LPC2106/2105/2104 使用手册

1. 简介

特性

- ARM7TDMI-S 处理器
- 128k 字节片内 Flash 程序存储器，具有 ISP 和 IAP 功能。
- Flash 编程时间：1ms 可编程 512 字节，扇区擦除或整片擦除只需 400ms。
- 64/32/16K 字节静态 RAM（LPC2106/2105/2104）
- 向量中断控制器
- 仿真跟踪模块，支持实时跟踪
- RealMonitor 模块支持实时调试
- 标准 ARM 测试/调试接口，兼容现有工具
- 极小封装：TQFP48 ($7\times7mm^2$)
- 双 UART，其中一个带有完全的调制解调器接口
- I^2C 串行接口
- SPI 串行接口
- 两个定时器，分别具有 4 路捕获/比较通道
- 多达 6 路输出的 PWM 单元
- 实时时钟
- 看门狗定时器
- 通用 I/O 口
- CPU 操作频率可达 60MHz
- 双电源

  －CPU 操作电压范围：1.65V~1.95V(1.8V±8.3%)

  －I/O 电压范围：3.0V~3.6V(3.3V±10%)
- 两个低功耗模式：空闲和掉电
- 通过外部中断将处理器从掉电模式中唤醒
- 外设功能可单独使能/禁止，实现功耗最优化
- 片内晶振的操作频率范围：10MHz~25MHz
- 片内 PLL 允许 CPU 以最大速度运行，可以在超过整个晶振操作频率范围的情况下使用。

应用

- Internet 网关
- 串行通信协议转换器
- 访问控制
- 工业控制
- 医疗设备

结构概述

LPC2106/2105/2104 包含一个支持仿真的 ARM7TDMI-S CPU、与片内存储器控制器接口的 ARM7 局部总线、与中断控制器接口的 AMBA 高性能总线（AHB）和连接片内外设功能的 VLSI 外设总线（VPB，ARM

AMBA 总线的兼容超集)。LPC2106/2105/2104 将 ARM7TDMI-S 配置为小端（little-endian）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 外设都分配了 16k 字节的地址空间。LPC2106/2105/2104 的外设功能（中断控制器除外）都连接到 VPB 总线。AHB 到 VPB 的桥接将 VPB 总线与 AHB 总线相连。VPB 外设也分配了 2M 字节的地址范围，从 3.5GB 地址点开始。每个 VPB 外设在 VPB 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。该模块必须由软件进行控制以符合外设功能与管脚在特定应用中的需求。

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 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。

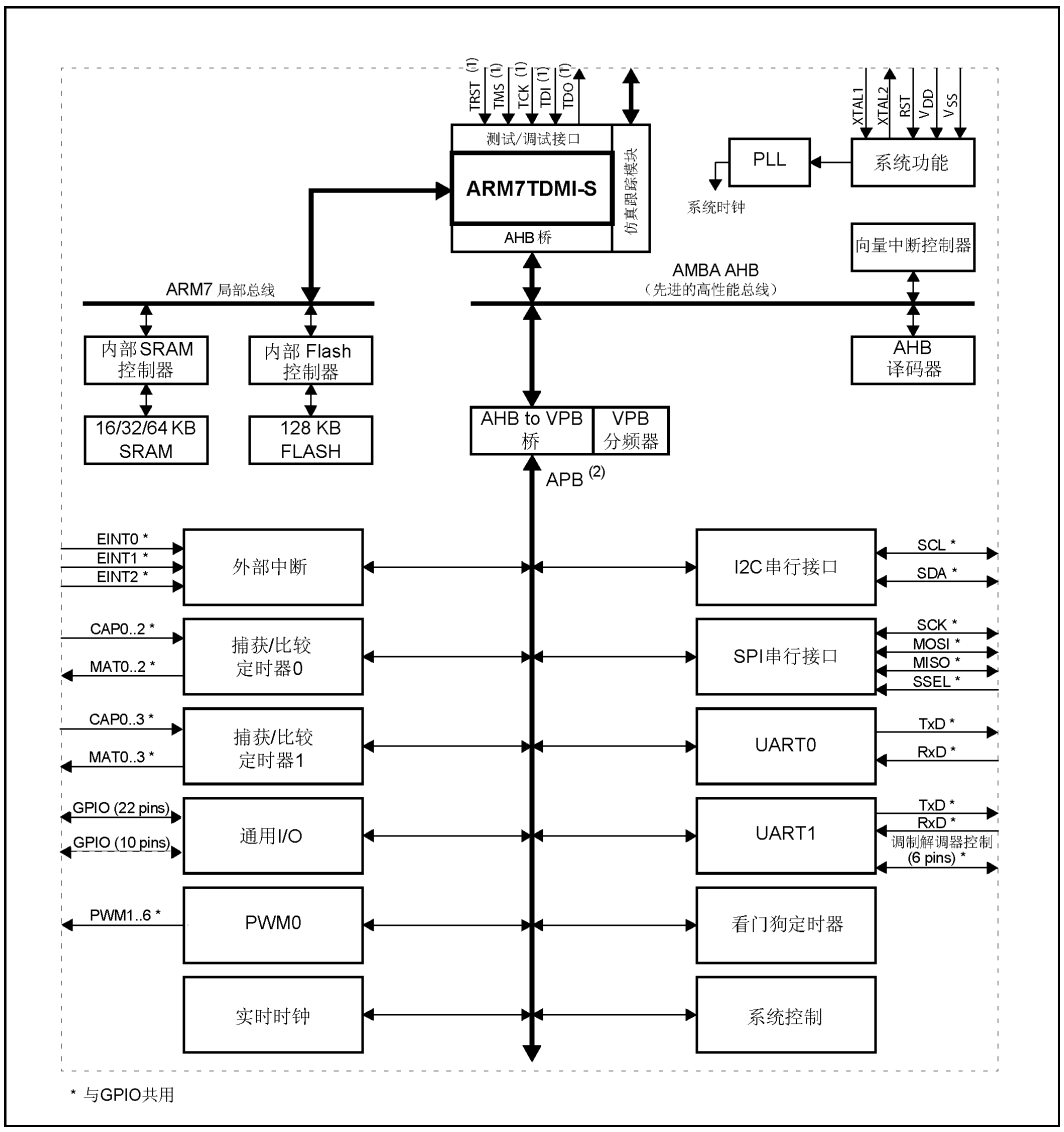
片内 FLASH 程序存储器

LPC2106/2105/2104 集成了一个 128K 字节的 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：通过内置的串行 JTAG 接口，通过串口进行在系统编程（ISP），也可以在应用程序运行时进行在应用编程（IAP）。这样为数据存储和现场固件的升级都带来了极大的灵活性。

片内静态 RAM

LPC2104/2105/2106 分别具有 16K/32K/64K 字节静态 RAM，SRAM 可用作代码和/或数据的存储。SRAM 支持 8 位、16 位和 32 位访问。

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。该数据只有在软件需要执行下一次写操作时才写入 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。任何在复位后检查 SRAM 内容的程序都必须注意这一点。在任何操作后附加一个对未使用地址的虚写（dummy write）操作可保证所有数据都真正写入了 SRAM。



1. 当使用测试/调试接口时，共用这些管脚的 GPIP/其它功能都不可用。

图 1 LPC2106/2105/2104 方框图

2. LPC2106/2105/2104 存储器寻址

LPC2104/2105/2106 包含几个不同的存储器区域，图 2 所示为复位后从用户角度所看到的整个地址空间映射。中断向量支持地址的重新映射，详见后面的章节。

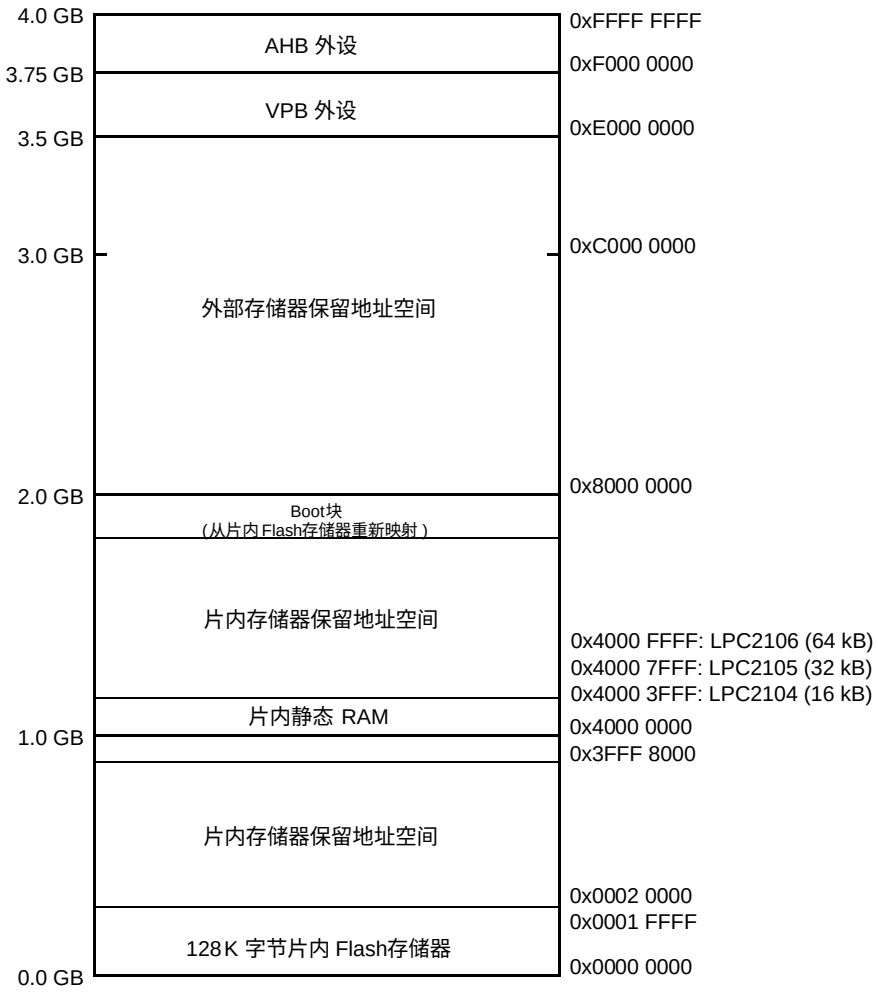


图 2 系统存储器映射

图 3~5 显示了从不同角度所观察到的外设地址空间。AHB 和 VPB 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。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。

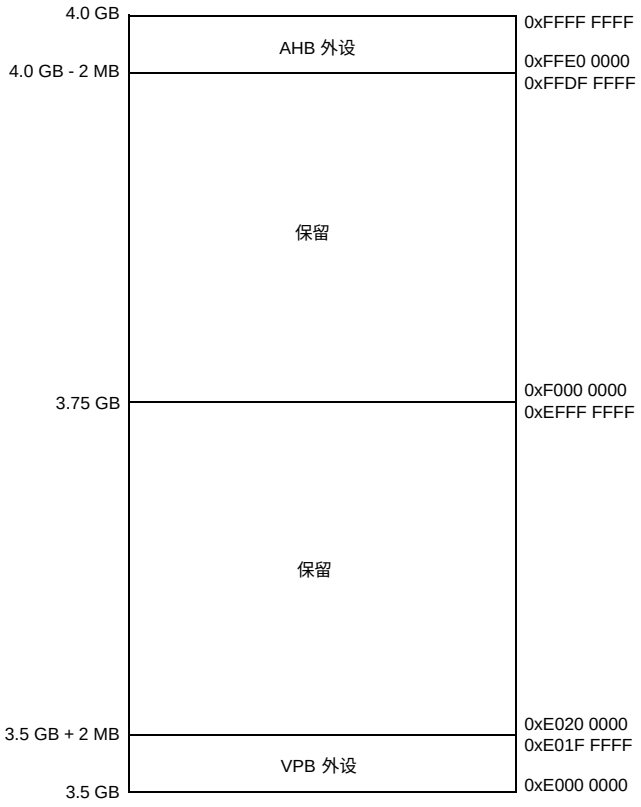


图 3 外设存储器映射

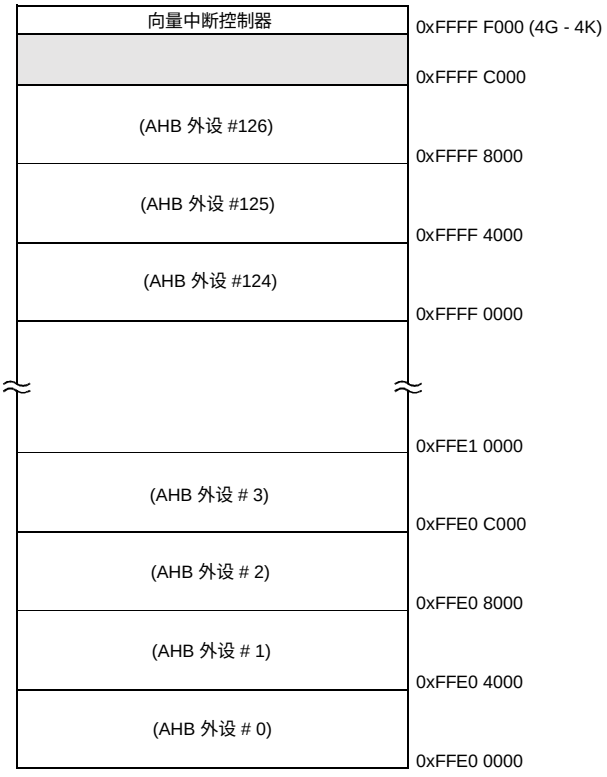


图 4 AHB 外设映射

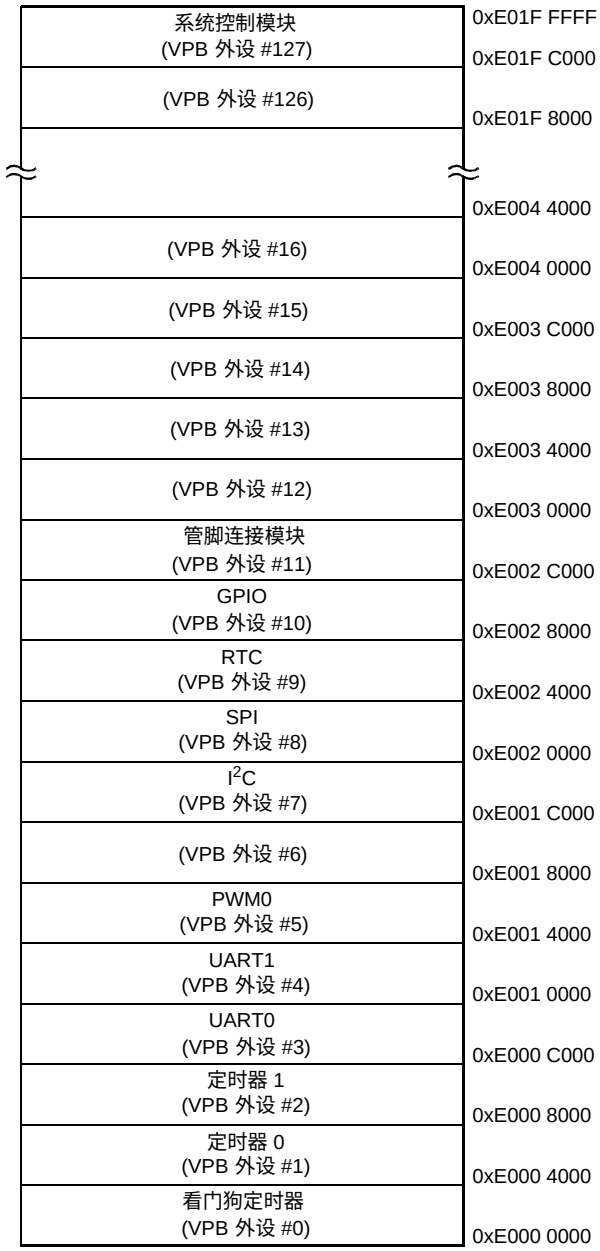


图 5 VPB 外设映射

LPC2106/2105/2104 存储器重新映射和 BOOT BLOCK

存储器映射概念和操作模式

LPC2106/2105/2104 的基本的概念是：每个存储器区域在存储器映射中都有一个“物理上的”位置。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，见表 1），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 2。中断的重新映射通过存储器映射控制特性来实现，详见*系统控制模块*一节。

表 1 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 *
0x0000 0018	IRQ
0x0000 001C	FIQ

* 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。

表 2 LPC2106/2105/2104 存储器映射模式

模式	激活	用途
Boot 装载程序模式	由任何复位硬件激活	在任何复位后都会执行 Boot 装载程序。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	由 Boot 代码软件激活	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	由用户程序软件激活	由用户程序激活。中断向量重新映射到静态 RAM 的底部。

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 6 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的转移指令。

选择这种配置有三个原因：

1. 使 Flash 存储器中的 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. 为超过单字转移指令范围的跳转提供空间来保存常量

重新映射的存储器区域，包括 Boot Block 和中断向量，除了重新映射的地址外，仍然继续出现在它们最初的位置。

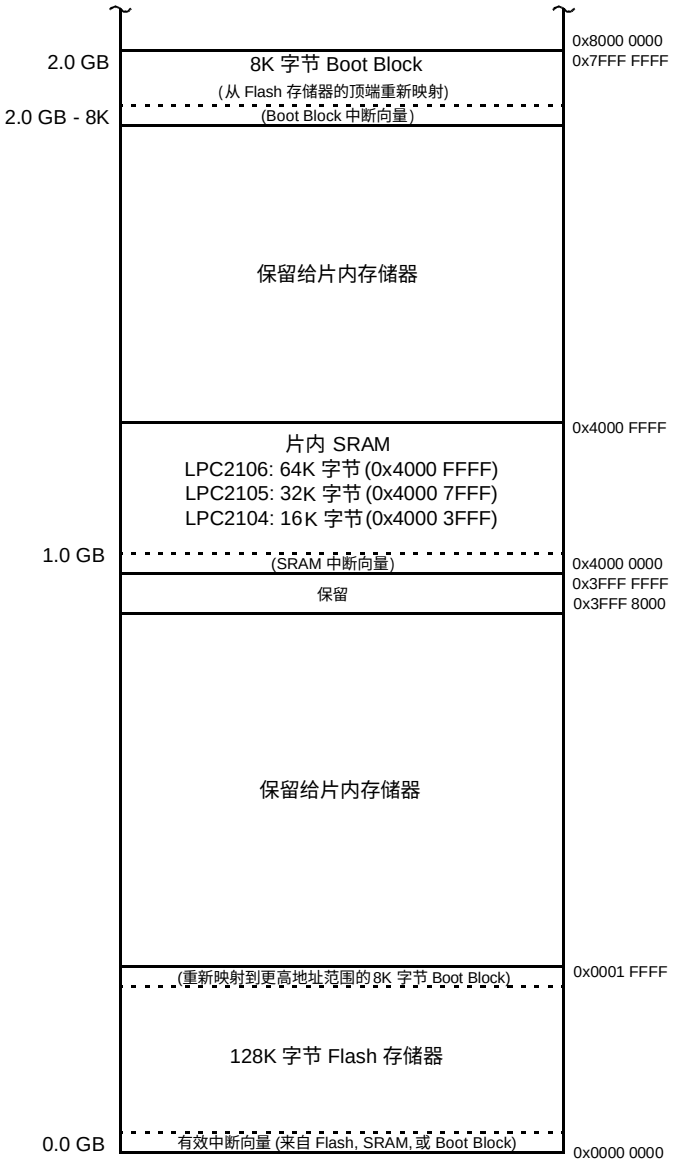


图 6 显示已重新映射和可重新映射区域的低存储器空间

预取指中止和数据中止异常

如果访问试图对一个保留地址或未分配区域的地址进行寻址，LPC2106/2105/2104 将产生适当的总线周期中止异常。这些区域包括：

- 特定的 ARM 器件所没有的存储器映射区域。对于 LPC2106/2105/2104，它们是：
  －片内非易失性存储器与特殊寄存器之间的地址空间，在图 2 和图 6 中标为“ 保留给片内存储器 ”。
  －片内静态 RAM 与外部存储器之间的地址空间，在图 2 中标为 “ 保留给片内存储器 ”
  －外部存储器（因为 LPC2106/2105/2104 没有包含外部存储器）
  －AHB 和 VPB 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 VPB 外设空间，见图 5。

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。此外，对 AHB 或 PVB 外设地址或特殊寄存器空间（位于地址 0x3FFF8000~0x3FFFFFFF）执行任何指令取指都会导致产生预取指中止异常。

在现有的 VPB 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判别的已定义寄存器。例如，对地址 0xE000D000（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 0xE000C000 处的寄存器进行访问。一个外设内的这样一种地址混淆在 LPC2106/2105/2104 文档中没有定义，并且它也不是一个被 LPC2106/2105/2104 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。

3. 系统控制模块

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
- 存储器映射控制
- PLL
- 功率控制
- 复位
- VPB 分频器
- 唤醒定时器

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

管脚描述

表 3 所示为系统控制模块功能相关的管脚。

表 3　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
$\overline{\text{EINT0}}$	输入	**外部中断输入 0**－低有效的通用中断输入。该管脚可用于将处理器从空闲或掉电模式中唤醒。
$\overline{\text{EINT1}}$	输入	**外部中断输入 1**－同上
$\overline{\text{EINT2}}$	输入	**外部中断输入 2**－同上
$\overline{\text{RST}}$	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 0 开始执行程序。

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，见表 4。

表 4　系统控制寄存器汇总。

地址	名称	描述	访问	复位值*
外部中断				
0xE01FC140	EXTINT	外部中断标志寄存器	R/W	0
0xE01FC144	EXTWAKE	外部中断唤醒寄存器	R/W	0
存储器映射控制				
0xE01FC040	MEMMAP	存储器映射控制	R/W	0
锁相环				
0xE01FC080	PLLCON	PLL 控制寄存器	R/W	0
0xE01FC084	PLLCFG	PLL 配置寄存器	R/W	0
0xE01FC088	PLLSTAT	PLL 状态寄存器	RO	0
0xE01FC08C	PLLFEED	PLL 馈送寄存器	WO	NA
功率控制				
0xE01FC0C0	PCON	功率控制寄存器	R/W	0
0xE01FC0C4	PCONP	外设功率控制	R/W	0x3BE
VPB 分频器				
0xE01FC100	VPBDIV	VPB 分频器控制	R/W	0

注：R/W: 读/写 RO: 只读 WO: 只写

* 复位值仅指已使用位中保存的数据，不包括保留位的内容。

晶体振荡器

振荡器支持的频率范围为 10MHz~25MHz。振荡器输出频率称为 F_{OSC}，而 ARM 处理器时钟频率称为 cclk。在该文档的其余部分，F_{OSC} 和 cclk 的值相同，除非运行并连接 PLL。详见 PLL 一节。

外部中断输入

LPC2106/2105/2104 包含 3 个外部中断输入。可用于将处理器从掉电模式唤醒。

寄存器描述

外部中断功能具有两个相关的寄存器。EXINT 寄存器包含中断标志。EXTWAKEUP 寄存器包含使能唤醒位，可使能独立的外部中断输入将处理器从掉电模式唤醒，见表 5。

表 5 外部中断寄存器

地址	名称	描述	访问
0xE01FC140	EXTINT	外部中断标志寄存器包含 ENIT0,EINT1 和 EINT2 的中断标志。见表 6。	R/W
0xE01FC144	EXTWAKE	外部中断唤醒寄存器包含 3 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 7。	R/W

EXTINT 寄存器（EXTINT - 0xE01FC140）

当外部中断映射到相关的管脚时，管脚上出现的逻辑 0 将置位 EXTINT 寄存器中对应的中断标志位。如果中断使能，这会使 VIC 作出正确的响应。一旦外部中断管脚的逻辑电平设置为 1，可以通过软件向 EXTINT 的相应位写入 1 将标志清零。只要管脚上的信号电平为 0，试图复位 EINT 位都是无效的。

表 6 外部中断标志寄存器（EXTINT- 0xE01FC140）

EXTINT	功能	描述	复位值
0	EINT0	当外部中断 $\overline{\text{EINT0}}$ 管脚变为低电平并且 EINT0 映射到它相关的管脚时，该位置位。当对应管脚的逻辑电平为 1 时，可向该位写入 1 将该位清零。	0
1	EINT1	当外部中断 $\overline{\text{EINT1}}$ 管脚变为低电平并且 EINT1 映射到它相关的管脚时，该位置位。当对应管脚的逻辑电平为 1 时，可向该位写入 1 将该位清零。	0
2	EINT2	当外部中断 $\overline{\text{EINT2}}$ 管脚变为低电平并且 EINT2 映射到它相关的管脚时，该位置位。当对应管脚的逻辑电平为 1 时，可向该位写入 1 将该位清零。	0
7:3	保留	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

EXTWAKE 寄存器（EXTWAKE - 0xE01FC144）

EXTWAKE 寄存器中的使能位允许外部中断将处理器从掉电模式唤醒。相关的 EINTn 功能必须映射到管脚才能实现掉电唤醒。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（关闭唤醒使能）。

表 7 外部中断唤醒寄存器（EXTWAKE – 0xE01FC144）

EXTWAKE	功能	描述	复位值
0	EINTWAKE0	该位为 1 时，使能 EINT0 将处理器从掉电模式唤醒。	0
1	EINTWAKE1	该位为 1 时，使能 EINT1 将处理器从掉电模式唤醒。	0
2	EINTWAKE2	该位为 1 时，使能 EINT2 将处理器从掉电模式唤醒。	0
7:2	保留	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

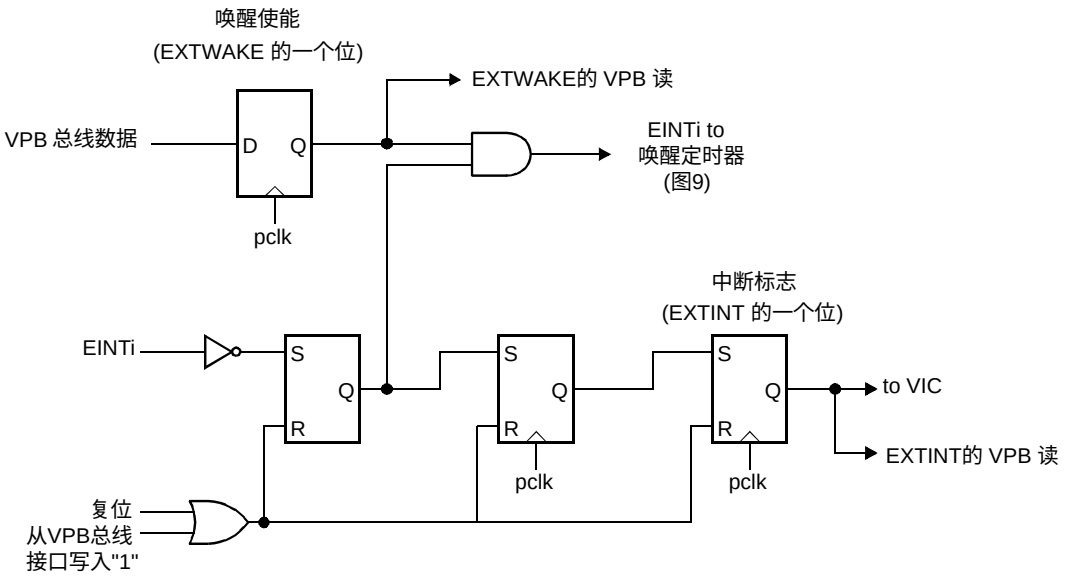


图 7 外部中断逻辑

存储器映射控制

存储器映射控制用于改变从地址 0x00000000 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。

存储器映射控制寄存器（MEMMAP – 0xE01FC040）

表 8 MEMMAP 寄存器

地址	名称	描述	访问
0xE01FC040	MEMMAP	存储器映射控制。选择从 Flash Boot Block、用户 Flash 或 RAM 中读取 ARM 中断向量。	R/W

表 9 存储器映射控制寄存器（MEMMAP – 0xE01FC040）

MEMMAP	功能	描述	复位值
1:0	MAP1:0	00: Boot 装载程序模式。中断向量从 Boot Block 重新映射。 01: 用户 Flash 模式。中断向量不重新映射，它位于 Flash 中。 10: 用户 RAM 模式。中断向量从静态 RAM 重新映射。 11: 保留，不可使用。 **警告**：不正确的设定会导致器件的异常操作。	0
7:2	保留	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

* LPC2106/2105/2104 的 MAP 位的硬件复位值为 00。Boot 装载程序会将用户看到的复位值更改，该程序总是在复位后立即运行。用户文档将反映这一区别。

PLL（锁相环）

PLL 接受的输入时钟频率范围为 10MHz~25MHz。输入频率通过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，LPC2106/2105/2104 的倍频值不能高于 6）。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 保持在频率范围内。输出分频器可设置为 2，4，8 或 16 分频。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 8 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 PLL 提供芯片时钟时，由于芯片的所有操作，包括看门狗定时器在内都依赖于它，因此 PLL 设置的意外改变将导致 CPU 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代码序列来实现。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，然后再连接 PLL。

寄存器描述

PLL 由表 10 所示的寄存器进行控制。

警告：PLL 值的不正确设定会导致芯片的错误操作。

表 10 PLL 寄存器

地址	名称	描述	访问
0xE01FC080	PLLCON	最新的 PLL 控制位的保持寄存器。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
0xE01FC084	PLLCFG	最新的 PLL 配置值的保持寄存器。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
0xE01FC088	PLLSTAT	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 状态的真实值。	RO
0xE01FC08C	PLLFEED	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。	WO

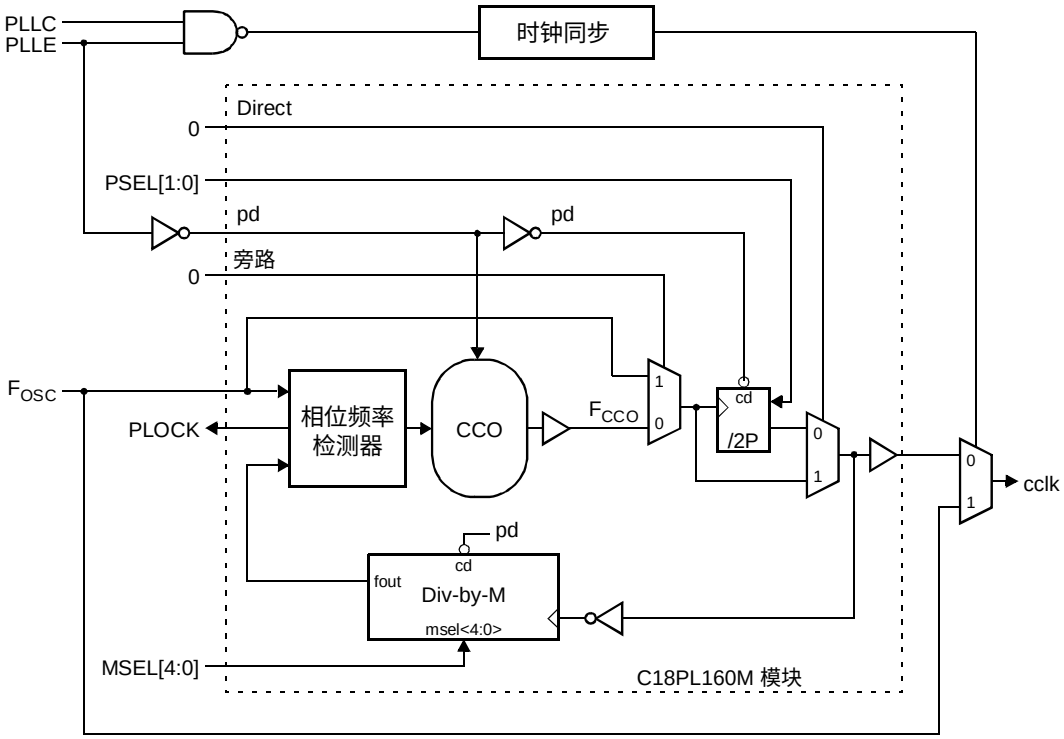


图 8 PLL 方框图

PLLCON 寄存器（PLLCON – 0xE01FC080）

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 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 PLLFEED - 0xE01FC08C 一节的描述）。

表 11 PLL 控制寄存器（PLLCON – 0xE01FC080）

PLLCON	功能	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。	0
1	PLLC	**PLL 连接**。当 PLLC 和 PLLE 都为 1 并且在有效的 PLL 馈送后，将 PLL 作为时钟源连接到 LPC2106/2105/2104。否则，LPC2106/2105/2104 直接使用振荡器时钟。见表 13 的 PLLSTAT 寄存器描述。	0
7:2	保留	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 在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。

PLLCFG 寄存器（PLLCFG – 0xE01FC084）

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。PLL 频率和倍频器以及分频器值的计算详见 *PLL 频率计算*一节。

表 12 PLL 配置寄存器（PLLCFG – 0xE01FC084）

PLLCFG	功能	描述	复位值
4:0	MSEL4:0	PLL 倍频器值。在 PLL 频率计算中其值为 M。	0
6:5	PSEL1:0	PLL 分频器值。在 PLL 频率计算中其值为 P。	0
7	保留	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

PLLSTAT 寄存器（PLLSTAT - 0xE01FC088）

从 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 PLLFEED 寄存器的描述。

表 13 PLL 状态寄存器（PLLSTAT – 0xE01FC088）

PLLSTAT	功能	描述	复位值
4:0	MSEL4:0	读出的 PLL 倍增器值。这是 PLL 当前使用的值。	0
6:5	PSEL1:0	读出的 PLL 分频器值。这是 PLL 当前使用的值。	0
7	保留	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 作为时钟源连接到 LPC2106/2105/2104；该位为 0 时，PLL 被旁路，LPC2106/2105/2104 直接使用振荡器时钟。当进入掉电模式时，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	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 中断

PLLSTAT 寄存器中的 PLOCK 位连接到中断控制器。这样可使用软件打开 PLL 并连接到其它功能，不需要等待 PLL 锁定。当发生中断时，可以连接 PLL 并禁止中断。

PLL 模式

PLLE 和 PLLC 的组合见表 14。

表 14 PLL 控制位组合

PLLC	PLLE	PLL 功能
0	0	PLL 被关闭并断开连接。系统使用未更改的时钟输入。
0	1	PLL 被激活但是尚未连接。PLL 可在 PLOCK 置位后连接。
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 已使能并连接到处理器作为系统时钟源。

PLLFEED 寄存器（PLLFEED – 0xE01FC08C）

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 VPB 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。不管是写入的值不正确还是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 15 PLLFEED 寄存器（PLLFEED – 0xE01FC08C）

PLLFEED	功能	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。	未定义

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。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。

PLL 频率计算

PLL 等式使用下列参数：

F_{OSC} 晶振频率

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 位的倍增器值

P PLLCFG 寄存器中 PSEL 位的分频器值

PLL 输出频率（当 PLL 激活并连接时）由下式得到：

$cclk = M * F_{OSC}$ 或 $cclk = F_{CCO} / (2*P)$

CCO 频率可由下式得到：

$F_{CCO} = cclk * 2 * P$ 或 $F_{CCO} = F_{OSC} * M * 2 * P$

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}（LPC2106/2105/2104 的最大允许频率）
- F_{CCO} 的范围：156MHz~320MHz

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. 选择处理器的操作频率（cclk）。这可以根据处理器的整体要求、UART 波特率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 VPB 分频器描述）。
2. 选择振荡器频率（F_{OSC}）。cclk 一定是 F_{OSC} 的整数倍。
3. 计算 M 值以配置 MSEL 位。M = cclk/FOSC，M 的取值范围为 1~32。写入 MSEL 位的值为 M-1(见表 17)。
4. 选择 P 值以配置 PSEL 位，FCCO 的频率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 PSEL 位的值 00 表示 P=1；01 表示 P=2；10 表示 P=4；11 表示 P=8 (见表 16)。

表 16 PLL 分频器值

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

表 17 PLL 倍增器值

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

功率控制

LPC2106/2105/2104 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保持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 值在掉电模式下被保持。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。

通过中断唤醒掉电模式不会使指令丢失、不完整或重复。从掉电模式唤醒将在唤醒定时器一节中作进一步讨论。

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。

寄存器描述

功率控制功能包含两个寄存器，如表 18 所示。

表 18 功率控制寄存器

地址	名称	描述	访问
0xE01FC0C0	PCON	功率控制寄存器。该寄存器包含两种节电模式的使能位。见表 19。	R/W
0xE01FC0C4	PCONP	外设功率控制寄存器。该寄存器包含使能和禁止单个外设功能的控制位。见表 20。	R/W

PCON 寄存器（PCON – 0xE01FC0C0）

PCON 寄存器包含两个位。置位其中一个位，将会进入对应的节电模式。如果两位都置位，则进入掉电模式。

表 19 功率控制寄存器（PCON – 0xE01FC0C0）

PCON	功能	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	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

外设功率控制寄存器（PCONP – 0xE01FC0C4）

PCONP 寄存器允许将所选的外设功能关闭以实现节电的目的。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）。PCONP 中的每个位都控制一个外设，见表 20。每个位所对应的外设编号见 *VPB 外设映射*一节。

表 20 外设功率控制寄存器（PCONP – 0xE01FC0C4）

PCONP	功能	描述	复位值
0	保留	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	该位为 1 时，定时器 0 使能。为 0 时，定时器 0 被关闭以实现节电。	1
2	PCTIM1	该位为 1 时，定时器 1 使能。为 0 时，定时器 1 被关闭以实现节电。	1
3	PCURT0	该位为 1 时，UART0 使能。为 0 时，UART0 被关闭以实现节电。	1
4	PCUART1	该位为 1 时，UART1 使能。为 0 时，UART1 被关闭以实现节电。	1
5	PCPWM0	该位为 1 时，PWM0 使能。为 0 时，PWM0 被关闭以实现节电。	1
6	保留	保留给 PWM1。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	PCI2C	该位为 1 时，I^2C 接口使能。为 0 时，I^2C 接口被关闭以实现节电。	1
8	PCSPI	该位为 1 时，SPI 接口使能。为 0 时，SPI 接口被关闭以实现节电。	1
9	PCRTC	该位为 1 时，RTC 使能。为 0 时，RTC 被关闭以实现节电。	1
31:10	保留	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

复位

LPC2106/2105/2104 有两个复位源：$\overline{RST}$ 管脚和看门狗复位。$\overline{RST}$ 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（详见唤醒定时器的描述），只有当外部复位撤除后，振荡器开始运行。当计数达到一个固定个数的时钟时，Flash 控制器完成其初始化。复位、振荡器以及唤醒定时器之间的关系见图 9。

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 $\overline{RST}$ 保证芯片复位所必须保持的最短时间。详细的复位时序要求可参阅 LPC2106/2105/2104 数据手册。

当内部复位撤除时，处理器从地址 0 开始运行，此处为复位向量。此时所有的处理器和外设寄存器都恢复为默认状态。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，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 DBGSEL, RTCK 和 EINT1 进行检测以实现不同的目的。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。

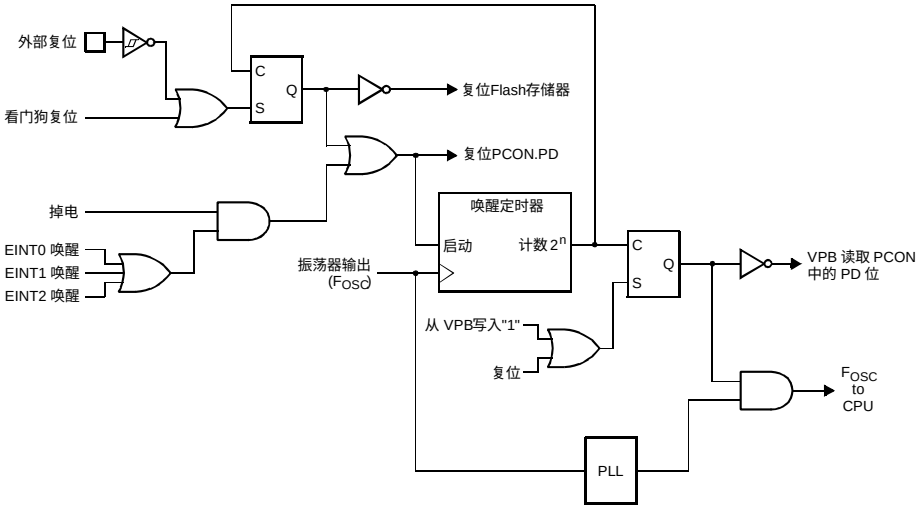


图 9 包括唤醒定时器的复位方框图

VPB 分频器

VPB 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。VPB 分频器有两个用途。第一个是 VPB 总线不能直接在 ARM 处理器的最高频率下工作。为了弥补这一点，VPB 总线可以降低到 1/2 或 1/4 处理器时钟速率。由于 VPB 总线必须在上电后正常工作（并且不能改变其时序），VPB 总线在复位后默认的状态是以 1/4 速度运行。VPB 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。

VPB 分频器与振荡器和处理器时钟的连接见图 10。由于 VPB 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）。

VPBDIV 寄存器（VPBDIV - 0xE01FC100）

VPB 分频器寄存器包含两个位，可以设定 3 个分频值，详见表 22。

表 21 VPBDIV 寄存器映射

地址	名称	描述	访问
0xE01FC100	VPBDIV	控制 VPB 时钟速率与处理器时钟之间的关系	R/W

表 22 VPBDIV 寄存器（VPBDIV - 0xE01FC100）

VPBDIV	功能	描述	复位值
1:0	VPBDIV	VPB 时钟速率如下： 00：VPB 总线时钟为处理器时钟的 1/4。 01：VPB 总线时钟与处理器时钟相同。 10：VPB 总线时钟为处理器时钟的 1/2。 11：保留。将该值写入 VPBDIV 寄存器无效（保留原来的设定）。 **警告**：对该值不正确的设定会导致器件的异常操作。	0
7:2	保留	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

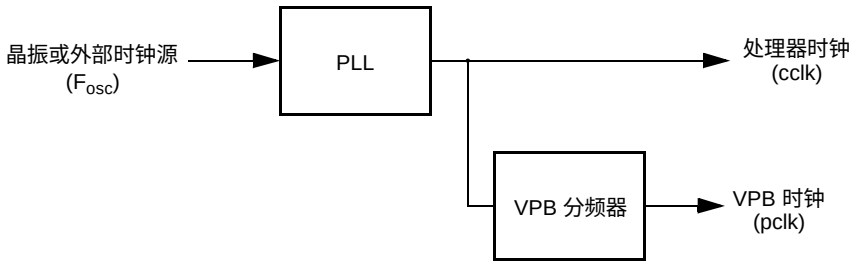


图 10 VPB 分频器连接

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，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 Vdd 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 Flash 进行初始化。当 Flash

存储器初始化完毕时，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，需要考虑的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。

LPC2106/2105/2104 不包含不需要时钟的比较器或者具有独立时钟源的看门狗振荡器这样的模拟功能。没有时钟源仍能工作的唯一功能是外部中断 EINT0, EINT1 和 EINT2。如果外部中断使能产生唤醒并被激活（外部驱动为低），那么必须启动振荡器唤醒。如果向量中断控制器中的中断也使能，中断处理的完成将会推迟到唤醒定时器停止之后。

总之，唤醒定时器根据晶振执行最短时间的复位，它在从掉电模式中唤醒或任何复位产生时激活。

4. 存储器加速模块（MAM）

介绍

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。所使用的方法是将 Flash 存储器分成两组，每一组都可独立进行访问。这两个 Flash 组都有自己的预取指缓冲区和分支跟踪缓冲区。当一个组的预取指缓冲区和分支跟踪缓冲区不能满足指令取指的需要，并且预取指还没有启动时，两个组的分支跟踪缓冲区捕获两个 128 位的 Flash 数据行。在 MAM 启动的预取指周期的结束，每个预取指缓冲区从它自身的 Flash 组捕获一个 128 位指令行。

每个 128 位值包括了 4 个 32 位 ARM 指令或 8 个 16 位 Thumb 指令。在连续执行代码时，通常一个 Flash 组包含或者正在取指当前的指令和包含该指令的整个 Flash 行。另一个 Flash 组则包含或正在预取指下一个连续的代码行。当一个代码行传送完最后一条指令时，包含它的 Flash 组开始对下一行进行取指。

对 Flash 读操作的时序可进行编程，这将在稍后讲述。使用 MAM 方式，当 CPU 时钟周期大于等于 Flash 访问时间的 1/4 时，对连续指令的执行不会对代码取指造成影响。花费在程序转移上的平均时间相对较小（小于 25%），而且通过使用 ARM 指令中具有的传统执行特性可使其在 ARM(不是 Thumb)代码中降到最低。这种条件执行可经常用来避免较小的前向分支，除非该分支是必要的。

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。当发生回溯分支时，表示很有可能正在执行一个循环。分支跟踪缓冲区有可能已经包含了目标指令。如果是，不需要执行 Flash 读周期就可执行指令。对于一个前向分支，新的地址也有可能包含在其中一个预取指缓冲区中。如果是，那么分支的执行不会有任何延迟。

当分支不在分支跟踪和预取指缓冲区当中时，则需要暂停几个时钟来装载分支跟踪缓冲区。接下来将不再有取指的延迟，除非发生了另一个这样的“ 指令丢失 ”。

Flash 存储器控制器检测访问 Flash 存储器的数据并使用一个单独的缓冲区保存结果，采用的方式类似于代码取指时使用的方式。这样就加快了按顺序访问数据的速度。数据访问使用一个单行的缓冲区，和访问代码时提供两个缓冲区不同。因为数据访问不需要预取指功能。

存储器加速器模块

存储器加速器模块分成以下几个功能块：

- 为每个组提供 Flash 地址锁存。用于 Flash 组 0 地址锁存的增量器功能。
- 两个 Flash 存储器组
- 指令锁存，数据锁存，地址比较锁存
- 等待逻辑

图 11 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后的地址执行 Flash 读操作。

Flash 存储器组

两个 Flash 存储器组实现了并行访问并消除了连续访问时的延迟。

Flash 编程功能不受存储器加速器模块的控制，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。

Flash 存储器的布线使其每个扇区同时存在于两个组当中，这样扇区擦除操作可同时对两个组执行。实际上，两个组的实体对于编程功能是透明的。

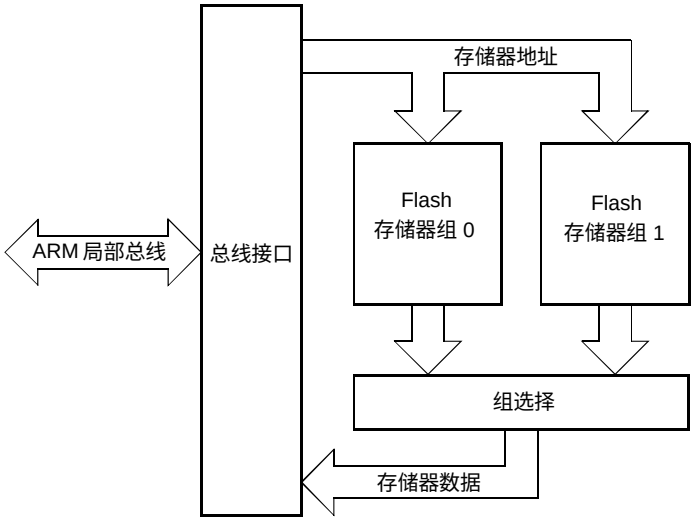


图 11 存储器加速器模块的简化框图

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 Flash 组都由两套 128 位指令锁存和 12 位比较地址锁存。其中一套称为分支跟踪缓冲区，用于保存最近一次指令丢失以来的数据和比较地址。另一套称为预取指缓冲区，用于保存预取指的数据和比较地址。每个指令锁存保存 4 个代码字（4 条 ARM 指令或 8 条 Thumb 指令）。

与之相似，在数据访问中使用了一个 128 位数据锁存和 13 位数据地址锁存。两个 Flash 组共用这一套锁存。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效果。

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 MAM 就必须强制 CPU 在 Flash 模块忙的时候等待（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不会出现非预期的看门狗复位所导致的系统故障。

为了防止从 Flash 存储器中读取无效的数据，MAM 使锁存在 Flash 编程或擦除操作的开始自动失效。在 Flash 操作结束后，任何对 Flash 地址的读操作将启动新的取指。

存储器加速器模块的操作模式

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：

0）MAM 关闭。所有存储器请求都会导致 Flash 的读操作。无指令预取指。

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。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。

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 读操作用于指令的预取指和当前缓冲区所没有的代码或数据的访问。

表 23 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]

表 24 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 [2]	启动取指 [2]	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 [2]	启动取指 [2]	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. 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。

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，而某些要求更精确定时的功能可以较慢但更可预测的速度运行。

寄存器描述

所有寄存器不管规格如何，都占用 1 个字的宽度。

表 25 系统控制寄存器汇总

地址	名称	描述	访问	复位值
MAM				
0xE01FC000	MAMCR	存储器加速器模块控制寄存器。决定 MAM 的操作模式。也就是说 MAM 性能增强的程度，见表 26。	R/W	0
0xE01FC004	MAMTIM	存储器加速器定时控制。决定 Flash 存储器取指所使用的时钟个数(1 到 7 个时钟)。	R/W	0x07

MAM 控制寄存器（MAMCR - 0xE01FC000）

两个配置位选择 MAM 的操作模式。见表 26。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 26 MAM 控制寄存器（MAMCR - 0xE01FC000）

MAMCR	功能	描述	复位值
1:0	MAM 模式控制	这两个位决定 MAM 的操作模式： 00－MAM 功能被禁止 01－MAM 功能部分使能 10－MAM 功能完全使能 11－保留	0
7:2	保留	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。	

MAM 定时寄存器（MAMTIM - 0xE01FC004）

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 访问实际上关闭了 MAM。这种情况下可以选择 MAM 模式对功耗进行优化。

表 27 MAM 定时寄存器（MAMTIM - 0xE01FC004）

MAMTIM	功能	描述	复位值
2:0	MAM 取指周期	这几个位决定 MAM Flash 取指操作的时间： 000＝0－保留 001＝1－MAM 取指周期为 1 个处理器时钟（cclk）。 010＝2－MAM 取指周期为 2 个处理器时钟（cclk）。 011＝3－MAM 取指周期为 3 个处理器时钟（cclk）。 100＝4－MAM 取指周期为 4 个处理器时钟（cclk）。 101＝5－MAM 取指周期为 5 个处理器时钟（cclk）。 110＝6－MAM 取指周期为 6 个处理器时钟（cclk）。 111＝7－MAM 取指周期为 7 个处理器时钟（cclk）。 警告：不正确的设定会导致器件的异常操作。	0x07
7:3	保留	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

MAM 使用注意事项

当改变 MAM 定时值时，必须先通过 MAMCR 关闭 MAM，然后将新值写入 MAMTIM。最后再根据所需要的操作模式打开 MAM。

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

5. 向量中断控制器

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

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。默认程序可读取任何 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*的相关文档。

寄存器描述

VIC 所包含的寄存器如表 28 所示。

表 28 VIC 寄存器映射

地址	名称	描述	访问	复位值
0xFFFF F000	VICIRQStatus	IRQ 状态。该寄存器读出定义为 IRQ 并使能的中断的状态。	RO	0
0xFFFF F004	VICFIQStatus	FIQ 状态请求。该寄存器读取定义为 FIQ 并使能的中断的状态。	RO	0
0xFFFF F008	VICRawIntr	所有中断的状态。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 F00C	VICIntSelect	中断选择。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 F010	VICIntEnable	中断使能。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 F014	VICIntEnClr	中断使能清零。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。	W	0
0xFFFF F018	VICSoftInt	软件中断。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 F01C	VICSoftIntClear	软件中断清零。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。	W	0
0xFFFF F020	VICProtection	保护使能。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 F030	VICVectAddr	向量地址。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 F034	VICDefVectAddr	默认向量地址。该寄存器保存了非向量中断的中断服务程序（ISR）地址。	R/W	0
0xFFFF F100	VICVectAddr0	向量地址 0。向量地址寄存器 0-15 保存了 16 个向量 IRQ slot 的中断服务程序地址。	R/W	0
0xFFFF F104	VICVectAddr1	向量地址 1 寄存器	R/W	0

续上表

地址	名称	描述	访问	复位值
0xFFFF F108	VICVectAddr2	向量地址 2 寄存器	R/W	0
0xFFFF F10C	VICVectAddr3	向量地址 3 寄存器	R/W	0
0xFFFF F110	VICVectAddr4	向量地址 4 寄存器	R/W	0
0xFFFF F114	VICVectAddr5	向量地址 5 寄存器	R/W	0
0xFFFF F118	VICVectAddr6	向量地址 6 寄存器	R/W	0
0xFFFF F11C	VICVectAddr7	向量地址 7 寄存器	R/W	0
0xFFFF F120	VICVectAddr8	向量地址 8 寄存器	R/W	0
0xFFFF F124	VICVectAddr9	向量地址 9 寄存器	R/W	0
0xFFFF F128	VICVectAddr10	向量地址 10 寄存器	R/W	0
0xFFFF F12C	VICVectAddr11	向量地址 11 寄存器	R/W	0
0xFFFF F130	VICVectAddr12	向量地址 12 寄存器	R/W	0
0xFFFF F134	VICVectAddr13	向量地址 13 寄存器	R/W	0
0xFFFF F138	VICVectAddr14	向量地址 14 寄存器	R/W	0
0xFFFF F13C	VICVectAddr15	向量地址 15 寄存器	R/W	0
0xFFFF F200	VICVectCntl0	向量控制 0。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 F204	VICVectCntl1	向量控制 1 寄存器	R/W	0
0xFFFF F208	VICVectCntl2	向量控制 2 寄存器	R/W	0
0xFFFF F20C	VICVectCntl3	向量控制 3 寄存器	R/W	0
0xFFFF F210	VICVectCntl4	向量控制 4 寄存器	R/W	0
0xFFFF F214	VICVectCntl5	向量控制 5 寄存器	R/W	0
0xFFFF F218	VICVectCntl6	向量控制 6 寄存器	R/W	0
0xFFFF F21C	VICVectCntl7	向量控制 7 寄存器	R/W	0
0xFFFF F220	VICVectCntl8	向量控制 8 寄存器	R/W	0
0xFFFF F224	VICVectCntl9	向量控制 9 寄存器	R/W	0
0xFFFF F228	VICVectCntl10	向量控制 10 寄存器	R/W	0
0xFFFF F22C	VICVectCntl11	向量控制 11 寄存器	R/W	0
0xFFFF F230	VICVectCntl12	向量控制 12 寄存器	R/W	0
0xFFFF F234	VICVectCntl13	向量控制 13 寄存器	R/W	0
0xFFFF F238	VICVectCntl14	向量控制 14 寄存器	R/W	0
0xFFFF F23C	VICVectCntl15	向量控制 15 寄存器	R/W	0

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。

软件中断寄存器（VICSoftInt - 0xFFFFF018，读/写）

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 29 软件中断寄存器（VICSoftInt - 0xFFFFF018，读/写）

VICSoftInt	功能	复位值
31:0	1：强制产生与该位相关的中断请求。 0：不强制产生中断请求。向 VICSoftInt 写入 0 无效，见 VICSoftIntClear。	0

软件中断清零寄存器（VICSoftIntClear - 0xFFFFF01C，只写）

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 30 软件中断清零寄存器（VICSoftIntClear - 0xFFFFF01C，只写）

VICSoftIntClear	功能	复位值
31:0	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。 0：写入 0 不会影响 VICSoftInt 中的相应位。	0

所有中断状态寄存器（VICRawIntr - 0xFFFFF008，只读）

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 31 所有中断状态寄存器（VICRawIntr - 0xFFFFF008，只读）

VICRawIntr	功能	复位值
31:0	1：对应位的中断请求或软件中断声明。 0：对应位的中断请求或软件中断未声明。	0

中断使能寄存器（VICIntEnable - 0xFFFFF010，读/写）

该寄存器使能分配为 FIQ 或 IRQ 的中断请求或软件中断。

表 32 中断使能寄存器（VICIntEnable - 0xFFFFF010，读/写）

VICIntEnable	功能	复位值
31:0	当读取该寄存器时，1 表示中断请求使能为 FIQ 或 IRQ。当写该寄存器时，1 使能中断请求或软件中断，0 无效。见 VICIntEnClear 寄存器（表 28）。	0

中断使能清零寄存器（VICIntEnClr - 0xFFFFF014，只写）

该寄存器在不需要读取中断使能寄存器的情况下，可用软件清零其中的一个或多个位。

表 33 中断使能清零寄存器（VICIntEnClr - 0xFFFFF014，只写）

VICIntEnClr	功能	复位值
31:0	1：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：写入 0 不影响中断使能寄存器中的位。	0

中断选择寄存器（VICIntSelect - 0xFFFFF00C，读/写）

该寄存器将 32 个中断请求分别分配为 FIQ 或 IRQ。

表 34 中断选择寄存器（VICIntSelect - 0xFFFFF00C，读/写）

VICIntSelect	功能	复位值
31:0	1：对应的中断请求分配为 FIQ。 0：对应的中断请求分配为 IRQ。	0

IRQ 状态寄存器（VICIRQStatus - 0xFFFFF000，只读）

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 35 IRQ 状态寄存器（VICIRQStatus - 0xFFFFF000，只读）

VICIRQStatus	功能	复位值
31:0	1：对应位的中断请求使能并分配为 IRQ 并且声明。	0

FIQ 状态寄存器（VICFIQStatus - 0xFFFFF004，只读）

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 36 FIQ 状态寄存器（VICFIQStatus - 0xFFFFF004，只读）

VICFIQStatus	功能	复位值
31:0	1：对应位的中断请求使能并分配为 FIQ 并且声明。	0

向量控制寄存器 0-15（VICVectCntI0-15 - 0xFFFFF200-23C，读/写）

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。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 37 向量控制寄存器 0-15（VICVectCntl0-15 - 0xFFFFF200-23C，读/写）

VICVectCntl0-15	功能	复位值
5	1：向量 IRQ 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 地址对应位的中断请求使能并分配为 FIQ 并且声明。	0
4:0	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

向量地址寄存器 0-15（VICVectAddr0-15 - 0xFFFFF100-13C，读/写）

这些寄存器保存 16 个向量 IRQ slot 中断服务程序的地址。

表 38 向量地址寄存器 0-15（VICVectAddr0-15 - 0xFFFFF100-13C，读/写）

VICVectAddr0-15	功能	复位值
31:0	当一个或多个分配为向量 IRQ slot 的中断请求使能，分配为 IRQ，声明并时，IRQ 服务程序读取向量地址寄存器（VICVectAddr）时会得到最高优先级 slot 寄存器的值。	0

默认向量地址寄存器（VICDefVectAddr - 0xFFFFF034，读/写）

这些寄存器保存非向量 IRQ 中断服务程序的地址。

表 39 默认向量地址寄存器（VICDefVectAddr - 0xFFFFF034，读/写）

VICDefVectAddr	功能	复位值
31:0	当一个 IRQ 服务程序读取向量地址寄存器，并且没有 IRQ slot 响应时，则返回该寄存器中的地址。	0

向量地址寄存器（VICVectAddr - 0xFFFFF030，读/写）

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 40 向量地址寄存器（VICVectAddr - 0xFFFFF030，读/写）

VICVectAddr	功能	复位值
31:0	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。	0

保护使能寄存器（VICProtection - 0xFFFFF020，读/写）

运行在用户模式下的软件使用该 1 位寄存器来控制对 VIC 寄存器的访问。

表 41 保护使能寄存器（VICProtection - 0xFFFFF020，读/写）

VICProtection	功能	复位值
0	1：VIC 寄存器只能在特权模式下访问。 0：VIC 寄存器可在用户模式或特权模式下访问。	0

中断源

表 42 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。

表 42 连接到向量中断控制器的中断源

模块	标志	VIC 通道#
WDT	看门狗中断（WDINT）	0
-	保留给软件中断	1
ARM 内核	EmbeddedICE, DbgCommRx	2
ARM 内核	EmbeddedICE, DbgCommTx	3
定时器 0	匹配 0-3（MR0, MR1, MR2, MR3） 捕获 0-3（CR0, CR1, CR2, CR3）	4
定时器 1	匹配 0-3（MR0, MR1, MR2, MR3） 捕获 0-3（CR0, CR1, CR2, CR3）	5
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
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
PWM0	匹配 0-6（MR0, MR1, MR2, MR3, MR4, MR5, MR6） 捕获 0-3（CR0, CR1, CR2, CR3）	8
I2C	SI（状态改变）	9
SPI	SPIF, MODF	10
-	保留	11
PLL	PLL 锁定（PLOCK）	12
RTC	RTCCIF（计数器增加），RTCALF（报警）	13
系统控制	外部中断 0（EINT0）	14
系统控制	外部中断 1（EINT1）	15
系统控制	外部中断 2（EINT2）	16

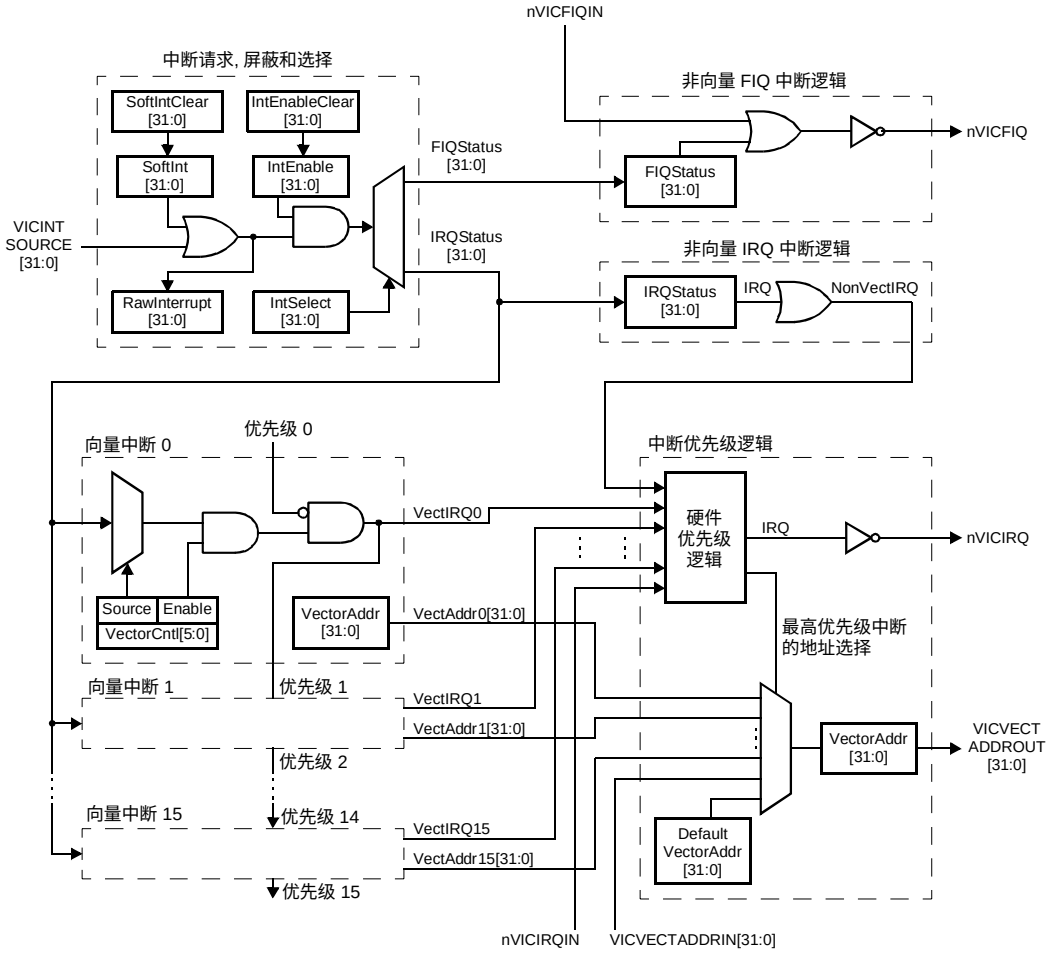


图 12 向量中断控制器方框图

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 Flash 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（位于系统控制模块当中）配置为 RAM 模式来实现这一点。

虽然可以选择多个中断源来产生 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。另外，为了能够服务下次中断，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。唯一的方法是在中断返回之前通过 VICIntEnClr 禁止 VIC 中断。

举例：

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

VICIntSelect = 0x0000 0000 (SPI, I2C, UART1 和 UART0 为 IRQ => bit10, bit9, bit7 和 bit6=0)

VICIntEnable = 0x0000 06C0 (SPI, I2C, 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 SPI IRQ 服务程序的起始地址)

VICVectCntl0 = 0x0000 0026 (UART0 中断源使能为优先级 0（最高级）)

VICVectCntl1 = 0x0000 002A (SPI 中断源使能为优先级 1)

在任何 IRQ 请求（SPI, I^2C, UART0 或 UART1）产生之后，微控制器跳转到地址 0x00000018 执行代码。对于向量和非向量 IRQ，可在地址 0x18 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 SPI 请求，VICVectAddr 等于 VICVectAddr1。如果 UART0 和 SPI 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。

6. GPIO

特性

- 单个位的方向控制
- 单独控制输出的置位和清零
- 所有 I/O 口在复位后默认为输入

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

管脚描述

表 43 GPIO 管脚描述

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

寄存器描述

GPIO 包含 4 个寄存器，见表 44。

表 44 GPIO 寄存器映射

地址	名称	描述	访问
0xE0028000	IOPIN	GPIO 管脚值寄存器。不管方向和模式如何设定，管脚的当前状态都可从该寄存器中读出。	只读
0xE0028004	IOSET	GPIO 0 输出置位寄存器。该寄存器 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。	读/置位
0xE0028008	IODIR	GPIO 0 方向控制寄存器。该寄存器单独控制每个 I/O 口的方向。	读/写
0xE002800C	IOCLR	GPIO 0 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。	只清零

GPIO 管脚值寄存器（IOPIN - 0xE0028000）

该寄存器提供 GPIO 管脚的值。它反映了外部环境对管脚的影响。

注：用于测试时，写该寄存器会将值保存到输出寄存器，不需要使用 IOSET 和 IOCLR 寄存器。该特性在应用中几乎毫无用处，因为不可能对该寄存器中单个字节执行写操作。

表 45 GPIO 管脚值寄存器（IOPIN - 0xE0028000）

IOPIN	描述	复位值
31:0	GPIO 管脚值。Bit0 对应于 P0.0 ... Bit31 对应于 P0.31	未定义

GPIO 输出置位寄存器（IOSET - 0xE0028004）

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，写 IOSET 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管脚的影响。

表 46 GPIO 输出置位寄存器（IOSET - 0xE0028004）

IOSET	描述	复位值
31:0	输出置位。Bit0 对应于 P0.0 ... Bit31 对应于 P0.31	0

GPIO 输出清零寄存器（IOCLR - 0xE002800C）

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。写入 0 无效。如果一个管脚被配置为输入或第二功能，写 IOCLR 无效。

表 47 GPIO 输出清零寄存器（IOCLR - 0xE002800C）

IOCLR	描述	复位值
31:0	输出清零。Bit0 对应于 P0.0 ... Bit31 对应于 P0.31	0

GPIO 方向寄存器（IODIR - 0xE0028008）

当管脚配置为 GPIO 模式时，可使用该寄存器控制管脚的方向。任意管脚的方向位的设置必须与管脚功能一致。

表 48 GPIO 方向寄存器（IODIR - 0xE0028008）

IODIR	描述	复位值
31:0	方向控制位（0＝输入，1＝输出）。Bit0 控制 P0.0 ... Bit31 控制 P0.31	0

GPIO 使用注意事项

如果指定输出管脚在 GPIO 输出置位寄存器（IOSET）和 GPIO 输出清零寄存器（IOCLR）中的对应位都置位，那么管脚的输出电平取决于后写入的寄存器的值。例如：

IOSET = 0x0000 0080

IOCLR = 0x0000 0080

P0.7 输出电平为低，因为写 GPIO 清零寄存器在写置位寄存器之后。

7. 管脚连接模块

特性

- 可实现独立的管脚配置

应用

管脚连接模块的用途是将管脚配置为需要的功能。

描述

管脚连接模块可以使所选管脚具有 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只读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。

寄存器描述

管脚连接模块包含两个寄存器，见表 49。

表 49 管脚连接模块寄存器映射

地址	名称	描述	访问
0xE002C000	PINSEL0	管脚选择寄存器 0	读/写
0xE002C004	PINSEL1	管脚选择寄存器 1	读/写

管脚功能选择寄存器 0（PINSEL0 - 0xE002C000）

PINSEL0 寄存器按照表 50 当中的设定来控制管脚的功能。IODIR 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 50 管脚选择寄存器 0（PINSEL0 - 0xE002C000）

PINSEL0	管脚名称	值		功能	复位值
1:0	P0.0	0	0	GPIO P0.0	0
		0	1	TxD (UART 0)	
		1	0	PWM1	
		1	1	保留	
3:2	P0.1	0	0	GPIO P0.1	0
		0	1	RxD (UART 0)	
		1	0	PWM3	
		1	1	保留	
5:4	P0.2	0	0	GPIO P0.2	0
		0	1	SCL (I^2C)	
		1	0	CAP0.0 (定时器 0)	
		1	1	保留	
7:6	P0.3	0	0	GPIO P0.3	0
		0	1	SDA (I^2C)	
		1	0	MAT0.0 (定时器 0)	
		1	1	保留	
9:8	P0.4	0	0	GPIO P0.4	0
		0	1	SCK (SPI)	
		1	0	CAP0.1 (定时器 0)	
		1	1	保留	

续上表

PINSEL0	管脚名称	值		功能	复位值
11:10	P0.5	0	0	GPIO P0.5	0
		0	1	MISO (SPI)	
		1	0	MAT0.1 (定时器 0)	
		1	1	保留	
13:12	P0.6	0	0	GPIO P0.6	0
		0	1	MOSI (SPI)	
		1	0	CAP0.2 (定时器 0)	
		1	1	保留	
15:14	P0.7	0	0	GPIO P0.7	0
		0	1	SSEL (SPI)	
		1	0	PWM2	
		1	1	保留	
17:16	P0.8	0	0	GPIO P0.8	0
		0	1	TxD (UART 1)	
		1	0	PWM4	
		1	1	保留	
19:18	P0.9	0	0	GPIO P0.9	0
		0	1	RxD (UART 1)	
		1	0	PWM6	
		1	1	保留	
21:20	P0.10	0	0	GPIO P0.10	0
		0	1	RTS (UART 1)	
		1	0	CAP1.0 (定时器 1)	
		1	1	保留	
23:22	P0.11	0	0	GPIO P0.11	0
		0	1	CTS (UART 1)	
		1	0	CAP1.1 (定时器 1)	
		1	1	保留	
25:24	P0.12	0	0	GPIO P0.12	0
		0	1	DSR (UART 1)	
		1	0	MAT1.0 (定时器 1)	
		1	1	保留	
27:26	P0.13	0	0	GPIO P0.13	0
		0	1	DTR (UART 1)	
		1	0	MAT1.1 (定时器 1)	
		1	1	保留	
29:28	P0.14	0	0	GPIO P0.14	0
		0	1	CD (UART 1)	
		1	0	EINT1	
		1	1	保留	

续上表

PINSEL0	管脚名称	值		功能	复位值
31:30	P0.15	0	0	GPIO P0.15	0
		0	1	RI (UART 1)	
		1	0	EINT2	
		1	1	保留	

管脚功能选择寄存器 1（PINSEL1 - 0xE002C004）

PINSEL1 寄存器按照表 51 当中的设定来控制管脚的功能。IODIR 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。在复位时拉低 DBGSEL 时，只有管脚 P0.17-P0.31 的功能控制有效。

表 51 管脚选择寄存器 1（PINSEL1 - 0xE002C004）

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	CAP1.2 (定时器 1)	
		1	0	保留	
		1	1	保留	
5:4	P0.18	0	0	GPIO P0.18	0
		0	1	CAP1.3 (定时器 1)	
		1	0	保留	
		1	1	保留	
7:6	P0.19	0	0	GPIO P0.19	0
		0	1	MAT1.2 (定时器 1)	
		1	0	保留	
		1	1	保留	
9:8	P0.20	0	0	GPIO P0.20	0
		0	1	MAT1.3 (定时器 1)	
		1	0	保留	
		1	1	保留	
11:10	P0.21	0	0	GPIO P0.21	0
		0	1	PWM5	
		1	0	保留	
		1	1	保留	
13:12	P0.22	0	0	GPIO P0.22	0
		0	1	保留	
		1	0	保留	
		1	1	保留	

续上表

PINSEL1	管脚名称	值		功能	复位值
15:14	P0.23	0	0	GPIO P0.23	0
		0	1	保留	
		1	0	保留	
		1	1	保留	
17:16	P0.24	0	0	GPIO P0.24	0
		0	1	保留	
		1	0	保留	
		1	1	保留	
19:18	P0.25	0	0	GPIO P0.25	0
		0	1	保留	
		1	0	保留	
		1	1	保留	
21:20	P0.26	0	0	GPIO P0.26	0
		0	1	保留	
		1	0	保留	
		1	1	保留	
23:22	P0.27	0	0	GPIO P0.27	0
		0	1	TRST	
		1	0	保留	
		1	1	保留	
25:24	P0.28	0	0	GPIO P0.28	0
		0	1	TMS	
		1	0	保留	
		1	1	保留	
27:26	P0.29	0	0	GPIO P0.29	0
		0	1	TCK	
		1	0	保留	
		1	1	保留	
29:28	P0.30	0	0	GPIO P0.30	0
		0	1	TDI	
		1	0	保留	
		1	1	保留	
31:30	P0.31	0	0	GPIO P0.31	0
		0	1	TDO	
		1	0	保留	
		1	1	保留	

管脚功能选择寄存器值

PINSEL 寄存器控制器件管脚的功能，如表 52 所示。每一对寄存器位对应一个特定的器件管脚。

表 52 管脚功能选择寄存器位

PINSEL0 和 PINSLE1 的值		功能	复位值
0	0	首选（默认）功能，通常为 GPIO 口	0
0	1	第一可选功能	
1	0	第二可选功能	
1	1	保留	

只有当管脚选择 GPIO 功能时，IODIR 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分布，因此每个管脚可能有不同的功能。详见对应的器件手册。

8. UART 0

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
- 内置波特率发生器

管脚描述

表 53 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

寄存器描述

表 54 UART0 寄存器映射

地址偏移	名称	描述	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	访问	复位值
0xE000C000 DLAB=0	U0RBR	接收缓冲	MSB 读数据 LSB								RO	未定义
0xE000C000 DLAB=0	U0THR	发送保持	MSB 写数据 LSB								WO	NA
0xE000C004 DLAB=0	U0IER	中断使能	0	0	0	0	0	使能 Rx 线状态中断	使能 THRE 中断	使能 Rx 数据可用中断	R / W	0
0xE000C008	U0IIR	中断 ID	FIFO 使能		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01
0xE000C008	U0FCR	FIFO 控制	Rx 触发		保留		DMA 模式选择	Tx FIFO 复位	Rx FIFO 复位	FIFO 使能	WO	0
0xE000C00C	U0LCR	线控制	DLAB	设置间隔	奇偶选择		奇偶使能	停止位个数	字长度选择		R/W	0
0xE000C014	U0LSR	线状态	Rx FIFO 错误	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60
0xE000C01C	U0SCR	高速缓存	MSB LSB								R/W	0
0xE000C000 DLAB=1	U0DLL	除数锁存 LSB	MSB LSB								R/W	0x01
0xE000C004 DLAB=1	U0DLM	除数锁存 MSB	MSB LSB								R/W	0

UART0 包含 10 个 8 位寄存器，见表 54。除数锁存访问位（DLAB）位于 U0LCR7，它使能对除数锁存的访问。

接收器缓存寄存器（U0RBR - 0xE000C000，DLAB=0，只读）

U0RBR 是 U0Rx 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 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，除数锁存访问位（DLAB）必须为 0。U0RBR 为只读寄存器。

表 55 接收器缓存寄存器（U0RBR - 0xE000C000，DLAB=0，只读）

U0RBR	功能	描述	复位值
7:0	接收器缓存	接收器缓存寄存器包含 U0Rx FIFO 当中最早接收到的字节	未定义

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

U0THR 是 U0Tx 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，除数锁存访问位（DLAB）必须为 0。U0THR 为只写寄存器。

表 56 发送器保持寄存器（U0THR - 0xE000C000，DLAB=0，只写）

U0THR	功能	描述	复位值
7:0	发送器保持	写发送器保持寄存器使数据保存到 U0Tx 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。	N/A

除数锁存 LSB 寄存器（U0DLL - 0xE000C000，DLAB=1）

除数锁存 MSB 寄存器（U0DLM - 0xE000C004，DLAB=1）

除数锁存是波特率发生器的一部分，它保存了用于产生波特率时钟的 VPB 时钟（pclk）分频值，波特率时钟必须是波特率的 16 倍。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'h0000 被看作是'h0001，因为除数是不允许为 0 的。当访问除数锁存寄存器时，除数锁存访问位（DLAB）必须为 1。

表 57 除数锁存 LSB 寄存器（U0DLL - 0xE000C000，DLAB=1）

U0DLL	功能	描述	复位值
7:0	除数锁存 LSB 寄存器	除数锁存 LSB 寄存器与 U0DLM 寄存器一起决定 UART0 的波特率。	N/A

表 58 除数锁存 MSB 寄存器（U0DLM - 0xE000C004，DLAB=1）

U0DLM	功能	描述	复位值
7:0	除数锁存 MSB 寄存器	除数锁存 MSB 寄存器与 U0DLL 寄存器一起决定 UART0 的波特率。	N/A

中断使能寄存器（U0IER - 0xE000C004，DLAB=0）

U0IER 用于使能 4 个 UART0 中断源。

表 59 中断使能寄存器（U0IER - 0xE000C004，DLAB=0）

U0IER	功能	描述	复位值
0	RBR 中断使能	0：禁止 RDA 中断 1：使能 RDA 中断 U0IER0 使能接收数据可用中断。它还控制字符接收超时中断。	0
1	THRE 中断使能	0：禁止 THRE 中断 1：使能 THRE 中断 U0IER1 使能 THRE 中断。该中断的状态可从 U0LSR5 读出。	0
2	Rx 线状态中断使能	0：禁止 Rx 线状态中断 1：使能 Rx 线状态中断 U0IER1 使能 U0Rx 线状态中断。该中断的状态可从 U0LSR[4:1]读出。	0
7:3	保留	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

中断标识寄存器（U0IIR - 0xE000C008，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 60 中断标识寄存器（U0IIR - 0xE000C008，只读）

U0IIR	功能	描述	复位值
0	中断挂起	0：至少有 1 个中断被挂起 1：没有挂起的中断 U0IIR0 为低有效。挂起的中断可通过 U0IIR3:1 确定。	0
3:1	中断标识	011：1. 接收线状态（RLS） 010：2a. 接收数据可用（RDA） 110：2b. 字符超时指示（CTI） 001：3. THRE 中断 U0IIR3:1 指示对应于 Rx FIFO 的中断。上面未列出的 U0IIR3:1 的其它组合都为保留值（000, 100, 101, 111）	0
5:4	保留	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
7:6	FIFO 使能	这些位等效于 U0FCR0	0

中断的处理见表 61。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（U0IIR3:1=011）是最高优先级的中断。只要产生 4 个错误条件（溢出错误 OE、优先级错误 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 U0Rx 错误条件可通过查看 U0LSR4:1 得到。当读取 U0LSR 时清除中断。

UART0 RDA 中断（U0IIR3:1=010）与 CTI 中断（IIR3:1=110）共用第二优先级。当 U0Rx FIFO 到达 U0FCR7:6 所定义的触发点时，RDA 被激活。当 U0Rx 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（U0IIR3:1=010）为第二优先级中断。当 U0Rx 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 U0Rx FIFO 动作时，产生该中断。U0Rx FIFO 的任何动作（读或写 U0RSR）都将清除该中断。当接收到的信息不是触发值的倍数时，CTI 中断将会清空 U0RBR。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，那么前 100 个字符将使 CPU 接收 10 个 RDA 中断，而剩下的 5 个字符使 CPU 接收 1 到 5 个 CTI 中断（取决于服务程序）。

表 61 中断处理

U0IIR[3:0]	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, PE, FE, 或 BI	LSR 读操作
0100	第二	Rx 数据可用	Rx 数据可用或 FIFO 模式下（FCR0=1）到达触发点	U0RBR 读或 FIFO 低于触发值
1100	第二	字符超时指示	Rx FIFO 包含至少 1 个字符并且在接收 3.5 到 4.5 字符的时间内没有发生 Rx FIFO 动作。 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
0010	第三	THRE	THRE	U0IIR 读或 THR 写操作
注：“0000”, “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”, “1101”, “1110”, “1111”为保留值。				

UART0 THRE 中断(U0IIR3:1)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 事件之后没有在 U0THR 中存在至少 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（U0IIR3:1=001）时，THRE 中断复位。

FIFO 控制寄存器（U0FCR - 0xE000C008）

U0FCR 控制 UART0 Rx 和 Tx FIFO 的操作。

表 62 UART0 FIFO 控制寄存器（U0FCR - 0xE000C008）

U0FCR	功能	描述	复位值
0	FIFO 使能	高电平使能对 UART0 Rx 和 Tx FIFO 以及 U0FCR7:1 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 FIFO 清空。	0
1	Rx FIFO 复位	该位置位会清零 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	该位置位会清零 UART0 Tx FIFO 中的所有字节并复位指针逻辑。该位自动清零。	0
3	DMA 模式选择	0：单传送模式 1：多传送模式 U0FCR3 影响 RXRDY_N 和 TXRDY_N 管脚的操作。	0
5:4	保留	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
7:6	Rx 触发选择	00：触发点 1（默认’h1） 01：触发点 2（默认’h4） 10：触发点 3（默认’h8） 11：触发点 4（默认’he） 这两个位决定在激活中断之前，UAT0 接收 FIFO 必须写入多少个字符。4 个触发点由用户在编译时定义，可以选择所需要的触发深度。	0

线控制寄存器（U0LCR - 0xE000C00C）

U0LCR 决定发送和接收数据字符的格式。

表 63 线控制寄存器（U0LCR - 0xE000C00C）

U0LCR	功能	描述	复位值
1:0	字长度选择	00：5 位字符长度 01：6 位字符长度 10：7 位字符长度 11：8 位字符长度	0
2	停止位选择	0：1 个停止位 1：2 个停止位（如果 U0LCR[1:0]=00 则为 1.5）	0
3	奇偶使能	0：禁止奇偶产生和校验 1：使能奇偶产生和校验	0
5:4	奇偶选择	00：奇数 01：偶数 10：强制为 1 11：强制为 0	0
6	间隔控制	0：禁止间隔发送 1：使能间隔发送 当 U0LCR6=1 时，输出管脚 TxD 强制为逻辑 0。	0
7	除数锁存访问位	0：禁止访问除数锁存 1：使能访问除数锁存	0

线状态寄存器（U0LSR - 0xE000C014，只读）

U0LSR 为只读寄存器，它提供 UART0 发送和接收模块的状态信息。

表 64 线状态寄存器（U0LSR - 0xE000C014，只读）

U0LSR	功能	描述	复位值
0	接收数据就绪（RDR）	0：U0RBR 为空 1：U0RBR 包含有效数据 当 U0RBR 包含未读取的字符时，U0LSR0 置位；当 U0RBR FIFO 为空时，U0LSR0 清零。	0
1	溢出错误（OE）	0：溢出错误状态未激活 1：溢出错误状态激活 溢出错误条件在错误发生后立即设置。U0LSR 读操作清零 U0LSR1。当 U0RSR 已经有新的字符就绪而 U0RBR FIFO 已满时，U0LSR1 置位。此时 U0RBR FIFO 不会被覆盖，U0RSR 中的字符将丢失。	0
2	奇偶错误（PE）	0：奇偶错误状态未激活 1：奇偶错误状态激活 当接收字符的奇偶位处于错误状态时产生一个奇偶错误。U0LSR 读操作清零 U0LSR2 位。奇偶错误检测时间取决于 U0FCR0。奇偶错误与 U0RBR FIFO 中读出的字符相关。	0

续上表

U0LSR	功能	描述	复位值
3	帧错误（FE）	0：帧错误状态未激活 1：帧错误状态激活 当接收字符的停止位为 0 时，产生帧错误。U0LSR 读操作清零 U0LSR3。帧错误检测时间取决于 U0FCR0。帧错误与 U0RBR FIFO 中读出的字符相关。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。	0
4	间隔中断（BI）	0：间隔中断状态未激活 1：间隔中断状态激活 在发送整个字符（起始位、数据、奇偶位和停止位）过程中 RxD 如果都保持逻辑 0，则产生间隔中断。当检测到中断条件时，接收器立即进入空闲状态直到 RxD 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 U0FCR0。间隔中断与 U0RBR FIFO 中读出的字符相关。	0
5	发送保持寄存器空（THRE）	0：U0THR 包含有效数据 1：U0THR 空 当检测到 THR 空时，THRE 置位，U0THR 写操作清零该位。	1
6	发送器空（TEMT）	0：U0THR 和/或 U0TSR 包含有效数据 1：U0THR 和 U0TSR 空 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。	1
7	Rx FIFO 错误（RXFE）	0：U0RBR 中没有 Rx 错误，或 U0FCR0=0 1：U0RBR 包含至少一个 Rx 错误 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，U0LSR7 置位。当读取 U0LSR 寄存器并且 FIFO 中不再有错误时，U0LSR7 清零。	0

高速缓存寄存器（U0SCR - 0xE000C01C）

在 UART 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 65 高速缓存寄存器（U0SCR - 0xE000C01C）

U0SCR	功能	描述	复位值
7:0	-	一个可读可写的字节	0

结构

UART 的结构如图 12 所示。VPB 接口提供 CPU 或主机与 UART 之间的通信连接。

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

UART0 发送器模块 Tx 接受 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 THR 中的数据并将数据通过串行输出管脚 TxD 发送。

波特率发生器模块 U0BRG 产生 Tx 模块所使用的定时。U0BRG 模块时钟源为 pclk。主时钟与 U0DLL

和 U0DLM 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。

中断接口包含寄存器 U0IER 和 U0IIR。中断接口接收几个由 U0Tx 和 U0Rx 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。

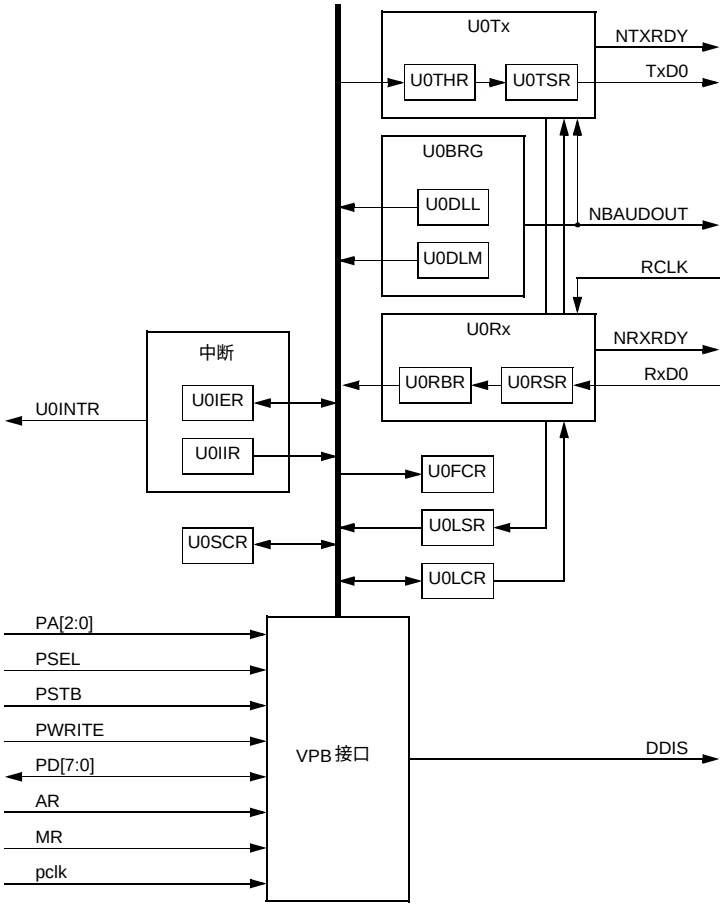


图 13 UART0 方框图

9. UART1

特性

- UART1 与 UART0 相同，只是增加了一个调制解调器（modem）接口
- 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
- 包含标准调制解调器接口信号

管脚描述

表 66 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	**清零以发送** 有效低电平信号指示外部 modem 的接收是否已经准备就绪，UART 数据可通过 TxD 发送。在 modem 的正常操作中(U1MCR4=0)，该信号的补码保存在 MSR4 中。状态改变信息保存在 U1MSR0 中，如果第 4 优先级中断使能(U1IER3=1)，该信息将作为中断源。
DCD1	输入	**数据载波检测** 有效低信号指示外部 modem 是否已经与 UART 建立了通信连接，可以进行数据交换。在 modem 的正常操作中(U1MCR4=0)，该信号的补码保存在 U1MSR7 中。状态改变信息保存在 U1MSR3 中，如果第 4 优先级中断使能(U1IER3=1)，该信息将作为中断源。
DSR1	输入	**数据设置就绪** 有效低电平指示外部 modem 是否准备建立与 UART 的连接。在 modem 的正常操作中(U1MCR4=0)，该信号的补码保存在 U1MSR5 中。状态改变信息保存在 U1MSR1 中，如果第 4 优先级中断使能(U1IER3=1)，该信息将作为中断源。
DTR1	输出	**数据终止就绪** 有效低电平指示 UART 准备建立与外部 modem 的连接。该信号的补码保存在 U1MCR0 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 的正常操作中(U1MCR4=0)，该信号的补码保存在 U1MSR6 中。状态改变信息保存在 U1MSR2 中，如果第 4 优先级中断使能(U1IER3=1)，该信息将作为中断源。
RTS1	输出	**请求发送** 有效低电平指示 UART 打算向外部 modem 发送数据。该信号的补码保存在 U1MCR1 中。

寄存器描述

表 67 UART1 寄存器映射

地址偏移	名称	描述	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	访问	复位值
0xE0010000 DLAB=0	U1RBR	接收缓冲	MSB	读数据						LSB	RO	未定义
0xE0010000 DLAB=0	U1THR	发送保持	MSB	写数据						LSB	WO	NA
0xE0010004 DLAB=0	U1IER	中断使能	0	0	0	0	使能 Modem 状态中断	使能 Rx 线状态中断	使能 THRE 中断	使能 Rx 数据可用中断	R/W	0
0xE0010008	U1IIR	中断 ID	FIFO 使能		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01
0xE0010008	U1FCR	FIFO 控制	Rx 触发		保留		DMA 模式选择	Tx FIFO 复位	Rx FIFO 复位	FIFO 使能	WO	0
0xE001000C	U1LCR	线控制	DLAB	设置间隔	奇偶选择		奇偶使能	停止位个数	字长度选择		R/W	0
0xE0010010	U1MCR	Modem 控制	0	0	0	回送	0	0	RTS	DTR	R/W	0
0xE0010014	U1LSR	线状态	Rx FIFO 错误	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60
0xE0010018	U1MSR	Modem 状态	DCD	RI	DSR	CTS	Delta DCD	后沿 RI	Delta DSR	Delta CTS	RO	0
0xE001001C	U1SCR	高速缓存	MSB							LSB	R/W	0
0xE0010000 DLAB=1	U1DLL	除数锁存 LSB	MSB							LSB	R/W	0x01
0xE0010004 DLAB=1	U1DLM	除数锁存 MSB	MSB							LSB	R/W	0

UART1 包含 12 个 8 位寄存器，见表 67。除数锁存访问位（DLAB）位于 U1LCR7，它使能对除数锁存的访问。

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 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，除数锁存访问位（DLAB）必须为 0。U1RBR 为只读寄存器。

表 68 接收器缓存寄存器（U1RBR - 0xE0010000，DLAB=0，只读）

U1RBR	功能	描述	复位值
7:0	接收器缓存	接收器缓存寄存器包含 Rx FIFO 当中最早接收到的字节	未定义

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

U1THR 是 UART1 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 U1THR，除数锁存访问位（DLAB）必须为 0。U1THR 为只写寄存器。

表 69 发送器保持寄存器（U1THR - 0xE0010000，DLAB=0，只写）

U1THR	功能	描述	复位值
7:0	发送器保持	写发送器保持寄存器使数据保存到 Tx FIFO 当中。当字节到达 FIFO 的最低部并且发送器就绪时，该字节将被发送。	N/A

UART1 除数锁存 LSB 寄存器（U1DLL - 0xE0010000，DLAB=1）

UART1 除数锁存 MSB 寄存器（U1DLM - 0xE0010004，DLAB=1）

UART1 的除数锁存是波特率发生器的一部分，它保存了用于产生波特率时钟的 VPB 时钟（pclk）分频值，波特率时钟必须是波特率的 16 倍。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'h0000 被看作是'h0001，因为除数是不允许为 0 的。当访问除数锁存寄存器时，除数锁存访问位（DLAB）必须为 1。

表 70 UART1 除数锁存 LSB 寄存器（U1DLL - 0xE0010000，DLAB=1）

U1DLL	功能	描述	复位值
7:0	除数锁存 LSB 寄存器	除数锁存 LSB 寄存器与 U1DLM 寄存器一起决定 UART1 的波特率。	N/A

表 71 除数锁存 MSB 寄存器（DLM - 0xE0010004，DLAB=1）

U1DLM	功能	描述	复位值
7:0	除数锁存 MSB 寄存器	除数锁存 MSB 寄存器与 U1DLL 寄存器一起决定 UART1 的波特率。	N/A

中断使能寄存器（U1IER - 0xE0010004，DLAB=0）

U1IER 用于使能 4 个中断源。

表 72 中断使能寄存器（U1IER - 0xE0010004，DLAB=0）

U1IER	功能	描述	复位值
0	RBR 中断使能	0：禁止 RDA 中断 1：使能 RDA 中断 U1IER0 使能接收数据可用中断。在 FIFO 模式下，U1IER0 还控制字符接收超时中断。	0
1	THRE 中断使能	0：禁止 THRE 中断 1：使能 THRE 中断 U1IER1 使能 THRE 中断。该中断的状态可从 U1LSR5 读出。	0
2	Rx 线状态中断使能	0：禁止 Rx 线状态中断 1：使能 Rx 线状态中断 U1IER1 使能 UART1 Rx 线状态中断。该中断的状态可从 U1LSR[4:1]读出。	0
3	Modem 状态中断使能	0：禁止 Modem 中断 1：使能 Modem 中断 U1IER3 使能 modem 中断。中断的状态可从 U1MSR[3:0]读取。	0
7:4	保留	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

中断标识寄存器（U1IIR - 0xE0010008，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 73 中断标识寄存器（U1IIR - 0xE0010008，只读）

U1IIR	功能	描述	复位值
0	中断挂起	0：至少有 1 个中断被挂起 1：没有挂起的中断 U1IIR0 为低有效。挂起的中断可通过 U1IIR3:1 确定。	0
3:1	中断标识	011：1. 接收线状态（RLS） 010：2a. 接收数据可用（RDA） 110：2b. 字符超时指示（CTI） 001：3. THRE 中断 000：4. Modem 中断 U1IIR3:1 指示对应于 Rx FIFO 的中断。上面未列出的 U1IIR3:1 的其它组合都为保留值（100, 101, 111）	0
5:4	保留	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
7:6	FIFO 使能	这些位等效于 U1FCR0	0

中断的处理见表 74。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 IIR 来清除中断。

RLS 中断（U1IIR3:1=011）是最高优先级的中断。只要产生 4 个错误条件（溢出错误 OE、优先级错误 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 U1LSR4:1 得到。当读取 U1LSR 时清除中断。

RDA 中断（U1IIR3:1=010）与 CTI 中断（U1IIR3:1=110）共用第二优先级。当 UART1 Rx FIFO 到达 FCR7:6 所定义的触发点时，RDA 被激活。当 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（U1IIR3:1=010）为第二优先级中断。当 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 Rx FIFO 动作时，产生该中断。Rx FIFO 的任何动作（读或写 U1RSR）都将清除该中断。当接收到的信息不是触发值的倍数时，CTI 中断将会清空 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，那么前 100 个字符将使 CPU 接收 10 个 RDA 中断，而剩下的 5 个字符使 CPU 接收 1 到 5 个 CTI 中断（取决于服务程序）。

表 74 中断处理

U1IIR[3:0]	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, PE, FE, 或 BI	U1LSR 读操作
0100	第二	Rx 数据可用	Rx 数据可用或 FIFO 模式下（FCR0=1）到达触发点	U1RBR 读或 FIFO 低于触发值
1100	第二	字符超时指示	Rx FIFO 包含至少 1 个字符并且在接收 3.5 到 4.5 字符的时间内没有发生 Rx FIFO 动作。 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
0010	第三	THRE	THRE	U1IIR 读或 THR 写操作
0000	第四	Modem 状态	CTS, DSR, RI 或 DCD	MSR 读操作
注：“0000”, “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”, “1101”, “1110”, “1111”为保留值。				

THRE 中断(U1IIR3:1)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 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 事件之后没有在 THR 中存在至少 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（U1IIR3:1=001）时，THRE 中断复位。

Modem 中断（U1IIR3:1=000）是最低优先级中断，只要在 modem 输入管脚、DCD、DSR 或 CTS 上发生任何状态变化，该中断就会激活。此外，modem 输入口 RI 上低到高电平的跳变会产生一个 modem 中断。modem 中断源可通过检查 U1MSR3:0 得到。读取 U1MSR 将清除 modem 中断。

FIFO 控制寄存器（U1FCR - 0xE0010008）

U1FCR 控制 Rx 和 Tx FIFO 的操作。

表 75 FIFO 控制寄存器（U1FCR - 0xE0010008）

U1FCR	功能	描述	复位值
0	FIFO 使能	高电平使能对 Rx 和 Tx FIFO 以及 U1FCR7:1 的访问。该位必须置位以实现正确的 UART 操作。该位的任何变化都将使 FIFO 清空。	0
1	Rx FIFO 复位	该位置位会清零 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	该位置位会清零 Tx FIFO 中的所有字节并复位指针逻辑。该位自动清零。	0
3	保留	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	保留	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

续上表

U1FCR	功能	描述	复位值
7:6	Rx 触发选择	00：触发点 1（默认 h1） 01：触发点 2（默认 h4） 10：触发点 3（默认 h8） 11：触发点 4（默认 he） 这两个位决定在激活中断之前，Rx FIFO 必须写入多少个字符。4 个触发点由用户在编译时定义，可以选择所需要的触发深度。	0

线控制寄存器（U1LCR - 0xE001000C）

U1LCR 决定发送和接收数据字符的格式。

表 76 线控制寄存器（U1LCR - 0xE001000C）

U1LCR	功能	描述	复位值
1:0	字长度选择	00：5 位字符长度 01：6 位字符长度 10：7 位字符长度 11：8 位字符长度	0
2	停止位选择	0：1 个停止位 1：2 个停止位（如果 U1LCR[1:0]=00 则为 1.5）	0
3	奇偶使能	0：禁止奇偶产生和校验 1：使能奇偶产生和校验	0
5:4	奇偶选择	00：奇数 01：偶数 10：强制为 1 11：强制为 0	0
6	间隔控制	0：禁止间隔发送 1：使能间隔发送 当 U1LCR6=1 时，输出管脚 TxD 强制为逻辑 0。	0
7	除数锁存访问位	0：禁止访问除数锁存 1：使能访问除数锁存	0

Modem 控制寄存器（U1MCR - 0xE0010010）

U1MCR 使能 modem 的回送模式并控制 modem 的输出信号。

表 77 U1MCR 位描述（U1MCR - 0xE0010010）

U1MCR	功能	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。	
2	保留	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	保留	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

续上表

U1MCR	功能	描述	复位值
4	回写模式选择	0：禁止 modem 回写模式 1：使能 modem 回写模式 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。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 位就可可产生 modem 状态中断。	0
7:5	保留	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

线状态寄存器（U1LSR - 0xE0010014，只读）

U1LSR 为只读寄存器，它提供发送和接收模块的状态信息。

表 78 线状态寄存器（U1LSR - 0xE0010014，只读）

U1LSR	功能	描述	复位值
0	接收数据就绪（RDR）	0：U1RBR 为空 1：U1RBR 包含有效数据 当 U1RBR 包含未读取的字符时，U1LSR0 置位；当 U1RBR FIFO 为空时，U1LSR0 清零。	0
1	溢出错误（OE）	0：溢出错误状态未激活 1：溢出错误状态激活 溢出错误条件在错误发生后立即设置。U1LSR 读操作清零 U1LSR1。当 U1RSR 已经有新的字符就绪而 U1RBR FIFO 已满时，U1LSR1 置位。此时 U1RBR FIFO 不会被覆盖，U1RSR 中的字符将丢失。	0
2	奇偶错误（PE）	0：奇偶错误状态未激活 1：奇偶错误状态激活 当接收字符的奇偶位处于错误状态时产生一个奇偶错误。U1LSR 读操作清零 U1LSR2 位。奇偶错误检测时间取决于 U1FCR0。奇偶错误与 U1RBR FIFO 中读出的字符相关。	0
3	帧错误（FE）	0：帧错误状态未激活 1：帧错误状态激活 当接收字符的停止位为 0 时，产生帧错误。U1LSR 读操作清零 U1LSR3。帧错误检测时间取决于 U1FCR0。帧错误与 U1RBR FIFO 中读出的字符相关。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。	0
4	间隔中断（BI）	0：间隔中断状态未激活 1：间隔中断状态激活 在发送整个字符（起始位、数据、奇偶位和停止位）过程中 RxD 如果都保持逻辑 0，则产生间隔中断。当检测到中断条件时，接收器立即进入空闲状态直到 RxD 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 U1FCR0。间隔中断与 U1RBR 中读出的字符相关。	0

续上表

U1LSR	功能	描述	复位值
5	发送保持寄存器空（THRE）	0：U1THR 包含有效数据 1：U1THR 空 当检测到 U1THR 空时，THRE 置位，U1THR 写操作清零该位。	1
6	发送器空（TEMT）	0：U1THR 和/或 TSR 包含有效数据 1：U1THR 和 U1TSR 空 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。	1
7	Rx FIFO 错误（RXFE）	0：U1RBR 中没有 Rx 错误，或 U1FCR0=0 1：U1RBR 包含至少一个 Rx 错误 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 RBR 时，U1LSR7 置位。当读取 U1LSR 寄存器并且 FIFO 中不再有错误时，U1LSR7 清零。	0

Modem 状态寄存器（U1MSR - 0x0E0010018）

U1MSR 是一个只读寄存器，它提供 modem 输入信号的状态信息。U1MSR3:0 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 79 Modem 状态寄存器（U1MSR - 0x0E0010018）

U1MSR	功能	描述	复位值
0	Delta CTS	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 当输入 CTS 状态发生变化时，该位置位。读取 U1MSR 时清零。	0
1	Delta DSR	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 当输入 DSR 状态发生变化时，该位置位。读取 U1MSR 时清零。	0
2	后沿 RI	0：没有检测到 modem 输入 RI 上的状态变化 1：检测到 modem 输入 RI 上的状态变化 当输入 RI 状态发生变化时，该位置位。读取 U1MSR 时清零。	0
3	Delta DCD	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 当输入 DCD 状态发生变化时，该位置位。读取 U1MSR 时清零。	0
4	CTS	清零以发送状态 输入信号 CTS 的补码。在回写模式下，该位连接到 U1MCR[1]。	0
5	DSR	数据设置就绪状态 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

高速缓存寄存器（U1SCR - 0xE001001C）

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 80 高速缓存寄存器（U1SCR - 0xE001001C）

U1SCR	功能	描述	复位值
7:0	-	一个可读可写的字节	0

结构

UART1 的结构如图 14 所示。VPB 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 缓冲寄存器 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 U1Tx 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 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 并控制中断输出管脚 INTR。中断接口接收几个由 U1Tx 和 U1Rx 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。

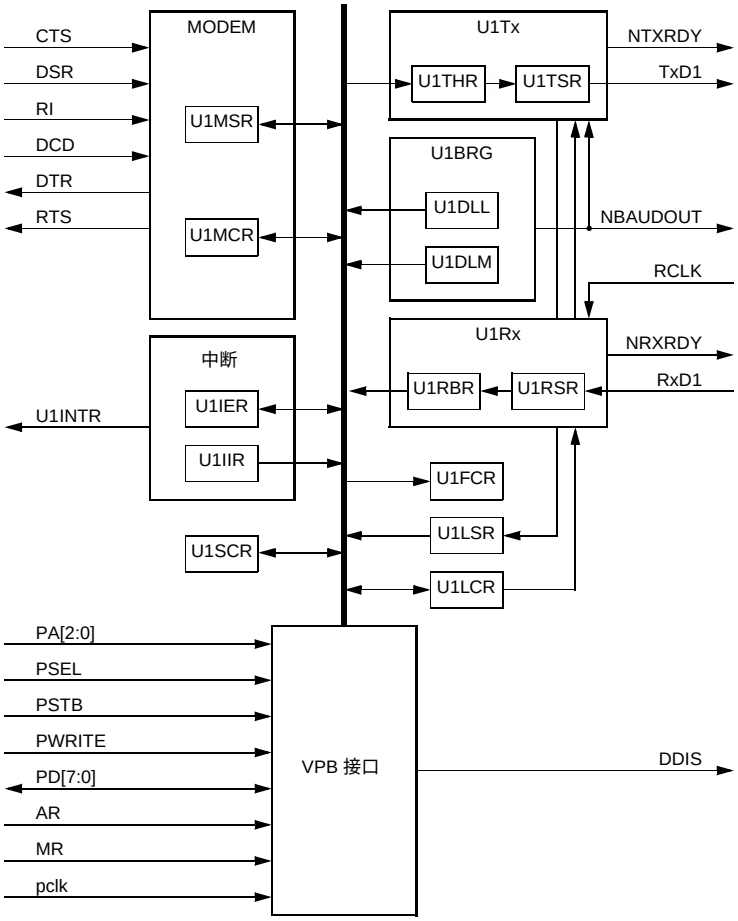


图 14 UART1 方框图

10. I²C 接口

- 标准的 I²C 总线接口
- 可配置为主机、从机或主/从机
- 可编程时钟可实现通用速率控制
- 主机从机之间双向数据传输
- 多主机总线(无中央主机)
- 同时发送的主机之间进行仲裁，避免了总线数据的冲突
- 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

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

描述

I²C 总线的典型配置如图 15 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：

- 从主发送器向从接收器发送数据。主机发送的第一个字节使从机地址。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。主机每接收一个字节返回一个应答位。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时下一个串行发送的开始，因此 I²C 总线不会被释放。

该器件提供字节方式的 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。

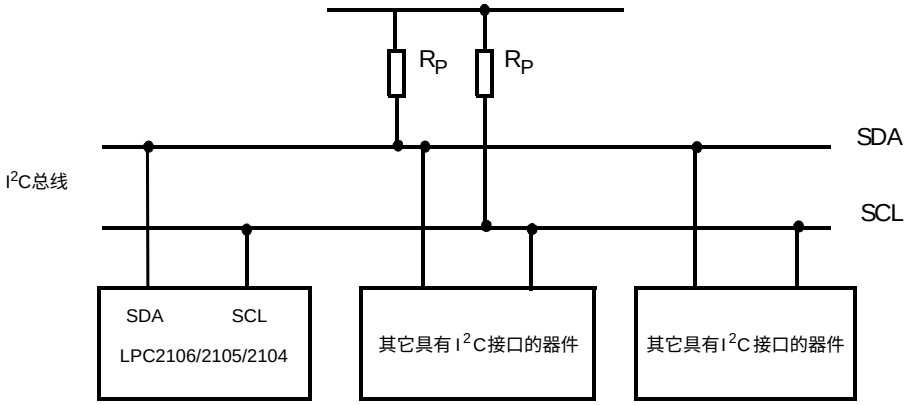


图 15 I²C 总线配置

I2C 操作模式

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 必须按照图 16 进行初始

化：

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

图 16 主模式配置

必须置位 I2EN 来使能 I^2C 功能。如果 AA 位为 0，而另一个器件成为总线的主控器时，I^2C 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 必须设置为 0。向 I2CONCLR 寄存器中的 SIC 位写入 1 可清零 SI。

第一个发送的数据包含接收器件的从地址（7 位）和数据方向位。在此模式下，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 SLA+W。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。起始和停止条件用于指示串行传输的起始和结束。

通过软件置位 STA 进入 I^2C 主发送器模式。I^2C 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。此时状态寄存器（I2STAT）中的状态代码应当为 08H。该状态代码用于指向一个中断服务程序。该中断程序将从地址和数据方向位（SLA+W）装入 I2DAT，然后清零 SI 位。向 I2CONCLR 寄存器中的 SIC 位写入 1 可清零 SI。

当从地址和方向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 18H，20H 或 38H，如果从模式使能（AA=1），可能的状态代码为 68H，78H 或 B0H。每个状态代码对应的执行动作见*《P89LPC932 使用指南》*中的表 16~19。

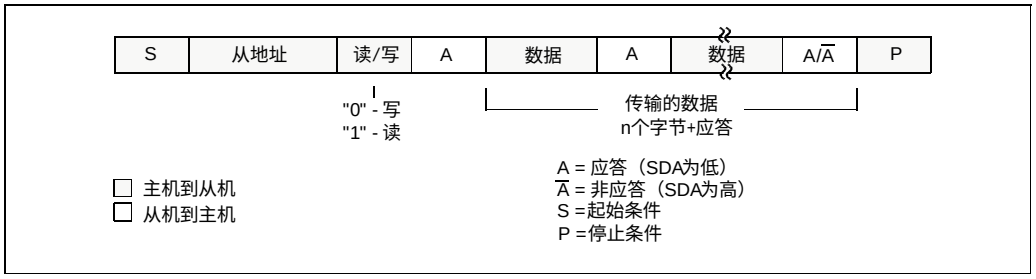


图 17 主发送器模式中的格式

主接收器模式

在主接收器模式中，数据字节接收自从发送器。传输的初始化与主发送器模式相同。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（SLA+R）装入 I^2C 数据寄存器（I2DAT），然后清零 RI 位。

当从地址和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 40H，48H 或 38H，对于从模式，可能的状态代码为 68H，78H 或 B0H。

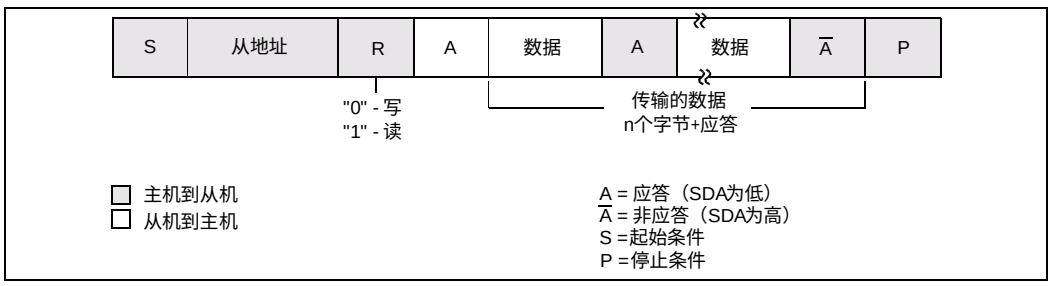


图 18 主接收器模式的格式

在一个重复的起始条件之后，I^2C 可以切换到主发送器模式。

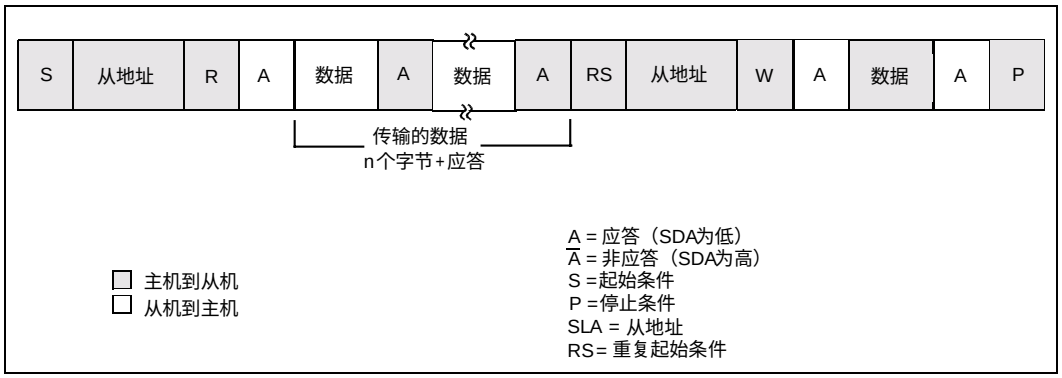


图 19 在发送重复起始条件后，主接收器切换到主发送器

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，用户必须将从地址写入从地址寄存器（I2ADR）并按照图 20 配置 I²C 控制寄存器（I2CON）：

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

图 20 从模式配置

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寻址为止。如果数据方向位为 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 I2STAT 中读出有效的状态代码。

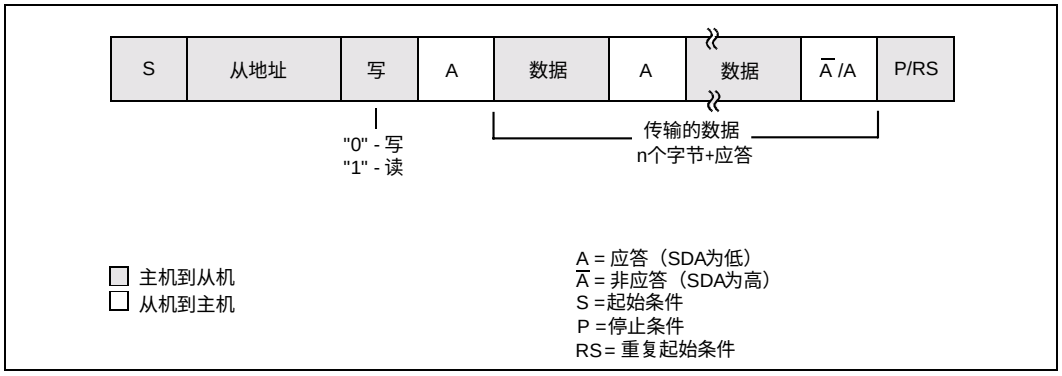


图 21 从接收器模式的格式

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，方向位指示传输的方向掉转。串行数据通过 P1.3/SDA 发送而串行时钟通过 P1.2/SCL 输入。在串行传输的开始和结束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 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 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。

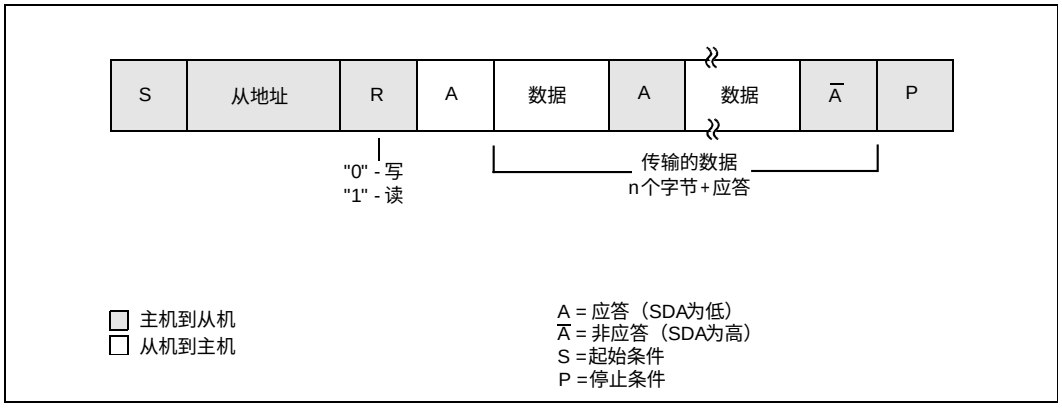


图 22 从发送器模式的格式

管脚描述

表 81 I²C 管脚描述

管脚名称	类型	描述
SDA	输入/输出	**串行数据** I²C 数据输出和输出。相关端口为开漏输出以符合 I²C 规范。
SCL	输入/输出	**串行时钟** I²C 时钟输出和输出。相关端口为开漏输出以符合 I²C 规范。

寄存器描述

I²C 接口包含 7 个寄存器，如表 82 所示。

表 82 I²C 寄存器映射

地址	名称	描述	访问	复位值
0xE001C000	I2CONSET	I²C 控制置位寄存器	读/置位	0
0xE001C004	I2STAT	I²C 状态寄存器	只读	0xF8
0xE001C008	I2DAT	I²C 数据寄存器	读/写	0
0xE001C00C	I2ADR	I²C 从地址寄存器	读/写	0
0xE001C010	I2SCLH	SCL 占空比寄存器高半字	读/写	0x04
0xE001C014	I2SCLL	SCL 占空比寄存器低半字	读/写	0x04
0xE001C018	I2CONCLR	I²C 控制清零寄存器	只清零	NA

I²C 控制置位寄存器（I2CONSET - 0xE001C000）

AA 为声明应答标志。当该位置位时，下面的任意条件之一将产生一个应答：

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，下列情况下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。

SI 为 I²C 中断标志。当进入 25 种可能的 I²C 状态中的任何一个后，该位置位。通常，I²C 中断只在空闲的从器件中用于指示一个起始条件，或在一个空闲的主器件（如果它等待使用 I²C 总线）中指示一个停止条件。向 I2CONCLR 寄存器中的 SIC 位写入 1 使 SI 位清零。

STO 为停止标志。当 STO 为 1 时，在主模式中，向 I^2C 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，向总线发送停止条件。当总线检测到停止条件时，STO 自动清零。

在从模式中，置位该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。

STA 为起始标志。当 STA=1 时，I^2C 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 I^2C 接口还没进入主模式时，I^2C 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 I^2C 接口已经处于主模式中并发送或接收了数据时，I^2C 接口会发送一个重复的起始条件。STA 可在任何时候置位，当 I^2C 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 I^2C 接口处于主模式，I^2C 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 I^2C 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。

I2EN 为 I^2C 接口使能。当该位置位时，使能 I^2C 接口；该位为 0 时，I^2C 功能被禁止。向 I2CONCLR 寄存器中的 I2ENC 位写入 1 将使 I2EN 位清零。

表 83 I^2C 控制置位寄存器（I2CONSET - 0xE001C000）

I2CONSET	功能	描述	复位值
0	保留	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	保留	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	0
3	SI	I^2C 中断标志	0
4	STO	停止标志	0
5	STA	起始标志	0
6	I2EN	I^2C 接口模式	0
7	保留	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

表 84 I^2C 控制清零寄存器（I2CONCLR - 0xE001C018）

I2CONCLR	功能	描述	复位值
0	保留	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	保留	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。向该位写入 1 清零 I2CONSET 寄存器中的 AA 位。写入 0 无效。	0
3	SIC	I^2C 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。	0
4	保留	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。向该位写入 1 清零 I2CONSET 寄存器中的 STA 位。写入 0 无效。	0
6	I2ENC	I^2C 接口禁止。向该位写入 1 清零 I2CONSET 寄存器中的 I2EN 位。写入 0 无效。	0
7	保留	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

I^2C 状态寄存器（I2STAT - 0xE001C004）

这是一个只读寄存器。它包含 I^2C 接口的状态代码。最低 3 位总是为 0。一共有 26 种可能存在的状态代码。当代码为 F8H 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 I^2C 状态。当进入其中一种状态时，SI 位将置位。完整的状态代码描述见*《P89LPC932 使用指南》*中的表 16~19。

表 85 I^2C 状态寄存器（I2STAT - 0xE001C004）

I2STAT	功能	描述	复位值
2:0	状态	这 3 个位总是为 0	0
7:3	状态	状态位	1

I^2C 数据寄存器（I2DAT - 0xE001C008）

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。在 SI 置位期间，I2DAT 中的数据保持稳定。I2DAT 中的数据移位总是从右至左进行：第一个发送的位是 MSB(bit7)，在接收字节时，第一个接收到的位存放在 I2DAT 的 MSB。

表 86 I^2C 数据寄存器（I2DAT - 0xE001C008）

I2DAT	功能	描述	复位值
7:0	状态	发送/接收数据位	0

I^2C 从地址寄存器（I2ADR - 0xE001C00C）

该寄存器可读可写，但只能在 I^2C 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（00h）被忽略。

表 87 I^2C 从地址寄存器（I2ADR - 0xE001C00C）

I2ADR	功能	描述	复位值
0	GC	通用调用位	0
7:1	地址	从模式地址	0

I^2C SCL 占空比寄存器（I2SCLH - 0xE001C010 和 I2SCLL - 0xE001C014）

软件必须通过对 I2SCLH 和 I2SCLL 寄存器进行设置来选择合适的波特率。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：

位频率 ＝ f_{CLK}/ (I2SCLH + I2SCLL)

此处为 f_{CLK} 为 pclk 频率。

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。但寄存器的值必须确保 I^2C 数据通信速率在 0 到 400KHz 之间。这样对 I2SCLL 和 I2SCLH 的值就有一些限制。每个寄存器的值都必须大于等于 4。

表 88 I^2C SCL 高电平占空比寄存器（I2SCLH - 0xE001C010）

I2SCLH	功能	描述	复位值
15:0	计数值	SCL 高电平周期选择计数	0x0004

表 89 I^2C SCL 低电平占空比寄存器（I2SCLL - 0xE001C014）

I2SCLL	功能	描述	复位值
15:0	计数值	SCL 低电平周期选择计数	0x0004

表 90 VPB 时钟分频＝1 时的 I^2C 时钟速率选择

I2SCLL+ I2SCLH	位数据速率（KHz）@ f_{CCLK}(MHz) & VPB 时钟分频＝1				
	16	20	40	60	80
8	-	-	-	-	-
10	-	-	-	-	-
25	-	-	-	-	-
50	320.0	400.0	-	-	-
75	213.333	266.67	-	-	-
100	160.0	200.0	400.0	-	-
160	100.0	125.0	250.0	375.0	-
200	80.0	100.0	200.0	300.0	400.0
320	50.0	62.5	125.0	187.5	250.0
400	40.0	50.0	100.0	150.0	200.0
510	31.373	39.216	78.431	117.647	156.863
800	20.0	25.0	50.	75.0	100.0
1280	12.5	15.625	31.25	46.875	62.5

表 91 VPB 时钟分频＝2 时的 I^2C 时钟速率选择

I2SCLL+ I2SCLH	位数据速率（KHz）@ f_{CCLK}(MHz) & VPB 时钟分频＝2				
	16	20	40	60	80
8	-	-	-	-	-
10	-	-	-	-	-
25	320.0	400.0	-	-	-
50	160.0	200.0	400.0	-	-
75	106.667	133.333	266.667	400.0	-
100	80.0	100.0	200.0	300.0	400.0
160	50.0	62.5	125.0	187.5	250.0
200	40.0	50.0	100.0	150.0	200.0
320	25.0	31.25	62.5	93.75	125.0
400	20.0	25.0	50.0	75.0	100.0
510	15.686	19.608	39.216	58.824	78.431
800	10.0	12.5	25.0	37.5	50.0
1280	6.25	7.813	15.625	23.438	31.25

表 92 VPB 时钟分频＝4 时的 I^2C 时钟速率选择

I2SCLL+ I2SCLH	位数据速率（KHz）@ f_{CCLK}(MHz) & VPB 时钟分频＝4				
	16	20	40	60	80
8	500.0	-	-	-	-
10	400.0	-	-	-	-
25	160.0	200.0	400.0	-	-
50	80.0	100.0	200.0	300.0	400.0
75	53.333	66.667	133.333	200.0	266.667

续上表

I2SCLL+ I2SCLH	位数据速率（KHz）@ f_{CCLK}(MHz) & VPB 时钟分频＝4				
	16	20	40	60	80
100	40.0	50.0	100.0	150	200.0
160	25.0	31.25	62.5	93.75	125.0
200	20.0	25.0	50.0	75.0	100.0
320	12.5	15.625	31.25	46.875	62.5
400	10.0	12.5	25.0	37.5	50.0
510	7.843	9.804	19.608	29.412	39.216
800	5.0	6.25	12.5	18.75	25.0
1280	3.125	3.906	7.813	11.719	15.625

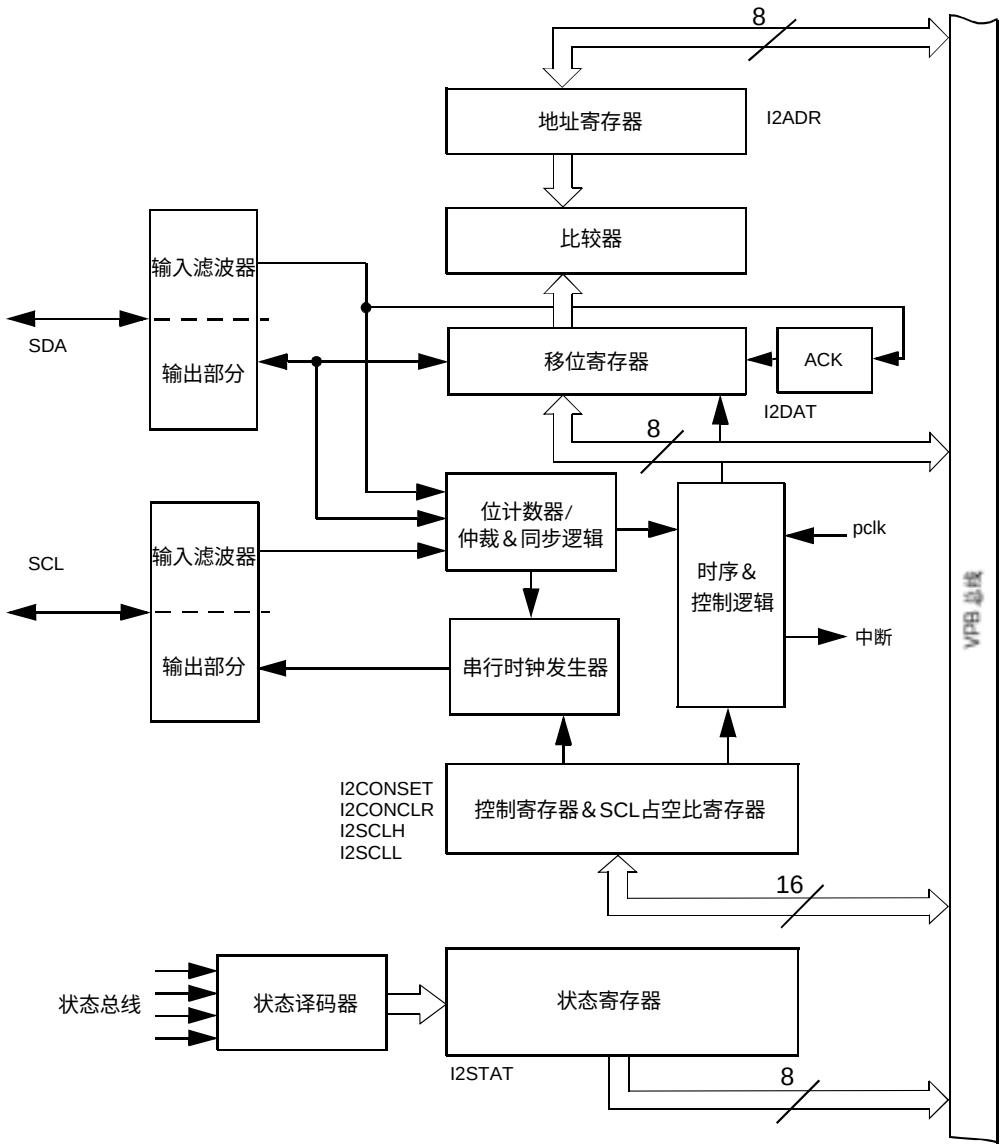


图 23 I^2C 结构

11. SPI 接口

特性

- 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

描述

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。在一定数据传输过程中，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一个字节数据，而从机也总是向主机发送一个字节数据。

SPI 数据传输

图 24 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 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）。

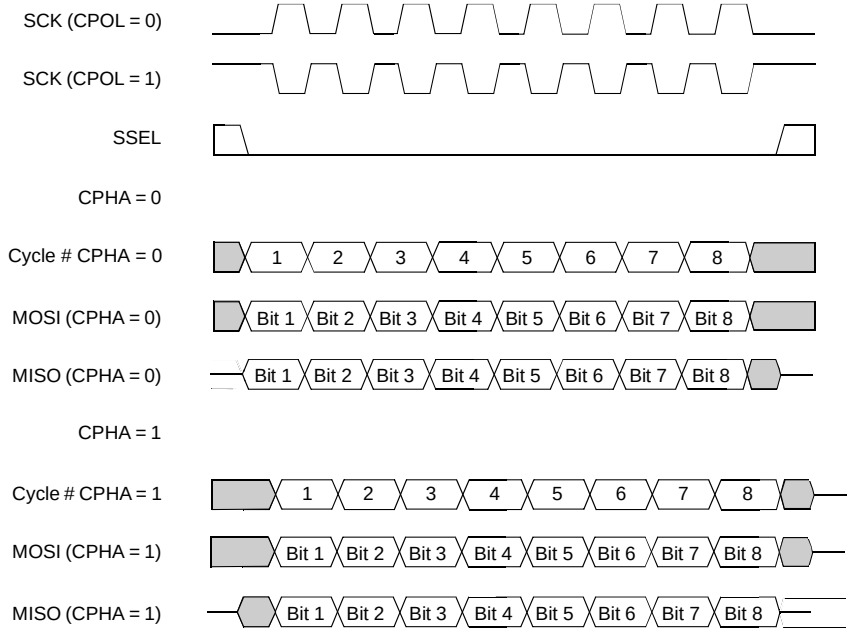


图 24 SPI 数据传输格式（CPHA=0 和 CPHA=1）

数据和时钟的相位关系在表 93 中描述。改变汇集了下面情况下 CPOL 和 CPHA 的每一种设定：

- 当驱动第一个数据位时
- 当驱动所有其它数据位时
- 当采样数据时

表 93 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（低电平）时开始，并在 SSEL 变为高电平时结束。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。

SPI 外设描述

有 4 个寄存器控制 SPI 外设。将在 *寄存器描述* 一节中详细讲述。

SPI 控制寄存器包含一些可编程位来控制 SPI 时钟的功能。该寄存器必须在数据传输之前进行设定。

SPI 状态寄存器包含只读位，用于监视 SPI 外设的状态，包括一般性功能和异常状况。该寄存器的主要用途时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。

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内部移位寄存器来实现。在发送时向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。因此在没有执行数据发送时不要向该寄存器写入数据。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，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。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。

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。

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周期之后置位。
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。

异常状况

读溢出－当 SPI 模块内部读缓冲区包含没有读出的数据，而新的传输已经完成，那么这时候就会发生读溢出。SPIF 位置位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，而读溢出 (ROVR)位将置位。

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。

模式错误－SSEL 信号在 SPI 模块为主机时必须保持高电平（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，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。

主机中止－如果 SSEL 信号在传输结束之前变为高电平，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，从机中止（ABRT）位置位。

管脚描述

表 94 SPI 管脚描述

管脚名称	类型	描述
SCK	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，即三态。
SSEL	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。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。
MISO	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。
MOSI	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。

寄存器描述

SPI 包含 7 个寄存器，见表 95。所有寄存器都可以字节、半字和字的形式访问。

表 95 SPI 寄存器映射

地址	名称	描述	访问	复位值
0xE0020000	SPCR	SPI 控制寄存器。该寄存器控制 SPI 的操作。	R/W	0
0xE0020004	SPSR	SPI 状态寄存器。该寄存器显示 SPI 的状态。	RO	0
0xE0020008	SPDR	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。SPI 接收的数据可从该寄存器读出。	R/W	0
0xE002000C	SPCCR	SPI 时钟计数寄存器。该寄存器控制主机 SCK 的频率。	R/W	0
0xE002001C	SPINT	SPI 中断寄存器。该寄存器包含 SPI 接口的中断标志。	R/W	0

SPI 控制寄存器（SPCR - 0xE0020000）

SPCR 寄存器根据每个配置位的设定来控制 SPI 的操作。

表 96 SPI 控制寄存器（SPCR - 0xE0020000）

SPCR	功能	描述	复位值
2:0	保留	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	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。当该位为 1 时，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。 当该位为 0 时，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。	0
4	CPOL	时钟极性控制。当该位为 1 时，SCK 为低有效。为 0 时，SCK 为高有效。	0
5	MSTR	主模式选择。为 1 时，SPI 处于主模式。为 0 时，SPI 处于从模式。	0
6	LSBF	移位方向控制。为 1 时，SPI 数据传输 LSB (bit0)在先。为 0 时，SPI 数据传输 MSB (bit7)在先。	0
7	SPIE	SPI 中断使能。为 1 时，每次 SPIF 或 MODF 置位时都会产生硬件中断。为 0 时，SPI 中断被禁止。	0

SPI 状态寄存器（SPSR - 0xE0020004）

SPSR 寄存器根据配置位的设定来控制 SPI 的操作。

表 97 SPI 状态寄存器（SPSR - 0xE0020004）

SPSR	功能	描述	复位值
2:0	保留	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。当读取该寄存器时，该位清零。	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。当读取该寄存器时，该位清零。	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

SPI 数据寄存器（SPDR - 0xE0020008）

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 98 SPI 数据寄存器（SPDR - 0xE0020008）

SPDR	功能	描述	复位值
7:0	数据	SPI 双向数据	0

SPI 时钟计数寄存器（SPCCR - 0xE002000C）

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 周期的数据。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动作。

表 99 SPI 时钟计数寄存器（SPCCR - 0xE002000C）

SPCCR	功能	描述	复位值
7:0	计数值	SPI 时钟计数值设定	0

SPI 数量可以这样进行计算：PCLK 速率/SPCCR 值。pclk 速率为 CCLK/VPB 除数。

SPI 中断寄存器（SPINT - 0xE002001C）

该寄存器包含 SPI 接口的中断标志。

表 100 SPI 中断寄存器（SPINT - 0xE002001C）

SPINT	功能	描述	复位值
0	SPI 中断	SPI 中断标志。由 SPI 接口置位以产生中断。向该位写入 1 清零。	0
7:1	保留	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

结构

SPI 方框图见图 25。

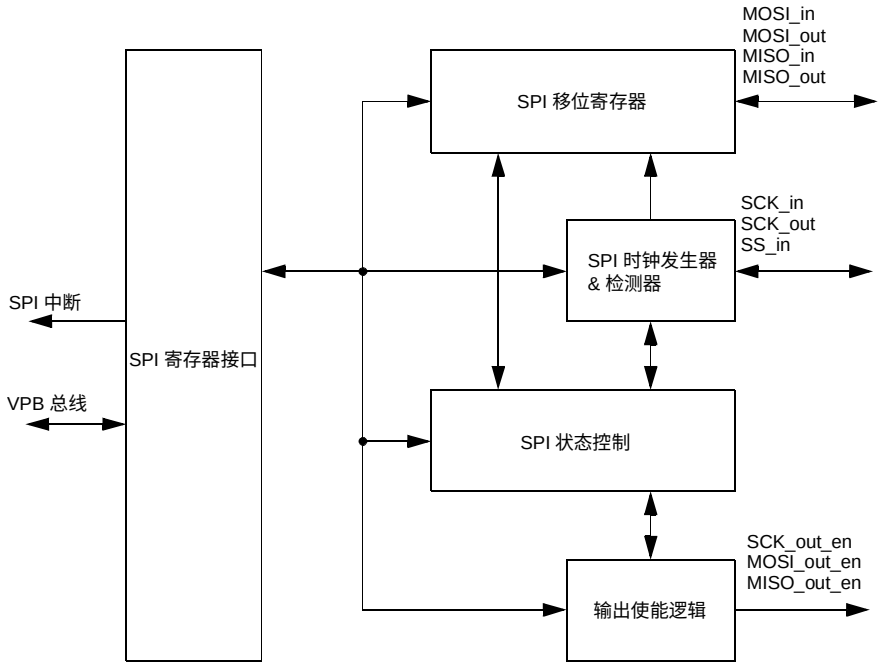


图 25 SPI 方框图

12. 定时器 0 和定时器 1

定时器 0 和定时器 1 除了外设基地址以外，其它都相同。

特性

- 带可编程 32 位预分频器的 32 位定时器/计数器
- 分别具有多达 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

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

描述

定时器对外设时钟（pclk）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 LPC2106/2105/2104 的管脚数量有限，定时器 0 只有 3 个捕获输入和 3 个匹配输出连接到器件的管脚。

管脚描述

表 101 所示为每个定时器相关管脚的简要描述。

表 101 管脚概况

管脚名称	管脚方向	管脚描述
CAP0.3..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。
MAT0.0 MAT1.0	输入	**外部匹配输出 0/1** 当匹配寄存器 0(MR0)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)控制该输出的功能。
MAT0.1 MAT1.1	输入	**外部匹配输出 1**－ 见 MAT0/MAT1 的描述
MAT0.2 MAT1.2	输入	**外部匹配输出 2**－ 见 MAT0/MAT1 的描述
MAP1.3	输入	**外部匹配输出 3**－ 见 MAT1 的描述

寄存器描述

每个定时器所包含的寄存器如表 102 所示。

表 102 定时器 0 和定时器 1 寄存器映射

名称	定时器 0 地址&名称	定时器 1 地址&名称	描述	访问	复位值
IR	0xE0004000 T0IR	0xE0008000 T1IR	中断寄存器 可以写 IR 来清除中断。可读取 IR 来识别哪个中断源被挂起。	R/W	0
TCR	0xE0004004 T0TCR	0xE0008004 T1TCR	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
TC	0xE0004008 T0TC	0xE0008008 T1TC	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
PR	0xE000400C T0PR	0xE000800C T1PR	预分频寄存器 32 位 TC 每经过 PR+1 个 pclk 周期加 1。	R/W	0
PC	0xE0004010 T0PC	0xE0008010 T1PC	预分频计数器 每当 32 位 PC 的值增加到等于 PR 中保存的值时，TC 加 1。	R/W	0
MCR	0xE0004014 T0MCR	0xE0008014 T1MCR	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
MR0	0xE0004018 T0MR0	0xE0008018 T1MR0	匹配寄存器 0 MR0 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。	R/W	0
MR1	0xE000401C T0MR1	0xE000801C T1MR1	匹配寄存器 1 同上	R/W	0
MR2	0xE0004020 T0MR2	0xE0008020 T1MR2	匹配寄存器 2 同上	R/W	0
MR3	0xE0004024 T0MR3	0xE0008024 T1MR3	匹配寄存器 3 同上	R/W	0
CCR	0xE0004028 T0CCR	0xE0008028 T1CCR	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
CR0	0xE000402C T0CR0	0xE000802C T1CR0	捕获寄存器 0 当产生捕获事件时，CR0 装载 TC 的值。	RO	0
CR1	0xE0004030 T0CR1	0xE0008030 T1CR1	捕获寄存器 1 同上	RO	0
CR2	0xE0004034 T0CR2	0xE0008034 T1CR2	捕获寄存器 2 同上	RO	0
CR3	0xE0004038 T0CR3	0xE0008038 T1CR3	捕获寄存器 3 同上，未用于定时器 0。	RO	0
EMR	0xE000403C T0EMR	0xE000803C T1EMR	外部匹配寄存器 EMR 控制外部匹配管脚 MATn。	R/W	0

中断寄存器（IR: 定时器 0 - T0IR：0xE00040000；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（定时器 0 为 3 个）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 103 中断寄存器（IR: 定时器 0 - T0IR：0xE00040000；定时器 1 - T1IR：0xE0008000）

IR	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	0

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 104 定时器控制寄存器（TCR - 定时器 0 – T0TCR：0xE0004004；定时器 1 – T1TCR：0xE0008004）

TCR	功能	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

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 0x00000000。该事件不会产生中断。如果需要，可用匹配寄存器检测溢出。

预分频寄存器（PR: 定时器 0 – T0PR：0xE000400C；定时器 1 – T1PR：0xE000800C）

32 位预分频寄存器直到预分频寄存器的最大值。

预分频计数器寄存器（PC: 定时器 0 – T0PC：0xE0004010；定时器 1 – T1PC：0xE0008010）

预分频计数器使用某个常量来控制 pclk 的分频。这样可实现控制定时器分辨率和定时器溢出时间之间的关系。预分频计数器每个 pclk 周期加 1。当其到达 PR 中保存的值时，定时器计数器 TC 加 1，PC 在下个 pclk 周期复位。这样，当 PR=0 时，TC 每 1 个 pclk 周期加 1，当 PR=1 时，TC 每 2 个 pclk 周期加 1。

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 寄存器控制。

匹配控制寄存器（MCR: 定时器 0 – T0MCR：0xE0004014；定时器 1 – T1MCR：0xE00080014）

匹配控制寄存器用于控制在发生匹配时所执行的操作。每个位的功能见表 105。

表 105 匹配控制寄存器（MCR: 定时器 0 – T0MCR：0xE0004014；定时器 1 – T1MCR：0xE00080014）

MCR	功能	描述	复位值
0	中断(MR0)	为 1 时，MR0 与 TC 值的匹配将产生中断。为 0 时，中断被禁止。	0
1	复位(MR0)	为 1 时，MR0 与 TC 值的匹配将使 TC 复位。为 0 时，该特性被禁止。	0
2	停止(MR0)	为 1 时，MR0 与 TC 值的匹配将使 TC 和 PC 停止。为 0 时，中断被禁止。	0
3	中断(MR1)	为 1 时，MR1 与 TC 值的匹配将产生中断。为 0 时，中断被禁止。	0
4	复位(MR1)	为 1 时，MR1 与 TC 值的匹配将使 TC 复位。为 0 时，该特性被禁止。	0
5	停止(MR1)	为 1 时，MR1 与 TC 值的匹配将使 TC 和 PC 停止。为 0 时，中断被禁止。	0
6	中断(MR2)	为 1 时，MR2 与 TC 值的匹配将产生中断。为 0 时，中断被禁止。	0
7	复位(MR2)	为 1 时，MR2 与 TC 值的匹配将使 TC 复位。为 0 时，该特性被禁止。	0
8	停止(MR2)	为 1 时，MR2 与 TC 值的匹配将使 TC 和 PC 停止。为 0 时，中断被禁止。	0
9	中断(MR3)	为 1 时，MR3 与 TC 值的匹配将产生中断。为 0 时，中断被禁止。	0
10	复位(MR3)	为 1 时，MR3 与 TC 值的匹配将使 TC 复位。为 0 时，该特性被禁止。	0
11	停止(MR3)	为 1 时，MR3 与 TC 值的匹配将使 TC 和 PC 停止。为 0 时，中断被禁止。	0

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。

捕获控制寄存器（CCR: 定时器 0 – T0CCR：0xE0004028；定时器 1 – T1CCR：0xE00080028）

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装入 4 个捕获寄存器中的哪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。

表 106 捕获控制寄存器（CCR: 定时器 0 – T0CCR：0xE0004028；定时器 1 – T1CCR：0xE00080028）

CCR	功能	描述	复位值
0	CAP[0] 上升沿捕获	为 1 时，CAP[0]上 0 到 1 的跳变将导致 TC 的内容装入 CR0。为 0 时，该特性被禁止。	0
1	CAP[0] 下降沿捕获	为 1 时，CAP[0]上 1 到 0 的跳变将导致 TC 的内容装入 CR0。为 0 时，该特性被禁止。	0
2	CAP[0] 事件中断	为 1 时，CAP[0]的捕获事件所导致的 CR0 装载将产生一个中断。为 0 时，该特性被禁止。	0
3	CAP[1] 上升沿捕获	为 1 时，CAP[1]上 0 到 1 的跳变将导致 TC 的内容装入 CR0。为 0 时，该特性被禁止。	0
4	CAP[1] 下降沿捕获	为 1 时，CAP[1]上 1 到 0 的跳变将导致 TC 的内容装入 CR0。为 0 时，该特性被禁止。	0
5	CAP[1] 事件中断	为 1 时，CAP[1]的捕获事件所导致的 CR0 装载将产生一个中断。为 0 时，该特性被禁止。	0
6	CAP[2] 上升沿捕获	为 1 时，CAP[2]上 0 到 1 的跳变将导致 TC 的内容装入 CR0。为 0 时，该特性被禁止。	0
7	CAP[2] 下降沿捕获	为 1 时，CAP[2]上 1 到 0 的跳变将导致 TC 的内容装入 CR0。为 0 时，该特性被禁止。	0
8	CAP[2] 事件中断	为 1 时，CAP[2]的捕获事件所导致的 CR0 装载将产生一个中断。为 0 时，该特性被禁止。	0

续上表

CCR	功能	描述	复位值
9	CAP[3]上升沿捕获	（仅用于定时器 1）。为 1 时，CAP[3]上 0 到 1 的跳变将导致 TC 的内容装入 CR0。为 0 时，该特性被禁止。	0
10	CAP[3]下降沿捕获	（仅用于定时器 1）。为 1 时，CAP[3]上 1 到 0 的跳变将导致 TC 的内容装入 CR0。为 0 时，该特性被禁止。	0
11	CAP[3]事件中断	（仅用于定时器 1）。为 1 时，CAP[3]的捕获事件所导致的 CR0 装载将产生一个中断。为 0 时，该特性被禁止。	0

外部匹配寄存器（EMR: 定时器 0 – T0EMR：0xE000403C；定时器 1 – T1EMR：0xE0008003C）

外部匹配寄存器提供外部匹配管脚 M(0-3)的控制和状态。

表 107 外部匹配寄存器（EMR: 定时器 0 – T0EMR：0xE000403C；定时器 1 – T1EMR：0xE0008003C）

EMR	功能	描述	复位值
0	外部匹配 0	不管 MAT0/1 是否连接到管脚，该位都会反映 MAT0/1 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	外部匹配 1	不管 MAT0/1 是否连接到管脚，该位都会反映 MAT0/1 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	外部匹配 2	不管 MAT0/1 是否连接到管脚，该位都会反映 MAT0/1 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	外部匹配 3	不管 MAT0/1 是否连接到管脚，该位都会反映 MAT0/1 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。 注：定时器 0 的该输出不能连接到器件管脚。	0
5:4	外部匹配控制 0	决定外部匹配 0 的功能。表 108 所示为这两个位的编码。	0
7:6	外部匹配控制 1	决定外部匹配 1 的功能。表 108 所示为这两个位的编码。	0
9:8	外部匹配控制 2	决定外部匹配 2 的功能。表 108 所示为这两个位的编码。	0
11:10	外部匹配控制 3	决定外部匹配 3 的功能。表 108 所示为这两个位的编码。	0

表 108 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	将对应的外部匹配输出设置为 0（如果连接到管脚，则输出低电平）
10	将对应的外部匹配输出设置为 1（如果连接到管脚，则输出高电平）
11	使对应的外部匹配输出翻转

定时器举例操作

图 26 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 27 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。

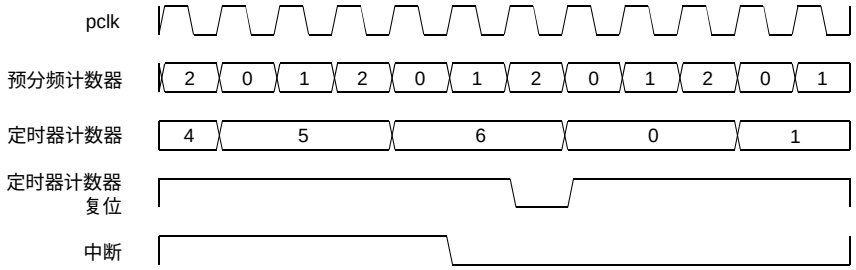


图 26　定时器周期设置为 PR=2, MRx=6, 匹配时使能中断和复位

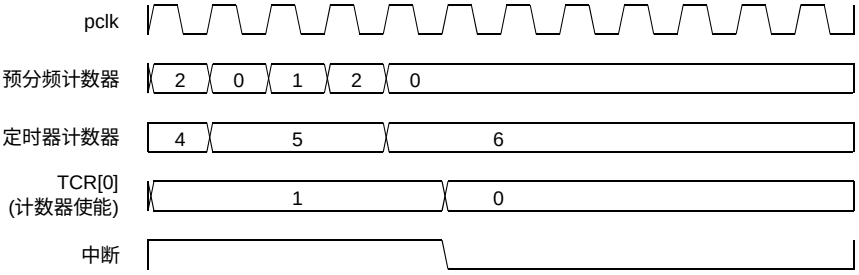
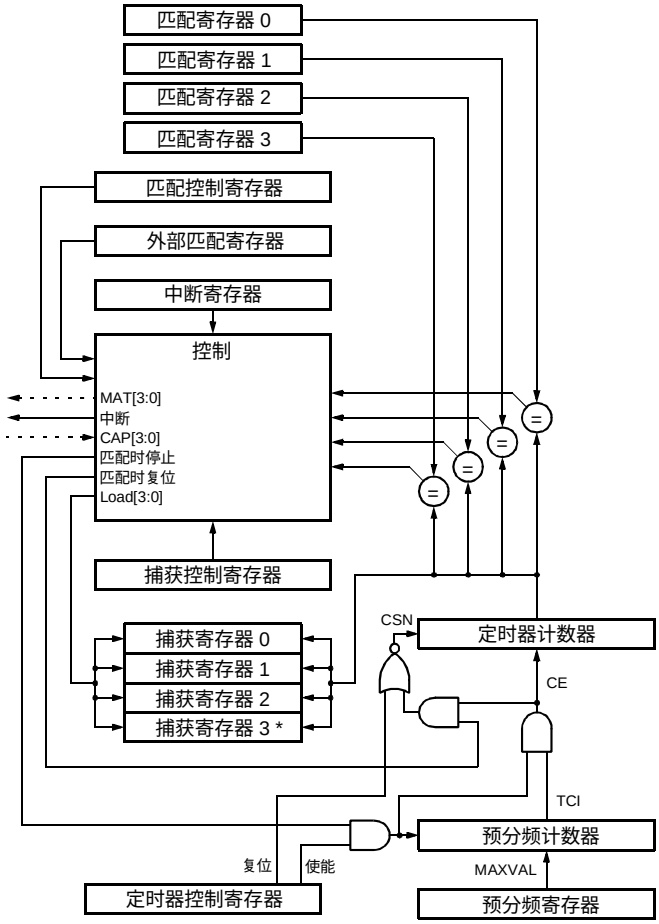


图 27　定时器周期设置为 PR=2, MRx=6, 匹配时使能中断和停止

结构

定时器 0 和定时器 1 的方框图，见图 28。



* 注：捕获寄存器 3 不能用于定时器 0

图 28 定时器方框图

13. 脉宽调制器（PWM）

LPC2106/2105/2104 的脉宽调制器建立在前一章的标准定时器 0/1 之上。应用可在 PWM 和匹配功能当中进行选择。

特性

- 7 个匹配寄存器，可实现 6 个单边沿控制或 3 个双边沿控制 PWM 输出，或这两种类型的混合输出：
  －连续操作，可选择在匹配时产生中断
  －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
- 支持单边沿控制和/或双边沿控制的 PWM 输出。单边沿控制 PWM 输出在每个周期开始时总是为高电平，除非输出保持恒定低电平。双边沿控制 PWM 输出可在一个周期内的任何位置产生边沿。这样可同时产生正和负脉冲。
- 脉冲周期和宽度可以是任何的定时器计数值。这样可实现灵活的分辨率和重复速率的设定。所有 PWM 输出都以相同的重复率发生。
- 双边沿控制的 PWM 输出可编程为正脉冲或负脉冲

- 匹配寄存器更新与脉冲输出同步，防止产生错误的脉冲。软件必须在新的匹配值生效之前将它们释放。
- 如果不使能 PWM 模式，可作为一个标准定时器
- 带可编程 32 位预分频器的 32 位定时器/计数器

描述

PWM 基于标准的定时器模块并具有其所有特性。不过 LPC2106/2105/2104 只将其 PWM 功能输出到管脚。定时器对外设时钟(pclk)进行计数，可选择产生中断或基于 7 个匹配寄存器，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，并可选择在事件发生时产生中断。PWM 功能是一个附加特性，建立在匹配寄存器事件基础之上。

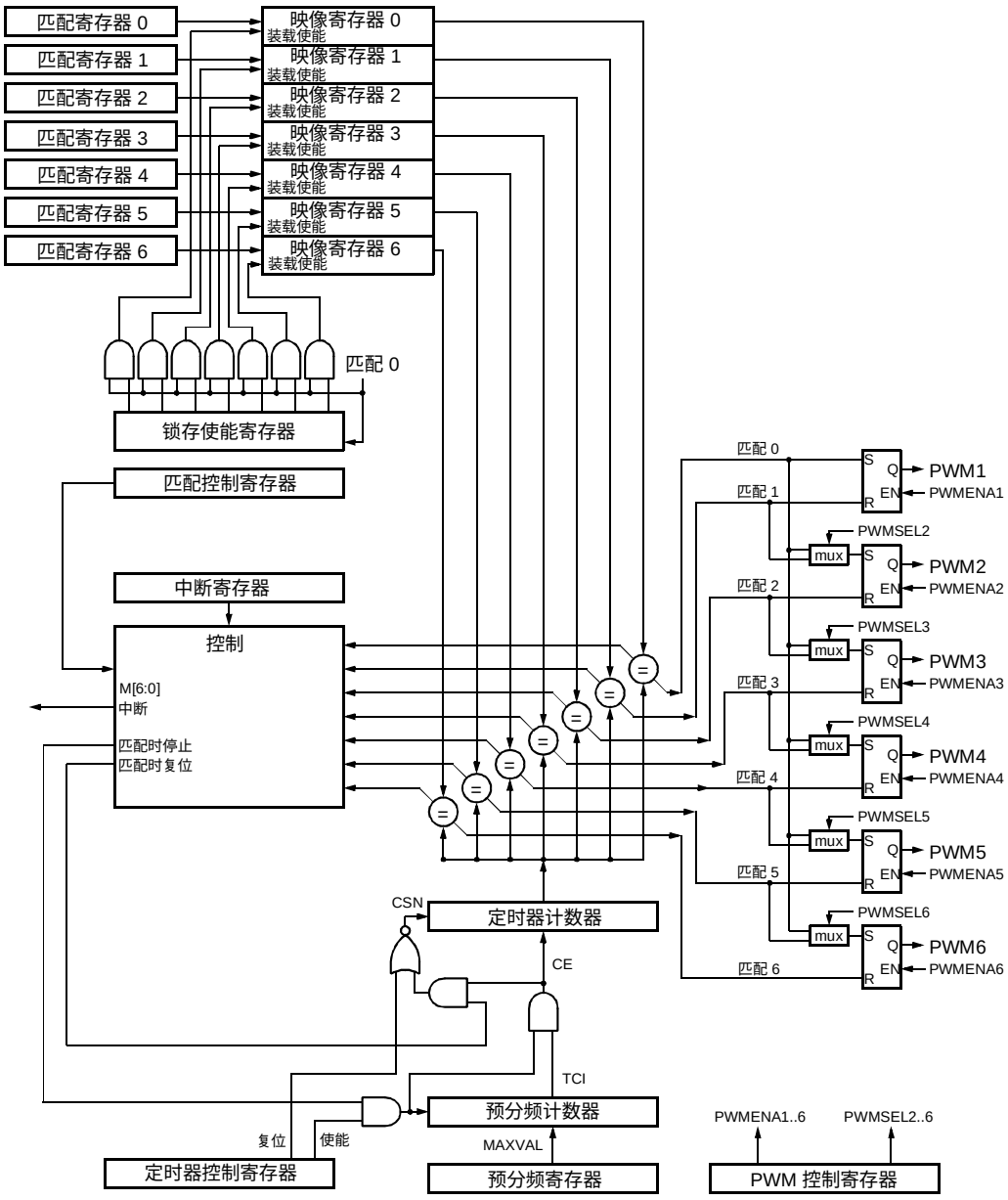
独立控制上升和下降沿位置的能力使 PWM 可以应用于更多的领域。例如，多相位电机控制通常需要 3 个非重叠的 PWM 输出，而这 3 个输出的脉宽和位置需要独立进行控制。

两个匹配寄存器可用于提供单边沿控制的 PWM 输出。匹配寄存器 MR0 通过匹配时重新设置计数值来控制 PWM 周期率。其它的匹配寄存器控制 PWM 边沿的位置。每个额外的单边沿控制 PWM 输出只需要一个匹配寄存器，因为所有 PWM 输出的重复率速率是相同的。多个单边沿控制 PWM 输出在每个 PWM 周期的开始，当 MR0 发生匹配时，都有一个上升沿。

3 个匹配寄存器可用于提供一个双边沿控制 PWM 输出。也就是说，MR0 匹配寄存器控制 PWM 周期速率，其它匹配寄存器控制两个 PWM 边沿位置。每个额外的双边沿控制 PWM 输出只需要两个匹配寄存器，因为所有 PWM 输出的重复率速率是相同的。

使用双边沿控制 PWM 输出时，指定的匹配寄存器控制输出的上升和下降沿。这样就产生了正脉冲（当上升沿先于下降沿时）和负脉冲（当下降沿先于上升沿时）。

图 29 所示为 PWM 的方框图。在标准定时器模块上增加的部分位于图的右边和顶端。



注：该图用来解释 PWM 的功能，不是一个具体的设计方案。

图 29 PWM 方框图

图 30 所示为一个用来说明 PWM 值与波形输出之间关系的例子。PWM 输出逻辑允许通过 PWMSELn 位选择单边沿或者双边沿控制的 PWM 输出。表 109 所示为不同 PWM 输出的匹配寄存器选项。支持 N-1 个单边沿 PWM 输出或(N-1)/2 个双边沿 PWM 输出，其中 N 为匹配寄存器的个数。如果需要也可使用混合的 PWM 类型。

下面所示的波形是单个 PWM 周期，它演示了在下列条件下的 PWM 输出：

- 定时器配置为 PWM 模式
- 匹配寄存器 0 配置为在发生匹配事件时复位定时器/计数器
- 控制位 PWMSEL2 和 PWMSEL4 置位
- 匹配寄存器值如下：

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

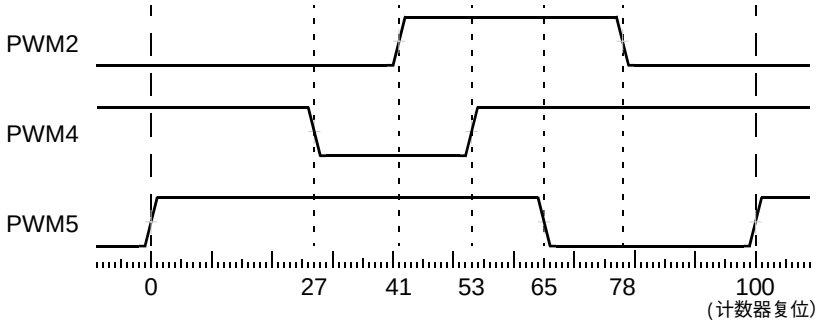


图 30 PWM 波形举例

表 109 PWM 触发器的置位和复位输入

PWM 通道	单边沿 PWM(PWMSELn=0)		双边沿 PWM(PWMSELn=1)	
	置位	复位	置位	复位
1	匹配 0	匹配 1	匹配 0[1]	匹配 1[1]
2	匹配 0	匹配 2	匹配 1	匹配 2
3	匹配 0	匹配 3	匹配 2[2]	匹配 3[2]
4	匹配 0	匹配 4	匹配 3	匹配 4
5	匹配 0	匹配 5	匹配 4[2]	匹配 5[2]
6	匹配 0	匹配 6	匹配 5	匹配 6

注：

1. 这种情况下与单边沿模式相同，因为匹配 0 是相邻的匹配寄存器。基本上 PWM1 不能用作双边沿输出。
2. 通常不建议使用 PWM 通道 3 和通道 5 作为双边沿 PWM 输出，因为这样会减少可用的双边沿 PWM 的个数。使用 PWM2, PWM4 和 PWM6 可得到最多个数的双边沿 PWM 输出。

单边沿控制的 PWM 输出规则

1. 所有单边沿控制的 PWM 输出在 PWM 周期开始时都为高电平，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都会变为低电平。如果没有发生匹配（即匹配值大于 PWM 速率），PWM 将一直保持高电平。

双边沿控制的 PWM 输出规则

当一个新的周期将要开始时，使用以下 5 个规则来决定下一个 PMW 输出的值：

1. 在一个 PWM 周期结束时（与下一个 PWM 周期的开始重合的时间点），使用下一个 PMW 周期的匹配值，例外见规则 3。
2. 等于 0 或当前 PWM 速率（与匹配通道 0 的值相同）的匹配值等效。例外见规则 3。例如，在 PWM 周期开始时的下降沿请求与 PWM 周期结束时的下降沿请求等效。
3. 当匹配值正在改变时，如果有其中一个“旧”匹配值等于 PWM 速率，并且新的匹配值不等于 0 或 PWM 速率，旧的匹配值不等于 0，那么旧的匹配值将再次被使用。
4. 如果同时请求 PWM 输出置位和清零，清零优先。当置位和清零匹配值相同时，或者置位或清零

值等于 0 并且其它值等于 PWM 速率时，可能发生这种状况。

5. 如果匹配值超出范围（大于 PMW 速率值），将不会发生匹配事件，匹配通道对输出不起作用。也就是说 PWM 输出将一直保持一种状态，可以为低电平、高电平或是“无变化”输出。

管脚描述

表 110 汇集了所有与 PWM 相关的管脚。

表 110 PWM 管脚汇总

管脚名称	管脚方向	管脚描述
PWM1	输出	PWM 通道 1 输出
PWM2	输出	PWM 通道 2 输出
PWM3	输出	PWM 通道 3 输出
PWM4	输出	PWM 通道 4 输出
PWM5	输出	PWM 通道 5 输出
PWM6	输出	PWM 通道 6 输出

寄存器描述

PWM 功能增加了新的寄存器和寄存器位，见表 111。

表 111 PWM 寄存器映射

地址	名称	描述	访问	复位值
0xE0014000	PWMIR	PWM 中断寄存器 可以写 IR 来清除中断。可读取 IR 来识别哪个中断源被挂起。	R/W	0
0xE0014004	PWMTCR	PWM 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
0xE0014008	PWMTC	PWM 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
0xE001400C	PWMPR	PWM 预分频寄存器 TC 每经过 PR+1 个 pclk 周期加 1。	R/W	0
0xE0014010	PWMPC	PWM 预分频计数器 每当 32 位 PC 的值增加到等于 PR 中保存的值时，TC 加 1。	R/W	0
0xE0014014	PWMMCR	PWM 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
0xE0014018	PWMMR0	PWM 匹配寄存器 0 MR0 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR0 和 TC 的匹配将置位所有单边沿模式的 PWM 输出，并置位双边沿模式下的 PWM1 输出。	R/W	0
0xE001401C	PWMMR1	PWM 匹配寄存器 1 MR1 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR1 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM1，并置位双边沿模式下的 PWM2 输出。	R/W	0
0xE0014020	PWMMR2	PWM 匹配寄存器 2 MR2 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR2 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM2，并置位双边沿模式下的 PWM3 输出。	R/W	0
0xE0014024	PWMMR3	PWM 匹配寄存器 3 MR3 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR3 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM3，并置位双边沿模式下的 PWM4 输出。	R/W	0
0xE0014040	PWMMR4	PWM 匹配寄存器 4 MR4 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR4 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM4，并置位双边沿模式下的 PWM5 输出。	R/W	0

续上表

地址	名称	描述	访问	复位值
0xE0014044	PWMMR5	PWM 匹配寄存器 5 MR5 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR5 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM5，并置位双边沿模式下的 PWM6 输出。	R/W	0
0xE0014048	PWMMR6	PWM 匹配寄存器 6 MR6 可通过 MCR 设定为在匹配时复位 TC，停止 TC 和 PC 和/或产生中断。此外，MR6 和 TC 的匹配将清零单边沿模式或双边沿模式下的 PWM6，	R/W	0
0xE001404C	PWMPCR	PWM 控制寄存器 使能 PWM 输出并选择 PWM 通道类型为单边沿或双边沿控制。	R/W	0
0xE0014050	PWMLER	PWM 锁存使能寄存器 使能使用新的 PWM 匹配值。	R/W	0

中断寄存器（PWMIR - 0xE0014000）

中断寄存器包含 11 个位，见表 112。其中 7 个位用于匹配中断，4 个位保留将来之用。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 112 中断寄存器（PWMIR - 0xE00140000）

PWMIR	功能	描述	复位值
0	PWMMR0 中断	PWM 匹配通道 0 的中断标志	0
1	PWMMR1 中断	PWM 匹配通道 1 的中断标志	0
2	PWMMR2 中断	PWM 匹配通道 2 的中断标志	0
3	PWMMR3 中断	PWM 匹配通道 3 的中断标志	0
4	保留	应用程序不能向该位写入 1	0
5	保留	应用程序不能向该位写入 1	0
6	保留	应用程序不能向该位写入 1	0
7	保留	应用程序不能向该位写入 1	0
8	PWMMR4 中断	PWM 匹配通道 4 的中断标志	0
9	PWMMR5 中断	PWM 匹配通道 5 的中断标志	0
10	PWMMR6 中断	PWM 匹配通道 6 的中断标志	0

PWM 定时器控制寄存器（PWMTCR - 0xE0014004）

定时器控制寄存器 PWMTCR 用于控制 PWM 定时器计数器的操作。

表 113 定时器控制寄存器（PWMTCR - 0xE0014004）

PWMTCR	功能	描述	复位值
0	计数器使能	为 1 时，PWM 定时器计数器和 PWM 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，PWM 定时器计数器和 PWM 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
2	保留	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	PWM 使能	为 1 时，PWM 模式使能。PWM 模式将映像寄存器连接到匹配寄存器。只有在 PWMLER 中的相应位置位后发生的匹配 0 事件才会使程序写入匹配寄存器的值生效。需要注意的是，决定 PWM 速率的匹配寄存器（PWMMR0）必须在使能 PWM 之前设定。否则不会发生使映像寄存器内容生效的匹配事件。	0

定时器计数器（PWMTC - 0xE0014008）

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 PWMTC 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 0x00000000。该事件不会产生中断。如果需要，可用匹配寄存器检测溢出。

预分频寄存器（PWMPR - 0xE001400C）

32 位预分频寄存器直到预分频寄存器的最大值。

预分频计数器寄存器（PWMPC - 0xE0014010）

预分频计数器使用某个常量来控制 pclk 的分频。这样可实现控制定时器分辨率和定时器溢出时间之间的关系。预分频计数器每个 pclk 周期加 1。当其到达 PWMPR 中保存的值时，PWM 定时器计数器加 1，PWMPC 在下个 pclk 周期复位。这样，当 PWMPR=0 时，PWMTC 每 1 个 pclk 周期加 1，当 PWMPR=1 时，PWMTC 每 2 个 pclk 周期加 1。

匹配寄存器（PWMMR0 –PWMMR6）

PWM 匹配寄存器值连续与 PWM 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 PWMMCR 寄存器控制。

匹配控制寄存器（PWMMCR - 0xE0014014）

PWM 匹配控制寄存器用于控制在发生匹配时所执行的操作。每个位的功能见表 114。

表 114 匹配控制寄存器（PWMMCR - 0xE0014014）

PWMMCR	功能	描述	复位值
0	中断(PWMMR0)	为 1 时，PWMMR0 与 PWMTC 值的匹配将产生中断。为 0 时，该中断被禁止。	0
1	复位(PWMMR0)	为 1 时，PWMMR0 与 PWMTC 值的匹配将使 PWMTC 复位。为 0 时，该特性被禁止。	0
2	停止(PWMMR0)	为 1 时，PWMMR0 与 PWMTC 值的匹配将使 PWMTC 和 PWMPC 停止并使 PWMTCR[0]复位为 0。为 0 时，该特性被禁止。	0
3	中断(PWMMR1)	为 1 时，PWMMR1 与 PWMTC 值的匹配将产生中断。为 0 时，该中断被禁止。	0
4	复位(PWMMR1)	为 1 时，PWMMR1 与 PWMTC 值的匹配将使 PWMTC 复位。为 0 时，该特性被禁止。	0
5	停止(PWMMR1)	为 1 时，PWMMR1 与 PWMTC 值的匹配将使 PWMTC 和 PWMPC 停止并使 PWMTCR[0]复位为 0。为 0 时，该特性被禁止。	0
6	中断(PWMMR2)	为 1 时，PWMMR2 与 PWMTC 值的匹配将产生中断。为 0 时，该中断被禁止。	0
7	复位(PWMMR2)	为 1 时，PWMMR2 与 PWMTC 值的匹配将使 PWMTC 复位。为 0 时，该特性被禁止。	0
8	停止(PWMMR2)	为 1 时，PWMMR2 与 PWMTC 值的匹配将使 PWMTC 和 PWMPC 停止并使 PWMTCR[0]复位为 0。为 0 时，该特性被禁止。	0
9	中断(PWMMR3)	为 1 时，PWMMR3 与 PWMTC 值的匹配将产生中断。为 0 时，该中断被禁止。	0
10	复位(PWMMR3)	为 1 时，PWMMR3 与 PWMTC 值的匹配将使 PWMTC 复位。为 0 时，该特性被禁止。	0
11	停止(PWMMR3)	为 1 时，PWMMR3 与 PWMTC 值的匹配将使 PWMTC 和 PWMPC 停止并使 PWMTCR[0]复位为 0。为 0 时，该特性被禁止。	0
12	中断(PWMMR4)	为 1 时，PWMMR4 与 PWMTC 值的匹配将产生中断。为 0 时，该中断被禁止。	0
13	复位(PWMMR4)	为 1 时，PWMMR4 与 PWMTC 值的匹配将使 PWMTC 复位。为 0 时，该特性被禁止。	0
14	停止(PWMMR4)	为 1 时，PWMMR4 与 PWMTC 值的匹配将使 PWMTC 和 PWMPC 停止并使 PWMTCR[0]复位为 0。为 0 时，该特性被禁止。	0
15	中断(PWMMR5)	为 1 时，PWMMR5 与 PWMTC 值的匹配将产生中断。为 0 时，该中断被禁止。	0
16	复位(PWMMR5)	为 1 时，PWMMR5 与 PWMTC 值的匹配将使 PWMTC 复位。为 0 时，该特性被禁止。	0

续上表

PWMMCR	功能	描述	复位值
17	停止(PWMMR5)	为 1 时，PWMMR5 与 PWMTC 值的匹配将使 PWMTC 和 PWMPC 停止并使 PWMTCR[0]复位为 0。为 0 时，该特性被禁止。	0
18	中断(PWMMR6)	为 1 时，PWMMR6 与 PWMTC 值的匹配将产生中断。为 0 时，该中断被禁止。	0
19	复位(PWMMR6)	为 1 时，PWMMR6 与 PWMTC 值的匹配将使 PWMTC 复位。为 0 时，该特性被禁止。	0
20	停止(PWMMR6)	为 1 时，PWMMR6 与 PWMTC 值的匹配将使 PWMTC 和 PWMPC 停止并使 PWMTCR[0]复位为 0。为 0 时，该特性被禁止。	0

PWM 控制寄存器（PWMPCR - 0xE001404C）

PWM 控制寄存器用于使能并选择每个 PMW 通道的类型。每个位的功能详见表 115。

表 115 PWM 控制寄存器（PWMPCR - 0xE001404C）

PWMPCR	功能	描述	复位值
1:0	保留	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	PWMSEL2	为 0 时，PWM2 选择单边沿控制模式；为 1 时，选择双边沿控制模式。	0
3	PWMSEL3	为 0 时，PWM3 选择单边沿控制模式；为 1 时，选择双边沿控制模式。	0
4	PWMSEL4	为 0 时，PWM4 选择单边沿控制模式；为 1 时，选择双边沿控制模式。	0
5	PWMSEL5	为 0 时，PWM5 选择单边沿控制模式；为 1 时，选择双边沿控制模式。	0
6	PWMSEL6	为 0 时，PWM6 选择单边沿控制模式；为 1 时，选择双边沿控制模式。	0
8:7	保留	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
9	PWMENA1	为 1 时，使能 PWM1 输出；为 0 时，禁止 PWM1 输出。	0
10	PWMENA2	为 1 时，使能 PWM2 输出；为 0 时，禁止 PWM2 输出。	0
11	PWMENA3	为 1 时，使能 PWM3 输出；为 0 时，禁止 PWM3 输出。	0
12	PWMENA4	为 1 时，使能 PWM4 输出；为 0 时，禁止 PWM4 输出。	0
13	PWMENA5	为 1 时，使能 PWM5 输出；为 0 时，禁止 PWM5 输出。	0
14	PWMENA6	为 1 时，使能 PWM6 输出；为 0 时，禁止 PWM6 输出。	0
15	保留	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

PWM 锁存使能寄存器（PWMLER - 0xE00140050）

当 PWM 匹配寄存器用于产生 PWM 时，PWM 锁存使能寄存器用于控制 PWM 匹配寄存器的更新。当定时器处于 PWM 模式时如果软件对 PWM 匹配寄存器位置执行写操作，写入的值将保存在一个映像寄存器中。当 PWM 匹配 0 事件发生时（在 PWM 模式下，通常也会复位定时器），如果对应的锁存使能寄存器位已经置位，那么映像寄存器的内容将传送到实际的匹配寄存器中。此时，新的值将生效并决定下一个 PWM 周期。当发生新值传送时，PWMLER 中的所有位都自动清零。在 PWMLER 中相应位置位和 PWM 匹配 0 事件发生之前，任何写入 PWM 匹配寄存器的值都不会影响 PWM 操作。

例如，当 PWM2 配置为双边沿操作并处于运行中时，改变定时的典型事件顺序如下：

- 将新值写入 PWM 匹配 1 寄存器；
- 将新值写入 PWM 匹配 2 寄存器；
- 写 PWMLER，同时置位 bit1 和 bit2；
- 更改的值将在下一次定时器复位时（当 PWM 匹配 0 事件发生时）生效。

写两个 PWM 匹配寄存器的顺序并不重要，因为在写 PWMLER 之前，写入的新匹配值都无效。这样就确保了两个值同时生效。如果使用单个值，也可用同样的方法更改。

PWMLER 中所有位的功能如表 116 所示。

表 116 PWM 锁存使能寄存器（PWMLER - 0xE0014050）

PWMLER	功能	描述	复位值
0	使能 PWM 匹配 0 锁存	将该位置位允许最后写入 PWM 匹配 0 寄存器的值在下次定时器复位时生效。见 PWM 匹配控制寄存器 PWMMCR 的描述。	0
1	使能 PWM 匹配 1 锁存	将该位置位允许最后写入 PWM 匹配 1 寄存器的值在下次定时器复位时生效。见 PWM 匹配控制寄存器 PWMMCR 的描述。	0
2	使能 PWM 匹配 2 锁存	将该位置位允许最后写入 PWM 匹配 2 寄存器的值在下次定时器复位时生效。见 PWM 匹配控制寄存器 PWMMCR 的描述。	0
3	使能 PWM 匹配 3 锁存	将该位置位允许最后写入 PWM 匹配 3 寄存器的值在下次定时器复位时生效。见 PWM 匹配控制寄存器 PWMMCR 的描述。	0
4	使能 PWM 匹配 4 锁存	将该位置位允许最后写入 PWM 匹配 4 寄存器的值在下次定时器复位时生效。见 PWM 匹配控制寄存器 PWMMCR 的描述。	0
5	使能 PWM 匹配 5 锁存	将该位置位允许最后写入 PWM 匹配 5 寄存器的值在下次定时器复位时生效。见 PWM 匹配控制寄存器 PWMMCR 的描述。	0
6	使能 PWM 匹配 6 锁存	将该位置位允许最后写入 PWM 匹配 6 寄存器的值在下次定时器复位时生效。见 PWM 匹配控制寄存器 PWMMCR 的描述。	0
7	保留	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. 实时时钟

特性

- 测量保持日历或时钟的时间通路
- 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
- 可编程基准时钟分频器允许调节 RTC 以适应不同的晶振频率

描述

实时时钟(RTC)提供一套寄存器在系统上电和关闭操作时对时间进行测量。RTC 消耗的功率非常低，这使其适合于由电池供电的，CPU 不连续工作(空闲模式)的系统。

结构

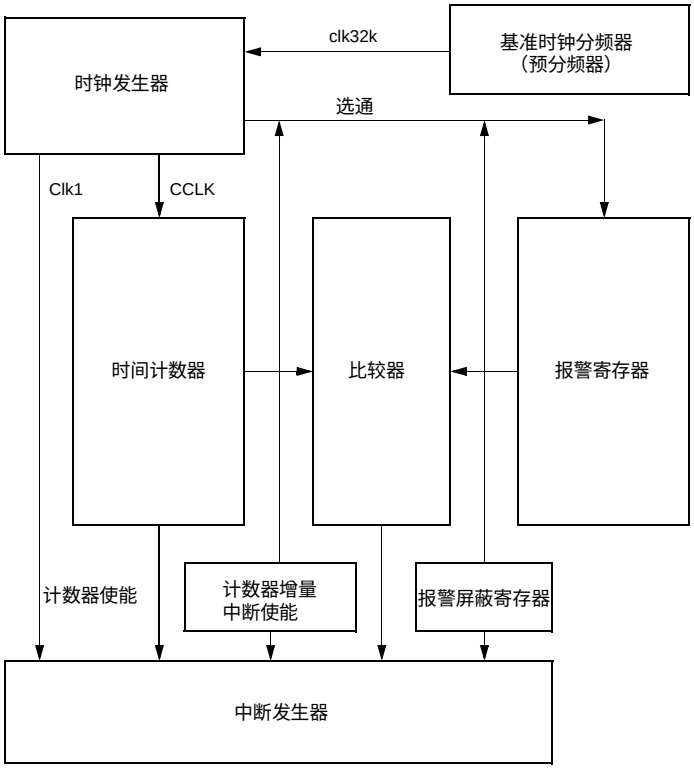


图 31 RTC 方框图

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。第二部分的 8 个地址为定时器计数器组，第三部分的 8 个地址为报警寄存器组。最后一部分为基准时钟分频器。

实时时钟模块所包含的寄存器见表 117。

表 117 实时时钟寄存器映射

地址	名称	规格	描述	访问	复位值
0xE0024000	ILR	2	中断位置寄存器	R/W	*
0xE0024004	CTC	15	时钟节拍计数器	RO	*
0xE0024008	CCR	4	时钟控制寄存器	R/W	*
0xE002400C	CIIR	8	计数器递增中断寄存器	R/W	*
0xE0024010	AMR	8	报警屏蔽寄存器	R/W	*
0xE0024014	CTIME0	(32)	完整时间寄存器 0	RO	*
0xE0024018	CTIME1	(32)	完整时间寄存器 1	RO	*
0xE002401C	CTIME2	(32)	完整时间寄存器 2	RO	*
0xE0024020	SEC	6	秒寄存器	R/W	*
0xE0024024	MIN	6	分寄存器	R/W	*
0xE0024028	HOUR	5	小时寄存器	R/W	*
0xE002402C	DOM	5	日期（月）寄存器	R/W	*
0xE0024030	DOW	3	星期寄存器	R/W	*
0xE0024034	DOY	9	日期（年）寄存器	R/W	*
0xE0024038	MONTH	4	月寄存器	R/W	*

续上表

地址	名称	规格	描述	访问	复位值
0xE002403C	YEAR	12	年寄存器	R/W	*
0xE0024060	ALSEC	6	秒报警值	R/W	*
0xE0024064	ALMIN	6	分报警值	R/W	*
0xE0024068	ALHOUR	5	小时报警值	R/W	*
0xE002406C	ALDOM	5	日期（月）报警值	R/W	*
0xE0024070	ALDOW	3	星期报警值	R/W	*
0xE0024074	ALDOY	9	日期（年）报警值	R/W	*
0xE0024078	ALMON	4	月报警值	R/W	*
0xE002407C	ALYEAR	12	年报警值	R/W	*
0xE0024080	PREINT	13	预分频值，整数部分	R/W	0
0xE0024084	PREFRAC	15	预分频值，小数部分	R/W	0

* RTC 当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。

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数据或时间。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。

混合寄存器组

表 118 所示为位置 0 到 7 的寄存器。

表 118 混合寄存器

地址	名称	规格	描述	访问
0xE0024000	ILR	2	中断位置寄存器	R/W
0xE0024004	CTC	15	时钟节拍计数器	RO
0xE0024008	CCR	4	时钟控制寄存器	R/W
0xE002400C	CIIR	8	计数器递增中断寄存器	R/W
0xE0024010	AMR	8	报警屏蔽寄存器	R/W
0xE0024014	CTIME0	32	完整时间寄存器 0	RO
0xE0024018	CTIME1	32	完整时间寄存器 1	RO
0xE002401C	CTIME2	32	完整时间寄存器 2	RO

中断位置（ILR - 0xE0024000）

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 119）。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 119 中断位置（ILR - 0xE0024000）

ILR	功能	描述
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。
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。

时钟节拍计数器（CTC - 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 120 时钟节拍计数器（CTC - 0xE0024004）

CTC	功能	描述
0	保留	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。
15:1	时钟节拍计数器	位于秒计数器之前，CTC 每秒计数 32768 个时钟。由于 RTC 预分频器的关系，这 32768 个时间增量的长度可能并不全部相同。详见基准时钟分频器（预分频器）。

时钟控制寄存器（CCR - 0xE0024008）

时钟控制寄存器是一个 4 位寄存器，它控制时钟分频电路的操作。见表 121。

表 121 时钟控制寄存器（CCR - 0xE0024008）

CCR	功能	描述
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。
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。
3:2	CTTEST	测试使能 在正常操作中，这些位应当全为 0。

计数器增量中断

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 ILR[0]位写入 1 之前，该中断一直保持有效。

表 122 计数器增量中断寄存器位（CIIR - 0xE002400C）

CIIR	功能	描述
0	IMSEC	为 1 时，秒值的增加产生一次中断。
1	IMMIN	为 1 时，分值的增加产生一次中断。
2	IMHOUR	为 1 时，小时值的增加产生一次中断。
3	IMDOM	为 1 时，日期（月）值的增加产生一次中断。
4	IMDOW	为 1 时，星期秒值的增加产生一次中断。
5	IMDOY	为 1 时，日期（年）值的增加产生一次中断。
6	IMMON	为 1 时，月值的增加产生一次中断。
7	IMYEAR	为 1 时，年值的增加产生一次中断。

报警屏蔽

报警屏蔽寄存器（AMR）允许用户屏蔽任意报警寄存器。表 123 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 123 报警屏蔽寄存器位（AMR - 0xE0024010）

AMR	功能	描述
0	AMRSEC	为 1 时，秒值不与报警寄存器比较。
1	AMRMIN	为 1 时，分值不与报警寄存器比较。
2	AMRHOUR	为 1 时，小时值不与报警寄存器比较。
3	AMRDOM	为 1 时，日期（月）值不与报警寄存器比较。
4	AMRDOW	为 1 时，星期秒值不与报警寄存器比较。
5	AMRDOY	为 1 时，日期（年）值不与报警寄存器比较。
6	AMRMON	为 1 时，月值不与报警寄存器比较。
7	AMRYEAR	为 1 时，年值不与报警寄存器比较。

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，见表 124，125 和 126。每个寄存器的最低位分别位于 bit0, 8, 16 和 24。

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。

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 124 完整时间寄存器 0（CTIME0 - 0xE0024014）

CTIME0	功能	描述
31:27	保留	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。
26:24	星期	星期值 该值的范围为 0~6。
23:21	保留	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。
20:16	小时	小时值 该值的范围为 0~23。
15:14	保留	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。
13:8	分	分值 该值的范围为 0~59。
7:6	保留	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。
5:0	秒	秒值 该值的范围为 0~59。

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 125 完整时间寄存器 1（CTIME0 - 0xE0024018）

CTIME1	功能	描述
31:28	保留	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。
27:16	年	年值 该值的范围为 0~4095。
15:14	保留	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。
13:8	月	小时值 该值的范围为 1~12。
7:6	保留	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。
5:0	日期（月）	日期（月）值 该值的范围为 1~28, 29, 30 或 31（取决于月份以及是否为闰年）。

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 126 完整时间寄存器 2（CTIME2 - 0xE002401C）

CTIME2	功能	描述
11:0	日期（年）	日期（年）值 该值的范围为 1~365（闰年为 366）。
31:12	保留	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。

时间计数器组

时间值包含 8 个寄存器，见表 127 和 128。表 128 所示的寄存器可执行读或写操作。

表 127 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 31）	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 128 时间计数器寄存器

地址	名称	规格	描述	访问
0xE0024020	SEC	6	秒值 该值的范围为 0~59。	R/W
0xE0024024	MIN	6	分值 该值的范围为 0~59。	R/W
0xE0024028	HOUR	5	小时值 该值的范围为 0~23。	R/W
0xE002402C	DOM	5	日期（月）值 该值的范围为 1~28,29,30 或 31（取决于月份以及是否为闰年）。	R/W
0xE0024030	DOW	3	星期值 该值的范围为 0~6。	R/W
0xE0024034	DOY	9	日期（年）值 该值的范围为 1~365（闰年为 366）。	R/W
0xE0024038	MONTH	4	月值 该值的范围为 1~12。	R/W
0xE002403C	YEAR	12	年值 该值的范围为 0~4095。	R/W

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。

报警寄存器组

报警寄存器见表 129。这些寄存器的值与时间计数器相比较。如果所有未屏蔽的报警寄存器都它们对应的时间计数器相匹配，那么将产生一次中断。向中断位置寄存器 ILR[1]写入 1 清除中断。

表 129 报警寄存器

地址	名称	规格	描述	访问
0xE0024060	ALSEC	6	秒报警值	R/W
0xE0024064	ALMIN	6	分报警值	R/W
0xE0024068	ALHOUR	5	小时报警值	R/W
0xE002406C	ALDOM	5	日期（月）报警值	R/W
0xE0024070	ALDOW	3	星期报警值	R/W
0xE0024074	ALDOY	9	日期（年）报警值	R/W
0xE0024078	ALMON	4	月报警值	R/W
0xE002407C	ALYEAR	12	年报警值	R/W

RTC 使用注意事项

由于 RTC 的时钟源为 VPB 时钟（pclk），时钟出现的任何中断都会导致时间值的偏移。

LPC2106/2105/2104 在断电时不能保持 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。芯片的断电将使 RTC 寄存器的内容完全丢失。进入掉电模式会使时间的更新出现误差。在系统操作过程中（重新配置 PLL、VPB 定时器或 RTC 预分频器）改变 RTC 的时间基准会使累加时间出现错误。

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 32768。

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 LPC2106/2105/2104 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 32768 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（32768×8192）。
2. 余数的最大值为 32767，需要 15 位来保存。

表 130 基准时钟分频寄存器

地址	名称	规格	描述	访问
0xE0024080	PREINT	13	预分频值，整数部分	R/W
0xE0024084	PREFRAC	15	预分频值，小数部分	R/W

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 的值必须大于等于 1。

表 131 预分频整数寄存器（PREINT - 0xE0024080）

PREINT	功能	描述	复位值
15:13	保留	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
12:0	预分频整数	包含 RTC 预分频值的整数部分	0

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 132 预分频小数寄存器（PREFRAC - 0xE0024084）

PREFRAC	功能	描述	复位值
15	保留	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
14:0	预分频小数	包含 RTC 预分频值的小数部分	0

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, PREFRAC = pclk – ((PREINT+1)×32768) = 1

使用此设定，每秒钟有 32767 次 2 个 pclk 周期计数，1 次 3pclk 周期计数，加起来每秒刚好为 RTC 提供 32768 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, PREFRAC = pclk – ((PREINT+1)×32768) = 5760

这时，有 5760 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 65536kHz 的 pclk 频率（每秒钟的周期数必须是偶数）转换成 RTC 的 32768Hz 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 32768 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的应用中可能需要注意这种“抖动”。

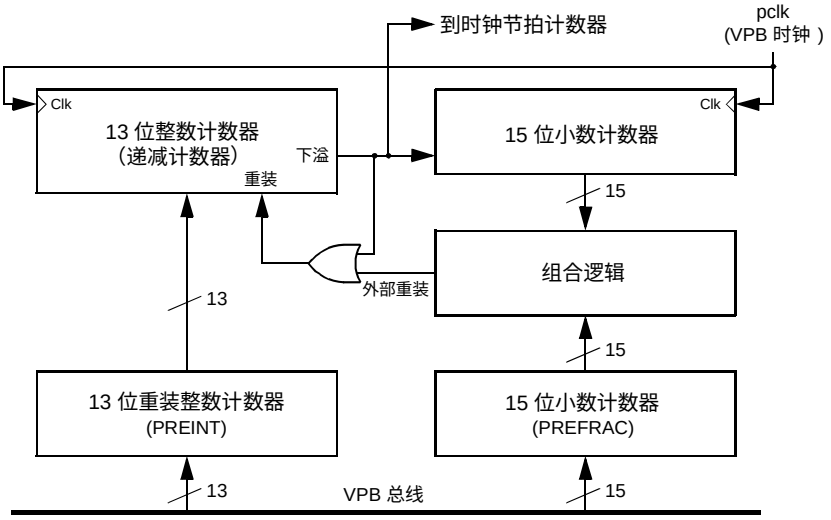


图 32 RTC 预分频器方框图

15. 看门狗

特性

- 如果没有周期性重装，则产生片内复位
- 调试模式
- 由软件使能，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}×4 倍数的时间周期：从(t_{pclk}×256×4) 到 (t_{pclk}×2^{32}×4)

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，看门狗会产生一个系统复位。

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。定时器递减计数。定时器递减的最小值为 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}×256×4)，最大间隔为(t_{pclk}×2^{32}×4)。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 0x00000000 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。

寄存器描述

表 133 看门狗寄存器映射

地址	名称	描述	访问	复位值
0xE0000000	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。	读/设置	0
0xE0000004	WDTC	看门狗定时器常数寄存器 该寄存器决定超时值。	读/写	0xFF
0xE0000008	WDFEED	看门狗喂狗寄存器 向该寄存器顺序写入 AAh 和 55h 使看门狗定时器重新装入预设值。	只写	NA
0xE000000C	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。	只读	0xFF

看门狗模式寄存器（WDMOD - 0xE00000000）

WDMOD 寄存器通过 WDEN 和 RESET 的组合来控制看门狗的操作。

WDEN	WDRESET	
0	X	看门狗关闭时的调试/操作
1	0	带看门狗中断的调试，但没有 WDRESET
1	1	带看门狗中断和 WDRESET 的操作

一旦 WDEN 和/或 WDRESET 位设置以后，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位清零。

表 134 看门狗模式寄存器（WDMOD - 0xE00000000）

WDMOD	功能	描述	复位值
0	WDEN	看门狗中断使能位（只能置位）	0
1	WDRESET	看门狗复位使能位（只能置位）	0
2	WDTOF	看门狗超时标志	0（外部复位）
3	WDINT	看门狗中断标志（只读）	0
7:4	保留	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

看门狗定时器常数寄存器（WDTC - 0xE00000004）

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}×256×4。

WDTC	功能	描述	复位值
31:0	计数值	看门狗超时间隔	0xFF

看门狗喂狗寄存器（WDFEED - 0xE00000008）

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，除非看门狗被触发。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 135 看门狗喂狗寄存器（WDFEED - 0xE00000008）

WDFEED	功能	描述	复位值
31:0	喂狗	喂狗值应当为 0xAA，然后是 0x55。	未定义

看门狗定时器值寄存器（WDTV - 0xE0000000C）

WDTV 寄存器用于读取看门狗定时器的当前值。

表 136 看门狗定时器值寄存器（WDTV - 0xE0000000C）

WDTV	功能	描述	复位值
31:0	计数	当前定时器值	0xFF

方框图

看门狗方框图如图 33 所示。

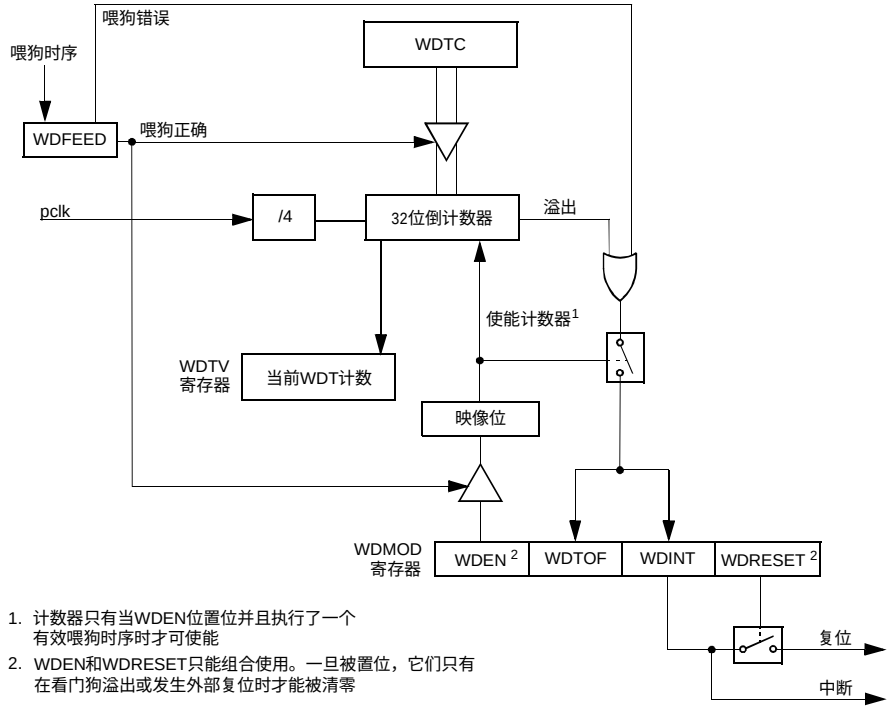


图 33 看门狗方框图

16. FLASH 存储器系统和编程

本章讲述 Flash 存储器系统和 Boot 装载程序。

FLASH 存储器系统

Flash 存储器系统包含 16 个扇区，每个扇区规格为 8K 字节。Flash 存储器从地址 0 开始并向上增加。详见*第 2 章 存储器寻址*。

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。

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。
- 在应用编程：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。

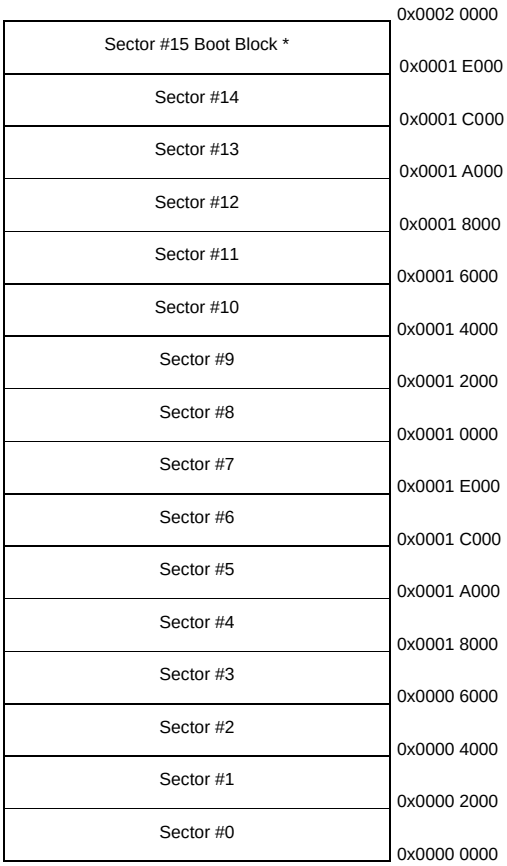
应用

Flash boot 装载程序同时提供片内 Flash 存储器的 ISP 和 IAP 编程接口。

描述

Flash boot 装载程序代码在上电或复位时执行。装载程序可执行 ISP 命令处理器或用户应用代码。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。该管脚由软件采样。假定在 RST 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，启动 ISP 命令处理器的外部硬件请求将被忽略。如果没有外部请求（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 在复位后处于高速模式，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。

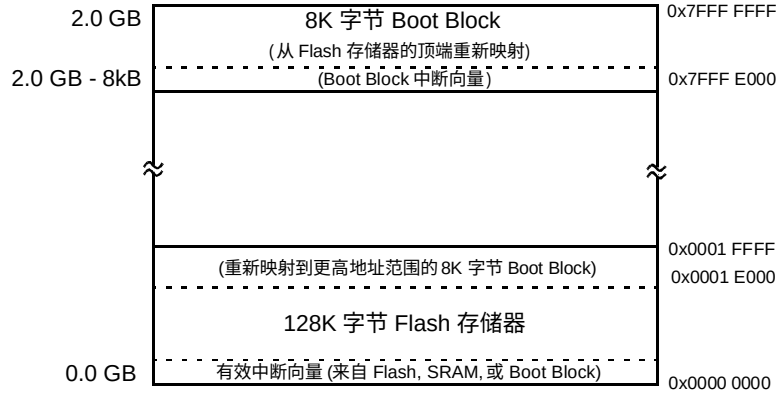


* 显示的地址并不反映Boot Block的重新映射。

图 34 Flash 扇区映射

复位后的存储器映射

Boot 扇区规格为 8kB，它位于片内 Flash 存储器最顶端的部分（从 0x0001 E000 开始）。在任何复位后，整个 boot 扇区映射到片内存储器空间的顶端，也就是说 boot 扇区出现在存储器从地址 0x7FFF E000 的区域。Flash boot 装载程序运行在这部分 Flash 区域，但 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 扇区当中，它在复位后被激活，即 boot 扇区的最低 64 字节也出现在存储器从地址 0x0000 0000 的区域。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：存储器区域并不是按照比例绘制的。

图 35 复位后低地址存储器的映射

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 扇区内重叠，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“？”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。解锁命令在 *ISP 命令*一节讲述。

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。

ISP 命令格式

“命令 参数_0 参数_1 ... 参数_n<CR><LF>”“数据”（只适用于写命令）

ISP 响应格式

“返回代码<CR><LF>响应_0<CR><LF>响应_1<CR><LF> ... 响应_n<CR><LF>" "数据"（只适用于读命令）

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。软化 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。

ISP 流程控制

软件 XON/XOFF 流程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控制机制。

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。

ISP 过程中的中断

在任何复位后，位于 Flash boot 扇区内的 boot block 中断向量都有效。

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。IAP 代码不使用或禁止中断。

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。

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。

Flash boot 装载程序的升级

ISP 和 IAP 命令不允许对片内 Flash 存储器中的 boot 扇区执行写操作。这样避免了用户破坏 Flash boot 扇区的风险。为了实现 Flash boot 装载程序的现场升级，片内 Flash 存储器的 boot 扇区不应连接硬件写保护。Boot 代码可通过下载到片内 RAM 的升级代码实现升级。使用 ISP“ Write ”和“ Go ”命令即可完成。现场升级代码包含新的 Flash boot 装载程序映像以及编程 boot 扇区所需要的代码。

BOOT 处理流程图

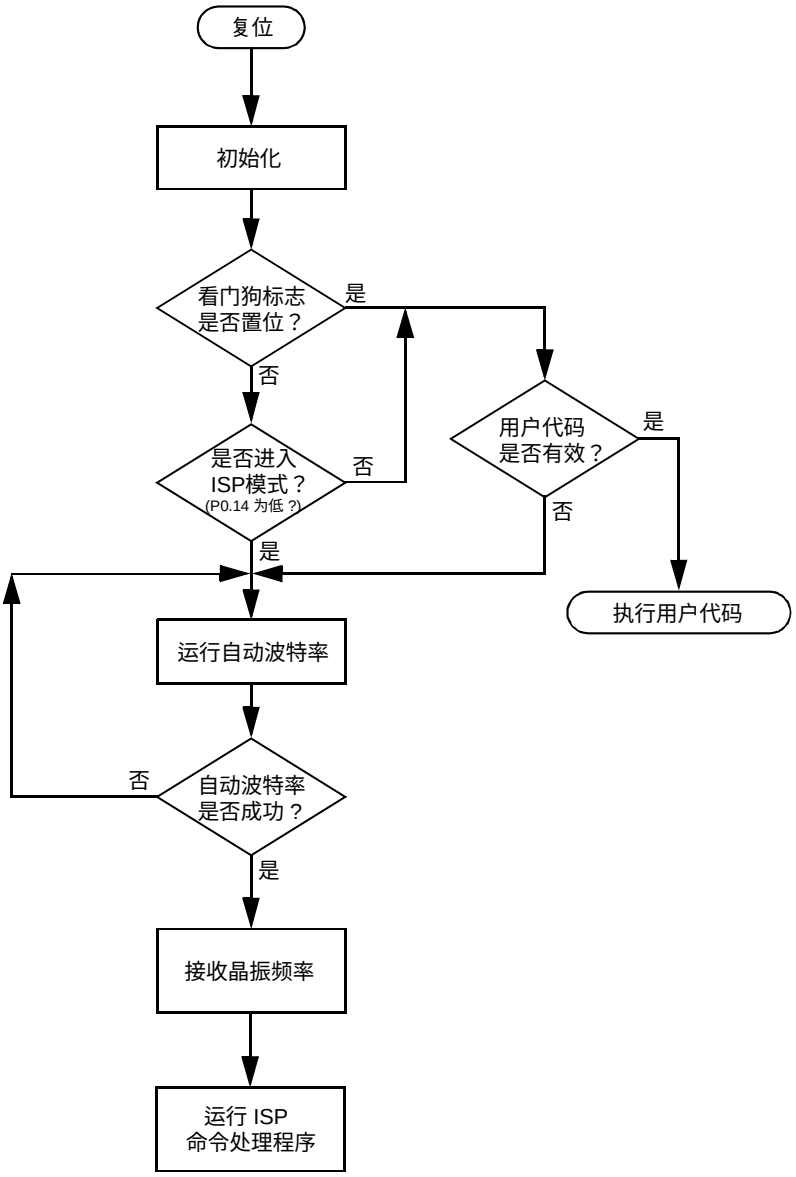


图 36 Boot 处理流程图

扇区数

有些 IAP 和 ISP 命令根据‘ 扇区 ’进行操作并指定扇区数。下面 3 个表列出了 32kB、64kB 和 128kB Flash 器件所包含的扇区数和存储器地址。IAP、ISP 和 RealMonitor 程序都位于扇区 15（boot 扇区）。boot 扇区存在于所有的器件当中。ISP 和 IAP 命令不允许对 boot 扇区执行写/擦除/运行操作。在 128K 器件中只有 120K Flash 可供用户程序使用。

表 137 32K 字节 Flash 器件中的扇区

扇区号	存储器地址
0	0x0000 0000 - 1FFF
1	0x0000 2000 - 3FFF
2	0x0000 4000 - 5FFF
3	0x0000 6000 - 7FFF

表 138 64K 字节 Flash 器件中的扇区

扇区号	存储器地址
0 - 3	同上表
4	0x0000 8000 - 9FFF
5	0x0000 A000 - BFFF
6	0x0000 C000 - DFFF
7	0x0000 E000 - FFFF

表 139 128K 字节 Flash 器件中的扇区

扇区号	存储器地址
0 - 7	同上表
8	0x0001 0000 - 1FFF
9	0x0001 2000 - 3FFF
10	0x0001 4000 - 5FFF
11	0x0001 6000 - 7FFF
12	0x0001 8000 - 9FFF
13	0x0001 A000 - BFFF
14	0x0001 C000 - DFFF
15	0x0001 E000 - FFFF

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返回代码。当接收到未定义的命令时，命令处理程序返回代码 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，处理程序才发送 CMD_SUCCESS。这时主机才能发送新的 ISP 命令。

表 140 ISP 命令汇总

ISP 命令	使用	描述
解锁	U <解锁代码>	见表 141
设置波特率	B <波特率> <停止位>	见表 142
回声	A <设定>	见表 143
写 RAM	W <起始地址> <字节数>	见表 144
读存储器	R <地址> <字节数>	见表 145
准备写操作的扇区	P <起始扇区号> <结束扇区号>	见表 146
将 RAM 内容复制到 Flash	C <Flash 地址> <RAM 地址> <字节数>	见表 147
运行	G <地址> <模式>	见表 148
擦除扇区	E <起始扇区号> <结束扇区号>	见表 149
扇区查空	I <起始扇区号> <结束扇区号>	见表 150
读器件 ID	J	见表 151
读 Boot 代码版本	K	见表 152
比较	M <地址 1> <地址 2> <字节数>	见表 153

解锁 <解锁代码>

表 141 ISP 解锁命令描述

命令	U
输入	解锁代码：23130
返回代码	CMD_SUCCESS | INVALID_CODE | PARAM_ERROR
描述	该命令用于解锁 Flash 写/擦除&运行命令。
举例	“ U 23130<CR><LF> ”解锁 Flash 写/擦除&运行命令。

设置波特率 <波特率> <停止位>

表 142 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。

回声 <设定>

表 143 ISP 回音命令描述

命令	A
输入	设定: 打开=1 | 关闭=0
返回代码	CMD_SUCCESS | PARAM_ERROR
描述	回声命令的默认设定是打开。当打开时，ISP 命令处理器将接收到的数据发送回主机。
举例	"A 0<CR><LF>" 回声关闭。

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。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，接收器响应 “ RESEND<CR><LF> ”。作为响应，发送器应当将字节重新发送。

表 144 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
描述	该命令用于将数据下载到 RAM。数据应当为 UU 编码格式。
举例	"W 1073741824 4<CR><LF>"向地址 0x4000 0000 写入 4 个字节数据。

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。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。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 145 ISP 写 RAM 命令描述

命令	R
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
描述	该命令用于读出 RAM 或 Flash 存储器的数据。
举例	"W 1073741824 4<CR><LF>"从地址 0x4000 0000 读出 4 个字节数据。

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 146 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 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。
举例	"P 0 0<CR><LF>"准备 Flash 扇区 0。

将 RAM 内容复制到 Flash <Flash 地址> <RAM 地址> <字节数>

表 147 ISP 将 RAM 内容复制到 Flash 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 512 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 512 | 1024 | 4096 | 8192.
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 512 | 1024 | 4096 | 8192) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR
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，扇区将自动受到保护。该命令不能写 boot 扇区。
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。

运行<地址><模式>

表 148 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
描述	该命令用于执行（调用）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。如果执行的代码以返回指令结束，则恢复 ISP 处理程序的执行。
举例	"G 0 A<CR><LF>"跳转到 ARM 模式下的地址 0x0000 0000 处。

擦除扇区<起始扇区号><结束扇区号>

表 149 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
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。
举例	"E 2 3<CR><LF>"擦除 Flash 扇区 2 和 3。

扇区查空<起始扇区号><结束扇区号>

表 150 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。要查空单个扇区可将起始和结束扇区号设定为相同值。
举例	"I 2 3<CR><LF>"对 Flash 扇区 2 和 3 进行查空。**由于扇区 0 的前 64 字节重新映射到 Flash boot 扇区，因此对其进行查空一定会失败。**

读器件 ID

表 151 ISP 读器件 ID 命令描述

命令	J
输入	无
返回代码	CMD_SUCCESS 后跟 ASCII 格式的 ID 号。
描述	该命令用于读取器件的 ID 号。
举例	"J<CR><LF>"

读 Boot 代码版本

表 152 ISP 读 Boot 代码版本命令描述

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。
举例	"K<CR><LF>"

比较<地址 1><地址 2><字节数>

表 153 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 | (源和目标数据相同) COMPARE_ERROR | (后跟第一个不匹配字节的地址) COUNT_ERROR (字节数不是 4 的倍数) | ADDR_ERROR | ADDR_NOT_MAPPED | PARAM_ERROR
描述	该命令用于读取 boot 代码版本号。
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。**当源或目标地址包含从地址 0 开始的前 64 字节中的任意一个时，比较的结果不一定正确。前 64 字节重新映射到 Flash boot 扇区。**

表 154 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	目标地址没有位于到存储器映射中。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小于起始扇区号。
8	SECTOR_NOT_BLANK	扇区非空
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	为写操作准备扇区命令未执行。
10	COMPARE_ERROR	源和目标数据不相等。
11	BUSY	Flash 编程硬件接口忙
12	PARAM_ERROR	参数不足或无效参数
13	ADDR_ERROR	地址没有以字为边界
14	ADDR_NOT_MAPPED	地址没有位于存储器映射中。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

IAP 命令

对于在应用编程来说，应当通过寄存器 r0 中的字指针指向存储器(RAM)包含的命令代码和参数来调用 IAP 程序。IAP 命令的结果返回到寄存器 r1 所指向的返回表。用户可通过传递寄存器 r0 和 r1 中的相同指针重用命令表来得到结果。参数表应当大到足够保存所有的结果以防结果的数目大于参数的数目。参数传递见图 37。参数和结果的数目根据 IAP 命令而有所不同。参数的最大数目为 5，由“将 RAM 内容复制到 Flash”命令传递。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 程序位于地址 0x7FFFFFF0。下面的符号定义可用于连接 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 编程，那么用户程序不应使用该空间。

表 155 IAP 命令汇总

ISP 命令	命令代码	描述
准备编程扇区	50	见表 156
将 RAM 内容复制到 Flash	51	见表 157
擦除扇区	52	见表 158
扇区查空	53	见表 159
读器件 ID	54	见表 160
读 boot 代码版本	55	见表 161
比较	56	见表 162

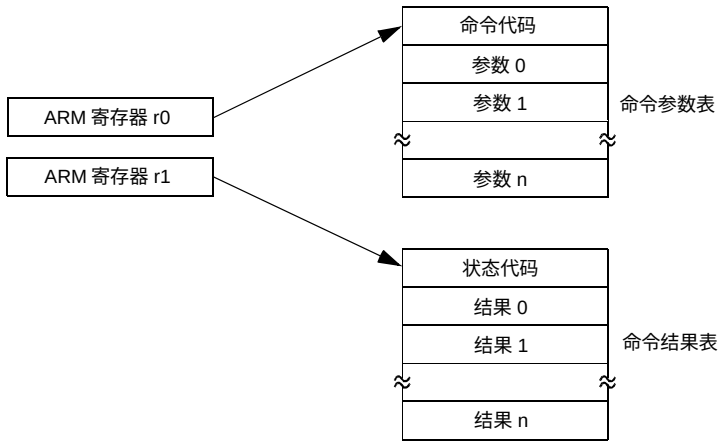


图 37 IAP 参数传递

准备编程扇区

该命令使 Flash 写/擦除操作分两步执行。

表 156 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 参数 0：起始扇区号 参数 1：结束扇区号：应当大于等于起始扇区号。
状态代码	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。

将 RAM 内容复制到 Flash

表 157 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 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 512 字节。 参数 1 (SRC)： 读出数据字节的源 RAM 地址。 参数 2：写入字节的数目。应当为 512 | 1024 | 4096 | 8192. 参数 3：系统时钟频率（CCLK）(单位：KHz)
状态代码	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 512 | 1024 | 4096 | 8192) | 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，扇区将自动受到保护。该命令不能写 boot 扇区。

擦除扇区

表 158 IAP 擦除扇区命令描述

命令	擦除扇区
输入	命令代码：52 参数 0：起始扇区号 参数 1：结束扇区号：应当大于等于起始扇区号。 参数 2：系统时钟频率（CCLK）(单位：KHz)
状态代码	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。

扇区查空

表 159 IAP 扇区查空命令描述

命令	扇区查空
输入	命令代码：53 参数 0：起始扇区号 参数 1：结束扇区号：应当大于等于起始扇区号。
状态代码	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。

读器件 ID

表 160 IAP 读器件 ID 命令描述

命令	读器件 ID
输入	命令代码：54 参数：无
返回代码	CMD_SUCCESS
结果	结果 0：器件 ID 号
描述	该命令用于读取器件的 ID 号。

读 Boot 代码版本

表 161 ISP 读 Boot 代码版本命令描述

命令	读 Boot 代码版本
输入	命令代码：55 参数：无
返回代码	CMD_SUCCESS
结果	结果 0：2 字节 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。

比较

表 162 IAP 比较命令描述

命令	比较
输入	命令代码：56 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	该命令用于读取 boot 代码版本号。**当源或目标地址包含从地址 0 开始的前 64 字节中的任意一个时，比较的结果不一定正确。前 64 字节重新映射到 Flash boot 扇区。**

表 163 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。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	目标地址没有位于到存储器映射中。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小于起始扇区号。
8	SECTOR_NOT_BLANK	扇区非空
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	为写操作准备扇区命令未执行。
10	COMPARE_ERROR	源和目标数据不相等。
11	BUSY	Flash 编程硬件接口忙

JTAG Flash 编程接口

3 种可能实现该接口：

1. 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 命令“ 将 RAM 内容复制到 Flash ”。
2. 调试工具可通过 JTAG 端口执行 Flash 编程代码。
3. 调试工具可直接使用 JTAG Flash 测试接口写 Flash 存储器。

17. EmbeddedICE 逻辑

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 直接与内核进行对话
- 在 ARM7TDMI-S 内核中直接插入指令
- 通过插入不同类型的指令可对 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

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。

描述

ARM7TDMI-S 调试结构使用现有的 JTAG* 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号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-S 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。
- 断点可以配置为在一段地址范围内对观察点有效。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。

* 详见 IEEE 标准 1149.1-1990《标准测试访问端口和边界扫描结构》。

管脚描述

表 164 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
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 信号的下降沿输出。
nTRST	输入	**测试复位** nTRST 管脚可用于复位 EmbeddedICE 逻辑中的测试逻辑。
DBGSEL	输入	**调试选择** 如果复位时为低电平，P0.17-P0.21 管脚通过管脚连接模块配置为可选功能。如果复位时为高电平，则进入调试模式。
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 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不同时钟频率的目标系统的同步。该信号还用于在进入调试模式时选择首要或次要 JTAG 管脚。

寄存器描述

EmbeddedICE 逻辑包含 16 个寄存器，见表 165。ARM7TDMI-S 的调试结构在“ ARM7TDMI-S(Rev 4) 技术参考手册 ”中作了详细描述。

表 165 EmbeddedICE 逻辑寄存器

地址	宽度	名称	描述
00000	6	调试控制	强制调试状态，禁止中断
00001	5	调试状态	调试状态
00100	32	调试通信控制寄存器	调试通信控制寄存器
00101	32	调试通信数据寄存器	调试通信数据寄存器
01000	32	观察点 0 地址值	保存观察点 0 地址值
01001	32	观察点 0 地址屏蔽	保存观察点 0 地址屏蔽
01010	32	观察点 0 数据值	保存观察点 0 数据值
01011	32	观察点 0 数据屏蔽	保存观察点 0 数据屏蔽
01100	9	观察点 0 控制值	保存观察点 0 控制值
01101	8	观察点 0 控制屏蔽	保存观察点 0 控制屏蔽
10000	32	观察点 1 地址值	保存观察点 1 地址值
10001	32	观察点 1 地址屏蔽	保存观察点 1 地址屏蔽
10010	32	观察点 1 数据值	保存观察点 1 数据值
10011	32	观察点 1 数据屏蔽	保存观察点 1 数据屏蔽
10100	9	观察点 1 控制值	保存观察点 1 控制值
10101	8	观察点 1 控制屏蔽	保存观察点 1 控制屏蔽

方框图

调试环境的方框图如图 38 所示。

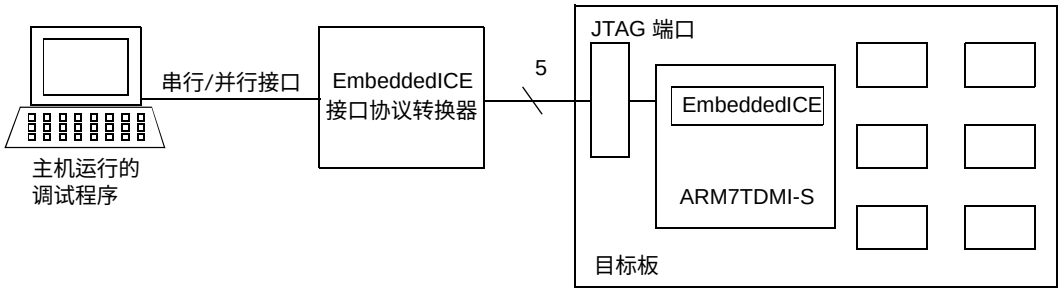


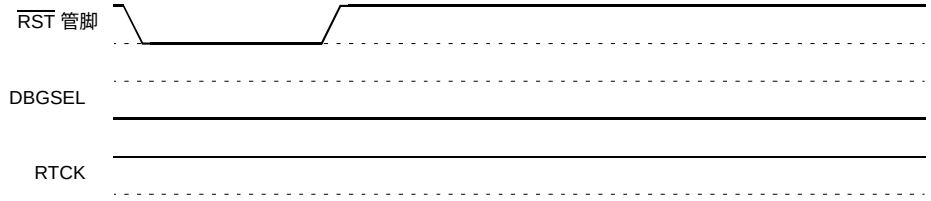
图 38 EmbeddedICE 调试环境方框图

调试模式

调试模式允许使用仿真器或其它开发工具通过 JTAG 接口进行程序调试。

调试模式入口

调试模式通过管脚 DBGSEL 进入。在管脚 RTCK 的帮助下选择两个调试变量。要进入调试模式，DBGSEL 必须在 CPU 复位过程和复位结束之后保持高电平。要实现正常（非调试）操作，DBGSEL 必须一直保持低电平，见图 39。



- DBGSEL 总是接低电平。如果没有它没有被外部拉高，内部下列将使 DBGSEL 保持低电平。

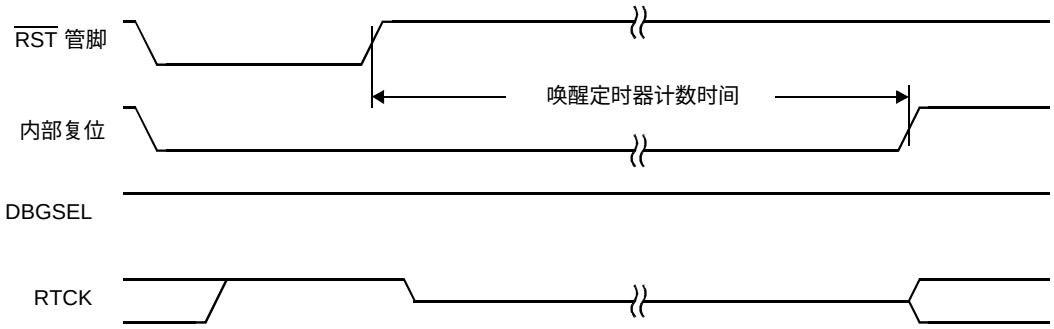
- RTCK 在应用中没有连接，它被内部上拉为高电平。

图 39 正常操作（非调试模式）的波形

如果使用首要 JTAG 管脚进行调试，RTCK 必须在 $\overline{\text{RST}}$ 管脚释放时保持高电平，见图 40。RTCK 可以被外部驱动为高电平，或通过它的内部上拉保持高电平。在内部唤醒时间内，RTCK 输出驱动器被禁止。外部复位撤除和内部复位撤除之间有一个间隔。在这段间隔时间里，如果有必要可以使用外部信号驱动 RTCK。

这一过程将 P0.17 – P0.21 设置为 JTAG 测试/调试接口。如果这些管脚初始化为 JTAG 管脚，管脚连接模块的设定将对它们无效。

如果使用次要 JTAG 管脚进行调试，必须通过软件将相关的管脚连接到 JTAG 端口。这通过管脚连接模块来完成。



- DBGSEL 必须为高电平。
- 当 RST 释放时，RTCK 必须为高电平。内部上拉会使 RTCK 保持高电平（如果没有被外部拉低）
- 当内部唤醒定时器解除内部芯片复位时，RTCK 输出驱动器必须打开。

图 40 使用首要 JTAG 管脚时的调试模式入口波形

18. 嵌入式跟踪宏单元

特性

- 跟踪 ARM 内核正在执行的指令
- 10 线接口
- 1 个外部触发输入
- 所有寄存器都通过 JTAG 接口编程
- 不使用跟踪时不消耗功率
- 可以编程为当超过跟踪端口带宽时停止 ARM 内核
- 支持 THUMB 指令集

应用

由于微控制器带有大量的片内存储器，因此不能简单地通过观察外部管脚来确定处理器核是如何运行的。ETM 对深嵌入处理器内核提供了实时跟踪能力。它向一个跟踪端口输出处理器执行的信息。软件调试器允许使用 JTAG 接口对 ETM 进行配置并以用户易于理解的格式显示捕获到的跟踪信息。

描述

ETM 直接连接到 ARM 内核而不是主 AMBA 系统总线。它将跟踪信息压缩并通过一个窄带跟踪端口输出。外部跟踪端口分析仪在软件调试器的控制下捕获跟踪信息。跟踪端口可以广播指令跟踪信息。指令跟踪(或 PC 跟踪)显示了处理器的执行流程并提供所有已执行指令的列表。指令跟踪被显著压缩为广播分支地址和一套用于指示流水线状态的状态信号。跟踪信息的产生可通过选择触发源进行控制。触发源包括地址比较器、计数器和序列发生器。由于跟踪信息被压缩，软件调试器需要一个执行代码的静态映像。由于这个限制，自修改代码无法被跟踪。

ETM 配置

ETM 宏单元使用下面的标准配置。

表 166 ETM 配置

资源数/类型	Small [1]
地址比较器对	1
数据比较器	0（不支持数据跟踪）
存储器映射译码器	4
计数器	1
时序发生器	无
外部输入	2
外部输出	0
FIFOFULL 信号	0（未连接）
FIFO 深度	10 字节
跟踪包宽度	4/8

1. 详见 ARM 文档“嵌入式跟踪宏单元规范”(ARM IHI 0014E)。

管脚描述

表 167 ETM 管脚描述

管脚名称	类型	描述
TRACECLK	输出	**跟踪时钟** 跟踪时钟信号为跟踪端口提供时钟。PIPESTAT[2:0], TRACESYNC 和 TRACEPKT[3:0]信号以跟踪时钟的上升沿为参照。该时钟并不由 ETM 时钟所产生，而是由系统时钟得来的。该时钟应当为跟踪数据信号提供足够的保持时间。支持半速率时钟模式。跟踪数据信号应当从 TRACECLK 的时钟相位移出。详见“ETM7 技术参考手册”(ARM DDI 0158B)，而 TRACECLK 时序请参阅“嵌入式跟踪宏单元规范”(ARM IHI 0014E)。
PIPESTAT[2:0]	输出	**流水线状态** 流水线状态信号提供处理器流水线执行阶段中所发生状况的指示
TRACESYNC	输出	**跟踪同步** 跟踪同步信号用于指示一组跟踪包当中的第一个包。并且仅为任何分支地址的第一个包声明为高电平。
TRACEPKT[3:0]	输出	**跟踪包** 跟踪包信号用于输出打包的关于流水线状态的地址和数据信息。所有包的长度都为 8 个字节。一个包需要两个周期输出，在第一个周期中，Packet[3:0]输出，第二个周期 Packet[7:4]输出。
EXTING[0]	输入	**外部触发输入**

寄存器描述

ETM 包含 29 个寄存器，见表 168。有关它们的详细描述见 ARM 有限公司出版的 ARM IHI 0014E 文档，可从网站 http://www.arm.com 下载。

表 168 ETM 寄存器

寄存器编码	名称	描述	访问
000 0000	ETM 控制	控制 ETM 的一般操作	R/W
000 0001	ETM 配置代码	允许调试器读取每种资源类型的数目。	RO
000 0010	触发事件	保存控制事件	WO
000 0011	存储映射译码控制	8 位寄存器，用于静态配置存储器映射译码器。	WO
000 0100	ETM 状态	保存挂起的溢出状态位	RO
000 0101	系统配置	保存使用 SYSOPT 总线的配置信息	RO

续上表

寄存器编码	名称	描述	访问
000 0110	跟踪使能控制 3	保存跟踪使能/禁止地址	WO
000 0111	跟踪使能控制 2	保存比较的地址	WO
000 1000	跟踪使能事件	保存使能的事件	WO
000 1001	跟踪使能控制 1	保存包含和排除的区域	WO
000 1010	FIFOFULL 区域	保存包含和排除的区域	WO
000 1011	FIFOFULL 水平	保存认为 FIFO 已满的值	WO
000 1100	ViewData 事件	保存使能的事件	
000 1101	ViewData 控制 1	保存包含/排除的区域	WO
000 1110	ViewData 控制 2	保存包含/排除的区域	WO
000 1111	ViewData 控制 3	保存包含/排除的区域	WO
001 xxxx	地址比较器 1~16	保存比较的地址	WO
010 xxxx	地址访问类型 1~16	保存访问的类型和规格	WO
000 xxxx	保留	—	—
100 xxxx	保留	—	—
101 00xx	初始计数值 1~4	保存计数器的初始值	WO
101 01xx	计数器使能 1~4	保存计数器时钟使能控制和事件	WO
101 10xx	计数器重装 1~4	保存计数器重装事件	WO
101 11xx	计数值 1~4	保存当前计数器值	RO
110 00xx	时序状态和控制	保存下一个状态触发的事件	—
110 10xx	外部输出 1~4	保存每个输出的控制事件	WO
110 11xx	保留	—	—
111 0xxx	保留	—	—
111 1xxx	保留	—	—

方框图

ETM 调试环境的方框图如图 41 所示。

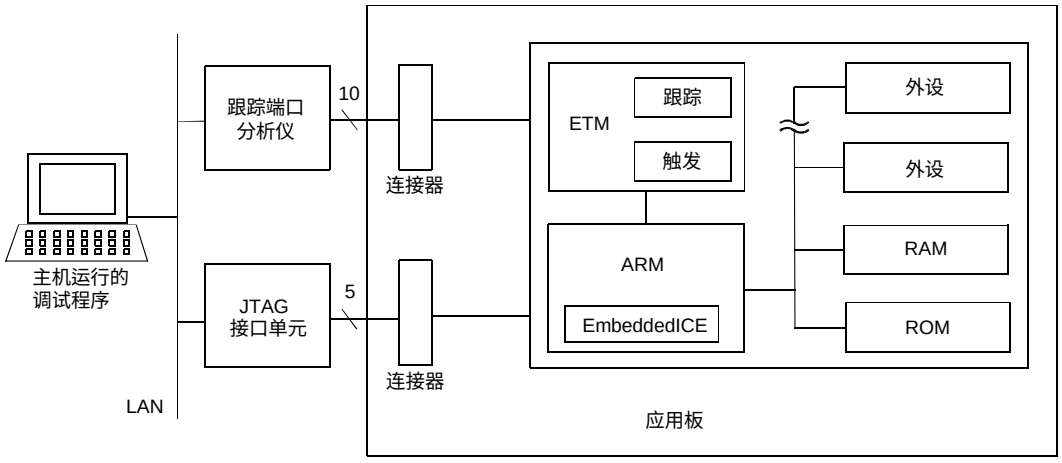


图 41 ETM 调试环境方框图

19. 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。它是一个非常小的调试监控器，当用户对运行在前台的应用程序进行调试时，它运行在后台。它使用 DCC(调试通信通道)（EmbeddedICE 逻辑中包含了 DCC）与主机进行通信。LPC2104/2105/2106 包含一个编程到片内 Flash 存储器内的 RealMonitor 软件，用于时序特定的配置。

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。

应用

- 实时调试

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，此外，它还包含了 Multi-ICE 的通信机制（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 Flash 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。

RealMonitor 部件

如图 42 所示，RealMonitor 分成两个功能部件：

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 RDI 信息。详细信息请参考“ ARM RMHost 用户指南 ”（ARM DUI 0137A）。

RMTarget

这部分预先编程到片内 Flash 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。详见“ RealMonitor 目标集成指南 ”（ARM DUI 0142A）。

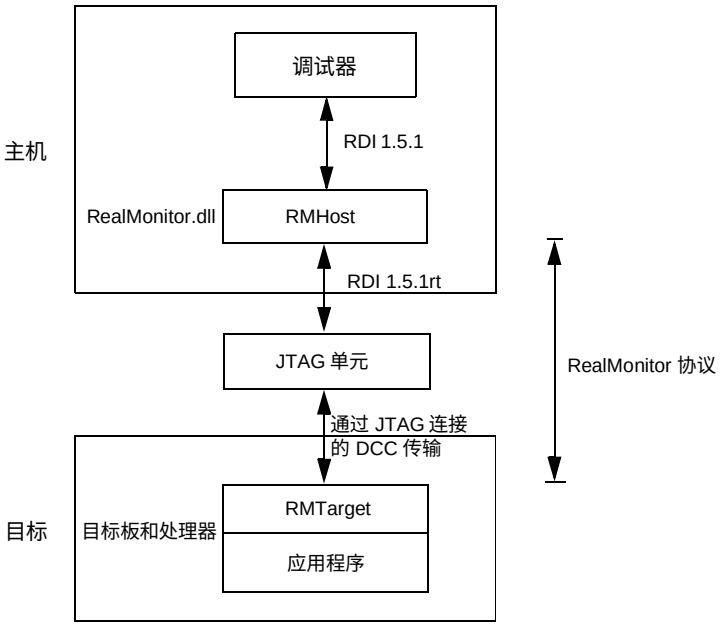


图 42 RealMonitor 部件

RealMonitor 是如何工作的

通常情况下，RealMonitor 作为一个状态机，如图 43 所示。为了响应从主机接收到的包，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。

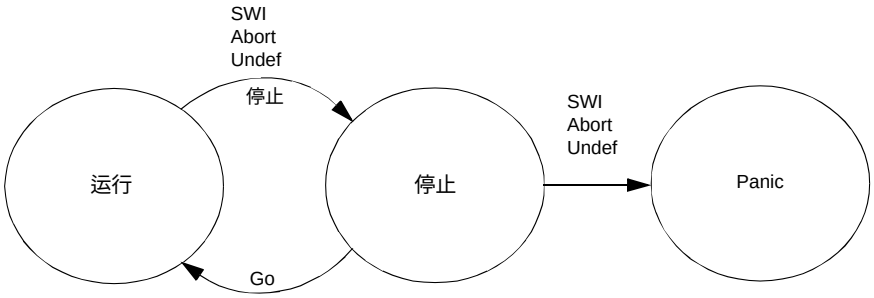


图 43 作为状态机的 RealMonitor

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 42 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。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。

- 包含中断和溢出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 m_ReceiveData()（RealMonitor 内部函数）。如果 DCC 写缓冲区空闲，控制权传递给 m_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操作的优先级高于写操作。
- 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。

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。

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，见表 139。

表 139 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

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。

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。

SVC 模式

RealMonitor 不使用该堆栈。

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。

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。

用户/系统模式

RealMonitor 不使用该堆栈。

FIQ 模式

RealMonitor 不使用该堆栈。

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。

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 44 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_IRQHandler()）。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，转移指令的目标地址为相关的 RealMonitor 异常处理程序的起始地址。

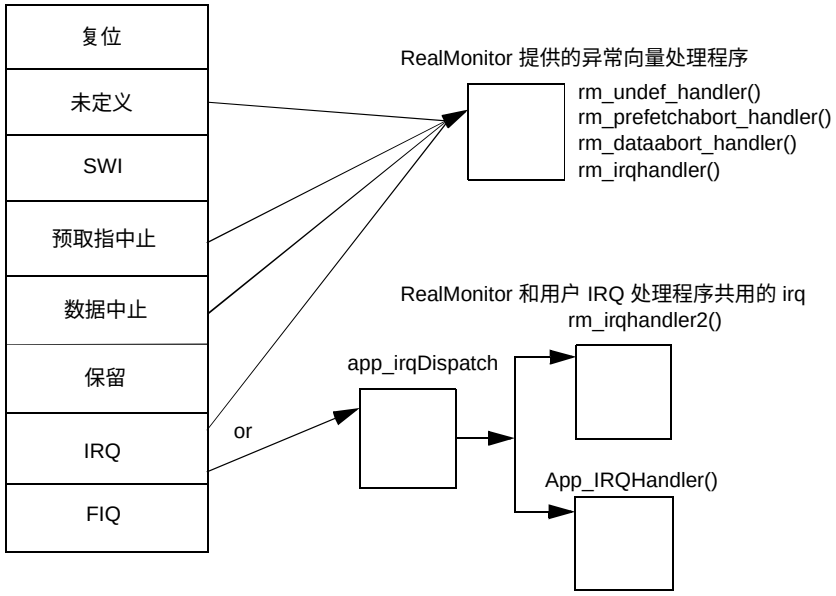


图 44 异常处理程序

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：call rm_init_entry()

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。指令连接器敬爱你跟代码放置在地址 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
; /*********************************************************************
; * 获取用户程序的入口。
; *********************************************************************/
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，通过硬件使
; 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
```

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 (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