UM10601

LPC800 用户手册

修订版：1.2 — 2013 年 3 月 14 日 **用户手册**

文档信息

信息	内容
关键词	ARM Cortex M0+， LPC800， USART， I2C， LPC810M021FN8，LPC811M001FDH16， LPC812M101FDH16， LPC812M101FD20，LPC812M101FDH20
摘要	LPC800 用户手册

修订记录

版本	日期	说明
1.2	20130314	LPC800 用户手册
变更内容：		• 编辑更新。 • 更新了表 40 “PLL 配置示例 ”。 • 更新了表 92 “ 模式匹配位片源寄存器 （PMSRC，地址 0xA000 402C）位说明 ” 中的寄存器位说明。 • 更新了章节 5 “LPC800 节能模式和电源管理单元 (PMU)”。 • 增加了章节 5.3.1 “ARM Cortex-M0+ 内核的低功耗模式 ”。 • 移除了表 212 “ 寄存器概述：FMC （基址 0x4004 0000） ” 中取决于系统频率的闪存访问时间。 • 增加了如何防止内部引脚浮空的说明。参见章节 6.3。 • 更新了图 30 “I2C 时钟 ”。 • 更新了 NMISRC 寄存器说明。参见章节 4.6.26 “NMI 源选择寄存器 ”。 • 增加了章节 16.3.1 “ 主机模式下的 I2C 发送 / 接收 ”。 • 增加了章节 14 “LPC800 ARM Cortex SysTick 定时器 (SysTick)”。 • 更正了 DEVICE_ID 寄存器的地址偏移。参见表 38 “ 器件 ID 寄存器（DEVICE_ID，地址 0x4004 83F8）位说明 ”。 • 在表28 “BOD控制寄存器（BODCTRL、地址0x4004 8150）位说明”中，“BOD复位电平0”改成“已保留”。
1.1	20130124	LPC800 用户手册
变更内容：		• 更正了闪存签名创建算法。参见第 19.5.1 节 “ 闪存签名生成 ”。 • 系统 PLL 输出频率限制为低于 100 MHz。 • 图 2 “LPC800 存储器映射 ” 中的 MTB 寄存器存储器空间更改为 1 kB。 • 更新了外部跟踪缓冲区命令寄存器的说明。参见第 4.6.20 节 “ 外部跟踪缓冲区命令寄存器 ”。 • 移除了表 3 中的 “ 闪存中断 ”。 • 增加第 27 章 “ARM Cortex-M0+ 指令集汇总 ”。 • 增加了 ISP 读取 CRC 校验和命令。增加了第 21.5.1.15 节 “ 读取 CRC 校验和 < 地址 > < 字节数 >”。 • 增加了第 20.3.1 节 “ 引导加载程序版本 ”。 • MRT 实施更改为 31 位定时器。参见第 11 章。更正了表 140“ 空闲通道寄存器（IDLE_CH，地址 0x4000 40F4）位说明 ” 中的位说明。 • 更新了第 17 章 “LPC800 SPI0/1” 中的说明。 • 更新了第 16 章 “LPC800 I2C 总线接口 ” 中的说明。 • 更新了第 15 章 “LPC800 USART0/1/2” 中的说明。 • 更新了第 8 章 “LPC800 引脚中断 / 模式匹配引擎 ” 中的说明。 • 更新了第 9.4 节 （开关矩阵 - 引脚功能框图）中的说明。 • 更新了第 5 章 “LPC800 低功耗模式和电源管理单元 (PMU)” 中的说明。 • 增加了第 3.3.2 节 “ 非屏蔽中断 (NMI)” 和第 3.3.3 节 “ 向量表偏移 ”。 • 更正了第 10.6 节中的位字段。 • 从 USART 特性中移除了 USART 波特率时钟输出。

修订记录 *（续）*

版本	日期	说明
1	20121109	初始版 LPC800 用户手册

联系信息

有关详细信息，请访问：http://www.nxp.com

欲咨询销售办事处地址，请发送电子邮件至：salesaddresses@nxp.com

1.1 前言

LPC800 是基于 ARM Cortex-M0+ 的低成本 32 位 MCU 系列产品，工作时 CPU 频率高达 30 MHz。 LPC800 支持最高 16 kB 的闪存和 4 kB 的 SRAM。

LPC800 的外设包括：一个 CRC 引擎、一个 I^2C 总线接口、多达三个 USART、多达两个 SPI 接口、一个多速率定时器、自唤醒定时器、状态可配置定时器、一个比较器、采用开关矩阵的功能可配置 I/O 端口、一个输入模式匹配引擎和多达 18 个通用 I/O 引脚。

1.2 特性

- 系统：
  - ARM Cortex-M0+ 处理器，运行时频率高达 30 MHz，集成了单周期乘法器和快速单周期 I/O 端口。
  - ARM Cortex-M0+ 内置可嵌套中断向量控制器 (NVIC)。
  - 系统节拍定时器。
  - 支持串行线调试 (SWD) 模式和 JTAG 边界扫描模式。
  - 支持微跟踪缓冲区 (MTB)。
- 存储器：
  - 最高 16 kB 片内可编程闪存，带 64 字节页面写入和擦除功能。
  - 4 kB SRAM。
- ROM API 支持：
  - 引导加载程序。
  - USART 驱动器。
  - I2C 驱动器。
  - 电源配置。
  - 闪存在应用编程 (IAP) 和在系统编程 (ISP)。
- 数字外设：
  - 连接至 ARM Cortex-M0+ IO 总线的高速 GPIO 接口，集成了多达 18 个通用 I/O (GPIO) 引脚，并具备可配置上拉 / 下拉电阻、可编程开漏模式、输入反相器和干扰滤波器。
  - 4 个引脚具备大电流输出驱动能力 (20 mA)
  - 2 个真正开漏引脚具备大电流灌入驱动能力 (20 mA)。
  - GPIO 中断生成能力， 8 个 GPIO 输入具有布尔模式匹配特性。
  - 开关矩阵，用于灵活配置每个 I/O 引脚功能。
  - 状态可配置定时器 (SCT)，输入和输出功能 （包括捕获和匹配）通过开关矩阵分配到引脚。
  - 多通道多速率定时器 (MRT)，以多达 4 种可编程固定速率生成可重复中断。
  - 自唤醒定时器 (WKT)，采用 IRC 或低功耗、低频率内部振荡器作为时钟输入。

  - CRC 引擎。
  - 窗口看门狗定时器 (WWDT)。
- 模拟外设：
  - 集成外部基准电压源的比较器，引脚功能通过开关矩阵分配或使能。
- 串行接口：
  - 3 个 USART 接口，引脚功能通过开关矩阵分配。
  - 2 个 SPI 控制器，引脚功能通过开关矩阵分配。
  - 1 个 I^2C 总线接口，引脚功能通过开关矩阵分配。
- 时钟生成：
  - 调整到 1 % 精度的 12 MHz 内部 RC 振荡器，可选择性地用作系统时钟。
  - 晶体振荡器，工作频率范围为 1 MHz 至 25 MHz。
  - 可编程看门狗振荡器，频率范围为 9.4 kHz 至 2.3 MHz。
  - 用于 WKT 的 10 kHz 低功耗振荡器。
  - PLL 使 CPU 无需使用高频晶体即可生成最高 CPU 主频。可从系统振荡器、外部时钟输入 CLKIN 或内部 RC 振荡器运行。
  - 带分频器的时钟输出功能，可反映晶体振荡器、主时钟、 IRC 或看门狗振荡器。
- 功率控制：
  - 可最大程度降低功耗的集成式 PMU （电源管理单元）。
  - 节能模式：睡眠模式、深度睡眠模式、掉电模式和深度掉电模式。
  - 深度睡眠模式和掉电模式可由 USART、 SPI 和 I2C 外设唤醒。
  - 深度掉电模式可由定时器控制进行自唤醒。
  - 上电复位 (POR)。
  - 掉电检测。
- 用于器件识别的独特序列号。
- 单电源。
- 采用的封装有：SO20 封装、 TSSOP20 封装、 TSSOP16 封装和 DIP8 封装。

1.3 订购信息

表 1. 订购信息

型号	封装		
	名称	说明	版本
LPC810M021FN8	DIP8	塑料双列直插式封装； 8 引脚 (300 mil)	SOT097-2
LPC811M001FDH16	TSSOP16	塑料减薄紧缩小型封装； 16 引脚；体宽 4.4 mm	SOT403-1
LPC812M101FDH16	TSSOP16	塑料减薄紧缩小型封装； 16 引脚；体宽 4.4 mm	SOT403-1
LPC812M101FD20	SO20	塑料小型封装； 20 引脚；体宽 7.5 mm	SOT163-1
LPC812M101FDH20	TSSOP20	塑料减薄紧缩小型封装； 20 引脚；体宽 4.4 mm	SOT360-1

表 2. 订购选项

型号	闪存 /kB	SRAM/kB	USART	I^2C	SPI	比较器	GPIO	封装
LPC810M021FN8	4	1	2	1	1	1	6	DIP8
LPC811M001FDH16	8	2	2	1	1	1	14	TSSOP16
LPC812M101FDH16	16	4	3	1	2	1	14	TSSOP16
LPC812M101FD20	16	4	2	1	1	1	18	SO20
LPC812M101FDH20	16	4	3	1	2	1	18	TSSOP20

1.4 功能框图

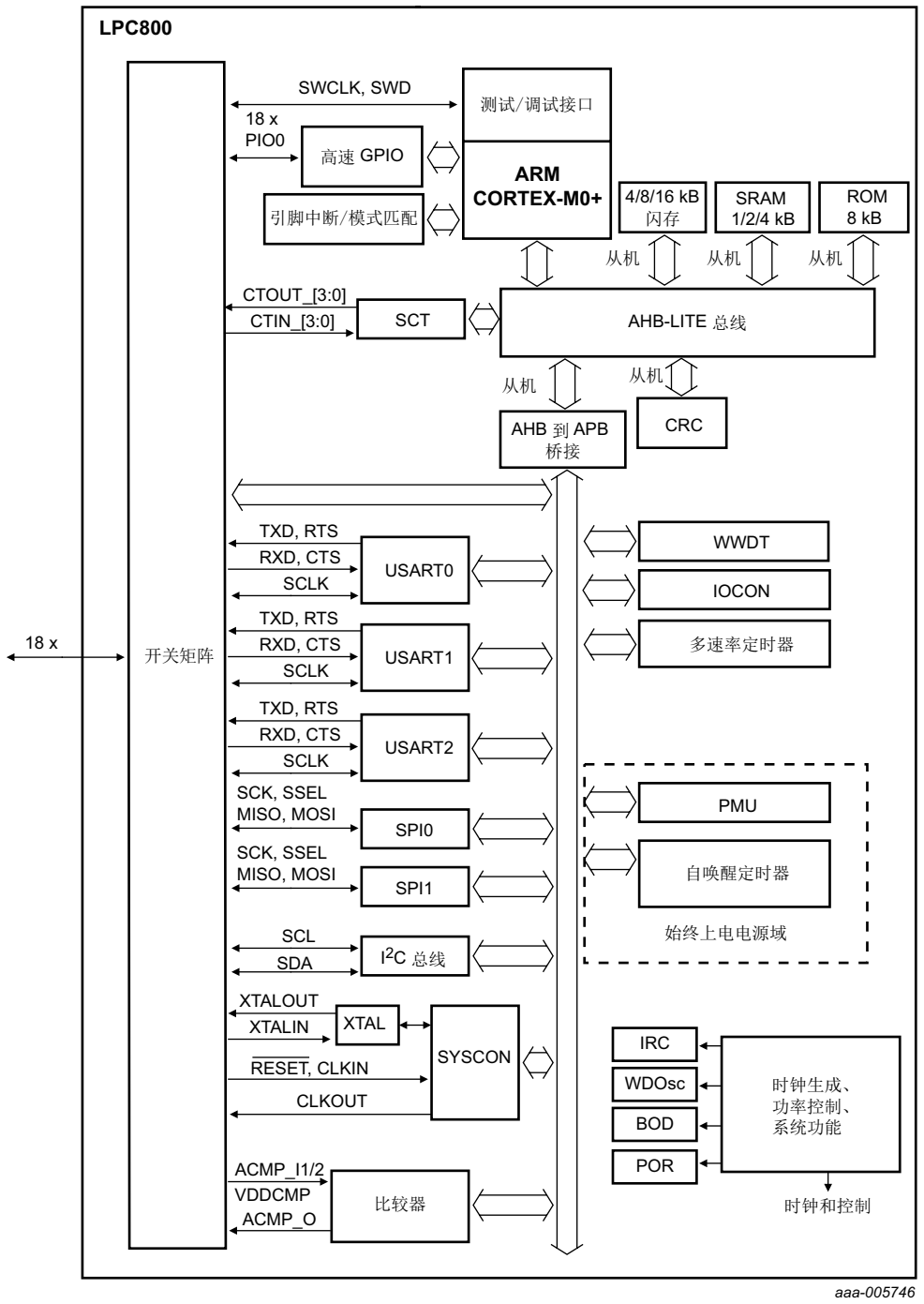


图 1. LPC800 功能框图

1.5 简介

1.5.1 ARM Cortex-M0+ 内核配置

ARM Cortex-M0+ 内核运行时的最高频率为 30 MHz。内核集成了 NVIC 和串行调试功能，具有 4 个断点和 2 个观察点。ARM Cortex-M0+ 内核支持单周期 I/O 使能端口 (IOP)，在地址 0xA000 0000 处可快速访问 GPIO。

内核集成了一个单周期乘法器和一个系统节拍定时器 (SysTick)。

2.1 本章导读

所有 LPC800 器件的存储器映射都是一样的。不同 LPC800 器件支持不同的闪存大小。

2.2 简介

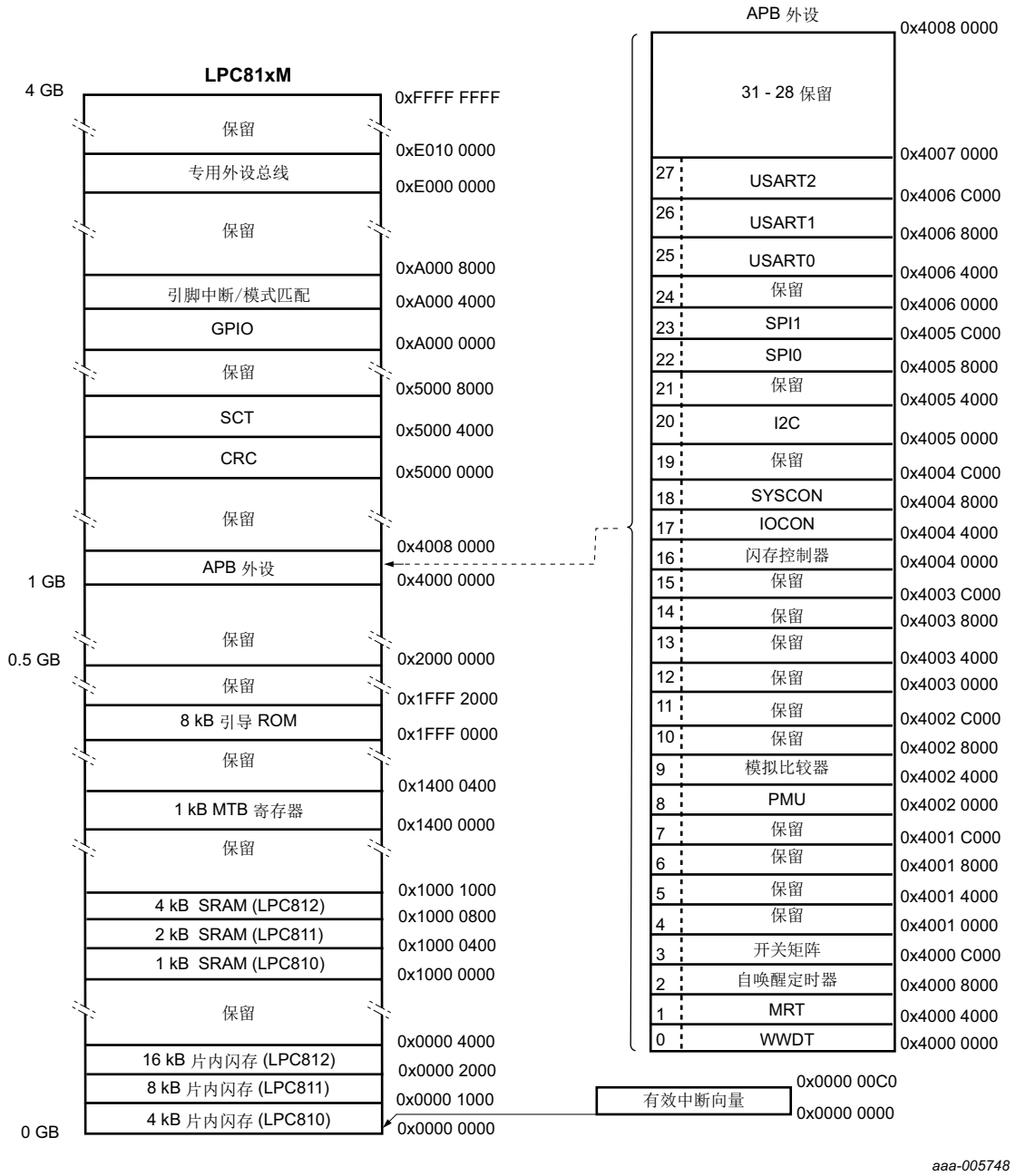
LPC800 集成了几个不同的存储区域。图 2 显示的是从用户编程角度出发，器件复位后整个地址空间的总映射图。

APB 外设区的大小为 512 kB，可分配多达 32 个外设。每个外设的空间大小均为 16 kB，地址译码得到简化。

NVIC、SysTick 和睡眠模式控制等 ARM Cortex-M0+ 内核集成的寄存器位于专用的外设总线上。

GPIO 端口和引脚中断 / 模式匹配寄存器可通过 ARM Cortex-M0+ 单周期 I/O 使能端口 (IOP) 访问。

2.2.1 存储器映射



专用外设总线包括 NVIC、 SysTick 和内核控制寄存器等 ARM Cortex-M0+ 外设。

图 2. LPC800 存储器映射

2.2.2 微跟踪缓冲区 (MTB)

LPC800 支持 ARM Cortex-M0+ 微跟踪缓冲区。参见章节 26.5.4。

3.1 本章导读

所有 LPC800 器件上的 NVIC 都是一样的。

仅器件 LPC812M101FDH20 和 LPC812M101FDH16 实施了 SPI1 中断和 USART2 中断。

3.2 特性

- 可嵌套中断向量控制器是 ARM Cortex-M0+ 的一个重要组成部分。
- 紧耦合方式使中断延迟大大缩短。
- 可控制系统的异常及外设中断。
- NVIC 支持 32 个向量中断。
- 4 个可编程的中断优先级 （带硬件优先级屏蔽功能）。
- 由 ARM 异常 （SVCall 和 PendSV）生成的软件中断 （参见参考 1）。
- 支持 NMI。
- 提供 ARM Cortex M0+ 向量表偏移寄存器 VTOR。

3.3 简介

可嵌套中断向量控制器 (NVIC) 是 Cortex-M0+ 的一个重要组成部分。它与 CPU 紧密结合，降低了中断延时，并让新进中断可以得到高效处理。

3.3.1 中断源

表 3 列出了每种外设功能的中断源。每个外设可以有一条或多条中断线连接到中断向量控制器。每条线可代表多个中断源。以相同优先级产生的中断将根据中断编号进行处理。

有关 NVIC 和 NVIC 寄存器描述的详细信息，请参见参考 1。

表 3. 中断源到 NVIC 的连接

中断编号	名称	说明	标志
0	SPI0_IRQ	SPI0 中断	参见表 194 “SPI 中断使能读取和设置寄存器（INTENSET，地址 0x4005 800C (SPI0)，0x4005 C00C (SPI1)）位说明 ”。
1	SPI1_IRQ	SPI1 中断	与 SPI0_IRQ 相同
2	-	保留	-
3	UART0_IRQ	USART0 中断	参见表 163 “USART 中断使能读取和设置寄存器（INTENSET，地址 0x4006 400C (USART0)，0x4006 800C (USART1)，0x4006 C00C (USART2)）位说明 ”。
4	UART1_IRQ	USART1 中断	与 UART0_IRQ 相同
5	UART2_IRQ	USART2 中断	与 UART0_IRQ 相同
6	-	保留	-
7	-	保留	-

表 3. 中断源到 NVIC 的连接 *（续）*

中断编号	名称	说明	标志
8	I2C0_IRQ	I2C0 中断	参见表 177 “中断使能清零寄存器（INTENCLR，地址 0x4005 000C）位 说明”。
9	SCT_IRQ	状态可配置定时器中断	EVFLAG SCT 事件
10	MRT_IRQ	多速率定时器中断	全局 MRT 中断 GFLAG0 GFLAG1 GFLAG2 GFLAG3
11	CMP_IRQ	模拟比较器中断	COMPEDGE - 上升沿、下降沿或两个边沿均可设置该位。
12	WDT_IRQ	窗口看门狗定时器中断	WARNINT - 看门狗警告中断
13	BOD_IRQ	BOD 中断	BODINTVAL - BOD 中断等级
14	-	-	保留
15	WKT_IRQ	自唤醒定时器中断	ALARMFLAG
23:16	-	保留	-
24	PININT0_IRQ	引脚中断 0 或模式匹配引擎片 0 中断	PSTAT - 引脚中断状态
25	PININT1_IRQ	引脚中断 1 或模式匹配引擎片 1 中断	PSTAT - 引脚中断状态
26	PININT2_IRQ	引脚中断 2 或模式匹配引擎片 2 中断	PSTAT - 引脚中断状态
27	PININT3_IRQ	引脚中断 3 或模式匹配引擎片 3 中断	PSTAT - 引脚中断状态
28	PININT4_IRQ	引脚中断 4 或模式匹配引擎片 4 中断	PSTAT - 引脚中断状态
29	PININT5_IRQ	引脚中断 5 或模式匹配引擎片 5 中断	PSTAT - 引脚中断状态
30	PININT6_IRQ	引脚中断 6 或模式匹配引擎片 6 中断	PSTAT - 引脚中断状态
31	PININT7_IRQ	引脚中断 7 或模式匹配引擎片 7 中断	PSTAT - 引脚中断状态

3.3.2 非屏蔽中断 (NMI)

LPC800 支持 NMI，可通过外设中断或软件触发。 NMI 是除复位之外优先级最高的异常。

由于 NMI 使用 SYSCON 模块中的 NMISRC 寄存器（参见表 3），因此可设置表 31 中所列的全部外设中断。为避免 NMI 异常和正常中断共用一个外设中断，当配置为 NMI 时，可在 NVIC 中禁用中断。

3.3.3 向量表偏移

向量表包含堆栈指针的复位值以及所有异常处理程序的起始地址（也称为异常向量）。系统复位时，向量表的地址为 0x0000 0000。软件可向 NVIC 中的 VTOR 寄存器写入数据，以便将向量表的起始地址重分配给不同的存储器位置。有关 VTOR 寄存器的说明，请参见参考 3。

4.1 本章导读

所有LPC800器件的系统配置模块都是一样的。仅器件LPC812M101FDH20和LPC812M101FDH16才提供 USART2 和 SPI1，相应时钟、复位和唤醒控制位在所有其他器件中均予以保留。

4.2 特性

- 时钟控制
  - 配置系统 PLL。
  - 配置系统振荡器和看门狗振荡器。
  - 使能独立外设时钟和存储器时钟。
  - 配置时钟输出。
  - 配置时钟分频器、数字滤波器时钟和 USART 波特率时钟。
- 监控和解除独立外设的复位。
- 选择外部引脚中断和模式匹配引擎的引脚。
- 低功耗模式配置。
- 唤醒控制。
- BOD 配置。
- MTB 跟踪启动和停止。
- 中断延迟控制。
- 选择 NMI 的信号源。
- 校准系统节拍定时器。

4.3 基本配置

根据下列步骤配置 SYSCON 模块：

- SYSCON 使用 CLKIN、CLKOUT、$\overline{\text{RESET}}$ 和 XTALIN/OUT 引脚。通过开关矩阵配置引脚功能。参见章节 4.4。
- 无需时钟配置。SYSCON 模块时钟始终使能。默认情况下，SYSCON 模块由 IRC 计时。

4.3.1 设置 PLL

PLL 可在频率高于输入时钟频率的情况下建立稳定的输出时钟。如需主机时钟的频率高于 IRC 时钟的 12 MHz，则使用 PLL 提高输入频率。

1. 在 PDRUNCFG 寄存器中使系统 PLL 上电。
   章节 4.6.32 “ 电源配置寄存器 ”
2. 在 SYSPLLCLKSEL 寄存器中选择 PLL 输入。输入选项如下：
   - IRC：12 MHz 内部振荡器。
   - 系统振荡器：使用 XTALIN/XTALOUT 引脚的外部晶体振荡器。

– 外部时钟输入 CLKIN。通过开关矩阵选择该引脚。

章节 4.6.8 “ 系统 PLL 时钟源选择寄存器 ”

3. 在 SYSPLLCLKUEN 寄存器中更新 PLL 时钟源。

章节 4.6.9 “ 系统 PLL 时钟源更新寄存器 ”

4. 配置 PLL M 分频器和 N 分频器。

章节 4.6.3 “ 系统 PLL 控制寄存器 ”

5. 监控 PLL 锁定状态，等待 PLL 锁定。

章节 4.6.4 “ 系统 PLL 状态寄存器 ”

4.3.2 配置主时钟和系统时钟。

寄存器和存储器的时钟源产生于主时钟。主机时钟可在 12 MHz 固定时钟频率下产生于 IRC 或产生于 PLL。

分频后的主时钟被称为系统时钟，为内核、存储器和外设（寄存器接口和外设时钟）计时。

1. 选择主时钟。选项如下：
   – IRC：12 MHz 内部振荡器 （默认）。
   – PLL 输出：必须配置 PLL 才可使用 PLL 输出。

   章节 4.6.10 “ 主时钟源选择寄存器 ”
2. 更新主时钟源。

   章节 4.6.11 “ 主时钟源更新使能寄存器 ”
3. 选择系统时钟的分频器值。分频器数值如果为 0 会禁用系统时钟。

   章节 4.6.12 “ 系统时钟分频器寄存器 ”
4. 在应用中选择处于工作状态的存储器和外设，因此必须具有有效时钟。内核始终会被计时。

   章节 4.6.13 “ 系统时钟控制寄存器 ”

4.3.3 使用 XTALIN 和 XTALOUT 设置系统振荡器

要使用 LPC800 系统振荡器，则需通过开关矩阵中的固定引脚功能分配与外部晶振连接的 XTALIN 和 XTALOUT 引脚。 XTALIN 和 XTALOUT 只能分配给引脚 PIO0_8 和 PIO0_9。

1. 在IOCON模块中，针对引脚PIO0_8和PIO0_9移除IOCON寄存器中的上拉/下拉电阻。
2. 在开关矩阵模块中，针对 XTALIN 和 XTALOUT 使能 1 位功能。
3. 在 SYSOSCCTRL 寄存器中，根据所需的振荡器输出时钟禁用 BYPASS 位和选择振荡器频率范围。

相关寄存器：

表 63 “PIO0_8 寄存器 （PIO0_8，地址 0x4004 4038）位说明 ”

表 62 “PIO0_9 寄存器 （PIO0_9，地址 0x4004 4034）位说明 ”

表 106 “ 引脚使能寄存器 0 （PINENABLE0，地址 0x4000 C1C0）位说明 ”

表 10 “ 系统振荡器控制寄存器 （SYSOSCCTRL、地址 0x4004 8020）位说明 ”

4.4 引脚说明

SYSCON 输入和输出通过开关矩阵分配给外部引脚。

有关如何将 CLKOUT 功能分配给 LPC800 封装上的引脚，请参见章节 9.3.1 “将内部信号与封装引脚相连 ”。

有关如何使能时钟输入、振荡器引脚和外部复位输入，请参见章节 9.3.2。

表 4. SYSCON 引脚说明

功能	方向	引脚	说明	SWM 寄存器	参考
CLKOUT	O	任意	CLKOUT 时钟输出。	PINASSIGN8	表 105
CLKIN	I	PIO0_1/ACMP_I2/CLKIN	系统PLL的外部时钟输入。在PINENABLE寄存器中禁用 ACMP_I2 功能。	PINENABLE0	表 106
XTALIN	I	PIO0_8/XTALIN	系统振荡器输入。	PINENABLE0	表 106
XTALOUT	O	PIO0_9/XTALOUT	系统振荡器输出。	PINENABLE0	表 106
$\overline{\text{RESET}}$	I	$\overline{\text{RESET}}$/PIO0_5	外部复位输入	PINENABLE0	表 106

4.5 简介

4.5.1 时钟生成

系统控制模块产生芯片所需的所有时钟信号。只有用于唤醒定时的低功耗振荡器才由 PMU 控制。除 USART 时钟和配置数字 I/O 引脚的干扰滤波器时钟外，内核时钟和外设时钟的工作频率相同。最大系统时钟频率为 30 MHz。参见图 3。

注：主时钟频率被限定在 100 MHz。

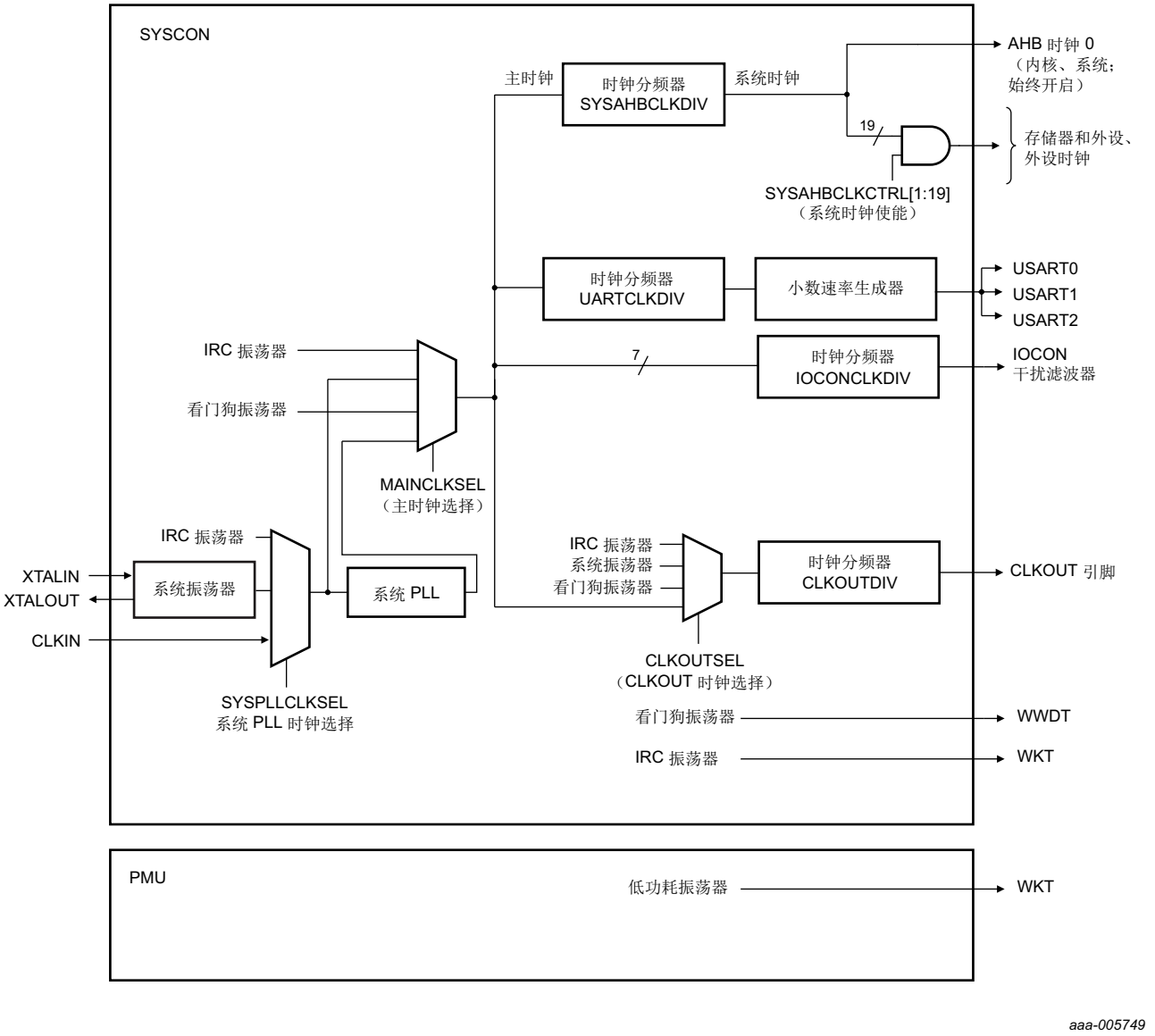


图 3. LPC800 时钟生成

4.5.2 模拟元件的电源控制

系统控制模块可控制振荡器和 PLL、 BOD 以及模拟比较器等模拟元件的电源。有关详情，请参见下列寄存器：

章节 4.6.30 “ 深度睡眠模式配置寄存器 ”

章节 4.6.3 “ 系统 PLL 控制寄存器 ”

章节 4.6.6 “ 看门狗振荡器控制寄存器 ”

章节 4.6.5 “ 系统振荡器控制寄存器 ”

4.5.3 低功耗模式配置

内部时钟在深度睡眠模式和掉电模式下被关断时，系统控制模块可配置在低功耗模式下能保持工作状态的模拟模块（可实现安全工作的 BOD 和看门狗振荡器）和使能各种中断以唤醒芯片。有关详情，请参见下列寄存器：

章节 4.6.32 “ 电源配置寄存器 ”

章节 4.6.29 “ 启动逻辑 1 中断唤醒使能寄存器 ”

4.5.4 复位和中断控制

系统控制寄存器中的外设复位控制寄存器允许置位和解除单个外设复位。参见表 7。

最多可将 8 个外部引脚中断分配给系统控制模块（参见章节 4.6.27 “ 引脚中断选择寄存器 ”）中的任意数字引脚。

4.6 寄存器说明

所有系统控制模块寄存器均位于字地址边界处。寄存器的详细信息都出现在每种功能的说明中。

执行引导加载程序后，复位值可描述寄存器内容。

表 5 中未显示的地址偏移将被保留，并且不应对其进行写操作。

表 5. 寄存器概述：系统配置 （基址 0x4004 8000）

名称	访问类型	偏移	说明	复位值	参考
SYSMEMREMAP	R/W	0x000	系统存储器重映射	0x2	表 6
PRESETCTRL	R/W	0x004	外设复位控制	0x0000 1FFF	表 7
SYSPLLCTRL	R/W	0x008	系统 PLL 控制	0	表 8
SYSPLLSTAT	R	0x00C	系统 PLL 状态	0	表 9
-	-	0x010	保留	-	-
-	-	0x014	保留	-	-
SYSOSCCTRL	R/W	0x020	系统振荡器控制	0x000	表 10
WDTOSCCTRL	R/W	0x024	看门狗振荡器控制	0x0A0	表 11
-	-	0x028	保留	-	-
-	-	0x02C	保留	-	-
SYSRSTSTAT	R/W	0x030	系统复位状态寄存器	0	表 12
SYSPLLCLKSEL	R/W	0x040	系统 PLL 时钟源选择	0	表 13
SYSPLLCLKUEN	R/W	0x044	系统 PLL 时钟源更新使能	0	表 14
MAINCLKSEL	R/W	0x070	主时钟源选择	0	表 15
MAINCLKUEN	R/W	0x074	主时钟源更新使能	0	表 16
SYSAHBCLKDIV	R/W	0x078	系统时钟分频器	1	表 17
SYSAHBCLKCTRL	R/W	0x080	系统时钟控制	0x1F	表 18
UARTCLKDIV	R/W	0x094	USART 时钟分频器	0	表 19
-	-	0x098	保留	-	-
-	-	0x09C	保留	-	-

表 5. 寄存器概述：系统配置 （基址 0x4004 8000）*（续）*

名称	访问类型	偏移	说明	复位值	参考
-	-	0x0A0 - 0x0BC	保留	-	-
-	-	0x0CC	保留	-	-
CLKOUTSEL	R/W	0x0E0	CLKOUT 时钟源选择	0	表 20
CLKOUTUEN	R/W	0x0E4	CLKOUT 时钟源更新使能	0	表 21
CLKOUTDIV	R/W	0x0E8	CLKOUT 时钟分频器	0	表 22
UARTFRGDIV	R/W	0x0F0	USART 小数生成器分频器值	0	表 23
UARTFRGMULT	R/W	0x0F4	USART 小数生成器乘法器值	0	表 24
EXTTRACECMD	R/W	0x0FC	外部跟踪缓冲区命令寄存器	0	表 25
PIOPORCAP0	R	0x100	POR 捕获 PIO 状态 0	由用户决定	表 26
-	-	0x104	保留	-	-
IOCONCLKDIV6	R/W	0x134	可编程干扰滤波器 IOCON 模块的外设时钟 6	0x0000 0000	表 27
IOCONCLKDIV5	R/W	0x138	可编程干扰滤波器 IOCON 模块的外设时钟 5	0x0000 0000	表 27
IOCONCLKDIV4	R/W	0x13C	可编程干扰滤波器 IOCON 模块的外设时钟 4	0x0000 0000	表 27
IOCONCLKDIV3	R/W	0x140	可编程干扰滤波器 IOCON 模块的外设时钟 3	0x0000 0000	表 27
IOCONCLKDIV2	R/W	0x144	可编程干扰滤波器 IOCON 模块的外设时钟 2	0x0000 0000	表 27
IOCONCLKDIV1	R/W	0x148	可编程干扰滤波器 IOCON 模块的外设时钟 1	0x0000 0000	表 27
IOCONCLKDIV0	R/W	0x14C	可编程干扰滤波器 IOCON 模块的外设时钟 0	0x0000 0000	表 27
BODCTRL	R/W	0x150	掉电检测	0	表 28
SYSTCKCAL	R/W	0x154	系统节拍计数器校准	0x0	表 29
-	R/W	0x168	保留	-	-
IRQLATENCY	R/W	0x170	四分位距 (IQR) 延迟。实现中断延迟和确定性之间的平衡。	0x0000 0010	表 30
NMISRC	R/W	0x174	NMI 源控制	0	表 31
PINTSEL0	R/W	0x178	GPIO 引脚中断选择寄存器 0	0	表 32
PINTSEL1	R/W	0x17C	GPIO 引脚中断选择寄存器 1	0	表 32
PINTSEL2	R/W	0x180	GPIO 引脚中断选择寄存器 2	0	表 32
PINTSEL3	R/W	0x184	GPIO 引脚中断选择寄存器 3	0	表 32
PINTSEL4	R/W	0x188	GPIO 引脚中断选择寄存器 4	0	表 32
PINTSEL5	R/W	0x18C	GPIO 引脚中断选择寄存器 5	0	表 32
PINTSEL6	R/W	0x190	GPIO 引脚中断选择寄存器 6	0	表 32
PINTSEL7	R/W	0x194	GPIO 引脚中断选择寄存器 7	0	表 32
STARTERP0	R/W	0x204	启动逻辑 0 引脚唤醒使能寄存器	0	表 33
STARTERP1	R/W	0x214	启动逻辑 1 中断唤醒使能寄存器	0	表 34
PDSLEEPCFG	R/W	0x230	深度睡眠模式掉电状态	0xFFFF	表 35
PDAWAKECFG	R/W	0x234	从深度睡眠模式唤醒的掉电状态	0xEDF0	表 36
PDRUNCFG	R/W	0x238	电源配置寄存器	0xEDF0	表 37
DEVICE_ID	R	0x3F8	器件 ID	取决于零部件	表 38

4.6.1 系统存储器重映射寄存器

系统存储器重映射寄存器选择是否从 Boot ROM、闪存或 SRAM 读取异常向量。默认情况下，闪存映射地址为 0x0000 0000。SYSMEMREMAP 寄存器中的 MAP 位设为 0x0 或 0x1 时，相应的 Boot ROM 或 RAM 映射至存储器映射的最后 512 字节（地址 0x0000 0000 至 0x0000 0200）。

表 6. 系统存储器重映射寄存器（SYSMEMREMAP、地址 0x4004 8000）位说明

位	符号	值	说明	复位值
1:0	MAP		系统存储器重映射。保留值 0x3。	0x2
		0x0	引导加载程序模式。中断向量重映射到引导 ROM。	
		0x1	用户 RAM 模式。中断向量重映射到静态 RAM。	
		0x2	用户闪存模式。中断向量不会被重映射，一直位于闪存中。	
31:2	-	-	保留	-

4.6.2 外设复位控制寄存器

PRESETCTRL 寄存器允许软件复位特定的外设。只要该寄存器中任意分配位为 0，便可复位特定外设。如果为 1 则可清零复位并允许外设运行。

表 7. 外设复位控制寄存器（PRESETCTRL、地址 0x4004 8004）位说明

位	符号	值	说明	复位值
0	SPI0_RST_N		SPI0 复位控制	1
		0	置位 SPI0 复位。	
		1	清零 SPI0 复位。	
1	SPI1_RST_N		SPI1 复位控制	1
		0	置位 SPI1 复位。	
		1	清零 SPI1 复位。	
2	UARTFRG_RST_N		USART 小数波特率生成器 (UARTFRG) 复位控制	1
		0	置位 UARTFRG 复位。	
		1	清零 UARTFRG 复位。	
3	USART0_RST_N		USART0 复位控制	1
		0	置位 USART0 复位。	
		1	清零 USART0 复位。	
4	UART1_RST_N		USART1 复位控制	1
		0	置位 USART 复位。	
		1	清零 USART1 复位。	
5	UART2_RST_N		USART2 复位控制	1
		0	置位 USART2 复位。	
		1	清零 USART2 复位。	
6	I2C_RST_N		I2C 复位控制	1
		0	置位 I2C 复位。	
		1	清零 I2C 复位。	
7	MRT_RST_N		多速率定时器 (MRT) 复位控制	1
		0	置位 MRT 复位。	
		1	清零 MRT 复位。	

表 7. 外设复位控制寄存器 （PRESETCTRL、地址 0x4004 8004）位说明 *（续）*

位	符号	值	说明	复位值
8	SCT_RST_N		SCT 复位控制	1
		0	置位 SCT 复位。	
		1	清零 SCT 复位。	
9	WKT_RST_N		自唤醒定时器 (WKT) 复位控制	1
		0	置位 WKT 复位。	
		1	清零 WKT 复位。	
10	GPIO_RST_N		GPIO 和 GPIO 引脚中断复位控制	1
		0	置位 GPIO 复位。	
		1	清零 GPIO 复位。	
11	FLASH_RST_N		闪存控制器复位控制	1
		0	置位闪存控制器复位。	
		1	清零闪存控制器复位。	
12	ACMP_RST_N		模拟比较器复位控制	1
		0	置位模拟比较器复位。	
		1	清零模拟比较器控制器复位。	
31:12	-	-	保留	-

4.6.3 系统 PLL 控制寄存器

该寄存器用于连接和启用系统 PLL，并配置 PLL 倍频器和分频器的值。 PLL 可从各种时钟源接收 10 MHz 到 25 MHz 的输入频率。将输入频率倍增至较高的频率，再进行分频，以提供 CPU、外设和存储器实际使用的时钟。 PLL 可产生 CPU 允许的最大时钟频率。

注：必须选择 P 和 M 的分频器值，以使 PLL 输出时钟的频率 FCLKOUT 小于 100 MHz。

表 8. 系统 PLL 控制寄存器 （SYSPLLCTRL、地址 0x4004 8008）位说明

位	符号	值	说明	复位值
4:0	MSEL		反馈分频器值。分频值 M 是编程的 MSEL 值 + 1。 00000: 分频比 M = 1 至 11111: 分频比 M = 32	0
6:5	PSEL		后置分频器比率 P。分频比为 2 × P。	0
		0x0	P = 1	
		0x1	P = 2	
		0x2	P = 4	
		0x3	P = 8	
31:7	-	-	保留。不能将 1 写入保留位。	-

4.6.4 系统 PLL 状态寄存器

该寄存器是只读寄存器，提供 PLL 锁定状态 （参见章节 4.7.1.1）。

表 9. 系统 PLL 状态寄存器 （SYSPLLSTAT、地址 0x4004 800C）位说明

位	符号	值	说明	复位值
0	LOCK		PLL 锁定状态	0
		0	PLL 未锁定	
		1	PLL 锁定	
31:1	-	-	保留	-

4.6.5 系统振荡器控制寄存器

该寄存器可配置系统振荡器的频率范围。

表 10. 系统振荡器控制寄存器 （SYSOSCCTRL、地址 0x4004 8020）位说明

位	符号	值	说明	复位值
0	旁通		旁路系统振荡器	0x0
		0	禁用。振荡器未被旁路。	
		1	使能。 PLL 输入 (sys_osc_clk) 不通过振荡器，直接由 XTALIN 引脚馈入。在使用外部时钟源而非晶体振荡器时，使用该模式。	
1	FREQRANGE		可确定低功耗振荡器的频率范围。	0x0
		0	1 - 20 MHz 频率范围。	
		1	15 - 25 MHz 频率范围。	
31:2	-	-	保留	0x00

4.6.6 看门狗振荡器控制寄存器

该寄存器可配置看门狗振荡器。该振荡器包含模拟和数字两个部分。模拟部分含有振荡器功能，可以生成模拟时钟 (Fclkana)。通过数字部分，模拟输出时钟 (Fclkana) 可以分成所需的输出时钟频率 wdt_osc_clk。模拟输出频率 (Fclkana) 可以根据 FREQSEL 位在 600 kHz 到 4.6 MHz 之间调整。通过数字部分，使用 DIVSEL 位将 Fclkana 分成（分频器比率 = 2、4、 ...、 64） wdt_osc_clk。

看门狗振荡器输出时钟频率可通过下式计算：
wdt_osc_clk = Fclkana/(2 × (1 + DIVSEL)) = 9.3 kHz 至 2.3 MHz （标称值）。

注：任何 FREQSEL 位的设置产生的 Fclkana 值都在所列出频率值的 ±40% 范围内。看门狗振荡器是功耗最低的时钟源。如需精确定时，则使用 IRC 或系统振荡器。

注：复位后，看门狗振荡器的频率未定义。使用看门狗振荡器之前，必须对 WDTOSCCTRL 寄存器进行写操作来设定看门狗振荡器频率。

表 11. 看门狗振荡器控制寄存器 （WDTOSCCTRL，地址 0x4004 8024）位说明

位	符号	值	说明	复位值
4:0	DIVSEL		选择 Fclkana 的分频器。 wdt_osc_clk = Fclkana/ (2 × (1 + DIVSEL)) 00000: 2 × (1 + DIVSEL) = 2 00001: 2 × (1 + DIVSEL) = 4 至 11111: 2 × (1 + DIVSEL) = 64	0
8:5	FREQSEL		选择看门狗振荡器模拟输出频率 (Fclkana)。	0x00
		0x1	0.6 MHz	
		0x2	1.05 MHz	
		0x3	1.4 MHz	
		0x4	1.75 MHz	
		0x5	2.1 MHz	
		0x6	2.4 MHz	
		0x7	2.7 MHz	
		0x8	3.0 MHz	
		0x9	3.25 MHz	
		0xA	3.5 MHz	
		0xB	3.75 MHz	
		0xC	4.0 MHz	
		0xD	4.2 MHz	
		0xE	4.4 MHz	
		0xF	4.6 MHz	
31:9	-	-	保留	0x00

4.6.7 系统复位状态寄存器

如果另一个复位信号 —— 例如外部 $\overline{\text{RESET}}$ 引脚 —— 在取消 POR 信号后依然有效，则其位将被设置为 “ 已检测到 ”。写入 1 将清零复位。

表 12 中的复位值适用于 POR 复位。

表 12. 系统复位状态寄存器 （SYSRSTSTAT，地址 0x4004 8030）位说明

位	符号	值	说明	复位值
0	POR		POR 复位状态	0
		0	未检测到 POR	
		1	检测到 POR。写入 1 将清零复位。	
1	EXTRST		外部复位状态。	0
		0	未检测到复位事件。	
		1	检测到复位。写入 1 将清零复位。	
2	WDT		看门狗复位状态	0
		0	未检测到 WDT 复位	
		1	检测到 WDT 复位。写入 1 将清零复位。	
3	BOD		掉电检测复位状态	0
		0	未检测到 BOD 复位	
		1	检测到 BOD 复位。写入 1 将清零复位。	
4	SYSRST		软件系统复位状态	0
		0	未检测到系统复位	
		1	检测到系统复位。写入 1 将清零复位。	
31:5	-	-	保留	-

4.6.8 系统 PLL 时钟源选择寄存器

该寄存器可选择系统 PLL 的时钟源。 SYSPLLCLKUEN 寄存器 （参见章节 4.6.9）必须从低电平转换到高电平，才能使更新生效。

表 13. 系统 PLL 时钟源选择寄存器 （SYSPLLCLKSEL、地址 0x4004 8040）位说明

位	符号	值	说明	复位值
1:0	SEL		系统 PLL 时钟源	0
		0x0	IRC	
		0x1	晶体振荡器 (SYSOSC)	
		0x2	保留。	
		0x3	CLKIN。外部时钟输入。	
31:2	-	-	保留	-

4.6.9 系统 PLL 时钟源更新寄存器

对 SYSPLLCLKSEL 寄存器进行写操作后，该寄存器可使用新输入时钟更新系统 PLL 的时钟源。必须先将 0 写入 SYSPLLUEN 寄存器，再将 1 写入，才能使更新生效。

表 14. 系统 PLL 时钟源更新使能寄存器 （SYSPLLCLKUEN、地址 0x4004 8044）位说明

位	符号	值	说明	复位值
0	ENA		使能系统 PLL 时钟源更新	0
		0	无变化	
		1	更新时钟源	
31:1	-	-	保留	-

4.6.10 主时钟源选择寄存器

该寄存器可选择主系统时钟，所选时钟可以是系统 PLL (sys_pllclkout)、看门狗振荡器或 IRC 振荡器。主系统时钟为内核、外设和存储器计时。

MAINCLKUEN 寄存器 （参见章节 4.6.11）的位 0 必须从 0 转换到 1，才能使更新生效。

表 15. 主时钟源选择寄存器 （MAINCLKSEL、地址 0x4004 8070）位说明

位	符号	值	说明	复位值
1:0	SEL		主时钟的时钟源	0
		0x0	IRC 振荡器	
		0x1	PLL 输入	
		0x2	看门狗振荡器	
		0x3	PLL 输出	
31:2	-	-	保留	-

4.6.11 主时钟源更新使能寄存器

对 MAINCLKSEL 寄存器进行写操作后，该寄存器可使用新输入时钟更新主时钟的时钟源。必须先将 0 写入该寄存器，再将 1 写入，才能使更新生效。

表 16. 主时钟源更新使能寄存器 （MAINCLKUEN、地址 0x4004 8074）位说明

位	符号	值	说明	复位值
0	ENA		使能主时钟源更新	0
		0	无变化	
		1	更新时钟源	
31:1	-	-	保留	-

4.6.12 系统时钟分频器寄存器

该寄存器可控制主时钟如何进行分频，以向内核、存储器和外设提供系统时钟。系统时钟可通过将 DIV 字段设置为 0 而完全关断。

表 17. 系统时钟分频器寄存器 （SYSAHBCLKDIV，地址 0x4004 8078）位说明

位	符号	说明	复位值
7:0	DIV	系统 AHB 时钟分频器值 0: 系统时钟禁用。 1:1 分频 至 255:255 分频	0x01
31:8	-	保留	-

4.6.13 系统时钟控制寄存器

SYSAHBCLKCTRL 存器可启用单个系统和外设模块的时钟。系统时钟 （位 0）为 AHB、APB 桥、 ARM Cortex-M0+、 SYSCON 模块和 PMU 提供时钟。无法禁用该时钟。

表 18. 系统时钟控制寄存器 （SYSAHBCLKCTRL，地址 0x4004 8080）位说明

位	符号	值	说明	复位值
0	SYS		使能 AHB、APB 桥、Cortex-M0+ 内核时钟、SYSCON 和 PMU 时钟。该位为只读，且始终以 1 读取。	1
		0	保留	
		1	使能	
1	ROM		使能 ROM 的时钟。	1
		0	禁用	
		1	使能	
2	RAM		使能 SRAM 的时钟。	1
		0	禁用	
		1	使能	
3	FLASHREG		使能闪存寄存器接口的时钟。	1
		0	禁用	
		1	使能	
4	闪存		使能闪存时钟。	1
		0	禁用	
		1	使能	
5	I2C		使能 I2C 时钟。	0
		0	禁用	
		1	使能	
6	GPIO		使能 GPIO 端口寄存器时钟和 GPIO 引脚中断寄存器时钟。	0
		0	禁用	
		1	使能	
7	SWM		使能开关矩阵时钟。	0
		0	禁用	
		1	使能	
8	SCT		使能状态配置定时器时钟。	0
		0	禁用	
		1	使能	

表 18. 系统时钟控制寄存器 （SYSAHBCLKCTRL，地址 0x4004 8080）位说明 *（续）*

位	符号	值	说明	复位值
9	WKT		使能自唤醒定时器时钟。	0
		0	禁用	
		1	使能	
10	MRT		使能多速率定时器时钟。	
		0	禁用	
		1	使能	
11	SPI0		使能 SPI0 时钟。	0
		0	禁用	
		1	使能	
12	SPI1		使能 SPI1 时钟。	
		0	禁用	
		1	使能	
13	CRC		使能 CRC 时钟。	0
		0	禁用	
		1	使能	
14	UART0		使能 USART0 时钟。	0
		0	禁用	
		1	使能	
15	UART1		使能 USART1 时钟。	0
		0	禁用	
		1	使能	
16	UART2		使能 USART2 时钟。	0
		0	禁用	
		1	使能	
17	WWDT		使能 WWDT 时钟。	0
		0	禁用	
		1	使能	
18	IOCON		使能 IOCON 模块时钟。	0
		0	禁用	
		1	使能	
19	ACMP		使能模拟比较器时钟。	0
		0	禁用	
		1	使能	
31:20	-	-	保留	-

4.6.14 USART 时钟分频器寄存器

该寄存器配置小数波特率生成器时钟和所有 USART 时钟。UART 时钟可通过将 DIV 字段设置为 0 而禁用 （默认设置）。

表 19. USART 时钟分频器寄存器 （UARTCLKDIV，地址 0x4004 8094）位说明

位	符号	说明	复位值
7:0	DIV	USART 小数波特率生成器时钟分频器值。 0: 时钟禁用。 1:1 分频。 至 255:255 分频。	0
31:8	-	保留	-

4.6.15 CLKOUT 时钟源选择寄存器

该寄存器可选择 CLKOUT 引脚上的信号。可选择任何振荡器或主时钟。

CLKOUTUEN 寄存器的位 0 （见章节 4.6.16）必须从 0 转换到 1，才能使更新生效。

表 20. CLKOUT 时钟源选择寄存器 （CKLOUTSEL，地址 0x4004 80E0）位说明

位	符号	值	说明	复位值
1:0	SEL		CLKOUT 时钟源	0
		0x0	IRC 振荡器	
		0x1	晶体振荡器 (SYSOSC)	
		0x2	看门狗振荡器	
		0x3	主时钟	
31:2	-	-	保留	0

4.6.16 CLKOUT 时钟源更新使能寄存器

对 CLKOUTSEL 寄存器进行写操作后，该寄存器可使用新时钟更新 CLKOUT 引脚的时钟源。必须先将 0 写入该寄存器，再将 1 写入，才能使 CLKOUT 引脚输入的更新生效。

表 21. CLKOUT 时钟源更新使能寄存器 （CLKOUTUEN、地址 0x4004 80E4）位说明

位	符号	值	说明	复位值
0	ENA		使能 CLKOUT 时钟源更新	0
		0	无变化	
		1	更新时钟源	
31:1	-	-	保留	-

4.6.17 CLKOUT 时钟分频寄存器

该寄存器可确定 CLKOUT 引脚信号的分频器值。

表 22. CLKOUT 时钟分频寄存器 （CLKOUTDIV、地址 0x4004 80E8）位说明

位	符号	说明	复位值
7:0	DIV	CLKOUT 时钟分频器值。 0: 禁用 CLKOUT 时钟分频器。 1:1 分频。 至 255:255 分频。	0
31:8	-	保留	-

4.6.18 USART 小数生成分频器值寄存器

所有 USART 外设共享同一个时钟 U_PCLK，该时钟可通过小数分频器进行调整：

U_PCLK = UARTCLKDIV/(1 + MULT/DIV)。

UARTCLKDIV 是在 UARTCLKDIV 寄存器中配置的 USART 时钟。

小数部分 (1 + MULT/DIV) 由 SYSCON 模块中的两个 USART 小数分频器寄存器确定：

1. 该寄存器中设置的 DIV 值是分频器的分母，小数速率生成器使用它产生 U_PCLK 的小数部分。
2. 小数分频器的 MULT 值在 UARTFRGMULT 寄存器中设定。参见表 24。

注：要使用小数波特率生成器，必须将 0xFF 写入该寄存器，以产生分母 256。不支持其他任何值。

另请参见：

章节 15.3.1 “ 配置 USART 时钟和波特率 ”

章节 15.7.1 “ 时钟和波特率 ”

表 23. USART 小数生成器分频器值寄存器 （UARTFRGDIV，地址 0x4004 80F0）位说明

位	符号	说明	复位值
7:0	DIV	小数分频器分母。 DIV 等于设定值 +1。要使用小数波特率生成器，应当始终设为 0xFF。	0
31:8	-	保留	-

4.6.19 USART 小数生成器乘法器值寄存器

所有 USART 外设共享同一个时钟 U_PCLK，该时钟可通过小数分频器进行调整：

U_PCLK = UARTCLKDIV/(1 + MULT/DIV)。

UARTCLKDIV 是在 UARTCLKDIV 寄存器中配置的 USART 时钟。

小数部分 (1 + MULT/DIV) 由 SYSCON 模块中的两个 USART 小数分频器寄存器确定：

1. 小数分频器值的 DIV 分母在 UARTFRGDIV 寄存器中设定。参见表 23。
2. 该寄存器中设定的 MULT 值是小数分频器值的分子，小数速率生成器用它产生波特率的小数部分。

另请参见：

章节 15.3.1 “ 配置 USART 时钟和波特率 ”

章节 15.7.1 “ 时钟和波特率 ”

表 24. USART 小数生成器乘法器值寄存器 （UARTFRGMULT，地址 0x4004 80F4）位说明

位	符号	说明	复位值
7:0	MULT	小数分频器分子。 MULT 等于设定值。	0
31:8	-	保留	-

4.6.20 外部跟踪缓冲区命令寄存器

该寄存器与 MTB 主机寄存器协同工作，可启动或停止跟踪。另请参见章节 26.5.4。

表 25. 外部跟踪缓冲区命令寄存器 （EXTTRACECMD，地址 0x4004 80FC）位说明

位	符号	说明	复位值
0	启动	跟踪开始命令。如果 MTB 主机寄存器中的 TSTARTEN 位同样也设为 1，则将 1 写入该位可将 MTB 的信号 TSTART 设为高电平并开始跟踪。	0
1	STOP	跟踪停止命令。如果 MTB 主机寄存器中的 TSTOPEN 位同样也设为 1，则将 1 写入该位可将 MTB 的信号 TSTOP 设为高电平并开始跟踪。	0
31:2	-	保留	0

4.6.21 POR 捕获 PIO 状态寄存器 0

PIOPORCAP0 寄存器在上电复位时可捕获 GPIO 端口 0 的状态。每个位代表一个 GPIO 引脚的复位状态。该寄存器是只读状态寄存器。

表 26. POR 捕获 PIO 状态寄存器 0 （PIOPORCAP0，地址 0x4004 8100）位说明

位	符号	说明	复位值
17:0	PIOSTAT	上电复位时的 PIO0_17 至 PIO0_0 状态	取决于实施
31:18	-	保留。	-

4.6.22 IOCON 干扰滤波器时钟分频器寄存器 6 到 0

这些寄存器可分别将 7 个外设输入时钟 (IOCONFILTR_PCLK) 配置到 IOCON 可编程干扰滤波器。时钟可通过将 DIV 位设置为 0x0 而关断。

表 27. IOCON 干扰滤波器时钟分频器寄存器 6 到 0 （IOCONCLKDIV[6:0]，地址 0x4004 8134 (IOCONCLKDIV6) 至 0x004 814C (IOCONFILTCLKDIV0)）位说明

位	符号	说明	复位值
7:0	DIV	IOCON 干扰滤波器时钟分频器值 0: 禁用 IOCONFILTR_PCLK。 1:1 分频 至 255:255 分频	0
31:8	-	保留	0x00

4.6.23 BOD 控制寄存器

BOD 控制寄存器可选择 4 个不同阈值，用于向 NVIC 发送 BOD 中断和强制复位。表 28 中所列的复位和中断阈值都是典型值。

BOD 中断和 BOD 复位均可将芯片从睡眠、深度睡眠和掉电模式中唤醒，具体取决于该寄存器中 BODRSTENA 位的值。

有关 BOD 复位和中断电平，请参见 LPC800 数据手册。

表 28. BOD 控制寄存器 （BODCTRL、地址 0x4004 8150）位说明

位	符号	值	说明	复位值
1:0	BODRSTLEV		BOD 复位电平	0
		0x0	保留。	
		0x1	1 级。	
		0x2	2 级。	
		0x3	3 级。	
3:2	BODINTVAL		BOD 中断电平	0
		0x0	保留	
		0x1	1 级。	
		0x2	2 级。	
		0x3	3 级。	
4	BODRSTENA		BOD 复位使能	0
		0	禁用复位功能。	
		1	使能复位功能。	
31:5	-	-	保留	0x00

4.6.24 系统节拍计数器校准寄存器

该寄存器可确定 SYST_CALIB 寄存器的值。

表 29. 系统节拍定时器校准寄存器 （SYSTCKCAL、地址 0x4004 8154）位说明

位	符号	说明	复位值
25:0	CAL	系统节拍定时器校准值	0
31:26	-	保留	-

4.6.25 中断请求 (IRQ) 延迟寄存器

IRQLATENCY 寄存器是一个 8 位寄存器，用于指定系统响应中断请求的最小允许周期数 (0-255)。该寄存器旨在使用户能在中断响应时间与确定性之间选择一个平衡点。

将该参数设置为一个非常小的值 （例如 0）可保证最佳中断性能，但同时也会带来非常明显的不确定性和抖动。要求系统始终使用较大的周期数（无论是否需要）将降低不确定量，但不一定会消除。

从理论上来讲，ARM Cortex-M0+ 内核应始终能够在 15 个周期内处理中断请求。但是，在处理中断之前， CPU 外部系统因素、总线延迟、外设响应时间等会增加完成上一个指令所需的时间。因此，精确地指定可保证确定性的最小周期数将取决于应用程序。

该寄存器的默认设置为 0x010。

表 30. IRQ 延迟寄存器 （IRQLATENCY、地址 0x4004 8170）位说明

位	符号	说明	复位值
7:0	LATENCY	8 位延迟值	0x010
31:8	-	保留	-

4.6.26 NMI 源选择寄存器

NMI 源选择寄存器可选择一个外设中断作为 ARM Cortex-M0+ 内核 NMI 中断的源。有关全部外设中断及其 IRQ 编号的列表，请参见表 3。有关 NMI 功能说明，请参见章节 3.3.2。

注：如需改变 NMI 的中断源，必须首先将该寄存器中的位 31 设置为 0 来禁用 NMI 源。然后，更新 IRQN 位来改变源以及将位 31 设置为 1 来重新使能 NMI 源。

表 31. NMI 源选择寄存器 （NMISRC，地址 0x4004 8174）位说明

位	符号	说明	复位值
4:0	IRQN	如果位 31 为 1，则中断的 IRQ 编号用作非屏蔽中断 (NMI)。有关中断源及其 IRQ 编号的列表，请参见表 3。	0
30:5	-	保留	-
31	NMIEN	将 1 写入该位可使能由位 4:0 选中的非屏蔽中断 (NMI) 源。	0

注：如果 NMISRC 寄存器用于选择作为非屏蔽中断源的某个中断并且所选中断使能，则单个中断请求可同时产生非屏蔽中断和普通中断。该情况可通过禁用 NVIC 中的普通中断来避免。

4.6.27 引脚中断选择寄存器

所有 8 个寄存器都从全部数字引脚中选出一个引脚，作为引脚中断源或作为模式匹配引擎的输入。要为 8 个引脚中断或模式匹配引擎输入中的任意一个选择引脚，则将引脚 PIO0_0 至 PIO0_17 的 GPIO 端口引脚编号 0 至 17 分别写入 INTPIN 位。例如，在 PINTSEL0 中将 INTPIN 设为 0x5 便可为引脚中断 0 选择引脚 PIO0_5。

要确定给定 LPC800 封装上 GPIO 端口引脚的编号，请参见数据手册中的引脚说明表。

注：GPIO 端口引脚编号用于检验连接 PINTSEL 寄存器的引脚。包括 GPIO 在内的所有数字输入功能都可通过开关矩阵分配给该引脚。

所有 8 个引脚中断都必须在 NVIC 中使用 24 至 31 号插槽使能 （参见表 3）。

要使用引脚中断或模式匹配引擎使用的选定引脚，请参见章节 8.5.2 “ 模式匹配引擎 ”。

表 32. 引脚中断选择寄存器 （PINTSEL[0:7]，地址 0x4004 8178 (PINTSEL0) 至 0x4004 8194 (PINTSEL7)）位说明

位	符号	说明	复位值
5:0	INTPIN	针对引脚中断或模式匹配引擎输入的引脚编号选择。（PIO0_0 至 PIO0_17 对应编号 0 至 17）。	0
31:6	-	保留	-

4.6.28 启动逻辑 0 引脚唤醒使能寄存器

STARTERP0 寄存器可使能选定的引脚中断，以从唤醒深度睡眠模式和掉电模式中唤醒。

注：还可在 NVIC 中使能相应中断。参见表 3 “ 中断源到 NVIC 的连接 ”。

表 33. 启动逻辑 0 引脚唤醒使能寄存器 0 （STARTERP0，地址 0x4004 8204）位说明

位	符号	值	说明	复位值
0	PINT0		GPIO 引脚中断 0 唤醒	0
		0	禁用	
		1	使能	

表 33. 启动逻辑 0 引脚唤醒使能寄存器 0 （STARTERP0，地址 0x4004 8204）位说明 *（续）*

位	符号	值	说明	复位值
1	PINT1		GPIO 引脚中断 1 唤醒	0
		0	禁用	
		1	使能	
2	PINT2		GPIO 引脚中断 2 唤醒	0
		0	禁用	
		1	使能	
3	PINT3		GPIO 引脚中断 3 唤醒	0
		0	禁用	
		1	使能	
4	PINT4		GPIO 引脚中断 4 唤醒	0
		0	禁用	
		1	使能	
5	PINT5		GPIO 引脚中断 5 唤醒	0
		0	禁用	
		1	使能	
6	PINT6		GPIO 引脚中断 6 唤醒	0
		0	禁用	
		1	使能	
7	PINT7		GPIO 引脚中断 7 唤醒	0
		0	禁用	
		1	使能	
31:8	-		保留	-

4.6.29 启动逻辑 1 中断唤醒使能寄存器

该寄存器选择哪个中断可将 LPC800 从深度睡眠和掉电模式中唤醒。

注：还可在 NVIC 中使能相应中断。参见表 3 “ 中断源到 NVIC 的连接 ”。

表 34. 启动逻辑 1 中断唤醒使能寄存器 （STARTERP1，地址 0x4004 8214）位说明

位	符号	值	说明	复位值
0	SPI0		SPI0 中断唤醒	0
		0	禁用	
		1	使能	
1	SPI1		SPI1 中断唤醒	0
		0	禁用	
		1	使能	
2	-		保留	-
3	USART0		USART0 中断唤醒。在同步从机模式下配置 USART。	0
		0	禁用	
		1	使能	

表 34. 启动逻辑 1 中断唤醒使能寄存器 （STARTERP1，地址 0x4004 8214）位说明 *（续）*

位	符号	值	说明	复位值
4	USART1		USART1 中断唤醒。在同步从机模式下配置 USART。	0
		0	禁用	
		1	使能	
5	USART2		USART2 中断唤醒。在同步从机模式下配置 USART。	0
		0	禁用	
		1	使能	
7:6	-		保留	-
8	I2C		I2C 中断唤醒。	0
		0	禁用	
		1	使能	
11:9	-		保留	-
12	WWDT		WWDT 中断唤醒	0
		0	禁用	
		1	使能	
13	BOD		BOD 中断唤醒	0
		0	禁用	
		1	使能	
14	-		保留	-
15	WKT		自唤醒定时器中断唤醒	0
		0	禁用	
		1	使能	
31:16			保留。	-

4.6.30 深度睡眠模式配置寄存器

该寄存器中的位 （BOD_PD 和 WDTOSC_OD）经过编程后可控制深度睡眠和掉电模式的各个方面。进入深度睡眠模式或掉电模式时，位将被载入 PDRUNCFG 寄存器中的相应位。

注：深度睡眠模式或掉电模式下，硬件可使模拟模块强制掉电。 BOD 和看门狗振荡器存在例外情况，它们通过寄存器配置后可保持运行状态。如果 WWDT MOD 寄存器 （参见表 143）中的 LOCK 位被设置，则写入 PDSLEEPCFG 寄存器的 WDTOSC_PD 值会被覆盖。有关详情，请参见章节 12.5.3。

表 35. 深度睡眠配置寄存器 （PDSLEEPCFG、地址 0x4004 8230）位说明

位	符号	值	说明	复位值
2:0			保留。	0b111
3	BOD_PD		深度睡眠和掉电模式的 BOD 掉电控制	1
		0	上电	
		1	掉电	
5:4			保留。	11

表 35. 深度睡眠配置寄存器 （PDSLEEPCFG、地址 0x4004 8230）位说明 *（续）*

位	符号	值	说明	复位值
6	WDTOSC_PD		深度睡眠和掉电模式的看门狗振荡器掉电控制 WWDT MOD 寄存器中的 LOCK 位被设置时，使该位掉电将无任何作用。在这种情况下，看门狗振荡器始终处于运行状态。	1
		0	上电	
		1	掉电	
15:7	-		保留	0b111 111111
31:16	-	-	保留	0

4.6.31 唤醒配置寄存器

该寄存器从深度睡眠模式或掉电模式唤醒时可控制器件的电源配置。

表 36. 唤醒配置寄存器 （PDAWAKECFG、地址 0x4004 8234）位说明

位	符号	值	说明	复位值
0	IRCOUT_PD		IRC 振荡器输出唤醒配置	0
		0	上电	
		1	掉电	
1	IRC_PD		IRC 振荡器掉电唤醒配置	0
		0	上电	
		1	掉电	
2	FLASH_PD		闪存唤醒配置	0
		0	上电	
		1	掉电	
3	BOD_PD		BOD 唤醒配置	0
		0	上电	
		1	掉电	
4	-		保留。	1
5	SYSOSC_PD		晶体振荡器唤醒配置	1
		0	上电	
		1	掉电	
6	WDTOSC_PD		看门狗振荡器唤醒配置。WWDT MOD 寄存器中的 LOCK 位被设置时，使该位掉电将无任何作用。在这种情况下，看门狗振荡器始终处于运行状态。	1
		0	上电	
		1	掉电	
7	SYSPLL_PD		系统 PLL 唤醒配置	1
		0	上电	
		1	掉电	
11:8	-		保留。始终将这些位写为 0b1101。	0b1101
14:12	-		保留。始终将这些位写为 0b110	0b110

表 36. 唤醒配置寄存器 （PDAWAKECFG、地址 0x4004 8234）位说明 *（续）*

位	符号	值	说明	复位值
15	ACMP		模拟比较器唤醒配置	1
		0	上电	
		1	掉电	
31:16	-	-	保留	0

4.6.32 电源配置寄存器

PDRUNCFG寄存器控制各种模拟模块的电源。芯片运行时的任何时刻都可对该寄存器进行写操作，而且该写操作将会立即生效，但 IRC 的掉电信号除外。

为避免在 IRC 掉电时产生干扰， IRC 时钟会在无干扰时自动关断。因此，对于 IRC 来说，在掉电状态生效之前可能会延时。

表 37. 电源配置寄存器 （PDRUNCFG、地址 0x4004 8238）位说明

位	符号	值	说明	复位值
0	IRCOUT_PD		IRC 振荡器输出电源	0
		0	上电	
		1	掉电	
1	IRC_PD		IRC 振荡器掉电	0
		0	上电	
		1	掉电	
2	FLASH_PD		闪存掉电	0
		0	上电	
		1	掉电	
3	BOD_PD		BOD 掉电	0
		0	上电	
		1	掉电	
4	-		保留。	1
5	SYSOSC_PD		晶体振荡器掉电	1
		0	上电	
		1	掉电	
6	WDTOSC_PD		看门狗振荡器掉电。 WWDT MOD 寄存器中的 LOCK 位被设置时，使该位掉电将无任何作用。在这种情况下，看门狗振荡器始终处于运行状态。	1
		0	上电	
		1	掉电	
7	SYSPLL_PD		系统 PLL 掉电	1
		0	上电	
		1	掉电	
11:8	-		保留。始终将这些位写为 0b1101。	0b1101
14:12	-		保留。始终将这些位写为 0b110	0b110

表 37. 电源配置寄存器 （PDRUNCFG、地址 0x4004 8238）位说明 *（续）*

位	符号	值	说明	复位值
15	ACMP		模拟比较器掉电	1
		0	上电	
		1	掉电	
31:16	-	-	保留	0

4.6.33 器件 ID 寄存器

器件 ID 寄存器为只读寄存器，包含每个 LPC800 的器件 ID。该寄存器也可通过 ISP/IAP 命令进行读操作 （参见表 235）。

表 38. 器件 ID 寄存器 （DEVICE_ID，地址 0x4004 83F8）位说明

位	符号	说明	复位值
31:0	DEVICEID	0x0000 8100 = LPC810M021FN8 0x0000 8110 = LPC811M001FDH16 0x0000 8120 = LPC812M101FDH16 0x0000 8121 = LPC812M101FD20 0x0000 8122 = LPC812M101FDH20	取决于零部件

4.7 功能说明

4.7.1 系统 PLL 的功能说明

LPC800 利用系统 PLL 为内核和外设创建时钟。

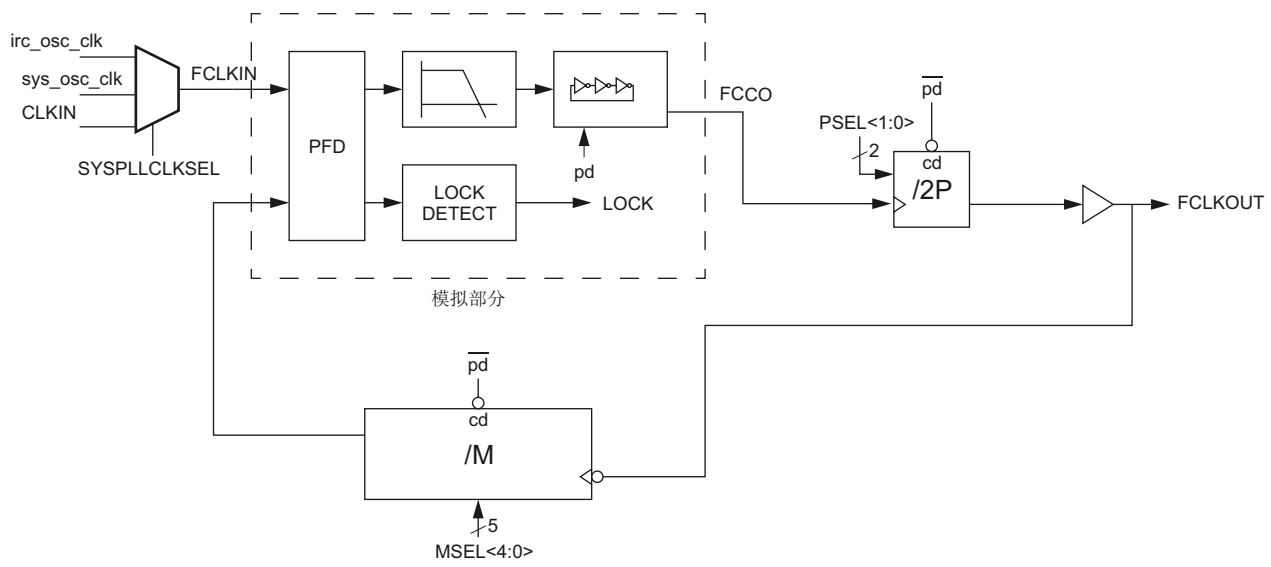


图 4. 系统 PLL 功能框图

该 PLL 的功能框图如图 4 所示。输入频率的范围从 10 MHz 到 25 MHz。输入时钟直接馈入相位频率检测器 (PFD)。该模块将比较其输入的相位和频率，如果它们不匹配，将生成控制信号。环形滤波器可过滤这些控制信号并驱动电流控制振荡器 (CCO)，从而生成主时钟以

及可选的两个额外相位。 CCO 频率范围为 156 MHz 至 320 MHz。这些时钟通过可编程后置分频器按 2 × P 分频以创建输出时钟，或者直接发送到输出。然后，主输出时钟由可编程反馈分频器进行 M 分频，以生成反馈时钟。相位频率检测器的输出信号也可由锁定检测器监控，在 PLL 已锁定到输入时钟时发出信号。

注：由于主时钟最大频率限值为 100 MHz，因此必须选择 P 和 M 的分频器值，以使 PLL 输出时钟的频率 FCLKOUT 小于 100 MHz。

4.7.1.1 锁定检测器

锁定检测器可测量输入时钟上升沿与反馈时钟上升沿之间的相位差。仅当该相位差小于八个以上连续输入时钟期间所谓的“锁定标准”时，锁定输出才会从低电平切换到高电平。单个相位差过大会立即复位计数器并导致锁定信号下降（如果原来为高电平信号）。每行要求有八个相位测量值低于某个特定的数，以确保锁定检测器在输入时钟和反馈时钟两者的相位和频率极其相符时，才会指示锁定。这样就能有效防止虚假的锁定指示，从而确保锁定信号不会出错。

4.7.1.2 掉电控制

为降低无需 PLL 时钟时的功耗，引入了 PLL 掉电模式。该模式可通过在掉电配置寄存器中将 SYSPLL_PD 位设置为 1 来使能 (表 37)。在该模式下，内部参考电流将被关断，振荡器和相位频率检测器会停止，分频器将进入复位状态。 PLL 掉电模式期间，锁定输出将变为低电平，从而表明 PLL 未被锁定。如果 PLL 掉电模式通过将 SYSPLL_PD 位设置为 0 而被终止，一旦 PLL 在输入时钟上被重新锁定，则 PLL 将恢复其正常工作并将锁定信号变为高电平。

4.7.1.3 分频器比率设定

4.7.1.3.1 后置分频器

后置分频器的分频比由 PSEL 位控制。分频率为 PSEL 位选择的 P 值的两倍，如表 8 所示。这样可保证输出时钟达到 50% 的占空比。

4.7.1.3.2 反馈分频器

反馈分频器的分频比由 MSEL 位控制。如表 8 中所规定的那样， PLL 输出时钟和输入时钟之间的分频率等于 MSEL 位十进制数值加 1。

4.7.1.3.3 更改分频器值

不建议在 PLL 运行时更改分频比。由于无法使 MSEL 和 PSEL 值的变更与分频器同步，因此计数器可能会读取未定义的值，从而造成输出时钟频率不必要地达到峰值或下降。要在分频器之间更改设置，建议使 PLL 掉电，调整分频器设置，然后再次启动 PLL。

4.7.1.4 频率选择

PLL 频率的计算公式使用下列参数 （另请参见图 4）：

表 39. PLL 频率参数

参数	系统 PLL
FCLKIN	SYSPLLCLKSEL 多路复用器中 sys_pllclkin （系统 PLL 的输入时钟）的频率（参见章节 4.6.8）。
FCCO	电流控制振荡器 (CCO) 的频率； 156 到 320 MHz。

表 39. PLL 频率参数 *（续）*

参数	系统 PLL
FCLKOUT	sys_pllclkout 的频率。这是 PLL 输出频率，并且必须小于 100 MHz。
P	系统 PLL 的后置分频器比率； SYSPLLCTRL（参见章节 4.6.3）中的 PSEL 位。
M	系统 PLL 的反馈分频寄存器； SYSPLLCTRL（参见章节 4.6.3）中的 MSEL 位。

4.7.1.4.1 正常模式

在该模式下后置分频器使能，占空比时钟为 50%，频率关系如下：

$$Fclkout = M \times Fclkin = (FCCO)/(2 \times P) \quad (1)$$

要选择合适的 M 值和 P 值，建议执行以下步骤：

1. 指定输入时钟频率 Fclkin。
2. 计算 M 值以获得所需的输出频率 Fclkout， $M = F_{clkout} / F_{clkin}$。
3. 找出一个值，使 $FCCO = 2 \times P \times F_{clkout}$。
4. 检查所有的频率和分频器值是否符合表 8 中指定的限值。

注：必须选择 P 和 M 的分频器值，以使 PLL 输出时钟的频率 FCLKOUT 小于 100 MHz。

表 40 显示的是如何使用 SYSPLLCTRL 寄存器（表 8）将 PLL 配置为 12 MHz 晶体振荡器。如果系统时钟分频器 SYSAHBCLKDIV 被设置为 1，则主时钟等于系统时钟（参见表 17）。

表 40. PLL 配置示例

PLL 输入时钟 sys_pllclkin (Fclkin)	主时钟 (Fclkout)	MSEL 位 表 8	M 分频器值	PSEL 位 表 8	P 分频器值	FCCO 频率	SYSAHBCLKDIV	系统时钟
12 MHz	60 MHz	00100（二进制）	5	01（二进制）	2	240 MHz	2	30 MHz
12 MHz	24 MHz	00001（二进制）	2	10（二进制）	4	192 MHz	1	24 MHz

4.7.1.4.2 PLL 掉电模式

在该模式下，内部参考电流将被关断，振荡器和相位频率检测器会停止，分频器将进入复位状态。PLL 掉电模式期间，锁定输出将变为低电平，从而表明 PLL 未被锁定。如果 PLL 掉电模式通过在掉电配置寄存器（表 37）中将 SYSPLL_PD 位设置为 0 而被终止，那么 PLL 一旦在输入时钟上被重新锁定，它就将恢复其正常工作并将锁定信号变为高电平。

5.1 本章导读

LPC800 在引导 ROM 内集成片内 API，可优化工作模式和睡眠模式下的功耗。参见表 255 “电源配置 API 调用 ”。

阅读本章，了解如何配置节能模式 （深度睡眠模式、掉电模式和深度掉电模式）。

5.2 特性

- 节能模式控制：
- 低功耗振荡器控制
- 五个通用备份寄存器可在深度掉电模式下保存数据

5.3 基本配置

只要 V_{DD} 有效， PMU 模块就处于工作状态。

在低功耗模式下，若开漏引脚 PIO0_10 和 PIO0_11 无引脚输出，则必须使能它们的输出驱动器，并在内部将输出驱动为低电平，以便在该模式下尽可能降低功耗。参见章节 6.3。

5.3.1 ARM Cortex-M0+ 内核的低功耗模式

进入与退出低功耗模式始终受 ARM Cortex-M0+ 内核的控制。进入低功耗模式时， SCR 寄存器用作控制内核操作的软件接口。SCR 寄存器位于 ARM 的专用外设总线上。更多详情，请参见参考 1。

5.3.1.1 系统控制寄存器

系统控制寄存器 (SCR) 控制低功耗状态的进出。该寄存器为 R/W 寄存器，复位值为 0x0000 0000。SCR 寄存器允许 ARM 内核进入睡眠模式，或整个系统进入深度睡眠 / 掉电模式。若要设置低功耗状态 (SLEEPDEEP = 1) 以进入深度睡眠模式或掉电模式，或者进入深度掉电模式，可使用 PCON 寄存器 （表 44）。

表 41. 系统控制寄存器 （SCR，地址 0xE000 ED10）位说明

位	符号	说明	复位值
0	-	保留。	0
1	SLEEPONEXIT	指示当从处理程序模式返回到线程模式时 sleep-on-exit（退出时进入睡眠）： 0 = 处理器返回到线程模式时不进入睡眠。 1 = 处理器从 ISR 返回到线程模式时进入睡眠或深度睡眠。 将该位设为 1 允许一个中断驱动的应用程序避免返回到一个空的主应用程序。	0
2	SLEEPDEEP	控制处理器是将睡眠模式还是深度睡眠模式作为低功耗模式： 0 = 睡眠。 1 = 深度睡眠。	0

表 41. 系统控制寄存器（SCR，地址 0xE000 ED10）位说明 *（续）*

位	符号	说明	复位值
3	-	保留。	0
4	SEVONPEND	挂起时发送事件位： 0 = 只有使能的中断或事件能够唤醒处理器，禁用的中断不包括在内。 1 = 使能的事件和包括禁用中断在内的所有中断都能唤醒处理器。 当一个事件或中断进入挂起状态时，事件信号将处理器从WFE唤醒。如果处理器并未在等待一个事件，该事件会被寄存并影响下一个 WFE。 处理器也可以在执行 SEV 指令时唤醒。	0
31:5	-	保留。	0

5.4 引脚说明

在深度掉电模式下，仅 WAKEUP 引脚（引脚 PIO0_4）可用。可在 DPDCTRL 寄存器中禁用 WAKEUP 功能，进一步降低功耗。这种情况下，可使能自唤醒定时器，提供内部唤醒信号。参见章节 5.6.3 “ 深度掉电控制寄存器 ”。

注：进入深度掉电模式后，需要在 WAKEUP 引脚上使用一个外部上拉电阻，以使该引脚保持在高电平。此外，在深度掉电模式下，可拉高 RESET 引脚以防止浮空。

5.5 简介

LPC800 的上电由 PMU、 SYSCON 模块和 ARM Cortex-M0+ 内核控制。支持下列低功耗模式 （按功耗从高到低排列）：

1. 睡眠模式：

   睡眠模式仅影响 ARM Cortex-M0+ 内核。外设和存储器处于工作状态。

2. 深度睡眠和掉电模式：

   深度睡眠和掉电模式影响内核和整个系统，包括存储器和外设。

   a. 在深度睡眠模式下，外设不接收内部时钟信号。闪存处于待机模式。 SRAM 存储器和所有外设寄存器以及处理器都保持在内部状态。WWDT、WKT 和 BOD 可保持工作状态以便在中断时唤醒系统。

   b. 掉电模式下，外设不接收内部时钟信号。内部 SRAM 存储器和所有外设寄存器以及处理器都保持在内部状态。闪存掉电。WWDT、WKT 和 BOD 可保持工作状态以便在中断时唤醒系统。

3. 深度掉电模式：

   要最大程度省电，整个系统除 PMU 中的通用寄存器以及自唤醒定时器外都处于掉电状态。仅 PMU 中的通用寄存器保持内部状态。可利用 WAKEUP 引脚上的脉冲信号唤醒器件，或者自唤醒定时器超时后也可唤醒器件。唤醒时，器件重启。

注：LPC800 重启并完全上电后即处于工作模式，可对其进行操作。

5.5.1 唤醒过程

若器件在低功耗模式下接收唤醒信号，则它将唤醒至工作模式。

有关寄存器及唤醒指令，请参见下列链接：

- 唤醒后配置系统：表 36 “ 唤醒配置寄存器 （PDAWAKECFG、地址 0x4004 8234）位说明 ”。
- 利用外部中断唤醒：表 33 “ 启动逻辑 0 引脚唤醒使能寄存器 0 （STARTERP0，地址 0x4004 8204）位说明 ” 和表 32 “ 引脚中断选择寄存器 （PINTSEL[0:7]，地址 0x4004 8178 (PINTSEL0) 至 0x4004 8194 (PINTSEL7)）位说明 ”。
- 使能外部或内部信号，将器件从深度睡眠模式或掉电模式中唤醒：表 34 “ 启动逻辑 1 中断唤醒使能寄存器 （STARTERP1，地址 0x4004 8214）位说明 ”。
- 配置 USART 以唤醒器件：章节 15.3.2 “ 配置 USART 以实现唤醒 ”。
- 配置自唤醒定时器：章节 13.5。
- 所有唤醒信号源列表：表 42 “ 低功耗模式的唤醒信号源 ”。

表 42. 低功耗模式的唤醒信号源

电源模式	唤醒信号源	条件
睡眠	任意中断	NVIC 中使能中断。

表 42. 低功耗模式的唤醒信号源 *（续）*

电源模式	唤醒信号源	条件
深度睡眠和掉电	引脚中断	在 NVIC 和 STARTERP0 寄存器中使能引脚中断。
	BOD 中断	• 在 NVIC 和 STARTERP1 寄存器中使能中断。 • 在 BODCTRL 寄存器中使能中断。 • PDSLEEPCFG 寄存器实现 BOD 上电。
	BOD 复位	• 在 BODCTRL 寄存器中使能复位。 • PDSLEEPCFG 寄存器实现 BOD 上电。
	WWDT 中断	• 在 NVIC 和 STARTERP1 寄存器中使能中断。 • WWDT 运行中。在 WWDT MOD 寄存器中使能 WWDT 并输入。 • 在 WWDT MOD 寄存器中使能中断。 • PDSLEEPCFG 寄存器实现 WDOsc 上电。
	WWDT 复位	• WWDT 运行中。 • 在 WWDT MOD 寄存器中使能复位。 • PDSLEEPCFG 寄存器实现 WDOsc 上电。
	自唤醒定时器 (WKT) 超时	• 在 NVIC 和 STARTERP1 寄存器中使能中断。 • 在 PCON 模块的 DPDCTRL 寄存器中使能低功耗振荡器。 • 在 WKT CTRL 寄存器中选择用于 WKT 时钟的低功耗时钟。 • 将超时值写入 WKT COUNT 寄存器以启动 WKT。
	USART/SPI/I2C 外设中断	• 在 NVIC 和 STARTERP1 寄存器中使能中断。 • 使能 USART/I2C/SPI 中断。 • 为外设提供外部时钟信号。 • USART 配置为同步从机模式， I2C 和 SPI 配置为从机模式。
深度掉电模式	WAKEUP 引脚 PIO0_4	在 PMU 的 DPDCTRL 寄存器中使能 WAKEUP 功能。
	WKT 超时。	• 在 PMU 的 DPDCTRL 寄存器中使能低功耗振荡器。 • 在 PMU 的 DPDCTRL 寄存器中使能低功耗振荡器，使其在深度掉电模式下继续运行。 • 在 WKT CTRL 寄存器中选择用于 WKT 时钟的低功耗时钟。 • 将超时值写入 WKT COUNT 寄存器以启动 WKT。

5.6 寄存器说明

表 43. 寄存器概述：PMU （基址 0x4002 0000）

名称	访问类型	地址偏移	说明	复位值	参考
PCON	R/W	0x000	电源控制寄存器	0x0	表 44
GPREG0	R/W	0x004	通用寄存器 0	0x0	表 45
GPREG1	R/W	0x008	通用寄存器 1	0x0	表 45
GPREG2	R/W	0x00C	通用寄存器 2	0x0	表 45
GPREG3	R/W	0x010	通用寄存器 3	0x0	表 45
DPDCTRL	R/W	0x014	深度掉电控制寄存器	0x0	表 46

5.6.1 电源控制寄存器

电源控制寄存器选择是否进入其中一种 ARM Cortex-M0+ 控制的掉电模式 （睡眠模式或深度睡眠模式 / 掉电模式）或深度掉电模式，并分别提供睡眠或深度睡眠 / 掉电模式及深度掉电模式的标志。

表 44. 电源控制寄存器 （PCON，地址 0x4002 0000）位说明

位	符号	值	说明	复位值
2:0	PM		电源模式。	000
		0x0	默认。器件处于工作模式或睡眠模式。	
		0x1	ARM WFI 将进入深度睡眠模式。	
		0x2	ARM WFI 将进入掉电模式。	
		0x3	ARM WFI 将进入深度掉电模式（ARM Cortex-M0+ 内核掉电）。	
3	NODPD		该位中的 1 阻止器件进入深度掉电模式，同时将 0x3 写入上述 PM 字段，设置 SLEEPDEEP 位，并执行 WFI。 该位只有通过上电复位才可清零，因此将 1 写入该位将锁定器件，使其无法进入深度掉电模式。	0
7:4	-	-	保留。不能将 1 写入该位。	0
8	SLEEPFLAG		睡眠模式标志。	0
		0	读：不会进入掉电模式。器件处于工作模式。 写入：无效。	
		1	读：进入睡眠 / 深度睡眠或深度掉电模式。 写入：写入 1 会将 SLEEPFLAG 位清零为 0。	
10:9	-	-	保留。不能将 1 写入该位。	0
11	DPDFLAG		深度掉电标志。	0
		0	读：**不会**进入深度掉电模式。 写入：无效。	0
		1	读：进入深度掉电模式。 写入：清零深度掉电标志。	
31:12	-	-	保留。不能将 1 写入该位。	0

5.6.2 通用寄存器 0 至 3

在深度掉电模式下，如果 V_{DD} 引脚仍通电，但芯片已进入深度掉电模式，通用寄存器可保留数据。只有在完全移除芯片上的所有电源后进行冷启动，才会复位通用寄存器。

表 45. 通用寄存器 0 ~ 3（GPREG[0:3]，地址 0x4002 0004 (GPREG0) 至 0x4002 0010 (GPREG3)）位说明

位	符号	说明	复位值
31:0	GPDATA	深度掉电模式下保留数据。	0x0

5.6.3 深度掉电控制寄存器

深度掉电控制寄存器控制低功耗振荡器，自唤醒定时器可利用该振荡器将器件从深度掉电模式中唤醒。此外，该寄存器还可配置 WAKEUP 引脚的功能 （引脚 PIO0_4）。

寄存器中未用于深度掉电控制的位（位 31:4）可用来存储深度掉电模式下获取的额外数据，而这些数据的获取方式与寄存器 GPREG0 至 GPREG3 相同。

注：在深度掉电模式下，若施加于引脚 V_{DD} 上的外部电压可能下降至 2.2 V 以下，则 WAKEUP 输入引脚的迟滞必须在进入深度掉电模式之前在该寄存器中将其禁用，否则芯片无法被唤醒。

注：深度掉电模式下使能低功耗振荡器将增加功耗。仅在需要利用自唤醒定时器将器件从深度掉电模式中唤醒时，才使能该振荡器。若唤醒引脚挪作他用且不提供唤醒功能，则可能需要用到自唤醒定时器。

表 46. 深度掉电控制寄存器 （DPDCTRL，地址 0x4002 0014）位说明

位	符号	值	说明	复位值
0	WAKEUPHYS		WAKEUP 引脚迟滞使能。	0
		0	禁用。 WAKEUP 引脚迟滞禁用。	
		1	使能。 WAKEUP 引脚迟滞使能。	
1	WAKEPAD_ DISABLE		WAKEUP 引脚禁用。设置该引脚可禁用唤醒引脚，以便将其挪作他用。 **注：**若需使用某个引脚将器件从深度掉电模式中唤醒，则不可设置该位。仅在使能并配置了自唤醒定时器的情况下才可禁用唤醒引脚。 **注：**若不使用深度掉电模式，则无需设置该位。	0
		0	使能。引脚 PIO0_4 可使能唤醒功能。	
		1	禁用。设置该位可禁用引脚 PIO0_4 上的唤醒功能。	
2	LPOSCEN		使能低功耗振荡器，与 10 kHz 自唤醒定时器时钟一起使用。若设置自唤醒定时器 CTRL 中的 CLKSEL 位，则必须设置该位。 若自唤醒定时器采用分频 IRC 作为时钟，则不可使能低功耗振荡器。	0
		0	禁用。	
		1	使能。	
3	LPOSCDPDEN		深度掉电模式下使能低功耗振荡器。设置该位可让低功耗振荡器在深度掉电模式下保持工作，前提是该寄存器中的位 2 亦被设置。 必须设置该位，自唤醒定时器才可将器件从深度掉电模式中唤醒。 **注：**不要设置该位，除非使用自唤醒定时器将器件从深度掉电模式唤醒。	0
		0	禁用。	
		1	使能。	
31:4	-		深度掉电模式下保留数据。	0x0

5.7 功能说明

5.7.1 电源管理

LPC800 支持多种电源控制功能。在工作模式下，当芯片运行时，可以对所选外设的电源和时钟进行优化，从而降低功耗。此外，处理器还有四种特殊的低功耗模式，可让不同外设分别运行在：睡眠模式、深度睡眠模式、掉电模式和深度掉电模式。

表 47. 低功耗模式下的外设配置

外设	睡眠模式	深度睡眠模式	掉电模式	深度掉电模式
IRC	软件可配置	开	关	关
IRC 输出	软件可配置	关	关	关
闪存	软件可配置	开	关	关
BOD	软件可配置	软件可配置	软件可配置	关
PLL	软件可配置	关	关	关
SysOsc	软件可配置	关	关	关

表 47. 低功耗模式下的外设配置 *（续）*

外设	睡眠模式	深度睡眠模式	掉电模式	深度掉电模式
WDosc/WWDT	软件可配置	软件可配置	软件可配置	关
数字外设	软件可配置	关	关	关
WKT/ 低功耗振荡器	软件可配置	软件可配置	软件可配置	软件可配置

注：在睡眠模式、深度睡眠模式、掉电模式或深度掉电模式下，不支持调试模式。

5.7.2 低功耗模式和 WWDT 锁定特性

WWDT 锁定特性影响所有电源模式下的功耗，因为锁定 WWDT 时钟源可迫使看门狗振荡器开启，而无视 PDSLEEPCFG 寄存器以软件方式配置的深度睡眠和掉电模式。有关详情，请参见章节 12.5.3 “ 使用 WWDT 锁定特性 ”。

5.7.3 工作模式

工作模式下， ARM Cortex-M0+ 内核、存储器和外设均采用系统时钟或主时钟。

芯片复位后处于工作模式，而且默认电源配置由 PDRUNCFG 和 SYSAHBCLKCTRL 寄存器的复位值决定。在运行时可以更改电源配置。

5.7.3.1 工作模式下的电源配置

工作模式下的功耗由以下配置选项决定：

- SYSAHBCLKCTRL 寄存器控制所运行的存储器和外设 （表 18）。
- 各模块 （PLL、振荡器、 BOD 电路和闪存模块）的电源可通过 PDRUNCFG 寄存器随时单独控制（章节 37 “ 电源配置寄存器（PDRUNCFG、地址 0x4004 8238）位说明 ”）。
- 系统时钟的时钟源可以从 IRC （默认）、系统振荡器或看门狗振荡器中选择（参见图 3 和相关寄存器）。
- 系统时钟的频率可通过 SYSPLLCTRL （表 8）和 SYSAHBCLKDIV 寄存器 （表 17）来选择。
- USART 和 CLKOUT 使用单独的外设时钟及其自身的时钟分频器。外设时钟可以通过相应的时钟分频寄存器来关闭。

5.7.4 睡眠模式

在睡眠模式下，ARM Cortex-M0+ 内核的系统时钟停止，且在复位或中断出现之前都不能执行指令。

在睡眠模式下，外设功能 （如果选择在 SYSAHBCLKCTRL 寄存器中计时）继续运行，并可能产生中断使处理器继续运行。睡眠模式消除了处理器自身、存储器系统及其相关控制器和内部总线的动态功耗。处理器的状态和寄存器、外设寄存器和内部 SRAM 的值都会保留，引脚的逻辑电平保持静态。

5.7.4.1 睡眠模式下的电源配置

睡眠模式下的功耗根据工作模式下相同的设置进行配置：

- 时钟继续运行。
- 系统时钟的频率与工作模式下的频率保持一致，但处理器未定时。
- 模拟和数字外设与工作模式下选择的一样。

5.7.4.2 设置睡眠模式

进入睡眠模式的步骤如下：

1. PCON 寄存器中的 PM 位必须设置为默认值 0x0。
2. ARM Cortex-M0+ SCR 寄存器中的 SLEEPDEEP 位必须设置为 0 （表 41）。
3. 使用 ARM Cortex-M0+ 等待中断 (WFI) 指令。

5.7.4.3 从睡眠模式唤醒

在 NVIC 启用的中断到达处理器或发生复位时，系统将自动退出睡眠模式。因中断而唤醒之后，微控制器将返回由 PDRUNCFG 和 SYSAHBCLKDIV 寄存器的内容定义的原始电源配置。如果发生复位，微控制器会进入工作模式下的默认配置。

5.7.5 深度睡眠模式

深度睡眠模式下，处理器的系统时钟如睡眠模式中一样被禁用。在 PDSLEEPCFG 寄存器处于深度睡眠模式时，除了 BOD 电路和看门狗振荡器之外，所有模拟模块掉电，必须选择或取消选择。主时钟 （以至于所有外设时钟）均被禁用，但看门狗定时器除外 （若选中看门狗振荡器）。 IRC 正常工作，但其输出被禁用。闪存处于待机模式。

深度睡眠模式消除了模拟外设使用的所有功耗以及处理器自身、存储器系统及其相关控制器和内部总线所使用的所有动态功耗。处理器的状态和寄存器、外设寄存器和内部 SRAM 的值都会保留，引脚的逻辑电平保持静态。

5.7.5.1 深度睡眠模式下的电源配置

深度睡眠模式下的功耗由 PDSLEEPCFG （表 35）寄存器中的深度睡眠电源配置的设置决定：

- 如果 WWDT 需要，看门狗振荡器可在深度睡眠模式下继续运行。
- 如果应用程序需要， BOD 电路可在深度睡眠模式下继续运行。

5.7.5.2 设置深度睡眠模式

进入深度睡眠模式的步骤如下：

1. PCON 寄存器中的 PM 位必须设置为 0x1 （表 44）。
2. 选择 PDSLEEPCFG （表 35）寄存器深度睡眠模式下的电源配置
3. 在 PDAWAKECFG （表 36）寄存器中唤醒之后选择电源配置。
4. 若唤醒操作需要用到任何可用的唤醒中断，则在中断唤醒寄存器（表 33，表 34）和 NVIC 中使能中断。
5. 将 1 写入 ARM Cortex-M0+ SCR 寄存器 （表 41）中的 SLEEPDEEP 位。
6. 使用 ARM WFI 指令。

5.7.5.3 从深度睡眠模式唤醒

微控制器从深度睡眠模式唤醒的方式有以下几种：

- 在某个引脚中断（表32中选定，共8个）上发出信号。所有引脚中断同时必须在STARTERP0 寄存器 （表 33）和 NVIC 中使能。
- 发出 BOD 信号，如果 BOD 在 PDSLEEPCFG 寄存器中使能：
  - 使用深度睡眠中断唤醒寄存器 1 （表 34）的 BOD 中断。必须在 NVIC 中使能 BOD 中断。必须在 BODCTRL 寄存器中选定 BOD 中断。

- 从 BOD 电路复位。在这种情况下，必须在 PDSLEEPCFG 寄存器中使能 BOD 电路，而且必须在 BODCTRL 寄存器 （表 28）中使能 BOD 复位。

- 发出 WWDT 信号，如果看门狗振荡器在 PDSLEEPCFG 寄存器中使能：
  - 使用中断唤醒寄存器 1 （表 34）的 WWDT 中断。必须在 NVIC 中使能 WWDT 中断。必须在 WWDT MOD 寄存器中设置 WWDT 中断；必须在 SYSAHBCLKCTRL 寄存器中使能 WWDT。
  - 从看门狗定时器复位。必须在 WWDT MOD 寄存器中复位 WWDT。在这种情况下，看门狗振荡器必须在深度睡眠模式下运行 （参见 PDSLEEPCFG 寄存器），而且必须在 SYSAHBCLKCTRL 寄存器中启用 WDT。
- 若 USART 配置为同步模式，则可通过任意 USART 模块唤醒。参见章节 15.3.2 “ 配置 USART 以实现唤醒 ”。
- 通过 I2C 唤醒。参见章节 16.3.2。
- 通过任意 SPI 模块唤醒。参见章节 17.3.1。

注：

5.7.6 掉电模式

在掉电模式下，处理器的系统时钟如睡眠模式中一样被禁用。在 PDSLEEPCFG 寄存器处于掉电模式时，除了 BOD 电路和看门狗振荡器之外，所有模拟模块掉电，必须选择或取消选择。主时钟 （以至于所有外设时钟）均被禁用，但看门狗定时器除外 （若选中看门狗振荡器）。 IRC 自身以及闪存均掉电，与深度睡眠模式相比可进一步降低功耗。

掉电模式消除了模拟外设使用的所有功耗以及处理器自身、存储器系统及其相关控制器和内部总线所使用的所有动态功耗。处理器的状态和寄存器、外设寄存器和内部 SRAM 的值都会保留，引脚的逻辑电平保持静态。与深度睡眠模式相比，唤醒时间更长。

5.7.6.1 掉电模式下的电源配置

掉电模式下的功耗可通过 PDSLEEPCFG （表 35）寄存器中的电源配置设置、采用和深度睡眠模式相同的方式进行配置 （参见章节 5.7.5.1）：

- 如果 WWDT 需要，看门狗振荡器可在掉电模式下继续运行。
- 如果应用程序需要， BOD 电路可在掉电模式下继续运行。

5.7.6.2 设置掉电模式

进入掉电模式的步骤如下：

1. PCON 寄存器中的 PM 位必须设置为 0x2 （表 44）。
2. 选择 PDSLEEPCFG （表 35）寄存器掉电模式下的电源配置。
3. 在 PDAWAKECFG （表 36）寄存器中唤醒之后选择电源配置。
4. 若唤醒操作需要用到任何可用的唤醒中断，则在中断唤醒寄存器 （表 33，表 34）和 NVIC 中使能中断。
5. 将 1 写入 ARM Cortex-M0+ SCR 寄存器 （表 41）中的 SLEEPDEEP 位。
6. 使用 ARM WFI 指令。

5.7.6.3 从掉电模式唤醒

可从掉电模式中唤醒微控制器，方法与从深度睡眠模式中唤醒相同：

- 在某个引脚中断（表 32 中选定，共 8 个）上发出信号。所有引脚中断同时必须在 STARTERP0 寄存器（表 33）和 NVIC 中使能。
- 发出 BOD 信号，如果 BOD 在 PDSLEEPCFG 寄存器中使能：
  - 使用中断唤醒寄存器 1（表 34）的 BOD 中断。必须在 NVIC 中使能 BOD 中断。必须在 BODCTRL 寄存器中选定 BOD 中断。
  - 从BOD电路复位。这种情况下，必须在BODCTRL寄存器（表28）中使能BOD复位。
- 发出 WWDT 信号，如果看门狗振荡器在 PDSLEEPCFG 寄存器中使能：
  - 使用中断唤醒寄存器 1（表 34）的 WWDT 中断。必须在 NVIC 中使能 WWDT 中断。必须在 WWDT MOD 寄存器中设置 WWDT 中断。
  - 从看门狗定时器复位。必须在 WWDT MOD 寄存器中复位 WWDT。
  - 通过任意 USART 模块唤醒。参见章节 15.3.2 “ 配置 USART 以实现唤醒 ”。
  - 通过 I2C 唤醒。参见章节 16.3.2。
  - 通过任意 SPI 模块唤醒。参见章节 17.3.1。

5.7.7 深度掉电模式

深度掉电模式下，除了 WAKEUP 引脚和自唤醒定时器外，整个芯片的电源和时钟都被关闭。

深度掉电模式下，除了少量数据可以存储在 PMU 模块的通用寄存器中之外，SRAM 和寄存器的内容将不会被保留。

深度掉电模式下，除了 WAKEUP 引脚之外，所有功能引脚都是三态的。该模式下，必须从外部将 RESET 引脚拉至高电平。

注：设置 PCON 寄存器中的位 3（表 44）可防止器件进入深度掉电模式。

5.7.7.1 深度掉电模式下的电源配置

深度掉电模式没有配置选项。所有时钟、内核和所有外设均掉电。只有 WAKEUP 引脚和自唤醒定时器上电。

5.7.7.2 通过 WAKEUP 引脚设置深度掉电模式：

使用 WAKEUP 引脚进行唤醒时，必须执行下列步骤，以便进入深度掉电模式：

1. 从外部将 WAKEUP 引脚拉至高电平。
2. 确保 PCON 寄存器中的位 3（表 44）已被清零。
3. 将 0x3 写入 PCON 寄存器（参见表 44）中的 PM 位。
4. 存储将要保留在通用寄存器（章节 5.6.2）中的数据。
5. 将 1 写入 ARM Cortex-M0+ SCR 寄存器（表 41）中的 SLEEPDEEP 位。
6. 使用 ARM WFI 指令。

5.7.7.3 通过 WAKEUP 引脚唤醒深度掉电模式：

将 WAKEUP 引脚拉至低电平会将 LPC800 从深度掉电模式唤醒，并且器件将完成整个复位过程。

1. WAKEUP 引脚从高电平跃迁为低电平。
   - PMU 将会开启片内调压器。内核电压达到上电复位 (POR) 的跳变点时，系统会复位，芯片也将重新启动。

- 除 DPDCTRL 和 GPREG0 ~ GPREG3 外的所有寄存器将处于复位状态。

2. 引导芯片后，读取 PCON 寄存器（表 44）中的深度掉电标志，以证明复位是由深度掉电模式的唤醒事件引起，而非冷复位引起。
3. 清零 PCON 寄存器（表 44）中的深度掉电标志。
4. （可选）读取通用寄存器（章节 5.6.2）中存储的数据。
5. 为下一个深度掉电循环设置 PMU。

注：在深度掉电模式下，$\overline{\text{RESET}}$ 引脚没有任何功能。

5.7.7.4 通过自唤醒定时器设置深度掉电模式：

使用自唤醒定时器进行唤醒时，必须执行下列步骤，以便进入深度掉电模式：

1. 通过将 DPDCTRL 寄存器中的位 2 和位 3 设置为 1，使能深度掉电模式下的低功耗振荡器（参见表 46）
2. 确保 PCON 寄存器中的位 3（表 44）已被清零。
3. 将 0x3 写入 PCON 寄存器（参见表 44）中的 PM 位。
4. 存储将要保留在通用寄存器（章节 5.6.2）中的数据。
5. 将 1 写入 ARM Cortex-M0+ SCR 寄存器中的 SLEEPDEEP 位。
6. 将数值写入 WKT COUNT 寄存器（表 152）以启动自唤醒定时器。
7. 使用 ARM WFI 指令。

5.7.7.5 通过自唤醒定时器从深度掉电模式唤醒：

若自唤醒定时器超时，则器件将完成整个复位过程。

1. 当 WKT 计数至 0 时，将发生下列情况：
   - PMU 将会开启片内调压器。内核电压达到上电复位 (POR) 的跳变点时，系统会复位，芯片也将重新启动。
   - 除 DPDCTRL 和 GPREG0 ~ GPREG3 外的所有寄存器将处于复位状态。
2. 引导芯片后，读取 PCON 寄存器（表 44）中的深度掉电标志，以证明复位是由深度掉电模式的唤醒事件引起，而非冷复位引起。
3. 清零 PCON 寄存器（表 44）中的深度掉电标志。
4. （可选）读取通用寄存器（章节 5.6.2）中存储的数据。
5. 为下一个深度掉电循环设置 PMU。

注：在深度掉电模式下，$\overline{\text{RESET}}$ 引脚没有任何功能。

6.1 本章导读

所有 LPC800 器件的 IOCON 模块都是一样的。对于不提供某些引脚的特定封装，其相应寄存器被保留。

表 48. 引脚配置摘要

封装	可用的引脚 / 配置寄存器
TSSOP16	PIO0_0 至 PIO0_13
TSSOP20	PIO0_0 至 PIO0_17
SOP20	PIO0_0 至 PIO0_17
DIP8	PIO0_0 至 PIO0_5

6.2 特性

每个引脚都可配置下列电气性能：

- 上拉 / 下拉电阻
- 开漏模式
- 迟滞
- 带可编程时间常数的数字干扰滤波器
- 模拟模式 （有关引脚子集，请参见 LPC81xM 数据手册）

真正的开漏引脚 PIO0_10 和 PIO0_11 可配置成不同的 I2C 总线速度。

6.3 基本配置

在 SYSAHBCLKCTRL 寄存器 （表 18，位 18）中使能 IOCON 的时钟。一旦完成引脚配置，便可禁用 IOCON 时钟，降低功耗。

注：若封装不提供开漏引脚 PIO0_10 和 PIO0_11，则以如下方式防止引脚内部浮空：将 GPIO DIR0 寄存器中的位 10 和位 11 设置为 1 可使能输出驱动器，将 1 写入 GPIO CLR0 寄存器的位 10 和位 11 可内部驱动输出至低电平。

6.4 简介

6.4.1 引脚配置

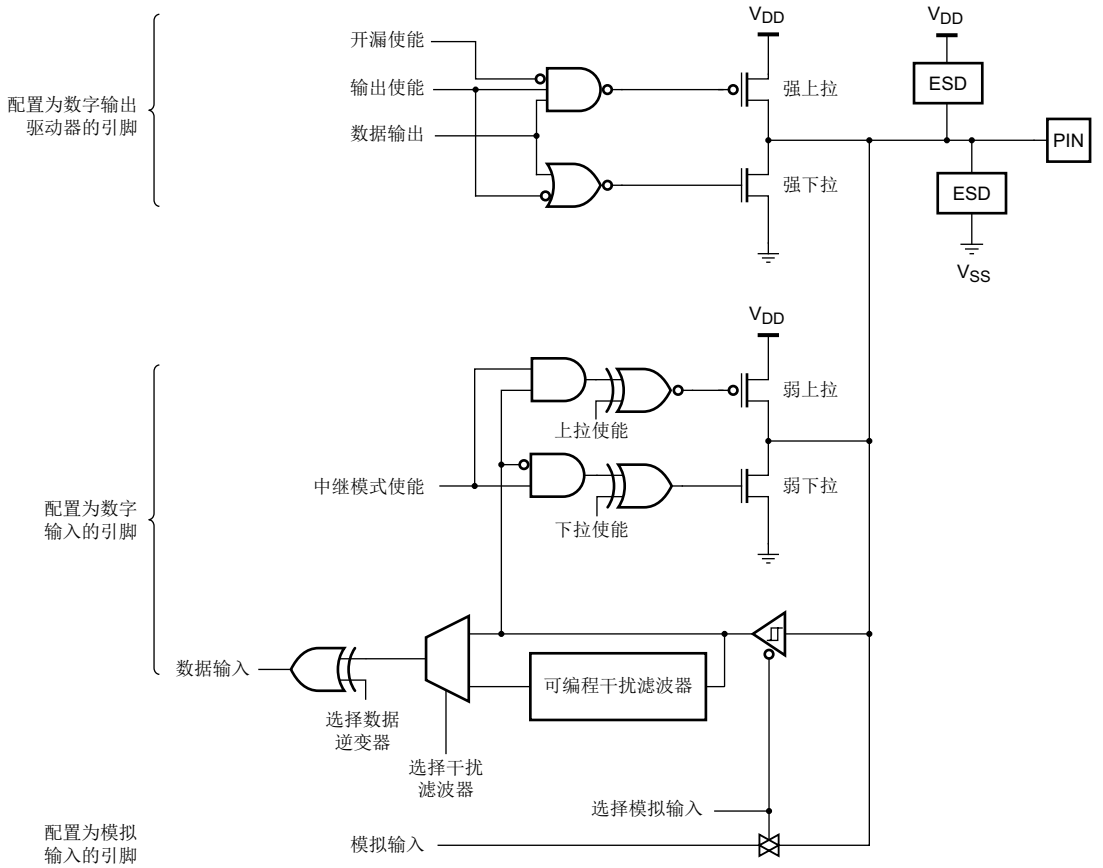


图 5. 引脚配置

6.4.2 引脚功能

引脚功能完全通过开关矩阵配置。默认情况下，GPIO 的某项功能会被分配给每个引脚。开关矩阵可将可转移功能表中的所有功能分配给 IOCON 模块的任意引脚，或针对特定的引脚使能特定功能，如模拟输入。

相关链接：

表 95 “ 可转移功能 （通过开关矩阵分配给引脚 PIO0_0 至 PIO0_17） ”

6.4.3 引脚模式

IOCON 寄存器的 MODE 位允许针对每个引脚使能或禁用片内上拉电阻。默认情况下，除 I^2C 总线引脚 PIO0_10 和 PIO0_11 外 - 这两个引脚没有可编程的上拉电阻，所有引脚的上拉电阻都被使能。

引脚处于高电平时，中继模式会使能上拉电阻；引脚处于低电平时，则会使能下拉电阻。这样，当引脚配置为输入且由外部驱动时，会使引脚保持其上一已知状态。如果暂时不驱动引脚，通常可用中继模式来防止引脚浮空（当引脚处于不确定状态时，可能会使用大量电源）。

6.4.4 开漏模式

开漏模式可使能用于所有数字 I/O 引脚。除引脚 PIO0_10 和 PIO0_11 外，该模式不是真正的开漏模式。输入上拉后的电平不得超过 V_{DD}。

6.4.5 模拟模式

一旦模拟输入或输出被选为引脚功能，开关矩阵就会自动将引脚配置为模拟模式。

6.4.6 I^2C 总线模式

I^2C 总线引脚 PIO0_10 和 PIO0_11 可编程为支持真正的开漏模式，无论选择了 I2C 功能还是其它数字功能。如果选择了 I^2C 功能，则支持所有三个 I^2C 模式，即标准模式、快速模式和超快速模式。针对所有功能都可配置一个数字干扰滤波器。引脚 PIO0_10 和 PIO0_11 独立于被编程的功能作为高电流接收器驱动程序 (20 mA) 工作。

6.4.7 可编程干扰滤波器

所有 GPIO 引脚都配有可编程数字干扰滤波器。滤波器在一个可选择的时间段内抑制输入脉冲，这个时间段可短于一个、两个或三个滤波器时钟周期（S_MODE = 1、2 或 3）。就每个单独引脚而言，滤波器时钟可从 7 个外设时钟 PCLK0 到 6 中选择，这些时钟源自使用 IOCONCLKDIV0 到 6 寄存器的主时钟。也可以完全绕过滤波器。

满足以下条件时，持续时间为 T_{pulse} 且具有任意极性的输入脉冲将被抑制：
$T_{pulse} < T_{PCLKn} \times S\_MODE$

长一个滤波器时钟周期的输入脉冲也可能被抑制：

$T_{pulse} = T_{PCLKn} \times (S\_MODE + 1)$

注：滤波效果通过以下过程实现：要求输入信号在滤波器时钟内 (S_MODE +1) 个连续边沿保持稳定然后才传递到芯片。使能滤波器会导致延迟将信号发往内部逻辑，仅当某个应用特别要求时才应这样做。对于高速或时间关键功能，确保绕过该滤波器。

如果必须最大限度缩短输入信号的延时，请选择更快的 PCLK 和更高的采样模式 (S_MODE)，以最大限度减少潜在额外时钟周期的影响。

如果必须最大限度降低对噪音尖峰的敏感性，应选择更慢的 PCLK 和更低的采样模式。

相关寄存器和链接：

表 27 “IOCON 干扰滤波器时钟分频器寄存器 6 到 0（IOCONCLKDIV[6:0]，地址 0x4004 8134 (IOCONCLKDIV6) 至 0x004 814C (IOCONFILTCLKDIV0)）位说明 ”

6.5 寄存器说明

每个端口引脚 PIOn_m 都有一个 IOCON 寄存器专门负责控制引脚功能和电气特性。

表 49. 寄存器概述：I/O 配置 （基址 0x4004 4000）

名称	访问类型	地址偏移	说明	复位值	参考
PIO0_17	R/W	0x000	引脚 PIO0_17 的 I/O 配置	0x0000 0090	表 50
PIO0_13	R/W	0x004	引脚 PIO0_13 的 I/O 配置	0x0000 0090	表 51
PIO0_12	R/W	0x008	引脚 PIO0_12 的 I/O 配置	0x0000 0090	表 52
PIO0_5	R/W	0x00C	引脚 PIO0_5 的 I/O 配置 /RESET	0x0000 0090	表 53
PIO0_4	R/W	0x010	引脚 PIO0_4 的 I/O 配置	0x0000 0090	表 54
PIO0_3	R/W	0x014	引脚 PIO0_3/SWCLK 的 I/O 配置	0x0000 0090	表 55
PIO0_2	R/W	0x018	引脚 PIO0_2/SWDIO 的 I/O 配置	0x0000 0090	表 56
PIO0_11	R/W	0x01C	引脚 PIO0_11 的 I/O 配置。这是真正开漏引脚的引脚配置。	0x0000 0080	表 57
PIO0_10	R/W	0x020	引脚 PIO0_10 的 I/O 配置。这是真正开漏引脚的引脚配置。	0x0000 0080	表 58
PIO0_16	R/W	0x024	引脚 PIO0_16 的 I/O 配置	0x0000 0090	表 59
PIO0_15	R/W	0x028	引脚 PIO0_15 的 I/O 配置	0x0000 0090	表 60
PIO0_1	R/W	0x02C	引脚 PIO0_1/ACMP_I1/CLKIN 的 I/O 配置	0x0000 0090	表 61
-	-	0x030	保留	-	-
PIO0_9	R/W	0x034	引脚 PIO0_9/XTALOUT 的 I/O 配置	0x0000 0090	表 62
PIO0_8	R/W	0x038	引脚 PIO0_8/XTALIN 的 I/O 配置	0x0000 0090	表 63
PIO0_7	R/W	0x03C	引脚 PIO0_7 的 I/O 配置	0x0000 0090	表 64
PIO0_6	R/W	0x040	引脚 PIO0_6/VDDCMP 的 I/O 配置	0x0000 0090	表 65
PIO0_0	R/W	0x044	引脚 PIO0_0/ACMP_I0 的 I/O 配置	0x0000 0090	表 66
PIO0_14	R/W	0x048	引脚 PIO0_14 的 I/O 配置	0x0000 0090	表 67

6.5.1 PIO0_17 寄存器

表 50. PIO0_17 寄存器 （PIO0_17，地址 0x4004 4000）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001

表 50. PIO0_17 寄存器 （PIO0_17，地址 0x4004 4000）位说明 *（续）*

位	符号	值	说明	复位值
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.2 PIO0_13 寄存器

表 51. PIO0_13 寄存器 （PIO0_13，地址 0x4004 4004）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	

表 51. PIO0_13 寄存器 （PIO0_13，地址 0x4004 4004）位说明 *（续）*

位	符号	值	说明	复位值
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.3 PIO0_12 寄存器

表 52. PIO0_12 寄存器 （PIO0_12，地址 0x4004 4008）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	

表 52. PIO0_12 寄存器 （PIO0_12，地址 0x4004 4008）位说明 *（续）*

位	符号	值	说明	复位值
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.4 PIO0_5 寄存器

表 53. PIO0_5 寄存器 （PIO0_5，地址 0x4004 400C）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	

表 53. PIO0_5 寄存器 （PIO0_5，地址 0x4004 400C）位说明 *（续）*

位	符号	值	说明	复位值
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.5 PIO0_4 寄存器

表 54. PIO0_4 寄存器 （PIO0_4，地址 0x4004 4010）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	

表 54. PIO0_4 寄存器 （PIO0_4，地址 0x4004 4010）位说明 *（续）*

位	符号	值	说明	复位值
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.6 PIO0_3 寄存器

表 55. PIO0_3 寄存器 （PIO0_3，地址 0x4004 4014）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	

表 55. PIO0_3 寄存器 （PIO0_3，地址 0x4004 4014）位说明 *（续）*

位	符号	值	说明	复位值
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.7 PIO0_2 寄存器

表 56. PIO0_2 寄存器 （PIO0_2，地址 0x4004 4018）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	

表 56. PIO0_2 寄存器 （PIO0_2，地址 0x4004 4018）位说明 *（续）*

位	符号	值	说明	复位值
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.8 PIO0_11 寄存器

表 57. PIO0_11 寄存器 （PIO0_11，地址 0x4004 401C）位说明

位	符号	值	说明	复位值
5:0	-		保留。	0
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
7	-		保留。	1
9:8	I2CMODE		选择 I2C 模式。 若引脚功能为 GPIO (FUNC = 000)，则选择标准模式（I2CMODE = 00，默认）或标准 I/O 功能 (I2CMODE = 01)。	00
		0x0	标准模式 / 快速模式 I2C。	
		0x1	标准 I/O 功能	
		0x2	超快速模式 I2C	
		0x3	保留。	
10	-	-	保留。	-
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	-

6.5.9 PIO0_10 寄存器

表 58. PIO0_10 寄存器 （PIO0_10，地址 0x4004 4020）位说明

位	符号	值	说明	复位值
5:0	-		保留。	0
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
7	-		保留。	1
9:8	I2CMODE		选择 I2C 模式。 若引脚功能为 GPIO (FUNC = 000)，则选择标准模式（I2CMODE = 00，默认）或标准 I/O 功能 (I2CMODE = 01)。	00
		0x0	标准模式 / 快速模式 I2C。	
		0x1	标准 I/O 功能	
		0x2	超快速模式 I2C	
		0x3	保留。	
10	-	-	保留。	-
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	-

6.5.10 PIO0_16 寄存器

表 59. PIO0_16 寄存器 （PIO0_16，地址 0x4004 4024）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	

表 59. PIO0_16 寄存器 （PIO0_16，地址 0x4004 4024）位说明 *（续）*

位	符号	值	说明	复位值
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.11 PIO0_15 寄存器

表 60. PIO0_15 寄存器 （PIO0_15，地址 0x4004 4028）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	

表 60. PIO0_15 寄存器 （PIO0_15，地址 0x4004 4028）位说明 *（续）*

位	符号	值	说明	复位值
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.12 PIO0_1 寄存器

表 61. PIO0_1 寄存器 （PIO0_1，地址 0x4004 402C）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001

表 61. PIO0_1 寄存器 （PIO0_1，地址 0x4004 402C）位说明 *（续）*

位	符号	值	说明	复位值
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.13 PIO0_9 寄存器

表 62. PIO0_9 寄存器 （PIO0_9，地址 0x4004 4034）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0

表 62. PIO0_9 寄存器 （PIO0_9，地址 0x4004 4034）位说明 *（续）*

位	符号	值	说明	复位值
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.14 PIO0_8 寄存器

表 63. PIO0_8 寄存器 （PIO0_8，地址 0x4004 4038）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	

表 63. PIO0_8 寄存器 （PIO0_8，地址 0x4004 4038）位说明 *（续）*

位	符号	值	说明	复位值
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.15 PIO0_7 寄存器

表 64. PIO0_7 寄存器 （PIO0_7，地址 0x4004 403C）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	

表 64. PIO0_7 寄存器 （PIO0_7，地址 0x4004 403C）位说明 *（续）*

位	符号	值	说明	复位值
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.16 PIO0_6 寄存器

表 65. PIO0_6 寄存器 （PIO0_6，地址 0x4004 4040）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	

表 65. PIO0_6 寄存器 （PIO0_6，地址 0x4004 4040）位说明 *（续）*

位	符号	值	说明	复位值
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.17 PIO0_0 寄存器

表 66. PIO0_0 寄存器 （PIO0_0，地址 0x4004 4044）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

6.5.18 PIO0_14 寄存器

表 67. PIO0_14 寄存器 （PIO0_14，地址 0x4004 4048）位说明

位	符号	值	说明	复位值
2:0	-		保留。	0
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0b10
		0x0	无效 （未使能上拉 / 下拉电阻）。	
		0x1	下拉电阻使能。	
		0x2	上拉电阻使能。	
		0x3	中继模式。	
5	HYS		迟滞。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入。	0
		0	输入未反转 （引脚高电平以 1 读取；引脚低电平以 0 读取）。	
		1	输入反转 （引脚高电平以 0 读取；引脚低电平以 1 读取）。	
9:7	-	-	保留。	0b001
10	OD		开漏模式。	0
		0	禁用。	
		1	开漏模式使能。 **注：**这不是真正的开漏模式。	
12:11	S_MODE		数字滤波器采样模式。	0
		0x0	绕过输入滤波器。	
		0x1	1 个时钟周期。不足 1 个滤波器时钟的输入脉冲将被抑制。	
		0x2	2 个时钟周期。不足 2 个滤波器时钟的输入脉冲将被抑制。	
		0x3	3 个时钟周期。不足 3 个滤波器时钟的输入脉冲将被抑制。	
15:13	CLK_DIV		选择用于输入滤波器采样时钟的外设时钟分频器。值 0x7 保留。	0
		0x0	IOCONCLKDIV0。	
		0x1	IOCONCLKDIV1。	
		0x2	IOCONCLKDIV2。	
		0x3	IOCONCLKDIV3。	
		0x4	IOCONCLKDIV4。	
		0x5	IOCONCLKDIV5。	
		0x6	IOCONCLKDIV6。	
31:16	-	-	保留。	0

7.1 本章导读

所有 GPIO 寄存器的每个端口都有 32 个引脚。并非所有引脚都可用，具体取决于封装类型；另外， GPIO 寄存器中的对应位保留 （参见表 68）。

表 68. 可用的 GPIO 引脚

封装	GPIO 端口 0
TSSOP16	PIO0_0 至 PIO0_13
TSSOP20	PIO0_0 至 PIO0_17
SOP20	PIO0_0 至 PIO0_17
DIP8	PIO0_0 至 PIO0_5

7.2 特性

- GPIO 端口寄存器位于 ARM Cortex M0+ I/O 端口上，以支持快速访问。
- ARM Cortex M0+ I/O 端口支持单周期访问。
- GPIO 端口
  - GPIO 引脚可通过软件配置为输入或输出。
  - 复位时所有 GPIO 引脚默认为输入。
  - 引脚中断寄存器允许单独感测和设置各引脚。

7.3 基本配置

对于 GPIO 端口寄存器，在 SYSAHBCLKCTRL 寄存器中使能 GPIO 端口寄存器的时钟（表 18，位 6）。

7.4 引脚说明

所有 GPIO 功能均为固定引脚功能。开关矩阵分配所有 GPIO 端口引脚至 LPC800 封装的一个 （且只有一个）引脚。默认情况下，开关矩阵可将除电源引脚和接地引脚外的所有封装引脚连接至它们的 GPIO 端口引脚。

引脚说明表 （参见表 291）显示的是 GPIO 端口引脚如何分配到 LPC800 封装引脚。

7.5 简介

GPIO 端口寄存器可用于将每个 GPIO 引脚配置为输入或输出；引脚配置为输入时，读取每个引脚的状态；引脚配置为输出时，设置每个引脚的状态。

7.6 寄存器说明

GPIO 端口寄存器和 GPIO 引脚中断寄存器位于 ARM M0+ I/O 端口。I/O 端口支持单周期访问。

GPIO 端口地址的读写方式可以是字节、半字或字。

“ext” 是指复位后的数据读取取决于引脚的状态，而该状态可能取决于外部源。

注：可对 GPIO 寄存器中的保留位进行编程以防止开漏 I2C 引脚在未进行引脚输出的情况下内部浮空。参见章节 6.3。

表 69. 寄存器概述：GPIO 端口 （基址 0xA000 0000）

名称	访问类型	地址偏移	说明	复位值	宽度	参考
B0 至 B17	R/W	0x0000 至 0x0012	字节引脚寄存器端口 0 ；引脚 PIO0_0 至 PIO0_17	ext	字节 （8 位）	表 70
W0 至 W17	R/W	0x1000 至 0x1048	字引脚寄存器端口 0	ext	字 （32 位）	表 71
DIR0	R/W	0x2000	方向寄存器端口 0	0	字 （32 位）	表 72
MASK0	R/W	0x2080	屏蔽寄存器端口 0	0	字 （32 位）	表 73
PIN0	R/W	0x2100	端口引脚寄存器端口 0	ext	字 （32 位）	表 74
MPIN0	R/W	0x2180	屏蔽端口寄存器端口 0	ext	字 （32 位）	表 75
SET0	R/W	0x2200	写入：端口 0 的设置寄存器 读：端口 0 的输出位	0	字 （32 位）	表 76
CLR0	WO	0x2280	清零端口 0	不适用	字 （32 位）	表 77
NOT0	WO	0x2300	切换端口 0	不适用	字 （32 位）	表 78

7.6.1 GPIO 端口字节引脚寄存器

每个GPIO引脚在该地址范围内都拥有一个字节寄存器。软件通常读取和写入字节来访问各个引脚，读取或写入半字来感测或设置 2 个引脚的状态，读取或写入字来感测或设置 4 个引脚的状态。

表 70. GPIO 端口 0 字节引脚寄存器 （B[0:17]，地址 0xA000 0000 (B0) 至 0xA000 0012 (B17)）位说明

位	符号	说明	复位值	访问类型
0	PBYTE	读：引脚 PIO0_n 的状态不受方向、屏蔽或其他功能的影响，除非引脚配置为模拟 I/O，否则会始终以 0 读取。 写入：加载引脚的输出位。	ext	R/W
7:1		保留 （读取时为 0，写入时忽略）	0	-

7.6.2 GPIO 端口字引脚寄存器

每个 GPIO 引脚在该地址范围内都拥有一个字寄存器。如果引脚为低电平，则该范围内的任何字节、半字或字读取为全 0 ；如果引脚为高电平，则为全 1。结果不受引脚的方向、屏蔽或其他功能的影响，除非引脚配置为模拟 I/O，否则会始终以 0 读取。如果写入的值均为 0，则任何写入都将清零引脚的输出位，否则会设置引脚的输出位。

表 71. GPIO 端口 0 字引脚寄存器（W[0:17]，地址 0xA000 1000 (W0) 至 0x5000 1048 (W17)）位说明

位	符号	说明	复位值	访问类型
31:0	PWORD	读 0：引脚为低电平。 写 0：清零输出位。 读 0xFFFF FFFF：引脚为高电平。 写 0x0000 0001 至 0xFFFF FFFF 之间的任何值：设置输出位。 **注：**仅可读取 0 或 0xFFFF FFFF。写 0 以外的任何值时将设置输出位。	ext	R/W

7.6.3 GPIO 端口方向寄存器

每个 GPIO 端口都有一个方向寄存器，用于将端口引脚配置为输入或输出。

表 72. GPIO 方向端口 0 寄存器 （DIR0，地址 0xA000 2000）位说明

位	符号	说明	复位值	访问类型
17:0	DIRP0	选择引脚 PIO0_n 的引脚方向 （位 0 = PIO0_0，位 1 = PIO0_1， ...，位 17 = PIO0_17）。 0 = 输入。 1 = 输出。	0	R/W
31:18	-	保留。	0	-

7.6.4 GPIO 端口掩码寄存器

这些寄存器会影响对 MPORT 寄存器的读写操作。将这些寄存器设为 0 可使能读写操作；设为 1 则可禁用写操作并在读操作时将对应位置设为 0。

表 73. GPIO 掩码端口 0 寄存器 （MASK0，地址 0xA000 2080）位说明

位	符号	说明	复位值	访问类型
17:0	MASKP0	控制在P0MPORT寄存器中有效的与PIO0_n对应的位（位 0 = PIO0_0，位 1 = PIO0_1， ...，位 17 = PIO0_17）。 0 = 读 MPORT：引脚状态；写 MPORT：加载输出位。 1 = 读 MPORT：0 ；写 MPORT：输出位不受影响。	0	R/W
31:18	-	保留。	0	-

7.6.5 GPIO 端口引脚寄存器

对这些寄存器进行读操作会返回所读引脚的当前状态，结果不受引脚的方向、屏蔽或其他功能的影响，除非引脚配置为模拟 I/O，否则会始终以 0 读取。对这些寄存器进行写操作会加载所写引脚的输出位，无论是否 Mask 寄存器。

表 74. GPIO 端口 0 引脚寄存器 （PIN0，地址 0xA000 2100）位说明

位	符号	说明	复位值	访问类型
17:0	PORT0	读取引脚状态或加载输出位 （位 0 = PIO0_0，位 1 = PIO0_1， ...，位 17 = PIO0_17）。 0 = 读：引脚为低电平；写：清零输出位。 1 = 读：引脚为高电平；写：设置输出位。	ext	R/W
31:18	-	保留。	0	-

7.6.6 GPIO 屏蔽端口引脚寄存器

这些寄存器与引脚寄存器类似，不同之处在于通过和相应 MASK 寄存器中的反向内容进行“与”操作，可以屏蔽读取某些值；同时，对其中某个寄存器进行写操作仅影响在相应MASK 寄存器中由 0 使能的输出寄存器位

表 75. GPIO 屏蔽端口 0 引脚寄存器 （MPIN0，地址 0xA000 2180）位说明

位	符号	说明	复位值	访问类型
17:0	MPORTP0	屏蔽端口寄存器（位 0 = PIO0_0，位 1 = PIO0_1，...，位 17 = PIO0_17）。 0 = 读：引脚为低电平和 / 或 MASK 寄存器中的对应位为 1；写：当 MASK 寄存器中的对应位为 0 时，清零输出位。 1 = 读：引脚为高电平和 MASK 寄存器中的对应位为 0；写：当 MASK 寄存器中的对应位为 0 时，设置输出位。	ext	R/W
31:18	-	保留。	0	-

7.6.7 GPIO 端口设置寄存器

将 1 写入这些寄存器可设置输出位，无论其是否为 MASK 寄存器。对这些寄存器进行读操作会返回端口的输出位，无论其引脚方向如何。

表 76. GPIO 设置端口 0 寄存器 （SET0，地址 0xA000 2200）位说明

位	符号	说明	复位值	访问类型
17:0	SETP0	读取或设置输出位。 0 = 读：输出位；写入：无操作。 1 = 读：输出位；写：设置输出位。	0	R/W
31:18	-	保留。	0	-

7.6.8 GPIO 端口清零寄存器

将 1 写入这些只写寄存器可清零输出位，无论是否为 MASK 寄存器。

表 77. GPIO 清零端口 0 寄存器 （CLR0，地址 0xA000 2280）位说明

位	符号	说明	复位值	访问类型
17:0	CLRP0	清零输出位： 0 = 无操作。 1 = 清零输出位。	不适用	WO
31:18	-	保留。	0	-

7.6.9 GPIO 端口切换寄存器

将 1 写入这些只写寄存器可切换 / 反转 / 补充输出位，无论是否为 MASK 寄存器。

表 78. GPIO 切换端口 0 寄存器 （NOT0，地址 0xA000 2300）位说明

位	符号	说明	复位值	访问类型
17:0	NOTP0	切换输出位： 0 = 无操作。 1 = 切换输出位。	不适用	WO
31:18	-	保留。	0	-

7.7 功能说明

7.7.1 读取引脚状态

软件可以读取除在开关矩阵逻辑中选为模拟功能的引脚外的所有 GPIO 引脚的状态。要读取某个引脚的状态，并不一定要在开关矩阵中将其选为GPIO。有几种方式可以读取引脚状态：

- 可以从字节引脚寄存器中读取单个引脚的状态，为 7 个高位 0。
- 可以从字引脚寄存器中读取单个引脚的状态，为 1 个字节、半字或整字的所有位。
- 可以从 PORT 寄存器中读取多个引脚的状态，为 1 个字节、半字或整字。
- 可以从屏蔽端口 (MPORT) 寄存器中读取某个端口的所选引脚子集的状态。引脚在端口的 Mask 寄存器中读为 1，在 MPORT 寄存器中将读为 0。

7.7.2 GPIO 输出

每个 GPIO 引脚在 GPIO 模块中都有一个输出位。这些输出位是写操作 “ 到引脚 ” 的目标位。要将引脚的输出位驱动到引脚，必须符合两个条件：

1. 必须选择该引脚用于开关矩阵中的 GPIO 操作。
2. 必须选择该引脚用于输出，方法是将其端口的 DIR 寄存器设为 1。

如果某个条件无法满足或两个条件都无法满足，则写入到引脚无效。

有多种方式可以更改 GPIO 输出位：

- 对字节引脚寄存器进行写操作可从最低有效位加载输出位。
- 写入到字引脚寄存器可通过所有写入字节的 “ 或 ” 操作加载输出位。（该特性遵循程序设计语言中多位值为 “ 真 ” 的定义。）
- 对端口的 PORT 寄存器进行写操作可加载所有写入引脚的输出位。
- 对端口的 MPORT 寄存器进行写操作可加载通过将该端口的 MASK 寄存器对应位置设为 0 而确定的引脚的输出位。
- 将 1 写入端口的 SET 寄存器可设置输出位。
- 将 1 写入端口的 CLR 寄存器可清零输出位。
- 将 1 写入端口的 NOT 寄存器可切换 / 补充 / 反转输出位。

某个端口的输出位状态可从 SET 寄存器读取。对章节 7.7.1 中所述的任何寄存器进行写操作会返回引脚状态，无论其方向或可选功能如何。

7.7.3 屏蔽 I/O

端口的 MASK 寄存器可确定哪些引脚在 MPORT 寄存器中应当可访问。将 MASK 寄存器中的位设为 0 可使能从 MPORT 中读取和写入的对应引脚。 MASK 寄存器中的位为 1 时引脚会被强制以 0 读取，其输出位将不会受写入 MPORT 的影响。当端口的 MASK 寄存器中的位全为 0 时，其 PORT 寄存器和 MPORT 寄存器将以完全一致的方式进行读写。

在一些应用程序中，中断可能会引起屏蔽 GPIO 操作，或执行屏蔽 GPIO 操作的任务之间的切换，这些应用程序必须将使用 MASK 寄存器的代码作为受保护 / 受限制的区域。中断禁用或使用信号量都可以实现这一点。

保护使用 MASK 寄存器的代码块有更简单的方法：设置 MASK 寄存器前禁用中断，并在使用 MPORT 或 MASK 寄存器的最后一次操作结束后重新使能中断。

更有效的是，软件可以为 MASK 寄存器指定专用信号量；在设置 MASK 寄存器前，设置 / 捕获控制 MASK 寄存器专用的信号量；并在使用 MPORT 或 MASK 寄存器的最后一次操作结束后释放信号量。

7.7.4 推荐用法

下表列出了 GPIO 端口寄存器的若干推荐用法：

- 要在复位或重新初始化后进行初始设置，对 PORT 寄存器进行写操作。
- 要更改某个引脚的状态，对字节引脚寄存器或字引脚寄存器进行写操作。
- 要一次性更改多个引脚的状态，对 SET 寄存器和 / 或 CLR 寄存器进行写操作。
- 要在严格控制的环境（如软件状态机）中更改多个引脚的状态，则考虑使用 NOT 寄存器。这比 SET 和 CLR 需要的写操作要少。
- 要读取某个引脚的状态，对字节引脚寄存器或字引脚寄存器进行读操作。
- 要根据多个引脚作出决定，对 PORT 寄存器进行读操作并将其屏蔽。

8.1 本章导读

所有 LPC800 器件都提供引脚中断发生器和模式匹配引擎。

8.2 特性

- 引脚中断
  - 最多可从所有 GPIO 引脚中选择 8 个引脚作为边沿或电平敏感中断请求。每个请求会在 NVIC 中创建一个单独的中断。
  - 边沿敏感中断引脚可在上升沿和 / 或下降沿上发生中断。
  - 电平敏感中断引脚可以是高电平或低电平有效。
- 模式匹配引擎
  - 最多可从所有 GPIO 引脚中选择 8 个引脚组成布尔表达式。布尔表达式由经过各种组合后的这些引脚上指定的电平和 / 或跃迁构成。
  - 组成特定布尔表达式的每个位片最小项 （乘积项）都可生成其专有的中断请求。
  - 出现的任何模式匹配在经过编程后都可产生 ARM CPU 的 RXEV 通知。RXEV 信号可连接至引脚。
  - 模式匹配与软件搭配使用可创建基于引脚输入的复杂状态机。

8.3 基本配置

- 引脚中断：
  - 在 SYSCON 模块（参见表 32）中，最多可从所有 GPIO 端口引脚中选择 8 个外部中断引脚。引脚中断和模式匹配引擎的引脚选择过程是相同的。这两种特性是互斥的。
  - 在 SYSAHBCLKCTRL 寄存器 （表 18，位 6）中使能引脚中断寄存器模块的时钟。
  - 要使用引脚中断将器件从深度睡眠模式或掉电模式中唤醒，可在 STARTERP0 寄存器 （表 33）中使能引脚中断唤醒特性。
  - 每个选中的引脚中断都被分配到 NVIC 中的一个中断 （24 号中断到 31 号中断对应引脚中断 0 到引脚中断 7）。
- 模式匹配引擎：
  - 在 SYSCON 模块（表 32）中，最多可从所有 GPIO 端口引脚中选择 8 个外部引脚。引脚中断和模式匹配引擎的引脚选择过程是相同的。这两种特性是互斥的。
  - 在 SYSAHBCLKCTRL 寄存器 （表 18，位 6）中使能引脚中断寄存器模块的时钟。
  - 每个模式匹配引擎的位片都被分配到 NVIC 中的一个中断 （24 号中断到 31 号中断对应逻辑片 0 到逻辑片 7）。
  - 来自所有逻辑片或逻辑片组合的中断组合可通过开关矩阵可转移功能寄存器（PINASSIGN8，图 105）与 ARM RXEV 请求和引脚功能 GPIO_INT_BMAT 关联。

8.3.1 将引脚配置为引脚中断或模式匹配引擎的输入

按照下列步骤将引脚配置为引脚中断：

1. 确定 LPC800 封装上作为引脚中断的引脚。有关如何确定与封装引脚有关的 GPIO 端口引脚数，请参见数据手册。
2. 对于每个引脚中断，可在 SYSCON 模块中进行 GPIO 端口引脚数的编程，将其写入 8 个 PINTSEL 寄存器中的某一个。

   注：端口引脚号用于识别连接 PINTSEL 寄存器的引脚。包括 GPIO 在内的所有功能都可通过开关矩阵分配给该引脚。
3. 在 NVIC 中使能所有引脚中断。

一旦引脚中断或模式匹配输入完成配置，便可设定引脚中断检测电平或模拟匹配布尔表达式。

有关 PINTSEL 寄存器，请参见章节 4.6.27 “ 引脚中断选择寄存器 ” 中的 SYSCON 模块。

8.4 引脚说明

引脚中断和模式匹配引擎的输入由 SYSCON 模块中的引脚中断选择寄存器决定。参见章节 8.3.1。

模式匹配引擎输出通过开关矩阵分配给外部引脚。

有关将 GPIO 模式匹配功能分配给 LPC800 封装引脚的步骤，请参见章节 9.3.1 “ 将内部信号与封装引脚相连 ”。

表 79. 引脚中断 / 模式匹配引擎引脚说明

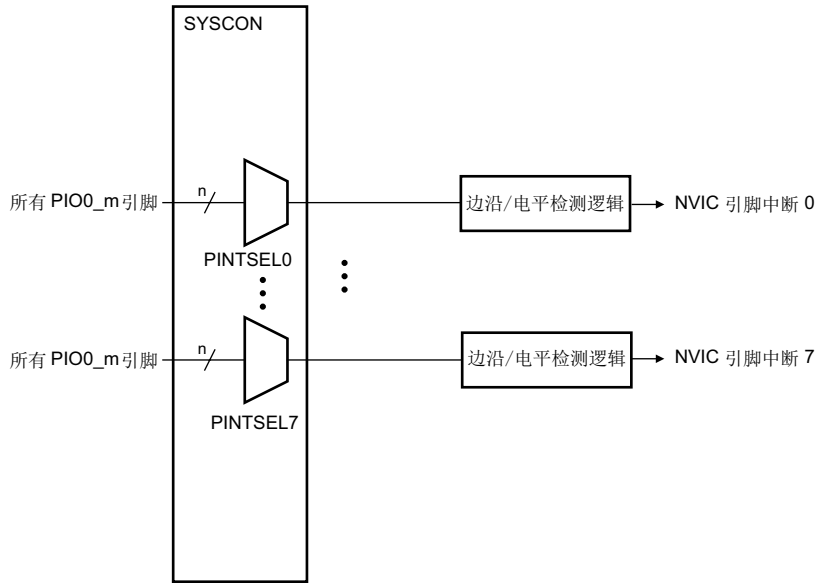
功能	方向	引脚	说明	SWM 寄存器	参考
GPIO_INT_BMAT	O	任意	GPIO 模式匹配输出	PINASSIGN8	表 105

8.5 简介

带可配置功能的引脚可作为模拟匹配引擎的外部中断或输入。可针对这些功能使用 SYSCON 模块中的 PINTSEL 寄存器配置最多 8 个引脚。

8.5.1 引脚中断

在系统控制模块中，最多可选择所有可用 GPIO 引脚中的 8 个引脚作为外部中断引脚（参见表 32）。外部中断引脚连接至 NVIC 的 8 个独立中断，并根据上升沿 / 下降沿或引脚上的输入电平创建。



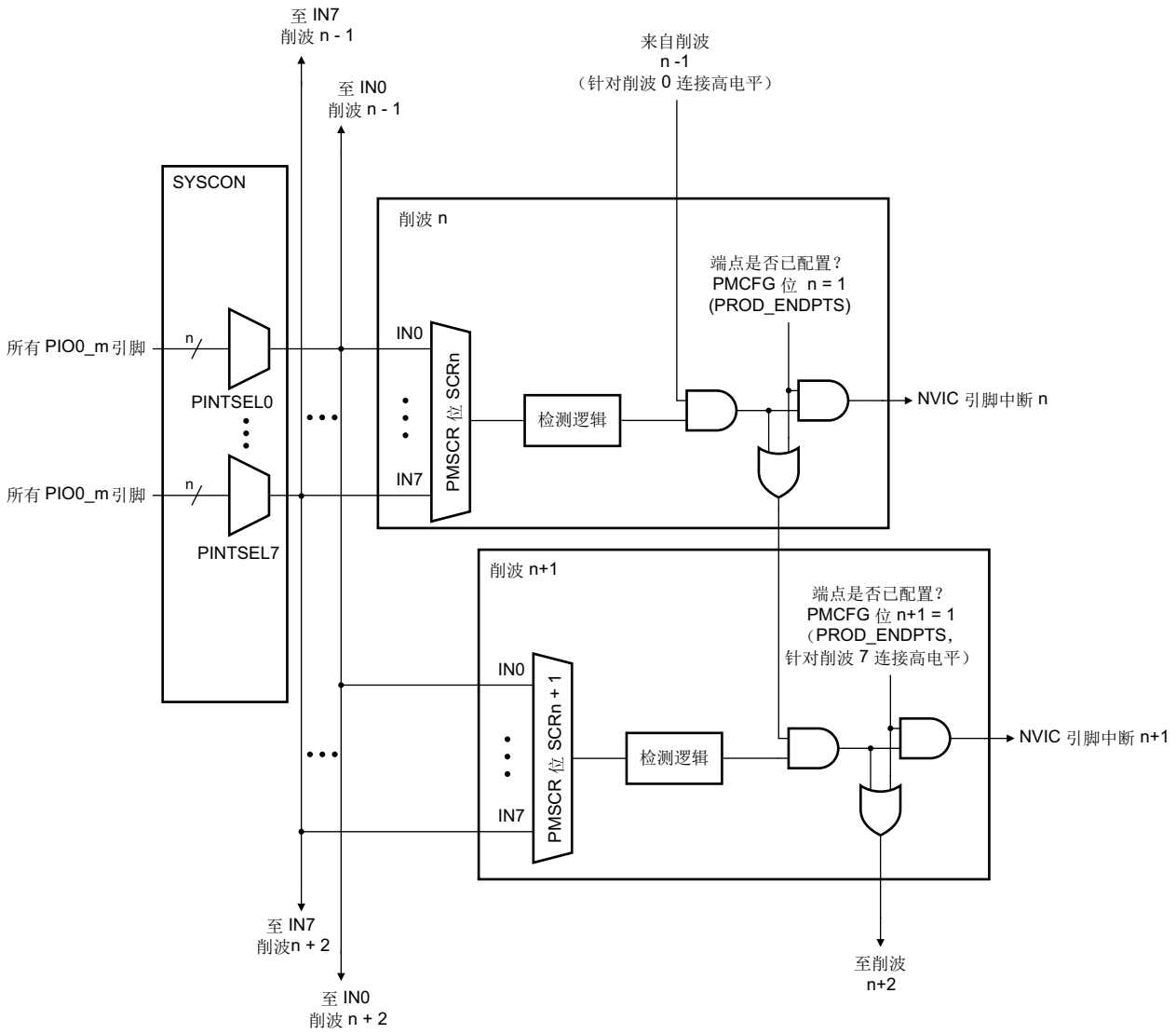
n = 6 （DIP8 封装）； n= 14 （TSSOP16 封装）； n = 18 （TSSOP/SOP20 封装）。

图 6. 引脚中断连接

8.5.2 模式匹配引擎

模式匹配特性允许根据针对 GPIO 引脚中断选择的同一组 8 个 GPIO 引脚构建复杂的布尔表达式。布尔表达式中的每一项都部署为模式匹配引擎的一个逻辑片。一个逻辑片由输入选择器和检测逻辑组成。逻辑片输入选择器从可用的 8 个输入中选择某个输入，每个输入均通过 PINTSEL 寄存器连接一个引脚。

检测逻辑连续监控选定输入，若输入满足检测条件则建立高电平输出。通过指定某个逻辑片为表达式的端点，某些项可合并为一个最小项。最小项被认为为真时，该逻辑片的引脚中断会被置位。



有关检测逻辑模块，请参见图 8。

图 7. 模式匹配引擎的连接

每个逻辑片上的检测逻辑可检测选定输入的下列事件：

- 存储器边沿（粘滞）：边沿检测机制清零后的任意时刻检测到的上升沿、下降沿或上升 / 下降沿。在模式匹配引擎检测逻辑被再次清零之前，输入都满足检测条件 （检测逻辑输出保持高电平）。
- 事件 （非粘滞）：每次检测到边沿 （上升沿或下降沿），该引脚的检测逻辑输出都会变为高电平。该位在一个时钟周期后会被清零，因此检测逻辑可检测另一个边沿。
- 电平：选定输入端上的高电平或低电平。

图 8 显示的是每个逻辑片的边沿检测逻辑的详情。

在一次验证边沿事件后，无论此时为上升沿或是下降沿，都可将粘滞事件与非粘滞事件相结合，创建一次引脚中断。

可创建一个时间窗口，使上升沿或下降沿在该时间窗口内能通过结合电平检测和事件检测创建一个引脚中断。有关详情，请参见章节 8.7.3。

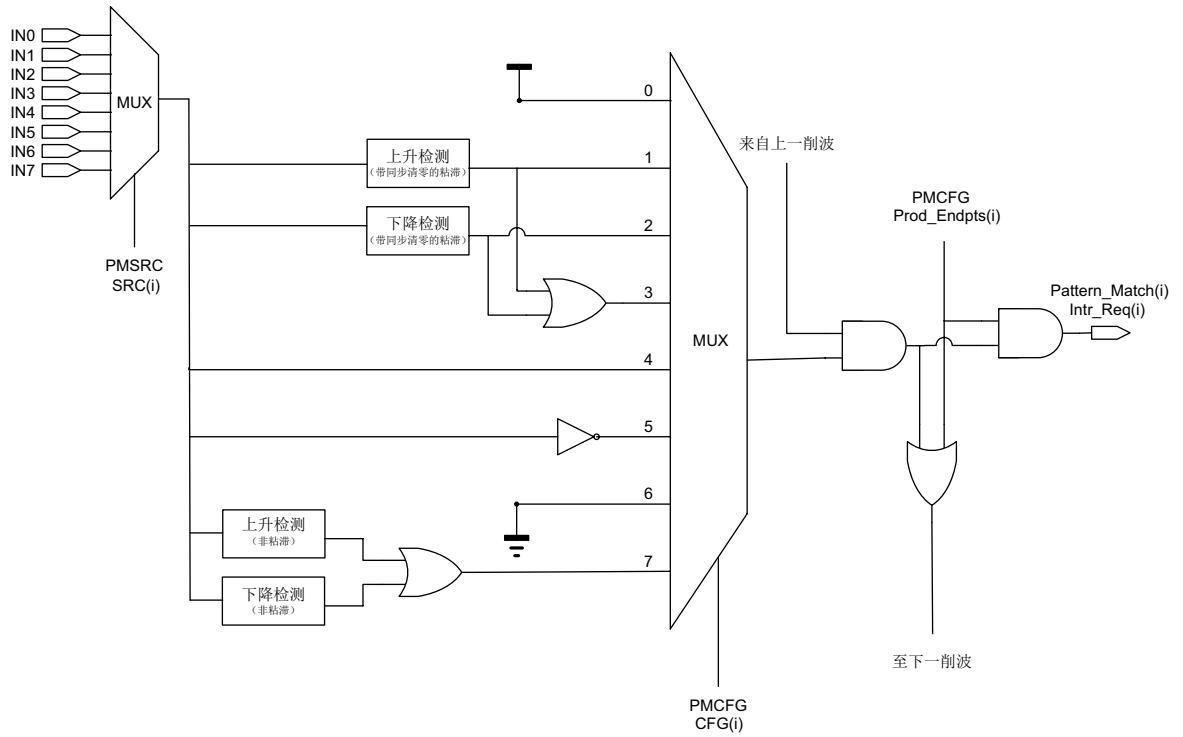


图 8. 带检测逻辑的模式匹配位片

8.5.2.1 模式匹配引擎的输入和输出

引脚和模式匹配引擎间的连接如图 7 所示。所有模式匹配引擎的输入均在 SYSCON 模块中选定，并且取决于不同的开关矩阵配置，可以是 GPIO 端口引脚或另一个引脚功能。

模式匹配逻辑可连续监控 8 个输入，并在指定布尔表达式的任意一个或多个最小项 （乘积项）匹配时产生中断。针对每一个最小项，都会产生一个独立的中断请求。

此外，当布尔表达式为真时 （例如：任意最小项匹配时），可使能模式匹配模块生成输出至 ARM 内核的接收事件 (RXEV)。

RXEV 输出同样被路由至 GPIO_INT_BMAT 引脚。 GPIO 模块因此能提供可部署多达 8 个输入和 1 个输出的基本可编程逻辑功能。

模式匹配功能利用相同的 8 条中断请求线路作为引脚中断，因此只要和中断产生有关，这两种特性便是互斥的。提供控制位是为了选择是否产生中断请求以响应标准引脚中断或模式匹配。需要注意的是，如果选定引脚中断，则发送至 CPU 的 RXEV 请求依然能被使能用于模式匹配。

注：模式匹配无法用于将器件从深度睡眠模式或掉电模式中唤醒。必须选定引脚中断，引脚才可用于唤醒。

8.5.2.2 布尔表达式

模式匹配模块由 8 个位片元素构成。经过编程的每个位片代表布尔表达式一个最小项 （乘积项）的一部分。一旦特定最小项匹配，则针对该最小项且与最后一个位片关联的中断请求将被置位。（参见位片原理图图 8）。

模式匹配功能可用于创建复杂的软件状态机。每个最小项 （及其相关专用中断）都代表不同的新状态转换事件。软件之后便可建立可转换出当前状态的一组新条件 （即新的布尔表达式）。

示例：

假设表达式：(IN0)~(IN1)(IN3)^ + (IN1)(IN2) + (IN0)~(IN3)~(IN4) 通过寄存器 PMSRC （表 92）和 PMCFG （表 93）指定。布尔表达式中的每一项 （(IN0)、~(IN1)、(IN3)^ 等）代表模式匹配引擎的每个位片。

- 在第一个最小项 （(IN0)~(IN1)(IN3)^）中，位片 0 监控输入 (IN0) 上的高电平，位片 1 监控输入 (IN1) 上的低电平，位片 2 监控输入 (IN3) 上的上升沿。如果检测到该组合，即如果全部三个项都为真，则与位片 2 (PININT2_IRQ) 有关的中断将被置位。
- 在第二个最小项 （(IN1)(IN2)）中，位片 3 监控输入 (IN1) 上的高电平，位片 4 监控输入 (IN2) 上的高电平。如果检测到该组合，则与位片 4 (PININT4_IRQ) 有关的中断将被置位。
- 在第三个最小项 （(IN0)~(IN3)~(IN4)）中，位片 5 监控输入 (IN0) 上的高电平，位片 6 监控输入 (IN3) 上的低电平，位片 7 监控输入 (IN4) 上的低电平。如果检测到该组合，则与位片 7 (PININT7_IRQ) 有关的中断将被置位。
- 所有三个最小项的 ORed 运算结果将置位到 CPU 和 GPIO_INT_BMAT 输出的 RXEV 请求。也就是说，只要三个最小项中的任意一项为真，输出即被置位。

相关链接：

章节 8.7.2

8.6 寄存器说明

表 80. 寄存器概述：引脚中断和模式匹配引擎 （基址：0xA000 4000）

名称	访问类型	地址偏移	说明	复位值	参考
ISEL	R/W	0x000	引脚中断模式寄存器	0	表 81
IENR	R/W	0x004	引脚中断电平或上升沿中断使能寄存器	0	表 82
SIENR	WO	0x008	引脚中断电平或上升沿中断设置寄存器	不适用	表 83
CIENR	WO	0x00C	引脚中断电平 （上升沿中断）清零寄存器	不适用	表 84

表 80. 寄存器概述：引脚中断和模式匹配引擎 （基址：0xA000 4000）*（续）*

名称	访问类型	地址偏移	说明	复位值	参考
IENF	R/W	0x010	引脚中断有效电平或下降沿中断使能寄存器	0	表 85
SIENF	WO	0x014	引脚中断有效电平或下降沿中断设置寄存器	不适用	表 86
CIENF	WO	0x018	引脚中断有效电平或下降沿中断清零寄存器	不适用	表 87
RISE	R/W	0x01C	引脚中断上升沿寄存器	0	表 88
FALL	R/W	0x020	引脚中断下降沿寄存器	0	表 89
IST	R/W	0x024	引脚中断状态寄存器	0	表 90
PMCTRL	R/W	0x028	模式匹配中断控制寄存器	0	表 91
PMSRC	R/W	0x02C	模式匹配中断位片源寄存器	0	表 92
PMCFG	R/W	0x030	模式匹配中断位片配置寄存器	0	表 93

8.6.1 引脚中断模式寄存器

对于 PINTSELn 寄存器（参见章节 4.6.27）中选择的每个引脚中断（共 8 个），ISEL 寄存器中的某个位可确定这些中断是边沿敏感还是电平敏感。

表 81. 引脚中断模式寄存器 （ISEL，地址 0xA000 4000）位说明

位	符号	说明	复位值	访问类型
7:0	PMODE	选择每个引脚中断的中断模式。位 n 可配置 PINTSELn 中所选的引脚中断。 0 = 边沿敏感 1 = 电平敏感	0	R/W
31:8	-	保留。	-	-

8.6.2 引脚中断电平或上升沿中断使能寄存器

对于 PINTSELn 寄存器 （参见章节 4.6.27）中所选的每个引脚中断 （共 8 个）， IENR 寄存器中的某个位可根据 ISEL 寄存器中配置的引脚中断模式使能中断：

- 如果引脚中断模式是边沿敏感 (PMODE = 0)，则使能上升沿中断。
- 如果引脚中断模式是电平敏感 (PMODE = 1)，则使能电平中断。IENF 寄存器可配置该中断的有效电平 （高电平或低电平）。

表 82. 引脚中断电平或上升沿中断使能寄存器 （IENR，地址 0xA000 4004）位说明

位	符号	说明	复位值	访问类型
7:0	ENRL	使能每个引脚中断的上升沿或电平中断。位 n 可配置 PINTSELn 中所选的引脚中断。 0 = 禁用上升沿或电平中断。 1 = 使能上升沿或电平中断。	0	R/W
31:8	-	保留。	-	-

8.6.3 引脚中断电平或上升沿中断设置寄存器

对于 PINTSELn 寄存器（参见章节 4.6.27）中所选的每个引脚中断（共 8 个），SIENR 寄存器中的某个位可根据 ISEL 寄存器中配置的引脚中断模式设置 IENR 寄存器中的对应位：

- 如果引脚中断模式是边沿敏感 (PMODE = 0)，则设置上升沿中断。
- 如果引脚中断模式是电平敏感 (PMODE = 1)，则设置电平中断。

表 83. 引脚中断电平或上升沿中断设置寄存器 （SIENR，地址 0xA000 4008）位说明

位	符号	说明	复位值	访问类型
7:0	SETENRL	将 1 写入该地址会设置 IENR 中的位，从而使能中断。位 n 可设置 IENR 寄存器中的位 n。 0 = 无操作。 1 = 使能上升沿或电平中断。	不适用	WO
31:8	-	保留。	-	-

8.6.4 引脚中断电平或上升沿中断清零寄存器

对于 PINTSELn 寄存器（参见章节 4.6.27）中所选的每个引脚中断（共 8 个），CIENR 寄存器中的某个位可根据 ISEL 寄存器中配置的引脚中断模式清零 IENR 寄存器中的对应位：

- 如果引脚中断模式是边沿敏感 (PMODE = 0)，则清零上升沿中断。
- 如果引脚中断模式是电平敏感 (PMODE = 1)，则清零电平中断。

表 84. 引脚中断电平或上升沿中断清零寄存器 （CIENR，地址 0xA000 400C）位说明

位	符号	说明	复位值	访问类型
7:0	CENRL	将 1 写入该地址可清零 IENR 中的位，从而禁用中断。位 n 可清零 IENR 寄存器中的位 n。 0 = 无操作。 1 = 禁用上升沿或电平中断。	不适用	WO
31:8	-	保留。	-	-

8.6.5 引脚中断有效电平或下降沿中断使能寄存器

对于 PINTSELn 寄存器（参见章节 4.6.27）中所选的每个引脚中断（共 8 个），IENF 寄存器中的某个位可根据 ISEL 寄存器中配置的引脚中断模式使能下降沿中断或配置电平敏感性：

- 如果引脚中断模式是边沿敏感 (PMODE = 0)，则使能下降沿中断。
- 如果引脚中断模式是电平敏感 (PMODE = 1)，则配置电平中断的有效电平 （高电平或低电平）。

表 85. 引脚中断有效电平或下降沿中断使能寄存器 （IENF，地址 0xA000 4010）位说明

位	符号	说明	复位值	访问类型
7:0	ENAF	使能每个引脚中断的下降沿中断或配置有效电平中断。位 n 可配置 PINTSELn 中所选的引脚中断。 0 = 禁用下降沿中断或将有效中断电平设置为低电平 1 = 使能下降沿中断或将有效中断电平设置为高电平	0	R/W
31:8	-	保留。	-	-

8.6.6 引脚中断有效电平或下降沿中断设置寄存器

对于 PINTSELn 寄存器（参见章节 4.6.27）中所选的每个引脚中断（共 8 个），SIENF 寄存器中的某个位可根据 ISEL 寄存器中配置的引脚中断模式设置 IENF 寄存器中的对应位：

- 如果引脚中断模式是边沿敏感 (PMODE = 0)，则设置下降沿中断。
- 如果引脚中断模式是电平敏感 (PMODE = 1)，则选择高电平有效中断。

表 86. 引脚中断有效电平或下降沿中断设置寄存器 （SIENF，地址 0xA000 4014）位说明

位	符号	说明	复位值	访问类型
7:0	SETENAF	将 1 写入该地址可设置 IENF 中的位，从而使能中断。位 n 可设置 IENF 寄存器中的位 n。 0 = 无操作。 1 = 选择高电平有效中断或使能下降沿中断。	不适用	WO
31:8	-	保留。	-	-

8.6.7 引脚中断有效电平或下降沿中断清零寄存器

对于 PINTSELn 寄存器（参见章节 4.6.27）中所选的每个引脚中断（共 8 个），CIENF 寄存器中的某个位可根据 ISEL 寄存器中配置的引脚中断模式设置 IENF 寄存器中的对应位：

- 如果引脚中断模式是边沿敏感 (PMODE = 0)，则清零下降沿中断。
- 如果引脚中断模式是电平敏感 (PMODE = 1)，则选择低电平有效中断。

表 87. 引脚中断有效电平或下降沿中断清零寄存器 （CIENF，地址 0xA000 4018）位说明

位	符号	说明	复位值	访问类型
7:0	CENAF	将 1 写入该地址可清零 IENF 中的位，从而禁用中断。位 n 可清零 IENF 寄存器中的位 n。 0 = 无操作。 1 = 选择低电平有效中断或禁用下降沿中断。	不适用	WO
31:8	-	保留。	-	-

8.6.8 引脚中断上升沿寄存器

该寄存器包含 1 表示 PINTSELn 寄存器 （参见章节 4.6.27）中所选的引脚中断上已检测到上升沿。将 1 写入该寄存器可清零上升沿检测。该寄存器中的 1 会针对上升沿中断使能的引脚发送一个中断请求。针对 PINTSELn 寄存器中选择的所有引脚检测所有边沿，无论这些引脚是否是中断使能。

表 88. 引脚中断上升沿寄存器 （RISE，地址 0xA000 401C）位说明

位	符号	说明	复位值	访问类型
7:0	RDET	上升沿检测。位 n 可检测 PINTSELn 中所选引脚的上升沿。 读 0：复位或上一次将 1 写入该位后，未在该引脚上检测到上升沿。 写 0：无操作。 读 1：复位或上一次将 1 写入该位后，已检测到上升沿。 写 1：清零该引脚的上升沿检测。	0	R/W
31:8	-	保留。	-	-

8.6.9 引脚中断下降沿寄存器

该寄存器包含 1 表示 PINTSELn 寄存器 （参见章节 4.6.27）中所选的引脚中断上已检测到下降沿。将 1 写入寄存器可清零下降沿检测。该寄存器中的 1 会针对降沿中断使能的引脚发送一个中断请求。针对 PINTSELn 寄存器中选择的所有引脚检测所有边沿，无论这些引脚是否是中断使能。

表 89. 引脚中断下降沿寄存器 （FALL，地址 0xA000 4020）位说明

位	符号	说明	复位值	访问类型
7:0	FDET	下降沿检测。位 n 可检测 PINTSELn 中所选引脚的下降沿。 读 0：复位或上一次将写入该位后，未在该引脚上检测到下降沿。 写 0：无操作。 读 1：复位或上一次将 1 写入该位后，已检测到下降沿。 写 1：清零该引脚的下降沿检测。	0	R/W
31:8	-	保留。	-	-

8.6.10 引脚中断状态寄存器

引脚中断当前正在请求中断时，读该寄存器会返回 1。对于在中断选择寄存器中确定为边沿敏感的引脚，将 1 写入该寄存器中会清零该引脚的上升和下降沿检测。对于电平敏感引脚，写 1 会反转有效电平寄存器中的对应位，从而切换引脚的有效电平。

表 90. 引脚中断状态寄存器 （IST，地址 0xA000 4024）位说明

位	符号	说明	复位值	访问类型
7:0	PSTAT	引脚中断状态。位 n 返回状态、清零边沿中断，或反转 PINTSELn 所选引脚的有效电平。 读 0：该中断引脚无正在请求的中断。 写 0：无操作。 读 1：该中断引脚有正在请求的中断。 写 1 （边沿敏感）：清零该引脚的上升和下降沿检测。 写 1（电平敏感）：切换（IENF 寄存器中）该引脚的有效电平。	0	R/W
31:8	-	保留。	-	-

8.6.11 模式匹配中断控制寄存器

模式匹配控制寄存器含有两位，一位用于选择模式匹配中断生成 （与引脚中断共享同一条中断请求线路相反），另一位用于使能 RXEV 输出至 CPU。该寄存器还允许读取任何模式匹配的当前状态。

如果未使用模式匹配特性 （无论针对中断生成还是 RXEV 置位），该寄存器的位 SEL_PMATCH 和 ENA_RXEV 应保留为 0 以降低功耗。

注：对该寄存器进行写操作以使能 （或再次使能）模式匹配功能前，应先配置 PMSRC 和 PMCFG 寄存器内的模式匹配。由于使能了该特性，因此将不再产生杂散中断。

表 91. 模式匹配中断控制寄存器 （PMCTRL，地址 0xA000 4028）位说明

位	符号	值	说明	复位值
0	SEL_PMATCH		指定 8 个引脚中断由引脚中断功能控制还是由模式匹配功能控制。	0
		0	引脚中断。中断被驱动以响应标准引脚中断功能	
		1	模式匹配。中断被驱动以响应模式匹配。	

表 91. 模式匹配中断控制寄存器 （PMCTRL，地址 0xA000 4028）位说明 *（续）*

位	符号	值	说明	复位值
1	ENA_RXEV		指定的布尔表达式被认为为真时，使能 RXEV 至 ARM CPU 和 / 或 GPIO 输出的输出。	0
		0	禁用。 RXEV 至 CPU 的输出禁用。	
		1	使能。 RXEV 至 CPU 的输出使能。	
23:2	-		保留。不要将 1 写入未使用的位。	0
31:24	PMAT	-	该字段显示模式匹配的当前状态。该字段中任意位为 1 表示相应的乘积项与相适应输出的当前状态匹配。	0x0

8.6.12 模式匹配中断位片源寄存器

位片源寄存器指定所有 8 个模式匹配的位片。

在 SYSCON 模块的引脚中断选择寄存器中选择所有可能的 8 个输入。参见章节 4.6.27。输入 0 对应于 PINTSEL0 寄存器选定的引脚，输入 1 对应于 PINTSEL1 寄存器选定的引脚，以此类推。

注：无论是将任何值写入 PMCFG 寄存器或 PMSRC 寄存器还是禁用模式匹配特性 （通过清零 PMCTRL 寄存器中的 SEL_PMATCH 和 ENA_RXEV 位）都将擦除所有边沿检测历史数据。

表 92. 模式匹配位片源寄存器 （PMSRC，地址 0xA000 402C）位说明

位	符号	值	说明	复位值
7:0	保留		软件不应将 1 写入未使用的位。	0
10:8	SRC0		选择位片 0 的输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器中选定的引脚作为位片 0 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器中选定的引脚作为位片 0 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器中选定的引脚作为位片 0 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器中选定的引脚作为位片 0 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器中选定的引脚作为位片 0 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器中选定的引脚作为位片 0 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器中选定的引脚作为位片 0 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器中选定的引脚作为位片 0 的源。	
13:11	SRC1		选择位片 1 的输入源	0
		0x0	输入 0。选择引脚中断输入 0 作为位片 1 的源。	
		0x1	输入 0。选择 PINTSEL0 寄存器中选定的引脚作为位片 0 的源。	
		0x2	输入 1。选择 PINTSEL1 寄存器中选定的引脚作为位片 0 的源。	
		0x3	输入 2。选择 PINTSEL2 寄存器中选定的引脚作为位片 0 的源。	
		0x4	输入 3。选择 PINTSEL3 寄存器中选定的引脚作为位片 0 的源。	
		0x5	输入 4。选择 PINTSEL4 寄存器中选定的引脚作为位片 0 的源。	
		0x6	输入 5。选择 PINTSEL5 寄存器中选定的引脚作为位片 0 的源。	
		0x7	输入 6。选择 PINTSEL6 寄存器中选定的引脚作为位片 0 的源。	

表 92. 模式匹配位片源寄存器 （PMSRC，地址 0xA000 402C）位说明 *（续）*

位	符号	值	说明	复位值
16:14	SRC2		为位片 2 选择输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器中选定的引脚作为位片 0 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器中选定的引脚作为位片 0 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器中选定的引脚作为位片 0 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器中选定的引脚作为位片 0 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器中选定的引脚作为位片 0 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器中选定的引脚作为位片 0 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器中选定的引脚作为位片 0 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器中选定的引脚作为位片 0 的源。	
19:17	SRC3		为位片 3 选择输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器中选定的引脚作为位片 0 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器中选定的引脚作为位片 0 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器中选定的引脚作为位片 0 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器中选定的引脚作为位片 0 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器中选定的引脚作为位片 0 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器中选定的引脚作为位片 0 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器中选定的引脚作为位片 0 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器中选定的引脚作为位片 0 的源。	
22:20	SRC4		为位片 4 选择输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器中选定的引脚作为位片 0 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器中选定的引脚作为位片 0 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器中选定的引脚作为位片 0 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器中选定的引脚作为位片 0 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器中选定的引脚作为位片 0 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器中选定的引脚作为位片 0 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器中选定的引脚作为位片 0 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器中选定的引脚作为位片 0 的源。	
25:23	SRC5		为位片 5 选择输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器中选定的引脚作为位片 0 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器中选定的引脚作为位片 0 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器中选定的引脚作为位片 0 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器中选定的引脚作为位片 0 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器中选定的引脚作为位片 0 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器中选定的引脚作为位片 0 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器中选定的引脚作为位片 0 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器中选定的引脚作为位片 0 的源。	

表 92. 模式匹配位片源寄存器 （PMSRC，地址 0xA000 402C）位说明 *（续）*

位	符号	值	说明	复位值
28:26	SRC6		为位片 6 选择输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器中选定的引脚作为位片 0 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器中选定的引脚作为位片 0 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器中选定的引脚作为位片 0 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器中选定的引脚作为位片 0 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器中选定的引脚作为位片 0 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器中选定的引脚作为位片 0 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器中选定的引脚作为位片 0 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器中选定的引脚作为位片 0 的源。	
31:29	SRC7		为位片 7 选择输入源	0
		0x0	输入 0。选择 PINTSEL0 寄存器中选定的引脚作为位片 0 的源。	
		0x1	输入 1。选择 PINTSEL1 寄存器中选定的引脚作为位片 0 的源。	
		0x2	输入 2。选择 PINTSEL2 寄存器中选定的引脚作为位片 0 的源。	
		0x3	输入 3。选择 PINTSEL3 寄存器中选定的引脚作为位片 0 的源。	
		0x4	输入 4。选择 PINTSEL4 寄存器中选定的引脚作为位片 0 的源。	
		0x5	输入 5。选择 PINTSEL5 寄存器中选定的引脚作为位片 0 的源。	
		0x6	输入 6。选择 PINTSEL6 寄存器中选定的引脚作为位片 0 的源。	
		0x7	输入 7。选择 PINTSEL7 寄存器中选定的引脚作为位片 0 的源。	

8.6.13 模式匹配中断位片配置寄存器

位片配置寄存器可配置检测逻辑，并包含针对每个位片从 8 个可选条件中选择的位，而该位的内容使位片得以提供模式匹配的信息。该寄存器的 7 个最低有效位可指定作为布尔表达式中乘积项端点的位片 （例如， OR 项在表达式中的哪个位置插入）。

每个输入可以有两种类型的边沿检测：

- 粘滞：边沿检测机制清零后的任意时刻检测到的上升沿、下降沿或上升 / 下降沿。在模式匹配引擎检测逻辑被再次清零之前，输入都满足检测条件（检测逻辑输出保持高电平）。
- 非粘滞：每次检测到边沿 （上升沿或下降沿），该引脚的检测逻辑输出都会变为高电平。该位在一个时钟周期后会被清零，因此该边沿检测逻辑可检测另一个边沿。

注：要清零模式匹配引擎检测逻辑，可将任意数值写入 PMCFG 寄存器或 PMSRC 寄存器，或者禁用模式匹配特性（清零 PMCTRL 寄存器中的 SEL_PMATCH 和 ENA_RXEV 位）。这将擦除所有边沿检测历史数据。

要选择某个逻辑片是否标记布尔表达式中最小项的最终元素，可将 1 写入相应的 PROD_ENPTSn 位。设置某一项为最终元素有两个效果：

1. 一旦检测到该乘积项的匹配项，即置位与该位片有关的中断请求。
2. 下一个位片将在布尔表达式中开始一个全新、独立的乘积项（例如，OR 将在布尔表达式中插入，后接由该位片控制的元素）。

表 93. 模式匹配位片配置寄存器 （PMCFG，地址 0xA000 4030）位说明

位	符号	值	说明	复位值
0	PROD_ENDPTS0		确定片 0 是否为端点。	0
		0	无效。片 0 不是端点。	
		1	端点。片 0 是乘积项的端点 （最小项）。如果最小项被认为为真，则会发起 NVIC 中的引脚中断 0。	
1	PROD_ENDPTS1		确定片 1 是否为端点。	0
		0	无效。片 1 不是端点。	
		1	端点。片 1 是乘积项的端点 （最小项）。如果最小项被认为为真，则会发起 NVIC 中的引脚中断 1。	
2	PROD_ENDPTS2		确定片 2 是否为端点。	0
		0	无效。片 2 不是端点。	
		1	端点。片 2 是乘积项的端点 （最小项）。如果最小项被认为为真，则会发起 NVIC 中的引脚中断 2。	
3	PROD_ENDPTS3		确定片 3 是否为端点。	0
		0	无效。片 3 不是端点。	
		1	端点。片 3 是乘积项的端点 （最小项）。如果最小项被认为为真，则会发起 NVIC 中的引脚中断 3。	
4	PROD_ENDPTS4		确定片 4 是否为端点。	0
		0	无效。片 4 不是端点。	
		1	端点。片 4 是乘积项的端点 （最小项）。如果最小项被认为为真，则会发起 NVIC 中的引脚中断 4。	
5	PROD_ENDPTS5		确定片 5 是否为端点。	0
		0	无效。片 5 不是端点。	
		1	端点。片 5 是乘积项的端点 （最小项）。如果最小项被认为为真，则会发起 NVIC 中的引脚中断 5。	
6	PROD_ENDPTS6		确定片 6 是否为端点。	0
		0	无效。片 6 不是端点。	
		1	端点。片 6 是乘积项的端点 （最小项）。如果最小项被认为为真，则会发起 NVIC 中的引脚中断 6。	
7	-		保留。位片 7 自动成为乘积端点。	0
10:8	CFG0		指定位片 0 的匹配影响条件。	0b000
		0x0	恒定高电平。该位片始终影响乘积项匹配。	
		0x1	粘滞上升沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x2	粘滞下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生下降沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x3	粘滞上升或下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿或下降沿事件，则会匹配。仅当对 PMCFG 或 PMSRC 寄存器进行写操作时，该位才被清零。	
		0x4	高电平。 PMSRC 寄存器中该位片指定的输入端上存在高电平时，则匹配 （就该位片而言）。	
		0x5	低电平。指定输入端上存在低电平时，则匹配。	
		0x6	恒定 0 电平。该位片从未影响匹配 （应当用于禁用所有未使用的位片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时匹配 —— 例如，指定输入端首先检测到上升或下降沿 （为数值 0x3 的非粘滞版本）。该位在一个时钟周期后会被清零。	

表 93. 模式匹配位片配置寄存器 （PMCFG，地址 0xA000 4030）位说明 *（续）*

位	符号	值	说明	复位值
13:11	CFG1		指定位片 1 的匹配影响条件。	0b000
		0x0	恒定高电平。该位片始终影响乘积项匹配。	
		0x1	粘滞上升沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x2	粘滞下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生下降沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x3	粘滞上升或下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿或下降沿事件，则会匹配。仅当对 PMCFG 或 PMSRC 寄存器进行写操作时，该位才被清零。	
		0x4	高电平。 PMSRC 寄存器中该位片指定的输入端上存在高电平时，则匹配 （就该位片而言）。	
		0x5	低电平。指定输入端上存在低电平时，则匹配。	
		0x6	恒定 0 电平。该位片从未影响匹配 （应当用于禁用所有未使用的位片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时匹配 —— 例如，指定输入端首先检测到上升或下降沿 （为数值 0x3 的非粘滞版本）。该位在一个时钟周期后会被清零。	
16:14	CFG2		指定位片 2 的匹配影响条件。	0b000
		0x0	恒定高电平。该位片始终影响乘积项匹配。	
		0x1	粘滞上升沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x2	粘滞下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生下降沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x3	粘滞上升或下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿或下降沿事件，则会匹配。仅当对 PMCFG 或 PMSRC 寄存器进行写操作时，该位才被清零。	
		0x4	高电平。 PMSRC 寄存器中该位片指定的输入端上存在高电平时，则匹配 （就该位片而言）。	
		0x5	低电平。指定输入端上存在低电平时，则匹配。	
		0x6	恒定 0 电平。该位片从未影响匹配 （应当用于禁用所有未使用的位片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时匹配 —— 例如，指定输入端首先检测到上升或下降沿 （为数值 0x3 的非粘滞版本）。该位在一个时钟周期后会被清零。	
19:17	CFG3		指定位片 3 的匹配影响条件。	0b000
		0x0	恒定高电平。该位片始终影响乘积项匹配。	
		0x1	粘滞上升沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x2	粘滞下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生下降沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x3	粘滞上升或下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿或下降沿事件，则会匹配。仅当对 PMCFG 或 PMSRC 寄存器进行写操作时，该位才被清零。	
		0x4	高电平。 PMSRC 寄存器中该位片指定的输入端上存在高电平时，则匹配 （就该位片而言）。	
		0x5	低电平。指定输入端上存在低电平时，则匹配。	
		0x6	恒定 0 电平。该位片从未影响匹配 （应当用于禁用所有未使用的位片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时匹配 —— 例如，指定输入端首先检测到上升或下降沿 （为数值 0x3 的非粘滞版本）。该位在一个时钟周期后会被清零。	

表 93. 模式匹配位片配置寄存器 （PMCFG，地址 0xA000 4030）位说明 *（续）*

位	符号	值	说明	复位值
22:20	CFG4		指定位片 4 的匹配影响条件。	0b000
		0x0	恒定高电平。该位片始终影响乘积项匹配。	
		0x1	粘滞上升沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x2	粘滞下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生下降沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x3	粘滞上升或下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿或下降沿事件，则会匹配。仅当对 PMCFG 或 PMSRC 寄存器进行写操作时，该位才被清零。	
		0x4	高电平。 PMSRC 寄存器中该位片指定的输入端上存在高电平时，则匹配 （就该位片而言）。	
		0x5	低电平。指定输入端上存在低电平时，则匹配。	
		0x6	恒定 0 电平。该位片从未影响匹配 （应当用于禁用所有未使用的位片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时匹配 —— 例如，指定输入端首先检测到上升或下降沿 （为数值 0x3 的非粘滞版本）。该位在一个时钟周期后会被清零。	
25:23	CFG5		指定位片 5 的匹配影响条件。	0b000
		0x0	恒定高电平。该位片始终影响乘积项匹配。	
		0x1	粘滞上升沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x2	粘滞下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生下降沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x3	粘滞上升或下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿或下降沿事件，则会匹配。仅当对 PMCFG 或 PMSRC 寄存器进行写操作时，该位才被清零。	
		0x4	高电平。 PMSRC 寄存器中该位片指定的输入端上存在高电平时，则匹配 （就该位片而言）。	
		0x5	低电平。指定输入端上存在低电平时，则匹配。	
		0x6	恒定 0 电平。该位片从未影响匹配 （应当用于禁用所有未使用的位片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时匹配 —— 例如，指定输入端首先检测到上升或下降沿 （为数值 0x3 的非粘滞版本）。该位在一个时钟周期后会被清零。	
28:26	CFG6		指定位片 6 的匹配影响条件。	0b000
		0x0	恒定高电平。该位片始终影响乘积项匹配。	
		0x1	粘滞上升沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x2	粘滞下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生下降沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x3	粘滞上升或下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿或下降沿事件，则会匹配。仅当对 PMCFG 或 PMSRC 寄存器进行写操作时，该位才被清零。	
		0x4	高电平。 PMSRC 寄存器中该位片指定的输入端上存在高电平时，则匹配 （就该位片而言）。	
		0x5	低电平。指定输入端上存在低电平时，则匹配。	
		0x6	恒定 0 电平。该位片从未影响匹配 （应当用于禁用所有未使用的位片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时匹配 —— 例如，指定输入端首先检测到上升或下降沿 （为数值 0x3 的非粘滞版本）。该位在一个时钟周期后会被清零。	

表 93. 模式匹配位片配置寄存器 （PMCFG，地址 0xA000 4030）位说明 *（续）*

位	符号	值	说明	复位值
31:29	CFG7		指定位片 7 的匹配影响条件。	0b000
		0x0	恒定高电平。该位片始终影响乘积项匹配。	
		0x1	粘滞上升沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x2	粘滞下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生下降沿事件，则会匹配。仅当对PMCFG或PMSRC寄存器进行写操作时，该位才被清零。	
		0x3	粘滞上升或下降沿。由于上一次清零了该位片的边沿检测，因此，如果指定输入端上发生上升沿或下降沿事件，则会匹配。仅当对 PMCFG 或 PMSRC 寄存器进行写操作时，该位才被清零。	
		0x4	高电平。 PMSRC 寄存器中该位片指定的输入端上存在高电平时，则匹配 （就该位片而言）。	
		0x5	低电平。指定输入端上存在低电平时，则匹配。	
		0x6	恒定 0 电平。该位片从未影响匹配 （应当用于禁用所有未使用的位片）。	
		0x7	事件。非粘滞上升或下降沿。事件发生时匹配 —— 例如，指定输入端首先检测到上升或下降沿 （为数值 0x3 的非粘滞版本）。该位在一个时钟周期后会被清零。	

8.7 功能说明

8.7.1 引脚中断

在该中断机制中，引脚中断选择寄存器 (PINTSEL0-7) 将多达 8 个引脚确认为中断源。所有引脚中断模块中的寄存器均包含 8 个位，对应 PINTSEL0-7 寄存器调用的各个引脚。 ISEL 寄存器可确定每个中断引脚是边沿敏感还是电平敏感。 RISE 寄存器和 FALL 寄存器可检测每个中断引脚的边沿，其写操作可以清零 （和设置）边沿检测。 IST 寄存器指示每个中断引脚当前是否在请求中断，也可通过写入该寄存器而清零中断。

其他引脚中断寄存器对边沿敏感和电平敏感引脚起到不同的作用，如表 94 中所述。

表 94. 用于边沿敏感和电平敏感引脚的引脚中断寄存器

名称	边沿敏感功能	电平敏感功能
IENR	使能上升沿中断。	使能电平中断。
SIENR	写入使能上升沿中断。	写入使能电平中断。
CIENR	写入禁用上升沿中断。	写入禁用电平中断。
IENF	使能下降沿中断。	选择有效电平。
SIENF	写入使能下降沿中断。	写入选择高电平有效。
CIENF	写入禁用下降沿中断。	写入选择低电平有效。

8.7.2 模式匹配引擎示例

假设用于匹配的所需布尔模式为：
(IN1) + (IN1 * IN2) + (~IN2 * ~IN3 * IN6fe) + (IN5 * IN7ev)

其中：

IN6fe = 输入 6 上的 （粘滞）下降沿

IN7ev = 输入 7 上的 （非粘滞）事件 （上升或下降沿）

如上所示表达式的每一项都通过一个位片控制。要指定该表达式，可通过如下方式设定模式匹配位片源并配置寄存器字段：

- PMSRC 寄存器 （表 92）：
  - 由于位片 5 将被用于输入 6 上的粘滞事件，可将 1 写入 SRC5 位以清零任何位片 5 上预留的边沿检测数据。
  - SRC0：001 - 位片 0 选择输入 1
  - SRC1：001 - 位片 1 选择输入 1
  - SRC2：010 - 位片 2 选择输入 2
  - SRC3：010 - 位片 3 选择输入 2
  - SRC4：011 - 位片 4 选择输入 3
  - SRC5：110 - 位片 5 选择输入 6
  - SRC6：101 - 位片 6 选择输入 5
  - SRC7：111 - 位片 7 选择输入 7
- PMCFG 寄存器 （表 93）：
  - PROD_ENDPTS0 = 1
  - PROD_ENDPTS02 = 1
  - PROD_ENDPTS5 = 1
  - 所有其他片都不是乘积项端点，并且它们的 PROD_ENDPTS 位等于 0。片 7 始终是乘积项端点，并且不存在相关的寄存器位。
  - = 0100101 - 位片 0、2、5 和 7 为乘积项端点。（位片 7 默认为端点 - 无相关寄存器位）。
  - CFG0：000 - 位片 0 指定输入 （输入 1）上为高电平
  - CFG1：000 - 位片 1 指定输入 （输入 1）上为高电平
  - CFG2：000 - 位片 2 指定输入 （输入 2）上为高电平
  - CFG3：101 - 位片 3 指定输入 （输入 2）上为低电平
  - CFG4：101 - 位片 4 指定输入 （输入 3）上为低电平
  - CFG5：010 - 位片 5 指定输入 （输入 6）上的 （粘滞）下降沿
  - CFG6：000 - 位片 6 指定输入 （输入 5）上为高电平
  - CFG7：111 - 位片 7 指定输入 （输入 7）上的事件 （任意边沿，非粘滞）
- PMCTRL 寄存器 （表 91）：
  - 位 0：设置该位将选定模式匹配，使其在普通引脚中断机制发挥作用时产生引脚中断。

    本例中，当第一个乘积项检测到匹配时，引脚中断 0 将被置位 （此处仅为输出 1 高电平）。

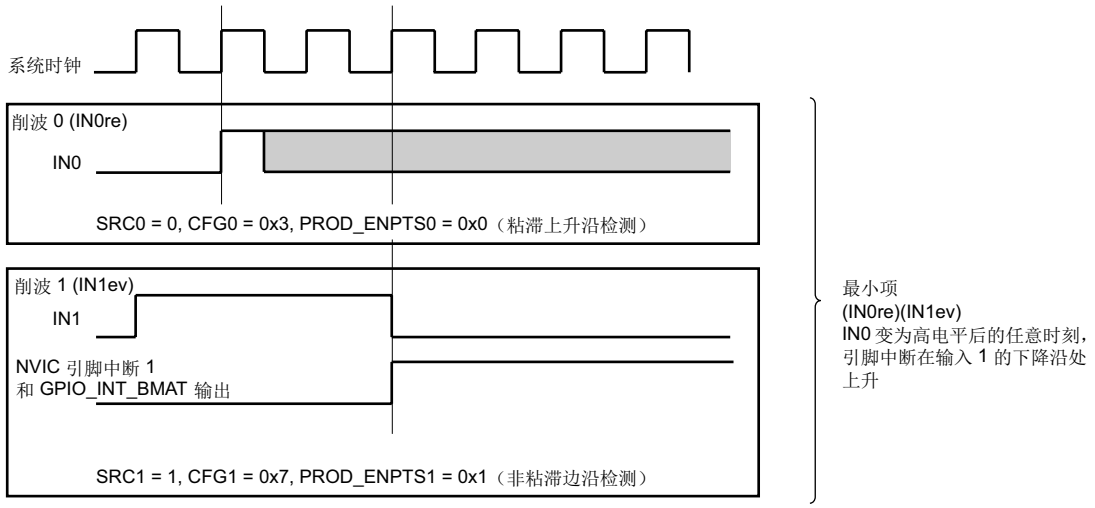
    引脚中断 2 将被置位以响应第二个乘积项的匹配。

    引脚中断 5 在第三个乘积项匹配时将被置位。

    引脚中断 7 在最后一项匹配时将被置位。
  - 位 1：一旦表达式的任意乘积项发生匹配，则设置该位将导致 RxEv 向 ARM CPU 发送置位信号。否则，将不使用 RXEV 线路。
  - 位 31:24：在任意给定的时间内，若相应的乘积项匹配，则位 0、2、5 和 / 或 7 可能为高电平。

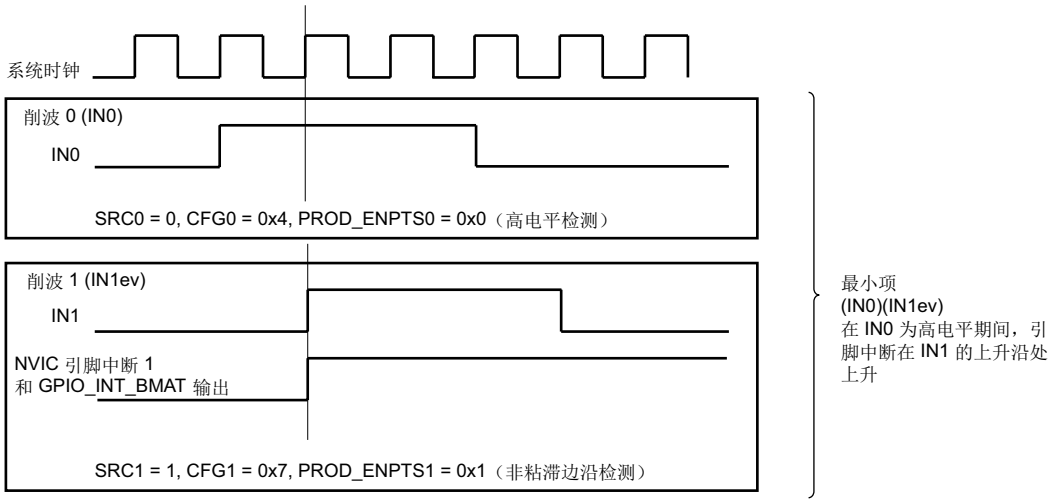
– 其余位将始终为低电平。

8.7.3 模式匹配引擎边沿检测示例



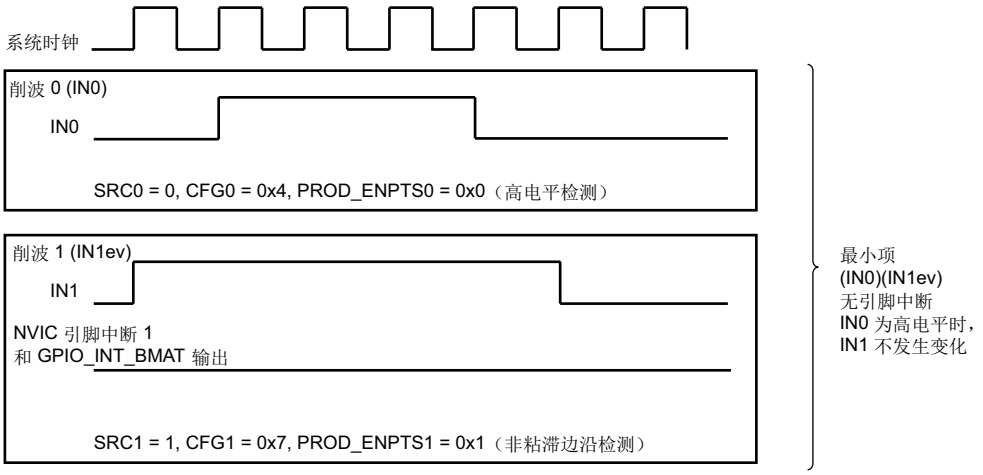
图中仅显示了模式匹配功能，未显示精确时序。为简便起见，输入 (INn) 显示为与系统时钟同步。

图 9. 模式匹配引擎示例：粘滞边沿检测



图中仅显示了模式匹配功能，未显示精确时序。为简便起见，输入 (INn) 显示为与系统时钟同步。

图 10. 模式匹配引擎示例：窗口化非粘滞边沿检测评价为真



图中仅显示了模式匹配功能，未显示精确时序。为简便起见，输入 (INn) 显示为与系统时钟同步。

图 11. 模式匹配引擎示例：窗口化非粘滞边沿检测评价为假

9.1 本章导读

所有LPC800器件的开关矩阵都是一样的。仅器件LPC812M101FDH20和LPC812M101FDH16才提供 USART2 功能和 SPI1 功能，相应矩阵开关位在所有其他器件中均予以保留。

9.2 特性

- 可将数字外设功能灵活分配给引脚
- 可使能 / 禁用模拟功能

9.3 基本配置

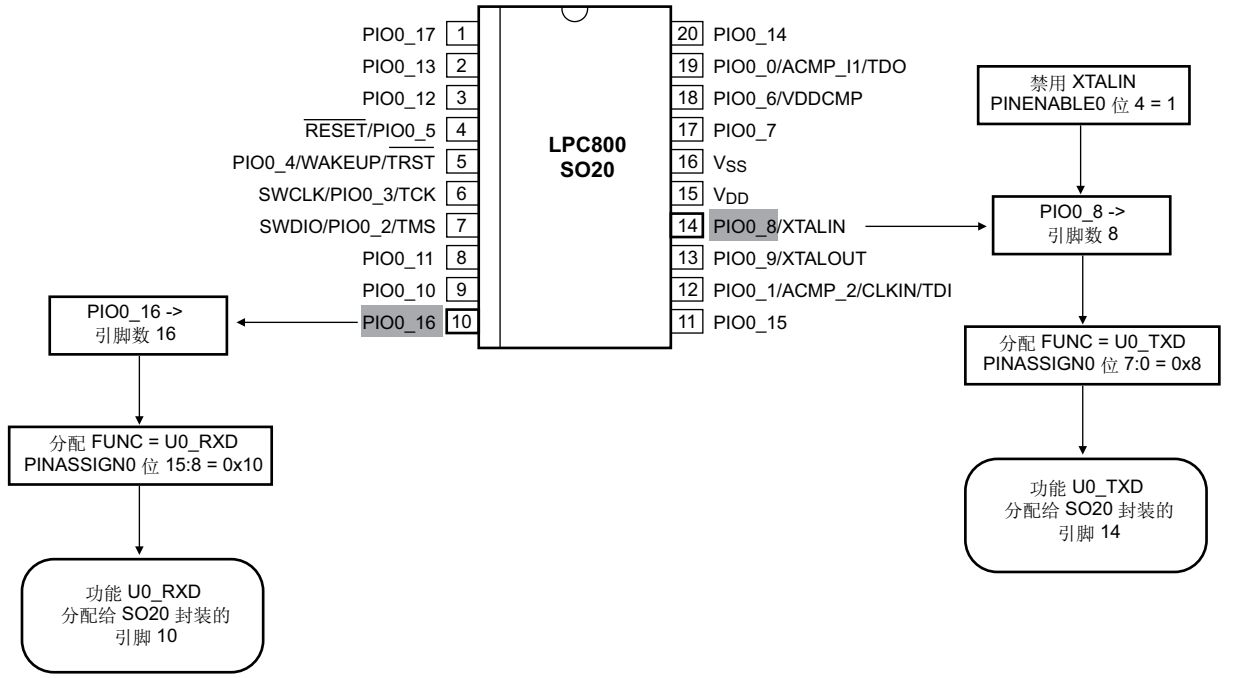
一旦完成配置，则开关矩阵无需时钟即可工作。系统时钟仅用于写入引脚分配寄存器，或从该寄存器中读取数据。开关矩阵完成配置后，在 SYSAHBCLKCTRL 寄存器中可禁用开关矩阵模块的时钟。

激活某个外设或使能其中断之前，使用开关矩阵将外设连接至外部引脚。

引导加载程序将 SWD 功能分配给 PIO0_2 和 PIO0_3。如果用户代码通过开关矩阵禁用 SWD 功能以便为引脚分配其他功能，则 SWD 端口禁用。

注：出于利用开关矩阵编程引脚功能的需要，除电源和接地引脚外的所有引脚均采用其 GPIO 端口引脚编号以独立于封装的方式进行标识。

9.3.1 将内部信号与封装引脚相连



标识为通过默认 GPIO 端口引脚对开关矩阵进行编程的引脚。

图 12. 示例：将 U0_RXD 和 U0_TXD 功能连接到 SO20 封装上的引脚 10 和引脚 14

利用引脚分配寄存器，开关矩阵可将可转移功能表中列出的所有内部信号与外部封装引脚相连。外部引脚由其默认 GPIO 引脚编号 PIO0_n 标识。按照以下步骤将内部信号 FUNC 连接到外部引脚。可转移功能的一个例子即 UART 发送信号 TXD：

1. 在表 95 的可转移功能列表或数据手册中，可查找功能 FUNC。
2. 参考 LPC800 数据手册，决定哪个 LPC800 封装引脚 x 与 FUNC 相连。
3. 使用引脚说明表，查找分配给封装引脚 x 的默认 GPIO 功能 PIO0_n。m 代表引脚编号。
4. 定位在开关矩阵寄存器说明中功能 FUNC 的引脚分配寄存器。
5. 在 PINENABLE0 寄存器中禁用 PIO0_n 引脚上的所有特殊功能。
6. 将引脚编号 n 编入分配给 FUNC 的位。

FUNC 现已分配给封装引脚 x。

9.3.2 使能模拟输入或其他特殊功能

开关矩阵使能仅可分配至一个引脚的功能。例如，模拟输入、所有 GPIO 引脚和调试 SWD 引脚。

- 如需将 GPIO 引脚分配给任意 LPC800 封装上的某个引脚，则在 PINENABLE0 寄存器中禁用该引脚的任何特殊功能，并且不要为其分配任何可转移功能。

  默认情况下，除引脚 PIO0_2、引脚 PIO0_3 和引脚 PIO0_5 外的所有引脚均分配给 GPIO。
- 对于所有未在可转移功能表中列出的功能，可采取下列步骤：

a. 确定该功能在数据手册引脚说明表中的位置。这样做可以显示该功能的封装引脚。

b. 在 PINENABLE0 寄存器中使能该功能。该引脚上的所有其他可用功能现在都会被禁用。

9.4 简介

开关矩阵将内部信号 （功能）连接到外部引脚。功能是来自 （或输出到）器件封装的单一引脚上的信号，或来自 （或输出到）片内外设时钟的信号。功能有：GPIO、UART 发送输出 (TXD)、时钟输出 CLKOUT 等。许多外设都提供多个功能，必须将它们分配到外部引脚上。

开关矩阵还使能输出驱动器，用于输出数字功能。每个引脚的输入和输出电气引脚特性（内部上拉 / 下拉电阻、反相器、数字滤波器、开漏模式）通过 IOCON 模块配置。

LPC800 的大部分功能都可通过开关矩阵分配给除电源和接地引脚外的任意外部引脚。这些功能称为可转移功能。

晶体振荡器 (XTALIN/XTALOUT) 或模拟比较器输入等某些功能仅可分配给具有适当电气特性的特定外部引脚。这些功能称为固定引脚功能。如果某个固定引脚功能未被使用，则可将其替换为任意可转移功能。

对于固定引脚模拟功能，开关矩阵可使能模拟输入或输出并禁用数字焊盘。

GPIO 是特殊的固定引脚功能。默认情况下，每个 GPIO 仅分配给一个外部引脚。外部引脚随后便由其固定引脚 GPIO 功能标识。数字输入的电平始终反映在 GPIO 端口寄存器中，以及反映在引脚中断 / 模式匹配状态中 （若选定），与哪个 （数字）功能被通过开关矩阵分配到引脚上无关。

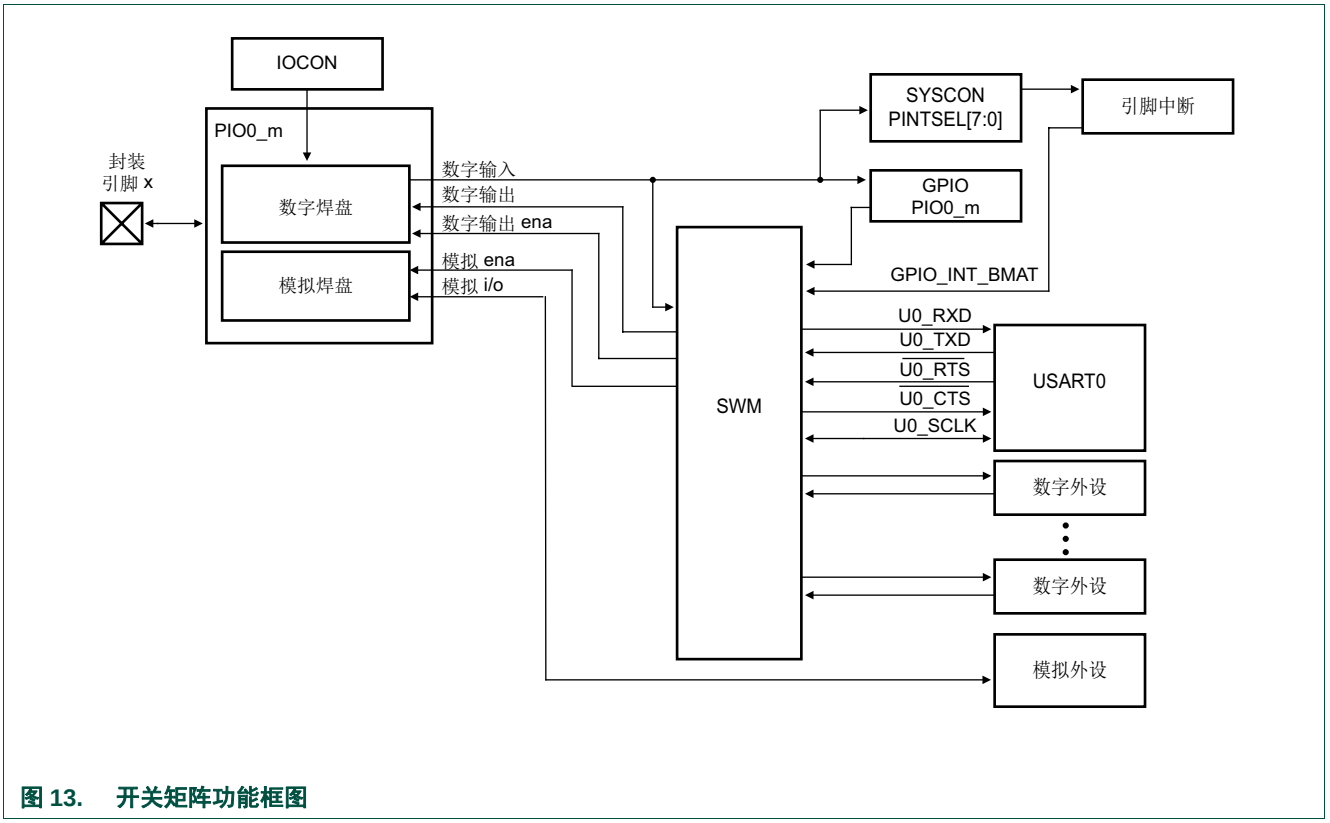


图 13. 开关矩阵功能框图

注：在所有可转移功能和固定引脚功能中，可将多个功能分配给同一个引脚，但分配的输出或双向功能不可超过一个（见图 13）。分配引脚功能时，请遵循如下指南：

- 允许在一个或多个 PINASSIGN 寄存器中，针对同一个引脚编号进行设置，以便在一个引脚上向多个内部输入发送一个输入信号。

  示例：

  在 PINENABLE0 寄存器内可使能引脚 PIO0_1 上的 CLKIN 输入，同时通过 PINASSIGN 寄存器将一个或多个 SCT 输入分配给引脚 PIO0_1，从而使 CLKIN 馈入 SCT。

  可将一个引脚上的输入信号发送至所有 SCT 输入端，用作 SCT 中止信号。

- 通过在 PINASSIGN 寄存器位字段中设置同一个引脚编号，可允许一个数字输出功能控制一个或多个数字输入，用于输出和输入。

  示例：

  可将同一个引脚编号分配给 ACMP_OUT 功能和一个 SCT 输入 CTIN_n。这就能将比较器输出与 SCT 的输入 n 连接。

  通过将同一个引脚编号分配给 Un_RXD 和 Un_TXD，可将 USART 发送输出环回至接收输入。

- 不允许将多个输出或双向功能分配给一个引脚。
- 通过开关矩阵将任意功能分配给某个引脚时， GPIO 输出会变为禁用。

9.4.1 可转移功能

表 95. 可转移功能（通过开关矩阵分配给引脚 PIO0_0 至 PIO0_17）

功能名称	类型	说明	SWM 引脚分配寄存器	参考
U0_TXD	O	USART0 的发送器输出。	PINASSIGN0	表 97
U0_RXD	I	USART0 的接收器输入。	PINASSIGN0	表 97
U0_RTS	O	USART0 的请求发送输出。	PINASSIGN0	表 97
U0_CTS	I	USART0 的清零发送输入。	PINASSIGN0	表 97
U0_SCLK	I/O	同步模式下 USART0 的串行时钟输入 / 输出。	PINASSIGN1	表 98
U1_TXD	O	USART1 的发送器输出。	PINASSIGN1	表 98
U1_RXD	I	USART1 的接收器输入。	PINASSIGN1	表 98
U1_RTS	O	USART1 的请求发送输出。	PINASSIGN1	表 98
U1_CTS	I	USART1 的清零发送输入。	PINASSIGN2	表 99
U1_SCLK	I/O	同步模式下 USART1 的串行时钟输入 / 输出。	PINASSIGN2	表 99
U2_TXD	O	USART2 的发送器输出。	PINASSIGN2	表 99
U2_RXD	I	USART2 的接收器输入。	PINASSIGN2	表 99
U2_RTS	O	USART1 的请求发送输出。	PINASSIGN3	表 100
U2_CTS	I	USART1 的清零发送输入。	PINASSIGN3	表 100
U2_SCLK	I/O	同步模式下 USART1 的串行时钟输入 / 输出。	PINASSIGN3	表 100
SPI0_SCK	I/O	SPI0 的串行时钟。	PINASSIGN3	表 100
SPI0_MOSI	I/O	SPI0 的主机输出从机输入。	PINASSIGN4	表 101
SPI0_MISO	I/O	SPI0 的主机输入从机输出。	PINASSIGN4	表 101
SPI0_SSEL	I/O	SPI0 的从机选择。	PINASSIGN4	表 101
SPI1_SCK	I/O	SPI1 的串行时钟。	PINASSIGN4	表 101
SPI1_MOSI	I/O	SPI1 的主机输出从机输入。	PINASSIGN5	表 102

表 95. 可转移功能 （通过开关矩阵分配给引脚 PIO0_0 至 PIO0_17）*（续）*

功能名称	类型	说明	SWM 引脚分配寄存器	参考
SPI1_MISO	I/O	SPI1 的主机输入从机输出。	PINASSIGN5	表 102
SPI1_SSEL	I/O	SPI1 的从机选择。	PINASSIGN5	表 102
CTIN_0	I	SCT 输入 0。	PINASSIGN5	表 102
CTIN_1	I	SCT 输入 1。	PINASSIGN6	表 103
CTIN_2	I	SCT 输入 2。	PINASSIGN6	表 103
CTIN_3	I	SCT 输入 3。	PINASSIGN6	表 103
CTOUT_0	O	SCT 输出 0。	PINASSIGN6	表 103
CTOUT_1	O	SCT 输出 1。	PINASSIGN7	表 104
CTOUT_2	O	SCT 输出 2。	PINASSIGN7	表 104
CTOUT_3	O	SCT 输出 3。	PINASSIGN7	表 104
I2C0_SDA	I/O	I^2C 总线数据输入 / 输出 （若分配给引脚 PIO0_11，则处于开漏状态）。仅当分配给引脚 PIO0_11 且在 I/O 配置寄存器中选择 I^2C 超快速模式时，才会有大电流灌入。	PINASSIGN7	表 104
I2C0_SCL	I/O	I^2C 总线时钟输入 / 输出 （如果分配给引脚 PIO0_10，则处于开漏状态）。仅当分配给引脚 PIO0_10 且在 I/O 配置寄存器中选择 I^2C 超快速模式时，才会有大电流灌入。	PINASSIGN8	表 105
ACMP_O	O	模拟比较器输出。	PINASSIGN8	表 105
CLKOUT	O	时钟输出。	PINASSIGN8	表 105
GPIO_INT_BMAT	O	模式匹配引擎输出。	PINASSIGN8	表 105

9.4.2 开关矩阵寄存器接口

开关矩阵含有两个引脚分配寄存器模块：PINASSIGN 和 PINENABLE。每个功能在该寄存器组中都有一个分配字段 （1 位或 8 位宽度），可通过它们设置需要分配功能的外部引脚（由 GPIO 功能标识）。

范围为 PIO0_0 至 PIO0_17 的 GPIO 编号为 0 至 17，用于通过引脚分配寄存器执行分配。

有两类功能必须以不同方式分配至端口引脚：

1. **可转移功能** （PINASSIGN0 ~ 8）：

   所有可转移功能都是数字功能。通过 PINASSIGN 寄存器中与可转移功能相关的 8 个位分配该功能至引脚编号。一旦将某功能分配给引脚 PIO0_n，它便通过该引脚与封装的物理引脚相连。

   注：在任意给定时刻，只能分配一个数字输出功能至一个外部引脚。

   注：可分配多个数字输入功能至一个外部引脚。

2. **固定引脚功能** (PINENABLE0)：

   某些功能要求引脚具有特殊的特性，并且不可转移到别的物理引脚上。因此，这些功能被映射到固定端口引脚上。固定引脚功能的示例有：振荡器引脚和比较器引脚。

   每个固定引脚功能都与 PINENABLE0 寄存器中的某个位相关联，该位可选中或取消选中该功能。

   – 若某个固定引脚功能被取消选择，则任意可转移功能均可分配至该功能的端口和引脚。

   – 若固定引脚功能取消选择并且该引脚未分配可转移功能，则其 GPIO 功能将被分配给此引脚。

– 复位时，所有固定引脚功能都将取消选择。
– 若选中某个固定引脚功能，则其分配引脚无法用于其他功能。

9.5 寄存器说明

表 96. 寄存器概述：开关矩阵 （基址 0x4000 C000）

名称	访问类型	偏移	说明	复位值	参考
PINASSIGN0	R/W	0x000	引脚分配寄存器 0。分配可转移功能 U0_TXD、U0_RXD、 U0_RTS、 U0_CTS。	0xFFFF FFFF	表 97
PINASSIGN1	R/W	0x004	引脚分配寄存器 1。分配可转移功能 U0_SCLK、 U1_TXD、 U1_RXD、 U1_RTS。	0xFFFF FFFF	表 98
PINASSIGN2	R/W	0x008	引脚分配寄存器 2。分配可转移功能 U1_CTS、U1_SCLK、 U2_TXD、 U2_RXD。	0xFFFF FFFF	表 99
PINASSIGN3	R/W	0x00C	引脚分配寄存器 3。分配可转移功能 U2_RTS、U2_CTS、 U2_SCLK、 SPI0_SCK。	0xFFFF FFFF	表 100
PINASSIGN4	R/W	0x010	引脚分配寄存器 4。分配可转移功能 SPI0_MOSI、 SPI0_MISO、 SPI0_SSEL、SPI1_SCK。	0xFFFF FFFF	表 101
PINASSIGN5	R/W	0x014	引脚分配寄存器 5。分配可转移功能 SPI1_MOSI、 SPI1_MISO、 SPI1_SSEL、CTIN_0。	0xFFFF FFFF	表 102
PINASSIGN6	R/W	0x018	引脚分配寄存器 6。分配可转移功能 CTIN_1、CTIN_2、 CTIN_3、 CTOUT_0。	0xFFFF FFFF	表 103
PINASSIGN7	R/W	0x01C	引脚分配寄存器 7。分配可转移功能 CTOUT_1、 CTOUT_2、 CTOUT_3、I2C_SDA。	0xFFFF FFFF	表 104
PINASSIGN8	R/W	0x020	引脚分配寄存器 8。分配可转移功能 I2C_SCL、 ACMP_O、 CLKOUT、GPIO_INT_BMAT。	0xFFFF FFFF	表 105
-	-	0x024	保留。	-	-
PINENABLE0	R/W	0x1C0	引脚使能寄存器 0。使能固定引脚功能 ACMP_I0、 ACMP_I1、SWCLK、 SWDIO、XTALIN、 XTALOUT、 RESET、 CLKIN、VDDCMP。	0x1B3	表 106

9.5.1 引脚分配寄存器 0

表 97. 引脚分配寄存器 0 （PINASSIGN0，地址 0x4000 C000）位说明

位	符号	说明	复位值
7:0	U0_TXD_O	U0_TXD 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
15:8	U0_RXD_I	U0_RXD 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
23:16	U0_RTS_O	U0_RTS 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
31:24	U0_CTS_I	U0_CTS 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF

9.5.2 引脚分配寄存器 1

表 98. 引脚分配寄存器 1（PINASSIGN1，地址 0x4000 C004）位说明

位	符号	说明	复位值
7:0	U0_SCLK_IO	U0_SCLK 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
15:8	U1_TXD_O	U1_TXD 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
23:16	U1_RXD_I	U1_RXD 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
31:24	U1_RTS_O	U1_RTS 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF

9.5.3 引脚分配寄存器 2

表 99. 引脚分配寄存器 2（PINASSIGN2，地址 0x4000 C008）位说明

位	符号	说明	复位值
7:0	U1_CTS_I	U1_CTS 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
15:8	U1_SCLK_IO	U1_SCLK 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
23:16	U2_TXD_O	U2_TXD 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
31:24	U2_RXD_I	U2_RXD 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF

9.5.4 引脚分配寄存器 3

表 100. 引脚分配寄存器 3（PINASSIGN3，地址 0x4000 C00C）位说明

位	符号	说明	复位值
7:0	U2_RTS_O	U2_RTS 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
15:8	U2_CTS_I	U2_CTS 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
23:16	U2_SCLK_IO	U2_SCLK 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
31:24	SPI0_SCK_IO	SPI0_SCK 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF

9.5.5 引脚分配寄存器 4

表 101. 引脚分配寄存器 4（PINASSIGN4，地址 0x4000 C010）位说明

位	符号	说明	复位值
7:0	SPI0_MOSI_IO	SPI0_MOSI 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF

表 101. 引脚分配寄存器 4 （PINASSIGN4，地址 0x4000 C010）位说明 *（续）*

位	符号	说明	复位值
15:8	SPI0_MISO_IO	SPI0_MISIO 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
23:16	SPI0_SSEL_IO	SPI0_SSEL 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
31:24	SPI1_SCK_IO	SPI1_SCK 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF

9.5.6 引脚分配寄存器 5

表 102. 引脚分配寄存器 5 （PINASSIGN5，地址 0x4000 C014）位说明

位	符号	说明	复位值
7:0	SPI1_MOSI_IO	SPI1_MOSI 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
15:8	SPI1_MISO_IO	SPI1_MISIO 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
23:16	SPI1_SSEL_IO	SPI1_SSEL 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
31:24	CTIN_0_I	CTIN_0 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF

9.5.7 引脚分配寄存器 6

表 103. 引脚分配寄存器 6 （PINASSIGN6，地址 0x4000 C018）位说明

位	符号	说明	复位值
7:0	CTIN_1_I	CTIN_1 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
15:8	CTIN_2_I	CTIN_2 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
23:16	CTIN_3_I	CTIN_3 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
31:24	CTOUT_0_O	CTOUT_0 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF

9.5.8 引脚分配寄存器 7

表 104. 引脚分配寄存器 7 （PINASSIGN7，地址 0x4000 C01C）位说明

位	符号	说明	复位值
7:0	CTOUT_1_O	CTOUT_1 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
15:8	CTOUT_2_O	CTOUT_2 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
23:16	CTOUT_3_O	CTOUT_3 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
31:24	I2C_SDA_IO	I2C_SDA 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF

9.5.9 引脚分配寄存器 8

表 105. 引脚分配寄存器 8 （PINASSIGN8，地址 0x4000 C020）位说明

位	符号	说明	复位值
7:0	I2C_SCL_IO	I2C_SCL 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
15:8	ACMP_O_O	ACMP_O_O 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
23:16	CLKOUT_O	CLKOUT 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF
31:24	GPIO_INT_BMAT_O	GPIO_INT_BMAT 功能分配。数值为待分配给该功能的引脚编号。可使用下列引脚：PIO0_0 (= 0) 至 PIO0_17 (= 0x11)。	0xFF

9.5.10 引脚使能寄存器 0

表 106. 引脚使能寄存器 0 （PINENABLE0，地址 0x4000 C1C0）位说明

位	符号	值	说明	复位值
0	ACMP_I1_EN		使能固定引脚功能。写入 1 将取消选择该功能，之后任意可转移功能便可分配给该引脚。默认情况下，固定引脚功能会被取消选择且 GPIO 会被分配给该引脚。	1
		0	使能 ACMP_I1。该功能在引脚 PIO0_0 上使能。	
		1	禁用 ACMP_I1。 GPIO 功能 PIO0_0 （默认）或任何其他可转移功能可分配给引脚 PIO0_0。	
1	ACMP_I2_EN		使能固定引脚功能。写入 1 将取消选择该功能，之后任意可转移功能便可分配给该引脚。默认情况下，固定引脚功能会被取消选择且 GPIO 会被分配给该引脚。功能 CLKIN 和 ACMP_I2 分配给同一引脚 PIO0_1。要使用 ACMP_I2，则需禁用该寄存器中位 7 的 CLKIN 功能并使能 ACMP_I2。	1
		0	使能 ACMP_I2。该功能在引脚 PIO0_1 上使能。	
		1	禁用 ACMP_I2。 GPIO 功能 PIO0_1 （默认）或任何其他可转移功能可分配给引脚 PIO0_1。	
2	SWCLK_EN		使能固定引脚功能。写入 1 将取消选择该功能，之后任意可转移功能便可分配给该引脚。默认选定该功能。	0
		0	使能 SWCLK。该功能在引脚 PIO0_3 上使能。	
		1	禁用 SWCLK。GPIO 功能 PIO0_3 在该引脚上选择。其他任意可转移功能均可分配给 PIO0_3。	
3	SWDIO_EN		使能固定引脚功能。写入 1 将取消选择该功能，之后任意可转移功能便可分配给该引脚。默认选定该功能。	0
		0	使能 SWDIO。该功能在引脚 PIO0_2 上使能。	
		1	禁用 SWDIO。 GPIO 功能 PIO0_2 在该引脚上选择。其他任意可转移功能均可分配给 PIO0_2。	
4	XTALIN_EN		使能固定引脚功能。写入 1 将取消选择该功能，之后任意可转移功能便可分配给该引脚。默认情况下，固定引脚功能会被取消选择且 GPIO 会被分配给该引脚。	1
		0	使能 XTALIN。引脚 PIO0_8 上使能该功能。	
		1	禁用 XTALIN。 GPIO 功能 PIO0_8 （默认）或任何其他可转移功能可分配给引脚 PIO0_8。	

表 106. 引脚使能寄存器 0（PINENABLE0，地址 0x4000 C1C0）位说明 *（续）*

位	符号	值	说明	复位值
5	XTALOUT_EN		使能固定引脚功能。写入 1 将取消选择该功能，之后任意可转移功能便可分配给该引脚。默认情况下，固定引脚功能会被取消选择且 GPIO 会被分配给该引脚。	1
		0	使能 XTALOUT。该功能在引脚 PIO0_9 上使能。	
		1	禁用 XTALOUT。 GPIO 功能 PIO0_9（默认）或任何其他可转移功能可分配给引脚 PIO0_9。	
6	RESET_EN		使能固定引脚功能。写入 1 将取消选择该功能，之后任意可转移功能便可分配给该引脚。默认选定该功能。	0
		0	使能 $\overline{\text{RESET}}$。该功能在引脚 PIO0_5 上使能。	
		1	禁用 $\overline{\text{RESET}}$。 GPIO 功能 PIO0_5 在该引脚上选择。其他任意可转移功能均可分配给 PIO0_5。	
7	CLKIN		使能固定引脚功能。写入 1 将取消选择该功能，之后任意可转移功能便可分配给该引脚。默认情况下，固定引脚功能会被取消选择且 GPIO 会被分配给该引脚。功能 CLKIN 和 ACMP_I2 分配给同一引脚 PIO0_1。要使用 CLKIN，则需禁用该寄存器中位 1 的 ACMP_I2 并使能 CLKIN。	1
		0	使能 CLKIN。该功能在引脚 PIO0_1 上使能。	
		1	禁用 CLKIN。 GPIO 功能 PIO0_1（默认）或任何其他可转移功能可分配给引脚 CKLIN。	
8	VDDCMP		使能固定引脚功能。写入 1 将取消选择该功能，之后任意可转移功能便可分配给该引脚。默认情况下，固定引脚功能会被取消选择且 GPIO 会被分配给该引脚。	1
		0	使能 VDDCMP。该功能在引脚 PIO0_6 上使能。	
		1	禁用 VDDCMP。 GPIO 功能 PIO0_6（默认）或任何其他可转移功能可分配给引脚 PIO0_6。	
31:9	-		保留。	<tbd>

10.1 本章导读

所有 LPC800 器件都提供 SCT。

10.2 特性

- 两个 16 位计数器或一个 32 位计数器。
- 由总线时钟或选定输入计时的计数器。
- 加法计数器或加减计数器。
- 状态变量可实现多个计数器周期中的定序。
- 下列条件可定义一次事件：计数器匹配条件、输入 （或输出）条件、指定状态中的匹配和 / 或输入 / 输出条件组合、计数方向。
- 事件控制输出、中断和 SCT 状态。
  - 匹配寄存器 0 可用作自动限值。
  - 在双向模式下，事件可根据计数方向使能。
  - 在发生另一次认证事件前，可保持匹配事件。
- 选定的事件可以限制、终止、启动或停止计数器操作。
- 提供如下支持：
  - 4 个输入
  - 4 个输出
  - 5 个匹配 / 捕获寄存器
  - 6 个事件
  - 2 个状态

10.3 基本配置

配置 SCT 采用的是下列步骤：

- 使用 SYSAHBCLKCTRL 寄存器（表 18）使能 SCT 寄存器接口时钟和外设时钟。LPC800 系统时钟用作 SCT 时钟处理的输入时钟以及 SCT 的时钟源。
- 使用 PRESETCTRL 寄存器 （表 7）清零 SCT 外设复位。
- SCT 组合中断连接至 NVIC 的 8 号插槽。
- 使用开关矩阵将 SCT 输入与输出连接至引脚 （参见章节 10.4），并且采用的连接方式是内部连接 （参见章节 10.5）。

10.3.1 SCT 用作简单定时器

要将 SCT 配置为具有匹配或捕获功能的简单定时器，请根据下列步骤：

1. 将 SCT 设置为一个 32 位定时器或者将其设置为一个或两个 16 位定时器。参见表 109。
2. 用某个计数值预载 32 位定时器或 16 位定时器。参见表 115。

3. 要在定时器达到匹配值时创建匹配事件，则：
   a. 配置匹配寄存器的寄存器映射。参见表 118。
   b. 用匹配值配置一个或多个匹配寄存器。参见表 126。
   c. 对于每个匹配值，都创建一个匹配事件。参见表 131。
   d. 要创建匹配事件中断，则使能事件中断。参见表 123。
   e. 要在引脚上创建匹配输出，则将 CTOUTn 功能分配给某一引脚 （参见章节 10.4），然后在 EVn_CTRL 寄存器中为匹配事件选择某个输出。参见表 131。EVn_CTRL 寄存器还可控制所创建输出信号的类型。
4. 要捕获信号的定时器值，则：
   a. 配置捕获寄存器的寄存器映射。参见表 118。
   b. 创建一个或多个捕获事件。参见表 131。
   c. 将 CTIN 功能分配给引脚（参见章节 10.4），然后配置信号以创建事件。参见表 131。
5. 通过对 CRTL 寄存器进行写操作启动定时器。参见表 110。
6. 对捕获寄存器进行读操作，以读取捕获事件的定时器值。

10.4 引脚说明

SCT 输入和输出均为可转移功能，通过开关矩阵分配给外部引脚。

有关如何将 SCT 功能分配给 LPC800 封装上的引脚，请参见章节 9.3.1 “ 将内部信号与封装引脚相连 ”。

表 107. SCT 引脚说明

功能	方向	引脚	说明	SWM 寄存器	参考
CTIN_0	I	任意	SCT 输入 0	PINASSIGN5	表 102
CTIN_1	I	任意	SCT 输入 1	PINASSIGN6	表 103
CTIN_2	I	任意	SCT 输入 2	PINASSIGN6	表 103
CTIN_3	I	任意	SCT 输入 3	PINASSIGN6	表 103
CTOUT_0	O	任意	SCT 输出 0	PINASSIGN6	表 103
CTOUT_1	O	任意	SCT 输出 1	PINASSIGN7	表 104
CTOUT_2	O	任意	SCT 输出 2	PINASSIGN7	表 104
CTOUT_3	O	任意	SCT 输出 3	PINASSIGN7	表 104

10.5 简介

状态可配置定时器 (SCT) 可以执行各种定时、计数、输出调制和输入捕获操作。

最基本的一个用户可编程选项是 SCT 是用作两个 16 位计数器还是一个统一的 32 位计数器。如果用作两个计数器，则除了计数器值外，下列操作要素对于每个计数器来说都是独立的：

- 状态变量
- 限制、终止、停止和启动条件
- 匹配 / 捕获寄存器的值，以及重载值或捕获控制值

如果用作两个计数器，则下列 SCT 操作要素为全局要素：

- 时钟选择
- 输入
- 事件
- 输出
- 中断

事件、输出和中断可使用任意计数器的匹配条件。

注：本章中的术语“总线错误”表示使处理器接受异常的 SCT 响应。

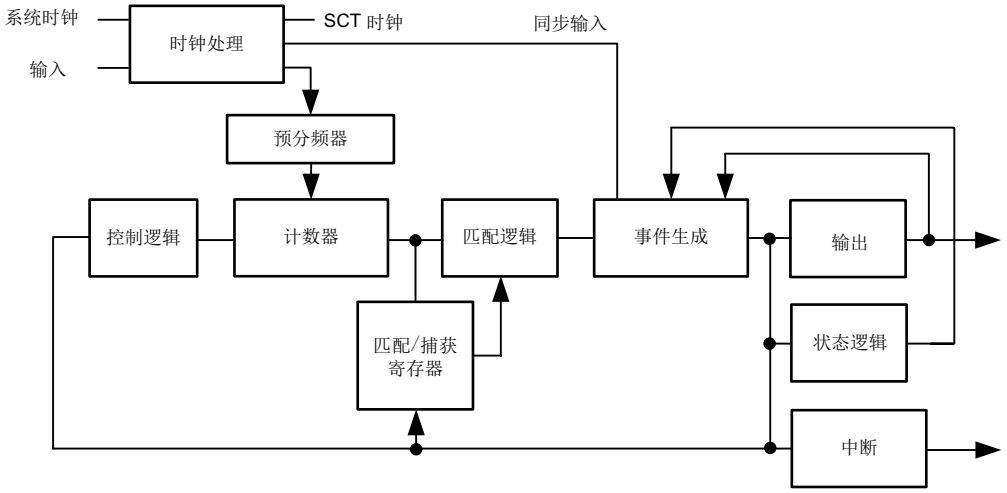


图 14. SCT 功能框图

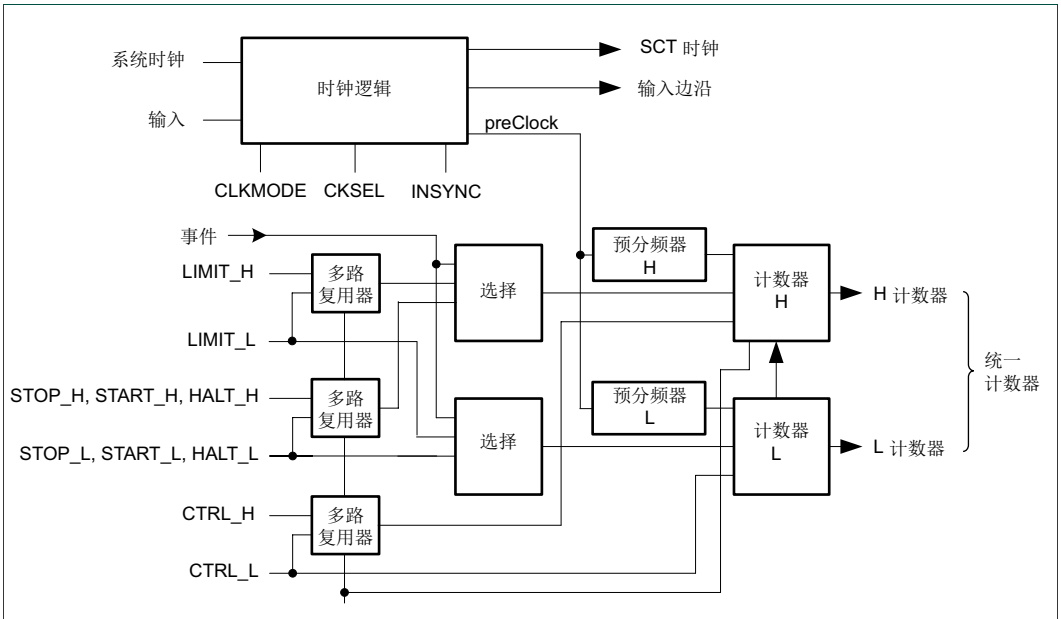


图 15. SCT 计数器和选择逻辑

10.6 寄存器说明

状态配置定时器的寄存器地址如表 108 所示。对于大部分 SCT 寄存器，寄存器功能取决于其他特定的寄存器位的设置：

1. CONFIG 寄存器中的 UNIFY 位可确定是将 SCT 用作一个 32 位寄存器 （用作一个 32 位计数器 / 定时器）还是两个名为 L 和 H 的 16 位计数器 / 定时器。UNIFY 位的设置会体现在寄存器映射中：
   - UNIFY = 1：仅使用一个寄存器 （用作一个 32 位计数器 / 定时器）。
   - UNIFY = 0:L 寄存器和 H 寄存器可通过 32 位读操作或写操作访问，也可以对这两者分别执行读操作或写操作 （用作两个 16 位计数器 / 定时器）。

   通常，在访问任何其他寄存器之前，对 CONFIG 寄存器执行写操作来配置 UNIFY 位。
2. REGMODE 寄存器中的 REGMODEn 位可确定每组匹配 / 捕获寄存器是使用匹配功能还是捕获功能：
   - REGMODEn = 1：寄存器用作匹配寄存器和重新载入寄存器。
   - REGMODEn = 0：寄存器用作捕获寄存器和捕获控制寄存器。

表 108. 寄存器概述：状态可配置定时器 （基址 0x5000 4000）

名称	访问类型	地址偏移	说明	复位值	参考
CONFIG	R/W	0x000	SCT 配置寄存器	0x0000 7E00	表 109
CTRL	R/W	0x004	SCT 控制寄存器	0x0004 0004	表 110
CTRL_L	R/W	0x004	SCT 控制寄存器低速计数器 16 位	-	表 110
CTRL_H	R/W	0x006	SCT 控制寄存器高速计数器 16 位	-	表 110
LIMIT	R/W	0x008	SCT 限值寄存器	0x0000 0000	表 111
LIMIT_L	R/W	0x008	SCT 限值寄存器低速计数器 16 位	-	表 111
LIMIT_H	R/W	0x00A	SCT 限值寄存器高速计数器 16 位	-	表 111
HALT	R/W	0x00C	SCT 终止条件寄存器	0x0000 0000	表 112
HALT_L	R/W	0x00C	SCT 终止条件寄存器低速计数器 16 位	-	表 112
HALT_H	R/W	0x00E	SCT 终止条件寄存器高速计数器 16 位	-	表 112
STOP	R/W	0x010	SCT 停止条件寄存器	0x0000 0000	表 113
STOP_L	R/W	0x010	SCT 停止条件寄存器低速计数器 16 位	-	表 113
STOP_H	R/W	0x012	SCT 停止条件寄存器高速计数器 16 位	-	表 113
启动	R/W	0x014	SCT 启动条件寄存器	0x0000 0000	表 114
START_L	R/W	0x014	SCT 启动条件寄存器低速计数器 16 位	-	表 114
START_H	R/W	0x016	SCT 启动条件寄存器高速计数器 16 位	-	表 114
-	-	0x018 - 0x03C	保留		-
COUNT	R/W	0x040	SCT 计数器寄存器	0x0000 0000	表 115
COUNT_L	R/W	0x040	SCT 计数器寄存器低速计数器 16 位	-	表 115
COUNT_H	R/W	0x042	SCT 计数器寄存器高速计数器 16 位	-	表 115
STATE	R/W	0x044	SCT 状态寄存器	0x0000 0000	表 116
STATE_L	R/W	0x044	SCT 状态寄存器低速计数器 16 位	-	表 116
STATE_H	R/W	0x046	SCT 状态寄存器高速计数器 16 位	-	表 116
INPUT	RO	0x048	SCT 输入寄存器	0x0000 0000	表 117
REGMODE	R/W	0x04C	SCT 匹配 / 捕获寄存器模式寄存器	0x0000 0000	表 118
REGMODE_L	R/W	0x04C	SCT匹配/捕获寄存器模式寄存器低速计数器16位	-	表 118
REGMODE_H	R/W	0x04E	SCT匹配/捕获寄存器模式寄存器高速计数器16位	-	表 118
OUTPUT	R/W	0x050	SCT 输出寄存器	0x0000 0000	表 119
OUTPUTDIRCTRL	R/W	0x054	SCT 输出计数器方向控制寄存器	0x0000 0000	表 120
RES	R/W	0x058	SCT 冲突解决寄存器	0x0000 0000	表 121
-	-	0x05C	-	-	-
-	-	0x060	-	-	-
-	-	0x064 - 0x0EC	保留	-	-
EVEN	R/W	0x0F0	SCT 事件使能寄存器	0x0000 0000	表 122
EVFLAG	R/W	0x0F4	SCT 事件标志寄存器	0x0000 0000	表 123
CONEN	R/W	0x0F8	SCT 冲突使能寄存器	0x0000 0000	表 124
CONFLAG	R/W	0x0FC	SCT 冲突标志寄存器	0x0000 0000	表 125
MATCH0 至 MATCH4	R/W	0x100 至 0x110	SCT 匹配值寄存器的匹配通道为 0 至 4；REGMOD0 至 REGMODE4 = 0	0x0000 0000	表 125

表 108. 寄存器概述：状态可配置定时器 （基址 0x5000 4000）*（续）*

名称	访问类型	地址偏移	说明	复位值	参考
MATCH_L0 至 MATCH_L4	R/W	0x100 至 0x110	SCT 匹配值寄存器的匹配通道为 0 至 4；低速计数器 16 位； REGMOD0_L 至 REGMODE4_L = 0	-	表 125
MATCH_H0 至 MATCH_H4	R/W	0x102 至 0x112	SCT 匹配值寄存器的匹配通道为 0 至 4；高速计数器 16 位； REGMOD0_H 至 REGMODE4_H = 0	-	表 125
CAP0 至 CAP4		0x100 至 0x110	SCT 捕获寄存器的捕获通道为 0 至 4； REGMOD0 至 REGMODE4 = 1	0x0000 0000	表 127
CAP_L0 至 CAP_L4		0x100 至 0x110	SCT 捕获寄存器的捕获通道为 0 至 4；低速计数器 16 位； REGMOD0_L 至 REGMODE4_L = 1	-	表 127
CAP_H0 至 CAP_H4		0x102 至 0x13E	SCT 捕获寄存器的捕获通道为 0 至 4；高速计数器 16 位； REGMOD0_H 至 REGMODE4_H = 1	-	表 127
MATCHREL0 至 MATCHREL4	R/W	0x200 至 0x210	SCT 匹配重新载入值寄存器 0 至 4； REGMOD0 = 0 至 REGMODE4 = 0	0x0000 0000	表 128
MATCHREL_L0 至 MATCHREL_L4	R/W	0x200 至 0x210	SCT 匹配重新载入值寄存器 0 至 4；低速计数器 16 位； REGMOD0_L = 0 至 REGMODE4_L = 0	-	表 128
MATCHREL_H0 至 MATCHREL_H4	R/W	0x202 至 0x212	SCT 匹配重新载入值寄存器 0 至 4；高速计数器 16 位； REGMOD0_H = 0 至 REGMODE4_H = 0	-	表 128
CAPCTRL0 至 CAPCTRL4		0x200 至 0x210	SCT 捕获控制寄存器 0 至 4； REGMOD0 = 1 至 REGMODE4 = 1	0x0000 0000	表 129
CAPCTRL_L0 至 CAPCTRL_L4		0x200 至 0x210	SCT 捕获控制寄存器 0 至 4；低速计数器 16 位； REGMOD0_L = 1 至 REGMODE4_L = 1	-	表 129
CAPCTRL_H0 至 CAPCTRL_H4		0x202 至 0x212	SCT 捕获控制寄存器 0 至 4；高速计数器 16 位； REGMOD0 = 1 至 REGMODE4 = 1	-	表 129
EV0_STATE	R/W	0x300	SCT 事件 0 状态寄存器	0x0000 0000	表 130
EV0_CTRL	R/W	0x304	SCT 事件 0 控制寄存器	0x0000 0000	表 131
EV1_STATE	R/W	0x308	SCT 事件 1 状态寄存器	0x0000 0000	表 130
EV1_CTRL	R/W	0x30C	SCT 事件 1 控制寄存器	0x0000 0000	表 131
EV2_STATE	R/W	0x310	SCT 事件 2 状态寄存器	0x0000 0000	表 130
EV2_CTRL	R/W	0x314	SCT 事件 2 控制寄存器	0x0000 0000	表 131
EV3_STATE	R/W	0x318	SCT 事件 3 状态寄存器	0x0000 0000	表 130
EV3_CTRL	R/W	0x31C	SCT 事件 3 控制寄存器	0x0000 0000	表 131
EV4_STATE	R/W	0x320	SCT 事件 4 状态寄存器	0x0000 0000	表 130
EV4_CTRL	R/W	0x324	SCT 事件 4 控制寄存器	0x0000 0000	表 131
EV5_STATE	R/W	0x328	SCT 事件 5 状态寄存器	0x0000 0000	表 130
EV5_CTRL	R/W	0x32C	SCT 事件 5 控制寄存器	0x0000 0000	表 131
OUT0_SET	R/W	0x500	SCT 输出 0 设置寄存器	0x0000 0000	表 132
OUT0_CLR	R/W	0x504	SCT 输出 0 清零寄存器	0x0000 0000	表 133
OUT1_SET	R/W	0x508	SCT 输出 1 设置寄存器	0x0000 0000	表 132
OUT1_CLR	R/W	0x50C	SCT 输出 1 清零寄存器	0x0000 0000	表 133
OUT2_SET	R/W	0x510	SCT 输出 2 设置寄存器	0x0000 0000	表 132
OUT2_CLR	R/W	0x514	SCT 输出 2 清零寄存器	0x0000 0000	表 133
OUT3_SET	R/W	0x518	SCT 输出 3 设置寄存器	0x0000 0000	表 132
OUT3_CLR	R/W	0x51C	SCT 输出 3 清零寄存器	0x0000 0000	表 133

10.6.1 SCT 配置寄存器

该寄存器配置 SCT 的整体运行。先对该寄存器进行写操作，再对其他任何寄存器进行写操作。

表 109. SCT 配置寄存器 （CONFIG，地址 0x5000 4000）位说明

位	符号	值	说明	复位值
0	UNIFY		SCT 操作	0
		0	16 位。 SCT 用作两个 16 位计数器，分别为 L 计数器和 H 计数器。	
		1	32 位。 SCT 用作统一的 32 位计数器。	
2:1	CLKMODE		SCT 时钟模式	0
		0x0	总线时钟。总线时钟对 SCT 和预分频器进行计时。	
		0x1	预分频器总线时钟。SCT 时钟是总线时钟。但仅在 CKSEL 字段所选的输入采样找到所选的边缘时才能使用预分频器计数。时钟输入的最小脉冲宽度为 1 个总线时钟周期。该模式为高性能采样时钟模式。	
		0x2	输入。CKSEL 选定的输入作为 SCT 和预分频器的时钟输入。输入同步至总线时钟，且可反转。时钟输入的最小脉冲宽度为 1 个总线时钟周期。该模式为低功耗采样时钟模式。	
		0x3	保留。	
6:3	CKSEL		SCT 时钟选择。保留所有其它值。	0
		0x0	输入 0 上升沿。	
		0x1	输入 0 下降沿。	
		0x2	输入 1 上升沿。	
		0x3	输入 1 下降沿。	
		0x4	输入 2 上升沿。	
		0x5	输入 2 下降沿。	
		0x6	输入 3 上升沿。	
		0x7	输入 3 下降沿。	
7	NORELAOD_L	-	该位如果为 1 可防止较低位匹配寄存器从其对应的重新载入寄存器中重新载入。在任何时候都可以编写软件来设置或清零该位。UNIFY 位被设置时，该位对于较高位寄存器和较低位寄存器都适用。	0
8	NORELOAD_H	-	该位如果为 1 可防止较高位匹配寄存器从其对应的重新载入寄存器中重新载入。在任何时候都可以编写软件来设置或清零该位。UNIFY 位被设置时，该位不使用。	0
16:9	INSYNC	-	输入 N （位 9 = 输入 0、位 10 = 输入 1... 位 16 = 输入 7）同步。在使用该位创建事件之前，这些位中的任意一个位中的数值 1 会使相应的输入同步到 SCT 时钟中。如果输入与 SCT 时钟同步，则保留其位 0 以实现更快的响应。 CLKMODE 字段为 1x 时，将不使用该字段中的位 （与 CKSEL 字段所选的输入对应）。	1

表 109. SCT 配置寄存器 （CONFIG，地址 0x5000 4000）位说明 *（续）*

位	符号	值	说明	复位值
17	AUTOLIMIT_L	-	该位如果为 1，则会使匹配寄存器 0 的匹配作为事实上的限制条件，而无需定义相关事件。 在单向模式或改变计数方向的双向模式中，该自动限制与任何限制事件一同造成计数器清零。 在任何时候都可以编写软件来设置或清零该位。UNIFY 位被设置时，该位对于较高位寄存器和较低位寄存器都适用。	
18	AUTOLIMIT_H	-	该位如果为 1，则会使匹配寄存器 0 的匹配作为事实上的限制条件，而无需定义相关事件。 在单向模式或改变计数方向的双向模式中，该自动限制与任何限制事件一同造成计数器清零。 在任何时候都可以编写软件来设置或清零该位。UNIFY 位被设置时，该位不使用。	
31:19	-		保留	-

10.6.2 SCT 控制寄存器

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用 _L 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则可像 CTRL_L 和 CTRL_H 这两个寄存器一样对该寄存器进行写操作。可以单独对 L 寄存器和 H 寄存器执行读 / 写操作，也可在单一的 32 位读 / 写操作中对它们执行操作。

在计数器停止或终止操作时，可写入该寄存器中的所有位。在计数器运行时，只能写入 STOP 或 HALT 位。（在 HALT 设为 1 时，可在后续写操作中写入其他位。）

表 110. SCT 控制寄存器 （CTRL，地址 0x5000 4004）位说明

位	符号	值	说明	复位值
0	DOWN_L	-	当 L 计数器或统一计数器向下计数时，该位为 1。当到达计数器限值且 BIDIR 为 1 时，则通过硬件设置该位。当计数器向下计数且发生限值条件或计数器到达 0 时，则通过硬件将该位清零。	0
1	STOP_L	-	该位为 1 且 HALT 为 0 时， L 计数器或统一计数器不运行，但会发生与计数器相关的 I/O 事件。如果该类事件与启动寄存器中的掩码相匹配，则该位会被清零且计数会继续进行。	0
2	HALT_L	-	该位为 1 时，L 计数器或统一计数器不运行，也不会发生任何事件。复位可设置该位。HALT_L 位为 1 时，STOP_L 位会被清零。要移除终止条件并使 SCT 保持在停止状态（非运行），可对该寄存器执行单次写操作，以更改终止和停止条件。 **注：**该位一旦被设置，只能通过软件清零以恢复计数器运行。	1
3	CLRCTR_L	-	将 1 写入该位时会清零 L 计数器或统一计数器。该位始终以 0 读取。	0
4	BIDIR_L		L 计数器或统一计数器方向选择。	0
		0	向上。计数器会一直向上计数至其所限范围，然后清零。	
		1	双向。计数器会一直向上计数至其所限范围，然后向下计数至限制条件或 0。	
12:5	PRE_L	-	指定对 SCT 时钟进行预分频以产生 L 计数器或统一计数器时钟的系数。计数器时钟将以被 PRE_L+1 分频的 SCT 时钟速率来计时。 **注：**无论何时更改 PRE 值，都会清零计数器 （通过将 1 写入 CLRCTR 位的方式）。	0
15:13	-		保留	
16	DOWN_H	-	当 H 计数器向下计数时，该位为 1。当到达计数器限值且 BIDIR 为 1 时，则通过硬件设置该位。当计数器向下计数且发生限值条件或计数器到达 0 时，则通过硬件将该位清零。	0

表 110. SCT 控制寄存器 （CTRL，地址 0x5000 4004）位说明

位	符号	值	说明	复位值
17	STOP_H	-	该位为 1 且 HALT 为 0 时， H 计数器不运行，但会发生与计数器相关的 I/O 事件。如果该类事件与启动寄存器中的掩码相匹配，则该位会被清零且计数会继续进行。	0
18	HALT_H	-	该位为 1 时， H 计数器不运行，也不会发生任何事件。复位可设置该位。 HALT_H 位为 1 时， STOP_H 位会被清零。要移除终止条件并使 SCT 保持在停止状态 （非运行），可对该寄存器执行单次写操作，以更改终止和停止条件。 **注：**该位一旦被设置，则只能通过恢复计数器操作的软件来清零。	1
19	CLRCTR_H	-	将 1 写入该位可清零 H 计数器。该位始终以 0 读取。	0
20	BIDIR_H		方向选择	0
		0	向上。 H 计数器会一直向上计数至其所限范围，然后清零。	
		1	双向。 H 计数器会一直向上计数至其所限范围，然后向下计数至限制条件或 0。	
28:21	PRE_H	-	指定对SCT时钟进行预分频以产生H计数器时钟的系数。计数器时钟将以被PRELH+1分频的 SCT 时钟速率来计时。 **注：**无论何时更改 PRE 值，都会清零计数器 （通过将 1 写入 CLRCTR 位的方式）。	0
31:29	-		保留	

10.6.3 SCT 限值寄存器

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用 _L 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会如同对 LIMIT_L 和 LIMIT_H 这两个寄存器一样对该寄存器进行写操作。可以单独对 L 寄存器和 H 寄存器执行读 / 写操作，也可在单一的 32 位读 / 写操作中对它们执行操作。

该寄存器中的位可设置将哪个事件作为计数器限值。在发生限值事件时，计数器将在单向模式中清零，或在双向模式改变计数方向。计数器计数完毕后，该状态将始终被视为限值事件，并且计数器会在单向模式中清零，或在双向模式中，开始在下一个时钟边缘向下计数，即使已发生 SCT 限值计数器定义的限值事件。

需要注意的是，除使用该寄存器指定作为限制的事件之外，还可在一发生匹配寄存器 0 的匹配事件时，自动导致限制条件。用户因此就无需为了创建限制条件而特地定义一次事件。配置寄存器中的 AUTOLIMITL 和 AUTOLIMITH 位可使能 / 禁用该特性 （参见表 109）。

表 111. SCT 限制寄存器 （LIMIT，地址 0x5000 4008）位说明

位	符号	说明	复位值
5:0	LIMMSK_L	如果位 n 等于 1，则会将事件 n 作为 L 计数器或统一计数器的计数器限值（事件 0 = 位 0，事件 1 = 位 1，事件 5 = 位 5）。	0
15:6	-	保留。	-
21:16	LIMMSK_H	如果位 n 等于 1，则会将事件 n 作为 H 计数器的计数器限值（事件 0 = 位 16，事件 1 = 位 17，事件 5 = 位 21）。	0
31:22	-	保留。	-

10.6.4 SCT 终止条件寄存器

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用 _L 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会如同对 HALT_L 和 HALT_H 这两个寄存器一样对该寄存器进行写操作。可以单独对 L 寄存器和 H 寄存器执行读 / 写操作，也可在单一的 32 位读 / 写操作中对它们执行操作。

注：终止计数器的任何事件在软件清零 CTRL 寄存器 （表 110）中的 HALT 位之前都可禁用其操作。

表 112. SCT 终止条件寄存器 （HALT，地址 0x5004 400C）位说明

位	符号	说明	复位值
5:0	HALTMSK_L	如果位 n 等于 1，则事件 n 会在 CTRL 寄存器中设置 HALT_L 位（事件 0 = 位 0，事件 1 = 位 1，事件 5 = 位 5）。	0
15:6	-	保留。	-
21:16	HALTMSK_H	如果位 n 等于 1，则事件 n 会在 CTRL 寄存器中设置 HALT_H 位（事件 0 = 位 16，事件 1 = 位 17，事件 5 = 位 21）。	0
31:22	-	保留。	-

10.6.5 SCT 停止条件寄存器

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用 _L 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会如同对 STOPT_L 和 STOP_H 这两个寄存器一样对该寄存器进行写操作。可以单独对 L 寄存器和 H 寄存器执行读 / 写操作，也可在单一的 32 位读 / 写操作中对它们执行操作。

表 113. SCT 停止条件寄存器 （STOP，地址 0x5000 4010）位说明

位	符号	说明	复位值
5:0	STOPMSK_L	如果位 n 等于 1，则事件 n 会在 CTRL 寄存器中设置 STOP_L 位（事件 0 = 位 0，事件 1 = 位 1，事件 5 = 位 5）。	0
15:6	-	保留。	-
21:16	STOPMSK_H	如果位 n 等于 1，则事件 n 会在 CTRL 寄存器中设置 STOP_H 位（事件 0 = 位 16，事件 1 = 位 17，事件 5 = 位 21）。	0
31:22	-	保留。	-

10.6.6 SCT 启动条件寄存器

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用 _L 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会如同对 START_L 和 START_H 这两个寄存器一样对该寄存器进行写操作。可以单独对 L 寄存器和 H 寄存器执行读 / 写操作，也可在单一的 32 位读 / 写操作中对它们执行操作。

该寄存器中的位可选择使用哪个事件 （如果有）来清零控制寄存器中的 STOP 位。（由于HALT为1时不会发生任何事件，因此只能由软件通过对控制寄存器进行写操作来清零HALT位。）

表 114. SCT 启动条件寄存器 （START，地址 0x5000 4014）位说明

位	符号	说明	复位值
5:0	STARTMSK_L	如果位 n 等于 1，则事件 n 会清零 CTRL 寄存器中的 STOP_L 位（事件 0 = 位 0，事件 1 = 位 1，事件 5 = 位 5）。	0
15:6	-	保留。	-
21:16	STARTMSK_H	如果位 n 等于 1，则事件 n 会清零 CTRL 寄存器中的 STOP_H 位 （事件 0 = 位 16，事件 1 = 位 17，事件 5 = 位 21）。	0
31:22	-	保留。	-

10.6.7 SCT 计数器寄存器

如果 CONFIG 寄存器中的 UNIFY 等于 1，则计数器为统一的 32 位寄存器，并且会同时使用 _L 和 _H 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会如同对 COUNT_L 和 COUNT_H 这两个寄存器一样对该寄存器进行写操作。可以单独对 L 寄存器和 H 寄存器执行读 / 写操作，也可在单一的 32 位读 / 写操作中对它们执行操作。在本例中，L 寄存器和 H 寄存器可在其他寄存器的控制下进行单独计数。
在计数器运行时尝试对计数器进行写操作不会影响计数器，但会产生总线错误。软件可以随时对计数器寄存器进行读操作。

表 115. SCT 计数器寄存器 （COUNT，地址 0x5000 4040）位说明

位	符号	说明	复位值
15:0	CTR_L	当 UNIFY = 0 时，读取或写入 16 位 L 计数器值。当 UNIFY = 1 时，读取或写入 32 位统一计数器的较低 16 位。	0
31:16	CTR_H	当 UNIFY = 0 时，读取或写入 16 位 H 计数器值。当 UNIFY = 1 时，读取或写入 32 位统一计数器的较高 16 位。	0

10.6.8 SCT 状态寄存器

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用 _L 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会如同对 STATE_L 和 STATE_H 这两个寄存器一样对该寄存器进行写操作。可以单独对 L 寄存器和 H 寄存器执行读 / 写操作，也可在单一的 32 位读 / 写操作中对它们执行操作。

软件可以随时读取与计数器相关的状态。仅当计数器的 HALT 位为 1 时才允许写入该状态；当 HALT 为 0 时，写入尝试不会更改该状态，但会导致总线错误。

状态变量是区分 SCT 与其他计数器 / 定时器 /PWM 模块的主要方式。仅在特定状态下才产生事件。而事件可以执行下列操作：

- 设置和清零输出
- 限制、停止和启动计数器
- 造成中断
- 修改状态变量

状态变量值完全在应用程序的控制之下。如果应用程序不使用状态，则状态变量值将保留为零，这是系统默认值。

可以使用状态变量跟踪和控制任何所需操作序列中的相关计数器的多个周期。状态变量在逻辑上与代表 SCT 配置的状态机图相关。有关状态与事件之间关系的更多信息，请参见章节 10.6.22 和 10.6.23。

所有已定义事件的事件控制寄存器STATELD/STADEV字段可设置状态变量的所有可能值。在多个计数器周期期间，状态变量的变化反映了相关状态机是如何从一种状态变为另一种状态的。

表 116. SCT 状态寄存器 （STATE，地址 0x5000 4044）位说明

位	符号	说明	复位值
4:0	STATE_L	状态变量。	0
15:5	-	保留。	-
20:16	STATE_H	状态变量。	0
31:21	-	保留。	

10.6.9 SCT 输入寄存器

软件可以稍微不同的两种格式来读取此只读寄存器中 SCT 输入的状态。如果要以完全不同的两种方式来读取该状态，则唯一的条件是 CONFIG 寄存器中的 CLKMODE 等于 2。

表 117. SCT 输入寄存器 （INPUT，地址 0x5000 4048）位说明

位	符号	说明	复位值
0	AIN0	输入 0 的实时状态。	引脚
1	AIN1	输入 1 的实时状态。	引脚
2	AIN2	输入 2 的实时状态。	引脚
3	AIN3	输入 3 的实时状态。	引脚
15:4	-	保留。	-
16	SIN0	同步到 SCT 时钟的输入 0 状态。	-
17	SIN1	同步到 SCT 时钟的输入 1 状态。	-
18	SIN2	同步到 SCT 时钟的输入 2 状态。	-
19	SIN3	同步到 SCT 时钟的输入 3 状态。	-
31:20	-	保留	-

10.6.10 SCT 匹配 / 捕获寄存器模式寄存器

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用该寄存器中的 _L 位。 L 位可控制每组匹配 / 捕获寄存器是否用作统一的 32 位捕获 / 匹配寄存器。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会如同对 REGMODE_L 和 REGMODE_H 这两个寄存器一样对该寄存器进行写操作。可单独对 L 寄存器和 H 寄存器执行读 / 写操作，也可以同时对它们执行单一的 32 位读 / 写操作。_L 位 / 寄存器用于控制 L 匹配 / 捕获寄存器，而 _H 位 / 寄存器用于控制 H 匹配 / 捕获寄存器。

SCT 包含了 5 个匹配 / 捕获寄存器对。“ 寄存器模式 ” 寄存器可选择每个寄存器对是用作匹配寄存器 （参见章节 10.6.18）还是捕获寄存器 （参见章节 10.6.19）。每个匹配 / 捕获寄存器都有一个随附寄存器，在将寄存器用作匹配寄存器（章节 10.6.20）时被用作重新载入寄存器，或者在将寄存器用作捕获寄存器（章节 10.6.21）时被用作捕获控制寄存器。仅在 UNIFY 位为 0 时才会使用 REGMODE_H。

表 118. SCT 匹配 / 捕获寄存器模式寄存器 （REGMODE，地址 0x5000 404C）位说明

位	符号	说明	复位值
4:0	REGMOD_L	每个位会控制一对匹配 / 捕获寄存器 （寄存器 0= 位 0，寄存器 1= 位 1， ...，寄存器 4= 位 4）。 0 表示寄存器用作匹配寄存器。 1 表示寄存器用作捕获寄存器。	0
15:5	-	保留。	-
20:16	REGMOD_H	每个位会控制一对匹配 / 捕获寄存器（寄存器 0= 位 16，寄存器 1= 位 17， ...，寄存器 4= 位 20）。 0 表示寄存器用作匹配寄存器。 1 表示寄存器用作捕获寄存器。	0
31:21	-	保留。	-

10.6.11 SCT 输出寄存器

SCT 支持 4 个输出，其中每个输出在该寄存器中都有相应的位。
当终止两个计数器来直接控制输出时，软件可以对任何一个输出寄存器执行写入操作。在任何一个计数器停止或运行时，对该寄存器执行写操作不会影响输出，但会导致总线错误。

软件可随时对该寄存器执行读操作来了解输出的状态。

表 119. SCT 输出寄存器 （OUTPUT，地址 0x5000 4050）位说明

位	符号	说明	复位值
3:0	OUT	将 1 写入位 n 会使相应的输出变为高电平。写入 0 会使相应的输出变为低电平 （输出 0= 位 0、输出 1= 位 1、 ...、输出 3= 位 3）。	0
31:4	-	保留	

10.6.12 SCT 双向输出控制寄存器

该寄存器可指定 （针对每个输出）计数方向对输出的设置和清零操作的影响 （参见章节 10.6.24 和章节 10.6.25）。

表 120. SCT 双向输出控制寄存器 （OUTPUTDIRCTRL，地址 0x5000 4054）位说明

位	符号	值	说明	复位值
1:0	SETCLR0		输出 0 的设置 / 清零操作。保留值 0x3。不要编写该值。	0
		0x0	任意。设置操作和清零操作与任何计数器无关。	
		0x1	L 向下计数。当计数器 L 或统一计数器向下计数时，会反向执行设置操作和清零操作。	
		0x2	H 向下计数。当计数器 H 向下计数时，会反向执行设置操作和清零操作。如果 UNIFY = 1，则不要使用该位。	
3:2	SETCLR1		输出 1 的设置 / 清零操作。保留值 0x3。不要编写该值。	0
		0x0	任意。设置操作和清零操作与任何计数器无关。	
		0x1	L 向下计数。当计数器 L 或统一计数器向下计数时，会反向执行设置操作和清零操作。	
		0x2	H 向下计数。当计数器 H 向下计数时，会反向执行设置操作和清零操作。如果 UNIFY = 1，则不要使用该位。	

表 120. SCT 双向输出控制寄存器 （OUTPUTDIRCTRL，地址 0x5000 4054）位说明

位	符号	值	说明	复位值
5:4	SETCLR2		输出 2 的设置 / 清零操作。保留值 0x3。不要编写该值。	0
		0x0	任意。设置操作和清零操作与任何计数器无关。	
		0x1	L 向下计数。当计数器 L 或统一计数器向下计数时，会反向执行设置操作和清零操作。	
		0x2	H 向下计数。当计数器 H 向下计数时，会反向执行设置操作和清零操作。如果 UNIFY = 1，则不要使用该位。	
7:6	SETCLR3		输出 3 的设置 / 清零操作。保留值 0x3。不要编写该值。	0
		0x0	任意。设置操作和清零操作与任何计数器无关。	
		0x1	L 向下计数。当计数器 L 或统一计数器向下计数时，会反向执行设置操作和清零操作。	
		0x2	H 向下计数。当计数器 H 向下计数时，会反向执行设置操作和清零操作。如果 UNIFY = 1，则不要使用该位。	
31:8	-		保留	-

10.6.13 SCT 冲突解决寄存器

寄存器 OUTn_SETn （章节 10.6.24）和 OUTnCLRn （章节 10.6.25）都允许对同一时钟周期中（甚至是相同事件）的输出指示设置和清零操作。该 SCT 冲突解决寄存器可解决该冲突。

要使能事件来切换输出，则在该寄存器中将 OnRES 值设置为 0x3，并在设置寄存器和清零寄存器中设置事件位。

表 121. SCT 冲突解决寄存器 （RES，地址 0x5000 4058）位说明

位	符号	值	说明	复位值
1:0	O0RES		输出 0 的同步设置和清零效应。	0
		0x0	无变化。	
		0x1	基于 SETCLR0 字段的设置输出 （或清零输出）。	
		0x2	基于 SETCLR0 字段的清零输出 （或设置输出）。	
		0x3	切换输出。	
3:2	O1RES		输出 1 的同步设置和清零效应。	0
		0x0	无变化。	
		0x1	基于 SETCLR1 字段的设置输出 （或清零输出）。	
		0x2	基于 SETCLR1 字段的清零输出 （或设置输出）。	
		0x3	切换输出。	
5:4	O2RES		输出 2 的同步设置和清零效应。	0
		0x0	无变化。	
		0x1	基于 SETCLR2 字段的设置输出 （或清零输出）。	
		0x2	基于 SETCLR2 字段的清零输出 （或设置输出）。	
		0x3	切换输出。	
7:6	O3RES		输出 3 的同步设置和清零效应。	0
		0x0	无变化。	
		0x1	基于 SETCLR3 字段的设置输出 （或清零输出）。	
		0x2	基于 SETCLR3 字段的清零输出 （或设置输出）。	
		0x3	切换输出。	
31:8	-	-	保留	-

10.6.14 SCT 标志使能寄存器

如果在 SCT 事件标志寄存器（章节 10.6.15）中同样也设置了 FLAGn 位，则该寄存器会使能请求中断的标志。

表 122. SCT 标志使能寄存器 （EVEN，地址 0x5000 40F0）位说明

位	符号	说明	复位值
5:0	IEN	当该寄存器和事件标志寄存器的位 n 均等于 1 时， SCT 会请求中断（事件 0= 位 0，事件 1= 位 1， ...，事件 5= 位 5）。	0
31:6	-	保留	

10.6.15 SCT 事件标志寄存器

该寄存器会记录各种事件。将 1 写入该寄存器可清零相应的标志，并且如果所有使能的标志位为零，则会取消 SCT 中断请求。

表 123. SCT 事件标志寄存器 （EVFLAG，地址 0x5000 40F4）位说明

位	符号	说明	复位值
5:0	FLAG	在进行复位后或将 1 最后写入该位后，如果发生了事件 n，则位 n 为 1（事件 0= 位 0，事件 1= 位 1， ...，事件 5= 位 5）。	0
31:6	-	保留	-

10.6.16 SCT 冲突使能寄存器

该寄存器能够使 SCT 冲突解决寄存器中所指定的 “ 无变化冲突 ” 事件进行 IRQ 请求。

表 124. SCT 冲突使能寄存器 （CONEN，地址 0x5000 40F8）位说明

位	符号	说明	复位值
3:0	NCEN	该寄存器和 SCT 冲突标志寄存器的位 n 均等于 1 时， SCT 会请求中断 （事件 0= 位 0，事件 1= 位 1， ...，事件 3= 位 3）。	0
31:4	-	保留	

10.6.17 SCT 冲突标志寄存器

该寄存器会记录中断使能的无变化冲突事件，并提供详细的总线错误信息。将 1 写入 NCFLAG 位可清零相应的读取位，并且如果所有使能的标志位为零，则会取消 SCT 中断请求。

表 125. SCT 冲突标志寄存器 （CONFLAG，地址 0x5000 40FC）位说明

位	符号	说明	复位值
3:0	NCFLAG	进行复位后或将 1 最后写入该位后，如果输出 n 发生了无变化冲突事件，则位n为1（输出0=位0，输出1=位1，...，输出3=位3）。	0

表 125. SCT 冲突标志寄存器 （CONFLAG，地址 0x5000 40FC）位说明

位	符号	说明	复位值	位
29:4	-	保留。	-	
30	BUSERRL	该 SCT 中出现的最新总线错误，与在 L/U 计数器未停止运行时对 CTR L/ 统一寄存器、状态 L/ 统一寄存器、匹配 L/ 统一寄存器或输出寄存器执行写操作有关。对特定的 L 寄存器和 H 寄存器执行字写入操作，其成功概率只占一半。	0	
31	BUSERRH	该 SCT 中所出现的最新总线错误，与在 H 计数器未停止运行时对 CTR H、 STATE H、 MATCH H 或输出寄存器执行写入操作有关。	0	

10.6.18 SCT 匹配寄存器 0 至 4 （REGMODEn 位 =0）

匹配寄存器通过与计数器进行比较来创建事件。当 UNIFY 位为 0 时， L 寄存器和 H 寄存器分别与 L 计数器和 H 计数器进行比较。当 UNIFY 为 1 时，L 寄存器和 H 寄存器会保留与统一计数器比较的 32 位值。匹配事件仅发生在计数器运行 （STOP 和 HALT 均为 0）所在的时钟内。

在任何时候都可对匹配寄存器进行读操作。相关计数器运行时，如果对匹配寄存器执行写操作，不会影响匹配寄存器，但会造成总线错误。匹配事件发生在 SCT 时钟内，在该时钟内，计数器的计数递增至下一个值 （或将递增至下一个值）。“ 匹配 ” 事件限制其计数器 （如章节 10.6.3 所述）时，“ 匹配 ” 寄存器中的值在其清零之前将为该计数器的最后一个值（或在 BIDIR 为 1 时递减）。

无论是重新载入寄存器，还是匹配寄存器，都不能采用 “ 透写 ” 机制。启动计数器之前，软件可将值 1 写入在计数器第一个周期中使用的匹配寄存器，并且将另一个值写入在计数器第二个周期中所对应的匹配重新载入寄存器。

表 126. SCT 匹配寄存器 0 至 4 （MATCH[0:4]，地址 0x5000 4100 (MATCH0) 至 0x5000 4110 (MATCH4)）位说明 （REGMODEn 位 = 0）

位	符号	说明	复位值
15:0	VALMATCH_L	当 UNIFY = 0 时，读取或写入要与 L 计数器进行比较的 16 位值。当 UNIFY = 1 时，读取或写入要与统一计数器进行比较的 32 位值的较低 16 位。	0
31:16	VALMATCH_H	当 UNIFY = 0 时，读取或写入要与 H 计数器进行比较的 16 位值。当 UNIFY = 1 时，读取或写入要与统一计数器进行比较的 32 位值的较高 16 位。	0

10.6.19 SCT 捕获寄存器 0 到 4 （REGMODEn 位 = 1）

这些寄存器允许软件读取计数器的值，在达到所对应的计数器的值时会发生捕获控制寄存器所选取的事件。

表 127. SCT 捕获寄存器 0 到 4 （CAP[0:4]，地址 0x5000 4100 (CAP0) 至 0x5000 4110 (CAP4)）位说明 （REGMODEn 位 = 1）

位	符号	说明	复位值
15:0	VALCAP_L	当 UNIFY = 0 时，读取最后一次捕获该寄存器时的 16 位计数器值。当 UNIFY = 1 时，读取最后一次捕获该寄存器时的 32 位值的较低 16 位。	0
31:16	VALCAP_H	当 UNIFY = 0 时，读取最后一次捕获该寄存器时的 16 位计数器值。当 UNIFY = 1 时，读取最后一次捕获该寄存器时的 32 位值的较高 16 位。	0

10.6.20 SCT 匹配重新载入寄存器 0 至 4 （REGMODEn 位 = 0）

BIDIR 为 0 且计数器的值到达其限值范围、或 BIDIR 为 1 且计数器的值到达 0 时，匹配寄存器 （L 寄存器、 H 寄存器或统一 32 位寄存器）会从相应的重新载入寄存器中加载。

表 128. SCT 匹配寄存器 0 到 4 （MATCHREL[0:4]，地址 0x5000 4200 (MATCHREL0) 至 0x5000 4210 (MATCHREL4)）位说明 （REGMODEn 位 = 0）

位	符号	说明	复位值
15:0	RELOAD_L	当 UNIFY = 0 时，读取或写入加载入 SCTMATCHn_L 寄存器的 16 位值。当 UNIFY = 1 时，将读取或写入加载入 MATCHn 寄存器的 32 位值的较低 16 位。	0
31:16	RELOAD_H	当 UNIFY = 0 时，读取或写入加载入 MATCHn_H 寄存器的 16 位值。当 UNIFY = 1 时，读取或写入加载入 MATCHn 寄存器的 32 位值的较高 16 位。	0

10.6.21 SCT 捕获控制寄存器 0 到 4 （REGMODEn 位 =1）

如果 CONFIG 寄存器中的 UNIFY 等于 1，则仅使用 _L 位。

如果 CONFIG 寄存器中的 UNIFY 等于 0，则会如同对 CAPCTRLn_L 和 CAPCTRLn_H 这两个寄存器一样对该寄存器进行写操作。可以单独对 L 寄存器和 H 寄存器执行读 / 写操作，也可在单一的 32 位读 / 写操作中对它们执行操作。

每个捕获控制寄存器（L 寄存器、H 寄存器或统一 32 位寄存器）可控制使用哪个事件从计数器加载相应的捕获寄存器。

表 129. SCT 捕获控制寄存器 0 到 4（CAPCTRL[0:4]，地址 0x5000 4200 (CAPCTRL0) 至 0x5000 4210 (CAPCTRL4)）位说明 （REGMODEn 位 = 1）

位	符号	说明	复位值
5:0	CAPCONm_L	如果位 m 等于 1，则事件 m 会加载 CAPn_L (UNIFY = 0) 或 CAPn (UNIFY = 1)寄存器（事件0=位0，事件1=位1，...，事件5=位5）。	0
15:6	-	保留。	-
21:16	CAPCONm_H	如果位 m 等于 1，则事件 m 会加载 CAPn_H (UNIFY = 0) 寄存器（事件 0= 位 16，事件 1= 位 17， ...，事件 5= 位 21）。	0
31:22	-	保留。	-

10.6.22 SCT 事件状态掩码寄存器 0 到 5

每个事件都有一个相关的 SCT 事件状态掩码寄存器，它允许该事件以 HEVENT 位 （位于相应的 EVn_CTRL 寄存器中）所选的计数器的一个或多个状态中下发生。

当其 EVn_STATE 寄存器全部为零时，事件 n 会被禁用，这是因为无论当前状态如何它都会被屏蔽。

在不使用各状态的简单应用中，将 0x01 写入该寄存器可使能事件。由于该状态值始终保持为其复位值 0，因此写入 0x01 可始终通过状态来使能该事件。

表 130. SCT 事件状态掩码寄存器 0 到 5 （EV[0:5]_STATE，地址 0x5000 4300 (EV0_STATE) 至 0x5000 4328 (EV5_STATE)）位说明

位	符号	说明	复位值
1:0	STATEMSKm	如果位 m 等于 1，则事件 n（n= 0 至 5）将发生在 HEVENT 位所选的计数器的 m 状态（m= 状态号；状态 0= 位 0，状态 1= 位 1）。	0
31:2	-	保留。	-

10.6.23 SCT 事件控制寄存器 0 到 5

该寄存器用于定义事件 n 发生的条件，而不是状态掩码寄存器定义的状态变量。大部分事件都与特定的计数器（高速计数器、低速计数器或统一计数器）有关，因此，事件取决于是否与该寄存器相匹配。事件的其他可能组成部分为选定的输入信号或输出信号。

当 UNIFY 位为 0 时，每个事件都与其事件控制寄存器中 HEVENT 位所选的特定计数器有关。当事件相关的计数器停止运行或当前的状态无法触发其事件掩码寄存器中所指定的事件时，则不会发生事件。当其事件状态掩码寄存器始终为 0 时，将会永久禁用事件。

根据所选的输入（输出）边沿或输入（输出）电平以及 / 或者根据与所选的匹配寄存器相匹配的计数器值，可对已启用的事件编写来使其发生（STOP 位 = 0）。事件可通过事件计数器的 HALT 位和 STATE 寄存器使能。在双向模式中，还可根据计数方向使能事件。

每个事件都可以修改其计数器的 STATE 值。如果在指定的时钟周期内发生了与相同计数器相关的多个事件，则仅会发生为最大编号事件指定的状态更改。任何同步发生的事件所指示的其他操作也会发生。

表 131. SCT 事件控制寄存器 0 到 5（EV[0:5]_CTRL，地址 0x5000 4304 (EV0_CTRL) 至 0x5000 432C (EV5_CTRL)）位说明

位	符号	值	说明	复位值
3:0	MATCHSEL	-	选择与该事件（如果存在）相关的匹配寄存器。仅在 HEVENT 位所选的计数器运行时，才会发生匹配事件。	0
4	HEVENT		选择 L/H 计数器。如果 UNIFY = 1，不设置该位。	0
		0	L 状态。选择由 MATCHSEL 所选的 L 状态寄存器和 L 匹配寄存器。	
		1	H 状态。选择由 MATCHSEL 所选的 H 状态寄存器和 H 匹配寄存器。	
5	OUTSEL		输入 / 输出选择	0
		0	输入。选择 IOSEL 所选的输入。	
		1	输出。选择 IOSEL 所选的输出。	
9:6	IOSEL	-	选择与该事件（如果存在）相关的输入或输出信号。如果 CLKMODE 为 1x，则不在该寄存器中选择输入。这种情况下，时钟输入是每个事件的隐式部分。 IOSEL = 0 选择引脚 CTIN_0 或 CTOUT_0，...，IOSEL = 3 选择引脚 CTIN_3 或 CTOUT_3。	0
11:10	IOCOND		选择事件 n 的 I/O 条件（对输出边沿的检测操作会使通过一个 SCT 时钟来切换输出的条件发生延迟）。输入必须具有至少一个 SCT 时钟周期的最小脉冲宽度，这样才能确保进行正确的边沿 / 状态检测。	0
		0x0	低电平	
		0x1	上升	
		0x2	下降	
		0x3	高电平	
13:12	COMBMODE		选择指定的匹配和 I/O 条件的使用方法和合并方法。	
		0x0	或条件。无论是出现指定的匹配，还是出现 I/O 条件，都将发生该事件。	
		0x1	匹配条件。仅使用指定的匹配条件。	
		0x2	IO。仅使用指定的 I/O 条件。	
		0x3	与条件。在同时出现指定的匹配与 I/O 条件时将发生该事件。	
14	STATELD		如果该事件是在该状态下所发生的最大编号事件，则该位会控制如何使用 STATEV 值来修改 HEVENT 所选的状态。	
		0	加法。STATEV 值被添加入 STATE（将忽略进位输出）。	
		1	加载。STATEV 值被加载入 STATE。	

表 131. SCT 事件控制寄存器 0 到 5（EV[0:5]_CTRL，地址 0x5000 4304 (EV0_CTRL) 至 0x5000 432C (EV5_CTRL)）位说明

位	符号	值	说明	复位值
19:15	STATEV		如果该事件是在该状态下所发生的最大编号事件，则该值将加载或添加到 HEVENT 所选的状态中，具体取决于 STATELD。如果 STATELD 和 STATEV 均为零，则 STATE 值不会发生变化。	
20	MATCHMEM		如果该位为 1 且 COMBMODE 字段指定了匹配组件以触发该事件，则在计数器值不小于匹配寄存器中的值（向上计数）或不大于匹配值（向下计数）时，匹配即视为有效。 如果该位为 0，则匹配仅在计数器等于匹配值的技术周期内才有效。	
22:21	方向		用于事件生成的方向验证器。该字段只适用于计数器在 BIDIR 模式下工作的情况。如果 BIDIR = 0， SCT 会忽略该字段。保留值 0x3。	
		0x0	独立于方向。无论计数方向，该事件均会触发。	
		0x1	向上计数。 BIDIR = 1 时，该事件仅在向上计数时才触发。	
		0x2	向下计数。 BIDIR = 1 时，该事件仅在向下计数时才触发。	
31:23	-		保留	

10.6.24 SCT 输出设置寄存器 0 至 3

每个输出 n 都有一个设置寄存器，可控制事件对每个输出的影响方式。对输出执行设置操作还是清零操作，取决于 SCT OUTPUTDIRCTRL 寄存器中的 SETCLRn 字段设置。

表 132. SCT 输出设置寄存器 （OUT[0:3]_SET，地址 0x5000 4500 (OUT0_SET) 至 0x5000 4518 (OUT3_SET)）位说明

位	符号	说明	复位值
5:0	设置	位 m 为 1 时，会选择事件 m 来设置输出 n，而在 SETCLRn = 0x1 或0x2时，会清零输出n（事件0=位0，事件1=位1，...，事件5=位5）。	0
31:6	-	保留	

10.6.25 SCT 输出清零寄存器 0 到 3

每个输出 n 都有一个清零寄存器，可控制事件对每个输出的影响方式。对输出执行设置操作还是清零操作，要取决于 OUTPUTDIRCTRL 寄存器中的 SETCLRn 字段设置。

表 133. SCT 输出清零寄存器 （OUT[0:3]_CLR，地址 0x5000 0504 (OUT0_CLR) 至 0x5000 051C (OUT3_CLR)）位说明

位	符号	说明	复位值
5:0	CLR	位 m 为 1 时，会选择事件 m 来清零输出 n，而在 SETCLRn = 0x1 或0x2时，会设置输出n（事件0=位0，事件1=位1，...，事件5=位5）。	0
31:6	-	保留	

10.7 功能说明

10.7.1 匹配逻辑

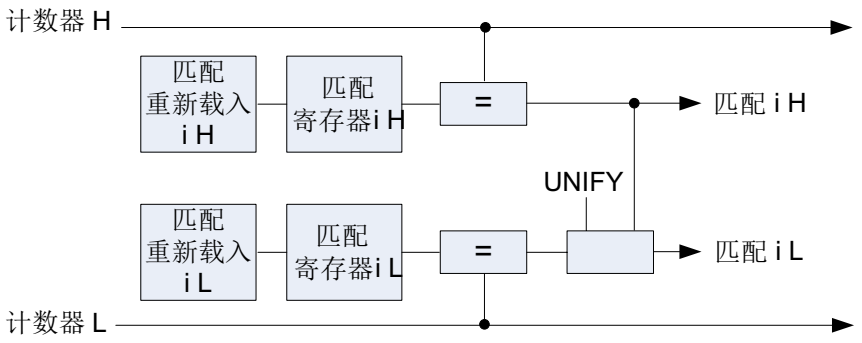


图 16. 匹配逻辑

10.7.2 捕获逻辑

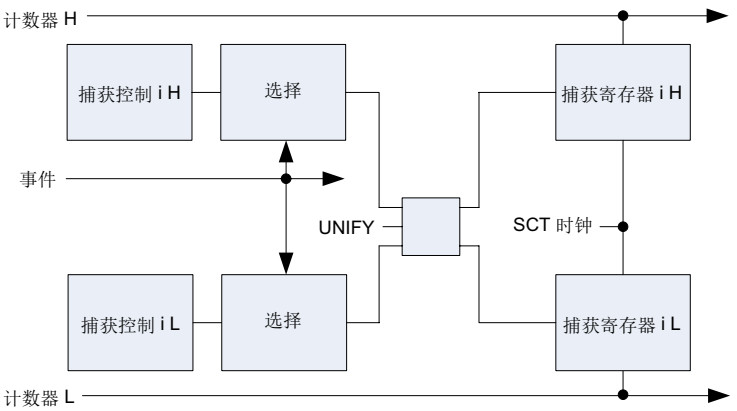


图 17. 捕获逻辑

10.7.3 事件选择

状态变量可控制跨计数器多个周期的 SCT。计数器匹配、输入 / 输出边缘和状态值将会合并为一组通用的事件，这些事件可以切换输出、请求中断，并更改状态值。

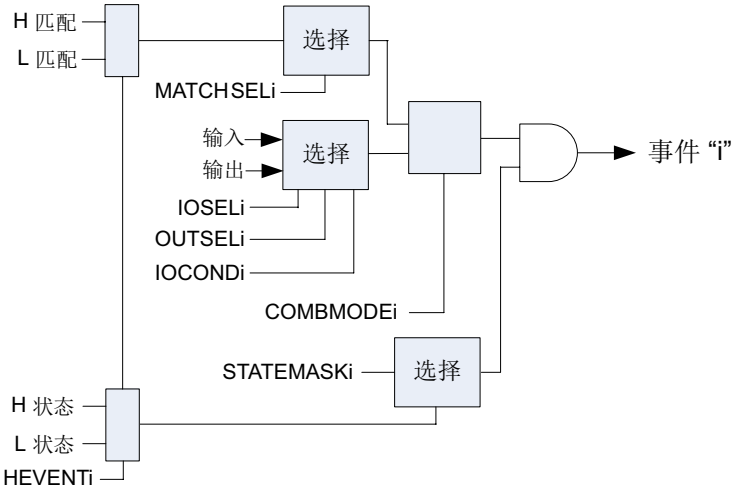


图 18. 事件选择

10.7.4 输出生成

图 19 显示的是 SCT 的某个输出削波。

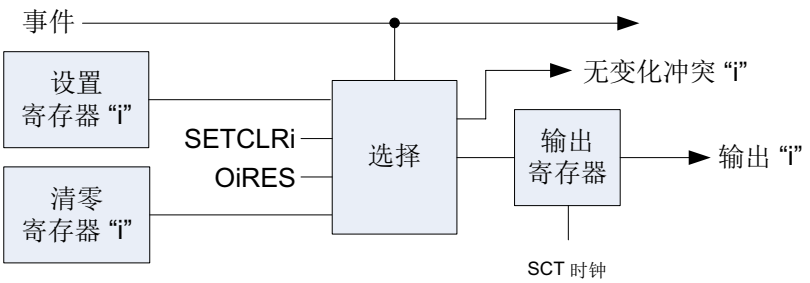


图 19. 输出片 i

10.7.5 中断生成

SCT 会为 NVIC 生成一个中断。

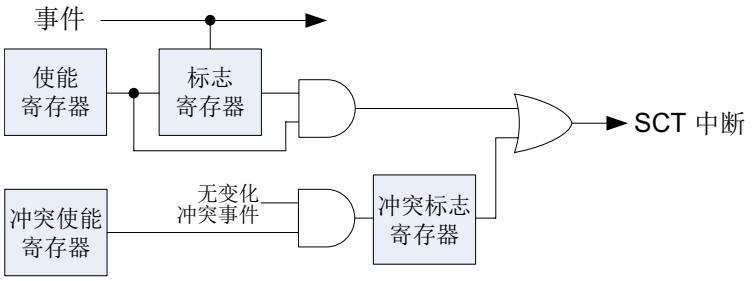


图 20. SCT 中断生成

10.7.6 清零预分频器

预分频器由控制寄存器中的非零 PRE 字段使能后，如同计数器值的小数部分一样，会作为计数器的时钟分频器。计数器因下列任何原因被清零或加载时，预分频器会被清零：

- 硬件复位
- 软件对计数器寄存器执行写操作
- 在控制器寄存器中将 1 写入 CLRCTR 位的软件
- BIDIR 等于 0 时，计数器的限值寄存器中的 1 会选择某个事件

BIDIR 为 0 时，I/O 信号产生的限值事件会清零非零预分频器。但匹配所产生的限值事件仅会在一个特殊的情况下清零非零预分频器，如章节 10.7.7 所述。

当 BIDIR 为 1 时，限值事件不会清零预分频器，而是会清零控制寄存器中的 DOWN 位，并且如果在该时钟内使能计数器，则会在相同的时钟上使计数器的计数递减。

10.7.7 匹配事件与 I/O 事件

预分频器以及时钟模式 01（SCT 时钟为总线时钟）会使计数器操作复杂化。然而，仅在时钟输入中检测到所选的边缘时才会启用预分频器和计数器来计数。

- 如果时钟模式非 01 或检测到 CLKSEL 字段所选的输入边缘时，会启用预分频器。
- 预分频器使能时，以及 PRELIM=0 或预分频器等于 PRELIM 中的值时，计数器使能。

当其计数器的 HALT 位为 0 时，会在任何 SCT 时钟内出现事件的 I/O 组件。一般情况下，当事件的计数器的 HALT 位和 STOP 位均为 0 且计数器已使能时，事件的匹配组件仅会出现在 UT 时钟内。

表 134 显示的是各种事件可能发生的时间。

表 134. 事件条件

COMBMODE	IOMODE	在时钟内发生事件：
IO	任意	当 HALT = 0 （类型 A）时可能会发生事件。
MATCH	任意	当 HALT = 0、STOP = 0 且计数器（类型 C）使能时可能会发生事件。
OR	任意	就 IO 组件而言：当 HALT = 0 （类型 A）时可能会发生事件。 就匹配组件而言：当 HALT = 0、 STOP = 0 且计数器 （类型 C）使能时可能会发生事件。
AND	低电平或高电平	当 HALT = 0、STOP = 0 且计数器（类型 C）使能时可能会发生事件。
AND	上升或下降	当 HALT = 0 （类型 A）时可能会发生事件。

10.7.8 SCT 操作

在其最简单的、单一状态配置中， SCT 将用作事件控制的单向或双向计数器。事件可配置为计数器匹配事件、输入或输出电平、输入或输出引脚的跃迁或者匹配与输入 / 输出行为的组合。在响应事件时， SCT 输出可以跃迁或 SCT 可以执行其他操作，如创建中断或者起动、停止或复位计数器。每个事件都可以包含多个同步操作。并且，任意数量的事件都可以触发 SCT 的一个特定操作。

事件由 SCT 的一个或多个操作唯一性地定义。状态是通过触发计数器任意阶段中的某个 SCT 操作或多个操作所启用的事件来定义的。系统会忽略该状态未选择的事件。

在多状态配置中，状态会随着事件而发生更改。状态更改是在事件发生时 SCT 可以执行的另一种操作。在将事件配置为更改状态时，新的状态会定义一组用于产生其他 SCT 操作的新事件。在计数器的多个周期内，事件可以多次更改状态，并因此创建 SCT 输出和 / 或中断的大量事件控制的跃迁。

在完成配置后， SCT 可以在无软件介入的情况下连续运行，并且可以生成多个完全受事件所控制的输出模式。

- 要配置 SCT，请参见章节 10.7.9。
- 要对 SCT 执行启动、运行和停止操作，请参见章节 10.7.10。
- 要将 SCT 配置为简单的事件控制计数器 / 定时器，请参见章节 10.7.11。

10.7.9 配置 SCT

要为 SCT 设置多个事件和多个状态，则执行下列配置步骤：

10.7.9.1 配置计数器

1. 在 CONFIG 寄存器中配置 L 计数器和 H 计数器，方法是在 UNIFY 字段中选择两个独立的 16 位计数器 （L 计数器和 H 计数器）或一个组合的 32 位计数器。
2. 从任意输入或内部时钟中为 CONFIG 寄存器（CLKMODE 字段和 CLKSEL 字段）选择 SCT 时钟源。

10.7.9.2 配置匹配寄存器和捕获寄存器

1. 选择应用程序使用多少个匹配寄存器和捕获寄存器 （总共 5 个）：
   - 在 REGMODE 寄存器中，针对 5 个匹配 / 捕获寄存器对，选择将该寄存器用作匹配寄存器还是捕获寄存器。
2. 为每个所选的匹配寄存器定义匹配条件：
   - 如果使用 32 位计数器，则每个匹配寄存器 MATCH 可设置一个匹配值，如果使用 L 16 位计数器和 H 16 位计数器，则允许设置两个匹配值。
   - 计数器值达到其限值范围或值 0 时，每个匹配重新载入寄存器 MATCHRELOAD 将可对加载到匹配寄存器中的重新载入值进行设置。

10.7.9.3 配置事件和事件响应

1. 在 EVn_CTRL 寄存器 （共 6 个寄存器，每个事件对应一个寄存器）中定义每个事件将在下列何种情况下发生：
   - 在 COMBMODE 字段中选择是在输入或输出发生变化时，还是输入电平或输出电平、计数器的匹配条件或匹配和输入 / 输出条件的组合下发生该事件。
   - 对于匹配条件：

选择包含事件发生的匹配条件的匹配寄存器。在MATCHSEL字段中输入所选匹配寄存器的数量。

如果使用 L 计数器和 H 计数器，则定义在 HEVENT 字段中进行 L 计数器匹配或 H 计数器匹配时是否发生该事件。

– 对于 SCT 输入或输出电平或跃迁：

在字段 IOSEL 和 OUTSEL 中选择与该事件相关的输入编号或输出编号。

在IOCOND字段中定义所选的输入或输出将如何触发该事件（与边缘或电平相关）。

2. 在 OUTn_SET 或 OUTn_CLR 寄存器中定义每个事件对 SCT 输出（共 4 个输出，每个输出对应一个寄存器）的作用：
   – 对于每个 SCT 输出，选择使用哪些事件来设置或清零该输出。可以使用多个事件更改输出，并且每个事件可以更改多个输出。
3. 定义每个事件对计数器的影响：
   – 在 LIMIT 寄存器中设置相应的事件位，以通过该事件来设置计数器的上限值。

     在单向模式中发生限值事件时，计数器会被清零，并从下一个时钟沿开始向上计数。

     在双向模式中发生限值事件时，计数器会从下一个时钟沿的当前值开始向下计数。
   – 在 HALT 寄存器中设置相应的事件位，以通过该事件来终止计数器运行。如果该计数器终止运行，则它会停止计数，并且不会发生任何新事件。只有清零 CTRL 计数器中的 HALT_L 位和 / 或 HALT_H 位，才能恢复计数器操作。
   – 在 STOP 寄存器中设置相应的事件位，以通过该事件来停止计数器运行。如果该计数器停止运行，尽管计数停止，但是仍然可由配置为输入 / 输出跃迁的事件将其重新启动。
   – 在 START 寄存器中设置相应的事件位，以通过该事件来重新启动计数。只有输入变更定义的事件才能用于重新启动计数器。
4. 定义哪些事件可以触发 SCT 中断：
   – 在 EVEN 和 EVFLAG 寄存器中设置相应的事件位，以通过该事件来触发 SCT 中断。

10.7.9.4 配置多个状态

1. 在每个事件 （共 6 个事件，每个事件对应一个寄存器）的 EVn_STATE 寄存器中，选择发生该事件所处于的某个状态或多个状态（共 2 个）。所选的每个状态可用于多个事件。
2. 确定事件对系统状态的影响：

   在 EVn_CTRL 寄存器（最多 6 个，每个事件一个寄存器）中，在 STATEV 字段中为该事件设置新的状态值。如果该事件是当前状态中的最大编号事件，则该值会添加到现有状态值中或替换现有的状态值，具体取决于 STATELD 字段。

   注：如果在当前状态中还具有其他更大编号的事件，则此事件无法更改状态。

   如果 STATEV 和 STATELD 值被设置为零，则状态不会发生改变。

10.7.9.5 其他选项

- 捕获寄存器的数量是特定的（可选）。每个捕获寄存器可被设定为在发生一个或多个事件时捕获计数器内容。
- 如果计数器在双向模式下，则可建立输出的设置和清零功能，具体取决于计数器是向上计数还是向下计数 （通过对 OUTPUTDIRCTRL 计数器执行写操作而确定）。

10.7.10 运行 SCT

1. 配置 SCT （参见表 10.7.9 “ 配置 SCT”）。
2. 对状态寄存器执行写操作来定义初始状态。默认情况下，初始状态为 0。
3. 要启动 SCT，则对 CTRL 寄存器执行写操作：
   - 清零计数器。
   - 清零或设置 STOP_L 位和 / 或 STOP_H 位。

     注：计数器将在清零 STOP 位后开始计数。如果设置了 STOP 位，则 SCT 将等待配有启动计数器的事件发生。
   - 对于每个计数器，可以选择单向计数或双向计数模式 （字段 BIDIR_L 和 / 或 BIDIR_H）。
   - 选择计数器时钟 （CTRL 寄存器）的预分频系数。
   - 清零 HALT_L 位和 / 或 HALT_H 位。默认情况下，计数器会终止运行，并且不会发生任何事件。
4. 要通过软件随时停止计数器运行，则在 CTRL 寄存器中写入 STOP_L 位和 / 或 STOP_H 位或 HALT_L 位和 / 或 HALT_H 位来停止或终止计数器运行。
   - 计数器停止运行时，可将事件配置为清零 STOP 位或通过软件将零写入 STOP 位，重新启动计数器。
   - 计数器终止运行时，只有可清零 HALT 位的软件写操作能重新启动计数器，并且不会发生任何事件。
   - 计数器终止运行时，软件可对 OUT 寄存器执行写操作，以直接将任何 SCT 输出设置为高电平或低电平。

对状态寄存器进行写操作可随时读取当前状态。

要通过软件更改当前状态（与任何事件的发生无关），则应设置 HALT 位并对状态寄存器执行写操作来更改状态值。计数器终止运行 （HALT_L 位和 / 或 HALT_H 位被设置）时，仅允许对状态寄存器执行写操作，并且此时不会发生任何事件。

10.7.11 在不使用状态的情况下配置 SCT

SCT 可用作带有外部捕获输入和匹配输出的标准计数器 / 定时器，但它不使用状态逻辑。要在无任何状态的情况下使用 SCT，则用下列方式配置 SCT：

- 将 0 写入状态寄存器 （默认值为 0）。
- 针对每个事件，将 0 写入 EVCTRL 寄存器中的 STATELD 字段和 STATEV 字段。
- 将 0x1 写入每个事件的 EVn_STATE 寄存器。写入 0x1 可使能事件。

  实际上，事件可以在计数器运行的同时，在无变化的单一状态下发生。

11.1 本章导读

所有 LPC800 器件都提供 MRT。

11.2 特性

- 31 位中断定时器
- 4 个通道从单独设置的数值开始，独立向下计数
- 重复模式和单次中断模式

11.3 基本配置

使用下列寄存器配置 MRT：

- 在 SYSAHBCLKCTRL 中，设置位 10 (表 18) 以使能寄存器接口时钟。
- 使用 PRESETCTRL 寄存器 （表 7）清零 MRT 复位。
- 全局 MRT 中断连接至 NVIC 的 10 号中断。

11.4 引脚说明

MRT 无可配置引脚。

11.5 简介

多速率定时器 (MRT) 提供具有 4 个通道的重复性中断定时器。每通道经过编程后可具有独立的时间间隔。

每个通道工作时均独立于其他通道，采用的工作模式为下列中的某一种：

- 重复中断模式。参见章节 11.5.1。
- 单次中断模式。参见章节 11.5.2。

每个定时器的工作模式在定时器控制寄存器中设置。参见表 138。

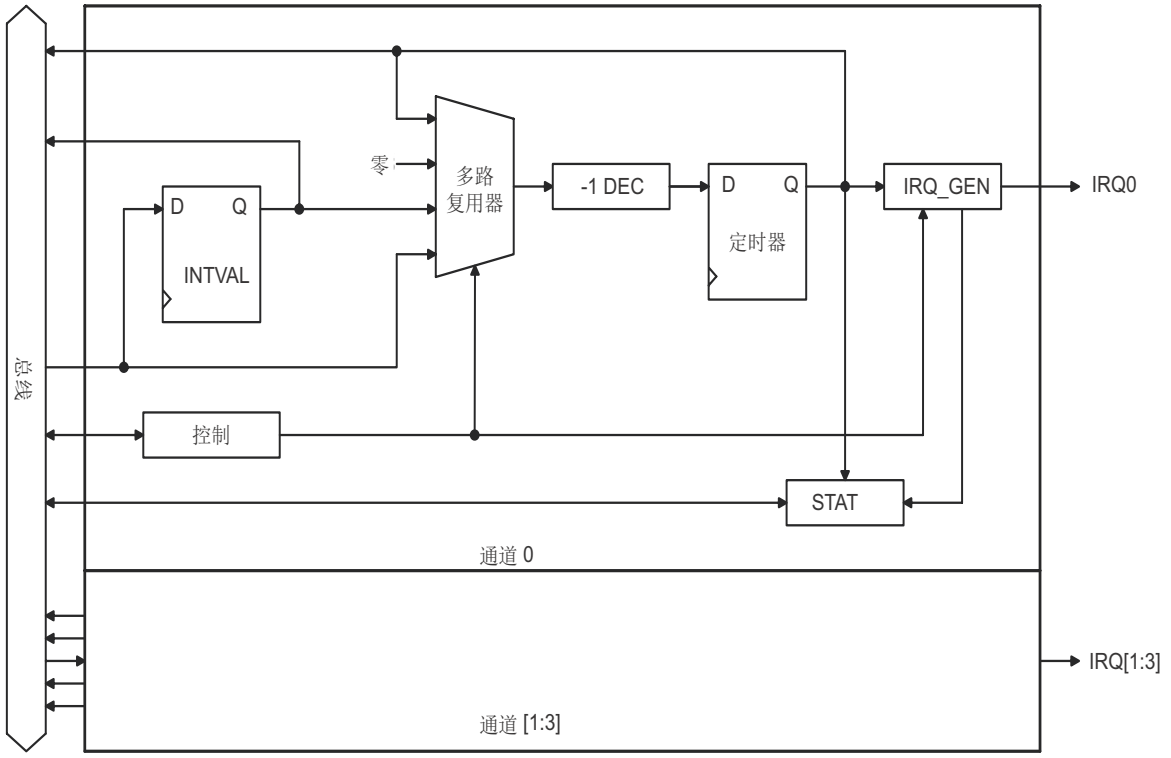


图 21. MRT 功能框图

11.5.1 重复中断模式

重复中断模式可在选定的时间间隔之后产生重复中断。该模式可用于基于软件的 PWM 或 PPM 应用。

定时器 n 处于空闲状态时，将非零值 IVALUE 写入 INTVALn 寄存器可立刻加载时间间隔值 IVALUE - 1，且定时器从该值开始向下计数。定时器达到 0 时会产生一个中断， INTVALn 寄存器中的数值 IVALUE - 1 会自动重新加载，并且定时器再次开始向下计数。

定时器在重复中断模式下运行时，可执行下列操作：

- 将新数值 （大于 0）写入 INTVALn 寄存器并将 LOAD 位设置为 0 可更改下一个定时器周期的间隔值。定时器达到 0 时会产生一个中断。定时器在下一个周期从新的数值开始向下计数。
- 通过将新数值 （大于 0）写入 INTVALn 寄存器并将 LOAD 位设置为 1，即可立即实时改变间隔值。定时器立刻开始从新的间隔值向下计数。定时器达到 0 时会产生一个中断。
- 将 0 写入 INTVALn 寄存器并将 LOAD 位设置为 0，即可在时间间隔结束后停止定时器。定时器达到 0 时会产生一个中断。
- 将 0 写入 INTVALn 寄存器并将 LOAD 位设置为 1，即可立即停止定时器。对 INTVALn 寄存器进行写操作时，不会产生任何中断。

11.5.2 单次中断模式

单次中断在完成一次计数后，会产生一次中断。该模式下，可在任何时刻产生单次中断。该模式可用于在软件任务中生成指定的延迟。

定时器处于空闲状态时，将非零值 IVALUE 写入 INTVALn 寄存器可立刻加载时间间隔值 IVALUE - 1，且定时器开始向下计数。定时器达到 0 时，会产生一个中断，随后定时器会停止工作并进入空闲状态。

定时器在单次中断模式下运行时，可执行下列操作：

- 用新的时间间隔值 （大于 0）更新 INTVALn 寄存器，并将 LOAD 位设置为 1。定时器会立即重新加载新的时间间隔值，并从新值开始向下计数。更新 TIME_INTVALn 寄存器时，不会产生任何中断。
- 将 0 写入 INTVALn 寄存器并将 LOAD 位设置为 1。定时器立即停止计数并进入空闲模式。更新 INTVALn 寄存器时，不会产生任何中断。

11.6 寄存器说明

表 136 中显示的复位值为 POR 复位值。

表 135. 寄存器概述：MRT （基址 0x4000 4000）

名称	访问类型	地址偏移	说明	复位值	参考
INTVAL0	R/W	0x0	MRT0 时间间隔数值寄存器。该值被载入 TIMER0 寄存器。	0	表 136
定时器 0	R	0x4	MRT0 定时器寄存器。该寄存器读取向下计数器的数值。	0x7FFF FFFF	表 137
CTRL0	R/W	0x8	MRT0 控制寄存器该寄存器控制 MRT0 模式。	0	表 138
STAT0	R/W	0xC	MRT0 状态寄存器。	0	表 139
INTVAL1	R/W	0x10	MRT1 时间间隔数值寄存器。该值被载入 TIMER1 寄存器。	0	表 136
定时器 1	R/W	0x14	MRT1 定时器寄存器。该寄存器读取向下计数器的数值。	0x7FFF FFFF	表 137
CTRL1	R/W	0x18	MRT1 控制寄存器该寄存器控制 MRT1 模式。	0	表 138
STAT1	R/W	0x1C	MRT1 状态寄存器。	0	表 139
INTVAL2	R/W	0x20	MRT2 时间间隔数值寄存器。该值被载入 TIMER2 寄存器。	0	表 136
定时器 2	R/W	0x24	MRT2 定时器寄存器。该寄存器读取向下计数器的数值。	0x7FFF FFFF	表 137
CTRL2	R/W	0x28	MRT2 控制寄存器该寄存器控制 MRT2 模式。	0	表 138
STAT2	R/W	0x2C	MRT2 状态寄存器。	0	表 139
INTVAL3	R/W	0x30	MRT3 时间间隔数值寄存器。该值被载入 TIMER3 寄存器。	0	表 136
定时器 3	R/W	0x34	MRT3 定时器寄存器。该寄存器读取向下计数器的数值。	0x7FFF FFFF	表 137
CTRL3	R/W	0x38	MRT3 控制寄存器该寄存器控制 MRT 模式。	0	表 138
STAT3	R/W	0x3C	MRT3 状态寄存器。	0	表 139
IDLE_CH	R	0xF4	空闲通道寄存器。该寄存器返回第一个空闲通道的编号。	0	表 140
IRQ_FLAG	R/W	0xF8	全局中断标志寄存器	0	表 141

11.6.1 时间间隔寄存器

该寄存器包含 MRT 加载值并控制定时器的重新加载方式。加载值为 IVALUE -1。

表 136. 时间间隔寄存器（INTVAL[0:3]，地址 0x4000 4000 (INTVAL0) 至 0x4000 4030 (INTVAL3)）位说明

位	符号	值	说明	复位值
30:0	IVALUE		时间间隔加载值。该值被载入 TIMERn 寄存器，并且 MRTn 开始从 IVALUE -1 向下计数。 若定时器空闲，则将非零值写入该位字段即可立即启动定时器。 若定时器正在工作中，则将 0 值写入该位字段后的情况如下： • 若 LOAD = 1，则定时器立即停止工作。 • 若 LOAD = 0，则定时器在时间间隔之后停止工作。	0
31	LOAD		决定定时器间隔值 IVALUE -1 如何载入 TIMERn 计时器。该位为只写。对该位进行读操作将始终返回 0。	0
		0	无强制加载。若选定重复模式，则 INTVALn 寄存器向 TIMERn 寄存器加载内容的操作将在时间间隔结束时进行。	
		1	强制加载。在 TIMERn 运行时，INTVALn 的间隔值 IVALUE -1 将被立即载入 TIMERn 寄存器。	

11.6.2 定时器寄存器

定时器寄存器可保持当前定时器值。该寄存器为只读寄存器。

表 137. 定时器寄存器 （TIMER[0:3]，地址 0x4000 4004 (TIMER0) 至 0x4000 4034 (TIMER3)）位说明

位	符号	说明	复位值
30:0	VALUE	保存向下计数器的当前定时器值。TIMERn 寄存器的初始值以 IVALUE - 1 的数值从 INTVALn 寄存器载入，载入时间为时间间隔结束时或立即载入，分为如下情况： INTVALn 寄存器在空闲状态下更新。 INTVALn 在 LOAD = 1 时更新。 定时器空闲时，读取该位字段将返回 -1 (0x00FF FFFF)。	0x00FF FFFF
31	-	保留。	0

11.6.3 控制寄存器

控制寄存器配置每个 MRT 的工作模式，并使能中断。

表 138. 控制寄存器 （CTRL[0:3]，地址 0x4000 4008 (CTRL0) 至 0x4000 4038 (CTRL3)）位说明

位	符号	值	说明	复位值
0	INTEN		使能 TIMERn 中断。	0
		0	禁用。	
		1	使能。	

表 138. 控制寄存器（CTRL[0:3]，地址0x4000 4008 (CTRL0)至0x4000 4038 (CTRL3)）位说明（续）

位	符号	值	说明	复位值
2:1	MODE		选择定时器模式。	0
		0x0	重复中断模式。	
		0x1	单次中断模式。	
		0x2	保留。	
		0x3	保留。	
31:3	-		保留。	0

11.6.4 状态寄存器

该寄存器表示每个 MRT 的状态。

表 139. 状态寄存器 （STAT[0:3]，地址 0x4000 400C (STAT0) 至 0x4000 403C (STAT3)）位说明

位	符号	值	说明	复位值
0	INTFLAG		监控中断标志。	0
		0	无挂起中断。写入 0 相当于无操作。	
		1	挂起中断。由于 TIMERn 到达时间间隔尾部，因此中断挂起。若 CONTROLn 中的 INTEN 位亦设为 1，则将触发定时器通道 n 中断和全局中断。 将 1 写入该位可清零中断请求。	
1	RUN		表示 TIMERn 的状态。该位为只读位。	0
		0	空闲状态。 TIMERn 停止。	
		1	运行。 TIMERn 正在运行。	
31:2	-		保留。	0

11.6.5 空闲通道寄存器

空闲通道寄存器可返回最小空闲通道的编号。当 STATUS 寄存器中的两个标志 （RUN 和 INTFLAG）均为 0 时，通道即视为空闲。

在多个定时器同时独立运行的应用中，可通过读取该寄存器中空闲通道的编号计算下一个空闲定时器的寄存器偏移量。空闲通道寄存器可让您无需检查每一个定时器的空闲状态即可设置下一个空闲定时器。

表 140. 空闲通道寄存器 （IDLE_CH，地址 0x4000 40F4）位说明

位	符号	说明	复位值
3:0	-	保留。	0
7:4	CHAN	空闲通道。读取 CHAN 位，返回最小空闲定时器通道值。若所有定时器通道都运行，则 CHAN = 4。	0
31:8	-	保留。	0

11.6.6 全局中断标志寄存器

全局中断寄存器将来自独立定时器通道的中断标志集中在一个寄存器中。设置 / 清零每个标志的操作与设置 / 清零每个 STATUSn 寄存器中 INTFLAG 位的操作相同。

表 141. 全局中断标志寄存器 （IRQ_FLAG，地址 0x4000 40F8）位说明

位	符号	值	说明	复位值
0	GFLAG0		监控 TIMER0 的中断标志。	0
		0	无挂起中断。写入 0 相当于无操作。	
		1	挂起中断。由于 TIMER0 到达时间间隔尾部，因此中断挂起。若 CONTROL0 寄存器中的 INTEN 位亦设为 1，则将触发定时器通道 0 中断和全局中断。 将 1 写入该位可清零中断请求。	
1	GFLAG1		监控 TIMER1 的中断标志。	0
		0	无挂起中断。写入 0 相当于无操作。	
		1	挂起中断。由于 TIMER1 到达时间间隔尾部，因此中断挂起。若 CONTROL1 寄存器中的 INTEN 位亦设为 1，则将触发定时器通道 1 中断和全局中断。 将 1 写入该位可清零中断请求。	
2	GFLAG2		监控 TIMER2 的中断标志。	0
		0	无挂起中断。写入 0 相当于无操作。	
		1	挂起中断。由于 TIMER2 到达时间间隔尾部，因此中断挂起。若 CONTROL2 寄存器中的 INTEN 位亦设为 1，则将触发定时器通道 2 中断和全局中断。 将 1 写入该位可清零中断请求。	
3	GFLAG3		监控 TIMER3 的中断标志。	0
		0	无挂起中断。写入 0 相当于无操作。	
		1	挂起中断。由于 TIMER3 到达时间间隔尾部，因此中断挂起。若 CONTROL3 寄存器中的 INTEN 位亦设为 1，则将触发定时器通道 3 中断和全局中断。 将 1 写入该位可清零中断请求。	
31:4	-		保留。	0

12.1 本章导读

所有 LPC800 器件的看门狗定时器都是一样的。

12.2 特性

- 如果在可编程超时期限内未重新载入，则可在内部复位芯片。
- 可选的窗口化操作要求在最大和最小超时期限之间发生重新载入，两个时限都可设定。
- 可选警报中断可在看门狗超时之前的可编程时间生成。
- 带内部固定预分频器的可编程 24 位定时器。
- 时间周期可选，从 1,024 个看门狗时钟 ($T_{WDCLK} \times 256 \times 4$) 到超过 6,700 万个看门狗时钟 ($T_{WDCLK} \times 2^{24} \times 4$)，取 4 个看门狗时钟的位数。
- “ 安全 ” 看门狗操作。一旦使能，则要求禁用软件复位或看门狗复位。
- 馈入序列不正确会导致看门狗即时事件 （如使能）。
- 由于对看门狗重载值的保护为选择性，因此只能在 “ 警报中断 ” 时间后更改该值。
- 指示看门狗复位的标志。
- 看门狗时钟 (WDCLK) 源是看门狗振荡器。
- 看门狗定时器可配置为在深度睡眠模式或掉电模式下运行。
- 调试模式。

12.3 基本配置

配置 WWDT 使用的是下列寄存器：

- 对寄存器接口供电 （WWDT PCLK 时钟）：在 SYSAHBCLKCTRL 寄存器中，设置表 18 中的位 17。
- 在 PDRUNCFG 寄存器 （表 37）中使能 WWDT 时钟源 （看门狗振荡器）。它是定时器的基础时钟源。
- 要从 WWDT 中断唤醒，则在 STARTERP1 寄存器 （表 34）中使能用于唤醒的看门狗中断。

12.4 引脚说明

WWDT 无外部引脚。

12.5 简介

看门狗定时器的作用是，如果微控制器进入错误状态，在可编程时间内将其复位或中断。看门狗定时器使能时，如果用户程序未能在预定时间内馈入 （或重新载入）看门狗，则会生成看门狗复位。

编程看门狗窗口时，早期的看门狗馈入也被视为看门狗事件。这可防止有故障的系统仍然馈入看门狗的情况。例如，应用程序代码可能在包含看门狗馈入的中断服务中阻塞。这样设置窗口可能会导致早期馈入，从而生成看门狗事件，允许系统恢复。

看门狗包括一个固定的（4 分频）预换算器和一个 24 位计数器，后者在有时钟控制时递减。计数器递减的最小起始值是 0xFF。设置低于 0xFF 的值将使 0xFF 被载入计数器。因此，看门狗的最小间隔为 $(T_{WDCLK} \times 256 \times 4)$，最大间隔为 $(T_{WDCLK} \times 2^{24} \times 4)$，两者都是 $(T_{WDCLK} \times 4)$ 的倍数。使用看门狗时应当采用下列方式：

- 在 TC 寄存器中设置看门狗定时器恒定的重载值。
- 在 MOD 寄存器中设置看门狗定时器操作模式。
- 如果需要窗口化操作，在 WINDOW 寄存器中设置看门狗窗口时间的值。
- 如果需要警告中断，则在 WARNINT 寄存器中设置看门狗警告中断的值。
- 将 0xAA 写入 FEED 寄存器，然后写入 0x55，以使能看门狗。
- 在看门狗计数器达到零之前必须重新馈入看门狗，以防发生看门狗事件。如果窗口值经过编程，则还必须在看门狗计数器超过该值后才馈入。

如果“看门狗定时器”配置为看门狗事件会导致复位且计数器会达到零，则 CPU 将被复位，如同外部复位一样，从向量表中载入堆栈指针和程序计数器。看门狗超时标志 (WDTOF) 经过检查后可确定看门狗是否导致了复位条件。 WDTOF 标志必须通过软件清零。

如果看门狗定时器配置为产生警告中断，则在计数器与寄存器定义的值匹配时将出现中断。

12.5.1 功能框图

看门狗的功能框图如下面的图 22 所示。功能框图中未显示同步逻辑 (PCLK - WDCLK)。

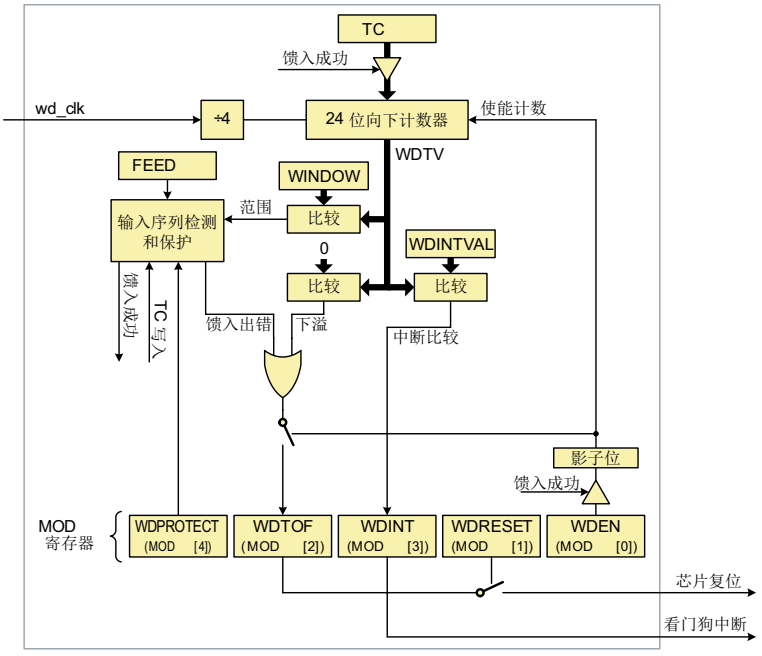


图 22. 窗口看门狗定时器功能框图

12.5.2 时钟和电源控制

看门狗定时器块使用两个时钟：PCLK 和 WDCLK。PCLK 供 APB 访问看门狗寄存器使用，由系统时钟生成（参见图 3）。 WDCLK 用于看门狗定时器计数，由看门狗振荡器生成。

两个时钟域之间同步逻辑的工作方式如下：当 MOD 和 TC 寄存器通过 APB 操作更新时，新的值将在 WDCLK 时钟域内逻辑的 3 个 WDCLK 周期内生效。

如果看门狗定时器根据 WDCLK 计数，同步逻辑将首先锁定 WDCLK 上计数器的值，然后使其与 PCLK 同步，这样 CPU 就能对 WDTV 寄存器进行读操作。

注：由于同步步骤，软件必须在馈入序列和 MOD 寄存器的 WDPROTECT 位使能时刻之间添加三个 WDCLK 时钟周期的延迟。延迟长度取决于所选的看门狗时钟 WDCLK。

12.5.3 使用 WWDT 锁定特性

WWDT 支持多种锁定特性，这些特性使能后可确保 WWDT 始终运行：

- 禁用 WWDT 时钟源
- 更改 WWDT 重载值

12.5.3.1 禁用 WWDT 时钟源

在睡眠模式、深度睡眠模式或掉电模式下，如果设置 WWDT MOD 寄存器中的位 5，则 WWDT 时钟源会被锁定，并且无法通过软件或硬件禁用。因此，用户在设置 MOD 寄存器中的位 5 **之前**，必须确保每种电源模式的看门狗振荡器均使能。

在深度掉电模式下，任何时钟锁定机制均无效，因为此时时钟不工作。但是， PMU 中的额外锁定位可设置，以防器件进入深度掉电模式 （参见图 43）。

12.5.3.2 更改 WWDT 重载值

如果设置WWDT MOD寄存器中的位4，则只能在计数器低于WDWARNINT和WDWINDOW值时才更改看门狗超时值 (TC)。

重载覆盖锁定机制只能通过任何类型的复位而禁用。

12.6 寄存器说明

看门狗定时器包含表 142 中所示的寄存器。

复位值仅反映已使用位中保存的数据，不包含保留位中的内容。

表 142. 寄存器概述：看门狗定时器 （基址 0x4000 4000）

名称	访问类型	地址偏移	说明	复位值	参考
MOD	R/W	0x000	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。	0	图 143
TC	R/W	0x004	看门狗定时器常数寄存器。该 24 位寄存器可确定超时值。	0xFF	图 145
FEED	WO	0x008	看门狗馈入序列寄存器。将 0xAA 写入该寄存器，然后写入0x55，可用WDTC中包含的值重新载入看门狗定时器。	不适用	图 146
TV	RO	0x00C	看门狗定时器值寄存器。该 24 位寄存器可读取看门狗定时器的当前值。	0xFF	图 147
-	-	0x010	保留	-	-
WARNINT	R/W	0x014	看门狗警告中断比较值。	0	图 148
WINDOW	R/W	0x018	看门狗窗口比较值。	0xFF FFFF	图 149

12.6.1 看门狗模式寄存器

WDMOD 寄存器控制看门狗的操作。需要注意的是，看门狗馈入必须在 WDMOD 寄存器的任何改变生效之前执行。

表 143. 看门狗模式寄存器 （MOD， 0x4000 4000）位说明

位	符号	值	说明	复位值
0	WDEN		看门狗使能位。一旦将 1 写入该位，便无法用 0 重新写入。一旦该位设为 1，则在发生看门狗馈入后，看门狗定时器会开始运行。	0
		0	看门狗定时器停止。	
		1	看门狗定时器运行。	
1	WDRESET		看门狗复位使能位。一旦将 1 写入该位，便无法用 0 重新写入。	0
		0	看门狗超时不会引起芯片复位。	
		1	看门狗超时会引起芯片复位。	
2	WDTOF		看门狗超时标志。在看门狗定时器因馈入错误或与 WDPROTECT 关联的事件而超时时设置。通过软件清零。如果 WDRESET=1，会引起芯片复位。	0（仅在外部复位之后）
3	WDINT		警报中断标志。在定时器达到 WDWARNINT 寄存器中的值时设置。通过软件清零。	0

表 143. 看门狗模式寄存器 （MOD， 0x4000 4000）位说明

位	符号	值	说明	复位值
4	WDPROTECT		看门狗更新模式。该位可通过软件进行一次性设置，并且只能通过复位清零。	0
		0	看门狗超时值 (TC) 可随时改变。	
		1	看门狗超时值 (TC) 仅在计数器低于 WDWARNINT 和 WDWINDOW 的值时才能改变。	
5	LOCK		该位为 1 时可防止禁用或关断看门狗振荡器。该位可通过软件进行一次性设置，并且只能通过复位清零。	0
31:6	-		保留，用户软件不应将 1 写入保留位。未定义从保留位读取的值。	不适用

WDEN、 **WDPROTECT** 或 **WDRESET** 位经过设置后，不能用软件清零。两个标志都须利用外部重复位或看门狗定时器复位来清零。

WDTOF 在看门狗超时、发生馈入错误，或当 PROTECT =1 且试图对 TC 寄存器进行写操作时，该看门狗超时标志会被设置。该标志通过将 0 写入该位的软件清零。

WDINT 当看门狗计数器达到 WARNINT 规定的值时，该看门狗中断标志会被设置。该标志在发生复位时被清零，并通过将 0 写入该位的软件清零。

在除深度掉电模式外的所有电源模式中，看门狗运行并具有操作时钟源的任何时刻都可发生看门狗复位或中断。看门狗振荡器可配置为在睡眠模式、深度睡眠模式和掉电模式下保持运行。

如果在睡眠模式、深度睡眠模式或掉电模式下发生看门狗中断，并且 WWDT 中断在 NVIC 中使能，则器件会唤醒。需要注意的是，在深度睡眠模式和掉电模式下，除了 NVIC 之外，还必须在 STARTERP1 寄存器中使能 WWDT 中断。

参见下列寄存器：

表 34 “ 启动逻辑 1 中断唤醒使能寄存器 （STARTERP1，地址 0x4004 8214）位说明 ”

表 144. 看门狗工作模式选择

WDEN	WDRESET	工作模式
0	X（0 或 1）	看门狗不运行时的调试 / 操作。
1	0	看门狗中断模式：将生成看门狗警报中断，但不会生成看门狗复位。 选择该模式后，在看门狗计数器达到 WDWARNINT 规定的值时将设置 WDINT 标志，并将生成看门狗中断请求。
1	1	看门狗复位模式：看门狗中断和看门狗复位都使能。 选择该模式后，在看门狗计数器达到 WDWARNINT 规定的值时将设置 WDINT 标志，并将生成看门狗中断请求。看门狗计数器达到零时将复位微控制器。达到 WDWINDOW 的值之前的看门狗馈入也将导致看门狗复位。

12.6.2 看门狗定时器常量寄存器

该 TC 寄存器可确定超时值。每次发生馈入序列时，TC 中的数值就会被载入看门狗定时器。TC 复位至 0x00 00FF。写入低于 0xFF 的值将导致 0x00 00FF 被载入 TC。因此，最小超时间隔为 $T_{WDCLK} \times 256 \times 4$。

如果 WDMOD 中的 WDPROTECT 位设置为 1，则在看门狗计数器低于 WDWARNINT 和 WDWINDOW 的值前试图改变 TC 的值，将导致看门狗复位并设置 WDTOF 标志。

表 145. 看门狗定时器常量寄存器 （TC， 0x4000 4004）位说明

位	符号	说明	复位值
23:0	COUNT	看门狗超时值。	0x00 00FF
31:24	-	保留，用户软件不应将 1 写入保留位。未定义从保留位读取的值。	不适用

12.6.3 看门狗馈入寄存器

将 0xAA 和 0x55 依次写入该寄存器将使 WDTC 的值重新载入看门狗定时器。如果看门狗已通过 WDMOD 寄存器使能，则该操作也会启动看门狗。设置 WDMOD 寄存器中的 WDEN 位不足以使能看门狗。设置 WDEN 后必须完成有效的馈入序列，这样看门狗才能生成复位。到那时，看门狗将忽略馈入错误。

将 0xAA 写入 WDFEED 后，必须紧接着写入 0x55，否则如果看门狗使能，访问任何看门狗寄存器都将会立即产生复位 / 中断，并设置 WDTOF 标志。在馈入序列期间，复位将在错误访问看门狗寄存器后的第二个 PCLK 期间生成。

如果某种应用的中断可能导致处理器控制在馈入时脱离当前任务而重新安排，则最佳实践是在馈入序列周围禁用中断，然后在控制返回执行被中断的任务之前导致一些其他的访问操作。

表 146. 看门狗馈入寄存器 （FEED， 0x4000 4008）位说明

位	符号	说明	复位值
7:0	FEED	馈入值应当是 0xAA，后跟 0x55。	不适用
31:8	-	保留，用户软件不应将 1 写入保留位。未定义从保留位读取的值。	不适用

12.6.4 看门狗定时器值寄存器

WDTV 寄存器用于读取看门狗定时器计数器的当前值。

读取 24 位计数器的值时，锁定和同步过程需要 6 个 WDCLK 周期和 6 个 PCLK 周期，因此， WDTV 的值比 CPU 正在读取的定时器的实际值要 “ 旧 ”。

表 147. 看门狗定时器值寄存器 （TV， 0x4000 400C）位说明

位	符号	说明	复位值
23:0	COUNT	计数器定时器值。	0x00 00FF
31:24	-	保留，用户软件不应将 1 写入保留位。未定义从保留位读取的值。	不适用

12.6.5 看门狗定时器警告中断寄存器

WDWARNINT 寄存器可确定将生成看门狗中断的看门狗定时器计数器值。当看门狗定时器计数器与 WARNINT 定义的值匹配时，在后续 WDCLK 之后会生成中断。

如果计数器后 10 位与 WARNINT 的 10 位值相同，则会发生看门狗计数器与 WARNINT 的匹配，计数器前面其余位的值因而将全都是 0。这使得看门狗事件之前发生中断的最大时间是 1,023 个看门狗定时器计数（4,096 个看门狗时钟）。如果 WARNINT 为 0，则中断将与看门狗事件同时发生。

表 148. 看门狗定时器警告中断寄存器 （WARNINT， 0x4000 4014）位说明

位	符号	说明	复位值
9:0	WARNINT	看门狗警告中断比较值。	0
31:10	-	保留，用户软件不应将 1 写入保留位。未定义从保留位读取的值。	不适用

12.6.6 看门狗定时器窗口寄存器

WINDOW寄存器可确定在执行看门狗馈入时允许的最大WDTV值。如果馈入序列在WDTV大于 WINDOW 中的值时发生，则会发生看门狗事件。

WINDOW 会复位到可能的最大 WDTV 值，这样窗口操作就会无效。

表 149. 看门狗定时器窗口寄存器 （WINDOW， 0x4000 4018）位说明

位	符号	说明	复位值
23:0	WINDOW	看门狗窗口值。	0xFF FFFF
31:24	-	保留，用户软件不应将 1 写入保留位。未定义从保留位读取的值。	不适用

12.7 功能说明

下图说明了看门狗定时器工作的几个方面。

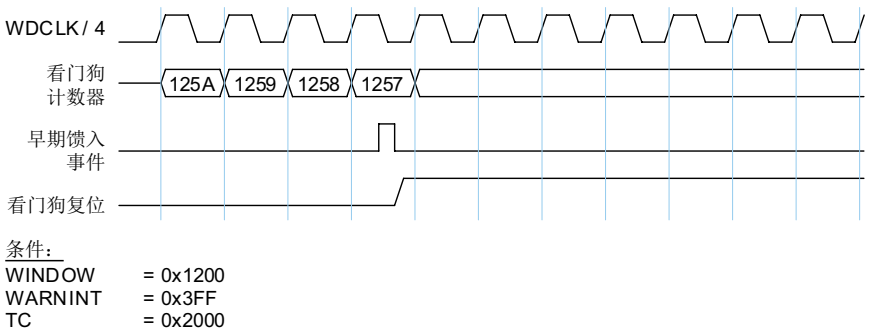


图 23. 早期看门狗馈入，窗口化模式使能

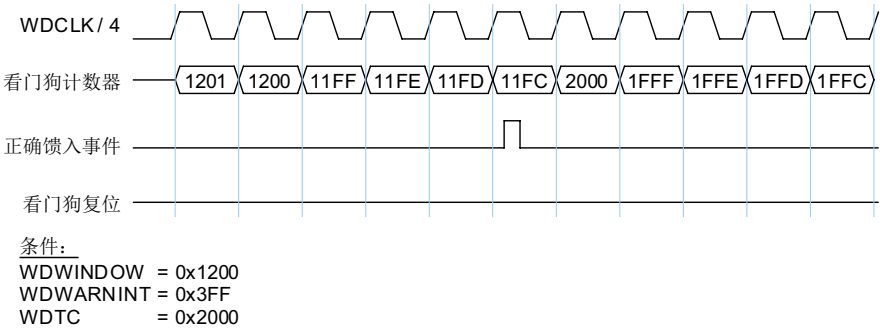


图 24. 正确看门狗馈入，窗口化模式使能

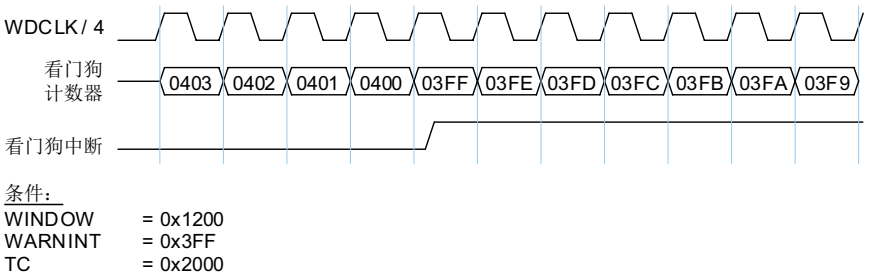


图 25. 看门狗警告中断

13.1 本章导读

所有 LPC800 器件都提供自唤醒定时器。

13.2 特性

- 32 位可加载减法计数器。加载计数值时计数器会自动启动。超时事件会生成一个中断 / 唤醒请求。
- WKT 位于一个独立且始终上电的电源域中。
- WKT 支持两个时钟源。一个时钟源来源于始终上电的电源域。
- WKT 可用于将器件从任何低功耗模式（包括深度掉电模式）中唤醒，也可用于通用定时。

13.3 基本配置

- 在 SYSAHBCLKCTRL 中，设置位 9（表 18）以使能寄存器接口时钟。
- 使用 PRESETCTRL 寄存器（表 7）清零 WKT 复位。
- WKT 中断连接至 NVIC 的 15 号中断。
- 在 PMU（表 46）中使能低功耗振荡器。
- 在 PDRUNCFG 寄存器（表 37）中使能 IRC 和 IRC 输出。
- 参见章节 5.7.1 以使能各种掉电模式。

13.4 引脚说明

WKT 无可配置引脚。

13.5 简介

自唤醒定时器是一个 32 位可加载减法计数器。将任何非零值写入该定时器可使能该计数器并启动递减序列。如果计数器用作唤醒定时器，则该次写操作可在进入低功耗模式前的那一刻发生。

载入起始计数值时，自唤醒定时器会自动开启，从预载入的值开始向下计数至零，产生中断和 / 或唤醒请求，然后关断，直到后续软件写操作将其重新启动。

13.5.1 WKT 时钟源

自唤醒定时器可根据两个替代时钟源进行计时：

- 来自 IRC 振荡器的 750 kHz 时钟。这是默认时钟。
- 使用带专用片内振荡器的 10 kHz 低功耗时钟作为时钟源。

来源于 IRC 的时钟比替代的低功耗时钟要精确的多。然而， IRC 在大多数低功耗模式下不可用。定时器用于从 IRC 被禁用的某个电源模式唤醒时，不得选择该时钟。

替代时钟源是一个（标称值为）10 kHz 的低功耗时钟，来源于某个专用振荡器。该振荡器位于始终上电的电压域中，因此可将其设置为在深度掉电模式下继续工作，该模式下电源不对器件的其余部分供电。 IRC 时钟被关断时，该时钟在其他低功耗模式下仍然可用。

低功耗振荡器并不精确（随制造工艺和温度而变化的范围为 +/- 40 %）。由于进入低功耗模式后，芯片温度降低，因此计数期间的频率可能还是会发生漂移。

13.6 寄存器说明

表 150. 寄存器概述：WKT （基址 0x4000 8000）

名称	访问类型	地址偏移	说明	复位值	参考
CTRL	R/W	0x0	自唤醒定时器控制寄存器。	0	表 151
COUNT	R/W	0xC	计数器寄存器。		

13.6.1 控制寄存器

必须在 NVIC 中使能 WKT 中断，以便通过自唤醒定时器唤醒器件。

表 151. 控制寄存器 （CTRL，地址 0x4000 8000）位说明

位	符号	值	说明	复位值
0	CLKSEL		选择自唤醒定时器时钟源。	0
		0	分频 IRC 时钟。该时钟的工作频率为 750 kHz，能够以 1.33 µs 的增量提供最长约为 95 分钟的超时周期。 **注：**深度睡眠模式、掉电模式、深度掉电模式下该时钟不可用。如需使用定时器唤醒这些模式，则不应选择该选项。	
		1	低功耗时钟。它是（标称值为） 10 kHz 的时钟，能够以 100 µs 的增量提供最长约为 119 小时的超时周期。随着温度和工艺的变化，该时钟的精度限制在 +/- 45 % 范围内。 **注：**该时钟在所有电源模式下都可用。使用前，必须先使能低功耗振荡器。如有必要，还必须将振荡器设置为在深度掉电模式下保持工作状态。	
1	ALARMFLAG		唤醒或警报定时器标志。	-
		0	未超时。自唤醒定时器未超时。写 0 无效。	
		1	超时。自唤醒定时器超时。若时钟源采用低功耗振荡器，则该标志产生中断请求，可将器件从任意低功耗模式中唤醒，包括深度掉电模式。写入 1 可清零该状态位。	
2	CLEARCTR		清零自唤醒定时器。	0
		0	无效。对该位进行读操作将始终返回 0。	
		1	清零计数器。计数在载入新的计数值之前不会停止。	
31:3	-		保留。	-

13.6.2 计数寄存器

计数过程中，不要对该寄存器进行写操作。

注：通常不建议读取定时器状态。如果读操作碰巧发生在自唤醒定时器的时钟沿上，则读取过程中将没有任何机制可确保该寄存器中的某些位不发生变化。如果一定要读取该数值，则建议连续读取两次。

表 152. 计数器寄存器 （COUNT，地址 0x4000 800C）位说明

位	符号	说明	复位值
31:0	VALUE	对该寄存器进行写操作可将起始计数值预载入定时器，并开始向下计数序列。 读操作可反映定时器的当前值。	-

14.1 本章导读

所有 LPC800 器件都提供 SysTick 定时器。

14.2 特性

- 简单的 24 位定时器。
- 使用专用的异常向量。
- 由系统时钟或系统时钟 /2 从内部计时。

14.3 基本配置

配置系统节拍定时器使用的是下列寄存器：

1. 系统节拍定时器通过 SysTick 控制寄存器（表 154）使能。系统节拍定时器时钟固定为系统时钟频率的二分之一。
2. 在 SYST_CSR 寄存器 （表 154）中使能 SysTick 的时钟源。

14.4 引脚说明

SysTick 没有可配置引脚。

14.5 简介

SysTick 定时器的功能框图如下面的图 26 所示。

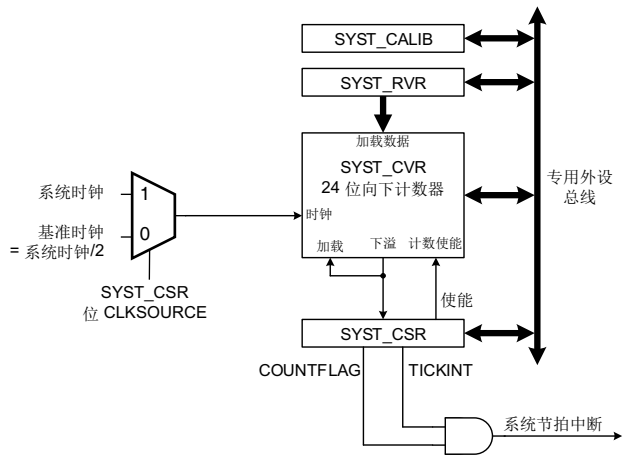


图 26. 系统节拍定时器功能框图

SysTick 定时器是 Cortex-M0+ 的主要组成部分。 SysTick 定时器为操作系统或其它系统管理软件提供固定 10 毫秒的中断。

由于 SysTick 定时器是 Cortex-M0+ 的一部分，它为基于 Cortex-M0 的器件提供一个标准定时器，有助于软件的移植。 SysTick 定时器可用于：

- 可编程设置频率的 RTOS 节拍定时器 （例如 100 Hz），调用一个 SysTick 例程。
- 使用内核时钟的高速报警定时器。
- 简单的计数器。软件可使用它测量完成任务所需时间和已使用时间。
- 基于丢失 / 命中期限控制的内部时钟源。控制和状态寄存器中的 COUNTFLAG 位字段可用于决定一个动作是否在设定的期限内完成，作为动态时钟管理控制环的一部分。

有关详情，请参见参考 3。

14.6 寄存器说明

Systick 定时器寄存器位于 ARM Cortex-M0+ 专用外设总线上 （参见图 2），是 ARM Cortex-M0+ 内核外设的一部分。有关详情，请参见参考 3。

表 153. 寄存器概述：SysTick 定时器 （基址 0xE000 E000）

名称	访问类型	地址偏移	说明	复位值 [1]
SYST_CSR	R/W	0x010	系统定时器控制和状态寄存器	0x000 0000
SYST_RVR	R/W	0x014	系统定时器重载值寄存器	0
SYST_CVR	R/W	0x018	系统定时器当前值寄存器	0
SYST_CALIB	R/W	0x01C	系统定时器校准值寄存器	0x4

[1] 复位值仅反映已使用位中保存的数据，不包括保留位的内容。

14.6.1 系统定时器控制和状态寄存器

SYST_CSR 寄存器包含 SysTick 定时器的控制信息，并提供状态标志。该寄存器是 ARM Cortex-M0+ 内核系统定时器寄存器模块的一部分。有关该寄存器的位说明，请参见参考 3。

该寄存器可确定系统节拍定时器的时钟源。

表 154. SysTick 定时器控制和状态寄存器 (SYST_CSR - 0xE000 E010) 位说明

位	符号	说明	复位值
0	使能	系统节拍计数器使能。为 1 时，计数器使能。为 0 时，计数器禁用。	0
1	TICKINT	系统节拍中断使能。为 1 时，系统节拍中断使能。为 0 时，系统节拍中断禁用。使能时，在系统节拍计数器倒计数到 0 时生成中断。	0
2	CLKSOURCE	系统节拍时钟源选择。为 1 时，选择系统时钟 (CPU)。为 0 时，选择系统时钟 /2 作为参考时钟。	0
15:3	-	保留，用户软件不应将 1 写入保留位。未定义从保留位读取的值。	不适用
16	COUNTFLAG	如果自上一次对该寄存器进行读操作以来 SysTick 定时器数到 0，则返回 1。	0
31:17	-	保留，用户软件不应将 1 写入保留位。未定义从保留位读取的值。	不适用

14.6.2 系统定时器重载值寄存器

SYST_RVR 寄存器被设置为 SysTick 定时器倒数到 0 时载入的值。在定时器进行初始化时，该寄存器由软件载入。如果 CPU 运行频率适合用 SYST_CALIB 值，则可对 SYST_CALIB 寄存器进行读操作，并将其用作 SYST_RVR 寄存器的值。

表 155. 系统定时器重载值寄存器 (SYST_RVR - 0xE000 E014) 位说明

位	符号	说明	复位值
23:0	RELOAD	该值在系统节拍计数器倒数到 0 时载入该计数器。	0
31:24	-	保留，用户软件不应将 1 写入保留位。未定义从保留位读取的值。	不适用

14.6.3 系统定时器当前值寄存器

当软件对 SYST_CVR 寄存器进行读操作时，该寄存器将从系统节拍计数器返回当前计数。

表 156. 系统定时器当前值寄存器 (SYST_CVR - 0xE000 E018) 位说明

位	符号	说明	复位值
23:0	CURRENT	对该寄存器会进行读操作会返回系统节拍计数器的当前值。写任意值都可将系统节拍计数器和 STCTRL 中的 COUNTFLAG 位清零。	0
31:24	-	保留，用户软件不应将 1 写入保留位。未定义从保留位读取的值。	不适用

14.6.4 系统定时器校准值寄存器 (SYST_CALIB - 0xE000 E01C)

SYST_CALIB 寄存器的值由系统配置模块 SYSCON 中 SYSTCKCAL 寄存器的值驱动（参见表 29）。

表 157. 系统定时器校准值寄存器 (SYST_CALIB - 0xE000 E01C) 位说明

位	符号	值	说明	复位值
23:0	TENMS		参见参考 3。	0x4
29:24	-		保留，用户软件不应将 1 写入保留位。未定义从保留位读取的值。	不适用
30	SKEW		参见参考 3。	0
31	NOREF		参见参考 3。	0

14.7 功能说明

SysTick 定时器是一个 24 位定时器，可倒计数到 0，还可生成中断。SysTick 定时器的作用就是为每次中断之间提供一个 10 毫秒的固定时间间隔。SysTick 定时器的时钟信号可由 CPU 时钟（系统时钟，参见图 3）提供，也可由参考时钟（固定为 CPU 时钟频率的二分之一）提供。要在特定的间隔上生成循环中断，就必须使用所需间隔的正确值初始化 SYST_RVR 寄存器。默认值保存在 SYST_CALIB 寄存器中，可通过软件进行修改。如果将 CPU 时钟设置为 <tbd>MHz，则默认值为 10 毫秒中断速率。

14.7.1 定时器计算示例

使用系统节拍定时器需遵循以下规则：

1. 用重载值 RELOAD 对 SYST_RVR 寄存器进行编程，以获得所需的时间间隔。
2. 对 SYST_CVR 寄存器进行写操作将其清零。这样能确保定时器使能后可以从 SYST_RVR 值开始计数，而不是从任意值开始计数。

3. 用值 0x7 对 SYST_SCR 寄存器进行编程，即可使能 SysTick 定时器和 SysTick 定时器中断。

以下示例说明，如果将系统时钟设置为 20 MHz，选择 SysTick 定时器重载值可以获得 10 ms 的时间间隔。

示例 （系统时钟 = 20 MHz）。

系统节拍时钟 = 系统时钟 = 20 MHz。SYST_CSR 寄存器中的 CLKSOURCE 位设为 1（系统时钟）。

RELOAD =（系统节拍时钟频率 ×10 ms）-1 = (20 MHz × 10 ms) -1 = 200 000 -1 = 199999 = 0x00030D3F。

15.1 本章导读

所有器件都提供USART0和USART1。仅器件LPC812M101FDH16和LPC812M101FDH20提供 USART2。

有关 USART 外设和软件接口的说明，请阅读本章。

LPC800 同样提供可配置并操作 USART 的基于 ROM 的片内 USART API。参见图 279。

15.2 特性

- 7 个、 8 个或 9 个数据位， 1 个或 2 个停止位
- 支持主机或从机运行的同步模式。包括数据相位选择和连续时钟选项。
- 支持软件地址比较的多处理器 / 多点 （9 位）模式。（可进行软件地址检测和收发器方向控制的 RS-485。）
- 奇偶生成及校验：奇校验、偶校验或无校验。
- 一个发送数据缓冲器和一个接收数据缓冲器。
- 用于自动流控制的 RTS/CTS 硬件信号。通过增量 CTS 检测、发送禁用控制以及作为 RTS 输出的任意 GPIO 可进行软件流控制。
- 接收的数据和状态可以选择性地从单个寄存器中读取
- 中止产生及检测。
- 接收数据是3个样本“投票”中的2个。当有1个样本与其它不同时，便会设置状态标志。
- 内置波特率发生器。
- 所有 USART 共用同一个小数速率分频器。
- 可中断的状态有：接收器就绪、发送器就绪、接收器挂起、接收器断点检测变化、帧错误、奇偶校验错误、溢出、欠载、增量 CTS 检测和接收器样本噪声检测。
- 用于数据测试和流控制的环回模式。

15.3 基本配置

注：片内 USART API 提供可配置并使用 USART 的软件例程。参见图 279。

配置 USART0/1/2 用于接收和发送数据：

- 在 SYSAHBCLKCTRL 寄存器中，设置位 14 至 16 （表 18）以使能寄存器接口时钟。
- 使用 PRESETCTRL 寄存器 （表 7）清零 USART0/1/2 外设复位。
- 在 NVIC 的 3 号至 5 号插槽中使能或禁用 USART0/1/2 中断。
- 通过开关矩阵配置 USART0/1/2 引脚功能。参见章节 15.4。
- 配置 USART 时钟和波特率。参见章节 15.3.1。

配置 USART0/1/2，将器件从低功耗模式唤醒：

- 配置 USART，使其可在同步从机模式下接收和发送数据。参见章节 15.3.2。

15.3.1 配置 USART 时钟和波特率

所有三个 USART 都使用同一个外设时钟 (U_PCLK)，并且若有必要，还可使用同一个小数波特率发生器。外设时钟和小数分频器可在 SYSCON 模块中根据下列步骤设置，用于计算波特率 （参见图 27）：

1. 通过将一个大于零的UARTCLKDIV值写入USART外设时钟分频器寄存器可配置UART时钟。这是分频后的主时钟，可用于所有 USART。

   章节 4.6.14 “USART 时钟分频器寄存器 ”

2. 要使用某个小数值以获得特定的波特率，可对小数分频器进行编程。小数分频器值是 MULT/DIV 的小数部分。 MULT 值在 UARTFRGMULT 寄存器中设定，且 DIV 值在 SYSCON 模块的 UARTFRGDIV 寄存器中以 256 固定值设定。

   U_PCLK = UARTCLKDIV/(1+(MULT/DIV))

   下列规则适用于 MULT 和 DIV：

   – 用数值 0xFF 进行 UARTFRGDIV 寄存器的编程，始终将 DIV 设置为 256。

   – 在 UARTFRGMULT 寄存器中设定 0 ~ 255 之间的任意数值。

   章节 4.6.19 “USART 小数生成器乘法器值寄存器 ”

   章节 4.6.18 “USART 小数生成分频器值寄存器 ”

3. 在异步模式中：在 USARTn BRG 寄存器中配置波特率分频器 BRGVAL。波特率分频器用系数 16 进行通用 USART 外设时钟的分频，然后乘以波特率值，以便使
   波特率 = U_PCLK/16 x BRGVAL。

   章节 15.6.9 “USART 波特率发生器寄存器 ”

4. 在同步模式中：串行时钟为 Un_SCLK = U_PCLK/BRGVAL。

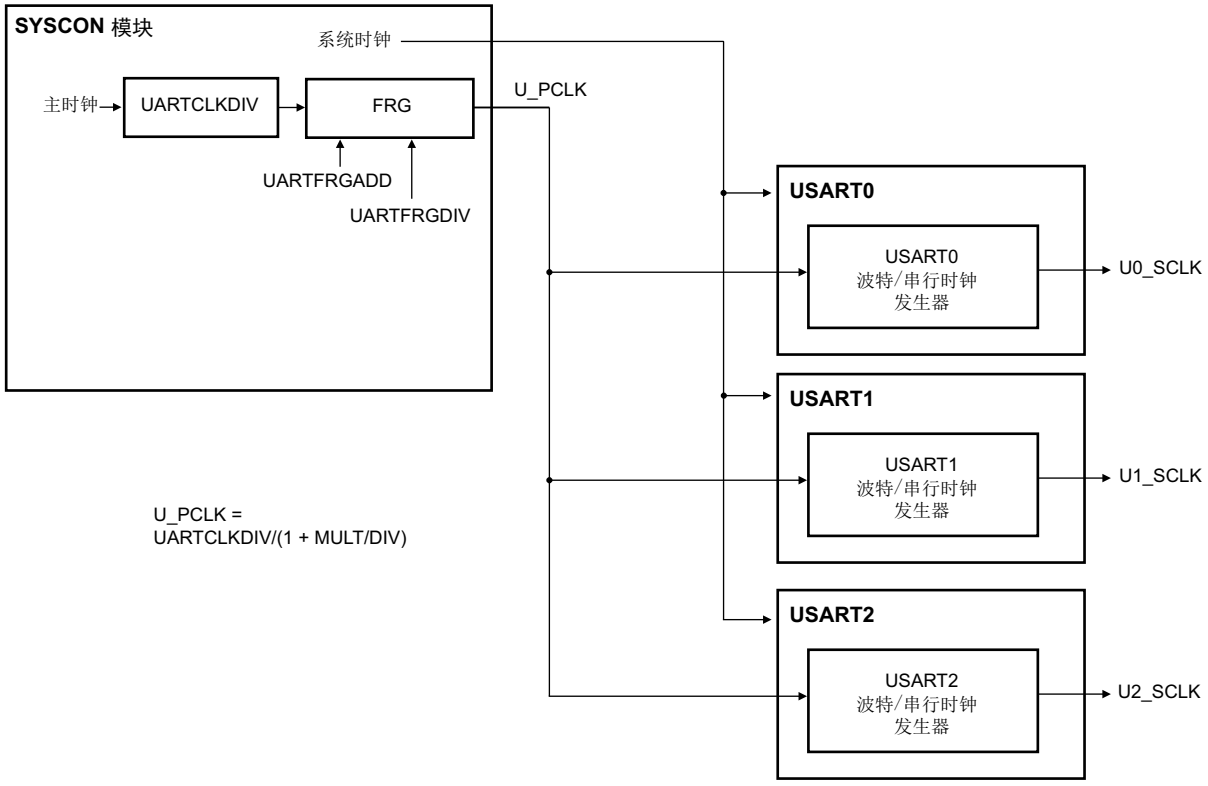


图 27. USART 计时

有关时钟配置的详细信息，请参见：

章节 15.7.1 “ 时钟和波特率 ”

15.3.2 配置 USART 以实现唤醒

USART 可通过任意使能的 USART 中断，将系统从异步或同步模式的睡眠模式中唤醒。

若 USART 配置为同步从机模式，则接收信号时 USART 模块可产生一个中断；甚至在深度睡眠模式或掉电模式下亦如此 （此时 ARM Cortex-M0+ 内核不提供 USART 模块时钟）。

在深度睡眠或掉电模式下，只要 USART 接收到来自主机的时钟信号， RXDATA 寄存器便可接收多达一个字节的数据。任何作为数据接收过程部分的中断均可唤醒器件。

15.3.2.1 从睡眠模式唤醒

- 在异步模式或同步模式下配置 USART。参见表 160。
- 在 NVIC 中使能 USART 中断。
- 任意 USART 中断均可将器件从睡眠模式中唤醒。在 INTENSET 寄存器 （表 163）中使能 USART 中断。

15.3.2.2 从深度睡眠模式或掉电模式唤醒

- 在同步从机模式下配置 USART。参见表 160。必须将 SCLK 功能分配给某个引脚，并将该引脚与主机相连。

- 在 STARTERP1 寄存器中使能 USART 中断。参见表 34 “ 启动逻辑 1 中断唤醒使能寄存器 （STARTERP1，地址 0x4004 8214）位说明 ”。
- 在 NVIC 中使能 USART 中断。
- 在 PDAWAKE 寄存器中，配置所有器件唤醒时需要运行的外设。
- 引发中断的任意事件都可使 USART 将器件从深度睡眠或掉电模式中唤醒，同样可在 INTENSET 寄存器中使能。典型的唤醒事件有：
  - 收到启动位数据。
  - RXDATA 缓冲区收到一个字节。
  - 数据在 TXDATA 缓冲区中准备发送就绪，并收到来自主机的串行时钟。
  - 若连接 CTS 功能，则 CTS 引脚状态会发生变化。 <tbd>
  - **注：**在 INTENSET 寄存器 （表 163）中使能或禁用中断，可在 USART 接收 / 发送协议中定制唤醒发生的时间。

15.4 引脚说明

USART 接收、发送和控制信号都是可转移功能，并且通过开关矩阵分配给引脚。

有关如何将 USART 功能分配给 LPC800 封装上的引脚，请参见章节 9.3.1 “ 将内部信号与封装引脚相连 ”。

表 158. USART 引脚说明

功能	方向	引脚	说明	SWM 寄存器	参考
U0_TXD	O	任意	USART0 的发送器输出。串行发送数据。	PINASSIGN0	表 97
U0_RXD	I	任意	USART0 的接收器输入。串行接收数据。	PINASSIGN0	表 97
$\overline{\text{U0\_RTS}}$	O	任意	USART0 的请求发送输出。低电平有效信号表示 USART0 接收数据准备就绪。通过使用硬件流控制，该信号可支持处理器间通信。当 USART RTS 信号配置为出现在器件引脚上时，该特性有效。	PINASSIGN0	表 97
$\overline{\text{U0\_CTS}}$	I	任意	USART0 的清零发送输入。低电平有效信号表示正在与 USART 通信的外部器件已做好接收数据的准备。该特性通过 CFG 寄存器中的 CTSEn 位使能且配置为出现在器件引脚上时有效。如果通过外部器件解除置位 （高电平），则 USART 将完成任何正在处理中的字符传输，然后停止，直到 CTS 被再次置位 （低电平）。	PINASSIGN0	表 97
U0_SCLK	I/O	任意	同步模式下 USART0 的串行时钟输入 / 输出。同步模式下的时钟输入或输出。	PINASSIGN1	表 98
U1_TXD	O	任意	USART1 的发送器输出。串行发送数据。	PINASSIGN1	表 98
U1_RXD	I	任意	USART1 的接收器输入。	PINASSIGN1	表 98
$\overline{\text{U1\_RTS}}$	O	任意	USART1 的请求发送输出。	PINASSIGN1	表 98
$\overline{\text{U1\_CTS}}$	I	任意	USART1 的清零发送输入。	PINASSIGN2	表 99
U1_SCLK	I/O	任意	同步模式下 USART1 的串行时钟输入 / 输出。	PINASSIGN2	表 99
U2_TXD	O	任意	USART2 的发送器输出。串行发送数据。	PINASSIGN2	表 99
U2_RXD	I	任意	USART2 的接收器输入。	PINASSIGN2	表 99
$\overline{\text{U2\_RTS}}$	O	任意	USART2 的请求发送输出。	PINASSIGN3	表 100
$\overline{\text{U2\_CTS}}$	I	任意	USART2 的清零发送输入。	PINASSIGN3	表 100
U2_SCLK	I/O	任意	同步模式下 USART2 的串行时钟输入 / 输出。	PINASSIGN3	表 100

15.5 简介

USART 接收器模块监控串行输入线路 Un_RXD，以便输入有效。接收器移位寄存器收到字符后即进行汇编，随后被发送到接收器缓冲区寄存器，等待 CPU 的访问。

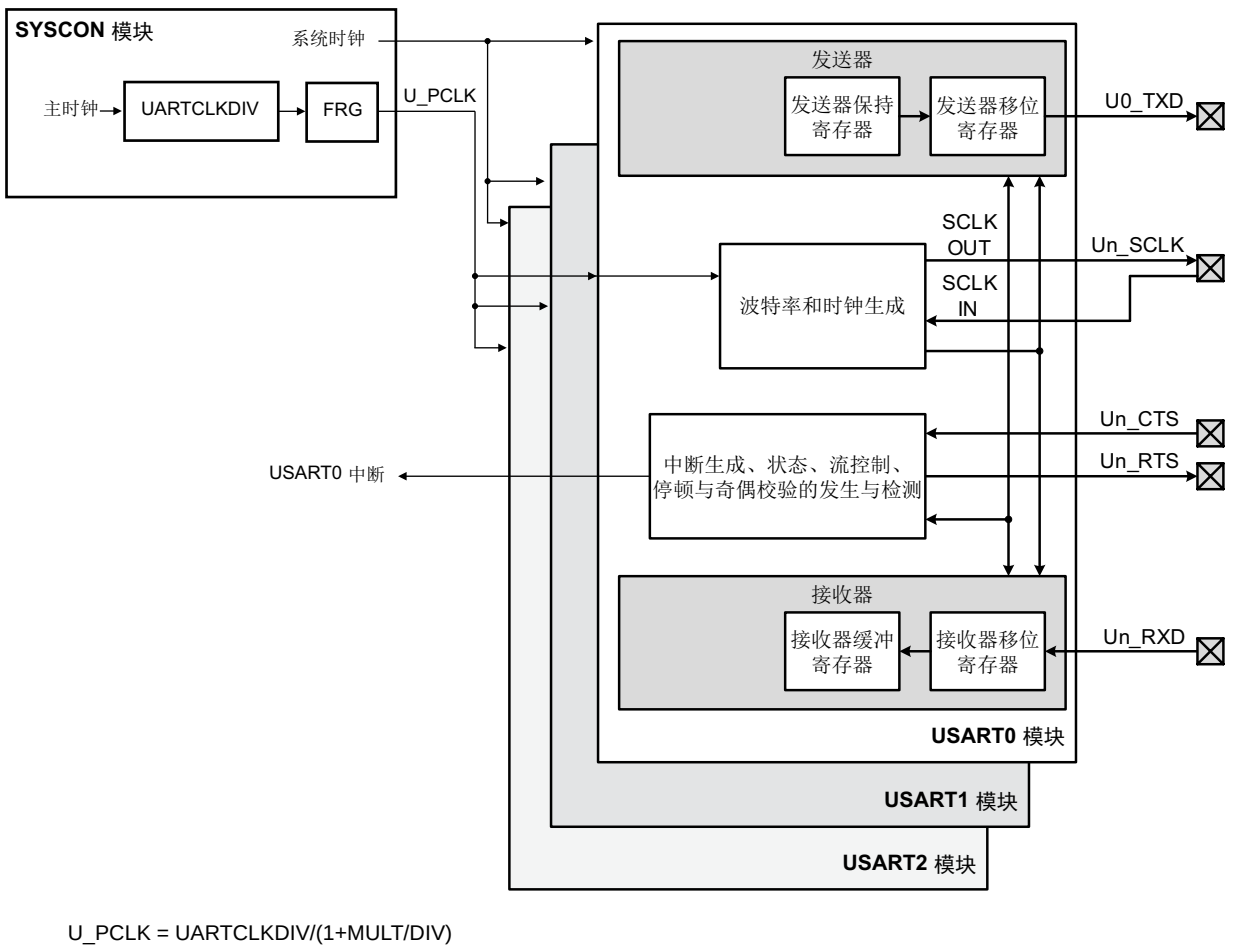
USART 发送器模块接收 CPU 写入的数据，并将数据缓存在发送保持寄存器中。一旦发送器可用，则发送移位寄存器会取出数据，对其进行格式化，然后进行串行处理，并通过 Un_TXD 串行输出。

标准异步工作模式下，波特率发生器模块对输入的时钟进行分频，产生 16 倍波特率时钟。BRG 时钟输入源是共用小数速率发生器，根据 USART 外设时钟 (U_PCLK) 运行。

同步从机模式下，直接使用串行时钟发送和接收数据。同步主机模式下，使用未分频的波特率时钟发送和接收数据。

状态寄存器保存并提供发送器和接收器的状态信息。许多能产生中断的状态标志可通过软件选择。

注：小数值和 USART 外设时钟可在所有 USART 之间共用。



U_PCLK = UARTCLKDIV/(1+MULT/DIV)

图 28. USART 功能框图

15.6 寄存器说明

复位值仅反映已使用位中保存的数据，不包含保留位中的内容。

表 159. 寄存器概述：USART （基址 0x4006 4000 (USART0)， 0x4006 8000 (USART1)， 0x4006 C000 (USART2)）

名称	访问类型	偏移	说明	复位值	参考
CFG	R/W	0x000	USART 配置寄存器。基本 USART 配置设置，通常在工作中不发生改变。	0	表 160
CTRL	R/W	0x004	USART 控制寄存器。 USART 控制设置，较容易在工作中发生改变。	0	表 161
STAT	R/W	0x008	USART 状态寄存器。完整状态值可从此处读取。写入 1 可清零寄存器中的某些位。某些位可通过写入 1 而清零。	0x000E	表 162
INTENSET	R/W	0x00C	中断使能读取和设置寄存器。包含所有潜在 USART 中断的各个中断使能位。完整数值可从该寄存器中读取。将 1 写入任意已实施的位都将设置该位。	0	表 163
INTENCLR	W	0x010	中断使能清零寄存器。可清零 INTENSET 寄存器中的所有组合位。将 1 写入任意已实施的位可清零对应位。	-	表 164
RXDATA	R	0x014	接收器数据寄存器。包含接收到的最后一个字符。	-	表 165
RXDATASTAT	R	0x018	含状态信息的接收器数据寄存器。将接收到的最后一个字符与当前 USART 接收状态相结合。允许软件将输入数据和状态信息一同恢复。	-	表 166
TXDATA	R/W	0x01C	发送数据寄存器。待发送数据在此写入。	0	表 167
BRG	R/W	0x020	波特率发生器寄存器。 16 位整数波特率除数值。	0	表 168
INTSTAT	R	0x024	中断状态寄存器。反映当前使能的中断。	0x0005	表 169

15.6.1 USART 配置寄存器

CFG 寄存器包含 USART 相关的通信和模式设置，通常这些设置在应用中仅配置一次。

注：软件如需更改配置值，则应当按照下列顺序：1) 确保 USART 当前不发送或接收数据。2) 通过将 0 写入使能位禁用 USART （可将 0 写入整个寄存器）。 3) 写入新配置值，将 ENABLE 位设置为 1。

表 160. USART 配置寄存器 （CFG，地址 0x4006 4000 (USART0)， 0x4006 8000 (USART1)，0x4006 C000 (USART2)）位说明

位	符号	值	说明	复位值
0	使能		USART 使能。	0
		0	禁用。 USART 禁用，内部状态机和计数器复位。使能位为 0 时，所有 USART 中断禁用。当再次设置使能位时， CFG 和其他大部分控制位都保持不变。例如，由于发送器已被复位，USART 一旦被重新使能，会立即产生 TxRdy 中断 （如果在 INTENSET 寄存器中已使能)，因而可投入使用。	
		1	使能。 USART 使能，可进行工作。	
1	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
3:2	DATALEN		选择 USART 的数据大小。	00
		0x0	7 位数据长度。	
		0x1	8 位数据长度。	
		0x2	9 位数据长度。第 9 位通常用于多点模式的寻址。参见 CTRL 寄存器中的 ADDRDET 位。	
		0x3	保留。	
5:4	PARITYSEL		选择 USART 使用的奇偶校验类型。	00
		0x0	无奇偶校验。	
		0x1	保留。	
		0x2	偶校验。向每个字符添加一个数据位，以使发送器中的 1 的个数为偶数，这样预期接收字符中 1 的个数也将为偶数。	
		0x3	奇数奇偶校验。向每个字符添加一个数据位，以使发送器中的 1 的个数为奇数，这样预期接收字符中 1 的个数也将为奇数。	
6	STOPLEN		附加到已发送数据的停止位个数。接收的数据仅需一个停止位。	0
		0	1 个停止位。	
		1	2 个停止位。该设置应当仅用于异步通信。	
7	-		保留。只能将 0 写入该位。	
8	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
9	CTSEN		CTS 使能。决定 CTS 是否用于流控制。若环回模式使能，CTS 既可来自输入引脚，也可来自 USART 自身的 RTS。更多信息，请参见章节 15.7.3。	0
		0	无流控制。发送器不接收任何自动流控制信号。	
		1	流控制使能。发送器使用 CTS 输入 （或环回模式下的 RTS 输出）控制流。	
10	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
11	SYNCEN		选择同步或异步运行。	0
		0	选定异步模式。	
		1	选定同步模式。	

表 160. USART 配置寄存器 （CFG，地址 0x4006 4000 (USART0)， 0x4006 8000 (USART1)，0x4006 C000 (USART2)）位说明 *（续）*

位	符号	值	说明	复位值
12	CLKPOL		选择时钟极性和同步模式下的接收数据采样边沿。	0
		0	下降沿。 Un_RXD 在 SCLK 的下降沿采样。	
		1	上升沿。 Un_RXD 在 SCLK 的上升沿采样。	
13	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
14	SYNCMST		同步模式主机选择。	0
		0	从机。使能同步模式时， USART 为从机。	
		1	主机。使能同步模式时， USART 为主机。	
15	LOOP		选择数据环回模式。	0
		0	正常操作。	
		1	环回模式。该模式提供执行 USART 数据诊断回路测试的机制。发送器中的串行数据 (Un_TXD) 在内部连接至接收器的串行输入 (Un_RXD)。若这些功能配置为在器件引脚上出现，则 Un_TXD 和 Un_RTS 活动将同样出现在外部引脚上。接收器 RTS 信号如果由 CTSEN 使能，则同样会环回到 CTS，并执行流控制。	
31:16	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

15.6.2 USART 控制寄存器

CTRL 寄存器控制 USART 运行的各个方面，容易在工作中发生改变。

表 161. USART 控制寄存器 （CTRL，地址 0x4006 4004 (USART0)， 0x4006 8004 (USART1)，0x4006 C004 (USART2)）位说明

位	符号	值	说明	复位值
0	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
1	TXBRKEN		中止使能。	0
		0	正常操作。	
		1	一旦设置该位，则连续中止会被立即发送，并在该位被清零前保持。 若首先禁用 （设置 CTRL 中的 TXDIS）的是发送器，则在不损坏当前传输字符的情况下可发送中止，然后在将 1 写入 TXBRKEN之前等待发送机被禁用（STAT中的TXDISINT = 1）。	
2	ADDRDET		使能地址检测模式。	0
		0	使能。针对所有输入数据使能 USART 接收器。	
		1	禁用。 USART 接收器忽略最高有效位 （通常为第 9 位）不为 1 的输入数据。数据最高有效位为 1 时，接收器正常处理输入数据，产生一个接收数据中断。软件随后便检查数据，判断是否是需要处理的地址。如果是需要处理的地址，则软件会清零 ADDRDET 位，并且后续输入数据会被正常处理。	
5:3	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
6	TXDIS		发送禁用。	0
		0	未禁用。 USART 发送器未禁用。	
		1	禁用。当前发送的所有字符完成发送后， USART 发送器会被禁用。该特性可用于为软件流控制提供便利。	
7	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

表 161. USART 控制寄存器 （CTRL，地址 0x4006 4004 (USART0)， 0x4006 8004 (USART1)，0x4006 C004 (USART2)）位说明

位	符号	值	说明	复位值
8	CC		连续时钟生成。默认情况下，只有在同步模式下传送数据时，SCLK 才输出。	0
		0	时钟字符。同步模式下，SCLK 仅在字符正在 Un_TXD 上发送时循环，或完成接收字符时才发送循环。	
		1	连续时钟。同步模式下 SCLK 连续运行，允许 Un_RxD 上的字符接收可独立于 Un_TXD 上的传输。	
9	CLRCC		清零连续时钟。	0
		0	对 CC 位无影响。	
		1	自动清零。接收到完整字符时， CC 位会被自动清零。同时，该位也会被清零。	
31:10	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

15.6.3 USART 状态寄存器

STAT 寄存器主要提供一组完整的 USART 状态标志，用于软件读取。除了只读标志，其他标志的清零可通过将 1 写入相应的 STAT 位实现。那些无法通过软件清零的只读中断状态标志可使用 INTENCLR 寄存器 （参见表 164）屏蔽。

一旦检测到接收噪声、奇偶校验错误、成帧错误和溢出等错误标志，则会立即对这些标志进行设置，并且在通过 STAT 中的软件动作得到清零之前保持设置。

表 162. USART 状态寄存器 （STAT，地址 0x4006 4008 (USART0)， 0x4006 8008 (USART1)， 0x4006 C008 (USART2)）位说明

位	符号	说明	复位值	访问类型 [1]
0	RXRDY	接收器就绪标志。该位为1时，表示可从接收器缓冲区读取数据。对RXDATA 寄存器或 RXDATASTAT 寄存器进行读操作后清零。	0	RO
1	RXIDLE	接收器空闲。该位为 0 时，表示接收器当前正在接收数据。该位为 1 时，表示接收器当前未接收数据。	1	RO
2	TXRDY	发送器就绪标志。该位为 1 时，表示数据可写入发送器缓冲区。先前的数据可能仍然在发送中。数据写入 TXDATA 时清零。数据从发送缓冲区移至发送移位寄存器时设置。	1	RO
3	TXIDLE	发送器空闲。该位为 0 时，表示发送器当前正在发送数据。该位为 1 时，表示发送器当前未发送数据。	1	RO
4	CTS	该位反映 CTS 信号的当前状态，而无论 CFG 寄存器中的 CTSEN 位设置如何。除非环回模式使能，否则它将作为 CTS 输入引脚值。	不适用	RO
5	DELTACTS	检测到上述 CTS 标志状态发生变化时，该位会被设置。该位通过软件清零。	0	W1
6	TXDISINT	发送器禁用中断标志。该位为 1 时，表示 USART 发送器通过 CFG 寄存器中的 TXIDS 被禁用 (TXDIS = 1) 后进入完全空闲状态。	0	RO
7	-	保留。读取的数值为未定义型，且只应写入零值。	不适用	不适用
8	OVERRUNINT	溢出错误中断标志。接收到新字符时，若此时接收器缓冲区正在使用中，则该标志会被设置。若发生这种情况则移位寄存器将丢失最近接收到的字符。	0	W1
9	-	保留。读取的数值为未定义型，且只应写入零值。	不适用	不适用

表 162. USART 状态寄存器 （STAT，地址 0x4006 4008 (USART0)， 0x4006 8008 (USART1)， 0x4006 C008 (USART2)）位说明 *（续）*

位	符号	说明	复位值	访问类型 [1]
10	RXBRK	接收器中止。该位反映接收器中止检测逻辑的当前状态。若 Un_RXD 引脚保持 16 位时间长度的低电平，则该位会被设置。需要注意的是，发生这种情况时，由于针对字符的停止位可能丢失，因此 FRAMERRINT 也将被设置。Un_RXD 引脚变为高电平时， RXBRK 会被清零。	0	RO
11	DELTARXBRK	检测到接收器中止状态发生变化时，该位会被设置。通过软件清零。	0	W1
12	启动	接收器输入端检测到启动信号时，该位会被设置。它的目的主要是检测到启动信号时，允许立即从深度睡眠或掉电模式中唤醒。通过软件清零。	0	W1
13	FRAMERRINT	成帧错误中断标志。在预期位置收到不含停止位的字符时，该标志会被设置。它可以表示波特率或配置与传送源发生了失配。	0	W1
14	PARITYERRINT	奇偶校验错误中断标志。在接收字符中检测到奇偶校验错误时，该标志会被设置。	0	W1
15	RXNOISEINT	接收噪声中断标志。获取接收数据的三个样本，以便决定每个接收数据位的值 （同步模式除外）。只要有一个样本不赞同，就表现为噪声滤波器。接收到的数据位包含一个不赞同样本时，该标志会被设置。它可以表示线路噪声、波特率或字符格式失配，或者数据接收过程中失去同步。	0	W1
31:16	-	保留。读取的数值为未定义型，且只应写入零值。	不适用	不适用

[1] RO = 只读， W1 = 写入 1 以清零。

15.6.4 USART 中断使能读取和设置寄存器

INTENSET 寄存器用于使能多个 USART 中断源。 INTENSET 中的使能位在与 STAT 寄存器中标志对应的位置中映射。完整的中断使能可从该寄存器中读取。将 1 写入该寄存器中已实施的位可设置这些位。 INTENCLR 寄存器用于清零该寄存器中的位。

表 163. USART 中断使能读取和设置寄存器（INTENSET，地址 0x4006 400C (USART0)，0x4006 800C (USART1)， 0x4006 C00C (USART2)）位说明

位	符号	说明	复位值
0	RXRDYEN	该位为 1 时，如果可从 RXDATA 寄存器读取接收字符，则会使能中断。	0
1	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
2	TXRDYEN	该位为 1 时，如果 TXDATA 寄存器可用于发送另一个字符，则会使能中断。	0
4:3	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
5	DELTACTSEN	该位为 1 时，如果 CTS 输入状态发生变化，则会使能中断。	0
6	TXDISINTEN	该位为 1 时，如果发送器如 STAT 中的 TXDISINT 标志所示被完全禁用，则会使能中断。有关详情，请参见 TXDISINT 位说明。	0
7	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
8	OVERRUNEN	该位为 1 时，如果发生溢出错误，则会使能中断。	0
10:9	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
11	DELTARXBRKEN	该位为 1 时，如果在检测已接收中止条件 （中止条件置位或解除置位）时状态已发生变化，则会使能中断。	0
12	STARTEN	该位为 1 时，如果已检测到接收启动位，则会使能中断。	0
13	FRAMERREN	该位为 1，如果已检测到成帧错误，则会使能中断。	0

表 163. USART 中断使能读取和设置寄存器（INTENSET，地址 0x4006 400C (USART0)，0x4006 800C (USART1)， 0x4006 C00C (USART2)）位说明 *（续）*

位	符号	说明	复位值
14	PARITYERREN	该位为 1 则检测到奇偶校验错误时，使能中断。	0
15	RXNOISEEN	该位为 1 时，如果检测到噪声，则会使能中断。参见表 162 中的 RXNOISEINT 位说明。	0
31:16	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

15.6.5 USART 中断使能清零寄存器

INTENCLR 寄存器用于清零 INTENSET 寄存器中的位。

表 164. USART 中断使能清零寄存器 （INTENCLR，地址 0x4006 4010 (USART0)， 0x4006 8010 (USART1)， 0x4006 C010 (USART2)）位说明

位	符号	说明	复位值
0	RXRDYCLR	写入 1 可清零 INTENSET 寄存器中的相应位。	0
1	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
2	TXRDYCLR	写入 1 可清零 INTENSET 寄存器中的相应位。	0
4:3	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
5	DELTACTSCLR	写入 1 可清零 INTENSET 寄存器中的相应位。	0
6	TXDISINTCLR	写入 1 可清零 INTENSET 寄存器中的相应位。	0
7	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
8	OVERRUNCLR	写入 1 可清零 INTENSET 寄存器中的相应位。	0
10:9	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
11	DELTARXBRKCLR	写入 1 可清零 INTENSET 寄存器中的相应位。	0
12	STARTCLR	写入 1 可清零 INTENSET 寄存器中的相应位。	0
13	FRAMERRCLR	写入 1 可清零 INTENSET 寄存器中的相应位。	0
14	PARITYERRCLR	写入 1 可清零 INTENSET 寄存器中的相应位。	0
15	RXNOISECLR	写入 1 可清零 INTENSET 寄存器中的相应位。	0
31:16	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

15.6.6 USART 接收器数据寄存器

RXDATA 寄存器包含溢出之前接收到的最后一个字符。

注：对该寄存器进行读操作会改变 RXDATASTAT 寄存器中的状态标志。

表 165. USART 接收器数据寄存器 （RXDATA，地址 0x4006 4014 (USART0)，0x4006 8014 (USART1)， 0x4006 C014 (USART2)）位说明

位	符号	说明	复位值
8:0	RXDAT	USART 接收器数据寄存器包含下一个接收到的字符。相关位的个数取决于 USART 配置设置。	0
31:9	-	保留，从保留位读取的值未定义。	不适用

15.6.7 含状态信息的 USART 接收器数据寄存器

RXDATASTAT 寄存器包含要被读取的下一个完整字符以及其相关的状态标志。它允许通过一次 16 位读操作获取所有与接收字符相关的信息。

注：对该寄存器进行读操作会改变状态标志。

表 166. 含状态信息的 USART 接收器数据寄存器（RXDATASTAT，地址 0x4006 4018 (USART0)，0x4006 8018 (USART1)， 0x4006 C018 (USART2)）位说明

位	符号	说明	复位值
8:0	RXDAT	USART 接收器数据寄存器包含下一个接收到的字符。相关位的个数取决于 USART 配置设置。	0
12:9	-	保留，从保留位读取的值未定义。	不适用
13	FRAMERR	成帧错误状态标志。 RXDATA 寄存器中存在可读取的字符时，该位有效；它反映那个字符的状态。在预期位置接收到不含停止位的 RXDAT 字符时，该位会被设置。它可以表示波特率或配置与传送源发生了失配。	0
14	PARITYERR	奇偶校验错误状态标志。 RXDATA 寄存器中存在可读取的字符时，该位有效；它反映那个字符的状态。在接收字符中检测到奇偶校验错误时，该位会被设置。	0
15	RXNOISE	接收噪声标志。参见表 162 中的 RxNoiseInt 位说明。	0
31:16	-	保留，从保留位读取的值未定义。	不适用

15.6.8 USART 发送器数据寄存器

对 TXDATA 寄存器进行写操作，以便通过 USART 发送器发送数据。该数据将在可用时发送至发送移位寄存器中，随后便可向 TXDATA 写入另一个字符。

表 167. USART 发送器数据寄存器（TXDATA，地址 0x4006 401C (USART0)， 0x4006 801C (USART1)， 0x4006 C01C (USART2)）位说明

位	符号	说明	复位值
8:0	TXDAT	向 USART 发送数据寄存器写入数据后，一旦发送移位寄存器可用并满足任意数据发送条件时，数据便可立即被发送：CTS 低电平（如果 CTSEN 位 = 1）， TXDIS 位 = 0。	0
31:9	-	保留。应当只写入 0。	不适用

15.6.9 USART 波特率发生器寄存器

波特率发生器是一个简单的 16 位整数分频器，由 BRG 寄存器控制。BRG 寄存器含有用于分频基础时钟的数值，可产生 USART 内部运行所需的时钟。

一个 16 位的数值允许产生的标准波特率范围为低至 300 波特甚至更低（最高器件频率）到高达 921,600 波特 （低至 14.7456 MHz 的基础时钟频率）。

通常，波特率时钟是实际波特率的16倍。该超频允许在一个位单元内将数据采样时间置中，并取走三个输入数据的样本用于减少噪声和检测。

为 BRG 选择正确值的方法详情，请参见本章稍后的章节 15.7.1。

注：如果软件需要更改波特率，则应当按照下列顺序：1) 确保 USART 当前不发送或接收数据。 2) 通过将 0 写入使能位禁用 USART （可将 0 写入整个寄存器） 。 3) 写入新的 BRGVAL。 4) 对 CFG 寄存器进行写操作，将使能位设置为 1。

表 168. USART 波特率发生器寄存器 （BRG，地址 0x4006 4020 (USART0)，0x4006 8020 (USART1)， 0x4006 C020 (USART2)）位说明

位	符号	说明	复位值
15:0	BRGVAL	该值基于 FRG 的输入时钟，用于分频 USART 输入时钟，以便决定波特率。 0 = FRG 时钟通过 USART 功能直接使用。 1 = FRG 时钟在使用前通过 USART 功能进行 2 分频。 2 = FRG 时钟在使用前通过 USART 功能进行 3 分频。 ... 0xFFFF = FRG 时钟在使用前通过 USART 功能进行 65,536 分频。	0
31:16	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

15.6.10 USART 中断状态寄存器

INTSTAT 只读寄存器提供当前使能的中断标志视图。它可以简化软件对中断的处理。有关中断标志的详细说明，请参见表 162。

表 169. USART 中断状态寄存器 （INTSTAT，地址 0x4006 4024 (USART0)，0x4006 8024 (USART1)， 0x4006 C024 (USART2)）位说明

位	符号	说明	复位值
0	RXRDY	接收器就绪标志。	0
1	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
2	TXRDY	发送器就绪标志。	1
4:3	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
5	DELTACTS	检测到 CST 输入的状态发生变化时，该位会被设置。	0
6	TXDISINT	发送器禁用中断标志。	0
7	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
8	OVERRUNINT	溢出错误中断标志。	0
10:9	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
11	DELTARXBRK	检测到接收器中止状态发生变化时，该位会被设置。	0
12	启动	接收器输入端检测到启动信号时，该位会被设置。	0
13	FRAMERRINT	成帧错误中断标志。	0

表 169. USART 中断状态寄存器 （INTSTAT，地址 0x4006 4024 (USART0)，0x4006 8024 (USART1)，0x4006 C024 (USART2)）位说明

位	符号	说明	复位值
14	PARITYERRINT	奇偶校验错误中断标志。	0
15	RXNOISEINT	接收噪声中断标志。	0
31:16	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

15.7 功能说明

15.7.1 时钟和波特率

要使用 USART，必须定义时钟详细信息，比如：设置 BRG，通常还需要设置 FRG。参见图 27。

15.7.1.1 小数速率发生器 (FRG)

如果外设时钟并非标准 （或所需）波特率的良好倍数，小数速率分频器可用于获取更精确的波特率。

FRG 通常被设置用于产生所需最高波特率的整数倍数（或与其极为近似的数）。BRG 随后用于获取所需的实际波特率。

FRG 寄存器可控制 USART 小数速率发生器，该发生器为 USART 提供基础时钟。小数速率发生器通过抑制选定的输入时钟，创建速率较低的输出时钟。不需要时，可针对 FRG 设置数值 0，这样随后便不会对输入时钟进行分频。

FRG 输出时钟定义为 1 + (MULT / 256) 分频的输入时钟，其中 MULT 的范围为 1 至 255。它允许产生一个输出时钟，范围为 1+1/256 至 1+255/256 （略大于 1 至略小于 2）分频的输入时钟。任何进一步分频都可通过每个 USART 模块完成，具体操作由每个 USART 中的整数 BRG 分频器完成。

FRG 产生的基础时钟不可能完全对称，因此 FRG 根据实际情况对输出时钟进行平均分配。由于 USART 通常使用 16 倍超频，因此这些情况下的小数速率时钟抖动将很有可能并不出现在最终的 USART 输出中。

通过下列寄存器设置小数分频器：

表 23 “USART 小数生成器分频器值寄存器（UARTFRGDIV，地址 0x4004 80F0）位说明 ”

表 24 “USART 小数生成器乘法器值寄存器（UARTFRGMULT，地址 0x4004 80F4）位说明 ”

有关详情，请参见章节 15.3.1 “ 配置 USART 时钟和波特率 ”。

15.7.1.2 波特率发生器 (BRG)

波特率发生器 （参见章节 15.6.9）用于分频基础时钟，产生所需波特率 16 倍的速率。通常，可通过对较高的波特率进行整数分频，产生标准波特率。

15.7.1.3 波特率计算

异步模式下的基础时钟为 16 倍，同步模式下为 1 倍。

15.7.2 同步模式

注：同步模式发送和接收在从机模式下的工作速率为输入时钟速率，在主机模式下的工作速率为 BRG 选择速率 （非 16 分频）。

15.7.3 流控制

USART 同时支持硬件和软件流量控制。

15.7.3.1 硬件流控制

USART 使用 RTS 和 / 或 CTS 信号支持硬件流控制。如果 RTS 配置为出现在某个器件引脚上以便被发送到外部器件，则表示接收器可从外部器件接收更多数据。

如果连接到某个引脚并使能，则 CTS 输入可允许外部器件对 USART 发送器进行节流控制。

图 29 显示的是 USART 中 RTS 和 CTS 的概述。

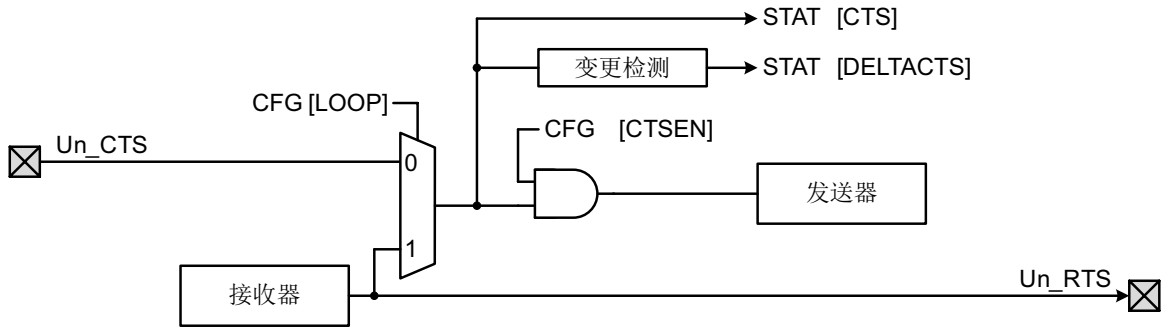


图 29. 使用 RTS 和 CTS 的硬件流控制

15.7.3.2 软件流控制

软件流控制可包含 XON/XOFF 流控制或其他机制。下述情况可实现这些机制：具有检查当前 CTS 输入状态的能力和 / 或 CTS 改变状态时发送一个中断的能力（分别通过 STAT 寄存器中的 CTS 和 DELTACTS 位），以及软件轻松关闭发送器的能力 （通过 CTRL 寄存器中的 TXDIS 位）。

16.1 本章导读

所有器件都提供 I2C 总线接口。

要想了解 I2C 的工作原理和软件接口，或者想学习如何使用 I2C 将器件从低功耗模式下唤醒，请阅读本章。

LPC800 提供基于 ROM 的片内 I2C API，以便配置并操作 I2C。参见表 258 “I2C API 调用”。

16.2 特性

- 独立的主机、从机和监控功能。
- 支持多主机功能以及带从机的多主机功能。
- 硬件中支持多个 I^2C 从机地址。
- 单个从机地址或地址范围可使用位掩码进行选择性认证，以便响应多个 I^2C 总线地址。
- 10 位寻址由软件协助提供支持。
- 支持 SMBus。

16.3 基本配置

使用下列寄存器配置 I2C：

- 在 SYSAHBCLKCTRL 中，设置位 5 （表 18）以使能寄存器接口时钟。
- 使用 PRESETCTRL 寄存器 （表 7）清零 I2C 外设复位。
- 在 NVIC 中使能 / 禁用 8 号中断插槽中的 I2C 中断。
- 通过开关矩阵配置 I2C 引脚功能。参见表 16.4。
- 采用系统时钟作为 I2C 外设时钟 （参见图 30）。

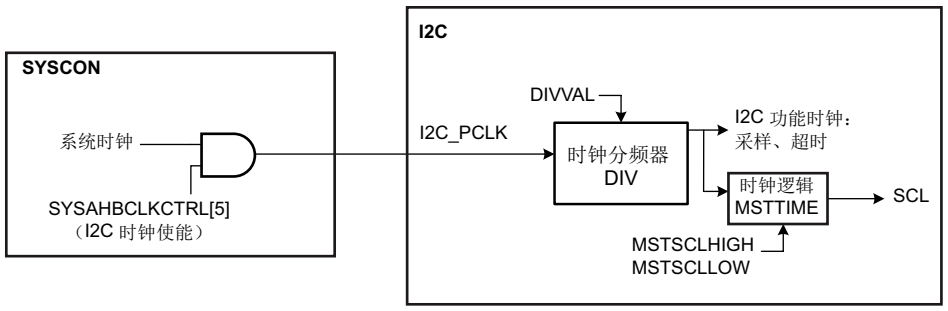


图 30. I2C 时钟

16.3.1 主机模式下的 I2C 发送 / 接收

本例中，LPC800 I2C 配置为主机。主机向从机发送 8 位数据，然后接收来自从机的 8 位数据。系统时钟设为 30 MHz，比特率约为 400 KHz。因此，可为 I2C0_SCL 和 I2C0_SDA 功能选择任意引脚。专用开漏 I2C 焊盘为可选。地址和数据位的传输由 MSTPENDING 状态位中的状态值控制。只要处于主机挂起状态，主机就可对 MSTDAT 寄存器进行读 / 写操作，然后通过写入 MSTCTRL 寄存器进入传输协议的下一步。

配置引脚：

- 通过开关矩阵为 I2C0_SCL 和 I2C0_SDA 选择两个引脚。参见表 170。
- 如果使用标准数字 I/O 引脚，则对于特定引脚需要在 IOCON 寄存器中禁用内部上拉。

配置 I2C 比特率：

- 系统时钟 (= I2C_PCLK) 的分频系数为 2。参见表 179 “I2C 分频器寄存器（DIV，地址 0x4005 0014）位说明 ”。
- 将 SCL 的高电平和低电平时间都设置为 2 个时钟周期。这是默认设置。参见表 182 “主机时间寄存器（MSTTIME，地址 0x4005 0024）位说明 ”。其结果是使 SCL 时钟频率为 375 kHz。

将 LPC800 I2C 配置为主机：在 CFG 寄存器中将 MSTEN 位设置为 1。参见表 172。

将数据写入从机：

1. $\overline{RW}$ 位设为 0 时，可将从机地址写入主机数据寄存器 MSTDAT。参见表 183。
2. 在主机控制寄存器中，将 MSTSTART 位设为 1 可开始进行传输。参见表 181。以下情况发生：
   - 挂起状态被清零， I2C 总线忙碌。
   - I2C 主机将开始位和带 $\overline{RW}$ 位的地址发送到从机。
3. 通过轮询 STAT 寄存器，等待挂起状态完成设置 (MSTPENDING = 1)。
4. 将 8 位数据写入 MSTDAT 寄存器。
5. 在主机控制寄存器中，将 MSTCONT 位设置为 1 可继续数据传输。参见表 181。以下情况发生：
   - 挂起状态被清零， I2C 总线忙碌。
   - I2C 主机将数据位发送到从机地址。
6. 通过轮询 STAT 寄存器，等待挂起状态完成设置 (MSTPENDING = 1)。
7. 在主机控制寄存器中，将 MSTSTOP 位设为 1 可结束传输。参见表 181。

从主机读取数据：

1. $\overline{RW}$ 位设为 1 时，可将从机地址写入主机数据寄存器 MSTDAT。参见图 183。
2. 在主机控制寄存器中，将 MSTSTART 位设为 1 可开始进行传输。参见表 181。以下情况发生：
   - 挂起状态被清零， I2C 总线忙碌。
   - I2C 主机将开始位和带 $\overline{RW}$ 位的地址发送到从机。
   - 从机发送 8 位数据。
3. 通过轮询 STAT 寄存器，等待挂起状态完成设置 (MSTPENDING = 1)。
4. 从 MSTDAT 寄存器读取 8 位数据。

5. 通过轮询 STAT 寄存器，等待挂起状态完成设置 (MSTPENDING = 1)。
6. 在主机控制寄存器中，将 MSTSTOP 位设为 1 可结束传输。参见表 181。

将数据写入从机并从从机回读 2 字节数据：

1. $\overline{RW}$ 位设为 0 时，可将从机地址写入主机数据寄存器 MSTDAT。参见图 183。
2. 在主机控制寄存器中，将 MSTSTART 位设为 1 可开始进行传输。参见表 181。以下情况发生：
   - 挂起状态被清零， I2C 总线忙碌。
   - I2C 主机将开始位和带 $\overline{RW}$ 位的地址发送到从机。
3. 通过轮询 STAT 寄存器，等待挂起状态完成设置 (MSTPENDING = 1)。
4. 将 8 位数据写入 MSTDAT 寄存器。
5. 在主机控制寄存器中，将 MSTCONTINUE 位设置为 1 可继续数据传输。参见表 181。以下情况发生：
   - 挂起状态被清零， I2C 总线忙碌。
   - I2C 主机将数据位发送到从机地址。
6. 通过轮询 STAT 寄存器，等待挂起状态完成设置 (MSTPENDING = 1)。
7. $\overline{RW}$ 位设为 1 时，可将从机地址写入主机数据寄存器 MSTDAT。参见图 183。
8. 在主机控制寄存器中，将 MSTSTART 位设为 1 可重新开始进行传输。参见表 181。以下情况发生：
   - 挂起状态被清零， I2C 总线忙碌。
   - I2C 主机将开始位和带 $\overline{RW}$ 位的地址发送到从机。
   - 从机发送 8 位数据。
9. 通过轮询 STAT 寄存器，等待挂起状态完成设置 (MSTPENDING = 1)。
10. 从 MSTDAT 寄存器读取首个数据字节。
11. 通过轮询 STAT 寄存器，等待挂起状态完成设置 (MSTPENDING = 1)。
12. 在主机控制寄存器中，将 MSTCONTINUE 位设为 1 可从从机重复读取数据。
13. 通过轮询 STAT 寄存器，等待挂起状态完成设置 (MSTPENDING = 1)。
14. 从 MSTDAT 寄存器读取第二个数据字节。
15. 在主机控制寄存器中，将 MSTSTOP 位设为 1 可结束传输。参见表 181。

16.3.2 配置 I2C 以实现唤醒

在睡眠模式下， I2C 总线上触发 I2C 中断的任何动作都会唤醒器件，前提是在 INTENSET 寄存器和 NVIC 中使能中断。在睡眠模式下，只要 I2C 时钟 I2C_PCLK 保持有效，I2C 就可唤醒器件，无论 I2C 模块是配置为主机模式还是从机模式。

在深度睡眠或关断模式下，I2C 时钟如所有外设时钟一样均被关断。然而，如果 I2C 配置为从机模式，并且 I2C 总线上的外部主机提供时钟信号，则 I2C 模块就能以异步方式建立中断。如果在 NVIC、 STARTERP1 寄存器和 I2C 模块的 INTENCLR 寄存器中使能该中断，则它可用于唤醒内核。

16.3.2.1 从睡眠模式唤醒

- 在 NVIC 中使能 I2C 中断。

- 在 I2C INTENSET 寄存器中使能 I2C 唤醒事件。任何已使能的中断均可唤醒（参见 INTENSET 寄存器）。参见以下事件实例：
  - 主机挂起
  - 转为空闲状态
  - 开始 / 停止错误
  - 从机挂起
  - 地址匹配（从机模式下）
  - 数据可用 / 就绪

16.3.2.2 从深度睡眠模式和掉电模式唤醒

- 在 NVIC 中使能 I2C 中断。
- 在 SYSCON 模块的 STARTERP1 寄存器中使能 I2C 中断，以便在内核和外设无时钟信号时，通过异步方式创建中断信号。参见表 34 “ 启动逻辑 1 中断唤醒使能寄存器（STARTERP1，地址 0x4004 8214）位说明 ”。
- 在 PDAWAKE 寄存器中，配置所有器件唤醒时需要运行的外设。
- 在从机模式下配置 I2C。
- 在 I2C INTENCLR 寄存器中使能 I2C 中断，可将中断配置为唤醒事件。参见以下事件实例：
  - 从机取消选定
  - 从机挂起（等待读取、写入或 ACK）
  - 地址匹配
  - 监控数据可用 / 就绪

16.4 引脚说明

I2C 引脚功能可转移，且通过开关矩阵分配给 LPC800 封装上的引脚。有两种方式可连接 I2C 引脚：

1. 连接专用 I2C 开漏引脚（PIO0_10 和 PIO0_11）。
2. 连接支持可移动功能的其他任意引脚。

I^2C 功能分配给专用 I^2C 引脚时，支持所有 I^2C 总线规格，最高可达超快速模式（高达 1 MHz I^2C）。

I^2C 功能分配给设定为开漏模式的标准引脚时，可以该方式使用功能性 I^2C 总线，但可能无法满足某些 I^2C 总线规格。这可能会影响总线速度、噪声过滤以及在不影响总线的情况下关断器件的能力。

有关如何将 I2C 引脚功能分配给 LPC800 封装上的任意引脚，请参见章节 9.3.1 “ 将内部信号与封装引脚相连 ”。

表 170. I2C 总线引脚说明

功能	类型	引脚	说明	SWM 寄存器	参考
I2C0_SCL	I/O	任意；使用引脚 PIO0_10 或 PIO0_11 以符合全部 I2C 总线规格。	I2C0 串行时钟。	PINASSIGN8	表 105
I2C0_SDA	I/O	任意；使用引脚 PIO0_10 或 PIO0_11 以符合全部 I2C 总线规格。	I2C0 串行数据。	PINASSIGN7	表 104

16.5 简介

I2C 总线接口架构如图 31 所示。

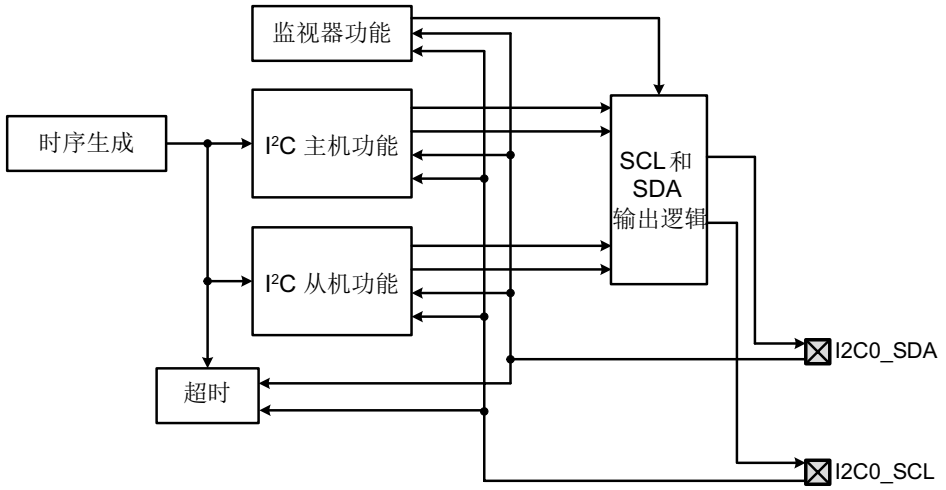


图 31. I2C 功能框图

16.6 寄存器说明

寄存器功能可按以下方式分组：

- 通用寄存器：
  - 表 172 “I2C 配置寄存器 （CFG，地址 0x4005 0000）位说明 ”
  - 表 173 “I²C 状态寄存器 （STAT，地址 0x4005 0004）位说明 ”
  - 表 180 “I²C 中断状态寄存器 （INTSTAT，地址 0x4005 0018）位说明 ”
  - 表 176 “ 中断使能设置和读寄存器 （INTENSET，地址 0x4005 0008）位说明 ”
  - 表 177 “ 中断使能清零寄存器 （INTENCLR，地址 0x4005 000C）位 说明 ”
  - 表 178 “ 超时寄存器 （TIMEOUT，地址 0x4005 0010）位说明 ”
  - 表 179 “I²C 分频器寄存器 （DIV，地址 0x4005 0014）位说明 ”
- 主机功能寄存器：
  - 表 181 “ 主机控制寄存器 （MSTCTL，地址 0x4005 0020）位说明 ”
  - 表 182 “ 主机时间寄存器 （MSTTIME，地址 0x4005 0024）位说明 ”
  - 表 183 “ 主机数据寄存器 （MSTDAT，地址 0x4005 0028）位说明 ”

- 从机功能寄存器：
  - 表 184 “ 从机控制寄存器 （SLVCTL，地址 0x4005 0040）位说明 ”
  - 表 184 “ 从机控制寄存器 （SLVCTL，地址 0x4005 0040）位说明 ”
  - 表 186 “ 从机地址寄存器（SLVADR[0:3]，地址 0x4005 0048 (SLVADR0) 至 0x4005 0054 (SLVADR3)）位说明 ”
  - 表 187 “ 从机地址验证器 0 寄存器 （SLVQUAL0， 0x4005 0058）位说明 ”
- 监控功能寄存器：表 188 “ 监控数据寄存器（MONRXDAT，地址 0x4005 0080）位说明 ”

表 171. 寄存器概述：I2C （基址 0x4005 0000）

名称	访问类型	偏移	说明	复位值	参考
CFG	R/W	0x00	共享功能配置。	0	表 172
STAT	R/W	0x04	主机、从机和监控功能状态寄存器。	0x000801	表 173
INTENSET	R/W	0x08	中断使能设置和读取寄存器。	0	表 176
INTENCLR	W	0x0C	中断使能清零寄存器。	不适用	表 177
TIMEOUT	R/W	0x10	超时值寄存器。	0xFFFF	表 178
DIV	R/W	0x14	整个 I²C 模块的时钟预分频器。它决定用于 MSTTIME 和 SLVTIME 寄存器的时间增量。	0	表 179
INTSTAT	R	0x18	主机、从机和监控功能中断状态寄存器。	0	表 180
MSTCTL	R/W	0x20	主机控制寄存器	0	表 181
MSTTIME	R/W	0x24	主机时序配置。	0x77	表 182
MSTDAT	R/W	0x28	主机接收器和发射器混合数据寄存器。	不适用	表 183
SLVCTL	R/W	0x40	从机控制寄存器。	0	表 184
SLVDAT	R/W	0x44	从机接收器和发送器混合数据寄存器。	不适用	表 185
SLVADR0	R/W	0x48	从机地址 0。	0x01	表 186
SLVADR1	R/W	0x4C	从机地址 1。	0x01	表 186
SLVADR2	R/W	0x50	从机地址 2。	0x01	表 186
SLVADR3	R/W	0x54	从机地址 3。	0x01	表 186
SLVQUAL0	R/W	0x58	地址 0 从机验证。	0	表 187
MONRXDAT	RO	0x80	监控接收器数据寄存器	0	表 188

16.6.1 I2C 配置寄存器

CFG 寄存器包含可用于主机、从机和监控功能的模式设定。

表 172. I2C 配置寄存器 （CFG，地址 0x4005 0000）位说明

位	符号	值	说明	复位值
0	MSTEN		主机使能。禁用时，主机功能的配置设定不会改变，但主机功能内部复位。	0
		0	禁用。 I²C 主机功能禁用。	
		1	使能。 I²C 主机功能使能。	
1	SLVEN		从机使能。禁用时，从机功能的配置设定不会改变，但从机功能内部复位。	0
		0	禁用。 I²C 从机功能禁用。	
		1	使能。 I²C 从机功能使能。	
2	MONEN		监控使能。禁用时，监控功能的配置设定不会改变，但监控功能内部复位。	0
		0	禁用。 I²C 监控功能禁用。	
		1	使能。 I²C 监控功能使能。	
3	TIMEOUTEN		I²C 总线超时使能。禁用时，超时功能内部复位。	0
		0	禁用。超时功能禁用。	
		1	使能。超时功能使能。将生成两种类型的超时标志，若使能则将产生中断。通常系统只需使用一次超时。	

表 172. I2C 配置寄存器 （CFG，地址 0x4005 0000）位说明 *（续）*

位	符号	值	说明	复位值
4	MONCLKSTR		监控功能时钟延伸。	0
		0	禁用。监控功能将不会执行时钟延伸。软件在数据写入前可能无法始终读取监控功能提供的数据。该模式在防入侵监控至关重要的情况下可使用。	
		1	使能。监控功能将执行时钟延伸，以便确保软件可读取该功能提供的所有输入数据。	
31:5	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

16.6.2 I2C 状态寄存器

STAT寄存器提供所有I²C模块功能的状态标志和状态信息。该寄存器内的某些信息为只读，并且某些标志可通过写入 1 来清零。

针对该寄存器中的位访问取决于：RO = 只读， W1 = 写入 1 以清零。

有关该寄存器中 MSTSTATE 位和 SLVSTATE 位针对主机和从机的状态说明，请参见表 174 和表 175。

表 173. I²C 状态寄存器 （STAT，地址 0x4005 0004）位说明

位	符号	值	说明	复位值	访问类型
0	MSTPENDING		主机挂起。表示主机等待 I2C 总线继续通信 （挂起）或主机空闲。主机挂起时， MSTSTATE 位表示主机等待的软件服务类型。如果通过 INTENSET 寄存器使能，则该标志将产生一个中断。若主机处于空闲状态且无需进行通信，则屏蔽该中断。	1	RO
		0	处理中。通信正在处理中，主机功能繁忙，无法立即接受命令。		
		1	挂起。主机功能需要软件服务，否则处于空闲状态。若主机不在空闲状态中，则等待接收或发送数据 （或 NACK 位）。		
3:1	MSTSTATE		主机状态码。当 MSTPENDING 位置位时，主机状态码反映主机状态，即主机挂起或空闲。该字段的每一个数值均表示主机功能的一个指定服务。保留所有其它值。	0	RO
		0x0	空闲。主机功能可用于下一次交换。		
		0x1	接收就绪。接收数据可用 （主机接收器模式） 。地址和读数据先前已由从机发送并获得认可。		
		0x2	发送就绪。数据可发送 （主机发送器模式） 。地址和写数据先前已由从机发送并获得认可。		
		0x3	NACK 地址。经 NACK 处理的从机地址。		
		0x4	NACK 数据。经 NACK 处理的从机发送数据。		
4	MSTARBLOSS		主机仲裁丢失标志。该标志可通过将 1 写入该位的软件来清除。也可通过将 1 写入 MSTCONTINUE 来自动清除。	0	W1
		0	无丢失。不发生仲裁丢失。		
		1	仲裁丢失。主机功能经历了一次仲裁丢失。 此时，主机功能已停止驱动总线并进入空闲状态。作为响应，软件可不采取任何动作，也可发送一个启动信号，以便在下次空闲时尝试获取总线控制。		
5	-		保留。读取的数值为未定义型，且只应写入零值。	不适用	不适用
6	MSTSTSTPERR		主机启动 / 停止错误标志。该标志可通过将 1 写入该位的软件来清除。也可通过将 1 写入 MstContinue 来自动清除。	0	W1
		0	未发生启动 / 停止错误。		
		1	发生启动 / 停止错误。主机功能经历了一次启动 / 停止错误。 在I²C规范不允许的时刻检测到启动或停止。主机接口停止驱动总线并进入空闲状态，无需采取任何行动。可建立一个启动请求，或者软件试图确保总线不会停止。		
7	-		保留。读取的数值为未定义型，且只应写入零值。	不适用	不适用

表 173. I²C 状态寄存器 （STAT，地址 0x4005 0004）位说明 *（续）*

位	符号	值	说明	复位值	访问类型
8	SLVPENDING		从机挂起。表示从机功能等待 I2C 总线继续通信并需要软件服务。如果通过 INTENSET 使能，则该标志将产生一个中断。SLVPENDING 标志为只读，并且在 SLVCTL 寄存器中将 1 写入 SLVCONTINUE 位时可自动清零。	0	RO
		0	处理中。从机功能当前不需要服务。		
		1	挂起。从机功能需要服务。有关需要何种内容的信息，可在相邻的 SLVSTATE 字段内找到。		
10:9	SLVSTATE		从机状态码。该字段的每一个数值均表示从机功能的一个指定服务。保留所有其它值。	0	RO
		0x0	从机地址 .. 接收到地址和 R/W 信息。至少有 1 个从机地址 （共 4 个）已通过硬件匹配。		
		0x1	从机接收。接收数据可用 （从机接收器模式）。		
		0x2	从机发送。数据可发送 （从机发送器模式）。		
		0x3	保留。		
11	SLVNOTSTR		从机未延伸。表示从机功能何时延伸 I²C 时钟。它用于在从机工作时轻松调用深度睡眠或掉电模式。该只读标志实时反映从机功能状态。	1	RO
		0	延伸。从机功能当前正在延伸 I²C 总线时钟。此时无法进入深度睡眠或掉电模式。		
		1	未延伸。从机功能当前没有延伸 I²C 总线时钟。此时可进入深度睡眠或掉电模式。		
13:12	SLVIDX		从机地址匹配索引。当 I²C 从机功能通过接收地址而选定时，该字段有效；该地址匹配其中一个从机地址，而该从机地址由任意使能从机地址寄存器定义，并提供匹配地址的鉴定。可能存在多个匹配地址，但此时只能汇报一个匹配地址。	0	RO
		0x0	从机地址 0 匹配。		
		0x1	从机地址 1 匹配。		
		0x2	从机地址 2 匹配。		
		0x3	从机地址 3 匹配。		
14	SLVSEL		从机选定标志。 SLVSEL 于地址匹配后进行置位，此时软件告知从机功能以确认地址时。从机软件决定 NACK 一个匹配地址或总线检测到停止信号时，如果另一个地址周期存在地址数据且该地址不匹配从机功能的某个使能地址，则清零该位。如果软件 NACK 了数据，则不清零 SLVSEL 位。	0	RO
		0	未选定。从机功能当前未选定。		
		1	选定。从机功能当前已选定。		
15	SLVDESEL		从机取消选定标志。如果通过 INTENSET 使能，则该标志将产生一个中断。该标志可通过将 1 写入该位来清零。	0	W1
		0	未取消选定。从机功能未变为取消选定。这不表示当前已选定。该信息可在 SLVSEL 标志中找到。		
		1	取消选定。从机功能变为取消选定。这主要是由于 SLVSEL 标志从 1 变为 0 而造成的。有关该事件何时发生的详情，请参见 SLVSEL 说明。		
16	MONRDY		监控就绪。该标志在对 MONRXDAT 寄存器进行读操作时被清零。	0	RO
		0	无数据。监控功能当前不提供数据。		
		1	数据等待。监控功能有数据等待被读取。		

表 173. I²C 状态寄存器 （STAT，地址 0x4005 0004）位说明 *（续）*

位	符号	值	说明	复位值	访问类型
17	MONOV		监控上溢标志。	0	W1
		0	无溢出。监控数据未溢出。		
		1	溢出。发生监控数据溢出。只有在 CFG 寄存器的 MONCLKSTR 位未使能监控时钟延伸的情况下才会发生。将 1 写入该位可清零该标志。		
18	MONACTIVE		监控激活标志。该标志表示监控功能认为 I²C 总线已激活。总线上有一些主机功能时，可定义该激活：总线启动先于总线停止发生。	0	RO
		0	无效。监控功能认为 I²C 总线未激活。		
		1	激活。监控功能认为 I²C 总线已激活。		
19	MONIDLE		监控空闲标志。当监控功能发现 I²C 总线从激活变为未激活时，设置该标志。软件可据此决定何时处理监控功能积累的数据。如果通过 INTENSET 寄存器使能，则该标志将产生一个中断。该标志可通过将 1 写入该位来清零。	0	W1
		0	未空闲。 I²C 总线未空闲，或软件已清零该标志。		
		1	空闲。自软件清零上一次标志起， I²C 总线至少经历了一次空闲。		
23:20	-		保留。读取的数值为未定义型，且只应写入零值。	不适用	不适用
24	EVENTTIMEOUT		事件超时中断标志。表示两次事件之间的时间长于 TIMEOUT 寄存器中指定的时间。事件包括启动、停止和时钟边沿。该标志可通过将 1 写入该位来清零。 I2C 总线空闲时，未造成超时。	0	W1
		0	未超时。 I²C 总线事件未造成超时。		
		1	事件超时。两次 I²C 总线事件之间的时间长于 I2C TIMEOUT 寄存器指定的时间。		
25	SCLTIMEOUT		SCL 超时中断标志。表示 SCL 保持低电平的时间长于 TIMEOUT 寄存器指定的时间。该标志可通过将 1 写入该位来清零。	0	W1
		0	未超时。 SCL 低电平时间未造成超时。		
		1	超时。 SCL 低电平时间造成超时。		
31:26	-		保留。读取的数值为未定义型，且只应写入零值。	不适用	不适用

表 174. 主机功能状态码 (MSTSTATE)

MstState	说明	操作
0	**空闲。**主机功能可用于下一次交换。	若当前不需要使用主机功能，则发送启动信号或禁用 MSTPENDING 中断。
1	**接收数据可用 （主机接收器模式）。**地址和读数据先前已由从机发送并获得认可。	读取数据，然后继续、发送停止信号或者发送重复启动信号。
2	**数据可发送 （主机发送器模式）。**地址和写数据先前已由从机发送并获得认可。	发送数据，然后继续、发送停止信号或者发送重复启动信号。
3	**经 NACK 处理的从机地址。**	发送停止或重复启动信号。
4	**经 NACK 处理的从机发送数据。**	发送停止或重复启动信号。

表 175. 从机功能状态码 (SLVSTATE)

SlvState	说明	操作
0	**接收到地址和 R/W 信息。**至少有 1 个从机地址 （共 4 个）已通过硬件匹配。	若有需要，软件可进一步检查地址，例如若需要使用经 SLVQUAL0 验证的地址子集。通过向 SLVCONTINUE 或 SLVNACK 写入 1，软件可 ACK 或 NACK 该地址。有关 10 位寻址，另请参见章节 16.7.2。
1	**接收数据可用 （从机接收器模式）。**	读取数据以 ACK 或 NACK 作出应答。
2	**数据可发送 （从机发送器模式）。**	发送数据。
3	保留。	-

16.6.3 中断使能设置和读寄存器

INTENSET 寄存器控制哪个 I²C 状态标志用于产生中断。若 STAT 寄存器支持中断，则将 1 写入 INTENSET 寄存器中的位位置可使能 STAT 寄存器中对应位的中断。读取 INTENSET 便可知当前使能的是哪一个中断。

表 176. 中断使能设置和读寄存器 （INTENSET，地址 0x4005 0008）位说明

位	符号	值	说明	复位值
0	MSTPENDINGEN		主机挂起中断使能。	0
		0	MstPending 中断禁用。	
		1	MstPending 中断使能。	
3:1	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
4	MSTARBLOSSEN		主机仲裁丢失中断使能。	0
		0	MstArbLoss 中断禁用。	
		1	MstArbLoss 中断使能。	
5	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
6	MSTSTSTPERREN		主机启动 / 停止错误中断使能。	0
		0	MstStStpErr 中断禁用。	
		1	MstStStpErr 中断使能。	
7	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
8	SLVPENDINGEN		从机挂起中断使能。	0
		0	SlvPending 中断禁用。	
		1	SlvPending 中断使能。	
10:9	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
11	SLVNOTSTREN		从机未延伸中断使能。	0
		0	SlvNotStr 中断禁用。	
		1	SlvNotStr 中断使能。	
14:12	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
15	SLVDESELEN		从机取消选定中断使能。	0
		0	SlvDeSel 中断禁用。	
		1	SlvDeSel 中断使能。	
16	MONRDYEN		监控数据就绪中断使能。	0
		0	MonRdy 中断禁用。	
		1	MonRdy 中断使能。	
17	MONOVEN		监控溢出中断使能。	0
		0	MonOv 中断禁用。	
		1	MonOv 中断使能。	
18	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
19	MONIDLEEN		监控空闲中断使能。	0
		0	MonIdle 中断禁用。	
		1	MonIdle 中断使能。	
23:20	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

表 176. 中断使能设置和读寄存器 （INTENSET，地址 0x4005 0008）位说明 *（续）*

位	符号	值	说明	复位值
24	EVENTTIMEOUTEN		事件超时中断使能。	0
		0	事件超时中断禁用。	
		1	事件超时中断使能。	
25	SCLTIMEOUTEN		SCL 超时中断使能。	0
		0	SCL 超时中断禁用。	
		1	SCL 超时中断使能。	
31:26	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

16.6.4 中断使能清零寄存器

向 INTENCLR 中的位写入 1 可清零 INTENSET 寄存器中的对应位，从而禁用中断。INTENCLR 为只写寄存器。

与 INTENSET 中定义的位无关的位予以保留，并且这些位应当只可写入零。

表 177. 中断使能清零寄存器 （INTENCLR，地址 0x4005 000C）位 说明

位	符号	说明	复位值
0	MSTPENDINGCLR	主机挂起中断清零。若实施，则将 1 写入该位可清零 INTENSET 寄存器中的相应位。	0
3:1	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
4	MSTARBLOSSCLR	主机仲裁丢失中断清零。	0
5	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
6	MSTSTSTPERRCLR	主机启动 / 停止错误中断清零。	0
7	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
8	SLVPENDINGCLR	从机挂起中断清零。	0
10:9	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
11	SLVNOTSTRCLR	从机未延伸中断清零。	0
14:12	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
15	SLVDESELCLR	从机取消选定中断清零。	0
16	MONRDYCLR	监控数据就绪中断清零。	0
17	MONOVCLR	监控溢出中断清零。	0
18	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
19	MONIDLECLR	监控空闲中断清零。	0
23:20	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
24	EVENTTIMEOUTCLR	事件超时中断清零。	0
25	SCLTIMEOUTCLR	SCL 超时中断清零。	0
31:26	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

16.6.5 超时值寄存器

TIMEOUT 寄存器允许设定某些 I^2C 总线时间的上限值，当那些时间超出设定值时通过状态标志和 / 或中断进行通知。

产生两个超时，且软件可选择使用其中任何一个。

1. 当总线非空闲时，EVENTTIMEOUT 可检查两次总线事件之间的时间间隔：启动、SCL 上升、SCL 下降和停止。若任意两次事件之间的时间间隔大于 TIMEOUT 寄存器中的设定值，则将设置 STAT 寄存器中的 EVENTTIMEOUT 状态标志。若 INTENSET 寄存器中的 EVENTTIMEOUTEN 位使能，则 EVENTTIMEOUT 状态标志可产生一次中断。
2. 当总线非空闲时，SCLTIMEOUT 仅检查 SCL 信号保持低电平的时间。若 SCL 保持低电平的时间长于 TIMEOUT 寄存器中的设定值，则将设置 STAT 寄存器中的 SCLTIMEOUT 状态标志。若 INTENSET 寄存器中的 SCLTIMEOUTEN 位使能，则 SCLTIMEOUT 状态标志可产生一次中断。 SCLTIMEOUT 可与 SMBus 一同使用。

另请参见表 16.7.2 “ 超时 ”。

表 178. 超时寄存器 （TIMEOUT，地址 0x4005 0010）位说明

位	符号	说明	复位值
3:0	TOMIN	超时时间值，最后 4 位。固定为 0xF。 I²C 功能时钟的最小超时为 16， I²C 功能时钟的超时分辨率为 16。	0xF
15:4	TO	超时时间值 I²C 功能时钟的指定超时间隔增量为 16，由 CLKDIV 寄存器定义。若要在 I²C 工作时改变该数值，则需禁用所有超时、向 TIMEOUT 写入新数值，并重新使能超时。 0x000 = I²C 功能时钟经 16 次计数后会产生一次超时。 0x001 = I²C 功能时钟经 32 次计数后会产生一次超时。 ... 0xFFF = I²C 功能时钟经 65,536 次计数后会产生一次超时。	0xFFF
31:16	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

16.6.6 I2C 时钟分频器寄存器

CLKDIV 寄存器对外设时钟 (PCLK) 分频，以产生 I²C 功能时钟，用于 I²C 接口的各种时序。I²C 功能时钟用于 I²C 模块中的某些内部操作，以及用于产生 I²C 总线规格所需的时序；其中，某些时钟在 MSTTIME 寄存器中可让用户配置，用作主机操作；或者在 SLVTIME 寄存器中配置，用作从机操作。

有关总线速率的设置，参见章节 16.7.1.1 “ 速率计算 ”。

表 179. I²C 分频器寄存器 （DIV，地址 0x4005 0014）位说明

位	符号	说明	复位值
15:0	DIVVAL	该字段控制 I²C 功能如何使用使用该时钟 (PCLK)； 该功能需要一个内部时钟才能工作。 0x0000 = I²C 功能直接使用 PCLK。 0x0001 = 供 I²C 功能使用前先对 PCLK 进行 2 分频处理。 0x0002 = 供 I²C 功能使用前先对 PCLK 进行 3 分频处理。 ... 0xFFFF = 供 I²C 功能使用前先对 PCLK 进行 65,536 分频处理。	0
31:16	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

16.6.7 I2C 中断状态寄存器

INTSTAT 寄存器提供当前使能的中断标志视图。它可以简化软件对中断的处理。有关中断标志的详细说明，请参见表 173。

表 180. I²C 中断状态寄存器 （INTSTAT，地址 0x4005 0018）位说明

位	符号	说明	复位值
0	MSTPENDING	主机挂起。	1
3:1	-	保留。	
4	MSTARBLOSS	主机仲裁丢失标志。	0
5	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
6	MSTSTSTPERR	主机启动 / 停止错误标志。	0
7	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
8	SLVPENDING	从机挂起。	0
10:9	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
11	SLVNOTSTR	从机未延伸状态。	1
14:12	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
15	SLVDESEL	从机取消选定标志。	0
16	MONRDY	监控就绪。	0
17	MONOV	监控上溢标志。	0
18	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
19	MONIDLE	监控空闲标志。	0
23:20	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
24	EVENTTIMEOUT	事件超时中断标志。	0
25	SCLTIMEOUT	SCL 超时中断标志。	0
31:26	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

16.6.8 主机控制寄存器

MSTCTL 寄存器含有控制 I²C 主机接口各种功能的位。主机挂起时，该寄存器为只写（STAT 寄存器中的 MSTPENDING = 1，表 173）。

表 181. 主机控制寄存器 （MSTCTL，地址 0x4005 0020）位说明

位	符号	值	说明	复位值
0	MSTCONTINUE		主机继续。该位为只写。	0
		0	无效。	
		1	继续。通知主机功能继续下一操作。该操作必须在写入传输数据、读取接收数据或完成任何与总线操作有关的数据保存操作之后才可进行。	
1	MSTSTART		主机启动控制。该位为只写。	0
		0	无效。	
		1	起动。在下一个允许的时间内，I²C 总线上将产生一个启动信号。	
2	MSTSTOP		主机停止控制。该位为只写。	0
		0	无效。	
		1	停止。若主机从从机接收数据 （主机接收器模式），则在下一个允许的时间内，I²C 总线上将产生一个停止信号，跟在 NACK 之后发送给从机。	
31:2	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

16.6.9 主机时间

MSTTIME 寄存器允许设置某几个时间，可通过主机功能控制。它们包括时钟 (SCL) 高电平和低电平时间、重复启动建立时间和传输数据建立时间。

I2C 时钟预分频器如表 179 所述。

表 182. 主机时间寄存器 （MSTTIME，地址 0x4005 0024）位说明

位	符号	值	说明	复位值
2:0	MSTSCLLOW		主机 SCL 低电平时间。指定将被该主机 SCL 置位的最小低电平时间。总线上的其他设备 （主机或从机）可延长该时间。在 I^2C 总线规格中，它对应参数 t_{LOW}。I^2C 总线规范参数 t_{BUF} 和 $t_{SU;STA}$ 具有相同的数值，且都由 MSTSCLLOW 控制。	0
		0x0	2 个时钟周期。SCL 低电平最小值为 I^2C 时钟预分频器的 2 个时钟周期。	
		0x1	3 个时钟周期。SCL 低电平最小值为 I^2C 时钟预分频器的 3 个时钟周期。	
		0x2	4 个时钟周期。SCL 低电平最小值为 I^2C 时钟预分频器的 4 个时钟周期。	
		0x3	5 个时钟周期。SCL 低电平最小值为 I^2C 时钟预分频器的 5 个时钟周期。	
		0x4	6 个时钟周期。SCL 低电平最小值为 I^2C 时钟预分频器的 6 个时钟周期。	
		0x5	7 个时钟周期。SCL 低电平最小值为 I^2C 时钟预分频器的 7 个时钟周期。	
		0x6	8 个时钟周期。SCL 低电平最小值为 I^2C 时钟预分频器的 8 个时钟周期。	
		0x7	9 个时钟周期。SCL 低电平最小值为 I^2C 时钟预分频器的 9 个时钟周期。	
6:4	MSTSCLHIGH		主机 SCL 高电平时间。指定将被该主机 SCL 断言的最小高电平时间。多主机系统中的其他主机可缩短该时间。在 I^2C 总线规格中，它对应参数 t_{HIGH}。I^2C 总线规格参数 $t_{SU;STO}$ 和 $t_{HD;STA}$ 具有相同的数值，且都由 MSTSCLHIGH 控制。	0
		0x0	2 个时钟周期。SCL 高电平最小值为 I^2C 时钟预分频器的 2 个时钟周期。	
		0x1	3 个时钟周期。SCL 高电平最小值为 I^2C 时钟预分频器的 3 个时钟周期。	
		0x2	4 个时钟周期。SCL 高电平最小值为 I^2C 时钟预分频器的 4 个时钟周期。	
		0x3	5 个时钟周期。SCL 高电平最小值为 I^2C 时钟预分频器的 5 个时钟周期。	
		0x4	6 个时钟周期。SCL 高电平最小值为 I^2C 时钟预分频器的 6 个时钟周期。	
		0x5	7 个时钟周期。SCL 高电平最小值为 I^2C 时钟预分频器的 7 个时钟周期。	
		0x6	8 个时钟周期。SCL 高电平最小值为 I^2C 时钟预分频器的 8 个时钟周期。	
		0x7	9 个时钟周期。SCL 高电平最小值为 I^2C 时钟预分频器的 9 个时钟周期。	
31:7	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

16.6.10 主机数据寄存器

MSTDAT 寄存器提供为主机功能读取最近接收到的数据，以及使用主机功能传输数据的方法。

表 183. 主机数据寄存器 （MSTDAT，地址 0x4005 0028）位说明

位	符号	说明	复位值
7:0	DATA	主机功能数据寄存器。 读：针对主机功能读取最近接收到的数据。 写入：使用主机功能传输数据。	0
31:8	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

16.6.11 从机控制寄存器

SLVCTL 寄存器含有控制 I^2C 从机接口各种功能的位。从机挂起时，该寄存器为只写（STAT 寄存器中的 SLVPENDING = 1，表 173）。

表 184. 从机控制寄存器 （SLVCTL，地址 0x4005 0040）位说明

位	符号	值	说明	复位值
0	SLVCONTINUE		从机继续。	0
		0	无效。	
		1	继续。通知从机功能继续下一操作。该操作必须在写入传输数据、读取接收数据或完成任何与总线操作有关的数据保存操作之后才可进行。	
1	SLVNACK		从机 NACK。	0
		0	无效。	
		1	NACK。当从机接收主机的数据时，可造成从机功能对主机的 NACK 处理 （从机接收器模式）。	
31:2	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

16.6.12 从机数据寄存器

SLVDAT 寄存器提供为从机功能读取最近接收到的数据，以及使用从机功能传输数据的方法。

表 185. 从机数据寄存器 （SLVDAT，地址 0x4005 0044）位说明

位	符号	说明	复位值
7:0	DATA	从机功能数据寄存器。 读：针对从机功能读取最近接收到的数据。 写入：使用从机功能传输数据。	0
31:8	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

16.6.13 从机地址寄存器

SLVADR[0:3] 寄存器允许使能并定义某个地址，该地址可被 I²C 从机硬件自动识别。SLVADR0 寄存器中的值由 SLVQUAL0 寄存器中的设定验证。

当从机地址与接收到的地址比较时，该比较会受到 SLVQUAL0 寄存器的设定影响（见章节 16.6.14）。

I²C 从机功能有 4 个地址比较器。另外 3 个地址比较器不支持地址验证功能。处理通用地址调用时， 4 个地址寄存器中的一个可设定为对地址 0 作出响应。

表 186. 从机地址寄存器 （SLVADR[0:3]，地址 0x4005 0048 (SLVADR0) 至 0x4005 0054 (SLVADR3)）位说明

位	符号	值	说明	复位值
0	SADISABLE		从机地址 n 禁用。	1
		0	使能。从机地址 n 使能，其变化将由 SLVQUAL0 寄存器中的指定内容识别。	
		1	忽略从机地址 n。	
7:1	SLVADR		若使能，则 7 位从机地址将与接收地址作比较。	0
31:8	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

16.6.14 从机地址验证器 0 寄存器

SLVQUAL0 寄存器可更改针对从机地址 0 的解读结果。

表 187. 从机地址验证器 0 寄存器 （SLVQUAL0， 0x4005 0058）位说明

位	符号	值	说明	复位值
0	QUALMODE0		保留。读取的数值为未定义型，且只应写入零值。	0
		0	SLVQUAL0 字段用于匹配地址 0 的逻辑屏蔽。	
		1	SLVQUAL0 字段用于扩展一定范围内地址 0 的匹配。	
7:1	SLVQUAL0		地址 0 从机地址验证器。数值 0 使 SLVADR0 中的地址按原样使用 （假设已使能）。 如果 QUALMODE0 = 0，则与 SLVADR0 寄存器比较时，本字段内任意设置为 1 的位将使接收到的地址与相应的位进行自动匹配。 如果 QUALMODE0 = 1，则地址范围匹配地址 0。该范围从 SLVADR0 定义的地址扩展至 SLVQUAL0 定义的地址（当 SLVADR0[7:1] <= 接收到的地址 <= SLVQUAL0[7:1] 时，地址匹配）。	0
31:8	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

16.6.15 监控数据寄存器

MONRXDAT 只读寄存器提供 I²C 总线上事件信息，主要在应用开发中为 I²C 调试提供方便。所有数据地址和总线上传递的数据、它们是否经确认、以及启动和停止事件均被报告。

监控功能必须通过 CFG 寄存器的 MONEN 位使能。如果数据未及时从 MONRXDAT 寄存器中读取，则可通过 CFG 寄存器中的 MONCLKSTR 位配置监控模式，以延伸 I²C 时钟。这有助于确保不错过任何数据，但一定程度上可能造成监控功能的侵略性 （取决于系统响应时间，可能会增加时钟延迟）。如果未使能时钟延伸，则为了增加收集所有监控信息的可能性，监控数据会被缓冲，以便其在下一个 I²C 总线数据结束前可访问。

表 188. 监控数据寄存器 （MONRXDAT，地址 0x4005 0080）位说明

位	符号	值	说明	复位值
7:0	MONRXDAT		监控功能接收器数据。它反映 I²C 引脚上通过的所有数据字节，并添加启动、重复启动和数据 NACK 指示。	0
8	MONSTART		监控接收启动。	0
		0	不检测。监控功能未检测到 I²C 总线上的启动事件。	
		1	启动检测。监控功能检测到 I²C 总线上的启动事件。	
9	MONRESTART		监控接收重复启动。	0
		0	不启动检测。监控功能未检测到 I²C 总线上的重复启动事件。	
		1	重复启动检测。监控功能检测到 I²C 总线上的重复启动事件。	
10	MONNACK		监控接收 NACK。	0
		0	确认。监控功能提供的当前数据受到至少一个主机或从机接收器的确认。	
		1	未确认。监控功能提供的当前数据未受到任何接收器确认。	
31:11	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

16.7 功能说明

16.7.1 总线速率和时序考虑因素

由于 I²C 总线的性质，通常无法确保 SCL 引脚具有特定的时钟速率。在 I2C 总线上，时钟可通过任意从机设备延伸、通过软件整理操作时间扩展等。在一个多主机系统中，只要某个主机正在进行 I2C 传输，则提供最短 SCL 高电平时间的该主机将使 SCL 具有该时间 （例如，当总线上仅有该主机时，或两个主机间的仲裁时）。

若无减速事件，则速率计算将给出代表 I²C 总线最快工作速度的基频。

16.7.1.1 速率计算

SCL 高电平时间 （I²C 功能时钟）= (CLKDIV + 1) * (MSTSCLHIGH + 2)

SCL 低电平时间 （I²C 功能时钟）= (CLKDIV + 1) * (MSTSCLLOW + 2)

标称 SCL 速率 = I²C 功能时钟速率 /（SCL 高电平时间 + SCL 低电平时间）

16.7.2 超时

I²C 接口上的超时特性可用于检测 “ 阻塞 ” 总线，并有可能作出一些操作以缓解该情况。支持两种不同类型的超时。一旦 I²C 模块和超时功能均使能，则两种类型都适用；主机、从机或监控功能无需使能。

第一类超时由 STAT 寄存器的 EVENTTIMEOUT 标志反映，两次总线事件之前的时间将主导超时检查。这些事件包括启动、停止和所有 I²C 时钟 (SCL) 的变化。当任意两个事件之间的时间超过了 TIMEOUT 寄存器中的配置，则该超时生效。该超时用于监控系统中的 I²C 总线时非常有效，可作为总线在发生问题时依然保持工作的方法之一。

第二类 I²C 超时通过 STAT 寄存器中的 SCLTIMEOUT 标志反映。当 SCL 信号保持低电平的时间超出了 TIMEOUT 寄存器中的配置，则该超时生效。它对应于 SMBus 超时参数：$T_{TIMEOUT}$。在该情况下，若它是一个冲突设备，则从机可复位其自身的 I²C 接口。如果所有监听从机（包括可被寻址为从机的主机）都执行该操作，除非有问题的是当前主机，否则总线将被释放。有关更多详情，请参见 SMBus 规范。

I²C 总线繁忙时，会同时产生两种类型的超时。

16.7.3 10 位寻址

通过 I²C 主机发送第二个地址字节，扩展标准 7 位地址的特定范围，达到 10 位寻址的目的。在主机向从机写入信息的情形中，I²C 帧在 2 个地址字节之后便继续提供内容。对于从从机读取数据的主机，在第二个地址字节后需将数据方向反转。这通过发送一个重复启动信号完成，后面重复出现同一个带有读取位的标准 7 位地址。从机必须记住，上一次写操作已寻址，并且从机需要保持选定状态，以便进行后续部分 I²C 地址的正确读取。

对于主机功能而言，I2C 被要求执行正常写操作的 2 字节寻址，后接更多的写入数据或重复启动信号，对从机地址的前 10 位部分进行重复，然后以正常方式读取。

对于从机功能而言，地址的第一部分以与 7 位寻址同样的方式进行自动匹配。从机地址验证功能（参见章节 16.6.14）可用于截取全部可能具有的 10 位地址（首位地址字节值范围为 F0 至 F6），也可仅截取一个。在从机接收器模式下，软件将首个数据字节与 10 位地址的剩余部分相匹配后，数据便以正常方式接收。从机功能应记录其自身已被寻址的事实，以防后续还有读操作。

对于从机发送器模式而言，从机功能以与从机接收器模式相同的方式对初始地址作出响应，并检查其之前是否已经过完整的 10 位寻址。若匹配的地址为地址 0，并且使能地址验证，则软件必须检查 10 位地址的第一部分与先前的地址完全匹配，随后才可确认该地址。

16.7.4 时钟和电源考虑因素

I²C 的主机功能需采用外设时钟才可工作。当从机未被寻址时，从机功能可在没有任何内部时钟的情况下工作。这表示器件可进入的模式为低功耗模式和掉电模式，以及这两种模式之间的所有模式；并且当 I²C 从机功能识别出一个地址时，器件会被唤醒。当接收到信息时，监控模式也能够以类似的方式将器件从低功耗模式中唤醒。

16.7.5 中断

I2C 提供单个中断输出，可处理主机、从机和监控功能的所有中断。

17.1 本章导读

所有器件都提供 SPI0。仅器件 LPC812M101FDH16 和 LPC812M101FDH20 提供 SPI1。

17.2 特性

- 直接支持 1 至 16 位数据帧。通过软件可支持位数更多的帧。
- 主机和从机运行。
- 无需读取输入数据，即可将数据发送到从机。在设置 SPI 存储器时，这可能非常有用。
- 控制信息可选择性地与数据一同写入器件。这就实现了各种操作，包括 “ 任意长度 ” 的帧。
- 具有可选极性和灵活用途的从机选择输入 / 输出。

注：不支持德州仪器公司的 SSI 模式和 Microwire 模式。

17.3 基本配置

使用下列寄存器配置 SPI0/1：

- 在 SYSAHBCLKCTRL 中，设置位 11 和 12 （表 18）以使能寄存器接口时钟。
- 使用 PRESETCTRL 寄存器 （表 7）清零 SPI0/1 外设复位。
- 在 NVIC 中使能 / 禁用 0 号和 1 号中断插槽中的 SPI0/1 中断。
- 通过开关矩阵配置 SPI0/1 引脚功能。参见表 17.4。
- 采用系统时钟作为这两个 SPI 的外设时钟 （参见表 3 “LPC800 时钟生成 ”）。

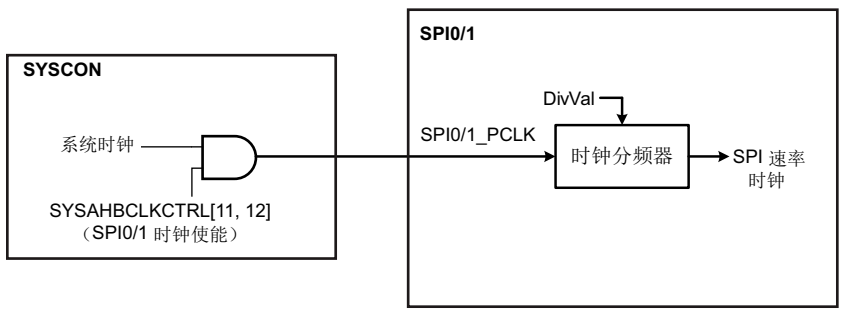


图 32. SPI 时钟

17.3.1 配置 SPI 以实现唤醒

在睡眠模式下，触发 SPI 中断的任何信号都会唤醒器件，前提是在 INTENSET 寄存器和 NVIC 中使能中断。在睡眠模式下，无论 SPI 模块是在主机模式下还是从机模式下进行配置，只要 SPI 时钟 SPI_PCLK 保持有效， SPI 就可唤醒器件。

在深度睡眠或掉电模式下，SPI 时钟如同所有外设时钟一样会被关断。然而，如果 SPI 在从机模式下配置且外部主机提供时钟信号，则 SPI 模块能以异步方式建立中断。如果该中断在 STARTERP1 寄存器、 NVIC 和 SPI 的 INTENSET 寄存器中使能，则可唤醒内核。

17.3.1.1 从睡眠模式唤醒

- 在主机模式或从机模式下配置 SPI。参见表 191。
- 在 NVIC 中使能 SPI 中断。
- 任意 SPI 中断均可将器件从睡眠模式中唤醒。在 INTENSET 寄存器 （表 194）中使能 SPI 中断。

17.3.1.2 从深度睡眠模式或掉电模式唤醒

- 在从机模式下配置 SPI。参见表 191。必须将 SCK 功能分配给某个引脚并将该引脚与主机相连。
- 在 STARTERP1 寄存器中使能 SPI 中断。参见表 34 “ 启动逻辑 1 中断唤醒使能寄存器（STARTERP1，地址 0x4004 8214）位说明 ”。
- 在 PDAWAKE 寄存器中，配置所有器件唤醒时需要运行的外设。
- 在 NVIC 中使能 SPI 中断。
- 在 INTENSET 寄存器中使能中断，可将中断配置为唤醒事件 （表 194）。参见以下唤醒事件实例：
  - SSEL 引脚状态的变化。
  - 要接收的可用数据。
  - 接收器溢出。

17.4 引脚说明

SPI 信号为可转移功能，并且通过开关矩阵分配到引脚。

有关如何将 SPI 功能分配给 LPC800 封装上的引脚，请参见表 9.3.1 “ 将内部信号与封装引脚相连 ”。

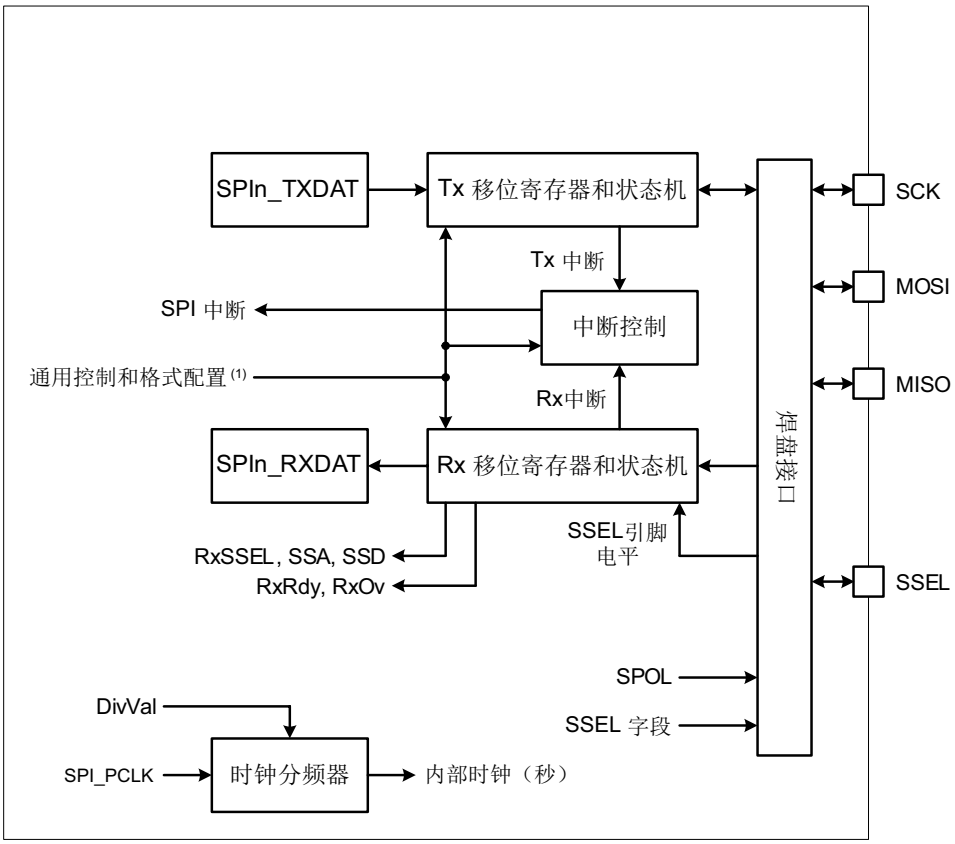
表 189. SPI 引脚说明

功能	方向	引脚	说明	SWM 寄存器	参考
SPI0_SCK	I/O	任意	**串行时钟。**SCK 是用来同步数据传送的时钟信号。它由主机驱动，从机接收。使用 SPI 接口时，时钟可编程为高电平有效或低电平有效。SCK 只在数据传输期间跳变。一旦 CFG 寄存器中的主机位等于 1，便将其驱动，而无论使能位的状态如何。	PINASSIGN3	表 100
SPI0_MOSI	I/O	任意	**主机输出从机输入。**MOSI 信号可将串行数据从主机传输到从机。SPI 为主机时，串行数据从该信号输出。SPI 为从机时，它记录从该信号发出的串行数据。一旦 CFG 寄存器中的主机位等于 1，便驱动 MOSI，而无论使能位的状态如何。	PINASSIGN4	表 101
SPI0_MISO	I/O	任意	**主机输入从机输出。**MISO 信号可将串行数据由从机传送到主机。SPI 为主机时，串行数据从该信号输入。SPI 为从机时，串行数据输出至该信号。SPI 模块使能、CFG 寄存器中的主机位等于 0 且一个或多个 SSEL 信号选中从机时，驱动 MISO。	PINASSIGN4	表 101

表 189. SPI 引脚说明

功能	方向	引脚	说明	SWM 寄存器	参考
SPI0_SSEL	I/O	任意	**从机选择。**SPI 接口为主机时，它会在串行数据启动前驱动 SSEL 信号，使之变为有效状态，并在串行数据发送后释放该信号，使之变为无效状态。默认情况下，该信号为低电平有效，但可将其选为高电平有效。SPI 为从机时，处于有效状态的任意 SSEL 信号都表示该从机正在被寻址。一旦 CFG 寄存器中的主机位等于 1，便驱动 SSEL 引脚，而无论使能位的状态如何。	PINASSIGN4	表 101
SPI1_SCK	I/O	任意	串行时钟。	PINASSIGN4	表 101
SPI1_MOSI	I/O	任意	主机输出从机输入。	PINASSIGN5	表 102
SPI1_MISO	I/O	任意	主机输入从机输出。	PINASSIGN5	表 102
SPI1_SSEL	I/O	任意	从机选择。	PINASSIGN5	表 102

17.5 简介



(1) 包括 CPOL、CPHA、LSBF、FLEN、主机、使能、transfer_delay、frame_delay、pre_delay、post_delay、SOT、EOT、EOF、RXIgnore、独立中断使能。

图 33. SPI 功能框图

17.6 寄存器说明

复位值仅反映已使用位中保存的数据，不包括保留位的内容。

表 190. 寄存器概述：SPI （基址 0x4005 8000 (SPI0) 和 0x4008 C000 (SPI1)）

名称	访问类型	偏移	说明	复位值	参考
CFG	R/W	0x000	SPI 配置寄存器	0	表 191
DLY	R/W	0x004	SPI 延迟寄存器	0	表 192
STAT	R/W	0x008	SPI 状态。某些状态标志的清零可通过将 1 写入该位完成。	0x0102	表 193
INTENSET	R/W	0x00C	SPI 中断使能读取和设置。完整数值可从该寄存器中读取。将 1 写入任意已实施的位都将设置该位。	0	表 194

表 190. 寄存器概述：SPI （基址 0x4005 8000 (SPI0) 和 0x4008 C000 (SPI1)）*（续）*

名称	访问类型	偏移	说明	复位值	参考
INTENCLR	W	0x010	SPI 中断使能清零。将 1 写入任意已实施的位都将导致INTENSET中的对应位被清零。	不适用	表 195
RXDAT	R	0x014	SPI 接收数据	不适用	表 196
TXDATCTL	R/W	0x018	SPI 控制数据传输	0	表 197
TXDAT	R/W	0x01C	SPI 传输数据	0	表 198
TXCTL	R/W	0x020	SPI 传输控制	0	表 199
DIV	R/W	0x024	SPI 时钟分频器	0	表 200
INTSTAT	R	0x028	SPI 中断状态	0x02	表 201

17.6.1 SPI 配置寄存器

CFG 寄存器含有进行 SPI 普通配置所需的信息。通常，该信息在工作时不会发生变化。SPI 未完全闲置时，不应进行 CPOL、CPHA 和 LSBF 等配置。有关更多信息，请参见（表 193 中的）闲置状态说明。

注: 如果接口从主机模式重新配置为从机模式，或者从从机模式配置为主机模式（较少见），则应当禁用 SPI，然后采用新配置再次使能。

表 191. SPI 配置寄存器 （CFG，地址 0x4005 8000 (SPI0)， 0x4005 C000 (SPI1)）位说明

位	符号	值	说明	复位值
0	使能		SPI 使能。	0
		0	禁用。 SPI 禁用，内部状态机和计数器复位。	
		1	使能。 SPI 使能，可进行工作。	
1	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
2	主机		主机模式选择	0
		0	从机模式。SPI 将在从机模式下工作。SCK、MOSI 和 SSEL 信号为输入，MISO 为输出。	
		1	主机模式。 PI 将在主机模式下工作。 SCK、 MOSI 和 SSEL 信号为输出， MISO 为输入。	
3	LSBF		LSB 优先模式使能。	0
		0	标准。数据以标准 MSB 优先的顺序发送和接收。	
		1	反向。数据以反向顺序 （LSB 优先）传输和接收。	
4	CPHA		时钟相位选择。	0
		0	改变。 SPI 在帧的首次时钟跃迁时捕获串行数据 （时钟从静态转变为其他状态）。数据在下列边沿上发生变化。	
		1	捕获。 SPI 在帧的首次时钟跃迁时更改串行数据 （时钟从静态转变为其他状态）。数据在下列边沿上被捕获。	
5	CPOL		时钟极性选择。	0
		0	低电平。时钟静态 （两帧之间）时为低电平。	
		1	高电平。时钟静态 （两帧之间）时为高电平。	
6	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
7	LOOP		环回模式使能。环回模式仅适用于主机模式，并且将传输和接收数据相连，以便软件可轻松测试。	0
		0	禁用。	
		1	使能。	
8	SPOL		SSEL 极性选择。	0
		0	低电平。SSEL 引脚为低电平有效。与 SSEL 有关的 RXDAT、TXDATCTL 和 TXCTL 寄存器中的 SSEL 字段值相对引脚未发生反向。	
		1	高电平。SSEL 引脚为高电平有效。与 SSEL 有关的 RXDAT、TXDATCTL 和 TXCTL 寄存器中的 SSEL 字段值相对引脚发生反向。	
31:9	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

17.6.2 SPI 延迟寄存器

DLY 寄存器控制数个与 SPI 信号有关的可编程延迟。这些延迟仅适用于主机模式，并且均在 SPI 时钟中声明。

时序详情见：

表 17.7.2.1 “Pre_delay 和 Post_delay”

章节 17.7.2.2 “Frame_delay”

章节 17.7.2.3 “Transfer_delay”

表 192. SPI 延迟寄存器 （DLY，地址 0x4005 8004 (SPI0)， 0x4005 C004 (SPI1)）位说明

位	符号	说明	复位值
3:0	PRE_DELAY	控制 SSEL 置位和数据帧起点之间的时间。 在SSEL置位和第一个时钟沿之间始终存在一个SPI时钟长度。不视为预延迟的一部分。 0x0 = 未插入额外时间。 0x1 = 插入 1 个 SPI 时钟时间。 0x2 = 插入 2 个 SPI 时钟时间。 ... 0xF = 插入 15 个 SPI 时钟时间。	0
7:4	POST_DELAY	控制数据帧末尾和 SSEL 解除置位之间的时间。 0x0 = 未插入额外时间。 0x1 = 插入 1 个 SPI 时钟时间。 0x2 = 插入 2 个 SPI 时钟时间。 ... 0xF = 插入 15 个 SPI 时钟时间。	0
11:8	FRAME_DELAY	控制相邻数据帧之间的最短时间。 0x0 = 未插入额外时间。 0x1 = 插入 1 个 SPI 时钟时间。 0x2 = 插入 2 个 SPI 时钟时间。 ... 0xF = 插入 15 个 SPI 时钟时间。	0
15:12	TRANSFER_DELAY	控制两次传输之间 SSEL 解除置位的最短时间。 0x0 = SSEL 解除置位的最短时间为 1 个 SPI 时钟时间。（零累加时间。） 0x1 = SSEL 解除置位的最短时间为 2 个 SPI 时钟时间。 0x2 = SSEL 解除置位的最短时间为 3 个 SPI 时钟时间。 ... 0xF = SSEL 解除置位的最短时间为 16 个 SPI 时钟时间。	0
31:16	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

17.6.3 SPI 状态寄存器

STAT 寄存器提供可被软件读取的 SPI 状态标志以及强制结束传输的控制位。除了只读标志，其他标志的清零可通过将 1 写入相应的 STAT 位实现。

STAT 含有 2 个错误标志（仅从机模式下）：RXOV 和 TXUR。它们分别是接收器溢出和发送器欠载。工作时如果发生任意一种错误，则应禁用 SPI，然后再次使能，以便尝试恢复工作前确保所有内部状态都被清零。

该寄存器中使用的是下列记号：RO = 只读， W1 = 写入 1 以清零。

表 193. SPI 状态寄存器 （STAT，地址 0x4005 8008 (SPI0)， 0x4005 C008 (SPI1)）位说明

位	符号	说明	复位值	访问 [1]
0	RXRDY	接收器就绪标志。该位为 1 时，表示可从接收器缓冲区读取数据。对 RXDAT 寄存器进行读操作后清零。	0	RO
1	TXRDY	发送器就绪标志。该位为 1 时，表示数据可写入发送器缓冲区。先前的数据可能仍然在发送中。数据写入 TXDAT 或 TXDATCTL 并转移至发送移位寄存器后，将其清零。	1	RO
2	RXOV	接收器溢出中断标志。该标志仅适用于从机模式 （主机 = 0）。检测到接收字符的起点时，如果此刻接收器缓冲区正在使用中，则设置该标志。如果发生这种情况，则接收器缓冲区内容将被保留，输入数据将丢失。如果设置了 RxOv，则 SPI 接收到的数据应视为未定义。	0	W1
3	TXUR	发送器欠载中断标志。该标志仅适用于从机模式 （主机 = 0） 。在这种情况下，如果发送器空闲，则应当从下一个输入时钟起，开始发送新的数据。如果此刻该数据不在发送器的保留寄存器中，则不发送数据，同时设置 TXUR 标志。如果设置了 TXUR，则 SPI 发送的数据应视为未定义。	0	W1
4	SSA	从机选择置位。无论在主机或是从机模式下，一旦任意从机选择从解除置位转换到置位状态，则设置该标志。这就能确定 SPI 发送 / 接收功能何时变为忙碌状态，从而在从机模式访问开始时将器件从低功耗模式中唤醒。该标志通过软件清零。	0	W1
5	SSD	从机选择解除置位。无论在主机模式还是从机模式下，一旦任意置位后的从机选择转换到解除置位状态，则设置该标志。这就能决定 SPI 发送 / 接收功能何时变为空闲状态。该标志通过软件清零。	0	W1
6	STALLED	停止状态标志。它会指示当前 SPI 是否处于停止条件。	0	RO
7	ENDTRANSFER	结束传输控制位。如同 EOT 标志先于最后传输而完成设置一样，如果发送器已完成所有处理中的活动，则可通过软件设置该位，强制结束当前传输。该功能用于为不确定是否完成数据传输（从而不确定是否结束传输）的情况提供支持。传输结束时，发送器变为空闲状态，该位会被清零。以这种方式强制结束传输将置位所有指定的 FrameDelay 和 TransferDelay。	0	RO/W1
8	IDLE	空闲状态标志。一旦 SPI 主机功能完全空闲，该位为 1。这表示发送保持寄存器内容为空，并且发送器当前未发送数据。	1	RO
31:9	-	保留。读取的数值为未定义型，且只应写入零值。	不适用	不适用

[1] RO = 只读， W1 = 写入 1 以清零。

17.6.4 SPI 中断使能读取和设置寄存器。

INTENSET 寄存器用于使能多个 SPI 中断源。INTENSET 中的使能位在与 STAT 寄存器中标志对应的位置中映射。完整的中断使能可从该寄存器中读取。将 1 写入该寄存器中已实施的位可设置这些位。 INTENCLR 寄存器用于清零该寄存器中的位。有关中断的详情，请参见表 193。

表 194. SPI 中断使能读取和设置寄存器 （INTENSET，地址 0x4005 800C (SPI0)， 0x4005 C00C (SPI1)）位说明

位	符号	值	说明	复位值
0	RXRDYEN		接收器数据可用时，可确定是否产生中断。	0
		0	接收器数据可用时，不会产生中断。	
		1	RXDAT 寄存器中的接收器数据可用时，会产生中断。	
1	TXRDYEN		发送器保持寄存器可用时，可确定是否产生中断。	0
		0	发送器保持寄存器可用时，不会产生中断。	
		1	数据可写入 TXDAT 时，会产生中断。	
2	RXOVEN		接收器溢出时，可确定是否产生中断。该判断发生在从机模式下，若接收器正在使用中，则此时它需将最近接收到的数据转入 RXDAT 寄存器。 通过不允许进行新的传输（否则可能造成接收器溢出），接口可在主机模式下防止接收器溢出。	0
		0	接收器溢出时，不会产生中断。	
		1	接收器溢出时，会产生中断。	
3	TXUREN		发送器溢出时，可确定是否产生中断。需要发送数据时，如果任何器件都不可用，则在从机中进行判断。	0
		0	发送器欠载时，不会产生中断。	
		1	发送器欠载时，会产生中断。	
4	SSAEN		置位从机选择时，可确定是否产生中断。	0
		0	任意从机选择从解除置位转换到置位都不会产生中断。	
		1	任意从机选择从解除置位转换到置位都会产生中断。	
5	SSDEN		解除置位从机选择时，可确定是否产生中断。	0
		0	所有从机选择从置位转换为解除置位时，不会产生中断。	
		1	所有从机选择从置位转换为解除置位时，会产生中断。	
31:6	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

17.6.5 SPI 中断使能清零寄存器

INTENCLR 寄存器用于清零 INTENSET 寄存器中的中断使能位。

表 195. SPI 中断使能清零寄存器（INTENCLR，地址 0x4005 8010 (SPI0)，0x4005 C010 (SPI1)）位说明

位	符号	说明	复位值
0	RXRDYEN	写入 1 可清零 INTENSET 寄存器中的相应位。	0
1	TXRDYEN	写入 1 可清零 INTENSET 寄存器中的相应位。	0
2	RXOVEN	写入 1 可清零 INTENSET 寄存器中的相应位。	0
3	TXUREN	写入 1 可清零 INTENSET 寄存器中的相应位。	0
4	SSAEN	写入 1 可清零 INTENSET 寄存器中的相应位。	0
5	SSDEN	写入 1 可清零 INTENSET 寄存器中的相应位。	0
31:6	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

17.6.6 SPI 接收器数据寄存器

RXDAT 只读寄存器提供了读取最新接收数据的方法。 SSEL 的值可随数据一同读取。

有关从机选择过程的详细信息，请参见章节 17.7.4。

表 196. SPI 接收器数据寄存器（RXDAT，地址 0x4005 8014 (SPI0)，0x4005 C014 (SPI1)）位说明

位	符号	说明	复位值
15:0	RXDAT	接收器数据。包含下一个接收到的数据。使用的位数取决于 TXCTL / TXDATCTL 中的 FLen 设置。	未定义
16	RXSSELN	接收数据的从机选择。该字段允许 SSEL 引脚状态与接收到的数据一同保存。该数值将同时反映主机操作和从机操作的 SSEL 引脚状态。0 表示从机选择有效。每个从机选择引脚的实际极性可在 CFG 寄存器的相应 SPOL 位中配置。	未定义
19:17	-	保留。	-
20	SOT	传送开始标志。如果该位为 SSEL 从解除置位转换到置位状态后的第一个帧数据（例如，先前的传输已结束），则该标志为 1。如果帧长度大于 16 位，则该信息可用于识别第一个数据。	
31:21	-	保留，从保留位读取的值未定义。	不适用

17.6.7 SPI 发送器数据和控制寄存器

TXDATCTL 寄存器提供了一处位置，在该处可同时写入发送数据和控制信息。这样就能对 SPI 进行详细控制，而无需对每个数据单独写入控制信息。

控制信息在传输中保持静态时，应当使用TXDAT寄存器（参见章节17.6.8）而非TXDATCTL寄存器。然后，便可通过 TXCTL 寄存器（参见章节 17.6.9）单独写入控制信息。TXDATCTL 的前半部（位 27 到位 16）与 TXCTL 寄存器中的位相同。这两个寄存器只不过提供了两种不同的访问方式。

有关从机选择过程的详细信息，请参见章节 17.7.4。

有关针对长度大于 16 位的帧使用多个连续帧的详细信息，请参见章节 17.7.5 “ 数据长度大于 16 位 ”。

表 197. SPI 发送器数据和控制寄存器 （TXDATCTL，地址 0x4005 8018 (SPI0)， 0x4005 C018 (SPI1)）位说明

位	符号	值	说明	复位值
15:0	TXDAT		发送数据。该字段可传送 1 至 16 位的数据。	0
16	TXSSELN		发送从机选择。该字段可控制主机模式下的 SSEL 输出。 **注：** SSEL 功能的有效状态可通过 CFG 寄存器中的位配置。	0
		0	置位 SSEL。	
		1	未置位 SSEL。	
19:17	-		保留。	
20	EOT		发送结束。置位后的 SSEL 将在传输结束时解除置位，并且至少保持 DLY 寄存器的 Transfer_delay 位指定的时间。	0
		0	未解除置位 SSEL。该数据不作为传输结束的数据。数据结束时 SSEL 不会解除置位。	
		1	解除置位 SSEL。该数据作为传输结束的数据。数据结束时 SSEL 将解除置位。	
21	EOF		帧结束。可在两帧之间插入一个延迟，由 DLY 寄存器的 Frame_delay 值定义。如果 FRAME_DELAY 值为 0，则帧结束可能并没有特殊含义。该控制可用于部分支持长度大于 16 位的帧。	0
		0	非 EOF 数据。发送的该数据不作为帧结束。	
		1	EOF 数据。该数据作为帧结束使用，可在后续数据发送前置位 FRAME_DELAY 时间。	
22	RXIGNORE		接收忽略。它允许使用 SPI 发送数据，而无需从接收器读取不需要的数据，简化发送过程。	0
		0	读取接收数据。必须读取接收数据，以便继续传输过程。从机模式下，如果未在新数据接收前读取接收数据，则发生溢出。	
		1	忽略接收数据。接收数据被忽略，从而允许以不读取不需要的接收数据的方式进行传输。不产生接收器标志。	

表 197. SPI 发送器数据和控制寄存器 （TXDATCTL，地址 0x4005 8018 (SPI0)， 0x4005 C018 (SPI1)）位说明 *（续）*

位	符号	值	说明	复位值
23	-		保留。读取的数值为未定义型，且只应写入零值。	不适用
27:24	FLEN		帧长度。指定帧长度为 1 至 16 位。需要注意的是，长度大于 16 位的帧将通过实施多个连续帧而支持。 需要注意的是，如果选定 1 位帧，则主机功能将始终在 SCK 引脚上的单个时钟长度之后插入一个长度为 SCK 时长的延迟。 0x0 = 数据帧长度为 1 位。 0x1 = 数据帧长度为 2 位。 0x2 = 数据帧长度为 3 位。 ... 0xF = 数据帧长度为 16 位。	0x0
31:28	-		保留。读取的数值为未定义型，且只应写入零值。	不适用

17.6.8 SPI 发送器数据寄存器

如果控制信息在传输中未发生变化，则对 TXDAT 寄存器进行写操作可通过 SPI 发送器发送数据 （参见章节 17.6.7）。该数据在可用时将发送至发送移位寄存器中，随后另一个字符可写入 TXDAT。

表 198. SPI 发送器数据寄存器（TXDAT，地址 0x4005 801C (SPI0)，0x4005 C01C (SPI1)）位说明

位	符号	说明	复位值
15:0	DATA	发送数据。该字段可传送 4 至 16 位的数据。	0
31:16	-	保留。应当只写入 0。	不适用

17.6.9 SPI 发送器控制寄存器

TXCTL 寄存器提供 SPI 单独访问控制信息的方法。这些位是 TXDATCTL 寄存器 （参见章节17.6.7）中名字相同位的另一种说法。除非数据稍后写入TXDAT寄存器，否则更改TXCTL中的位不会有任何效果。写入 TXDATCTL 的数据将覆盖 TXCTL 寄存器内容。

传输中如需更改控制信息，则应当使用TXDATCTL寄存器（参见章节17.6.7）而非TXDAT。然后，控制信息便可与数据一同写入器件。

表 199. SPI 发送器控制寄存器（TXCTL，地址 0x4005 8020 (SPI0)，0x4005 C020 (SPI1)）位说明

位	符号	说明	复位值
15:0	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
16	TX SSEL	发送从机选择。	0x0
19:17	-	保留。	0x0
20	EOT	发送结束。	0
21	EOF	帧结束。	0
22	RXIGNORE	接收忽略。	0
23	-	保留。读取的数值为未定义型，且只应写入零值。	不适用
27:24	FLEN	帧长度。	0x0
31:28	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

17.6.10 SPI 分频器寄存器

DIV 寄存器可确定主机模式下 SPI 所使用的时钟。

有关时钟的详细信息，请参见章节 17.7.3 “ 时钟和数据速率 ”。

表 200. SPI 分频器寄存器 （DIV，地址 0x4005 8024 (SPI0)， 0x4005 C024 (SPI1)）位说明

位	符号	说明	复位值
15:0	DIVVAL	速率分频器值。指定如何对 SPI 的 PCLK 分频，以产生主机模式下的 SPI 时钟速率。 DIVVAL 编码为 -1，以使 PCLK/1 的值为 0、 PCLK/2 的值为 1，直至可能达到的分母最大值 0xFFFF，即 PCLK/65536。	0
31:16	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

17.6.11 SPI 中断状态寄存器

INTSTAT 只读寄存器提供当前使能的中断标志视图。它可以简化软件对中断的处理。有关中断标志的详细说明，请参见表 193。

表 201. SPI 中断状态寄存器（INTSTAT，地址 0x4005 8028 (SPI0)，0x4005 C028 (SPI1)）位说明

位	符号	说明	复位值
0	RXRDY	接收器就绪标志。	0
1	TXRDY	发送器就绪标志。	1
2	RXOV	接收器溢出中断标志。	0
3	TXUR	发送器欠载中断标志。	0
4	SSA	从机选择置位。	0
5	SSD	从机选择解除置位。	0
31:6	-	保留。读取的数值为未定义型，且只应写入零值。	不适用

17.7 功能说明

17.7.1 工作模式：时钟和相位选择

SPI 接口通常可进行时钟相位和极性的配置。某些情况下，它们也称作编号 SPI 模式，如表 202中所述以及图34中所示。CPOL和CPHA可通过CFG寄存器（章节17.6.1）中的位配置。

表 202. SPI 模式汇总

CPOL	CPHA	SPI 模式	说明	SCK 静态	SCK 数据转变边沿	SCK 数据样本边沿
0	0	0	SPI 在帧的首次时钟跃迁时捕获串行数据（时钟从静态转变为其他状态）。数据在下列边沿上发生变化。	低电平	下降沿	上升沿
0	1	1	SPI 在帧的首次时钟跃迁时更改串行数据（时钟从静态转变为其他状态）。数据在下列边沿上被捕获。	低电平	上升沿	下降沿
1	0	2	与 SCK 反转时的模式 0 相同。	高电平	上升沿	下降沿
1	1	3	与 SCK 反转时的模式 1 相同。	高电平	下降沿	上升沿

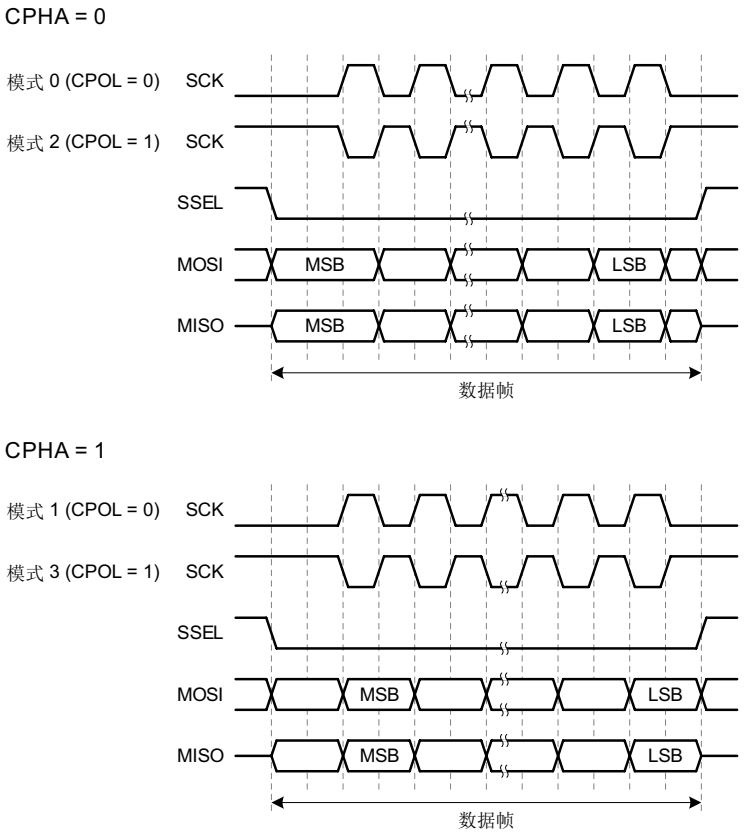


图 34. 基本 SPI 工作模式

17.7.2 帧延迟

可针对 SPI 帧指定数个延迟。包括：

- Pre_delay：在 SSEL 置位后、数据时钟开始前插入延迟
- Post_delay：在一个数据帧之后、 SSEL 置位前插入延迟
- Frame_delay：SSEL 未解除置位时，在两个数据帧之间插入延迟
- Transfer_delay：两次传输之间 SSEL 解除置位状态的最短持续时间

17.7.2.1 Pre_delay 和 Post_delay

Pre_delay 和 Post_delay 参见表 35 中的示例。Pre_delay 值控制 SSEL 置位和后续数据帧起点之间的时间长度。 Post_delay 控制数据帧末尾和 SSEL 解除置位之间的时间长度。

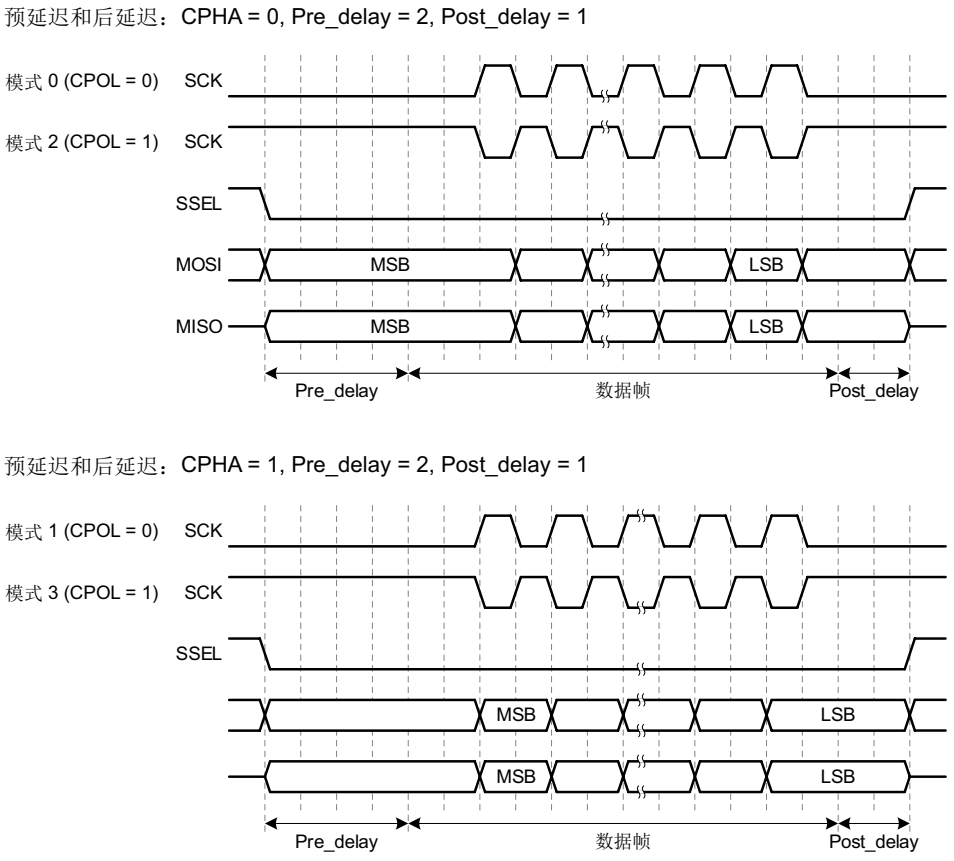


图 35. Pre_delay 和 Post_delay 时序

17.7.2.2 Frame_delay

Frame_delay 值控制的是每帧的结束时间。当 EOF 位为 1 时，该延迟会被插入。有关 Frame_delay 的说明，请参见表 36 中的示例。需要注意的是，帧边界仅在指定处存在。这是因为帧的长度大小是任意的，其中包含多次数据写操作。有关更多信息，请参见章节 17.7.5。

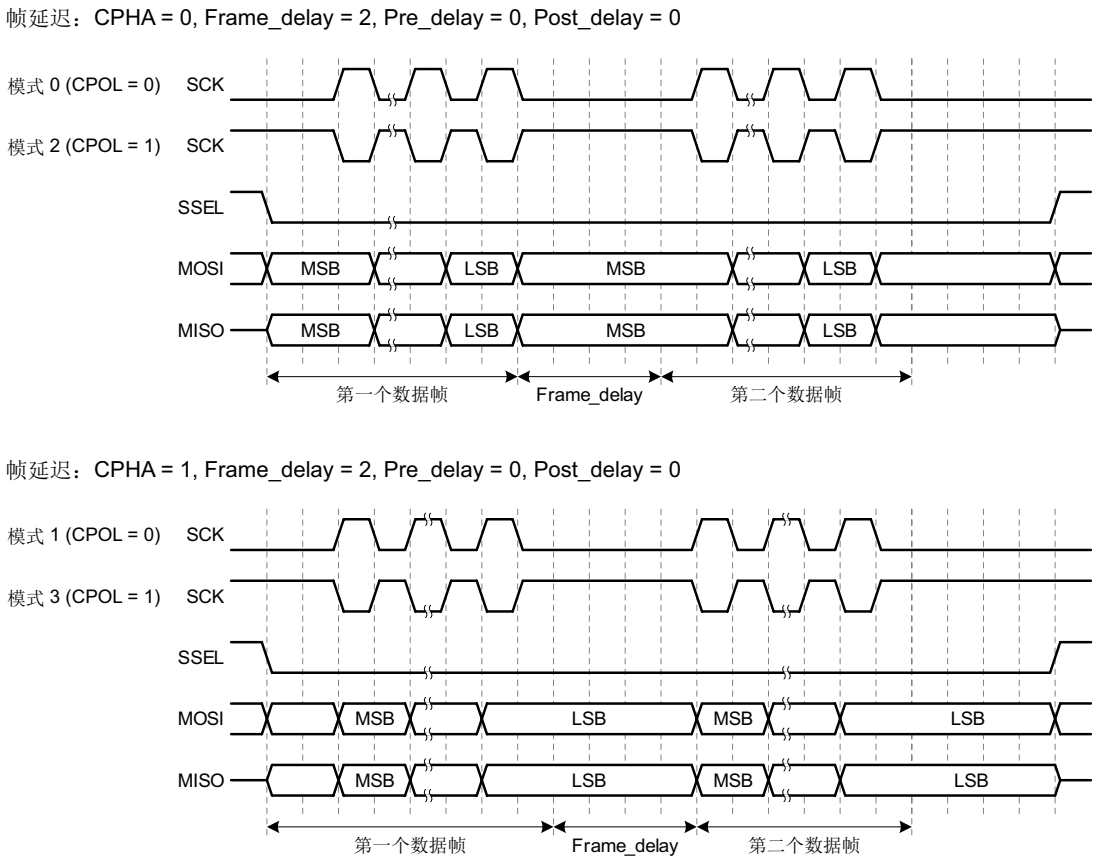


图 36. Frame_delay 时序

17.7.2.3 Transfer_delay

由于 EOT 位为 1，因此 Transfer_delay 值可控制 SSEL 在两次传输之间解除置位的最小时间长度。当 Transfer_delay 为 0 时， SSEL 可能被解除置位的最小时间为一个 SPI 时钟周期。 Transfer_delay 参见图 37 中的示例。

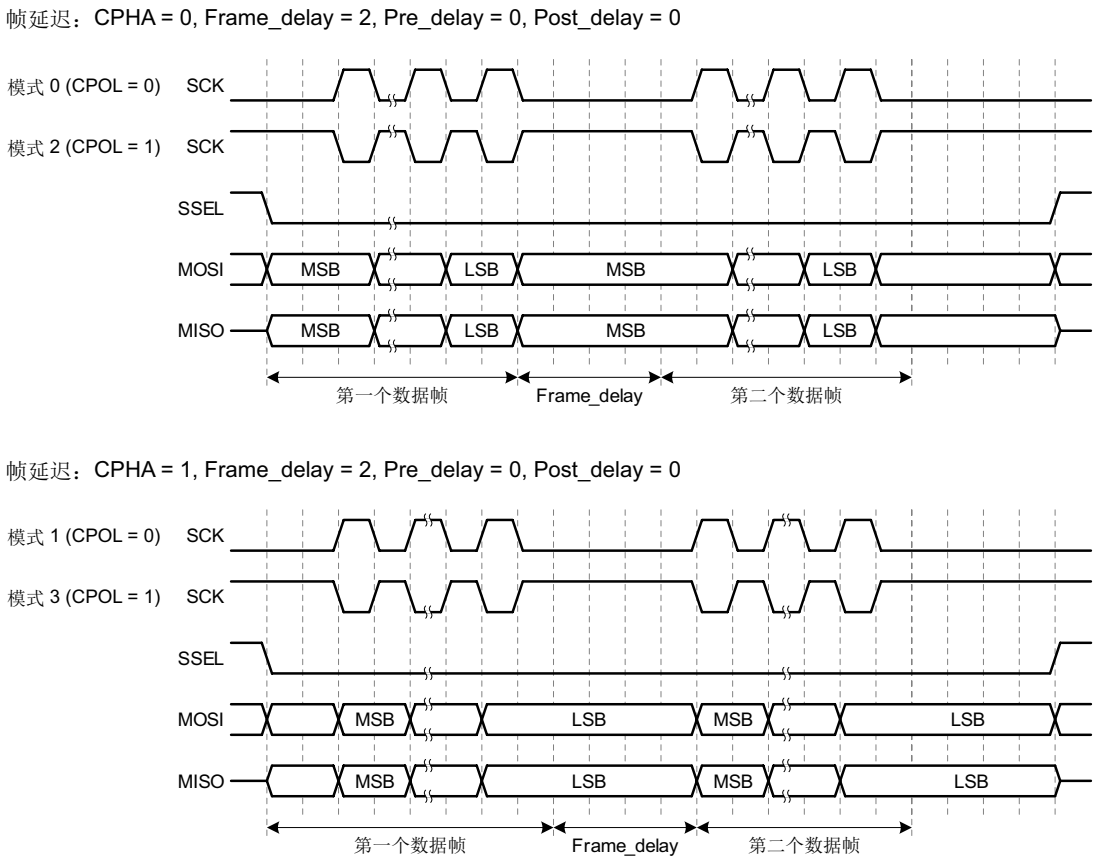


图 37. Transfer_delay 时序

17.7.3 时钟和数据速率

要使用 SPI，则需定义计时细节。这包括配置系统时钟和在 DIV 中选择时钟分频器值。参见图 32。

17.7.3.1 数据速率计算

SPI 接口设计为工作在异步于任何片内时钟的频率下，并且无需超频。

从机模式下，这表示直接使用外部主机的 SCK 运行发送和接收移位寄存器以及其他逻辑。

主机模式下， SPI 时钟分频器产生的 SPI 速率时钟直接用于输出 SCK。

SPI 时钟分频器是一个整数分频器。主机模式下的 SPI 可设置为运行在与选定的 PCLK 相同的速率，也可设置为运行在较低的整数分频速率。 SPI 速率为 PCLK_SPIn / DIVVAL。

从机模式下，时钟为 SCK 输入，不使用 SPI 时钟分频器。

17.7.4 从机选择

SPI 模块可在从机模式下提供一个从机选择输入，或者在主机模式下提供一个输出。 SSEL 可设置为普通极性 （低电平有效）或反转极性 （高电平有效）。寄存器中表示 SSEL 的位始终低电平有效。如果 SSEL 反转，则可通过信号离开 / 进入 SPI 模块完成。

从机模式下，连接至某个引脚并置位的 SSEL 将激活 SPI。主机模式下，连接某个引脚的 SSEL 将如同 SPI 寄存器中定义的那样被输出。

主机模式下，从机选择通过 TXSSELN 字段进行配置，出现在 TXCTL 和 TXDATCTL 寄存器中。从机模式下，SSEL 的状态以及 RXDAT 寄存器的 RXSSELN 字段接收数据将被一并保存。

17.7.5 数据长度大于 16 位

SPI 接口直接处理尺寸为 1 至 16 位的数据帧。以 16 位一组 （或更小）分割数据，便可处理更大尺寸的数据。例如，可将 24 位分为 16 位和 8 位两组，或两组 12 位数据，诸如此类。可以同样方式支持任意尺寸 （包括大于 32 位）的帧。

有关如何处理更大数据宽度的详情，将在一定程度上取决于其他 SPI 的配置选项。例如，如果试图在两帧之间置位从机选择，则必须在较大的帧被分割成多个部分的情况下抑制该操作。发送 24 位增量之间 SSEL 解除置位的两组 12 位数据将需要改变其他 12 位帧上的 EOF 位数值。

17.7.6 数据停止

如果 SCK 无法返回到静态，则在可将下一个数据帧的 MSB 驱动至 MOSI 之前，模式 0 和模式 2 可能发生主机数据传输停止的情况。在这种情况下，如果下一个数据暂时不可用，则数据的最终时钟沿之后会发生停止。

主机可能在这种情况下停止接收，即如果发送器不停机则接收器会溢出。在模式 0 和模式 2 中，若之前接收的数据未能在下一个数据到达之前被读取，则发生该情况。发生该停止的时间比发送器停止早一个时钟沿。

在模式 1 和模式 3 中，可能发生同样的接收器停机，但仅在接收数据的最终时钟沿之前发生。此外，模式 1 和模式 3 中将不会发生接收器停机，因为在原本停止传输的时刻之前即可完成数据的传输，从而无需停机。

停止可通过 STAT 寄存器中的停止状态标志反映，该标志表示当前 SPI 状态。

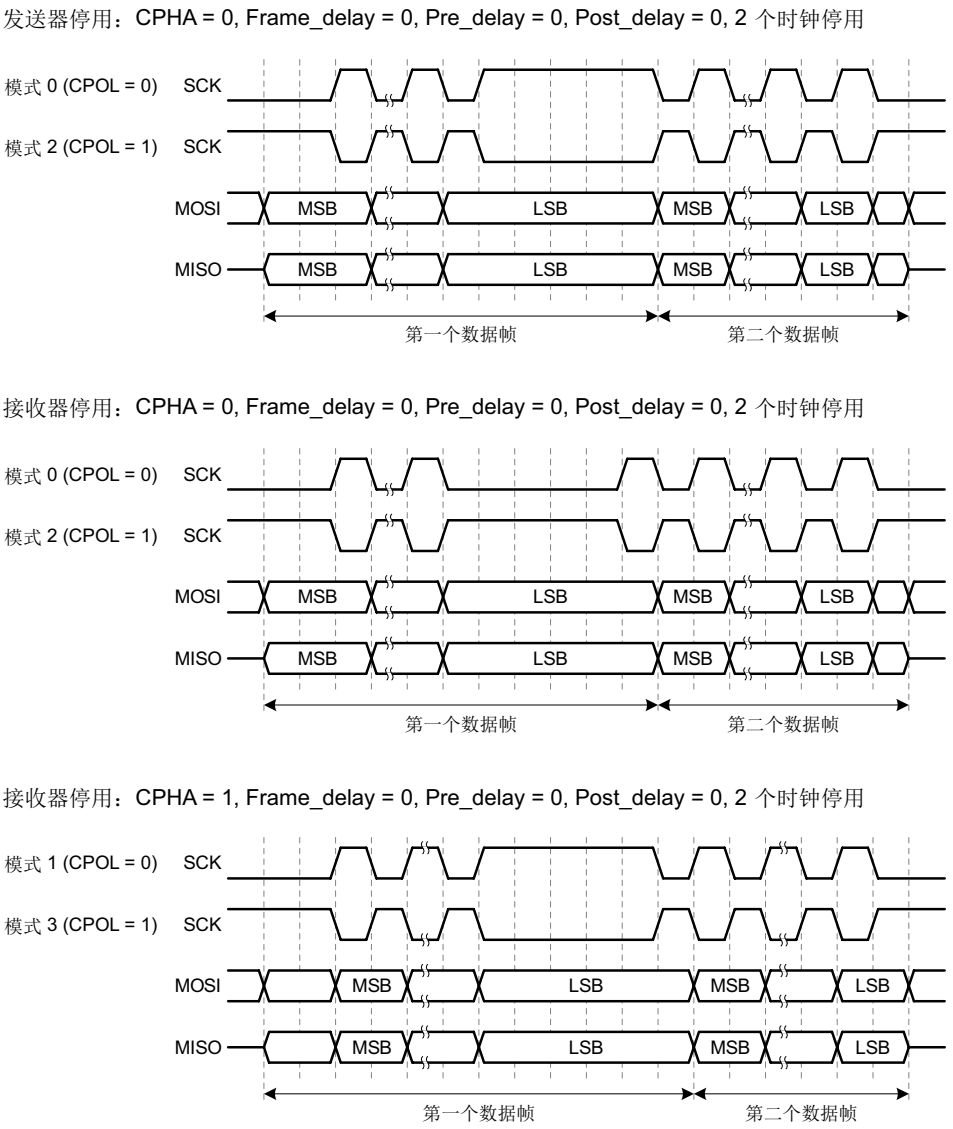


图 38. 数据停止示例

18.1 本章导读

所有 LPC800 器件都提供模拟比较器。

18.2 特性

- 可选外部输入既可用作比较器的正输入，也可用作比较器的负输入。
- 内部基准电压（0.9 V 带隙基准电压）既可用作比较器的正输入，也可用作比较器的负输入。
- 32 级阶梯电压既可用作比较器的正输入，也可用作比较器的负输入。
- 电压阶梯源的可选范围在电源引脚 V_{DD} 或 VDDCMP 引脚之间。
- 电压阶梯在不需要时可单独掉电。
- 中断功能。

18.3 基本配置

使用下列寄存器配置模拟比较器：

- 在 SYSAHBCLKCTRL 寄存器中，设置位 19（表 18）以使能寄存器接口时钟。
- 通过 PDRUNCFG 寄存器（表 37）可使能或禁用模拟比较器的电源。
- 使用 PRESETCTRL 寄存器（表 7）清零模拟比较器外设复位。
- 模拟比较器中断连接至 NVIC 的 11 号中断。
- 通过开关矩阵配置模拟比较器引脚功能。参见章节 18.4。

18.3.1 比较器输出连接至 SCT

可使用比较器输出功能 (ACMP_O) 启动或停止 SCT，而更为通常的是创建 SCT 事件。要创建 SCT 事件，则用下列方式连接 AMP_O：

1. 使用开关矩阵将 ACMP_O 连接至某个引脚。参见表 203。
2. 使用开关矩阵将任意一个 SCT 输入功能连接至同一引脚。参见表 107。

所选 SCT 输入随后便可监控 ACMP_O 功能。

18.4 引脚说明

模拟比较器基准电压、输入和输出通过开关矩阵分配给外部引脚。模拟比较器的输出可分配给封装上的任意非电源或接地引脚。比较器输入和基准电压为固定引脚功能，必须通过开关矩阵使能，并且只能分配给封装上的特定引脚。

有关如何将模拟比较器输出分配给 LPC800 封装上的任意引脚，请参见表 9.3.1 “ 将内部信号与封装引脚相连 ”。

有关如何使能模拟比较器输入和基准电压输入，请参见章节 9.3.2。

表 203. 模拟比较器引脚说明

功能	类型	引脚	说明	SWM 寄存器	参考
ACMP_I1	I	PIO0_0/ACMP_I1	比较器输入 1	PINENABLE0	章节 9.5.10
ACMP_I2	I	PIO0_0/ACMP_I2/CLKIN	比较器输入 2。禁用 PINENABLE0 寄存器的 CLKIN 功能。	PINENABLE0	章节 9.5.10
ACMP_O	O	任意	比较器输出	PINASSIGN8	章节 9.5.9
VDDCMP	I	PIO0_6/VDDCMP	32 级阶梯电压的外部基准电压源。	PINENABLE0	章节 9.5.10

18.5 简介

模拟比较器可将外部引脚上的电压电平与内部电压作比较。

该比较器有 4 个单独复用到各自正负输入的输入。多路复用器由比较寄存器 CTL（参见图 39 和表 205）控制。

多路复用器的输入 0 是可编程电压阶梯输出。

位 2:1 控制外部输入 ACMP_I[2:1]。

多路复用器的位 6 控制内部基准电压输入。

所有其他位都为保留位。

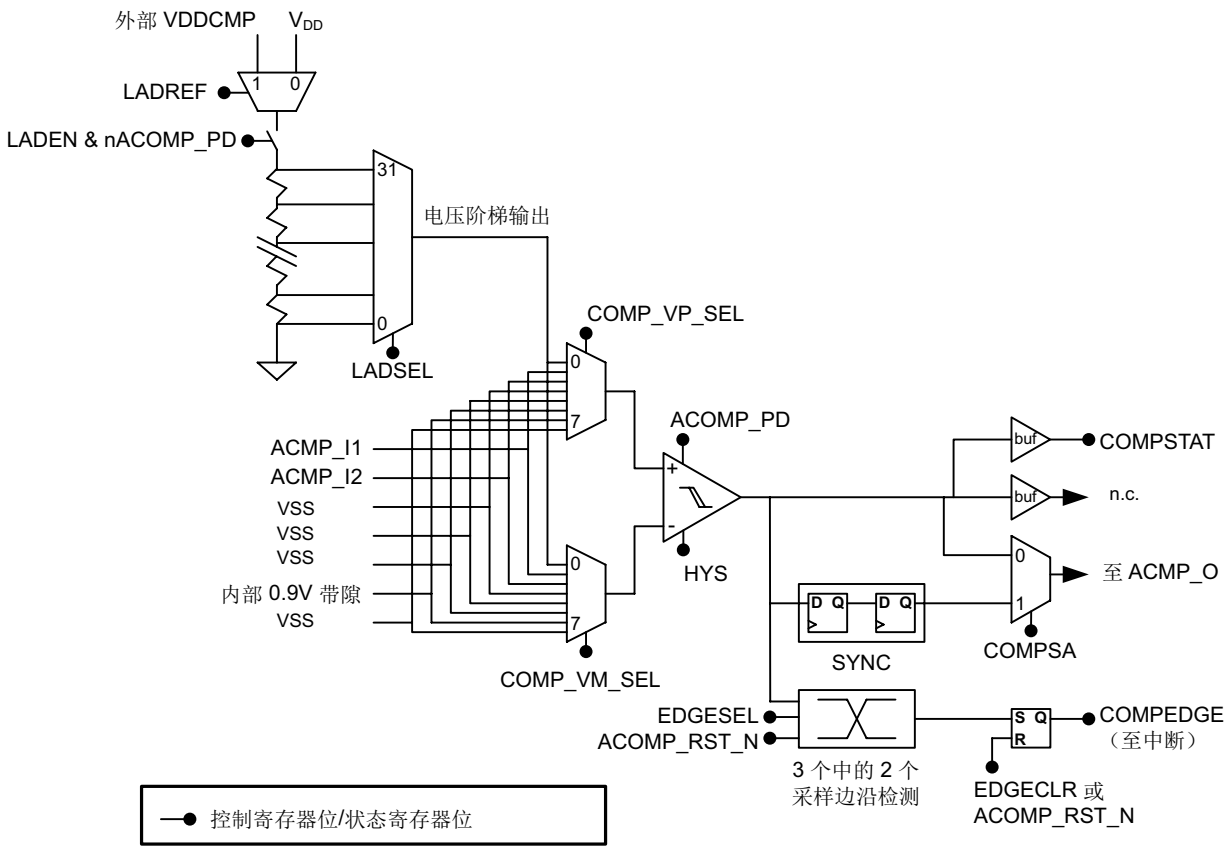


图 39. 比较器功能框图

18.5.1 基准电压

电压阶梯可使用来自 VDDCMP 引脚或 V_{DD} 引脚的两种基准电压源。电压阶梯在引脚电压和 V_{SS}（两者包括在内）之间选择 32 级中的某一级。VDDCMP 上的电压不应超过 V_{DD} 上的电压。

18.5.2 建立时间

电压阶梯上电后，需要一定的稳定工作时间才能进行精确比较。更短的建立时间适用于 LADSEL 值发生变化后以及其中某个电压源发生变化或两个电压源都发生变化的情况。软件可在多个读数产生相同结果之前通过重复读取比较器输出处理这些因素。

18.5.3 中断

中断输出来自该模块的边沿检测电路。上升沿、下降沿或这两个边沿都可设置 COMPEDGE 位，从而请求一个中断。软件将 1 写入 EDGECLR 时会清零 COMPEDGE 和中断请求。

18.5.4 比较器输出

比较器输出（取决于 COMPSA 位）可路由至某个外部引脚。COMPSA 为 0 且比较器中断被禁用时，如果无需对控制寄存器进行写操作，则可在总线时钟禁用（表 18 “ 系统时钟控制寄存器（SYSAHBCLKCTRL，地址 0x4004 8080）位说明 ”）的情况下使用比较器来降低功耗。

可通过比较器状态寄存位观察比较器输出的状态。

比较器输出可通过开关矩阵路由至 SCT，允许捕获电压交叉的时间和进行单向或双向交叉计数。参见章节 18.3.1 “ 比较器输出连接至 SCT”。

18.6 寄存器说明

表 204. 寄存器概述：模拟比较器（基址 0x4002 4000）

名称	访问类型	地址偏移	说明	复位值
CTRL	R/W	0x000	比较器控制寄存器	0
LAD	R/W	0x004	电压阶梯寄存器	0

18.6.1 比较器控制寄存器

该寄存器可使能比较器、配置中断、控制比较器两侧的输入多路复用器。表 205 中所有未显示的位均为保留位，并且应当以 0 写入。

表 205. 比较器控制寄存器（CTRL，地址 0x4002 4000）位说明

位	符号	值	说明	复位值
2:0	-		保留。以 0 写入。	0
4:3	EDGESEL		该字段可控制设置 COMPEDGE 位（下文中的位 23）的比较器输出边沿：	0
		0x0	下降沿	
		0x1	上升沿	
		0x2	上升沿和下降沿	
		0x3	上升沿和下降沿	
5	-		保留。以 0 写入。	0

表 205. 比较器控制寄存器 （CTRL，地址 0x4002 4000）位说明 *（续）*

位	符号	值	说明	复位值
6	COMPSA		比较器输出控制	0
		0	比较器输出被直接使用。	
		1	比较器输出与总线时钟同步以输出到其他模块。	
7	-		保留。以 0 写入。	0
10:8	COMP_VP_SEL		选择正电压输入	0
		0x0	电压阶梯输出	
		0x1	ACMP_I1	
		0x2	ACMP_I2	
		0x3	保留	
		0x4	保留	
		0x5	保留	
		0x6	内部基准电压 （带隙）	
		0x7	保留	
13:11	COMP_VM_SEL		选择负电压输入	0
		0x0	电压阶梯输出	
		0x1	ACMP_I1	
		0x2	ACMP_I2	
		0x3	保留	
		0x4	保留	
		0x5	保留	
		0x6	内部基准电压 （带隙）	
		0x7	保留	
19:14	-		保留。以 0 写入。	0
20	EDGECLR		中断清零位。要清零 COMPEDGE 位并进而取消中断请求，可通过先写入 1 再写入 0 的方式切换 EDGECLR 位。	0
21	COMPSTAT		比较器状态。该位反映的是比较器输出的状态。	0
22	-		保留。以 0 写入。	0
23	COMPEDGE		比较器边沿检测状态。	0
24	-		保留。以 0 写入。	0
26:25	HYS		控制比较器的迟滞。当比较器输出某种状态时，开关输出的正是该输出状态相反方向上所选信号之间的差值。	0
		0x0	未设置 （输出将以电压交叉进行开关）	
		0x1	5 mV	
		0x2	10 mV	
		0x3	20 mV	
31:27	-		保留	-

18.6.2 电压阶梯寄存器

该寄存器可使能和控制电压阶梯。由阶梯产生的这部分基准电压是可编程的，步长为 1，共 31 级。

表 206. 电压阶梯寄存器 （LAD，地址 0x4002 4004）位说明

位	符号	值	说明	复位值
0	LADEN		电压阶梯使能	0
5:1	LADSEL		电压阶梯值。基准电压 Vref 取决于下文中的 LADREF 位。 00000 = V_{SS} 00001 = 1 × Vref/31 00010 = 2 × Vref/31 ... 11111 = Vref	0
6	LADREF		选择电压阶梯的基准电压 Vref：	0
		0	电源引脚 V_{DD}	
		1	VDDCMP 引脚	
31:7	-		保留。	0

19.1 本章导读

所有 LPC800 器件均提供 CRC 引擎。

19.2 特性

- 支持三个通用多项式 CRC-CCITT、 CRC-16 和 CRC-32。
  - CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
  - CRC-16: $x^{16} + x^{15} + x^2 + 1$
  - CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- 针对输入数据及 CRC 和的位顺序取反和 1 的补码可编程设置。
- 可编程种子数设置。
- 可接受每次写操作的任意大小数据宽度：8 位、 16 位或 32 位。
  - 8 位写操作：1 周期操作
  - 16 位写操作：2 周期操作 （8 位 x 2 周期）
  - 32 位写操作：4 周期操作 （8 位 x 4 周期）

19.3 基本配置

在 SYSAHBCLKCTRL 寄存器 （表 18，位 13）中使能 CRC 引擎的时钟。

19.4 引脚说明

CRC 引擎无可配置引脚。

19.5 简介

带可编程多项式设置的循环冗余校验 (CRC) 生成器支持多个常用 CRC 标准。

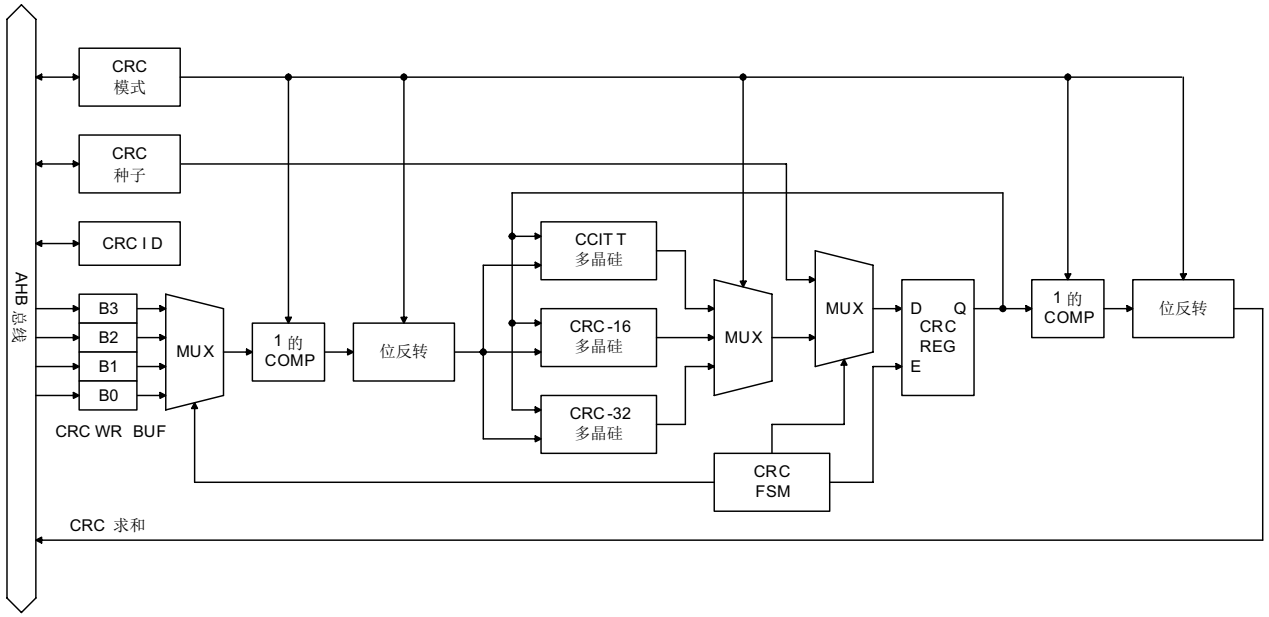


图 40. CRC 功能框图

19.6 寄存器说明

表 207. 寄存器概述：CRC 引擎 （基址 0x5000 0000）

名称	访问类型	地址偏移	说明	复位值	参考
MODE	R/W	0x000	CRC 模式寄存器	0x0000 0000	表 208
SEED	R/W	0x004	CRC 种子寄存器	0x0000 FFFF	表 209
SUM	RO	0x008	CRC 校验和寄存器	0x0000 FFFF	表 210
WR_DATA	WO	0x008	CRC 数据寄存器	-	表 211

19.6.1 CRC 模式寄存器

表 208. CRC 模式寄存器 （MODE，地址 0x5000 0000）位说明

位	符号	说明	复位值
1:0	CRC_POLY	CRC 多项式： 1X= CRC-32 多项式 01= CRC-16 多项式 00= CRC-CCITT 多项式	00
2	BIT_RVS_WR	数据位顺序： 1= 对 CRC_WR_DATA 进行位顺序取反 （每字节） 0= 不对 CRC_WR_DATA 进行位顺序取反 （每字节）	0
3	CMPL_WR	数据补码： 1= 对 CRC_WR_DATA 采用 1 的补码 0= 对 CRC_WR_DATA 不采用 1 的补码	0
4	BIT_RVS_SUM	CRC 和位顺序： 1= 对 CRC_SUM 进行位顺序取反 0= 不对 CRC_SUM 进行位顺序取反	0
5	CMPL_SUM	CRC 和补码： 1= 对 CRC_SUM 采用 1 的补码 0= 对 CRC_SUM 不采用 1 的补码	0
31:6	保留	读取时始终为 0	0x0000000

19.6.2 CRC 种子寄存器

表 209. CRC 种子寄存器 （SEED，地址 0x5000 0004）位说明

位	符号	说明	复位值
31:0	CRC_SEED	对该寄存器的写访问将按选定位顺序和 1 的补码预处理将 CRC 种子值加载到 CRC_SUM 寄存器。 **注：**对该寄存器的写访问将使正在进行的 CRC 计算无效。	0x0000 FFFF

19.6.3 CRC 校验和寄存器

该寄存器是只读寄存器，包含最近一次校验和。在结果有效且校验和运算完成前，对该寄存器的读请求会被有限个等待状态自动延迟。

表 210. CRC 校验和寄存器 （SUM，地址 0x5000 0008）位说明

位	符号	说明	复位值
31:0	CRC_SUM	可通过该寄存器按选定位顺序和1的补码后处理读取最近一次的 CRC 和。	0x0000 FFFF

19.6.4 CRC 数据寄存器

该寄存器是只写寄存器，包含将计算 CRC 和的数据块。

表 211. CRC 数据寄存器 （WR_DATA，地址 0x5000 0008）位说明

位	符号	说明	复位值
31:0	CRC_WR_DATA	写入该寄存器的数据将被用于按选定位顺序和 1 的补码预处理进行 CRC 计算。允许 8 位、 16 位或 32 位的任意大小的写操作，并接受连续交换。	-

19.7 功能说明

下列章节介绍的是适用于每个受支持 CRC 标准的寄存器设置：

19.7.1 CRC-CCITT 设置

多项式 = $x^{16} + x^{12} + x^5 + 1$

种子值 = 0xFFFF

对数据输入进行位顺序取反：否

对数据输入采用 1 的补码：否

对 CRC 和进行位顺序取反：否

对 CRC 和采用 1 的补码：否

CRC_MODE = 0x0000 0000

CRC_SEED = 0x0000 FFFF

19.7.2 CRC-16 设置

多项式 = $x^{16} + x^{15} + x^2 + 1$

种子值 = 0x0000

对数据输入进行位顺序取反：是

对数据输入采用 1 的补码：否

对 CRC 和进行位顺序取反：是

对 CRC 和采用 1 的补码：否

CRC_MODE = 0x0000 0015

CRC_SEED = 0x0000 0000

19.7.3 CRC-32 设置

多项式 = $x^{32}+ x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

种子值 = 0xFFFF FFFF

对数据输入进行位顺序取反：是

对数据输入采用 1 的补码：否

对 CRC 和进行位顺序取反：是

对 CRC 和采用 1 的补码：是

CRC_MODE = 0x0000 0036

CRC_SEED = 0xFFFF FFFF

20.1 本章导读

所有 LPC800 器件上的闪存控制器都是一样的。

20.2 特性

- 可控制闪存访问时间。
- 提供用于生成闪存签名的寄存器。

20.3 简介

可访问闪存控制器进行闪存等待状态的编程和生成闪存签名。

20.4 寄存器说明

表 212. 寄存器概述：FMC （基址 0x4004 0000）

名称	访问类型	地址偏移	说明	复位值	参考
FLASHCFG	R/W	0x010	闪存配置寄存器	<tbd>	表 213
FMSSTART	R/W	0x020	签名起始地址寄存器	0	表 214
FMSSTOP	R/W	0x024	签名停止地址寄存器	0	表 215
FMSW0	R	0x02C	签名字	-	表 216

20.4.1 闪存配置寄存器

对 FLASHCFG 寄存器进行写操作可配置与系统频率无关的闪存访问。

注：使用 Power API 时，不要在效率模式、低电流模式或性能模式下改变等待状态。

表 213. 闪存配置寄存器 （FLASHCFG，地址 0x4004 0010）位说明

位	符号	值	说明	复位值
1:0	FLASHTIM		闪存访问时间。 FLASHTIM +1 与用于闪存访问的系统时钟数相等。	0x1
		0x0	1 个系统时钟的闪存访问时间。	
		0x1	2 个系统时钟的闪存访问时间。	
		0x2	保留。	
		0x3	保留。	
31:2	-	-	保留。**用户软件不得更改这些位的数值。位 31:2 必须按照读取时的值写回。**	-

20.4.2 闪存签名起始地址寄存器

表 214. 闪存模块签名起始寄存器 (FMSSTART - 0x4004 0020) 位说明

位	符号	说明	复位值
16:0	启动	签名生成起始地址 （对应于 AHB 字节地址位 [20:4]）。	0
31:17	-	保留，用户软件不应将 1 写入保留位。未定义从保留位读取的值。	不适用

20.4.3 闪存签名结束地址寄存器

表 215. 闪存模块签名结束寄存器 (FMSSTOP - 0x4004 0024) 位说明

位	符号	值	说明	复位值
16:0	STOPA		签名生成结束地址（STOPA 指定字包含在地址范围内）。地址的单位为存储器字，而非字节。	0
30:17	-		保留，用户软件不应将 1 写入保留位。未定义从保留位读取的值。	0
31	STRTBIST		将 1 写入该位时，会开始签名生成。签名生成结束后，该位会自动清零。	0

20.4.4 闪存签名生成结果寄存器

签名生成结果寄存器可返回内置生成器生成的闪存签名。

生成的闪存签名可用于验证闪存内容。生成的签名可与预期签名作比较，从而节省时间和代码空间。生成签名的方法如章节 20.5.1 中所述。

表 216. FMSW0 寄存器位说明 （FMSW0，地址：0x4004 002C）

位	符号	说明	复位值
31:0	SIG	32 位签名。	-

20.5 功能说明

20.5.1 闪存签名生成

闪存模块包含内置的签名生成器。该生成器可从一系列闪存中生成 32 位的签名。典型的用途是根据计算签名验证闪存的内容 （例如编程期间）。

用于生成签名的地址范围必须在闪字边界上对齐，例如：32 位边界。一旦开始，签名生成将独立完成。在签名生成过程中，不允许出于其它目的访问闪存，而且试图读取会导致在签名生成完成前置位等待状态。签名生成期间，可执行闪存外部代码 （例如内部 RAM）。如果中断向量表被重新映射到闪存外的其它内存中，这也可包括中断服务。启动签名生成的代码也应置于闪存外。

20.5.1.1 签名生成地址和控制寄存器

这些寄存器可控制自动签名生成。可生成一个签名将其用于闪存内容的任何部分。将起始地址写入起始地址寄存器 (FMSSTART) 以及将停止地址写入停止地址寄存器 (FMSSTOP) 可界定生成签名的地址范围。起始和结束地址必须与 32 位边界对齐。

签名生成通过设置 FMSSTOP 寄存器中的 STRTBIST 位开始。在单次写操作中，设置 STRTBIST 位通常与签名停止地址相结合。

表 214 和表 215 分别显示的是 FMSSTART 和 FMSSTOP 寄存器中的位分配。

20.5.1.2 签名生成

可生成一个签名将其用于闪存内容的任何部分。将起始地址写入 FMSSTART 寄存器以及将停止地址写入 FMSSTOP 寄存器可界定生成签名的地址范围。

将 1 写入 FMSSTOP 寄存器中的 SIG_START 位可开始签名生成。开始签名生成通常与定义停止地址相结合，后者在同一寄存器的 STOP 位中完成。

签名生成占用的时间与用于生成签名的地址范围成正比。为生成签名而对闪存进行的读操作使用的是自定时读取机制，它不依赖任何可配置的闪存定时设置。安全的签名生成持续时间估计为：

持续时间 = int((60 / tcy) + 3) x (FMSSTOP - FMSSTART + 1)

通过软件触发签名生成时，持续时间以 AHB 时钟周期为单位，tcy 是一个 AHB 时钟的时间（单位：ns）。软件可通过轮询 FMSTAT 中的 SIG_DONE 位来确定签名完成的时间。

签名生成后，可从寄存器 FMSW0 读取 32 位的签名。该 32 位签名反映的是从闪存和闪存奇偶校验位中读取的正确数据以及检查位的值。

20.5.1.3 内容验证

从寄存器 FMSW0 读取的签名必须与参考签名相等。下列伪代码给出了推算参考签名的算法：

```
sign = 0
FOR address = FMSSTART.START to FMSSTOP.STOPA
{
        FOR i = 0 TO 30
        {
                nextSign[i] = f_Q[address][i] XOR sign[i + 1]
        }
        nextSign[31] = f_Q[address][31] XOR sign[0] XOR sign[10] XOR sign[30] XOR sign[31]
        sign = nextSign
}
signature32 = sign
```

21.1 本章导读

所有器件的引导加载程序都是一样的。引导 ROM 的实现随芯片版本不同而有所不同。参见章节 21.3.1。

21.2 特性

- 8 kB 片内引导 ROM
- 包含具有在系统编程 (ISP) 功能的引导加载程序和下列 API：
  - 闪存的在应用编程 (IAP)
  - 用于优化功耗和系统性能的电源配置
  - USART 驱动器
  - I2C 驱动器

21.3 基本配置

默认情况下 ROM 时钟为使能。使用 ROM 无需配置。

21.3.1 引导加载程序版本

LPC800 的引导加载程序已更新为最新芯片版本。通过 ISP 命令 “ 读取引导代码版本 (Read Boot code version)”（参见章节 22.5.1.12）或根据器件标识确定引导加载程序版本。

表 217. 引导加载程序版本

引导加载程序版本	标识	API	说明
v13.1（初始版本）	1A	ISP/IAP	与规格有所不同的下列情况适用： • IAP 擦除页命令仅允许单页擦除。起始页参数必须与结束页参数相同。参见表 252。 • 不返回 ISP 命令 C （从 RAM 写入闪存）中的代码 SECTOR_NOT_PREPARED_FORWRITE_OPERATION。参见表 231。 • ISP 模式使用 USART0 接口进行通信。如果应用中使用了 USART0，则在使用 IAP 命令 57 （重新调用 ISP）前复位 USART0 （参见表 7）。参见表 250。
		UART	与规格有所不同的下列情况适用： • 不支持 UART 同步模式。 • 故障时，API 功能 uart_put_line 和 uart_get_line 不返回中断。参见表 285 和表 286。 • UART API 返回代码编号为 0x0007 0001 至 0x0007 0005。
		I2C	无变化。
		电源配置	无变化。

表 217. 引导加载程序版本 *（续）*

引导加载程序版本	标识	API	说明
v13.2	2A	ISP/IAP	相比于 v13.1，下列更新适用： • IAP 擦除页命令允许多页擦除。在 IAP 擦除页命令中，允许将任意不大于结束页页码的起始页页码作为起始页。参见表 252。 • 返回 ISP 命令 C（从 RAM 写入闪存）中的代码 SECTOR_NOT_PREPARED_FOR WRITE_OPERATION。参见表 231。 • IAP 命令 57 （重新调用 ISP）可在不首先复位 USART0 的情况下调用。 • 添加了 ISP 命令 S （读取 CRC 校验和）。参见表 240。
		UART	相比于 v13.1，下列更新适用： • 支持 UART 同步模式。 • 故障时，API 功能 uart_put_line 和 uart_get_line 返回中断。参见表 285 和表 286。 • UART API 返回代码编号为 0x0008 0001 至 0x0008 0005。参见表 288。
		I2C	无变化。
		电源配置	无变化。

21.4 引脚说明

复位时，如果将 PIO0_1 引脚拉至低电平，则该器件会进入 ISP 模式，并且 ISP 命令处理程序会启动。在 ISP 模式下， USART0 模块上的引脚 PIO0_0 会连接至功能 U0_RXD，引脚 PIO0_4 会连接至功能 U0_TXD。

21.5 简介

21.5.1 引导加载程序

引导加载程序控制复位后的初始操作，并提供通过 USART 完成闪存编程的方法。这可能是清空设备的初始编程，已编程设备的擦除和重编程，或者是通过运行系统中的应用程序对闪存编程。

每当器件上电或复位时，引导加载程序代码即被执行一次。引导加载程序还可执行 ISP 命令处理程序或用户应用程序代码。复位后引脚 PIO0_1 的低电平被视为通过 USART 启动 ISP 命令处理程序的外部硬件请求。

有关引导过程的详细信息，请参见章节 21.6.2 “ 引导过程 ”。

注：引导加载程序不使用 SRAM 位置 0x1000 0000 至 0x1000 0050，并且该区域的存储器内容在复位时会予以保留。当器件进入掉电或深度掉电模式时， SRAM 存储器的内容不会予以保留。

假设在 $\overline{\text{RESET}}$ 引脚上产生上升沿时，电源引脚在标称电平上，那么在最多 <tbd>3 ms 后，会对 PIO0_1 引脚信号进行采样并确定是继续用户代码还是产生 ISP 处理程序。如果对 PIO0_1 引脚采样的结果为低电平，同时看门狗溢出标志被设置，那么外部硬件启动 ISP 命令处理程序的请求将被忽略。如果不存在执行 ISP 命令处理程序的请求 （PIO0_1 在复位后的采样结果为高电平），则会搜寻有效的用户程序。如果找到了有效的用户程序，则会将执行控制转交给它。如果未找到有效的用户程序，则会调用自动波特率例程。

注：通过对闪存地址 0x0000 02FC 进行编程可禁用引脚 PIO0_1 上的采样 （参见章节 22.4.3 “ 代码读保护 (CRP)”）。

21.5.2 基于 ROM 的 API

一旦器件完成引导，用户就可访问引导 ROM 内的某些 API，进而访问闪存、优化功耗并操作 USART 和 I2C 外设。

引导 ROM API 的结构如图 41 所示。

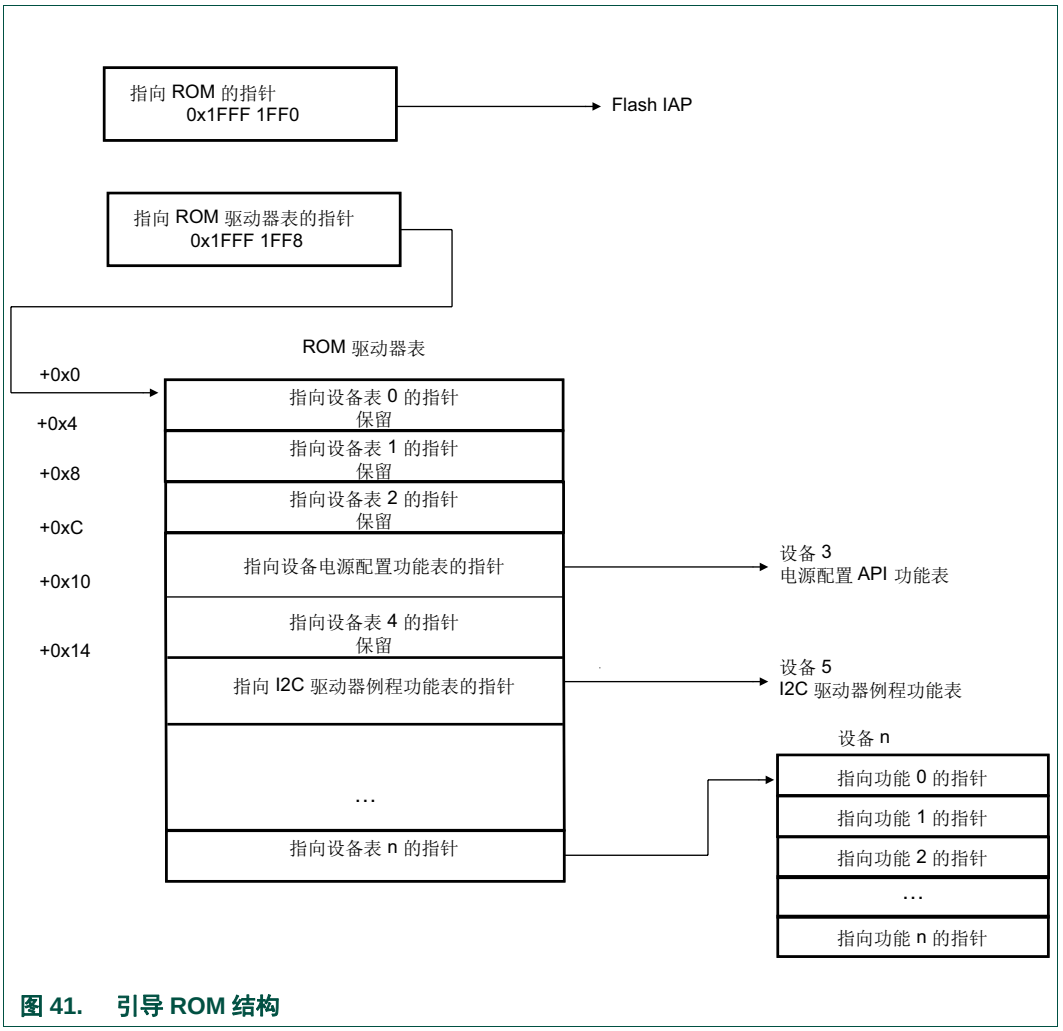


图 41. 引导 ROM 结构

表 218. API 调用

API	说明	参考
闪存 IAP	闪存在应用编程	表 242
电源配置 API	配置系统时钟和功耗	表 255
I2C 驱动器	I2C ROM 驱动器	表 258
UART 驱动器	USART ROM 驱动器	表 279

21.6 功能说明

21.6.1 复位后的存储器映射

引导模块的大小为 8 kB。引导模块在存储器区域中的起始地址为 0x1FFF 0000。引导加载程序设计为从该存储器区域运行，但 ISP 和 IAP 软件都会占用部分片内 RAM。本章后续部分描述了 RAM 的使用情况。复位后，驻留在片内闪存引导模块中的中断向量也会变为有效，也就是说，引导模块底部的 512 个字节也将出现在起始地址为 0x0000 0000 的存储器区域中。

21.6.2 引导过程

引导过程中，引导加载程序会检查闪存中是否存在有效用户代码。有效用户代码的判定标准如下：

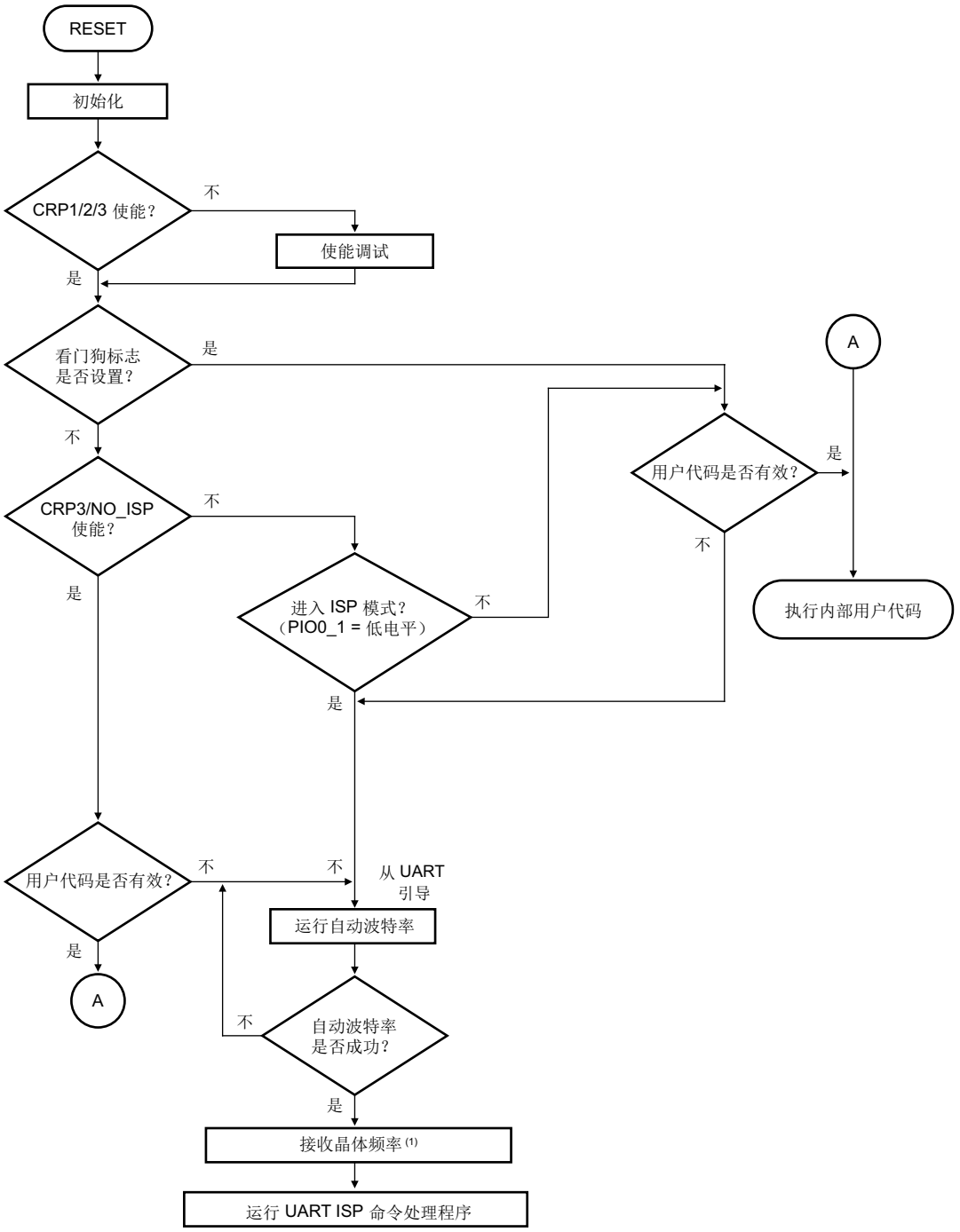
保留的 Cortex-M0+ 异常向量位置 7（向量表中的偏移量 0x0000 001C）应包含表中条目 0 到 6 的校验和的二进制补码。这将使表中头 8 个条目的校验和为 0。引导加载程序代码可求闪存扇区 0 的前 8 个位置的校验和。如果结果为 0，则执行控制会被转交到用户代码。

如果签名无效，则自动波特率例程会通过串行端口 USART0 与主机进行同步。主机应发送一个 “? ”(0x3F) 作为同步字符并等待响应。主机侧的串行端口设置应当是 8 个数据位、 1 个停止位以及无奇偶校验。自动波特率例程按其自身的频率（12 MHz IRC 频率）测量所接收同步字符的位时间，并对串行端口的波特率发生器进行编程。它还向主机发送 ASCII 字符串 ("Synchronized<CR><LF>")。作为响应，主机应当发送同样的字符串 ("Synchronized<CR><LF>")。

引导加载程序自动波特率例程可检查接收到的字符以验证同步。如果同步得到验证，则 "OK<CR><LF>" 字符串会被发送到主机。主机应当通过发送器件运行所使用的晶体频率（单位 kHz）进行响应。引导加载程序代码的向后兼容性要求作出响应，该响应在 LPC800 上被忽略。引导加载程序可配置该器件，使其在 12 MHz IRC 频率下运行。

接收到晶体频率响应后，会初始化器件并调用 ISP 命令处理程序。出于安全考虑，在执行可导致闪存擦除 / 写入操作的命令以及 “ 运行 (Go)” 命令之前必需使用 “ 解锁 (Unlock)” 命令。其余命令无需解锁命令即可直接执行。每个 ISP 会话都要求执行一次解锁命令。解锁命令在表 225 “UART ISP“ 解锁 ” 命令 ” 中作了说明。

21.6.3 引导过程流程图



(1) 该步骤包括在内是为了满足向后兼容性的要求，并且引导加载程序会忽略该响应。

图 42. 引导过程流程图

22.1 本章导读

有关各种闪存配置，请参见表 219。

表 219. LPC800 闪存配置

型号	闪存
LPC810M021FN8	4 kB
LPC811M001FDH16	8 kB
LPC812M101FDH16	16 kB
LPC812M101FD20	16 kB
LPC812M101FDH20	16 kB

22.2 特性

- 在系统编程：在系统编程 (ISP) 使用引导加载程序软件和 UART 串行端口对片内闪存进行编程或重新编程。
- 在应用编程：在应用编程 (IAP) 按照终端用户应用程序代码对片内闪存进行擦除及写入操作。
- 器件位于终端用户板内时，可使用 ISP 和 IAP。
- 支持闪存页的写入和擦除操作。

22.3 引脚说明

复位时，如果将 PIO0_1 引脚拉至低电平，则该器件会进入 ISP 模式，并且 ISP 命令处理程序会启动。在 ISP 模式下， USART0 模块上的引脚 PIO0_0 会连接至功能 U0_RXD，引脚 PIO0_4 会连接至功能 U0_TXD。

22.4 简介

22.4.1 闪存配置

大部分 IAP 和 ISP 命令在扇区执行并且有指定的扇区号。另外还支持页擦除命令。下表给出了页号、扇区号和存储器地址间的对应关系。

扇区大小为 1 kB，页大小为 64 字节。一个扇区包含 16 页。

表 220. LPC800 闪存配置

扇区号	扇区大小 [kB]	页号	地址范围	4 kB 闪存	8 kB 闪存	16 kB 闪存
0	1	0 -15	0x0000 0000 - 0x0000 03FF	是	是	是
1	1	16 - 31	0x0000 0400 - 0x0000 07FF	是	是	是
2	1	32 - 47	0x0000 0800 - 0x0000 0BFF	是	是	是
3	1	48 - 63	0x0000 0C00 - 0x0000 0FFF	是	是	是

表 220. LPC800 闪存配置 *（续）*

扇区号	扇区大小 [kB]	页号	地址范围	4 kB 闪存	8 kB 闪存	16 kB 闪存
4	1	64 - 79	0x0000 1000 - 0x0000 13FF	-	是	是
5	1	80 - 95	0x0000 1400 - 0x0000 17FF	-	是	是
6	1	96 - 111	0x0000 1800 - 0x0000 1BFF	-	是	是
7	1	112 - 127	0x0000 1C00 - 0x0000 1FFF	-	是	是
8	1	128 - 143	0x0000 2000 - 0x0000 23FF	-	-	是
9	1	144 - 159	0x0000 2400 - 0x0000 27FF		-	是
10	1	160 - 175	0x0000 2800 - 0x0000 2BFF		-	是
11	1	176 - 191	0x0000 2C00 - 0x0000 2FFF		-	是
12	1	192 - 207	0x0000 3000 - 0x0000 33FF		-	是
13	1	208 - 223	0x0000 3400 - 0x0000 37FF		-	是
14	1	224 - 239	0x0000 3800 - 0x0000 3BFF		-	是
15	1	240 - 255	0x0000 3C00 - 0x0000 3FFF		-	是

22.4.2 闪存内容保护机制

该器件集成了支持纠错代码 (ECC) 功能的闪存。使用纠错模块有两个目的：

ECC 首先将读取自存储器的数据字解码成输出数据字。然后，ECC 对要写入存储器的数据字进行编码。纠错功能由汉明码单位纠错构成。

ECC 的工作方式对运行中的应用程序而言是透明的。ECC 内容本身存储在闪存中，用户代码无法访问，既不能对其执行读操作也不能对其执行写操作。6 位 ECC 对应用户可访问闪存的每个连续 32 位。因此，地址范围为 0x0000 0000 至 0x0000 0003 的闪存字节受第一个 6 位 ECC 字节保护，而地址范围为 0x0000 0004 至 0x0000 0007 的闪存字节则受第二个 6 位 ECC 字节的保护，以此类推。

无论何时，只要 CPU 请求对用户可访问闪存进行读操作，就会同时评估包含特定存储器位置信息的 32 位原始数据以及与之匹配的 ECC 字节。如果 ECC 机制在获取的数据中检测到单个错误，则在数据提交 CPU 处理之前会进行纠错。当请求对用户可访问闪存进行写操作时，随用户指定内容一同写入的还有存储在 ECC 存储器中与之匹配的 ECC 计算值。

擦除某个闪存扇区时，相应的 ECC 位也会被一并擦除。一旦写入 6 位 ECC，除非先将其擦除，否则无法更新。因此，要使所实施的 ECC 机制正常工作，数据就必须以每组 4 字节（或 4 的倍数）的方式写入闪存中，并根据上文所述方式排列。

22.4.3 代码读保护 (CRP)

代码读保护是一种机制，它允许用户在系统中通过使能不同的安全级别来限制对片内闪存的访问和 ISP 的使用。需要时，通过对 0x0000 02FC 处闪存地址中的特定模式进行编程可调用 CRP。 IAP 命令不受代码读保护的影响。

注意事项：任何 CRP 更改都仅在设备经过一个电源周期后才变为有效。

表 221. 代码读保护选项

名称	0x0000 02FC 中编程的模式	说明
NO_ISP	0x4E69 7370	防止对 PIO0_1 引脚进行采样而进入 ISP 模式。 PIO0_1 引脚可用作其他用途。
CRP1	0x12345678	通过 SWD 引脚访问芯片被禁用。该模式允许使用下列 ISP 命令和限制更新部分闪存： • “ 写入 RAM” 命令不能访问 0x1000 0300 以下的 RAM。访问 0x1000 0200 以下地址被禁用。 • “ 从 RAM 复制到闪存 ” 命令不能写入扇区 0。 • 只有选中所有扇区进行擦除，“ 擦除 ” 命令才能擦除扇区 0。 • “ 比较 ” 命令被禁用。 • “ 读存储器 ” 命令被禁用。 该模式在要求 CRP 且需要更新闪存字段但不能擦除所有扇区时有用。由于在闪存部分更新的情况下比较命令会被禁用，因此辅助加载程序应执行校验和机制来验证闪存的完整性。
CRP2	0x87654321	通过 SWD 引脚访问芯片被禁用。下列 ISP 命令被禁用： • “ 读存储器 ” • “ 写入 RAM” • “ 运行 ” • “ 从 RAM 复制到闪存 ” • “ 比较 ” 使能 CRP2 时， ISP 擦除命令仅允许擦除所有用户扇区的内容。
CRP3	0x43218765	通过 SWD 引脚访问芯片被禁用。如果闪存扇区 0 中存在有效用户代码，则将 PIO0_1 拉至低电平会禁用 ISP 输入。 该模式可使用 PIO0_1 引脚有效禁用 ISP 覆盖。用户的应用程序可决定是通过调用 IAP 还是通过 UART 重新调用 ISP 命令来更新闪存。 **注意：如果选择了 CRP3，则无法对设备作进一步的出厂测试。**

表 222. 代码读保护硬件 / 软件的相互作用

CRP 选项	用户代码是否有效	复位时的PIO0_1 引脚电平	SWD 是否使能	器件是否进入 ISP 模式	ISP模式下是否更新部分闪存
无	否	x	是	是	是
无	是	高电平	是	否	不适用
无	是	低电平	是	是	是
CRP1	是	高电平	否	否	不适用
CRP1	是	低电平	否	是	是
CRP2	是	高电平	否	否	不适用
CRP2	是	低电平	否	是	否
CRP3	是	x	否	否	不适用
CRP1	否	x	否	是	是
CRP2	否	x	否	是	否
CRP3	否	x	否	是	否

表 223. 不同 CRP 级别允许使用的 ISP 命令

ISP 命令	CRP1	CRP2	CRP3 （ISP 模式下不允许进入）
解锁	是	是	不适用
设置波特率	是	是	不适用
应答	是	是	不适用

表 223. 不同 CRP 级别允许使用的 ISP 命令

ISP 命令	CRP1	CRP2	CRP3（ISP 模式下不允许进入）
写入 RAM	是；仅限于 0x1000 0300 以上	否	不适用
读存储器	否	否	不适用
准备写操作扇区	是	是	不适用
从 RAM 复制到闪存	是；不到扇区 0	否	不适用
运行	否	否	不适用
擦除扇区	是；仅在擦除所有扇区时才能擦除扇区 0。	是；仅限于所有扇区	不适用
空白检查扇区	否	否	不适用
读取器件 ID	是	是	不适用
读取引导代码版本	是	是	不适用
比较	否	否	不适用
读取 UID	是	是	不适用

如果使能某个 CRP 模式且通过 ISP 允许访问芯片，则不受支持或受限的 ISP 命令将由返回代码 CODE_READ_PROTECTION_ENABLED 终止。

22.4.3.1 ISP 输入保护

除这三种 CRP 模式外，用户还可通过防止对 PIO0_1 引脚采样而进入 ISP 模式，从而释放 PIO0_1引脚，使其用作它用。这被称为NO_ISP模式。可通过对地址0x0000 02FC处的模式 0x4E69 7370 进行编程进入 NO_ISP 模式。

22.5 API 说明

22.5.1 UART ISP 命令

下列命令由 ISP 命令处理程序接受。每条命令都支持详细的状态码。接收到未定义的命令时，命令处理程序会发送返回码 INVALID_COMMAND。命令和返回码都采用 ASCII 格式。

只有在接收到的 ISP 命令执行完毕且主机能发出新 ISP 命令时， ISP 命令处理程序才会发送 CMD_SUCCESS。“ 设置波特率 ”、“ 写入 RAM”、“ 读存储器 ” 和 “ 运行 ” 这几项命令不遵守此规则。

表 224. UART ISP 命令汇总

ISP 命令	用法	说明见：
解锁	U < 解锁代码 >	表 225
设置波特率	B < 波特率 > < 停止位 >	表 226
应答	A < 设置 >	表 227
写入 RAM	W < 起始地址 > < 字节数 >	表 228
读存储器	R < 地址 > < 字节数 >	表 229
准备写操作扇区	P < 起始扇区号 > < 结束扇区号 >	表 230
从 RAM 复制到闪存	C < 闪存地址 > <RAM 地址 > < 字节数 >	表 231
运行	G < 地址 > < 模式 >	表 232
擦除扇区	E < 起始扇区号 > < 结束扇区号 >	表 233

表 224. UART ISP 命令汇总 *（续）*

ISP 命令	用法	说明见：
空白检查扇区	I < 起始扇区号 > < 结束扇区号 >	表 234
读取器件 ID	J	表 235
读取引导代码版本	K	表 237
比较	M < 地址 1> < 地址 2> < 字节数 >	表 238
读取 UID	N	表 239
读取 CRC 校验和	S < 地址 > < 字节数 >	表 240

22.5.1.1 解锁 < 解锁代码 >

表 225. UART ISP“ 解锁 ” 命令

命令	U
输入	解锁代码：23130_{10}
返回码	CMD_SUCCESS | INVALID_CODE | PARAM_ERROR
说明	该命令用于解锁闪存 “ 写入 ”、“ 擦除 ” 和 “ 运行 ” 命令。
示例	"U 23130<CR><LF>" 解锁闪存 “ 写入 ”、“ 擦除 ” 和 “ 运行 ” 命令。

22.5.1.2 设置波特率 < 波特率 > < 终止位 >

表 226. UART ISP“ 设置波特率 ” 命令

命令	B
输入	波特率：9600 | 19200 | 38400 | 57600 | 115200 终止位：1 | 2
返回码	CMD_SUCCESS | INVALID_BAUD_RATE | INVALID_STOP_BIT | PARAM_ERROR
说明	该命令用于更改波特率。新的波特率在命令处理程序发送 CMD_SUCCESS 返回码后生效。
示例	"B 57600 1<CR><LF>" 将串行端口的波特率设置为 57600 bps 和 1 个终止位。

22.5.1.3 应答 < 设置 >

表 227. UART ISP“ 应答 ” 命令

命令	A
输入	设置：ON = 1 | OFF = 0
返回码	CMD_SUCCESS | PARAM_ERROR
说明	应答命令默认设置为 “ 开启 ”。该命令 “ 开启 ” 时，ISP 命令处理程序会将接收到的串行数据发送回主机。
示例	"A 0<CR><LF>" 关闭应答。

22.5.1.4 写入 RAM < 起始地址 > < 字节数 >

主机在接收到 CMD_SUCCESS 返回码后应当发送普通二进制编码。传输结束后，该 ISP 命令处理程序以 “OK<CR><LF>” 响应。

表 228. UART ISP“ 写入 RAM” 命令

命令	W
输入	**起始地址：**将写入数据字节的 RAM 地址。该地址应当是某个字边界。 **字节数：**要写入的字节数。该数字应当是 4 的倍数。
返回码	CMD_SUCCESS | ADDR_ERROR （地址不在字边界上）| ADDR_NOT_MAPPED | COUNT_ERROR （字节数不是 4 的倍数） | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
说明	该命令用于将数据下载到 RAM。启用 2 级或 3 级代码读保护后，该命令会被锁定。对于 CRP1，写入 0x1000 0300 以下地址被禁用。
示例	"W 268436224 4<CR><LF>" 将 4 字节数据写入地址 0x1000 0300。

22.5.1.5 读存储器 < 地址 > < 字节数 >

读取数据流的普通二进制编码，后跟 CMD_SUCCESS 返回码。

表 229. UART ISP“ 读存储器 ” 命令

命令	R
输入	**起始地址：**读取数据字节的起始地址。该地址应当是某个字边界。 **字节数：**要读取的字节数。该数字应当是 4 的倍数。
返回码	CMD_SUCCESS，后跟 < 实际数据 （普通二进制数） > | ADDR_ERROR （地址不在字边界上）| ADDR_NOT_MAPPED | COUNT_ERROR （字节数不是 4 的倍数）| PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
说明	该命令用于从 RAM 或闪存中读取数据。代码读保护使能时该命令会被禁止。
示例	"R 268435456 4<CR><LF>" 从地址 0x1000 0000 中读取 4 字节。

22.5.1.6 准备写操作扇区 < 起始扇区号 > < 结束扇区号 >

该命令使闪存写 / 擦除操作分为两步。

表 230. UART ISP“ 准备写操作扇区 ” 命令

命令	P
输入	**起始扇区号。** **结束扇区号：**应当不小于起始扇区号。
返回码	CMD_SUCCESS | BUSY | INVALID_SECTOR | PARAM_ERROR
说明	该命令必须在执行 “ 从 RAM 复制到闪存 ” 或 “ 擦除扇区 ” 命令之前执行。成功执行 “ 从 RAM 复制到闪存 ” 或 “ 擦除扇区 ” 命令可使相关扇区得到再次保护。不能使用该命令准备引导模块。要准备单一扇区，应当使用相同的 “ 起始 ” 扇区号和 “ 结束 ” 扇区号。
示例	"P 0 0<CR><LF>" 准备闪存扇区 0。

22.5.1.7 从 RAM 复制到闪存 < 闪存地址 > <RAM 地址 > < 字节数 >

写入闪存时存在下列限制：

1. “ 从 RAM 复制到闪存 ” 命令所能写入的最少数据为 64 字节 （相当于一页）。
2. 一页包含 16 个闪存字 （行），每次可供更改的闪存写入操作的最少数量为一个闪存字（一行）。该限制是由于在闪存写操作中应用了 ECC 而造成的，参见章节 22.4.2。
3. 为避免写入干扰 （一种闪存固有的机制），一页内完成 16 次连续写操作后应执行一次擦除操作。需要注意的是，擦除操作会擦除整个扇区的内容。

   注：一旦页内完成了 16 次写操作 ，在不执行扇区擦除的情况下依然可将内容写入同一扇区的另一页中 （假定那些页之前已擦除过）。

表 231. UART ISP“ 从 RAM 复制到闪存 ” 命令

命令	C
输入	**闪存地址 (DST)：**要写入数据字节的目标闪存地址。目标地址应当为某个 64 字节边界。 **RAM 地址 (SRC)：**要读取数据字节的源 RAM 地址。 **字节数：**要写入的字节数。应当为 64 | 128 | 256 | 512 | 1024。
返回码	CMD_SUCCESS | SRC_ADDR_ERROR （地址不在字边界上） | DST_ADDR_ERROR （地址不在正确边界上） | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR （字节数不是 64 | 128 | 256 | 512 | 1024） | SECTOR_NOT_PREPARED_FOR WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
说明	该命令用于对闪存进行编程。在该命令前应当执行 “ 准备写操作扇区 ” 命令。一旦成功执行复制命令，受影响的扇区将自动得到重新保护。不能用该命令对引导模块进行写操作。代码读保护使能时该命令会被禁止。
示例	"C 0 268437504 512<CR><LF>" 从 RAM 地址 0x1000 0800 复制 512 字节到闪存地址 0。

22.5.1.8 “ 运行 ”< 地址 > < 模式 >

表 232. UART ISP“ 运行 ” 命令

命令	G
输入	**地址：**开始执行代码的闪存或 RAM 地址。该地址应当在某个字边界上。 **模式：**T （在 Thumb 模式下执行程序）。
返回码	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
说明	该命令用于执行驻留在 RAM 或闪存中的程序。一旦成功执行该命令，则可能无法返回至 ISP 命令处理程序。代码读保护使能时该命令会被禁止。使用该命令的地址必须不低于 0x0000 0200。
示例	"G 512 T<CR><LF>" 在 Thumb 模式下转移到地址 0x0000 0200。

22.5.1.9 擦除扇区 < 起始扇区号 > < 结束扇区号 >

表 233. UART ISP“ 擦除扇区 ” 命令

命令	E
输入	**起始扇区号。** **结束扇区号：**应当不小于起始扇区号。
返回码	CMD_SUCCESS | BUSY | INVALID_SECTOR | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
说明	该命令用于擦除片内闪存的一个或多个扇区。不能使用该命令来擦除引导模块。代码读保护使能时，该命令只允许擦除所有用户扇区的内容。
示例	"E 2 3<CR><LF>" 擦除闪存扇区 2 和 3。

22.5.1.10 空白检查扇区 < 扇区号 > < 结束扇区号 >

表 234. UART ISP“ 空白检查扇区 ” 命令

命令	I
输入	**起始扇区号：** **结束扇区号：**应当不小于起始扇区号。
返回码	CMD_SUCCESS | SECTOR_NOT_BLANK（后跟 < 第一个非空白字位置的偏移 > < 非空白字位置的内容 >） | INVALID_SECTOR | PARAM_ERROR
说明	该命令用于空白检查片内闪存的一个或多个扇区。 **对扇区 0 进行的空白检查总是由于前 64 字节重新映射到闪存引导模块而失败。** CRP 使能时，空白检查命令会针对非空的扇区偏移和扇区值返回 0。无论 CRP 如何设置，空扇区都会被正确报告。
示例	"I 2 3<CR><LF>" 空白检查闪存扇区 2 和 3。

22.5.1.11 读取器件标识号

表 235. UART ISP“ 读取器件标识 ” 命令

命令	J
输入	无。
返回码	CMD_SUCCESS，后跟 ASCII 格式的器件标识号 （参见表 226）。
说明	该命令用于读取器件识别号。

表 236. 器件标识号

设备	Hex 编码
LPC810M021FN8	0x0000 8100
LPC811M001FDH16	0x0000 8110
LPC812M101FDH16	0x0000 8120
LPC812M101FD20	0x0000 8121
LPC812M101FDH20	0x0000 8122

22.5.1.12 读取引导代码版本号

表 237. UART ISP“ 读取引导代码版本号 ” 命令

命令	K
输入	无
返回码	CMD_SUCCESS，后跟 2 字节 ASCII 格式的引导代码版本号。它将被解析为 <byte1（主要） >.<byte0 （次要） >。
说明	该命令用于读取引导代码版本号。

22.5.1.13 比较 < 地址 1> < 地址 2> < 字节数 >

表 238. UART ISP“ 比较 ” 命令

命令	M
输入	**地址 1(DST)：**要比较数据字节的起始闪存或 RAM 地址。该地址应当是某个字边界。 **地址 2(SRC)：**要比较数据字节的起始闪存或 RAM 地址。该地址应当是某个字边界。 **字节数：**要比较的字节数；应当是 4 的倍数。
返回码	CMD_SUCCESS |（源数据和目标数据相等） COMPARE_ERROR |（后跟第一个不匹配的偏移） COUNT_ERROR （字节数不是 4 的倍数）| ADDR_ERROR | ADDR_NOT_MAPPED | PARAM_ERROR
说明	该命令用于比较两个位置的存储器内容。
示例	"M 8192 268468224 4<CR><LF>" 可比较 RAM 地址 0x1000 8000 中的 4 个字节和闪存地址 0x2000 中的 4 个字节。

22.5.1.14 读取 UID

表 239. UART ISP“ 读取 UID” 命令

命令	N
输入	无
返回码	CMD_SUCCESS，后跟 E 类测试信息的 4 个 32 位字 （ASCII 格式）。低位地址的字首先发送。
说明	该命令用于读取唯一的 ID。

22.5.1.15 读取 CRC 校验和 < 地址 > < 字节数 >

获取 RAM 或闪存块的 CRC 校验和。CMD_SUCCESS，后跟 8 字节 ASCII 格式的 CRC 校验和。

校验和按下式计算：

CRC-32 多项式：x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

种子值：0xFFFF FFFF

不对数据输入进行位顺序取反

不对数据输入采用 1 的补码

不对 CRC 和进行位顺序取反

不对 CRC 和采用 1 的补码

表 240. UART ISP 读取 CRC 校验和命令

命令	S
输入	**地址**：数据从该地址读取，用于 CRC 校验和的计算。该地址必须在字边界上。 **字节数**：CRC 校验和待计算的字节数；必须是 4 的倍数。
返回码	CMD_SUCCESS，后跟普通二进制格式数据 ADDR_ERROR （地址不在字边界上）| ADDR_NOT_MAPPED | COUNT_ERROR （字节数不是 4 的倍数）| PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
说明	该命令用于从 RAM 或闪存块中读取 CRC 校验和。代码读保护使能时该命令会被禁止。
示例	"S 268436736 4<CR><LF>" 从地址 0x1000 0500 中读取 4 字节 CRC 校验和数据。 若校验和数值为 0xCBF43926，则主机将收到： "3421780262 <CR><LF>"

22.5.1.16 UART ISP 返回码

表 241. UART ISP 返回码汇总

返回码	助记符	说明
0	CMD_SUCCESS	命令执行成功。主机给出的命令只有在完成其成功执行时才会由 ISP 处理程序发送。
1	INVALID_COMMAND	无效命令。
2	SRC_ADDR_ERROR	源地址不在字边界上。
3	DST_ADDR_ERROR	目标地址不在正确边界上。
4	SRC_ADDR_NOT_MAPPED	源地址未在存储器映射中映射。适用时应考虑计数值。
5	DST_ADDR_NOT_MAPPED	目标地址未在存储器映射中映射。适用时应考虑计数值。
6	COUNT_ERROR	计数的字节不是 4 的倍数或并非允许值。
7	INVALID_SECTOR	扇区号无效或结束扇区号大于始扇区号。
8	SECTOR_NOT_BLANK	扇区非空。
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	准备写操作扇区的命令未被执行。
10	COMPARE_ERROR	源数据和目标数据不相等。
11	BUSY	闪存编程硬件接口忙。
12	PARAM_ERROR	参数数量不足或参数无效。
13	ADDR_ERROR	地址不在字边界上。
14	ADDR_NOT_MAPPED	地址未在存储器映射中映射。适用时应考虑计数值。
15	CMD_LOCKED	命令被锁定。
16	INVALID_CODE	解锁代码无效。
17	INVALID_BAUD_RATE	无效的波特率设置。
18	INVALID_STOP_BIT	无效的终止位设置。
19	CODE_READ_PROTECTION_使能	代码读保护使能。

22.5.2 IAP 命令

对于在应用编程， IAP 例程应通过寄存器 r0 中的字指针来调用，该指针指向包含命令代码和参数的内存 (RAM)。 IAP 命令的结果在指向寄存器 r1 的结果表中返回。用户可赋予寄存器 r0 和 r1 中的指针相同的值，如此便能将命令表复用来存放结果。参数表应足够大，以便在结果数超过参数数量时可以保存所有的结果。参数传递参见图 43。参数和结果的数目根据 IAP 命令而有所不同。参数的最大数目为 5，传递到 “ 从 RAM 复制到闪存 ” 命令。结果的最大数目为 4，由 “ 读取 UID” 命令返回。当接收到未定义的命令时，命令处理程序会发送状态码 INVALID_COMMAND。IAP 程序是 Thumb 代码，驻留在地址 0x1FFF 1FF0 处。

IAP 函数可通过使用 C 语言的下列方法调用。

定义 IAP 位置入口点。由于设置了 IAP 地址的第 0 位，因此当程序计数器转移到该地址时会使当前指令集变为 Thumb 指令集。

```
#define IAP_LOCATION 0x1fff1ff1
```

定义 IAP 函数的数据架构或指针来传递 IAP 命令表和结果表：

```
unsigned long command[5];
unsigned long result[4];
```

或

```
unsigned long * command;
unsigned long * result;
command=(unsigned long *) 0x...
result= (unsigned long *) 0x...
```

定义函数类型的指针，它有两个参数并返回 void。注意，IAP 返回的结果带有驻留在 R1 中的表的基址。

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

设置函数指针：

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

无论何时想调用 IAP，都可以使用下面的语句。

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

按照 ARM 规范（ARM 拇指程序调用标准 SWS ESPC 0002 A-05），在寄存器 r0、r1、r2 和 r3 中分别最多传递 4 个参数。其它参数在堆栈上传递。在寄存器 r0、r1、r2 和 r3 中分别最多返回 4 个参数。其它参数通过内存间接返回。有些 IAP 调用要求的参数多于 4 个。如果使用 ARM 推荐的模式进行参数传递 / 返回，则可能由于来自不同供应商的 C 编译器之间的执行差异而产生问题。推荐的参数传递模式可降低这种风险。

写操作或擦除操作期间，无法访问闪存。可导致闪存写 / 擦除操作的 IAP 命令使用片内 RAM 顶部的 32 位空间来进行执行操作。如果应用程序中允许进行 IAP 闪存编程操作，则用户程序不应使用该空间。

表 242. IAP 命令汇总

IAP 命令	命令代码	说明见：
准备写操作扇区	50（十进制）	表 243
从 RAM 复制到闪存	51（十进制）	表 244
擦除扇区	52（十进制）	表 245
空白检查扇区	53（十进制）	表 246
读取器件 ID	54（十进制）	表 247
读取引导代码版本	55（十进制）	表 248
比较	56（十进制）	表 249
重新调用 ISP	57（十进制）	表 250
读取 UID	58（十进制）	表 251
擦除页	59（十进制）	表 252

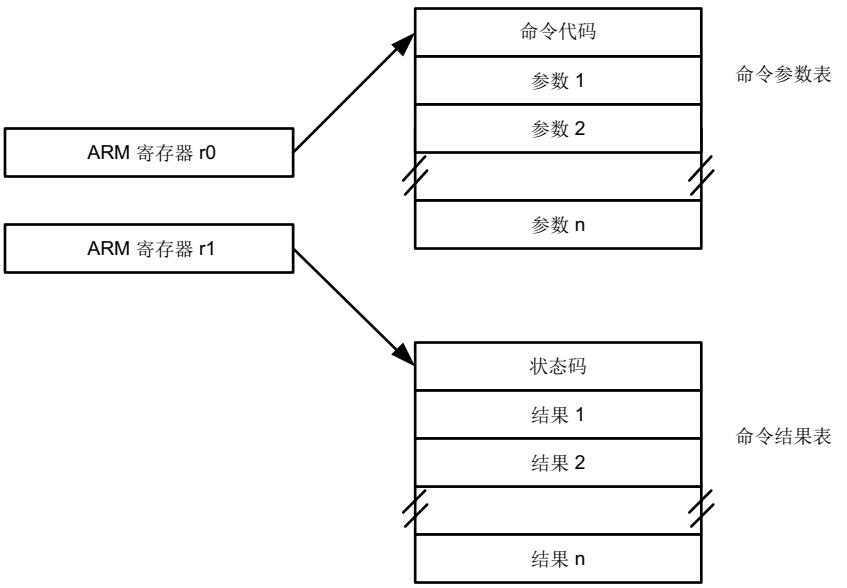


图 43. IAP 参数传递

22.5.2.1 准备写操作扇区 (IAP)

该命令使闪存写 / 擦除操作分为两步。

表 243. IAP 准备写操作命令的扇区

命令	准备写操作扇区
输入	**命令代码：50（十进制）** **参数 0：**起始扇区号。 **参数 1：**结束扇区号（应当不小于起始扇区号）。

表 243. IAP 准备写操作命令的扇区

命令	准备写操作扇区
返回码	CMD_SUCCESS | BUSY | INVALID_SECTOR
结果	无
说明	该命令必须在执行 “ 从 RAM 复制到闪存 ” 或 “ 擦除扇区 ” 命令之前执行。成功执行 “ 从 RAM 复制到闪存 ” 或 “ 擦除扇区 ” 命令可使相关扇区得到再次保护。不能使用该命令准备引导扇区。要准备单一扇区，应当使用相同的 “ 起始 ” 扇区号和 “ 结束 ” 扇区号。

22.5.2.2 从 RAM 复制到闪存 (IAP)

有关写入到闪存过程的限制，请参见章节 22.5.1.4。

表 244. IAP“ 从 RAM 复制到闪存 ” 命令

命令	从 RAM 复制到闪存
输入	**命令代码：51 （十进制）** **参数 0(DST)：**要写入数据字节的目标闪存地址。该地址应当是某个 64 字边界。 **参数 1(SRC)：**从中读取数据字节的源 RAM 地址。该地址应当是某个字边界。 **参数 2：**要写入的字节数。应当为 64 | 128 | 256 | 512 | 1024。 **参数 3：**系统时钟频率 (CCLK) （单位：kHz）。
返回码	CMD_SUCCESS | SRC_ADDR_ERROR （地址并非为某个字边界） | DST_ADDR_ERROR （地址不在正确边界上） | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR （字节数不是 256 | 512 | 1024 | 4096） | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY
结果	无
说明	该命令用于对闪存进行编程。受影响的扇区应首先通过调用 “ 准备写操作扇区 ” 命令准备好。一旦成功执行复制命令，受影响的扇区将自动得到重新保护。不能使用该命令对引导扇区进行写操作。

22.5.2.3 擦除扇区 (IAP)

表 245. IAP 擦除扇区命令

命令	擦除扇区
输入	**命令代码：52 （十进制）** **参数 0：**起始扇区号。 **参数 1：**结束扇区号 （应当不小于起始扇区号）。 **参数 2：**系统时钟频率 (CCLK) （单位：kHz）。

表 245. IAP 擦除扇区命令 *（续）*

命令	擦除扇区
返回码	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
结果	无
说明	该命令用于擦除片内闪存的一个或多个扇区。该命令不能擦除引导扇区。要擦除单一扇区，应当使用相同的 “ 起始 ” 扇区号和 “ 结束 ” 扇区号。

22.5.2.4 空白检查扇区 (IAP)

表 246. IAP 空白检查扇区命令

命令	空白检查扇区
输入	**命令代码：53 （十进制）** **参数 0：**起始扇区号。 **参数 1：**结束扇区号 （应当不小于起始扇区号）。
返回码	CMD_SUCCESS | BUSY | SECTOR_NOT_BLANK | INVALID_SECTOR
结果	**结果 0：**如果状态码是 SECTOR_NOT_BLANK，此项为第一个非空白字位置的偏移。 **结果 1：**非清空字位置的内容。
说明	该命令用于空白检查片内闪存的一个或多个扇区。要空白检查单一扇区，应当使用相同的 “ 起始 ” 扇区号和 “ 结束 ” 扇区号。

22.5.2.5 读取器件标识号 (IAP)

表 247. IAP“ 读取器件标识 ” 命令

命令	读取器件标识号
输入	**命令代码：54 （十进制）** **参数：**无
返回码	CMD_SUCCESS
结果	**结果 0：**器件识别号。
说明	该命令用于读取器件识别号。

22.5.2.6 读取引导代码版本号 (IAP)

表 248. IAP“ 读取引导代码版本号 ” 命令

命令	读取引导代码版本号
输入	**命令代码：55 （十进制）** **参数：**无
返回码	CMD_SUCCESS
结果	**结果 0：**2 字节引导代码版本号。读取为 <byte1 （主要） >.<byte0 （次要） >
说明	该命令用于读取引导代码版本号。

22.5.2.7 比较 < 地址 1> < 地址 2> < 字节数 > (IAP)

表 249. IAP“ 比较 ” 命令

命令	比较
输入	**命令代码：56 （十进制）** **参数 0(DST)：**要比较数据字节的起始闪存或 RAM 地址。该地址应当是某个字边界。 **参数 1(SRC)：**要比较数据字节的起始闪存或 RAM 地址。该地址应当是某个字边界。 **参数 2：**要比较的字节数；应当是 4 的倍数。
返回码	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。
说明	该命令用于比较两个位置的存储器内容。

22.5.2.8 重新调用 ISP (IAP)

表 250. IAP“ 重新调用 ISP”

命令	比较
输入	**命令代码：57 （十进制）**
返回码	无
结果	**无。**
说明	该命令用于调用 ISP 模式中的引导加载程序。它会映射引导向量，设定 PCLK = CCLK，并配置 USART0 引脚 U0_RXD 和 U0_TXD。内部闪存中出现有效的用户程序且 PIO0_1 引脚不可访问时，可使用该指令强制进入 ISP 模式。

22.5.2.9 读取 UID (IAP)

表 251. IAP“ 读取 UID” 命令

命令	比较
输入	**命令代码：58 （十进制）**
返回码	CMD_SUCCESS
结果	**结果 0：**第 1 个 32 位字 （在最低地址）。 **结果 1：**第 2 个 32 位字。 **结果 2：**第 3 个 32 位字。 **结果 3：**第 4 个 32 位字。
说明	该命令用于读取唯一的 ID。

22.5.2.10 擦除页

表 252. IAP“ 擦除页 ” 命令

命令	擦除页
输入	**命令代码：59 （十进制）** **参数 0：**起始页号。 **参数 1：**结束页号 （应当不小于起始页） **参数 2：**系统时钟频率 (CCLK) （单位：kHz）。
返回码	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
结果	无
说明	该命令用于擦除片内闪存的一页或多页。要擦除单页，应当使用相同的 “ 起始 ” 页号和 “ 结束 ” 页号。

22.5.2.11 IAP 状态代码

表 253. IAP 状态代码汇总

状态码	助记符	说明
0	CMD_SUCCESS	命令执行成功。
1	INVALID_COMMAND	无效命令。
2	SRC_ADDR_ERROR	源地址不在字边界上。
3	DST_ADDR_ERROR	目标地址不在正确边界上。
4	SRC_ADDR_NOT_MAPPED	源地址未在存储器映射中映射。适用时应考虑计数值。
5	DST_ADDR_NOT_MAPPED	目标地址未在存储器映射中映射。适用时应考虑计数值。
6	COUNT_ERROR	计数的字节不是 4 的倍数或并非允许值。
7	INVALID_SECTOR	扇区号无效。
8	SECTOR_NOT_BLANK	扇区非空。
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	准备写操作扇区的命令未被执行。
10	COMPARE_ERROR	源数据和目标数据不相同。
11	BUSY	闪存编程硬件接口忙。

22.6 功能说明

22.6.1 通信协议

所有 UART ISP 命令都应当作为单一 ASCII 字符串发送。这些字符串应使用回车 (CR) 和 / 或换行 (LF) 控制字符来终止。额外的 <CR> 和 <LF> 字符会被忽略。所有 ISP 响应都作为以 <CR><LF> 终止的 ASCII 字符串发送。数据以普通二进制格式发送和接收。

22.6.1.1 UART ISP 命令格式

“ 命令 参数 _0 参数 _1 ...Parameter_n<CR><LF>" "Data" （Data 只针对写命令）。

22.6.1.2 UART ISP 响应格式

"Return_Code<CR><LF>Response_0<CR><LF>Response_1<CR><LF>...Response_n <CR><LF>" "Data" （Data 只针对读命令）。

22.6.1.3 UART ISP 数据格式

数据流采用普通二进制格式。

22.6.2 针对 ISP 和 IAP 的存储器和中断使用

22.6.2.1 UART ISP 过程中的中断

在任何复位后，闪存引导模块中的引导模块中断向量都有效。

22.6.2.2 IAP 过程中的中断

片内闪存在擦除 / 写操作期间无法访问。用户应用程序代码开始执行时，来自用户闪存区的中断向量变为有效。执行任何 IAP 调用前，禁用中断或是确保 RAM 中的用户中断向量有效且中断处理程序驻留在 RAM 中。 IAP 代码不使用或禁用中断。

22.6.2.3 ISP 命令处理程序使用的 RAM

ISP 命令堆栈位于 0x1000 0270。堆栈最大使用量为 540 字节，向下增长。

22.6.2.4 IAP 命令处理程序使用的 RAM

在分配给用户的堆栈空间中，可使用的最大堆栈为 148 字节，向下增长。

22.6.3 调试

22.6.3.1 比较闪存映像

调试器连接时可见的存储器可能是引导 ROM、内部 SRAM 或闪存，具体取决于使用的调试器和 IDE 调试设置。为有助于确定当前调试环境下的存储器，应检查闪存地址 0x0000 0004 的值。该地址含有 ARM Cortex-M0+ 向量表代码的入口点，分别是引导 ROM、内部 SRAM 或闪存底部。

表 254. 调试模式下的存储器映射

存储器映射模式	存储器起始地址在 0x0000 0004 处可见
引导加载程序模式	0x1FFF 0000
用户闪存模式	0x0000 0000
用户 SRAM 模式	0x1000 0000

22.6.3.2 串行线调试 (SWD) 闪存编程接口

调试工具可将一部分闪存映像写入 RAM，然后按照正确的偏移地址重复调用 IAP 命令 “ 从 RAM 复制到闪存 ”。

23.1 本章导读

所有 LPC800 器件都提供电源配置。

23.2 特性

- 包括基于 ROM 的应用服务
- 电源管理服务
- 时序服务

23.3 简介

通过简单调用电源配置，可针对应用优化激活模式和睡眠模式下的功耗。电源配置例程可将 LPC800 配置为下列某种电源模式：

- 对应于复位后电源配置的默认模式。
- 对应于优化后处理功能的 CPU 性能模式。
- 对应于经过优化平衡的功耗和 CPU 性能的效率模式。
- 对应于最低功耗的低电流模式。

此外，电源配置还包括针对给定系统时钟和 PLL 输入时钟选择最优 PLL 设置的例程。

注：调用电源配置 API 前，禁用所有中断。完成电源配置 API 调用后可重新使能中断。

执行由 ROM 驱动器表中的指针所指向的功能可完成 ROM 对 API 的调用。图 44 显示的是调用电源配置 API 的指针结构。

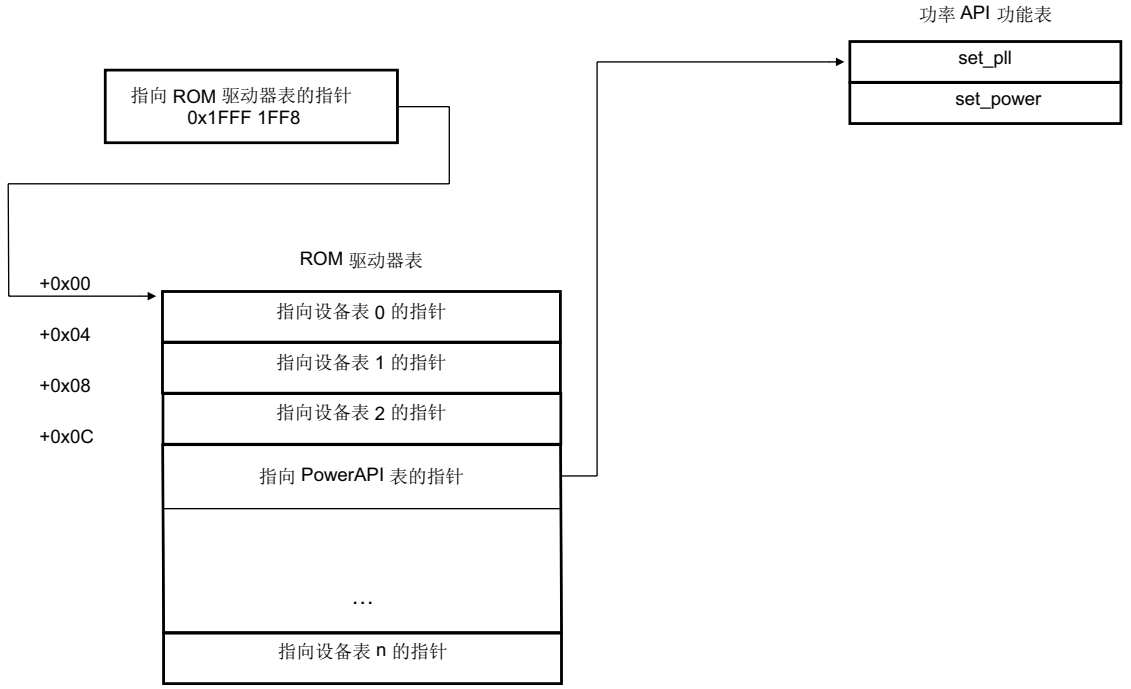


图 44. 电源配置指针结构

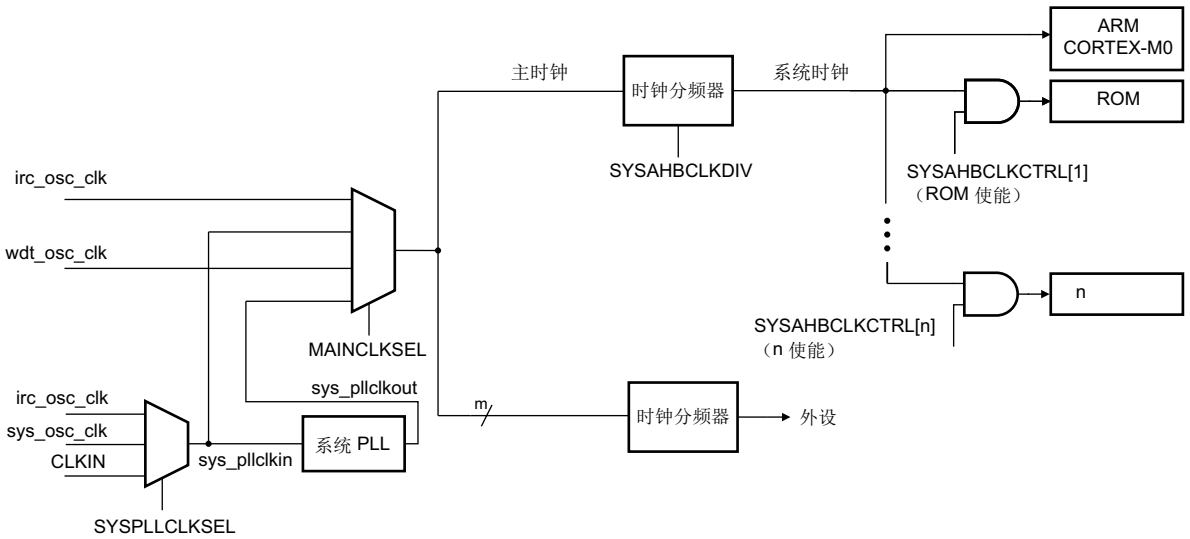


图 45. 用于电源 API 的 LPC800 时钟配置

23.4 API 说明

电源配置 API 提供配置系统时钟和优化系统设置的功能，可实现最低功耗。

表 255. 电源配置 API 调用

API 调用	说明	参考
set_pll （命令，结果）	电源 API 设置 PLL 例程	表 256
set_power （命令，结果）	电源 API 设置电源例程	表 257

下列元素必须在使用电源配置的应用中定义：

```
typedef struct _PWRD {
        void (*set_pll)(unsigned int cmd[], unsigned int resp[]);
        void (*set_power)(unsigned int cmd[], unsigned int resp[]);
} PWRD;
typedef struct _ROM {
        const PWRD * pWRD;
} ROM;
ROM ** rom = (ROM **) (0x1FFF1FF8 + 3 * sizeof(ROM**));
unsigned int command[4], result[2];
```

23.4.1 set_pll

该例程根据调用自变量设置系统 PLL。如果只需对系统 PLL 输入进行分频即可获取目标时钟，那么 *set_pll* 可旁路 PLL 以降低系统功耗。

注：调用该例程前，必须选定 PLL 时钟源（IRC/ 系统振荡器，表 13），主时钟源必须设置为系统 PLL 的输入时钟 （表 15），并且系统 /AHB 时钟分频器必须置位为 1 （表 17）。

*set_pll*将尝试查找与调用函数相匹配的PLL设置。一旦发现反馈分频器值（SYSPLLCTRL，M）、后置分频器比率 （SYSPLLCTRL， P）和系统 /AHB 时钟分频器 (SYSAHBCLKDIV) 的组合，*set_pll* 即输出选定值，并且在必要的时候切换主时钟源选项至系统 PLL 时钟输出。

例程返回结果代码，表示系统 PLL 是否正确设置；若正确则返回 PLL_CMD_SUCCESS，若不正确则识别错误。同时还将返回当前系统频率。应用程序应当利用该信息调整器件中的其他时钟 （SSP、 UART 和 WDT 时钟，和 / 或时钟输出）。

表 256. set_pll 例程

例程	set_pll
输入	**参数 0：**系统 PLL 输入频率 （单位：kHz） **参数 1：**期望系统时钟 （单位：kHz） **参数 2：**模式 （CPU_FREQ_EQU、 CPU_FREQ_LTE、 CPU_FREQ_GTE、 CPU_FREQ_APPROX） **参数 3：**系统 PLL 锁定时钟输出
结果	**结果 0：**PLL_CMD_SUCCESS | PLL_INVALID_FREQ | PLL_INVALID_MODE | PLL_FREQ_NOT_FOUND | PLL_NOT_LOCKED **结果 1：**系统时钟 （单位：kHz）

执行 set_pll 电源例程调用时，需用到下列定义：

```
/* set_pll mode options */
#defineCPU_FREQ_EQU
#defineCPU_FREQ_LTE
#defineCPU_FREQ_GTE
#defineCPU_FREQ_APPROX
/* set_pll result0 options */
#definePLL_CMD_SUCCESS
```

```
#definePLL_INVALID_FREQ
#definePLL_INVALID_MODE
#definePLL_FREQ_NOT_FOUND
#definePLL_NOT_LOCKED
```

有关时钟配置的简化原理图，请参见图 45。有关更多详情，请参见图 3。

23.4.1.1 参数 0：系统 PLL 输入频率和 Param1：期望系统时钟

set_pll 配置的主时钟频率不超过 30 MHz （参见图 45）。当预期系统时钟与系统 PLL 输入频率之比为整数时，可轻松找到该问题的解决方案；但其他情况下也一样有相应的解决方案。

系统 PLL 输入频率 (*Param0*) 必须在 10000 ~ 25000 kHz （10 MHz ~ 25 MHz，包括 10 MHz 和 25 MHz）范围内。预期系统时钟 (*Param1*) 必须在 1 ~ 30000 kHz（包括 1 kHz 和 30000kHz）范围内。若这些条件中的某一项不满足，则由于 PLL 设置未改变，*set_pll* 将返回 PLL_INVALID_FREQ 和 *Param0* 作为 *Result1*。

23.4.1.2 参数 2：mode

set_pll 的主要任务是找到以 *Param1* 中的指定速率产生系统时钟的设置。若无法找到确切匹配值，则应当使用输入参数模式 (*Param2*) 指定实际系统时钟是否可以小于等于、大于等于或近似等于预期系统时钟 (*Param1*) 值。

仅当PLL可确切输出*Param1*指定的频率时，针对CPU_FREQ_EQU的调用才可成功执行。

若不应超过请求频率 （例如出于总功耗和 / 或功率预算方面的原因），则可使用 CPU_FREQ_LTE。

若应用需要具有最低的 CPU 处理能力，则使用 CPU_FREQ_GTE。

CPU_FREQ_APPROX 使系统时钟尽可能接近请求值 （可能高于或低于请求值）。

若指定了某个非法模式，则 *set_pll* 返回 PLL_INVALID_MODE。若期望系统时钟超出该例程所能支持的范围，则 *set_pll* 返回 PLL_FREQ_NOT_FOUND。在这种情况下，当前 PLL 设置不会改变，并且 *Param0* 返回 *Result1*。

23.4.1.3 参数 3：系统 PLL 锁定时钟输出

若选定的配置有效，则系统PLL锁定时间应当不会超过100 ms。若 *Param3* 为0，则 *set_pll* 将一直等待 PLL 锁定。非零值表示在返回 PLL_NOT_LOCKED 前，代码将执行多少次 PLL 成功锁定事件检查。在这种情况下， PLL 设置不发生变化， *Param0* 返回 *Result1*。

注：PLL 锁定所需时间取决于所选 PLL 输入时钟源（IRC/ 系统振荡器）以及其特性。取决于工作条例（例如电源和 / 或环境温度），所选时钟源可能或多或少都会有些抖动。这便是建议当使用良好已知的时钟源并且 PLL_NOT_LOCKED 响应已接收时，应在声明所选 PLL 时钟源有效之前多次调用 set_pll 例程的原因。

提示：将 *Param3* 设为等于系统 PLL 频率 [Hz] 除以 10000，便可提供足够多的 PLL 锁定轮询周期。

23.4.2 set_power

根据调用自变量，该例程可配置器件内部电源控制设置。目标是降低工作功耗，同时尽量保持所需应用特性的性能最优。

注：使用 SYSAHBCLKDIV = 1（系统时钟分频器寄存器，参见表 17 和图 45）的 set_power 例程。

set_power 返回结果代码，汇报电源设置是否正确更改。

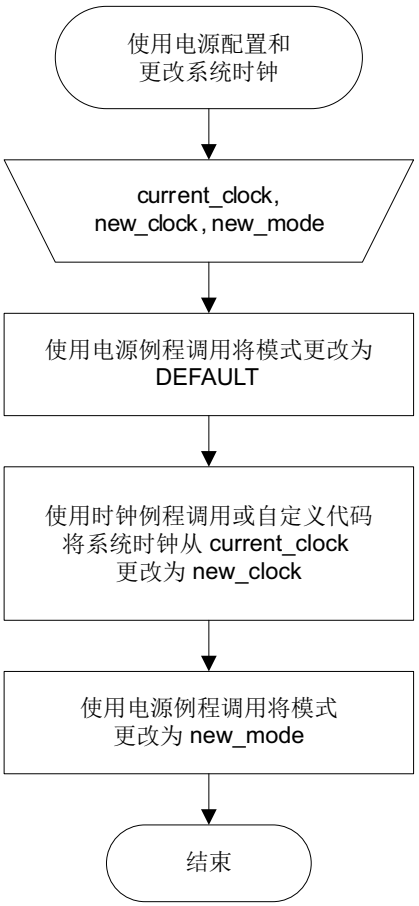


图 46. 使用电源配置

表 257. set_power 例程

例程	set_power
输入	**参数 0：**主时钟 （单位：MHz） **参数 1：**模式 （PWR_DEFAULT、 PWR_CPU_PERFORMANCE、 PWR_EFFICIENCY、 PWR_LOW_CURRENT） **参数 2：**系统时钟 （单位：MHz）
结果	**结果 0：**PWR_CMD_SUCCESS | PWR_INVALID_FREQ | PWR_INVALID_MODE

执行 set_power 例程调用时，需用到下列定义：

```
/* set_power mode options */
#definePWR_DEFAULT
#definePWR_CPU_PERFORMANCE
#definePWR_EFFICIENCY
#definePWR_LOW_CURRENT
```

```
/* set_power result0 options */
#definePWR_CMD_SUCCESS
#definePWR_INVALID_FREQ
#definePWR_INVALID_MODE
```

有关时钟配置的简化原理图，请参见图 45。有关更多详情，请参见图 3。

23.4.2.1 参数 0：主时钟

主时钟是微控制器用来为系统和外设提供时钟的时钟速率。通过正确执行时序例程调用或用户提供的类似代码，均可完成配置。该操作数必须为 1 ~ 30 MHz （包括 1 MHz 和 30 MHz）范围内的整数。若数值超出这一范围，则 *set_power* 返回 PWR_INVALID_FREQ，并且不会改变电源控制系统。

23.4.2.2 参数 1：mode

输入参数模式 (*Param1*) 指定 4 种可用电源设置中的一种。若选择无效，则 *set_power* 返回 PWR_INVALID_MODE 并且不会改变电源控制系统。

PWR_DEFAULT 将器件保持在于其复位状态相似的基准电源设置上。

PWR_CPU_PERFORMANCE 配置微控制器，以便为应用提供更多的处理能力。CPU 性能相比默认选项高 30%。

PWR_EFFICIENCY 设置设计为寻找当前电流大小和 CPU 执行代码以及处理数据的能力之间的平衡。该模式下，器件不仅在 CPU 性能方面，更在降低当前电流大小方面都优于默认模式。

PWR_LOW_CURRENT 用于重点在降低功耗而非提升 CPU 性能的解决方案。

23.4.2.3 参数 2：系统时钟

当调用*set_power*时，系统时钟即为微控制器工作时的时钟速率。该参数是范围为1 ~ 30 MHz（包括 1 MHz 和 30 MHz）的整数。

23.5 功能说明

23.5.1 时钟控制

有关时钟控制 API，请参见章节 23.5.1.1 至章节 23.5.1.6。

23.5.1.1 无效频率 （超出器件最大时钟速率）

```
command[0] = 12000;
command[1] = 60000;
command[2] = CPU_FREQ_EQU;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

上述代码指定 12 MHz PLL 输入时钟以及 60 MHz 系统时钟。应用就绪，无限等待 PLL 锁定。但 60 MHz 的预期系统时钟超出 30 MHz 的最大值。因此， *set_pll* 在 *result[0]* 中返回 PLL_INVALID_FREQ，在 *result[1]* 中返回 12000，不改变 PLL 设置。

23.5.1.2 所选频率无效 （系统时钟分频器限制）

```
command[0] = 12000;
```

```
command[1] = 40;
command[2] = CPU_FREQ_LTE;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

上述代码指定 12 MHz PLL 输入时钟，系统时钟不超过 40 kHz，等待 PLL 锁定时无超时。由于系统时钟的最大分频器值为 255 并且需除以 300 才可工作在 40 kHz 下，因此 *set_pll* 在 *result[0]* 中返回 PLL_INVALID_FREQ，在 *result[1]* 中返回 12000，不改变 PLL 设置。

23.5.1.3 无法找到确切解决方案 (PLL)

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_EQU;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

上述代码指定 12 MHz PLL 输入时钟以及 25 MHz 系统时钟。应用就绪，无限等待 PLL 锁定。由于在前文所述的限制内不存在有效 PLL 设置，因此 *set_pll* 在 *result[0]* 中返回 PLL_FREQ_NOT_FOUND，在 *result[1]* 中返回 12000，不改变 PLL 设置。

23.5.1.4 系统时钟小于等于预期值

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_LTE;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

上述代码指定 12 MHz PLL 输入时钟，系统时钟不超过 25 MHz，并且无锁定超时。*set_pll* 在 *result[0]* 中返回 PLL_CMD_SUCCESS，在 *result[1]* 中返回 24000。新系统时钟为 24 MHz。

23.5.1.5 系统时钟大于等于预期值

```
command[0] = 12000;
command[1] = 20000;
command[2] = CPU_FREQ_GTE;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

上述代码指定 12 MHz PLL 输入时钟，系统时钟最小值为 20 MHz，并且无锁定超时。*set_pll* 在 *result[0]* 中返回 PLL_CMD_SUCCESS，在 *result[1]* 中返回 24000。新系统时钟为 24 MHz。

23.5.1.6 系统时钟近似等于预期值

```
command[0] = 12000;
command[1] = 16500;
command[2] = CPU_FREQ_APPROX;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

上述代码指定 12 MHz PLL 输入时钟，系统时钟近似等于 16.5 MHz，并且无锁定超时。*set_pll* 在 *result[0]* 中返回 PLL_CMD_SUCCESS，在 *result[1]* 中返回 16000。新系统时钟为 16 MHz。

23.5.2 功率控制

有关电源控制 API 示例，请参见章节 23.5.1.1 和章节 23.5.2.2。

23.5.2.1 无效频率 （超出器件最大时钟速率）

```
command[0] = 30;
command[1] = PWR_CPU_PERFORMANCE;
command[2] = 40;
(*rom)->pWRD->set_power(command, result);
```

上述设置可用于运行在30 MHz主时钟和系统时钟、并且需要最大CPU处理功耗的系统中。由于指定的40 MHz时钟超过30 MHz最大值，*set_power*在*result[0]*中返回PWR_INVALID_FREQ，不改变任何当前电源设置。

23.5.2.2 一种可行的电源配置

```
command[0] = 24;
command[1] = PWR_CPU_EFFICIENCY;
command[2] = 24;
(*rom)->pWRD->set_power(command, result);
```

上述代码指定应用工作的主时钟和系统时钟为 24 MHz，侧重于效率。完成微控制器的内部电源控制后， *set_power* 在 *result[0]* 中返回 PWR_CMD_SUCCESS。

24.1 本章导读

所有 LPC800 器件都提供 I2C 总线 ROM API。

24.2 特性

- 简单 I2C 驱动器，在 I2C 总线上发送和接收数据。
- 轮询和中断驱动接收与发送功能，用于主机模式和从机模式下。

24.3 简介

任何应用程序都可调用驱动器，在 I2C 总线上发送或接收数据。通过 I2C 驱动器可轻松使用 I2C 接口产生工作项目。

ROM 例程允许用户以主机或从机方式操作 I2C 接口。软件例程不执行仲裁；仲裁可让主机模式在发送过程中切换到从机模式。

尽管主机仲裁不在这些 I2C 驱动器中实施，但可在具有多个主机的系统设计中使用。如果从驱动器返回的标志指示消息因仲裁丢失而发送失败，则应用程序会重新发送消息。

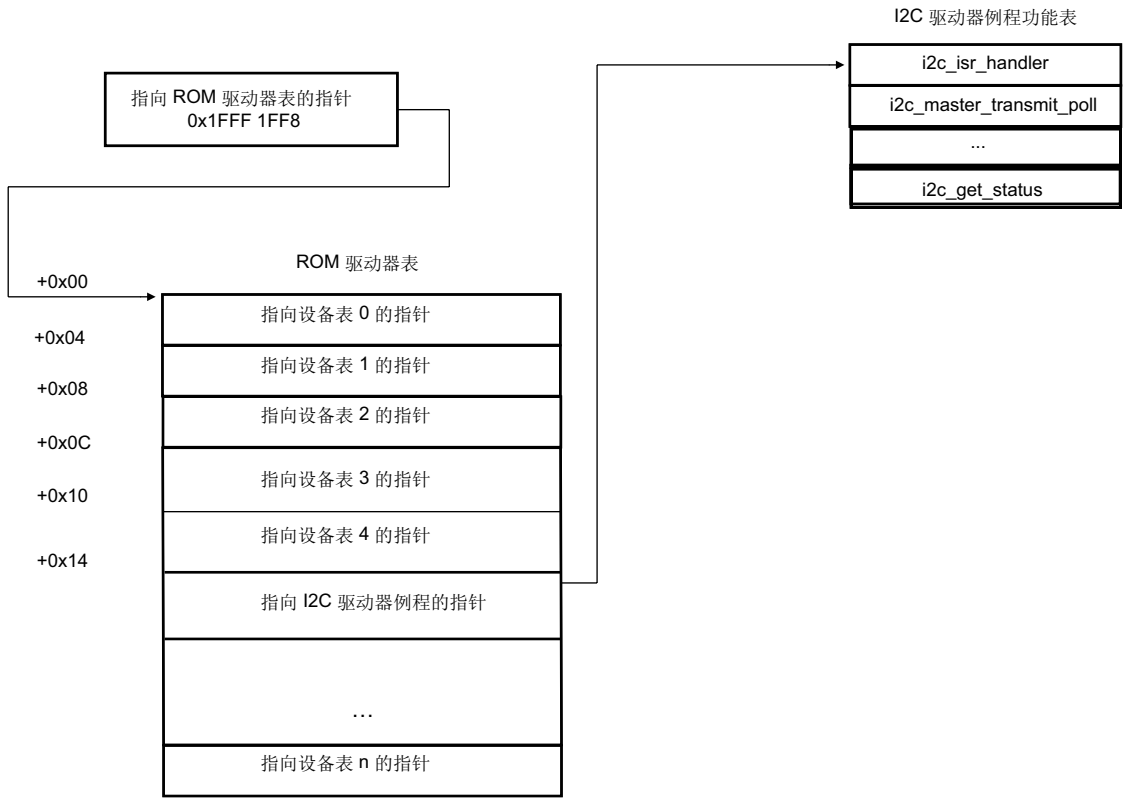


图 47. I2C 总线驱动器例程指针结构

24.4 API 说明

I2C API 包含可配置 I2C 并在主机和从机模式下发送和接收数据的函数。

表 258. I2C API 调用

API 调用	说明	参考
void i2c_isr_handler(I2C_HANDLE_T*)	I2C ROM 驱动器中断服务例程。	表 259
ErrorCode_t i2c_master_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)	I2C 主机发送轮询	表 260
ErrorCode_t i2c_master_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C 主机接收轮询	表 261
ErrorCode_t i2c_master_tx_rx_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C 主机发送和接收轮询	表 262
ErrorCode_t i2c_master_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C 主机发送中断	表 263
ErrorCode_t i2c_master_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C 主机接收中断	表 264
ErrorCode_t i2c_master_tx_rx_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)	I2C 主机发送接收中断	表 265

表 258. I2C API 调用

API 调用	说明	参考
ErrorCode_t i2c_slave_receive_poll(I2C_HANDLE_T* ， I2C_PARAM* ， I2C_RESULT*)	I2C 从机接收轮询	表 266
ErrorCode_t i2c_slave_transmit_poll(I2C_HANDLE_T*，I2C_PARAM* ， I2C_RESULT*)	I2C 从机发送轮询	表 267
ErrorCode_t i2c_slave_receive_intr(I2C_HANDLE_T* ， I2C_PARAM* ， I2C_RESULT*)	I2C 从机接收中断	表 268
ErrorCode_t i2c_slave_transmit_intr(I2C_HANDLE_T* ，I2C_PARAM* ， I2C_RESULT*)	I2C 从机发送中断	表 269
ErrorCode_t i2c_set_slave_addr(I2C_HANDLE_T*, slave_addr_0_3, slave_mask_0_3)	I2C 设置从机地址	表 270
uint32_t i2c_get_mem_size(void)	I2C 获取存储器大小	表 271
I2C_HANDLE_T* i2c_setup(i2c_base_addr, *start_of_ram)	I2C 设置	表 272
ErrorCode_t i2c_set_bitrate(I2C_HANDLE_T*, P_clk_in_hz, bitrate_in_bps)	I2C 设置比特率	表 273
uint32_t i2c_get_firmware_version(void)	I2C 获取固件版本	表 274
I2C_MODE_T i2c_get_status(I2C_HANDLE_T*)	I2C 获取状态	表 275
ErrorCode_t i2c_set_timeout(I2C_HANDLE_T* h_i2c, uint32_t timeout)	I2C 超时值	表 276

必须定义下列结构，才可使用 I2C API：

```
typedef struct I2CD_API {// index of all the i2c driver functions
void (*i2c_isr_handler) (I2C_HANDLE_T* h_i2c) ; // ISR interrupt service request
// MASTER functions ***
ErrorCode_t (*i2c_master_transmit_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
                                        I2C_RESULT* ptr );
ErrorCode_t (*i2c_master_receive_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
                                       I2C_RESULT* ptr );
ErrorCode_t (*i2c_master_tx_rx_poll)(I2C_HANDLE_T* h_i2c,I2C_PARAM* ptp,
                                     I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_master_transmit_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_master_receive_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_master_tx_rx_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT* ptr ) ;
// SLAVE functions ***
ErrorCode_t (*i2c_slave_receive_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_slave_transmit_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_slave_receive_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_slave_transmit_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_set_slave_addr)(I2C_HANDLE_T* h_i2c,
                    uint32_t slave_addr_0_3, uint32_t slave_mask_0_3)
// OTHER functions
uint32_t (*i2c_get_mem_size)(void) ; //ramsize_in_bytes memory needed by I2C drivers
I2C_HANDLE_T*  (*i2c_setup)(uint32_t i2c_base_addr, uint32_t *start_of_ram ) ;
ErrorCode_t  (*i2c_set_bitrate)(I2C_HANDLE_T* h_i2c, uint32_t P_clk_in_hz,
                    uint32_t bitrate_in_bps) ;
uint32_t (*i2c_get_firmware_version)() ;
I2C_MODE_T (*i2c_get_status)(I2C_HANDLE_T* h_i2c ) ;
} I2CD_API_T ;
```

24.4.1 ISR 处理程序

表 259. ISR 处理程序

例程	ISR 处理程序
原型	void i2c_isr_handler(I2C_HANDLE_T*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。
返回	无。
说明	I2C ROM 驱动器中断服务例程。使用 I2C ROM 驱动器中断模式时必须从 I2C ISR 调用该功能。

24.4.2 I2C 主机发送轮询

表 260. I2C 主机发送轮询

例程	I2C 主机发送轮询
原型	ErrorCode_t i2c_master_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。 I2C_PARAM - 指向 I2C PARAM 结构的指针。 I2C_RESULT - 指向 I2C RESULT 结构的指针。
返回	错误代码。
说明	将发送缓存区中的字节发送到从机。R/W 位为 0 的从机地址预期出现在发送缓冲区的第一个字节。除非 stop_flag =0，否则在结束时发送 STOP 条件。任务完成后，该函数会在发生调用后返回线路。

24.4.3 I2C 主机接收轮询

表 261. I2C 主机接收轮询

例程	I2C 主机接收轮询
原型	ErrorCode_t i2c_master_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。 I2C_PARAM - 指向 I2C PARAM 结构的指针。 I2C_RESULT - 指向 I2C RESULT 结构的指针。
返回	错误代码。
说明	接收来自从机的字节并将其置入接收缓冲区。R/W 位为 0 的从机地址预期出现在发送缓冲区的第一个字节。任务完成后，R/W 位为 1 的从机地址位于接收缓冲区的第一个字节中。除非 stop_flag =0，否则在结束时发送 STOP 条件。任务完成后，该函数会在发生调用后返回线路。

24.4.4 I2C 主机发送和接收轮询

表 262. I2C 主机发送和接收轮询

例程	I2C 主机发送和接收轮询
原型	ErrorCode_t i2c_master_tx_rx_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。 I2C_PARAM - 指向 I2C PARAM 结构的指针。 I2C_RESULT - 指向 I2C RESULT 结构的指针。
返回	错误代码。
说明	首先，将发送缓冲区内的字节发送到从机；其次，接收来自从机的字节并存储在接收缓冲区内。R/W 位为 0 的从机地址预期出现在发送缓冲区的第一个字节。任务完成后， R/W 位为 1 的从机地址位于接收缓冲区的第一个字节中。除非 stop_flag =0，否则在结束时发送 STOP 条件。任务完成后，该函数会在发生调用后返回线路。

24.4.5 I2C 主机发送中断

表 263. I2C 主机发送中断

例程	I2C 主机发送中断
原型	ErrorCode_t i2c_master_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。 I2C_PARAM - 指向 I2C PARAM 结构的指针。 I2C_RESULT - 指向 I2C RESULT 结构的指针。
返回	错误代码。
说明	将发送缓存区中的字节发送到从机。 R/W 位为 0 的从机地址预期出现在发送缓冲区的第一个字节。除非 stop_flag =0，否则在结束时发送 STOP 条件。程序控制将被立即返回，任务将以中断驱动的方式完成。任务完成后，回调函数被调用。

24.4.6 I2C 主机接收中断

表 264. I2C 主机接收中断

例程	I2C 主机接收中断
原型	ErrorCode_t i2c_master_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。 I2C_PARAM - 指向 I2C PARAM 结构的指针。 I2C_RESULT - 指向 I2C RESULT 结构的指针。
返回	错误代码。
说明	接收来自从机的字节并将其置入接收缓冲区。任务完成后， R/W 位为 1 的从机地址位于接收缓冲区的第一个字节中。除非 stop_flag =0，否则在结束时发送 STOP 条件。程序控制将被立即返回，任务将以中断驱动的方式完成。任务完成后，回调函数被调用。

24.4.7 I2C 主机发送接收中断

表 265. I2C 主机发送接收中断

例程	I2C 主机发送接收中断
原型	ErrorCode_t i2c_master_tx_rx_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。 I2C_PARAM - 指向 I2C PARAM 结构的指针。 I2C_RESULT - 指向 I2C RESULT 结构的指针。
返回	错误代码。
说明	首先，将发送缓冲区内的字节发送到从机；其次，接收来自从机的字节并存储在接收缓冲区内。R/W 位为 0 的从机地址预期出现在发送缓冲区的第一个字节。任务完成后， R/W 位为 1 的从机地址位于接收缓冲区的第一个字节中。除非 stop_flag =0，否则在结束时发送 STOP 条件。程序控制将被立即返回，任务将以中断驱动的方式完成。任务完成后，回调函数被调用。

24.4.8 I2C 从机接收轮询

表 266. I2C 从机接收轮询

例程	I2C 从机接收轮询
原型	ErrorCode_t i2c_slave_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。 I2C_PARAM - 指向 I2C PARAM 结构的指针。 I2C_RESULT - 指向 I2C RESULT 结构的指针。
返回	错误代码。
说明	接收来自主机的数据。任务完成后，该函数会在发生调用后返回线路。

24.4.9 I2C 从机发送轮询

表 267. I2C 从机发送轮询

例程	I2C 从机发送轮询
原型	ErrorCode_t i2c_slave_transmit_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。 I2C_PARAM - 指向 I2C PARAM 结构的指针。 I2C_RESULT - 指向 I2C RESULT 结构的指针。
返回	错误代码。
说明	将数据字节发送回主机。任务完成后，该函数会在发生调用后返回线路。

24.4.10 I2C 从机接收中断

表 268. I2C 从机接收中断

例程	I2C 从机接收中断
原型	ErrorCode_t i2c_slave_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。 I2C_PARAM - 指向 I2C PARAM 结构的指针。 I2C_RESULT - 指向 I2C RESULT 结构的指针。
返回	错误代码。
说明	接收来自主机的数据。程序控制将被立即返回，任务将以中断驱动的方式完成。任务完成后，回调函数被调用。

24.4.11 I2C 从机发送中断

表 269. I2C 从机发送中断

例程	I2C 从机发送中断
原型	ErrorCode_t i2c_slave_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。 I2C_PARAM - 指向 I2C PARAM 结构的指针。 I2C_RESULT - 指向 I2C RESULT 结构的指针。
返回	错误代码。
说明	向主机发送数据。程序控制将被立即返回，任务将以中断驱动的方式完成。任务完成后，回调函数被调用。

24.4.12 I2C 设置从机地址

表 270. I2C 设置从机地址

例程	I2C 设置从机地址
原型	ErrorCode_t i2c_set_slave_addr(I2C_HANDLE_T*, slave_addr_0_3, slave_mask_0_3)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。 Slave_addr_0_3 - unint32 变量。 7 位从机地址。 Slave_mask_0_3 - unint32 变量。从机地址掩码。
返回	错误代码。
说明	设置从机地址和相应的掩码。set_slave_addr() 函数支持 4 个 7 位从机地址和掩码。

24.4.13 I2C 获取存储器大小

表 271. I2C 获取存储器大小

例程	I2C 获取存储器大小
原型	uint32_t i2c_get_mem_size(void)
输入参数	无。
返回	uint32。
说明	返回 I2C 驱动器所需的 SRAM 中的字节数。

24.4.14 I2C 设置

表 272. I2C 设置

例程	I2C 设置
原型	I2C_HANDLE_T* i2c_setup(i2c_base_addr, *start_of_ram)
输入参数	I2C_base addr - unint32 变量。 I2C 外设的基本地址。 Start_of_ram - unint32 指针。指针指向分配的 SRAM。
返回	I2C_Handle。
说明	返回分配 SRAM 区域的句柄。

24.4.15 I2C 设置比特率

表 273. I2C 设置比特率

例程	I2C 设置比特率
原型	ErrorCode_t i2c_set_bitrate(I2C_HANDLE_T*, P_clk_in_hz, bitrate_in_bps)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。 P_clk_in_hz - unint32 变量。外设时钟单位为 Hz。 Bitrate_in_bps - unint32 变量。请求的 I2C 工作频率单位为 Hz。
返回	错误代码。
说明	配置 I2C 占空比寄存器 （SCLH 和 SCLL）。

24.4.16 I2C 获取固件版本

表 274. I2C 获取固件版本

例程	I2C 获取固件版本
原型	uint32_t i2c_get_firmware_version(void)
输入参数	无。
返回	I2C ROM 驱动器版本号。
说明	返回版本号。固件版本为无符号 32 位数字。

24.4.17 I2C 获取状态

表 275. I2C 获取状态

例程	I2C 获取状态
原型	I2C_MODE_T i2c_get_status(I2C_HANDLE_T*)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。
返回	状态码。
说明	返回状态码。状态码表示 I2C 总线状态。参见 I2C 状态码表。

24.4.18 I2C 超时值

表 276. I2C 超时值

例程	I2C 超时值
原型	ErrorCode_t i2c_set_timeout(I2C_HANDLE_T* h_i2c, uint32_t timeout)
输入参数	I2C_HANDLE_T - 已分配 SRAM 区域的句柄。 uint32_t timeout - 时间值为超时值乘以 16 的 I2C 功能时钟。如果超时等于 0，则超时特性会被禁用。
返回	状态码。
说明	返回状态码。状态码表示 I2C 总线状态。参见 I2C 状态码表。

24.4.19 错误码

表 277. 错误码

错误码	说明	备注
0	成功完成	函数成功执行。
1	常见错误	-
0x0006 0001	ERR_I2C_NAK	-
0x0006 0002	ERR_I2C_BUFFER_OVERFLOW	-
0x0006 0003	ERR_I2C_BYTE_COUNT_ERR	-
0x0006 0004	ERR_I2C_LOSS_OF_ARBRITRATION	-
0x0006 0005	ERR_I2C_SLAVE_NOT_ADDRESSED	-
0x0006 0006	ERR_I2C_LOSS_OF_ARBRITRATION_NAK_BIT	-
0x0006 0007	ERR_I2C_GENERAL_FAILURE	I2C 总线上检测到故障。
0x0006 0008	ERR_I2C_REGS_SET_TO_DEFAULT	I2C 时钟频率无法设置。0x04 默认值载入 SCLH 和 SCLL。

24.4.20 I2C 状态码

表 278. I2C 状态码

状态码	说明
0	IDLE
1	MASTER_SEND
2	MASTER_RECEIVE
3	SLAVE_SEND
4	SLAVE_RECEIVE

24.4.21 I2C ROM 驱动器变量

I2C ROM 驱动器要求声明并初始化特定变量以便正确使用。某些变量可以省略，具体取决于工作模式。

24.4.21.1 I2C 句柄

I2C 句柄是分配给 I2C ROM 驱动器的指针。句柄应当被定义为 I2C 句柄类型：

```
typedef void* I2C_HANDLE_T
```

句柄定义完成后，必须通过调用 i2c_setup() 函数，用 I2C 基址以及保留用于 I2C ROM 驱动器的 RAM 初始化句柄。

如果使用 I2C ROM 驱动器中断，则必须定义回调函数类型：

```
typedef void (*I2C_CALLBK_T) (uint32_t err_code, uint32_t n)
```

使用中断时，一旦某个任务完成， I2C ROM 驱动器将调用回调函数。

24.4.22 PARAM 和 RESULT 结构

I2C ROM 驱动器输入参数由两种结构组成：PARAM 结构和 RESULT 结构。 PARAM 结构包含发送给 I2C ROM 驱动器的参数， RESULT 结构包含调用 I2C ROM 驱动器后的结果。

PARAM 结构如下所示：

```
typedef struct i2c_A {//parameters passed to ROM function
        uint32_t num_bytes_send ;
        uint32_t num_bytes_rec ;
        uint8_t *buffer_ptr_send ;
        uint8_t *buffer_ptr_rec ;
        I2C_CALLBK_T func_pt;// callback function pointer
        uint8_t stop_flag;
        uint8_t dummy[3] ;        // required for word alignment
} I2C_PARAM ;
```

RESULT 结构如下所示：

```
typedef struct i2c_R {// RESULTs struct--results are here when returned
        uint32_t n_bytes_sent ;
        uint32_t n_bytes_recd ;
} I2C_RESULT ;
```

24.4.23 错误结构

I2C ROM 驱动器返回的错误码是一种枚举结构。错误结构如下所示：

```
typedef enum
{
                        LPC_OK=0, /**< enum value returned on Success */
                         ERROR,
 ERR_I2C_BASE = 0x00060000,
/*0x00060001*/ERR_I2C_NAK=ERR_I2C_BASE+1,
/*0x00060002*/ERR_I2C_BUFFER_OVERFLOW,
/*0x00060003*/ERR_I2C_BYTE_COUNT_ERR,
/*0x00060004*/ERR_I2C_LOSS_OF_ARBRITRATION,
/*0x00060005*/ERR_I2C_SLAVE_NOT_ADDRESSED,
/*0x00060006*/ERR_I2C_LOSS_OF_ARBRITRATION_NAK_BIT,
/*0x00060007*/ERR_I2C_GENERAL_FAILURE,
/*0x00060008*/ERR_I2C_REGS_SET_TO_DEFAULT
} ErrorCode_t;
```

24.4.24 I2C 模式

i2c_get_status() 函数返回 I2C 引擎的当前状态。返回码可定义为枚举结构：

```
typedef enum I2C_mode {
 IDLE,
 MASTER_SEND,
 MASTER_RECEIVE,
 SLAVE_SEND,
 SLAVE_RECEIVE
 } I2C_MODE_T ;
```

24.4.25 I2C ROM 驱动器指针

I2C ROM驱动器地址为0x1FFF1FF8。地址必须得到声明，以便允许对ROM驱动器的访问：

```
#define ROM_DRIVERS_PTR ((ROM *)(*((unsigned int *)0x1FFF1FF8)))
```

24.5 功能说明

24.5.1 I2C 设置

在 I2C ROM 中的任何设置函数前，应用程序负责执行下列步骤：

1. 使能 I2C 外设时钟。
2. 使能 I2C 外设的 SCL 和 SDA 输出所需的两个引脚。
3. 分配 I2C ROM 驱动器的专用 RAM 区域。

完成 I2C 模块配置后，必须设置 I2C ROM 驱动器变量：

1. 初始化指向 I2C API 函数表的指针。
2. 声明 PARAM 和 RESULT 结构。
3. 声明错误码结构。
4. 声明发送和接收缓冲区。

如果使用中断，则必须设置额外驱动器变量：

1. 声明 I2C_CALLBK_T 类。
2. 声明回调函数。
3. 在 I2C ISR 中声明 I2C ROM 驱动器 ISR。
4. 使能 I2C 中断。

24.5.2 I2C 主机模式设置

I2C ROM 驱动器支持轮询和中断。主机模式支持 7 位和 10 位寻址。设置步骤如下：

1. 通过调用 i2c_get_mem_size() 函数，为 I2C ROM 驱动器分配 SRAM。
2. 通过调用 i2c_setup() 函数，建立 I2C 句柄。
3. 通过调用 the i2c_set_bitrate() 函数，设置 I2C 工作频率。

```
pI2cApi = ROM_DRIVERS_PTR->pI2CD;//setup I2C function table pointer
```

```
size_in_bytes = pI2cApi->i2c_get_mem_size();
i2c_handle = pI2cApi->i2c_setup(LPC_I2C_BASE, (uint32_t *)&I2C_Handle[0] );
error_code = pI2cApi->i2c_set_bitrate((I2C_HANDLE_T*)i2c_handle, PCLK_in_Hz, bps_in_hz);
```

24.5.3 I2C 从机模式设置

从机模式下， I2C ROM 驱动器支持轮询和中断。从机模式下，仅支持 7 位寻址。设置步骤如下：

1. 通过调用 i2c_get_mem_size() 函数，为 I2C ROM 驱动器分配 SRAM。
2. 通过调用 i2c_setup() 函数，建立 I2C 句柄。
3. 通过调用 the i2c_set_bitrate() 函数，设置 I2C 工作频率。
4. 通过调用 i2c_set_slave_addr() 函数，设置从机地址。

I2C ROM 驱动器可设置最多 4 个从机地址和 4 个地址掩码，并且还可使能通用调用地址。

4 个从机地址字节组成一个 4 字节变量。从机地址字节 0 是最低有效字节，从机地址字节 3 是最高有效字节。其他 32 位变量中的从机地址掩码字节顺序也都与此相同。从机接收模式下，所有这些地址 （如果使用掩码则为分组）将被监控匹配。如果通用调用位 （四个从机地址字节的任意一个最低有效位）已被设置，则通用调用地址 0x00 亦被监控。

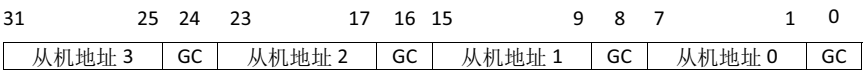


图 48. I2C 从机模式设置地址分组

```
pI2cApi = ROM_DRIVERS_PTR->pI2CD;//setup I2C function table pointer
size_in_bytes = pI2cApi->i2c_get_mem_size();
i2c_handle = pI2cApi->i2c_setup(LPC_I2C_BASE, (uint32_t *)&I2C_Handle[0] );
error_code = pI2cApi->i2c_set_bitrate((I2C_HANDLE_T*)i2c_handle, PCLK_in_Hz, bps_in_hz);
error_code = pI2cApi->i2c_set_slave_addr((I2C_HANDLE_T*)i2c_handle, slave_addr, slave_addr_mask) ;
```

24.5.4 I2C 主机发送 / 接收

主机模式驱动器供用户选择轮询 （等待消息完成）或中断驱动例程 （非阻塞式）。建议在测试或极其简单的 I2C 应用中使用轮询例程。这些例程允许主机向从机发送 7 位或 10 位地址。

下列例程为轮询例程：

```
err_code i2c_master_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_master_receive_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_master_tx_rx_poll (I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

下列例程为中断驱动例程：

```
err_code i2c_master_transmit_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_master_receive_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

err_code i2c_master_tx_rx_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

其中：

- err_code 是函数的返回状态。 “0” 表示成功。所有非零均表示错误。参见错误表。
- I2C_PARM* 为参数传递给函数的结构。请参阅章节 24.4.22。
- I2C_RESULT* 包含函数执行后的结果。

要发起主机模式的写 / 读操作，必须设置 I2C_PARAM。 I2C_PARAM 是 I2C ROM 驱动器正确工作所需的结构，含有各种变量。该结构由以下部分组成：

- 要发送的字节数。
- 要接收的字节数。
- 指向发送缓冲区的指针。
- 指向接收缓冲区的指针。
- 指向回调函数的指针。
- 停止标志。

RESULT 结构包含函数执行后的结果。该结构由以下部分组成：

- 发送的字节数。
- 接收的字节数。

注：发送的字节数更新 i2c_master_transmit_intr() 和 i2c_master_transmit_poll()。接收的字节数仅更新 i2c_master_receive_poll()、i2c_master_receive_intr()、i2c_master_tx_rx_poll() 和 i2c_master_tx_rx_intr()。

在所有主机模式例程中，发送缓冲区的第一个字节必须是从机地址，且 R/W 位设为 0。要使能主机读操作，接收缓冲区的第一个字节必须是从机地址，且 R/W 位设为 1。

在主机模式下使用 I2C 驱动器例程必须满足下列条件：

- 对于 7 位寻址，发送缓冲区的第一个字节必须在 7 位最高有效位中包含从机地址，且最低有效 (R/W) 位为 0。示例：从机地址 0x53，第一个字节为 0xA6。
- 对于 7 位寻址，接收缓冲区的第一个字节必须在 7 位最高有效位中包含从机地址，且最低有效 (R/W) 位为 1。示例：从机地址 0x53，第一个字节为 0xA7。
- 对于 10 位地址，发送缓冲区的第一个字节必须在 2 位最高有效位中包含从机地址，且 (R/W) 位为 0。第二个字节必须包含其余 8 位从机地址。
- 对于 10 位地址，接收缓冲区的第一个字节必须在 2 位最高有效位中包含从机地址，且 (R/W) 位为 1。第二个字节必须包含其余 8 位从机地址。
- 待发送的字节数应当包含缓冲区的第一个字节，即从机地址字节。示例：2 个数据字节 + 7 位从机地址 = 3。
- 应用程序必须使能I2C中断。发生I2C中断时，必须从该应用程序中调用i2c_isr_handler函数。

使用中断函数调用时，必须定义回调函数。一旦完成 PARAM 结构指定的读 / 写操作，回调函数即被调用。

24.5.5 I2C 从机模式发送 / 接收

从机模式下，轮询例程用于测试。由用户决定是使用轮询模式还是中断驱动模式。尽管在轮询模式下操作从机驱动器有利于程序开发和调试，但大多数应用程序需要在最终软件中使用从机接收和发送的中断驱动版本。

下列例程为轮询例程：

```
err_code i2c_slave_receive_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_slave_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

下列例程为中断驱动例程：

```
err_code i2c_slave_receive_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_slave_transmit_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

其中：

- err_code 是函数的返回状态。 0 表示成功。所有非零均表示错误。参见错误码表。
- I2C_PARM 为参数传递给函数的结构。章节 24.4.22
- I2C_RESULT 包含函数执行后的结果。章节 24.4.22

要发起主机模式的写 / 读操作，必须设置 I2C_PARAM。 I2C_PARAM 是 I2C ROM 驱动器正确工作所需的结构，含有各种变量。该结构由以下部分组成：

- 要发送的字节数。
- 要接收的字节数。
- 指向发送缓冲区的指针。
- 指向接收缓冲区的指针。
- 指向回调函数的指针。
- 停止标志。

RESULT 结构包含函数执行后的结果。该结构由以下部分组成：

- 发送的字节数。
- 接收的字节数。

注：发送的字节数仅更新 i2c_slave_send_poll() 和 i2c_slave_send_intr()。接收的字节数仅更新 i2c_slave_receive_poll() 和 i2c_slave_receive_intr()。

要发起从机模式通信，需调用接收函数。它可以是轮询函数，也可以是中断驱动函数，分别为 i2c_slave_receive_poll() 或 i2c_slave_receive_intr()。接收缓冲区应当不小于接收的任何数据或命令。如果数据的量超出接收缓冲区大小，则会返回错误码。

从机接收模式下，驱动器接收数据，直至下列情况之一为真：

- set_slave_addr() 函数中设置的地址匹配，其 R/W 位为 1
- 接收到 STOP 或重复的 START
- 检测到错误条件

使用中断函数调用时，必须定义回调函数。一旦完成 PARAM 结构指定的读 / 写操作，回调函数即被调用。

24.5.6 I2C 超时功能

```
//timeout:Timeout time value.Specifies the timeout interval value in increments of
// 16 I2C function clocks (Min value is 16).
if timeout = 0,  timeout feature is disabled
//        if timeout != 0, time value is timeout*16 i2c function clock.
ErrorCode_t i2c_set_timeout(I2C_HANDLE_T*  h_i2c, uint32_t timeout)
{
  I2C_DRIVER_TypeDef *h ;// declare pointer to i2c structure [handle]
  h = (I2C_DRIVER_TypeDef*) h_i2c ;   //assign handle pointer address
        if (timeout != 0){
                h->i2c_base->TimeOut = (timeout - 1)<<4;
                // Enable timeout feature
                h->i2c_base->CFG |= BI2C_TIMEOUT_EN;
        }
        else
                // disable timeout feature
                h->i2c_base->CFG &= ~BI2C_TIMEOUT_EN;

        return(LPC_OK) ;
}//i2c_set_timeout
```

25.1 本章导读

所有 LPC800 器件都提供 USART ROM 驱动器例程。

25.2 特性

- 在异步或同步模式下发送和接收字符
- 在异步或同步 UART 模式下发送和接收多个字符 （行）

25.3 简介

UART API 句柄在异步模式下使用任意 USART 模块发送和接收字符。

注：由于所有 USART 共用同一个小数分频器， uart_init 例程会返回共用分频器值。

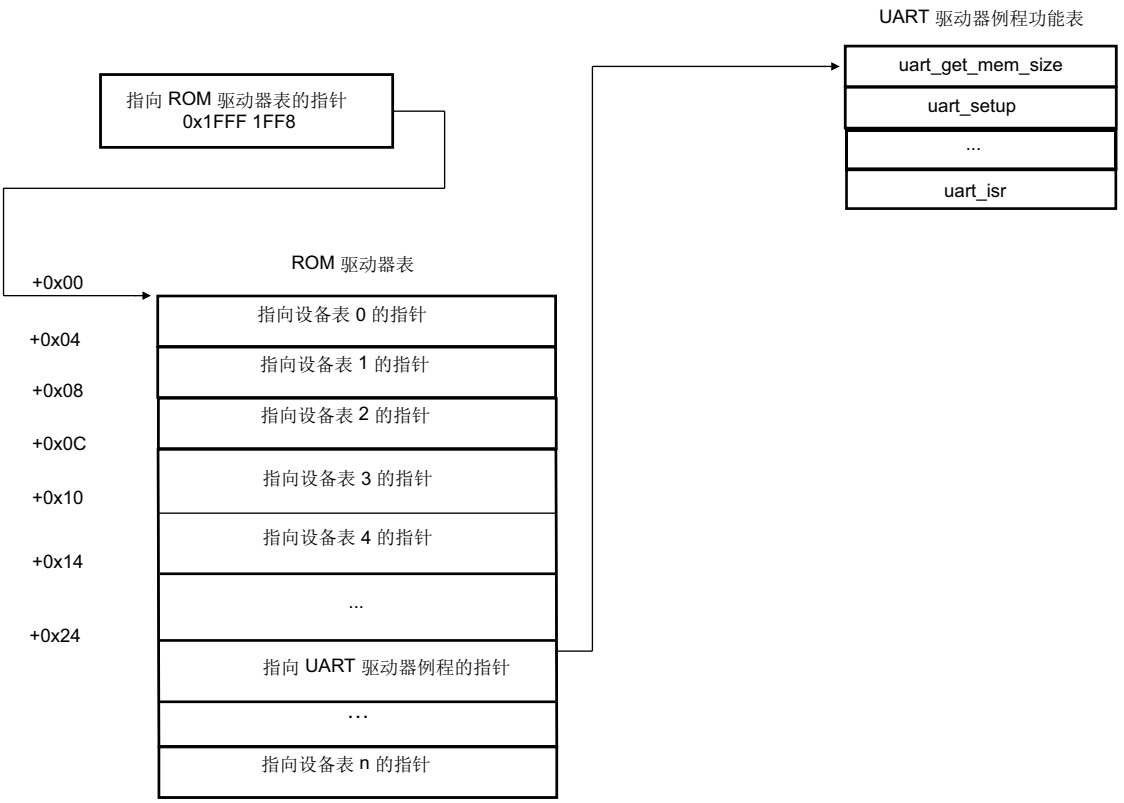


图 49. USART 驱动器例程指针结构

25.4 API 说明

UART API 含有可通过任何 USART 模块发送和接收字符的函数。

表 279. UART API 调用

API 调用	说明	参考
uint32_t ramsize_in_bytes uart_get_mem_size(void) ;	UART 获取存储器大小	表 280
UART_HANDLE_T* uart_setup(uint32_t base_addr, uint8_t *ram) ;	UART 设置	表 281
uint32_t uart_init(UART_HANDLE_T* handle, UART_CONFIG set);	UART init	表 282
uint8_t uart_get_char(UART_HANDLE_T* handle);	UART 获取字符	表 283
void uart_put_char(UART_HANDLE_T* handle, uint8_t data);	UART 放置字符	表 284
uint32_t uart_get_line(UART_HANDLE_T* handle, UART_PARAM_T param);	UART 获取行	表 285
uint32_t uart_put_line(UART_HANDLE_T* handle, UART_PARAM_T param);	UART 放置行	表 286
void uart_isr(UART_HANDLE_T* handle);	UART 中断服务例程	表 287

必须定义下列结构，才可使用 UART API：

```
typedef struct  UARTD_API {        // index of all the uart driver functions
            uint32_t (*uart_get_mem_size)(void);
            UART_HANDLE_T (*uart_setup)(uint32_t base_addr, uint8_t *ram);
            uint32_t (*uart_init)(UART_HANDLE_T handle, UART_CONFIG_T *set);
            //--polling functions--//
            uint8_t (*uart_get_char)(UART_HANDLE_T handle);
            void (*uart_put_char)(UART_HANDLE_T handle, uint8_t data);
            uint32_t (*uart_get_line)(UART_HANDLE_T handle, UART_PARAM_T * param);
            uint32_t (*uart_put_line)(UART_HANDLE_T handle, UART_PARAM_T * param);
            //--interrupt functions--//
            void (*uart_isr)(UART_HANDLE_T handle);
} UARTD_API_T  ;           // end of structure
```

25.4.1 UART 获取存储器大小

表 280. uart_get_mem_size

例程	uart_get_mem_size
原型	uint32_t ramsize_in_bytes uart_get_mem_size(void) ;
输入参数	无。
返回	存储器大小 （单位：字节）。
说明	以一个 UART 实例获取存储器大小。

25.4.2 UART 设置

表 281. uart_setup

例程	uart_setup
原型	UART_HANDLE_T* uart_setup(uint32_t base_addr, uint8_t *ram) ;
输入参数	base_addr：该 UART 模块的寄存器基址。 ram：指向 UART 实例存储器空间的指针。可通过 uart_get_mem_size 函数获取存储器空间大小。
返回	相应 UART 实例的句柄。
说明	用给定存储器设置 UART 实例并将句柄返回该实例。

25.4.3 UART init

表 282. uart_init

例程	uart_init
原型	uint32_t uart_init(UART_HANDLE_T* handle, UART_CONFIG set);
输入参数	handle：UART 实例句柄。 set：UART 操作配置。
返回	如果系统时钟不是波特率的整数倍，则返回小数分频器值。
说明	设置 UART 波特率和工作模式，然后使能 UART。

25.4.4 UART 获取字符

表 283. uart_get_char

例程	uart_get_char
原型	uint8_t uart_get_char(UART_HANDLE_T* handle);
输入参数	handle：UART 实例句柄。
返回	已接收数据
说明	从 UART 接收一个字符。该函数仅在接收字符后才返回。如果 “ 应答 ” 使能，则会立即发送已接收数据。

25.4.5 UART 放置字符

表 284. uart_put_char

例程	uart_put_char
原型	void uart_put_char(UART_HANDLE_T* handle, uint8_t data);
输入参数	handle：UART 实例句柄。 data：待发送数据。
返回	无。
说明	通过 UART 发送一个字符。该函数仅在数据发送后才返回。

25.4.6 UART 获取行

表 285. uart_get_line

例程	uart_get_line
原型	uint32_t uart_get_line(UART_HANDLE_T* handle, UART_PARAM_T param);
输入参数	handle： UART 实例句柄。 param： 参考 UART_PARAM_T 定义。
返回	错误码： ERR_UART_RECEIVE_ON - UART 正在接收。
说明	从 UART 接收多个字节。

25.4.7 UART 放置行

表 286. uart_put_line

例程	uart_put_line
原型	uint32_t uart_put_line(UART_HANDLE_T* handle, UART_PARAM_T param);
输入参数	handle： UART 实例句柄。 param： 参考 UART_PARAM_T 定义。
返回	错误码： ERR_UART_SEND_ON - UART 正在发送。
说明	通过 UART 发送字符串 （以 “\0” 结束）或原始数据。

25.4.8 UART 中断服务例程

表 287. uart_isr

例程	uart_isr
原型	void uart_isr(UART_HANDLE_T* handle);
输入参数	handle：UART 实例句柄。
返回	无。
说明	UART 中断服务例程。要使用该例程，必须使能相应的 USART 中断。该功能由用户 ISR 调用。

25.4.9 错误码

表 288. 错误码

返回码	错误码	说明
0x0008 0001	ERR_UART_RXD_BUSY = ERR_UART_BASE+1,	UART 接收忙碌
0x0008 0002	ERR_UART_TXD_BUSY	UART 发送忙碌
0x0008 0003	ERR_UART_OVERRUN_FRAME_PARITY_NOISE	溢出错误，帧错误，奇偶校验错误，RxNoise 错误
0x0008 0004	ERR_UART_UNDERRUN	欠载错误
0x0008 0005	ERR_UART_PARAM	参数错误

25.4.10 UART ROM 驱动器变量

25.4.10.1 UART_CONFIG 结构

```
typdef  struct UART_CONFIG {
        uint32_t sys_clk_in_hz; // Sytem clock in hz.
        uint32_t baudrate_in_hz; // Baudrate in hz
        uint8_t  config; //bit 1:0
                        // 00:7 bits length, 01:8 bits lenght, others:reserved
                        //bit3:2
                        //   00:No Parity, 01:reserved, 10:Even, 11:Odd
                        //bit4
                        //   0:1 Stop bit, 1:2 Stop bits
        uint8_t sync_mod;  //bit0:0(Async mode), 1(Sync mode)
                                //bit1:0(Un_RXD is sampled on the falling edge of SCLK)
                                //       1(Un_RXD is sampled on the rising edge of SCLK)
                                //bit2:0(Start and stop bits are transmitted as in asynchronousmode)
                                //       1(Start and stop bits are not transmitted)
                                //bit3:0(the UART is a slave on Sync mode)
                                //       1(the UART is a master on Sync mode)
uint16_t error_en; //Bit0:OverrunEn, bit1:UnderrunEn, bit2:FrameErrEn,
                        //bit3:ParityErrEn, bit4:RxNoiseEn
}
```

25.4.10.2 UART_HANDLE_T

UART 驱动器实例的句柄。每个 UART 都有一个句柄，因此针对最多三个 UART 模块可有多个句柄。该句柄由 INIT API 创建，并用于传输函数的相应 UART 模块中。

```
typedef  void    *UART_HANDLE_T   ;    // define TYPE for uart handle pointer
```

25.4.10.3 UART_PARAM_T

```
typedef struct  uart_A {// parms passed to uart driver function
uint8_t  *  buffer ; // The pointer of buffer.
                                // For uart_get_line function, buffer for receiving data.
                                // For uart_put_line function, buffer for transmitting data.
uint32_t   size;     // [IN] The size of buffer.
                                //[OUT] The number of bytes transmitted/received.
uint16_t    transfer_mode ;
                                // 0x00:For uart_get_line function, transfer without
                                // termination.
                                // For uart_put_line function, transfer without termination.
                                // 0x01:For uart_get_line function, stop transfer when
                                // <CR><LF> are received.
                                // For uart_put_line function, transfer is stopped after
                                // reaching \0.<CR><LF> characters are sent out after that.
                                // 0x02:For uart_get_line function, stop transfer when <LF>
                                // is received.
                                // For uart_put_line function, transfer is stopped after
                                // reaching \0.A <LF> character is sent out after that.
                                //0x03:For uart_get_line function, RESERVED.
                                // For uart_put_line function, transfer is stopped after
```

```
                                // reaching \0.
uint16_t   driver_mode;
                                // 0x00:Polling mode, function is blocked until transfer is
                                // finished.
                                // 0x01:Intr mode, function exit immediately, callback function
                                // is invoked when transfer is finished.
                                // 0x02: 保留
UART_CALLBK_T    callback_func_pt;// callback function
} UART_PARAM_T ;
```

26.1 本章导读

所有 LPC800 器件的调试功能都是一样的。

26.2 特性

- 支持 ARM 串行线调试模式。
- 可直接对所有存储器、寄存器和外设进行调试。
- 调试会话无需目标资源。
- 4 个断点。
- 两个数据观察点，也可用作触发器。
- 支持 JTAG 边界扫描。
- 支持微跟踪缓冲区 (MTB)。

26.3 简介

ARM Cortex-M0+ 集成了调试功能。支持串行线调试功能。ARM Cortex-M0+ 经过配置后可支持多达 4 个断点和 2 个观察点。

支持边界扫描和微跟踪缓冲区。

26.4 引脚说明

SWD 功能通过开关矩阵分配给引脚。 SWD 功能为固定引脚功能，通过开关矩阵使能，并且只能分配给封装上的特定引脚。默认情况下， SWD 功能为使能。

有关如何使能模拟比较器输入和基准电压输入，请参见章节 9.3.2。

表 289. SWD 引脚说明

功能	类型	引脚	说明	SWM 寄存器	参考
SWCLK	I/O	SWCLK/PIO0_3/ TCLK	**串行线时钟。**该引脚在串行线调试 (SWD) 模式下是 SWD 调试逻辑的时钟。该引脚内部上拉。	PINENABLE0	表 106
SWDIO	I/O	SWDIO/PIO0_2/ TMS	**串行线调试数据输入 / 输出。**外部调试工具使用 SWDIO 引脚与 LPC800 进行通信并对其进行控制。该引脚内部上拉。	PINENABLE0	表 106

边界扫描模式和所需的引脚通过硬件选择 （参见章节 26.5.2）。无法通过开关矩阵访问边界扫描引脚。

表 290. JTAG 边界扫描引脚说明

功能	引脚名称	类型	说明
TCK	SWCLK/PIO0_3/ TCK	I	**JTAG 测试时钟。** RESET 引脚为低电平时，该引脚是 JTAG 边界扫描的时钟。
TMS	SWDIO/PIO0_2/ TMS	I	**JTAG 测试模式选择。**TMS 引脚可选择 TAP 状态机中的下一状态。RESET 引脚为低电平时，该引脚含有内部上拉信号，并用于 JTAG 边界扫描。
TDI	PIO0_1/ACMP_I2/ CLKIN/TDI	I	**JTAG 测试数据输入。**这是移位寄存器的串行数据输入。 RESET 引脚为低电平时，该引脚含有内部上拉信号，并用于 JTAG 边界扫描。
TDO	PIO0_0/ACMP_I1/ TDO	O	**JTAG 测试数据输出。**这是移位寄存器的串行数据输出。数据在 TCK 信号的负边沿移出设备。 RESET 引脚为低电平时，该引脚用于 JTAG 边界扫描。
TRST	PIO0_4/ WAKEUP/TRST	I	**JTAG 测试复位。**TRST 引脚可用于复位调试逻辑内的测试逻辑。RESET 引脚为低电平时，该引脚含有内部上拉信号，并用于 JTAG 边界扫描。

26.5 功能说明

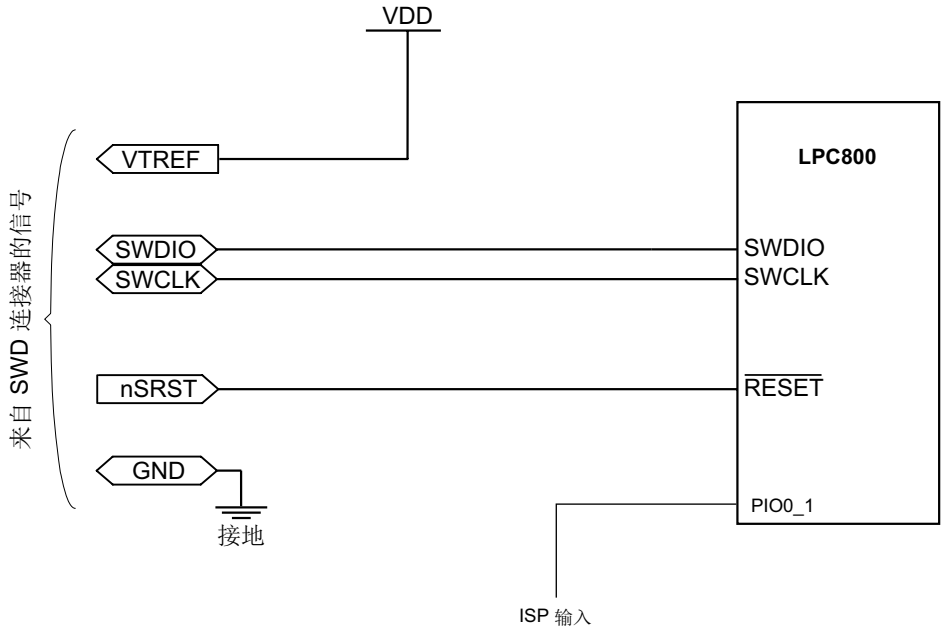
26.5.1 调试限制

建议在深度睡眠模式或掉电模式下不要使用调试模式。

在调试会话期间，每当 CPU 停止时系统节拍定时器就会自动停止。其他外设不受影响。

26.5.2 SWD 的调试连接

就调试而言，提供对 ISP 输入引脚 PIO0_1 的访问非常有用。该引脚可用于从会禁用 SWD 端口的不当 PLL 配置、SWD 引脚重新配置、退出复位后进入深度掉电模式等配置中恢复器件。该引脚可用于 GPIO 等其他功能，但在上电或复位时不应将其保持为低电平。



SWD 连接器上的 VTREF 引脚可使能调试连接器来匹配目标电压。

图 50. 将 SWD 引脚连接到标准的 SWD 连接器

26.5.3 边界扫描

RESET 引脚可在 JTAG 边界扫描模式（RESET = 低电平）和 ARM SWD 调试模式（RESET = 高电平）之间进行选择。器件复位时， ARM SWD 调试端口被禁用。

要执行边界扫描测试，可通过以下这些步骤：

1. 擦除驻留在闪存中的所有用户代码。
2. 通过在外部将 RESET 引脚拉至高电平使器件上电。
3. 等待至少 250 ms。
4. 在外部将 RESET 引脚拉至低电平。
5. 执行边界扫描操作。
6. 一旦完成边界扫描操作，则通过置位 TRST 引脚来使能 SWD 调试模式和释放 RESET 引脚 （拉至高电平）。

注：JTAG 接口无法用于调试。

注：POR、 BOD 复位，或 TRST 引脚上具有低电平都将使测试 TAP 控制器进入测试逻辑复位状态。 RESET = 高电平时的第一个 TCK 时钟可使测试 TAP 进入运行测试闲置模式。

26.5.4 微跟踪缓冲区 (MTB)

MTB 寄存器位于存储器地址 0x1400 0000，如参考 2 所述。 SYSCON 模块 （参见章节 4.6.20）中的 EXTTRACE 寄存器配合 MTB MASTER 寄存器中的 TSTARTEN 和 TSTOPEN 位启动并停止跟踪。跟踪结果保存在本地 SRAM 中，起始地址为 0x1000 0000。跟踪存储器位置在 MTB POSITION 寄存器中配置。

注：MTB BASE 寄存器未实施。对 BASE 寄存器进行读操作可返回 0x0，而与针对跟踪而配置的 SRAM 存储器区域无关。

27.1 封装

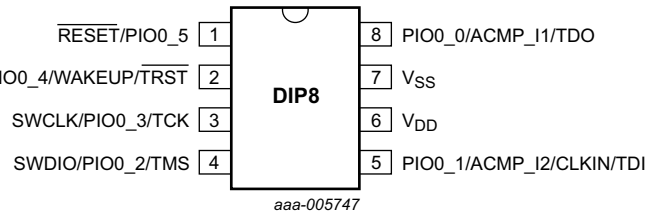


图 51. DIP8 封装 (LPC810M021FN8) 引脚配置

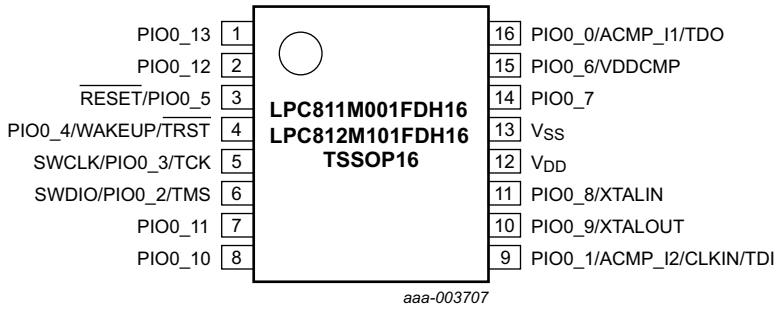


图 52. TSSOP16 封装引脚配置

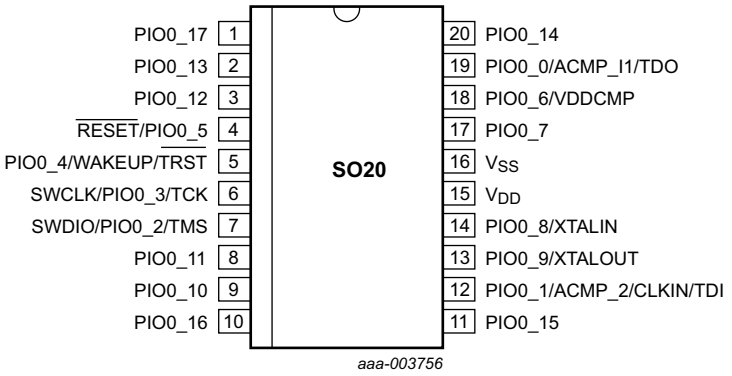


图 53. SO20 封装 (LPC812M101FD20) 引脚配置

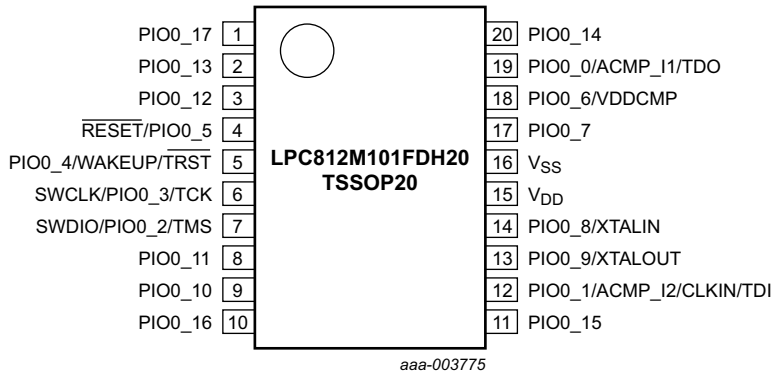


图 54. TSSOP20 封装引脚配置

27.2 引脚说明

引脚说明表表 291 显示的是每种封装特定引脚的固定引脚功能。这些固定引脚功能可在 GPIO、比较器、SWD 和 XTAL 引脚间进行选择。除引脚 PIO0_2、PIO0_3 和 PIO0_5 外，这些引脚均默认选择 GPIO 功能。 JTAG 功能仅在边界扫描模式下可用。

I2C、USART、SPI 和 SCT 的可转移引脚功能，可通过开关矩阵分配给除电源或接地外的任意引脚，代替引脚的固定功能。

下列例外情况适用：

为了完全兼容 I2C 总线，将 I2C 功能分配给开漏引脚 PIO0_11 和 PIO0_10。

任何引脚都不要分配一个以上的输出。但允许为引脚分配一个以上的输入。

引脚 PIO0_4 可触发从深度掉电模式的唤醒。如需通过外部引脚从深度掉电模式唤醒，则不要将任何可转移功能分配给该引脚。

器件处于边界扫描模式时，引脚 PIO0_0 至 PIO0_4 通过硬件选择 TDO、TDI、TCK、TMS 和 TRST 等 JTAG 功能。

表 291. 引脚说明表 （固定引脚）

符号	SO20/ TSSOP20	TSSOP16	DIP8		类型	复位状态 [1]	说明
PIO0_0/ACMP_I1/ TDO	19	16	8	[5]	I/O	I；PU	**PIO0_0 —** 通用数字输入 / 输出端口 0 引脚 0。 在 ISP 模式下，它是 USART0 接收引脚 U0_RXD。 在边界扫描模式下：TDO （测试数据输出）。
					AI	-	**ACMP_I1 —** 模拟比较器输入 1。
PIO0_1/ACMP_I2/ CLKIN/TDI	12	9	5	[5]	I/O	I；PU	**PIO0_1 —** 通用数字输入 / 输出引脚。 ISP 输入引脚。复位期间，该引脚为低电平时，会启动 ISP 命令处理程序。 在边界扫描模式下：TDI （测试数据输入）。
					AI	-	**ACMP_I2 —** 模拟比较器输入 2。
					I	-	**CLKIN —** 外部时钟输入。

表 291. 引脚说明表 （固定引脚）*（续）*

符号	SO20/ TSSOP20	TSSOP16	DIP8		类型	复位状态 [1]	说明
SWDIO/PIO0_2/TMS	7	6	4	[2]	I/O	I；PU	**SWDIO —** 串行线调试 I/O。 SWDIO 默认在该引脚上使能。在边界扫描模式下：TMS （测试模式选择）。
					I/O	-	**PIO0_2 —** 通用数字输入 / 输出引脚。
SWCLK/PIO0_3/ TCK	6	5	3	[2]	I/O	I；PU	**SWCLK —** 串行线时钟。 SWCLK 默认在该引脚上使能。在边界扫描模式下：TCK （测试时钟）。
					I/O	-	**PIO0_3 —** 通用数字输入 / 输出引脚。
PIO0_4/WAKEUP/ TRST	5	4	2	[6]	I/O	I；PU	**PIO0_4 —** 通用数字输入 / 输出引脚。 在 ISP 模式下，它是 USART0 发送引脚 U0_TXD。 在边界扫描模式下：$\overline{\text{TRST}}$ （测试复位）。 该引脚可触发从深度掉电模式的唤醒。如需通过外部引脚从深度掉电模式唤醒，则不要将任何可转移功能分配给该引脚。从外部将该引脚拉至高电平便可进入深度掉电模式。将该引脚拉至低电平便可退出深度掉电模式。持续时间低至 50 ns 的低电平脉冲可唤醒器件。
$\overline{\text{RESET}}$/PIO0_5	4	3	1	[4]	I/O	I；PU	**RESET —** 外部复位输入：该引脚上持续时间低至 50 ns 的低电平脉冲可复位器件，从而导致 I/O 端口和外围设备采用其默认状态，并且处理器从地址 0 开始执行。
					I	-	**PIO0_5 —** 通用数字输入 / 输出引脚。
PIO0_6/VDDCMP	18	15	-	[9]	I/O	I；PU	**PIO0_6 —** 通用数字输入 / 输出引脚。
					AI	-	**VDDCMP —** 模拟比较器的备用基准电压。
PIO0_7	17	14	-	[2]	I/O	I；PU	**PIO0_7 —** 通用数字输入 / 输出引脚。
PIO0_8/XTALIN	14	11	-	[8]	I/O	I；PU	**PIO0_8 —** 通用数字输入 / 输出引脚。
					I	-	**XTALIN —** 振荡器电路和内部时钟发生器电路的输入。输入电压不得超过 1.95 V。
PIO0_9/XTALOUT	13	10	-	[8]	I/O	I；PU	**PIO0_9 —** 通用数字输入 / 输出引脚。
					O	-	**XTALOUT —** 振荡器电路的输出。
PIO0_10	9	8	-	[3]	I	IA	**PIO0_10 —** 通用数字输入 / 输出引脚。当需要使用真正的开漏引脚，以使信号完全符合 I2C 规范时，可将 I2C 功能分配给该引脚。
PIO0_11	8	7	-	[3]	I	IA	**PIO0_11 —** 通用数字输入 / 输出引脚。当需要使用真正的开漏引脚，以使信号完全符合 I2C 规范时，可将 I2C 功能分配给该引脚。
PIO0_12	3	2	-	[2]	I/O	I；PU	**PIO0_12 —** 通用数字输入 / 输出引脚。
PIO0_13	2	1	-	[2]	I/O	I；PU	**PIO0_13 —** 通用数字输入 / 输出引脚。
PIO0_14	20	-	-	[7]	I/O	I；PU	**PIO0_14 —** 通用数字输入 / 输出引脚。
PIO0_15	11	-	-	[7]	I/O	I；PU	**PIO0_15 —** 通用数字输入 / 输出引脚。
PIO0_16	10	-	-	[7]	I/O	I；PU	**PIO0_16 —** 通用数字输入 / 输出引脚。
PIO0_17	1	-	-	[7]	I/O	I；PU	**PIO0_17 —** 通用数字输入 / 输出引脚。
V_{DD}	15	12	6		-	-	3.3 V 电源电压。
V_{SS}	16	13	7			-	地线。

[1] 复位后默认功能的引脚状态：I = 输入； AI = 模拟输入； O = 输出； PU = 内部上拉电阻使能 （引脚上拉至满量程 V_{DD} 电平）； IA = 无源，上拉电阻 / 下拉电阻未使能。

[2] 5 V 容压焊盘，提供数字 I/O 功能，带可配置上拉电阻 / 下拉电阻和可配置迟滞；集成了高电流输出驱动器。

[3] 真正的开漏引脚。 I^2C 总线引脚，符合 I^2C 总线规范，支持 I^2C 标准模式、 I^2C 快速模式和 I^2C 超快速模式。不要在 SPI 时钟等高速应用中使用该焊盘。

[4] 深度掉电模式下，RESET 功能不可用。使用 WAKEUP 引脚进行芯片复位，并从深度掉电模式唤醒。在深度掉电模式下，需要在该引脚上安装一个外部上拉电阻。

[5] 5 V 容压的引脚，提供标准数字 I/O 功能，带可配置模式、可配置迟滞和模拟输入。当配置为模拟输入时，引脚的数字部分被禁用，并且该引脚非 5 V 容压。

[6] 5 V 容压的焊盘，提供数字 I/O 功能，带可配置上拉电阻 / 下拉电阻和可配置迟滞。在深度掉电模式下，将该引脚拉至低电平可唤醒芯片。

[7] 5 V 容压的焊盘，提供数字 I/O 功能，带可配置上拉电阻 / 下拉电阻和可配置迟滞。

[8] 5 V 容压的引脚，提供标准数字 I/O 功能，带可配置模式、可配置迟滞和用于系统振荡器的模拟 I/O。当配置为模拟 I/O 时，引脚的数字部分被禁用，并且该引脚非 5 V 容压。

[9] 由于具有特殊的模拟功能，它并非 5 V 容压引脚。引脚提供带可配置模式、可配置迟滞和模拟 I/O 的标准数字 I/O 功能。当配置为模拟 I/O 时，引脚的数字部分被禁用。

表 292. 可转移功能 （通过开关矩阵分配给引脚 PIO0_0 至 PIO_17）

功能名称	类型	说明
U0_TXD	O	USART0 的发送器输出。
U0_RXD	I	USART0 的接收器输入。
U0_RTS	O	USART0 的请求发送输出。
U0_CTS	I	USART0 的清零发送输入。
U0_SCLK	I/O	同步模式下 USART0 的串行时钟输入 / 输出。
U1_TXD	O	USART1 的发送器输出。
U1_RXD	I	USART1 的接收器输入。
U1_RTS	O	USART1 的请求发送输出。
U1_CTS	I	USART1 的清零发送输入。
U1_SCLK	I/O	同步模式下 USART1 的串行时钟输入 / 输出。
U2_TXD	O	USART2 的发送器输出。
U2_RXD	I	USART2 的接收器输入。
U2_RTS	O	USART2 的请求发送输出。
U2_CTS	I	USART2 的清零发送输入。
U2_SCLK	I/O	同步模式下 USART2 的串行时钟输入 / 输出。
SPI0_SCK	I/O	SPI0 的串行时钟。
SPI0_MOSI	I/O	SPI0 的主机输出从机输入。
SPI0_MISO	I/O	SPI0 的主机输入从机输出。
SPI0_SSEL	I/O	SPI0 的从机选择。
SPI1_SCK	I/O	SPI1 的串行时钟。
SPI1_MOSI	I/O	SPI1 的主机输出从机输入。
SPI1_MISO	I/O	SPI1 的主机输入从机输出。
SPI1_SSEL	I/O	SPI1 的从机选择。
CTIN_0	I	SCT 输入 0。
CTIN_1	I	SCT 输入 1。
CTIN_2	I	SCT 输入 2。
CTIN_3	I	SCT 输入 3。
CTOUT_0	O	SCT 输出 0。
CTOUT_1	O	SCT 输出 1。
CTOUT_2	O	SCT 输出 2。
CTOUT_3	O	SCT 输出 3。

表 292. 可转移功能 （通过开关矩阵分配给引脚 PIO0_0 至 PIO_17）

功能名称	类型	说明
I2C0_SCL	I/O	I^2C 总线时钟输入 / 输出 （如果分配给引脚 PIO0_10，则处于开漏状态）。仅当分配给引脚 PIO0_10 且在 I/O 配置寄存器中选择 I^2C 超快速模式时，才会有大电流灌入。
I2C0_SDA	I/O	I^2C 总线数据输入 / 输出（若分配给引脚 PIO0_11，则处于开漏状态）。仅当分配给引脚 PIO0_11 且在 I/O 配置寄存器中选择 I^2C 超快速模式时，才会有大电流灌入。
ACMP_O	O	模拟比较器输出。
CLKOUT	O	时钟输出。
GPIO_INT_BMAT	O	模式匹配引擎输出。

28.1 本章导读

本章总结了 ARM Cortex-M0+ 指令。所有 LPC800 器件的指令集都是一样的。

28.2 简介

该处理器执行 ARMv6-M Thumb 指令集，包括大量使用 Thumb-2 技术的 32 位指令。ARMv6-M 指令集包括：

- 除 CBZ、 CBNZ 和 IT 外的所有 ARMv7-M 16 位 Thumb 指令。
- 32 位 Thumb 指令 BL、 DMB、 DSB、 ISB、 MRS 和 MSR。

表 293 显示的是 Cortex-M0+ 指令及其周期数。周期数基于的是具有零等待状态的系统。

表 293. Cortex M0- 指令汇总

操作	说明	汇编程序	周期
传输	8 位立即数	MOVS Rd, #<imm>	1
	Lo 至 Lo	MOVS Rd, Rm	1
	任意至任意	MOV Rd, Rm	1
	任意至 PC	MOV PC, Rm	2
加法	3 位立即数	ADDS Rd, Rn, #<imm>	1
	所有寄存器 Lo	ADDS Rd, Rn, Rm	1
	任意至任意	ADD Rd, Rd, Rm	1
	任意至 PC	ADD PC, PC, Rm	2
	8 位立即数	ADDS Rd, Rd, #<imm>	1
	带进位	ADCS Rd, Rd, Rm	1
	立即数至 SP	ADD SP, SP, #<imm>	1
	从 SP 形成地址	ADD Rd, SP, #<imm>	1
	从 PC 形成地址	ADR Rd, <label>	1
减法	Lo 与 Lo	SUBS Rd, Rn, Rm	1
	3 位立即数	SUBS Rd, Rn, #<imm>	1
	8 位立即数	SUBS Rd, Rd, #<imm>	1
	带进位	SBCS Rd, Rd, Rm	1
	来自 SP 的立即数	SUB SP, SP, #<imm>	1
	取反	RSBS Rd, Rn, #0	1
乘法	乘法	MULS Rd, Rm, Rd	1
比较	比较	CMP Rn, Rm	1
	负数	CMN Rn, Rm	1
	立即数	CMP Rn, #<imm>	1

表 293. Cortex M0- 指令汇总 *（续）*

操作	说明	汇编程序	周期
逻辑	与	ANDS Rd, Rd, Rm	1
	异或	EORS Rd, Rd, Rm	1
	或	ORRS Rd, Rd, Rm	1
	位清零	BICS Rd, Rd, Rm	1
	取反传输	MVNS Rd, Rm	1
	“ 与 ” 测试	TST Rn, Rm	1
移位	立即数逻辑左移	LSLS Rd, Rm, #<shift>	1
	寄存器逻辑左移	LSLS Rd, Rd, Rs	1
	立即数逻辑右移	LSRS Rd, Rm, #<shift>	1
	寄存器逻辑右移	LSRS Rd, Rd, Rs	1
	算术右移	ASRS Rd, Rm, #<shift>	1
	寄存器算术右移	ASRS Rd, Rd, Rs	1
旋转	寄存器右转	RORS Rd, Rd, Rs	1
加载	字，立即数偏移	LDR Rd, [Rn, #<imm>]	2 或 1[2]
	半字，立即数偏移	LDRH Rd, [Rn, #<imm>]	2 或 1[2]
	字节，立即数偏移	LDRB Rd, [Rn, #<imm>]	2 或 1[2]
	字，寄存器偏移	LDR Rd, [Rn, Rm]	2 或 1[2]
	半字，寄存器偏移	LDRH Rd, [Rn, Rm]	2 或 1[2]
	带符号半字，寄存器偏移	LDRSH Rd, [Rn, Rm]	2 或 1[2]
	字节，寄存器偏移	LDRB Rd, [Rn, Rm]	2 或 1[2]
	带符号字节，寄存器偏移	LDRSB Rd, [Rn, Rm]	2 或 1[2]
	PC 相对	LDR Rd, <label>	2 或 1[2]
	SP 相对	LDR Rd, [SP, #<imm>]	2 或 1[2]
	倍数，不包括基数	LDM Rn!, {<loreglist>}	1 + N[1]
	倍数，包括基数	LDM Rn, {<loreglist>}	1 + N[1]
存储	字，立即数偏移	STR Rd, [Rn, #<imm>]	2 或 1[2]
	半字，立即数偏移	STRH Rd, [Rn, #<imm>]	2 或 1[2]
	字节，立即数偏移	STRB Rd, [Rn, #<imm>]	2 或 1[2]
	字，寄存器偏移	STR Rd, [Rn, Rm]	2 或 1[2]
	半字，寄存器偏移	STRH Rd, [Rn, Rm]	2 或 1[2]
	字节，寄存器偏移	STRB Rd, [Rn, Rm]	2 或 1[2]
	SP 相对	STR Rd, [SP, #<imm>]	2 或 1[2]
	倍数	STM Rn!, {<loreglist>}	1 + N[1]
推入	推入	PUSH {<loreglist>}	1 + N[1]
	用链路寄存器推入	PUSH {<loreglist>, LR}	1 + N[3]
弹出	弹出	POP {<loreglist>}	1 + N[1]
	弹出并返回	POP {<loreglist>, PC}	3 + N[3]

表 293. Cortex M0- 指令汇总 *（续）*

操作	说明	汇编程序	周期
转移	条件	B<cc> <label>	1 或 2[4]
	无条件	B <label>	2
	带链路	BL <label>	3
	带交换	BX Rm	2
	带链路和交换	BLX Rm	2
扩展	带符号半字至字	SXTH Rd, Rm	1
	带符号字节至字	SXTB Rd, Rm	1
	无符号半字	UXTH Rd, Rm	1
	无符号字节	UXTB Rd, Rm	1
反转	字节 （以字为单位）	REV Rd, Rm	1
	字节 （以两个半字为单位）	REV16 Rd, Rm	1
	带符号的下半部字	REVSH Rd, Rm	1
状态变化	超级用户调用	SVC #<imm>	-[5]
	禁用中断	CPSID i	1
	使能中断	CPSIE i	1
	对特殊寄存器进行读操作	MRS Rd, <specreg>	3
	对特殊寄存器进行写操作	MSR <specreg>, Rn	3
	断点	BKPT #<imm>	-[5]
提示	发送事件	SEV	1
	等待中断	WFI	2[6]
	等待事件	WFE	2[6]
	结果	YIELD[7]	1
	无操作	NOP	1
限制	指令同步	ISB	3
	数据存储器	DMB	3
	数据同步	DSB	3

[1] N 是列表中元素的数量。

[2] 如果到 AHB 接口或 SCS，则周期数为 2；如果到单周期 I/O 端口，则周期数为 1。

[3] N 是列表中元素的数量，包括 PC 或 LR。

[4] 选中则为 2，未选中则为 1。

[5] 周期数取决于内核和调试配置。

[6] 不包括等待中断或事件的时间。

[7] 以 NOP 执行。

29.1 缩略词

表 294. 缩略词

首字母缩略词	说明
AHB	高级高性能总线
APB	高级外设总线
BOD	掉电检测
GPIO	通用输入 / 输出
PLL	锁相环
RC	电阻 - 电容
SPI	串行外设接口
SMBus	系统管理总线
TEM	横向电磁
UART	通用异步收发器

29.2 参考文献

[1] **DDI0484B_cortex_m0p_r0p0_trm** — ARM Cortex-M0+ 技术参考手册

[2] **DDI0486A** — ARM 技术参考手册

[3] DRMv6-M 架构参考手册

29.3 法律信息

29.3.1 定义

初稿 — 本文仅为初稿版本。内容仍在内部审查，尚未正式批准，可能会有进一步修改或补充。恩智浦半导体对本文信息的准确性或完整性不做任何说明或保证，并对因使用此信息而导致的后果不承担任何责任。

29.3.2 免责声明

有限担保与责任 — 本文中的信息据信是准确和可靠的。但是，恩智浦半导体对此处所含信息的准确性或完整性不做任何明示或暗示的说明或保证，并对因使用此信息而导致的后果不承担任何责任。若文中信息并非来自恩智浦半导体，则恩智浦半导体对该信息的内容概不负责。

在任何情况下，对于任何间接、意外、惩罚性、特殊或衍生性损害（包括但不限于利润损失、积蓄损失、业务中断、因拆卸或更换任何产品而产生的开支或返工费用），无论此等损害是否基于侵权行为（包括过失）、担保、违约或任何其他法理，恩智浦半导体均不承担任何责任。

对于因任何原因给客户带来的任何损害，恩智浦半导体对本文所述产品的总计责任和累积责任仅限于恩智浦*商业销售条款和条件*所规定的范围。

修改权利 — 恩智浦半导体保留对本文所发布的信息（包括但不限于规格和产品说明）随时进行修改的权利，恕不另行通知。本文件将取代并替换之前就此提供的所有信息。

适宜使用 — 恩智浦半导体产品并非设计、授权或担保适合用于生命保障、生命关键或安全关键系统或设备，亦非设计、授权或担保适合用于在恩智浦半导体产品失效或故障时可导致人员受伤、死亡或严重财产或环境损害的应用。恩智浦半导体及其供应商对在此类设备或应用中加入和 / 或使用恩智浦半导体产品不承担任何责任，客户需自行承担因加入和 / 或使用恩智浦半导体产品而带来的风险。

应用 — 本文件所载任何产品的应用只用于例证目的。此类应用如不经进一步测试或修改用于特定用途，恩智浦半导体对其适用性不做任何说明或保证。

客户负责自行利用恩智浦半导体的产品进行设计和应用，对于应用或客户产品设计，恩智浦半导体无义务提供任何协助。客户须自行判断恩智浦半导体的产品是否适用于其应用和设计计划，以及是否适用于其第三方客户的规划应用。客户须提供适当的设计和操作安全保障措施，以降低与应用和产品相关的风险。

对于因客户应用或产品的任何缺陷或故障，或者客户的第三方客户的应用或使用导致的任何故障、损害、开支或问题，恩智浦半导体均不承担任何责任。客户负责对自己基于恩智浦半导体的产品的应用和产品进行所有必要测试，以避免这些应用和产品或者客户的第三方客户的应用或使用存在任何缺陷。恩智浦不承担与此相关的任何责任。

出口管制 — 本文件以及此处所描述的产品可能受出口法规的管制。出口可能需要事先经主管部门批准。

29.3.3 商标

注意：所有引用的品牌、产品名称、服务名称以及商标均为其各自所有者的资产。

I^2C 总线 . 标志是恩智浦的商标。

29.4 表

29.5 图

29.6 目录

第 1 章：LPC800 简介信息

第 2 章：LPC800 存储器映射

第 3 章：LPC800 可嵌套中断向量控制器 (NVIC)

第 4 章：LPC800 系统配置 (SYSCON)

第 5 章：LPC800 节能模式和电源管理单元 (PMU)

第 6 章：LPC800 I/O 配置 (IOCON)

第 7 章：LPC800 GPIO 端口

第 8 章：LPC800 引脚中断 / 模式匹配引擎

第 9 章：LPC800 开关矩阵

第 10 章：LPC800 状态可配置定时器 (SCT)

第 11 章：LPC800 多速率定时器 (MRT)

第 12 章：LPC800 窗口看门狗定时器 (WWDT)

第 13 章：LPC800 自唤醒定时器 (WKT)

第 14 章：LPC800 ARM Cortex SysTick 定时器 (SysTick)

第 15 章：LPC800 USART0/1/2

第 16 章：LPC800 I2C 总线接口

第 17 章：LPC800 SPI0/1

第 18 章：LPC800 模拟比较器

第 19 章：LPC800 循环冗余检查 (CRC) 引擎

第 20 章：LPC800 闪存控制器

第 21 章：LPC800 引导 ROM

第 22 章：LPC800 闪存 ISP 和 IAP 编程

第 23 章：LPC800 电源配置 API ROM 驱动器

第 24 章：LPC800 I2C 总线 ROM API

第 25 章：LPC800 USART API ROM 驱动器例程

第 26 章：LPC800 调试

第 27 章：LPC800 封装和引脚说明

第 28 章：LPC800 附录

第 29 章：补充信息

欲了解更多信息，请访问：http://www.nxp.com
欲咨询销售办事处地址，请发送电子邮件至：salesaddresses@nxp.com

发布日期：2013 年 3 月 14 日
文档号：UM10601