至它开始发送额外的字节后。一旦接收 FIFO 到达原来的触发电平，RTS1 就自动重新有效（到一个低值）。RTS1 的重新有效指示发送的 UART 继续发送数据。

如果自动 RTS 模式被禁止，那么 RTSen 位控制 UART1 的 RTS1 输出。如果自动 RTS 模式被使能，那么硬件控制 RTS1 输出且 RTS1 的实际值将在 UART1 的 RTSen 位中复制。只要自动 RTS 被使能，则 RTSen 位的值对于软件只可读。

例子：假设 type550 中操作的 UART1 在 U1FCR 中的触发电平设为 0x2，那么若自动 RTS 被使能，一旦接收 FIFO 包含 8 字节（表 108），UART1 就使 RTS1 输出无效。一旦接收 FIFO 到达原来的触发电平 4 字节，那么 RTS1 输出就将重新有效。

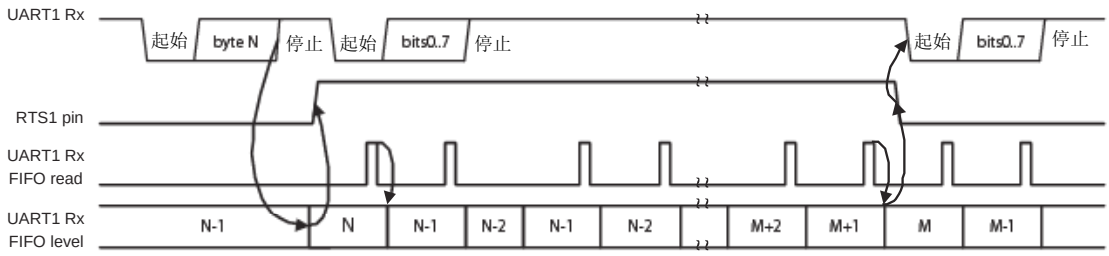


图 19　自动 RTS 功能时序

自动 CTS

自动 CTS 功能通过设置 CTSen 位使能。如果自动 CTS 被使能，U1TSR 模块中的发送器电路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，发送器发送下一个字节。为了停止发送器发送后面的字节，CTS1 必须在当前发送的最后一个停止位的中间之前被释放。在自动 CTS 模式中，CTS1 信号的变化不触发 modem 状态中断除非 CTS 中断使能位置位，但 U1MSR 中的 Delta CTS 位将被置位。表 111 列出产生 modem 状态中断的条件。

表 111　modem 状态中断产生

使能 modem 状态中断 (U1IER[3])	CTSen (U1MCR[7])	CTS 中断使能 (U1IER[7])	DeltaCTS (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

自动 CTS 功能减少到主机系统的中断。当流控制使能时，CTS1 状态变化不触发主机中断，因为器件自动控制其自身的发送器。无需自动 CTS，发送器发送任何出现在发送 FIFO 的数据并导致接收器溢出错误。图 20 描述自动 CTS 功能的时序。

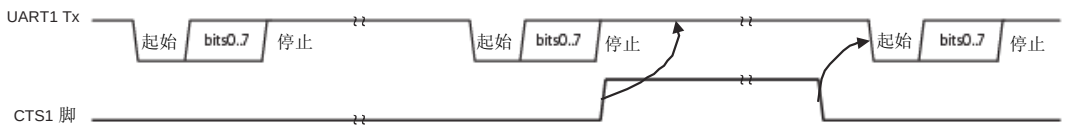


图 20　自动 CTS 功能的时序

当启动首个字符的发送时，CTS1 信号有效。一旦待处理的传输结束，传输就停止。只要 CTS1 无效（高电平），UART 就继续发送 1 位。一旦 CTS1 无效，传输恢复并发送起始位，后面跟着下一个字符的数据位。

10.3.11 UART1 线状态寄存器（U1LSR – 0xE001 0014，只读）

U1LSR 为只读寄存器，它提供 UART1 Tx 和 Rx 模块的状态信息。

表 112　UART1 线状态寄存器（U1LSR – 地址 0xE001 0014，只读）位描述

位	符号	值	描述	复位值
0	接收器数据就绪（RDR）	0 1	当 U1RBR 包含未读取的字符时，U1LSR[0]置位；当 UART1 RBR FIFO 为空时，U1LSR[0]清零。 U1RBR 为空 U1RBR 包含有效数据	0
1	溢出错误（OE）	0 1	溢出错误条件在错误发生后立即设置。U1LSR 读操作清零 U1LSR[1]。当 UART1 RSR 已经有新的字符组合而 UART1 RBR FIFO 已满时，U1LSR[1]置位。此时 UART1 RBR FIFO 不会被覆盖，UART1 RSR 中的字符将丢失。 溢出错误状态未激活 溢出错误状态激活	0
2	奇偶错误（PE）	0 1	当接收字符的校验位为错误状态时产生一个奇偶错误。U1LSR 读操作清零 U1LSR[2]位。奇偶错误检测时间取决于 U1FCR[0]。 **注：**奇偶错误与 UART1 RBR FIFO 中顶部的字符相关。 奇偶错误状态未激活 奇偶错误状态激活	0
3	帧错误（FE）	0 1	当接收字符的停止位为 0 时，产生帧错误。U1LSR 读操作清零 U1LSR[3]。帧错误检测时间取决于 U1FCR0。当检测到一个帧错误时，Rx 将尝试与数据重新同步并假设错误的停止位实际是一个超前的起始位。但即使没有出现帧错误，它也不能假设下一个接收到的字节是正确的。 **注：**帧错误与 UART1 RBR FIFO 中顶部的字符相关。 帧错误状态未激活 帧错误状态激活	0
4	间隔中断（BI）	0 1	在发送整个字符（起始位、数据、校验位和停止位）过程中 RxD1 如果都保持逻辑 0，则产生间隔中断。当检测到间隔条件时，接收器立即进入空闲状态直到 RxD1 变为全 1 状态。U1LSR 读操作清零该状态位。间隔检测的时间取决于 U1FCR[0]。 **注：**间隔中断与 UART1 RBR FIFO 中顶部的字符相关。 间隔中断状态未激活 间隔中断状态激活	0
5	发送器保持寄存器空（THRE）	0 1	当检测到 UART1 THR 空时，THRE 置位，U1THR 写操作清零该位。 U1THR 包含有效数据 U1THR 为空	1
6	发送器空（TEMT）	0 1	当 U1THR 和 U1TSR 都为空时，TEMT 置位。当 U1TSR 或 U1THR 包含有效数据时，TEMT 清零。 U1THR 和/或 U1TSR 包含有效数据 U1THR 和 U1TSR 为空	1

续上表

位	符号	值	描述	复位值
7	Rx FIFO 错误（RXFE）		当一个带有 Rx 错误（例如帧错误、奇偶错误或间隔中断）的字符装入 U1RBR 时，U1LSR[7]置位。当读取 U1LSR 寄存器并且 UART1 FIFO 中不再有错误时，U1LSR[7]清零。	0
		0	U1RBR 中没有 UART1 Rx 错误，或 U1FCR[0]=0	
		1	UART1 RBR 包含至少一个 UART1 Rx 错误	

10.3.12 UART1 Modem 状态寄存器（U1MSR - 0x0E001 0018）

U1MSR 是一个只读寄存器，它提供 modem 输入信号的状态信息。U1MSR[3:0]在读取 U1MSR 时清零。需要注意的是，modem 信号对 UART1 的操作没有直接影响，modem 信号的操作是通过软件来实现的。

表 113　UART1 Modem 状态寄存器（U1MSR – 地址 0xE001 0018）位描述

位	符号	值	描述	复位值
0	Delta CTS		当输入 CTS 状态发生变化时，该位置位。读取 U1MSR 时清零。	0
		0	没有检测到 modem 输入 CTS 上的状态变化	
		1	检测到 modem 输入 CTS 上的状态变化	
1	Delta DSR		当输入 DSR 状态发生变化时，该位置位。读取 U1MSR 时清零。	0
		0	没有检测到 modem 输入 DSR 上的状态变化	
		1	检测到 modem 输入 DSR 上的状态变化	
2	后沿 RI		当输入 RI 发生低到高的跳变时，该位置位。读取 U1MSR 时清零。	0
		0	没有检测到 modem 输入 RI 上的状态变化	
		1	检测到 modem 输入 RI 上低到高的跳变	
3	Delta DCD		当输入 DCD 状态发生变化时，该位置位。读取 U1MSR 时清零。	0
		0	没有检测到 modem 输入 DCD 上的状态变化	
		1	检测到 modem 输入 DCD 上的状态变化	
4	CTS		清零发送状态。输入信号 CTS 的补码。在回写模式下，该位连接到 U1MCR[1]。	0
5	DSR		数据设备就绪状态。输入信号 DSR 的补码。在回写模式下，该位连接到 U1MCR[0]。	0
6	RI		响铃指示状态。输入信号 RI 的补码。在回写模式下，该位连接到 U1MCR[2]。	0
7	DCD		数据载波检测状态。输入信号 DCD 的补码。在回写模式下，该位连接到 U1MCR[3]。	0

10.3.13 UART1 高速缓存寄存器（U1SCR - 0xE001 001C）

在 UART1 操作时 U1SCR 无效。用户可自由对该寄存器进行读或写。不提供中断接口向主机指示 U1SCR 所发生的读或写操作。

表 114　UART1 高速缓存寄存器（U1SCR – 地址 0xE001 001C）位描述

位	符号	描述	复位值
7:0	Pad	一个可读可写的字节	0x00

10.3.14 UART1 自动波特率控制寄存器(U1ACR – 0xE001 0020)

UART1 自动波特率控制寄存器(U1ACR)控制测量波特率生成的输入时钟/数据率的过程，用户可自由对该寄存器进行读或写。

表 115　自动波特率控制寄存器（U1ACR – 0xE001 0020）位描述

位	符号	值	描述	复位值
0	Start		自动波特率结束后该位自动清零。	0
		0	自动波特率停止（自动波特率不运行）。	
		1	自动波特率启动（自动波特率正在运行）。自动波特率运行位。该位在自动波特率结束后自动清零。	
1	Mode		自动波特率模式选择位。	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（计数器在下一个 UART1 Rx 下降沿重新启动）	
7:3	—	NA	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	0
8	ABEOIntClr		自动波特率中断结束清零位（仅可写访问）。写 1 将在 U1IIR 中清除相应的中断。写 0 无影响。	0
9	ABTOIntClr		自动波特率超时中断清零位（仅可写访问）。写 1 将在 U1IIR 中清除相应的中断。写 0 无影响。	0
31:10	—	NA	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	0

10.3.15 自动波特率

UART1 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。如果使能，那么自动波特率特性将测量接收数据流的位时并因此设置除数锁存寄存器 U1DLM 和 U1DLL。

自动波特率通过设置 U1ACR 起始位来启动。自动波特率可通过清零 U1ACR 起始位来停止。一旦自动波特率结束起始位将清零，并且读该位将返回自动波特率的状态（等待/完成）。

可通过选择 U1ACR 模式位来使用两种自动波特率测量模式。在模式 0 中，波特率在 UART1 Rx 管脚的两个连续的下降沿上测量（起始位的下降沿和最低位的下降沿）。在模式 1 中，波特率在 UART1 Rx 管脚的下降沿和后续的上升沿之间测量（起始位的长度）。

如果超时出现（速率测量计数器溢出），那么 U1ACR AutoRestart 位可用于自动重新启动速率测量。如果该位置位，速率测量将在 UART1 Rx 管脚的下一个下降沿重新启动。

自动波特率功能可产生两个中断。

- 如果中断使能，那么将设置 U1IIR ABTOInt 中断（U1IER ABToIntEn 置位且自动波特率测量计数器溢出）。
- 如果中断使能，那么将设置 U1IIR ABEOInt 中断（U1IER ABEoIntEn 置位且自动波特率成功完成）。

通过置位相应的 U1ACR ABTOIntClr 和 ABEOIntClr 位来清零自动波特率中断。

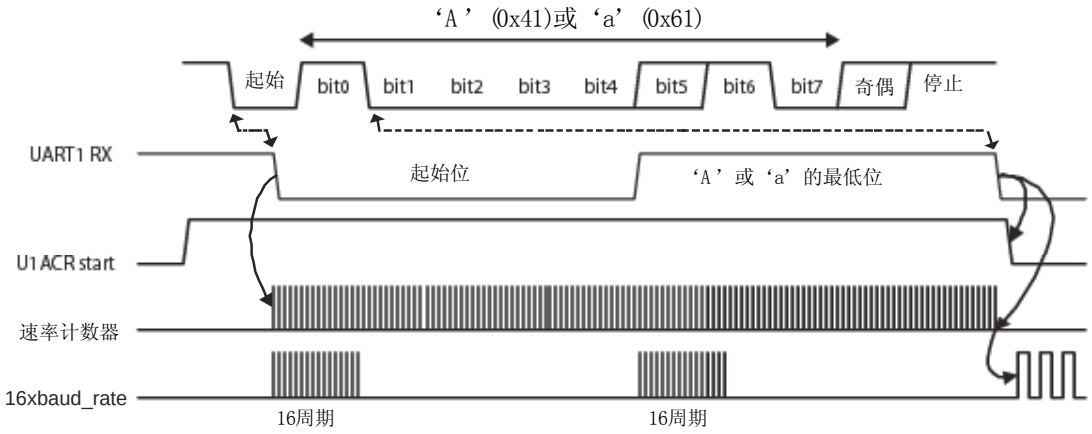
在自动波特率期间，小数波特率发生器被禁能(DIVADDVAL=0)。然而，如果小数波特率发生器被使能(DIVADDVAL＞0)，那么它将影响 UART1 Rx 管脚波特率的测量，但 U1FDR 寄存器的值在速率测量后不会被修改。同时，当使用波特率时，任何写 U1DLM 和 U1DLL 寄存器的操作应在写 U1ACR 寄存器前完成。UART1 支持的波特率最小和最大值受 PCLK, 数据位的个数、停止位和校验位影响。

$$\text{ratemin}=\frac{2\times \text{PCLK}}{16\times 2^{15}}\leqslant \text{UART1}_{\text{baudrate}}\leqslant \frac{\text{PCLK}}{16\times(2+\text{数据位}+\text{奇偶位}+\text{停止位})}=\text{ratemax} \tag{6}$$

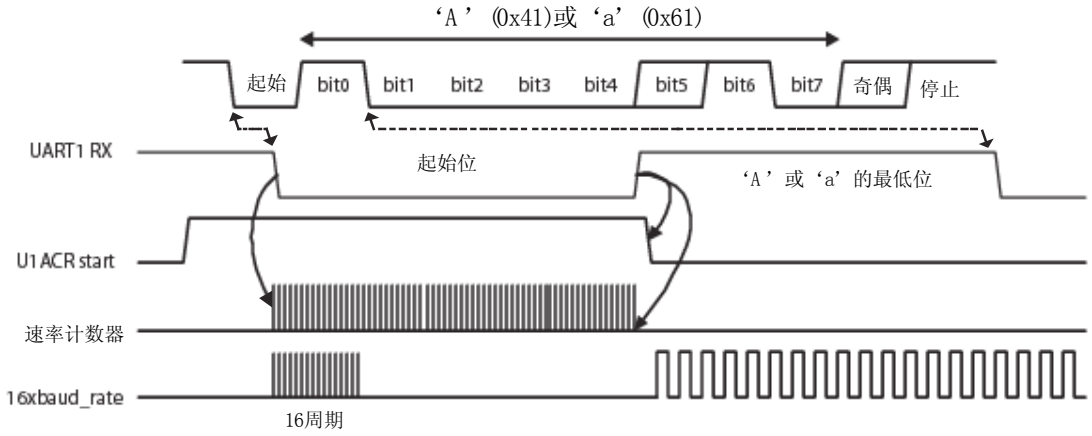
10.3.16 自动波特率模式

当软件正在期望“AT”命令时，它用期望的字符格式配置 UART1 并设置 U1ACR 起始位。可以不用关心除数锁存器 U1DLL 和 U1DLM 的初始值。由于“A”或“a”ASCII 编码（“A”=0x41, “a”=0x61），因此 UART1 Rx 管脚检测起始位且所期望字符的 LSB 由两个下降沿限定。当 U1ACR 起始位置位时，自动波特率协议将执行以下过程：

1. 在 U1ACR 起始位置位时，速率测量计数器复位且 UART1 U1RSR 复位。U1RSR 波特率切换为最高的速率。
2. UART1 Rx 管脚下降沿触发起始位的开始。速率测量计数器将开始对 PCLK（可选择被小数波特率发生器预分频）进行计数。
3. 在起始位的接收过程中，RSR 的波特率输入端产生 16 个具有 UART1 输入时钟频率（被小数波特率发生器预分频）的脉冲（脉冲频率和 UART1 输入时钟频率一致），保证起始位存储在 U1RSR 中。
4. 在接收起始位（和模式 0 的字符 LSB）的过程中，速率计数器将随着预分频的 UART1 输入时钟（PCLK）的出现继续增加。
5. 如果 Mode=0，那么速率计数器将在 UART1 Rx 管脚的下一个下降沿停止。如果 Mode=1，那么速率计数器将在 UART1 Rx 管脚的下一个上升沿停止。
6. 速率计数器被装入到 U1DLM/U1DLL 且波特率将自动切换为正常操作模式。设置完 U1DLM/U1DLL 后，U1IIR ABEOInt(自动波特率中断的结束)将被置位，如果使能。U1RSR 现将继续接收“A/a”字符剩下的位。



a. 模式0（起始位和最低位用于自动波特率）



b. 模式1（仅起始位用于自动波特率）

图 21　自动波特率 a) 模式 0 和 b) 模式 1 波形

10.3.17 UART1 发送使能寄存器（U1TER – 0xE001 0030）

LPC2101/02/03 的 U1TER 使能实现软件和硬件流控制。当 TxEn＝1 时，只要数据可用，UART1 发送器就将持续发送数据。一旦 TxEn 变为 0，UART1 发送器立刻停止工作。

表 116 描述了如何使用 TXEn 位来实现软件流控制。

表 116　UART1 发送使能寄存器（U1TER – 地址 0xE0010030）位描述

位	符号	描述	复位值
6:0	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
7	TxEn	该位为 1（复位值）时，如果以前的数据都被发送后，写入 THR 的数据输出到 TxD 管脚。如果一个字符正在发送时该位清零，则结束这个字符的发送，后面的字符也不再发送，直到该位重新置位。换言之，该位为 0 将终止 THR 或 Tx FIFO 到发送移位寄存器的字符传输。当检测到硬件握手 Tx-允许信号 CTS 出错或利用软件握手接收到一个 XOFF 字符（DC3）时，软件将该位清零。当检测到正确的 Tx-允许信号或接收到 XON 字符（DC1）时，软件又能将该位重新置位。	1

10.4 结构

UART1 的结构如下方框图所示。

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 监视串行输入线 RxD1 的有效输入。UART1 Rx 移位寄存器（U1RSR）通过 RxD1 接受有效的字符。当 U1RSR 接收到一个有效字符时，它将该字符传送到 UART1 Rx 缓存寄存器 FIFO 中，等待 CPU 或主机通过主机接口进行访问。

UART1 发送器模块 U1Tx 接受 CPU 或主机写入的数据并将数据缓存到 UART1 Tx 保持寄存器 FIFO (U1THR)中。UART1 Tx 移位寄存器（U1TSR）读取 U1THR 中的数据并将数据通过串行输出管脚 TxD1 发送。

UART1 波特率发生器模块 U1BRG 产生 UART1 Tx 模块所使用的定时。U1BRG 模块时钟源为 APB 时钟（PCLK）。主时钟与 U1DLL 和 U1DLM 寄存器所定义的除数相除得到 Tx 模块使用的时钟。该时钟为 16 倍过采样时钟 NBAUDOUT。

Modem 接口包含寄存器 U1MCR 和 U1MSR。该接口负责一个 modem 外设与 UART1 之间的握手。

中断接口包含寄存器 U1IER 和 U1IIR。中断接口接收几个由 U1Tx 和 U1Rx 模块发出的单时钟宽度的使能信号。

U1Tx 和 U1Rx 的状态信息保存在 U1LSR 中。U1Tx 和 U1Rx 的控制信息保存在 U1LCR 中。

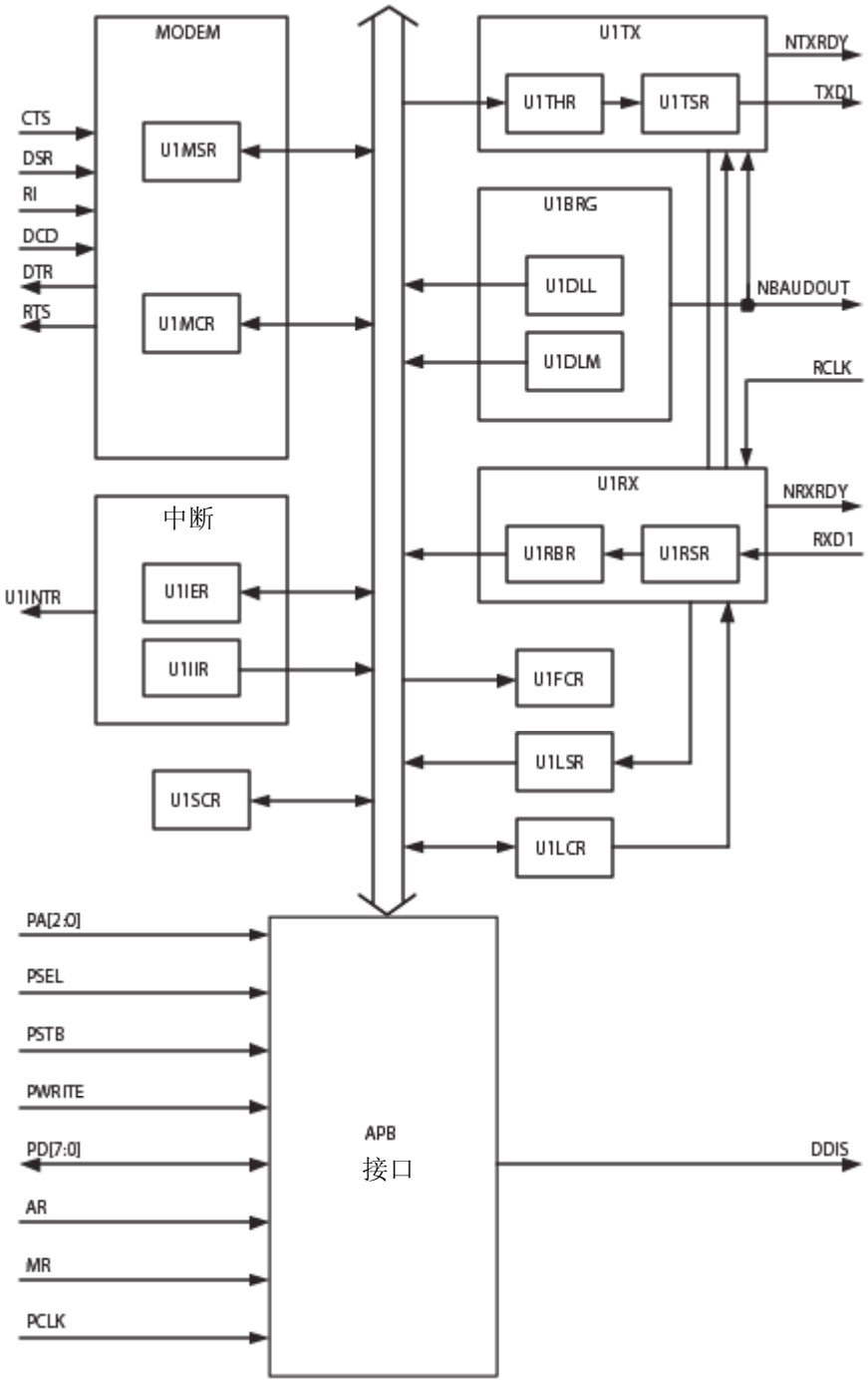


图 22　UART1 方框图

第11章 I²C 接口 I²C0 和 I²C1

11.1 特性

- 标准的 I²C 总线接口，可配置为主机，从机或主/从机
- 同时发送的主机之间进行仲裁，避免了总线数据的冲突
- 可编程时钟可实现 I²C 传输速率控制
- 主机从机之间双向数据传输
- 串行时钟同步使器件在一条串行总线上实现不同位速率的通信
- 串行时钟同步可作为握手机制使串行传输挂起和恢复
- I²C 总线可用于测试和诊断

11.2 应用

与外部标准 I²C 部件接口，例如串行 RAM、LCD、音调发生器等等。

11.3 描述

I²C 总线的典型配置如图 23 所示。根据方向位(R/W)状态的不同，I²C 总线上存在以下两种类型的数据传输：

- 主发送器向从接收器发送数据。主机发送的第一个字节是从机地址。接下来是数据字节流。从机每接收一个字节返回一个应答位。
- 从发送器向主接收器发送数据。第一个字节(从地址)由主机发送。从机返回一个应答位。接下来从机向主机发送数据字节。主机接收完除最后一个字节外的所有字节后返回一个应答位。接收完最后一个字节，主机返回一个“非应答位”。主器件产生所有串行时钟脉冲和起始以及停止条件。出现停止条件或重复的起始条件时传输结束。由于重复的起始条件同时是下一个串行发送的开始，因此 I²C 总线不会被释放。

LPC2101/02/03 提供字节方式的 I²C 接口。它有 4 种操作模式：主发送器模式、主接收器模式、从发送器模式和从接收器模式。

I²C 接口完全符合整个 I²C 规范，支持断开到 LPC2101/02/03 的连接而不影响同一 I²C 总线上的其它器件（见“快速模式”标题下的“I²C 总线规范”描述）。此功能有时很有用，但它实际上限制了 I²C 接口不使用时相同管脚的交替使用。当相同的微控制器中包含多个 I²C 接口时该功能几乎不用。

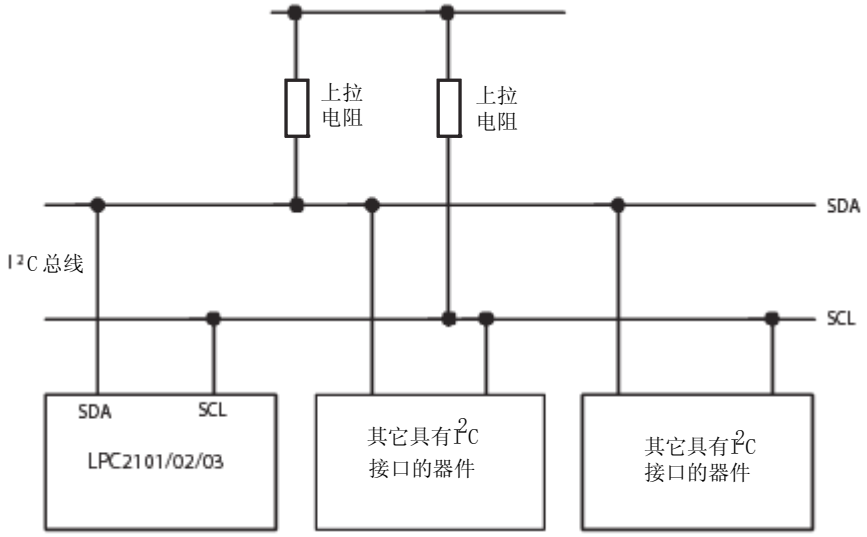


图 23 I²C 总线配置

11.4 管脚描述

表 117 I²C 管脚描述

管脚	类型	描述
SDA0, 1	输入/输出	**I²C 串行数据**
SCL0, 1	输入/输出	**I²C 串行时钟**

11.5 I²C 操作模式

在一个给定的应用中，I²C 模块可用作主机、从机或同时用作主机和从机。在从机模式中，I²C 硬件查找其自身的从地址和通用调用地址。如果检测到其中的一个地址，则产生中断请求。如果处理器想成为总线主机，在进入主机模式前硬件必须等到总线空闲，以便不中止从机操作。如果在主机模式中总线仲裁丢失，则 I²C 模块立刻切换到从机模式并在相同的串行传输中检测其自身的从机地址。

11.5.1 主发送器模式：

在该模式中，数据从主机发送到从机。在进入主发送器模式之前，I2CONSET 必须按照表 118 进行初始化。必须置位 I2EN 来使能 I²C 功能。如果 AA 位为 0，而另一个器件成为总线的主控器时，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 可清零 SI。

表 118 I2C0CONSET 和 I2C1CONSET 用于配置主模式

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器件的从地址（7 位）和数据方向位。在该模式下，数据方向位（R/W）应当为 0 表示执行写操作。因此第一个发送的字节为从地址和写方向位。数据

的发送每次为 8 位。每发送完一个字节，都接收到一个应答位。输出起始和停止条件来指示串行传输的起始和结束。

当软件置位 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。当发送完起始条件后，SI 置位。此时 I2STAT 中的状态代码应当为 0x08。该状态代码用于指向一个状态服务程序。该状态程序将从地址和写方向位装入 I2DAT（数据寄存器），然后清零 SI 位。向 I2CONCLR 寄存器中的 SIC 位写入 1 可清零 SI。

当从地址和 R/W 位已发送且接收到应答位之后，SI 位再次置位，并且对于主模式，可能的状态代码为 0x18，0x20 或 0x38，如果从模式使能（AA=1），可能的状态代码为 0x68，0x78H 或 0xB0。每个状态代码对应的执行动作如表 133 到表 136 所示。

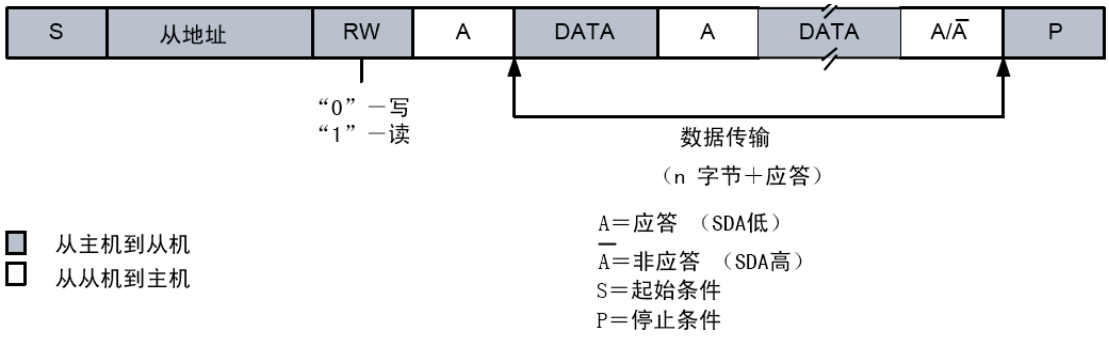


图 24　主发送器模式中的格式

11.5.2 主接收器模式

在主接收器模式中，接收的数据来自从发送器。传输的初始化与主发送器模式相同。当发送完起始条件后，中断服务程序必须将从地址和数据方向位装入 I²C 数据寄存器（I2DAT），然后清零 SI 位。在这种情况下，数据方向位(R/W)应为 1 来指示一个读操作。

当从地址和数据方向位已发送且接收到应答位之后，SI 置位而状态寄存器将显示状态代码。对于主模式，可能的状态代码为 0x40，0x48 或 0x38。对于从模式，可能的状态代码为 0x68，0x78 或 0xB0。详细内容请参考表 134。

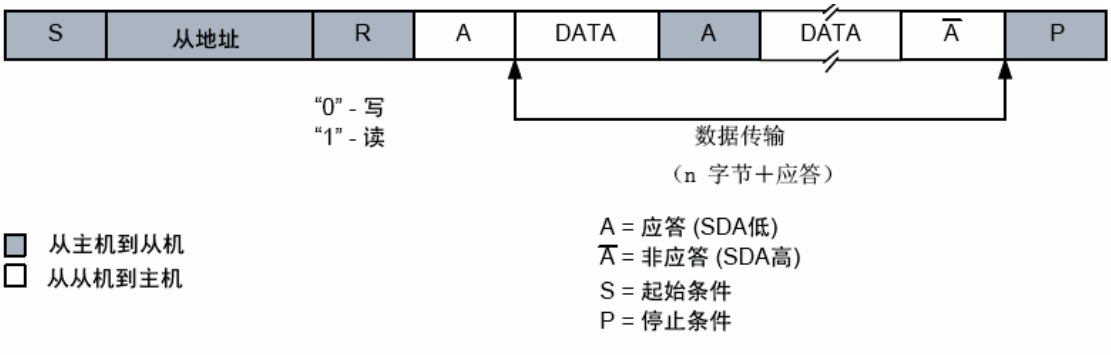


图 25　主接收器模式的格式

在一个重复的起始条件之后，I²C 可以切换到主发送器模式。

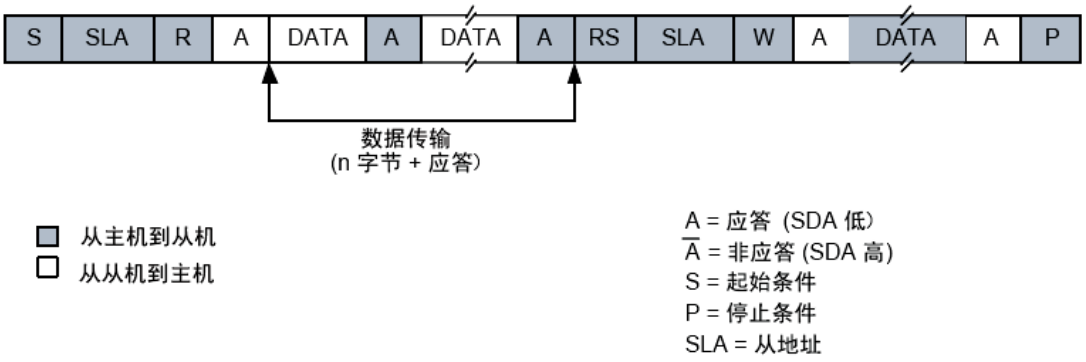


图 26 在发送重复起始条件后，主接收器切换到主发送器

11.5.3 从接收器模式

在从接收器模式中，从主发送器接收数据字节。要初始化从接收器模式，用户应写入从地址寄存器（I2ADR）和 I²C 控制置位寄存器（I2CONSET），如表 119 所示：

表 119 I2C0CONSET 和 I2C1CONSET 用于配置从模式

位	7	6	5	4	3	2	1	0
符号	-	I2EN	STA	STO	SI	AA	-	-
值	-	1	0	0	0	1	-	-

I2EN 必须置位以使能 I²C 功能。AA 位必须置位以使 I²C 应答自身的从地址或通用调用地址。STA，STO 和 SI 设置为 0。

当 I2ADR 和 I2CONSET 完成初始化时，I²C 一直等待到它被自身的从地址或通用地址（两者后面都紧跟数据方向位）寻址为止。如果方向位为 0(W)，I²C 将进入从接收器模式。如果方向位为 1(R)，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）中读出有效的状态代码。状态代码及操作请参考表 135。

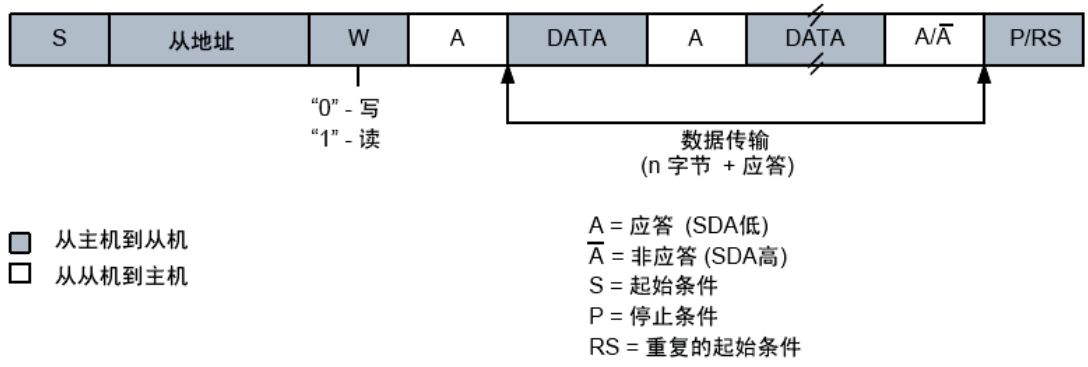


图 27 从接收器模式的格式

11.5.4 从发送器模式

第一个字节的接收和处理与从接收器模式相同。但在该模式中，方向位将为 1 指示一个读操作。串行数据通过 SDA 发送而串行时钟通过 SCL 输入。在串行传输开始和结束时对起始和停止条件进行识别。在一个给定的应用中，I²C 可以为主模式也可以为从模式。在从模式中，I²C 硬件寻找它自身的从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，硬件在进入主模式前一直等待，直到总线释放，这样就不会中断一个可能的从机动作。如果在主模式中总线仲裁丢失，I²C 接口将立即切换到从模式并能在同一个串行传输中检测自身的从地址。

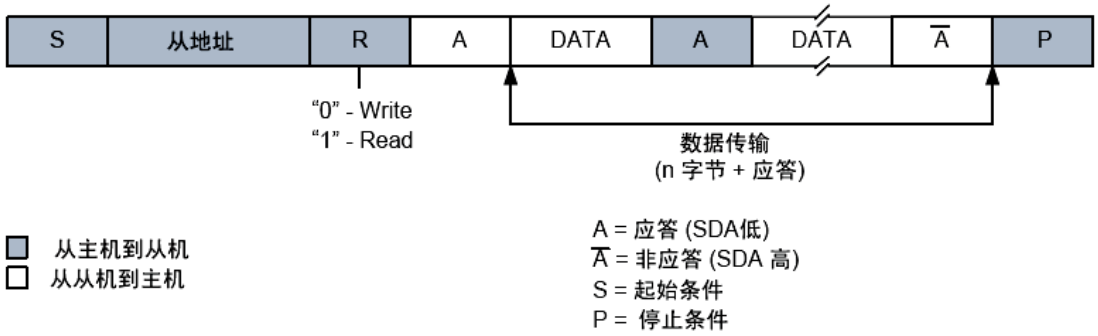


图 28 从发送器模式的格式

11.6 I²C 的实现和操作

图 29 所示为片内 I2C 总线接口的操作过程，下面分别对每个模块进行了描述。

11.6.1 输入滤波器和输出部分

输入信号与内部时钟同步，短于 3 个时钟的脉冲干扰可以滤除。

I²C 输出是一个特殊的端口，它是为了遵循 I²C 规范而设计的。

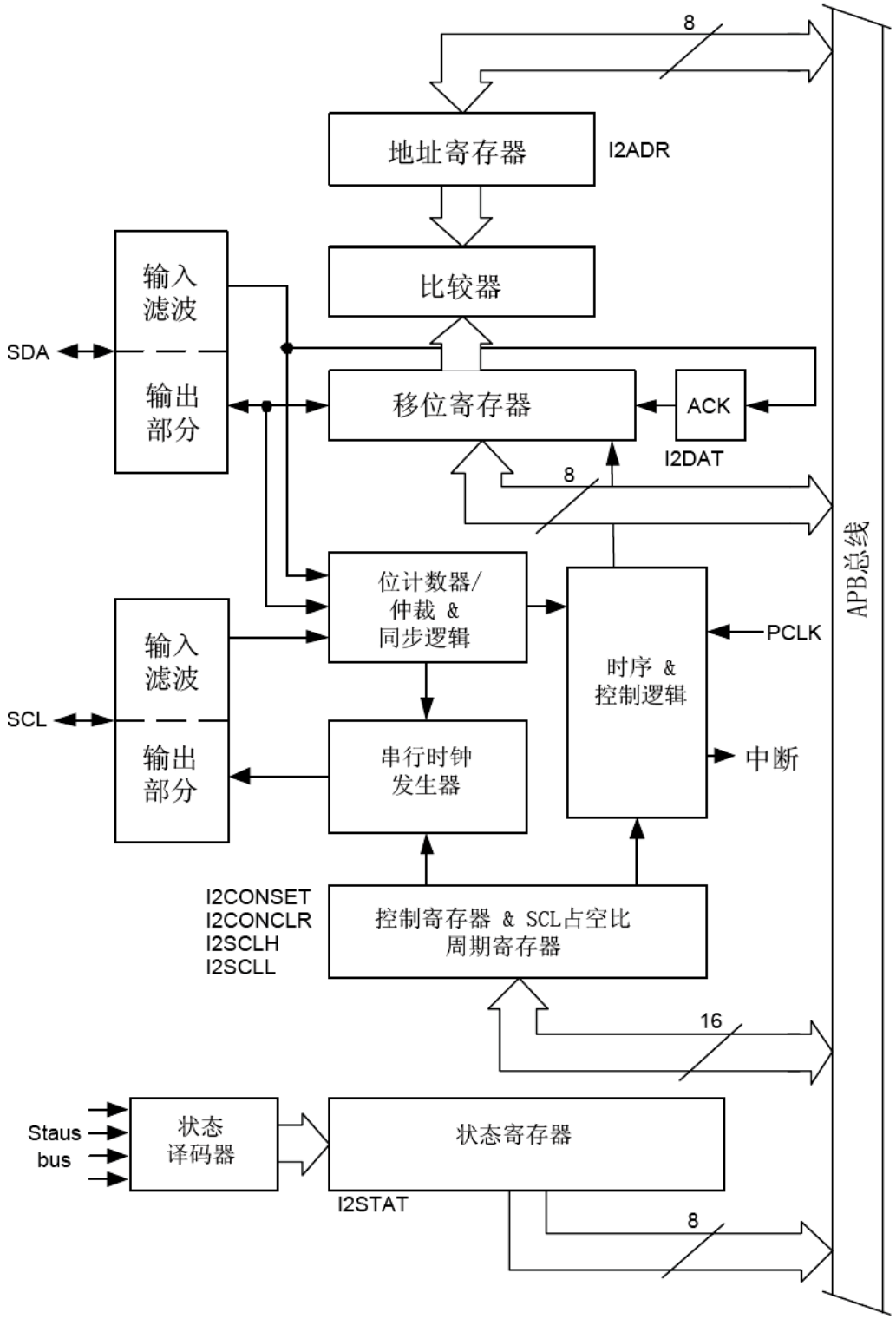


图 29 I²C 串行接口方框图

11.6.2 地址寄存器，I2ADDR

当器件编程用作从发送器或接收器时，该寄存器用来存放 I²C 模块响应的 7 位从地址（7 个最高位）。LSB（GC）用来使能通用调用地址（0x00）的识别。

11.6.3 比较器

比较器将接收到的 7 位从地址与其自身的从地址（I2ADR 的 7 个最高位）相比较。它还将首次接收到的 8 位字节与通用调用地址（0x00）相比较。如果任何一者相同，相应的状态位置位，产生中断请求。

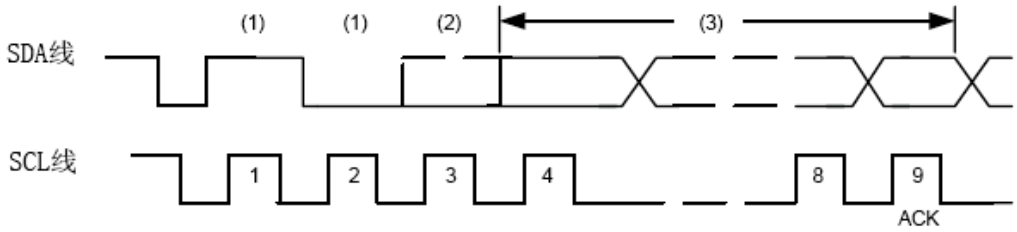
11.6.4 移位寄存器，I2DAT

该 8 位寄存器用来存放要发送的一个字节的串行数据或刚接收到的一个字节。I2DAT 的数据总是从右向左移动；最先发送 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。当数据正在移出时，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，在仲裁丢失时，主发送器到从接收器的转变和 I2DAT 中数据的更新同时进行。

11.6.5 仲裁和同步逻辑

在主发送器模式中，仲裁逻辑检查每个发送的逻辑 1 作为一个逻辑 1 真正出现在 I²C 总线上。如果总线的另一个器件撤消了一个逻辑 1 并将 SDA 线拉低，仲裁丢失，且 I²C 模块立刻由主发送器变为从接收器。I²C 模块继续输出时钟脉冲（在 SCL 上），直至发送完当前的串行字节。

仲裁也可能在主接收器模式中丢失。这种情况只在 I²C 模块正在向总线返回一个“非应答：（逻辑 1）”时出现。当总线的另一个器件将信号拉低时仲裁丢失。由于它只在串行字节结束时出现，因此 I²C 模块不会再产生时钟脉冲。图 30 所示为仲裁过程。



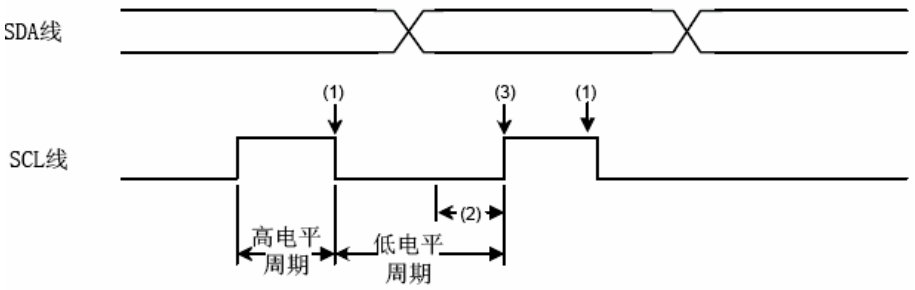
(1). 另一个器件发送相同的串行数据。

(2). 另一个器件通过拉低 SDA 线撤消了该 I²C 主机发送的一个逻辑 1（虚线）。仲裁丢失，且 I²C 进入从接收器模式。

(3). 此时 I²C 处于从接收器模式，但仍产生时钟脉冲，直至发送完当前字节。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。

图 30 仲裁过程

同步逻辑使得串行时钟发生器与另一个器件 SCL 线上的时钟脉冲同步。如果 2 个或更多主器件产生时钟脉冲，则高电平周期取决于产生最短高电平时间的器件；低电平周期取决于产生最长低电平时间的器件。图 31 所示为同步过程。



1. 另一个器件在 I²C 计时完一个完整的高电平时间之前拉低 SCL 线。其它器件决定了高电平（最短）的时间。
2. 另一个器件在 I²C 计时完一个完整的低电平时间并释放 SCL 之后继续拉低 SCL 线。I²C 时钟发生器被迫等待，直至 SCL 变高。其它器件决定了低电平（最长）的时间。
3. SCL 线被释放，时钟发生器开始计时高电平时间。

图 31 串行时钟同步

从机可以延长低电平时间来使总线主机减速。也可通过延长低电平时间来实现握手。延长低电平时间的操作在每位或一个完整字节传输之后执行。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 低电平时间。设置串行中断标志（SI），继续延长低电平时间，直至串行中断标志清除。

11.6.6 串行时钟发生器

当 I²C 模块在主发送器或主接收器模式时，该可编程时钟脉冲发生器提供 SCL 时钟脉冲。当 I²C 模块处于从机模式时时钟发生器关闭。I²C 输出时钟频率和占空比可通过 I²C 时钟控制寄存器编程。详见 I2CSCLL 和 I2CSCLH 寄存器的描述。输出时钟脉冲使用编程设定的占空比，除非总线与上面描述的其它 SCL 时钟源同步。

11.6.7 时序和控制

时序和控制逻辑产生串行字节处理的时序和控制信号。该逻辑模块为 I2DAT 提供移位脉冲，使能比较器，产生和检测起始和停止条件，接收和发送应答位，控制主机和从机模式，包含中断请求逻辑以及监控 I²C 总线状态。

11.6.8 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。

I2CONSET 可作为 I²C 控制寄存器的内容被读出。写 I2CONSET 将置位 I²C 控制器寄存器中相应的被写为 1 的位；相反，写 I2CONCLR 将清除 I²C 控制寄存器中相应的被写为 1 的位。

11.6.9 状态译码器和状态寄存器

状态译码器取出所有内部状态位并将它们压缩成一个 5 位的代码。该代码与每个 I²C 总线状态位一一对应。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 模块的所有四种模式都被使用，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

存器的高 5 位，直至中断标志被软件清除。状态寄存器的低 3 位总是为 0。如果状态代码用作服务程序的向量，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（见本节的软件例程）。

11.7 寄存器描述

每个 I²C 接口包含 7 个寄存器，如表 120 所示。

表 120 I²C 寄存器映射

名称	描述	访问	复位值[1]	I²C0 地址& 名称	I²C1 地址& 名称
I2CONSET	**I²C 控制置位寄存器。**当向该寄存器写入 1 时，I²C 控制寄存器中相应位置位。写 0 到 I²C 控制寄存器的相应位没有影响。	R/W	0x00	0xE001C000 I2C0CONSET	0xE005C000 I2C1CONSET
I2STAT	**I²C 状态寄存器。**在 I²C 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	0xE001C004 I2C0STAT	0xE005C004 I2C1STAT
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	0xE001C008 I2C0DAT	0xE005C008 I2C1DAT
I2ADR	**I²C 从地址寄存器**。包含从机模式下 I²C 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	0xE001C00C I2C0ADR	0xE005C00C I2C1ADR
I2SCLH	**SCH 占空比寄存器高半字**。确定 I²C 时钟的高时间。	R/W	0x04	0xE001C010 I2C0SCLH	0xE005C010 I2C1SCLH
I2SCLL	**SCL 占空比寄存器低半字**。确定 I²C 时钟的低时间。12nSCLL 和 12nSCLH 一起确定 I²C 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	0xE001C014 I2C0SCLL	0xE005C014 I2C1SCLL
I2CONCLR	**I²C 控制清零寄存器**。当向该寄存器中的位写入 1 时，I²C 控制寄存器中相应位被清零。写 0 到 I²C 控制寄存器的相应位没有影响。	WO	NA	0xE001C018 I2C0CONCLR	0xE005C018 I2C1CONCLR

[1] 复位值仅指已使用位中保存的数据。它不包括保留位的内容。

11.7.1 I²C 控制置位寄存器（I2CONSET：12C0 – I2C0CONSET: 0xE001C000 和 I2C1- I2C1CONSET: 0xE005C000）

I2CONSET 寄存器控制 I2CON 寄存器中位的设置，该寄存器中位的设置控制 I²C 接口的操作。写 1 到该寄存器使 I²C 控制寄存器中相应位置位。写 0 没有影响。

表 121 I²C 控制置位寄存器（I2CONSET：12C0，I2C0CONSET-地址 0xE001C000 和 I2C1，I2C1CONSET-地址 0xE005C000）位描述

位	名称	描述	复位值
1:0	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
2	AA	应答标志。见下文。	0
3	SI	I²C 中断标志。	0
4	STO	停止标志。见下文。	0
5	STA	起始标志。见下文。	0
6	I2EN	I²C 接口使能。见下文。	0
7	—	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

I2EN 为 I²C 接口使能。当该位置位时，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 位为 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”从机状态下，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 为起始标志。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 接口将进入主模式，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 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 为停止标志。在主模式中置位 STO，使 I²C 总线发送一个停止条件或在从模式中使总线从错误状态中恢复。当主模式中 STO=1 时，在总线上发送停止条件。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 为 I²C 中断标志。当 I²C 状态改变时该位置位。但是，由于在那种情况下中断服务程序不起作用，进入状态 F8 不置位 SI。

当 SI 置位时，SCL 线上的串行时钟低周期扩展，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。

AA 为声明应答标志。当该位置位时，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（SDA 上的低电平）：

1. 接收到从地址寄存器中的地址。
2. 当 I2ADR 中的通用调用位（GC）置位时，接收到通用调用地址。
3. 当 I²C 接口处于主接收器模式时，接收到一个数据字节。
4. 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 上的高电平）：

1. 当 I²C 接口处于主接收器模式时，接收到一个数据字节。
2. 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。

11.7.2 I²C 控制清零寄存器（I2CONCLR: I2C0 - I2C0CONCLR: 0xE001C018; I2C1-I2C1CONCLR:0xE005C018）

I2CONCLR 寄存器控制 I2CON 寄存器中位的清零，I2CON 寄存器控制 I²C 接口的操作。写 1 到该寄存器使 I²C 控制寄存器中相应位清零。写 0 没有影响。

表 122 I²C 控制清零寄存器（I2CONCLR: I2C0 - I2C0CONCLR: 地址 0xE001C018 和 I2C1 - I2C1CONCLR: 地址 0xE005C018）位描述

位	名称	描述	复位值
1:0	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
2	AAC	应答标志清零位。	0
3	SIC	I²C 中断标志清零位。	0
4	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
5	STAC	起始标志清零位。	0
6	I2ENC	I²C 接口禁止位。	0
7	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

AAC 应答标志清零位。向该位写入 1 清零 I2CONSET 寄存器中的 AA 位。写入 0 无效。

SIC 为 I²C 中断标志清零位。向该位写入 1 清零 I2CONSET 寄存器中的 SI 位。写入 0 无效。

STAC 为起始标志清零位。向该位写入 1 清零 I2CONSET 寄存器中的 STA 位。写入 0 无效。

I2ENC 为 I²C 接口禁止位。向该位写入 1 清零 I2CONSET 寄存器中的 I2EN 位。写入 0 无效。

11.7.3 I²C 状态寄存器（I2STAT: I2C0-I2C0STAT: 0xE001C004 和 I2C1-I2C1STAT: 0xE005C004）

每个 I²C 状态寄存器表示相应 I²C 接口的条件。I²C 状态寄存器为只读寄存器。

表 123 I²C 状态寄存器（I2STAT: I2C0-I2C0STAT: 地址 0xE001C004 和 I2C1-I2C1STAT: 地址 0xE005C004）位描述

位	符号	描述	复位值
2:0	-	这 3 个位不使用且总是为 0	0
7:3	Status	这些位给出 I²C 接口的真实状态位	0x1F

最低 3 位总是为 0。状态寄存器中的一个字节表示一个状态代码。一共有 26 种可能存

在的状态代码。当代码为 0xF8 时，无可用的相关信息，SI 位不会置位。所有其它 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。所有状态代码的描述见表 133 到表 136。

11.7.4 I²C 数据寄存器（I2DAT: I2C0-I2C0DAT: 0xE001C008 和 I2C1-I2C1DAT: 0xE005C008）

该寄存器包含要发送或刚接收的数据。当它没有处理字节的移位时，CPU 可对其进行读写。该寄存器只能在 SI 置位时访问。只要 SI 置位，I2DAT 中的数据就保持稳定。I2DAT 中的数据移位总是从右至左进行：第一个发送的位是 MSB（位 7），在接收字节时，第一个接收到的位存放在 I2DAT 的 MSB。

表 124 I²C 数据寄存器（I2DAT: I2C0-I2C0DAT: 地址 0xE001C008; I2C1-I2C1DAT: 地址 0xE005C008）位描述

位	符号	描述	复位值
7:0	Data	该寄存器保留已经接收，或准备要发送的数据值	0

11.7.5 I²C 从地址寄存器（I2ADR：I2C0-I2C0ADR: 0xE001C00C 和 I2C1-I2C1ADR: 地址 0xE005C00C）位描述

该寄存器可读可写，但只能在 I²C 设置为从模式时才能使用。在主模式中，该寄存器无效。I2ADR 的 LSB 为通用调用位。当该位置位时，通用调用地址（0x00）被识别。

表 125 I²C 从地址寄存器（I2ADR：I2C0-I2C0ADR: 0xE001C00C; I2C1-I2C1ADR: 0xE005C00C）位描述

位	符号	描述	复位值
0	GC	通用调用使能位	0
7:1	Address	从模式的 I²C 器件地址	0x00

11.7.6 I²C SCL 高电平占空比寄存器（I2SCLH: I2C0-I2C0SCLH: 0xE001 C010 和 I2C1-I2C1SCLH: 0xE005 C010）

表 126 I²C SCL 高电平占空比寄存器（I2SCLH: I2C0-I2C0SCLH: 地址 0xE001C010; I2C1-I2C1SCLH: 地址 0xE005C010）位描述

位	符号	描述	复位值
15:0	SCLH	SCL 高电平周期选择计数	0x 0004

11.7.7 I²C SCL 低电平占空比寄存器（I2SCLL: I2C0-I2C0SCLL: 0xE001 C014; I2C1-I2C1SCLL: 0xE005 C014）

表 127 I²C SCL 低电平占空比寄存器（I2SCLL: I2C0-I2C0SCLL: 地址 0xE001 C014; I2C1-I2C1SCLL: 地址 0xE005 C014）位描述

位	符号	描述	复位值
15:0	SCLL	SCL 低电平周期选择计数	0x 0004

11.7.8 选择合适的 I²C 数据率和占空比

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。频率由下面的公式得出（PCLK 是外设总线 APB 的频率）：

$$I^2C_{bitfrequency} = \frac{PCLK}{I2CSCLH + I2CSCLL} \quad (7)$$

I2SCLL 和 I2SCLH 的值不一定要相同。通过设定这两个寄存器可得到 SCL 的不同占空比。例如，I²C 总线规范定义 400KHz I²C 速率下不同值的 SCL 低时间和高时间。但寄存器的值必须确保 I²C 数据通信速率在 0 到 400KHz 之间。每个寄存器的值都必须大于等于 4。表 128 给出基于 PCLK 频率和 I2SCLL 和 I2SCLH 值的 I²C 总线速率的实例。

表 128 I²C 时钟速率的实例

I2SCLL+ I2SCLH	PCLK (MHz)下 I²C 的位速率（KHz）						
	1	5	10	16	20	40	60
8	125	-	-	-	-	-	-
10	100	-	-	-	-	-	-
25	40	200	400	-	-	-	-
50	20	100	200	320	400	-	-
100	10	50	100	160	200	400	-
160	6.25	31.25	62.5	100	125	250	375
200	5	25	50	80	100	200	300
400	2.5	12.5	25	40	50	100	150
800	1.25	6.25	12.5	20	25	50	75

11.8 I²C 操作模式的细节

4 种操作模式：

- 主发送器
- 主接收器
- 从接收器
- 从发送器

每种操作模式的数据传输如图 32，图 33，图 34，图 35 和图 36 所示。表 129 列出描述 I²C 操作模式时这些图中使用的缩写：

表 129 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）
$\overline{A}$	非应答位（SDA 为高电平）
Data	8 位数据字节
P	停止条件

在图 32 到图 36 中，圆圈用来指示串行中断标志何时被置位。圆圈中的数字表示的是 I2STAT 寄存器中的状态代码。每当出现这些状态代码时，必须执行服务程序来继续或结束串行传输。由于串行传输被挂起，这些服务程序并不重要，直至串行中断标志被软件清除。

当进入串行中断程序时，I2STAT 的状态代码用来指向跳转到的相应的服务程序。对于每个状态代码需要的软件操作以及后面串行传输的详细情况见表 133 到表 137。

11.8.1 主发送器模式

在主发送器模式下，数据字节发送到从接收器（见图 32）。在进入主发送器模式之前，I2CON 必须进行如下初始化：

表 130 I2CONSET 用于初始化主发送器模式

位	7	6	5	4	3	2	1	0
符号	-	I2EN	STA	STO	SI	AA	-	-
值	-	1	0	0	0	x	-	-

I²C 的速率也必须在 I2SCLL 和 I2SCLH 寄存器中配置。I2EN 必须设置为逻辑 1 来使能 I²C 模块。如果 AA 位复位，当另一个器件变成总线主机时，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 接口就不能进入从机模式。STA, STO 和 SI 必须复位。

可通过置位 STA 位进入主发送器模式。一旦总线空闲，I²C 逻辑将测试 I²C 总线并产生一个起始条件。当 START 条件被发送时，串行中断标志（SI）置位，状态寄存器（I2STAT）中的状态代码变为 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 的 SI 位必须在串行传输持续之前复位。

当发送完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)被重新置位，I2STAT 中可能是一系列不同的状态代码。主机模式下为 0x18, 0x20 或 0x38，从机模式(AA=逻辑 1)为 0x68, 0x78 或 0xB0。每个状态代码的相应操作详见表 133。在重复起始条件（状态 0x10）后，I²C 模块通过将 SLA+R 装入 I2DAT 切换为主接收器模式。

11.8.2 主接收器模式

在主接收器模式下，接收来自从发送器的数据字节（见图 33）。传输在主发送器模式中初始化。当发送完起始条件后，中断服务程序必须向 I2DAT 装入 7 位从机地址和数据方向位(SLA+R)。必须先清除 I2CON 中的 SI 位，再继续执行串行传输。

当发送完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)被重新置位，I2STAT 中可能是一系列不同的状态代码。主机模式下为 0x40, 0x48 或 0x38，从机模式(AA=1)为 0x68, 0x78 或 0xB0。每个状态代码的相应操作详见表 134。在重复起始条件（状态 0x10）后，I²C 模块通过将 SLA+W 装入 I2DAT 切换为主发送器模式。

11.8.3 从接收器模式

在从接收器模式下，接收来自主发送器的数据字节（见图 34）。要初始化从接收器模式，I2ADR 和 I2CON 必须被装入以下内容：

表 131 在从接收器模式中使用 I2C0ADR 和 I2C1ADR

位	7	6	5	4	3	2	1	0
符号	自身 7 位从地址							GC

高 7 位是主机寻址时 I^2C 模块响应的地址。如果 LSB(GC)被置位，I^2C 模块将响应通用调用地址(0x00)；否则忽略通用调用地址。

表 132 I2C0CONSET 和 I2C1CONSET 用于初始化从接收器模式

位	7	6	5	4	3	2	1	0
符号	-	I2EN	STA	STO	SI	AA	-	-
值	-	1	0	0	0	1	-	-

I^2C 总线的速率设置不影响从机模式下的 I^2C 模块。必须置位 I2EN 来使能 I^2C 模块。AA 位必须置位来使能 I^2C 模块应答其自身从机地址或通用调用地址。STA, STO 和 SI 必须被复位。

当 I2ADR 和 I2CON 完成初始化后，I^2C 模块一直等待，直至被从机地址及其后面的数据方向位寻址，该数据方向位必须为“0”(W)，便于 I^2C 模块工作在从接收器模式下。接收完其自身的从机地址和 W 位后，串行中断标志(SI)置位，可从 I2STAT 中读出一个有效的状态代码。该状态代码用作状态服务程序的向量。每个状态代码的相应操作见表 104。如果 I^2C 模块工作在主机模式下时仲裁丢失，也可进入从接收器模式（见状态 0x68 和 0x78）。

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 I^2C 模块将向 SDA 返回一个非应答（逻辑 1）。当 AA 复位时，I^2C 模块不响应其自身的从机地址或通用调用地址。但是，I^2C 总线仍被监控，而且，地址识别可随时通过置位 AA 来恢复。这就意味着 AA 位可用来暂时将 I^2C 模块从 I^2C 总线上分离出来。

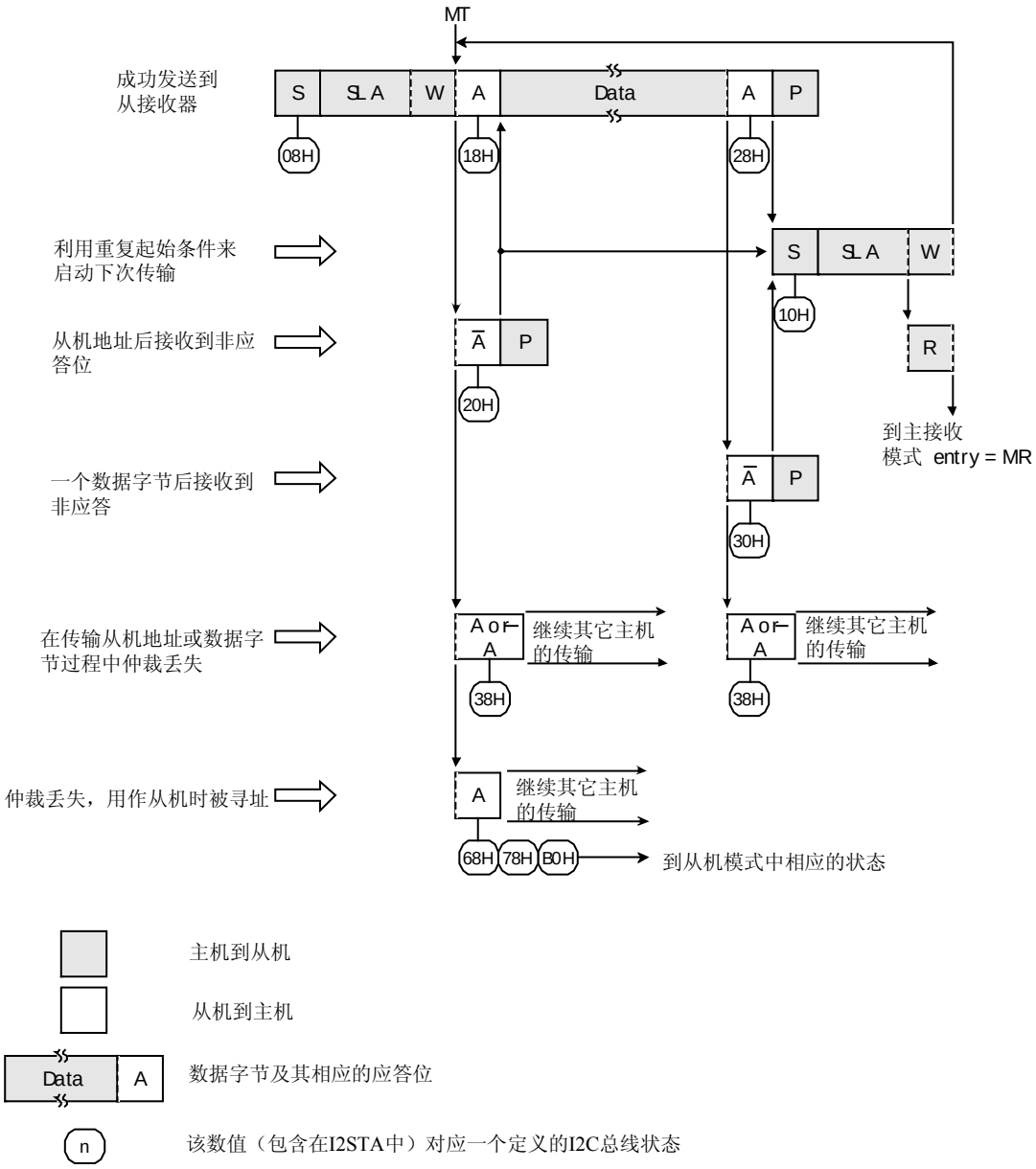


图 32 主发送器模式的格式和状态

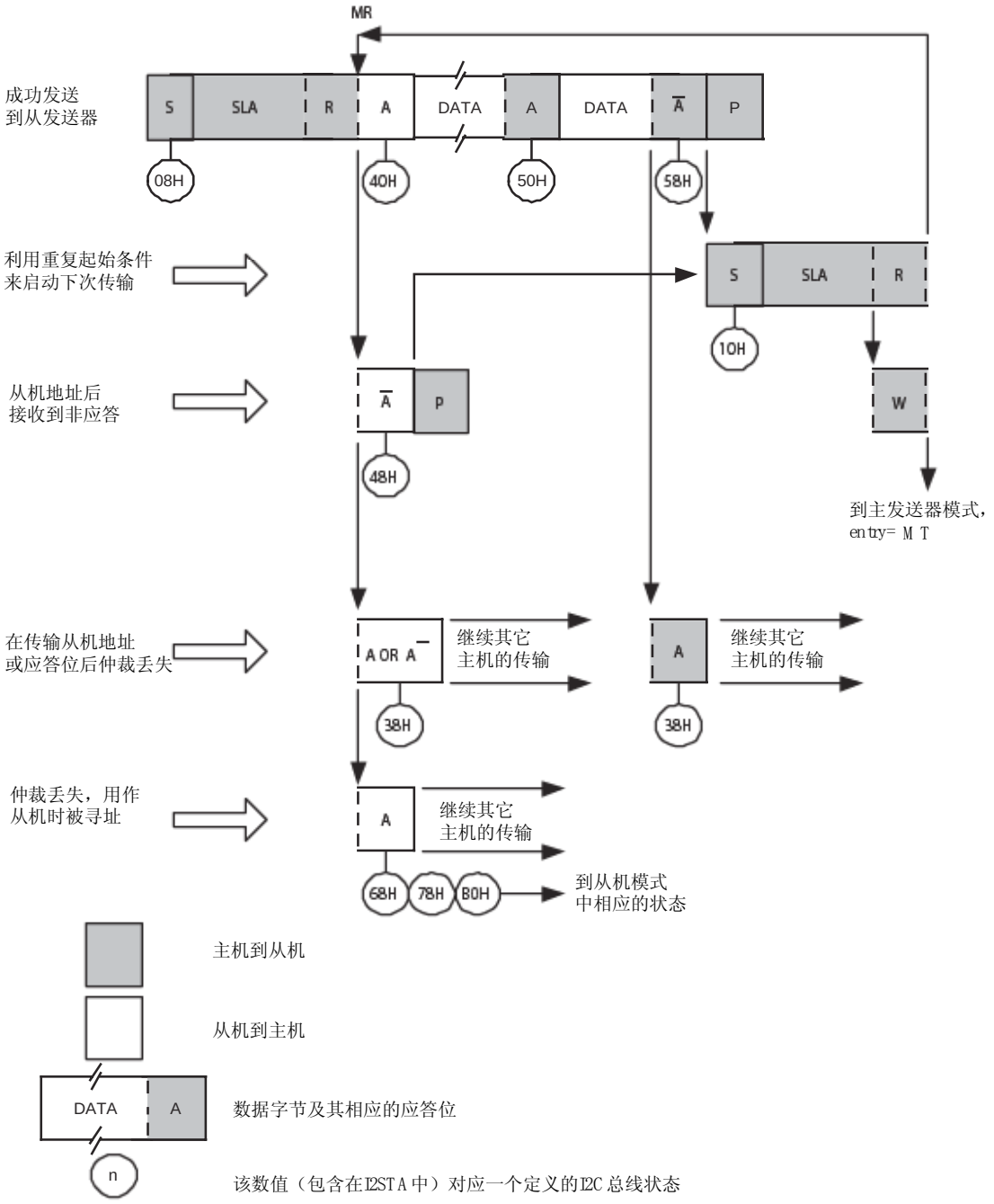


图 33 主接收器模式的格式和状态

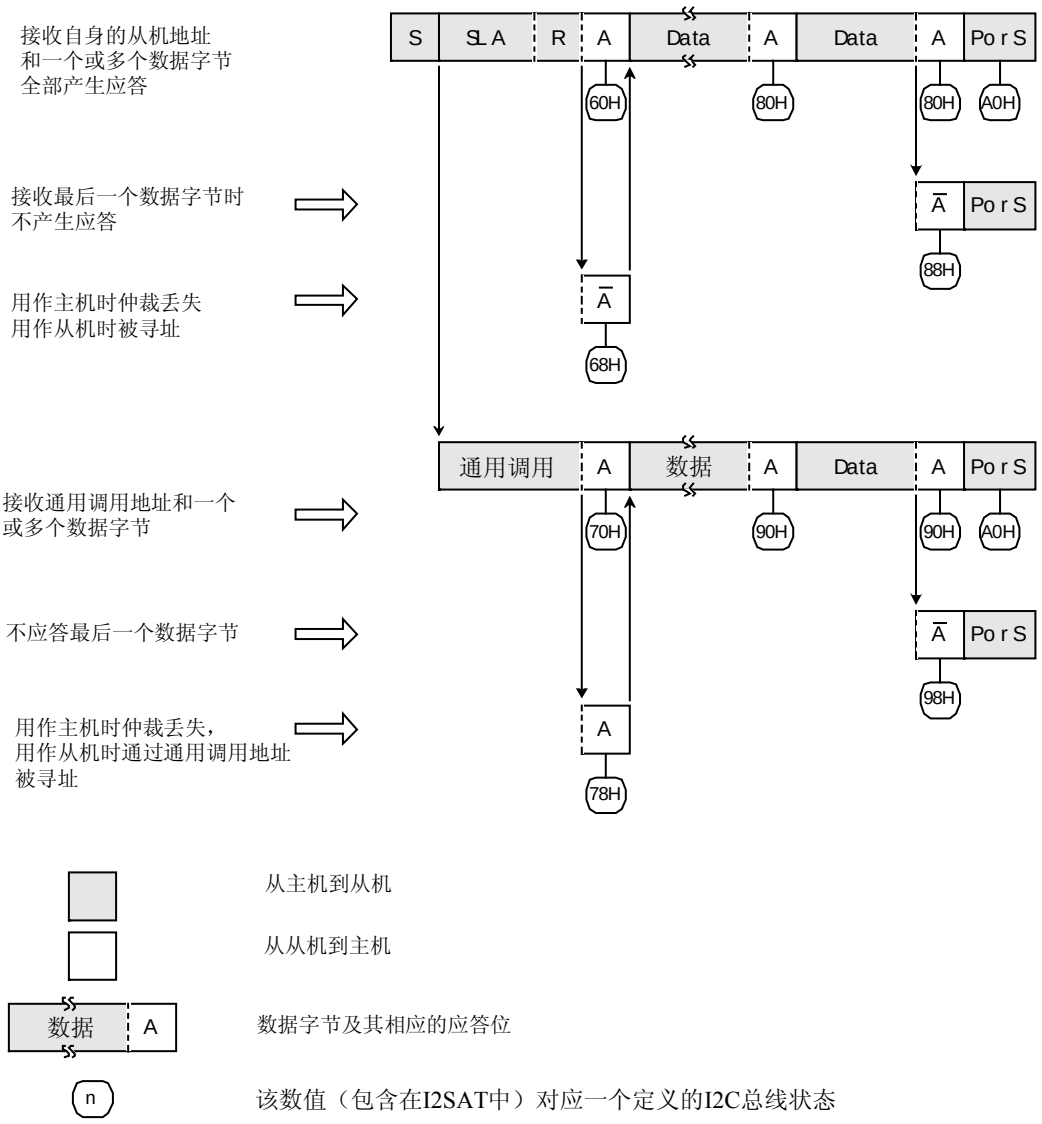


图 34 从接收器模式的格式和状态

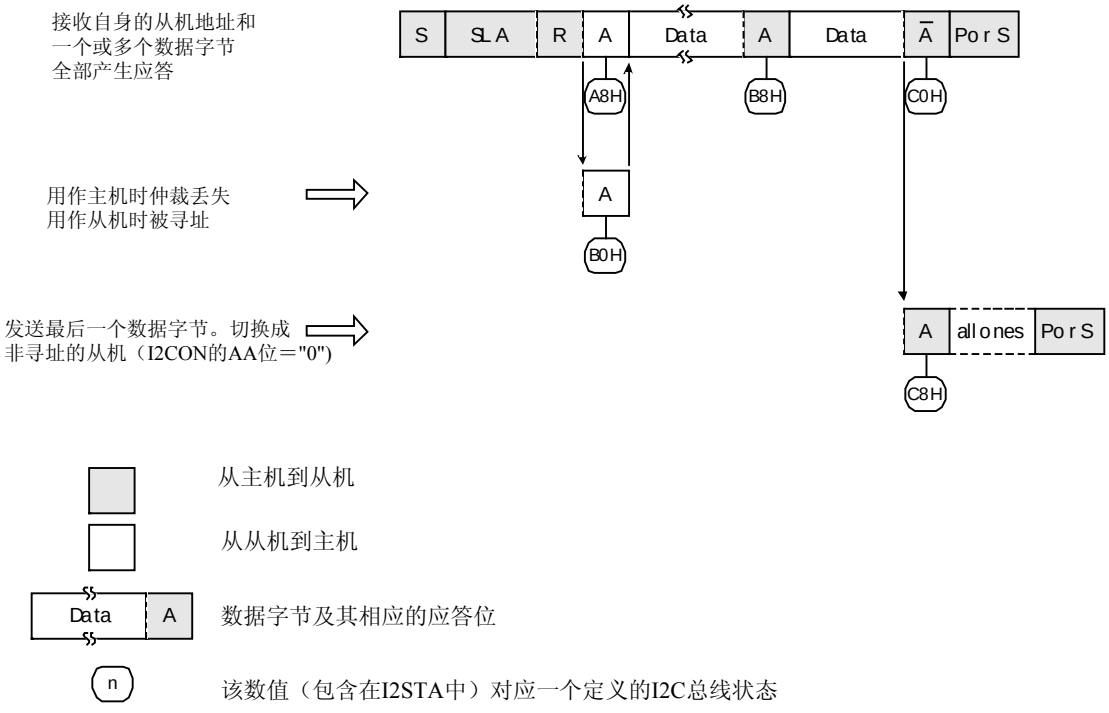


图 35 从发送器模式的格式和状态

11.8.4 从发送器模式

在从发送器模式下，向主接收器发送数据字节（见图 35）。数据传输在从接收器模式下初始化。当 I2ADR 和 I2CON 完成初始化后，I²C 模块一直等待，直至被自身的从机地址及其后面的数据方向位寻址，该数据方向位必须为“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 位后，串行中断标志(SI)置位，可从 I2STAT 中读出一个有效的状态代码。该状态代码用作状态服务程序的向量，每个状态代码的相应操作见表 136。如果 I²C 模块工作在主机模式下时仲裁丢失，也可进入从发送器模式（见状态 0xB0）。

如果 AA 位在传输过程中复位，则 I²C 模块将发送传输的最后一个字节并进入状态 0xC0 或 0xC8。I²C 模块切换为非寻址的从机模式，如果它继续传输将忽略主接收器。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，地址识别可随时通过置位 AA 来恢复。这就意味着 AA 位可用来暂时将 I²C 模块从 I²C 总线上分离出来。

表 133 主发送器模式

状态代码 (I2CSTAT)	I²C 总线硬件状态	应用软件的响应					I²C 硬件执行的下一个动作
		读/写 I2DAT	写 I2CON				
			STA	STO	SI	AA	
0x08	已发送起始条件	装入 SLA+W	x	0	0	x	将发送 SLA+W，接收 ACK 位
0x10	已发送重复起始条件	装入 SLA+W 或	X	0	0	X	同上；
		装入 SLA+R	x	0	0	x	将发送 SLA+W，I²C 模块将切换为 MST/REC 模式

续上表

状态代码 (I2CSTAT)	I²C 总线硬件状态	应用软件的响应					I²C 硬件执行的下一个动作
		读/写 I2DAT	写 I2CON				
			STA	STO	SI	AA	
0x18	已发送 SLA+W；已接收 ACK	装入数据字节	0	0	0	x	将发送数据字节，接收 ACK 位
		无 I2DAT 动作	1	0	0	x	将发送重复起始条件
		无 I2DAT 动作	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	0	0	0	x	将发送数据字节，接收 ACK 位
		无 I2DAT 动作	1	0	0	x	将发送重复起始条件
		无 I2DAT 动作	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 中的数据字节；已接收 ACK	装入数据字节	0	0	0	x	将发送数据字节，接收 ACK 位
		无 I2DAT 动作	1	0	0	x	将发送重复起始条件
		无 I2DAT 动作	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 中的数据字节；已接收非 ACK	装入数据字节	0	0	0	x	将发送数据字节，接收 ACK 位
		无 I2DAT 动作	1	0	0	x	将发送重复起始条件
		无 I2DAT 动作	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	0	0	0	x	I²C 总线将被释放；进入不可寻址从模式
		无 I2DAT 动作	1	0	0	x	当总线变为空闲时发送起始条件

表 134 主接收器模式

状态代码 (I2CSTAT)	I²C 总线硬件状态	应用软件的响应					I²C 硬件执行的下一个动作
		读/写 I2DAT	写 I2CON				
			STA	STO	SI	AA	
0x08	已发送起始条件	装入 SLA+R	x	0	0	x	将发送 SLA+R，接收 ACK 位
0x10	已发送重复起始条件	装入 SLA+R	x	0	0	x	同上
		装入 SLA+W	x	0	0	x	将发送 SLA+W，I²C 将切换到 MST/TRX 模式
0x38	在非 ACK 位中丢失仲裁	无 I2DAT 动作	0	0	0	x	I²C 总线将被释放；I²C 将进入从模式
		无 I2DAT 动作	1	0	0	x	当总线恢复空闲后发送起始条件
0x40	已发送 SLA+R；已接收 ACK	无 I2DAT 动作	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	1	0	0	x	将发送重复起始条件
		无 I2DAT 动作	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；已返回 ACK	读数据字节	0	0	0	0	将接收数据字节，返回非 ACK 位
		读数据字节	0	0	0	1	将接收数据字节；返回 ACK 位
0x58	已接收数据字节；已返回非 ACK	读数据字节	1	0	0	x	将发送重复起始条件
		读数据字节	0	1	0	x	将发送停止条件；STO 标志将复位
		读数据字节	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 135 从接收器模式

状态代码 (I2CSTAT)	I²C 总线硬件状态	应用软件的响应					I²C 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 SLA+W;	无 I2DAT 动作	x	0	0	0	将接收数据字节并返回非 ACK 位
	已返回 ACK	无 I2DAT 动作	x	0	0	1	将接收数据字节并返回 ACK 位
0x68	主控器时在 SLA+R/W 中丢失仲裁;已接收自身 SLA+W,已返回 ACK	无 I2DAT 动作	x	0	0	0	将接收数据字节并返回非 ACK 位
		无 I2DAT 动作	x	0	0	1	将接收数据字节并返回 ACK 位
0x70	已接收通用调用地址	无 I2DAT 动作	x	0	0	0	将接收数据字节并返回非 ACK 位
	(0x00);已返回 ACK	无 I2DAT 动作	x	0	0	1	将接收数据字节并返回 ACK 位
0x78	主控器时在 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	x	0	0	0	将接收数据字节并返回非 ACK 位
		无 I2DAT 动作	x	0	0	1	将接收数据字节并返回 ACK 位
0x80	前一次寻址使用自身从地址；已接收数据字节；已返回 ACK	读数据字节	x	0	0	0	将接收数据字节并返回非 ACK 位
		读数据字节	x	0	0	1	将接收数据字节并返回 ACK 位
0x88	前一次寻址使用自身 SLA 地址；已接收数据字节；已返回非 ACK	读数据字节	0	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址
		读数据字节	0	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=1，将识别通用调用地址
		读数据字节	1	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址；当总线空闲后发送起始条件
		读数据字节	1	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=1，将识别通用调用地址；当总线空闲后发送起始条件
0x90	前一次寻址使用通用调用；已接收数据字节；已返回 ACK	读数据字节	x	0	0	0	将接收数据字节并返回非 ACK 位
		读数据字节	x	0	0	1	将接收数据字节并返回 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	读数据字节	0	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址
		读数据字节或	0	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=1，将识别通用调用地址
		读数据字节或	1	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址；当总线空闲后发送起始条件
		读数据字节	1	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=1，将识别通用调用地址；当总线空闲后发送起始条件

续上表

状态代码 (I2CSTAT)	I²C 总线硬件状态	应用软件的响应					I²C 硬件执行的下一个动作
		读/写 I2DAT	写 I2CON				
			STA	STO	SI	AA	
0xA0	当使用 SLV/REC 或 SLV/TRX 静态寻址时,接收到停止条件或重复的起始条件	无 STDAT 动作	0	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址
		无 STDAT 动作	0	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果 I2ADR[0]=1，将识别通用调用地址
		无 STDAT 动作	1	0	0	0	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址；当总线空闲后发送起始条件
		无 STDAT 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]=1，将识别通用调用地址；当总线空闲后发送起始条件

表 136 从发送器模式

状态代码 (I2CSTAT)	I²C 总线硬件状态	应用软件的响应					I²C 硬件执行的下一个动作
		读/写 I2DAT	写 I2CON				
			STA	STO	SI	AA	
0xA8	已接收自身 SLA+R；已返回 ACK	装入数据字节或	x	0	0	0	将发送最后的数据字节并接收 ACK 位
		装入数据字节	x	0	0	1	将发送数据字节并接收 ACK 位
0xB0	主控器时在 SLA+R/W 中丢失仲裁；已接收自身 SLA+R，已返回 ACK	装入数据字节或	x	0	0	0	将发送最后的数据字节并接收 ACK 位
		装入数据字节	x	0	0	1	将发送数据字节并接收 ACK 位
0xB8	已发送 I2DAT 中数据字节；已返回 ACK	装入数据字节或	x	0	0	0	将发送最后的数据字节并接收 ACK 位
		装入数据字节	x	0	0	1	将发送数据字节并接收 ACK 位
0xC0	已发送 I2DAT 中数据字节；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 或通用调用地址
		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]=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 或通用调用地址；当总线空闲后发送起始条件
		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]=1，将识别通用调用地址；当总线空闲后发送起始条件
0xC8	已发送 I2DAT 中最后的数据字节（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 或通用调用地址
		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]=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 或通用调用地址；当总线空闲后发送起始条件
		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]=1，将识别通用调用地址；当总线空闲后发送起始条件

11.8.5 各种不同的状态

有 2 个 I2STAT 代码与定义的 I²C 硬件状态不对应（见表 137），下面将对这两种状态进行讨论：

11.8.6 I2STAT= 0xF8:

该状态码表示没有任何可用的相关信息，因为串行中断标志 SI 还没有被置位。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。

11.8.7 I2STAT= 0x00：

该状态代码表示一个总线错误在 I²C 串行传输过程中出现。当起始或停止条件出现在格式帧的非法位置上时产生总线错误。这些非法位置的例子为地址字节、数据字节或应答位的串行传输过程。当外部干扰影响到内部 I²C 模块信号时也会产生总线错误。总线错误出现时 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。这使得 I²C 模块进入“非寻址的”从机模式（已定义的状态）并清除 STO 标志（I2CON 中的其它位不受影响）。SDA 和 SCL 线被释放（不发送停止条件）。

表 137 各种不同的状态

状态代码 (I2CSTAT)	I²C 总线和硬件状态	应用软件的响应					I²C 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 I2CON 动作				等待或执行当前传输
0x00	由于非法的起始或停止条件，在 MST 或选择的从机模式中出现总线错误。当外部干扰使 I²C 模块进入一个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。所有情况下，总线被释放、I²C 模块切换到非寻址的 SLV 模式。STO 复位。

11.8.8 一些特殊的情况：

I²C 硬件可以处理出现在串行传输过程中的以下特殊情况：

11.8.9 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可能产生重复起始条件。如果此时另一个主机同时产生重复起始条件将出现一种特殊情况（见图 36）。即使出现这种情况，任何一个主机都不会丢失仲裁，因为它们都发送相同的数据。

如果 I²C 硬件在产生重复起始条件之前在 I²C 总线上检测到重复起始条件，它将释放总线，并且不产生中断请求。如果另一个主机通过产生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），并开始启动重新尝试完整的串行数据传输。

11.8.10 仲裁丢失后的数据传输

仲裁可能在主发送器和主接收器模式中丢失（见图 30）。仲裁丢失通过 I2STAT 中的下

列状态代码来指示：0x38, 0x68, 0x78 和 0xB0（见图 32 和图 33）。

如果 I2CON 中的 STA 标志被服务这些状态的程序置位，则当总线再次空闲时，发送起始条件（状态 0x08），不受 CPU 的影响，并开始启动重新尝试完整的串行数据传输。

11.8.11 强制访问 I²C 总线

在某些应用中，非控制源有可能造成总线挂起。在这些情况下，干扰、总线的暂时中断或 SDA 和 SCL 之间的暂时短路都会引发问题的产生。

如果非控制源产生一个多余的起始条件或屏蔽一个停止条件，则 I²C 总线一直处于忙状态。如果 STA 标志被设置且在相应的时间内未访问总线，那么有可能强制访问 I²C 总线。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停止条件一样，可以发送起始条件。STO 标志通过硬件清除（见图 34）。

11.8.12 SCL 或 SDA 低电平妨碍 I²C 总线的操作

如果 SDA 或 SCL 被一个非控制源拉低，将出现 I²C 总线挂起。如果 SCL 线被总线上的器件妨碍（拉低），则不能进行更进一步的串行传输，并且 I²C 硬件也无法解决此类问题。这时，问题必须由将 SCL 线拉低的器件来解决。

如果 SDA 线被总线上的另一个器件妨碍（例如从机器件不能位同步），可通过向 SCL 线发送额外的时钟脉冲来解决（见图 38）。STA 标志置位时 I²C 硬件发送额外的时钟脉冲，但不会产生起始条件，因为 I²C 总线空闲时 SDA 线被拉低。每当在 SCL 线上产生 2 个额外的时钟脉冲后，I²C 硬件就尝试产生一个起始条件。当 SDA 线最终被释放时，发送正常的起始条件，进入 0x08 状态，并继续串行传输。

当 SDA 被干扰（拉低）时，若出现强制总线访问或重复发送起始条件，I²C 硬件如上所述执行相同的操作。每一种情况下，成功发送起始条件后进入 0x08 状态，继续进行正常的串行传输。注意 CPU 不参与此类总线挂起问题的解决。

11.8.13 总线错误

当起始或停止条件出现在格式帧的非法位置上时出现总线错误。非法位置是指串行传输过程中的地址字节、数据位或应答位。

当 I²C 硬件作为主机或被寻址的从机处理串行传输时，它仅对总线错误有反应。检测到总线错误时，I²C 模块立即切换成非寻址的从机模式，释放 SDA 和 SCL 线，设置中断标志，并将 0x00 装入状态寄存器。该状态代码可用作状态服务程序的向量，尝试再次终止串行传输或仅从错误条件中恢复，如表 137 所示。

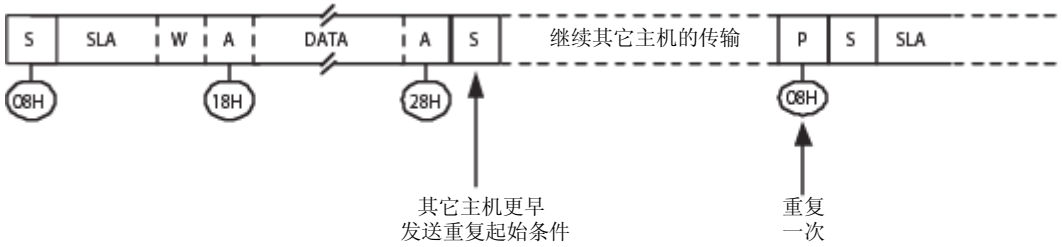


图 36　两个主机同时发送重复起始条件

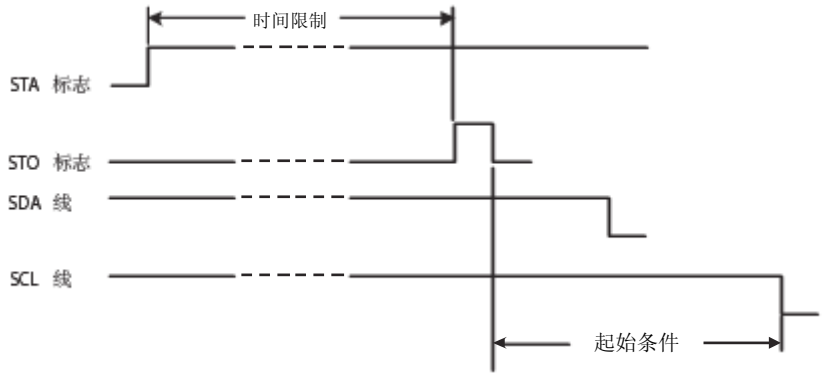
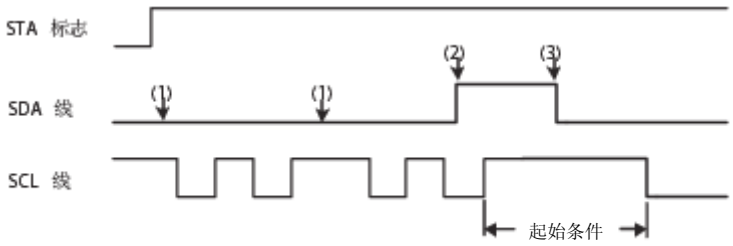


图 37 强制访问忙 I²C 总线



1. 发送起始条件失败
2. SDA 线释放
3. 成功发送起始条件；进入 08H 状态。

图 38 从 SDA 低电平造成的总线操作妨碍中恢复

11.8.14 I²C 状态服务程序

本节提供了不同 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

26 个状态服务程序支持 4 种 I²C 操作模式

11.8.15 初始化

在初始化示例中，I²C 模块可工作在主机和从机模式。每种模式下缓冲区都可用于发送和接收。初始化程序执行以下功能：

I2ADR 装入器件自身的从机地址和通用调用位(GC)

I²C 中断使能，设置中断优先级位

从机模式通过同时设置 I2CON 寄存器中的 I2EN 和 AA 位来使能，串行时钟频率（主机模式）由装入 I2CON 中 CR0 和 CR1 的值来定义。主机程序必须在主程序中开始执行。

I²C 硬件开始在 I²C 总线上检查自身的从机地址和通用调用位。一旦检测到通用调用或自身从地址，请求中断，相应的状态信息装入 I2STAT。

11.8.16 I²C 中断服务

当进入 I²C 中断时，I2STAT 包含一个状态代码，用来识别要执行的 26 个状态服务中的一个。

11.8.17 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。

11.8.18 实际应用中的状态服务

状态服务程序显示了响应 26 个 I²C 状态代码必须执行的典型操作。如果 4 种 I²C 操作模式中的一个或多个未使用，则未使用模式的相关状态服务可被忽略，只要小心处理那些不会出现的状态即可。

在应用中，可能需要在 I²C 操作过程中执行一些超时处理，来限制不起作用的总线或丢失的服务程序。

11.9 软件举例

11.9.1 初始化程序

将 I²C 接口初始化用作从机和/或主机的例子。

1. 将自身的从机地址装入 I2ADR，使能通用调用识别（如果需要）。
2. 使能 I²C 中断。
3. 写 0x44 到 I2CONSET 来置位 I2EN 和 AA 位，使能从机功能。对于主机功能，写 0x40 到 I2CONSET。

11.9.2 启动主机发送功能

通过建立缓冲区、指针和数据计数器来开始主机发送操作，然后启动。

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，并且添加写位。
3. 写 0x20 到 I2CONSET 来置位 STA 位。
4. 在主机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的信息长度。
6. 退出。

11.9.3 启动主机接收功能

通过建立缓冲区、指针和数据计数器来开始主机接收操作，然后启动。

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，并且添加读位。
3. 写 0x20 到 I2CONSET 来置位 STA 位。
4. 建立主机接收缓冲区。

5. 初始化主机数据计数器来匹配接收到的信息长度。
6. 退出。

11.9.4 I²C 中断程序

确定 I²C 状态和要使用的状态程序。

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。

11.9.5 非模式指定的状态

11.9.5.1 状态：0x00

总线错误。进入非寻址的从机模式并释放总线。

1. 写 0x14 到 I2CONSET 来置位 STO 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

11.9.5.2 主机状态

状态 08 和状态 10 适用于主发送和主接收模式。R/W 位决定下一个状态是否在主发送模式或主接收模式中。

11.9.5.3 状态：0x08

已发送起始条件。将发送从机地址＋R/W 位和接收 ACK 位。

1. 向 I2DAT 写入从机地址和 R/W 位。
2. 写 0x04 到 I2CONSET 来置位 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 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。

11.9.5.4 状态：0x10

已发送重复起始条件。将发送从机地址＋R/W 位和接收 ACK 位。

1. 向 I2DAT 写入从机地址和 R/W 位。
2. 写 0x04 到 I2CONSET 来置位 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 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。

11.9.6 主机发送器状态

11.9.6.1 状态：0x18

前面的状态是状态 8 或状态 10，已发送从机地址＋写，已接收 ACK。将发送第一个数据字节和接收 ACK 位。

1. 将主机发送缓冲区的第一个数据字节装入 I2DAT。
2. 写 0x04 到 I2CONSET 来置位 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 主机发送缓冲区指针加 1。
5. 退出。

11.9.6.2 状态：0x20

已发送从机地址＋写，已接收到非应答。将发送停止条件。

1. 写 0x14 到 I2CONSET 来置位 STO 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

11.9.6.3 状态：0x28

已发送数据，已接收 ACK。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. 主机数据计数器减 1，如果不是最后一个数据字节就跳到第 5 步。
2. 写 0x14 到 I2CONSET 来置位 STO 和 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 退出。
5. 向 I2DAT 装入主机发送缓冲区的下一个数据字节。
6. 写 0x04 到 I2CONSET 来置位 AA 位。
7. 写 0x08 到 I2CONCLR 来清除 SI 标志。
8. 主机发送缓冲区指针加 1。
9. 退出。

11.9.6.4 状态：0x30

已发送数据，已接收到非应答。将发送停止条件。

1. 写 0x14 到 I2CONSET 来置位 STO 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

11.9.6.5 状态：0x38

仲裁在传输从机地址＋写或数据过程中已丢失。总线已被释放且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. 写 0x24 到 I2CONSET 来置位 STA 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

11.9.7 主机接收器状态

11.9.7.1 状态：0x40

前面的状态是状态 08 或状态 10，已发送从机地址＋读，已接收到 ACK。将接收数据和返回 ACK。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

11.9.7.2 状态：0x48

已发送从机地址＋读，已接收到非应答。将发送停止条件。

1. 写 0x14 到 I2CONSET 来置位 STO 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

11.9.7.3 状态：0x50

已接收数据，已返回 ACK。将从 I2DAT 读取数据。将接收其它的数据。如果这是最后一个数据字节，则将返回非应答，否则将返回 ACK。

1. 读取 I2DAT 中的数据字节，存放到主机接收缓冲区。
2. 主机数据计数器减 1，如果不是最后一个数据字节就跳到第 5 步。
3. 写 0x0C 到 I2CONCLR 来清除 SI 标志和 AA 位。
4. 退出。
5. 写 0x04 到 I2CONSET 来置位 AA 位。
6. 写 0x08 到 I2CONCLR 来清除 SI 标志。
7. 主机接收缓冲区指针加 1。
8. 退出。

11.9.7.4 状态：0x58

已接收到数据，已返回非应答。将从 I2DAT 中读取数据和发送停止条件。

1. 读取 I2DAT 中的数据字节，存放到主机接收缓冲区。
2. 写 0x14 到 I2CONSET 来置位 STO 和 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 退出。

11.9.8 从机接收器状态

11.9.8.1 状态：0x60

已接收到自身从机地址＋写和返回 ACK。将接收数据和返回 ACK。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 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。

11.9.8.2 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＋写和返回 ACK。将接收数据和返回 ACK。当总线再次空闲后置位 STA 来重新启动主机模式。

1. 写 0x24 到 I2CONSET 来置位 STA 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 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。

11.9.8.3 状态：0x70

已接收到通用调用位和返回 ACK。将接收数据和返回 ACK。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 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。

11.9.8.4 状态：0x78

用作总线主机时仲裁已在传输从机地址＋R/W 位时丢失。已接收到通用调用和返回 ACK。将接收数据和返回 ACK。当总线再次空闲后置位 STA 来重新启动主机模式。

1. 写 0x24 到 I2CONSET 来置位 STA 和 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 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。

11.9.8.5 状态：0x80

以前用自身从机地址寻址。已接收到数据并返回 ACK。将读取其它的数据。

1. 读取 I2DAT 的数据字节，存放到从机接收缓冲区。
2. 从机数据计数器减 1，如果不是最后一个数据字节就跳到第 5 步。
3. 写 0x0C 到 I2CONCLR 来清除 SI 标志和 AA 位。
4. 退出。
5. 写 0x04 到 I2CONSET 来置位 AA 位。
6. 写 0x08 到 I2CONCLR 来清除 SI 标志。
7. 从机接收缓冲区指针加 1。
8. 退出。

11.9.8.6 状态：0x88

以前用自身从机地址寻址。已接收到数据并返回了非应答。接收到的数据将不保存。进入非寻址的从机模式。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

11.9.8.7 状态：0x90

以前用通用调用寻址。已接收到数据和返回了 ACK。将保存接收到的数据。只有将要接收的第一个数据字节带有 ACK，其它数据字节都是非应答。

1. 读取 I2DAT 的数据字节，存放到从机接收缓冲区。
2. 写 0x0C 到 I2CONCLR 来清除 SI 标志和 AA 位。
3. 退出。

11.9.8.8 状态：0x98

以前用通用调用寻址。已接收到数据和返回非应答。接收到的数据将不保存。进入非寻址的从机模式。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

11.9.8.9 状态：0xA0

接收到停止条件或重复起始条件，但仍作为从机寻址。将不保存数据。进入非寻址的从机模式。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

11.9.9 从机发送器状态

11.9.9.1 状态：0xA8

已接收到自身从机地址＋读和返回 ACK。将发送数据和接收 ACK 位。

1. 将从机发送缓冲区的第一个数据字节装入 I2DAT。
2. 写 0x04 到 I2CONSET 来置位 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 建立从发送模式数据缓冲区。
5. 从机发送缓冲区指针加 1。
6. 退出。

11.9.9.2 状态：0xB0

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＋读和返回 ACK。将发送数据和接收 ACK 位。当总线再次空闲后置位 STA 来重新启动主机模式。

1. 将从机发送缓冲区的第一个数据字节装入 I2DAT。
2. 写 0x24 到 I2CONSET 来置位 STA 和 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 建立从发送模式数据缓冲区。
5. 从机发送缓冲区指针加 1。
6. 退出。

11.9.9.3 状态：0xB8

已发送数据和接收到 ACK。将发送数据和接收 ACK 位。

1. 将从机发送缓冲区的数据字节装入 I2DAT。
2. 写 0x04 到 I2CONSET 来置位 AA 位。
3. 写 0x08 到 I2CONCLR 来清除 SI 标志。
4. 从机发送缓冲区指针加 1。
5. 退出。

11.9.9.4 状态：0xC0

已发送数据，已接收到非应答。进入非寻址的从机模式。

1. 写 0x04 到 I2CONSET 来置位 AA 位。
2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

11.9.9.5 状态：0xC8

已发送最后一个数据字节和接收到 ACK。进入非寻址的从机模式。

1. 写 0x04 到 I2CONSET 来置位 AA 位。

2. 写 0x08 到 I2CONCLR 来清除 SI 标志。
3. 退出。

第12章 SPI 接口（SPI0）

12.1 特性

单个完整和独立的 SPI 控制器

遵循串行外设接口(SPI)规范

同步、串行、全双工通信

组合的 SPI 主机和从机

最大数据位速率为输入时钟速率的 1/8

每次传输 8 到 16 位

12.2 描述

12.2.1 SPI 概述

SPI 是一个全双工的串行接口。它设计成可以处理在一个给定总线上多个互连的主机和从机。在给定的数据传输过程中，接口上只能有一个主机和一个从机能够通信。在一次数据传输中，主机总是向从机发送数据的 8～16 位，而从机也总是向主机发送一个字节数据。

12.2.2 SPI 数据传输

图 39 所示为 SPI 的 4 种不同数据传输格式的时序。该时序图描述的是一个 8 位数据的传输。需要注意的是，该时序图分成了 3 个水平的部分。第一部分描述 SCK 和 SSEL 信号。第二部分描述了 CPHA=0 时的 MOSI 和 MISO 信号。第三部分描述了 CPHA＝1 时的 MOSI 和 MISO 信号。

在时序图的第一部分需要注意两点。第一，时序图包含了 CPOL 设置为 0 和 1 的情况。第二，SSEL 信号的激活和未激活。当 CPHA=1 时，SSEL 信号在数据传输之间时总是保持未激活状态。当 CPHA=0 时则不能保证这一点（信号有可能保持激活状态）。

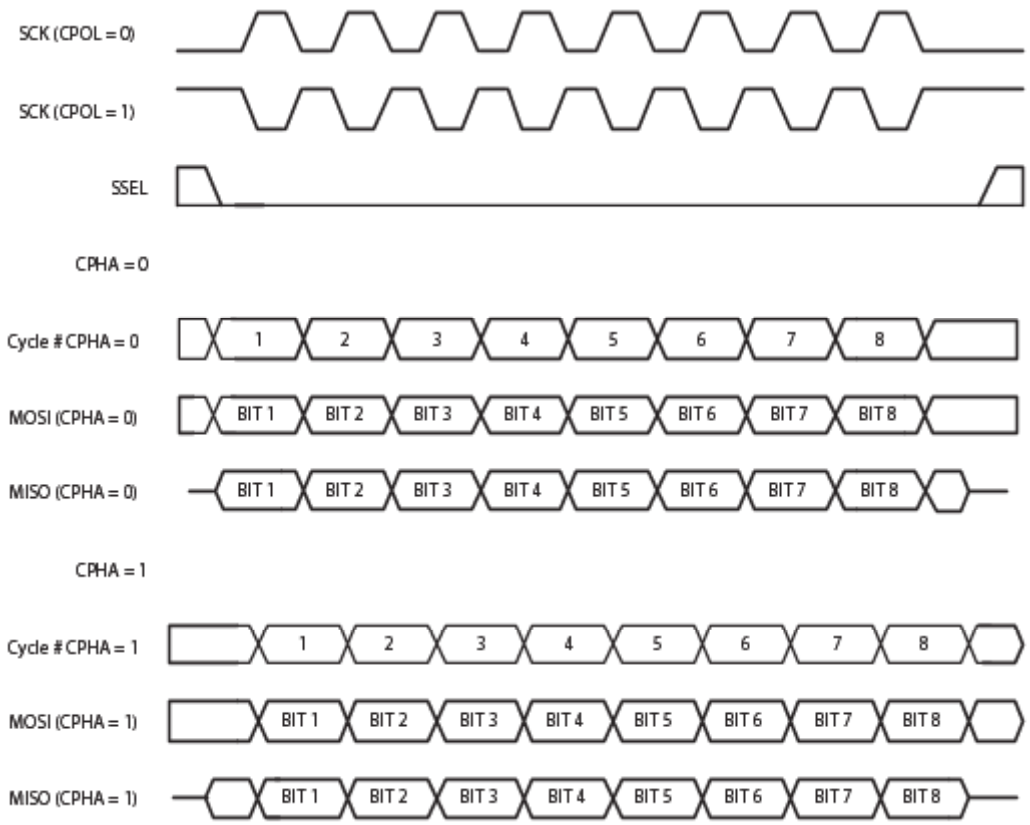


图 39 SPI 数据传输格式（CPHA=0 和 CPHA=1）

数据和时钟的相位关系在表 138 中描述。该表汇集了下面情况下 CPOL 和 CPHA 的每一种设定：

- 当驱动第一个数据位时
- 当驱动其它所有数据位时
- 当采样数据时

表 138 SPI 数据和时钟的相位关系

CPOL 和 CPHA 的设定	驱动的第一个数据	驱动的其它数据	采样的数据
CPOL=0, CPHA=0	在第一个 SCK 上升沿之前	SCK 下降沿	SCK 上升沿
CPOL=0, CPHA=1	第一个 SCK 上升沿	SCK 上升沿	SCK 下降沿
CPOL=1, CPHA=0	在第一个 SCK 下降沿之前	SCK 上升沿	SCK 下降沿
CPOL=1, CPHA=1	第一个 SCK 下降沿	SCK 下降沿	SCK 上升沿

8 位传输起始和停止时间的定义取决于器件为主机还是从机，以及 CPHA 变量的设定。

当器件为主机时，传输的起始由包含发送数据字节的主机来指示。此时，主机可激活时钟并开始传输。当传输的最后一个时钟周期结束时，传输结束。

当器件为从机并且 CPHA=0 时，传输在 SSEL 信号激活时开始，并在 SSEL 变为无效时结束。当器件为从机且 CPHA=1 时，如果该器件被选择，传输从第一个时钟沿开始，并在数据采样的最后一个时钟沿结束。

12.2.3 概述

有 4 个寄存器控制 SPI 外设。它们在 12.4 节“寄存器描述”中详细讲述。

SPI 控制寄存器包含一些可编程位来控制 SPI 模块的功能。该寄存器必须在给定的数据传输发生之前进行设定。

SPI 状态寄存器包含只读位，用于监视 SPI 接口的状态，包括一般性功能和异常状况。该寄存器的主要用途是检测数据传输的完成，这通过 SPIF 位来实现。其它位用于指示异常状况。异常情况将在后面描述。

SPI 数据寄存器用于提供发送和接收的数据字节。串行数据实际的发送和接收通过 SPI 模块逻辑的内部移位寄存器来实现。在发送时向 SPI 数据寄存器写入数据。数据寄存器和内部移位寄存器之间没有缓冲区。写数据寄存器会使数据直接进入内部移位寄存器。因此数据只能在没有执行数据发送时写入该寄存器。读数据带有缓冲区。当传输结束时，接收到的数据转移到一个单字节的数据缓冲区，下次传输时将其读出。读 SPI 数据寄存器将返回读数据缓冲区的值。

当 SPI 模块处于主模式时，SPI 时钟计数器寄存器用于控制时钟速率。SPI 模块为主模式时该寄存器必须在数据传输之前设定。当 SPI 模块处于从模式时，该寄存器无效。

SPI 所使用的 I/O 口为标准 CMOS I/O 口。设计上没有实现开漏 SPI 选项。当器件设置为从机时，它的 I/O 口只有在被有效的 SSEL 信号选择时才有效。

12.2.4 主机操作

下面的步骤描述了 SPI 设置为主机时如何处理数据传输。该处理假设任何之前的数据传输已经结束。

1. 将 SPI 时钟计数寄存器设置为所需要的值。
2. 将 SPI 控制寄存器设置为所需要的设定。
3. 将要发送的数据写入 SPI 数据寄存器。此写操作启动 SPI 数据传输。
4. 等待 SPI 状态寄存器中的 SPIF 位置位。SPIF 位将在 SPI 数据传输的最后一个周期之后置位。
5. 读取 SPI 状态寄存器。
6. 从 SPI 数据寄存器中读出接收到的数据（可选）。
7. 如果有更多数据需要发送，则跳到第 3 步。

注：读或写 SPI 数据寄存器用来清零 SPIF 状态位。因此，如果不执行可选的 SPI 数据寄存器读操作，则需要执行该寄存器的写操作以清零 SPIF 状态位。

12.2.5 从机操作

下面的步骤描述了 SPI 设置为从机时如何处理数据传输。该处理假设任何之前的数据传输已经结束。要求驱动 SPI 逻辑的系统时钟速度至少 8 倍于 SPI。

1. 将 SPI 控制寄存器设置为所需要的设定。
2. 将要发送的数据写入 SPI 数据寄存器（可选）。注意这只能在从 SPI 传输没有进行时执行。

3. 等待 SPI 状态寄存器中的 SPIF 位置位。SPIF 位将在 SPI 数据传输的最后一个采样时钟沿后置位。
4. 读取 SPI 状态寄存器。
5. 从 SPI 数据寄存器中读出接收到的数据（可选）。
6. 如果有更多数据需要发送，则跳到第 2 步。

注：读或写 SPI 数据寄存器用来清零 SPIF 状态位。因此至少需要执行一个该寄存器的读或写操作来清零 SPIF 状态位。

12.2.6 异常状况

12.2.7 读溢出

当 SPI 模块内部读缓冲区包含没有被处理器读出的数据，而新的传输已经完成时，那么就会发生读溢出。SPIF 位在状态寄存器中有效则表示读缓冲区包含了有效数据。当一次传输结束时，SPI 模块需要将接收到的数据移到读缓冲区。如果 SPIF 位置位 (读缓冲区已满)，新接收到的数据将会丢失，而状态寄存器的读溢出 (ROVR)位将置位。

12.2.8 写冲突

我们在前面提到过，在 SPI 总线接口与内部移位寄存器之间没有写缓冲区。这样在 SPI 数据传输过程当中不应向 SPI 数据寄存器写入数据。不能向 SPI 数据寄存器写入数据的时间帧从传输启动时开始，直到 SPIF 置位时读取状态寄存器为止。如果在这段时间帧内写 SPI 数据寄存器，写入的数据将会丢失，状态寄存器中的写冲突位（WCOL）置位。

12.2.9 模式错误

SSEL 信号在 SPI 模块为主机时必须无效。当 SPI 模块为主机时，如果 SSEL 信号被激活，表示有另外一个主机将该器件选择为从机。这种状态称为模式错误。当检测到一个模式错误时，状态寄存器的模式错误位（MODF）位置位，SPI 信号驱动器关闭，而 SPI 模式转换为从模式。

12.2.10 从机中止

如果 SSEL 信号在传输结束之前变为无效，从传输将被认为中止。此时，正在处理的发送或接收数据都将丢失，状态寄存器的从机中止（ABRT）位置位。

12.3 管脚描述

表 139 SPI 管脚描述

管脚名称	类型	描述
SCK0	输入/输出	**串行时钟** 用于同步 SPI 接口间数据传输的时钟信号。该时钟总是由主机驱动并且从机接收。时钟可编程为高有效或低有效。它只在数据传输时才被激活，其它任何时候都处于非激活状态或三态。
SSEL0	输入	**从机选择** SPI 从机选择信号是一个低有效信号，用于指示被选择参与数据传输的从机。每个从机都有各自特定的从机选择输入信号。在数据处理之前，SSEL 必须为低电平并在整个处理过程中保持低电平。如果在数据传输中 SSEL 信号变为高电平，传输将被中止。这种情况下，从机返回到空闲状态并将任何接收到的数据丢弃。对于这样的异常没有其它的指示。该信号不直接由主机驱动。可通过软件使用一个通用 I/O 口来驱动。 **当 SPI0 接口仅用于主机模式下时，LPC2101/02/03（不像早期的 Philips ARM 器件）的 SSEL0 脚可用于不同的功能。例如，SSEL0 的管脚功能可配置为一个输出数字 GPIO 脚并用于选择匹配输出的其中一个。**
MISO0	输入/输出	**主入从出** MISO 信号是一个单向的信号，它将数据从从机传输到主机。当器件为从机时，串行数据从该端口输出。当器件为主机时，串行数据从该端口输入。当从机没有被选择时，将该信号驱动为高阻态。
MOSI0	输入/输出	**主出从入** MOSI 信号是一个单向的信号，它将数据从主机传输到从机。当器件为主机时，串行数据从该端口输出。当器件为从机时，串行数据从该端口输入。

12.4 寄存器描述

SPI 包含 5 个寄存器，见表 140。所有寄存器都可以字节、半字和字的形式访问。

表 140 SPI 寄存器映射

名称	描述	访问	复位值[1]	地址
S0SPCR	SPI 控制寄存器。该寄存器控制 SPI 的操作。	R/W	0x00	0xE0020000
S0SPSR	SPI 状态寄存器。该寄存器显示 SPI 的状态。	RO	0x00	0xE0020004
S0SPDR	SPI 数据寄存器。该双向寄存器为 SPI 提供发送和接收的数据。发送数据通过写该寄存器提供。SPI0 接收的数据可从该寄存器读出。	R/W	0x00	0xE0020008
S0SPCCR	SPI 时钟计数寄存器。 该寄存器控制主机 SCK0 的频率。	R/W	0x00	0xE002000C
S0SPINT	SPI 中断标志寄存器。该寄存器包含 SPI 接口的中断标志。	R/W	0x00	0xE002001C

[1] 复位值仅指已使用位中保存的数据，不包括保留位的内容。

12.4.1 SPI 控制寄存器（S0SPCR - 0xE002 0000）

S0SPCR 寄存器根据每个配置位的设定来控制 SPI0 的操作。

表 141 SPI 控制寄存器（S0SPCR – 地址 0xE002 0000）位描述

位	符号	值	描述	复位值
1:0	-		保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
2	BitEnable	0	SPI 控制器每次传输发送和接收数据的 8 位。	0
3	CPHA		时钟相位控制决定 SPI 传输时数据和时钟的关系并控制从机传输的起始和结束。	0
		0	数据在 SCK 的第一个时钟沿采样。传输从 SSEL 信号激活时开始，并在 SSEL 信号无效时结束。	
		1	数据在 SCK 的第二个时钟沿采样。当 SSEL 信号激活时，传输从第一个时钟沿开始并在最后一个采样时钟沿结束。	
4	CPOL		时钟极性控制。	0
		0	SCK 为高有效。	
		1	SCK 为低有效。	
5	MSTR		主模式选择。	0
		0	SPI 处于从模式。	
		1	SPI 处于主模式。	
6	LSBF		LSBF 用来控制传输的每个字节的移动方向。	0
		0	SPI 数据传输 MSB (位 7)在先。	
		1	SPI 数据传输 LSB (位 0)在先。	
7	SPIE		串行外设中断使能。	0
		0	SPI 中断被禁止。	
		1	每次 SPIF 或 MODF 置位时都会产生硬件中断。	
11:8	BITS		当该寄存器的位 2 为 1 时，该字段控制每次传输的位数：	0000
		1000	每次传输 8 位	
		1001	每次传输 9 位	
		1010	每次传输 10 位	
		1011	每次传输 11 位	
		1100	每次传输 12 位	
		1101	每次传输 13 位	
		1110	每次传输 14 位	
		1111	每次传输 15 位	
		0000	每次传输 16 位	
15:12	-		保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

12.4.2 SPI 状态寄存器（S0SPSR - 0xE002 0004）

S0SPSR 寄存器根据配置位的设定来控制 SPI0 的操作。

表 142 SPI 状态寄存器（S0SPSR - 0xE002 0004）位描述

位	符号	描述	复位值
2:0	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
3	ABRT	从机中止。该位为 1 时表示发生了从机中止。当读取该寄存器时，该位清零。	0
4	MODF	模式错误。为 1 时表示发生了模式错误。先通过读取该寄存器清零 MODF 位，再写 SPI0 控制寄存器。	0
5	ROVR	读溢出。为 1 时表示发生了读溢出。当读取该寄存器时，该位清零。	0
6	WCOL	写冲突。为 1 时表示发生了写冲突。先通过读取该寄存器清零 WCOL 位，再访问 SPI 数据寄存器。	0
7	SPIF	SPI 传输完成标志。为 1 时表示一次 SPI 数据传输完成。在主模式下，该位在传输的最后一个周期置位。在从机模式下，该位在 SCK 的最后一个数据采样边沿置位。当第一次读取该寄存器时，该位清零。然后才能访问 SPI 数据寄存器。 **注**：SPIF 不是 SPI 中断标志。中断标志位于 SPINT 寄存器中。	0

12.4.3 SPI 数据寄存器（S0SPDR - 0xE002 0008）

双向数据寄存器为 SPI 提供数据的发送和接收。发送数据通过将数据写入该寄存器来实现。SPI 接收的数据可从该寄存器中读出。处于主模式时，写该寄存器将启动 SPI 数据传输。从数据传输开始到 SPIF 状态位置位并且还没有读取状态寄存器的这段时间内不能对该寄存器执行写操作。

表 143 SPI 数据寄存器（S0SPDR – 地址 0xE002 0008）位描述

位	符号	描述	复位值
7:0	DataLow	SPI 双向数据口。	0x00
15:8	DataHigh	如果 SPCR 的位 2 为 1 且位 11:8 为除 1000 以外的数，那么这些位中的某些或全部位包含额外的发送和接收位。当 16 位没有全部被选择时，最高位读为 0。	0x00

12.4.4 SPI 时钟计数寄存器（S0SPCCR - 0xE002 000C）

该寄存器控制主机 SCK 的频率。寄存器指示构成一个 SPI 时钟的 PCLK 周期的数目。该寄存器的值必须为偶数。因此 bit0 必须为 0。该寄存器的值还必须大于等于 8。如果寄存器的值不符合上述条件，可能导致产生不可预测的操作。

表 144 SPI 时钟计数寄存器（S0SPCCR – 地址 0xE002 000C）位描述

位	符号	描述	复位值
7:0	Counter	SPI0 时钟计数值设定	0x00

SPI0 速率可以这样进行计算：PCLK/SPCCR0 值。PCLK 速率为 CCLK/APB 除数，由 APBDIV 寄存器的内容决定。

12.4.5 SPI 中断寄存器（S0SPINT - 0xE002 001C）

该寄存器包含 SPI0 接口的中断标志。

表 145 SPI 中断寄存器（S0SPINT – 地址 0xE002 001C）位描述

位	符号	描述	复位值
0	SPI 中断标志	SPI 中断标志。由 SPI 接口置位以产生中断。向该位写入 1 清零。 **注：**当 SPIE=1 并且 SPIF 和 WCOL 位中至少有一位为 1 时该位置位。但是，只有当 SPI 中断位置位并且 SPI0 中断在 VIC 中被使能，SPI 中断才能由中断处理软件处理。	0
7:1	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

12.5 结构

SPI0 接口中的 SPI 方框图见图 40。

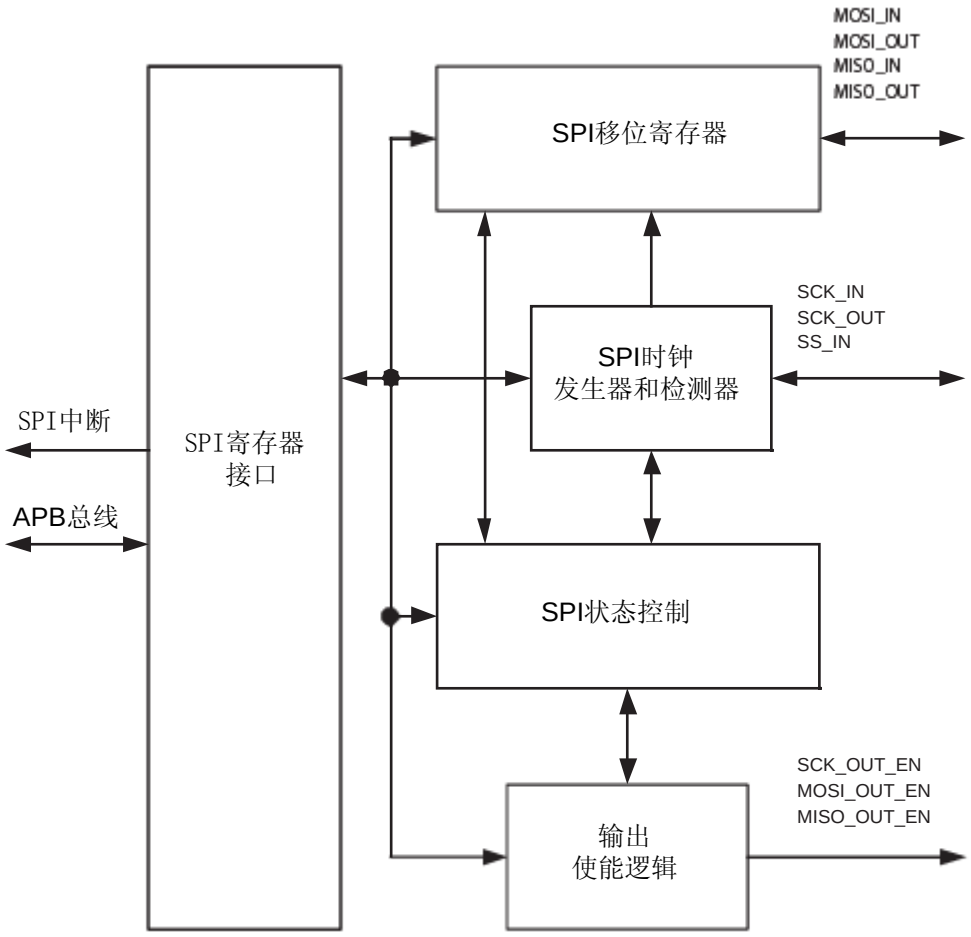


图 40 SPI 方框图

第13章 SSP 控制器(SPI1)

13.1 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
- 主机或从机操作
- 8 帧收发 FIFO
- 每帧 4～16 位

13.2 描述

SSP 是一个同步串行端口控制器，可控制 SPI、4 线 SSI 或 Microwire 总线的操作。它可处理总线上多个主机和从机的相互作用。在一定数据传输过程中，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的，4～16 位帧的数据由主机发送到从机或由从机发送到主机。但实际上，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。

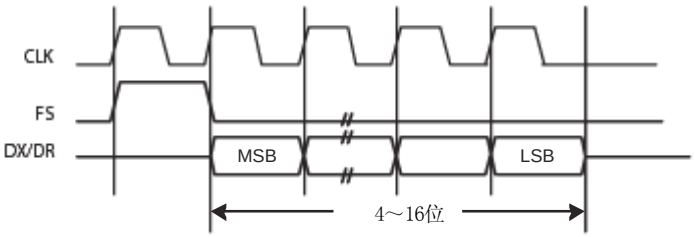
表 146 SSP 管脚描述

管脚名称	类型	接口管脚名称/功能			管脚描述
		SPI	SSI	Microwire	
SCK1	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。SCK1 的状态只能在数据传输过程中改变，在任何其它时间里，SSP 使其保持无效状态或不驱动它（使其处于高阻态）。
SSEL1	I/O	SSEL	FS	CS	**从机选择/帧同步/芯片选择。**当 SSP 是总线主机时，它在串行数据启动前或串行数据传输终止后立刻驱动该信号来指示一次所选总线和模式的合适数据传输。当 SSP 是总线从机时，该信号用来根据使用的通信协议来限制主机数据的去向。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。当总线上有多个从机时，必须要进一步限制这些从机的帧选择/从机选择输入，以防一次传输有多个从机响应。
MISO1	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 SSP 是从机时，串行数据从该信号输出。当 SSP 是主机时，该信号输出串行数据的时钟。当 SSP 是从机且未被 SSEL 选择时，它将不驱动该信号（使其处于高阻态）。
MOSI1	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 SSP 是主机时，串行数据从该信号输出。当 SSP 是从机时，该信号输出串行数据的时钟。

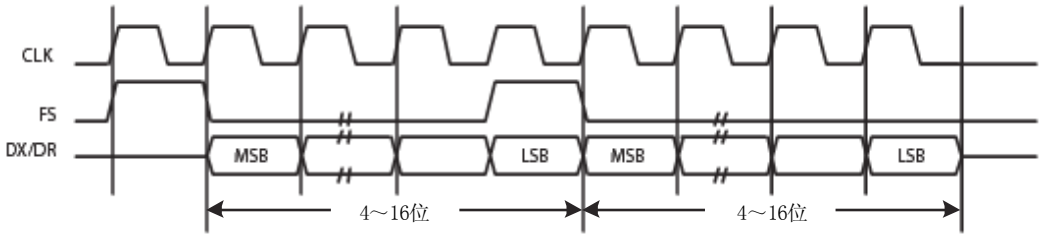
13.3 总线描述

13.3.1 Texas 仪器同步串行(SSI)数据帧格式

图 41 给出了 SSP 模块支持的 4 线 Texas 仪器同步串行数据帧的格式。



a. 单个帧传输



b. 连续/连续两帧传输

图 41 Texas 仪器同步串行数据帧的格式：

a) 单帧传输 b) 连续/连续两帧的传输

假设器件在该模式下配置为主机，当 SSP 处于空闲模式时，CLK 和 FS 强制为低，发送数据线 DX 为三态。一旦发送 FIFO 的底端入口包含数据，FS 将在一个 CLK 周期内保持高电平。被发送的数据从发送 FIFO 传输到发送逻辑的串行移位寄存器。在 CLK 的下个上升沿，4～16 位数据帧的 MSB 从 DX 管脚移出。同样地，接收到的 MSB 从片外串行从器件的 DR 管脚移入。

然后，SSP 和片外串行从器件在每个 CLK 下降沿的控制下将每一个数据位送入相应的串行移位器。LSB 被锁存后，接收到的数据在 CLK 的首个上升沿从串行移位器传递到接收 FIFO。

13.3.2 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。

13.3.3 时钟极性（CPOL）和时钟相位（CPHA）控制

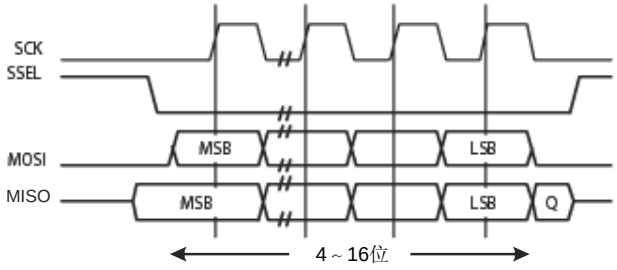
当 CPOL 时钟极性控制位为 0 时，它将在 SCK 管脚上产生一个稳定的低电平。如果 CPOL 时钟极性控制位为 1，不发生数据传输时 CLK 管脚上将出现一个稳定的高电平。

CPHA 控制位用来选择捕获数据的时钟沿，这个边沿的状态允许改变。它将对第一个数据捕获沿出现前由于允许或不允许时钟跳变而发送的第一个位产生最重大的影响。当 CPHA

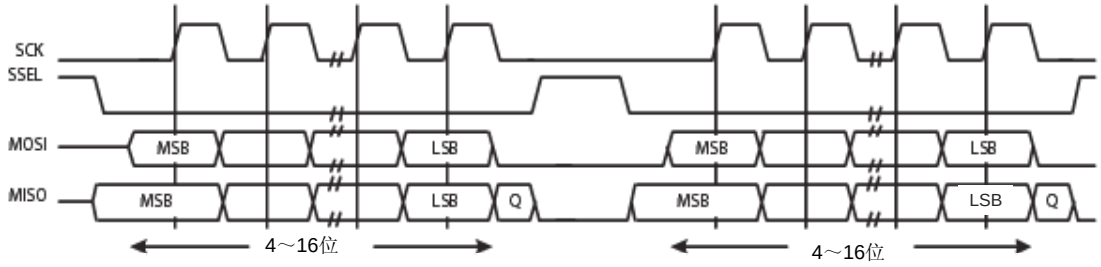
相位控制位为 0 时，数据在第 1 个时钟沿跳变时被捕获。如果 CPHA 相位控制位为 1，数据在第 2 个时钟沿跳变时被捕获。

13.3.4 CPOL=0, CPHA=0 时的 SPI 格式

CPOL=0，CPHA=0 时 SPI 格式的单帧传输和连续传输信号时序如图 42 所示。



a. CPOL=0和CPHA=0时的单帧传输



b. CPOL=0和CPHA=0时的连续传输

图 42 CPOL＝0 和 CPHA＝0 时的 SPI 格式 （a）单帧 和 b）连续传输）

这种配置在空闲期间：

- CLK 信号强制为低
- SSEL 强制为高
- 发送 MOSI/MISO 处于高阻态

如果 SSP 使能并且发送 FIFO 中含有有效数据，则 SSEL 主机信号被驱动为低表示开始发送数据。这就使得从机数据使能到主机的 MISO 输入线上。主机的 MOSI 被使能。

1/2 个 SCK 周期后，有效主机数据被传输到 MOSI 管脚。由于主机和从机数据都被设置，再过 1/2 个 SCK 周期后 SCK 主机时钟管脚将变高。

此时，数据在 SCK 信号的上升沿被捕获，保持到 SCK 的下降沿。

在发送单个字时，当数据字的所有位发送完后，最后一位被捕获后的一个 SCK 周期内，SSEL 返回到空闲的高电平状态。

但是，在连续顺序的发送过程中，在每个数据字传输之间 SSEL 信号必须为高。这是因为当 CPHA 位为逻辑 0 时，从机选择管脚冻结了串行外围寄存器中的数据，不允许改变。因此，在每次数据传输之间主器件必须拉高从器件的 SSEL 管脚来使能串行外设数据的写操作。当连续传输结束后，最后一位被捕获后一个 SCK 周期内，SSEL 返回到空闲状态。

13.3.5 CPOL=0，CPHA=1 时的 SPI 格式

CPOL=0，CPHA=1 时 SPI 格式的传输信号时序如图 43 所示，它包括单帧传输和连续传输两种方式。

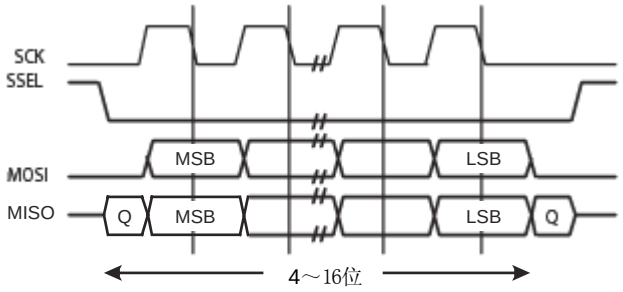


图 43 CPOL＝0 和 CPHA=1 时的 SPI 帧格式（单帧传输）

这种配置在空闲期间：

- CLK 信号强制为低
- SSEL 强制为高
- 发送 MOSI/MISO 管脚处于高阻态

如果 SSP 使能并且发送 FIFO 中含有有效数据，则 SSEL 主机信号被驱动为低表示开始发送数据。主机的 MOSI 管脚使能。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，SCK 出现上升沿跳变。

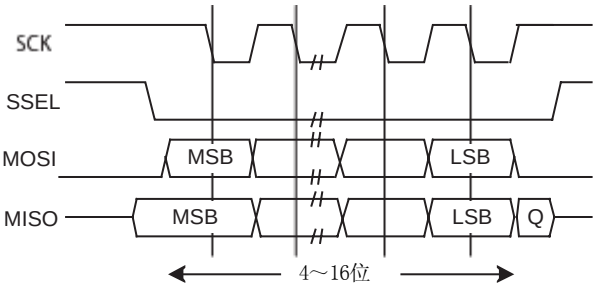
然后，数据在 SCK 信号的下降沿被捕获并保持到 SCK 信号的上升沿。

在单个字的传输过程中，当所有位传输结束后，最后一位被捕获后一个 SCK 周期内，SSEL 线返回到空闲的高电平状态。

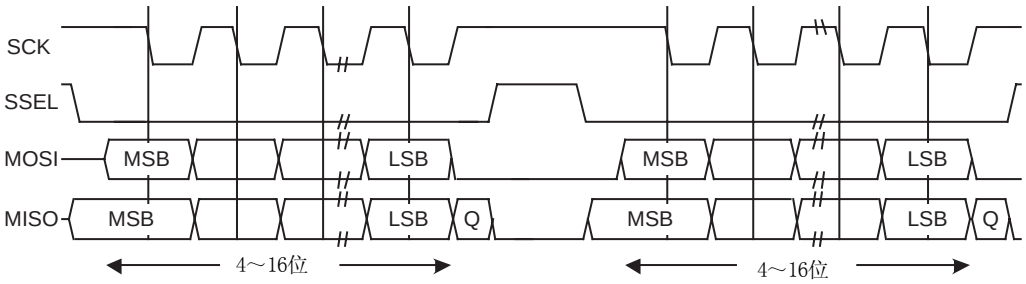
对于连续的顺序传输，SSEL 在两个连续的数据字传输之间保持低电平，终止传输的方法与单个字传输相同。

13.3.6 CPOL=1, CPHA=0 时的 SPI 格式

CPOL=1，CPHA=0 时 SPI 格式的单帧和连续传输信号时序如图 44 所示。



a) CPOL=1和CPHA=0时的单帧传输



b) CPOL=1和CPHA=0时的连续传输

图 44 CPOL=1 和 CPHA=0 时的 SPI 帧格式（a）单帧和 b) 连续传输）

这种配置在空闲期间：

- CLK 信号强制为高
- SSEL 强制为高
- 发送 MOSI/MISO 管脚处于高阻态

如果 SSP 使能并且发送 FIFO 中含有有效数据，则 SSEL 主机信号被驱动为低表示开始发送数据，使得从机数据立即传输到主机的 MISO 线上。主机的 MOSI 管脚被使能。

1/2 个 SCK 周期后，有效主机数据被传输到 MOSI 线。由于主机和从机数据都被设置，再过 1/2 个 SCK 周期后 SCK 主机时钟管脚将变低。这就意味着数据在 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，当数据字的所有位发送完后，最后一位被捕获后一个 SCK 周期内，SSEL 线返回到空闲的高电平状态。

但是，在连续顺序的发送过程中，在每个数据字传输之间 SSEL 信号必须为高。这是因为当 CPHA 位为逻辑 0 时，从机选择管脚冻结了串行外围寄存器中的数据，不允许改变。因此，在每次数据传输之间主器件必须拉高从器件的 SSEL 管脚来使能串行外设数据的写操作。当连续传输结束后，最后一位被捕获后一个 SCK 周期内，SSEL 返回到空闲状态。

13.3.7 CPOL=1，CPHA=1 时的 SPI 格式

CPOL=1，CPHA=1 时 SPI 格式的传输信号时序如图 45 所示，它包含单帧传输和连续传输两种方式。

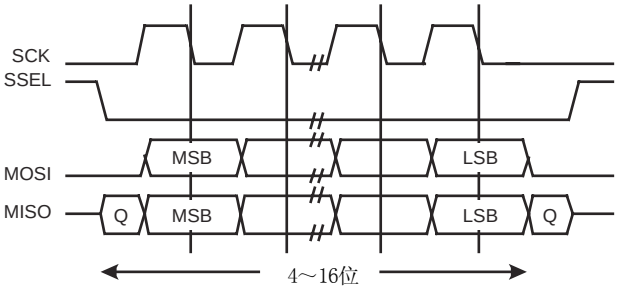


图 45 CPOL=1 和 CPHA=1 时的 SPI 帧格式（单帧传输）

这种配置在空闲期间：

- CLK 信号强制为高
- SSEL 强制为高
- 发送 MOSI/MISO 管脚处于高阻态

如果 SSP 使能并且发送 FIFO 中含有有效数据，则 SSEL 主机信号被驱动为低表示开始发送数据。主机的 MOSI 管脚使能。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，SCK 出现上升沿跳变使能。然后，数据在 SCK 信号的上升沿被捕获并保持到 SCK 信号的下降沿。

在单个字的传输过程中，当所有位传输结束后，最后一位被捕获后一个 SCK 周期内，SSEL 返回到空闲的高电平状态。对于连续的顺序传输，SSEL 管脚仍然保持有效的低电平状态，直到最后一个字的最后一位捕获结束后，它再返回到上述的空闲状态。总的说来，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。

13.3.8 半导体 Microwire 帧格式

图 46 所示为 Microwire 帧格式的单帧传输。图 47 所示为 Microwire 帧格式的连续帧传输。

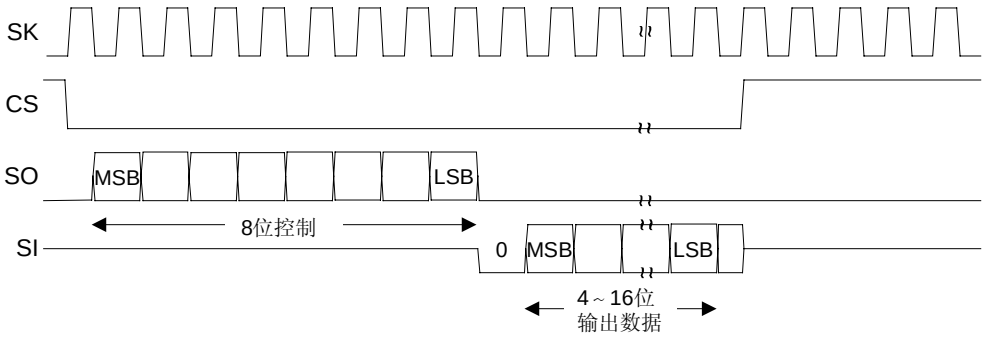


图 46 Microwire 帧格式（单帧传输）

Microwire 格式与 SPI 格式类似，但它的发送是半双工而非全双工模式，数据从主机传输到从机。每次串行发送以 SSP 传输到片外从器件的一个 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，片外从器件对其进行译码，在 8 位控制信息的最后一位发送完的一个串行时钟后，才响应到来的所需数据。返回的数据长度为 4～16 位，使得总的数据帧长度在 13～25 位之间。

这种配置在空闲期间：

- SK 信号强制为低
- CS 强制为高
- 发送数据线 SO 可强制为低

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 底端入口的数据传输到发送逻辑的串行移位寄存器，8 位控制帧的 MSB 移出到 SO 管脚。CS 在帧发送过程中保持低电平。SI 在帧发送过程中保持三态。

片外串行从器件在每个 SK 的上升沿将每个控制位锁存到串行移位器。当从器件完成最后一位的锁存后，一个时钟等待周期内对控制字节进行译码，然后从器件通过将数据发回给 SSP 来响应。每一位在 SK 的下降沿驱动到 SI 上。SSP 又在 SK 的上升沿将每一位锁存。在帧传输结束后，对于单帧传输，在最后一位被锁存到接收串行移位器后的一个时钟周期内 CS 信号被拉高，使数据传输到接收 FIFO。

注：在 LSB 被接收移位器锁存后或当 CS 变为高电平时，片外从器件的接收线在任何一个 SK 的下降沿都呈现三态。

连续传输过程的数据发送开始和结束的方法都与单帧传输相同。所不同的是，在连续传输过程中，CS 持续有效（保持低电平），数据连续发送。当前数据帧的 LSB 被接收后，下一帧的控制字节立刻直接发送。在一帧数据的 LSB 被锁存到 SSP 后，每个接收到的数据在 SK 的下降沿传送到接收移位器。

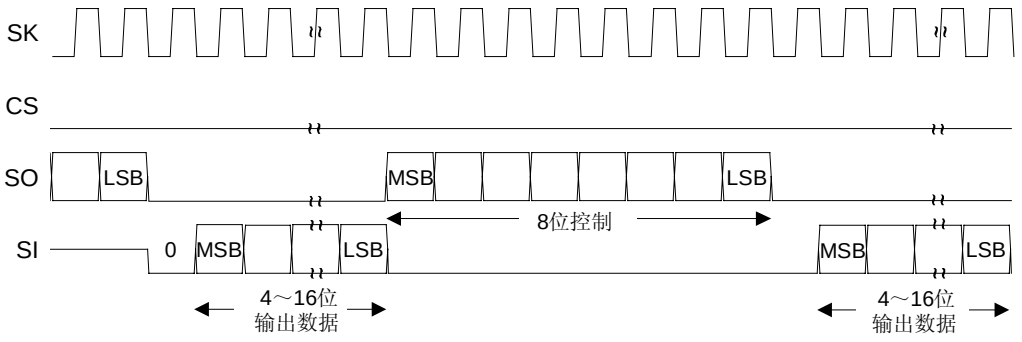


图 47 Microwire 帧格式（连续传输）

13.3.9 Microwire 模式中 CS 相对 SK 的建立和保持时间

Microwire 模式中，当 CS 变低后，SSP 从机在 SK 的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 的上升沿有足够的建立和保持时间。

图 48 给出了建立和保持时间的要求。相对 SSP 从机采样接收数据的首个位的 SK 上升沿，CS 的建立时间至少为 SSP 的 SK 周期的 2 倍。相对于之前的 SK 上升沿，CS 的保持时间至少为一个 SK 周期。

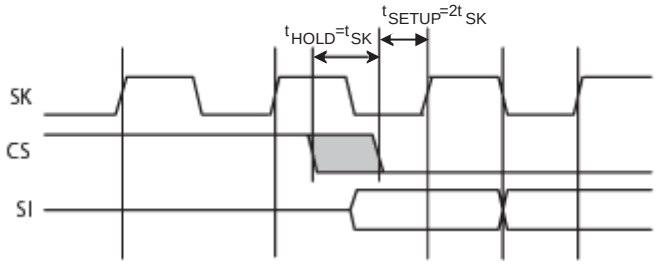


图 48 Microwire 帧格式（连续传输）—详细描述

13.4 寄存器描述

SSP 包括表 147 中给出的 9 个寄存器。所有寄存器可进行字节、半字和字访问。

表 147 SSP 寄存器映射

名称	描述	访问	复位值[1]	地址
SSPCR0	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	0x0000	0xE006 8000
SSPCR1	控制寄存器 1。选择主机/从机和其它模式。	R/W	0x00	0xE006 8004
SSPDR	数据寄存器。写满发送 FIFO 和读空接收 FIFO。	R/W	0x0000	0xE006 8008
SSPSR	状态寄存器	RO	0x03	0xE006 800C
SSPCPSR	时钟预分频寄存器。	R/W	0x00	0xE006 8010
SSPIMSC	中断屏蔽设置和清零寄存器。	R/W	0x00	0xE006 8014
SSPRIS	原始中断状态寄存器。	R/W	0x04	0xE006 8018
SSPMIS	屏蔽中断状态寄存器。	RO	0x00	0xE006 801C
SSPICR	SSPICR 中断清零寄存器。	WO	NA	0xE006 8020

[1] 复位值仅指已使用位中保存的数据，不包括保留位的内容。

13.4.1 SSP 控制寄存器 0 (SSPCR0-0xE006 8000)

该寄存器控制着 SSP 控制器的基本操作。

表 148 SSP 控制寄存器 0（SSPCR0 – 地址 0xE006 8000）位描述

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	SSI	
		10	Microwire	
		11	不支持且不应使用这个组合。	
6	CPOL		时钟输出极性。该位只用在 SPI 模式。	0
		0	SSP 控制器在帧传输的第一个时钟跳变上升沿捕获串行数据，即跳变**远离**时钟线在帧内部的状态。	
		1	SSP 控制器在帧传输的第二个时钟跳变沿捕获串行数据，即跳变**后退**到时钟线在帧内部的状态。	
7	CPHA		时钟输出相位。该位只用在 SPI 模式。	0
		0	SSP 控制器使总线时钟在每帧传输之间保持低电平。	
		1	SSP 控制器使总线时钟在每帧传输之间保持高电平。	
15:8	SCR		串行时钟速率。其值为总线上每位的预分频器输出时钟数减 1。假设 CPSDVR 为预分频器分频值，APB 时钟 PCLK 为预分频器时钟，则位频率为 PCLK / (CPSDVSR*[SCR+1])。	0x00

13.4.2 SSP 控制寄存器 1（SSPCR1 – 0xE006 8004）

该寄存器控制着 SSP 控制器的工作方式。

表 149 SSP 控制寄存器 1（SSPCR1 – 地址 0xE006 8004）位描述

位	符号	值	描述	复位值
0	LBM		回写模式。	0
		0	正常工作模式。	
		1	串行输入脚可用作串行输出脚（MOSI 或 MISO），而不是仅用作串行输入脚（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器件相互通信。在置位该位前，软件应将合适的控制信息写入其它 SSP 寄存器和中断控制器寄存器。	
2	MS		主机/从机模式。该位只能在 SSE 位为 0 时写入。	0
		0	SSP 控制器用作总线主机，驱动 SCLK、MOSI 和 SSEL 并接收 MISO 线。	
		1	SSP 控制器用作总线从机，驱动 MISO 线和接收 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
7:4	-		保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

13.4.3 SSP 数据寄存器（SSPDR – 0xE006 8008）

软件可将要发送的数据写入该寄存器，或从该寄存器读出接收到的数据。

表 150 SSP 数据寄存器（SSPDR – 地址 0xE006 8008）位描述

位	符号	描述	复位值
15:0	DATA	**写：**当状态寄存器的 TNF 位为 1 指示 Tx FIFO 未满时，软件可将要发送的帧的数据写入该寄存器。如果 Tx FIFO 以前为空且 SSP 控制器空闲，则立刻开始发送数据。否则，写入该寄存器的数据要等到所有数据发送（或接收）完后才能发送。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：**当状态寄存器的 RNE 位为 1 指示 Rx FIFO 不为空时，软件可读取该寄存器。软件读取该寄存器时，SSP 控制器将返回从 Rx FIFO 中读走的最后一个数据。如果数据长度小于 16 位，该字段的数据必须进行调整，低位对齐，高位补零。	0x0000

13.4.4 SSP 状态寄存器（SSPSR – 0xE006 800C）

该只读寄存器反映了 SSP 控制器的当前状态。

表 151 SSP 状态寄存器（SSPSR – 地址 0xE006 800C）位描述

位	名称	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 未满。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。SSP 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 Tx FIFO 不为空时该位为 1。	0
7:5	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

13.4.5 SSP 时钟预分频寄存器（SSPCPSR – 0xE006 8010）

该寄存器控制着预分频器分频 APB 时钟 PCLK 得到预分频器时钟的因子，反过来，预分频时钟被 SSPCR0 中的 SCR 分频后得到位时钟。

表 152 SSP 时钟预分频寄存器（SSPCPSR – 地址 0xE006 8010）位描述

位	名称	描述	复位值
7:0	CPSDVSR	这是一个 2～254 中的一个偶数。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

重要：必须适当地初始化 SSPCPSR 值，否则 SSP 控制器将不能正确发送数据。如果 SSP 在主控器模式下操作则 $CPSDVSR_{min}$＝2，而如果 SSP 在从机模式下操作则 $CPSDVSR_{min}$＝12。

13.4.6 SSP 中断屏蔽设置/清除寄存器（SSPIMSC – 0xE006 8014）

该寄存器控制着 SSP 控制器 4 个中断条件的使能。注意：ARM 使用“masked”一词时的理解与经典计算机术语相反，计算机术语中“masked”意思是“禁能”。而 ARM 中“masked”的意思是“使能“。为了防止混淆，ARM 文档中通常不用“masked”一词。

表 153 SSP 中断屏蔽设置/清除寄存器(SSPIMSC –地址 0xE0068014)位描述

位	名称	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 满时又完成另一个帧的接收时该位置位。ARM 特别指出，接收溢出时新的数据帧会将前面的数据帧覆盖。	0
1	RTIM	当出现接收超时条件时，软件置位该位来使能中断。当 Rx FIFO 不为空且在 32 个位时间内既未接收新数据又未从 FIFO 中读出数据时，产生接收超时。	0
2	RXIM	当 Rx FIFO 至少有一半为满时，软件置位该位来使能中断。	0
3	TXIM	当 Tx FIFO 至少有一半为空时，软件置位该位来使能中断。	0
7:4	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

13.4.7 SSP 原始中断状态寄存器（SSPRIS – 0xE006 8018）

当一个中断条件出现时，该只读寄存器中对应的位置位，与中断是否通过 SSPIMSC 使能无关。

表 154 SSP 原始中断状态寄存器（SSPRIS – 地址 0xE006 8018）位描述

位	名称	描述	复位值
0	RORRIS	当 Rx FIFO 满时又接收到另一帧数据时该位置位。ARM 特别指出，此时接收到的新数据帧会将前面的数据帧覆盖。	0
1	RTRIS	当接收超时条件出现时该位置位。注意：如果要接收更多的数据，接收超时可撤除。	0
2	RXRIS	当 Rx FIFO 至少一半为满时该位置位。	0
3	TXRIS	当 Tx FIFO 至少一半为空时该位置位。	1
7:4	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

13.4.8 SSP 屏蔽中断状态寄存器（SSPMIS – 0xE006 801C）

当一个中断条件出现且相应的中断在 SSPIMSC 中被使能时，对应在该只读寄存器的位置位。当产生 SSP 中断时，中断服务程序可通过读该寄存器来判断中断源。

表 155 SSP 屏蔽中断状态寄存器（SSPMIS – 地址 0xE006 801C）位描述

位	名称	描述	复位值
0	RORMIS	当 Rx FIFO 满时又接收到另一帧数据，并且中断被使能时该位置位。	0
1	RTMIS	当接收超时条件出现且中断被使能时该位置位。注意：如果要接收更多的数据，接收超时可撤除。	0
2	RXMIS	当 Rx FIFO 至少一半为满且中断被使能时该位置位。	0
3	TXMIS	当 Tx FIFO 至少一半为空且中断被使能时该位置位。	0
7:5	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

13.4.9 SSP 中断清除寄存器（SSPICR – 0xE006 8020）

软件可通过向该只写寄存器写入 1 或多个 1 来清除 SSP 控制器的相关中断。注意：另外 2 个中断条件可通过写或读相应的 FIFO 来清除，或通过清除 SSPIMSC 中对应的位来禁能。

表 156 SSP 中断清除寄存器（SSPICR – 地址 0xE006 8020）位描述

位	名称	描述	复位值
0	RORIC	向该位写 1 来清除“Rx FIFO 满时还接收到数据”的中断。	NA
1	RTIC	向该位写 1 来清除接收超时中断。	NA
7:2	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未被定义。	NA

第14章 模数转换器(ADC)

14.1 特性

- 10 位逐次逼近式模数转换器
- 掉电模式
- 测量范围：0V～$V_{DD(3V3)}$ （通常为 3V; 不超过 V_{DDA} 电压电平)
- 10 位转换时间≥2.44us
- 一个或多个输入的突发转换模式
- 可选择由输入跳变或定时器匹配信号触发转换
- 特定结果寄存器用于每个模拟输入来减少中断开销

14.2 描述

A/D 转换器的基本时钟由 APB 时钟提供。每个转换器包含一个可编程分频器，可将时钟调整至逐步逼近转换所需的 4.5MHz（最大）。完全满足精度要求的转换需要 11 个这样的时钟。

14.3 管脚描述

表 157 给出了每个 ADC 相关管脚的简单总结。

表 157 ADC 管脚描述

管脚名称	类型	管脚描述
AD0.7:0	输入	**模拟输入**。A/D 转换器单元可测量输入信号的电压。注意：这些模拟输入通常连接到管脚上，即使管脚功能选择寄存器将它们设定为端口管脚。通过将这些管脚驱动成端口输出来实现 A/D 转换器的简单自测。 **注：**当使用 A/D 转换器时，模拟输入管脚的信号电平在任何时候都不能大于 V_{3A}，否则，读出的 A/D 值无效。如果在应用中未使用 A/D 转换器，则 A/D 输入管脚用作可承受 5V 电压的数字 I/O 口。 **警告：**当 ADC 管脚指定为 5V 电压时（见表 58 “管脚描述”），ADC 模块中的模拟复用功能不可用。3.3V(V_{DDA})+10%以上的电压不应该应用到选择作为 ADC 输入的任何管脚，否则 ADC 读操作将不正确。例如，若 AD0.0 和 AD0.1 作为 ADC0 输入，AD0.0=4.5V 而 AD0.1=2.5V，AD0.0 上过量的电压会造成 AD0.1 的错误读操作，尽管 AD0.1 输入电压在正确的范围以内。
$V_{DD(3V3)}$	参考电压	**参考电压。**该管脚为 A/D 转换器提供参考电压电平。
V_{DDA},V_{SSA}	电源	**模拟电源和地**。它们分别与标称为 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。

14.4 寄存器描述

A/D 转换器寄存器如表 158 所示。

表 158 ADC 寄存器

通用名称	描述	访问	复位值[1]	AD0 地址& 名称
ADCR	A/D 控制寄存器。A/D 转换开始前，必须写入 ADCR 寄存器来选择工作模式。	R/W	0x0000 0001	0xE003 4000 AD0CR
ADGDR	A/D 全局数据寄存器。该寄存器包含 ADC 的 DONE 位和最当前 A/D 转换的结果。	R/W	NA	0xE003 4004 AD0GDR
ADSTAT	A/D 状态寄存器。该寄存器包括所有 A/D 通道的 DONE 和 OVERRUN 标志，以及 A/D 中断标志。	RO	0x0000 0000	0xE003 4030 AD0STAT
ADINTEN	A/D 中断使能寄存器。该寄存器含有使能位，这些使能位允许每个 A/D 通道的 DONE 标志是否产生 A/D 中断。	R/W	0x0000 0100	0xE003 400C AD0INTEN
ADDR0	A/D 通道 0 数据寄存器。该寄存器包括在通道 0 完成的最当前的转换结果。	RO	NA	0xE003 4010 AD0DR0
ADDR1	A/D 通道 1 数据寄存器。该寄存器包括在通道 1 完成的最当前的转换结果。	RO	NA	0xE003 4014 AD0DR1
ADDR2	A/D 通道 2 数据寄存器。该寄存器包括在通道 2 完成的最当前的转换结果。	RO	NA	0xE003 4018 AD0DR2
ADDR3	A/D 通道 3 数据寄存器。该寄存器包括在通道 3 完成的最当前的转换结果。	RO	NA	0xE003 401C AD0DR3
ADDR4	A/D 通道 4 数据寄存器。该寄存器包括在通道 4 完成的最当前的转换结果。	RO	NA	0xE003 4020 AD0DR4
ADDR5	A/D 通道 5 数据寄存器。该寄存器包括在通道 5 完成的最当前的转换结果。	RO	NA	0xE003 4024 AD0DR5
ADDR6	A/D 通道 6 数据寄存器。该寄存器包括在通道 6 完成的最当前的转换结果。	RO	NA	0xE003 4028 AD0DR6
ADDR7	A/D 通道 7 数据寄存器。该寄存器包括在通道 7 完成的最当前的转换结果。	RO	NA	0xE003 402C AD0DR7

[1] 复位值仅指已使用位中保存的数据。它不包括保留位的内容。

14.4.1 A/D 控制寄存器（AD0CR – 0xE003 4000）

表 159 A/D 控制寄存器（AD0CR –地址 0xE003 4000）位描述

位	名称	值	描述	复位值
7:0	SEL		从 AD0.7:0 管脚中选择采样和转换输入脚。对于 AD0，位 0 选择管脚 AD0.0，位 7 选择管脚 AD0.7。软件控制模式下，这些位中只有一位可被置位。硬件扫描模式下，SEL 可包含 1～8 个 1。SEL 为零时等效于为 0x01。	0x01

续上表

位	名称	值	描述	复位值
15:8	CLKDIV		将 APB 时钟（PCLK）进行（CLKDIV 的值+1）分频得到 A/D 转换时钟，该时钟必须小于或等于 4.5MHz。典型地，软件将 CLKDIV 编程为最小值来得到 4.5MHz 或稍低于 4.5MHz 的时钟，但某些情况下（例如高阻抗模拟电源）可能需要更低的时钟。	0
16	BURST	1	AD 转换器以 CLKS 字段选择的速率重复执行转换，（如果必要）并从 SEL 字段中为 1 的位对应的管脚开始扫描。A/D 转换器启动后的第一次转换的是 SEL 字段中为 1 的位中的最低有效位对应的模拟输入，然后是为 1 的更高有效位对应的模拟输入（如果可用）。重复转换通过清零该位终止，但该位被清零时正在进行的转换将会结束。 **重要**：当 BURST=1 时 START 位必须为 000，否则不启动转换。	0
		0	转换由软件控制，需要 11 个时钟方能完成。	
19:17	CLKS		该字段用来选择 Burst 模式下每次转换使用的时钟数和所得 ADDR 转换结果的 RESULT 位中可确保精度的位的数目，CLKS 可在 11 个时钟（10 位）～4 个时钟（3 位）之间选择：	000
		000	11 个时钟/10 位	
		001	10 个时钟/9 位	
		010	9 个时钟/8 位	
		011	8 个时钟/7 位	
		100	7 个时钟/6 位	
		101	6 个时钟/5 位	
		110	5 个时钟/4 位	
		111	4 个时钟/3 位。	
20	-		保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
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，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
26:24	START		当 BURST 为 0 时，这些位控制着 A/D 转换是否启动和何时启动：	0
		000	不启动（PDN 清零时使用该值）	
		001	立即启动转换	
		010	位 27 选择的边沿出现在 P0.16/EINT0/ MAT0.2 脚时启动转换	
		011	ADCR 寄存器位 27 选择的边沿出现在 P0.22 时启动转换	
		100	ADCR 寄存器位 27 选择的边沿出现在 MAT0.1 时启动转换	
		101	ADCR 寄存器位 27 选择的边沿出现在 MAT0.3 时启动转换	
		110	ADCR 寄存器位 27 选择的边沿出现在 MAT1.0 时启动转换	
		111	ADCR 寄存器位 27 选择的边沿出现在 MAT1.1 时启动转换	
27	EDGE		该位只有在 START 字段为 010～111 时有效。在这些情况下：	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，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

14.4.2 A/D 全局数据寄存器（AD0GDR–0xE003 4004）

表 160 A/D 全局数据寄存器（AD0GDR –地址 0xE003 4004）位描述

位	符号	描述	复位值
5:0	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
15:6	RESULT	当 DONE 为 1 时，该字段包含一个二进制数，用来代表 SEL 字段选中的 Ain 脚的电压。该字段根据 V_{DDA} 脚上的电压(V/V_{REF})对 Ain 脚的电压进行划分。该字段为 0 表明 Ain 脚的电压小于，等于或接近于 V_{SSA}；该字段为 0x3FF 表明 Ain 脚的电压接近于，等于或大于 V_{REF}。	NA
23:16	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
26:24	CHN	这些位包含的是 RESULT 位的转换通道（例如，000 表示通道 0，001 表示通道 1）。	NA
29:27	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
30	OVERUN	Burst 模式下，如果在转换产生 RESULT 位的结果前一个或多个转换结果被丢失和覆盖，该位置位。该位通过读 ADDR 寄存器清零。	0
31	DONE	A/D 转换结束时该位置位。该位在 ADDR 被读出和 ADCR 被写入时清零。如果 ADCR 在转换过程中被写入，该位置位，启动一次新的转换。	0

14.4.3 A/D 状态寄存器(ADSTAT, ADC0: AD0CR-0xE003 4004)

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 寄存器中出现的 DONE 和 OVERRUN 标志在 ADSTAT 中反映。中断标志（所有 DONE 标志的逻辑或）也可在 ADSTAT 中找到。

表 161 A/D 状态寄存器（ADSTAT, ADC0: AD0STAT-地址 0xE003 4004 和 ADC1: AD1STAT-地址 0xE006 0004)位描述

位	符号	描述	复位值
0	DONE0	该位反映 A/D 通道 0 结果寄存器的 DONE 状态标志。	0
1	DONE1	该位反映 A/D 通道 1 结果寄存器的 DONE 状态标志。	0
2	DONE2	该位反映 A/D 通道 2 结果寄存器的 DONE 状态标志。	0
3	DONE3	该位反映 A/D 通道 3 结果寄存器的 DONE 状态标志。	0
4	DONE4	该位反映 A/D 通道 4 结果寄存器的 DONE 状态标志。	0
5	DONE5	该位反映 A/D 通道 5 结果寄存器的 DONE 状态标志。	0
6	DONE6	该位反映 A/D 通道 6 结果寄存器的 DONE 状态标志。	0
7	DONE7	该位反映 A/D 通道 7 结果寄存器的 DONE 状态标志。	0
8	OVERRUN0	该位反映 A/D 通道 0 结果寄存器的 OVERRUN 状态标志。	0
9	OVERRUN1	该位反映 A/D 通道 1 结果寄存器的 OVERRUN 状态标志。	0
10	OVERRUN2	该位反映 A/D 通道 2 结果寄存器的 OVERRUN 状态标志。	0
11	OVERRUN3	该位反映 A/D 通道 3 结果寄存器的 OVERRUN 状态标志。	0
12	OVERRUN4	该位反映 A/D 通道 4 结果寄存器的 OVERRUN 状态标志。	0
13	OVERRUN5	该位反映 A/D 通道 5 结果寄存器的 OVERRUN 状态标志。	0
14	OVERRUN6	该位反映 A/D 通道 6 结果寄存器的 OVERRUN 状态标志。	0
15	OVERRUN7	该位反映 A/D 通道 7 结果寄存器的 OVERRUN 状态标志。	0
16	ADINT	该位是 A/D 中断标志。当任何单个 A/D 通道 DONE 标志有效且通过 ADINTEN 寄存器使能 A/D 中断时，该位为 1。	0
31:17	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

14.4.4 A/D 中断使能寄存器(ADINTEN,ADC0:AD0INTEN-0xE003 400C）

该寄存器控制转换结束时产生中断的 A/D 通道。例如，使用一些 A/D 通道来监控检测器（通过在 A/D 通道上继续进行转换来实现）。通过应用程序读最当前的结果(只要它们有需要)。此时，在某些 A/D 通道的每个转换结束不需要中断。

表 162 A/D 中断使能寄存器(ADINTEN,ADC0:AD0INTEN-地址 0xE003 400C)

位描述

位	符号	值	描述	复位值
0	ADINTEN0	0 1	ADC 通道 0 上转换的结束将不会产生中断。 ADC 通道 0 上转换的结束将产生中断。	0
1	ADINTEN1	0 1	ADC 通道 1 上转换的结束将不会产生中断。 ADC 通道 1 上转换的结束将产生中断。	0
2	ADINTEN2	0 1	ADC 通道 2 上转换的结束将不会产生中断。 ADC 通道 2 上转换的结束将产生中断。	0
3	ADINTEN3	0 1	ADC 通道 3 上转换的结束将不会产生中断。 ADC 通道 3 上转换的结束将产生中断。	0
4	ADINTEN4	0 1	ADC 通道 4 上转换的结束将不会产生中断。 ADC 通道 4 上转换的结束将产生中断。	0
5	ADINTEN5	0 1	ADC 通道 5 上转换的结束将不会产生中断。 ADC 通道 5 上转换的结束将产生中断。	0
6	ADINTEN6	0 1	ADC 通道 6 上转换的结束将不会产生中断。 ADC 通道 6 上转换的结束将产生中断。	0
7	ADINTEN7	0 1	ADC 通道 7 上转换的结束将不会产生中断。 ADC 通道 7 上转换的结束将产生中断。	0
8	ADGINTEN	0 1	只有通过 ADINTEN7:0 使能的单个 ADC 通道将产生中断。 仅使能 ADDR 中的全局 DONE 标志来产生中断。	1
31:17	-		保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

14.4.5 A/D 数据寄存器(ADDR0～ADDR7，ADC0：AD0DR0 ~AD0DR7 - 0xE003 4010～0xE003 402C)

当 A/D 转换结束时 A/D 数据寄存器保存结果，同时包括表示转换结束和转换溢出发生时的标志。

表 163 A/D 数据寄存器(ADDR0~ADDR7, ADC0: AD0DR0~AD0DR7)位描述

位	符号	描述	复位值
5:0	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
15:6	RESULT	当 DONE 为 1 时，该字段包含一个二进制数，表示 AIN 脚的电压，V_{REF} 脚(V/V_{REF})的电压对 AIN 脚的电压进行划分。该字段为 0 表明 AIN 脚的电压小于，等于或接近于 V_{SSA}；该字段为 0x3FF 表明 AIN 脚的电压接近于，等于或大于 V_{REF}。	NA
29:16	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
30	OVERUN	Burst 模式下，如果在转换产生 RESULT 位的结果前一个或多个转换结果被丢失和覆盖，该位置位。该位通过读 ADDR 寄存器清零。	
31	DONE	A/D 转换结束时该位置位。读这个寄存器时该位清零。	NA

14.5 操作

14.5.1 硬件触发转换

如果 ADCR 的 BURST 位为 0 且 START 字段的值包含在 010-111 之内，当所选管脚或定时器匹配信号发生跳变时 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 位 27 相异或所得的结果用作边沿检测逻辑。

14.5.2 中断

当 DONE 位为 1 时，中断请求声明到向量中断控制器（VIC）。软件可通过 VIC 中 A/D 转换器的中断使能位来控制是否产生中断。DONE 在 ADDR 读出时无效。

14.5.3 精度和数字接收器

为了在监测管脚上读取到精确的电压值，必须在相应的管脚选择寄存器（见 7.4 节“寄存器描述”中的“管脚连接模块”）中选择 AIN 功能。对于连接 ADC 输入的管脚，不可能具有选择的数字功能和得到有效的 ADC 读取值。只要在该管脚上选择数字功能，内部电路就断开 ADC 硬件和相关管脚的连接。

第15章 定时器/计数器 定时器 0 和定时器 1

定时器/计数器 0 和定时器/计数器 1 除了外设基地址以外，其它都相同。

15.1 特性

- 带可编程 32 位预分频器的 32 位定时器/计数器
- 计数器或定时器操作
- 具有多达 4 路（定时器 1）和 3 路（定时器 0）32 位的捕获通道。当输入信号跳变时可取得定时器的瞬时值。也可选择使捕获事件产生中断。
- 4 个 32 位匹配寄存器：

  －匹配时定时器继续工作，可选择产生中断
  －匹配时停止定时器，可选择产生中断
  －匹配时复位定时器，可选择产生中断
- 多达 4 个(定时器 1)和 3 个(定时器 0)对应于匹配寄存器的外部输出，具有下列特性：

  －匹配时设置为低电平
  －匹配时设置为高电平
  －匹配时翻转
  －匹配时无动作
- 对于每个定时器，多达 4 个匹配寄存器可配置为 PWM，允许使用多达 3 个匹配输出作为单边沿控制的 PWM 输出。

15.2 应用

- 用于对内部事件进行计数的间隔定时器
- 通过捕获输入实现脉宽调制
- 自由运行的定时器
- 通过匹配输出的脉宽调节器

15.3 描述

定时器/计数器对外设时钟（PCLK）或外部提供的时钟周期进行计数，可选择产生中断或根据 4 个匹配寄存器的设定，在到达指定的定时值时执行其它动作。它还包括 4 个捕获输入，用于在输入信号发生跳变时捕获定时器值，并可选择产生中断。

由于 LPC2101/02/03 器件的管脚数目有限，只有 3 个捕获输入和定时器 0 的 3 个匹配输出被连接到器件的管脚。

2 个匹配寄存器可以在 MATn.2..0 管脚上提供单边沿控制的 PWM 输出。由于 MAT0.3 寄存器在定时器 0 上没有管脚输出，因此建议使用 MRn.3 寄存器来控制 PWM 周期长度。需要用另一个匹配寄存器来控制 PWM 边沿位置。剩余的两个寄存器可用来创建 PWM 输出，该输出具有 MRn.3 决定的 PWM 周期率。

15.4 管脚描述

表 164 所示为每个定时器/计数器相关管脚的简要描述。

表 164 定时器/计数器管脚描述

管脚名称	管脚方向	管脚描述
CAP0.2..0 CAP1.3..0	输入	**捕获信号** 捕获管脚的跳变可配置为将定时器值装入一个捕获寄存器，并可选择产生一个中断。 下面是所有捕获信号及其选择的管脚的列表： ● CAP0.0：P0.2 ● CAP0.1：P0.4。 ● CAP0.2：P0.6 ● CAP1.0：P0.10 ● CAP1.1：P0.11 ● CAP1.2：P0.17 ● CAP1.3：P0.18 计数器/定时器可选择一个捕获信号作为时钟源来代替 PCLK 分频时钟。详见 15.5.3 节 “计数控制寄存器(CTCR，TIMER0：T0CTCR-0xE0004070 和 TIMER1：T1TCR- 0xE0008070)”。
MAT0.2..0 MAT1.3..0	输出	**外部匹配输出 0/1** 当匹配寄存器 0/1（MR3:0）等于定时器计数器（TC）时，该输出可翻转，变为低电平、变为高电平或不变。外部匹配寄存器（EMR）和 PWM 控制寄存器(PWMCON)控制该输出的功能。 下面是所有匹配信号及其选择的管脚的列表： ● MAT0.0：P0.3 ● MAT0.1：P0.5 ● MAT0.2：P0.16 ● MAT1.0：P0.12 ● MAT1.1：P0.13 ● MAT1.2：P0.19 ● MAT1.3：P0.20

15.5 寄存器描述

每个定时器/计数器所包含的寄存器如表 165 所示。详见下列的描述。

表 165　定时器/计数器 0 和定时器/计数器 1 寄存器映射

通用名称	描述	访问	复位值[1]	定时器/计数器 0 地址&名称	定时器/计数器 1 地址&名称
IR	中断寄存器。可以写 IR 来清除中断。可读取 IR 来识别哪个中断源（8 个可能中断源中）被挂起。	R/W	0	0xE0004000 T0IR	0xE0008000 T1IR
TCR	定时器控制寄存器。TCR 用于控制定时器计数器功能。定时器计数器可通过 TCR 禁止或复位。	R/W	0	0xE0004004 T0TCR	0xE0008004 T1TCR
TC	定时器计数器。32 位 TC 每经过 PR+1 个 PCLK 周期加 1。TC 通过 TCR 进行控制。	R/W	0	0xE0004008 T0TC	0xE0008008 T1TC
PR	预分频寄存器。预分频计数器(下面)等于该值。下个时钟增加 TC 并清除 PC。	R/W	0	0xE000400C T0PR	0xE000800C T1PR
PC	预分频计数器。32 位 PC 是加 1 到 PR 内的存储值的计数器。当到达 PR 中保存的值时，TC 加 1 且 PC 被清除。通过总线接口可观察和控制 PC。	R/W	0	0xE0004010 T0PC	0xE0008010 T1PC
MCR	匹配控制寄存器。MCR 用于控制在匹配时是否产生中断或复位 TC。	R/W	0	0xE0004014 T0MCR	0xE0008014 T1MCR
MR0	匹配寄存器 0。MR0 可通过 MCR 设定为在每次 MR0 匹配 TC 时复位 TC，停止 TC 和 PC，和/或产生中断。	R/W	0	0xE0004018 T0MR0	0xE0008018 T1MR0
MR1	匹配寄存器 1。见 MR0 描述	R/W	0	0xE000401C T0MR1	0xE000801C T1MR1
MR2	匹配寄存器 2。见 MR0 描述	R/W	0	0xE0004020 T0MR2	0xE0008020 T1MR2
MR3	匹配寄存器 3。见 MR0 描述	R/W	0	0xE0004024 T0MR3	0xE0008024 T1MR3
CCR	捕获控制寄存器。CCR 控制用于装载捕获寄存器的捕获输入边沿以及在发生捕获时是否产生中断。	R/W	0	0xE0004028 T0CCR	0xE0008028 T1CCR
CR0	捕获寄存器 0。当在 CAPn.0(分别为 CAP0.0 或 CAP1.0)上产生捕获事件时，CR0 装载 TC 的值。	RO	0	0xE000402C T0CR0	0xE000802C T1CR0
CR1	捕获寄存器 1。见 CR0 描述。	RO	0	0xE0004030 T0CR1	0xE0008030 T1CR1
CR2	捕获寄存器 2。见 CR0 描述。	RO	0	0xE0004034 T0CR2	0xE0008034 T1CR2
CR3	捕获寄存器 3。见 CR0 描述。 **注**：CAP0.3 在定时器 0 上不可用。	RO	0	0xE0004038 T0CR3 不可用	0xE0008038 T1CR3

续上表

通用名称	描述	访问	复位值[1]	定时器/计数器 0 地址&名称	定时器/计数器 1 地址&名称
EMR	外部匹配寄存器。EMR 控制外部匹配管脚 MAT0.2..0 和 MAT1.3..0。 **注**：MAT0.3 不连接到 LPC2101/02/03 的管脚。	R/W	0	0xE000403C T0EMR	0xE000803C T1EMR
CTCR	计数控制寄存器。CTCR 选择定时器或计数器模式，且在计数器模式下选择计数的信号和边沿。	R/W	0	0xE0004070 T0CTCR	0xE0008070 T1CTCR
PWMCON	PWM 控制寄存器。PWMCON 使能外部匹配管脚 MAT0.3..0 和 MAT1.3..0 的 PWM 模式。	R/W	0	0xE0004074 PWM0CON	0xE0008074 PWM1CON

[1] 复位值仅指已使用位中保存的数据。它不包括保留位的内容。

15.5.1 中断寄存器（IR，定时器 0：T0IR-0xE000 4000 和定时器 1：T1IR-0xE0008000）

中断寄存器包含 4 个位用于匹配中断，4 个位用于捕获中断。如果有中断产生，IR 中的对应位会置位，否则为 0。向对应的 IR 位写入 1 会复位中断。写入 0 无效。

表 166　中断寄存器（IR，定时器 0 - T0IR：地址 0xE0004000；定时器 1 - T1IR：地址 0xE0008000）位描述

位	功能	描述	复位值
0	MR0 中断	匹配通道 0 的中断标志	0
1	MR1 中断	匹配通道 1 的中断标志	0
2	MR2 中断	匹配通道 2 的中断标志	0
3	MR3 中断	匹配通道 3 的中断标志	0
4	CR0 中断	捕获通道 0 事件的中断标志	0
5	CR1 中断	捕获通道 1 事件的中断标志	0
6	CR2 中断	捕获通道 2 事件的中断标志	0
7	CR3 中断	捕获通道 3 事件的中断标志 **注**：CAP0.3 在定时器 0 上不可使用	0

15.5.2 定时器控制寄存器（TCR，定时器 0 – T0TCR：0xE0004004；定时器 1 – T1TCR：0xE0008004）

定时器控制寄存器(TCR)用于控制定时器/计数器的操作。

表 167　定时器控制寄存器（TCR，定时器 0：T0TCR- 地址 0xE0004004 和定时器 1：T1TCR- 地址 0xE0008004）位描述

位	符号	描述	复位值
0	计数器使能	为 1 时，定时器计数器和预分频计数器使能计数。为 0 时，计数器被禁止。	0
1	计数器复位	为 1 时，定时器计数器和预分频计数器在 PCLK 的下一个上升沿同步复位。计数器在 TCR[1]恢复为 0 之前保持复位状态。	0
7:2	-	保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

15.5.3 计数控制寄存器(CTCR: 定时器 0-T0CTCR:0xE0004070 和定时器 1-T1TCR: 0xE0008070)

计数控制寄存器(CTCR)用来选择定时器或计数器模式，且在计数器模式下选择计数的管脚和边沿。

当选择工作在计数器模式时，在每个 PCLK 时钟的上升沿对 CAP 输入（由 CTCR 位 3:2 选择）进行采样。比较完 CAP 输入的 2 次连续采样结果后，可以识别下面四个事件中的一个：上升沿、下降沿、任一边沿或选择的 CAP 输入的电平无变化。只要识别到的事件与 CTCR 寄存器中位 1:0 选择的事件相对应时，定时器计数器寄存器加 1。

计数器外部时钟源的有效操作受到一些限制。由于 PCLK 时钟的 2 个连续的上升沿用来识别 CAP 选择输入的一个边沿，所以 CAP 输入的频率不能大于 1/2 个 PCLK 时钟。因此，这种情况下同一 CAP 输入的高/低电平持续时间不能小于 1/PCLK。

表 168　计数控制寄存器(CTCR，定时器 0-T0CTCR: 地址 0xE000 4070 和定时器 1-T1TCR: 地址 0xE000 8070)位描述

位	功能	值	描述	复位值
1:0	计数器/定时器模式		该字段选择触发定时器的预分频计数器(PC)递增、清除 PC 和定时器计数器(TC)递增的 PCLK 边沿：	00
		00	定时器模式：每个 PCLK 的上升沿。	
		01	计数器模式：TC 在位 3:2 选择的 CAP 输入的上升沿递增。	
		10	计数器模式：TC 在位 3:2 选择的 CAP 输入的下降沿递增。	
		11	计数器模式：TC 在位 3:2 选择的 CAP 输入的上升和下降沿递增。	
3:2	计数器输入选择		当位 1:0 不是 00 时，这些位用来选择对哪一个 CAP 管脚进行采样计时：	00
		00	CAPn.0（CAP0.0 用于定时器 0 和 CAP1.0 用于定时器 1）	
		01	CAPn.1（CAP0.1 用于定时器 0 和 CAP1.1 用于定时器 1）	
		10	CAPn.2（CAP0.2 用于定时器 0 和 CAP1.2 用于定时器 1）	
		11	CAP1.3 用于定时器 1	
			注：如果在 TnCTCR 中选择计数器模式用于某个特定的 CAPn 输入，则捕获控制寄存器(TnCCR)中对应该输入的 3 位必须编程设为 000。但是，可在相同的定时器中选择其它 3 个 CAPn 输入用于捕获和/或中断。	
			注：CAP0.3 在定时器 0 上不可使用	
7:4	-	-	保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

15.5.4 定时器计数器（TC：定时器 0 – T0TC：0xE0004008；定时器 1 – T1TC：0xE0008008）

当预分频计数器到达计数的上限时，32 位定时器计数器加 1。如果 TC 在到达计数上限之前没有被复位，它将一直计数到 0xFFFFFFFF 然后翻转到 0x0000 0000。该事件不会产生中断。如果需要，可用匹配寄存器检测溢出。

15.5.5 预分频寄存器（PR：定时器 0 – T0PR：0xE000400C；定时器 1 – T1PR：0xE000800C）

32 位预分频寄存器指定到预分频计数器的最大值。

15.5.6 预分频计数器寄存器（PC：定时器 0 – T0PC：0xE0004010；定时器 1 – T1PC：0xE0008010）

32 位预分频计数器在其应用于定时器计数器之前，使用某个常量来控制 PCLK 的分频。这样可在定时器溢出之前实现控制定时器分辨率和最大时间之间的关系。预分频计数器每个 PCLK 周期加 1。当其到达预分频寄存器中保存的值时，定时器计数器加 1，预分频计数器在下个 PCLK 周期复位。这样，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 2 个 PCLK 周期加 1。

15.5.7 匹配寄存器（MR0 - MR3）

匹配寄存器值连续与定时器计数值相比较。当两个值相等时自动触发相应动作。这些动作包括产生中断，复位定时器计数器或停止定时器。所执行的动作由 MCR 寄存器的设定来控制。

15.5.8 匹配控制寄存器（MCR：定时器 0 – T0MCR：0xE0004014；定时器 1 – T1MCR：0xE0008014）

匹配控制寄存器用于控制在其中一个匹配寄存器匹配定时器计数器时所执行的操作。每个位的功能见表 169。

表 169　匹配控制寄存器（MCR：定时器 0 – T0MCR：地址 0xE0004014；定时器 1 – T1MCR：地址 0xE0008014）位描述

位	符号	值	描述	复位值
0	MR0I	1 0	MR0 上的中断：MR0 与 TC 值的匹配将产生中断。 中断被禁止。	0
1	MR0R	1 0	MR0 上的复位：MR0 与 TC 值的匹配将使 TC 复位。 该特性被禁止。	0
2	MR0S	1 0	MR0 上的停止：MR0 与 TC 值的匹配将使 TC 和 PC 停止，TCR[0]清零。 该特性被禁止。	0
3	MR1I	1 0	MR1 的中断：MR1 与 TC 值的匹配将产生中断。 中断被禁止。	0
4	MR1R	1 0	MR1 的复位：MR1 与 TC 值的匹配将使 TC 复位。 该特性被禁止。	0
5	MR1S	1 0	MR1 的停止：MR1 与 TC 值的匹配将使 TC 和 PC 停止，TCR[0]清零。 该特性被禁止。	0
6	MR2I	1 0	MR2 的中断：MR2 与 TC 值的匹配将产生中断。 中断被禁止。	0
7	MR2R	1 0	MR2 上的复位：MR2 与 TC 值的匹配将使 TC 复位。 该特性被禁止。	0
8	MR2S	1 0	MR2 上的停止：MR2 与 TC 值的匹配将使 TC 和 PC 停止，TCR[0]清零。 该特性被禁止。	0
9	MR3I	1 0	MR3 上的中断：MR3 与 TC 值的匹配将产生中断。 中断被禁止。	0

续上表

位	符号	值	描述	复位值
10	MR3R	1 0	MR3 上的复位：MR3 与 TC 值的匹配将使 TC 复位。 该特性被禁止。	0
11	MR3S	1 0	MR3 上的停止：MR3 与 TC 值的匹配将使 TC 和 PC 停止，TCR[0]清零。 该特性被禁止。	0
15:12	-		保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

15.5.9 捕获寄存器（CR0 - CR3）

每个捕获寄存器都与一个器件管脚相关联。当管脚发生特定的事件时，可将定时器计数值装入该寄存器。捕获控制寄存器的设定决定捕获功能是否使能以及捕获事件在管脚的上升沿、下降沿或是双边沿发生。

15.5.10 捕获控制寄存器（CCR：定时器 0 – T0CCR：0xE0004028；定时器 1 – T1CCR：0xE0008028）

当发生捕获事件时，捕获控制寄存器用于控制将定时器计数值是否装入 4 个捕获寄存器中的一个以及是否产生中断。同时设置上升沿和下降沿位也是有效的配置，这样会在双边沿触发捕获事件。在下面的描述中，“n”代表定时器的编号 0 或 1。

表 170　捕获控制寄存器（CCR：定时器 0 – T0CCR：地址 0xE0004028；定时器 1 – T1CCR：地址 0xE0008028）位描述

位	符号	值	描述	复位值
0	CAP0RE	1 0	CAPn.0 上升沿上的捕获：CAPn.0 上 0 到 1 的跳变将导致 TC 的内容装入 CR0。 该特性被禁止。	0
1	CAP0FE	1 0	CAPn.0 下降沿上的捕获：CAPn.0 上 1 到 0 的跳变将导致 TC 的内容装入 CR0。 该特性被禁止。	0
2	CAP0I	1 0	CAPn.0 事件上的中断：CAPn.0 的捕获事件所导致的 CR0 装载将产生一个中断。 该特性被禁止。	0
3	CAP1RE	1 0	CAPn.1 上升沿上的捕获：CAPn.1 上 0 到 1 的跳变将导致 TC 的内容装入 CR1。 该特性被禁止。	0
4	CAP1FE	1 0	CAPn.1 下降沿上的捕获：CAPn.1 上 1 到 0 的跳变将导致 TC 的内容装入 CR1。 该特性被禁止。	0
5	CAP1I	1 0	CAPn.1 事件上的中断：CAPn.1 的捕获事件所导致的 CR1 装载将产生一个中断。 该特性被禁止。	0
6	CAP2RE	1 0	CAPn.2 上升沿的捕获：CAPn.2 上 0 到 1 的跳变将导致 TC 的内容装入 CR2。 该特性被禁止。	0

续上表

位	符号	值	描述	复位值
7	CAP2FE	1 0	CAPn.2 下降沿上的捕获：CAPn.2 上 1 到 0 的跳变将导致 TC 的内容装入 CR2。 该特性被禁止。	0
8	CAP2I	1 0	CAPn.2 事件上的中断：CAPn.2 的捕获事件所导致的 CR2 装载将产生一个中断。 该特性被禁止。	0
9	CAP3RE	1 0	CAPn.3 上升沿上的捕获：CAPn.3 上 0 到 1 的跳变将导致 TC 的内容装入 CR3。[1] 该特性被禁止。	0
10	CAP3FE	1 0	CAPn.3 下降沿上的捕获：CAPn.3 上 1 到 0 的跳变将导致 TC 的内容装入 CR3。 该特性被禁止。	0
11	CAP3I	1 0	CAPn.3 事件上的中断：CAPn.3 的捕获事件所导致的 CR3 装载将产生一个中断。 该特性被禁止。	0
15:12	-		保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

[1] 在定时器 0 上，CAP0.3 被禁止且 CAP3RE, CAP3FE 和 CAP3I 的值未被定义。

15.5.11 外部匹配寄存器（EMR：定时器 0 – T0EMR：0xE000403C 和定时器 1 – T1EMR：0xE000803C）

外部匹配寄存器提供外部匹配管脚 MAT(0-3)的控制和状态。

如果匹配输出配置为 PWM 输出，外部匹配寄存器的功能由 PWM 规则决定（请见 15.5.13 节“单边沿控制 PWM 输出的规则”）。

表 171　外部匹配寄存器（EMR：定时器 0 – T0EMR：地址 0xE000403C 和定时器 1 – T1EMR：地址 0xE000803C）位描述

位	符号	描述	复位值
0	EM0	外部匹配 0。不管 MAT0.0/MAT1.0 是否连接到管脚，该位都会反映 MAT0.0/MAT1.0 的状态。当 MR0 发生匹配时，该输出可翻转，变为低电平，变为高电平或不执行任何动作。位 EMR[5:4]控制该输出的功能。	0
1	EM1	外部匹配 1。不管 MAT0.1/MAT1.1 是否连接到管脚，该位都会反映 MAT0.1/MAT1.1 的状态。当 MR1 发生匹配时，该输出可翻转，变为低电平，变为高电平或不执行任何动作。位 EMR[7:6]控制该输出的功能。	0
2	EM2	外部匹配 2。不管 MAT0.2/MAT1.2 是否连接到管脚，该位都会反映 MAT0.2/MAT1.2 的状态。当 MR2 发生匹配时，该输出可翻转，变为低电平，变为高电平或不执行任何动作。位 EMR[9:8]控制该输出的功能。	0
3	EM3	外部匹配 3。不管 MAT0.3/MAT1.3 是否连接到管脚，该位都会反映 MAT0.3/MAT1.3 的状态。当 MR3 发生匹配时，该输出可翻转，变为低电平，变为高电平或不执行任何动作。位 EMR[10:11]控制该输出的功能。	0
5:4	EMC0	外部匹配控制 0。决定外部匹配 0 的功能。表 172 所示为这两个位的编码。	00
7:6	EMC1	外部匹配控制 1。决定外部匹配 1 的功能。表 172 所示为这两个位的编码。	00

续上表

位	符号	描述	复位值
9:8	EMC2	外部匹配控制 2。决定外部匹配 2 的功能。表 172 所示为这两个位的编码。	00
11:10	EMC3	外部匹配控制 3。决定外部匹配 3 的功能。表 172 所示为这两个位的编码。	00
15:12	-	保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

表 172　外部匹配控制

EMR[11:10], EMR[9:8] EMR[7:6],或 EMR[5:4]	功能
00	不执行任何动作
01	将对应的外部匹配位/输出设置为 0（如果连接到管脚，则 MATn.m 管脚为低电平）
10	将对应的外部匹配位/输出设置为 1（如果连接到管脚，则 MATn.m 管脚为高电平）
11	使对应的外部匹配位/输出翻转

15.5.12 PWM 控制寄存器(PWMCON，定时器 0：PWM0CON- 0xE0004074 和定时器 1：PWM1CON- 0xE0008074)

PWM 控制寄存器可用来配置匹配输出为 PWM 输出。可单独设置每个匹配输出以执行 PWM 输出或匹配输出，该匹配输出的功能由外部匹配寄存器(EMR)来控制。

对于每个定时器，可在 MATn.2:0 输出上选择 3 个单边沿控制 PWM 输出的最大值。一个匹配寄存器决定 PWM 周期的长度。当匹配在其它任何一个匹配寄存器中出现时，PWM 输出设置为高电平。定时器通过匹配寄存器配置为设定 PWM 周期长度来复位。当定时器被复位为 0 时，清除所有当前高电平匹配输出配置为 PWM 的输出。

表 173　PWM 控制寄存器(PWMCON，定时器 0：PWM0CON-0xE0004074 和定时器 1：PWM1CON-0xE0008074)位描述

位	符号	描述	复位值
0	PWM 使能	该位为 1 时，PWM 模式被使能用于 MATn.0。该位为 0 时，MATn.0 由 EM0 控制。	0
1	PWM 使能	该位为 1 时，PWM 模式被使能用于 MATn.1。该位为 0 时，MATn.1 由 EM1 控制。	0
1	PWM 使能	该位为 1 时，PWM 模式被使能用于 MATn.2。该位为 0 时，MATn.2 由 EM2 控制。	0
1	PWM 使能	该位为 1 时，PWM 模式被使能用于 MATn.3。该位为 0 时，MATn.3 由 EM3 控制。 **注**：建议使用 MATn.3 来设置 PWM 周期，因为定时器 0 没有管脚输出。	0
4:32	-	保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

15.5.13 单边沿控制 PWM 输出的规则

1. 所有单边沿控制的 PWM 输出在每个 PWM 周期的开始变低（定时器设为 0），除非它们的匹配值等于 0。
2. 每个 PWM 输出在到达其匹配值时将变高。如果没有匹配发生（如匹配值大于 PWM 周期长度），那么 PWM 输出保持为持续的低电平。
3. 如果匹配值大于写入匹配寄存器的 PWM 周期长度，且 PWM 信号已为高电平，那么 PWM 信号将在下个定时器复位时清零。
4. 如果匹配寄存器含有与定时器复位值相同的值（PWM 周期长度），那么 PWM 输出将在下个时钟节拍复位为低电平。因此，PWM 输出将总是包括一个时钟节拍宽正向脉冲，该脉冲带有 PWM 周期长度确定的周期（即定时器重装值）。
5. 如果匹配寄存器设为 0，那么 PWM 输出将在定时器到达其复位值的第一时间内变为高电平且将持续保持高电平。

注：当选择匹配输出用作 PWM 输出时，匹配控制寄存器 MCR 中的定时器复位(MRnR)和定时器停止(MRnS)位必须设为 0，但匹配寄存器设置 PWM 周期长度除外。对于该寄存器，当定时器值匹配对应的匹配寄存器值时，设置 MRnR 位为 1 来使能定时器复位。

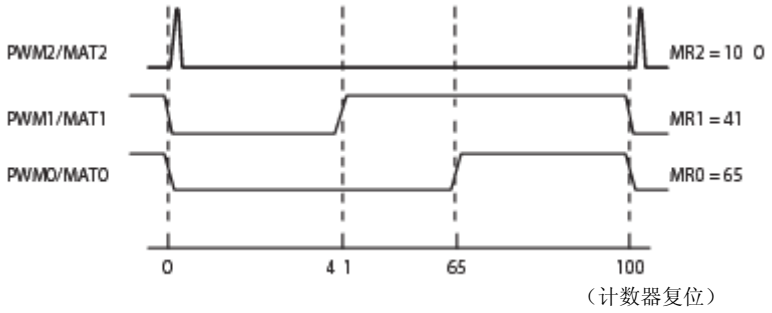


图 49　具有 PWM 周期长度 100（由 MR3 选择）和通过 PWCON 寄存器 MAT3:0 使能为 PWM 输出的采样 PWM 波形

15.6 定时器举例操作

图 50 所示为定时器配置为在匹配时复位计数并产生中断。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 51 所示为定时器配置为在匹配时停止并产生中断。预分频器设置为 2，匹配寄存器设置为 6。在定时器到达匹配值的下一个周期中，TCR 中的定时器使能位清零并产生指示匹配发生的中断。

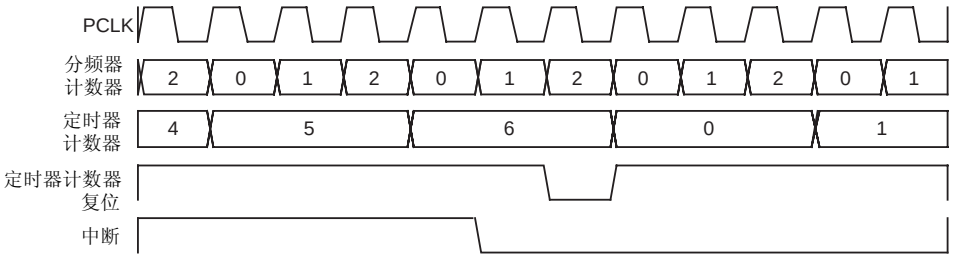


图 50 定时器周期设置为 PR=2, MRx=6, 匹配时使能中断和复位

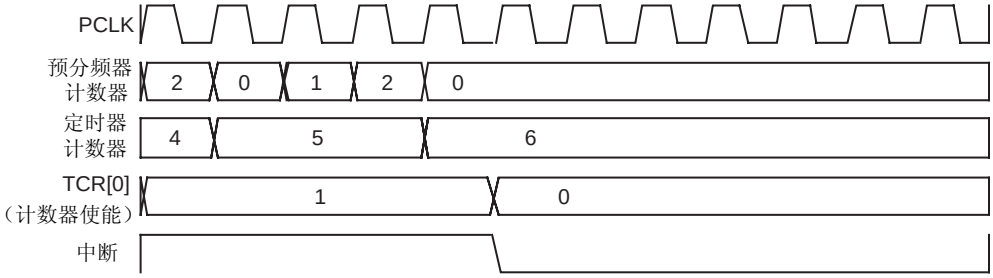
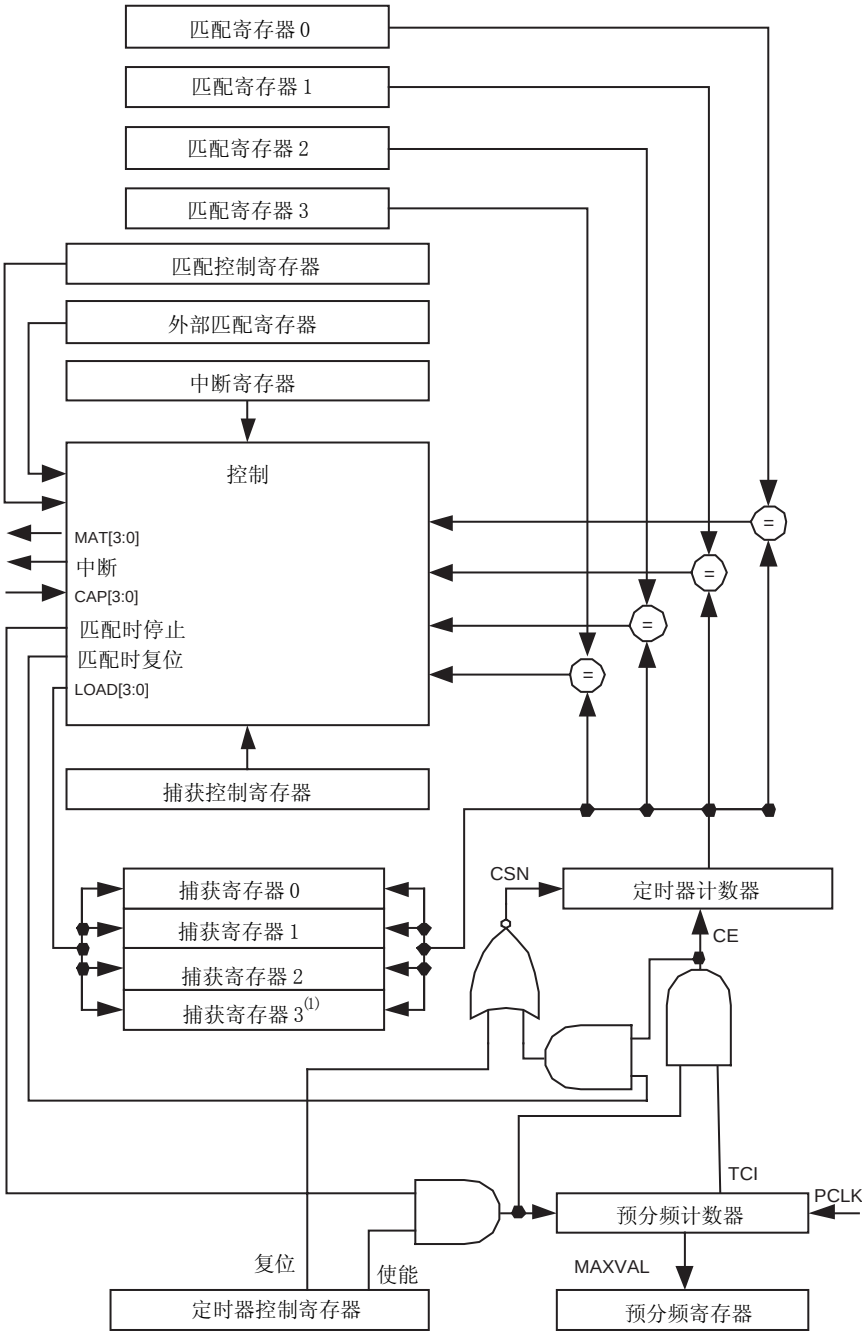


图 51 定时器周期设置为 PR=2, MRx=6, 匹配时使能中断和停止

15.7 结构

定时器/计数器 0 和定时器/计数器 1 的方框图，见图 52。



⑴ 捕获寄存器3 (CAP0.3)不能用于定时器 0。

图 52　定时器 0/1 方框图

第16章 定时器/计数器 定时器 2 和定时器 3

定时器/计数器 2 和定时器/计数器 3 除了外设基地址以外，其它都相同。

16.1 特性

- 带可编程 16 位预分频器的 16 位定时器/计数器
- 计数器或定时器操作
- 具有 3 路（定时器 2）16 位的捕获通道。当输入信号跳变时可取得定时器的瞬时值。也可选择使捕获事件产生中断。
- 4 个 16 位匹配寄存器：

  －匹配时定时器继续工作，可选择产生中断
  －匹配时停止定时器，可选择产生中断
  －匹配时复位定时器，可选择产生中断
- 多达 4 个(定时器 3)和 3 个(定时器 2)对应于匹配寄存器的外部输出，具有下列特性：

  －匹配时设置为低电平
  －匹配时设置为高电平
  －匹配时翻转
  －匹配时无动作
- 对于每个定时器，多达 4 个匹配寄存器可配置为 PWM，允许使用多达 3 个匹配输出作为单边沿控制的 PWM 输出。

16.2 应用

- 用于对内部事件进行计数的间隔定时器
- 通过捕获输入实现脉宽调制
- 自由运行的定时器
- 通过匹配输出的脉宽调节器

16.3 描述

定时器/计数器对外设时钟（PCLK）或外部提供的时钟周期进行计数，可选择产生中断或根据 4 个匹配寄存器的设定，在到达指定的定时值时执行其它动作。它还包括 4 个捕获输入，用于在输入信号发生跳变时捕获定时器值，并可选择产生中断。

由于 LPC2101/02/03 器件的管脚数目有限，因此定时器 3 没有捕获输入，只有定时器 2 的 3 个捕获输入和 3 个匹配输出，以及定时器 3 的 4 个匹配输出被连接到器件的管脚。

2 个匹配寄存器可以在 MATn.2..0 管脚上提供单边沿控制的 PWM 输出。由于 MAT2.3 寄存器在定时器 2 上没有管脚输出，因此建议使用 MRn.3 寄存器来控制 PWM 周期长度。需要用另一个匹配寄存器来控制 PWM 边沿位置。剩余的两个匹配寄存器可用来创建 PWM

输出，该输出具有 MRn.3 决定的 PWM 周期率。

16.4 管脚描述

表 174 所示为每个定时器/计数器相关管脚的简要描述。

表 174 定时器/计数器管脚描述

管脚名称	管脚方向	管脚描述
CAP2.2..0	输入	**捕获信号** 捕获管脚的跳变可配置为将定时器值装入一个捕获寄存器，并可选择产生一个中断。 下面是所有捕获信号及其选择的管脚的列表： ● CAP2.0：P0.27 ● CAP2.1：P0.28 ● CAP2.2：P0.29 计数器/定时器可选择一个捕获信号作为时钟源来代替 PCLK 分频时钟。详见 16.5.3 节 “计数控制寄存器(CTCR，TIMER2：T2CTCR-0xE0070070 和 TIMER3：T3TCR- 0xE0074070)”。
MAT2.2..0 MAT3.3..0	输出	**外部匹配输出 0/1** 当匹配寄存器 0/1（MR3:0）等于定时器计数器（TC）时，该输出可翻转，变为低电平、变为高电平或不变。外部匹配寄存器（EMR）和 PWM 控制寄存器(PWMCON)控制该输出的功能。 下面是所有匹配信号及其选择的管脚的列表： ● MAT2.0：P0.7 ● MAT2.1：P0.8 ● MAT2.2：P0.9 ● MAT3.0：P0.21 ● MAT3.1：P0.13 ● MAT3.2：P0.14 ● MAT3.3：P0.15

16.5 寄存器描述

每个定时器/计数器所包含的寄存器如表 175 所示。详见下列的描述。

表 175　定时器/计数器 2 和定时器/计数器 3 寄存器映射

通用名称	描述	访问	复位值[1]	定时器/计数器 2 地址&名称	定时器/计数器 3 地址&名称
IR	中断寄存器。可以写 IR 来清除中断。可读取 IR 来识别哪个中断源（8 个可能中断源中）被挂起。	R/W	0	0xE0070000 T2IR	0xE0074000 T3IR
TCR	定时器控制寄存器。TCR 用于控制定时器计数器功能。定时器计数器可通过 TCR 禁止或复位。	R/W	0	0xE0070004 T2TCR	0xE0074004 T3TCR
TC	定时器计数器。16 位 TC 每经过 PR+1 个 PCLK 周期加 1。TC 通过 TCR 进行控制。	R/W	0	0xE0070008 T2TC	0xE0074008 T3TC
PR	预分频寄存器。预分频计数器(下面)等于该值。下个时钟增加 TC 并清除 PC。	R/W	0	0xE007000C T2PR	0xE007400C T3PR
PC	预分频计数器。16 位 PC 是加 1 到 PR 内的存储值的计数器。当到达 PR 中保存的值时，TC 加 1 且 PC 被清除。通过总线接口可观察和控制 PC。	R/W	0	0xE0070010 T2PC	0xE0074010 T3PC
MCR	匹配控制寄存器。MCR 用于控制在匹配时是否产生中断或复位 TC。	R/W	0	0xE0070014 T2MCR	0xE0074014 T3MCR
MR0	匹配寄存器 0。MR0 可通过 MCR 设定为在每次 MR0 匹配 TC 时复位 TC，停止 TC 和 PC，和/或产生中断。	R/W	0	0xE0070018 T2MR0	0xE0074018 T3MR0
MR1	匹配寄存器 1。见 MR0 描述	R/W	0	0xE007001C T2MR1	0xE007401C T3MR1
MR2	匹配寄存器 2。见 MR0 描述	R/W	0	0xE0070020 T2MR2	0xE0074020 T3MR2
MR3	匹配寄存器 3。见 MR0 描述	R/W	0	0xE0070024 T2MR3	0xE0074024 T3MR3
CCR	捕获控制寄存器。CCR 控制用于装载捕获寄存器的捕获输入边沿以及在发生捕获时是否产生中断。	R/W	0	0xE0070028 T2CCR	0xE0074028 T3CCR
CR0	捕获寄存器 0。当在 CAP2.0 上产生捕获事件时，CR0 装载 TC 的值。 **注**：CAP3.0 在定时器 3 上不可用。	RO	0	0xE007002C T2CR0	0xE007402C T3CR0
CR1	捕获寄存器 1。见 CR0 描述。 **注**：CAP3.1 在定时器 3 上不可用。	RO	0	0xE0070030 T2CR1	0xE0074030 T3CR1
CR2	捕获寄存器 2。见 CR0 描述。 **注**：CAP3.2 在定时器 3 上不可用。	RO	0	0xE0070034 T2CR2	0xE0074034 T3CR2

续上表

通用名称	描述	访问	复位值[1]	定时器/计数器 2 地址&名称	定时器/计数器 3 地址&名称
EMR	外部匹配寄存器。EMR 控制外部匹配管脚 MAT2.2..0 和 MAT3.3..0。 **注**：MAT2.3 不连接到 LPC2101/02/03 的管脚。	R/W	0	0xE007003C T2EMR	0xE007403C T3EMR
CTCR	计数控制寄存器。CTCR 选择定时器或计数器模式，且在计数器模式下选择计数的信号和边沿。	R/W	0	0xE0070070 T2CTCR	0xE0074070 T3CTCR
PWMCON	PWM 控制寄存器。PWMCON 使能外部匹配管脚 MAT2.3..0 和 MAT3.3..0 的 PWM 模式。	R/W	0	0xE0070074 PWM0CON	0xE0074074 PWM1CON

[1] 复位值仅指已使用位中保存的数据。它不包括保留位的内容。

16.5.1 中断寄存器（IR，定时器 2：T2IR-0xE007 0000 和定时器 3：T3IR-0xE0074000）

中断寄存器包含 4 个位用于匹配中断，4 个位用于捕获中断。如果有中断产生，则 IR 中的对应位会置位，否则为 0。向对应的 IR 位写入 1 会复位中断。写入 0 无效。

表 176　中断寄存器（IR，定时器 2 – T2IR：地址 0xE0070000；定时器 3 – T3IR：地址 0xE0074000）位描述

位	功能	描述	复位值
0	MR0 中断	匹配通道 0 的中断标志	0
1	MR1 中断	匹配通道 1 的中断标志	0
2	MR2 中断	匹配通道 2 的中断标志	0
3	MR3 中断	匹配通道 3 的中断标志	0
4	CR0 中断	捕获通道 0 事件的中断标志	0
5	CR1 中断	捕获通道 1 事件的中断标志	0
6	CR2 中断	捕获通道 2 事件的中断标志	0
7	CR3 中断	捕获通道 3 事件的中断标志 **注**：CAPn.3 在定时器 2/定时器 3 上不可使用	0

16.5.2 定时器控制寄存器（TCR，定时器 2– T2TCR：0xE0070004 和定时器 3 – T3TCR：0xE0074004）

定时器控制寄存器(TCR)用于控制定时器/计数器的操作。

表 177　定时器控制寄存器（TCR，定时器 2：T2TCR- 地址 0xE0070004 和定时器 3：T3TCR- 地址 0xE0074004）位描述

位	符号	描述	复位值
0	计数器使能	为 1 时，定时器计数器和预分频计数器使能计数。为 0 时，计数器被禁止。	0
1	计数器复位	为 1 时，定时器计数器和预分频计数器在 PCLK 的下一个上升沿同步复位。计数器在 TCR[1]恢复为 0 之前保持复位状态。	0
7:2	-	保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

16.5.3 计数控制寄存器(CTCR: 定时器 2-T2CTCR:0xE0070070 和定时器 3-T3TCR: 0xE0074070)

计数控制寄存器(CTCR)用来选择定时器或计数器模式，且在计数器模式下选择计数的管脚和边沿。

当选择工作在计数器模式时，在每个 PCLK 时钟的上升沿对 CAP 输入（由 CTCR 位 3:2 选择）进行采样。比较完 CAP 输入的 2 次连续采样结果后，可以识别下面四个事件中的一个：上升沿、下降沿、任一边沿或选择的 CAP 输入的电平无变化。只要识别到的事件与 CTCR 寄存器中位 1:0 选择的事件相对应时，定时器计数器寄存器加 1。

计数器外部时钟源的有效操作受到一些限制。由于 PCLK 时钟的 2 个连续的上升沿用来识别 CAP 选择输入的一个边沿，所以 CAP 输入的频率不能大于 1/2 个 PCLK 时钟。因此，这种情况下同一 CAP 输入的高/低电平持续时间不能小于 1/PCLK。

表 178　计数控制寄存器(CTCR，定时器 2-T2CTCR: 地址 0xE007 0070 和定时器 3-T3TCR: 地址 0xE007 4070)位描述

位	功能	值	描述	复位值
1:0	计数器/定时器模式		该字段选择触发定时器的预分频计数器(PC)递增、清除 PC 和定时器计数器(TC)递增的 PCLK 边沿：	00
		00	定时器模式：每个 PCLK 的上升沿。	
		01	计数器模式：TC 在位 3:2 选择的 CAP 输入的上升沿递增。	
		10	计数器模式：TC 在位 3:2 选择的 CAP 输入的下降沿递增。	
		11	计数器模式：TC 在位 3:2 选择的 CAP 输入的上升和下降沿递增。	
3:2	计数器输入选择		当位 1:0 不是 00 时，这些位用来选择对哪一个 CAP 管脚进行采样计时：	00
		00	CAP2.0	
		01	CAP2.1	
		10	CAP2.2	
			注：如果在 TnCTCR 中选择计数器模式用于某个特定的 CAPn 输入，则捕获控制寄存器(TnCCR)中对应该输入的 3 位必须编程设为 000。但是，可在相同的定时器中选择其它 3 个 CAPn 输入用于捕获和/或中断。	
		-	CAPn.3 在定时器 2/3 上不可使用	
7:4	-	-	保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

16.5.4 定时器计数器（TC：定时器 2 – T2TC：0xE0070008；定时器 3 – T3TC：0xE0074008）

当预分频计数器到达计数的上限时，16 位定时器计数器加 1。如果 TC 在到达计数上限之前没有被复位，它将一直计数到 0xFFFFFFFF 然后翻转到 0xE000 0000。该事件不会产生中断。如果需要，可用匹配寄存器检测溢出。

16.5.5 预分频寄存器（PR：定时器 2 – T2PR：0xE007000C；定时器 3– T3PR：0xE007400C）

16 位预分频寄存器指定到预分频计数器的最大值。

16.5.6 预分频计数器寄存器（PC：定时器 2 – T2PC：0xE0070010；定时器 3 – T3PC：0xE0074010）

16 位预分频计数器在其应用于定时器计数器之前，使用某个常量来控制 PCLK 的分频。这样可在定时器溢出之前实现控制定时器分辨率和最大时间之间的关系。预分频计数器每个 PCLK 周期加 1。当其到达预分频寄存器中保存的值时，定时器计数器加 1，预分频计数器在下个 PCLK 周期复位。这样，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 2 个 PCLK 周期加 1。

16.5.7 匹配寄存器（MR0 - MR3）

匹配寄存器值连续与定时器计数值相比较。当两个值相等时自动触发相应动作。这些动作包括产生中断，复位定时器计数器或停止定时器。所执行的动作由 MCR 寄存器的设定来控制。

16.5.8 匹配控制寄存器（MCR：定时器 2 – T2MCR：0xE0070014 和定时器 3 – T3MCR：0xE0074014）

匹配控制寄存器用于控制在其中一个匹配寄存器匹配定时器计数器时所执行的操作。每个位的功能见表 179。

表 179 匹配控制寄存器（MCR：定时器 2 – T2MCR：地址 0xE0070014；定时器 3 – T3MCR：地址 0xE0074014）位描述

位	符号	值	描述	复位值
0	MR0I	1 0	MR0 上的中断：MR0 与 TC 值的匹配将产生中断。 中断被禁止。	0
1	MR0R	1 0	MR0 上的复位：MR0 与 TC 值的匹配将使 TC 复位。 该特性被禁止。	0
2	MR0S	1 0	MR0 上的停止：MR0 与 TC 值的匹配将使 TC 和 PC 停止，TCR[0]清零。 该特性被禁止。	0
3	MR1I	1 0	MR1 的中断：MR1 与 TC 值的匹配将产生中断。 中断被禁止。	0
4	MR1R	1 0	MR1 的复位：MR1 与 TC 值的匹配将使 TC 复位。 该特性被禁止。	0
5	MR1S	1 0	MR1 的停止：MR1 与 TC 值的匹配将使 TC 和 PC 停止，TCR[0]清零。 该特性被禁止。	0
6	MR2I	1 0	MR2 的中断：MR2 与 TC 值的匹配将产生中断。 中断被禁止。	0
7	MR2R	1 0	MR2 上的复位：MR2 与 TC 值的匹配将使 TC 复位。 该特性被禁止。	0
8	MR2S	1 0	MR2 上的停止：MR2 与 TC 值的匹配将使 TC 和 PC 停止，TCR[0]清零。 该特性被禁止。	0
9	MR3I	1 0	MR3 上的中断：MR3 与 TC 值的匹配将产生中断。 中断被禁止。	0

续上表

位	符号	值	描述	复位值
10	MR3R	1 0	MR3 上的复位：MR3 与 TC 值的匹配将使 TC 复位。 该特性被禁止。	0
11	MR3S	1 0	MR3 上的停止：MR3 与 TC 值的匹配将使 TC 和 PC 停止，TCR[0]清零。 该特性被禁止。	0
15:12	-		保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

16.5.9 捕获寄存器（CR0 - CR3）

每个捕获寄存器都与一个器件管脚相关联。当管脚发生特定的事件时，可将定时器计数值装入该寄存器。捕获控制寄存器的设定决定捕获功能是否使能以及捕获事件是否在管脚的上升沿、下降沿或是双边沿发生。

16.5.10 捕获控制寄存器（CCR：定时器 2 – T2CCR：0xE0070028；定时器 3 – T3CCR：0xE0074028）

当发生捕获事件时，捕获控制寄存器用于控制将定时器计数值是否装入 4 个捕获寄存器中的一个以及是否产生中断。同时设置上升沿和下降沿位也是有效的配置，这样会在双边沿触发捕获事件。在下面的描述中，“n”代表定时器的编号 2 或 3。

表 180　捕获控制寄存器（CCR：定时器 2 – T2CCR：地址 0xE0070028；定时器 3 – T3CCR：地址 0xE0074028）位描述

位	符号	值	描述	复位值
0	CAP0RE	1 0	CAPn.0 上升沿上的捕获：CAPn.0 上 0 到 1 的跳变将导致 TC 的内容装入 CR0。 该特性被禁止。	0
1	CAP0FE	1 0	CAPn.0 下降沿上的捕获：CAPn.0 上 1 到 0 的跳变将导致 TC 的内容装入 CR0。 该特性被禁止。	0
2	CAP0I	1 0	CAPn.0 事件上的中断：CAPn.0 的捕获事件所导致的 CR0 装载将产生一个中断。 该特性被禁止。	0
3	CAP1RE	1 0	CAPn.1 上升沿上的捕获：CAPn.1 上 0 到 1 的跳变将导致 TC 的内容装入 CR1。 该特性被禁止。	0
4	CAP1FE	1 0	CAPn.1 下降沿上的捕获：CAPn.1 上 1 到 0 的跳变将导致 TC 的内容装入 CR1。 该特性被禁止。	0
5	CAP1I	1 0	CAPn.1 事件上的中断：CAPn.1 的捕获事件所导致的 CR1 装载将产生一个中断。 该特性被禁止。	0
6	CAP2RE	1 0	CAPn.2 上升沿的捕获：CAPn.2 上 0 到 1 的跳变将导致 TC 的内容装入 CR2。 该特性被禁止。	0

续上表

位	符号	值	描述	复位值
7	CAP2FE	1 0	CAPn.2 下降沿上的捕获：CAPn.2 上 1 到 0 的跳变将导致 TC 的内容装入 CR2。 该特性被禁止。	0
8	CAP2I	1 0	CAPn.2 事件上的中断：CAPn.2 的捕获事件所导致的 CR2 装载将产生一个中断。 该特性被禁止。	0
15:9	- [1]		保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

[1] 在定时器 2/3 上，CAPn.3 被禁止且 CAP3RE, CAP3FE 和 CAP3I 的值未被定义。

16.5.11 外部匹配寄存器（EMR: 定时器 2 – T2EMR：0xE007003C 和定时器 3 – T3EMR：0xE007403C）

外部匹配寄存器提供外部匹配管脚 MAT(0-3)的控制和状态。

如果匹配输出配置为 PWM 输出，外部匹配寄存器的功能由 PWM 规则决定（请见 16.7 节“单边沿控制 PWM 输出的规则”）。

表 181 外部匹配寄存器（EMR: 定时器 2 – T2EMR：地址 0xE007003C 和定时器 3 – T3EMR：地址 0xE0074016 位 3C）位描述

位	符号	描述	复位值
0	EM0	外部匹配 0。不管 MAT2.0/MAT3.0 是否连接到管脚，该位都会反映 MAT2.0/MAT3.0 的状态。当 MR0 发生匹配时，该输出可翻转，变为低电平，变为高电平或不执行任何动作。位 EMR[5:4]控制该输出的功能。	0
1	EM1	外部匹配 1。不管 MAT2.1/MAT3.1 是否连接到管脚，该位都会反映 MAT2.1/MAT3.1 的状态。当 MR1 发生匹配时，该输出可翻转，变为低电平，变为高电平或不执行任何动作。位 EMR[7:6]控制该输出的功能。	0
2	EM2	外部匹配 2。不管 MAT2.2/MAT3.2 是否连接到管脚，该位都会反映 MAT2.2/MAT3.2 的状态。当 MR2 发生匹配时，该输出可翻转，变为低电平，变为高电平或不执行任何动作。位 EMR[9:8]控制该输出的功能。	0
3	EM3	外部匹配 3。不管 MAT2.3/MAT3.3 是否连接到管脚，该位都会反映 MAT2.3/MAT3.3 的状态。当 MR3 发生匹配时，该输出可翻转，变为低电平，变为高电平或不执行任何动作。位 EMR[11:10]控制该输出的功能。	0
5:4	EMC0	外部匹配控制 0。决定外部匹配 0 的功能。表 182 所示为这两个位的编码。	00
7:6	EMC1	外部匹配控制 1。决定外部匹配 1 的功能。表 182 所示为这两个位的编码。	00
9:8	EMC2	外部匹配控制 2。决定外部匹配 2 的功能。表 182 所示为这两个位的编码。	00
11:10	EMC3	外部匹配控制 3。决定外部匹配 3 的功能。表 182 所示为这两个位的编码。	00
15:12	-	保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

表 182 外部匹配控制

EMR[11:10], EMR[9:8] EMR[7:6],或 EMR[5:4]	功能
00	不执行任何动作
01	将对应的外部匹配位/输出设置为 0（如果连接到管脚，则 MATn.m 管脚为低电平）
10	将对应的外部匹配位/输出设置为 1（如果连接到管脚，则 MATn.m 管脚为高电平）
11	使对应的外部匹配位/输出翻转

16.6 PWM 控制寄存器(PWMCON，定时器 2： PWM0CON-0xE0070074 和定时器 3：PWM1CON- 0xE0074074)

PWM 控制寄存器可用来配置匹配输出为 PWM 输出。可单独设置每个匹配输出以执行 PWM 输出或匹配输出，该匹配输出的功能由外部匹配寄存器(EMR)来控制。

对于每个定时器，可在 MATn.2:0 输出上选择 3 个单边沿控制 PWM 输出的最大值。一个匹配寄存器决定 PWM 周期的长度。当匹配在其它任何一个匹配寄存器中出现时，PWM 输出设置为高电平。定时器通过匹配寄存器配置为设定 PWM 周期长度来复位。当定时器被复位为 0 时，清除所有当前高电平匹配输出配置为 PWM 的输出。

表 183 PWM 控制寄存器(PWMCON，定时器 0：PWM0CON-0xE0070074 和定时器 1：PWM1CON-0xE0074074)位描述

位	符号	描述	复位值
0	PWM 使能	该位为 1 时，PWM 模式被使能用于 MATn.0。该位为 0 时，MATn.0 由 EM0 控制。	0
1	PWM 使能	该位为 1 时，PWM 模式被使能用于 MATn.1。该位为 0 时，MATn.1 由 EM1 控制。	0
1	PWM 使能	该位为 1 时，PWM 模式被使能用于 MATn.2。该位为 0 时，MATn.2 由 EM2 控制。	0
1	PWM 使能	该位为 1 时，PWM 模式被使能用于 MATn.3。该位为 0 时，MATn.3 由 EM3 控制。 **注**：建议使用 MATn.3 来设置 PWM 周期，因为定时器 0 没有管脚输出。	0
4:32	-	保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

16.7 单边沿控制 PWM 输出的规则

1. 所有单边沿控制的 PWM 输出在每个 PWM 周期的开始变低（定时器设为 0），除非它们的匹配值等于 0。
2. 每个 PWM 输出在到达其匹配值时将变高。如果没有匹配发生（如匹配值大于 PWM 周期长度），那么 PWM 输出保持为持续的低电平。
3. 如果匹配值大于写入匹配寄存器的 PWM 周期长度，且 PWM 信号已为高电平，那么 PWM 信号将在下个定时器复位时清零。
4. 如果匹配寄存器含有与定时器复位值相同的值（PWM 周期长度），那么 PWM 输出将在下个时钟节拍复位为低电平。因此，PWM 输出将总是包括一个时钟节拍宽正向脉冲，该脉冲带有 PWM 周期长度确定的周期（即定时器重装值）。
5. 如果匹配寄存器设为 0，那么 PWM 输出将在定时器到达其复位值的第一时间内变为高电平且将持续保持高电平。

注：当选择匹配输出用作 PWM 输出时，匹配控制寄存器 MCR 中的定时器复位(MRnR)和定时器停止(MRnS)位必须设为 0，但匹配寄存器设置 PWM 周期长度除外。对于该寄存器，当定时器值匹配对应的匹配寄存器值时，设置 MRnR 位为 1 来使能定时器复位。

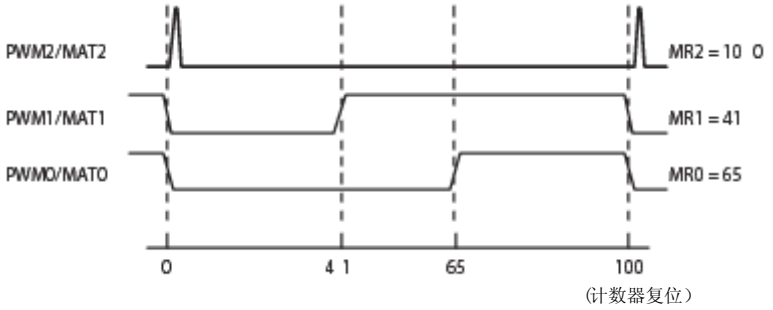


图 53　具有 PWM 周期长度 100（由 MR3 选择）和通过 PWCON 寄存器 MAT3:0 使能为 PWM 输出的采样 PWM 波形

16.8 定时器举例操作

图 54 所示为定时器配置为在匹配时复位计数并产生中断。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 55 所示为定时器配置为在匹配时停止并产生中断。预分频器设置为 2，匹配寄存器设置为 6。在定时器到达匹配值的下一个周期中，TCR 中的定时器使能位清零并产生指示匹配发生的中断。

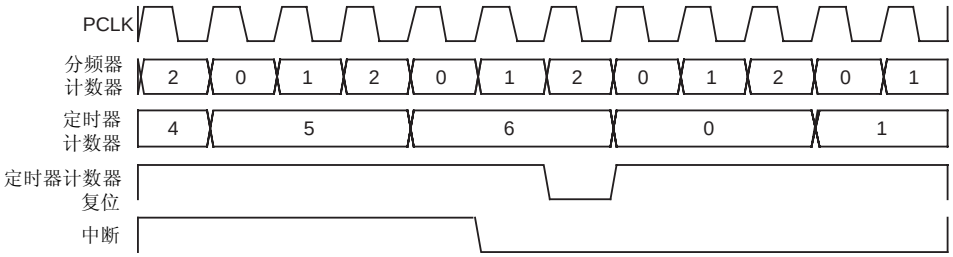


图 54　定时器周期设置为 PR=2, MRx=6, 匹配时使能中断和复位

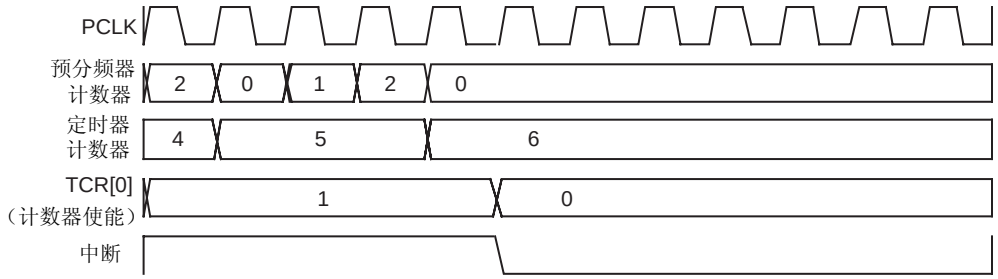
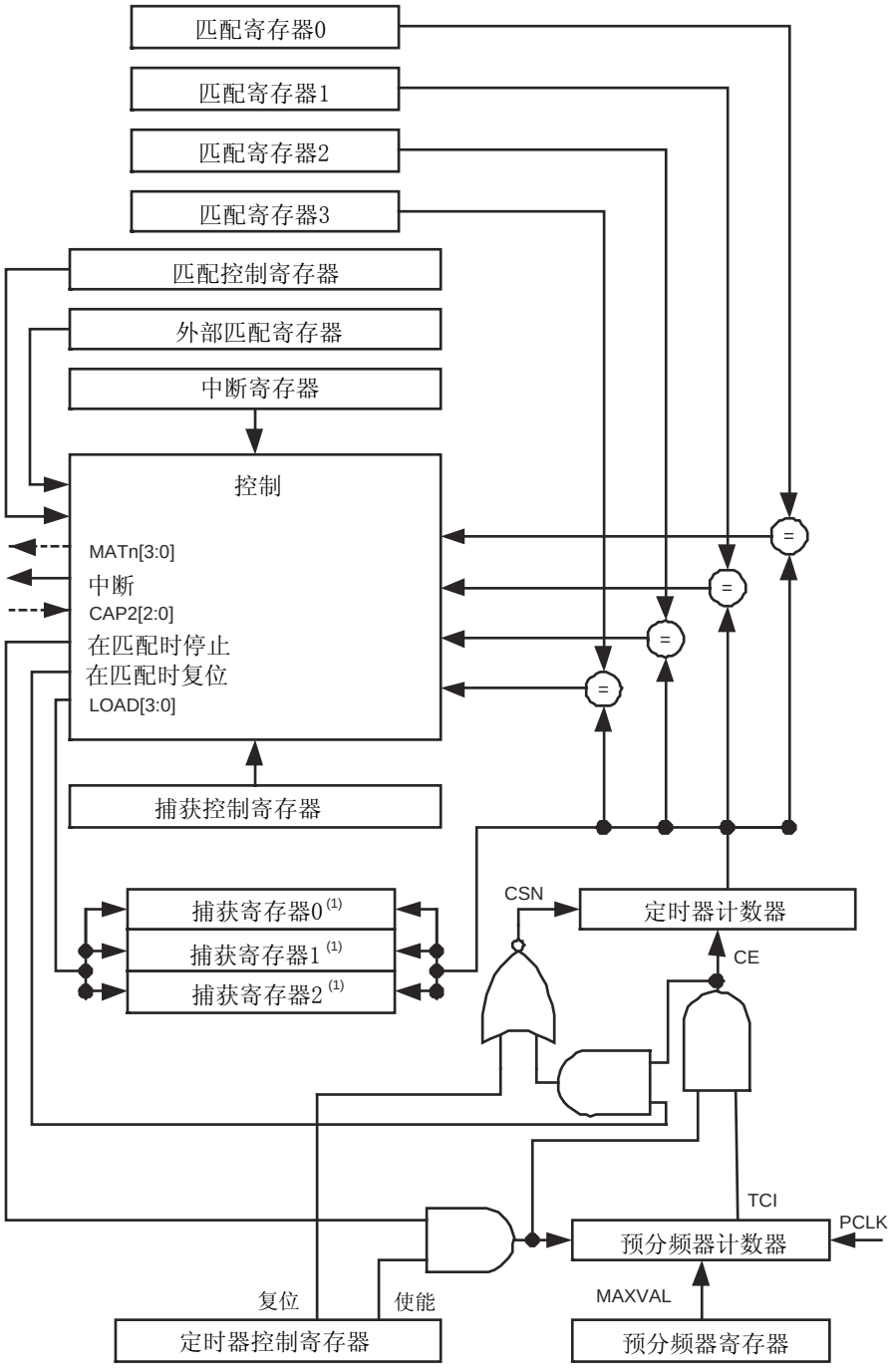


图 55　定时器周期设置为 PR=2, MRx=6, 匹配时使能中断和停止

16.9 结构

定时器/计数器 2 和定时器/计数器 3 的方框图，见图 56。



（1）定时器 3 不可使用捕获寄存器。

图 56　定时器 2/3 方框图

第17章 实时时钟

17.1 特性

- 测量保持日历和时钟的时间通路
- 超低功耗设计，支持电池供电系统
- 提供秒、分、小时、日、月、年和星期
- 可使用指定的 32kHz 振荡器或可编程 APB 时钟预分频器
- 专用电源管脚可与电池或 3.3V 的电压相连

17.2 描述

实时时钟(RTC)提供一套计数器在系统上电和关闭操作时对时间进行测量。RTC 在掉电模式下消耗的功率非常低。LPC2101/02/03 的 RTC 时钟可由独立的 32.768kHz 振荡器或基于 APB 时钟的可编程预分频器来提供。另外，RTC 还具有专用的电源管脚 V_{BAT}，可连接到电池或其它器件使用的相同的 3.3V 电压上。

17.3 结构

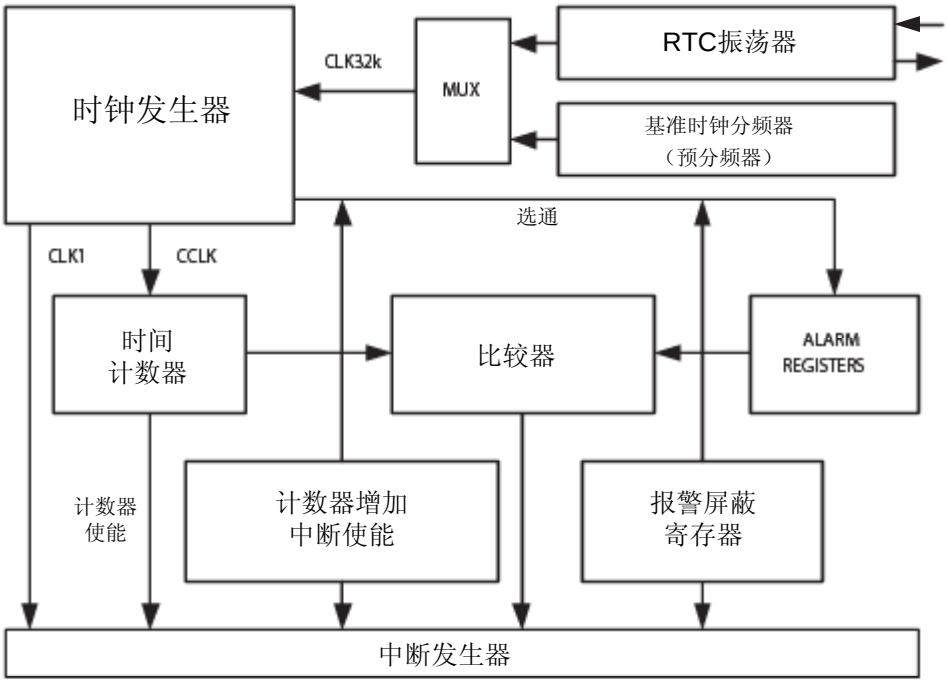


图 57 RTC 方框图

17.4 寄存器描述

RTC 包含了许多寄存器。地址空间按照功能分成 4 个部分。前 8 个地址为混合寄存器组（见 17.4.2 节）。第二部分的 8 个地址为定时器计数器组（见 17.4.12 节），第三部分的 8 个地址为报警寄存器组（见 17.4.14 节）。剩余的寄存器控制基准时钟分频器。

实时时钟模块所包含的寄存器见表 184。详细的描述见后面的寄存器。

表 184 实时时钟(RTC)寄存器映射

名称	规格	描述	访问	复位值[1]	地址
ILR	2	中断位置寄存器	R/W	*	0xE0024000
CTC	15	时钟节拍计数器	RO	*	0xE0024004
CCR	4	时钟控制寄存器	R/W	*	0xE0024008
CIIR	8	计数器递增中断寄存器	R/W	*	0xE002400C
AMR	8	报警屏蔽寄存器	R/W	*	0xE0024010
CTIME0	32	完整时间寄存器 0	RO	*	0xE0024014
CTIME1	32	完整时间寄存器 1	RO	*	0xE0024018
CTIME2	32	完整时间寄存器 2	RO	*	0xE002401C
SEC	6	秒计数器	R/W	*	0xE0024020
MIN	6	分寄存器	R/W	*	0xE0024024
HOUR	5	小时寄存器	R/W	*	0xE0024028
DOM	5	日期（月）寄存器	R/W	*	0xE002402C
DOW	3	星期寄存器	R/W	*	0xE0024030
DOY	9	日期（年）寄存器	R/W	*	0xE0024034
MONTH	4	月寄存器	R/W	*	0xE0024038
YEAR	12	年寄存器	R/W	*	0xE002403C
ALSEC	6	秒报警值	R/W	*	0xE0024060
ALMIN	6	分报警值	R/W	*	0xE0024064
ALHOUR	5	小时报警值	R/W	*	0xE0024068
ALDOM	5	日期（月）报警值	R/W	*	0xE002406C
ALDOW	3	星期报警值	R/W	*	0xE0024070
ALDOY	9	日期（年）报警值	R/W	*	0xE0024074
ALMON	4	月报警值	R/W	*	0xE0024078
ALYEAR	12	年报警值	R/W	*	0xE002407C
PREINT	13	预分频值，整数部分	R/W	0	0xE0024080
PREFRAC	15	预分频值，小数部分	R/W	0	0xE0024084

[1] RTC 中除预分频器部分之外的其它寄存器都不受器件复位的影响。如果 RTC 使能，这些寄存器必须通过软件来初始化。复位值仅指已使用位中保存的数据。它不包括保留位的内容。

17.4.1 RTC 中断

中断的产生由中断位置寄存器(ILR)、计数器递增中断寄存器(CIIR)、报警寄存器和报警屏蔽寄存器(AMR) 控制。只有转换到中断状态才能产生中断。ILR 单独使能 CIIR 和 AMR 中断。CIIR 中的每个位都对应一个时间计数器。如果 CIIR 使能用于一个特定的计数器，那

么该计数器的值每增加一次就产生一个中断。报警寄存器允许用户设定产生中断的日期或时间。AMR 提供一个屏蔽报警比较的机制。如果所有非屏蔽报警寄存器与它们对应的时间计数器的值相匹配时，则会产生中断。

RTC 中断（如果 RTC 工作在其自身的 RTCX1-2 管脚的振荡器下）可使微控制器退出掉电模式。当 RTC 中断使能产生唤醒并且所选中断事件出现时，启动 X1/2 管脚相关的振荡器唤醒周期。有关 RTC 唤醒处理的内容详见 3.5.3 节“中断唤醒寄存器(INTWAKE -0xE01FC144)”和 3.12 节“唤醒定时器”。

17.4.2 混合寄存器组

表 185 所示为 A[6:2]的 0 到 7 的寄存器。详见下面的描述。

表 185 混合寄存器

名称	规格	描述	访问	地址
ILR	2	中断位置寄存器。读出的该位置寄存器的值指示了中断源。向寄存器的一个位写入 1 来清除相应的中断。	RW	0xE0024000
CTC	15	时钟节拍计数器。该寄存器的值来自时钟分频器。	RO	0xE0024004
CCR	4	时钟控制寄存器。控制时钟分频器的功能。	RW	0xE0024008
CIIR	8	计数器递增中断寄存器。当计数器递增时，选择一个计数器产生中断。	RW	0xE002400C
AMR	8	报警屏蔽寄存器。控制报警寄存器的屏蔽。	RW	0xE0024010
CTIME0	32	完整时间寄存器 0	RO	0xE0024014
CTIME1	32	完整时间寄存器 1	RO	0xE0024018
CTIME2	32	完整时间寄存器 2	RO	0xE002401C

17.4.3 中断位置寄存器（ILR - 0xE002 4000）

中断位置寄存器为 2 位寄存器，它指定哪些模块产生中断（见表 186）。向一个位写入 1 会清除相应的中断。写入 0 无效。这样程序员可以读取该寄存器并将读出的值回写到寄存器中清除检测到的中断。

表 186 中断位置寄存器（ILR – 地址 0xE0024000）位描述

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。	NA
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。	NA
7:2	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

17.4.4 时钟节拍计数器寄存器（CTCR - 0xE0024004）

时钟节拍计数器只可读。它可通过时钟控制寄存器（CCR）复位为 0。CTC 包含时钟分频计数器位。

表 187 时钟节拍计数器寄存器（CTCR – 地址 0xE0024004）位描述

位	符号	描述	复位值
14:0	时钟节拍计数器	位于秒计数器之前，CTC 每秒计数 32,768 个时钟。由于 RTC 预分频器的关系，这 32,768 个时间增量的长度可能并不全部相同。详见 17.6 节“基准时钟分频器（预分频器）”。	NA
15	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

17.4.5 时钟控制寄存器（CCR - 0xE0024008）

时钟寄存器是一个 5 位寄存器，它控制时钟分频电路的操作。每一位的功能见表 188。

表 188 时钟控制寄存器（CCR – 地址 0xE0024008）位描述

位	符号	描述	复位值
0	CLKEN	时钟使能。当该位为 1 时，时间计数器使能。为 0 时，时间计数器都被禁止，这时可对其进行初始化。	NA
1	CTCRST	CTC 复位。为 1 时，时钟节拍计数器复位。在 CCR[1]变为 0 之前，它将一直保持复位状态。	NA
3:2	CTTEST	测试使能。在正常操作中，这些位应当全为 0。	NA
4	CLKSRC	如果该位为 0，时钟节拍计数器计数预分频器的时钟，这与早先的 Philips 嵌入式 ARM 系列器件相吻合。如果该位为 1，CTC 计数连接在 RTCX1 和 RTCX2 两端的 32kHz 振荡器信号（关于硬件的详细内容请见 17.7 节“RTC 外部 32KHz 振荡器元件选择”）。	NA
7:5	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

17.4.6 计数器增量中断寄存器（CIIR-0xE002400C）

计数器增量中断寄存器（CIIR）可使计数器每次增加时产生一次中断。在中断位置寄存器的位 0 （ILR[0]）写入 1 之前，该中断一直保持有效。

表 189 计数器增量中断寄存器（CIIR – 地址 0xE002400C）位描述

位	符号	描述	复位值
0	IMSEC	为 1 时，秒值的增加产生一次中断。	NA
1	IMMIN	为 1 时，分值的增加产生一次中断。	NA
2	IMHOUR	为 1 时，小时值的增加产生一次中断。	NA
3	IMDOM	为 1 时，日期（月）值的增加产生一次中断。	NA
4	IMDOW	为 1 时，星期值的增加产生一次中断。	NA
5	IMDOY	为 1 时，日期（年）值的增加产生一次中断。	NA
6	IMMON	为 1 时，月值的增加产生一次中断。	NA
7	IMYEAR	为 1 时，年值的增加产生一次中断。	NA

17.4.7 报警屏蔽寄存器（AMR - 0xE0024010）

报警屏蔽寄存器（AMR）允许用户屏蔽任意报警寄存器。表 190 所示为 AMR 位与报警寄存器位之间的关系。对于报警功能来说，要产生中断，非屏蔽的报警寄存器必须匹配对应的时间计数值。只有当计数器之间的比较第一次从不匹配到匹配时才会产生中断。向中断

位置寄存器（ILR）的位写入 1 会清除相应的中断。如果所有屏蔽位都置位，报警将被禁止。

表 190 报警屏蔽寄存器（AMR – 地址 0xE0024010）位描述

位	符号	描述	复位值
0	AMRSEC	为 1 时，秒值不与报警寄存器比较。	NA
1	AMRMIN	为 1 时，分值不与报警寄存器比较。	NA
2	AMRHOUR	为 1 时，小时值不与报警寄存器比较。	NA
3	AMRDOM	为 1 时，日期（月）值不与报警寄存器比较。	NA
4	AMRDOW	为 1 时，星期值不与报警寄存器比较。	NA
5	AMRDOY	为 1 时，日期（年）值不与报警寄存器比较。	NA
6	AMRMON	为 1 时，月值不与报警寄存器比较。	NA
7	AMRYEAR	为 1 时，年值不与报警寄存器比较。	NA

17.4.8 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，程序员只需执行 3 次读操作即可读出所有的时间计数器值。32 位值的不同寄存器见表 191，表 192 和表 193。每个寄存器的最低位分别在 bit0, 8, 16 和 24 被读回。

完整时间寄存器为只读寄存器。要更新时间计数器的值，必须对时间计数器寻址。

17.4.9 完整时间寄存器 0（CTIME0 - 0xE0024014）

完整时间寄存器 0 包含的时间值为：秒、分、小时和星期。

表 191 完整时间寄存器 0（CTIME0 – 地址 0xE0024014）位描述

位	符号	描述	复位值
5:0	秒	秒值的范围为 0~59。	NA
7:6	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
13:8	分	分值的范围为 0~59。	NA
15:14	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
20:16	小时	小时值的范围为 0~23。	NA
23:21	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
26:24	星期	星期值的范围为 0~6。	NA
31:27	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

17.4.10 完整时间寄存器 1（CTIME1 - 0xE0024018）

完整时间寄存器 1 包含的时间值为：日期（月）、月和年。

表 192 完整时间寄存器 1（CTIME1 – 地址 0xE0024018）位描述

位	符号	描述	复位值
4:0	日期（月）	日期（月）值的范围为 1~28, 29, 30 或 31（取决于月份以及是否为闰年）。	NA
7:5	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
11:8	月	月值的范围为 1~12。	NA
15:12	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA
27:16	年	年值的范围为 0~4095。	NA
31:28	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

17.4.11 完整时间寄存器 2（CTIME2 - 0xE002401C）

完整时间寄存器 2 仅包含日期（年）。

表 193 完整时间寄存器 2（CTIME2 - 0xE002401C）位描述

位	符号	描述	复位值
11:0	日期（年）	日期（年）值的范围为 1~365（闰年为 366）。	NA
31:12	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

17.4.12 时间计数器组

时间值包含 8 个计数器，见表 194 和 195。表 195 所示的计数器可执行读或写操作。

表 194 时间计数器的关系和值

计数器	规格	使能	最小值	最大值
秒	6	Clk1（见图 57）	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 195 时间计数器寄存器

名称	规格	描述	访问	地址
SEC	6	秒值的范围为 0~59。	R/W	0xE0024020
MIN	6	分值的范围为 0~59。	R/W	0xE0024024
HOUR	5	小时值的范围为 0~23。	R/W	0xE0024028
DOM	5	日期（月）值的范围为 1~28,29,30 或 31（取决于月份以及是否为闰年）。[1]	R/W	0xE002402C
DOW	3	星期值的范围为 0~6。[1]	R/W	0xE0024030
DOY	9	日期（年）值的范围为 1~365（闰年为 366）。[1]	R/W	0xE0024034

续上表

名称	规格	描述	访问	地址
MONTH	4	月值的范围为 1~12。	R/W	0xE0024038
YEAR	12	年值的范围为 0~4095。	R/W	0xE002403C

[1]. 这些值只能在适当的时间间隔处递增且在定义的溢出点复位。为了使这些值有意义，它们不能进行计算且必须被正确初始化。

17.4.13 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 月份的长度、日期（月）和年的计数值。

17.4.14 报警寄存器组

报警寄存器见表 196。这些寄存器的值与时间计数器相比较。如果所有未屏蔽（见 17.4.7 节“报警屏蔽寄存器（AMR-0xE0024010）”）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。向中断位置寄存器的位 1（ILR[1]）写入 1 清除中断。

表 196 报警寄存器

名称	规格	描述	访问	地址
ALSEC	6	秒报警值	R/W	0xE0024060
ALMIN	6	分报警值	R/W	0xE0024064
ALHOUR	5	小时报警值	R/W	0xE0024068
ALDOM	5	日期（月）报警值	R/W	0xE002406C
ALDOW	3	星期报警值	R/W	0xE0024070
ALDOY	9	日期（年）报警值	R/W	0xE0024074
ALMON	4	月报警值	R/W	0xE0024078
ALYEAR	12	年报警值	R/W	0xE002407C

17.5 RTC 使用注意事项

如果使用 RTC，V_{BAT} 必须连接到 V_3 脚或一个独立的电源（外部电池）。否则，V_{BAT} 应该接地(Vss)。V_{BAT} 断电时 LPC2101/02/03 不能保存 RTC 的状态，如果时钟源丢失、中断或改变，RTC 也无法维持时间计数。

由于 RTC 有两个可用的时钟（APB 时钟(PCLK)或来自 RTCX1-2 管脚的 32KHz 的信号），所选择时钟的任何中断都会导致时间值的偏移。如果 RTC 初始化成这个时间值或从 RTC 激活后运行的一段时间内出现了一个错误，它们带来的变化都将影响真实的时钟时间。

RTCX1-2 管脚的信号可随时为 RTC 提供时钟，选择 PCLK 作为 RTC 时钟和进入掉电模式会使时间的更新出现误差。而且，在系统操作过程中（重新配置 PLL、APB 分频器或 RTC 预分频器）改变 RTC 的时间基准会使累加时间出现错误。当 RTC 时钟由 PCLK 转变为 RTCX 管脚信号时也会出现累加时间误差。

一旦 RTCX1-2 管脚的 32KHz 信号被选择用作 RTC 的时钟源，RTC 可完全独立工作，

与 APB 时钟(PCLK)无关。因此，在要用到 RTC 且对功耗敏感的应用中（如电池供电设备）可通过使用 RTCX1-2 管脚的信号和清除 PCONP 功率控制寄存器的 PCRTC 位来降低功耗（见 3.9 节“功率控制”）。

17.6 基准时钟分频器（预分频器）

基准时钟分频器（在下文中称为预分频器）允许从任何频率等于或高于 65.536kHz（2×32.768kHz）的外设时钟源产生一个 32.768kHz 的基准时钟。这样，不管外设时钟的频率为多少，RTC 总是以正确的速率运行。预分频器通过一个包含整数和小数部分的值对外设时钟(PCLK)进行分频。这样就产生了一个不是恒定频率的连续输出。有些时钟周期比其它周期多 1 个 PCLK 周期。但是每秒钟的计数总数总是 32,768。

基准时钟分频器包含一个 13 位整数计数器和一个 15 位小数计数器。使用该规格的原因如下：

1. 对于 LPC2101/02/03 所支持的频率，13 位整数计数器是必要的。可以这样进行计算：频率 160MHz 除以 32,768 再减去 1 等于 4881，余数为 26,624。保存 4881 需要 13 个位。13 位实际所能支持的最高频率为 268.4MHz（32,768×8192）。
2. 余数的最大值为 32,767，需要 15 位来保存。

表 197 基准时钟分频寄存器

名称	规格	描述	访问	地址
PREINT	13	预分频值，整数部分	R/W	0xE0024080
PREFRAC	15	预分频值，小数部分	R/W	0xE0024084

17.6.1 预分频整数寄存器（PREINT - 0xE0024080）

预分频值的整数部分计算如下：

PREINT = int (PCLK/32768) – 1。PREINT 的值必须大于或等于 1。

表 198 预分频整数寄存器（PREINT – 地址 0xE0024080）位描述

位	符号	描述	复位值
12:0	预分频整数	包含 RTC 预分频值的整数部分	0
15:13	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

17.6.2 预分频小数寄存器（PREFRAC - 0xE0024084）

预分频值的小数部分计算如下：

PREFRAC = PCLK – ((PREINT+1)×32768)

表 199 预分频小数寄存器（PREFRAC –地址 0xE0024084）位描述

位	符号	描述	复位值
14:0	预分频小数	包含 RTC 预分频值的小数部分	0
15	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

17.6.3 预分频器的使用举例

先假设一个最简单的状况，PCLK 频率为 65.537kHz。那么：

PREINT = int (PCLK/32768) – 1 = 1 且 PREFRAC = PCLK – ((PREINT+1)×32768) = 1

使用此设定，每秒钟有 32,767 次 2 个 PCLK 周期计数，1 次 3 个 PCLK 周期计数，加起来每秒刚好为 RTC 提供 32,768 个时钟。

再假设一个比较实际的状况，PCLK 频率为 10MHz。那么：

PREINT = int (PCLK/32768) – 1 = 304 且

PREFRAC = PCLK – ((PREINT+1)×32768) = 5,760

这时，有 5,760 个预分频器输出时钟宽度为 306 (305+1)PCLK 周期。余下的时钟宽度为 305 个 PCLK 周期。

采用相似的方法可以将任何高于 65.536kHz 的 PCLK 频率（每秒钟的周期数必须是偶数）转换成 RTC 的 32kHz 基准时钟。唯一需要注意的是，如果 PREFRAC 不等于 0，那么每秒当中的 32,768 个时钟长度是不完全相同的，有些时钟会比其它时钟多 1 个 PCLK 周期。虽然较长的脉冲已经尽可能地分配到剩余的脉冲当中，但是在希望直接观察时钟节拍计数器(CTC)（见 17.4.4 节“时钟节拍计数器寄存器(CTCR-0xE002 4004)”）的应用中可能需要注意这种“抖动”。

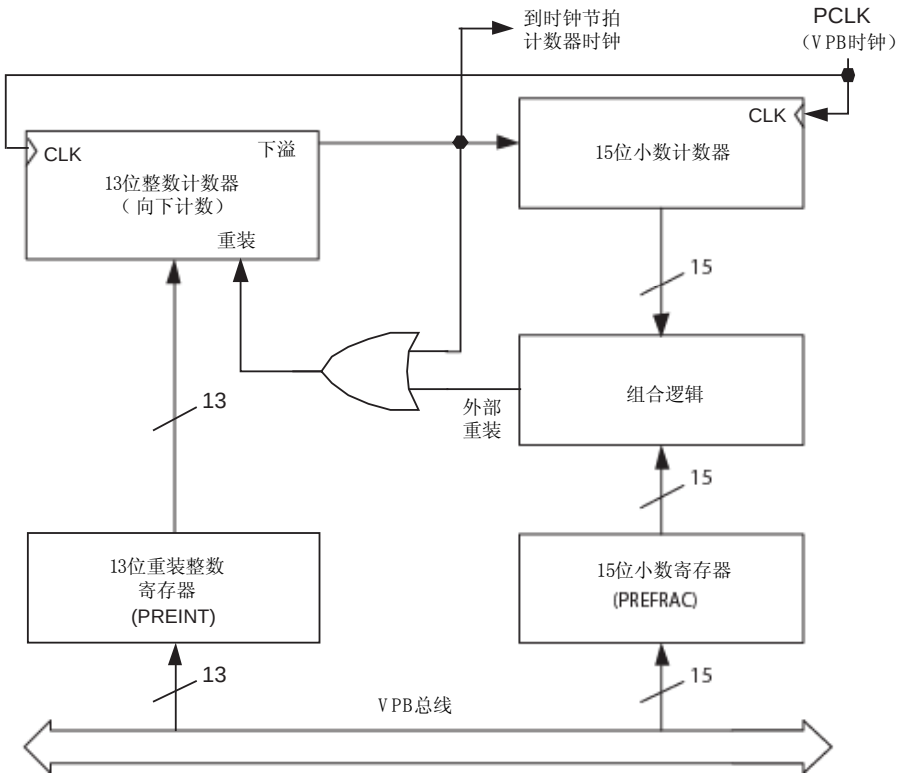


图 58 RTC 预分频器方框图

17.6.4 预分频器操作

图 58 预分频器模块中被称为“组合逻辑”的模块决定了何时 13 位 PREINT 计数器的递减操作延长 1 个 PCLK。为了使插入的延长周期数目正确并对它们进行合理地分配，组合逻

辑将 PREFRAC 的每一位与 15 位小数计数器的值对应起来。这种对应关系如表 200 所示。

例如，如果 PREFRAC 的位 14 为 1（表示小数 1/2），则 13 位计数器计数半个周期时延长。当小数计数器的 LSB 位为 1 时，组合逻辑将使每次计数变化（每当小数计数器的 LSB 位为 1 时）的时间延长 1 个 PCLK，并尽量平均分配脉冲宽度。类似地，PREFRAC 的位 13 为 1（表示小数 1/4）时 13 位计数器计数 1/4 个周期（每当小数计数器的低 2 位＝10 时）时延长。

表 200 整数计数器重装值递增时预分频器的情况

小数计数器	PREFRAC 位														
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
--- ---- ---- ---1	1	–	–	–	–	–	–	–	–	–	–	–	–	–	–
--- ---- ---- --10	–	1	–	–	–	–	–	–	–	–	–	–	–	–	–
--- ---- ---- -100	–	–	1	–	–	–	–	–	–	–	–	–	–	–	–
--- ---- ---- 1000	–	–	–	1	–	–	–	–	–	–	–	–	–	–	–
--- ---- ---1 0000	–	–	–	–	1	–	–	–	–	–	–	–	–	–	–
--- ---- --10 0000	–	–	–	–	–	1	–	–	–	–	–	–	–	–	–
--- ---- -100 0000	–	–	–	–	–	–	1	–	–	–	–	–	–	–	–
--- ---- 1000 0000	–	–	–	–	–	–	–	1	–	–	–	–	–	–	–
--- ---1 0000 0000	–	–	–	–	–	–	–	–	1	–	–	–	–	–	–
--- --10 0000 0000	–	–	–	–	–	–	–	–	–	1	–	–	–	–	–
--- -100 0000 0000	–	–	–	–	–	–	–	–	–	–	1	–	–	–	–
--- 1000 0000 0000	–	–	–	–	–	–	–	–	–	–	–	1	–	–	–
--1 0000 0000 0000	–	–	–	–	–	–	–	–	–	–	–	–	1	–	–
-10 0000 0000 0000	–	–	–	–	–	–	–	–	–	–	–	–	–	1	–
100 0000 0000 0000	–	–	–	–	–	–	–	–	–	–	–	–	–	–	1

17.7 RTC 外部 32kHz 振荡器元件选择

RTC 外部振荡器电路如图 59 所示。由于反馈电阻在片内集成，只有一个晶体，因此电容 C_{X1} 和 C_{X2} 需要外部连接到微控制器。

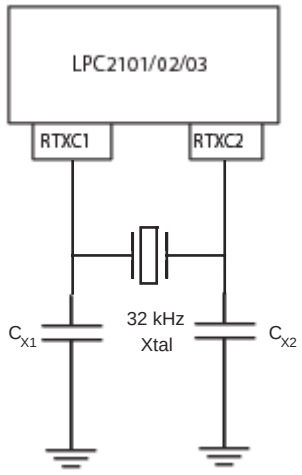


图 59 RTC 32kHz 晶体振荡器电路

表 201 给出需要使用的晶体参数。C_L 是典型的晶体负载电容并通常由晶体制造商指定。实际的 C_L 影响振荡器的频率。当使用不同负载电容制造的晶体时，电路相对于指定的频率，将在稍微不同的频率下振荡（取决于晶体的质量）。因此对于精确的时间基准，建议使用表 201 中特定 C_L 的指定负载电容。该表中外部电容 C_{X1} 和 C_{X2} 指定的值通过内部寄生电容和 C_L 计算。PCB 的寄生电容和封装都可以不考虑在内。

表 201 RTC 外部 32kHz 振荡器 $C_{X1/X2}$ 元件的推荐值

晶体负载电容 C_L	晶体串联电阻 R_S 的最大值	外部负载电容 C_{X1}, C_{X2}
11pF	＜100KΩ	18pF, 18pF
13pF	＜100KΩ	22pF, 22pF
15pF	＜100KΩ	27pF, 27pF

第18章 看门狗定时器（WDT）

18.1 特性

- 如果没有周期性重装，则产生片内复位
- 支持调试模式
- 由软件使能看门狗定时器，但要求禁止硬件复位或看门狗复位/中断
- 错误/不完整的喂狗时序会导致复位/中断(如果使能)
- 指示看门狗复位的标志
- 带内部预分频器的可编程 32 位定时器
- 可选择 $T_{PCLK}\times4$ 倍数的时间周期：从($T_{PCLK}\times256\times4$) 到 ($T_{PCLK}\times2^{32}\times4$)

18.2 应用

看门狗的用途是使微控制器在进入错误状态后的一定时间内复位。当看门狗使能时，如果用户程序没有在周期时间内喂狗（重装），看门狗会产生一个系统复位。

有关片内看门狗和外围功能的相互作用，尤其是复位和引导过程的关系，请参考本文档 3.10 节“复位”的内容。

18.3 描述

看门狗包括一个 4 分频的预分频器和一个 32 位计数器。时钟通过预分频器输入定时器。计时时定时器递减。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。因此最小看门狗间隔为($T_{PCLK}\times256\times4$)，最大看门狗间隔为($T_{PCLK}\times2^{32}\times4$)，两者都是 $T_{PCLK}\times4$ 的倍数。看门狗应当根据下面的方法来使用：

- 在 WDTC 寄存器中设置看门狗定时器的固定装载值
- 在 WDMOD 寄存器中设置模式
- 通过向 WDFEED 寄存器顺序写入 0xAA 和 0x55 启动看门狗
- 在看门狗计数器向下溢出之前应当再次喂狗以防止复位/中断

当看门狗计数器向下溢出时，程序计数器将从 0x0000 0000 开始，和外部复位一样。可以检查看门狗超时标志（WDTOF）来确定看门狗是否产生复位条件。WDTOF 标志必须由软件清零。

18.4 寄存器描述

看门狗包含 4 个寄存器，见表 202。

表 202 看门狗寄存器映射

名称	描述	访问	复位值[1]	地址
WDMOD	看门狗模式寄存器。该寄存器包含看门狗定时器的基本模式和状态。	R/W	0	0xE0000000
WDTC	看门狗定时器常数寄存器。该寄存器决定超时值。	R/W	0xFF	0xE0000004
WDFEED	看门狗喂狗序列寄存器。向该寄存器顺序写入 0xAA 和 0x55 使看门狗定时器重新装入预设值。	WO	NA	0xE0000008
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	0xE000000C

[1] 复位值仅表示存储在使用位的数据，它不包括保留位的内容。

18.4.1 看门狗模式寄存器（WDMOD - 0xE0000000）

WDMOD 寄存器通过 WDEN 和 RESET 位的组合来控制看门狗的操作。

表 203 看门狗操作模式选择

WDEN	WDRESET	操作模式
0	X（0 或 1）	看门狗关闭时的调试/操作
1	0	看门狗中断模式：带看门狗中断的调试，但没有使能 WDRESET。 当选择该模式时，看门狗计数器溢出将设置 WDINT 标志且将产生看门狗中断请求。
1	1	看门狗复位模式：带看门狗中断和 WDRESET 使能的操作。 当选择该模式时，看门狗计数器溢出将复位微控制器。当看门狗中断也在这种情况(WDEN=1)下使能时它将不会被识别，因为看门狗复位将清除 WDINT 标志。

一旦 **WDEN** 和/或 **WDRESET** 位设置，就无法使用软件将其清零。这两个标志由外部复位或看门狗定时器溢出清零。

WDTOF 当看门狗发生超时，看门狗超时标志置位。该标志由软件清零。

WDINT 当看门狗发生超时，看门狗中断标志置位。产生的任何复位都会使该标志清零。一旦看门狗中断被服务，它会在 VIC 中禁能或将不限制地产生看门狗中断请求。

表 204 看门狗模式寄存器（WDMOD – 地址 0xE0000000）位描述

位	符号	描述	复位值
0	WDEN	WDEN 看门狗中断使能位（只能置位）	0
1	WDRESET	WDRESET 看门狗复位使能位（只能置位）	0
2	WDTOF	WDTOF 看门狗超时标志	0（仅在外部复位后）
3	WDINT	WDINT 看门狗中断标志（只读）	0
7:4	-	保留，用户软件不要向其写入 1。从保留位读出的值未被定义。	NA

18.4.2 看门狗定时器常数寄存器（WDTC - 0xE0000004）

WDTC 寄存器决定看门狗超时值。当喂狗时序产生时，WDTC 的内容重新装入看门狗定时器。它是一个 32 位寄存器，低 8 位在复位时设置为 1。写入一个小于 0xFF 的值会使 0xFF 装入 WDTC，因此超时的最小时间间隔为 $T_{PCLK} \times 256 \times 4$。

表 205 看门狗定时器常数寄存器（WDTC – 地址 0xE0000004）位描述

位	符号	描述	复位值
31:0	Count	看门狗超时间隔	0x0000 00FF

18.4.3 看门狗喂狗寄存器（WDFEED - 0xE0000008）

向该寄存器写入 0xAA，然后写入 0x55 会使 WDTC 的值重新装入看门狗定时器。如果看门狗通过 WDMOD 寄存器使能，该操作还将启动看门狗运行。置位 WDMOD 中的 WDEN 位不足以使能看门狗。在看门狗能够产生中断/复位之前，必须完成一次有效的喂狗时序。否则，看门狗将忽略喂狗错误。向 WDFEED 寄存器写入 0xAA 的下一个操作应当是向 WDFEED 寄存器写入 0x55，否则看门狗被触发。在一个喂狗时序中，一次对看门狗定时器寄存器不正确的访问之后第二个 PCLK 周期将产生中断/复位。

表 206 看门狗喂狗寄存器（WDFEED – 地址 0xE0000008）位描述

位	符号	描述	复位值
7:0	Feed	喂狗值应当为 0xAA，然后是 0x55。	NA

18.4.4 看门狗定时器值寄存器（WDTV - 0xE000000C）

WDTV 寄存器用于读取看门狗定时器的当前值。

表 207 看门狗定时器值寄存器（WDTV – 地址 0xE000000C）位描述

位	符号	描述	复位值
31:0	Count	计数器定时器值。	0x0000 00FF

18.5 方框图

看门狗方框图如图 60 所示。

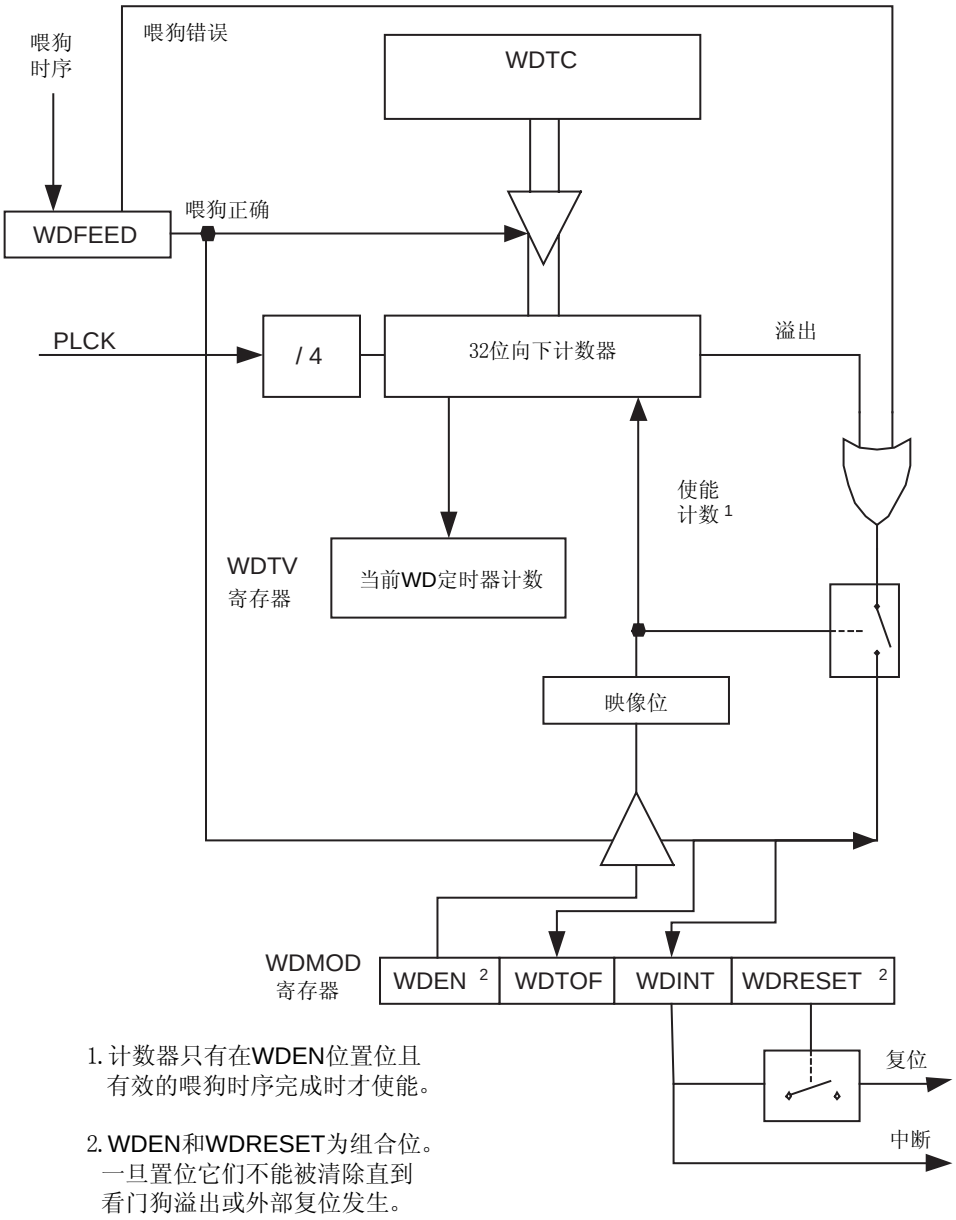


图 60 看门狗方框图

第19章 FLASH 存储器系统和编程

19.1 FLASH BOOT 装载程序

Boot 装载程序控制复位后的初始化操作，并提供实现 Flash 编程的方法。BOOT 装载器可启动对空片的编程、已编程器件的擦除和再编程以及在运行的系统中由应用程序对 Flash 存储器进行编程。

19.2 特性

- 在系统编程：在系统编程 (ISP)通过 boot 装载程序和串口对片内 Flash 存储器进行编程和再编程。当器件在终端用户板上使用时可实现该操作。
- 在应用中编程：最终用户代码直接执行在应用编程 (IAP)对片内 Flash 存储器进行擦除和编程操作。

19.3 应用

Flash boot 装载程序同时提供片内 Flash 存储器的 ISP 和 IAP 编程接口。

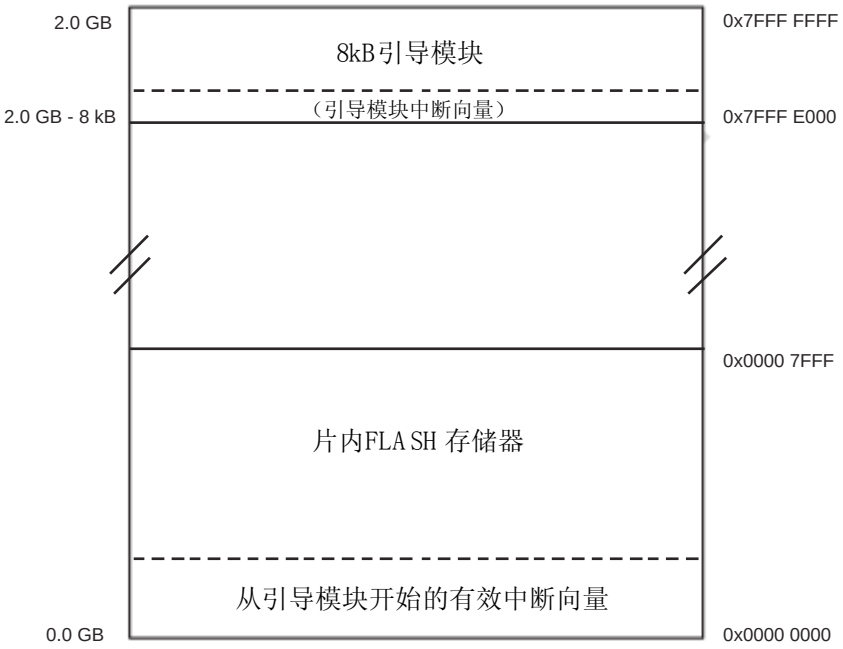
19.4 描述

Flash boot 装载程序代码在器件上电或复位时执行。装载程序可执行 ISP 命令处理程序或用户应用代码。复位后 P0.14 的低电平被认为是启动 ISP 命令处理器的外部硬件请求。假定在 $\overline{\text{RESET}}$ 脚产生上升沿时 X1 管脚上有正确的信号，在 P0.14 被采样之前最多为 3ms 并决定执行用户代码还是 ISP 处理程序。如果 P0.14 采样为低电平并且看门狗溢出标志置位，则启动 ISP 命令处理器的外部硬件请求将被忽略。如果没有请求用于 ISP 命令处理程序的执行（P0.14 复位后采样为高电平），那么将搜索有效的用户程序。如果找到有效的用户程序，执行的控制就转移给用户程序。如果没有找到有效的用户程序，那么就调用自动波特率程序。

管脚 P0.14 作为 ISP 硬件请求时要特别注意。由于 P0.14 在复位后处于高阻模式，用户需要提供外部硬件（上拉电阻或其它器件）使管脚处于一个确定的状态。否则可能导致非预期的进入 ISP 模式。

19.4.1 复位后的存储器映射

Boot block 规格为 8kB，它位于片内 Flash 存储器最顶端的部分（从 0x7FFF E000 开始）。ISP 和 IAP 程序使用部分片内 RAM。RAM 的使用在稍后讲述。中断向量位于片内 Flash 存储器的 Boot block 当中，它在复位后被激活，即 Boot block 的最低 64 字节也出现在地址 0x0000 0000 开始的存储器区域内。复位向量包含一条跳转指令，用来跳转到 Flash boot 装载程序的入口。



注：存储器区并不是按比例绘制的。

图 61 带 32kB Flash 存储器 LPC2103 复位后低地址存储器的映射

19.4.2 有效用户代码的判定标准

有效用户代码的判定标准：保留的 ARM 中断向量位置（0x0000 0014）应当包含剩余中断向量校验和的 2 的补码。这样就使所有向量的校验和为 0。boot 装载程序代码禁止中断向量在 boot block 内重叠，然后计算 Flash 扇区 0 当中的中断向量的校验和。如果结果匹配，那么通过将 0x0000 0000 装入程序计数器使执行控制权转移给用户代码。此后，用户 Flash 复位扇区应当包含一条跳转到用户代码的跳转指令。

如果结果不匹配，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，主机应当发送接收到的字符串（“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。如果通过验证，则向主机发送“OK<CR><LF>”。主机应当通过发送正在运行部分的晶振频率（单位为 kHz）作为响应。例如，如果运行在 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“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。要使自动波特率正确工作，晶振频率应当大于或等于 10MHz。Boot 代码没有使用片内 PLL。

在接收到晶振频率后，执行初始化并调用 ISP 命令处理器。处于安全性的考虑，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。其它命令不需要解锁命令就可执行。每次 ISP 命令处理都要执行一次“Unlock（解锁）”命令。解锁命令在 19.8 节“ISP 命令”中讲述。

19.4.3 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。多余的<CR>和<LF>将被忽略。所有 ISP 的响应都以<CR><LF>结束的字符串形式发送。数据以 UU 编码格式发送和接收。

19.4.4 ISP 命令格式

“命令 参数_0 参数_1 ... 参数_n<CR><LF>”“数据”（“数据”只适用于写命令）

19.4.5 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"（“数据”只适用于读命令）

19.4.6 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。发送器应当在发送 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相比较。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，发送器应当将字节重新发送。

UU编码的描述见 *http://www.wotsit.org*。

19.4.7 ISP 流控制

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(停止)使数据流停止。发送 ASCII 控制字符 DC1(启动)恢复数据流。主机也应支持相同的流控制机制。

19.4.8 ISP 命令中止

命令可通过发送 ASCII 控制字符“ESC”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 ESC 代码，ISP 命令处理器将等待一个新命令。

19.4.9 ISP 过程中的中断

在任何复位后，位于 Flash boot block 内的 boot block 中断向量都有效。

19.4.10 IAP 过程中的中断

在擦除/编程操作过程中，片内 Flash 存储器不可访问。当用户应用程序启动执行时，用户 Flash 区域的中断向量有效。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 代码不使用或禁止中断。

19.4.11 ISP 命令处理器使用的 RAM

ISP 命令使用片内地址 0x4000 0120 到 0x4000 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 编程命令使用片内 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，堆栈是向下增加的。

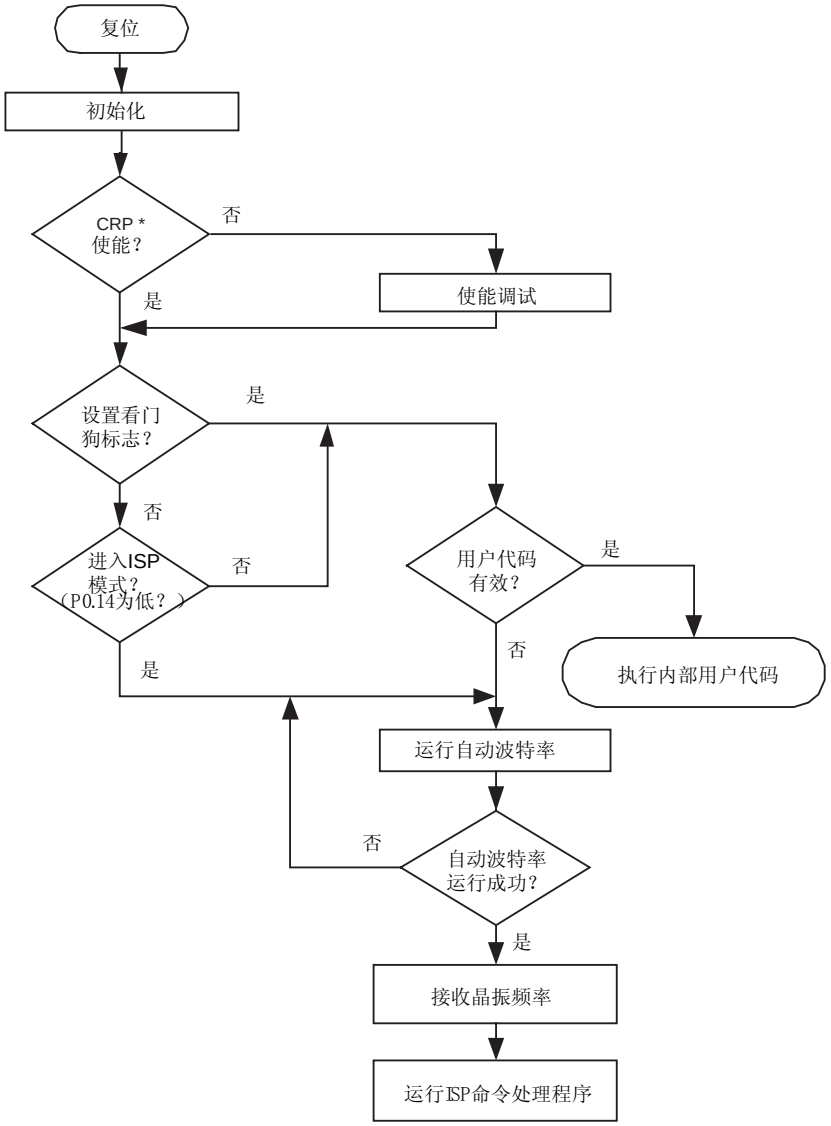
19.4.12 IAP 命令处理器使用的 RAM

Flash 编程命令使用片内 RAM 最顶端的 32 字节。用户可使用的最大堆栈为 128 字节，堆栈是向下增加的。

19.4.13 RealMonitor 使用的 RAM

RealMonitor 使用的片内 RAM 地址范围为 0x4000 0040~0x4000 011F。如果不需要基于 RealMonitor 的调试，用户可使用该区域。Flash boot 装载程序不初始化 RealMonitor 的堆栈。

19.4.14 BOOT 处理流程图



(1) 代码读保护

图 62　Boot 处理流程图

19.5 扇区数

有些 IAP 和 ISP 命令根据“扇区”进行操作并指定扇区数。LPC2101/02/03 器件分别包含 8、16 和 32kB 的 Flash 存储器，下表列出它们所包含的扇区数和存储器地址。IAP、ISP 和 RealMonitor 程序都位于 boot block。boot block 存在于所有器件的 0x7FFF E000 – 0x7FFF FFFF 中。ISP 和 IAP 命令不允许对 boot block 执行写/擦除/运行操作。在 LPC2101/02/03 微控制器操作下，用户可应用所有的 8/16/32kB Flash 存储器。

表 208 LPC2101,LPC2102,LPC2103 的 Flash 扇区

扇区号	扇区大小 [kB]	地址范围	LPC2101 (8kB)	LPC2102 (16kB)	LPC2103 (32kB)
0	4	0x0000 0000-0x0000 0FFF	+	+	+
1	4	0x0000 1000-0x0000 1FFF	+	+	+
2	4	0x0000 2000-0x0000 2FFF		+	+
3	4	0x0000 3000-0x0000 3FFF		+	+
4	4	0x0000 4000-0x0000 4FFF			+
5	4	0x0000 5000-0x0000 5FFF			+
6	4	0x0000 6000-0x0000 6FFF			+
7	4	0x0000 7000-0x0000 7FFF			+

19.6 FLASH 内容保护机制

LPC2101/02/03 配置有 Flash 存储器的错误校验码(ECC)。错误校验模块的用途有两部分。第一，它译码从存储器读取的数据字为输出数据字。第二，它编码数据字写入存储器。错误校验能力包含 Hamming 代码的单个位错误校验。

ECC 的操作可在运行应用中看出。ECC 自身的内容储存在 flash 存储器中，不能通过用户代码访问进行自身的读取或写入。ECC 的字节响应用户可访问 Flash 的每个连续的 128 位。因此，从 0x0000 0000 到 0x0000 0003 的 Flash 字节由第一个 ECC 字节保护，从 0x0000 0004 到 0x0000 0007 的 Flash 字节由第二个 ECC 字节保护，等等。

每当 CPU 请求读用户的 Flash，原始数据的 128 位含有指定的存储器单元和匹配的 ECC 字节。如果 ECC 机制在读取数据中检测到一个错误，校验将会在数据提供到 CPU 之前应用。当请求用户 Flash 的写操作时，用户指定内容的写操作可通过匹配计算的 ECC 值来实现并储存在 ECC 存储器中。

当用户 Flash 存储器的一个扇区被擦除时，相应的 ECC 字节也擦除。一旦写入一个 ECC 字节，它不能被更新除非它首先被擦除。因此，为了合适地操作 ECC 机制，必须以 4 个字节(或 4 的倍数)一组向 Flash 存储器写入数据，排列如上所述。

19.7 代码读保护(CRP)

代码读保护通过向 Flash 地址单元 0x1FC（用户 Flash 扇区 0）写入 0x8765 4321（十进制表示为 227 1560481）来使能。地址单元 0x1FC 允许为 FIQ 异常处理程序保留部分空间。当读保护被使能时，JTAG 调试端口、外部存储器引导和以下 ISP 命令将被禁能：

- 读存储器
- 写 RAM
- 运行
- 将 RAM 内容复制到 Flash

上述 ISP 命令终止时返回 CODE_READ_PROTECTION_ENABLED。

代码读保护使能时，ISP 擦除命令只允许擦除用户扇区的内容。这种限制是代码读保护不使能时所没有的。IAP 命令不受代码读保护的影响。

重要：一旦器件经过电源周期，代码读保护为有效/无效。

19.8 ISP 命令

下面的命令是 ISP 命令处理程序所接受的命令。每个命令都有具体的状态代码。当接收到未定义的命令时，命令处理程序发送返回代码 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 命令处理程序才发送 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 209 ISP 命令汇总

ISP 命令	使用	描述
解锁	U <解锁代码>	见表 210
设置波特率	B <波特率> <停止位>	见表 211
回声	A <设定>	见表 213
写 RAM	W <起始地址> <字节数>	见表 214
读存储器	R <地址> <字节数>	见表 215
准备写操作的扇区	P <起始扇区号> <结束扇区号>	见表 216
将 RAM 内容复制到 Flash	C <Flash 地址> <RAM 地址> <字节数>	见表 217
运行	G <地址> <模式>	见表 218
擦除扇区	E <起始扇区号> <结束扇区号>	见表 219
扇区查空	I <起始扇区号> <结束扇区号>	见表 220
读器件 ID	J	见表 221
读 Boot 代码版本	K	见表 223
比较	M <地址 1> <地址 2> <字节数>	见表 224

19.8.1 解锁 <解锁代码>

表 210 ISP 解锁命令描述

命令	U
输入	解锁代码：23130_{10}
返回代码	CMD_SUCCESS | INVALID_CODE | PARAM_ERROR
描述	该命令用于解锁 Flash 写、擦除和运行命令。
举例	“U 23130<CR><LF>”解锁 Flash 写/擦除&运行命令。

19.8.2 设置波特率 <波特率> <停止位>

表 211 ISP 设置波特率命令描述

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	该命令用于改变波特率。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。

表 212 ISP 波特率和外部晶体频率的关系（以 MHz 为单位）

ISP 波特率.vs. 外部晶体频率	9600	19200	38400	57600	115200	230400
10.0000	+	+	+			
11.0592	+	+		+		
12.2880	+	+	+			
14.7456	+	+	+	+	+	+
15.3600	+					
18.4320	+	+		+		
19.6608	+	+	+			
24.5760	+	+	+			
25.0000	+	+	+			

19.8.3 回声 <设定>

表 213 ISP 回声命令描述

命令	A
输入	设定: ON=1 | OFF=0
返回代码	CMD_SUCCESS | PARAM_ERROR
描述	回声命令的默认设定是 ON(打开)。当 ON(打开)时，ISP 命令处理器将接收到的串行数据发送回主机。
举例	"A 0<CR><LF>" 回声关闭。

19.8.4 写 RAM<起始地址> <字节数>

主机应当在接收到 CMD_SUCCESS 返回代码后发送数据。主机应当在发送 20 个 UU 编码行之后发送校验和。校验和通过增加原始数据（UU 编码前）字节而产生，发送完 20 个编码行后重新设置。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相比较。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，ISP 命令处理器响应“RESEND<CR><LF>”。作为响应，主机应当将字节重新发送。

表 214 ISP 写 RAM 命令描述

命令	W
输入	**起始地址：**被写 RAM 的起始地址，该地址应当以字为边界。 **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。当代码读保护使能时该命令被禁止。
举例	"W 1073742336 4<CR><LF>"向地址 0x4000 0200 写入 4 个字节数据。

19.8.5 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。校验和在增加原始数据（UU 编码前）字节时产生，发送完 20 个 UU 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相比较。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 命令处理器应当将字节重新发送。

表 215 ISP 读存储器命令描述

命令	R
输入	**起始地址**：被读出数据字节的地址，该地址应当以字为边界。 **字节数**：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。当代码读保护使能时该命令被禁止。
举例	"R 1073741824 4<CR><LF>"从地址 0x4000 0000 读出 4 个字节数据。

19.8.6 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 216 ISP 准备写操作的扇区命令描述

命令	P
输入	**起始扇区号** **结束扇区号**：应当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“将 RAM 内容复制到 Flash”或“擦除扇区”命令之前执行。这两个命令的成功执行会导致相关的扇区再次被保护。该命令不能用于 boot block。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。

19.8.7 将 RAM 内容复制到 Flash <Flash 地址> <RAM 地址> <字节数>

表 217 ISP 复制命令描述

命令	C
输入	**Flash 地址(DST)**: 要写入数据字节的目标 Flash 地址。目标地址的边界应当为 256 字节。 **RAM 地址(SRC)**: 读出数据字节的源 RAM 地址。 **字节数**: 写入字节的数目。应当为 256 |512 | 1024 | 4096。
返回代码	CMD_SUCCESS | SRC_ADDR_ERROR (地址不以字为边界) | 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_ENABLE
描述	该命令用于编程 Flash 存储器。“准备写操作的扇区”命令应当在该命令之前被执行。当成功执行复制命令后，扇区将自动受到保护。该命令不能写 bootblock。当代码读保护使能时该命令被禁止。
举例	"C 0 1073774592 512<CR><LF>"将 RAM 地址 0x4000 8000 开始的 512 字节复制到 Flash 地址 0。

19.8.8 运行<地址><模式>

表 218 ISP 运行命令描述

命令	G
输入	**地址**：代码执行起始的 Flash 或 RAM 地址。该地址应当以字为边界。 **模式**：T（执行 Thumb 模式下的程序）| A（执行 ARM 模式下的程序）
返回代码	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLE
描述	该命令用于执行位于 RAM 或 Flash 存储器当中的程序。一旦成功执行该命令，就有可能不再返回 ISP 命令处理程序。当代码读保护使能时该命令被禁止。
举例	"G 0 A<CR><LF>"跳转到 ARM 模式下的地址 0x0000 0000 处。

19.8.9 擦除扇区<起始扇区号><结束扇区号>

表 219 ISP 擦除扇区命令描述

命令	E
输入	**起始扇区号** **结束扇区号**：应当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_ENABLE
描述	该命令用于擦除片内 Flash 存储器的一个或多个扇区。Boot block 不能由该命令擦除。当代码读保护使能时，该命令只允许擦除所有用户扇区的内容。
举例	"E 2 3<CR><LF>"擦除 Flash 扇区 2 和 3。

19.8.10 扇区查空<起始扇区号><结束扇区号>

表 220 ISP 扇区查空命令描述

命令	I
输入	**起始扇区号** **结束扇区号**：应当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	该命令用于对片内 Flash 存储器的一个或多个扇区进行查空。 **由于扇区 0 的前 64 字节重新映射到 Flash boot block，因此对其进行查空一定会失败。**
举例	"I 2 3<CR><LF>"对 Flash 扇区 2 和 3 进行查空。

19.8.11 读器件标识号

表 221 ISP 读器件标识号命令描述

命令	J
输入	无
返回代码	CMD_SUCCESS 后跟 ASCII 格式的 ID 号（见表 222）。
描述	该命令用于读取器件的标识号。

表 222 LPC2101/02/03 器件标识号

器件	ASCII/dec 编码	Hex 编码
LPC2103	327441	0x0004 FF11

19.8.12 读 Boot 代码版本号

表 223 ISP 读 Boot 代码版本号命令描述

命令	K
输入	无
返回代码	CMD_SUCCESS 后跟 2 字节 ASCII 格式的 boot 代码版本号。将其解释为<字节 1 (主)>.<字节 0 (次)>
描述	该命令用于读取 boot 代码版本号。

19.8.13 比较<地址 1><地址 2><字节数>

表 224 ISP 比较命令描述

命令	M
输入	**地址 1(DST)**: 要比较数据字节的起始 Flash 或 RAM 地址。该地址应当以字为边界。 **地址 2(SRC)**: 要比较数据字节的起始 Flash 或 RAM 地址。该地址应当以字为边界。 **字节数**: 待比较的字节数. 计数值应当为 4 的倍数。
返回代码	CMD_SUCCESS | (源和目标数据相等) COMPARE_ERROR | (后跟第一个不匹配字节的偏移值) COUNT_ERROR (字节数不是 4 的倍数) | ADDR_ERROR | ADDR_NOT_MAPPED | PARAM_ERROR |
描述	该命令用来比较两个地址单元的存储器内容。**当源或目标地址包含从地址 0 开始的前 64 字节中的任意一个地址时，比较的结果可能不正确。因为前 64 字节重新映射到 Flash boot 扇区。**
举例	"M 8192 1073741824 4<CR><LF>"将 RAM 地址 0x4000 0000 开始的 4 个字节与 Flash 地址 0x2000 开始的 4 个字节相比较。

19.8.14 ISP 返回代码

表 225 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。只有当主机发出的命令被成功执行完毕后，才由 ISP 处理程序发送。
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	源地址没有位于存储器映射中。计数值必须考虑可用性。
5	DST_ADDR_NOT_MAPPED	目标地址没有位于存储器映射中。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大于起始扇区号。
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	地址不在存储器映射中映射。计数值必须考虑可用性。
15	CMD_LOCKED	命令被锁定。
16	INVALID_CODE	解锁代码无效。
17	INVALID_BAUD_RATE	无效波特率设定。
18	INVALID_STOP_BIT	无效停止位设定。
19	CODE_READ_PROTECTION_ENABLE	代码读保护使能。

19.9 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 所指向的结果表。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 63。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，传递到“将 RAM 内容复制到 Flash”命令。结果的最大数目为 2，由“扇区查空”命令返回。命令处理程序在接收到一个未定义的命令时发送状态代码 INVALID_COMMAND。IAP 程序是 thumb 代码，位于地址 0x7FFF FFF0。

IAP 功能可用下面的 C 代码来调用。

定义 IAP 程序的入口地址。由于 IAP 地址的第 0 位是 1，因此，当程序计数器转移到该地址时会引起 Thumb 指令集的变化。

```
#define  IAP_LOCATION 0x7ffffff1
```

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 result[2];
```

或

```
unsigned long * command;
unsigned long * result;
command = (unsigned long *) 0x….
result = (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 将函数结果和 R1 中的表格基址一同返回。

```
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 , result);
```

使用 ADS（ARM 开发套件）的 ARM 链接器支持的符号定义文件可以进一步简化 IAP 的调用。用户还可使用汇编程序来调用 IAP 程序。

下面的符号定义可用于链接 IAP 程序和用户代码：

```
#<SYMDEFS># ARM Linker, ADS1.2 [Build 826]: Last Updated: Wed May 08 16:12:23 2002
0x7fffff90 T rm_init_entry
0x7fffffa0 A rm_undef_handler
0x7fffffb0 A rm_prefetchabort_handler
0x7fffffc0 A rm_dataabort_handler
0x7fffffd0 A rm_irqhandler
0x7fffffe0 A rm_irqhandler2
0x7ffffff0 T iap_entry
```

根据 ARM 规范（ARM Thumb 过程调用标准 SWS ESPC 0002 A-05），r0, r1, r2 和 r3 寄存器能够传递最多 4 个参数。另外的参数通过堆栈传递。最多有 4 个参数可以返回 r0, r1, r2 和 r3 寄存器。另外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 编译器的差异而产生问题。建议的参数传递机制降低了这样的风险。

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 命令使用片内 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 226 IAP 命令汇总

IAP 命令	命令代码	描述
准备编程扇区	50_{10}	见表 227
将 RAM 内容复制到 Flash	51_{10}	见表 228
擦除扇区	52_{10}	见表 229
扇区查空	53_{10}	见表 230
读器件 ID	54_{10}	见表 231
读 boot 代码版本	55_{10}	见表 232
比较	56_{10}	见表 233
重新调用 ISP	57_{10}	见表 234

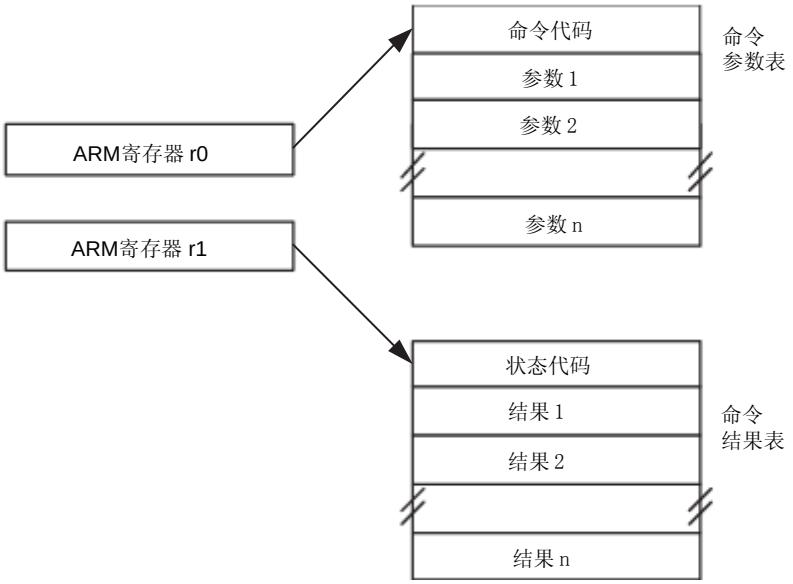


图 63 IAP 参数传递

19.9.1 准备编程扇区

该命令使 Flash 写/擦除操作分两步执行。

表 227 IAP 准备编程扇区命令描述

命令	准备编程扇区
输入	**命令代码：50_{10}** **参数 0**：起始扇区号 **参数 1**：结束扇区号（应当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”命令之前执行。这两个命令的成功执行会导致相关的扇区再次被保护。该命令不能用于 boot 扇区。要准备单个扇区，可将起始和结束扇区号设置为相同值。

19.9.2 将 RAM 内容复制到 Flash

表 228 IAP 将 RAM 内容复制到 Flash 命令描述

命令	将 RAM 内容复制到 Flash
输入	**命令代码：51_{10}** **参数 0 (DST)：**要写入数据字节的目标 Flash 地址。目标地址的边界应当为 256 字节。 **参数 1 (SRC)：** 读出数据字节的源 RAM 地址。该地址应当以字为边界。 **参数 2：**写入字节的数目。应当为 256 |512 | 1024 | 4096。 **参数 3：**系统时钟频率（CCLK）(单位：KHz)
返回代码	CMD_SUCCESS | SRC_ADDR_ERROR (地址不以字为边界) | 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	该命令用于编程 Flash 存储器。受影响的扇区应当先通过调用“准备写操作的扇区”命令准备。当成功执行复制命令后，受影响的扇区将再次自动受到保护。该命令不能写 boot 扇区。

19.9.3 擦除扇区

表 229 IAP 擦除扇区命令描述

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（应当大于或等于起始扇区号）。 **参数 2**：系统时钟频率（CCLK）(单位：KHz)
返回代码	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR |
结果	无
描述	该命令用于擦除片内 Flash 存储器的一个或多个扇区。boot 扇区不能由该命令擦除。要擦除单个扇区可将起始和结束扇区号设定为相同值。

19.9.4 扇区查空

表 230 IAP 扇区查空命令描述

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（应当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	该命令用于对片内 Flash 存储器的一个或多个扇区进行查空。要查空单个扇区可将起始和结束扇区号设定为相同值。

19.9.5 读器件标识号

表 231 IAP 读器件标识号命令描述

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（详见表 222“LPC2101/02/03 器件标识号”）。
描述	该命令用于读取器件的标识号。

19.9.6 读 Boot 代码版本号

表 232 IAP 读 Boot 代码版本号命令描述

命令	读 Boot 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 boot 代码版本号（ASCII 格式）。将其解释为<字节 1 (主)>.<字节 0 (次)>
描述	该命令用于读取 boot 代码版本号。

19.9.7 比较 <地址 1><地址 2> <字节数>

表 233 IAP 比较命令描述

命令	比较
输入	**命令代码：56_{10}** **参数 0 (DST)**：要比较数据字节的起始 Flash 或 RAM 地址。该地址应当以字为边界。 **参数 1 (SRC)**：要比较数据字节的起始 Flash 或 RAM 地址。该地址应当以字为边界。 **参数 2**：待比较的字节数。计数值应当为 4 的倍数。
返回代码	CMD_SUCCESS | COMPARE_ERROR | COUNT_ERROR (字节数不是 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地址。
描述	该命令用来比较两个地址单元的存储器内容。**当源或目标地址包含从地址 0 开始的前 64 字节中的任意一个地址时，比较的结果可能不正确。前 64 字节可重新映射到 RAM。**

19.9.8 重新调用 ISP

表 234 重新调用 ISP

命令	比较
输入	**命令代码：57_{10}**
返回代码	无
结果	**无**
描述	该命令用于调用 ISP 模式下的引导装入。该命令映射引导向量，配置 P0.1 为输入并在进入 ISP 模式之前设置 APB 分频器寄存器为 0。当一个有效的用户程序出现在内部 flash 存储器且 P0.14 脚不可进入来强制 ISP 模式时，可使用该命令。该命令不禁止 PLL，因此当器件脱离 PLL 运行时该命令可调用引导装入。在这些情况下，ISP 程序应在自动波特率握手后传输 PLL 频率。另一种选择是在调用 IAP 之前禁止 PLL。 **重要**：在使用“重新调用 ISP”命令之前，必须用复位值编程定时器 1 寄存器。

19.9.9 IAP 状态代码

表 235 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	源地址没有位于存储器映射中。计数值必须考虑可用性。
5	DST_ADDR_NOT_MAPPED	目标地址没有位于存储器映射中。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。

19.10 JTAG Flash 编程接口

调试工具可以将 Flash 映像部分写入 RAM，然后根据正确的偏移重复执行 IAP 调用“将 RAM 内容复制到 Flash”。

第20章 EmbeddedICE 逻辑

20.1 特性

- 通过软件调试器启动调试会话，不需要目标资源
- 允许软件调试器通过 JTAG 端口直接与内核进行对话
- 在 ARM7TDMI-S 内核中直接插入指令
- 根据插入不同类型的指令对 ARM7TDMI-S 内核或系统状态进行检查、保存或修改
- 允许指令在低调试速度或高系统速度下执行

20.2 应用

EmbeddedICE 逻辑提供对片内调试的支持。对目标系统进行调试需要一个主机来运行调试软件和 EmbeddedICE 协议转换器。EmbeddedICE 协议转换器将远程调试协议命令转换成所需要的 JTAG 数据，从而对目标系统上的 ARM7TDMI-S 内核进行访问。

20.3 描述

ARM7TDMI-S 调试结构使用现有的 JTAG [1] 端口来访问内核。供产品测试用的扫描链在调试状态下重新用来捕获数据总线上的信息并向内核或存储器插入新的信息。在 ARM7TDMI-S 当中有两个 JTAG 类型的扫描链。一个 JTAG 类型的测试访问端口控制器控制扫描链。除了扫描链之外，调试结构还使用位于 ARM7TDMI-S 核内部的 EmbeddedICE 逻辑。EmbeddedICE 使用自身的扫描链向 ARM7TDMI-S 内核插入观察点和断点。EmbeddedICE 逻辑包含 2 个实时观察点寄存器和 1 个控制和状态寄存器。这两个观察点寄存器或其中的一个可编程为暂停 ARM7TDMI-S 内核。当编程到 EmbeddedICE 逻辑中的值与当前出现在地址总线、数据总线和某些控制信号上的值匹配时，内核的运行将暂停。可以屏蔽任何位使其不会影响比较操作。观察点寄存器可以配置为观察点（即对于数据的访问）或断点（指令取指）。观察点和断点可以按照下面的方式进行组合：

- 在停止 ARM7TDMI 内核之前，必须满足观察点的两个条件。CHAIN 的功能要求在暂停内核之前满足两个连续的条件。例如，将第一个断点设定为在访问外设时触发，而第二个断点在执行任务切换的代码段时触发。当断点触发时，与任务切换有关的信息准备就绪以备检查。
- 可以配置观察点以使在一段地址范围内观察点有效。RANGE 功能允许实现一个组合的断点，例如在访问存储器最低 256 字节但不访问最低 32 字节时产生断点。

ARM7TDMI-S 内核有一个内置的调试通信通道功能。调试通信通道允许程序在目标系统上运行，即使进入调试状态，目标系统程序与主机调试器或其它独立的主机进行通信时也不会中断程序流程。ARM7TDMI-S 内核上运行的程序将调试通信通道作为协处理器 14 进行访问。调试通信通道允许 JTAG 端口发送和接收数据，但不影响正常的程序流程。调试通信通道数据和控制寄存器映射到 EmbeddedICE 逻辑中的地址。

1. 详见 IEEE 标准 1149.1-1990《标准测试访问端口和边界扫描结构》。

20.4 管脚描述

表 236 EmbeddedICE 管脚描述

管脚名称	类型	描述
TMS	输入	**测试模式选择。**TMS 管脚选择 TAP 状态机中的下一个状态。
TCK	输入	**测试时钟。**该管脚允许 TMS 和 TDI 管脚上数据的转换。它是一个上升沿触发时钟，由 TMS 和 TCK 信号定义器件的内部状态。
TDI	输入	**测试数据输入。**移位寄存器的串行数据输入端。
TDO	输出	**测试数据输出。**移位寄存器的串行数据输出端。器件中的数据在 TCK 信号的下降沿移出。
TRST	输入	**测试复位。**TRST 管脚可用于复位 EmbeddedICE 逻辑中的测试逻辑。
DBGSEL	输入	**调试选择。**当复位时该管脚为低电平，P0.27-P0.31 脚通过管脚连接模块配置为其中一种使用。当复位时该管脚为高电平，则进入调试模式。 对于 DBGSEL 提供的功能，请见 20.8 节“DEBUG 模式”。
RTCK	输出	**返回的测试时钟。**叠加到 JTAG 端口的额外信号。 基于 ARM7TDMI-S 处理器内核进行设计时需要该信号。Multi-ICE（ARM 的开发系统）使用该信号来保持与低或宽范围时钟频率的目标系统的同步。详见“Multi-ICE 系统设计注意事项应用注释 72（ARM DAI 0072A）”。同时在进入调试模式中使用。

20.5 复用管脚的复位状态

LPC2101/02/03 器件的 TMS, TCK, TDI, TDO, AND TRST 管脚是 P0.27-P0.31 的复用功能。要使它们用作调试端口，则在 V_{SS} 和 RTCK 管脚之间连接一个弱偏置电阻（4.7 - 10kΩ，取决于外部 JTAG 电路）。如果它们用作 GPIO 管脚，无需连接偏置电阻，并且确保了任何连接到管脚 26(RTCK)的外部驱动器在复位时驱动为高或处于高阻态。

20.6 寄存器描述

EmbeddedICE逻辑包含 16 个寄存器，见表 237。有关ARM7TDMI-S调试结构的详细描述见ARM有限公司出版的“ARM7TDMI-S(rev 4)技术参考手册”（ARM DDI 0234A），可从网站 http://www.arm.com下载。

表 237 EmbeddedICE 逻辑寄存器

名称	宽度	描述	地址
调试控制	6	强制调试状态，禁止中断	00000
调试状态	5	调试状态	00001
调试通信控制寄存器	32	调试通信控制寄存器	00100
调试通信数据寄存器	32	调试通信数据寄存器	00101
观察点 0 地址值	32	保存观察点 0 地址值	01000
观察点 0 地址屏蔽	32	保存观察点 0 地址屏蔽	01001
观察点 0 数据值	32	保存观察点 0 数据值	01010
观察点 0 数据屏蔽	32	保存观察点 0 数据屏蔽	01011
观察点 0 控制值	9	保存观察点 0 控制值	01100
观察点 0 控制屏蔽	8	保存观察点 0 控制屏蔽	01101

续上表

名称	宽度	描述	地址
观察点 1 地址值	32	保存观察点 1 地址值	10000
观察点 1 地址屏蔽	32	保存观察点 1 地址屏蔽	10001
观察点 1 数据值	32	保存观察点 1 数据值	10010
观察点 1 数据屏蔽	32	保存观察点 1 数据屏蔽	10011
观察点 1 控制值	9	保存观察点 1 控制值	10100
观察点 1 控制屏蔽	8	保存观察点 1 控制屏蔽	10101

20.7 方框图

调试环境的方框图如图 64 所示。

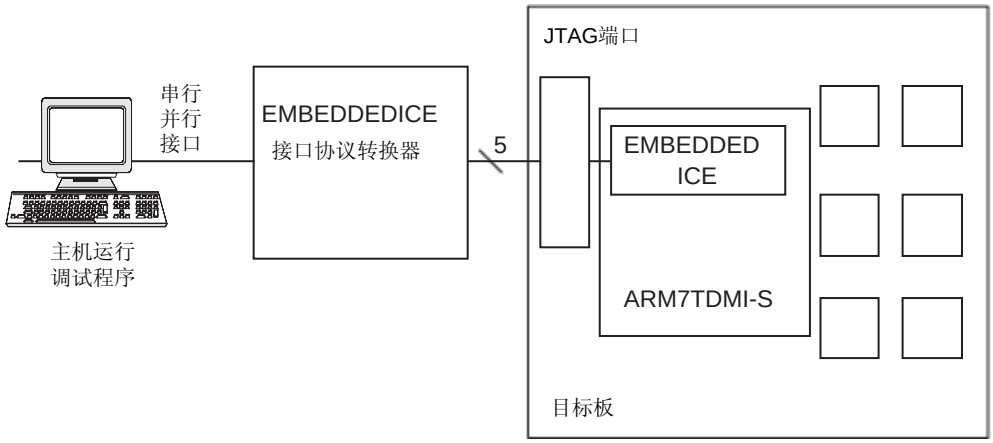


图 64　EmbeddedICE 调试环境方框图

20.8 DEBUG 模式

调试模式连接 JTAG 管脚到嵌入式 ICE 以便于使用仿真器或其它开发套件来编程调试。

20.8.1 使能调试模式

调试模式通过使用 DBGSEL 和 RTCK 管脚使能。

为了使能调试模式，DBGSEL 必须在 CPU 复位期间和复位后保持为高电平。对于正常的操作（非调试），DBGSEL 必须在任何时候保持为低电平（见图 65）。

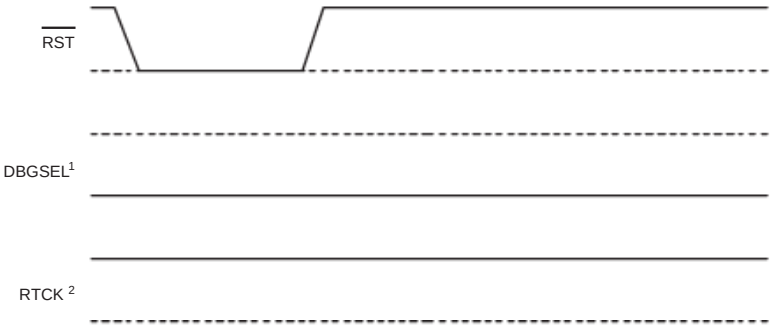
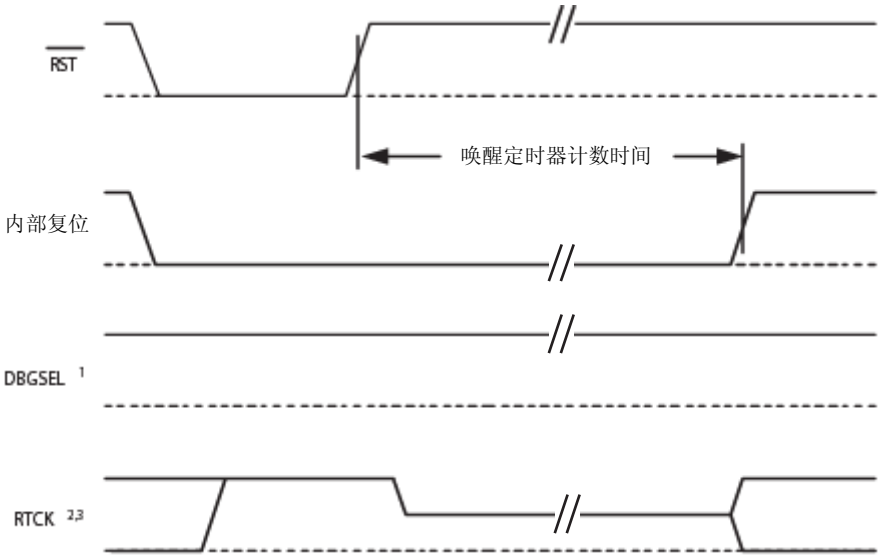


图 65　正常操作的波形（在非调试模式下）

对于带有 JTAG 管脚的调试，RTCK 必须在 $\overline{\text{RST}}$ 管脚释放时为高电平（见图 66）。RTCK 可通过其片内上拉被外部驱动为高电平或悬浮在高电平状态。RTCK 输出驱动器被禁能直至内部唤醒时间超时，允许 RTCK 必要时由外部信号驱动的过程中在释放外部复位和释放内部复位之间的间隔。

这个过程设置 P0.27-P0.31 管脚作为 JTAG 测试/调试接口。如果 P0.27-P0.31 管脚初始化为 JTAG 管脚，则管脚连接模块的设定对它们没有影响。

关于硬件不考虑与 DBGSEL 和 RTCK 相关的影响，请见 6.2 节“LPC2101/02/03 管脚描述”的表 58。



（1）　DBGSEL 必须为高电平。

（2）　RTCK 必须在 $\overline{\text{RST}}$ 管脚释放时为高电平。如果 RTCK 被外部拉低，则内部上拉将使 RTCK 为高电平。

（3）　当内部芯片复位通过唤醒定时器释放时，RTCK 输出驱动器将会打开。

图 66　使用主要 JTAG 管脚调试模式的波形

20.8.2 JTAG 管脚选择

仅当 DBGSEL 或 RTCK 脚在复位期间为高电平时，可选择主要的 JTAG 端口用来调试（见图 66）。如果复位时至少有一条 DBGSEL 或 RTCK 线为低，则 JTAG 将不会被使能且不能用作后面的调试。

第21章 RealMonitor

RealMonitor 是一个可配置的软件模块，它由 ARM 公司开发，可以提供实时的调试。本章所描述的信息摘自 ARM 文档 *RealMonitor 目标集成指南*（ARM DUI 0142A）。该器件包含一个编程到片内 ROM boot 存储器内的 RealMonitor 软件，用于特定的配置。

请参考白皮书“*片上系统的实时调试*”，可从 http://www.arm.com/support/White_Papers?Open Document 下载。

21.1 特性

- 允许用户在不暂停或复位系统的情况下，与当前运行的系统建立调试会话。
- 在其它用户应用代码调试过程中，允许用户的实时中断代码连续执行。

21.2 应用

实时调试。

21.3 描述

RealMonitor 是一个轻巧的调试监视器，它允许在用户调试前台应用程序时对中断进行服务。它通过 DCC(调试通信通道)与主机进行通信，DCC 位于 EmbeddedICE 逻辑当中。RealMonitor 比 ARM 系统中传统的调试方法更具优势。传统的调试方法包括：

- Angel（基于目标的调试监视器）
- Multi-ICE 或其它 JTAG 单元和 EmbeddedICE 逻辑（基于硬件的调试方案）

尽管这两种方法都提供健壮的调试环境，但并不适合作为一个轻巧的实时监视器。

Angel 设计成可以装载和调试在各种不同模式下独立运行的应用程序，它通过不同的连接与调试主机进行通信（例如串口或者以太网）。Angel 要求保存和恢复所有的处理器上下文，这样做的结果是使中断产生延迟。Angel 作为一个全功能的基于目标的调试器，对于执行实时监视来说显得过于笨重了。

Multi-ICE 是一种硬件调试方案。它使用内置在大多数 ARM 处理器中的 EmbeddedICE 单元来执行操作。为了执行访问存储器或处理器寄存器这样的调试任务，Multi-ICE 必须使内核进入调试状态。处理器处于调试状态的时间可能长达数百万个周期，这样正常的程序执行被挂起，中断也无法执行。

RealMonitor 结合了 Angel 和 Multi-ICE 的特性和机制，它提供了必需的服务和功能。特别是，它还包含了 Multi-ICE 的通信机制（使用 JTAG 的 DCC）和类似 Angel 的保存和恢复处理器上下文。RealMonitor 被预先编程在片内 ROM 存储器当中（boot 扇区）。当用户将其使能后，可以在部分应用程序继续运行时进行观察和调试。详见 21.4 节“如何使能 RealMonitor”的内容。

21.3.1 RealMonitor 部件

如图 67 所示，RealMonitor 分成两个功能部件：

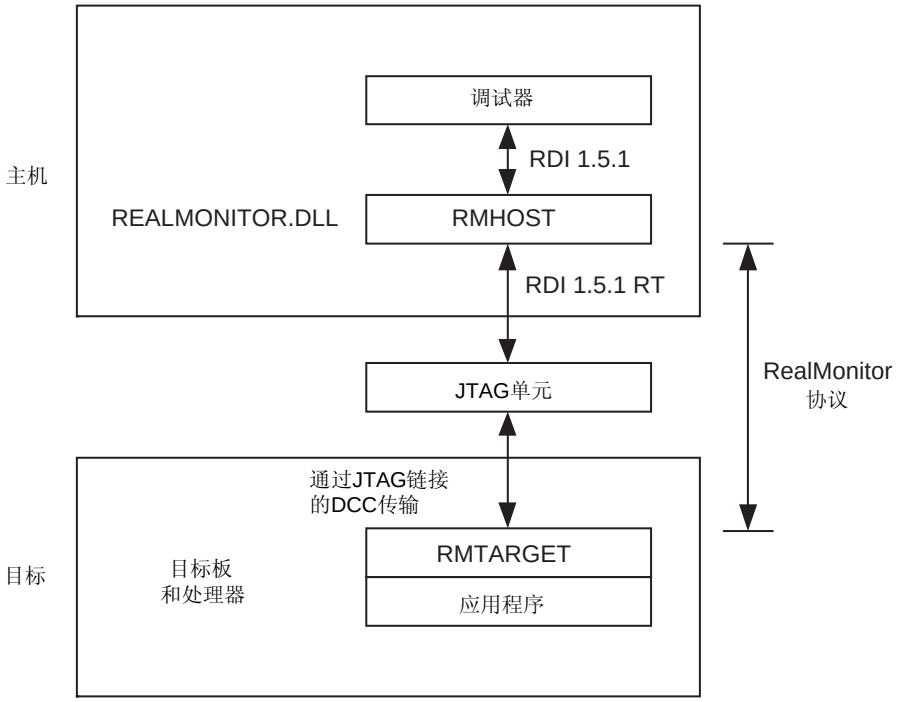


图 67 RealMonitor 部件

21.3.2 RMHost

RMHost 位于调试器和 JTAG 单元之间。RMHost 控制器和 RealMonitor.dll 将通用的远程调试接口（RDI）请求从调试器转换为 JTAG 单元的仅为 DCC 的 RDI 信息。有关主机 RealMonitor 集成应用调试的完整信息，请参考“ARM RMHost 用户指南”（ARM DUI 0137A）。

21.3.3 RMTarget

这部分预先编程到片内 ROM 存储器（boot 扇区）并在目标硬件上运行。它使用 EmbeddedICE 逻辑并通过 DCC 与主机进行通信。有关 RMTarget 功能的详细信息见 RealMonitor 目标集成指南（ARM DUI 0142A）。

21.3.4 RealMonitor 是如何工作的

通常情况下，RealMonitor 作为一个状态机，如图 68 所示。为了响应主机接收到的包，或者因为目标板上的异步事件，RealMonitor 在运行和停止状态之间进行切换。RMTarget 一次只支持一个断点、观察点、停止或半主机 SWI 的触发。不提供嵌套事件的保存和恢复。因此，如果用户应用程序因为一个断点而停止，而在 IRQ 处理程序中产生了另一个断点，那么 RealMonitor 进入 Panic 状态。RealMonitor 进入该状态后将不执行任何调试。

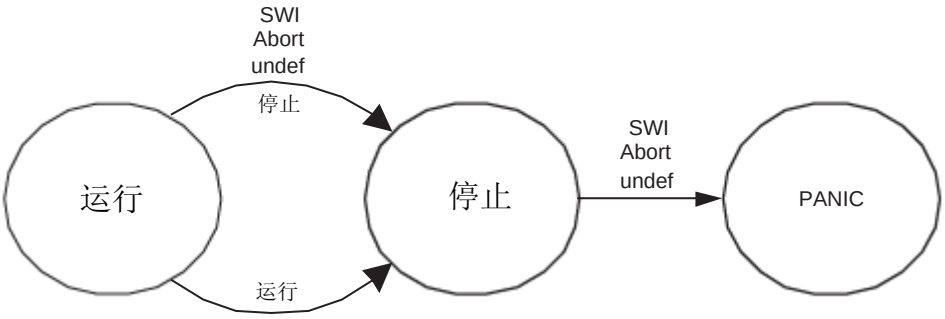


图 68 作为状态机的 RealMonitor

一个运行在主计算机上的调试器，例如 ARM eXtended 调试器（AXD）或其它 RealMonitor 调试器可以连接到目标，实现命令的发送和数据的接收。主机和目标之间的通信如图 67 所示。

RealMonitor 的目标部件 RMTarget 与主机部件 RMHost 通过调试通信通道（DCC）进行通信。DCC 通过 JTAG 连接来传递数据。

当用户应用程序运行时，RMTarget 通常使用 DCC 所产生的 IRQ。这意味着，如果用户应用程序也打算使用 IRQ，它必须将 DCC 产生的中断传递给 RealMonitor。

为了实现不间断的调试，处理器中的 EmbeddedICE-RT 逻辑在接收到一个断点时产生一个预取指中止异常或在接收到观察点时产生一个数据中止异常。这些异常由 RealMonitor 异常处理程序进行处理，并由调试器告知用户。这样用户应用程序可以在不停止处理器的情况下继续运行。RealMonitor 认为用户应用程序包含下面两部分：

- 连续运行的前台应用程序，通常处于用户、系统或 SVC 模式。
- 包含中断和异常处理程序的后台应用程序，由用户系统的特定事件触发，这些事件包括：

  －IRQ 或 FIQ
  －由用户前台应用程序产生的数据和预取指中止，表示应用程序在调试时出现错误。这两种情况都会通知主机并停止用户应用程序。
  －由用户前台程序中的未定义指令所导致的未定义异常，表示在调试程序时出现错误。RealMonitor 使用户程序一直停止直到从主机接收到一个“Go”包为止。

当一个不是由用户应用程序处理的异常发生时，执行下面的操作：

- RealMonitor 进入查询 DCC 的一个循环。如果 DCC 读缓冲区已满，控制权传递给 rm_ReceiveData()（RealMonitor 内部函数）。如果 DCC 写缓冲区空闲，控制权传递给 rm_TransmitData()（RealMonitor 内部函数）。如果没有别的事要做，函数返回调用程序。上述比较的顺序使读 DCC 的优先级高于写通信链接。
- RealMonitor 停止前台应用程序。如果 IRQ 和 FIQ 在前台程序停止时已经使能，那么 IRQ 和 FIQ 可以继续得到服务。

21.4 如何使能 RealMonitor

必须执行下面的步骤才可使能 RealMonitor。在这一节的末尾给出了执行所有步骤的例程。

21.4.1 增加堆栈

用户必须确保在 RealMonitor 所使用的每一个处理器模式下的应用程序中都建立了堆栈。对于每一种模式，RealMonitor 都要求一个固定数目字的堆栈空间。用户必须为 RealMonitor 和应用程序提供足够的堆栈空间。

RealMonitor 对堆栈有下列要求：

表 238 RealMonitor 的堆栈要求

处理器模式	RealMonitor 堆栈使用（字节）
未定义	48
预取指中止	16
数据中止	16
IRQ	8

21.4.2 IRQ 模式

该模式下的堆栈是必不可少的。RealMonitor 用两个字保存中断处理程序的入口。在嵌套中断使能前将它们释放。

21.4.3 未定义模式

该模式的堆栈也是必要的。RealMonitor 在处理一个未定义指令异常时使用 12 个字。

21.4.4 SVC 模式

RealMonitor 不使用该堆栈。

21.4.5 预取指中止模式

RealMonitor 使用 4 个字保存预取指中止中断处理程序的入口。

21.4.6 数据中止模式

RealMonitor 使用 4 个字保存数据中止中断处理程序的入口。

21.4.7 用户/系统模式

RealMonitor 不使用该堆栈。

21.4.8 FIQ 模式

RealMonitor 不使用该堆栈。

21.4.9 处理异常

这一节讲述 RealMonitor 和用户程序共用异常处理程序的重要性。

21.4.10 RealMonitor 异常处理

为了正常工作，RealMonitor 必须能够截获特定的中断和异常。图 69 所示为如何由 RealMonitor 自身声明异常或与应用程序共用异常处理程序。如果用户应用程序要求共用异常，那么它必须提供函数（例如 *app_IRQDispatch( )*）。根据异常的特性，该处理程序可以：

- 将控制权传递给 RealMonitor 处理程序，例如 rm_irqhandler2( )
- 为应用程序自身声明异常，例如 app_IRQHandler( )

一个自身不带异常处理程序的应用程序可以安装 RealMonitor 低级异常处理程序，该程序直接指向处理器的向量表。IRQ 处理程序必须得到向量中断控制器的地址。最简单的方法是在向量表中写一条转移指令(<地址>)，转移指令的目标地址为相关 RealMonitor 异常处理程序的起始地址。

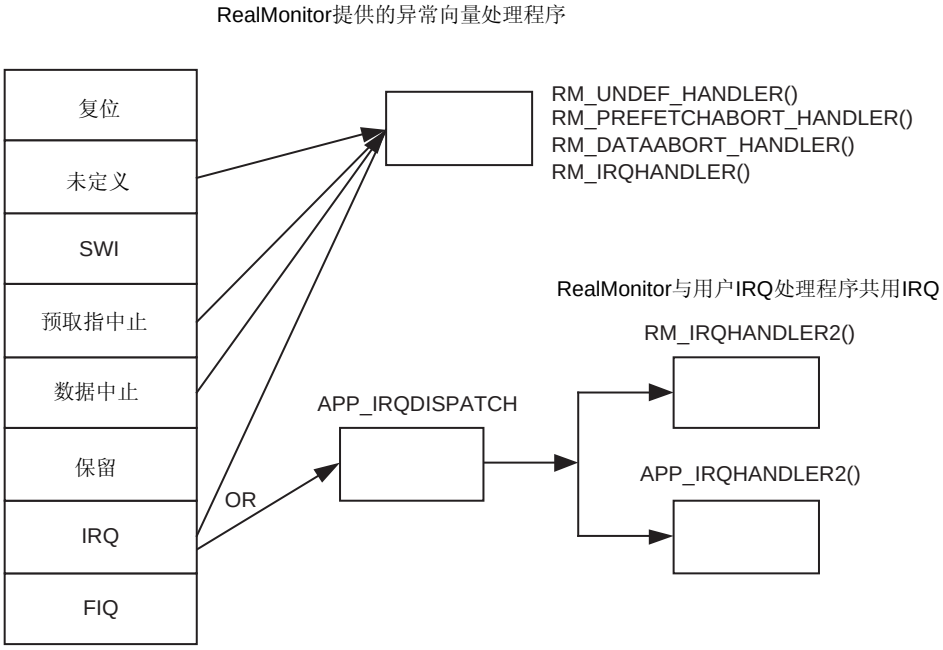


图 69 异常处理程序

21.4.11 RMTarget 初始化

当处理器处于特权模式并且 IRQ 禁止时，用户必须在应用程序的初始化代码中加入一行指令来调用 rm_init_entry( )。

21.4.12 例程

下面的例子显示了如何建立堆栈、VIC、初始化 RealMonitor 以及共用非向量中断：

```
IMPORT  rm_init_entry
IMPORT  rm_prefetchabort_handler
IMPORT  rm_dataabort_handler
IMPORT  rm_irqhandler2
IMPORT  rm_undef_handler
IMPORT  User_Entry                          ; 用户应用程序入口
CODE32
```

```
    ENTRY
    ; 定义异常表。指令连接器将代码放置在地址 0x0000 0000。

    AREA    exception_table, CODE
    LDR     pc, Reset_Address
    LDR     pc, Undefined_Address
    LDR     pc, SWI_Address
    LDR     pc, Prefetch_Address
    LDR     pc, Abort_Address
    NOP                                         ; 在此处插入用户代码有效签名
    LDR     pc, [pc, #-0xFF0]                   ; 从 VIC 装载 IRQ 向量
    LDR     PC, FIQ_Address

Reset_Address       DCD     __init              ; 复位入口
Undefined_Address   DCD     rm_undef_handler    ; 由 RealMonitor 提供
SWI_Address         DCD     0                   ; 用户可将 SWI 处理程序的地址放置在此处
Prefetch_Address    DCD     rm_prefetchabort_handler ; 由 RealMonitor 提供
Abort_Address       DCD     rm_dataabort_handler ; 由 RealMonitor 提供
FIQ_Address         DCD     0                   ; 用户可将 FIQ 处理程序的地址放置在此处

AREA    init_code,    CODE

ram_end    EQU    0x4000xxxx                    ; 片内 RAM 的顶端
__init
; /*********************************************************************
; *  为不同的处理模式建立堆栈指针。堆栈向下增加。
; *********************************************************************/
    LDR     r2, =ram_end        ; 得到 RAM 顶端地址
    MRS     r0, CPSR            ; 保存当前处理器模式
    ; 初始化未定义模式堆栈，供 RealMonitor 使用
    BIC     r1, r0, #0x1f
    ORR     r1, r1, #0x1b
    MSR     CPSR_c, r1
    ; 为 Flash 编程程序保留顶端 32 字节。参见 Flash 存储器系统和编程章节。
    SUB     sp,r2,#0x1F

    ; 初始化中止模式堆栈，供 RealMonitor 使用
    BIC     r1, r0, #0x1f
    ORR     r1, r1, #0x17
    MSR     CPSR_c, r1
    ; 为未定义模式堆栈保留 64 字节
    SUB     sp,r2,#0x5F

    ; 初始化 IRQ 模式堆栈，供 RealMonitor 和用户程序使用
```

```
    BIC     r1, r0, #0x1f
    ORR     r1, r1, #0x12
    MSR     CPSR_c, r1
    ; 为中止模式堆栈保留 32 字节
    SUB     sp,r2,#0x7F

    ; 返回初始模式
    MSR     CPSR_c, r0

    ; 初始化用户应用程序堆栈
    ; 为 IRQ 模式堆栈保留 256 字节
    SUB     sp,r2,#0x17F

; /*********************************************************************
; *  建立向量中断控制器。DCC Rx 和 Tx 中断产生非向量 IRQ 请求。
; * rm_init_entry 意识到 VIC 并使能 DBGCommRX 和 DBGCommTx 中断。
; *  将默认向量地址寄存器编程为非向量 app_irqDispatch 的地址。
; *  在此例中，用户可在此处建立向量 IRQ 或 FIQ。
; *********************************************************************/
    VICBaseAddr             EQU   0xFFFFF000 ; VIC 基地址
    VICDefVectAddrOffset    EQU   0x34

    LDR     r0, =VICBaseAddr
    LDR     r1, =app_irqDispatch
    STR     r1, [r0,#VICDefVectAddrOffset]

    BL      rm_init_entry       ; 初始化 RealMonitor
    ;使能 ARM 处理器中的 FIQ 和 IRQ
    MRS     r1, CPSR            ; 读取 CPSR
    BIC     r1, r1, #0xC0       ; 使能 IRQ 和 FIQ
    MSR     CPSR_c, r1          ; 更新 CPSR
; /*********************************************************************
; *  获取用户程序入口的地址。
; *********************************************************************/
    LDR     lr, =User_Entry
    MOV     pc, lr

; /*********************************************************************
; *  非向量 irq 处理程序  (app_irqDispatch)
; *********************************************************************/
AREA app_irqDispatch, CODE
VICVectAddrOffset EQU 0x30
app_irqDispatch
```

```
    ; 使能中断嵌套
    STMFD   sp!, {r12,r14}
    MRS     r12, spsr               ; 将 SPSR 保存到 r12
    MSR     cpsr_c,0x1F             ; 重新使能 IRQ, 进入系统模式

; 如果要求共用非向量中断，用户应当在此处插入代码。每个非向量共用 irq 处理程序都必须使用下列代码
; 返回到被中断的指令。
;MSR     cpsr_c, #0x52                   ; 禁止 irq, 进入 IRQ 模式
;MSR     spsr, r12                       ; 从 r12 恢复 SPSR
;STMFD   sp!, {r0}
;LDR     r0, =VICBaseAddr
;STR     r1, [r0,#VICVectAddrOffset]     ; 应答。非向量 irq 已经执行完毕
;LDMFD   sp!, {r12,r14,r0}               ; 恢复寄存器
;SUBS    pc, r14, #4                     ; 返回到被中断的指令

    ; 用户中断没有发生，因此调用 rm_irqhandler2。该处理程序没有意识到 VIC 的中断优先级
    ; 因此硬件使 rm_irqhandler2 返回到此处。

STMFD   sp!, {ip,pc}
LDR     pc, rm_irqhandler2
;rm_irqhandler2 返回到此处
MSR     cpsr_c, #0x52                   ; 禁止 irq, 进入 IRQ 模式
MSR     spsr, r12                       ; 将 SPSR 从 r12 恢复
STMFD   sp!, {r0}
LDR     r0, =VICBaseAddr
STR     r1, [r0,#VICVectAddrOffset]     ; 应答。非向量 irq 已经执行完毕
LDMFD   sp!, {r12,r14,r0}               ; 恢复寄存器
SUBS    pc, r14, #4                     ; 返回到被中断的指令
END
```

21.5 RealMonitor 建立选项

RealMonitor 使用下列选项建立：

```
RM_OPT_DATALOGGING=FALSE
```

该选项使能或禁止在非 RealMonitor（第三方）通道上发送任何从目标到主机的包。

```
RM_OPT_STOPSTART=TRUE
```

该选择使能或禁止对所有停止和启动调试特性的支持。

```
RM_OPT_SOFTBREAKPOINT=TRUE
```

该选项使能或禁止对软件断点的支持。

RM_OPT_HARDBREAKPOINT=TRUE

在带有 EmbeddedICE-RT 的内核上使能。该器件使用带有 EmbeddedICE-RT 的 ARM-7TDMI-S Rev 4 内核。

RM_OPT_HARDWATCHPOINT=TRUE

在带有 EmbeddedICE-RT 的内核上使能。该器件使用带有 EmbeddedICE-RT 的 ARM-7TDMI-S Rev 4 内核。

RM_OPT_SEMIHOSTING=FALSE

该选项使能或禁止对 SWI 半主机 (semi-hosting)的支持。Semi-hosting 提供运行在 ARM 目标板上的代码，这些代码具有运行 ARM 调试器的主机的一些功能。这些功能包括键盘输入、屏幕输出和磁盘 I/O 等等。

RM_OPT_SAVE_FIQ_REGISTERS=TRUE

当 RealMonitor 停止时，该选项决定是否将 FIQ 模式寄存器保存到寄存器块。

RM_OPT_READBYTES=TRUE

RM_OPT_WRITEBYTES=TRUE

RM_OPT_READHALFWORDS=TRUE

RM_OPT_WRITEHALFWORDS=TRUE

RM_OPT_READWORDS=TRUE

RM_OPT_WRITEWORDS=TRUE

使能或禁止对 8/16/32 位读/写的支持。

RM_OPT_EXECUTECODE=FALSE

使能或禁止对“执行代码”缓冲区的执行代码的支持。该代码必须先下载。

RM_OPT_GETPC=TRUE

该选项使能或禁止对 RealMonitor GetPC 包的支持。当中断模式中使用了实时监视时，它可用于代码的成型。

RM_EXECUTECODE_SIZE=NA

"执行代码"缓冲器规格。参见 RM_OPT_EXECUTECODE 选项。

RM_OPT_GATHER_STATISTICS=FALSE

该选项使能或禁止关于 RealMonitor 内部操作的集中统计表代码。

RM_DEBUG=FALSE

该选项使能或禁止在 RealMonitor 中额外的调试和错误检测代码。

RM_OPT_BUILDIDENTIFIER=FALSE

该选项决定是否在 RMTarget 的性能表中建立一个“建立标识符”。性能表保存在 ROM 中。

RM_OPT_SDM_INFO=FALSE

SDM 向调试工具提供关于应用板和处理器的额外信息。

RM_OPT_MEMORYMAP=FALSE

该选项决定板的存储器映射是否建立到目标当中。并通过性能表得到。

RM_OPT_USE_INTERRUPTS=TRUE

该选项指定是否为中断驱动模式和查询模式建立 RMTarget。

RM_FIFOSIZE=NA

该选项指定数据记录 FIFO 缓冲区的规格（以字为单位）。

CHAIN_VECTORS=FALSE

该选项允许 RMTarget 通过μHAL(ARM HW abstraction API)支持向量链。

第22章 补充信息

22.1 缩写词

表 239 首字母缩写表

缩写	描述
ADC	模数转换器
CPU	中央处理单元
DAC	数模转换器
DCC	调试通信通道
FIFO	先入先出
GPIO	通用输入/输出
NA	不可应用
PLL	锁相环
POR	上电复位
PWM	脉宽调节器
RAM	随机存取存储器
SRAM	静态随机存取存储器
UART	通用异步接收器/发送器
VIC	向量中断控制器
APB	ARM 外围总线

注：蓝笔部分是对该版本所作的修改。

附录A 周立功公司相关信息

总部

- **广州周立功单片机发展有限公司**

  地址： 广州市天河北路 689 号**光大银行大厦 15 楼 F1**

  邮编： 510630

  电话：(020) 38730916 38730917 38730976 38730977

  传真：(020) 38730925

  E-mail：chen@zlgmcu.com

  联系人： 陈智红(13902273164) 、周立新

- **销售一部(负责广州地区)：**

  电话：(020) 38730727

  传真：(020) 38730925

  E-mail：sales@zlgmcu.com

- **广州致远电子有限公司销售部 （销售致远产品）**

  地址：广州市天河区车陂路黄洲工业区 7 栋 2 楼

  邮编： 510660

  电话： (020) 22644249 22644399 28872569 28872571

  传真： (020) 38601859

- **技术支持：**

CAN-bus	020-22644381 22644382	CAN@zlgmcu.com
ARM	020-22644383 22644384	ARM@zlgmcu.com
金卡产品	020-22644377 22644378	mifare@zlgmcu.com
编程器	020-22644371 22644372	program@zlgmcu.com
分析仪器	020- 22644375	analyser@zlgmcu.com
USB	020-22644386	USB@zlgmcu.com
总体支持	020-22644358 22644359	LPC900@zlgmcu.com
	020-22644360 22644361	TKS@zlgmcu.com

分公司

- **武汉周立功**

  地址：武汉市洪山区广埠屯珞瑜路 158 号 12128 室(华中电脑数码市场)

  邮编：430079

  电话：(027) 87168497 87168397 87168297

  传真：(027) 87163755

  E-mail： wuhan@zlgmcu.com

  联系人：周东进(13971423627)

- **重庆周立功**

  地址：重庆市石桥铺科园一路二号大西洋国际大厦(赛格电子市场)1116 室

  邮编： 400039

  电话： (023) 68796438 68796439

  传真： (023) 68796439

  E-mail：chongqing@zlgmcu.com

  联系人：田清奎(13980708389)

- **北京周立功**

  地址：北京市海淀区知春路 113 号银网中心 712 室

  邮编：100086

  电话：(010) 62536178 62536179 82628073 82614433

  传真：(010) 82614433

  E-mail： beijing@zlgmcu.com

  联系人：朱利华(13801204663)

- **杭州周立功**

  地址：杭州市登云路 428 号浙江时代电子商城 205 号

  邮编：310000

  电话： (0571) 88009205 88009932 88009933

  传真： (0571) 88009204

  E-mail：hangzhou@zlgmcu.com

  联系人：黄森栋(13958004066)

- **成都周立功**

  地址： 成都市一环路南一段 57 号金城大厦 612 室

  邮编： 610041

  电话： (028) 85499320 85437446

  传真： (028) 85439505

  E-mail：chengdu@zlgmcu.com

联系人： 田清奎(13980708389)

- **深圳周立功**

  地址： 深圳市深南中路 2070 号电子科技大厦 A 座 24 楼 2403 室

  邮编： 518031

  电话：(0755) 83783298 83781768 83781788 83782922

  传真：(0755) 83793285

  E-mail：shenzhen@zlgmcu.com

  联系人： 周庆峰(13332983100)

- **上海周立功**

  地址： 上海市北京东路 668 号科技京城东座 7E 室

  邮编： 200001

  电话： (021) 53083452 53083453 53083496 53083497

  传真： (021) 53083491

  E-mail： shanghai@zlgmcu.com

  联系人： 曾成奇(13564533057)

- **南京周立功**

  地址：南京市珠江路 280 号珠江大厦 2006 室

  邮编： 210018

  电话： (025) 83613221 83613271 83603500 83603005

  传真： (025) 83613271

  E-mail： nanjing@zlgmcu.com

  联系人： 王涛(13512515168)

- **广州专卖店**

  地址： 广州市天河区新赛格电子城 203--204 室

  邮编： 510630

  电话： (020) 87578634 87578842 87569917

  传真： (020) 87578842

  E-mail： guangzhou@zlgmcu.com

  联系人： 陈智德(13352867848)

- **西安办事处**

  地址：西安市长安北路 54 号太平洋大厦 1201 室

  邮编：710061

  电话：(029) 87881296 87881295 83063000

  传真：(029) 87880865

  销售部：西安市长安北路 40 号电子大楼三楼西区 06 室

  电 话：(029) 85399492

  E-mail：XAagent@zlgmcu.com

  联系人：金龙(13619259555)