UM10524

LPC1315/16/17/45/46/47 User manual

修订版：1 — 2012 年 2 月 17 日 **用户手册**

文档信息

项目	内容
关键字	LPC1315/16/17/45/46/47、 ARM Cortex-M3、微控制器、 USB
摘要	LPC1315/16/17/45/46/47 用户手册

修订记录

版本	日期	描述
1	20120217	初始版本

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 简介

LPC1315/16/17/45/46/47 系列是基于 ARM Cortex-M3 的微控制器，适合高度集成和低功耗的嵌入式应用。 ARM Cortex-M3 是下一代微控制器内核，具有易调试、易集成等系统增强优势。

LPC1315/16/17/45/46/47 系列 CPU 工作频率高达 72 MHz。ARM Cortex-M3 内核 CPU 采用 3 级流水线和哈佛架构，具有独立的本地指令和数据总线以及用于系统外设的第三总线。此外， ARM Cortex-M3 内核 CPU 还包含一个内部预取单元支持不确定的分支操作。

LPC1345/46/47 具备高度灵活和可配置的全速 USB 2.0 设备控制器，该系列为当今要求苛刻的连接解决方案带来无与伦比的设计灵活性以及无缝集成。

LPC1315/16/17/45/46/47 上的补充外设包括高达 64 kB 闪存、8 kB 或 10 kB SRAM 数据存储器、一个超快速模式 I^2C 总线接口、一个支持同步模式和智能卡接口的 RS-485/EIA-485 USART、两个 SSP 接口、四个通用计数器 / 定时器、一个 8 通道 12 位 ADC 和最多 51 个通用 I/O 引脚。

1.2 特性

- 系统：

  ARM Cortex-M3 r2p1 处理器，工作频率高达 72 MHz。

  ARM Cortex-M3 内置可嵌套中断向量控制器 (NVIC)。

  可以从多个输入源选择非屏蔽中断 (NMI) 输入。

  系统节拍定时器。
- 存储器：
  - 最高 64 kB 片内闪存程序存储器，具有 256 字节页擦除功能。
  - 通过片内启动引导程序软件执行的在系统编程 (ISP) 和在应用编程 (IAP)。支持通过 USB 更新闪存。
  - 最高 4 kB 片内 EEPROM 数据存储器，支持片内 API。
  - 最高 12 kB SRAM 数据存储器。
  - 16 kB Boot ROM，具有 USB API 支持、功率控制、 EEPROM 和闪存 IAP/ISP。

- 调试选项：
  - 用于 BSDL 的标准 JTAG 测试接口。
  - 串行线调试。
  - 支持 ETM ARM Cortex-M3 调试时间戳。
- 数字外设：
  - 多达 51 个通用 I/O (GPIO) 引脚，上拉 / 下拉电阻、中继模式、输入逆变器和伪开漏模式可配置。 8 个引脚支持可编程干扰滤波器。
  - 最多可以选择 8 个 GPIO 引脚作为边沿和电平敏感中断源。
  - 2 个 GPIO 分组中断模块基于一组 GPIO 引脚的输入状态的可编程模式使能中断。
  - 一个引脚 (P0_7) 上的大电流源输出驱动器 (20 mA)。
  - 真正的开漏引脚 （P0_4 和 P0_5）上的大电流吸收驱动器 (20 mA)。
  - 四个通用计数器 / 定时器，总共具有多达 8 个捕获输入和 13 个匹配输出。
  - 可编程窗口化看门狗定时器 (WWDT)，包含内部低功率看门狗振荡器 (WDO)。
  - 重复中断定时器 （RI 定时器）。
- 模拟外设：
  - 12 位 ADC，具有 8 个输入通道和高达 500 kSamples/s 的采样率。
- 串行接口：
  - USB 2.0全速设备控制器(LPC1345/46/47)，具有片内基于ROM的USB驱动程序库。
  - USART，生成小数波特率，内部 FIFO，全调制解调器控制信号交换接口，支持 RS-485/9 位模式以及同步模式。 USART 支持异步智能卡接口 (ISO 7816-3)。
  - 两个 SSP 控制器，搭载 FIFO 和多协议功能。
  - I^2C 总线接口，支持完整 I^2C 总线规范以及超快速模式（可识别多个地址且数据速率高达 1 Mbit/s）和监控模式。
- 时钟生成：
  - 晶振，工作频率范围为 1 MHz 至 25 MHz （系统振荡器），带故障检测器。
  - 可对 12 MHz 高频内部 RC 振荡器 (IRC) 进行调整，使其在整个电压和温度范围内精确到 1%。 IRC 可选择性用作系统时钟。
  - 内部低功率低频看门狗振荡器 (WDO)，具有可编程频率输出。
  - 系统振荡器或 IRC 作为时钟源时， PLL 允许 CPU 以最大 CPU 速率运行。
  - 为 USB 提供第二个专用 PLL (LPC1345/46/47)。
  - 时钟输出功能，其中分频器可反映系统振荡器、主时钟、 IRC 或看门狗振荡器。
- 功率控制：
  - 4 种节能模式：睡眠模式、深度睡眠模式、掉电模式和深度掉电模式。
  - 功率配置驻留在 Boot ROM 内，可通过一次简单函数调用，针对任何给定应用来优化性能并最大限度降低功耗。
  - 通过复位、可选择的 GPIO 引脚、看门狗中断或 USB 端口活动从深度睡眠模式和掉电模式唤醒处理器。
  - 使用一个特殊功能引脚从深度掉电模式唤醒处理器。
  - 集成 PMU（电源管理单元）可最大程度降低睡眠模式、深度睡眠模式、掉电模式和深度掉电模式的功耗。
  - 上电复位 (POR)。

– 掉电检测，为中断和强制复位设有 4 个独立的阈值。

- 识别器件的唯一序列号。
- 3.3 V 单电源 （2.0 V 至 3.6 V）。
- 温度范围 –40 °C 至 +85 °C。
- 提供 LQFP64、 LQFP48 和 HVQFN33 封装。

1.3 订购信息

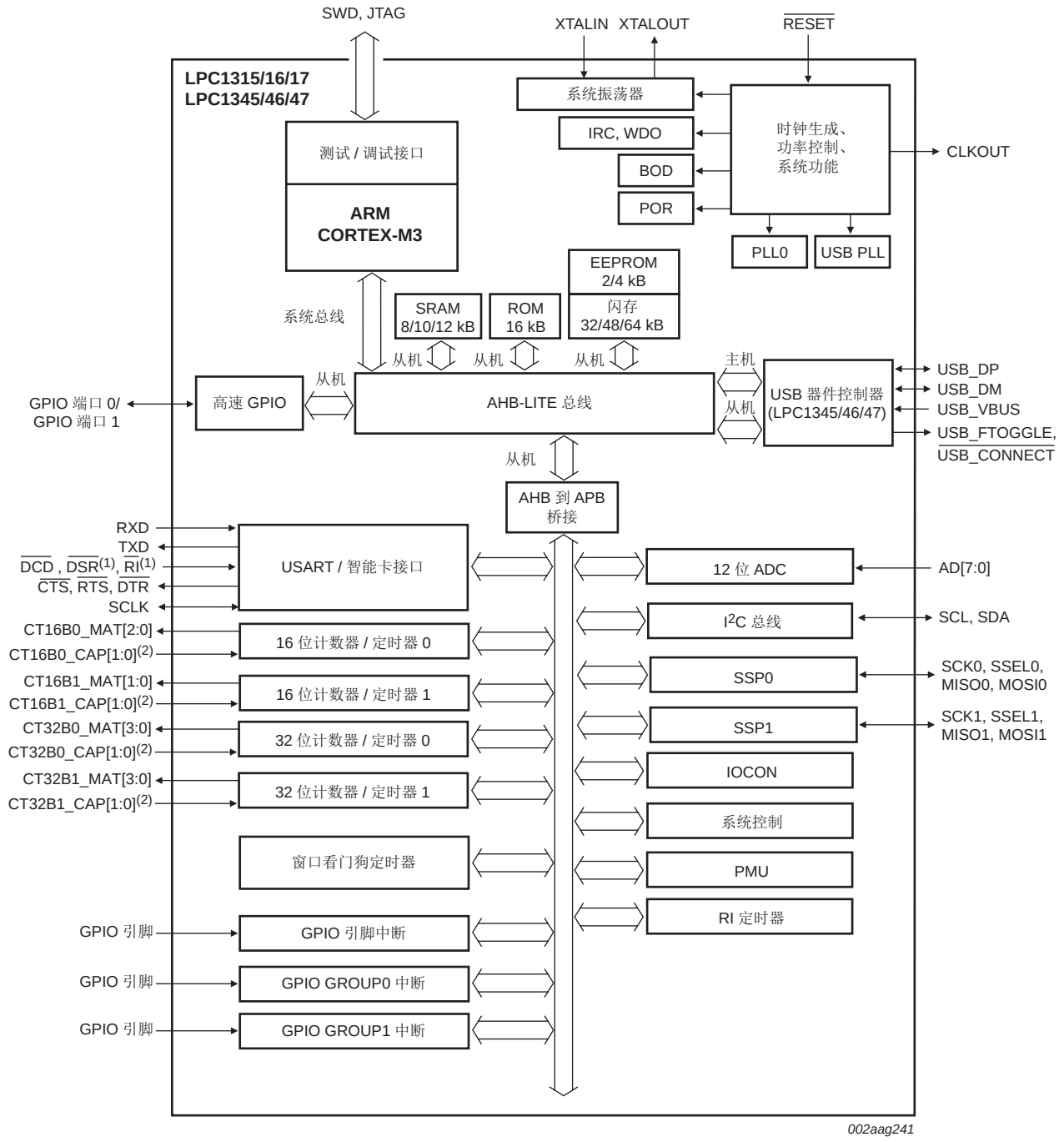
表 1. 订购信息

产品型号	封装		
	名称	描述	版本
LPC1345FHN33	HVQFN33	塑料热性能优化型超薄四侧扁平封装；无引脚； 33 个端子；主体尺寸 7 × 7 × 0.85 mm	不适用
LPC1345FBD48	LQFP48	塑封薄型四侧扁平封装； 48 引脚；主体尺寸 7 × 7 × 1.4 mm	SOT313-2
LPC1346FHN33	HVQFN33	塑料热性能优化型超薄四侧扁平封装；无引脚； 33 个端子；主体尺寸 7 × 7 × 0.85 mm	不适用
LPC1346FBD48	LQFP48	塑封薄型四侧扁平封装； 48 引脚；主体尺寸 7 × 7 × 1.4 mm	SOT313-2
LPC1347FHN33	HVQFN33	塑料热性能优化型超薄四侧扁平封装；无引脚； 33 个端子；主体尺寸 7 × 7 × 0.85 mm	不适用
LPC1347FBD48	LQFP48	塑封薄型四侧扁平封装； 48 引脚；主体尺寸 7 × 7 × 1.4 mm	SOT313-2
LPC1347FBD64	LQFP64	LQFP64：塑封薄型四侧扁平封装； 64 引脚；主体尺寸 10 × 10 × 1.4 mm	SOT314-2
LPC1315FHN33	HVQFN33	塑料热性能优化型超薄四侧扁平封装；无引脚； 33 个端子；主体尺寸 7 × 7 × 0.85 mm	不适用
LPC1315FBD48	LQFP48	塑封薄型四侧扁平封装； 48 引脚；主体尺寸 7 × 7 × 1.4 mm	SOT313-2
LPC1316FHN33	HVQFN33	塑料热性能优化型超薄四侧扁平封装；无引脚； 33 个端子；主体尺寸 7 × 7 × 0.85 mm	不适用
LPC1316FBD48	LQFP48	塑封薄型四侧扁平封装； 48 引脚；主体尺寸 7 × 7 × 1.4 mm	SOT313-2
LPC1317FHN33	HVQFN33	塑料热性能优化型超薄四侧扁平封装；无引脚； 33 个端子；主体尺寸 7 × 7 × 0.85 mm	不适用
LPC1317FBD48	LQFP48	塑封薄型四侧扁平封装； 48 引脚；主体尺寸 7 × 7 × 1.4 mm	SOT313-2
LPC1317FBD64	LQFP64	LQFP64：塑封薄型四侧扁平封装； 64 引脚；主体尺寸 10 × 10 × 1.4 mm	SOT314-2

表 2. 订购选项

产品型号	闪存 [kB]	SRAM [kB]			EEPROM [kB]	USB 设备	SSP	I2C/ FM+	ADC 通道	GPIO 引脚
		SRAM0	USB SRAM	SRAM1						
LPC1345FHN33	32	8	2	-	2	是	2	1	8	26
LPC1345FBD48	32	8	2	-	2	是	2	1	8	40
LPC1346FHN33	48	8	2	-	4	是	2	1	8	26
LPC1346FBD48	48	8	2	-	4	是	2	1	8	40
LPC1347FHN33	64	8	2	2	4	是	2	1	8	26
LPC1347FBD48	64	8	2	2	4	是	2	1	8	40
LPC1347FBD64	64	8	2	2	4	是	2	1	8	51
LPC1315FHN33	32	8	-	-	2	否	2	1	8	28
LPC1315FBD48	32	8	-	-	2	否	2	1	8	40
LPC1316FHN33	48	8	-	-	4	否	2	1	8	28
LPC1316FBD48	48	8	-	-	4	否	2	1	8	40
LPC1317FHN33	64	8	-	2	4	否	2	1	8	28
LPC1317FBD48	64	8	-	2	4	否	2	1	8	40
LPC1317FBD64	64	8	-	2	4	否	2	1	8	51

1.4 功能框图



(1) 仅提供 LQFP48 和 LQFP64 封装。

(2) CT16B0_CAP1、CT16B1_CAP1、CT32B1_CAP1 输入仅可用于 LQFP64 封装。CT32B0_CAP0 输入仅可用于 LQFP48 和 LQFP64 封装。

图 1. 功能框图

2.1 本章导读

有关 LPC1315/16/17/45/46/47 器件的存储器配置，请参见表 3。

表 3. LPC1315/16/17/45/46/47 存储器配置

产品型号	闪存 [kB]	SRAM [kB]			EEPROM [kB]
		SRAM0	USB SRAM	SRAM1	
LPC1345FHN33	32	8	2	-	2
LPC1345FBD48	32	8	2	-	2
LPC1346FHN33	48	8	2	-	4
LPC1346FBD48	48	8	2	-	4
LPC1347FHN33	64	8	2	2	4
LPC1347FBD48	64	8	2	2	4
LPC1347FBD64	64	8	2	2	4
LPC1315FHN33	32	8	-	-	2
LPC1315FBD48	32	8	-	-	2
LPC1316FHN33	48	8	-	-	4
LPC1316FBD48	48	8	-	-	4
LPC1317FHN33	64	8	-	2	4
LPC1317FBD48	64	8	-	2	4
LPC1317FBD64	64	8	-	2	4

2.2 存储器映射

LPC1315/16/17/45/46/47 包含几个不同的存储器区域，如下列各图所示。图 2 显示了从用户程序角度来看，复位后的整个地址空间的总映射。

AHB 外设区的大小为 2 Mb，可分配多达 128 个外设。APB 外设区的大小为 512 kB，可分配多达 32 个外设。每种类型的每一个外设空间的大小均为 16 kB。从而简化了每个外设的地址译码。

2.2.1 片内闪存编程存储器

LPC1315/16/17/45/46/47 包含最高 128 kB 片内闪存程序存储器。通过片内启动引导程序软件使用在系统编程(ISP)和在应用编程(IAP)可进行闪存的编程。也支持通过USB更新闪存。

闪存分成大小为 4 kB 的扇区，每个扇区包含 16 页。可以使用 IAP 擦除页命令擦除各含 256 字节的单个页。

2.2.2 EEPROM

LPC1315/16/17/45/46/47 包含 2 kB 或 4 kB 的片内字节可擦除和字节可编程 EEPROM 数据存储器。通过片内启动引导程序软件使用在应用编程 (IAP) 可进行 EEPROM 的编程。

2.2.3 SRAM

LPC1315/16/17/45/46/47 总共包含 8 kB、 10 kB 或 12 kB 片内静态 RAM 存储器。 USB SRAM 块仅在 LPC134x 器件中可用。 SRAM 块 SRAM1 仅在 LPC1347/17 器件中可用。

SRAM1 和 USB SRAM 时钟默认为关闭状态。在 SYSHBCLKCTRL 寄存器中使能时钟（表 19）。

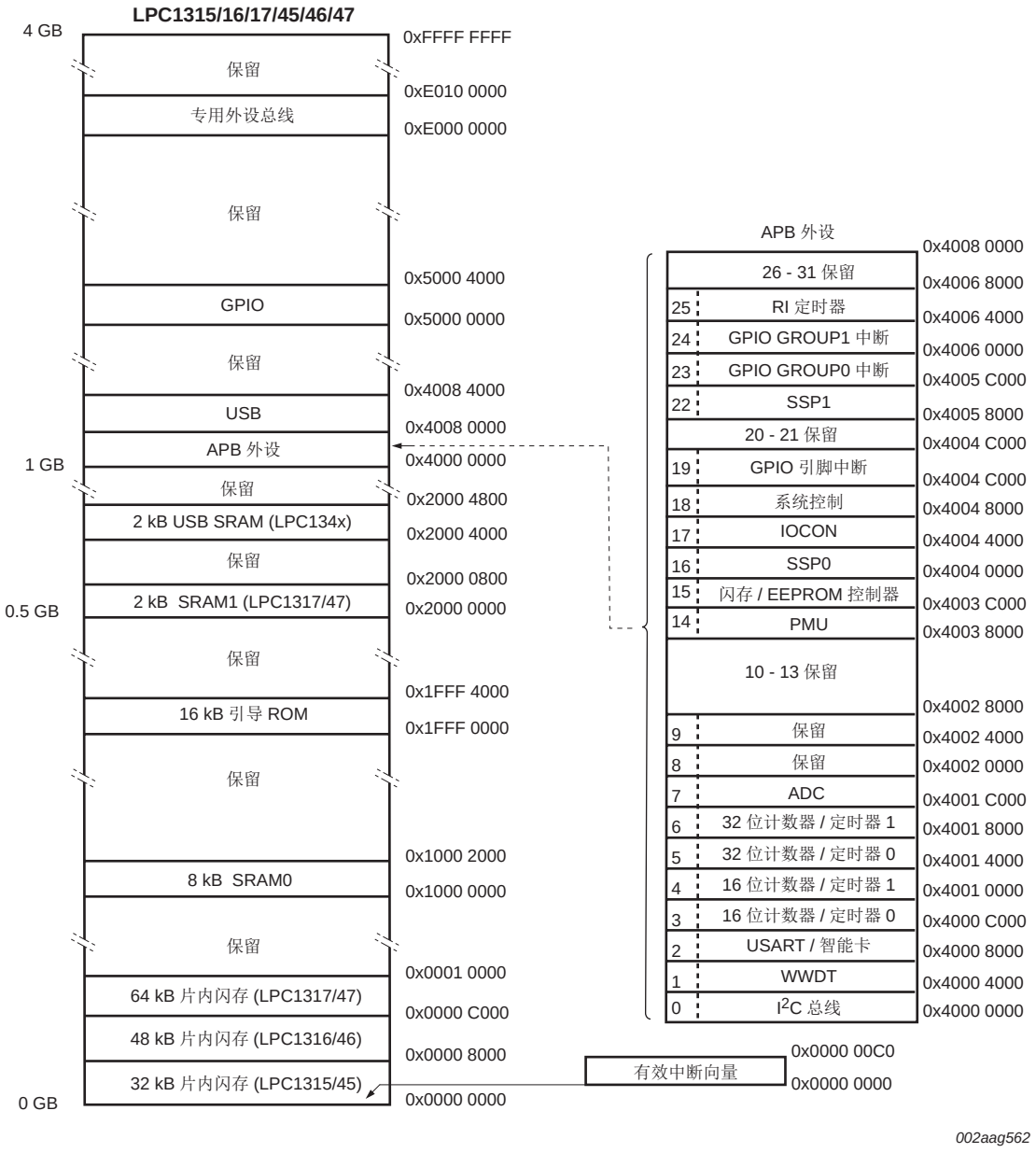


图 2. LPC1315/16/17/45/46/47 存储器映射

3.1 本章导读

所有 USB 相关寄存器和寄存器位仅在 LPC134x 器件中可用。 USB PLL 仅在 LPC134x 器件中可用。

注：DEVICE_ID 寄存器位于地址偏移 0xF8。该寄存器位置和其他 LPC1xxx 器件不同。

3.2 简介

系统配置模块可控制 LPC1315/16/17/45/46/47 的振荡器、电源管理的某些方面和时钟产生。该模块中还包含用于重映射闪存、 SRAM 和 ROM 存储器区域的寄存器。

3.3 引脚说明

表 4 显示了与系统控制模块功能相关的引脚。

表 4. 引脚摘要

引脚名称	引脚方向	引脚说明
CLKOUT	O	Clockout 引脚
PIO0 和 PIO1 引脚	I	可从所有可用的 GPIO 引脚中选择八个引脚作为外部中断引脚 （参见表 35）。

3.4 时钟和电源控制

有关 LPC1315/16/17/45/46/47 时钟产生单元 (CGU) 的概述，参见图 3。

LPC1315/16/17/45/46/47 包括三个独立的振荡器：系统振荡器、内部 RC 振荡器 (IRC) 和看门狗振荡器。每个振荡器都可用于特定应用中所要求的多种用途。

在复位之后，LPC1315/16/17/45/46/47 将从内部 RC 振荡器运行，直到通过软件进行切换。这就使得系统可以在没有外部晶体的情况下运行，并使启动引导程序代码按照已知频率运行。

SYSAHBCLKCTRL 寄存器可开启各种外设和存储器的系统时钟。USART 和 SSP 具有独立的时钟分频器，以从主时钟获得外设时钟。

主时钟以及 IRC、系统振荡器和看门狗振荡器的时钟输出均可以直接在 CLKOUT 引脚上观察到。

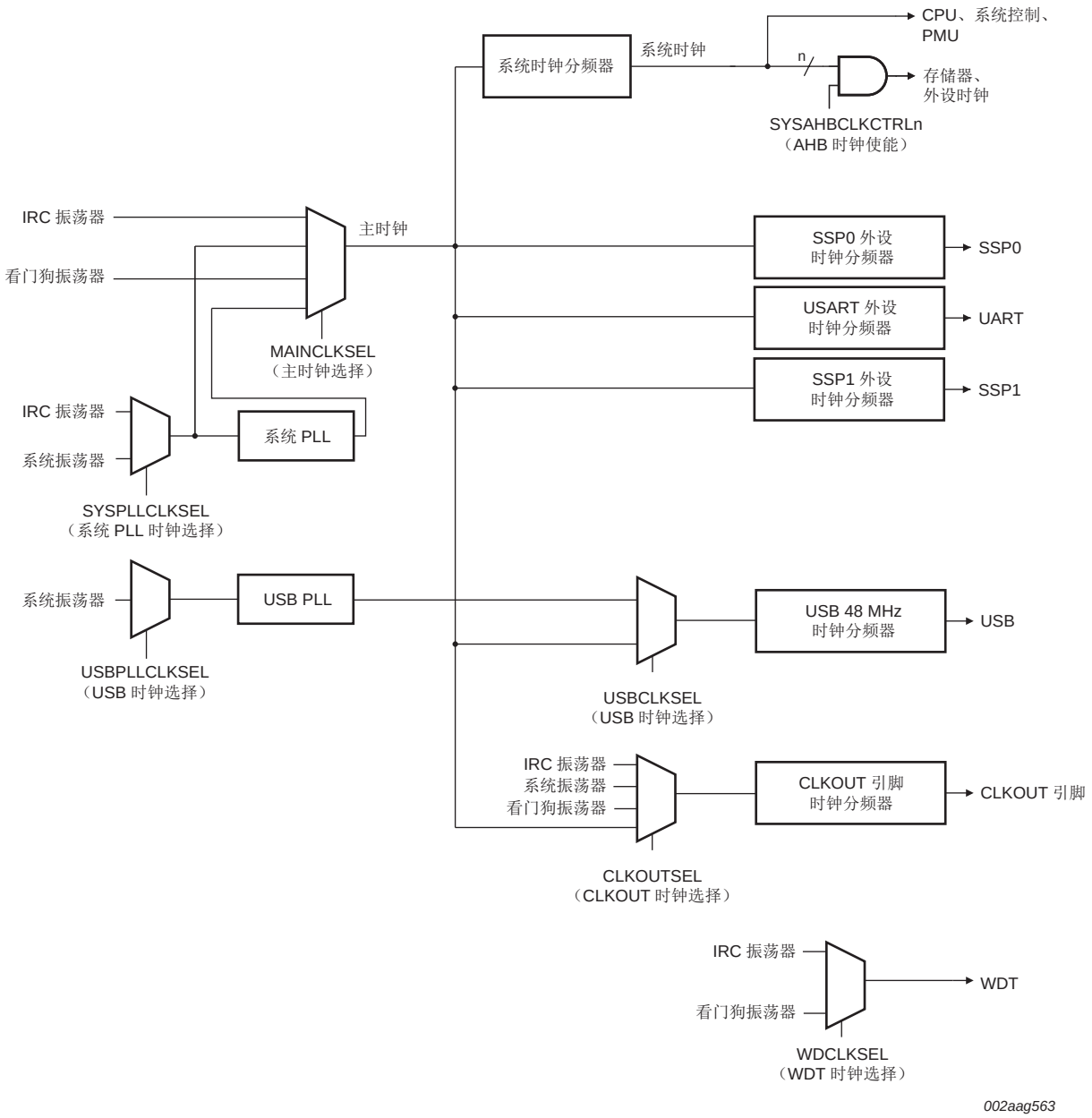


图 3.　LPC1315/16/17/45/46/47 CGU 框图

3.5 寄存器描述

表 5. 寄存器简介：SYSCON （基址：0x4004 8000）

名称	访问类型	地址偏移	描述	复位值	参考
SYSMEMREMAP	读 - 写	0x000	系统存储器重映射	0	表 6
PRESETCTRL	读 - 写	0x004	外设复位控制	0	表 7
SYSPLLCTRL	读 - 写	0x008	系统 PLL 控制	0	表 8
SYSPLLSTAT	读	0x00C	系统 PLL 状态	0	表 9
USBPLLCTRL	读 - 写	0x010	USB PLL 控制	0	表 10
USBPLLSTAT	读	0x014	USB PLL 状态	0	表 11
SYSOSCCTRL	读 - 写	0x020	系统振荡器控制	0x000	表 12
WDTOSCCTRL	读 - 写	0x024	看门狗振荡器控制	0x0A0	表 13
-	-	0x028	保留	-	-
SYSRSTSTAT	读 - 写	0x030	系统复位状态寄存器	0	表 14
SYSPLLCLKSEL	读 - 写	0x040	系统 PLL 时钟源选择	0	表 15
-	-	0x044	保留	-	-
USBPLLCLKSEL	读 - 写	0x048	USB PLL 时钟源选择	0	表 16
-	-	0x04C	保留	-	
MAINCLKSEL	读 - 写	0x070	主时钟源选择	0	表 17
-	-	0x074	保留	-	
SYSAHBCLKDIV	读 - 写	0x078	系统时钟分频器	0x001	表 18
SYSAHBCLKCTRL	读 - 写	0x080	系统时钟控制		表 19
SSP0CLKDIV	读 - 写	0x094	SSP0 时钟分频器	0	表 20
UARTCLKDIV	读 - 写	0x098	UART 时钟分频器	0	表 21
SSP1CLKDIV	读 - 写	0x09C	SSP1 时钟分频器	0x0000	表 22
TRACECLKDIV	读 - 写	0x0AC	ARM 跟踪时钟分频器	0x0000 0000	表 23
SYSTICKCLKDIV	读 - 写	0x0B0	SYSTICK 时钟分频器	0x0000 0000	表 24
USBCLKSEL	读 - 写	0x0C0	USB 时钟源选择	0	表 25
-	-	0x0C4	保留	-	
USBCLKDIV	读 - 写	0x0C8	USB 时钟源分频器	0	表 26
CLKOUTSEL	读 - 写	0x0E0	CLKOUT 时钟源选择	0	表 27
-	-	0x0E4	保留	-	
CLKOUTDIV	读 - 写	0x0E8	CLKOUT 时钟分频器	0	表 28
PIOPORCAP0	读	0x100	POR 捕获 PIO 状态 0	由用户决定	表 29
PIOPORCAP1	读	0x104	POR 捕获 PIO 状态 1	由用户决定	表 30
BODCTRL	读 - 写	0x150	掉电检测	0	表 31
SYSTCKCAL	读 - 写	0x154	系统节拍计数器校准		表 32
IRQLATENCY	读 - 写	0x170	四分位距 (IQR) 延迟。使中断延迟和确定性之间达到平衡。	0x0000 0010	表 33
NMISRC	读 - 写	0x174	NMI 源控制	0	表 34
PINTSEL	读 - 写	0x178	GPIO 引脚中断选择寄存器	0	表 35
USBCLKCTRL	读 - 写	0x198	USB 时钟控制		表 36
USBCLKST	读	0x19C	USB 时钟状态		表 37

表 5. 寄存器简介：SYSCON （基址：0x4004 8000）*（续）*

名称	访问类型	地址偏移	描述	复位值	参考
STARTERP0	读 - 写	0x204	启动逻辑 0 中断唤醒使能寄存器 0	0	表 38
STARTERP1	读 - 写	0x214	启动逻辑 1 中断唤醒使能寄存器 1	0	表 39
PDSLEEPCFG	读 - 写	0x230	深度睡眠模式掉电状态		表 40
PDAWAKECFG	读 - 写	0x234	从深度睡眠唤醒的掉电状态		表 41
PDRUNCFG	读 - 写	0x238	电源配置寄存器		表 42
DEVICE_ID	读	0x3F8	器件 ID	取决于零部件	表 43

3.5.1 系统存储器重映射寄存器 (SYSMEMREMAP)

系统存储器重映射寄存器选择是从 Boot ROM、闪存还是 SRAM 读取异常向量。默认情况下，闪存映射到地址 0x0000 0000。当 SYSMEMREMAP 寄存器的 MAP 位设为 0x0 或 0x1 时， Boot ROM 或 RAM 将分别映射到存储器映射 （地址 0x0000 0000 至 0x0000 0200）的底部 512 字节。

表 6. 系统存储器重映射 （SYSMEMREMAP，地址 0x4004 8000）位描述

位	符号	值	描述	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 Boot 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	-

3.5.2 外设复位控制寄存器 (PRESETCTRL)

该寄存器使软件可以复位特定外设。该寄存器的某个分配位中的 0 可以复位指定外设。1 将取消复位并允许外设操作。

注：访问 SSP 和 I2C 外设之前，向此寄存器写入 1，确保取消 SSP 和 I2C 的复位信号。

表 7. 外设复位控制 （PRESETCTRL，地址 0x4004 0004）位描述

位	符号	值	描述	复位值
0	SSP0_RST_N		SSP0 复位控制	0
		0	复位 SSP0 外设。	
		1	SSP0 复位已取消。	
1	I2C_RST_N		I2C 复位控制	0
		0	复位 I2C 外设。	
		1	I2C 复位已取消。	
2	SSP1_RST_N		SSP1 复位控制	0
		0	复位 SSP0 外设。	
		1	SSP1 复位已取消。	
3	-		保留	-
31:4	-		保留	-

3.5.3 系统 PLL 控制寄存器 (SYSPLLCTRL)

该寄存器用于连接和使能系统 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。

表 8. 系统 PLL 控制 （SYSPLLCTRL，地址 0x4004 8008）位描述

位	符号	值	描述	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		后分频器比 P。分频比为 2 x P。	0
		0x0	P = 1	
		0x1	P = 2	
		0x2	P = 4	
		0x3	P = 8	
31:7	-		保留。保留位不能写 1。	-

3.5.4 系统 PLL 状态寄存器 (SYSPLLSTAT)

该寄存器是只读寄存器，提供 PLL 锁定状态 （参见第 3.10.1 节）。

表 9. 系统 PLL 状态 （SYSPLLSTAT，地址 0x4004 800C）位描述

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

3.5.5 USB PLL 控制寄存器 (USBPLLCTRL)

USB PLL 和系统 PLL 一样，用于向 USB 模块提供专用时钟（如果可用）（参见第 3.1 节）。

该寄存器用于连接和使能 USB 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 USB 子系统实际使用的 48 MHz 时钟。

表 10. USB PLL 控制 （USBPLLCTRL，地址 0x4004 8010）位描述

位	符号	值	描述	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	0x000
6:5	PSEL		后分频器比 P。分频比为 2 x P。	0x00
		0x0	P = 1	
		0x1	P = 2	
		0x2	P = 4	
		0x3	P = 8	
31:7	-		保留。保留位不能写 1。	0x00

3.5.6 USB PLL 状态寄存器 (USBPLLSTAT)

该寄存器是只读寄存器，提供 PLL 锁定状态 （参见第 3.10.1 节）。

表 11. USB PLL 状态 （USBPLLSTAT，地址 0x4004 8014）位描述

位	符号	值	描述	复位值
0	LOCK		PLL 锁定状态	0x0
		0	PLL 未锁定	
		1	PLL 锁定	
31:1	-		保留	0x00

3.5.7 系统振荡器控制寄存器 (SYSOSCCTRL)

该寄存器可配置系统振荡器的频率范围。

表 12. 系统振荡器控制 （SYSOSCCTRL，地址 0x4004 8020）位描述

位	符号	值	描述	复位值
0	BYPASS		旁路系统振荡器	0x0
		0	振荡器未被旁路。	
		1	振荡器被旁路。PLL 输入 (sys_osc_clk) 直接由 XTALIN 和 XTALOUT 引脚提供。	
1	FREQRANGE		决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

3.5.8 看门狗振荡器控制寄存器 (WDTOSCCTRL)

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 (F_{clkana})。FREQSEL 字段选择 0.5 和 3.4 MHz 之间的 F_{clkana}。在数字部分中， F_{clkana} 在 DIVSEL 字段的控制下进行分频，产生振荡器输出时钟。

输出时钟频率的计算如下：
wdt_osc_clk = Fclkana/(2 × (1 + DIVSEL)。

注：FREQSEL 字段的任何非零设置产生的 F_{clkana} 值都在所列频率值的 ±40% 范围内。

表 13. 看门狗振荡器控制 （WDTOSCCTRL，地址 0x4004 8024）位描述

位	符号	值	描述	复位值
4:0	DIVSEL		选择 Fclkana 分频器。 wdt_osc_clk = Fclkana/ (2 x(1 + DIVSEL)) 00000：2 x (1 + DIVSEL) = 2 00001：2 x (1 + DIVSEL) = 4 至 11111：2 x (1 + DIVSEL) = 64	0

表 13. 看门狗振荡器控制 （WDTOSCCTRL，地址 0x4004 8024）位描述 *（续）*

位	符号	值	描述	复位值
8:5	FREQSEL		选择看门狗振荡器模拟输出频率 (Fclkana)。值 0x0 保留。此值的操作未定义。复位后，启动代码应尽快在该字段中编程非零值。	0
		0x1	0.5 MHz	
		0x2	0.8 MHz	
		0x3	1.1 MHz	
		0x4	1.4 MHz	
		0x5	1.6 MHz	
		0x6	1.8 MHz	
		0x7	2.0 MHz	
		0x8	2.2 MHz	
		0x9	2.4 MHz	
		0xA	2.6 MHz	
		0xB	2.7 MHz	
		0xC	2.9 MHz	
		0xD	3.1 MHz	
		0xE	3.2 MHz	
		0xF	3.4 MHz	
31:9	-		保留	-

3.5.9 系统复位状态寄存器 (SYSRSTSTAT)

如果取消 POR 信号后，其他复位信号（例如，外部 $\overline{\text{RESET}}$ 引脚）仍然有效，则其位将设置为 “检测到”。

表 14. 系统复位状态寄存器 （SYSRSTSTAT，地址 0x4004 8030）位描述

位	符号	值	描述	复位值
0	POR		POR 复位状态	0
		0	未检测到 POR	
		1	检测到 POR	
1	EXTRST		外部 RESET 引脚的状态	0
		0	未检测到复位事件	
		1	检测到复位	
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	
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	
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	
31:5	-		保留	-

3.5.10 系统 PLL 时钟源选择寄存器 (SYSPLLCLKSEL)

该寄存器为系统 PLL 选择时钟源。

表 15. 系统 PLL 时钟源选择 （SYSPLLCLKSEL，地址 0x4004 8040）位描述

位	符号	值	描述	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	保留	
31:2	-		保留	-

3.5.11 USB PLL 时钟源选择寄存器 (USBPLLCLKSEL)

该寄存器为专用 USB PLL 选择时钟源。

注：切换时钟源时，必须先运行两个时钟。

表 16. USB PLL 时钟源选择 （USBPLLCLKSEL，地址 0x4004 8048）位描述

位	符号	值	描述	复位值
1:0	SEL		USB PLL 时钟源	0x00
		0x0	IRC。必须将 USB PLL 时钟源切换到系统振荡器以确保正确的 USB 操作。	
		0x1	系统振荡器	
		0x2	保留	
		0x3	保留	
31:2	-		保留	0x00

3.5.12 主时钟源选择寄存器 (MAINCLKSEL)

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

表 17. 主时钟源选择 （MAINCLKSEL，地址 0x4004 8070）位描述

位	符号	值	描述	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	-

3.5.13 系统时钟分频寄存器 (SYSAHBCLKDIV)

该寄存器控制主时钟的分频方式，以向内核、存储器和外设提供系统时钟。可以通过将 DIV 字段设置为 0 完全关闭系统时钟。

表 18. 系统时钟分频器 （SYSAHBCLKDIV，地址 0x4004 8078）位描述

位	符号	描述	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	-

3.5.14 系统时钟控制寄存器 (SYSAHBCLKCTRL)

SYSAHBCLKCTRL 寄存器可使能单个系统和外设模块的时钟。系统时钟（位 0）为 AHB、APB 桥接、 ARM Cortex-M3、 SYSCON 模块和 PMU 提供时钟。无法禁用该时钟。

表 19. 系统时钟控制 （SYSAHBCLKCTRL，地址 0x4004 8080）位描述

位	符号	值	描述	复位值
0	SYS		使能 AHB、 APB 桥接、 Cortex-M3 FCLK 和 HCLK、 SysCon 及 PMU 的时钟。此位为只读且始终为 1。	1
		0	保留	
		1	使能	
1	ROM		使能 ROM 时钟。	1
		0	禁用	
		1	使能	
2	RAM0		使能 SRAM0 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	已禁用	
		1	使能	
4	FLASHARRAY		使能闪存阵列时钟访问。	1
		0	已禁用	
		1	使能	
5	I2C		使能 I2C 时钟。	1
		0	禁用	
		1	使能	
6	GPIO		使能 GPIO 端口寄存器的时钟。	0
		0	禁用	
		1	使能	
7	CT16B0		使能 16 位计数器 / 定时器 0 的时钟。	0
		0	禁用	
		1	使能	

表 19. 系统时钟控制 （SYSAHBCLKCTRL，地址 0x4004 8080）位描述 *（续）*

位	符号	值	描述	复位值
8	CT16B1		使能 16 位计数器 / 定时器 1 的时钟。	0
		0	禁用	
		1	使能	
9	CT32B0		使能 32 位计数器 / 定时器 0 的时钟。	0
		0	禁用	
		1	使能	
10	CT32B1		使能 32 位计数器 / 定时器 1 的时钟。	0
		0	禁用	
		1	使能	
11	SSP0		使能 SSP0 的时钟。	0
		0	禁用	
		1	使能	
12	USART		使能 UART 的时钟。	0
		0	禁用	
		1	使能	
13	ADC		使能 ADC 时钟。	0
		0	禁用	
		1	使能	
14	USB		使能 USB 寄存器接口的时钟。	0
		0	禁用	
		1	使能	
15	WWDT		使能 WWDT 的时钟。	0
		0	禁用	
		1	使能	
16	IOCON		使能 I/O 配置模块的时钟。	0
		0	禁用	
		1	使能	
17	-		保留	0
18	SSP1		使能 SSP1 的时钟。	0
		0	禁用	
		1	使能	
19	PINT		使能 GPIO 引脚中断寄存器接口的时钟。	0
		0	禁用	
		1	使能	
22:20	-		保留	-
23	GROUP0INT		使能 GPIO 组 0 中断寄存器接口的时钟。	0
		0	禁用	
		1	使能	
24	GROUP1INT		使能 GPIO 组 1 中断寄存器接口的时钟。	0
		0	禁用	
		1	使能	

表 19. 系统时钟控制 （SYSAHBCLKCTRL，地址 0x4004 8080）位描述 *（续）*

位	符号	值	描述	复位值
25	-		保留	-
26	RAM1		使能位于 0x2000 0000 到 0x2000 0800 的 SRAM1 的时钟。	1
		0	禁用	
		1	使能	
27	USBSRAM		使能位于 0x2000 4000 到 0x2000 4800 的 USB SRAM 块。	0
		0	禁用	
		1	使能	
31:28	-		保留	-

3.5.15 SSP0 时钟分频寄存器 (SSP0CLKDIV)

该寄存器可配置 SSP0 外设时钟 SPI0_PCLK。可以通过将 DIV 字段设置为 0 关闭 SPI0_PCLK。

表 20. SSP0 时钟分频器 （SSP0CLKDIV，地址 0x4004 8094）位描述

位	符号	描述	复位值
7:0	DIV	SPI0_PCLK 时钟分频器值。 0：系统时钟禁用。 1：1 分频。 至 255：255 分频。	0
31:8	-	保留	-

3.5.16 UART 时钟分频寄存器 (UARTCLKDIV)

该寄存器可配置 USART 外设时钟 UART_PCLK。可以通过将 DIV 字段设置为 0 关闭 UART_PCLK。

表 21. UART 时钟分频器 （UARTCLKDIV，地址 0x4004 8098）位描述

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

3.5.17 SSP1 时钟分频寄存器 (SSP1CLKDIV)

该寄存器可配置 SSP1 外设时钟 SSP1_PCLK。可以通过将 DIV 位设置为 0x0 关闭 SSP1_PCLK。

表 22. SSP1 时钟分频器 （SSP1CLKDIV，地址 0x4004 809C）位描述

位	符号	描述	复位值
7:0	DIV	SSP1_PCLK 时钟分频器值 0：禁用 SSP1_PCLK。 1：1 分频。 至 255：255 分频。	0x00
31:8	-	保留	0x00

3.5.18 ARM 跟踪时钟分频寄存器 (TRACECLKDIV)

该寄存器可配置 ARM 跟踪时钟。可以通过将 DIV 字段设置为 0 关闭 ARM 跟踪时钟。

表 23. ARM 跟踪时钟分频器 （TRACECLKDIV，地址 0x4004 80AC）位描述

位	符号	描述	复位值
7:0	DIV	ARM 跟踪时钟分频器值。 0：禁用 TRACE_CLK。 1：1 分频。 至 255：255 分频。	0x00
31:8	-	保留	0x00

3.5.19 SYSTICK 时钟分频寄存器 (SYSTICKCLKDIV)

该寄存器可配置 SYSTICK 外设时钟。可以通过将 DIV 字段设置为 0 关闭 SYSTICK 定时器时钟。

表 24. SYSTICK 时钟分频器 （SYSTICKCLKDIV，地址 0x4004 80B0）位描述

位	符号	描述	复位值
7:0	DIV	SYSTICK 时钟分频器值。 0：禁用 SYSTICK 定时器时钟。 1：1 分频。 至 255：255 分频。	0x00
31:8	-	保留	0x00

3.5.20 USB 时钟源选择寄存器 (USBCLKSEL)

该寄存器为 USB usb_clk 选择时钟源。时钟源可以是 USB PLL 输出或主时钟，该时钟可通过 USBCLKDIV 寄存器 （参见表 26）进一步分频，得到一个 48 MHz 时钟。

注：切换时钟源时，必须先运行两个时钟，然后才能更新时钟源。 USB 控制器的默认时钟源为 USB PLL 输出。要将时钟源切换到主时钟，应确保系统 PLL 和 USB PLL 都运行，使两个时钟源可用于切换。主时钟必须设为 48 MHz 并配置了主 PLL 和系统振荡器。切换后，可关闭 USB PLL。

表 25. USB 时钟源选择 （USBCLKSEL，地址 0x4004 80C0）位描述

位	符号	值	描述	复位值
1:0	SEL		USB 时钟源。值 0x2 和 0x3 保留。	0x00
		0x0	USB PLL 输出	
		0x1	主时钟	
31:2	-		保留	0x00

3.5.21 USB 时钟源分频寄存器 (USBCLKDIV)

该寄存器允许将 USB 时钟 usb_clk 分频为 48 MHz。可以通过将 DIV 位设置为 0x0 关闭 usb_clk。

表 26. USB 时钟源分频器 （USBCLKDIV，地址 0x4004 80C8）位描述

位	符号	描述	复位值
7:0	DIV	USB 时钟分频器值 0：禁用 USB 时钟。 1：1 分频。 至 255：255 分频。	0x01
31:8	-	保留	0x00

3.5.22 CLKOUT 时钟源选择寄存器 (CLKOUTSEL)

此寄存器选择 CLKOUT 引脚上可见的信号。可选择任意振荡器或主时钟。

要改变 CLKOUT 引脚上可见的时钟源，首先使用仍在运行的当前选定时钟源使能新时钟源，使用 SEL 位改变时钟源，然后取消当前时钟源。

如果 CLKOUT 引脚上选定的时钟源在 PDRUNCFG 或 PDSLEEPCFG 寄存器中掉电，则必须先重新使能该时钟源，才可通过此寄存器选择另一个时钟源。

表 27. CLKOUT 时钟源选择 （CLKOUTSEL，地址 0x4004 80E0）位描述

位	符号	值	描述	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

3.5.23 CLKOUT 时钟分频寄存器 (CLKOUTDIV)

该寄存器可确定 CLKOUT 引脚上信号的分频器值。

表 28. CLKOUT 时钟分频器 （CLKOUTDIV，地址 0x4004 80E8）位描述

位	符号	描述	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	-

3.5.24 POR 捕获 PIO 状态 0 寄存器 (PIOPORCAP0)

PIOPORCAP0 寄存器用于在上电复位时捕获 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 29. POR 捕获 PIO 状态 0 （PIOPORCAP0，地址 0x4004 8100）位描述

位	符号	描述	复位值
23:0	PIOSTAT	上电复位时 P0_23 到 P0_0 的状态	依赖于执行
31:24	-	保留	-

3.5.25 POR 捕获 PIO 状态 1 寄存器 (PIOPORCAP1)

PIOPORCAP1 寄存器用于在上电复位时捕获 GPIO 端口 1 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 30. POR 捕获 PIO 状态 1 （PIOPORCAP1，地址 0x4004 8104）位描述

位	符号	描述	复位值
31:0	PIOSTAT	上电复位时 P1_31 到 P1_0 的状态	依赖于执行

3.5.26 掉电检测寄存器 (BODCTRL)

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 31 中所列的复位和中断阈值都是典型值。

BOD 中断和 BOD 复位 （取决于该寄存器中位 BODRSTENA 的值）都可从睡眠模式、深度睡眠模式和掉电模式唤醒芯片。参见第 3.9 节。

表 31. 掉电检测 （BODCTRL，地址 0x4004 8150）位描述

位	符号	值	描述	复位值
1:0	BODRSTLEV		BOD 复位电平	00
		0x0	0 级：能引起复位的阈值电压为 1.46 V；能使复位无效的阈值电压为 1.63 V。	
		0x1	1 级：能引起复位的阈值电压为 2.06 V；能使复位无效的阈值电压为 2.15 V。	
		0x2	2 级：能引起复位的阈值电压为 2.35 V；能使复位无效的阈值电压为 2.43 V。	
		0x3	3 级：能引起复位的阈值电压为 2.63 V；能使复位无效的阈值电压为 2.71 V。	
3:2	BODINTVAL		BOD 中断电平	00
		0x0	0 级：能引起中断的阈值电压为 1.65 V；能使中断无效的阈值电压为 1.80 V。	
		0x1	1 级：能引起中断的阈值电压为 2.22 V；能使中断无效的阈值电压为 2.35 V。	
		0x2	2 级：能引起中断的阈值电压为 2.52 V；能使中断无效的阈值电压为 2.66 V。	
		0x3	3 级：能引起中断的阈值电压为 2.80 V；能使中断无效的阈值电压为 2.90 V。	

表 31. 掉电检测 （BODCTRL，地址 0x4004 8150）位描述 *（续）*

位	符号	值	描述	复位值
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

3.5.27 系统节拍计数器校准寄存器 (SYSTCKCAL)

该寄存器可确定 SYST_CALIB 寄存器的值 （参见表 313）。

表 32. 系统节拍计数器校准 （SYSTCKCAL，地址 0x4004 8154）位描述

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

3.5.28 IQR 延迟寄存器 (IRQLATENCY)

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)。该寄存器的意图在于使用户可以在中断响应时间与确定性之间选择一个平衡点。

将此参数设置为一个非常小的值（例如 0），可以保证最佳中断性能，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-M3 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 33. IQR 延迟 （IRQLATENCY，地址 0x4004 8170）位描述

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

3.5.29 NMI 源控制寄存器 (NMISRC)

NMI 源选择寄存器选择外设中断作为 ARM Cortex-M3 内核的 NMI 中断源。有关所有外设中断及其 IRQ 编号的列表，请参见表 53。有关 NMI 功能描述，请参见 《ARM Cortex-M3 技术参考手册》。

表 34. NMI 源控制 （NMISRC，地址 0x4004 8174）位描述

位	符号	描述	复位值
4:0	IRQNO	位 31 为 1 时，用作非屏蔽中断 (NMI) 的中断的 IRQ 编号。有关中断源及其 IRQ 编号的列表，请参见表 53。	0
30:5	-	保留	-
31	NMIEN	向该位写入 1，使能位 4:0 选择的非屏蔽中断 (NMI) 源。	0

注：如果 NMISRC 寄存器用于选择某个中断作为非屏蔽中断，并使能所选的中断，则一个中断请求会同时产生非屏蔽中断和正常中断。这可通过禁用 NVIC 中的正常中断来避免，如《ARM Cortex-M3 技术参考手册》中所述。

3.5.30 GPIO 引脚中断选择寄存器 (PINTSEL)

这 8 个寄存器中的每一个都能从两个端口的所有 GPIO 引脚中选择一个 GPIO 引脚，作为引脚中断源。要为八个引脚中断中的任何一个选择引脚，将引脚编号写入 INTPIN 位（0 至 23 对应于引脚 PIO0_0 至 PIO0_23， 24 至 55 对应于引脚 PIO1_0 至 PIO1_31）。例如，在 PINTSEL0 中将 INTPIN 设为 0x5 会选择引脚 PIO0_5 用于引脚中断 0。在 PINTSEL7 中将 INTPIN 设为 0x32 （引脚 50）会选择引脚 PIO1_26 用于引脚中断 7。

8 个引脚中断中的每一个都必须使用中断槽 0 至 7 在 NVIC 中使能 （参见表 53）。

如需使能每个引脚中断并配置其边沿或电平敏感性，使用GPIO引脚中断寄存器（见第 9.4.1 节）。

表 35. GPIO 引脚中断选择寄存器 （PINTSEL，地址 0x4004 8178）位描述

位	符号	值	描述	复位值
4:0	INTPIN		该寄存器的 PORTSEL 位所选择的端口中的引脚编号。	0
5	PORTSEL		为该寄存器的 INTPIN 位要选择的引脚编号选择端口。	0
		0	端口 0	
		1	端口 1	
31:6	-		保留	-

3.5.31 USB 时钟控制寄存器 (USBCLKCTRL)

该寄存器控制 USB need_clock 信号的使用和 need_clock 信号的极性，用于触发 USB 唤醒中断。有关如何使用 USB need_clock 信号从深度睡眠模式或掉电模式唤醒器件，请参见第 10.7.6 节。

表 36. USB 时钟控制 （USBCLKCTRL，地址 0x4004 8198）位描述

位	符号	值	描述	复位值
0	AP_CLK		USB need_clock 信号控制	0x0
		0	受硬件控制。	
		1	强制为高电平。	
1	POL_CLK		触发 USB 唤醒中断的 USB need_clock 极性	0x0
		0	USB need_clock 的下降沿触发 USB 唤醒 （默认）。	
		1	USB need_clock 的上升沿触发 USB 唤醒。	
31:2	-		保留	0x00

3.5.32 USB 时钟状态寄存器 (USBCLKST)

寄存器为只读，将返回 USB need_clock 信号的状态。有关如何使用 USB need_clock 信号从深度睡眠模式或掉电模式唤醒器件，请参见第 10.7.6 节。

表 37. USB 时钟状态 （USBCLKST，地址 0x4004 819C）位描述

位	符号	值	描述	复位值
0	NEED_CLKST		USB need_clock 信号状态	0x0
		0	低	
		1	高	
31:1	-		保留	0x00

3.5.33 启动逻辑 0 中断唤醒使能寄存器 0 (STARTERP0)

STARTERP0 寄存器使能通过引脚中断选择寄存器（参见表 35）选择的各个 GPIO 引脚进行唤醒。引脚中断还必须在 NVIC 中使能 （表 53 中的中断 0 至 8）。

表 38. 启动逻辑 0 中断唤醒使能寄存器 0 （STARTERP0，地址 0x4004 8204）位描述

位	符号	值	描述	复位值
0	PINT0		引脚中断 0 唤醒	0
		0	已禁用	
		1	使能	
1	PINT1		引脚中断 1 唤醒	0
		0	已禁用	
		1	使能	
2	PINT2		引脚中断 2 唤醒	0
		0	已禁用	
		1	使能	
3	PINT3		引脚中断 3 唤醒	0
		0	已禁用	
		1	使能	
4	PINT4		引脚中断 4 唤醒	0
		0	已禁用	
		1	使能	
5	PINT5		引脚中断 5 唤醒	0
		0	已禁用	
		1	使能	
6	PINT6		引脚中断 6 唤醒	0
		0	已禁用	
		1	使能	
7	PINT7		引脚中断 7 唤醒	0
		0	已禁用	
		1	使能	
31:8	-		保留	-

3.5.34 启动逻辑 1 中断唤醒使能寄存器 (STARTERP1)

此寄存器选择能将器件从深度睡眠模式和掉电模式唤醒的中断。这些寄存器中的 1 所选定的中断必须在 NVIC 中使能 （表 53）。

STARTERP1 寄存器使能用于唤醒的 WWDT 中断、BOD 中断、USB 唤醒中断和两个 GPIO 组中断。

表 39. 启动逻辑 1 中断唤醒使能寄存器 （STARTERP1，地址 0x4004 8214）位描述

位	符号	值	描述	复位值
11:0			保留。	-
12	WWDTINT		WWDT 中断唤醒	0
		0	已禁用	
		1	使能	
13	BODINT		掉电检测 (BOD) 中断唤醒	0
		0	已禁用	
		1	使能	
18:14	-		保留	-
19	USB_WAKEUP		USB need_clock 信号唤醒	0
		0	已禁用	
		1	使能	
20	GPIOINT0		GPIO 组 0 中断唤醒	0
		0	已禁用	
		1	使能	
21	GPIOINT1		GPIO 组 1 中断唤醒	0
		0	已禁用	
		1	使能	
31:22			保留。	-

3.5.35 深度睡眠模式配置寄存器 (PDSLEEPCFG)

可以编程该寄存器中的位 （BOD_PD 和 WDTOSC_OD），控制深度睡眠模式和掉电模式的各个方面。进入深度睡眠模式或掉电模式后，会将这些位加载到 PDRUNCFG 寄存器的相应位。

注：根据第 3.9.4.1 节和第 3.9.5.1 节中描述的电源配置硬件，强制模拟模块在深度睡眠模式和掉电模式下掉电。 BOD 和看门狗振荡器除外，它们可通过此寄存器配置为继续运行。如果设置了WWDT MOD寄存器（参见表301）中的LOCK位，则会覆盖写入PDSLEEPCFG寄存器的 WDTOSC_PD 值。参见第 17.7 节以了解详细信息。

表 40. 深度睡眠模式配置寄存器 （PDSLEEPCFG，地址 0x4004 8230）位描述

位	符号	值	描述	复位值
2:0			保留。	111
3	BOD_PD		深度睡眠模式和掉电模式下的 BOD 掉电控制	1
		1	掉电	
		0	上电	

表 40. 深度睡眠模式配置寄存器 （PDSLEEPCFG，地址 0x4004 8230）位描述 *（续）*

位	符号	值	描述	复位值
6	WDTOSC_PD		深度睡眠模式和掉电模式下的看门狗振荡器掉电控制	1
		1	掉电	
		0	上电	
31:7	-		保留	-

3.5.36 唤醒配置 (PDAWAKECFG)

从深度睡眠模式或掉电模式唤醒时，此寄存器控制设备的电源配置。

表 41. 唤醒配置 （PDAWAKECFG，地址 0x4004 8234）位描述

位	符号	值	描述	复位值
0	IRCOUT_PD		IRC 振荡器输出唤醒配置	0
		1	掉电	
		0	上电	
1	IRC_PD		IRC 振荡器掉电唤醒配置	0
		1	掉电	
		0	上电	
2	FLASH_PD		闪存唤醒配置	0
		1	掉电	
		0	上电	
3	BOD_PD		BOD 唤醒配置	0
		1	掉电	
		0	上电	
4	ADC_PD		ADC 唤醒配置	1
		1	掉电	
		0	上电	
5	SYSOSC_PD		晶体振荡器唤醒配置	1
		1	掉电	
		0	上电	
6	WDTOSC_PD		看门狗振荡器唤醒配置	1
		1	掉电	
		0	上电	
7	SYSPLL_PD		系统 PLL 唤醒配置	1
		1	掉电	
		0	上电	
8	USBPLL_PD		USB PLL 唤醒配置	1
		0	上电	
		1	掉电	
9	-		保留。始终将此位写为 0。	
10	USBPAD_PD		USB 发送器唤醒配置	1
		0	USB 收发器供电	
		1	USB 收发器掉电	

表 41. 唤醒配置 （PDAWAKECFG，地址 0x4004 8234）位描述 *（续）*

位	符号	值	描述	复位值
11	-		保留。在运行模式中，必须设置此位为 1。	1
12	-		保留。	0
31:13	-		保留	-

3.5.37 电源配置寄存器 (PDRUNCFG)

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以写入该寄存器，而且该写入操作将会立即生效， IRC 的掉电信号除外。

为了避免在 IRC 掉电时产生干扰，IRC 时钟在无干扰时会自动关闭。因此，对于 IRC 来说，在掉电状态生效之前可能会延时。

表 42. 电源配置寄存器 （PDRUNCFG，地址 0x4004 8238）位描述

位	符号	值	描述	复位值
0	IRCOUT_PD		IRC 振荡器输出掉电	0
		1	掉电	
		0	上电	
1	IRC_PD		IRC 振荡器掉电	0
		1	掉电	
		0	上电	
2	FLASH_PD		闪存掉电	0
		1	掉电	
		0	上电	
3	BOD_PD		BOD 掉电	0
		1	掉电	
		0	上电	
4	ADC_PD		ADC 掉电	1
		1	掉电	
		0	上电	
5	SYSOSC_PD		晶体振荡器掉电	1
		1	掉电	
		0	上电	
6	WDTOSC_PD		看门狗振荡器掉电	1
		1	掉电	
		0	上电	
7	SYSPLL_PD		系统 PLL 掉电	1
		1	掉电	
		0	上电	
8	USBPLL_PD		USB PLL 掉电	1
		0	上电	
		1	掉电	
9	-		保留。始终将此位写为 0。	

表 42. 电源配置寄存器 （PDRUNCFG，地址 0x4004 8238）位描述 *（续）*

位	符号	值	描述	复位值
10	USBPAD_PD		USB 收发器掉电配置	1
		0	USB 收发器供电	
		1	USB 收发器掉电 （挂起模式）	
11	-		保留。在运行模式中，必须设置此位为 1。	1
12	-		保留。	0
15:13	-		保留。**始终将这些位写为 111。**	111
31:16	-		保留	-

3.5.38 器件 ID (DEVICE_ID)

该器件 ID 寄存器为只读寄存器，包含每个器件的器件 ID。该寄存器也可以通过 ISP/IAP 命令读取 （参见表 348）。

表 43. 器件 ID （DEVICE_ID，地址 0x4004 83F8）位描述

位	符号	描述	复位值
31:0	DEVICEID	LPC1345FHN33 = 0x2801 0541	取决于零部件
		LPC1345FBD48 = 0x2801 0541	
		LPC1346FHN33 = 0x0801 8542	
		LPC1346FBD48 = 0x0801 8542	
		LPC1347FHN33 = 0x0802 0543	
		LPC1347FBD48 = 0x0802 0543	
		LPC1347FBD64 = 0x0802 0543	
		LPC1315FHN33 = 0x3A01 0523	
		LPC1315FBD48 = 0x3A01 0523	
		LPC1316FHN33 = 0x1A01 8524	
		LPC1316FBD48 = 0x1A01 8524	
		LPC1317FHN33 = 0x1A02 0525	
		LPC1317FBD48 = 0x1A02 0525	
		LPC1317FBD64 = 0x1A02 0525	

3.6 复位

LPC1315/16/17/45/46/47 有 4 个复位源：$\overline{\text{RESET}}$ 引脚、看门狗复位、上电复位 (POR) 和掉电检测 (BOD)。此外，还有一个 ARM 软件复位。

$\overline{\text{RESET}}$ 引脚为施密特触发输入引脚。芯片复位可以由任意一个复位源引起，只要工作电压达到可用电平，就会启动 IRC （可引起复位）来保持有效，直到外部复位取消为止，振荡器运行，同时闪存控制器完成初始化。

当复位源 （ARM 软件复位、 POR、 BOD 复位、外部复位和看门狗复位）有效时，将发起以下进程：

1. IRC 启动。 IRC 启动 （上电时最多 6 µs）以后， IRC 就可以提供稳定的时钟输出。
2. ROM 中的引导代码启动。引导代码可以执行引导任务，也可以跳转到闪存。
3. 闪存上电。闪存上电大约需要 100 µs。然后，闪存初始化序列启动，这大约需要 250 个周期。

当移除内部复位时，处理器将在地址 0 处开始执行，该地址最初是从引导模块映射的复位向量。这时，所有处理器和外设寄存器都已经初始化为预定值。

3.7 启动特性

有关复位后的启动定时，请参见图 4。IRC 是复位后的默认时钟，当电源电压达到 1.8 V 阈值后不久会提供优化系统时钟。

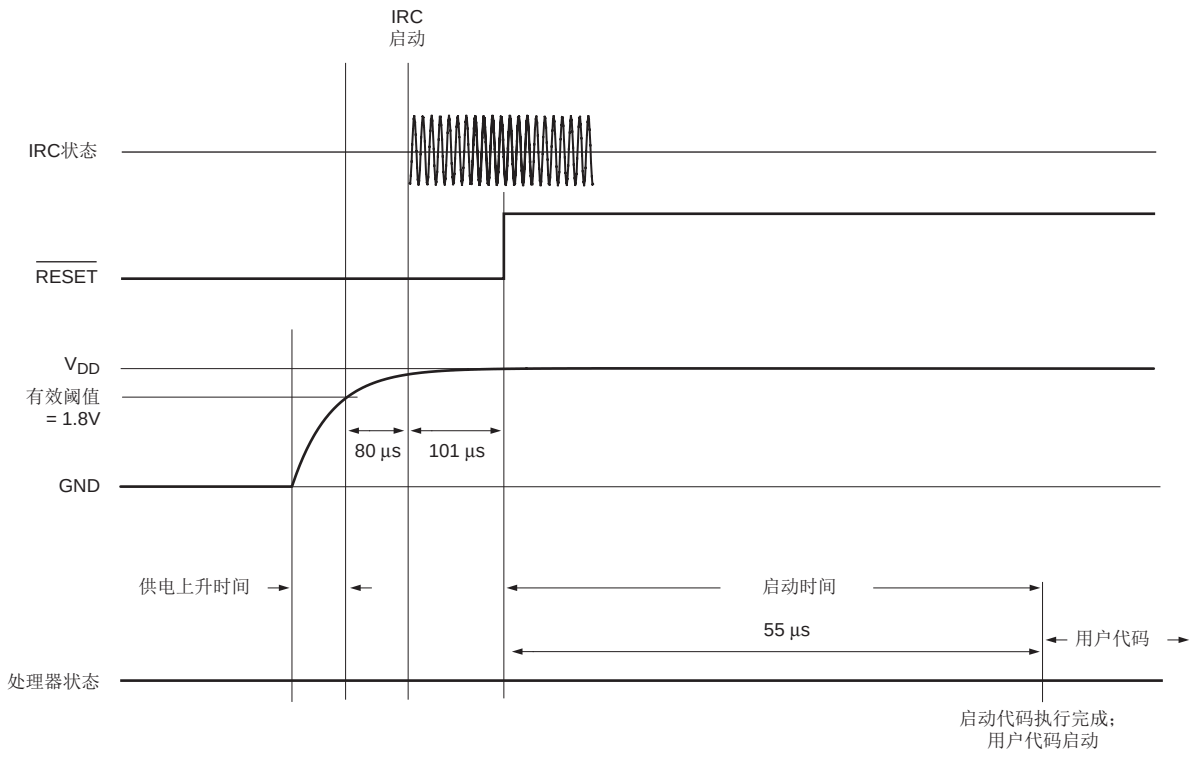


图 4. 启动计时

3.8 掉电检测

LPC1315/16/17/45/46/47 包括四个用于监控 V_{DD} 引脚上电压的电平。如果该电压低于四个所选电平之一，则 BOD 会产生对 NVIC 的中断信号或发出复位，具体取决于 BOD 控制寄存器 （表 31）中 BODRSTENA 位的值。

可针对 NVIC 中断使能寄存器（参见表 348）中的中断来使能该中断信号以产生 CPU 中断；否则，软件可通过读取专用状态寄存器来监控信号。

如果在 STARTERP1 寄存器（参见表 39）和 NVIC 中使能 BOD 中断，BOD 中断可从深度睡眠模式和掉电模式唤醒芯片。

如果使能 BOD 复位，强制 BOD 复位可从深度睡眠模式或掉电模式唤醒芯片。

3.9 电源管理

LPC1315/16/17/45/46/47 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，不同外设运行时处理器有四种特殊的功耗降低模式：睡眠模式、深度睡眠模式、掉电模式和深度掉电模式。

表 44. 低功耗模式中的外设配置

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 [1]	关
IRC 输出	软件可配置	关 [1]	关 [1]	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
WDosc/WWDT	软件可配置	软件可配置	软件可配置	关
ADC	软件可配置	关	关	关
数字外设	软件可配置	关	关	关
USB	软件可配置	关	关	关

[1] 如果设置 WWDT MOD 寄存器中的时钟源锁定位 （位 5），并选择 IRC 作为 WWDT 时钟源，则在此模式中强制 IRC 和 IRC 输出开启 （表 306）。这会增加功耗并可能导致器件不能正确进入掉电模式。有关详情，请参阅第 17.7 节。

注：在睡眠、深度睡眠、掉电或深度掉电模式下，不支持调试模式。

3.9.1 低功耗模式和 WWDT 锁定功能

在任何功耗模式中， WWDT 时钟选择锁定功能都会影响功耗，因为锁定 WWDT 时钟源会强制所选的 WWDT 时钟源开启，独立于通过 PDSLEEPCFG 寄存器进行的深度睡眠模式和掉电模式软件配置。有关详情，请参阅第 17.7 节。

如果器件使用深度睡眠模式的同时保持 WWDT 运行，则看门狗振荡器为首选时钟源，因为它可以最大限度降低功耗。如果时钟源未锁定，则必须使用 PDSLEEPCFG 寄存器为看门狗振荡器供电。或者，可在 WWDT MOD 寄存器中选择并锁定 IRC，强制 IRC 在深度睡眠模式中开启。

如果器件使用掉电模式的同时保持 WWDT 运行，则必须选择看门狗振荡器作为时钟源。如果时钟源未锁定，则必须使用 PDSLEEPCFG 寄存器为看门狗振荡器供电。不要在选择 IRC 时锁定时钟源。

3.9.2 工作模式

在工作模式下， ARM Cortex-M3 内核和存储器由系统时钟计时，而外设由系统时钟或专用外设时钟计时。

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。

3.9.2.1 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 19）。
- 各模拟模块（PLL、振荡器、ADC、BOD 电路和闪存模块）的电源可以通过 PDRUNCFG 寄存器随时单独控制 （表 42）。
- 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 18）来选择。
- 选定的外设（USART、SSP0/1、USB、CLKOUT）使用单独的外设时钟及其自己的时钟分频器。外设时钟可以通过相应的时钟分频寄存器 （表 20 到表 24）来关闭。

3.9.3 睡眠模式

在睡眠模式下， ARM Cortex-M3 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。

3.9.3.1 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。

3.9.3.2 对睡眠模式进行编程

进入睡眠模式的步骤如下：

1. PCON 寄存器中的 PD 位必须设置为默认值 0x0。
2. ARM Cortex-M3 SCR 寄存器中的 SLEEPDEEP 位必须设置为 0。
3. 使用 ARM Cortex-M3 等待中断 (WFI) 指令。

3.9.3.3 从睡眠模式唤醒

在 NVIC 使能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。

3.9.4 深度睡眠模式

在深度睡眠模式下，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。选择看门狗振荡器后，禁用主时钟以及全部外设时钟 （看门狗定时器时钟除外）。 IRC 保持运行，但禁用其输出。闪存处于待机模式。

注：如果设置 WWDT MOD 寄存器 （表 301）中的 LOCK 位，并选择 IRC 作为 WWDT 时钟源，则在深度睡眠模式中， IRC 继续为 WWDT 计时。

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。

3.9.4.1 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 40）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。
- 如果锁定 IRC 作为 WWDT 时钟源（参见第 17.7 节），IRC 将在深度睡眠模式中继续运行并为 WWDT 计时，不受 PDSLEEPCFG 寄存器设置的影响。
- 如果应用程序需要， BOD 电路可在深度睡眠模式下继续运行。

3.9.4.2 对深度睡眠模式进行编程

进入深度睡眠模式的步骤如下：

1. PCON 寄存器中的 PD 位必须设置为 0x1 （表 48）。
2. 选择 PDSLEEPCFG （表 40）寄存器深度睡眠模式下的电源配置。
3. 确定在选择 IRC 时，是否必须锁定 WWDT 时钟源才能覆盖电源配置（参见第 17.7 节）。
4. 如果看门狗振荡器关闭，则应确保 IRC 在 PDRUNCFG 寄存器中上电，并将时钟源切换到 MAINCLKSEL 寄存器 （表 17）中的 IRC。这样可以确保系统时钟无干扰关闭。
5. 在 PDAWAKECFG （表 41）寄存器中唤醒之后选择电源配置。
6. 如果需要任何可用唤醒中断来进行唤醒，应使能中断唤醒寄存器 （表 38、表 39）和 NVIC 中的中断。
7. 向 ARM Cortex-M3 SCR 寄存器中的 SLEEPDEEP 位写入 1。
8. 使用 ARM WFI 指令。

3.9.4.3 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：

- 表35中选择的八个引脚中断之一上的信号。每个引脚中断也必须在STARTERP0寄存器（表 38）和 NVIC 中使能。
- BOD 信号 （如果在 PDSLEEPCFG 寄存器中使能 BOD）：
  - 使用深度睡眠中断唤醒寄存器 1 的 BOD 中断 （表 39）。必须在 NVIC 中使能 BOD 中断。必须在 BODCTRL 寄存器中选择 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 31）中使能 BOD 复位。
- WWDT 信号 （如果在 PDSLEEPCFG 寄存器中使能看门狗振荡器）：
  - 使用中断唤醒寄存器 1 的 WWDT 中断 （表 39）。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 寄存器中设置 WWDT 复位。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 寄存器中使能 WDT。
- 使用中断唤醒寄存器 1 的 USB 唤醒信号 （表 39）。有关详情，请参阅第 10.7.6 节。
- GPIO 分组中断信号 （参见表 39）。

注：如果看门狗振荡器在深度睡眠模式中运行，则其频率决定唤醒时间。

3.9.5 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。选择看门狗振荡器后，禁用主时钟以及全部外设时钟 （看门狗定时器时钟除外） 。IRC 本身和闪存掉电，与深度睡眠模式相比可降低功耗。

注：选择 IRC 作为 WWDT 时钟源时，不要设置 WWDT MOD 寄存器（表 301）中的 LOCK 位。这将阻止器件正确进入掉电模式。

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。唤醒时间比深度睡眠模式更长。

3.9.5.1 掉电模式下的电源配置

掉电模式下的功耗可通过 PDSLEEPCFG （表 40）寄存器中的电源配置设置来配置，方式与深度睡眠模式相同 （参见第 3.9.4.1 节）：

- 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。

3.9.5.2 编程掉电模式

进入掉电模式的步骤如下：

1. PCON 寄存器中的 PD 位必须设置为 0x2 （表 48）。
2. 选择 PDSLEEPCFG （表 40）寄存器掉电模式下的电源配置。
3. 如果设置了 WWDT MOD 寄存器中的锁定位 5（表 301）并选择了 IRC 作为 WWDT 时钟源，则复位器件以清除锁定位，然后选择看门狗振荡器作为 WWDT 时钟源。
4. 如果看门狗振荡器关闭，则应确保 IRC 在 PDRUNCFG 寄存器中上电，并将时钟源切换到 MAINCLKSEL 寄存器 （表 17）中的 IRC。这样可以确保系统时钟无干扰关闭。
5. 在 PDAWAKECFG （表 41）寄存器中唤醒之后选择电源配置。
6. 如果使用任何可用唤醒中断来进行唤醒，应使能中断唤醒寄存器 （表 38、表 39）和 NVIC 中的中断。
7. 向 ARM Cortex-M3 SCR 寄存器中的 SLEEPDEEP 位写入 1。
8. 使用 ARM WFI 指令。

3.9.5.3 从掉电模式唤醒

微控制器可从掉电模式唤醒，方式与深度睡眠模式相同：

- 表35中选择的八个引脚中断之一上的信号。每个引脚中断也必须在STARTERP0寄存器（表 38）和 NVIC 中使能。
- BOD 信号 （如果在 PDSLEEPCFG 寄存器中使能 BOD）：
  - 使用中断唤醒寄存器 1 的 BOD 中断（表 39）。必须在 NVIC 中使能 BOD 中断。必须在 BODCTRL 寄存器中选择 BOD 中断。
  - 从BOD电路复位。这种情况下，必须在BODCTRL寄存器中使能BOD复位（表31）。
- WWDT 信号 （如果在 PDSLEEPCFG 寄存器中使能看门狗振荡器）：
  - 使用中断唤醒寄存器 1 的 WWDT 中断 （表 39）。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 寄存器中设置 WWDT 复位。
- USB 唤醒信号中断唤醒寄存器 1 （表 41）。有关详情，请参阅第 10.7.6 节。
- GPIO 分组中断信号 （参见表 39）。

3.9.6 深度掉电模式

在深度掉电模式下，除了 WAKEUP 引脚之外，整个芯片的电源和时钟都被关闭。深度掉电模式由 PMU 进行控制 （参见第 4 章）。

在深度掉电模式下，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。

在深度掉电模式下，除了 WAKEUP 引脚之外，所有功能引脚都是三态的。

注：设置 PCON 寄存器中的位 3 （第 4.3.1 节）将阻止器件进入深度掉电模式。

3.9.6.1 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、内核和所有外设都已掉电。只有 WAKEUP 引脚处于上电状态。

3.9.6.2 对深度掉电模式进行编程

进入深度掉电模式的步骤如下：

1. 从外部将 WAKEUP 引脚上拉至高电平。
2. 确保清除 PCON 寄存器中的位 3 （表 48）。
3. 向 PCON 寄存器 （参见表 48）中的 PD 位写入 0x3。
4. 存储将要保留在通用寄存器 （第 4.3.2 节）中的数据。
5. 向 ARM Cortex-M3 SCR 寄存器中的 SLEEPDEEP 位写入 1。
6. 在进入深度掉电模式之前，通过将 PDRUNCFG 寄存器中的位 IRCOUT_PD 和 IRC_PD 设置为 0，确保 IRC 上电。
7. 使用 ARM WFI 指令。

3.9.6.3 从深度掉电模式唤醒

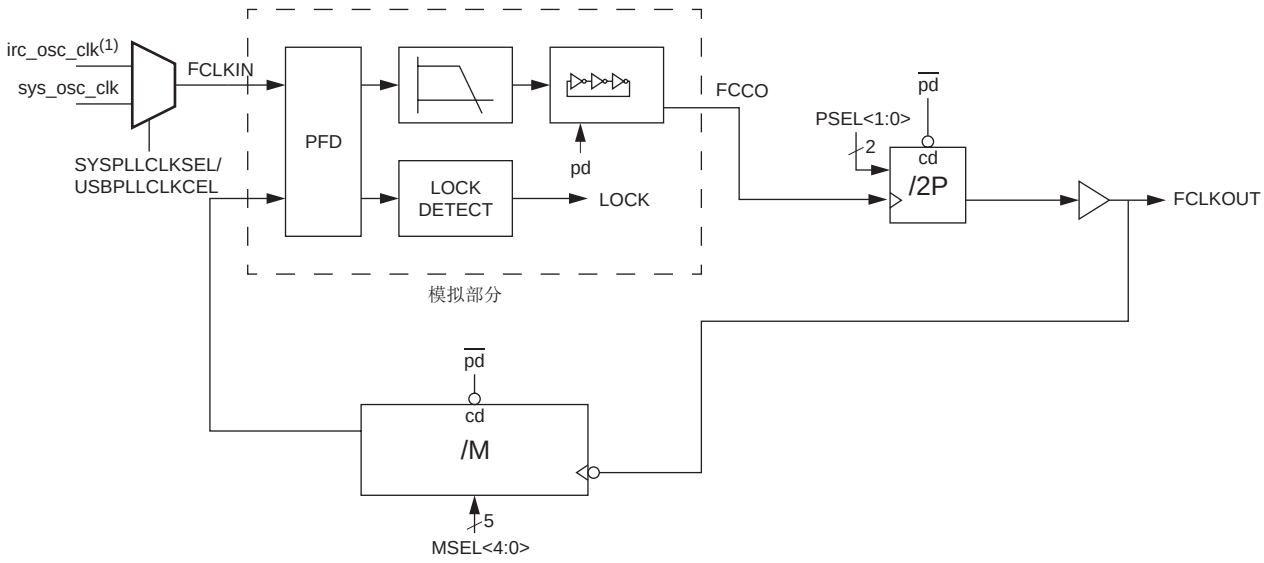
将 WAKEUP 引脚下拉到低电平，会从深度掉电唤醒 LPC1315/16/17/45/46/47，并且芯片将完成整个复位过程 （第 3.6 节）。

1. 在 WAKEUP 引脚上，从高电平跳变到低电平。
   - PMU 将会开启片内调压器。当内核电压达到上电复位 (POR) 的跳变点时，系统就会复位，芯片也将重新启动。
   - 除了 GPREG0 到 GPREG4 之外，所有寄存器都将处于复位状态。
2. 启动芯片之后，可以读取 PCON 寄存器（表 48）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的，而不是冷复位。
3. 清除 PCON 寄存器 （表 48）中的深度掉电标志。
4. （可选）读取通用寄存器 （第 4.3.2 节）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。

3.10 系统 PLL/USB PLL 的功能描述

LPC1315/16/17/45/46/47 利用系统 PLL 为内核和外设创建时钟。相同的 PLL 可用于 USB。



(1) 仅系统 PLL。

图 5. 系统 PLL 功能框图

此 PLL 的功能框图如图 5 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)。此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。锁相环滤波器过滤掉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 到 320 MHz。这些时钟通过可编程的后分频器按 2×P 分频以创建输出时钟，或者直接发送到输出。然后，可编程的反馈分频器将主输出时钟按 M 分频，以生成反馈时钟。锁定探测器还会监控相位频率检测器的输出信号，在 PLL 已锁定到输入时钟时发出信号。

3.10.1 锁定检测器

锁定探测器测量输入时钟上升沿与反馈时钟上升沿之间的相位差异。只有当此差异小于八个以上连续输入时钟期间所谓的锁定标准时，锁定输出才会从低切换到高。单一过大的相位差异会立即复位计数器，并导致锁定信号下降（如果原来较高）。每行要求有八次相位测量值低于某个特定的数，以确保锁定探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，可有效防止虚假的锁定指示，由此确保锁定信号不会出错。

3.10.2 掉电控制

为降低无需 PLL 时钟时的功耗，引入了掉电模式。可通过在掉电配置寄存器中将 SYSPLL_PD 位设置为 1 来使能该模式（表 42）。在此模式下，将关闭内部电流参考，停止振荡器和相位频率检测器，并且分频器将进入复位状态。同时，在掉电模式下，锁定输出将变低以指示 PLL 未处于锁定状态。当通过设置 SYSPLL_PD 位为 0 结束掉电模式时，PLL 将恢复其正常工作，并且一旦输入时钟重新获得锁定，锁定信号将变为高电平。

3.10.3 分频器比率编程

后置分频器

后分频器的分频比由 PSEL 位控制。分频率为 PSEL 位选择的 P 值的两倍，如表 8 和表 10 所示。这样可保证输出时钟达到 50% 的占空比。

反馈分频器

反馈分频器的分频比由 MSEL 位控制。 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，如表 8 和表 10 中所规定的。

更改分频器值

不建议在 PLL 运行时更改分频比。由于无法将 MSEL 和 PSEL 值的变更同步到分频器，因此计数器可能会读到未定义的值，从而造成输出时钟频率不必要地升高或下降。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。

3.10.4 频率选择

PLL 频率的计算公式使用下列参数 （另请参见图 3）：

表 45. PLL 频率参数

参数	系统 PLL
FCLKIN	PLL 时钟多路复用器中 sys_pllclkin （系统 PLL 的输入时钟）的频率 （参见表 15 和表 16）。
FCCO	电流控制振荡器 (CCO) 的频率； 156 到 320 MHz。
FCLKOUT	sys_pllclkout 的频率
P	系统 PLL 的后分频器比； PLL 控制寄存器中的 PSEL 位（参见表 8 和表 10）。
M	系统 PLL 反馈分频寄存器； PLL 控制寄存器中的 MSEL 位（参见表 8 和表 10）。

3.10.4.1 正常模式

在此模式下使能后分频器，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，使得 $FCCO = 2 \times P \times F_{clkout}$。
4. 检查所有的频率和分频器值是否符合表 8 和表 10 中指定的限值。

表 46 显示了如何使用 SYSPLLCTRL 寄存器 （表 8）为 12 MHz 晶体振荡器配置 PLL。将系统时钟分频器 SYSAHBCLKDIV 设为 1 时，主时钟等于系统时钟 （参见表 19）。

表 46. 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
12 MHz	48 MHz	00011（二进制）	4	01（二进制）	2	192 MHz
12 MHz	36 MHz	00010（二进制）	3	10（二进制）	4	288 MHz
12 MHz	24 MHz	00001（二进制）	2	10（二进制）	4	192 MHz

3.10.4.2 掉电模式

在此模式下，将关闭内部电流参考，停止振荡器和相位频率检测器，并且分频器将进入复位状态。同时，在掉电模式下，锁定输出将变低以指示 PLL 未处于锁定状态。当通过设置掉电配置寄存器（表 42）中的 SYSPLL_PD 位为 0 结束掉电模式时， PLL 将恢复其正常工作，并且一旦输入时钟重新获得锁定，锁定信号将变为高电平。

4.1 本章导读

所有LPC1315/16/17/45/46/47器件中的PMU都是一样的。有关电源控制，另请参阅第5章。

4.2 简介

PMU 控制深度掉电模式。 PMU 中的四个通用寄存器可用于在深度掉电模式下保存数据。

4.3 寄存器描述

表 47. 寄存器简介：PMU （基址 0x4003 8000）

名称	访问类型	地址偏移	描述	复位值	参考
PCON	R/W	0x000	电源控制寄存器	0x0	表 48
GPREG0	R/W	0x004	通用寄存器 0	0x0	表 49
GPREG1	R/W	0x008	通用寄存器 1	0x0	表 49
GPREG2	R/W	0x00C	通用寄存器 2	0x0	表 49
GPREG3	R/W	0x010	通用寄存器 3	0x0	表 49
GPREG4	R/W	0x014	通用寄存器 4	0x0	表 50

4.3.1 电源控制寄存器

电源控制寄存器选择是否进入 ARM Cortex-M3 控制的其中一种掉电模式（睡眠模式、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。有关如何进入掉电模式的详情，请参阅第 3.9 节。

表 48. 电源控制寄存器 （PCON，地址 0x4003 8000）位描述

位	符号	值	描述	复位值
2:0	PM		电源模式	000
		0x0	默认。该器件处于工作或睡眠模式。	
		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-M3 内核掉电）。	
3	NODPD		将 0x3 写入上述 PM 字段，设置 SLEEPDEEP 位并执行 WFI 时，该位中的 1 禁止进入深度掉电模式。此位为 1 时， WFI 后会继续执行。 只有在上电复位时才清除该位，因此，向该位写入 1 将锁定处于禁用深度掉电模式的模式中的器件。	0
7:4	-	-	保留。此位不能写 1。	0

表 48. 电源控制寄存器 （PCON，地址 0x4003 8000）位描述 *（续）*

位	符号	值	描述	复位值
8	SLEEPFLAG		睡眼模式标志	0
		0	读：不会进入掉电模式。 LPC1315/16/17/45/46/47 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 位清除为 0。	
10:9	-	-	保留。此位不能写 1。	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：清除深度掉电标志。	
31:12	-	-	保留。此位不能写 1。	0

4.3.2 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 49. 通用寄存器 0 至 3 （GPREG0 - GPREG3，地址 0x4003 8004 至 0x4003 8010）位描述

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

4.3.3 通用寄存器 4

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 4 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。

注：在深度掉电时，如果引脚 V_{DD} 上的外加电压可能降到低于 2.2 V，为了能唤醒芯片，必须在进入深度掉电模式之前，在此寄存器中禁用 WAKEUP 输入引脚的滞回。

表 50. 通用寄存器 4 （GPREG4，地址 0x4003 8014）位描述

位	符号	值	描述	复位值
9:0	-	-	保留。此位不能写 1。	0x0
10	WAKEUPHYS		WAKEUP 引脚滞回使能	0x0
		0	禁用 WAKUP 引脚滞回。	
		1	使能 WAKEUP 引脚滞回。	
31:11	GPDATA		深度掉电模式下保留数据。	0x0

4.4 功能说明

有关进入和退出低功耗模式的详细信息，请参见第 3.9 节。

5.1 本章导读

所有 LPC1315/16/17/45/46/47 器件上均配有功率配置。

5.2 特性

- 包含基于 ROM 的应用服务
- 电源管理服务
- 时钟服务

5.3 描述

通过对功率配置的简单调用，工作模式和睡眠模式下的功耗可针对具体应用进行优化。电源配置程序配置 LPC1315/16/17/45/46/47 的下列某个功耗模式：

- 对应于复位后电源配置的默认模式。
- 对应于优化后处理能力的 CPU 性能模式。
- 对应于电流消耗和 CPU 性能之间优化后平衡的效率模式。
- 对应于最低功耗的低电流模式。

此外，功率配置包括针对给定系统时钟和 PLL 输入时钟选择最佳 PLL 设置的程序。

执行 ROM 驱动器表中指针所指向的函数来进行 ROM 的 API 调用。图 6 显示用于调用功率配置 API 的指针结构。

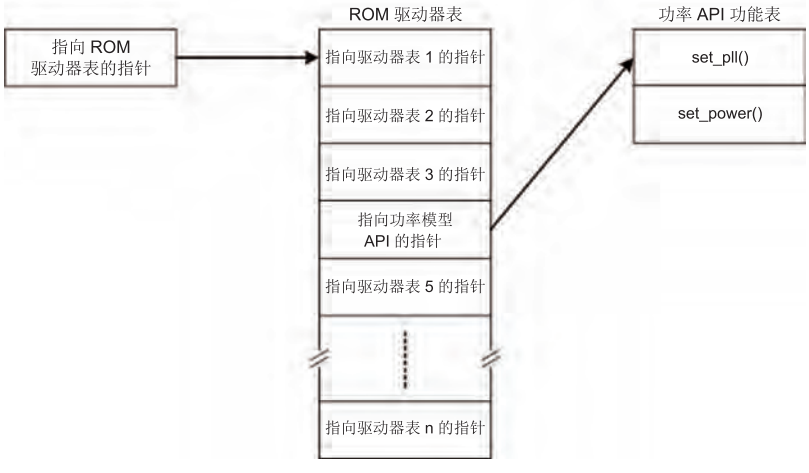


图 6. 功率配置指针结构

5.4 定义

必须在使用功率配置的应用中定义以下元素：

```
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;
unsigned int command[4], result[2];
```

5.5 时钟例程

5.5.1 set_pll

此例程根据调用参数设置系统 PLL。如果可通过简单分频系统 PLL 输入获取预期时钟，则 *set_pll* 将绕过 PLL 以降低系统功耗。

注意事项：调用此例程之前，必须选择 PLL 时钟源 （IRC/ 系统振荡器）（表 15），将主时钟源设为系统 PLL 的输入时钟（表 17），并且必须将系统 AHB 时钟分频器设为 1（表 18）。

set_pll 试图找到匹配调用参数的 PLL 设置。找到反馈分频器值（SYSPLLCTRL，M）、后分频器比率（SYSPLLCTRL，P）和系统 /AHB 时钟分频器 (SYSAHBCLKDIV) 的组合后，*set_pll* 会应用选定值并将主时钟源选择切换到系统 PLL 时钟输出 （必要时）。

此例程返回一个结果代码，指示系统 PLL 已成功设置 (PLL_CMD_SUCCESS) 或未成功设置（此时结果代码将指出问题所在）。还将返回当前系统频率值。此应用应使用此信息调整设备中的其他时钟 （SSP、 UART 和 USB 时钟，和 / 或时钟输出）。

表 51. 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：**预期系统时钟 （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 锁定超时
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）

调用 set_pll 功率例程时需要以下定义：

```
/* set_pll mode options */
#define    CPU_FREQ_EQU          0
#define    CPU_FREQ_LTE          1
#define    CPU_FREQ_GTE          2
#define    CPU_FREQ_APPROX       3
/* set_pll result0 options */
#define    PLL_CMD_SUCCESS       0
#define    PLL_INVALID_FREQ      1
#define    PLL_INVALID_MODE      2
#define    PLL_FREQ_NOT_FOUND    3
#define    PLL_NOT_LOCKED        4
```

5.5.1.1 系统 PLL 输入频率和预期系统时钟

set_pll 寻找系统 PLL 时钟不超过 50 MHz 的设置。当预期系统时钟与系统 PLL 输入频率之间的比率是整数值时，可以轻松找到解决方案，但在其他情况下也能找到解决方案。

系统 PLL 输入频率 (*Param0*) 必须在 10000 （含）到 25000 kHz （含）（10 MHz 到 25 MHz）之间。预期系统时钟 (*Param1*) 必须在 1 （含）到 50000 kHz （含）之间。上述任一要求不满足，*set_pll* 都将返回 PLL_INVALID_FREQ，并将 *Param0* 作为 *Result1* 返回，因为 PLL 的设置保持不变。

5.5.1.2 模式

set_pll 的第一优先级是找到可以完全按照 *Param1* 中指定的比率生成系统时钟的设置。如果不可能找到完全匹配的设置，应使用输入参数模式 (*Param2*) 指定实际系统时钟可小于或等于、大于或等于还是约等于指定为预期系统时钟 (*Param1*) 的值。

仅当 PLL 可以准确输出 *Param1* 请求的频率时，指定 CPU_FREQ_EQU 的调用才会成功。

如果不应超越请求的频率 （由于总电流消耗和 / 或功率预算等原因），则可使用 CPU_FREQ_LTE。

CPU_FREQ_GTE 可为需要最低水平 CPU 处理能力的应用提供帮助。

CPU_FREQ_APPROX 会产生一个和请求值尽可能接近的系统时钟（可能大于或小于请求值）。

如果指定了非法模式， *set_pll* 将返回 PLL_INVALID_MODE。如果预期系统时钟不在此例程支持的范围内， *set_pll* 将返回 PLL_FREQ_NOT_FOUND。这种情况下，当前 PLL 设置不会改变，并且 *Param0* 作为 *Result1* 返回。

5.5.1.3 系统 PLL 锁定超时

若选择了有效的配置，系统 PLL 的锁定时间应该不超过 100 µs。如果 *Param3* 为 0，*set_pll* 将无限期等待 PLL 锁定。如果提供非零值，该值表示返回 PLL_NOT_LOCKED 前，代码检查成功PLL锁定事件的次数。这种情况下，PLL 的设置不会改变，并且*Param0*作为*Result1*返回。

提示：将 *Param3* 设置为系统 PLL 频率 [Hz] 除以 10000 的值，将提供绰绰有余的 PLL 锁定轮询周期。

5.5.1.4 代码示例

以下示例说明了上文讨论的 *set_pll* 的部分功能。

5.5.1.4.1 无效频率 （超过设备的最大时钟频率）

```
command[0] = 12000;
command[1] = 60000;
command[2] = CPU_FREQ_EQU;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

以上代码指定了 12 MHz PLL 输入时钟和正好 60 MHz 的系统时钟。此应用准备无限期等待 PLL 锁定。但 60 MHz 的预期系统时钟超过了最大值 50 MHz。因此，*set_pll* 在 *result[0]* 中返回 PLL_INVALID_FREQ，在 *result[1]* 中返回 12000，不会改变 PLL 的设置。

5.5.1.4.2 无效频率选择 （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);
```

以上代码指定了 12 MHz PLL 输入时钟，不超过 40 kHz 的系统时钟，且等待 PLL 锁定时无超时。由于时钟系统的最大分频器值为 255，而以 40 kHz 运行将需要除以 300，因此 *set_pll* 在 *result[0]* 中返回 PLL_INVALID_FREQ，在 *result[1]* 中返回 12000，不会改变 PLL 的设置。

5.5.1.4.3 找不到精确的解决方案 (PLL)

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_EQU;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

以上代码指定了 12 MHz PLL 输入时钟和正好 25 MHz 的系统时钟。此应用准备无限期等待 PLL 锁定。由于在上文提到的限制内没有有效的 PLL 设置，因此 *set_pll* 在 *result[0]* 中返回 PLL_FREQ_NOT_FOUND，在 *result[1]* 中返回 12000，不会改变 PLL 的设置。

5.5.1.4.4 系统时钟小于或等于预期值

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_LTE;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

以上代码指定了 12 MHz PLL 输入时钟，不超过 25 MHz 的系统时钟，无锁定超时。*set_pll* 在 *result[0]* 中返回 PLL_CMD_SUCCESS，在 *result[1]* 中返回 24000。新的系统时钟为 24 MHz。

5.5.1.4.5 系统时钟大于或等于预期值

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_GTE;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

以上代码指定了 12 MHz PLL 输入时钟，不小于 25 MHz 的系统时钟，无锁定超时。*set_pll* 在 *result[0]* 中返回 PLL_CMD_SUCCESS，在 *result[1]* 中返回 36000。新的系统时钟为 36 MHz。

5.5.1.4.6 系统时钟约等于预期值

```
command[0] = 12000;
command[1] = 16500;
command[2] = CPU_FREQ_APPROX;
command[3] = 0;
(*rom)->pWRD->set_pll(command, result);
```

以上代码指定了 12 MHz PLL 输入时钟，约等于 16.5 MHz 的系统时钟，无锁定超时。*set_pll* 在 *result[0]* 中返回 PLL_CMD_SUCCESS，在 *result[1]* 中返回 16000。新的系统时钟为 16 MHz。

5.6 功率例程

5.6.1 set_power

此例程根据调用参数配置设备的内部功率控制设置。其目标是降低有源功耗，同时使关乎到应用的功能接近最佳水平。

set_power 返回一个结果代码，报告功率设置是否成功改变。

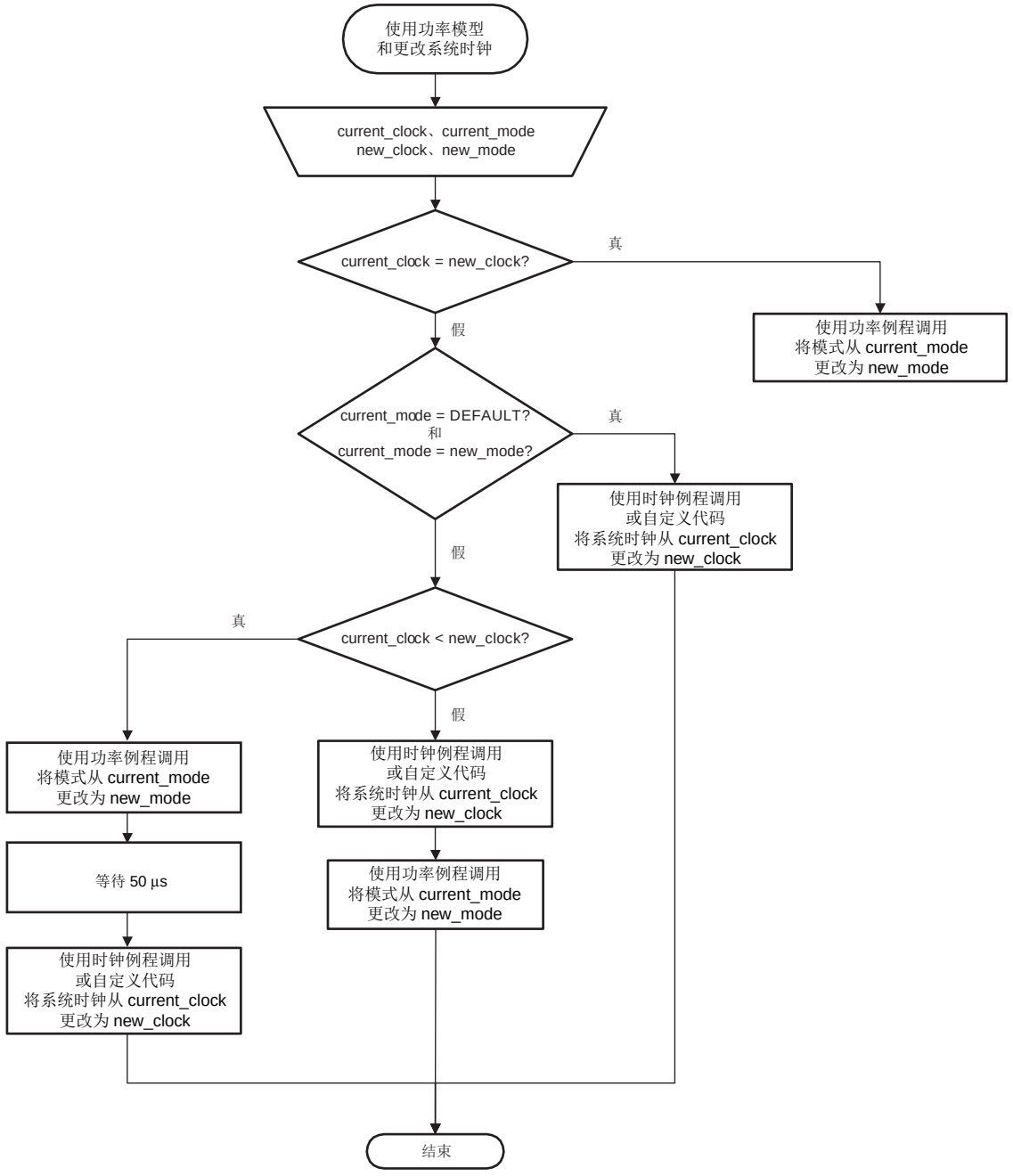


图 7.　功率配置的使用

表 52. set_power 例程

例程	set_power
输入	**参数 0**：新系统时钟 （单位：MHz） **参数 1**：模式 （PWR_DEFAULT、 PWR_CPU_PERFORMANCE、 PWR_EFFICIENCY、 PWR_LOW_CURRENT） **参数 2**：当前系统时钟 （单位：MHz）
结果	**结果 0**：PWR_CMD_SUCCESS | PWR_INVALID_FREQ | PWR_INVALID_MODE

调用 set_pll 例程需要以下定义：

```
/* set_power mode options */
#define    PWR_DEFAULT          0
#define    PWR_CPU_PERFORMANCE  1
#define    PWR_EFFICIENCY       2
#define    PWR_LOW_CURRENT      3
/* set_power result0 options */
#define    PWR_CMD_SUCCESS      0
#define    PWR_INVALID_FREQ     1
#define    PWR_INVALID_MODE     2
```

5.6.1.1 新系统时钟

新系统时钟是指成功执行时钟例程调用或用户提供相似代码后，微控制器运行的时钟频率。操作数必须是 1（含）到 50 MHz（含）之间的整数。如果提供的值不在此范围内，*set_power* 将返回 PWR_INVALID_FREQ，且不会改变功率控制系统。

5.6.1.2 模式

输入参数模式 (*Param1*) 指定四个可用功率设置之一。如果提供了非法选择，*set_power* 将返回 PWR_INVALID_MODE，且不会改变功率控制系统。

PWR_DEFAULT 将设备保持在和复位状态相似的基准功率设置。

PWR_CPU_PERFORMANCE 配置微控制器，以便其可以为应用提供更多处理能力。CPU 性能优于默认选项 30%。

PWR_EFFICIENCY 设置旨在找到有源电流和 CPU 执行代码和处理数据的能力之间的平衡。这种模式下，设备性能高于默认模式，既可以提供更好的 CPU 性能，又能降低有源电流。

PWR_LOW_CURRENT 针对的是注重降低功耗而不是注重 CPU 性能的解决方案。

5.6.1.3 当前系统时钟

当前系统时钟是指调用 *set_power* 时，微控制器运行的时钟频率。此参数是 1 （含）到 50 MHz （含）之间的整数。

5.6.1.4 代码示例

以下示例说明了上文讨论的部分 *set_power* 功能。

5.6.1.4.1 无效频率 （超过设备的最大时钟频率）

```
command[0] = 55;
command[1] = PWR_CPU_PERFORMANCE;
command[2] = 12;
(*rom)->pWRD->set_power(command, result);
```

以上设置将用于以 12 MHz 运行、尝试切换到 55 MHz 系统时钟的系统，需要最大 CPU 处理能力。由于指定的 55 MHz 时钟超出最大值 50 MHz， *set_power* 在 *result[0]* 中返回 PWR_INVALID_FREQ，不会改变现有功率设置中的任何内容。

5.6.1.4.2 适用的功率设置

```
command[0] = 24;
command[1] = PWR_CPU_EFFICIENCY;
command[2] = 12;
(*rom)->pWRD->set_power(command, result);
```

以上代码指定了某个以 12 MHz 运行的应用将切换到 24 MHz 的系统时钟，强调效率。配置微控制器的内部功率控制功能后，*set_power* 在 *result[0]* 中返回 PWR_CMD_SUCCESS。

6.1 本章导读

USB 相关的中断 #22、 23 和 30 仅在 LPC134x 器件中可用。

6.2 简介

可嵌套中断向量控制器 (NVIC) 是 Cortex-M3 不可或缺的一部分。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。

6.3 特性

- 可嵌套中断向量控制器是 ARM Cortex-M3 不可或缺的一部分
- 紧密连接的中断控制器提供低中断延迟
- 控制系统异常和外设中断
- NVIC 支持最多 32 个向量中断
- 8 个可编程的中断优先级，带硬件优先级屏蔽
- 软件中断生成
- 支持 NMI

6.4 中断源

表 53 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。除 ARM 的特定标准外，中断线的连接位置没有重要性或优先级的区别。

有关 NVIC 寄存器位描述，请参见第 21.5.2 节。

表 53. 中断源与中断向量控制器的连接

异常编号	名称	描述	标志
0	PIN_INT0	GPIO 引脚中断 0	-
1	PIN_INT1	GPIO 引脚中断 1	-
2	PIN_INT2	GPIO 引脚中断 2	-
3	PIN_INT3	GPIO 引脚中断 3	-
4	PIN_INT4	GPIO 引脚中断 4	-
5	PIN_INT5	GPIO 引脚中断 5	-
6	PIN_INT6	GPIO 引脚中断 6	-
7	PIN_INT7	GPIO 引脚中断 7	-
8	GINT0	GPIO 组合 0 中断	-
9	GINT1	GPIO 组合 1 中断	-
11 至 10	-	-	保留
12	RIT	RIT 中断	保留
13	-	-	保留

表 53. 中断源与中断向量控制器的连接 *（续）*

异常编号	名称	描述	标志
14	SSP1	SSP1 中断	Tx FIFO 半空 Rx FIFO 半满 Rx 超时 Rx 溢出
15	I2C	I2C 中断	SI （状态更改）
16	CT16B0	CT16B0 中断	匹配 0 - 2 捕获 0
17	CT16B1	CT16B1 中断	匹配 0 - 1 捕获 0
18	CT32B0	CT32B0 中断	匹配 0 - 3 捕获 0
19	CT32B1	CT32B1 中断	匹配 0 - 3 捕获 0
20	SSP0	SSP0 中断	Tx FIFO 半空 Rx FIFO 半满 Rx 超时 Rx 溢出
21	USART	USART 中断	Rx 线状态 (RLS) 发送保持寄存器空 (THRE) Rx 数据可用 (RDA) 字符超时指示 (CTI) 自动波特率结束 (ABEO) 自动波特率超时 (ABTO) 调制解调器控制中断
22	USB_IRQ	USB_IRQ 中断	USB IRQ 中断
23	USB_FIQ	USB_FIQ 中断	USB FIQ 中断
24	ADC	ADC 中断	A/D 转换器转换结束
25	WWDT	WWDT 中断	窗口看门狗中断 (WDINT)
26	BOD	BOD 中断	掉电检测
27	闪存	闪存中断	-
28	-	-	保留
29	-	-	保留
30	USB_WAKEUP	USB_WAKEUP 中断	USB 唤醒中断
31	-	-	保留

6.5 寄存器描述

请参见 《ARM Cortex-M3 技术参考手册》。

7.1 本章导读

IOCON 寄存器镜像取决于封装类型 （参见表 54）。保留不可用引脚的寄存器。

表 54. IOCON 寄存器可用

封装	端口 0	端口 1
LQFP64	PIO0_0 至 PIO0_23	PIO1_0 至 PIO1_5； PIO1_7 至 PIO1_8； PIO1_10 至 PIO1_29
LQFP48	PIO0_0 至 PIO0_23	PIO1_13 至 PIO1_16； PIO1_19 至 PIO1_23 至 PIO1_29； PIO1_31
HVQFN（无USB）	PIO0_0 至 PIO0_23	PIO1_15； PIO1_19； PIO1_23 至 PIO1_24
HVQFN (USB)	PIO0_0 至 PIO0_23	PIO1_15； PIO1_19

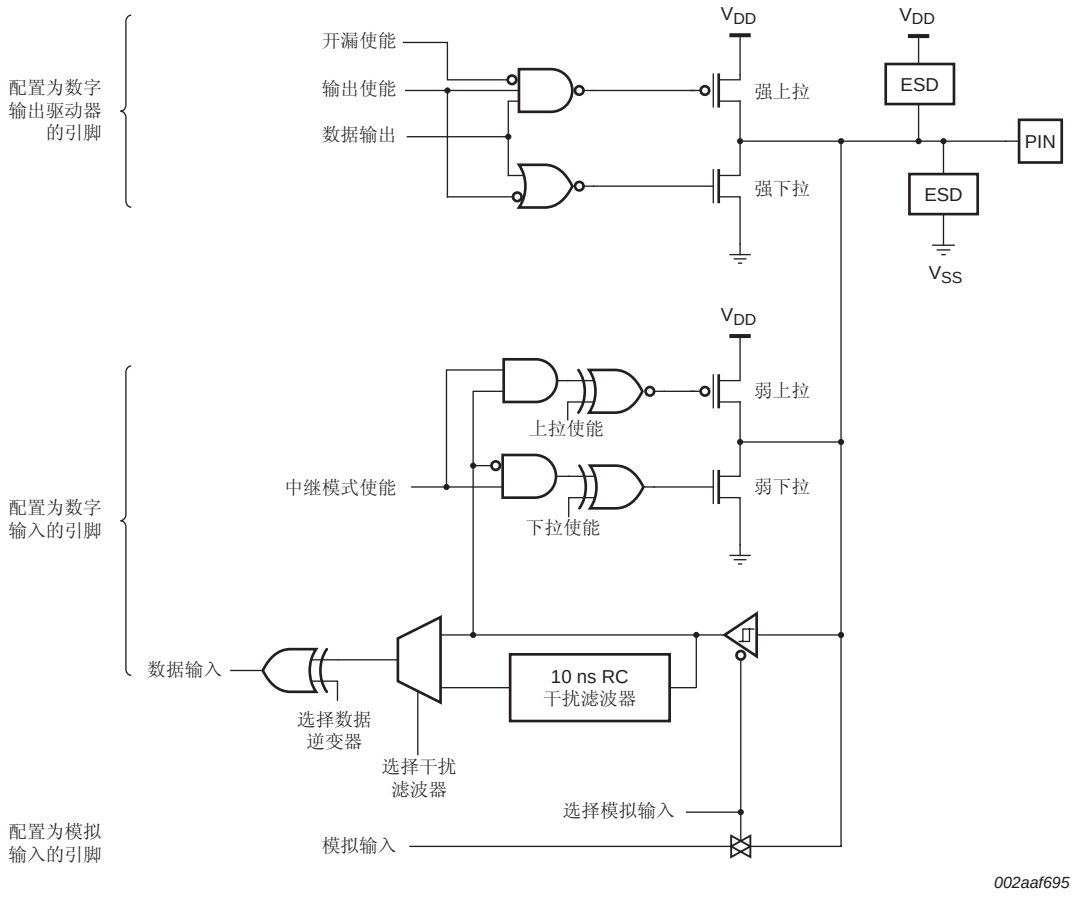
7.2 简介

I/O 配置寄存器控制焊盘的电气特性。可编程下列特性：

- 引脚功能
- 内部上拉 / 下拉电阻或总线保持器功能 （中继模式）
- 用于标准 I/O 引脚的开漏模式
- 滞回
- 输入逆变器
- 选定引脚的干扰滤波器
- 托管 ADC 输入的焊盘的模拟输入或数字模式
- 托管 I^2C 总线功能的焊盘的 I^2C 模式

7.3 简介

IOCON 寄存器控制端口引脚的功能 （GPIO 或外设功能）和电气特性 （参见图 8）。



10 ns 干扰滤波器仅适用于选定引脚。

图 8. 标准 I/O 引脚配置

7.3.1 引脚功能

IOCON 寄存器中的 FUNC 位可设为 GPIO (FUNC = 0) 或外设功能。如果引脚为 GPIO 引脚，则 DIR 寄存器决定引脚是配置为输入还是输出（见第 9.5.3.3 节）。对于任何外设功能，引脚方向根据引脚的功能自动控制。 DIR 寄存器对外设功能没有影响。

7.3.2 引脚模式

IOCON 寄存器的 MODE 位可为每个引脚选择片内上拉或下拉电阻，或者选择中继模式。

片内电阻配置可能是上拉使能、下拉使能，或是两者都不使能。默认值为上拉使能。

当引脚处于逻辑高电平时，中继模式会使能上拉电阻；当引脚处于逻辑低电平时，则使能下拉电阻。这样，当引脚配置为输入且不由外部驱动时，会使引脚保持其上一已知状态。状态保持不适用于深度掉电模式。如果暂时不驱动引脚，通常可用中继模式来防止引脚悬空（当引脚处于不确定状态时，可能会使用大量电量）。

7.3.3 滞回

可通过 IOCON 寄存器将数字功能的输入缓冲配置为具有滞回或配置为普通缓冲。

如果外部焊盘电源电压 V_{DD} 介于 2.5 V 和 3.6 V 之间，可使能或禁用滞回缓冲。如果 V_{DD} 低于 2.5 V，必须**禁用**滞回缓冲，才可使用输入模式的引脚。

7.3.4 输入逆变器

如果输入逆变器使能，高电平引脚电平反转至 0，低电平引脚电平反转至 1。

7.3.5 输入干扰滤波器

选定引脚 （引脚 PIO0_22、 PIO0_23 和 PIO0_11 至 PIO0_16）可选择打开或关闭 10 ns 输入干扰滤波器。干扰滤波器默认为关闭状态。 RESET 引脚包括一个 20 ns 干扰滤波器（不可配置）。

7.3.6 开漏模式

所有数字引脚都支持伪开漏模式。请注意，除 I^2C 总线引脚外，这不是真正的开漏模式。

7.3.7 模拟模式

在模拟模式下，数字接收器断开以获得准确的输入电压用于模数转换。该模式可在用模拟功能控制引脚的 IOCON 寄存器中选择。选择模拟模式时，滞回、引脚模式、逆变器、干扰滤波器和开漏设置无效。

对于没有模拟功能的引脚，模拟模式设置无效。

7.3.8 I^2C 模式

如果寄存器 PIO0_4 （表 60）和 PIO0_5 （表 61）的 FUNC 位选择了 I^2C 功能，则 I^2C 总线引脚可配置为不同 I^2C 模式：

- 带50 ns输入干扰滤波器的标准模式/快速模式I^2C。可单独配置符合I^2C总线规范的开漏输出。
- 带 50 ns 输入干扰滤波器的超快速模式 I^2C。在此模式下，引脚用作高电流接收器。可单独配置符合 I^2C 总线规范的开漏输出。
- 不带输入滤波器的标准功能。

注：当引脚用作 GPIO 引脚时，应选择标准模式 / 快速模式 I^2C 或标准 I/O 功能。

7.3.9 RESET 引脚 （引脚 RESET_PIO0_0）

关于复位焊盘配置，请参见图9。在深度掉电模式下，不能使用RESET功能。使用WAKEUP引脚复位芯片和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。此复位引脚包括一个固定 20 ns 干扰滤波器。

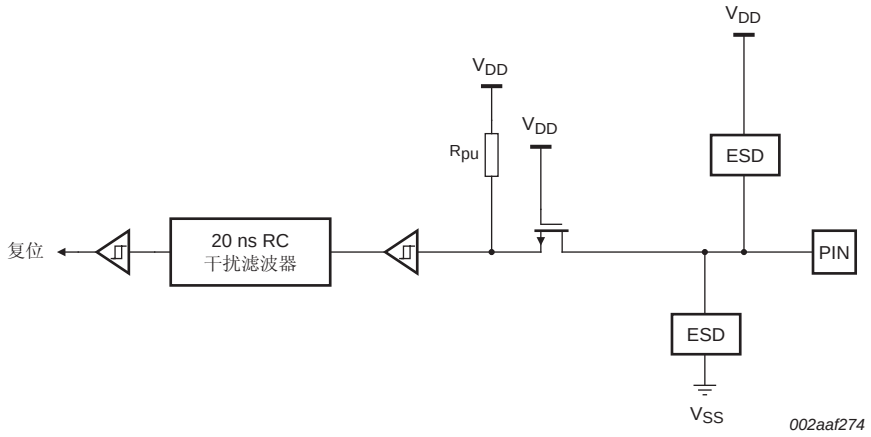


图 9. 复位焊盘配置

7.3.10 WAKEUP 引脚 （引脚 PIO0_16）

WAKEUP 引脚与 PIO0_16 引脚组合，包括一个 20 ns 固定干扰滤波器。要进入深度掉电模式，必须从外部将此引脚上拉到高电平；要退出深度掉电模式，必须将其下拉到低电平。以短至 50 ns 的下降脉冲唤醒器件。

7.4 寄存器描述

表 55. 寄存器简介：IOCON （基址：0x4004 4000）

名称	访问类型	地址偏移	描述	复位值	参考
RESET_PIO0_0	读 - 写	0x000	引脚 RESET/PIO0_0 的 I/O 配置	0x0000090	表 56
PIO0_1	读 - 写	0x004	引脚 PIO0_1/CLKOUT/CT32B0_MAT2/ USB_FTOGGLE 的 I/O 配置	0x0000090	表 57
PIO0_2	读 - 写	0x008	引脚 PIO0_2/SSEL0/CT16B0_CAP0 的 I/O 配置	0x0000090	表 58
PIO0_3	读 - 写	0x00C	引脚 PIO0_3/USB_VBUS 的 I/O 配置	0x0000090	表 59
PIO0_4	读 - 写	0x010	引脚 PIO0_4/SCL 的 I/O 配置	0x0000080	表 60
PIO0_5	读 - 写	0x014	引脚 PIO0_5/SDA 的 I/O 配置	0x0000080	表 61
PIO0_6	读 - 写	0x018	引脚 PIO0_6/USB_CONNECT/SCK0 的 I/O 配置	0x0000090	表 62
PIO0_7	读 - 写	0x01C	引脚 PIO0_7/CTS 的 I/O 配置	0x0000090	表 63
PIO0_8	读 - 写	0x020	引脚 PIO0_8/MISO0/CT16B0_MAT0/ ARM_TRACE_CLK 的 I/O 配置	0x0000090	表 64
PIO0_9	读 - 写	0x024	引脚 PIO0_9/MOSI0/CT16B0_MAT1/ ARM_TRACE_SWV 的 I/O 配置	0x0000090	表 65
SWCLK_PIO0_10	读 - 写	0x028	引脚 SWCLK/PIO0_10/SCK0/CT16B0_MAT2 的 I/O 配置	0x0000090	表 66
TDI_PIO0_11	读 - 写	0x02C	引脚 TDI/PIO0_11/AD0/CT32B0_MAT3 的 I/O 配置	0x0000090	表 67

表 55. 寄存器简介：IOCON （基址：0x4004 4000）*（续）*

名称	访问类型	地址偏移	描述	复位值	参考
TMS_PIO0_12	读 - 写	0x030	引脚 TMS/PIO0_12/AD1/CT32B1_CAP0 的 I/O 配置	0x0000090	表 68
TDO_PIO0_13	读 - 写	0x034	引脚 TDO/PIO0_13/AD2/CT32B1_MAT0 的 I/O 配置	0x0000090	表 69
TRST_PIO0_14	读 - 写	0x038	引脚 TRST/PIO0_14/AD3/CT32B1_MAT1 的 I/O 配置	0x0000090	表 70
SWDIO_PIO0_15	读 - 写	0x03C	引脚 SWDIO/PIO0_15/AD4/CT32B1_MAT2 的 I/O 配置	0x0000090	表 71
PIO0_16	读 - 写	0x040	引脚 PIO0_16/AD5/CT32B1_MAT3/WAKEUP 的 I/O 配置	0x0000090	表 72
PIO0_17	读 - 写	0x044	引脚 PIO0_17/RTS/CT32B0_CAP0/SCLK 的 I/O 配置	0x0000090	表 73
PIO0_18	读 - 写	0x048	引脚 PIO0_18/RXD/CT32B0_MAT0 的 I/O 配置	0x0000090	表 74
PIO0_19	读 - 写	0x04C	引脚 PIO0_19/TXD/CT32B0_MAT1 的 I/O 配置	0x0000090	表 75
PIO0_20	读 - 写	0x050	引脚 PIO0_20/CT16B1_CAP0 的 I/O 配置	0x0000090	表 76
PIO0_21	读 - 写	0x054	引脚 PIO0_21/CT16B1_MAT0/MOSI1 的 I/O 配置	0x0000090	表 77
PIO0_22	读 - 写	0x058	引脚 PIO0_22/AD6/CT16B1_MAT1/MISO1 的 I/O 配置	0x0000090	表 78
PIO0_23	读 - 写	0x05C	引脚 PIO0_23/AD7 的 I/O 配置	0x0000090	表 79
PIO1_0	读 - 写	0x060	引脚 PIO1_0/CT32B1_MAT0 的 I/O 配置	0x0000090	表 80
PIO1_1	读 - 写	0x064	引脚 PIO1_1/CT32B1_MAT1 的 I/O 配置	0x0000090	表 81
PIO1_2	读 - 写	0x068	引脚 PIO1_2/CT32B1_MAT2 的 I/O 配置	0x0000090	表 82
PIO1_3	读 - 写	0x06C	引脚 PIO1_3/CT32B1_MAT3 的 I/O 配置	0x0000090	表 83
PIO1_4	读 - 写	0x070	引脚 PIO1_4/CT32B1_CAP0 的 I/O 配置	0x0000090	表 84
PIO1_5	读 - 写	0x074	引脚 PIO1_5/CT32B1_CAP1 的 I/O 配置	0x0000090	表 85
-	-	0x078	保留	-	-
PIO1_7	读 - 写	0x07C	引脚 PIO1_7 的 I/O 配置	0x0000090	表 86
PIO1_8	读 - 写	0x080	引脚 PIO1_8 的 I/O 配置	0x0000090	表 87
-	-	0x084	保留	-	-
PIO1_10	读 - 写	0x088	引脚 PIO1_10 的 I/O 配置	0x0000090	表 88
PIO1_11	读 - 写	0x08C	引脚 PIO1_11 的 I/O 配置	0x0000090	表 89
-	-	0x090	保留	-	-
PIO1_13	访问类型	0x094	PIO1_13/DTR/CT16B0_MAT0/TXD 的 I/O 配置	0x0000090	表 90
PIO1_14	访问类型	0x098	PIO1_14/DSR/CT16B0_MAT1/RXD 的 I/O 配置	0x0000090	表 91
PIO1_15	读 - 写	0x09C	引脚 PIO1_15/DCD/ CT16B0_MAT2/SCK1 的 I/O 配置	0x0000090	表 92
PIO1_16	读 - 写	0x0A0	引脚 PIO1_16/RI/CT16B0_CAP0 的 I/O 配置	0x0000090	表 93
PIO1_17	访问类型	0x0A4	PIO1_17/CT16B0_CAP1/RXD 的 I/O 配置	0x0000090	表 94
PIO1_18	访问类型	0x0A8	PIO1_18/CT16B1_CAP1/TXD 的 I/O 配置	0x0000090	表 95
PIO1_19	读 - 写	0x0AC	引脚 PIO1_19/DTR/SSEL1 的 I/O 配置	0x0000090	表 96
PIO1_20	读 - 写	0x0B0	引脚 PIO1_20/DSR/SCK1 的 I/O 配置	0x0000090	表 97
PIO1_21	读 - 写	0x0B4	引脚 PIO1_21/DCD/MISO1 的 I/O 配置	0x0000090	表 98

表 55. 寄存器简介：IOCON （基址：0x4004 4000）*（续）*

名称	访问类型	地址偏移	描述	复位值	参考
PIO1_22	读 - 写	0x0B8	引脚 PIO1_22/RI/MOSI1 的 I/O 配置	0x0000090	表 99
PIO1_23	读 - 写	0x0BC	引脚 PIO1_23/CT16B1_MAT1/SSEL1 的 I/O 配置	0x0000090	表 100
PIO1_24	读 - 写	0x0C0	引脚 PIO1_24/ CT32B0_MAT0 的 I/O 配置	0x0000090	表 101
PIO1_25	读 - 写	0x0C4	引脚 PIO1_25/CT32B0_MAT1 的 I/O 配置	0x0000090	表 102
PIO1_26	读 - 写	0x0C8	引脚 PIO1_26/CT32B0_MAT2/RXD 的 I/O 配置	0x0000090	表 103
PIO1_27	读 - 写	0x0CC	引脚 PIO1_27/CT32B0_MAT3/TXD 的 I/O 配置	0x0000090	表 104
PIO1_28	读 - 写	0x0D0	引脚 PIO1_28/CT32B0_CAP0/SCLK 的 I/O 配置	0x0000090	表 105
PIO1_29	读 - 写	0x0D4	引脚 PIO1_29/SCK0/ CT32B0_CAP1 的 I/O 配置	0x0000090	表 106
-	-	0x0D8	保留	-	-
PIO1_31	读 - 写	0x0DC	引脚 PIO1_31 的 I/O 配置	0x0000090	表 107

7.4.1 引脚 RESET_PIO0_0 的 I/O 配置

表 56. 引脚 RESET/PIO0_0 的 I/O 配置 （RESET_PIO0_0，地址 0x4004 4000）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	RESET。	
		0x1	PIO0_0。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.2 引脚 PIO0_1 的 I/O 配置

表 57. 引脚PIO0_1/CLKOUT/CT32B0_MAT2/USB_FTOGGLE的I/O配置（PIO0_1，地址0x4004 4004）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	PIO0_1。	
		0x1	CLKOUT。	
		0x2	CT32B0_MAT2。	
		0x3	USB_FTOGGLE。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.3 引脚 PIO0_2 的 I/O 配置

表 58. 引脚 PIO0_2/SSEL0/CT16B0_CAP0 的 I/O 配置 （PIO0_2，地址 0x4004 4008）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO0_2。	
		0x1	SSEL0。	
		0x2	CT16B0_CAP0。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	

表 58. 引脚 PIO0_2/SSEL0/CT16B0_CAP0 的 I/O 配置（PIO0_2，地址 0x4004 4008）位描述 *（续）*

位	符号	值	描述	复位值
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.4 引脚 PIO0_3 的 I/O 配置

表 59. 引脚 PIO0_3/USB_VBUS 的 I/O 配置 （PIO0_3，地址 0x4004 400C）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO0_3。	
		0x1	USB_VBUS。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.5 引脚 PIO0_4 的 I/O 配置

表 60. 引脚 PIO0_4/SCL 的 I/O 配置 （PIO0_4，地址 0x4004 4010）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO0_4 （开漏引脚）。	
		0x1	I2C SCL （开漏引脚）。	
7:3	-		保留。	
9:8	I2CMODE		选择 I2C 模式（见第 6.3.8 节）。如果引脚功能为 GPIO (FUNC = 0)，选择标准模式（I2CMODE = 0，默认）或标准 I/O 功能 (I2CMODE = 1)。	0
		0x0	标准模式 / 快速模式 I2C。	
		0x1	标准 I/O 功能	
		0x2	I2C 超快速模式	
		0x3	保留。	
31:10	-		保留。	-

7.4.6 引脚 PIO0_5 的 I/O 配置

表 61. 引脚 PIO0_5/SDA 的 I/O 配置 （PIO0_5，地址 0x4004 4014）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO0_5 （开漏引脚）。	
		0x1	I2C SDA （开漏引脚）。	
7:3			保留。	
9:8	I2CMODE		选择 I2C 模式（见第 6.3.8 节）。如果引脚功能为 GPIO (FUNC = 0)，选择标准模式（I2CMODE = 00，默认）或标准 I/O 功能 (I2CMODE = 01)。	0
		0x0	标准模式 / 快速模式 I2C。	
		0x1	标准 I/O 功能	
		0x2	I2C 超快速模式	
		0x3	保留。	
31:10	-		保留。	-

7.4.7 引脚 PIO0_6 的 I/O 配置

表 62. 引脚 PIO0_6/USB_CONNECT/SCK0 的 I/O 配置 （PIO0_6，地址 0x4004 4018）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO0_6。	
		0x1	USB_CONNECT。	
		0x2	SCK0。	

表 62. 引脚PIO0_6/USB_CONNECT/SCK0的I/O配置（PIO0_6，地址0x4004 4018）位描述*（续）*

位	符号	值	描述	复位值
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.8 引脚 PIO0_7 的 I/O 配置

表 63. 引脚 PIO0_7/CTS 的 I/O 配置 （PIO0_7，地址 0x4004 401C）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO0_7。	
		0x1	CTS。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.9 引脚 PIO0_8 的 I/O 配置

表 64. 引脚PIO0_8/MISO0/CT16B0_MAT0/ARM_TRACE_CLK的I/O配置（PIO0_8，地址0x4004 4020）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	PIO0_8。	
		0x1	MISO0。	
		0x2	CT16B0_MAT0。	
		0x3	ARM_TRACE_CLK	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.10 引脚 PIO0_9 的 I/O 配置

表 65. 引脚 PIO0_9/MOSI0/CT16B0_MAT1/ARM_TRACE_SWV 的 I/O 配置 （PIO0_9，地址 0x4004 4024）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	PIO0_9。	
		0x1	MOSI0。	
		0x2	CT16B0_MAT1。	
		0x3	ARM_TRACE_SWV	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	

表 65. 引脚 PIO0_9/MOSI0/CT16B0_MAT1/ARM_TRACE_SWV 的 I/O 配置（PIO0_9，地址 0x4004 4024）位描述 *（续）*

位	符号	值	描述	复位值
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向（引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向（引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.11 引脚 SWCLK/PIO0_10 的 I/O 配置

表 66. 引脚SWCLK/PIO0_10/ SCK0/CT16B0_MAT2的I/O配置（SWCLK_PIO0_10，地址0x4004 4028）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	SWCLK。	
		0x1	PIO0_10。	
		0x2	SCK0。	
		0x3	CT16B0_MAT2。	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效（未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向（引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向（引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.12 引脚 TDI/PIO0_11 的 I/O 配置

表 67. 引脚 TDI/PIO0_11/AD0/CT32B0_MAT3 的 I/O 配置 （TDI_PIO0_11，地址 0x4004 402C）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	TDI。	
		0x1	PIO0_11。	
		0x2	AD0。	
		0x3	CT32B0_MAT3。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
7	ADMODE		选择模拟 / 数字模式。	1
		0	模拟输入模式。	
		1	数字功能模式。	
8	FILTR		选择 10 ns 输入干扰滤波器。	0
		0	禁用滤波器。	
		1	使能滤波器。	
9	-		保留。	0
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.13 引脚 TMS/PIO0_12 的 I/O 配置

表 68. 引脚 TMS/PIO0_12/AD1/CT32B1_CAP0 的 I/O 配置（TMS_PIO0_12，地址 0x4004 4030）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	TMS。	
		0x1	PIO0_12。	
		0x2	AD1。	
		0x3	CT32B1_CAP0。	

表 68. 引脚 TMS/PIO0_12/AD1/CT32B1_CAP0 的 I/O 配置（TMS_PIO0_12，地址 0x4004 4030）位描述 *（续）*

位	符号	值	描述	复位值
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
7	ADMODE		选择模拟 / 数字模式。	1
		0	模拟输入模式。	
		1	数字功能模式。	
8	FILTR		选择 10 ns 输入干扰滤波器。	0
		0	禁用滤波器。	
		1	使能滤波器。	
9	-		保留。	0
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.14 引脚 TDO/PIO0_13 的 I/O 配置

表 69. 引脚 TDO/PIO0_13/AD2/CT32B1_MAT0 的 I/O 配置（TDO_PIO0_13，地址 0x4004 4034）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	TDO。	
		0x1	PIO0_13。	
		0x2	AD2。	
		0x3	CT32B1_MAT0。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	

表 69. 引脚 TDO/PIO0_13/AD2/CT32B1_MAT0 的 I/O 配置（TDO_PIO0_13，地址 0x4004 4034）位描述 *（续）*

位	符号	值	描述	复位值
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
7	ADMODE		选择模拟 / 数字模式。	1
		0	模拟输入模式。	
		1	数字功能模式。	
8	FILTR		选择 10 ns 输入干扰滤波器。	0
		0	禁用滤波器。	
		1	使能滤波器。	
9	-		保留。	0
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.15 引脚 TRST/PIO0_14 的 I/O 配置

表 70. 引脚 TRST/PIO0_14/AD3/CT32B1_MAT1 的 I/O 配置 （TRST_PIO0_14，地址 0x4004 4038）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	TRST。	
		0x1	PIO0_14。	
		0x2	AD3。	
		0x3	CT32B1_MAT1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	

表 70. 引脚 TRST/PIO0_14/AD3/CT32B1_MAT1 的 I/O 配置 （TRST_PIO0_14，地址 0x4004 4038）位描述 *（续）*

位	符号	值	描述	复位值
7	ADMODE		选择模拟 / 数字模式。	1
		0	模拟输入模式。	
		1	数字功能模式。	
8	FILTR		选择 10 ns 输入干扰滤波器。	0
		0	禁用滤波器。	
		1	使能滤波器。	
9	-		保留。	0
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.16 引脚 SWDIO/PIO0_15 的 I/O 配置

表 71. 引脚 SWDIO/PIO0_15/AD4/CT32B1_MAT2 的 I/O 配置（SWDIO_PIO0_15，地址 0x4004 403C）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	SWDIO。	
		0x1	PIO0_15。	
		0x2	AD4。	
		0x3	CT32B1_MAT2。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
7	ADMODE		选择模拟 / 数字模式。	1
		0	模拟输入模式。	
		1	数字功能模式。	
8	FILTR		选择 10 ns 输入干扰滤波器。	0
		0	禁用滤波器。	
		1	使能滤波器。	
9	-		保留。	0

表 71. 引脚 SWDIO/PIO0_15/AD4/CT32B1_MAT2 的 I/O 配置（SWDIO_PIO0_15，地址 0x4004 403C）位描述 *（续）*

位	符号	值	描述	复位值
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.17 引脚 PIO0_16 的 I/O 配置

表 72. 引脚 PIO0_16/AD5/CT32B1_MAT3/ WAKEUP 的 I/O 配置（PIO0_16，地址 0x4004 4040）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。如果 LPC1315/16/17/45/46/47 处于深度掉电模式，该引脚用作 WAKEUP 引脚，而不管 FUNC 值如何。值 0x3 至 0x7 保留。	0
		0x0	PIO0_16。	
		0x1	AD5。	
		0x2	CT32B1_MAT3。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
7	ADMODE		选择模拟 / 数字模式。	1
		0	模拟输入模式。	
		1	数字功能模式。	
8	FILTR		选择 10 ns 输入干扰滤波器。	0
		0	禁用滤波器。	
		1	使能滤波器。	
9	-		保留。	0
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.18 引脚 PIO0_17 的 I/O 配置

表 73. 引脚PIO0_17/RTS/CT32B0_CAP0/SCLK的I/O配置（PIO0_17，地址0x4004 4044）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	PIO0_17。	
		0x1	RTS。	
		0x2	CT32B0_CAP0。	
		0x3	SCLK （UART 同步时钟）。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.19 引脚 PIO0_18 的 I/O 配置

表 74. 引脚 PIO0_18/RXD/CT32B0_MAT0 的 I/O 配置 （PIO0_18，地址 0x4004 4048）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO0_18。	
		0x1	RXD。	
		0x2	CT32B0_MAT0。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	

表 74. 引脚 PIO0_18/RXD/CT32B0_MAT0 的 I/O 配置（PIO0_18，地址 0x4004 4048）位描述 *（续）*

位	符号	值	描述	复位值
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.20 引脚 PIO0_19 的 I/O 配置

表 75. 引脚 PIO0_19/TXD/CT32B0_MAT1 的 I/O 配置 （PIO0_19，地址 0x4004 404C）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO0_19。	
		0x1	TXD。	
		0x2	CT32B0_MAT1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.21 引脚 PIO0_20 的 I/O 配置

表 76. 引脚 PIO0_20/CT16B1_CAP0 的 I/O 配置 （PIO0_20，地址 0x4004 4050）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO0_20。	
		0x1	CT16B1_CAP0。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		00	无效 （未使能下拉 / 上拉电阻）。	
		01	已使能下拉电阻。	
		10	已使能上拉电阻。	
		11	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.22 引脚 PIO0_21 的 I/O 配置

表 77. 引脚 PIO0_21/CT16B1_MAT0/MOSI1 的 I/O 配置 （PIO0_21，地址 0x4004 4054）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO0_21。	
		0x1	CT16B1_MAT0。	
		0x2	MOSI1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	

表 77. 引脚 PIO0_21/CT16B1_MAT0/MOSI1 的 I/O 配置 （PIO0_21，地址 0x4004 4054）位描述 *（续）*

位	符号	值	描述	复位值
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.23 引脚 PIO0_22 的 I/O 配置

表 78. 引脚 PIO0_22/AD6/CT16B1_MAT1/MISO1 的 I/O 配置（PIO0_22，地址 0x4004 4058）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	PIO0_22。	
		0x1	AD6。	
		0x2	CT16B1_MAT1。	
		0x3	MISO1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
7	ADMODE		选择模拟 / 数字模式。	1
		0	模拟输入模式。	
		1	数字功能模式。	
8	FILTR		选择 10 ns 输入干扰滤波器。	0
		0	禁用滤波器。	
		1	使能滤波器。	
9	-		保留。	0

表 78. 引脚 PIO0_22/AD6/CT16B1_MAT1/MISO1 的 I/O 配置（PIO0_22，地址 0x4004 4058）位描述 *（续）*

位	符号	值	描述	复位值
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.24 引脚 PIO0_23 的 I/O 配置

表 79. 引脚 PIO0_23/AD7 的 I/O 配置 （PIO0_23，地址 0x4004 405C）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO0_23。	
		0x1	AD7。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
7	ADMODE		选择模拟 / 数字模式。	1
		0	模拟输入模式。	
		1	数字功能模式。	
8	FILTR		选择 10 ns 输入干扰滤波器。	0
		0	禁用滤波器。	
		1	使能滤波器。	
9	-		保留。	0
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.25 引脚 PIO1_0 的 I/O 配置

表 80. 引脚 PIO1_0/CT32B1_MAT0 的 I/O 配置 （PIO1_0，地址 0x4004 4060）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO1_0。	
		0x1	CT32B1_MAT1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.26 引脚 PIO1_1 的 I/O 配置

表 81. 引脚 PIO1_1/CT32B1_MAT1 的 I/O 配置 （PIO1_1，地址 0x4004 4064）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO1_1。	
		0x1	CT32B1_MAT1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1

表 81. 引脚 PIO1_1/CT32B1_MAT1 的 I/O 配置 （PIO1_1，地址 0x4004 4064）位描述 *（续）*

位	符号	值	描述	复位值
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.27 引脚 PIO1_2 的 I/O 配置

表 82. 引脚 PIO1_2/CT32B1_MAT2 的 I/O 配置 （PIO1_2，地址 0x4004 4068）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO1_2。	
		0x1	CT32B1_MAT2。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.28 引脚 PIO1_3 的 I/O 配置

表 83. 引脚 PIO1_3/CT32B1_MAT3 的 I/O 配置 （PIO1_3，地址 0x4004 406C）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO1_3。	
		0x1	CT32B1_MAT3。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	

表 83. 引脚 PIO1_3/CT32B1_MAT3 的 I/O 配置 （PIO1_3，地址 0x4004 406C）位描述 *（续）*

位	符号	值	描述	复位值
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.29 引脚 PIO1_4 的 I/O 配置

表 84. 引脚 PIO1_4/CT32B1_CAP0 的 I/O 配置 （PIO1_4，地址 0x4004 4070）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO1_4。	
		0x1	CT32B1_CAP0。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.30 引脚 PIO1_5 的 I/O 配置

表 85. 引脚 PIO1_5/CT32B1_CAP1 的 I/O 配置 （PIO1_5，地址 0x4004 4074）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO1_5。	
		0x1	CT32B1_CAP1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.31 引脚 PIO1_7 的 I/O 配置

表 86. 引脚 PIO1_7 的 I/O 配置 （PIO1_7，地址 0x4004 407C）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO1_7。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	001

表 86. 引脚 PIO1_7 的 I/O 配置 （PIO1_7，地址 0x4004 407C）位描述 *（续）*

位	符号	值	描述	复位值
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.32 引脚 PIO1_8 的 I/O 配置

表 87. 引脚 PIO1_8 的 I/O 配置 （PIO1_8，地址 0x4004 4080）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x1 至 0x7 保留。	0
		0x0	PIO1_8。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	001
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.33 引脚 PIO1_10 的 I/O 配置

表 88. 引脚 PIO1_10 的 I/O 配置 （PIO1_10，地址 0x4004 4088）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x1 至 0x7 保留。	0
		0x0	PIO1_10。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	

表 88. 引脚 PIO1_10 的 I/O 配置 （PIO1_10，地址 0x4004 4088）位描述 *（续）*

位	符号	值	描述	复位值
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.34 引脚 PIO1_11 的 I/O 配置

表 89. 引脚 PIO1_11 的 I/O 配置 （PIO1_11，地址 0x4004 408C）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x1 至 0x7 保留。	000
		0x0	PIO1_11。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.35 PIO1_13 的 I/O 配置

表 90. PIO1_13/DTR/CT16B0_MAT0/TXD 的 I/O 配置 （PIO1_13，地址 0x4004 4094）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	PIO1_13	
		0x1	DTR	
		0x2	CT16B0_MAT0	
		0x3	TXD	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	001
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.36 PIO1_14 的 I/O 配置

表 91. PIO1_14/DSR/CT16B0_MAT1/RXD 的 I/O 配置（PIO1_14，地址 0x4004 4098）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	PIO1_14。	
		0x1	DSR	
		0x2	CT16B0_MAT1	
		0x3	RXD	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	

表 91. PIO1_14/DSR/CT16B0_MAT1/RXD 的 I/O 配置（PIO1_14，地址 0x4004 4098）位描述 *（续）*

位	符号	值	描述	复位值
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.37 引脚 PIO1_15 的 I/O 配置

表 92. 引脚 PIO1_15/DCD/ CT16B0_MAT2/SCK1 的 I/O 配置 （PIO1_15，地址 0x4004 409C）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x4 至 0x7 保留。	0
		0x0	PIO1_15。	
		0x1	DCD。	
		0x2	CT16B0_MAT2。	
		0x3	SCK1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	001
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.38 引脚 PIO1_16 的 I/O 配置

表 93. 引脚 PIO1_16/RI/CT16B0_CAP0 的 I/O 配置 （PIO1_16，地址 0x4004 40A0）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO1_16。	
		0x1	RI。	
		0x2	CT16B0_CAP0。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	001
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.39 PIO1_17 的 I/O 配置

表 94. PIO1_17/CT16B0_CAP1/RXD 的 I/O 配置 （PIO1_17，地址 0x4004 40A4）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO1_17。	
		0x1	CT16B0_CAP1	
		0x2	RXD	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	

表 94. PIO1_17/CT16B0_CAP1/RXD 的 I/O 配置 （PIO1_17，地址 0x4004 40A4）位描述 *（续）*

位	符号	值	描述	复位值
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	001
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.40 PIO1_18 的 I/O 配置

表 95. PIO1_18/CT16B1_CAP1/TXD 的 I/O 配置 （PIO1_18，地址 0x4004 40A8）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO1_18	
		0x1	CT16B1_CAP1	
		0x2	TXD	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.41 引脚 PIO1_19 的 I/O 配置

表 96. 引脚 PIO1_19/DTR/SSEL1 的 I/O 配置 （PIO1_19，地址 0x4004 40AC）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO1_19。	
		0x1	DTR。	
		0x2	SSEL1。	
4:3	MODE		模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	001
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.42 引脚 PIO1_20 的 I/O 配置

表 97. 引脚 PIO1_20/DSR/SCK1 的 I/O 配置 （PIO1_20，地址 0x4004 40B0）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO1_20。	
		0x1	DSR。	
		0x2	SCK1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	

表 97. 引脚 PIO1_20/DSR/SCK1 的 I/O 配置 （PIO1_20，地址 0x4004 40B0）位描述 *（续）*

位	符号	值	描述	复位值
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	001
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.43 引脚 PIO1_21 的 I/O 配置

表 98. 引脚 PIO1_21/DCD/MISO1 的 I/O 配置 （PIO1_21，地址 0x4004 40B4）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO1_21。	
		0x1	DCD。	
		0x2	MISO1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.44 引脚 PIO1_22 的 I/O 配置

表 99. 引脚 PIO1_22/RI/MOSI1 的 I/O 配置 （PIO1_22，地址 0x4004 40B8）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO1_22。	
		0x1	RI。	
		0x2	MOSI1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	001
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.45 引脚 PIO1_23 的 I/O 配置

表 100. 引脚 PIO1_23/CT16B1_MAT1/SSEL1 的 I/O 配置（PIO1_23，地址 0x4004 40BC）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO1_23。	
		0x1	CT16B1_MAT1。	
		0x2	SSEL1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	

表 100. 引脚 PIO1_23/CT16B1_MAT1/SSEL1 的 I/O 配置（PIO1_23，地址 0x4004 40BC）位描述 *（续）*

位	符号	值	描述	复位值
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.46 引脚 PIO1_24 的 I/O 配置

表 101. 引脚 PIO1_24/ CT32B0_MAT0 的 I/O 配置 （PIO1_24，地址 0x4004 40C0）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO1_24。	
		0x1	CT32B0_MAT0。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.47 引脚 PIO1_25 的 I/O 配置

表 102. 引脚 PIO1_25/CT32B0_MAT1 的 I/O 配置 （PIO1_25，地址 0x4004 40C4）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x2 至 0x7 保留。	0
		0x0	PIO1_25。	
		0x1	CT32B0_MAT1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.48 引脚 PIO1_26 的 I/O 配置

表 103. 引脚 PIO1_26/CT32B0_MAT2/ RXD 的 I/O 配置（PIO1_26，地址 0x4004 40C8）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO1_26。	
		0x1	CT32B0_MAT2	
		0x2	RXD。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	

表 103. 引脚PIO1_26/CT32B0_MAT2/ RXD的I/O配置（PIO1_26，地址0x4004 40C8）位描述（续）

位	符号	值	描述	复位值
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.49 引脚 PIO1_27 的 I/O 配置

表 104. 引脚 PIO1_27/CT32B0_MAT3/ TXD 的 I/O 配置 （PIO1_27，地址 0x4004 40CC）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO1_27。	
		0x1	CT32B0_MAT3。	
		0x2	TXD。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.50 引脚 PIO1_28 的 I/O 配置

表 105. 引脚 PIO1_28/CT32B0_CAP0/ SCLK 的 I/O 配置（PIO1_28，地址 0x4004 40D0）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO1_28。	
		0x1	CT32B0_CAP0。	
		0x2	SCLK。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.51 引脚 PIO1_29 的 I/O 配置

表 106. 引脚 PIO1_29/SCK0/ CT32B0_CAP1 的 I/O 配置 （PIO1_29，地址 0x4004 40D4）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x3 至 0x7 保留。	0
		0x0	PIO1_29。	
		0x1	SCK0。	
		0x2	CT32B0_CAP1。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	

表 106. 引脚 PIO1_29/SCK0/ CT32B0_CAP1 的 I/O 配置 （PIO1_29，地址 0x4004 40D4）位描述 *（续）*

位	符号	值	描述	复位值
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

7.4.52 引脚 PIO1_31 的 I/O 配置

表 107. 引脚 PIO1_31 的 I/O 配置 （PIO1_31，地址 0x4004 40DC）位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。值 0x1 至 0x7 保留。	0
		0x0	PIO1_31。	
4:3	MODE		选择功能模式 （片内上拉 / 下拉电阻控制）。	0x2
		0x0	无效 （未使能下拉 / 上拉电阻）。	
		0x1	已使能下拉电阻。	
		0x2	已使能上拉电阻。	
		0x3	中继模式。	
5	HYS		滞回。	0
		0	禁用。	
		1	使能。	
6	INV		反转输入	0
		0	输入未反向 （引脚高电平读作 1，引脚低电平读作 0）。	
		1	输入反向 （引脚高电平读作 0，引脚低电平读作 1）。	
9:7	-		保留。	0x1
10	OD		开漏模式。	0
		0	禁用。	
		1	已使能开漏模式。这不是真开漏模式。输入不能上拉至超过 VDD。	
31:11	-		保留。	0

8.1 引脚配置

8.1.1 引脚说明

表 108 和表 109 显示了全部引脚及其所分配到的数字或模拟功能（按 GPIO 端口号排序）。首先列出的是复位后的默认功能。所有端口引脚会在复位后使能内部上拉电阻，但真正的开漏引脚 PIO0_4 和 PIO0_5 除外。

每个端口引脚都有一个相应的 IOCON 寄存器，用于编程数字或模拟功能、上拉 / 下拉配置、中继器和开漏模式。

要为某个外设功能选择端口引脚，可使用此功能在端口引脚的 IOCON 寄存器中编程 FUNC 位。对于多路复用到多个端口引脚的功能，用户必须确保将功能明确分配给端口引脚。

默认情况下，在相应的 IOCON 寄存器中选择 JTAG 和 SWD 的调试功能。所有其他功能必须在 IOCON 模块中编程后才能使用。

表 108. 引脚描述 （LPC1315/16/17 - 无 USB）

符号	LQFP64	LQFP48	HVQFN33		复位状态 [1]	类型	描述
RESET/PIO0_0	4	3	2	[2]	I；PU	I	**RESET** — 外部复位输入，具有 20 ns 干扰滤波器。此引脚上短至 50 ns 的下降脉冲唤将复位器件，导致 I/O 端口和外设呈现默认状态，并且处理器从地址 0 开始执行。此引脚也用作调试选择输入。低电平选择 JTAG 边界扫描。高电平选择 ARM SWD 调试模式。
					-	I/O	**PIO0_0** — 通用数字输入 / 输出引脚。
PIO0_1/CLKOUT/ CT32B0_MAT2	5	4	3	[3]	I；PU	I/O	**PIO0_1** — 通用数字输入 / 输出引脚。复位期间，当此引脚为低电平时，启动 ISP 命令处理程序。
					-	O	**CLKOUT** — Clockout 引脚。
					-	O	**CT32B0_MAT2** — 32 位定时器 0 的匹配输出 2。
PIO0_2/SSEL0/ CT16B0_CAP0	13	10	8	[3]	I；PU	I/O	**PIO0_2** — 通用数字输入 / 输出引脚。
						I/O	**SSEL0** — SSP0 的从机选择。
						I	**CT16B0_CAP0** — 16 位定时器 0 的捕获输入 0。
PIO0_3	19	14	9	[3]	I；PU	I/O	**PIO0_3** — 通用数字输入 / 输出引脚。
PIO0_4/SCL	20	15	10	[4]	IA	I/O	**PIO0_4** — 通用数字输入 / 输出引脚 （开漏）。
					-	I/O	**SCL** — I^2C 总线时钟输入 / 输出（开漏）。仅当在 I/O 配置寄存器中选择 I^2C 超快速模式时，用作高电流接收器。
PIO0_5/SDA	21	16	11	[4]	IA	I/O	**PIO0_5** — 通用数字输入 / 输出引脚 （开漏）。
					-	I/O	**SDA** — I^2C 总线数据输入 / 输出（开漏）。仅当在 I/O 配置寄存器中选择 I^2C 超快速模式时，用作高电流接收器。

表 108. 引脚描述 （LPC1315/16/17 - 无 USB）*（续）*

符号	LQFP64	LQFP48	HVQFN33		复位状态[2]	类型	描述
PIO0_6/R/ SCK0	29	22	15	[3]	I； PU	I/O	**PIO0_6** — 通用数字输入 / 输出引脚。
					-	-	**R** — 保留。
					-	I/O	**SCK0** — SSP0 的串行时钟。
PIO0_7/$\overline{\text{CTS}}$	30	23	16	[5]	I； PU	I/O	**PIO0_7** — 通用数字输入 / 输出引脚 （大电流输出驱动器）。
					-	I	**$\overline{\text{CTS}}$** — USART 的 “准许发送”输入。
PIO0_8/MISO0/ CT16B0_MAT0	36	27	17	[3]	I； PU	I/O	**PIO0_8** — 通用数字输入 / 输出引脚。
					-	I/O	**MISO0** — SSP0 的主机输入从机输出。
					-	O	**CT16B0_MAT0** — 16 位定时器 0 的匹配输出 0。
PIO0_9/MOSI0/ CT16B0_MAT1/ SWO	37	28	18	[3]	I； PU	I/O	**PIO0_9** — 通用数字输入 / 输出引脚。
					-	I/O	**MOSI0** — SSP0 的主机输出从机输入。
					-	O	**CT16B0_MAT1** — 16 位定时器 0 的匹配输出 1。
					-	O	**SWO** — 串行线跟踪输出。
SWCLK/PIO0_10/SCK0/ CT16B0_MAT2	38	29	19	[3]	I； PU	I	**SWCLK** — JTAG 接口的串行线时钟和测试时钟 TCK。
					-	I/O	**PIO0_10** — 通用数字输入 / 输出引脚。
					-	O	**SCK0** — SSP0 的串行时钟。
					-	O	**CT16B0_MAT2** — 16 位定时器 0 的匹配输出 2。
TDI/PIO0_11/AD0/ CT32B0_MAT3	42	32	21	[6]	I； PU	I	**TDI** — JTAG 接口的测试数据输入。
					-	I/O	**PIO0_11** — 通用数字输入 / 输出引脚。
					-	I	**AD0** — A/D 转换器，输入 0。
					-	O	**CT32B0_MAT3** — 32 位定时器 0 的匹配输出 3。
TMS/PIO0_12/AD1/ CT32B1_CAP0	44	33	22	[6]	I； PU	I	**TMS** — JTAG 接口的测试模式选择。
					-	I/O	**PIO_12** — 通用数字输入 / 输出引脚。
					-	I	**AD1** — A/D 转换器，输入 1。
					-	I	**CT32B1_CAP0** — 32 位定时器 1 的捕获输入 0。
TDO/PIO0_13/AD2/ CT32B1_MAT0	45	34	23	[6]	I； PU	O	**TDO** — JTAG 接口的测试数据输出。
					-	I/O	**PIO0_13** — 通用数字输入 / 输出引脚。
					-	I	**AD2** — A/D 转换器，输入 2。
					-	O	**CT32B1_MAT0** — 32 位定时器 1 的匹配输出 0。
$\overline{\text{TRST}}$/PIO0_14/AD3/ CT32B1_MAT1	46	35	24	[6]	I； PU	I	**$\overline{\text{TRST}}$** — JTAG 接口的测试重置。
					-	I/O	**PIO0_14** — 通用数字输入 / 输出引脚。
					-	I	**AD3** — A/D 转换器，输入 3。
					-	O	**CT32B1_MAT1** — 32 位定时器 1 的匹配输出 1。

表 108. 引脚描述 （LPC1315/16/17 - 无 USB）*（续）*

符号	LQFP64	LQFP48	HVQFN33		复位状态[1]	类型	描述
SWDIO/PIO0_15/AD4/ CT32B1_MAT2	52	39	25	[6]	I；PU	I/O	**SWDIO** — 串行调试接口输入 / 输出。
					-	I/O	**PIO0_15** — 通用数字输入 / 输出引脚。
					-	I	**AD4** — A/D 转换器，输入 4。
					-	O	**CT32B1_MAT2** — 32 位定时器 1 的匹配输出 2。
PIO0_16/AD5/ CT32B1_MAT3/WAKEUP	53	40	26	[7]	I；PU	I/O	**PIO0_16** — 通用数字输入 / 输出引脚。
					-	I	**AD5** — A/D 转换器，输入 5。
					-	O	**CT32B1_MAT3** — 32 位定时器 1 的匹配输出 3。
					-	I	**WAKEUP** — 带 20 ns 干扰滤波器的深度掉电模式唤醒引脚。要进入深度掉电模式，必须从外部将此引脚上拉到高电平；要退出深度掉电模式，必须将其下拉到低电平。以短至 50 ns 的下降脉冲唤醒器件。
PIO0_17/$\overline{\text{RTS}}$/ CT32B0_CAP0/SCLK	60	45	30	[3]	I；PU	I/O	**PIO0_17** — 通用数字输入 / 输出引脚。
					-	O	**$\overline{\text{RTS}}$** — USART 的 “请求发送” 输出。
					-	I	**CT32B0_CAP0** — 32 位定时器 0 的捕获输入 0。
					-	I/O	**SCLK** — 同步模式下USART的串行时钟输入/输出。
PIO0_18/RXD/ CT32B0_MAT0	61	46	31	[3]	I；PU	I/O	**PIO0_18** — 通用数字输入 / 输出引脚。
					-	I	**RXD** — USART的接收器输入。用于UART ISP模式。
					-	O	**CT32B0_MAT0** — 32 位定时器 0 的匹配输出 0。
PIO0_19/TXD/ CT32B0_MAT1	62	47	32	[3]	I；PU	I/O	**PIO0_19** — 通用数字输入 / 输出引脚。
					-	O	**TXD** — USART的发送器输出。用于UART ISP模式。
					-	O	**CT32B0_MAT1** — 32 位定时器 0 的匹配输出 1。
PIO0_20/CT16B1_CAP0	11	9	7	[3]	I；PU	I/O	**PIO0_20** — 通用数字输入 / 输出引脚。
					-	I	**CT16B1_CAP0** — 16 位定时器 1 的捕获输入 0。
PIO0_21/CT16B1_MAT0/ MOSI1	22	17	12	[3]	I；PU	I/O	**PIO0_21** — 通用数字输入 / 输出引脚。
					-	O	**CT16B1_MAT0** — 16 位定时器 1 的匹配输出 0。
					-	I/O	**MOSI1** — SSP1 的主机输出从机输入。
PIO0_22/AD6/ CT16B1_MAT1/MISO1	40	30	20	[6]	I；PU	I/O	**PIO0_22** — 通用数字输入 / 输出引脚。
					-	I	**AD6** — A/D 转换器，输入 6。
					-	O	**CT16B1_MAT1** — 16 位定时器 1 的匹配输出 1。
					-	I/O	**MISO1** — SSP1 的主机输入从机输出。
PIO0_23/AD7	56	42	27	[6]	I；PU	I/O	**PIO0_23** — 通用数字输入 / 输出引脚。
					-	I	**AD7** — A/D 转换器，输入 7。
PIO1_0/CT32B1_MAT0	1	-	-	[3]	I；PU	I/O	**PIO1_0** — 通用数字输入 / 输出引脚。
					-	O	**CT32B1_MAT0** — 32 位定时器 1 的匹配输出 0。
PIO1_1/CT32B1_MAT1	17	-	-	[3]	I；PU	I/O	**PIO1_1** — 通用数字输入 / 输出引脚。
					-	O	**CT32B1_MAT1** — 32 位定时器 1 的匹配输出 1。
PIO1_2/CT32B1_MAT2	34	-	-	[3]	I；PU	I/O	**PIO1_2** — 通用数字输入 / 输出引脚。
					-	O	**CT32B1_MAT2** — 32 位定时器 1 的匹配输出 2。

表 108. 引脚描述 （LPC1315/16/17 - 无 USB）*（续）*

符号	LQFP64	LQFP48	HVQFN33		复位状态[2]	类型	描述
PIO1_3/CT32B1_MAT3	50	-	-	[3]	I；PU	I/O	**PIO1_3** — 通用数字输入 / 输出引脚。
					-	O	**CT32B1_MAT3** — 32 位定时器 1 的匹配输出 3。
PIO1_4/CT32B1_CAP0	16	-	-	[3]	I；PU	I/O	**PIO1_4** — 通用数字输入 / 输出引脚。
					-	I	**CT32B1_CAP0** — 32 位定时器 1 的捕获输入 0。
PIO1_5/CT32B1_CAP1	32	-	-	[3]	I；PU	I/O	**PIO1_5** — 通用数字输入 / 输出引脚。
					-	I	**CT32B1_CAP1** — 32 位定时器 1 的捕获输入 1。
PIO1_7	6	-	-	[3]	I；PU	I/O	**PIO1_7** — 通用数字输入 / 输出引脚。
PIO1_8	39	-	-	[3]	I；PU	I/O	**PIO1_8** — 通用数字输入 / 输出引脚。
PIO1_10	12	-	-	[3]	I；PU	I/O	**PIO1_10** — 通用数字输入 / 输出引脚。
PIO1_11	43	-	-	[3]	I；PU	I/O	**PIO1_11** — 通用数字输入 / 输出引脚。
PIO1_13/$\overline{\text{DTR}}$/CT16B0_MAT0/TXD	47	36	-	[3]	I；PU	I/O	**PIO1_13** — 通用数字输入 / 输出引脚。
					-	O	**$\overline{\text{DTR}}$** — USART 的 “数据终端就绪” 输出。
					-	O	**CT16B0_MAT0** — 16 位定时器 0 的匹配输出 0。
					-	O	**TXD** — USART 的发送器输出。
PIO1_14/$\overline{\text{DSR}}$/CT16B0_MAT1/RXD	49	37	-	[3]	I；PU	I/O	**PIO1_14** — 通用数字输入 / 输出引脚。
					-	I	**$\overline{\text{DSR}}$** — USART 的 “数据设置就绪” 输入。
					-	O	**CT16B0_MAT1** — 16 位定时器 0 的匹配输出 1。
					-	I	**RXD** — USART 的接收器输入。
PIO1_15/$\overline{\text{DCD}}$/CT16B0_MAT2/SCK1	57	43	28	[3]	I；PU	I/O	**PIO1_15** — 通用数字输入 / 输出引脚。
					-	I	**$\overline{\text{DCD}}$** — USART 的 “数据载波检测” 输入。
					-	O	**CT16B0_MAT2** — 16 位定时器 0 的匹配输出 2。
					-	I/O	**SCK1** — SSP1 的串行时钟。
PIO1_16/$\overline{\text{RI}}$/CT16B0_CAP0	63	48	-	[3]	I；PU	I/O	**PIO1_16** — 通用数字输入 / 输出引脚。
					-	I	**$\overline{\text{RI}}$** — USART 的振铃指示器输入。
					-	I	**CT16B0_CAP0** — 16 位定时器 0 的捕获输入 0。
PIO1_17/CT16B0_CAP1/RXD	23	-	-	[3]	I；PU	I/O	**PIO1_17** — 通用数字输入 / 输出引脚。
					-	I	**CT16B0_CAP1** — 16 位定时器 0 的捕获输入 1。
					-	I	**RXD** — USART 的接收器输入。
PIO1_18/CT16B1_CAP1/TXD	28	-	-	[3]	I；PU	I/O	**PIO1_18** — 通用数字输入 / 输出引脚。
					-	I	**CT16B1_CAP1** — 16 位定时器 1 的捕获输入 1。
					-	O	**TXD** — USART 的发送器输出。
PIO1_19/$\overline{\text{DTR}}$/SSEL1	3	2	1	[3]	I；PU	I/O	**PIO1_19** — 通用数字输入 / 输出引脚。
					-	O	**$\overline{\text{DTR}}$** — USART 的 “数据终端就绪” 输出。
					-	I/O	**SSEL1** — SSP1 的从机选择。
PIO1_20/$\overline{\text{DSR}}$/SCK1	18	13	-	[3]	I；PU	I/O	**PIO1_20** — 通用数字输入 / 输出引脚。
					-	I	**$\overline{\text{DSR}}$** — USART 的 “数据设置就绪” 输入。
					-	I/O	**SCK1** — SSP1 的串行时钟。

表 108. 引脚描述 （LPC1315/16/17 - 无 USB）*（续）*

符号	LQFP64	LQFP48	HVQFN33		复位状态[2]	类型	描述
PIO1_21/DCD/MISO1	35	26	-	[3]	I；PU	I/O	**PIO1_21** — 通用数字输入 / 输出引脚。
					-	I	**DCD** — USART 的 “数据载波检测” 输入。
					-	I/O	**MISO1** — SSP1 的主机输入从机输出。
PIO1_22/RI/MOSI1	51	38	-	[3]	I；PU	I/O	**PIO1_22** — 通用数字输入 / 输出引脚。
					-	I	**RI** — USART 的振铃指示器输入。
					-	I/O	**MOSI1** — SSP1 的主机输出从机输入。
PIO1_23/CT16B1_MAT1/ SSEL1	24	18	13	[3]	I；PU	I/O	**PIO1_23** — 通用数字输入 / 输出引脚。
					-	O	**CT16B1_MAT1** — 16 位定时器 1 的匹配输出 1。
					-	I/O	**SSEL1** — SSP1 的从机选择。
PIO1_24/CT32B0_MAT0	27	21	14	[3]	I；PU	I/O	**PIO1_24** — 通用数字输入 / 输出引脚。
					-	O	**CT32B0_MAT0** — 32 位定时器 0 的匹配输出 0。
PIO1_25/CT32B0_MAT1	2	1	-	[3]	I；PU	I/O	**PIO1_25** — 通用数字输入 / 输出引脚。
					-	O	**CT32B0_MAT1** — 32 位定时器 0 的匹配输出 1。
PIO1_26/CT32B0_MAT2/ RXD	14	11	-	[3]	I；PU	I/O	**PIO1_26** — 通用数字输入 / 输出引脚。
					-	O	**CT32B0_MAT2** — 32 位定时器 0 的匹配输出 2。
					-	I	**RXD** — USART 的接收器输入。
PIO1_27/CT32B0_MAT3/ TXD	15	12	-	[3]	I；PU	I/O	**PIO1_27** — 通用数字输入 / 输出引脚。
					-	O	**CT32B0_MAT3** — 32 位定时器 0 的匹配输出 3。
					-	O	**TXD** — USART 的发送器输出。
PIO1_28/CT32B0_CAP0/ SCLK	31	24	-	[3]	I；PU	I/O	**PIO1_28** — 通用数字输入 / 输出引脚。
					-	I	**CT32B0_CAP0** — 32 位定时器 0 的捕获输入 0。
					-	I/O	**SCLK** — 同步模式下 USART 的串行时钟输入 / 输出。
PIO1_29/SCK0/ CT32B0_CAP1	41	31	-	[3]	I；PU	I/O	**PIO1_29** — 通用数字输入 / 输出引脚。
					-	I/O	**SCK0** — SSP0 的串行时钟。
					-	I	**CT32B0_CAP1** — 32 位定时器 0 的捕获输入 1。
PIO1_31	-	25	-	[3]	I；PU	I/O	**PIO1_31** — 通用数字输入 / 输出引脚。
n.c.	25	19	-		-	-	未连接。
n.c.	26	20	-		-	-	未连接。
XTALIN	8	6	4	[8]	-	-	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 1.8 V。
XTALOUT	9	7	5	[8]	-	-	振荡器放大器的输出。
V_{DDA}	59	-	-		-	-	模拟 3.3 V 焊盘电源电压：名义上该电压应当与 V_{DD} 相同，但应将其隔离以最大限度降低噪声和错误。该电压用来为 ADC 供电。如果未使用 ADC，则该引脚应当与 3.3V 电源连接。

表 108. 引脚描述 （LPC1315/16/17 - 无 USB） *（续）*

符号	LQFP64	LQFP48	HVQFN33	复位状态[1]	类型	描述
VREFN	48	-	-	-	-	ADC 负基准电压：名义上该电压应当与 V_{SS} 相同，但应将其隔离以最大限度降低噪声和错误。该引脚上的电平用作 ADC 的基准。
VREFP	64	-	-	-	-	ADC 正基准电压：名义上该电压应当与 V_{DDA} 相同，但应将其隔离以最大限度降低噪声和错误。该引脚上的电平用作 ADC 的基准。如果未使用 ADC，则该引脚应当与 3.3V 电源连接。
V_{SSA}	55	-	-	-	-	模拟接地：0 V 参考。名义上该电压应当与 V_{SS} 相同，但应将其隔离以最大限度降低噪声和错误。
V_{DD}	10; 33; 58	8; 44	6; 29	-	-	供给内部调压器和外部轨的电源电压。在 LQFP48 和 HVQFN33 封装上，此引脚也将连接到 3.3 V ADC 电源和基准电压。
V_{SS}	7; 54	5; 41	33	-	-	地线。

[1] 复位后默认功能的引脚状态：I= 输入； O= 输出； PU= 使能内部上拉电阻； IA= 非工作，未使能上拉电阻 / 下拉电阻；F= 浮动；未使用时，浮动引脚应接地或者连接电源以最大限度降低功耗。

[2] 关于复位焊盘配置，请参见图 9。在深度掉电模式下，不能使用 $\overline{\text{RESET}}$ 功能。使用 WAKEUP 引脚复位芯片和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。

[3] 5V 容限的焊盘，提供带可配置上拉 / 下拉电阻和可配置滞回的数字 I/O 功能 （参见图 8）。

[4] I^2C 总线引脚符合 I^2C 总线规范，用于 I^2C 标准模式、 I^2C 快速模式和 I^2C 超快速模式。

[5] 5 V 容限的焊盘，提供带可配置上拉 / 下拉电阻和可配置滞回的数字 I/O 功能 （参见图 8）； 包括大电流输出驱动器。

[6] 5 V 容限的焊盘，提供带可配置上拉 / 下拉电阻、可配置滞回和模拟输入的数字 I/O 功能。当配置为 ADC 输入时，禁用焊盘的数字部分，引脚非 5 V 容限 （参见图 8）； 包括可编程数字输入干扰滤波器。

[7] WAKEUP 引脚。 5 V 容限的焊盘，提供带可配置上拉 / 下拉电阻、可配置滞回和模拟输入的数字 I/O 功能。当配置为 ADC 输入时，禁用焊盘的数字部分，引脚非 5 V 容限 （参见图 8）； 包括数字输入干扰滤波器。

[8] 未使用系统振荡器时，按如下所示连接 XTALIN 和 XTALOUT：XTALIN 可悬空或接地（最好接地以降低噪声敏感性）。XTALOUT 应悬空。

表 109. 引脚描述 （LPC1345/46/47 - 有 USB）

符号	LQFP64	LQFP48	HVQFN33		复位状态 [1]	类型	描述
RESET/PIO0_0	4	3	2	[2]	I；PU	I	**RESET** — 外部复位输入，具有 20 ns 干扰滤波器。此引脚上短至 50 ns 的下降脉冲唤将复位器件，导致 I/O 端口和外设呈现默认状态，并且处理器从地址 0 开始执行。此引脚也用作调试选择输入。低电平选择 JTAG 边界扫描。高电平选择 ARM SWD 调试模式。
					-	I/O	**PIO0_0** — 通用数字输入 / 输出引脚。
PIO0_1/CLKOUT/ CT32B0_MAT2/ USB_FTOGGLE	5	4	3	[3]	I；PU	I/O	**PIO0_1** — 通用数字输入 / 输出引脚。复位期间，当此引脚为低电平时，启动 ISP 命令处理程序或者 USB 设备枚举。
					-	O	**CLKOUT** — Clockout 引脚。
					-	O	**CT32B0_MAT2** — 32 位定时器 0 的匹配输出 2。
					-	O	**USB_FTOGGLE** — USB 1 ms 帧起始信号。
PIO0_2/SSEL0/ CT16B0_CAP0	13	10	8	[3]	I；PU	I/O	**PIO0_2** — 通用数字输入 / 输出引脚。
						I/O	**SSEL0** — SSP0 的从机选择。
						I	**CT16B0_CAP0** — 16 位定时器 0 的捕获输入 0。
PIO0_3/USB_VBUS	19	14	9	[3]	I；PU	I/O	**PIO0_3** — 通用数字输入 / 输出引脚。复位期间，当此引脚为低电平时，启动 ISP 命令处理程序。复位期间，高电平启动 USB 设备枚举。
					-	I	**USB_VBUS** — 监控是否存在 USB 总线供电。
PIO0_4/SCL	20	15	10	[4]	IA	I/O	**PIO0_4** — 通用数字输入 / 输出引脚 （开漏）。
					-	I/O	**SCL** — I^2C 总线时钟输入 / 输出（开漏）。仅当在 I/O 配置寄存器中选择 I^2C 超快速模式时，用作高电流接收器。
PIO0_5/SDA	21	16	11	[4]	IA	I/O	**PIO0_5** — 通用数字输入 / 输出引脚 （开漏）。
					-	I/O	**SDA** — I^2C 总线数据输入 / 输出（开漏）。仅当在 I/O 配置寄存器中选择 I^2C 超快速模式时，用作高电流接收器。
PIO0_6/USB_CONNECT/ SCK0	29	22	15	[3]	I；PU	I/O	**PIO0_6** — 通用数字输入 / 输出引脚。
					-	O	**USB_CONNECT** — 用于在软件控制下开关1.5 kΩ电阻的信号。与 SoftConnect USB 功能搭配使用。
					-	I/O	**SCK0** — SSP0 的串行时钟。
PIO0_7/CTS	30	23	16	[5]	I；PU	I/O	**PIO0_7** — 通用数字输入 / 输出引脚 （大电流输出驱动器）。
					-	I	**CTS** — USART 的 “准许发送” 输入。
PIO0_8/MISO0/ CT16B0_MAT0	36	27	17	[3]	I；PU	I/O	**PIO0_8** — 通用数字输入 / 输出引脚。
					-	I/O	**MISO0** — SSP0 的主机输入从机输出。
					-	O	**CT16B0_MAT0** — 16 位定时器 0 的匹配输出 0。
PIO0_9/MOSI0/ CT16B0_MAT1/ SWO	37	28	18	[3]	I；PU	I/O	**PIO0_9** — 通用数字输入 / 输出引脚。
					-	I/O	**MOSI0** — SSP0 的主机输出从机输入。
					-	O	**CT16B0_MAT1** — 16 位定时器 0 的匹配输出 1。
					-	O	**SWO** — 串行线跟踪输出。

表 109. 引脚描述 （LPC1345/46/47 - 有 USB）（续）

符号	LQFP64	LQFP48	HVQFN33		复位状态[1]	类型	描述
SWCLK/PIO0_10/SCK0/ CT16B0_MAT2	38	29	19	[3]	I；PU	I	**SWCLK** — JTAG 接口的串行线时钟和测试时钟 TCK。
					-	I/O	**PIO0_10** — 通用数字输入 / 输出引脚。
					-	O	**SCK0** — SSP0 的串行时钟。
					-	O	**CT16B0_MAT2** — 16 位定时器 0 的匹配输出 2。
TDI/PIO0_11/AD0/ CT32B0_MAT3	42	32	21	[6]	I；PU	I	**TDI** — JTAG 接口的测试数据输入。
					-	I/O	**PIO0_11** — 通用数字输入 / 输出引脚。
					-	I	**AD0** — A/D 转换器，输入 0。
					-	O	**CT32B0_MAT3** — 32 位定时器 0 的匹配输出 3。
TMS/PIO0_12/AD1/ CT32B1_CAP0	44	33	22	[6]	I；PU	I	**TMS** — JTAG 接口的测试模式选择。
					-	I/O	**PIO_12** — 通用数字输入 / 输出引脚。
					-	I	**AD1** — A/D 转换器，输入 1。
					-	I	**CT32B1_CAP0** — 32 位定时器 1 的捕获输入 0。
TDO/PIO0_13/AD2/ CT32B1_MAT0	45	34	23	[6]	I；PU	O	**TDO** — JTAG 接口的测试数据输出。
					-	I/O	**PIO0_13** — 通用数字输入 / 输出引脚。
					-	I	**AD2** — A/D 转换器，输入 2。
					-	O	**CT32B1_MAT0** — 32 位定时器 1 的匹配输出 0。
$\overline{\text{TRST}}$/PIO0_14/AD3/ CT32B1_MAT1	46	35	24	[6]	I；PU	I	$\overline{\text{TRST}}$ — JTAG 接口的测试重置。
					-	I/O	**PIO0_14** — 通用数字输入 / 输出引脚。
					-	I	**AD3** — A/D 转换器，输入 3。
					-	O	**CT32B1_MAT1** — 32 位定时器 1 的匹配输出 1。
SWDIO/PIO0_15/AD4/ CT32B1_MAT2	52	39	25	[6]	I；PU	I/O	**SWDIO** — 串行调试接口输入 / 输出。
					-	I/O	**PIO0_15** — 通用数字输入 / 输出引脚。
					-	I	**AD4** — A/D 转换器，输入 4。
					-	O	**CT32B1_MAT2** — 32 位定时器 1 的匹配输出 2。
PIO0_16/AD5/ CT32B1_MAT3/WAKEUP	53	40	26	[7]	I；PU	I/O	**PIO0_16** — 通用数字输入 / 输出引脚。
					-	I	**AD5** — A/D 转换器，输入 5。
					-	O	**CT32B1_MAT3** — 32 位定时器 1 的匹配输出 3。
					-	I	**WAKEUP** — 带 20 ns 干扰滤波器的深度掉电模式唤醒引脚。要进入深度掉电模式，必须从外部将此引脚上拉到高电平；要退出深度掉电模式，必须将其下拉到低电平。以短至 50 ns 的下降脉冲唤醒器件。
PIO0_17/$\overline{\text{RTS}}$/ CT32B0_CAP0/SCLK	60	45	30	[3]	I；PU	I/O	**PIO0_17** — 通用数字输入 / 输出引脚。
					-	O	$\overline{\text{RTS}}$ — USART 的 “请求发送” 输出。
					-	I	**CT32B0_CAP0** — 32 位定时器 0 的捕获输入 0。
					-	I/O	**SCLK** — 同步模式下 USART 的串行时钟输入 / 输出。

表 109. 引脚描述 （LPC1345/46/47 - 有 USB）*（续）*

符号	LQFP64	LQFP48	HVQFN33		复位状态[1]	类型	描述
PIO0_18/RXD/ CT32B0_MAT0	61	46	31	[3]	I；PU	I/O	**PIO0_18** — 通用数字输入 / 输出引脚。
					-	I	**RXD** — USART的接收器输入。用于UART ISP模式。
					-	O	**CT32B0_MAT0** — 32 位定时器 0 的匹配输出 0。
PIO0_19/TXD/ CT32B0_MAT1	62	47	32	[3]	I；PU	I/O	**PIO0_19** — 通用数字输入 / 输出引脚。
					-	O	**TXD** — USART的发送器输出。用于UART ISP模式。
					-	O	**CT32B0_MAT1** — 32 位定时器 0 的匹配输出 1。
PIO0_20/CT16B1_CAP0	11	9	7	[3]	I；PU	I/O	**PIO0_20** — 通用数字输入 / 输出引脚。
					-	I	**CT16B1_CAP0** — 16 位定时器 1 的捕获输入 0。
PIO0_21/CT16B1_MAT0/ MOSI1	22	17	12	[3]	I；PU	I/O	**PIO0_21** — 通用数字输入 / 输出引脚。
					-	O	**CT16B1_MAT0** — 16 位定时器 1 的匹配输出 0。
					-	I/O	**MOSI1** — SSP1 的主机输出从机输入。
PIO0_22/AD6/ CT16B1_MAT1/MISO1	40	30	20	[6]	I；PU	I/O	**PIO0_22** — 通用数字输入 / 输出引脚。
					-	I	**AD6** — A/D 转换器，输入 6。
					-	O	**CT16B1_MAT1** — 16 位定时器 1 的匹配输出 1。
					-	I/O	**MISO1** — SSP1 的主机输入从机输出。
PIO0_23/AD7	56	42	27	[6]	I；PU	I/O	**PIO0_23** — 通用数字输入 / 输出引脚。
					-	I	**AD7** — A/D 转换器，输入 7。
PIO1_0/CT32B1_MAT0	1	-	-	[3]	I；PU	I/O	**PIO1_0** — 通用数字输入 / 输出引脚。
					-	O	**CT32B1_MAT0** — 32 位定时器 1 的匹配输出 0。
PIO1_1/CT32B1_MAT1	17	-	-	[3]	I；PU	I/O	**PIO1_1** — 通用数字输入 / 输出引脚。
					-	O	**CT32B1_MAT1** — 32 位定时器 1 的匹配输出 1。
PIO1_2/CT32B1_MAT2	34	-	-	[3]	I；PU	I/O	**PIO1_2** — 通用数字输入 / 输出引脚。
					-	O	**CT32B1_MAT2** — 32 位定时器 1 的匹配输出 2。
PIO1_3/CT32B1_MAT3	50	-	-	[3]	I；PU	I/O	**PIO1_3** — 通用数字输入 / 输出引脚。
					-	O	**CT32B1_MAT3** — 32 位定时器 1 的匹配输出 3。
PIO1_4/CT32B1_CAP0	16	-	-	[3]	I；PU	I/O	**PIO1_4** — 通用数字输入 / 输出引脚。
					-	I	**CT32B1_CAP0** — 32 位定时器 1 的捕获输入 0。
PIO1_5/CT32B1_CAP1	32	-	-	[3]	I；PU	I/O	**PIO1_5** — 通用数字输入 / 输出引脚。
					-	I	**CT32B1_CAP1** — 32 位定时器 1 的捕获输入 1。
PIO1_7	6	-	-	[3]	I；PU	I/O	**PIO1_7** — 通用数字输入 / 输出引脚。
PIO1_8	39	-	-	[3]	I；PU	I/O	**PIO1_8** — 通用数字输入 / 输出引脚。
PIO1_10	12	-	-	[3]	I；PU	I/O	**PIO1_10** — 通用数字输入 / 输出引脚。
PIO1_11	43	-	-	[3]	I；PU	I/O	**PIO1_11** — 通用数字输入 / 输出引脚。
PIO1_13/$\overline{\text{DTR}}$/ CT16B0_MAT0/TXD	47	36	-	[3]	I；PU	I/O	**PIO1_13** — 通用数字输入 / 输出引脚。
					-	O	**$\overline{\text{DTR}}$** — USART 的 “数据终端就绪” 输出。
					-	O	**CT16B0_MAT0** — 16 位定时器 0 的匹配输出 0。
					-	O	**TXD** — USART 的发送器输出。

表 109. 引脚描述 （LPC1345/46/47 - 有 USB）*（续）*

符号	LQFP64	LQFP48	HVQFN33		复位状态[1]	类型	描述
PIO1_14/DSR/ CT16B0_MAT1/RXD	49	37	-	[3]	I；PU	I/O	**PIO1_14** — 通用数字输入 / 输出引脚。
					-	I	**DSR** — USART 的 “数据设置就绪” 输入。
					-	O	**CT16B0_MAT1** — 16 位定时器 0 的匹配输出 1。
					-	I	**RXD** — USART 的接收器输入。
PIO1_15/DCD/ CT16B0_MAT2/SCK1	57	43	28	[3]	I；PU	I/O	**PIO1_15** — 通用数字输入 / 输出引脚。
					-	I	**DCD** — USART 的 “数据载波检测” 输入。
					-	O	**CT16B0_MAT2** — 16 位定时器 0 的匹配输出 2。
					-	I/O	**SCK1** — SSP1 的串行时钟。
PIO1_16/RI/CT16B0_CAP0	63	48	-	[3]	I；PU	I/O	**PIO1_16** — 通用数字输入 / 输出引脚。
					-	I	**RI** — USART 的振铃指示器输入。
					-	I	**CT16B0_CAP0** — 16 位定时器 0 的捕获输入 0。
PIO1_17/CT16B0_CAP1/ RXD	23	-	-	[3]	I；PU	I/O	**PIO1_17** — 通用数字输入 / 输出引脚。
					-	I	**CT16B0_CAP1** — 16 位定时器 0 的捕获输入 1。
					-	I	**RXD** — USART 的接收器输入。
PIO1_18/CT16B1_CAP1/ TXD	28	-	-	[3]	I；PU	I/O	**PIO1_18** — 通用数字输入 / 输出引脚。
					-	I	**CT16B1_CAP1** — 16 位定时器 1 的捕获输入 1。
					-	O	**TXD** — USART 的发送器输出。
PIO1_19/DTR/SSEL1	3	2	1	[3]	I；PU	I/O	**PIO1_19** — 通用数字输入 / 输出引脚。
					-	O	**DTR** — USART 的 “数据终端就绪” 输出。
					-	I/O	**SSEL1** — SSP1 的从机选择。
PIO1_20/DSR/SCK1	18	13	-	[3]	I；PU	I/O	**PIO1_20** — 通用数字输入 / 输出引脚。
					-	I	**DSR** — USART 的 “数据设置就绪” 输入。
					-	I/O	**SCK1** — SSP1 的串行时钟。
PIO1_21/DCD/MISO1	35	26	-	[3]	I；PU	I/O	**PIO1_21** — 通用数字输入 / 输出引脚。
					-	I	**DCD** — USART 的 “数据载波检测” 输入。
					-	I/O	**MISO1** — SSP1 的主机输入从机输出。
PIO1_22/RI/MOSI1	51	38	-	[3]	I；PU	I/O	**PIO1_22** — 通用数字输入 / 输出引脚。
					-	I	**RI** — USART 的振铃指示器输入。
					-	I/O	**MOSI1** — SSP1 的主机输出从机输入。
PIO1_23/CT16B1_MAT1/ SSEL1	24	18	-	[3]	I；PU	I/O	**PIO1_23** — 通用数字输入 / 输出引脚。
					-	O	**CT16B1_MAT1** — 16 位定时器 1 的匹配输出 1。
					-	I/O	**SSEL1** — SSP1 的从机选择。
PIO1_24/CT32B0_MAT0	27	21	-	[3]	I；PU	I/O	**PIO1_24** — 通用数字输入 / 输出引脚。
					-	O	**CT32B0_MAT0** — 32 位定时器 0 的匹配输出 0。
PIO1_25/CT32B0_MAT1	2	1	-	[3]	I；PU	I/O	**PIO1_25** — 通用数字输入 / 输出引脚。
					-	O	**CT32B0_MAT1** — 32 位定时器 0 的匹配输出 1。

表 109. 引脚描述 （LPC1345/46/47 - 有 USB）*（续）*

符号	LQFP64	LQFP48	HVQFN33		复位状态 [1]	类型	描述
PIO1_26/CT32B0_MAT2/ RXD	14	11	-	[3]	I； PU	I/O	**PIO1_26** — 通用数字输入 / 输出引脚。
					-	O	**CT32B0_MAT2** — 32 位定时器 0 的匹配输出 2。
					-	I	**RXD** — USART 的接收器输入。
PIO1_27/CT32B0_MAT3/ TXD	15	12	-	[3]	I； PU	I/O	**PIO1_27** — 通用数字输入 / 输出引脚。
					-	O	**CT32B0_MAT3** — 32 位定时器 0 的匹配输出 3。
					-	O	**TXD** — USART 的发送器输出。
PIO1_28/CT32B0_CAP0/ SCLK	31	24	-	[3]	I； PU	I/O	**PIO1_28** — 通用数字输入 / 输出引脚。
					-	I	**CT32B0_CAP0** — 32 位定时器 0 的捕获输入 0。
					-	I/O	**SCLK** — 同步模式下 USART 的串行时钟输入 / 输出。
PIO1_29/SCK0/ CT32B0_CAP1	41	31	-	[3]	I； PU	I/O	**PIO1_29** — 通用数字输入 / 输出引脚。
					-	I/O	**SCK0** — SSP0 的串行时钟。
					-	I	**CT32B0_CAP1** — 32 位定时器 0 的捕获输入 1。
PIO1_31	-	25	-	[3]	I； PU	I/O	**PIO1_31** — 通用数字输入 / 输出引脚。
USB_DM	25	19	13	[8]	F	-	**USB_DM** — USB 双向 D– 线路。（仅 LPC1345/46/46。）
USB_DP	26	20	14	[8]	F	-	**USB_DP** — USB 双向 D+ 线路。（仅 LPC1345/46/46。）
XTALIN	8	6	4	[9]	-	-	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 1.8 V。
XTALOUT	9	7	5	[9]	-	-	振荡器放大器的输出。
V_{DDA}	59	-	-		-	-	模拟 3.3 V 焊盘电源电压：名义上该电压应当与 V_{DD} 相同，但应将其隔离以最大限度降低噪声和错误。该电压用来为 ADC 供电。如果未使用 ADC，则该引脚应当与 3.3V 电源连接。
VREFN	48	-	-		-	-	ADC 负基准电压：名义上该电压应当与 V_{SS} 相同，但应将其隔离以最大限度降低噪声和错误。该引脚上的电平用作 ADC 的基准。
VREFP	64	-	-		-	-	ADC 正基准电压：名义上该电压应当与 V_{DDA} 相同，但应将其隔离以最大限度降低噪声和错误。该引脚上的电平用作 ADC 的基准。如果未使用 ADC，则该引脚应当与 3.3V 电源连接。
V_{SSA}	55	-	-		-	-	模拟接地：0 V 参考。名义上该电压应当与 V_{SS} 相同，但应将其隔离以最大限度降低噪声和错误。
V_{DD}	10; 33; 58	8; 44	6; 29		-	-	供给内部调压器和外部轨的电源电压。在 LQFP48 和 HVQFN33 封装上，此引脚也将连接到 3.3 V ADC 电源和基准电压。
V_{SS}	7; 54	5; 41	33		-	-	地线。

[1] 复位后默认功能的引脚状态：I= 输入； O= 输出； PU= 使能内部上拉电阻； IA= 非工作，未使能上拉电阻 / 下拉电阻；F= 浮动；未使用时，浮动引脚应接地或者连接电源以最大限度降低功耗。

[2] 关于复位焊盘配置，请参见图 9。在深度掉电模式下，不能使用 $\overline{\text{RESET}}$ 功能。使用 WAKEUP 引脚复位芯片和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。

[3] 5V 容限的焊盘，提供带可配置上拉 / 下拉电阻和可配置滞回的数字 I/O 功能 （参见图 8）。

[4] I^2C 总线引脚符合 I^2C 总线规范，用于 I^2C 标准模式、 I^2C 快速模式和 I^2C 超快速模式。

[5] 5 V 容限的焊盘，提供带可配置上拉 / 下拉电阻和可配置滞回的数字 I/O 功能 （参见图 8）；包括大电流输出驱动器。

[6] 5 V 容限的焊盘，提供带可配置上拉 / 下拉电阻、可配置滞回和模拟输入的数字 I/O 功能。当配置为 ADC 输入时，禁用焊盘的数字部分，引脚非 5 V 容限 （参见图 8）；包括可编程数字输入干扰滤波器。

[7] WAKEUP 引脚。 5 V 容限的焊盘，提供带可配置上拉 / 下拉电阻、可配置滞回和模拟输入的数字 I/O 功能。当配置为 ADC 输入时，禁用焊盘的数字部分，引脚非 5 V 容限 （参见图 8）；包括数字输入干扰滤波器。

[8] Pad 提供 USB 功能。它们根据 USB 规格修订版 2.0 进行的设计 （仅全速和低速模式）。此 pad 无法承受 5 V 电压。

[9] 未使用系统振荡器时，按如下所示连接XTALIN和XTALOUT：XTALIN可悬空或接地（最好接地以降低噪声敏感性）。XTALOUT应悬空。15

9.1 本章导读

所有 GPIO 寄存器涵盖每个端口的 32 个引脚。并非所有引脚均可用，这要取决于封装类型；另外， GPIO 寄存器中的对应位保留 （参见表 110）。

表 110. GPIO 引脚可用

封装	GPIO 端口 0	GPIO 端口 1
LQFP64	PIO0_0 至 PIO0_23	PIO1_0 至 PIO1_5； PIO1_7 至 PIO1_8； PIO1_10 至 PIO1_29
LQFP48	PIO0_0 至 PIO0_23	PIO1_13 至 PIO1_16； PIO1_19 至 PIO1_23 至 PIO1_29； PIO1_31
HVQFN（无USB）	PIO0_0 至 PIO0_23	PIO1_15； PIO1_19； PIO1_23 至 PIO1_24
HVQFN (USB)	PIO0_0 至 PIO0_23	PIO1_15； PIO1_19

9.2 基本配置

必须使能不同寄存器模块，才可使用 GPIO 端口和引脚中断功能：

- 对于引脚中断，从 SYSCON 模块的所有 GPIO 端口引脚中选择多达 8 个外部中断引脚（表 35），并在 SYSAHBCLKCTRL 寄存器中使能引脚中断寄存器模块的时钟（表 19，位 19）。在 STARTERP0 寄存器中使能引脚中断唤醒功能 （表 38）。
- 对于分组中断功能，在 SYSAHBCLKCTRL 寄存器中使能组 0 和组 1 寄存器接口的时钟（表 19，位 19）。在 STARTERP1 寄存器中使能分组中断唤醒功能 （表 39）。
- 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 19，位 6）。

9.3 特性

9.3.1 GPIO 引脚中断特性

- 可从所有 GPIO 引脚中选择多达 8 个引脚，作为边沿或电平敏感中断请求。每个请求在 NVIC 中创建一个单独的中断。
- 边沿敏感中断引脚可以在上升沿和 / 或下降沿发生中断。
- 电平敏感中断引脚可以是高电平或低电平有效。

9.3.2 GPIO 分组中断特性

- 可以使能任意数量的 GPIO 引脚的输入，以构成一个组合的分组中断。
- 为分组中断使能的每个输入的极性可以配置为高电平或低电平。
- 使能中断可以通过 OR 或 AND 操作进行逻辑组合。
- 支持两种分组中断，体现两种不同的中断模式。
- GPIO 分组中断可以将器件从睡眠、深度睡眠或掉电模式下唤醒。

9.3.3 GPIO 端口特性

- GPIO 引脚可以通过软件配置为输入或输出。
- 所有 GPIO 引脚都默认设置为输入，复位时禁用中断。
- 引脚寄存器允许单独感测和设置各引脚。

9.4 简介

GPIO 引脚可用于通过多种方式将引脚设置为输入或输出，并将输入用作电平和边沿敏感中断的组合。

9.4.1 GPIO 引脚中断

在所有可用的 GPIO 引脚中，可在系统控制模块中选择多达八个引脚作为外部中断引脚（参见表 35）。外部中断引脚连接至 NVIC 中 8 个独立的中断，并根据上升 / 下降沿或引脚输入电平创建。

9.4.2 GPIO 分组中断

对于连接至两个 GPIO 分组中断模块（组 0 和组 1）之一的每个端口 / 引脚，GPIO 分组中断寄存器决定使能哪些引脚用于产生中断以及每个此类引脚的活动极性。

此外，GPIO 分组中断寄存器还会选择中断输出是由电平触发还是边沿触发，以及是基于所有使能输入的 OR 还是 AND 操作。

所选的输入引脚上检测到指定模式后，GPIO 分组中断模块将产生一个中断。如果器件处于省电模式，则将先异步唤醒器件，然后才产生中断请求。向控制寄存器的中断状态位写入 1，可以清除中断请求线。

9.4.3 GPIO 端口

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；或引脚配置为输出时，设置每个引脚的状态。

9.5 寄存器描述

GPIO 由以下模块组成：

- GPIO引脚中断模块位于地址0x4004 C000。该模块的寄存器使能syscon模块PINTSEL寄存器（参见表 35）中选择的多达 8 个引脚中断，并为每个所选的引脚中断配置电平和边沿敏感性。 GPIO 中断寄存器列示于表 111 和第 9.5.1 节
- GPIO组0中断模块位于地址0x4005 C000。该模块的寄存器可以配置端口0和1上的任何引脚，以形成组合中断。 GPIO 组 0 寄存器列示于表 112 和第 9.5.2 节。
- GPIO组1中断模块位于地址0x4005 8000。该模块的寄存器可以配置端口0和1上的任何引脚，以形成组合中断。 GPIO 组 1 寄存器列示于表 113 和第 9.5.2 节。
- GPIO 端口模块位于地址 0x5000 0000。该模块的寄存器可以读写端口引脚，并将端口引脚配置为输入或输出。 GPIO 端口寄存器列示于表 114 和第 9.5.3 节。

注：在所有 GPIO 寄存器中，未显示的位为**保留位**。

表 111. 寄存器简介：GPIO 引脚中断 （基址：0x4004 C000）

名称	访问类型	地址偏移	描述	复位值	参考
ISEL	R/W	0x000	引脚中断模式寄存器	0	表 115
IENR	R/W	0x004	引脚中断电平 （上升沿）中断使能寄存器	0	表 116
SIENR	WO	0x008	引脚中断电平 （上升沿）中断设置寄存器	不适用	表 117
CIENR	WO	0x00C	引脚中断电平 （上升沿中断）清除寄存器	不适用	表 118
IENF	R/W	0x010	引脚中断有效电平（下降沿）中断使能寄存器	0	表 119
SIENF	WO	0x014	引脚中断有效电平（下降沿）中断设置寄存器	不适用	表 120
CIENF	WO	0x018	引脚中断有效电平（下降沿）中断清除寄存器	不适用	表 121
RISE	R/W	0x01C	引脚中断上升沿寄存器	0	表 122
FALL	R/W	0x020	引脚中断下降沿寄存器	0	表 123
IST	R/W	0x024	引脚中断状态寄存器	0	表 124

表 112. 寄存器简介：GPIO 组 0 中断 （基址 0x4005 C000）

名称	访问类型	地址偏移	描述	复位值	参考
CTRL	R/W	0x000	GPIO 分组中断控制寄存器	0	表 125
PORT_POL0	R/W	0x020	GPIO 分组中断端口 0 极性寄存器	0xFFFF FFFF	表 126
PORT_POL1	R/W	0x024	GPIO 分组中断端口 1 极性寄存器	0xFFFF FFFF	表 127
PORT_ENA0	R/W	0x040	GPIO 分组中断端口 0 使能寄存器	0	表 128
PORT_ENA1	R/W	0x044	GPIO 分组中断端口 1 使能寄存器	0	表 129

表 113. 寄存器简介：GPIO 组 1 中断 （基址 0x4006 0000）

名称	访问类型	地址偏移	描述	复位值	参考
CTRL	R/W	0x000	GPIO 分组中断控制寄存器	0	表 125
PORT_POL0	R/W	0x020	GPIO 分组中断端口 0 极性寄存器	0xFFFF FFFF	表 126
PORT_POL1	R/W	0x024	GPIO 分组中断端口 1 极性寄存器	0xFFFF FFFF	表 127
PORT_ENA0	R/W	0x040	GPIO 分组中断端口 0 使能寄存器	0	表 128
PORT_ENA1	R/W	0x044	GPIO 分组中断端口 1 使能寄存器	0	表 129

GPIO 端口地址可以字节、半字或字进行读写。

表 114. 寄存器简介：GPIO 端口 （基址 0x5000 0000）

名称	访问类型	地址偏移	描述	复位值	宽度	参考
B0 至 B31	R/W	0x0000 至 0x001F	字节引脚寄存器端口 0；引脚 PIO0_0 至 PIO0_31	ext[1]	字节 （8 位）	表 130
B32 至 B63	R/W	0x0020 至 0x002F	字节引脚寄存器端口 1	ext[1]	字节 （8 位）	表 131
W0 至 W31	R/W	0x1000 至 0x107C	字引脚寄存器端口 0	ext[1]	字 （32 位）	表 132
W32 至 W63	R/W	0x1080 至 0x10FC	字引脚寄存器端口 1	ext[1]	字 （32 位）	表 133
DIR0	R/W	0x2000	方向寄存器端口 0	0	字 （32 位）	表 134
DIR1	R/W	0x2004	方向寄存器端口 1	0	字 （32 位）	表 135
MASK0	R/W	0x2080	屏蔽寄存器端口 0	0	字 （32 位）	表 136
MASK1	R/W	0x2084	屏蔽寄存器端口 1	0	字 （32 位）	表 137
PIN0	R/W	0x2100	端口引脚寄存器端口 0	ext[1]	字 （32 位）	表 138
PIN1	R/W	0x2104	端口引脚寄存器端口 1	ext[1]	字 （32 位）	表 139
MPIN0	R/W	0x2180	屏蔽端口寄存器端口 0	ext[1]	字 （32 位）	表 140
MPIN1	R/W	0x2184	屏蔽端口寄存器端口 1	ext[1]	字 （32 位）	表 141
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 142
SET1	R/W	0x2204	写入：端口 1 的设置寄存器 读：端口 1 的输出位	0	字 （32 位）	表 143
CLR0	WO	0x2280	清除端口 0	不适用	字 （32 位）	表 144
CLR1	WO	0x2284	清除端口 1	不适用	字 （32 位）	表 145
NOT0	WO	0x2300	切换端口 0	不适用	字 （32 位）	表 146
NOT1	WO	0x2304	切换端口 1	不适用	字 （32 位）	表 147

[1] ext 是指复位后的数据读取取决于引脚的状态，这进一步又可能取决于外部源。

9.5.1 GPIO 引脚中断寄存器描述

9.5.1.1 引脚中断模式寄存器

对于 PINTSELn 寄存器 （参见表 35）中选择的八个引脚中断， ISEL 寄存器的某个位决定了这些中断是边沿还是电平敏感。

表 115. 引脚中断模式寄存器 （ISEL，地址 0x4008 7000）位描述

位	符号	描述	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 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。	-	-

9.5.1.2 引脚中断电平 （上升沿）中断使能寄存器

对于 PINTSELn 寄存器中所选的八个引脚中断 （参见表 35）， IENR 寄存器的某个位会根据 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 寄存器配置该中断的有效电平 （高或低电平）。

表 116. 引脚中断电平 （上升沿）中断使能寄存器 （IENR，地址 0x4008 7004）位描述

位	符号	描述	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 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。	-	-

9.5.1.3 引脚中断电平 （上升沿）中断设置寄存器

对于 PINTSELn 寄存器中所选的八个引脚中断（参见表 35），SIENR 寄存器的某个位会根据 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)，则设置为上升沿中断。
- 如果引脚中断模式是电平敏感 (PMODE = 1)，则设置为电平中断。

表 117. 引脚中断电平 （上升沿）中断设置寄存器 （SIENR，地址 0x4008 7008）位描述

位	符号	描述	复位值	访问类型
7:0	SETENRL	向该地址写 1 会设置 IENR 中的位，从而使能中断。位 n 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。	-	-

9.5.1.4 引脚中断电平 （上升沿中断）清除寄存器

对于 PINTSELn 寄存器中所选的八个引脚中断（参见表 35），CIENR 寄存器的某个位会根据 ISEL 寄存器中配置的引脚中断模式清除 IENR 寄存器中的对应位：

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

表 118. 引脚中断电平 （上升沿中断）清除寄存器 （CIENR，地址 0x4008 700C）位描述

位	符号	描述	复位值	访问类型
7:0	CENRL	向该地址写 1 会清除 IENR 中的位，从而禁用中断。位 n 清除 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。	-	-

9.5.1.5 引脚中断有效电平 （下降沿）中断使能寄存器

对于 PINTSELn 寄存器中所选的八个引脚中断 （参见表 35）， IENF 寄存器的某个位会根据 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 119. 引脚中断有效电平 （下降沿）中断使能寄存器 （IENF，地址 0x4008 7010）位描述

位	符号	描述	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 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。	-	-

9.5.1.6 引脚中断有效电平 （下降沿）中断设置寄存器

对于 PINTSELn 寄存器中所选的八个引脚中断（参见表 35），SIENF 寄存器的某个位会根据 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)，则设置为下降沿中断。
- 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 120. 引脚中断有效电平 （下降沿中断）设置寄存器 （SIENF，地址 0x4008 7014）位描述

位	符号	描述	复位值	访问类型
7:0	SETENAF	向该地址写 1 会置位 IENF 中的位，从而使能中断。位 n 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。	-	-

9.5.1.7 引脚中断有效电平 （下降沿中断）清除寄存器

对于 PINTSELn 寄存器中所选的八个引脚中断（参见表 35），CIENF 寄存器的某个位会根据 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)，则清除下降沿中断。
- 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 121. 引脚中断有效电平 （下降沿）中断清除寄存器 （CIENF，地址 0x4008 7018）位描述

位	符号	描述	复位值	访问类型
7:0	CENAF	向该地址写 1 会清除 IENF 中的位，从而禁用中断。位 n 清除 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。	-	-

9.5.1.8 引脚中断上升沿寄存器

该寄存器的 1 表示 PINTSELn 寄存器中所选的引脚中断（参见表 35）上已检测到上升沿。向该寄存器写 1，清除上升沿检测。对于上升沿中断使能的引脚，该寄存器中的 1 产生一个中断请求。为 PINTSELn 寄存器中所选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 122. 引脚中断上升沿寄存器 （RISE，地址 0x4008 701C）位描述

位	符号	描述	复位值	访问类型
7:0	RDET	上升沿检测。位 n 检测 PINTSELn 中所选引脚的上升沿。 读 0：自复位或上一次向该位写 1 起，未在该引脚上检测到上升沿。 写 0：无操作。 读 1：自复位或上一次向该位写 1 起，检测到上升沿。 写 1：清除该引脚的上升沿检测。	0	R/W
31:8	-	保留。	-	-

9.5.1.9 引脚中断下降沿寄存器

该寄存器的 1 表示 PINTSELn 寄存器中所选的引脚中断（参见表 35）上已检测到下降沿。向该寄存器写 1，清除下降沿检测。对于下降沿中断使能的引脚，该寄存器中的 1 产生一个中断请求。为 PINTSELn 寄存器中所选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 123. 引脚中断下降沿寄存器 （FALL，地址 0x4008 7020）位描述

位	符号	描述	复位值	访问类型
7:0	FDET	下降沿检测。位 n 检测 PINTSELn 中所选引脚的下降沿。 读 0：自复位或上一次向该位写 1 起，未在该引脚上检测到下降沿。 写 0：无操作。 读 1：自复位或上一次向该位写 1 起，检测到下降沿。 写 1：清除该引脚的下降沿检测。	0	R/W
31:8	-	保留。	-	-

9.5.1.10 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，向该寄存器中写 1 会清除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 124. 引脚中断状态寄存器 （IST，地址 0x4008 7024）位描述

位	符号	描述	复位值	访问类型
7:0	PSTAT	引脚中断状态。位 n 返回状态、清除边沿中断，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）：清除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。	-	-

9.5.2 GPIO 组 0/ 组 1 中断寄存器描述

9.5.2.1 分组中断控制寄存器

表 125. GPIO 分组中断控制寄存器 （CTRL，地址 0x4005 C000 (GROUP0 INT) 和 0x4006 0000 (GROUP1 INT)）位描述

位	符号	值	描述	复位值
0	INT		分组中断状态。写 1 则清除该位。写入 0 无效。	0
		0	无中断请求挂起。	
		1	中断请求有效。	
1	COMB		组合分组中断的使能输入	0
		0	OR 功能：当任意一个使能输入有效时 （基于可编程的极性），生成一个分组中断。	
		1	AND 功能：当所有使能位有效时 （基于可编程的极性），生成一个中断。	
2	TRIG		分组中断触发	0
		0	边沿触发	
		1	电平触发	
31:3	-	-	保留	0

9.5.2.2 GPIO 分组中断端口极性寄存器

分组中断端口极性寄存器决定每个使能引脚的极性如何有助于构成分组中断。每个端口与其端口极性寄存器相关，且两个寄存器的值一起决定分组中断。

表 126. GPIO 分组中断端口 0 极性寄存器 （PORT_POL0，地址 0x4005 C020 (GROUP0 INT) 和 0x4006 0020 (GROUP1 INT)）位描述

位	符号	描述	复位值	访问类型
31:0	POL0	配置分组中断的端口 0 引脚的引脚极性。位 n 对应端口 0 的引脚 P0_n。 0 = 引脚为有效低电平。如果该引脚为低电平，则该引脚构成分组中断。 1 = 引脚为有效高电平。如果该引脚为高电平，则该引脚构成分组中断。	1	-

表 127. GPIO 分组中断端口 1 极性寄存器 （PORT_POL1，地址 0x4005 C024 (GROUP0 INT) 和 0x4006 0024 (GROUP1 INT)）位描述

位	符号	描述	复位值	访问类型
31:0	POL1	配置分组中断的端口 1 引脚的引脚极性。位 n 对应端口 1 的引脚 P1_n。 0 = 引脚为有效低电平。如果该引脚为低电平，则该引脚构成分组中断。 1 = 引脚为有效高电平。如果该引脚为高电平，则该引脚构成分组中断。	1	-

9.5.2.3 GPIO 分组中断端口使能寄存器

分组中断端口使能寄存器使能构成分组中断的引脚。每个端口与其端口使能寄存器相关，且两个寄存器的值一起决定构成分组中断的引脚。

表 128. GPIO 分组中断端口 0 使能寄存器 （PORT_ENA0，地址 0x4005 C040 (GROUP0 INT) 和 0x4006 0040 (GROUP1 INT)）位描述

位	符号	描述	复位值	访问类型
31:0	ENA0	使能分组中断的端口0引脚。位n对应端口0的引脚P0_n。 0 = 端口 0 引脚禁用，不构成分组中断。 1 = 端口 0 引脚使能，构成分组中断。	0	-

表 129. GPIO 分组中断端口 1 使能寄存器 （PORT_ENA1，地址 0x4005 C044 (GROUP0 INT) 和 0x4006 0044 (GROUP1 INT)）位描述

位	符号	描述	复位值	访问类型
31:0	ENA1	使能分组中断的端口1引脚。位n对应端口0的引脚P1_n。 0 = 端口 1 引脚禁用，不构成分组中断。 1 = 端口 1 引脚使能，构成分组中断。	0	-

9.5.3 GPIO 端口寄存器描述

9.5.3.1 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。通常，软件通过读取和写入字节来访问各个引脚，也可以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 130. GPIO 端口 0 字节引脚寄存器 （B0 至 B31，地址 0x5000 0000 至 0x5000 001F）位描述

位	符号	描述	复位值	访问类型
0	PBYTE	读：引脚 P0_n 的状态，引脚的方向、屏蔽或可选功能都不会影响结果，除非引脚配置为模拟 I/O，则会始终读为 0。 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	-

表 131. GPIO 端口 1 字节引脚寄存器 （B32 至 B63，地址 0x5000 0020 至 0x5000 002F）位描述

位	符号	描述	复位值	访问类型
0	PBYTE	读：引脚 P1_n 的状态，引脚的方向、屏蔽或可选功能都不会影响结果，除非引脚配置为模拟 I/O，则会始终读为 0。 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	-

9.5.3.2 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，引脚的方向、屏蔽或可选功能都不会影响结果，除非引脚配置为模拟 I/O，则会始终读为 0。如果写入的值均为 0，则任何写入都将清除引脚的输出位，否则会设置引脚的输出位。

表 132. GPIO 端口 0 字引脚寄存器（W0 至 W31，地址 0x5000 1000 至 0x5000 107C）位描述

位	符号	描述	复位值	访问类型
31:0	PWORD	读 0：引脚为低电平。 写 0：清除输出位。 读 0xFFFF FFFF：引脚为高电平。 写 0x0000 0001 至 0xFFFF FFFF 之间的任何值：置位输出位。 **注：**仅可读取 0 或 0xFFFF FFFF。写 0 以外的任何值时，将置位输出位。	ext	R/W

表 133. GPIO 端口 1 字引脚寄存器（W32 至 W63，地址 0x5000 1080 至 0x5000 10FC）位描述

位	符号	描述	复位值	访问类型
31:0	PWORD	读 0：引脚为低电平。 写 0：清除输出位。 读 0xFFFF FFFF：引脚为高电平。 写 0x0000 0001 至 0xFFFF FFFF 之间的任何值：置位输出位。 **注：**仅可读取 0 或 0xFFFF FFFF。写 0 以外的任何值时，将置位输出位。	ext	R/W

9.5.3.3 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 134. GPIO 方向端口 0 寄存器（DIR0，地址 0x5000 2000）位描述

位	符号	描述	复位值	访问类型
31:0	DIRP0	选择引脚 P0_n 的引脚方向（位 0 = P0_0，位 1 = P0_1，...，位 31 = P0_31）。 0 = 输入。 1 = 输出。	0	R/W

表 135. GPIO 方向端口 1 寄存器（DIR1，地址 0x5000 2004）位描述

位	符号	描述	复位值	访问类型
31:0	DIRP1	选择引脚 P1_n 的引脚方向（位 0 = P1_0，位 1 = P1_1，...，位 31 = P1_31）。 0 = 输入。 1 = 输出。	0	R/W

9.5.3.4 GPIO 端口屏蔽寄存器

这些寄存器会影响读写 MPORT 寄存器。将这些寄存器设为 0，可以使能读写；设为 1 则禁用写入以及让对应位置读为 0。

表 136. GPIO 屏蔽端口 0 寄存器 （MASK0，地址 0x5000 2080）位描述

位	符号	描述	复位值	访问类型
31:0	MASKP0	控制 P0MPORT 寄存器中哪些对应 P0_n 的位是有效的（位 0 = P0_0，位 1 = P0_1， ...，位 31 = P0_31）。 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

表 137. GPIO 屏蔽端口 1 寄存器 （MASK1，地址 0x5000 2084）位描述

位	符号	描述	复位值	访问类型
31:0	MASKP1	控制 P1MPORT 寄存器中哪些对应 P1_n 的位是有效的（位 0 = P1_0，位 1 = P1_1， ...，位 31 = P1_31）。 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

9.5.3.5 GPIO 端口引脚寄存器

读取这些寄存器会返回所读引脚的当前状态，引脚的方向、屏蔽或可选功能都不会影响结果，除非引脚配置为模拟 I/O，则会始终读为 0。写这些寄存器会加载所写引脚的输出位，无论是否为 MASK 寄存器。

表 138. GPIO 端口 0 引脚寄存器 （PIN0，地址 0x5000 2100）位描述

位	符号	描述	复位值	访问类型
31:0	PORT0	读取引脚状态或加载输出位（位 0 = P0_0，位 1 = P0_1，...，位 31 = P0_31）。 0 = 读：引脚为低电平；写入：清除输出位。 1 = 读：引脚为高电平；写入：置位输出位。	ext	R/W

表 139. GPIO 端口 1 引脚寄存器 （PIN1，地址 0x5000 2104）位描述

位	符号	描述	复位值	访问类型
31:0	PORT1	读取引脚状态或加载输出位（位 0 = P1_0，位 1 = P1_1，...，位 31 = P1_31）。 0 = 读：引脚为低电平；写入：清除输出位。 1 = 读：引脚为高电平；写入：置位输出位。	ext	R/W

9.5.3.6 GPIO 屏蔽端口引脚寄存器

这些寄存器与 PORT 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 AND 操作，可以屏蔽读取某些值；同时，写其中一个寄存器仅影响在相应 MASK 寄存器中由 0 使能的输出寄存器位

表 140. GPIO 屏蔽端口 0 引脚寄存器 （MPIN0，地址 0x5000 2180）位描述

位	符号	描述	复位值	访问类型
31:0	MPORTP0	屏蔽端口寄存器（位 0 = P0_0，位 1 = P0_1， ...，位 31 = P0_31）。 0 = 读：引脚为低电平和 / 或 MASK 寄存器中的对应位为 1；写入：当 MASK 寄存器中的对应位为 0 时，清除输出位。 1 = 读：引脚为高电平和 MASK 寄存器中的对应位为 0 ；写入：当 MASK 寄存器中的对应位为 0 时，置位输出位。	ext	R/W

表 141. GPIO 屏蔽端口 1 引脚寄存器 （MPIN1，地址 0x5000 2184）位描述

位	符号	描述	复位值	访问类型
31:0	MPORTP1	屏蔽端口寄存器（位 0 = P1_0，位 1 = P1_1， ...，位 31 = P1_31）。 0 = 读：引脚为低电平和 / 或 MASK 寄存器中的对应位为 1；写入：当 MASK 寄存器中的对应位为 0 时，清除输出位。 1 = 读：引脚为高电平和 MASK 寄存器中的对应位为 0；写入：当 MASK 寄存器中的对应位为 0 时，置位输出位。	ext	R/W

9.5.3.7 GPIO 端口设置寄存器

向这些寄存器写 1，可以置位输出位，无论其是否为 MASK 寄存器。读这些寄存器会返回端口的输出位，无论其引脚方向如何。

表 142. GPIO 设置端口 0 寄存器 （SET0，地址 0x5000 2200）位描述

位	符号	描述	复位值	访问类型
31:0	SETP0	读取或设置输出位。 0 = 读：输出位；写入：无操作。 1 = 读：输出位；写入：置位输出位。	0	R/W

表 143. GPIO 设置端口 1 寄存器 （SET1，地址 0x5000 2204）位描述

位	符号	描述	复位值	访问类型
31:0	SETP1	读取或设置输出位。 0 = 读：输出位；写入：无操作。 1 = 读：输出位；写入：置位输出位。	0	R/W

9.5.3.8 GPIO 端口清除寄存器

向这些只写寄存器写 1，可以清除输出位，无论是否为 MASK 寄存器。

表 144. GPIO 清除端口 0 寄存器 （CLR0，地址 0x5000 2280）位描述

位	符号	描述	复位值	访问类型
31:0	CLRP0	清除输出位： 0 = 无操作。 1 = 清除输出位。	不适用	WO

表 145. GPIO 清除端口 1 寄存器 （CLR1，地址 0x5000 2284）位描述

位	符号	描述	复位值	访问类型
31:0	CLRP1	清除输出位： 0 = 无操作。 1 = 清除输出位。	不适用	WO

9.5.3.9 GPIO 端口切换寄存器

向这些只写寄存器写 1，可以切换 / 反转 / 补充输出位，无论是否为 MASK 寄存器。

表 146. GPIO 切换端口 0 寄存器 （NOT0，地址 0x5000 2300）位描述

位	符号	描述	复位值	访问类型
31:0	NOTP0	切换输出位： 0 = 无操作。 1 = 切换输出位。	不适用	WO

表 147. GPIO 切换端口 1 寄存器 （NOT1，地址 0x5000 2304）位描述

位	符号	描述	复位值	访问类型
31:0	NOTP1	切换输出位： 0 = 无操作。 1 = 切换输出位。	不适用	WO

9.6 功能说明

9.6.1 读取引脚状态

软件可以读取所有 GPIO 引脚的状态，除在 I/O 配置逻辑中选为模拟输入或输出的引脚外。要读取某个引脚的状态，并不一定要在 I/O 配置中将其选为 GPIO。共有 4 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。引脚在端口的屏蔽寄存器中读为 1，在 MPORT 寄存器中将读为 0。

9.6.2 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。如需将引脚的输出位驱动到引脚，必须符合两个条件：

1. 在 I/O 配置模块中，必须将该引脚选为 GPIO 操作；
2. 该引脚必须选为输出，方法为将其端口的 DIR 寄存器设为 1。

如果其中一个或两个条件未能满足，写入到引脚无效。

共有 7 种方式可以更改 GPIO 输出位：

- 写入到字节引脚寄存器会从最低有效位加载输出位。
- 写入到字引脚寄存器会通过所有写入位的 OR 操作加载输出位。（该功能遵循程序设计语言中多位值为真的定义。）
- 写入到端口的 PORT 寄存器会加载所有写入引脚的输出位。
- 写入到端口的 MPORT 寄存器会加载由该端口 MASK 寄存器对应位置的 0 确定的引脚输出位。
- 向端口的 SET 寄存器写 1 会设置输出位。
- 向端口的 CLR 寄存器写 1 会清除输出位。
- 向端口的 NOT 寄存器写 1 会切换 / 补充 / 反转输出位。

可以从 SET 寄存器中读取某个端口的输出位状态。读第 9.6.1 节中描述的任何寄存器会返回引脚状态，无论其方向或可选功能如何。

9.6.3 屏蔽 I/O

端口的 MASK 寄存器确定哪些引脚应当在 MPORT 寄存器中可访问。MASK 中的位为 0 时，可以使能从 MPORT 中读取和写入的对应引脚。MASK 中的位为 1 时，引脚会强制读为 0，其输出位将不会受写入到 MPORT 的影响。当端口的 MASK 寄存器中的位全为 0 时，其 PORT 和 MPORT 寄存器将以完全一致的方式进行读写。

对于使用具有类似 GPIO 模块的前代恩智浦设备的用户，应注意其不可兼容性：在 LPC11A1x 上，写入到 SET、 CLR 和 NOT 寄存器不会受 MASK 寄存器的影响。前代设备上屏蔽了这些寄存器。

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屏蔽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 寄存器的代码块的更简单方式是：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。

9.6.4 GPIO 中断

提供两种独立的 GPIO 中断机制。引脚中断：多达 8 个 GPIO 引脚可以拥有独立的向量、边沿或电平敏感中断。

分组中断：每个端口的任一引脚子集都可以构成一个通用中断。可以使能任何引脚和端口中断，将器件从深度睡眠模式或掉电模式下唤醒。

9.6.4.1 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。引脚中断模块中的所有寄存器中均包含 8 个位，对应 PINTSEL0-7 寄存器调用的各个引脚。ISEL 寄存器确定每个中断引脚是边沿敏感还是电平敏感。 RISE 和 FALL 寄存器检测每个中断引脚的边沿，其写操作可以清除 （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，写此寄存器还可以清除中断。

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 148 中的描述。

表 148. 边沿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。

9.6.4.2 分组中断

在该中断机制中，根据选择的每个端口的任一引脚子集，都可以为每个端口请求中断。将端口的使能寄存器设为 1，选出构成每个端口中断的引脚，同时还可在端口的极性寄存器中选择每个引脚的中断极性。每个引脚的电平与其极性位进行异或操作，结果与其使能位进行 AND 操作；然后这些结果再与该端口的所有引脚进行兼或操作，从而创建了该端口的原始中断请求。

两个分组中断的原始中断请求发送到 NVIC，经过编程会将其作为电平或边沿敏感（参见第 6.4 节）或者由唤醒中断逻辑对其进行边沿检测 （参见表 39）。

9.6.5 推荐用法

下表列出了 GPIO 端口寄存器的若干推荐用法：

- 对于复位或重新初始化后的初始设置，写 PORT 寄存器。
- 如需更改某个引脚的状态，写字节引脚或字引脚寄存器。
- 如需一次性更改多个引脚的状态，写 SET 和 / 或 CLR 寄存器。
- 如需在严格控制的环境（如软件状态机）中更改多个引脚的状态，考虑使用 NOT 寄存器。这比 SET 和 CLR 需要的写操作更少。
- 如需读取某个引脚的状态，读字节引脚或字引脚寄存器。
- 如需根据多个引脚作出决定，读取并屏蔽 PORT 寄存器。

10.1 本章导读

LPC1345/46/47 器件上包含 USB 模块。

10.2 基本配置

- 引脚：在 IOCON 寄存器模块中配置 USB 引脚。
- 在SYSAHBCLKCTRL寄存器中，设置位14使能USB控制器寄存器接口的时钟；设置位27 使能 USB RAM 的时钟 （参见表 19）。
- 电源：在 PDRUNCFG 寄存器（表 42）中使能 USB PHY 和 USB PLL 的电源（若使用）。
- 配置 USB 主时钟 （参见表 25）。
- 需要时，配置 USB 唤醒信号 （参见第 10.7.6 节）。

10.3 特性

- USB2.0 全速设备控制器。
- 支持 10 个物理 （5 个逻辑）端点，包括一个控制端点。
- 支持单缓冲和双缓冲。
- 每个非控制端点都支持批量、中断或等时端点类型。
- 支持从 USB 活动上的深度睡眠模式唤醒和远程唤醒。
- 支持 SoftConnect。

10.4 简介

通用串行总线 (USB) 是一条四线总线，支持主机和一或多个（最多 127 个）外设之间的通信。主机控制器通过一个基于令牌的协议将 USB 带宽分配到连接的设备上。总线支持热插拔和动态配置设备。所有传送都由主机控制器发起。

主机以 1 ms 的帧计划传送。每个帧都包含帧起始 (SOF) 标志以及与设备端点之间传输数据的传送。每个设备最多可以有 16 个逻辑端点或者 32 个物理端点。LPC1315/16/17/45/46/47 设备控制器支持多达 10 个物理端点。这些端点有四种定义的传输类型。控制传输用于配置设备。

中断传输用于定期数据传输。传输延迟不重要时，使用批量传输。同步传输能够保证传输时间但不能保证错误纠正。

有关通用串行总线的更多信息，请参见 USB 实施者论坛网站。

LPC1315/16/17/45/46/47 上的 USB 设备控制器使能与 USB 主机控制器的全速 (12 Mb/s) 数据交换。

图 10 显示了 USB 设备控制器框图。

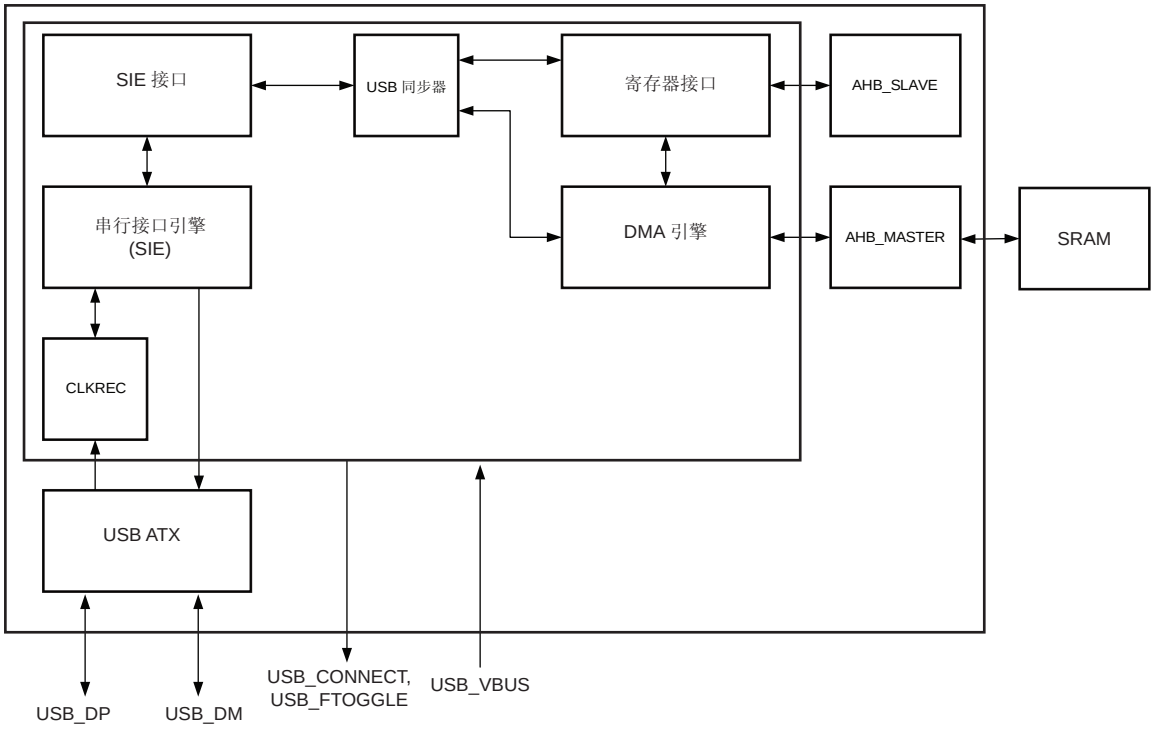


图 10. USB 框图

USB 设备控制器含一个内置模拟收发器 (ATX)。USB ATX 发送 / 接收 USB 总线的双向 USB_DP 和 USB_DM 信号。

SIE 实施完整的 USB 协议层。它完全采用硬连线以确保速度，不需要软件干预。它将处理 USB RAM 中的端点缓冲区和 USB 总线之间的数据传输。此模块的功能包括：同步模式识别、并行 / 串行转换、位填充 / 解除填充、CRC 检查 / 生成、PID 验证 / 生成、地址识别和信号交换评估 / 生成。

10.4.1 USB 软件接口

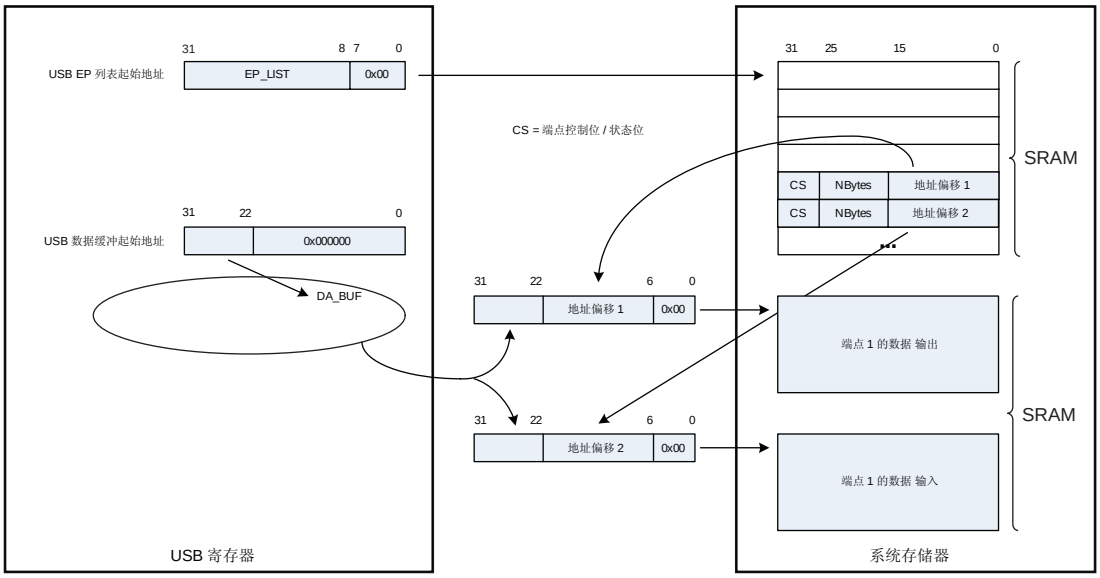


图 11. USB 软件接口

10.4.2 固定的端点配置

表 149 列出了所支持的端点配置。每种端点类型的数据包大小可配置到表 149 中显示的最大值。

表 149. 固定的端点配置

逻辑端点	物理端点	端点类型	方向	最大数据包大小（字节）	双缓冲
0	0	控制	输出	64	无
0	1	控制	输入	64	无
1	2	中断 / 批量 / 等时	输出	64/64/1023	有
1	3	中断 / 批量 / 等时	输入	64/64/1023	有
2	4	中断 / 批量 / 等时	输出	64/64/1023	有
2	5	中断 / 批量 / 等时	输入	64/64/1023	有
3	6	中断 / 批量 / 等时	输出	64/64/1023	有
3	7	中断 / 批量 / 等时	输入	64/64/1023	有
4	8	中断 / 批量 / 等时	输出	64/64/1023	有
4	9	中断 / 批量 / 等时	输入	64/64/1023	有

10.4.3 SoftConnect

通过 1.5 kOhm 上拉电阻将 USB_DP （用于全速设备）上拉至高电平，完成与 USB 的连接。 SoftConnect 功能可用于允许软件在决定建立与 USB 的连接之前完成其初始化序列。无需拔掉电缆就可重新初始化 USB 总线连接。

要使用 SoftConnect 功能，CONNECT 信号应控制一个连接到 USB_DP 和 V_{DD} (+3.3 V) 之间 1.5 kOhm 电阻的外部开关。然后，软件可以通过写入 DEVCMDSTAT 寄存器的 DCON 位控制 CONNECT 信号。

10.4.4 中断

USB 控制器有两个中断线路 USB_Int_Req_IRQ 和 USB_Int_Req_FIQ。软件可编程 USB 中断路由寄存器的相应位，以便将中断条件发送至 NVIC 表表 53 中的一个条目。如果同时设置中断状态位和相应中断使能位，硬件将生成一个中断。如果发生中断条件 （不论是否设置中断使能位），硬件将设置中断状态位。

10.4.5 挂起和恢复

USB 协议坚持由 USB 设备管理电源。设备消耗总线电源时（总线供电设备），这将变得更加重要。总线供电设备应满足以下条件。

- 非配置状态中的设备从 USB 总线消耗的电流不得超过 100mA。
- 已配置设备消耗的电流不得超过配置描述符中最大功率字段指定的值。最大值为500 mA。
- 挂起设备消耗的电流不得超过 500 µA。

当 USB 总线超过 3 ms 没有活动时，设备将进入 L2 挂起状态。如果有来自主机的传输，将唤醒挂起设备（主机启动的唤醒）。LPC1315/16/17/45/46/47 中的 USB 控制器还支持软件启动的远程唤醒。要启动远程唤醒，设备上的软件必须使能全部时钟并清除挂起位。这将引发硬件在上游生成远程唤醒信号。

USB 控制器支持链路电源管理。链路电源管理用于定义附加链接电源管理状态 L1，利用大多数现有挂起/恢复基础设施补充现有 L2 状态，但提供 L1 和 L0（开）之间更快的跳变延迟。

USB 挂起信号的产生表示过去 3 ms 内 USB 总线上没有活动。这时，将中断发送至处理器，软件即可开始准备挂起设备。

如果在接下来的 2 ms 内也没有活动， USB need_clock 信号将变为低电平。这表明可关断 USB 主时钟。

检测到 USB 总线上的活动时，停用 USB 挂起信号并激活 USB need_clock 信号。此过程是完全组合的，因此不需要使用 USB 主时钟激活 USB need_clock 信号。

10.4.6 帧切换输出

USB_FTOGGLE 输出引脚反映源自 USB 主机发送的输入帧起始令牌的 1 kHz 时钟。

10.4.7 时钟

LPC1315/16/17/45/46/47 USB 设备控制器有以下时钟连接：

- USB 主时钟：USB 主时钟是来自专用 USB PLL 的 48 MHz +/- 500 ppm 时钟或主时钟（参见表 25）。如果使用主时钟，系统 PLL 输出必须为 48 MHz 并且源自系统振荡器。USB 主时钟用于从 USB 主机恢复 12 MHz 时钟。
- AHB 时钟：这是 AHB 系统总线时钟。 USB 设备控制器接收或发送 USB 数据包时，AHB 时钟的最低频率是 16 MHz。

10.5 引脚说明

设备控制器可访问一个 USB 端口。

表 150. USB 设备引脚描述

名称	方向	描述
V_{BUS}	I	V_{BUS} 状态输入。没有通过相应的 IOCON 寄存器使能此功能时，从内部将此功能驱动为高电平。
USB_CONNECT	O	SoftConnect 控制信号。
USB_FTOGGLE	O	USB 1 ms SoF 信号。
USB_DP	I/O	正差分数据。
USB_DM	I/O	负差分数据。

10.6 寄存器描述

表 151. 寄存器简介：USB （基址：0x4008 0000）

名称	访问类型	地址偏移	描述	复位值	参考
DEVCMDSTAT	R/W	0x000	USB 设备命令 / 状态寄存器	0x00000800	表 152
INFO	R/W	0x004	USB 信息寄存器	0	表 153
EPLISTSTART	R/W	0x008	USB EP 命令 / 状态列表起始地址	0	表 154
DATABUFSTART	R/W	0x00C	USB 数据缓冲起始地址	0	表 155
LPM	R/W	0x010	链路电源管理寄存器	0	表 156
EPSKIP	R/W	0x014	USB 端点跳过	0	表 157
EPINUSE	R/W	0x018	使用中的 USB 端点缓冲区	0	表 158
EPBUFCFG	R/W	0x01C	USB 端点缓冲区配置寄存器	0	表 159
INTSTAT	R/W	0x020	USB 中断状态寄存器	0	表 160
INTEN	R/W	0x024	USB 中断使能寄存器	0	表 161
INTSETSTAT	R/W	0x028	USB 设置中断状态寄存器	0	表 162
INTROUTING	R/W	0x02C	USB 中断路由寄存器	0	表 163
EPTOGGLE	R	0x034	USB 端点切换寄存器	0	表 164

10.6.1 USB 设备命令 / 状态寄存器 (DEVCMDSTAT)

表 152. USB 设备命令 / 状态寄存器 （DEVCMDSTAT，地址 0x4008 0000）位描述

位	符号	值	描述	复位值	访问类型
6:0	DEV_ADDR		USB 设备地址。总线复位后，将地址复位为 0x00。如果设置使能位，设备将响应功能地址 DEV_ADDR 的数据包。从 USB 主机接收到 SetAddress 控制请求时，软件必须在完成 SetAddress 控制请求的状态阶段前编程新地址。	0	RW
7	DEV_EN		USB 设备使能。如果设置此位， HW 将开始响应功能地址 DEV_ADDR 的数据包。	0	RW
8	SETUP		已接收到 SETUP 令牌。如果设备接收到并应答 SETUP 令牌，则设置此位。一旦设置此位，HW 将 NAK 所有接收到的 IN 和 OUT 令牌。SW 必须通过写入 1 清除此位。如果此位为 0，HW 将按照 SW 编程的 CTRL EP0 IN 和 OUT 数据信息，处理发送到 CTRL EP0 的令牌。	0	RWC
9	PLL_ON		USB 时钟 /PLL 控制。	0	RW
		0	USB_NeedClk 功能		
		1	USB_NeedClk 总是为 1。挂起时，时钟不会停止。		
10	-		保留。	0	RO
11	LPM_SUP		支持 LPM。	1	RW
		0	不支持 LPM。		
		1	支持 LPM。		
12	INTONNAK_AO		NAK 上用于中断和批量 OUT EP 的中断	0	RW
		0	只有应答的数据包可产生中断		
		1	应答和 NAK 的数据包都可产生中断。		
13	INTONNAK_AI		NAK 上用于中断和批量 IN EP 的中断	0	RW
		0	只有应答的数据包可产生中断		
		1	应答和 NAK 的数据包都可产生中断。		
14	INTONNAK_CO		NAK 上用于控制 OUT EP 的中断	0	RW
		0	只有应答的数据包可产生中断		
		1	应答和 NAK 的数据包都可产生中断。		
15	INTONNAK_CI		NAK 上用于控制 IN EP 的中断	0	RW
		0	只有应答的数据包可产生中断		
		1	应答和 NAK 的数据包都可产生中断。		
16	DCON		设备状态 - 连接。 连接位须由 SW 设置，表示设备必须发出连接信号。设置此位且 VbusDebounced 位为 1 时，使能 USB_DP 上的上拉电阻。	0	RW
17	DSUS		设备状态 - 挂起。 挂起位表示当前挂起状态。如果设备在超过 3 毫秒以上未在上游端口检测到任何活动，将此位设为 1。如果检测到任何活动，则将此位复位为 0。设备挂起 （挂起位 DSUS = 1）且软件向其写入 0 时，设备将生成一个远程唤醒。仅当设备已连接时 （连接位 = 1）时会发生此种情况。设备未连接或者未挂起时，写入 0 无效。写入 1 总是无效。	0	RW
18	-		保留。	0	RO

表 152. USB 设备命令 / 状态寄存器 （DEVCMDSTAT，地址 0x4008 0000）位描述 *（续）*

位	符号	值	描述	复位值	访问类型
19	LPM_SUS		设备状态 - LPM 挂起。 此位代表当前 LPM 挂起状态。当设备应答来自 USB 主机的 LPM 请求且令牌重试时间 10 us 已过时， HW 将此位设为 1。当设备处于 LPM 挂起状态（LPM 挂起位 = 1）且软件向此位写入 0 时，设备将生成一个远程唤醒。当 LPM_REWP 位设为 1 时，软件只能向此位写入 0。接收到主机启动的恢复时， HW 复位此位。当 LPM_SUPP 位等于 1 时， HW 只更新 LPM_SUS 位。	0	RW
20	LPM_REWP		USB 主机使能的 LPM 远程唤醒。 将 LPM 扩展令牌中的 bRemoteWake 位设为 1 时，HW 将此位设为 1。接收到主机启动的 LPM 恢复、设备发送远程唤醒或者接收到 USB 总线复位时， HW 将此位复位为 0。软件将使用此位检查主机是否使能远程唤醒功能用于 LPM 传送。	0	RO
23:21	-		保留。	0	RO
24	DCON_C		设备状态 - 连接更改。 由于 VBus 消失而断开设备的上拉电阻时，设置连接更改位。写 1 则复位该位。	0	RWC
25	DSUS_C		设备状态 - 挂起更改。 切换挂起位时，将挂起更改位设为 1。由于以下原因，挂起位可切换： - 设备进入挂起状态 - 设备断开 - 设备接收到上游端口的恢复信号。 写 1 则复位该位。	0	RWC
26	DRES_C		设备状态 - 复位更改。 设备接收到总线复位时，设置此位。总线复位时，设备将自动进入默认状态 （未配置且响应地址 0）。写 1 则复位该位。	0	RWC
27	-		保留。	0	RO
28	VBUSDEBOUNCED		此位表示是否检测到 Vbus。Vbus 到达高电平时，此位立即上升。Vbus 至少有 3 ms 处于低电平时，此位下降至 0。如果此位上升至高电平并设置 DCon 位，HW 将使能上拉电阻以发出连接信号。	0	RO
31:29	-		保留。	0	RO

10.6.2 USB 信息寄存器 (INFO)

表 153. USB 信息寄存器 （INFO，地址 0x4008 0004）位描述

位	符号	值	描述	复位值	访问类型
10:0	FRAME_NR		帧编号。这包括最后一个成功接收到的 SOF 的帧编号。如果在帧起始时，设备未接收到 SOF，则返回的帧编号是最后一个成功接收到的 SOF 的编号。如果 SOF 帧编号包含一个 CRC 错误，则返回的帧编号将是设备接收到的损坏的帧编号。	0	RO
14:11	ERR_CODE		最后出现的错误代码：	0	RW
		0x0	无错误		
		0x1	PID 编码错误		
		0x2	PID 未知		
		0x3	意外数据包		
		0x4	令牌 CRC 错误		
		0x5	数据 CRC 错误		
		0x6	超时		
		0x7	混串音		
		0x8	截断 EOP		
		0x9	发送 / 接收 NAK		
		0xA	发送停止		
		0xB	溢出		
		0xC	发送空数据包		
		0xD	位填充错误		
		0xE	同步错误		
		0xF	错误数据切换		
15	-		保留。	0	RO
31:16	-	-	保留	-	RO

10.6.3 USB EP 命令 / 状态列表起始地址 (EPLISTSTART)

32 位寄存器表示 USB EP 命令 / 状态列表起始地址。软件只编程这些位的子集。由于此列表必须在256字节边界上开始，所以将8个最低有效位硬编码为0。可用软件编程位31到8。

表 154. USB EP 命令 / 状态列表起始地址 （EPLISTSTART，地址 0x4008 0008）位描述

位	符号	描述	复位值	访问类型
7:0	-	保留	0	RO
31:8	EP_LIST	USB EP 命令 / 状态列表的起始地址。	0	R/W

10.6.4 USB 数据缓冲起始地址 (DATABUFSTART)

此寄存器指出可以放置端点数据的 AHB 地址页面。

表 155. USB 数据缓冲起始地址 （DATABUFSTART，地址 0x4008 000C）位描述

位	符号	描述	复位值	访问类型
21:0	-	保留	0	R
31:22	DA_BUF	可放置全部端点数据缓冲的缓冲指针页面的起始地址。	0	R/W

10.6.5 链路电源管理寄存器 (LPM)

表 156. 链路电源管理寄存器 （LPM，地址 0x4008 0010）位描述

位	符号	描述	复位值	访问类型
3:0	HIRD_HW	主机启动的恢复持续时间 - HW。这是来自最后接收到的 LPM 令牌的 HIRD 值	0	RO
7:4	HIRD_SW	主机启动的恢复持续时间 - SW。这是 USB 设备系统要求的从接收到主机启动的 LPM 恢复后到退出 LPM 启动的挂起的持续时间。	0	R/W
8	DATA_PENDING	一旦将此位设为 1 且 LPM 的支持位设为 1，HW 将在每接收到一个 LPM 令牌时返回 NYET 信号交换。如果将 LPM 的支持位设为 1 并将此位设为 0，HW 将在每接收到一个LPM令牌时返回ACK信号交换。如果 SW 还有数据挂起且支持 LPM，则必须将此位设置为 1。	0	R/W
31:9	-	保留	0	RO

10.6.6 USB 端点跳过 (EPSKIP)

表 157. USB 端点跳过 （EPSKIP，地址 0x4008 0014）位描述

位	符号	描述	复位值	访问类型
29:0	SKIP	端点跳过：向其中一位写入 1，向 HW 表示必须停用分配到此端点的缓冲并将控制返回到软件。HW 停用端点时，将清除此位，但不会修改 EPINUSE 位。 当有效位从 1 变为 0 时，将生成一个中断。 注意：在双缓冲中，HW 只会清除 EPINUSE 位指示的缓冲的有效位。	0	R/W
31:30	-	保留	0	R

10.6.7 使用中的 USB 端点缓冲 (EPINUSE)

表 158. 使用中的 USB 端点缓冲 （EPINUSE，地址 0x4008 0018）位描述

位	符号	描述	复位值	访问类型
1:0	-	保留。由于每个物理端点上的控制端点 0 都固定为单缓冲，此位固定为 0。	0	R
9:2	BUF	使用中的缓冲：此寄存器在每个物理端点上都有一位。 0：HW 正在访问缓冲 0。 1：HW 正在访问缓冲 1。	0	R/W
31:10	-	保留	0	R

10.6.8 USB 端点缓冲配置 (EPBUFCFG)

表 159. USB 端点缓冲配置 （EPBUFCFG，地址 0x4008 001C）位描述

位	符号	描述	复位值	访问类型
1:0	-	保留。由于每个物理端点上的控制端点 0 都固定为单缓冲，此位固定为 0。	0	R
9:2	BUF_SB	缓冲使用：此寄存器在每个物理端点上都有一位。 0：单缓冲。 1：双缓冲。 如果将此位设为单缓冲 (0)，清除有效位时不会切换相应 EPINUSE 位。如果将此位设为双缓冲 (1)，清除缓冲区的有效位时 HW 将切换 EPINUSE 位。	0	R/W
31:10	-	保留	0	R

10.6.9 USB 中断状态寄存器 (INTSTAT)

表 160. USB 中断状态寄存器 （INTSTAT，地址 0x4008 0020）位描述

位	符号	描述	复位值	访问类型
0	EP0OUT	控制 EP0 OUT 方向的中断状态寄存器位。 NByte 跳变为 0、软件设置了跳过位或者控制 EP0 成功接收到 SETUP 数据包时，将设置该位。 如果设置 IntOnNAK_CO，发送控制 EP0 OUT 方向的 NAK 时，也会设置该位。 软件可通过向该位写入 1 清除该位。	0	R/WC
1	EP0IN	控制 EP0 IN 方向的中断状态寄存器位。 NByte 跳变为 0 或者软件设置了跳过位时，将设置该位。 如果设置 IntOnNAK_CI，发送控制 EP0 IN 方向的 NAK 时，也会设置该位。 软件可通过向该位写入 1 清除该位。	0	R/WC
2	EP1OUT	EP1 OUT 方向的中断状态寄存器位。 如果 HW 清除相应的有效位，将设置该位。已编程的 NByte 跳变为 0 或者软件设置了跳过位时，将完成此操作。 如果设置 IntOnNAK_AO，发送 EP1 OUT 方向的 NAK 时，也会设置该位。 软件可通过向该位写入 1 清除该位。	0	R/WC
3	EP1IN	EP1 IN 方向的中断状态寄存器位。 如果 HW 清除相应的有效位，将设置该位。已编程的 NByte 跳变为 0 或者软件设置了跳过位时，将完成此操作。 如果设置 IntOnNAK_AI，发送 EP1 IN 方向的 NAK 时，也会设置该位。 软件可通过向该位写入 1 清除该位。	0	R/WC
4	EP2OUT	EP2 OUT 方向的中断状态寄存器位。 如果 HW 清除相应的有效位，将设置该位。已编程的 NByte 跳变为 0 或者软件设置了跳过位时，将完成此操作。 如果设置 IntOnNAK_AO，发送 EP2 OUT 方向的 NAK 时，也会设置该位。 软件可通过向该位写入 1 清除该位。	0	R/WC
5	EP2IN	EP2 IN 方向的中断状态寄存器位。 如果 HW 清除相应的有效位，将设置该位。已编程的 NByte 跳变为 0 或者软件设置了跳过位时，将完成此操作。 如果设置 IntOnNAK_AI，发送 EP2 IN 方向的 NAK 时，也会设置该位。 软件可通过向该位写入 1 清除该位。	0	R/WC
6	EP3OUT	EP3 OUT 方向的中断状态寄存器位。 如果 HW 清除相应的有效位，将设置该位。已编程的 NByte 跳变为 0 或者软件设置了跳过位时，将完成此操作。 如果设置 IntOnNAK_AO，发送 EP3 OUT 方向的 NAK 时，也会设置该位。 软件可通过向该位写入 1 清除该位。	0	R/WC

表 160. USB 中断状态寄存器 （INTSTAT，地址 0x4008 0020）位描述 *（续）*

位	符号	描述	复位值	访问类型
7	EP3IN	EP3 IN 方向的中断状态寄存器位。 如果 HW 清除相应的有效位，将设置该位。已编程的 NByte 跳变为 0 或者软件设置了跳过位时，将完成此操作。 如果设置 IntOnNAK_AI，发送 EP3 IN 方向的 NAK 时，也会设置该位。 软件可通过向该位写入 1 清除该位。	0	R/WC
8	EP4OUT	EP4 OUT 方向的中断状态寄存器位。 如果 HW 清除相应的有效位，将设置该位。已编程的 NByte 跳变为 0 或者软件设置了跳过位时，将完成此操作。 如果设置 IntOnNAK_AO，发送 EP4 OUT 方向的 NAK 时，也会设置该位。 软件可通过向该位写入 1 清除该位。	0	R/WC
9	EP4IN	EP4 IN 方向的中断状态寄存器位。 如果 HW 清除相应的有效位，将设置该位。已编程的 NByte 跳变为 0 或者软件设置了跳过位时，将完成此操作。 如果设置 IntOnNAK_AI，发送 EP4 IN 方向的 NAK 时，也会设置该位。 软件可通过向该位写入 1 清除该位。	0	R/WC
29:10	-	保留	0	RO
30	FRAME_INT	帧中断。 设置了 VbusDebounced 位和 DCON 位时，每毫秒会将该位设置为 1。处理等时端点时，软件可使用此位。 软件可通过向该位写入 1 清除该位。	0	R/WC
31	DEV_INT	设备状态中断。设置了设备状态更改寄存器中的一个位时，HW 将设置此位。软件可通过向该位写入 1 清除该位。	0	R/WC

10.6.10 USB 中断使能寄存器 (INTEN)

表 161. USB 中断使能寄存器 （INTEN，地址 0x4008 0024）位描述

位	符号	描述	复位值	访问类型
9:0	EP_INT_EN	如果设置此位和相应的 USB 中断状态位，相应的 USB 中断路由位指示的中断线上将生成 HW 中断。	0	R/W
29:10	-	保留	0	RO
30	FRAME_INT_EN	如果设置此位和相应的 USB 中断状态位，相应的 USB 中断路由位指示的中断线上将生成 HW 中断。	0	R/W
31	DEV_INT_EN	如果设置此位和相应的 USB 中断状态位，相应的 USB 中断路由位指示的中断线上将生成 HW 中断。	0	R/W

10.6.11 USB 设置中断状态寄存器 (INTSETSTAT)

表 162. USB 设置中断状态寄存器 （INTSETSTAT，地址 0x4008 0028）位描述

位	符号	描述	复位值	访问类型
9:0	EP_SET_INT	如果软件向这些位中的一个写入 1，则会设置相应的 USB 中断状态位。 读取此寄存器时，返回与 USB 中断状态寄存器相同的值。	0	R/W
29:10	-	保留	0	RO
30	FRAME_SET_INT	如果软件向这些位中的一个写入 1，则会设置相应的 USB 中断状态位。 读取此寄存器时，返回与 USB 中断状态寄存器相同的值。	0	R/W
31	DEV_SET_INT	如果软件向这些位中的一个写入 1，则会设置相应的 USB 中断状态位。 读取此寄存器时，返回与 USB 中断状态寄存器相同的值。	0	R/W

10.6.12 USB 中断路由寄存器 (INTROUTING)

表 163. USB 中断路由寄存器 （INTROUTING，地址 0x4008 002C）位描述

位	符号	描述	复位值	访问类型
9:0	ROUTE_INT	此位可以控制将要生成中断的硬件中断线： 0：为此中断位选择 IRQ 中断线 1：为此中断位选择 FIQ 中断线	0	R/W
29:10	-	保留	0	RO
30	ROUTE_INT	此位可以控制将要生成中断的硬件中断线： 0：为此中断位选择 IRQ 中断线 1：为此中断位选择 FIQ 中断线	0	R/W
31	ROUTE_INT	此位可以控制将要生成中断的硬件中断线： 0：为此中断位选择 IRQ 中断线 1：为此中断位选择 FIQ 中断线	0	R/W

10.6.13 USB 端点切换 (EPTOGGLE)

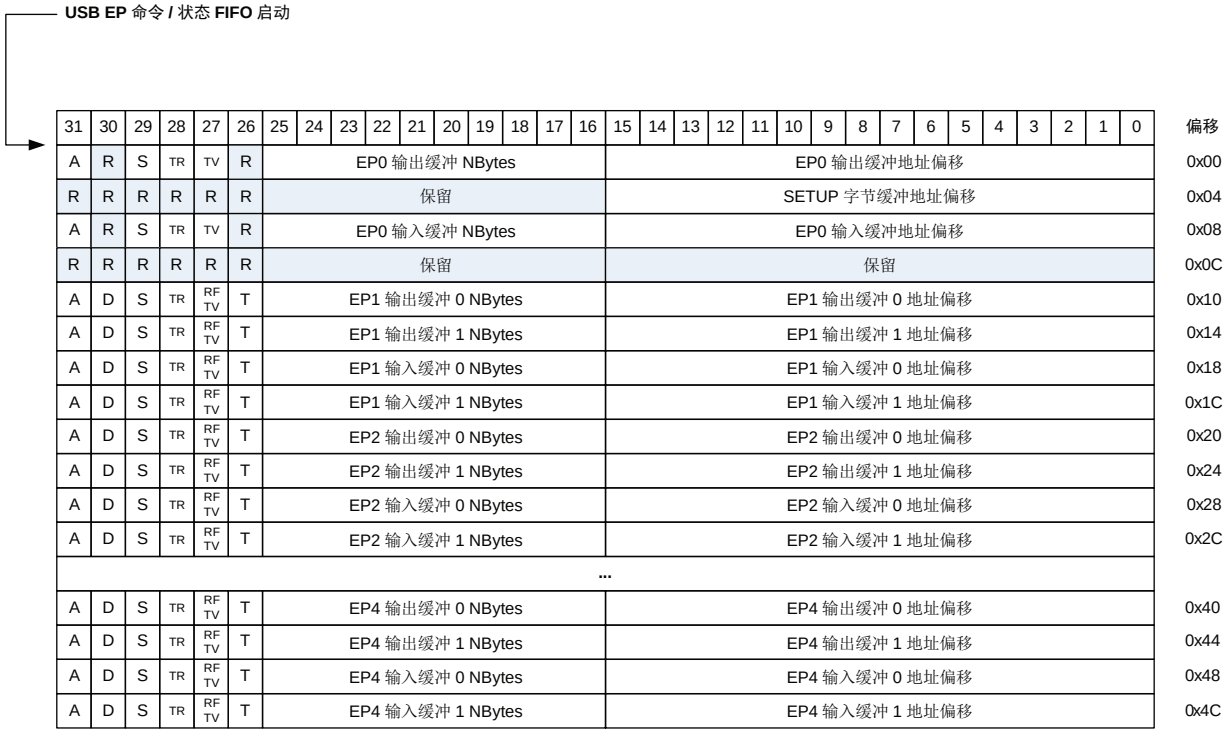
表 164. USB 端点切换 （EPTOGGLE，地址 0x4008 0034）位描述

位	符号	描述	复位值	访问类型
9:0	TOGGLE	端点数据切换：该字段表示相应端点上数据切换的当前值。	0	R
31:10	-	保留	0	R

10.7 功能说明

10.7.1 端点命令 / 状态列表

图 12 概述端点列表在存储器中的组织方式。 USB EP 命令 / 状态列表起始寄存器指向存储器中包含所有端点信息的列表的起始点。端点的顺序固定为下图所示。



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	偏移
A	R	S	TR	TV	R	EP0 输出缓冲 NBytes										EP0 输出缓冲地址偏移																	0x00
R	R	R	R	R	R	保留										SETUP 字节缓冲地址偏移																0x04
A	R	S	TR	TV	R	EP0 输入缓冲 NBytes										EP0 输入缓冲地址偏移																0x08
R	R	R	R	R	R	保留										保留																0x0C
A	D	S	TR	RF TV	T	EP1 输出缓冲 0 NBytes										EP1 输出缓冲 0 地址偏移																0x10
A	D	S	TR	RF TV	T	EP1 输出缓冲 1 NBytes										EP1 输出缓冲 1 地址偏移																0x14
A	D	S	TR	RF TV	T	EP1 输入缓冲 0 NBytes										EP1 输入缓冲 0 地址偏移																0x18
A	D	S	TR	RF TV	T	EP1 输入缓冲 1 NBytes										EP1 输入缓冲 1 地址偏移																0x1C
A	D	S	TR	RF TV	T	EP2 输出缓冲 0 NBytes										EP2 输出缓冲 0 地址偏移																0x20
A	D	S	TR	RF TV	T	EP2 输出缓冲 1 NBytes										EP2 输出缓冲 1 地址偏移																0x24
A	D	S	TR	RF TV	T	EP2 输入缓冲 0 NBytes										EP2 输入缓冲 0 地址偏移																0x28
A	D	S	TR	RF TV	T	EP2 输入缓冲 1 NBytes										EP2 输入缓冲 1 地址偏移																0x2C
...																																
A	D	S	TR	RF TV	T	EP4 输出缓冲 0 NBytes										EP4 输出缓冲 0 地址偏移																0x40
A	D	S	TR	RF TV	T	EP4 输出缓冲 1 NBytes										EP4 输出缓冲 1 地址偏移																0x44
A	D	S	TR	RF TV	T	EP4 输入缓冲 0 NBytes										EP4 输入缓冲 0 地址偏移																0x48
A	D	S	TR	RF TV	T	EP4 输入缓冲 1 NBytes										EP4 输入缓冲 1 地址偏移																0x4C

图 12. 端点命令 / 状态列表 （另请参见表 165）

表 165. 端点命令

符号	访问类型	描述
A	RW	活动 缓冲使能。HW 可以使用缓冲区存储接收到的 OUT 数据或者发送 IN 端点的数据。 软件只可以将此位设为 “1”。一旦将此位设为 1，就不允许软件更新此 32 位字中的任何值。如果软件想要停用缓冲，必须向 USB 端点跳过寄存器中的相应 “跳过” 位写入 1。硬件只可以向此位写入 0。接收一个短数据包，或者NByte字段跳变为0，或者软件已向“跳过”位写入1时，会执行此操作。
D	RW	已禁用 0：选定端点使能。 1：选定端点禁用。 如果已设置禁用位的端点接收到 USB 令牌，硬件将忽略令牌并且不会返回任何数据或信号交换。接收到总线复位时，软件必须将所有端点的禁用位设为 1。 有效位为 0 时，软件只能修改此位。
S	RW	停止 0：选定端点未停止。 1：选定端点停止。 有效位的优先级总是高于停止位。这意味着只有在有效位为 0，停止位为 1 时，才会发送停止信号交换。 有效位为 0 时，软件只能修改此位。
TR	RW	切换复位 软件将此位设为 1 时，HW 将切换值设为和 “切换值” (TV) 位中指示的值相同。 对于控制端点 0，由于在接收到设置令牌时硬件会将两个方向的端点切换复位到 1，因此不需要使用此位。 对于其他端点，端点复位时，只能将切换复位成 0。
RF / TV	RW	速率反馈模式 / 切换值 对于批量端点和等时端点，此位保留并且必须设为 0。 对于控制端点 0，该位用作切换值。设置切换复位位时，将数据切换更新到该位中编程的值。 端点用作中断端点时，可设为以下值。 0：“切换模式” 中的中断端点 1：“速率反馈模式” 中的中断端点。这意味着所有数据包的数据切换都固定为 0。 中断端点位于 “速率反馈模式” 时，必须始终将 TR 位设为 0。

表 165. 端点命令 *（续）*

符号	访问类型	描述
T	RW	端点类型 0：通用端点。端点配置为批量或中断端点 1：等时端点
NByte	RW	对于 OUT 端点，这是该缓冲区中可接收的字节数。 对于 IN 端点，这是必须发送的字节数。 每次成功传输数据包后， HW 都将按数据包的大小递减该值。 注：如果在 OUT 端点接收到短数据包，则清除有效位，并且 NByte 的值表示剩余未使用的缓冲空间。软件通过从已编程的值中减去剩余 NByte，计算接收到的字节数。
地址偏移	RW	缓冲起始地址的位 21 至 6。 如果将端点类型设为 0 （通用端点），每次成功接收 / 发送数据包时，此地址都会递增。 如果将端点类型设为 1 （等时端点），此地址不会递增。

注：接收端点 0 的 SETUP 令牌时， HW 将只读取 SETUP 字节缓冲地址偏移，以便了解存储接收到的 SETUP 字节的位置。硬件将忽略所有其他字段。如果 SETUP 阶段包含 8 个以上字节，只会将前 8 个字节写入存储器。在 SETUP 阶段， USB 兼容主机不得发送 8 个以上字节。

10.7.2 控制端点 0

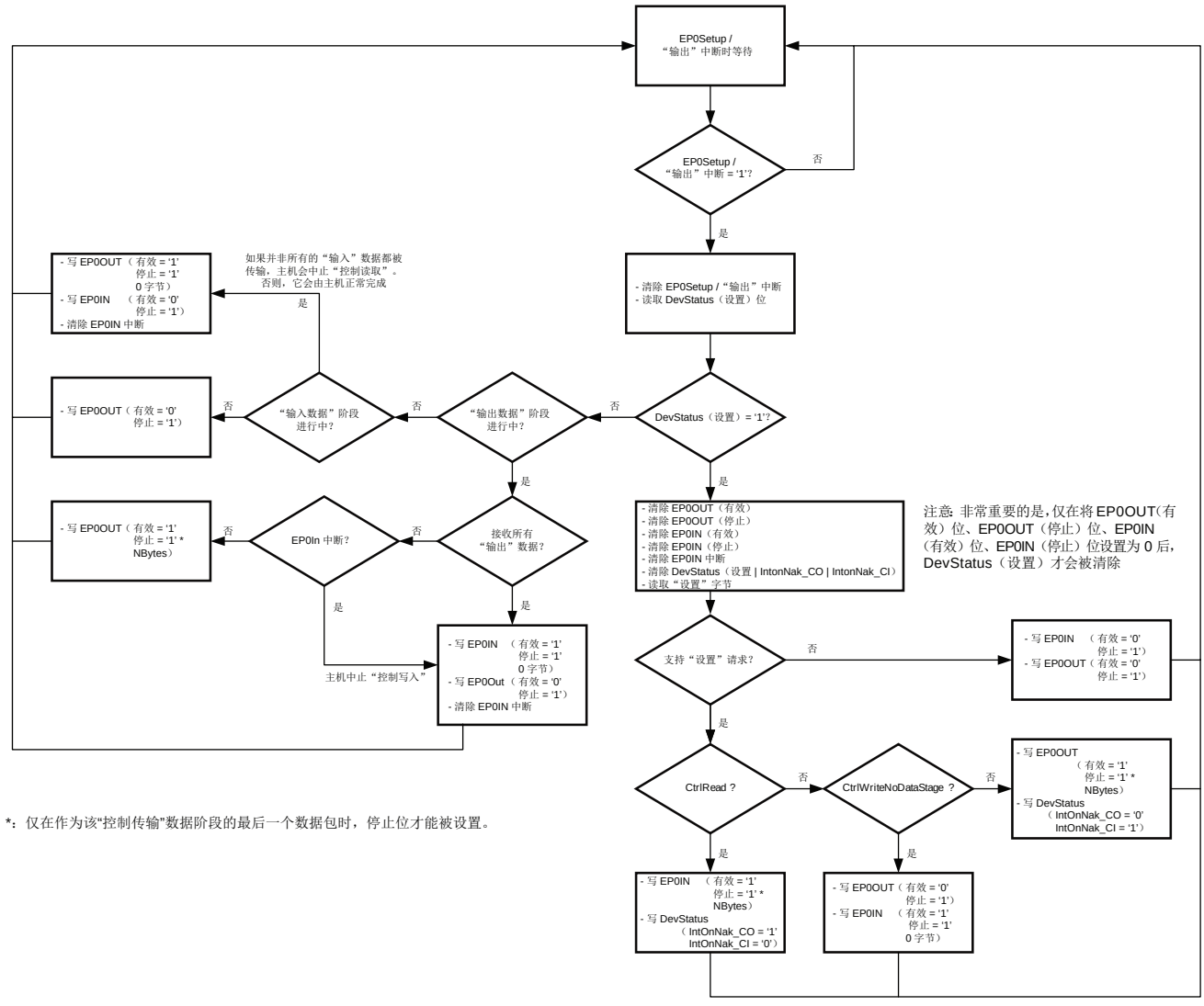
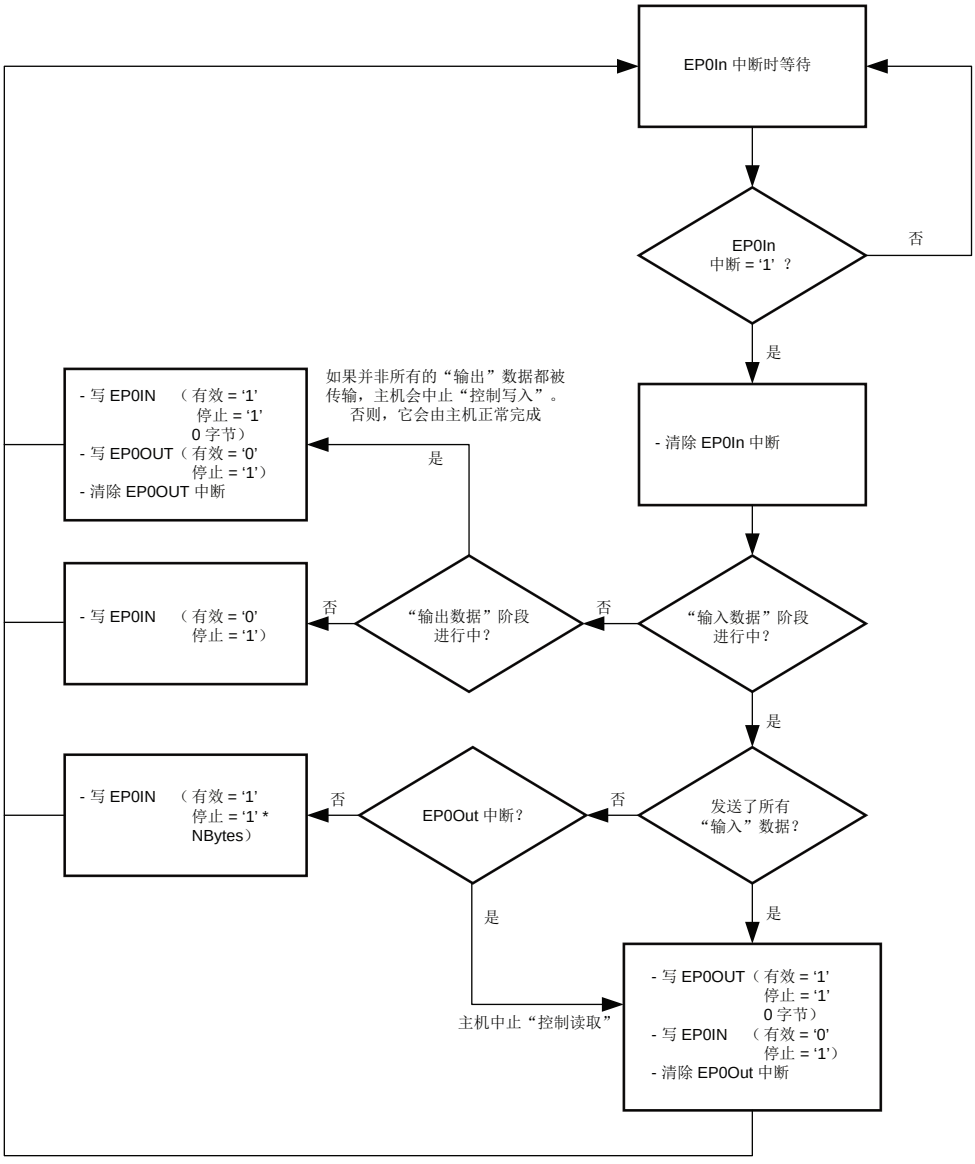


图 13. 控制端点 0 的流程图 - OUT 方向



*：仅在作为该“控制传输”数据阶段的最后一个数据包时，停止位才能被设置。

图 14. 控制端点 0 的流程图 - IN 方向

10.7.3 通用端点：单缓冲

要使能单缓冲，软件必须将相应 “USB EP 缓冲配置”位设为 0。在 “使用中的 USB EP 缓冲”寄存器上，软件可指出这种情况下使用的缓冲区。

软件想要传输数据时，要编程端点命令 / 状态条目中的不同位并设置有效位。硬件将为此端点发送 / 接收多个数据包，直到 NByte 的值等于 0。NByte 为 0 时，硬件清除有效位并设置相应的中断状态位。

软件必须等到硬件清除有效位后，才能更改一些命令 / 状态位。这将阻止硬件利用仍在缓存的一些旧值覆盖软件编程的新值。

如果软件想要在硬件处理完全部缓冲前禁用有效位，可以通过设置 USB 端点跳过寄存器中的相应端点跳过位执行此操作。

10.7.4 通用端点：双缓冲

要使能双缓冲，软件必须将相应 “USB EP 缓冲配置”位设为 1。“使用中的 USB EP 缓冲”寄存器指出接收到下一个令牌时 HW 使用的缓冲区。

HW 清除当前所用缓冲的有效位时，将关闭使用中的缓冲。软件还可通过写入 “使用中的 USB EP 缓冲”位强制 HW 使用特定缓冲。

10.7.5 特殊情况

10.7.5.1 有效位的使用

OUT 和 IN 端点上的有效位的使用有所不同。

必须在 OUT 端点接收数据时，软件会将有效位设为 1 并将 NByte 字段编程为可接收的最大字节数。

必须在 IN 端点发送数据时，软件会将有效位设为 1 并将 NByte 字段编程为必须发送的字节数。

10.7.5.2 生成 STALL 信号交换

编程端点以发送 STALL 信号交换时，必须特别小心。 STALL 信号交换只在以下情况下发送：

- 端点使能 （禁用位 = 0）
- 将端点的有效位设为 0。（该端点不需要接收 / 发送任何数据包）。
- 将端点的 STALL 位设为 1。

10.7.5.3 清除功能 （端点停止）

非控制端点返回了一个 STALL 信号交换时，主机将向该端点发送一个清除功能 （端点停止）。设备接收到该请求时，不能解除停止端点且该端点的切换位必须复位回 0。要执行此操作，软件必须为所指示的端点编程以下项目。

如果端点用于单缓冲模式，则编程以下内容：

- 设置 STALL 位 (S) 为 0。
- 设置切换复位位 (TR) 为 1，并设置切换值位 (TV) 为 0。

如果端点用于双缓冲模式，则编程以下内容：

- 将缓冲 0 和缓冲 1 的 STALL 位均设为 0。
- 读取该端点的 “使用中的缓冲”位。
- 对于 “使用中的缓冲”位指示的缓冲区，设置切换复位位 (TR) 为 1，并设置切换值位 (TV) 为 0。

10.7.5.4 设置配置

接收到配置值不为零的 SetConfiguration 请求时，设备软件必须使能将用于此配置的所有端点并复位所有切换值。要执行此操作，必须为将用于此配置的所有端点生成第 10.7.5.3 节中说明的流程。
对于不会用于此配置的所有端点，必须将禁用位 (D) 设为 1。

10.7.6 USB 唤醒

10.7.6.1 从 USB 活动上的深度睡眠和掉电模式中唤醒

要允许 LPC1315/16/17/45/46/47 从 USB 活动上的深度睡眠或掉电模式唤醒，需要完成以下步骤：

1. 将 USBCLKCTRL 寄存器 （表 36）中的位 AP_CLK 设为 0 （默认），以使能 USB need_clock 信号的自动控制。
2. 轮询 DSVCMD_STAT 寄存器中的 DSUS 位 (DSUS = 1)，等到 USB 活动挂起。
3. 再过 2 ms 后， USB need_clock 信号将被取消。轮询 USBCLKST 寄存器，直到 USB need_clock 状态位为 0 （表 37）。
4. 一旦 USBCLKST 寄存器返回 0，使能 NVIC (# 30) 中的 USB 活动唤醒中断并将其清除。
5. 将 USBCLKCTRL 寄存器的位 1 设为 1，触发 USB need_clock 信号上升沿的 USB 活动唤醒中断。
6. 通过使能 STARTERP1 寄存器（表 39，位 19）的 USB need_clock 信号，在此中断上使能从深度睡眠或掉电模式唤醒。
7. 通过写入 PCON 寄存器，进入深度睡眠或掉电模式。
8. 执行 WFI 指令。

LPC1315/16/17/45/46/47 将自动唤醒和恢复执行 USB 活动。

10.7.6.2 远程唤醒

要在 USB 活动挂起时发出远程唤醒，需完成以下步骤：

1. 将USBCLKCTRL寄存器中的位AP_CLK设为0（表36，默认），以使能USB need_clock信号的自动控制。
2. 要发出远程唤醒时，打开 USB 时钟并使能 USB 时钟源。
3. 通过向 USBCLKCTRL 寄存器的位 AP_CLK（表 36，位 0）写入 1，强制 USB 时钟开启。
4. 向 DSVCMD_STAT 寄存器的 DSUS 位写入 0。
5. 轮询 DSVCMD_STAT 寄存器的 DSUS 位 (DSUS = 0)，等到 USB 退出挂起状态。
6. 清除 USBCLKCTRL 寄存器的 AP_CLK 位（表 36，位 0），以使能自动 USB 时钟控制。

11.1 本章导读

LPC1345/46/47 器件上可以使用 USB 片上驱动器。

11.2 简介

Boot ROM 包含一个用于简化 USB 应用开发的 USB 驱动器。该 USB 驱动器实施通信设备类 (CDC)、人机接口设备 (HID) 和海量存储设备 (MSC) 设备类。USB 片上驱动器支持复合设备。

11.3 USB 驱动器函数

USB 设备驱动器 ROM API 包含以下模块：

- 通信设备类 (CDC) 函数驱动器
  - 通信设备类函数驱动器初始化参数数据结构 （表 193 “USBD_CDC_INIT_PARAM 类结构”）。
  - CDC 类 API 函数结构。该模块公开直接与 USB 设备控制器硬件交互的函数（表 192 “USBD_CDC_API 类结构”）。
- USB 内核层
  - struct （表 189 “_WB_T 类结构”）
  - union （表 166 “__WORD_BYTE 类结构”）
  - struct （表 167 “_BM_T 类结构”）
  - struct （表 180 “_REQUEST_TYPE 类结构”）
  - struct （表 187 “_USB_SETUP_PACKET 类结构”）
  - struct （表 183 “_USB_DEVICE_QUALIFIER_DESCRIPTOR 类结构”）
  - struct USB 设备描述符
  - struct （表 183 “_USB_DEVICE_QUALIFIER_DESCRIPTOR 类结构”）
  - struct USB 配置描述符
  - struct （表 185 “_USB_INTERFACE_DESCRIPTOR 类结构”）
  - struct USB 端点描述符
  - struct （表 188 “_USB_STRING_DESCRIPTOR 类结构”）
  - struct （表 181 “_USB_COMMON_DESCRIPTOR 类结构”）
  - struct （表 186 “_USB_OTHER_SPEED_CONFIGURATION 类结构”）
  - USB 描述符数据结构 （表 182 “_USB_CORE_DESCS_T 类结构”）
  - USB 设备协议栈初始化参数数据结构 （表 191 “USBD_API_INIT_PARAM 类结构”）。
  - USB 设备协议栈内核 API 函数结构 （表 194 “USBD_CORE_API 类结构”）。

- 设备固件升级 (DFU) 类函数驱动器
  - DFU 描述符数据结构 （表 196 “USBD_DFU_INIT_PARAM 类结构”）。
  - DFU 类 API 函数结构。该模块显示直接与 USB 设备控制器硬件交互的函数（表 195 “USBD_DFU_API 类结构”）。
- HID 类函数驱动器
  - struct （表 175 “ _HID_DESCRIPTOR 类结构”）。
  - struct （表 177 “ _HID_REPORT_T 类结构”）。
  - USB 描述符数据结构 （表 198 “USBD_HID_INIT_PARAM 类结构”）。
  - HID 类 API 函数结构。该结构包含指向 HID 函数驱动器模块所公开的所有功能的指针 （表 199 “USBD_HW_API 类结构”）。
- USB 设备控制器驱动器
  - 硬件 API 函数结构。该模块显示直接与 USB 设备控制器硬件交互的函数（表 199 “USBD_HW_API 类结构”）。
- 海量存储设备类 (MSC) 函数驱动器
  - 海量存储设备类函数驱动器初始化参数数据结构 （表 201）。
  - MSC 类 API 函数结构。该模块显示直接与 USB 设备控制器硬件交互的函数（表 200）。

11.4 调用 USB 设备驱动器

ROM 中的一个固定位置（即， 0x1FFF 1FF8）包含指向 ROM 驱动器表的指针。 ROM 驱动器表包含一个指向 USB 驱动器表的指针。指向各种 USB 驱动器函数的指针都存储在该表中。可使用 C 结构调用 USB 驱动器函数。图 15 说明了用于访问片上 USB 驱动器的指针机制。

```
typedef struct USBD_API

{

const USBD_HW_API_T* hw;

const USBD_CORE_API_T* core;

const USBD_MSC_API_T* msc;

const USBD_DFU_API_T* dfu;

const USBD_HID_API_T* hid;

const USBD_CDC_API_T* cdc;

const uint32_t* reserved6;

const uint32_t version;

} USBD_API_T;
```

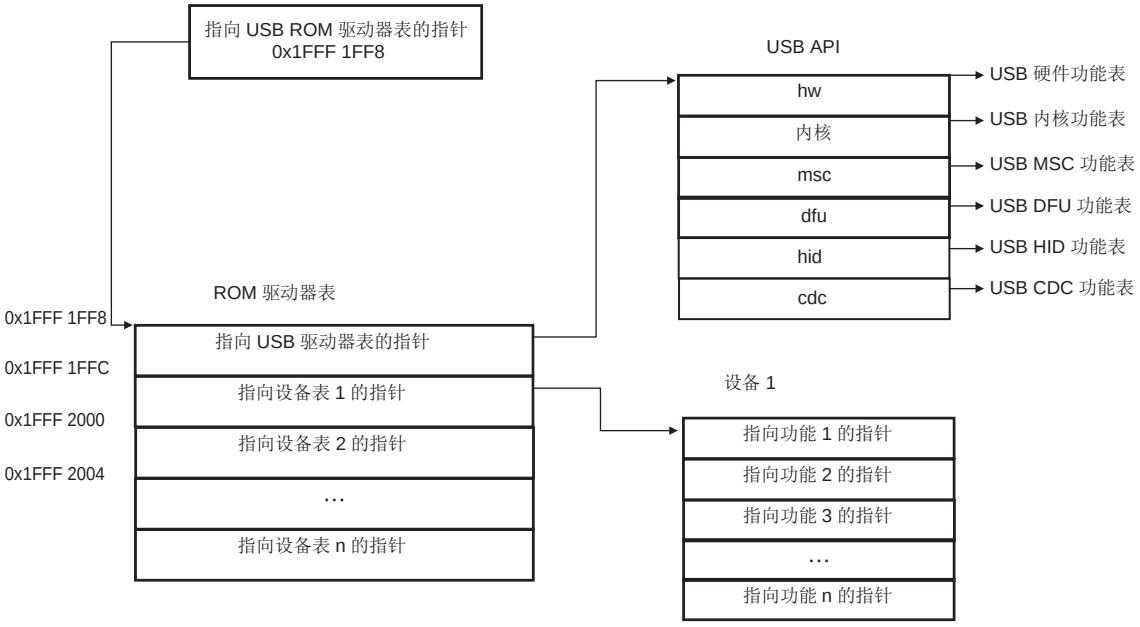


图 15. USB 设备驱动器指针结构

11.5 USB API

11.5.1 __WORD_BYTE

表 166. __WORD_BYTE 类结构

成员	描述
W	`uint16_tuint16_t __WORD_BYTE::W` 进行 16 位访问的数据成员
WB	`WB_TWB_T __WORD_BYTE::WB` 进行 8 位访问的数据成员

11.5.2 _BM_T

表 167. _BM_T 类结构

成员	描述
Recipient	`uint8_tuint8_t _BM_T::Recipient` 收件人类型。
Type	`uint8_tuint8_t _BM_T::Type` 请求类型。
Dir	`uint8_tuint8_t _BM_T::Dir` 方向类型。

11.5.3 _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR

表 168. _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR 类结构

成员	描述
bFunctionLength	uint8_tuint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bFunctionLength
bDescriptorType	uint8_tuint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bDescriptorType
bDescriptorSubtype	uint8_tuint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bDescriptorSubtype
bmCapabilities	uint8_tuint8_t _CDC_ABSTRACT_CONTROL_MANAGEMENT_DESCRIPTOR::bmCapabilities

11.5.4 _CDC_CALL_MANAGEMENT_DESCRIPTOR

表 169. _CDC_CALL_MANAGEMENT_DESCRIPTOR 类结构

成员	描述
bFunctionLength	uint8_tuint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bFunctionLength
bDescriptorType	uint8_tuint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bDescriptorType
bDescriptorSubtype	uint8_tuint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bDescriptorSubtype
bmCapabilities	uint8_tuint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bmCapabilities
bDataInterface	uint8_tuint8_t _CDC_CALL_MANAGEMENT_DESCRIPTOR::bDataInterface

11.5.5 _CDC_HEADER_DESCRIPTOR

表 170. _CDC_HEADER_DESCRIPTOR 类结构

成员	描述
bFunctionLength	uint8_tuint8_t _CDC_HEADER_DESCRIPTOR::bFunctionLength
bDescriptorType	uint8_tuint8_t _CDC_HEADER_DESCRIPTOR::bDescriptorType
bDescriptorSubtype	uint8_tuint8_t _CDC_HEADER_DESCRIPTOR::bDescriptorSubtype
bcdCDC	uint16_tuint16_t _CDC_HEADER_DESCRIPTOR::bcdCDC

11.5.6 _CDC_LINE_CODING

表 171. _CDC_LINE_CODING 类结构

成员	描述
dwDTERate	uint32_tuint32_t _CDC_LINE_CODING::dwDTERate
bCharFormat	uint8_tuint8_t _CDC_LINE_CODING::bCharFormat
bParityType	uint8_tuint8_t _CDC_LINE_CODING::bParityType
bDataBits	uint8_tuint8_t _CDC_LINE_CODING::bDataBits

11.5.7 _CDC_UNION_1SLAVE_DESCRIPTOR

表 172. _CDC_UNION_1SLAVE_DESCRIPTOR 类结构

成员	描述
sUnion	CDC_UNION_DESCRIPTORCDC_UNION_DESCRIPTOR _CDC_UNION_1SLAVE_DESCRIPTOR::sUnion
bSlaveInterfaces	uint8_tuint8_t _CDC_UNION_1SLAVE_DESCRIPTOR::bSlaveInterfaces[1][1]

11.5.8 _CDC_UNION_DESCRIPTOR

表 173. _CDC_UNION_DESCRIPTOR 类结构

成员	描述
bFunctionLength	uint8_tuint8_t _CDC_UNION_DESCRIPTOR::bFunctionLength
bDescriptorType	uint8_tuint8_t _CDC_UNION_DESCRIPTOR::bDescriptorType
bDescriptorSubtype	uint8_tuint8_t _CDC_UNION_DESCRIPTOR::bDescriptorSubtype
bMasterInterface	uint8_tuint8_t _CDC_UNION_DESCRIPTOR::bMasterInterface

11.5.9 _DFU_STATUS

表 174. _DFU_STATUS 类结构

成员	描述
bStatus	uint8_tuint8_t _DFU_STATUS::bStatus
bwPollTimeout	uint8_tuint8_t _DFU_STATUS::bwPollTimeout[3][3]
bState	uint8_tuint8_t _DFU_STATUS::bState
iString	uint8_tuint8_t _DFU_STATUS::iString

11.5.10 _HID_DESCRIPTOR

HID 特定类 HID 描述符。

表 175. _HID_DESCRIPTOR 类结构

成员	描述
bLength	uint8_tuint8_t _HID_DESCRIPTOR::bLength 描述符的大小（字节）。
bDescriptorType	uint8_tuint8_t _HID_DESCRIPTOR::bDescriptorType HID 描述符的类型。
bcdHID	uint16_tuint16_t _HID_DESCRIPTOR::bcdHID HID 描述符和设备符合的 BCD 编码版。

表 175. _HID_DESCRIPTOR 类结构 *（续）*

成员	描述
bCountryCode	uint8_tuint8_t _HID_DESCRIPTOR::bCountryCode 本地化设备的国家 / 地区代码，如果通用，则为 0。
bNumDescriptors	uint8_tuint8_t _HID_DESCRIPTOR::bNumDescriptors 接口的 HID 报告描述符总数。
DescriptorList	PRE_PACK struct POST_PACK _HID_DESCRIPTOR::_HID_DESCRIPTOR_LISTPRE_PACK struct POST_PACK _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST _HID_DESCRIPTOR::DescriptorList[1][1] 一个或多个描述符的数组

11.5.11 _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST

表 176. _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST 类结构

成员	描述
bDescriptorType	uint8_tuint8_t _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST::bDescriptorType HID 报告的类型。
wDescriptorLength	uint16_tuint16_t _HID_DESCRIPTOR::_HID_DESCRIPTOR_LIST::wDescriptorLength 相关 HID 报告描述符的长度 （字节）。

11.5.12 _HID_REPORT_T

HID 报告描述符数据结构。

表 177. _HID_REPORT_T 类结构

成员	描述
len	uint16_tuint16_t _HID_REPORT_T::len 报告描述符的大小 （字节）。
idle_time	uint8_tuint8_t _HID_REPORT_T::idle_time 协议栈使用该值响应指定报告 ID 的 Set_Idle 和 GET_Idle 请求。该字段的值指定了为指定报告 ID 生成两个报告的速率。例如，带两个输入报告的设备可指定报告 ID 1 的闲置率为 20 毫秒，报告 ID 2 的闲置率为 500 毫秒。
__pad	uint8_tuint8_t _HID_REPORT_T::__pad 填充空间。
desc	uint8_t *uint8_t* _HID_REPORT_T::desc 报告描述符。

11.5.13 _MSC_CBW

表 178. _MSC_CBW 类结构

成员	描述
dSignature	uint32_tuint32_t _MSC_CBW::dSignature
dTag	uint32_tuint32_t _MSC_CBW::dTag
dDataLength	uint32_tuint32_t _MSC_CBW::dDataLength
bmFlags	uint8_tuint8_t _MSC_CBW::bmFlags
bLUN	uint8_tuint8_t _MSC_CBW::bLUN
bCBLength	uint8_tuint8_t _MSC_CBW::bCBLength
CB	uint8_tuint8_t _MSC_CBW::CB[16][16]

11.5.14 _MSC_CSW

表 179. _MSC_CSW 类结构

成员	描述
dSignature	uint32_tuint32_t _MSC_CSW::dSignature
dTag	uint32_tuint32_t _MSC_CSW::dTag
dDataResidue	uint32_tuint32_t _MSC_CSW::dDataResidue
bStatus	uint8_tuint8_t _MSC_CSW::bStatus

11.5.15 _REQUEST_TYPE

表 180. _REQUEST_TYPE 类结构

成员	描述
B	uint8_tuint8_t _REQUEST_TYPE::B 字节宽访问成员
BM	BM_TBM_T _REQUEST_TYPE::BM 位字段结构访问成员

11.5.16 _USB_COMMON_DESCRIPTOR

表 181. _USB_COMMON_DESCRIPTOR 类结构

成员	描述
bLength	uint8_tuint8_t _USB_COMMON_DESCRIPTOR::bLength 该描述符的大小（字节）
bDescriptorType	uint8_tuint8_t _USB_COMMON_DESCRIPTOR::bDescriptorType 描述符类型

11.5.17 _USB_CORE_DESCS_T

USB 描述符数据结构。

表 182. _USB_CORE_DESCS_T 类结构

成员	描述
device_desc	uint8_t *uint8_t* _USB_CORE_DESCS_T::device_desc 指向 USB 设备描述符的指针
string_desc	uint8_t *uint8_t* _USB_CORE_DESCS_T::string_desc 指向 USB 字符串描述符数组的指针
full_speed_desc	uint8_t *uint8_t* _USB_CORE_DESCS_T::full_speed_desc 设备以全速模式运行时，指向 USB 设备配置描述符的指针。
high_speed_desc	uint8_t *uint8_t* _USB_CORE_DESCS_T::high_speed_desc 设备以高速模式运行时，指向 USB 设备配置描述符的指针。对于仅限于全速的实施，该指针应和 full_speed_desc 相同。
device_qualifier	uint8_t *uint8_t* _USB_CORE_DESCS_T::device_qualifier 指向 USB 设备限定符描述符的指针。对于仅限于全速的实施，应将该指针设置为空 (0)。

11.5.18 _USB_DEVICE_QUALIFIER_DESCRIPTOR

表 183. _USB_DEVICE_QUALIFIER_DESCRIPTOR 类结构

成员	描述
bLength	uint8_tuint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bLength 描述符大小
bDescriptorType	uint8_tuint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDescriptorType 设备限定符类型
bcdUSB	uint16_tuint16_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bcdUSB USB 规范版本号 （如 V2.00 为 0200H）
bDeviceClass	uint8_tuint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDeviceClass 类代码
bDeviceSubClass	uint8_tuint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDeviceSubClass 子类代码
bDeviceProtocol	uint8_tuint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bDeviceProtocol 协议代码
bMaxPacketSize0	uint8_tuint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bMaxPacketSize0 其他速度下的最大数据包大小
bNumConfigurations	uint8_tuint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bNumConfigurations 其他速度配置数
bReserved	uint8_tuint8_t _USB_DEVICE_QUALIFIER_DESCRIPTOR::bReserved 留待将来使用，必须为 0

11.5.19 _USB_DFU_FUNC_DESCRIPTOR

表 184. _USB_DFU_FUNC_DESCRIPTOR 类结构

成员	描述
bLength	uint8_tuint8_t _USB_DFU_FUNC_DESCRIPTOR::bLength
bDescriptorType	uint8_tuint8_t _USB_DFU_FUNC_DESCRIPTOR::bDescriptorType
bmAttributes	uint8_tuint8_t _USB_DFU_FUNC_DESCRIPTOR::bmAttributes
wDetachTimeOut	uint16_tuint16_t _USB_DFU_FUNC_DESCRIPTOR::wDetachTimeOut
wTransferSize	uint16_tuint16_t _USB_DFU_FUNC_DESCRIPTOR::wTransferSize
bcdDFUVersion	uint16_tuint16_t _USB_DFU_FUNC_DESCRIPTOR::bcdDFUVersion

11.5.20 _USB_INTERFACE_DESCRIPTOR

表 185. _USB_INTERFACE_DESCRIPTOR 类结构

成员	描述
bLength	uint8_tuint8_t _USB_INTERFACE_DESCRIPTOR::bLength 该描述符的大小 （字节）
bDescriptorType	uint8_tuint8_t _USB_INTERFACE_DESCRIPTOR::bDescriptorType INTERFACE 描述符类型
bInterfaceNumber	uint8_tuint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceNumber 该接口的编号。以零为基准的值，用于标识该配置所支持的并行接口数组的索引。
bAlternateSetting	uint8_tuint8_t _USB_INTERFACE_DESCRIPTOR::bAlternateSetting 用于为在先前字段标识的接口选择备用设置的值
bNumEndpoints	uint8_tuint8_t _USB_INTERFACE_DESCRIPTOR::bNumEndpoints 该接口使用的端点数量 （不包括端点 0）。如果该值为 0，则该接口仅使用默认控制管道。
bInterfaceClass	uint8_tuint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceClass 类代码 （由 USB-IF 分配）。
bInterfaceSubClass	uint8_tuint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceSubClass 子类代码 （由 USB-IF 分配）。
bInterfaceProtocol	uint8_tuint8_t _USB_INTERFACE_DESCRIPTOR::bInterfaceProtocol 协议代码 （由 USB 分配）。
iInterface	uint8_tuint8_t _USB_INTERFACE_DESCRIPTOR::iInterface 描述该接口的字符串描述符的索引

11.5.21 _USB_OTHER_SPEED_CONFIGURATION

表 186. _USB_OTHER_SPEED_CONFIGURATION 类结构

成员	描述
bLength	uint8_tuint8_t _USB_OTHER_SPEED_CONFIGURATION::bLength 描述符大小
bDescriptorType	uint8_tuint8_t _USB_OTHER_SPEED_CONFIGURATION::bDescriptorType Other_speed_Configuration 类型
wTotalLength	uint16_tuint16_t _USB_OTHER_SPEED_CONFIGURATION::wTotalLength 返回数据的总长度
bNumInterfaces	uint8_tuint8_t _USB_OTHER_SPEED_CONFIGURATION::bNumInterfaces 该速度配置支持的接口数量
bConfigurationValue	uint8_tuint8_t _USB_OTHER_SPEED_CONFIGURATION::bConfigurationValue 用于选择配置的值
IConfiguration	uint8_tuint8_t _USB_OTHER_SPEED_CONFIGURATION::IConfiguration 字符串描述符索引
bmAttributes	uint8_tuint8_t _USB_OTHER_SPEED_CONFIGURATION::bmAttributes 与配置描述符相同
bMaxPower	uint8_tuint8_t _USB_OTHER_SPEED_CONFIGURATION::bMaxPower 与配置描述符相同

11.5.22 _USB_SETUP_PACKET

表 187. _USB_SETUP_PACKET 类结构

成员	描述
bmRequestType	REQUEST_TYPEREQUEST_TYPE _USB_SETUP_PACKET::bmRequestType 该位映射字段可标识特定请求的特性。 _BM_T。
bRequest	uint8_tuint8_t _USB_SETUP_PACKET::bRequest 该字段可指定特定请求。 bmRequestType 字段的类型位可修改该字段的含义。 USBD_REQUEST。
wValue	WORD_BYTEWORD_BYTE _USB_SETUP_PACKET::wValue 用于将参数传递给设备，特定于请求。
wIndex	WORD_BYTEWORD_BYTE _USB_SETUP_PACKET::wIndex 用于将参数传递给设备，特定于请求。 wIndex 字段常在请求中用于指定端点或接口。
wLength	uint16_tuint16_t _USB_SETUP_PACKET::wLength 该字段指定在控制传输的第二个阶段传输的数据长度。

11.5.23 _USB_STRING_DESCRIPTOR

表 188. _USB_STRING_DESCRIPTOR 类结构

成员	描述
bLength	uint8_tuint8_t _USB_STRING_DESCRIPTOR::bLength 该描述符的大小（字节）
bDescriptorType	uint8_tuint8_t _USB_STRING_DESCRIPTOR::bDescriptorType STRING 描述符类型
bString	uint16_tuint16_t _USB_STRING_DESCRIPTOR::bString UNICODE 编码的字符串

11.5.24 _WB_T

表 189. _WB_T 类结构

成员	描述
L	uint8_tuint8_t _WB_T::L 低字节
H	uint8_tuint8_t _WB_T::H 高字节

11.5.25 USBD_API

主要的 USBD API 函数结构。该结构包含指向各种 USB 设备协议栈子模块函数表的指针。该结构作为主入口点，用于访问基于 ROM 的 USB 设备协议栈公开的各种方法（按子模块分组）。

表 190. USBD_API 类结构

成员	描述
hw	const USBD_HW_API_T *const USBD_HW_API_T* USBD_API::hw 指向函数表的指针，该函数表公开直接与 USB 设备协议栈内核层交互的函数。
core	const USBD_CORE_API_T *const USBD_CORE_API_T* USBD_API::core 指向函数表的指针，该函数表公开直接与 USB 设备控制器硬件交互的函数。
msc	const USBD_MSC_API_T *const USBD_MSC_API_T* USBD_API::msc 指向函数表的指针，该函数表公开 MSC 函数驱动器模块提供的函数。
dfu	const USBD_DFU_API_T *const USBD_DFU_API_T* USBD_API::dfu 指向函数表的指针，该函数表公开 DFU 函数驱动器模块提供的函数。
hid	const USBD_HID_API_T *const USBD_HID_API_T* USBD_API::hid 指向函数表的指针，该函数表公开 HID 函数驱动器模块提供的函数。

表 190. USBD_API 类结构 *（续）*

成员	描述
cdc	`const USBD_CDC_API_T *const USBD_CDC_API_T* USBD_API::cdc` 指向函数表的指针，该函数表公开 CDC-ACM 函数驱动器模块提供的函数。
reserved6	`const uint32_t *const uint32_t* USBD_API::reserved6` 保留用于未来的函数驱动器模块。
version	`const uint32_tconst uint32_t USBD_API::version` USB ROM 协议栈的版本标识符。版本定义为 0x0CHDMhCC，其中每个半字节代表相应元件的版本号。CC - 7:0 - 8 位内核版本号 h - 11:8 - 4 位硬件接口版本号 M - 15:12 - 4 位 MSC 类模块版本号 D - 19:16 - 4 位 DFU 类模块版本号 H - 23:20 - 4 位 HID 类模块版本号 C - 27:24 - 4 位 CDC 类模块版本号 H - 31:28 - 4 位保留

11.5.26 USBD_API_INIT_PARAM

USB 设备协议栈初始化参数数据结构。

表 191. USBD_API_INIT_PARAM 类结构

成员	描述
usb_reg_base	`uint32_tuint32_t USBD_API_INIT_PARAM::usb_reg_base` USB 设备控制器的寄存器基址。
mem_base	`uint32_tuint32_t USBD_API_INIT_PARAM::mem_base` 协议栈从中分配数据和缓冲的基本存储器位置。 **注：**此字段中设置的存储器地址应可由 USB DMA 控制器访问。该值还应在 2048 字节边界对齐。
mem_size	`uint32_tuint32_t USBD_API_INIT_PARAM::mem_size` 协议栈可使用的存储器缓冲区大小。 **注：**mem_size 应大于 USBD_HW_API::GetMemSize() 例程返回的大小。
max_num_ep	`uint8_tuint8_t USBD_API_INIT_PARAM::max_num_ep` USB 设备控制器实例支持的最大端点数 （由 指定
pad0	`uint8_tuint8_t USBD_API_INIT_PARAM::pad0[3][3]`
USB_Reset_Event	`USB_CB_TUSB_CB_T USBD_API_INIT_PARAM::USB_Reset_Event` 用于 USB 接口复位的事件。当 USB 主机请求设备复位其接口时，触发该事件。库自动配置控制端点后，触发该事件。 **注：**该事件从 USB_ISR 上下文调用，因此时间非常关键。如果此回调中具有延迟，将使设备无法正确枚举或正常工作。
USB_Suspend_Event	`USB_CB_TUSB_CB_T USBD_API_INIT_PARAM::USB_Suspend_Event` 用于 USB 挂起的事件。当 USB 主机通过中止将帧起始脉冲发送到设备的操作而挂起设备时，触发该事件。通常情况下，挂接该事件的目的是为了将设备移动到低功耗状态，直到主机唤醒设备为止。 **注：**该事件从 USB_ISR 上下文调用，因此时间非常关键。如果此回调中具有延迟，会引发其他系统问题。
USB_Resume_Event	`USB_CB_TUSB_CB_T USBD_API_INIT_PARAM::USB_Resume_Event` 用于 USB 唤醒或恢复的事件。当 USB 设备接口挂起，且主机通过提供帧起始脉冲唤醒设备时，触发该事件。通常情况下，挂接该事件的目的是为了让用户应用脱离低功耗状态，并返回到正常操作模式。 **注：**该事件从 USB_ISR 上下文调用，因此时间非常关键。如果此回调中具有延迟，会引发其他系统问题。
reserved_sbz	`USB_CB_TUSB_CB_T USBD_API_INIT_PARAM::reserved_sbz` 保留参数应设为 0。

表 191. USBD_API_INIT_PARAM 类结构 *（续）*

成员	描述
USB_SOF_Event	`USB_CB_TUSB_CB_T USBD_API_INIT_PARAM::USB_SOF_Event` 用于 USB 帧起始检测的事件（使能时）。每个 USB 帧起始时触发该事件（全速模式下，每毫秒一次；高速模式下，每 125 毫秒一次），且和 USB 总线同步。 该事件的处理时间要求严格，每毫秒运行一次（全速模式下），因此，长处理器将明显降低设备性能。仅应在需要时使能该事件以减少设备唤醒次数。 通常情况下，该事件未激活 - 必须手动使能并通过 USB 中断寄存器禁用。 **注：**通常情况下，该事件未激活 - 必须手动使能并通过 USB 中断寄存器禁用。
USB_WakeUpCfg	`USB_PARAM_CB_TUSB_PARAM_CB_T USBD_API_INIT_PARAM::USB_WakeUpCfg` 用于远程唤醒配置的事件（使能时）。当 USB 主机请求设备为其自己配置远程唤醒功能时，触发该事件。进入低功率状态前，USB 主机将该请求发送至设备（在设备描述符中报告远程唤醒功能）。如果设备拥有可触发远程唤醒事件的特殊板上电路，应用层应实施该回调。此外，应用还可使用本回调区分后续挂起事件是由电缆拔出还是主机挂起请求引起的。仅在收到此回调，且主机使能远程唤醒功能后，设备才可唤醒主机。要发送远程唤醒信号，设备必须通过调用 usapi.hw->WakeUp() 例程在主机上生成恢复信号。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. param1 = 为 0 时 - 清除唤醒配置，为 1 时 - 使能唤醒配置。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。
USB_Power_Event	`USB_PARAM_CB_TUSB_PARAM_CB_T USBD_API_INIT_PARAM::USB_Power_Event` 保留参数应设为 0。
USB_Error_Event	`USB_PARAM_CB_TUSB_PARAM_CB_T USBD_API_INIT_PARAM:USB_Error_Event` 用于错误条件的事件。当 USB 设备控制器检测到系统内的错误条件时，触发该事件。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. param1 = USB 设备中断状态寄存器。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。
USB_Configure_Event	`USB_CB_TUSB_CB_T USBD_API_INIT_PARAM::USB_Configure_Event` 用于已更改的 USB 配置编号的事件。当 USB 主机更改选定配置编号时，触发该事件。从主机接收到配置更改请求后，协议栈在调用该回调函数前，使能 / 配置新配置所需的端点。 **注：**该事件从 USB_ISR 上下文调用，因此时间非常关键。如果此回调中具有延迟，将使设备无法正确枚举或正常工作。
USB_Interface_Event	`USB_CB_TUSB_CB_T USBD_API_INIT_PARAM::USB_Interface_Event` 用于已更改的 USB 接口设置的事件。当 USB 主机将接口设置更改为备用接口设置之一时，触发该事件。从主机接收到接口更改请求后，协议栈在调用该回调函数前，使能 / 配置新备用接口设置所需的端点。 **注：**该事件从 USB_ISR 上下文调用，因此时间非常关键。如果此回调中具有延迟，将使设备无法正确枚举或正常工作。

表 191. USBD_API_INIT_PARAM 类结构 *（续）*

成员	描述
USB_Feature_Event	`USB_CB_TUSB_CB_T USBD_API_INIT_PARAM::USB_Feature_Event` 用于已更改的 USB 功能的事件。当 USB 主机发送设置 / 清除功能请求时，触发该事件。协议栈仅处理针对 USB_FEATURE_REMOTE_WAKEUP、USB_FEATURE_TEST_MODE 和 USB_FEATURE_ENDPOINT_STALL 功能的此类请求。从主机接收到功能请求后，协议栈将相应地处理请求，然后调用此回调函数。 **注：**该事件从 USB_ISR 上下文调用，因此时间非常关键。如果此回调中具有延迟，将使设备无法正确枚举或正常工作。
virt_to_phys	`uint32_t(*uint32_t(*  USBD_API_INIT_PARAM::virt_to_phys)(void *vaddr))(void *vaddr)` 留待将来使用的参数。应设为 0。
cache_flush	`void(*void(*  USBD_API_INIT_PARAM::cache_flush)(uint32_t *start_adr, uint32_t *end_adr))(uint32_t *start_adr, uint32_t *end_adr)` 留待将来使用的参数。应设为 0。

11.5.27 USBD_CDC_API

CDC 类 API 函数结构。该模块公开直接与 USB 设备控制器硬件交互的函数。

表 192. USBD_CDC_API 类结构

成员	描述
GetMemSize	`uint32_t(*uint32_t USBD_CDC_API::GetMemSize)(USBD_CDC_INIT_PARAM_T *param)` 用于确定 CDC 函数驱动器模块所要求的存储器的函数。 在调用 pUsbApi->CDC->Init() 前，应用层调用该函数以分配 CDC 函数驱动器模块所使用的存储器。应用程序应分配可由 USB 控制器 /DMA 控制器访问的存储器。 **注：**一些存储器区域无法由所有总线主机访问。 参数： 1. param = 含 CDC 函数驱动器模块初始化参数的结构。 返回： 返回需要的存储器大小 （字节）。
init	`ErrorCode_t(*ErrorCode_t USBD_CDC_API::init)(USBD_HANDLE_T hUsb, USBD_CDC_INIT_PARAM_T *param, USBD_HANDLE_T *phCDC)` 初始化 CDC 函数驱动器模块的函数。 应用层调用该函数以初始化 CDC 函数驱动器模块。 hUsb：USB 设备协议栈的句柄。 param：含 CDC 函数驱动器模块初始化参数的结构。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. param = 含 CDC 函数驱动器模块初始化参数的结构。 返回： 返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功 2. ERR_USBD_BAD_MEM_BUF = 传递的存储器缓冲区不以 4 字节对齐或小于要求。 3. ERR_API_INVALID_PARAM2 = CDC_Write()、CDC_Read() 或 CDC_Verify() 回调未定义。 4. ERR_USBD_BAD_INTF_DESC = 传递错误的接口描述符。 5. ERR_USBD_BAD_EP_DESC = 传递错误的端点描述符。

表 192. USBD_CDC_API 类结构 *（续）*

成员	描述
SendNotification	`ErrorCode_t(*ErrorCode_t USBD_CDC_API::SendNotification)(USBD_HANDLE_T hCdc, uint8_t bNotification, uint16_t data)` 初始化 CDC 函数驱动器模块的函数。 应用层调用该函数以初始化 CDC 函数驱动器模块。 hUsb：USB 设备协议栈的句柄。 param：含 CDC 函数驱动器模块初始化参数的结构。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. param = 含 CDC 函数驱动器模块初始化参数的结构。 返回： 返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功 2. ERR_USBD_BAD_MEM_BUF = 传递的存储器缓冲区不以 4 字节对齐或小于要求。 3. ERR_API_INVALID_PARAM2 = CDC_Write()、 CDC_Read() 或 CDC_Verify() 回调未定义。 4. ERR_USBD_BAD_INTF_DESC = 传递错误的接口描述符。 5. ERR_USBD_BAD_EP_DESC = 传递错误的端点描述符。

11.5.28 USBD_CDC_INIT_PARAM

通信设备类函数驱动器初始化参数数据结构。

表 193. USBD_CDC_INIT_PARAM 类结构

成员	描述
mem_base	`uint32_tuint32_t USBD_CDC_INIT_PARAM::mem_base` 协议栈从中分配数据和缓冲的基本存储器位置。 **注：**此字段中设置的存储器地址应可由 USB DMA 控制器访问。该值还应在 4 字节边界对齐。
mem_size	`uint32_tuint32_t USBD_CDC_INIT_PARAM::mem_size` 协议栈可使用的存储器缓冲区大小。 **注：**mem_size 应大于 USBD_CDC_API::GetMemSize() 例程返回的大小。
cif_intf_desc	`uint8_t *uint8_t* USBD_CDC_INIT_PARAM::cif_intf_desc` 指向描述符数组内的控制接口描述符的指针 （
dif_intf_desc	`uint8_t *uint8_t* USBD_CDC_INIT_PARAM::dif_intf_desc` 指向描述符数组内的数据接口描述符的指针 （

表 193. USBD_CDC_INIT_PARAM 类结构 *（续）*

成员	描述
CIC_GetRequest	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::CIC_GetRequest)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t *length))(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t *length)` 通信接口类特定获取请求回调函数。 该函数由应用软件提供。当主机发送 CIC 管理元素获取请求时，调用该函数。设置数据包 （ hCdc：CDC 函数驱动器的句柄。 pSetup：指向从主机接收的设置数据包的指针。 pBuffer：指向含请求数据的数据缓冲区指针的指针。指向指针的指针用于实施 0 复制缓冲区。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 length：发回主机的数据量。 参数： 1. hCdc = CDC 函数驱动器的句柄。 2. pSetup = 指向从主机接收的设置数据包的指针。 3. pBuffer = 指向含请求数据的数据缓冲区指针的指针。指向指针的指针用于实施 0 复制缓冲区。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 4. length = 发回主机的数据量。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。
CIC_SetRequest	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::CIC_SetRequest)(USBD_HANDLE_T hCdc, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t length))(USBD_HANDLE_T hCdc, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t length)` 通信接口类特定设置请求回调函数。 该函数由应用软件提供。当主机发送 CIC 管理元素请求时，调用该函数。设置数据包 （ hCdc：CDC 函数驱动器的句柄。 pSetup：指向从主机接收的设置数据包的指针。 pBuffer：指向含请求数据的数据缓冲区指针的指针。指向指针的指针用于实施 0 复制缓冲区。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 length：复制到目标缓冲区的数据量。 参数： 1. hCdc = CDC 函数驱动器的句柄。 2. pSetup = 指向从主机接收的设置数据包的指针。 3. pBuffer = 指向含请求数据的数据缓冲区指针的指针。指向指针的指针用于实施 0 复制缓冲区。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 4. length = 复制到目标缓冲区的数据量。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。

表 193. USBD_CDC_INIT_PARAM 类结构 *（续）*

成员	描述
CDC_BulkIN_Hdlr	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_BulkIN_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event))(USBD_HANDLE_T hUsb, void *data, uint32_t event)` 通信设备类特定 BULK IN 端点处理器。 应用软件应提供 BULK IN 端点处理器。应用程序应根据描述符中设置的通信协议类型传输数据。 **注：** 参数： 1. hUsb = USB 设备协议栈的句柄。 2. data = 指向协议栈调用回调函数时所传递数据的指针。 3. event = 端点事件的类型。更多详情请参见 USBD_EVENT_T。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。
CDC_BulkOUT_Hdlr	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_BulkOUT_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event))(USBD_HANDLE_T hUsb, void *data, uint32_t event)` 通信设备类特定 BULK OUT 端点处理器。 应用软件应提供 BULK OUT 端点处理器。应用程序应根据描述符中设置的通信协议类型传输数据。 **注：** 参数： 1. hUsb = USB 设备协议栈的句柄。 2. data = 指向协议栈调用回调函数时所传递数据的指针。 3. event = 端点事件的类型。更多详情请参见 USBD_EVENT_T。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。
SendEncpsCmd	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::SendEncpsCmd)(USBD_HANDLE_T hCDC, uint8_t *buffer, uint16_t len))(USBD_HANDLE_T hCDC, uint8_t *buffer, uint16_t len)`
GetEncpsResp	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::GetEncpsResp)(USBD_HANDLE_T hCDC, uint8_t **buffer, uint16_t *len))(USBD_HANDLE_T hCDC, uint8_t **buffer, uint16_t *len)`
SetCommFeature	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::SetCommFeature)(USBD_HANDLE_T hCDC, uint16_t feature, uint8_t *buffer, uint16_t len))(USBD_HANDLE_T hCDC, uint16_t feature, uint8_t *buffer, uint16_t len)`
GetCommFeature	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::GetCommFeature)(USBD_HANDLE_T hCDC, uint16_t feature, uint8_t **pBuffer, uint16_t *len))(USBD_HANDLE_T hCDC, uint16_t feature, uint8_t **pBuffer, uint16_t *len)`
ClrCommFeature	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::ClrCommFeature)(USBD_HANDLE_T hCDC, uint16_t feature))(USBD_HANDLE_T hCDC, uint16_t feature)`

表 193. USBD_CDC_INIT_PARAM 类结构 *（续）*

成员	描述
SetCtrlLineState	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::SetCtrlLineState)(USBD_HANDLE_T hCDC, uint16_t state))(USBD_HANDLE_T hCDC, uint16_t state)`
SendBreak	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::SendBreak)(USBD_HANDLE_T hCDC, uint16_t mstime))(USBD_HANDLE_T hCDC, uint16_t mstime)`
SetLineCode	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::SetLineCode)(USBD_HANDLE_T hCDC, CDC_LINE_CODING *line_coding))(USBD_HANDLE_T hCDC, CDC_LINE_CODING *line_coding)`
CDC_InterruptEP_Hdlr	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_InterruptEP_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event))(USBD_HANDLE_T hUsb, void *data, uint32_t event)` 可选通信设备类特定 INTERRUPT IN 端点处理器。 应用软件应提供 INT IN 端点处理器。应用程序应根据描述符中设置的通信协议类型传输数据。 **注：** 参数： 1. hUsb = USB 设备协议栈的句柄。 2. data = 指向协议栈调用回调函数时所传递数据的指针。 3. event = 端点事件的类型。更多详情请参见 USBD_EVENT_T。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。
CDC_Ep0_Hdlr	`ErrorCode_t(*ErrorCode_t(* USBD_CDC_INIT_PARAM::CDC_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event))(USBD_HANDLE_T hUsb, void *data, uint32_t event)` 代替默认 CDC 类处理器可选用户可覆盖函数。 应用软件提供处理器函数地址作为参数结构的此数据成员，从而用自己的处理器覆盖默认 EP0 类处理器。首选默认处理器的应用应在调用 USBD_CDC_API::Init() 前将该数据成员置零。 **注：** 参数： 1. hUsb = USB 设备协议栈的句柄。 2. data = 指向协议栈调用回调函数时所传递数据的指针。 3. event = 端点事件的类型。更多详情请参见 USBD_EVENT_T。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。

11.5.29 USBD_CORE_API

USBD 协议栈内核 API 函数结构。

表 194. USBD_CORE_API 类结构

成员	描述
RegisterClassHandler	`ErrorCode_t(*ErrorCode_t USBD_CORE_API::RegisterClassHandler)(USBD_HANDLE_T hUsb, USB_EP_HANDLER_T pfn, void *data)` 向 USB 设备协议栈注册类特定 EP0 事件处理器的函数。 应用层在必须注册自定义类 EP0 处理器时，将使用该函数。该协议栈在进行事件的默认处理前对任何 EP0 事件调用所有注册的类处理器。这就提供了类处理器来实施特定类的请求处理器，并覆盖面向接口的特定事件的默认协议栈处理。有关如何实施回调函数的更多详情，请参见 USB_EP_HANDLER_T。应用层还可使用该函数注册响应供应商特定请求的 EP0 处理器。 hUsb：USB 设备协议栈的句柄。 pfn：特定类的 EP0 处理器函数。 data：指向协议栈调用回调函数时所传递数据的指针。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. pfn = 特定类的 EP0 处理器函数。 3. data = 指向协议栈调用回调函数时所传递数据的指针。 返回： 返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功 2. ERR_USBD_TOO_MANY_CLASS_HDLR(0x0004000c) = 注册的类处理器数量大于协议栈允许的处理器数量。
RegisterEpHandler	`ErrorCode_t(*ErrorCode_t USBD_CORE_API::RegisterEpHandler)(USBD_HANDLE_T hUsb, uint32_t ep_index, USB_EP_HANDLER_T pfn, void *data)` 向 USB 设备协议栈注册所请求端点的中断 / 事件处理器的函数。 应用层使用该函数注册自定义类 EP0 处理器。该协议栈在进行事件的默认处理前对任何 EP0 事件调用所有注册的类处理器。这就提供了类处理器来实施特定类的请求处理器，并覆盖面向接口的特定事件的默认协议栈处理。有关如何实施回调函数的更多详情，请参见 USB_EP_HANDLER_T。 hUsb：USB 设备协议栈的句柄。 ep_index：特定类的 EP0 处理器函数。 pfn：特定类的 EP0 处理器函数。 data：指向协议栈调用回调函数时所传递数据的指针。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. ep_index = 特定类的 EP0 处理器函数。 3. pfn = 特定类的 EP0 处理器函数。 4. data = 指向协议栈调用回调函数时所传递数据的指针。 返回： 返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功 2. ERR_USBD_TOO_MANY_CLASS_HDLR(0x0004000c) = 端点处理器太多。

表 194. USBD_CORE_API 类结构 *（续）*

成员	描述
SetupStage	`void(*void USBD_CORE_API::SetupStage)(USBD_HANDLE_T hUsb)` 在设置状态设置 EP0 状态机的函数。 该函数由 USB 协议栈和应用层调用，以在设置状态设置 EP0 状态机。该函数会将从 USB 主机接收的设置数据包读入协议栈缓存。 **注：**该接口提供给用户以在 （当前协议栈未处理的）其他情况下调用此函数。在多数用户应用中，该函数未直接调用。该函数还可由用户用来通过协议栈公开的回调接口，选择性地修改 USB 设备协议栈的标准处理器。 参数： 1. hUsb = USB 设备协议栈的句柄。 返回： 无任何内容。
DataInStage	`void(*void USBD_CORE_API::DataInStage)(USBD_HANDLE_T hUsb)` 在 data_in 状态设置 EP0 状态机的函数。 该函数由 USB 协议栈或应用层调用，以在 data_in 状态设置 EP0 状态机。该函数会将 EP0Data 缓冲区中存在的数据写入 EP0 FIFO，以发送至主机。 **注：**该接口提供给用户以在 （当前协议栈未处理的）其他情况下调用此函数。在多数用户应用中，该函数未直接调用。该函数还可由用户用来通过协议栈公开的回调接口，选择性地修改 USB 设备协议栈的标准处理器。 参数： 1. hUsb = USB 设备协议栈的句柄。 返回： 无任何内容。
DataOutStage	`void(*void USBD_CORE_API::DataOutStage)(USBD_HANDLE_T hUsb)` 在 data_out 状态设置 EP0 状态机的函数。 该函数由 USB 协议栈或应用层调用，以在 data_out 状态设置 EP0 状态机。该函数会将从 USB 主机接收的控制数据 （EP0 OUT 数据包）读入 EP0Data 缓冲区。 **注：**该接口提供给用户以在 （当前协议栈未处理的）其他情况下调用此函数。在多数用户应用中，该函数未直接调用。该函数还可由用户用来通过协议栈公开的回调接口，选择性地修改 USB 设备协议栈的标准处理器。 参数： 1. hUsb = USB 设备协议栈的句柄。 返回： 无任何内容。
StatusInStage	`void(*void USBD_CORE_API::StatusInStage)(USBD_HANDLE_T hUsb)` 在 status_in 状态设置 EP0 状态机的函数。 该函数由 USB 协议栈或应用层调用，以在 status_in 状态设置 EP0 状态机。该函数会将 EP0 上的零长度 IN 数据包发送至主机，表示正状态。 **注：**该接口提供给用户以在 （当前协议栈未处理的）其他情况下调用此函数。在多数用户应用中，该函数未直接调用。该函数还可由用户用来通过协议栈公开的回调接口，选择性地修改 USB 设备协议栈的标准处理器。 参数： 1. hUsb = USB 设备协议栈的句柄。 返回： 无任何内容。

表 194. USBD_CORE_API 类结构 *（续）*

成员	描述
StatusOutStage	`void(*void USBD_CORE_API::StatusOutStage)(USBD_HANDLE_T hUsb)` 在 status_out 状态设置 EP0 状态机的函数。 该函数由 USB 协议栈或应用层调用，以在 status_out 状态设置 EP0 状态机。该函数会将从 USB 主机接收的零长度 OUT 数据包读到 EP0。 **注：**该接口提供给用户以在 （当前协议栈未处理的）其他情况下调用此函数。在多数用户应用中，该函数未直接调用。该函数还可由用户用来通过协议栈公开的回调接口，选择性地修改 USB 设备协议栈的标准处理器。 参数： 1. hUsb = USB 设备协议栈的句柄。 返回： 无任何内容。
StallEp0	`void(*void USBD_CORE_API::StallEp0)(USBD_HANDLE_T hUsb)` 用于在停止状态设置 EP0 状态机的函数。 该函数由 USB 协议栈和应用层调用，以在 EP0 端点产生 STALL 信号。该函数还将复位 EP0Data 缓冲区。 **注：**该接口提供给用户以在 （当前协议栈未处理的）其他情况下调用此函数。在多数用户应用中，该函数未直接调用。该函数还可由用户用来通过协议栈公开的回调接口，选择性地修改 USB 设备协议栈的标准处理器。 参数： 1. hUsb = USB 设备协议栈的句柄。 返回： 无任何内容。

11.5.30 USBD_DFU_API

DFU 类 API 函数结构。该模块公开直接与 USB 设备控制器硬件交互的函数。

表 195. USBD_DFU_API 类结构

成员	描述
GetMemSize	`uint32_t(*uint32_t USBD_DFU_API::GetMemSize)(USBD_DFU_INIT_PARAM_T *param)` 用于确定 DFU 函数驱动器模块所要求的存储器的函数。 在调用 pUsbApi->dfu->Init() 前，应用层调用该函数以分配 DFU 函数驱动器模块所使用的存储器。应用程序应分配可由 USB 控制器 /DMA 控制器访问的存储器。 **注：**一些存储器区域无法由所有总线主机访问。 参数： 1. param = 含 DFU 函数驱动器模块初始化参数的结构。 返回： 返回需要的存储器大小 （字节）。
init	`ErrorCode_t(*ErrorCode_t USBD_DFU_API::init)(USBD_HANDLE_T hUsb, USBD_DFU_INIT_PARAM_T *param, uint32_t init_state)` 初始化 DFU 函数驱动器模块的函数。 应用层调用该函数以初始化 DFU 函数驱动器模块。 hUsb：USB 设备协议栈的句柄。 param：含 DFU 函数驱动器模块初始化参数的结构。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. param = 含 DFU 函数驱动器模块初始化参数的结构。 返回： 返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功 2. ERR_USBD_BAD_MEM_BUF = 传递的存储器缓冲区不以 4 字节对齐或小于要求。 3. ERR_API_INVALID_PARAM2 = DFU_Write()、 DFU_Done() 或 DFU_Read() 回调未定义。 4. ERR_USBD_BAD_DESC = USB_DFU_DESCRIPTOR_TYPE 未紧接在接口描述符之后定义。描述符中的 wTransferSize 与 param->wTransferSize 中传递的值不匹配。在 DFU 描述符中设置了 USB_DFU_WILL_DETACH 时，未定义 DFU_Detach()。 5. ERR_USBD_BAD_INTF_DESC = 传递错误的接口描述符。

11.5.31 USBD_DFU_INIT_PARAM

USB 描述符数据结构。

表 196. USBD_DFU_INIT_PARAM 类结构

成员	描述
mem_base	`uint32_tuint32_t USBD_DFU_INIT_PARAM::mem_base` 协议栈从中分配数据和缓冲的基本存储器位置。 **注：**此字段中设置的存储器地址应可由 USB DMA 控制器访问。该值还应在 4 字节边界对齐。
mem_size	`uint32_tuint32_t USBD_DFU_INIT_PARAM::mem_size` 协议栈可使用的存储器缓冲区大小。 **注：**mem_size 应大于 USBD_DFU_API::GetMemSize() 例程返回的大小。
wTransferSize	`uint16_tuint16_t USBD_DFU_INIT_PARAM::wTransferSize` DFU 传输模块大小 （字节数）。该值应匹配作为描述符数组一部分的 DFU 描述符中设置的值 （
焊盘	`uint16_tuint16_t USBD_DFU_INIT_PARAM::pad`
intf_desc	`uint8_t *uint8_t* USBD_DFU_INIT_PARAM::intf_desc` 指向描述符数组内的 DFU 接口描述符的指针 （
DFU_Write	`uint8_t(*uint8_t(* USBD_DFU_INIT_PARAM::DFU_Write)(uint32_t block_num, uint8_t **src, uint32_t length, uint8_t *bwPollTimeout))(uint32_t block_num, uint8_t **src, uint32_t length, uint8_t *bwPollTimeout)` DFU 写回调函数。 该函数由应用软件提供。主机发送写命令时调用此函数。对于使用 0 复制缓冲机制的应用，该函数首次采用调用 block_num：目标起始地址。 src：指向数据源指针的指针。指向指针的指针用于实施 0 复制缓冲区。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 bwPollTimeout：指向 3 字节缓冲区的指针，回调实施者应在其中填充主机发送后续 DFU_GETSTATUS 请求前应等待的最短时间 （毫秒）。 length：要写入的字节数。 参数： 1. block_num = 目标起始地址。 2. src = 指向数据源指针的指针。指向指针的指针用于实施 0 复制缓冲区。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 3. bwPollTimeout = 指向 3 字节缓冲区的指针，回调实施者应在其中填充主机在发送后续 DFU_GETSTATUS 请求前应等待的最短时间 （毫秒）。 4. length = 要写入的字节数。 返回： 返回 mw_usbd_dfu.h 中定义的 DFU_STATUS_ 值。

表 196. USBD_DFU_INIT_PARAM 类结构 *（续）*

成员	描述
DFU_Read	`uint32_t(*uint32_t(* USBD_DFU_INIT_PARAM::DFU_Read)(uint32_t block_num, uint8_t **dst, uint32_t length))(uint32_t block_num, uint8_t **dst, uint32_t length)` DFU 读回调函数。 该函数由应用软件提供。主机发送读命令时调用此函数。 block_num：目标起始地址。 dst：指向数据源指针的指针。指向指针的指针用于实施 0 复制缓冲区。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 length：复制到目标缓冲区的数据量。 参数： 1. block_num = 目标起始地址。 2. dst = 指向数据源指针的指针。指向指针的指针用于实施 0 复制缓冲区。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 3. length = 复制到目标缓冲区的数据量。 返回： 返回 mw_usbd_dfu.h 中定义的 DFU_STATUS_ 值。
DFU_Done	`void(*void(* USBD_DFU_INIT_PARAM::DFU_Done)(void))(void)` DFU 完成回调函数。 该函数由应用软件提供。下载完成后调用该函数。 无任何内容。 返回： 无任何内容。
DFU_Detach	`void(*void(* USBD_DFU_INIT_PARAM::DFU_Detach)(USBD_HANDLE_T hUsb))(USBD_HANDLE_T hUsb)` DFU 分离回调函数。 该函数由应用软件提供。接收 USB_REQ_DFU_DETACH 后调用该函数。在 DFU 描述符中设置 USB_DFU_WILL_DETACH 位的应用应定义该函数。作为该函数的一部分，应用可调用 Connect() 例程以断开连接并和主机重新连接。由于 USB 复位只能由 Windows 主机上的内核驱动器调用，依靠基于 WinUSB 的主机的应用应使用该功能。实施该功能后，主机不需要发出重设，而是由设备通过连接和断开连接程序自动执行。 hUsb 处理 DFU 控制结构。 参数： 1. hUsb = 处理 DFU 控制结构。 返回： 无任何内容。

表 196. USBD_DFU_INIT_PARAM 类结构 *（续）*

成员	描述
DFU_Ep0_Hdlr	`ErrorCode_t(*ErrorCode_t(* USBD_DFU_INIT_PARAM::DFU_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event))(USBD_HANDLE_T hUsb, void *data, uint32_t event)` 代替默认 DFU 类处理器可选用户可覆盖函数。 应用软件提供处理器函数地址作为参数结构的此数据成员，从而用自己的处理器覆盖默认 EP0 类处理器。首选默认处理器的应用应在调用 USBD_DFU_API::Init() 前将该数据成员置零。 **注：** 参数： 1. hUsb = USB 设备协议栈的句柄。 2. data = 指向协议栈调用回调函数时所传递数据的指针。 3. event = 端点事件的类型。更多详情请参见 USBD_EVENT_T。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。

11.5.32 USBD_HID_API

HID 类 API 函数结构。该结构包含指向 HID 函数驱动器模块所公开的所有函数的指针。

表 197. USBD_HID_API 类结构

成员	描述
GetMemSize	`uint32_t(*uint32_t USBD_HID_API::GetMemSize)(USBD_HID_INIT_PARAM_T *param)` 用于确定 HID 函数驱动器模块所要求的存储器的函数。 在调用 pUsbApi->hid->Init() 前，应用层调用该函数以分配 HID 函数驱动器模块所使用的存储器。应用程序应分配可由 USB 控制器 /DMA 控制器访问的存储器。 **注：**一些存储器区域无法由所有总线主机访问。 参数： 1. param = 含 HID 函数驱动器模块初始化参数的结构。 返回： 返回需要的存储器大小 （字节）。
init	`ErrorCode_t(*ErrorCode_t USBD_HID_API::init)(USBD_HANDLE_T hUsb, USBD_HID_INIT_PARAM_T *param)` 初始化 HID 函数驱动器模块的函数。 应用层调用该函数以初始化 HID 函数驱动器模块。成功初始化后，该函数将返回已传递 param 结构中 HID 函数驱动器模块的一个句柄。 hUsb：USB 设备协议栈的句柄。 param：含 HID 函数驱动器模块初始化参数的结构。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. param = 含 HID 函数驱动器模块初始化参数的结构。 返回： 返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功 2. ERR_USBD_BAD_MEM_BUF = 传递的存储器缓冲区不以 4 字节对齐或小于要求。 3. ERR_API_INVALID_PARAM2 = HID_GetReport() 或 HID_SetReport() 回调未定义。 4. ERR_USBD_BAD_DESC = HID_HID_DESCRIPTOR_TYPE 未紧接在接口描述符之后定义。 5. ERR_USBD_BAD_INTF_DESC = 传递错误的接口描述符。 6. ERR_USBD_BAD_EP_DESC = 传递错误的端点描述符。

11.5.33 USBD_HID_INIT_PARAM

USB 描述符数据结构。

表 198. USBD_HID_INIT_PARAM 类结构

成员	描述
mem_base	`uint32_tuint32_t USBD_HID_INIT_PARAM::mem_base` 协议栈从中分配数据和缓冲的基本存储器位置。 **注：**此字段中设置的存储器地址应可由 USB DMA 控制器访问。该值还应在 4 字节边界对齐。
mem_size	`uint32_tuint32_t USBD_HID_INIT_PARAM::mem_size` 协议栈可使用的存储器缓冲区大小。 **注：**mem_size 应大于 USBD_HID_API::GetMemSize() 例程返回的大小。
max_reports	`uint8_tuint8_t USBD_HID_INIT_PARAM::max_reports` 该 HID 类驱动器实例支持的 HID 报告数量。
焊盘	`uint8_tuint8_t USBD_HID_INIT_PARAM::pad[3][3]`
intf_desc	`uint8_t *uint8_t* USBD_HID_INIT_PARAM::intf_desc` 指向描述符数组内的 HID 接口描述符的指针 （
report_data	`USB_HID_REPORT_T *USB_HID_REPORT_T* USBD_HID_INIT_PARAM::report_data` 指向 HID 报告描述符数据结构数组的指针 （ **注：**该数组应属于全局范围。
HID_GetReport	`ErrorCode_t(*ErrorCode_t(* USBD_HID_INIT_PARAM::HID_GetReport)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t *length))(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t *length)` HID 获取报告回调函数。 该函数由应用软件提供。主机发送 HID_REQUEST_GET_REPORT 请求时调用此函数。设置数据包（ **注：**还会通过中断 IN 端点发送 HID 报告。仅当在控制端点上接收报告请求时，才调用该函数。应用应实施 HID_EpIn_Hdlr 以通过中断 IN 端点将报告发送至主机。 参数： 1. hHid = HID 函数驱动器的句柄。 2. pSetup = 指向从主机接收的设置数据包的指针。 3. pBuffer = 指向含报告数据的数据缓冲区指针的指针。指向指针的指针用于实施 0 复制缓冲区。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 4. length = 复制到目标缓冲区的数据量。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。

表 198. USBD_HID_INIT_PARAM 类结构 *（续）*

成员	描述
HID_SetReport	`ErrorCode_t(*ErrorCode_t(* USBD_HID_INIT_PARAM::HID_SetReport)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t length))(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuffer, uint16_t length)` HID 设置报告回调函数。 该函数由应用软件提供。主机发送 HID_REQUEST_SET_REPORT 请求时调用此函数。设置数据包（ hHid：HID 函数驱动器的句柄。pSetup：指向从主机接收的设置数据包的指针。pBuffer：指向含报告数据的数据缓冲区指针的指针。指向指针的指针用于实施 0 复制缓冲区。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 length：复制到目标缓冲区的数据量。 参数： 1. hHid = HID 函数驱动器的句柄。 2. pSetup = 指向从主机接收的设置数据包的指针。 3. pBuffer = 指向含报告数据的数据缓冲区指针的指针。指向指针的指针用于实施 0 复制缓冲区。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 4. length = 复制到目标缓冲区的数据量。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。
HID_GetPhysDesc	`ErrorCode_t(*ErrorCode_t(* USBD_HID_INIT_PARAM::HID_GetPhysDesc)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuf, uint16_t *length))(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuf, uint16_t *length)` 用于处理 HID_GetPhysDesc 请求的可选回调函数。 应用软件可提供此回调 HID_GetPhysDesc 处理器以处理主机发送的获取物理描述符请求。当主机请求将物理描述符置零时，应用应返回一个可标识描述符集数量及其大小的特殊描述符。物理索引等于 1 的 Get_Descriptor 请求应返回第一个物理描述符集。一个设备的项目可能有备用的用法。这些可以通过发出后续 Get_Descriptor 请求，同时递增描述符索引来枚举。设备返回的最后一个描述符集应针对索引大于 HID 描述符中所定义的最后编号的请求。 **注：**不含物理描述符的应用应在调用 USBD_HID_API::Init() 前将该数据成员置零。 参数： 1. hHid = HID 函数驱动器的句柄。 2. pSetup = 指向从主机接收的设置数据包的指针。 3. pBuf = 指向含物理描述符数据的数据缓冲区指针的指针。如果物理描述符位于 USB 可访问存储器区域中，应用可更新指针，否则应将描述符复制到该指针指向的地址。 4. length = 复制到目标缓冲区的数据量或描述符长度。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。

表 198. USBD_HID_INIT_PARAM 类结构 *（续）*

成员	描述
HID_SetIdle	`ErrorCode_t(*ErrorCode_t(* USBD_HID_INIT_PARAM::HID_SetIdle)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t idleTime))(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t idleTime)` 用于处理 HID_REQUEST_SET_IDLE 请求的可选回调函数。 应用软件可提供此回调以处理主机发送的 HID_REQUEST_SET_IDLE 请求。该回调提供给应用，以调整与通过中断端点发送给主机的各种报告关联的定时器。设置数据包 （ **注：**未在中断端点上发送报告或者在两个报告之间没有空闲时间的应用应在调用 USBD_HID_API::Init() 前将该数据成员置零。 参数： 1. hHid = HID 函数驱动器的句柄。 2. pSetup = 指向从主机接收的设置数据包的指针。 3. idleTime = 要为指定报告设置的空闲时间。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。
HID_SetProtocol	`ErrorCode_t(*ErrorCode_t(* USBD_HID_INIT_PARAM::HID_SetProtocol)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t protocol))(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t protocol)` 用于处理 HID_REQUEST_SET_PROTOCOL 请求的可选回调函数。 应用软件可提供此回调以处理主机发送的 HID_REQUEST_SET_PROTOCOL 请求。该回调提供给应用，以调整启动模式和报告模式之间代码的模式。 **注：**不支持协议模式的应用应在调用 USBD_HID_API::Init() 前将该数据成员置零。 参数： 1. hHid = HID 函数驱动器的句柄。 2. pSetup = 指向从主机接收的设置数据包的指针。 3. protocol = 协议模式。 0 = 启用协议 1 = 报告协议 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。

表 198. USBD_HID_INIT_PARAM 类结构 *（续）*

成员	描述
HID_EpIn_Hdlr	`ErrorCode_t(*ErrorCode_t(* USBD_HID_INIT_PARAM::HID_EpIn_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event))(USBD_HANDLE_T hUsb, void *data, uint32_t event)` 可选中断 IN 端点事件处理器。 应用软件可提供中断 IN 端点事件处理器。在中断端点上将报告发送至主机的应用应通过该数据成员提供端点事件处理器。如果接口描述符，忽略此数据成员 hUsb：USB 设备协议栈的句柄。 data：HID 函数驱动器的句柄。 event：端点事件的类型。更多详情请参见 USBD_EVENT_T。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. data = HID 函数驱动器的句柄。 3. event = 端点事件的类型。更多详情请参见 USBD_EVENT_T。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。
HID_EpOut_Hdlr	`ErrorCode_t(*ErrorCode_t(* USBD_HID_INIT_PARAM::HID_EpOut_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event))(USBD_HANDLE_T hUsb, void *data, uint32_t event)` 可选中断 OUT 端点事件处理器。 应用软件可提供中断 OUT 端点事件处理器。在中断端点上从主机接收报告的应用应通过该数据成员提供端点事件处理器。如果接口描述符，忽略此数据成员 hUsb：USB 设备协议栈的句柄。 data：HID 函数驱动器的句柄。 event：端点事件的类型。更多详情请参见 USBD_EVENT_T。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. data = HID 函数驱动器的句柄。 3. event = 端点事件的类型。更多详情请参见 USBD_EVENT_T。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。

表 198. USBD_HID_INIT_PARAM 类结构 *（续）*

成员	描述
HID_GetReportDesc	`ErrorCode_t(*ErrorCode_t(* USBD_HID_INIT_PARAM::HID_GetReportDesc)(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuf, uint16_t *length))(USBD_HANDLE_T hHid, USB_SETUP_PACKET *pSetup, uint8_t **pBuf, uint16_t *length)` 代替默认 HID_GetReportDesc 处理器可选用户可覆盖函数。 应用软件提供处理器函数地址作为参数结构的此数据成员，从而用自己的处理器覆盖默认 HID_GetReportDesc 处理器。首选默认处理器的应用应在调用 USBD_HID_API::Init() 前将该数据成员置零，并提供报告数据阵列 **注：** 参数： 1. hUsb = USB 设备协议栈的句柄。 2. data = 指向协议栈调用回调函数时所传递数据的指针。 3. event = 端点事件的类型。更多详情请参见 USBD_EVENT_T。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。
HID_Ep0_Hdlr	`ErrorCode_t(*ErrorCode_t(* USBD_HID_INIT_PARAM::HID_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event))(USBD_HANDLE_T hUsb, void *data, uint32_t event)` 代替默认 HID 类处理器可选用户可覆盖函数。 应用软件提供处理器函数地址作为参数结构的此数据成员，从而用自己的处理器覆盖默认 EP0 类处理器。首选默认处理器的应用应在调用 USBD_HID_API::Init() 前将该数据成员置零。 **注：** 参数： 1. hUsb = USB 设备协议栈的句柄。 2. data = 指向协议栈调用回调函数时所传递数据的指针。 3. event = 端点事件的类型。更多详情请参见 USBD_EVENT_T。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。

11.5.34 USBD_HW_API

硬件 API 函数结构。该模块公开直接与 USB 设备控制器硬件交互的函数。

表 199. USBD_HW_API 类结构

成员	描述
GetMemSize	`uint32_t(*uint32_t USBD_HW_API::GetMemSize)(USBD_API_INIT_PARAM_T *param)` 用于确定 USB 设备协议栈的 DCD 和内核层所要求的存储器的函数。 应用层在调用 pUsbApi->hw-> 前调用该函数。 **注：**一些存储器区域无法由所有总线主机访问。 参数： 1. param = 含 USB 设备协议栈初始化参数的结构。 返回： 返回需要的存储器大小 （字节）。
Init	`ErrorCode_t(*ErrorCode_t USBD_HW_API::Init)(USBD_HANDLE_T *phUsb, USB_CORE_DESCS_T *pDesc, USBD_API_INIT_PARAM_T *param)` 用于初始化 USB 设备协议栈的 DCD 和内核层的函数。 应用层调用该函数以初始化 USB 硬件和内核层。成功初始化后，该函数将返回应传递给剩余函数的 USB 设备协议栈的一个句柄。 phUsb：指向 USBD_HANDLE_T 类型的 USB 设备协议栈句柄的指针。 param：含 USB 设备协议栈初始化参数的结构。 参数： 1. phUsb = 指向 USBD_HANDLE_T 类型的 USB 设备协议栈句柄的指针。 2. param = 含 USB 设备协议栈初始化参数的结构。 返回： 返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK(0) = 成功 2. ERR_USBD_BAD_MEM_BUF(0x0004000b) = 当未传递足够的存储器缓冲或者存储器未在 2048 边界上对齐时。
Connect	`void(*void USBD_HW_API::Connect)(USBD_HANDLE_T hUsb, uint32_t con)` 使 USB 设备在 USB 总线上可见 / 不可见的函数。 USB 初始化后，调用该函数。该函数使用软连接功能使设备在 USB 总线上可见。仅在应用准备好处理 USB 数据后，调用该函数。枚举过程在设备检测后由主机启动。驱动器根据在 USB 初始化函数中传递的 USB 描述符处理枚举过程。 hUsb：USB 设备协议栈的句柄。 con：连接 (1) 或断开连接 (0) 状态。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. con = 连接 (1) 或断开连接 (0) 状态。 返回： 无任何内容。

表 199. USBD_HW_API 类结构 *（续）*

成员	描述
ISR	`void(*void USBD_HW_API::ISR)(USBD_HANDLE_T hUsb)` USB 设备控制器中断事件函数。 激活用户应用时，中断处理器映射到用户闪存空间。用户应用必须为 USB 中断提供中断处理器，并在中断处理器例程中调用该函数。驱动器中断处理器根据 USB 总线上接收的数据采取适当操作。 hUsb：USB 设备协议栈的句柄。 参数： 1. hUsb = USB 设备协议栈的句柄。 返回： 无任何内容。
复位	`void(*void USBD_HW_API::Reset)(USBD_HANDLE_T hUsb)` 用于复位 USB 设备协议栈和硬件控制器的函数。 复位 USB 设备协议栈和硬件控制器。禁用除 EP0 外的所有端点。清除所有挂起中断并复位端点传输队列。在内部通过 pUsbApi->hw->init() 并从复位事件调用该函数。 hUsb：USB 设备协议栈的句柄。 参数： 1. hUsb = USB 设备协议栈的句柄。 返回： 无任何内容。
ForceFullSpeed	`void(*void USBD_HW_API::ForceFullSpeed)(USBD_HANDLE_T hUsb, uint32_t cfg)` 强制高速 USB 设备以全速模式运行的函数。 连接到仅限全速的主机时，该函数对检测当前设备的行为很有用。 hUsb：USB 设备协议栈的句柄。 cfg：为 1 时 - 设置强制全速或者为 0 时 - 清除强制全速。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. cfg = 为 1 时 - 设置强制全速或者为 0 时 - 清除强制全速。 返回： 无任何内容。
WakeUpCfg	`void(*void USBD_HW_API::WakeUpCfg)(USBD_HANDLE_T hUsb, uint32_t cfg)` 用于配置 USB 设备控制器以基于远程事件唤醒主机的函数。 应用层调用该函数以配置 USB 设备控制器以基于远程事件唤醒。建议从注册到协议栈的用户 USB_WakeUpCfg() 回调例程调用此函数。 **注：**调用 pUsbApi->hw->Init() 例程前，通过设置 USBD_API_INIT_PARAM_T 结构的 USB_WakeUpCfg 成员将用户的 USB_WakeUpCfg() 注册到协议栈。某些 USB 设备控制器需要保持某些时钟始终运行以通过 pUsbApi->hw->WakeUp() 产生恢复信号。该挂接可支持此类控制器。在多数控制器情况中，这是空例程。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. cfg = 为 1 时 - 配置控制器以基于远程事件唤醒或者为 0 时 - 配置控制器以不基于远程事件唤醒。 返回： 无任何内容。

表 199. USBD_HW_API 类结构 *（续）*

成员	描述
SetAddress	`void(*void USBD_HW_API::SetAddress)(USBD_HANDLE_T hUsb, uint32_t adr)` 用于设置设备控制器硬件中的主机分配的 USB 地址的函数。 协议栈从 USB 主机接收 USB_REQUEST_SET_ADDRESS 请求后会自动调用该函数。该接口提供给用户以在 （当前协议栈未处理的）其他情况下调用此函数。在多数用户应用中，该函数未直接调用。该函数还可由用户用来通过协议栈公开的回调接口，选择性地修改 USB 设备协议栈的标准处理器。 hUsb：USB 设备协议栈的句柄。 adr：设备控制器应响应的 USB 总线地址。通常由 USB 主机分配。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. adr = 设备控制器应响应的 USB 总线地址。通常由 USB 主机分配。 返回： 无任何内容。
Configure	`void(*void USBD_HW_API::Configure)(USBD_HANDLE_T hUsb, uint32_t cfg)` 用于将设备控制器硬件配置为选定配置的函数。 协议栈从 USB 主机接收 USB_REQUEST_SET_CONFIGURATION 请求后会自动调用该函数。该接口提供给用户以在 （当前协议栈未处理的）其他情况下调用此函数。在多数用户应用中，该函数未直接调用。该函数还可由用户用来通过协议栈公开的回调接口，选择性地修改 USB 设备协议栈的标准处理器。 hUsb：USB 设备协议栈的句柄。 cfg：配置索引。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. cfg = 配置索引。 返回： 无任何内容。
ConfigEP	`void(*void USBD_HW_API::ConfigEP)(USBD_HANDLE_T hUsb, USB_ENDPOINT_DESCRIPTOR *pEPD)` 用于根据描述符配置 USB 端点的函数。 协议栈从 USB 主机接收 USB_REQUEST_SET_CONFIGURATION 请求后会自动调用该函数。配置所有与选定配置相关的端点。该接口提供给用户以在 （当前协议栈未处理的）其他情况下调用此函数。在多数用户应用中，该函数未直接调用。该函数还可由用户用来通过协议栈公开的回调接口，选择性地修改 USB 设备协议栈的标准处理器。 hUsb：USB 设备协议栈的句柄。 pEPD：USB 2.0 规范中定义的端点描述符结构。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. pEPD = USB 2.0 规范中定义的端点描述符结构。 返回： 无任何内容。

表 199. USBD_HW_API 类结构 *（续）*

成员	描述
DirCtrlEP	`void(*void USBD_HW_API::DirCtrlEP)(USBD_HANDLE_T hUsb, uint32_t dir)` 用于设置 USB 控制端点 EP0 的方向的函数。 该函数由协议栈按需自动调用。该接口提供给用户以在（当前协议栈未处理的）其他情况下调用此函数。在多数用户应用中，该函数未直接调用。该函数还可由用户用来通过协议栈公开的回调接口，选择性地修改 USB 设备协议栈的标准处理器。 hUsb：USB 设备协议栈的句柄。 cfg：为 1 时 - 在 IN 传输模式中设置 EP0，为 0 时 - 在 OUT 传输模式中设置 EP0 参数： 1. hUsb = USB 设备协议栈的句柄。 2. cfg = 为 1 时 - 在 IN 传输模式中设置 EP0，为 0 时 - 在 OUT 传输模式中设置 EP0 返回： 无任何内容。
EnableEP	`void(*void USBD_HW_API::EnableEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` 用于使能选定 USB 端点的函数。 该函数可在所选端点使能中断。 hUsb：USB 设备协议栈的句柄。 EPNum：符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. EPNum = 符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 返回： 无任何内容。 该函数可在所选端点使能中断。 hUsb：USB 设备协议栈的句柄。 EPNum：符合 USB 规范的与事件对应的端点号。例如， EP1_IN 由 0x81 编号表示。对于设备事件，将此 param 设为 0x0。 event：端点事件的类型。更多详情请参见 USBD_EVENT_T。 enable：1 - 使能事件， 0 - 禁用事件。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. EPNum = 符合 USB 规范的与事件对应的端点号。例如， EP1_IN 由 0x81 编号表示。对于设备事件，将此 param 设为 0x0。 3. event = 端点事件的类型。更多详情请参见 USBD_EVENT_T。 4. enable = 1 - 使能事件， 0 - 禁用事件。 返回： 返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK(0) = - 成功 2. ERR_USBD_INVALID_REQ(0x00040001) = - 无效事件类型。
DisableEP	`void(*void USBD_HW_API::DisableEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` 用于禁用选定 USB 端点的函数。 该函数可在所选端点禁用中断。 hUsb：USB 设备协议栈的句柄。 EPNum：符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. EPNum = 符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 返回： 无任何内容。

表 199. USBD_HW_API 类结构 *（续）*

成员	描述
ResetEP	`void(*void USBD_HW_API::ResetEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` 用于复位选定 USB 端点的函数。 该函数可刷新端点缓冲区并复位数据切换逻辑。 hUsb：USB 设备协议栈的句柄。 EPNum：符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. EPNum = 符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 返回： 无任何内容。
SetStallEP	`void(*void USBD_HW_API::SetStallEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` 用于停止选定 USB 端点的函数。 为请求的端点产生停止信号。 hUsb：USB 设备协议栈的句柄。 EPNum：符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. EPNum = 符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 返回： 无任何内容。
ClrStallEP	`void(*void USBD_HW_API::ClrStallEP)(USBD_HANDLE_T hUsb, uint32_t EPNum)` 用于为请求的端点清除停止状态的函数。 该函数可为请求的端点清除停止状态。 hUsb：USB 设备协议栈的句柄。 EPNum：符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. EPNum = 符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 返回： 无任何内容。
SetTestMode	`ErrorCode_t(*ErrorCode_t USBD_HW_API::SetTestMode)(USBD_HANDLE_T hUsb, uint8_t mode)` 用于在请求的检测模式中设置高速 USB 设备控制器的函数。 USB-IF 要求将高速设备置于各种检测模式进行电气检测。每次接收到针对 USB_FEATURE_TEST_MODE 的 USB_REQUEST_CLEAR_FEATURE 请求时，该 USB 设备协议栈均会调用此函数。用户可以通过直接调用此函数将设备置于检测模式。设备控制器仅以全速运行时，返回 ERR_USBD_INVALID_REQ。 hUsb：USB 设备协议栈的句柄。 mode：USB 2.0 电气检测规范中定义的检测模式。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. mode = USB 2.0 电气检测规范中定义的检测模式。 返回： 返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK(0) = - 成功 2. ERR_USBD_INVALID_REQ(0x00040001) = - 无效的检测模式或设备控制器仅以全速运行。

表 199. USBD_HW_API 类结构 *（续）*

成员	描述
ReadEP	`uint32_t(*uint32_t USBD_HW_API::ReadEP)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint8_t *pData)` 用于读取在请求端点接收到的数据的函数。 该函数由 USB 协议栈和应用层调用，以读取在请求的端点接收到的数据。 hUsb：USB 设备协议栈的句柄。 EPNum：符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 pData：指向要在其中复制数据的数据缓冲区的指针。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. EPNum = 符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 3. pData = 指向要在其中复制数据的数据缓冲区的指针。 返回： 返回复制到缓冲区的字节数。
ReadReqEP	`uint32_t(*uint32_t USBD_HW_API::ReadReqEP)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint8_t *pData, uint32_t len)` 用于为指定端点的读请求排队的函数。 该函数由 USB 协议栈和应用层调用，为指定端点的读请求排队。 hUsb：USB 设备协议栈的句柄。 EPNum：符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 pData：指向要在其中复制数据的数据缓冲区的指针。该缓冲地址应可由 USB DMA 主机访问。 len：已传递缓冲区的长度。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. EPNum = 符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 3. pData = 指向要在其中复制数据的数据缓冲区的指针。该缓冲区地址应可由 USB DMA 主机访问。 4. len = 已传递缓冲区的长度。 返回： 返回所请求的缓冲区的长度。
ReadSetupPkt	`uint32_t(*uint32_t USBD_HW_API::ReadSetupPkt)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint32_t *pData)` 用于读取在请求端点接收到的设置数据包数据的函数。 该函数由 USB 协议栈和应用层调用，以读取在请求的端点接收到的设置数据包数据。 hUsb：USB 设备协议栈的句柄。 EPNum：符合 USB 规范的端点号。例如， EP0_IN 由 0x80 编号表示。 pData：指向要在其中复制数据的数据缓冲区的指针。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. EPNum = 符合 USB 规范的端点号。例如， EP0_IN 由 0x80 编号表示。 3. pData = 指向要在其中复制数据的数据缓冲区的指针。 返回： 返回复制到缓冲区的字节数。

表 199. USBD_HW_API 类结构 *（续）*

成员	描述
WriteEP	`uint32_t(*uint32_t USBD_HW_API::WriteEP)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint8_t *pData, uint32_t cnt)` 用于将要发送的数据写到请求端点的函数。 该函数由 USB 协议栈和应用层调用，以发送数据到请求端点。 hUsb：USB 设备协议栈的句柄。 EPNum：符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 pData：指向要在其中复制数据的数据缓冲区的指针。 cnt：要写入的字节数。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. EPNum = 符合 USB 规范的端点号。例如， EP1_IN 由 0x81 编号表示。 3. pData = 指向要从其中复制数据的数据缓冲区的指针。 4. cnt = 要写入的字节数。 返回： 返回写入的字节数。
WakeUp	`void(*void USBD_HW_API::WakeUp)(USBD_HANDLE_T hUsb)` 用于在主机上产生用于远程主机唤醒的恢复信号的函数。 该函数由应用层调用，在系统处于挂起状态时远程唤醒主机控制器。应用应在配置描述符的 bmAttributes 中设置 USB_CONFIG_REMOTE_WAKEUP，指示该远程唤醒功能。并且，仅当主机在挂起总线前通过发送 SET_FEATURE 请求使能了 USB_FEATURE_REMOTE_WAKEUP 时，该例程才会产生恢复信号。 hUsb：USB 设备协议栈的句柄。 参数： 1. hUsb = USB 设备协议栈的句柄。 返回： 无任何内容。
EnableEvent	`ErrorCode_t(*ErrorCode_t(* USBD_HW_API::EnableEvent)(USBD_HANDLE_T hUsb, uint32_t EPNum, uint32_t event_type, uint32_t enable))(USBD_HANDLE_T hUsb, uint32_t EPNum, uint32_t event_type, uint32_t enable)`

11.5.35 USBD_MSC_API

MSC 类 API 函数结构。该模块公开直接与 USB 设备控制器硬件交互的函数。

表 200. USBD_MSC_API 类结构

成员	描述
GetMemSize	`uint32_t(*uint32_t USBD_MSC_API::GetMemSize)(USBD_MSC_INIT_PARAM_T *param)` 用于确定 MSC 函数驱动器模块所要求的存储器的函数。 在调用 pUsbApi->msc->Init() 前，应用层调用该函数以分配 MSC 函数驱动器模块所使用的存储器。应用程序应分配可由 USB 控制器 /DMA 控制器访问的存储器。 **注：**一些存储器区域无法由所有总线主机访问。 参数： 1. param = 含 MSC 函数驱动器模块初始化参数的结构。 返回： 返回需要的存储器大小 （字节）。
init	`ErrorCode_t(*ErrorCode_t USBD_MSC_API::init)(USBD_HANDLE_T hUsb, USBD_MSC_INIT_PARAM_T *param)` 初始化 MSC 函数驱动器模块的函数。 应用层调用该函数以初始化 MSC 函数驱动器模块。 hUsb：USB 设备协议栈的句柄。 param：含 MSC 函数驱动器模块初始化参数的结构。 参数： 1. hUsb = USB 设备协议栈的句柄。 2. param = 含 MSC 函数驱动器模块初始化参数的结构。 返回： 返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功 2. ERR_USBD_BAD_MEM_BUF = 传递的存储器缓冲区不以 4 字节对齐或小于要求。 3. ERR_API_INVALID_PARAM2 = MSC_Write()、 MSC_Read() 或 MSC_Verify() 回调未定义。 4. ERR_USBD_BAD_INTF_DESC = 传递错误的接口描述符。 5. ERR_USBD_BAD_EP_DESC = 传递错误的端点描述符。

11.5.36 USBD_MSC_INIT_PARAM

海量存储设备类函数驱动器初始化参数数据结构。

表 201. USBD_MSC_INIT_PARAM 类结构

成员	描述
mem_base	`uint32_tuint32_t USBD_MSC_INIT_PARAM::mem_base` 协议栈从中分配数据和缓冲的基本存储器位置。 **注：**此字段中设置的存储器地址应可由 USB DMA 控制器访问。该值还应在 4 字节边界对齐。
mem_size	`uint32_tuint32_t USBD_MSC_INIT_PARAM::mem_size` 协议栈可使用的存储器缓冲区大小。 **注：**mem_size 应大于 USBD_MSC_API::GetMemSize() 例程返回的大小。
InquiryStr	`uint8_t *uint8_t* USBD_MSC_INIT_PARAM::InquiryStr` 指向 28 字符的字符串的指针。发送该字符串以响应 SCSI 查询命令。 **注：**指针指向的数据应属于全局范围。
BlockCount	`uint32_tuint32_t USBD_MSC_INIT_PARAM::BlockCount` 海量存储设备上存在的模块数量。
BlockSize	`uint32_tuint32_t USBD_MSC_INIT_PARAM::BlockSize` 模块大小 （字节数）
MemorySize	`uint32_tuint32_t USBD_MSC_INIT_PARAM::MemorySize` 存储器大小 （字节数）
intf_desc	`uint8_t *uint8_t* USBD_MSC_INIT_PARAM::intf_desc` 指向描述符数组内的接口描述符的指针 （
MSC_Write	`void(*void(* USBD_MSC_INIT_PARAM::MSC_Write)(uint32_t offset, uint8_t **src, uint32_t length))(uint32_t offset, uint8_t **src, uint32_t length)` MSC 写回调函数。 该函数由应用软件提供。主机发送写命令时调用此函数。 offset：目标起始地址。src：指向数据源指针的指针。指向指针的指针用于实施 0 复制缓冲区。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 length：要写入的字节数。 参数： 1. offset = 目标起始地址。 2. src = 指向数据源指针的指针。指向指针的指针用于实施 0 复制缓冲区。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 3. length = 要写入的字节数。 返回： 无任何内容。

表 201. USBD_MSC_INIT_PARAM 类结构 *（续）*

成员	描述
MSC_Read	`void(*void(* USBD_MSC_INIT_PARAM::MSC_Read)(uint32_t offset, uint8_t **dst, uint32_t length))(uint32_t offset, uint8_t **dst, uint32_t length)` MSC 读回调函数。 该函数由应用软件提供。主机发送读命令时调用此函数。 offset：源起始地址。 dst：指向数据源指针的指针。协议栈中实施的 MSC 函数驱动器以 0 复制模型写入。表示协议栈从 USB 硬件 FIFO 写入 / 读取数据前，无需另外复制缓冲区。因此，参数是指向含地址缓冲区的指针的指针 (uint8_t** dst)。因此，用户应用可更新缓冲区指针，而不是将数据复制到参数指向的地址。/note 更新后的缓冲区地址应可由 USB DMA 主机访问。如果用户不想使用 0 复制模型，还应将数据复制到已传递的缓冲区指针参数指向的地址，并不得改变地址值。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 length：要读取的字节数。 参数： 1. offset = 源起始地址。 2. dst = 指向数据源指针的指针。协议栈中实施的 MSC 函数驱动器以 0 复制模型写入。表示协议栈从 USB 硬件 FIFO 写入 / 读取数据前，无需另外复制缓冲区。因此，参数是指向含地址缓冲区的指针的指针 (uint8_t** dst)。因此，用户应用可更新缓冲区指针，而不是将数据复制到参数指向的地址。 /note 更新后的缓冲区地址应可由 USB DMA 主机访问。如果用户不想使用 0 复制模型，还应将数据复制到已传递的缓冲区指针参数指向的地址，并不得改变地址值。有关 0 复制概念的更多详情，请参见 0 复制数据传输模型。 3. length = 要读取的字节数。 返回： 无任何内容。
MSC_Verify	`ErrorCode_t(*ErrorCode_t(* USBD_MSC_INIT_PARAM::MSC_Verify)(uint32_t offset, uint8_t buf[], uint32_t length))(uint32_t offset, uint8_t buf[], uint32_t length)` MSC 验证回调函数。 该函数由应用软件提供。主机发送验证命令时调用此函数。回调函数应将缓冲区与所请求偏移量的目标存储器进行比较 offset：目标起始地址。 buf：含主机发送数据的缓冲区。 length：要验证的字节数。 参数： 1. offset = 目标起始地址。 2. buf = 含主机发送数据的缓冲区。 3. length = 要验证的字节数。 返回： 返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 如果缓冲区的数据匹配目标数据 2. ERR_FAILED = 至少有一个字节不同。

表 201. USBD_MSC_INIT_PARAM 类结构 *（续）*

成员	描述
MSC_GetWriteBuf	`void(*void(* USBD_MSC_INIT_PARAM::MSC_GetWriteBuf)(uint32_t offset, uint8_t **buff_adr, uint32_t length))(uint32_t offset, uint8_t **buff_adr, uint32_t length)` 用于优化 MSC_Write 缓冲传输的可选回调函数。 该函数由应用软件提供。主机发送 SCSI_WRITE10/SCSI_WRITE12 命令时调用此函数。回调函数应更新 offset：目标起始地址。 buf：含主机发送数据的缓冲区。 length：要写入的字节数。 参数： 1. offset = 目标起始地址。 2. buf = 含主机发送数据的缓冲区。 3. length = 要写入的字节数。 返回： 无任何内容。
MSC_Ep0_Hdlr	`ErrorCode_t(*ErrorCode_t(* USBD_MSC_INIT_PARAM::MSC_Ep0_Hdlr)(USBD_HANDLE_T hUsb, void *data, uint32_t event))(USBD_HANDLE_T hUsb, void *data, uint32_t event)` 代替默认 MSC 类处理器可选用户可覆盖函数。 应用软件提供处理器函数地址作为参数结构的此数据成员，从而用自己的处理器覆盖默认 EP0 类处理器。首选默认处理器的应用应在调用 USBD_MSC_API::Init() 前将该数据成员置零。 **注：** 参数： 1. hUsb = USB 设备协议栈的句柄。 2. data = 指向协议栈调用回调函数时所传递数据的指针。 3. event = 端点事件的类型。更多详情请参见 USBD_EVENT_T。 返回： 回调应返回 ErrorCode_t 类型，以指示成功或错误状态。 返回值： 1. LPC_OK = 成功。 2. ERR_USBD_UNHANDLED = 事件未处理，因此将事件传递到行中的下一个。 3. ERR_USBD_xxx = 针对其他错误状态。

12.1 本章导读

所有 LPC1315/16/17/45/46/47 器件上均配有 USART 控制器。

12.2 基本配置

USART 按如下方式配置：

- 通过 SYSAHBCLKCTRL 寄存器 （参见表 19）使能 USART 模块。
- USART波特率生成器所用的外设USART时钟(PCLK)由UARTCLKDIV寄存器（参见表 21）控制。

12.3 特性

- 16 字节接收和发送 FIFO。
- 寄存器位置符合 550 工业标准。
- 接收器 FIFO 触发点在 1、 4、 8 和 14 字节。
- 内置波特率发生器。
- 软件或硬件流控制。
- RS-485/EIA-485 9 位模式支持输出使能。
- $\overline{\text{RTS}}$/$\overline{\text{CTS}}$ 流控制和其他调制解调器控制信号。
- 1X- 时钟发送或接收。
- 符合 ISO 7816-3 标准的智能卡接口。
- IrDA 支持。

12.4 引脚说明

以下部分引脚不适用某些封装。

表 202. USART 引脚描述

引脚	类型	描述
RXD	输入	串行输入。串行接收数据。
TXD	输出	串行输出。串行发送数据 （智能卡模式下的输入 / 输出）。
$\overline{\text{RTS}}$	输出	请求发送。 RS-485 方向控制引脚。
CTS	输入	准许发送。
DTR	输出	数据终端就绪。
DSR	输入	数据设置就绪。
DCD	输入	数据载波检测。
RI	输入	振铃指示器。
SCLK	I/O	串行时钟。

12.5 寄存器描述

USART 所包含的寄存器，其结构如表 203 所示。除数锁存器访问位 (DLAB) 包含在 LCR[7] 中，可实现对除数锁存器的访问。

表 203 中未列出的偏移 / 地址保留。

表 203. 寄存器简介：USART （基址：0x4000 8000）

名称	访问类型	地址偏移	描述	复位值 [1]	参考
RBR	RO	0x000	接收缓冲寄存器。包含下一个待读的已接收字符。(DLAB=0)	不适用	表 204
THR	WO	0x000	发送保持寄存器。下一个待发送字符写入在此。(DLAB=0)	不适用	表 205
DLL	R/W	0x000	除数锁存 LSB。波特率除数值的最低有效字节。整除数用于从小数分频器产生波特率。 (DLAB=1)	0x01	表 206
DLM	R/W	0x004	除数锁存 MSB。波特率除数值的最高有效字节。整除数用于从小数分频器产生波特率。 (DLAB=1)	0	表 207
IER	R/W	0x004	中断使能寄存器。包含 7 个潜在 USART 中断的各个中断使能位。 (DLAB=0)	0	表 208
IIR	RO	0x008	中断 ID 寄存器，识别要挂起的中断。	0x01	表 209
FCR	WO	0x008	FIFO 控制寄存器，控制 USART FIFO 的使用和模式。	0	表 211
LCR	R/W	0x00C	线路控制寄存器。包含对帧格式化和断点产生的控制。	0	表 212
MCR	R/W	0x010	调制解调器控制寄存器。	0	表 213
LSR	RO	0x014	线路状态寄存器，包含发送和接收状态（包括线路错误）的标志。	0x60	表 214
MSR	RO	0x018	调制解调器状态寄存器。	0	表 216
SCR	R/W	0x01C	暂存寄存器，软件的八位临时存储器。	0	表 217
ACR	R/W	0x020	自动波特率控制寄存器，包含自动波特率功能的控制。	0	表 218
ICR	R/W	0x024	IrDA 控制寄存器。使能和配置 IrDA （远程控制）模式。	0	表 219
FDR	R/W	0x028	小数分频寄存器，为波特率分频器生成时钟输入。	0x10	表 221
OSR	R/W	0x02C	过采样寄存器。控制每个位时间内的过采样度。	0xF0	表 223
TER	R/W	0x030	发送使能寄存器。关闭 USART 发送器，以使用软件流控制。	0x80	表 224
HDEN	R/W	0x040	半双工使能寄存器。	0	表 225
SCICTRL	R/W	0x048	智能卡接口控制寄存器。使能和配置智能卡接口功能。	0	表 226
RS485CTRL	R/W	0x04C	RS-485/EIA-485 控制，包含对配置 RS-485/EIA-485 模式各个方面的控制。	0	表 227
RS485ADRMATCH	R/W	0x050	RS-485/EIA-485 地址匹配，包含 RS-485/EIA-485 模式的地址匹配值。	0	表 228
RS485DLY	R/W	0x054	RS-485/EIA-485 方向控制延迟。	0	表 229
SYNCCTRL	R/W	0x058	同步模式控制寄存器。	0	表 230

[1] 复位值仅反映用到的位中保存的数据，不包括保留位的内容。

12.5.1 USART 接收器缓冲寄存器 （当 DLAB = 0 时，只读）

RBR 是 USART RX FIFO 的最高字节。RX FIFO 最高字节包含了接收到的最旧的字符，可以通过总线接口读取。 LSB （位 0）包含最先接收到的数据位。如果接收到的字符少于 8 位，则未使用的 MSB 用 0 填充。

要访问 RBR， LCR 中的除数锁存器访问位 (DLAB) 必须为零。 RBR 始终为 “只读”。

由于奇偶错误 (PE)、帧错误 (FE) 和间隔中断 (BI) 位 （参见表 215）与 RBR FIFO 顶部的字节 （即下次从 RBR 读取时要读取的字节）相对应，因此，要正确提取有效的接收字节对及其状态位，应先读取 LSR 寄存器的内容，然后再读取 RBR 中的字节。

表 204. USART 接收器缓冲寄存器（当 DLAB = 0 时，只读）（RBR - 地址 0x4000 8000）位描述

位	符号	描述	复位值
7:0	RBR	USART 接收缓冲寄存器包含在 USART RX FIFO 中最早收到的字节。	未定义
31:8	-	保留	-

12.5.2 USART 发送器保持寄存器 （当 DLAB = 0 时，只写）

THR 是 USART TX FIFO 的最高字节。最高字节是 TX FIFO 中最新的字符，可通过总线接口写入。 LSB 代表第一个要发送的位。

要访问 THR， LCR 中的除数锁存器访问位 (DLAB) 必须为零。 THR 始终为 “只写”。

表 205. USART 发送器保持寄存器 （当 DLAB = 0 时，只写）（THR - 地址 0x4000 8000）位描述

位	符号	描述	复位值
7:0	THR	写入 USART 发送保持寄存器的数据将存储在 USART 发送 FIFO 中。字节将在其变为 FIFO 最旧字节且发送器可用时发出。	不适用
31:8	-	保留	-

12.5.3 USART 除数锁存器 LSB 和 MSB 寄存器 （当 DLAB = 1 时）

USART 除数锁存器是 USART 波特率发生器的一部分，持有所用的值（可选择包含小数分频器），以对 UART_PCLK 时钟进行分频，产生必须是过采样寄存器所指定的所需波特率倍数（通常为 16X）的波特率时钟。 DLL 和 DLM 寄存器共同组成一个 16 位分频器。 DLL 包含分频器的低 8 位， DLM 包含分频器的高 8 位。0 值被当作 0x0001 处理。LCR 中的除数锁存器访问位 (DLAB) 必须为 1，才可访问 USART 除数锁存器。有关如何为 DLL 和 DLM 选择正确值的详细信息，请参阅第 12.5.14 节。

表 206. USART 除数锁存器 LSB 寄存器 （当 DLAB = 1 时）（DLL - 地址 0x4000 8000）位描述

位	符号	描述	复位值
7:0	DLLSB	USART 除数锁存器 LSB 寄存器与 DLM 寄存器共同决定 USART 的波特率。	0x01
31:8	-	保留	-

表 207. USART 除数锁存器 MSB 寄存器 （当 DLAB = 1 时）（DLM - 地址 0x4000 8004）位描述

位	符号	描述	复位值
7:0	DLMSB	USART 除数锁存器 MSB 寄存器与 DLL 寄存器共同决定 USART 的波特率。	0x00
31:8	-	保留	-

12.5.4 USART 中断使能寄存器 （当 DLAB = 0 时）

IER 用于使能不同 USART 中断源。

表 208. USART 中断使能寄存器 （当 DLAB = 0 时）（IER - 地址 0x4000 8004）位描述

位	符号	值	描述	复位值
0	RBRINTEN		RBR 中断使能。使能接收数据可用中断。也控制字符接收超时中断。	0
		0	禁用 RDA 中断。	
		1	使能 RDA 中断。	
1	THREINTEN		THRE 中断使能。使能 THRE 中断。此中断的状态可从 LSR[5] 读取。	0
		0	禁用 THRE 中断。	
		1	使能 THRE 中断。	
2	RLSINTEN		使能接收线路状态中断。此中断的状态可从 LSR[4:1] 读取。	0
		0	禁用 RLS 中断。	
		1	使能 RLS 中断。	
3	MSINTEN		使能调制解调器状态中断。此中断的组成可从 MSR 读取。	0
		0	禁用 MS 中断。	
		1	使能 MS 中断。	
7:4	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
8	ABEOINTEN		使能自动波特率结束中断。	0
		0	禁用自动波特率结束中断。	
		1	使能自动波特率结束中断。	
9	ABTOINTEN		使能自动波特率超时中断。	0
		0	禁用自动波特率超时中断。	
		1	使能自动波特率超时中断。	
31:10	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

12.5.5 USART 中断识别寄存器 （只读）

IIR 提供一个状态代码，表示挂起中断的优先级和源。这些中断在 IIR 访问期间被冻结。如果在 IIR 访问期间出现中断，则为下次 IIR 访问记录该中断。

表 209. USART 中断识别寄存器 （只读）（IIR - 地址 0x4004 8008）位描述

位	符号	值	描述	复位值
0	INTSTATUS		中断状态。请注意， IIR[0] 低电平激活。挂起中断可通过评估 IIR[3:1] 来确定。	1
		0	至少一个中断挂起。	
		1	没有中断挂起。	
3:1	INTID		中断识别。IER[3:1] 识别与 USART Rx FIFO 对应的中断。下面未列出的 IER[3:1] 的其他值均为保留值。	0
		0x3	1 - 接收线路状态 (RLS)。	
		0x2	2a - 接收数据可用 (RDA)。	
		0x6	2b - 字符超时指示 (CTI)。	
		0x1	3 - THRE 中断。	
		0x0	4 - 调制解调器状态	
5:4	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
7:6	FIFOEN		这些位相当于 FCR[0]。	0
8	ABEOINT		自动波特率结束中断。自动波特率成功完成且中断使能时为真。	0
9	ABTOINT		自动波特率超时中断。自动波特率超时且中断使能时为真。	0
31:10	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

位 IIR[9:8] 由自动波特率功能设置，并发出超时信号或自动波特率条件结束信号。通过设置自动波特率控制寄存器中的相应 “清除”位，清除自动波特率中断条件。

如果 IntStatus 位为 1 且没有中断挂起，则 IntId 位将为零。如果 IntStatus 为 0，表示有一个非自动波特率中断被挂起，此时 IntId 位会确定中断的类型和处理方式，如表 210 中所述。如果指定 IIR[3:0] 的状态，中断处理程序就能确定中断原因以及如何清除有效的中断。必须读取 IIR，才能在退出中断服务例程前清除中断。

USART RLS 中断 (IIR[3:1] = 011) 为最高优先级中断，每当在 USART RX 输入端发生下面 4 个错误状况中的任意一个时，都可以进行设置：溢出错误 (OE)、奇偶校验错误 (PE)、帧处理错误 (FE) 和中止中断 (BI)。设置中断的 USART Rx 错误状况可通过 LSR[4:1] 查看。该中断在读取 LSR 时被清除。

USART RDA 中断 (IIR[3:1] = 010) 与 CTI 中断 (IIR[3:1] = 110) 均为第二优先级。 RDA 在 USART Rx FIFO 达到 FCR7:6 中定义的触发级别时激活，并在 USART Rx FIFO 降低到触发级别以下时复位。 RDA 中断激活时， CPU 可读取触发级别定义的数据块。

CTI 中断 (IIR[3:1] = 110) 为第二优先级中断，在 USART RX FIFO 含有至少一个字符并且在接收到 3.5 到 4.5 个字符的时间内没有发生任何 USART RX FIFO 操作时，可对该中断进行设置。任何 USART Rx FIFO 操作 （读或写 USART RSR）都将清除中断。此中断的目的是在收到一条大小不是触发级别大小的倍数的消息后，刷新 USART RBR。例如，如果要发送一条 105 个字符的消息，并且触发级别为 10 个字符，则 CPU 将接收 10 次 RDA 中断以传输100个字符，还将接收1到5次CTI中断（取决于服务例程）以传输剩余的5个字符。

表 210. USART 中断处理

IIR[3:0]值[1]	优先级	中断类型	中断源	中断复位
0001	-	无	无	-
0110	最高	RX 线路状态 / 错误	OE[2]、 PE[2]、 FE[2] 或 BI[2]	LSR 读操作[2]
0100	第二	RX 数据可用	Rx 数据可用或触发级别达到 FIFO (FCR0=1)	RBR 读操作[3]或USART FIFO低于触发级别
1100	第二	字符超时指示	RX FIFO 中至少有一个字符且一段时间内无字符输入或者无字符删除，取决于 FIFO 中有多少字符和触发级别的设置 （3.5 至 4.5 字符时间）。 精确时间为： [（字长度）× 7 - 2] × 8 + [（触发电平 - 字符数）× 8 + 1] RCLKs	RBR 读操作[3]
0010	第三	THRE	THRE[2]	IIR 读操作[4]（如果是中断源）或 THR 写操作
0000	第四	调制解调器状态	CTS、 DSR、 RI 或 DCD。	MSR 读取

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

[2] 有关详情，请参阅 第 12.5.9 节 “USART 线路状态寄存器 （只读）”

[3] 有关详情，请参阅 第 12.5.1 节 “USART 接收器缓冲寄存器 （当 DLAB = 0 时，只读）”

[4] 有关详情，请参阅 第 12.5.5 节 “USART 中断识别寄存器 （只读）” 和 第 12.5.2 节 “USART 发送器保持寄存器 （当 DLAB = 0 时，只写）”

USART THRE 中断 (IIR[3:1] = 001) 为第三优先级中断，当 USART THR FIFO 为空，且满足特定的初始化条件时，该中断激活。这些初始化条件的目的是为 USART THR FIFO 提供填充数据的机会，以消除在系统启动时出现的许多 THRE 中断。当 THRE = 1 时，且在上次的 THRE = 1 事件后， THR 中没有出现至少两个字符时，这些初始化条件就会实现一个字符减去停止位的延时。提供此延迟的目的是让CPU有时间将数据写入THR，而无需THRE中断来进行解码和服务。如果 USART THR FIFO 曾一次持有两个或多个字符，并且 THR 当前被清空，则 THRE 中断立即设置。当发生 THR 写操作或 IIR 读操作，且 THRE 为最高优先级中断 (IIR[3:1] = 001) 时， THRE 中断复位。

调制解调器状态中断 (IIR3:1 = 000) 是最低优先级 USART 中断，每当 CTS、DCD 或 DSR 或者 RI 引脚后沿的状态发生变化时被激活。调制解调器中断源可以在 MSR3:0 中读取。读取 MSR 将清除调制解调器中断。

12.5.6 USART FIFO 控制寄存器 （只写）

FCR 控制 USART RX 和 TX FIFO 的操作。

表 211. USART FIFO 控制寄存器 （只写）（FCR - 地址 0x4000 8008）位描述

位	符号	值	描述	复位值
0	FIFOEN		FIFO 使能	0
		0	USART FIFO 禁用，不得用于应用程序。	
		1	为 USART Rx 和 TX FIFO 以及 FCR[7:1] 访问使能有效高电平。必须设置此位，USART 才能正确操作。此位上的任何跳变都将自动清除 USART FIFO。	
1	RXFIFORES		RX FIFO 复位	0
		0	对任一 USART FIFO 都没有影响。	
		1	将逻辑 1 写入 FCR[1] 将清除 USART Rx FIFO 中的所有字节，复位指针逻辑。此位是自清除的。	
2	TXFIFORES		TX FIFO 复位	0
		0	对任一 USART FIFO 都没有影响。	
		1	将逻辑 1 写入 FCR[2] 将清除 USART TX FIFO 中的所有字节，复位指针逻辑。此位是自清除的。	
3	-	-	保留	0
5:4	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
7:6	RXTL		RX 触发级别。这两个位确定在激活中断前， FIFO 必须接收多少 USART FIFO 字符。	0
		0x0	触发级别 0 （1 个字符或 0x01）。	
		0x1	触发级别 1 （4 个字符或 0x04）。	
		0x2	触发级别 2 （8 个字符或 0x08）。	
		0x3	触发级别 3 （14 个字符或 0x0E）。	
31:8	-	-	保留	-

12.5.7 USART 线路控制寄存器

LCR 确定要发送或接收的数据字符的格式。

表 212. USART 线路控制寄存器 （LCR - 地址 0x4000 800C）位描述

位	符号	值	描述	复位值
1:0	WLS		字长度选择	0
		0x0	5 位字符长度。	
		0x1	6 位字符长度。	
		0x2	7 位字符长度。	
		0x3	8 位字符长度。	
2	SBS		停止位选择	0
		0	1 停止位。	
		1	2 停止位 （如果 LCR[1:0]=00，则为 1.5）。	
3	PE		奇偶校验使能	0
		0	禁用奇偶校验生成和检查。	
		1	使能奇偶校验生成和检查。	

表 212. USART 线路控制寄存器 （LCR - 地址 0x4000 800C）位描述 *（续）*

位	符号	值	描述	复位值
5:4	PS		奇偶检验选择	0
		0x0	奇数校验。已发送字符中 1 的数量，附加的奇偶校验位将为奇数。	
		0x1	偶数校验。已发送字符中 1 的数量，附加的奇偶校验位将为偶数。	
		0x2	强制 1 奇偶校验。	
		0x3	强制 0 奇偶校验。	
6	BC		断点控制	0
		0	禁用断点传输。	
		1	使能断点传输。LCR[6] 为高电平有效时，输出引脚 USART TXD 强制为逻辑 0。	
7	DLAB		除数锁存器访问位	0
		0	禁用对除数锁存的访问。	
		1	使能对除数锁存的访问。	
31:8	-	-	保留	-

12.5.8 USART 调制解调器控制寄存器

MCR 可使能调制解调器环回模式，并控制调制解调器输出信号。

表 213. USART 调制解调器控制寄存器 （MCR - 地址 0x4000 8010）位描述

位	符号	值	描述	复位值
0	DTRCTRL		调制解调器输出引脚 $\overline{\text{DTR}}$ 的源。当调制解调器环回模式激活时，该位读为 0。	0
1	RTSCTRL		调制解调器输出引脚 $\overline{\text{RTS}}$ 的源。当调制解调器环回模式激活时，该位读为 0。	0
3:2	-		保留，用户软件不应向保留位写入1。未定义从保留位读取的值。	0
4	LMS		环回模式选择。调制解调器环回模式提供了执行诊断环回测试的机制。发送器中的串行数据在内部连接至接收器的串行输入。输入引脚 RXD 对环回无任何影响，而输出引脚 TXD 保持在标记状态。DSR、CTS、DCD 和 RI 引脚均被忽略。从外部看来，DTR 和 $\overline{\text{RTS}}$ 被设置为无效。从内部看来， MSR 的高 4 位由 MCR 的低 4 位驱动。这样在环回模式下，写 MCR 的低 4 位就有可能生成调制解调器状态中断。	0
		0	禁用调制解调器环回模式。	
		1	使能调制解调器环回模式。	
5	-		保留，用户软件不应向保留位写入1。未定义从保留位读取的值。	0
6	RTSEN		RTS 使能	0
		0	禁用 auto-rts 流量控制。	
		1	使能 auto-rts 流量控制。	
7	CTSEN		CTS 使能	0
		0	禁用 auto-cts 流量控制。	
		1	使能 auto-cts 流量控制。	
31:8	-	-	保留	-

12.5.8.1 自动流控制

如果自动 RTS 模式被使能，则 USART 接收器 FIFO 硬件可控制 USART 的 $\overline{\text{RTS}}$ 输出。如果自动 CTS 模式被使能，则只有在 $\overline{\text{CTS}}$ 引脚为低电平时， USART 发送器才开始发送。

12.5.8.1.1 自动 RTS

通过设置 RTSen 位可以使能自动 RTS 功能。自动 RTS 数据流控制在 RBR 模块中产生，并链接至已编程好的接收器 FIFO 触发电平。如果自动 RTS 使能，则按照以下方式对数据流进行控制：

当接收器 FIFO 的电平达到已编程好的触发电平时， $\overline{\text{RTS}}$ 失效 （变为高电平值） 。发送 USART 在达到触发电平后，可能会发送一个额外字节（假设发送 USART 有额外字节要发送），因为它可能直到开始发送额外字节后才知道 $\overline{\text{RTS}}$ 取消。一旦接收器 FIFO 达到先前的触发电平，$\overline{\text{RTS}}$ 就会自动重新生效（变为低电平值）。$\overline{\text{RTS}}$ 重新断言将知会发送 USART 继续发送数据。

如果自动 RTS 模式被禁用，则 RTSen 位可控制 USART 的 $\overline{\text{RTS}}$ 输出。如果自动 RTS 模式被使能，则硬件可控制 $\overline{\text{RTS}}$ 输出，而且可以将 $\overline{\text{RTS}}$ 的实际值复制到 USART 的 RTS 控制位。如果自动 RTS 使能，则只有软件可对 RTS 控制位的值进行只读操作。

示例：假设 USART 在类型 ‘550 模式下操作，将 FCR 中的触发电平设置为 0x2，那么如果自动 RTS 被使能，接收 FIFO 只要包含 8 个字节（第 189 的表 211），USART 就会取消 $\overline{\text{RTS}}$ 输出。只要接收 FIFO 达到先前的触发电平， $\overline{\text{RTS}}$ 输出就会重新生效：4 字节。

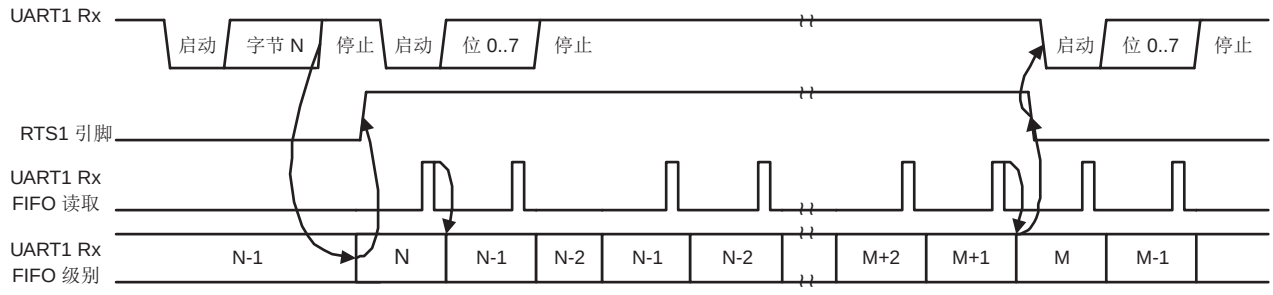


图 16. 自动 RTS 功能时序

12.5.8.1.2 自动 CTS

通过设置 CTSen 位可以使能自动 CTS 功能。如果自动 CTS 被使能，则发送器电路将在发送下一个数据字节之前检查 $\overline{\text{CTS}}$ 输入。当 $\overline{\text{CTS}}$ 有效 （低电平）时，发送器发送下一个字节。为了让发送器停止发送后面的字节，必须在当前正在发送的最后一个停止位发送一半以前释放 $\overline{\text{CTS}}$。在自动 CTS 模式下，$\overline{\text{CTS}}$ 信号的变化不会触发调制解调器状态中断，除非对 CTS 中断使能位进行设置，不过将会设置 MSR 中的 Delta CTS 位。表 214 列出了生成调制解调器状态中断的条件。

表 214. 调制解调器状态中断生成

使能调制解调器状态中断 (IER[3])	CTSen (MCR[7])	CTS 中断使能 (IER[7])	Delta CTS (MSR[0])	Delta DCD 或后沿 RI 或 Delta DSR (MSR[3:1])	调制解调器状态中断
0	x	x	x	x	无
1	0	x	0	0	无
1	0	x	1	x	有
1	0	x	x	1	有
1	1	0	x	0	无
1	1	0	x	1	有
1	1	1	0	0	无
1	1	1	1	x	有
1	1	1	x	1	有

自动 CTS 功能通常无需 CTS 中断。当流控制使能时，$\overline{\text{CTS}}$ 状态的改变不会触发主机中断，因为器件会自动控制其发送器。若不使用自动 CTS，则发送器会将发送 FIFO 中存在的所有数据都发送出去，从而导致接收器发生溢出错误。图 17 展示了自动 CTS 功能时序。

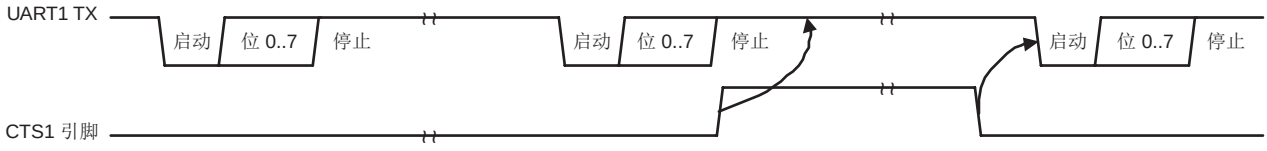


图 17. 自动 CTS 功能时序

发送第二个字符时，会取消 $\overline{\text{CTS}}$ 信号。之后不会发送第三个字符。只要取消 $\overline{\text{CTS}}$（高电平），USART 就会在 TXD 上保持 1。一旦 $\overline{\text{CTS}}$ 被断言，发送就会恢复且发送起始位，接着是下一个字符的数据位。

12.5.9 USART 线路状态寄存器 （只读）

LSR 是一个只读寄存器，提供有关 USART TX 和 RX 模块的状态信息。

表 215. USART 线路状态寄存器 （只读）（LSR - 地址 0x4000 8014）位描述

位	符号	值	描述	复位值
0	RDR		接收器数据就绪：当 RBR 包含未读字符时，LSR[0] 会被设置；当 USART RBR FIFO 为空时， LSR[0] 会被清零。	0
		0	RBR 被清空。	
		1	RBR 包含有效数据。	
1	OE		超限错误。超限错误条件在其出现时立即设置。 LSR 读取将清除 LSR[1]。在 USART RSR 汇编了新字符且 USART RBR FIFO 为满时，设置 LSR[1]。在此情况下，USART RBR FIFO 将不会被覆盖，且 USART RSR 中的字符将丢失。	0
		0	超限错误状态未激活。	
		1	超限错误状态激活。	

表 215. USART 线路状态寄存器 （只读）（LSR - 地址 0x4000 8014）位描述 *（续）*

位	符号	值	描述	复位值
2	PE		奇偶校验错误。已接收字符的奇偶校验位处于错误状态时，出现奇偶校验错误。LSR 读取将清除 LSR[2]。奇偶校验错误检测的时间取决于 FCR[0]。 **注：**奇偶检验错误与 USART RBR FIFO 顶部的字符相关。	0
		0	奇偶校验错误状态未激活。	
		1	奇偶校验错误状态激活。	
3	FE		帧处理错误。已接收字符的停止位为逻辑 0 时，出现成帧错误。LSR 读取将清除 LSR[3]。帧处理错误检测的时间取决于 FCR0。当检测到有帧处理错误时，RX 会尝试与数据重新同步，并假设错误的停止位实际是一个超前的起始位。当然，不能假定下一个收到的字节是正确的，即使没有帧处理错误。 **注：**帧处理错误与 USART RBR FIFO 顶部的字符相关。	0
		0	帧处理错误状态未激活。	
		1	帧处理错误状态激活。	
4	BI		中止中断。对于一个完整的字符发送 （开始、数据、优先级、停止）， RXD1 保持在间距状态 （全部为零）时，发生断点中断。检测到断点条件后，接收器即进入空闲，直至 RXD1 进入标记状态 （全部为 1）为止。 LSR 读取将清除此状态位。断点检测的时间取决于 FCR[0]。 **注：**中止中断与 USART RBR FIFO 顶部的字符相关。	0
		0	中止中断状态未激活。	
		1	中止中断状态激活。	
5	THRE		发送保持寄存器空。 THRE 在检测到空 USART THR 时立即设置，并在 THR 写入时清除。	1
		0	THR 包含有效数据。	
		1	THR 为空。	
6	TEMT		发送器空。TEMT 在 THR 和 TSR 都为空时设置； TEMT 在 TSR 或 THR 包含有效数据时被清除。	1
		0	THR 和 / 或 TSR 包含有效数据。	
		1	THR 和 TSR 为空。	
7	RXFE		RX FIFO 中的错误。在具有 RX 错误 （如成帧处理错误、奇偶校验错误或断点错误）的字符被加载到RBR中时，设置LSR[7]。此位在 LSR 寄存器被读取且 USART FIFO 中没有后续错误时被清除。	0
		0	RBR 不包含 USART RX 错误或 FCR[0]=0。	
		1	USART RBR 包含至少一个 USART RX 错误。	
8	TXERR		Tx 错误。在智能卡 T=0 操作期间，当智能卡用 NACK 填充发送字符 （比 TXRETRY 字段指示的次数多一次）时，设置该位。	0
31:9	-	-	保留	-

12.5.10 USART 调制解调器状态寄存器

MSR 为只读寄存器，提供 USART 输入信号的状态信息。读取该寄存器时 （之后）将清除位 0。

表 216. USART 调制解调器状态寄存器 （MSR - 地址 0x4000 8018）位描述

位	符号	值	描述	复位值
0	DCTS		Delta CTS。 当输入 CTS 的状态改变时，该位被设置。读 MSR 会清除该位。	0
		0	未检测到调制解调器输入 CTS 的状态变化。	
		1	检测到调制解调器输入 CTS 的状态变化。	
1	DDSR		Delta DSR。 发生 DSR 输入状态变化时被设置。 MSR 读操作会清除该位。	0
		0	未检测到调制解调器输入 DSR 的变化。	
		1	检测到调制解调器输入 DSR 的变化。	
2	TERI		后沿 RI。 在输入 RI 由低电平跳变为高电平时被设置。 MSR 读操作会清除该位。	0
		0	未检测到调制解调器输入 RI 的变化。	
		1	检测到 RI 上低电平至高电平的跳变。	
3	DDCD		Delta DCD。发生 DCD 输入状态变化时被设置。MSR 读操作会清除该位。	0
		0	未检测到调制解调器输入 DCD 的变化。	
		1	检测到调制解调器输入 DCD 的变化。	
4	CTS	-	准许发送状态。输入信号 CTS 的补码。在调制解调器环回模式下，该位连接到 MCR[1]。	0
5	DSR	-	数据集就绪状态。输入信号 DSR 的补值。在调制解调器环回模式下，该位连接到 MCR[0]。	0
6	RI	-	振铃指示器状态。输入 RI 的补值。在调制解调器环回模式下，该位连接到 MCR[2]。	0
7	DCD	-	数据载波检测状态。输入 DCD 的补值。在调制解调器环回模式下，该位连接到 MCR[3]。	0
31:8	-	-	保留，从保留位读取的值未定义。	不适用

12.5.11 USART 暂存寄存器

SCR 对 USART 操作没有影响。该寄存器可由用户决定写入和 / 或读取。中断接口中没有向主机指明已发生 SCR 读取或写入的规范。

表 217. USART 暂存寄存器 （SCR - 地址 0x4000 801C）位描述

位	符号	描述	复位值
7:0	PAD	一个可读、可写的字节。	0x00
31:8	-	保留	-

12.5.12 USART 自动波特率控制寄存器

USART 自动波特率控制寄存器 (ACR) 控制为波特率生成测量输入时钟 / 数据速率的过程，并且可由用户决定读取和写入。

表 218. USART 自动波特率控制寄存器 （ACR - 地址 0x4000 8020）位描述

位	符号	值	描述	复位值
0	START		自动波特率完成后此位自动清除。	0
		0	自动波特率停止 （自动波特率不运行）。	
		1	自动波特率起动 （自动波特率运行）。自动波特率运行位。自动波特率完成后此位自动清除。	
1	MODE		自动波特率模式选择位。	0
		0	模式 0。	
		1	模式 1。	
2	AUTORESTART		启动模式	0
		0	不重启	
		1	如果超时则重新启动 （计数器会在下一个 USART Rx 下降沿重新启动）	
7:3	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	0
8	ABEOINTCLR		自动波特率中断结束清除位 （仅可写访问）。	0
		0	写入 0 没有影响。	
		1	写入 1 将清除 IIR 中的相应中断。	
9	ABTOINTCLR		自动波特率超时中断清除位 （仅可写访问）。	0
		0	写入 0 没有影响。	
		1	写入 1 将清除 IIR 中的相应中断。	
31:10	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	0

12.5.12.1 自动波特率

USART 自动波特率功能可用于基于 AT 协议 （Hayes 命令）测量输入的波特率。使能后，自动波特率功能将测量接收数据流的位时间，并相应设置除数锁存寄存器 DLM 和 DLL。

自动波特率通过设置 ACR 起始位来起动。可通过清除 ACR 起始位来停止自动波特率。自动波特率完成后起始位将清除，且读取位的操作将返回自动波特率的状态 （挂起 / 完成）。

有两种自动波特率测量模式可用，可通过 ACR 模式位选择。在模式 0 中，在 USART Rx 引脚的两个后续下降沿上测量波特率 （起始位的下降沿和最低有效位的下降沿） 。在模式 1 中，在 USART Rx 引脚的下降沿和后续上升沿之间测量波特率 （起始位的长度）。

出现超时时，ACR AutoRestart 位可用于自动重启波特率测量（速率测量计数器溢出）。如果设置了此位，则将在 USART Rx 引脚的下一下降沿重启速率测量。

自动波特率功能可产生两种中断。

- IIR ABTOInt 中断将在中断使能时设置 （IER ABToIntEn 设置，并且自动波特率测量计数器溢出）。
- IIR ABEOInt 中断将在中断使能时设置（IER ABEOIntEn 设置，并且自动波特率成功完成）。

必须通过设置相应的 ACR ABTOIntClr 和 ABEOIntEn 位来清除自动波特率中断。

小数波特率发生器在自动波特率期间必须禁用 (DIVADDVAL = 0)。同样，使用自动波特率时，对 DLM 和 DLL 寄存器的任何写操作都必须在 ACR 寄存器写操作前完成。 USART 支持的最小和最大波特率是 USART_PCLK、数据位、停止位和奇偶校验位的数量的函数。

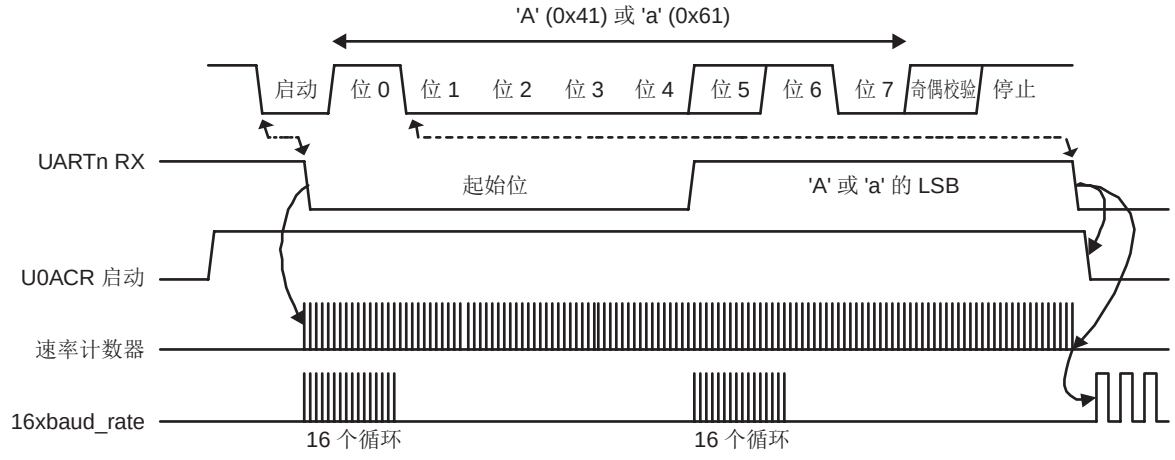
(2)

$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

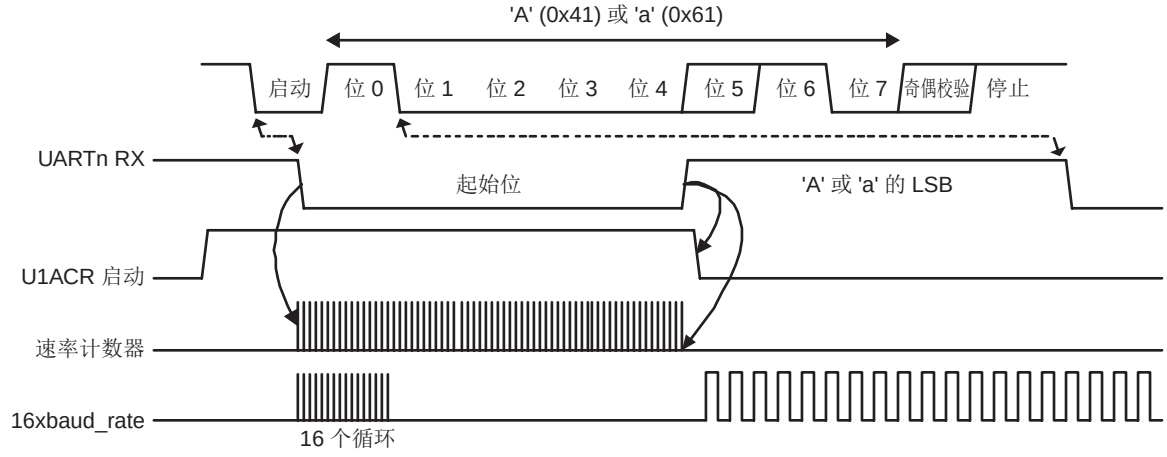
12.5.12.2 自动波特率模式

软件要使用 AT 命令时，可用预期的字符格式配置 USART 并设置 ACR 起始位。无需考虑除数锁存器 DLM 和 DLM 中的初始值。由于 A 或者 a 的 ASCII 编码 （“A” = 0x41，“a” = 0x61），USART Rx 引脚检测到起始位，且预期字符的最低有效位由两个下降沿分隔。 ACR 起始位设置后，自动波特率协议将执行以下几个阶段：

1. ACR 起始位设置时，波特率测量计数器被复位，USART RSR 也复位。RSR 波特率切换到最高速率。
2. USART Rx 引脚的下降沿触发起始位的开始。速率测量计数器将开始对 UART_PCLK 循环进行计数。
3. 接收起始位期间， RSR 波特率输入中产生 16 个脉冲，频率为 USART 输入时钟的频率，保证起始位存储在 RSR 中。
4. 在接收起始位的过程中（以及模式 = 0 下的字符 LSB），速率计数器将随着被预分频的 USART 输入时钟 (UART_PCLK) 递增。
5. 如果是模式 = 0，那么速率计数器将在 USART Rx 引脚的下一个下降沿停止。如果是模式 = 1，那么速率计数器将在 USART Rx 引脚的下一个上升沿停止。
6. 速率计数器被加载到 DLM/DLL 中，并且波特率将切换到正常操作。设置 DLM/DLL 后，自动波特率结束中断 IIR ABEOInt 将被设置 （如果使能）。 RSR 现在将继续接收字符的剩余位。



a. 模式 0（起始位和 LSB 均用于自动波特率）



b. 模式 1（仅起始位用于自动波特率）

图 18. 自动波特率 a) 模式 0 和 b) 模式 1 的波形图

12.5.13 USART IrDA 控制寄存器

IrDA 控制寄存器使能和配置 IrDA 模式。在发送或接收数据时，不应更改 ICR 的值，否则会造成数据丢失或损坏。

表 219. USART IrDA 控制寄存器（ICR – 0x4000 8024）位描述

位	符号	值	描述	复位值
0	IRDAEN		IrDA 模式使能	0
		0	禁用 IrDA 模式。	
		1	使能 IrDA 模式。	
1	IRDAINV		串行输入逆变器	0
		0	串行输入不反向。	
		1	串行输入反向。这不会影响串行输出。	
2	FIXPULSEEN		IrDA 固定脉冲宽度模式。	0
		0	IrDA 固定脉冲宽度模式禁用。	
		1	IrDA 固定脉冲宽度模式使能。	
5:3	PULSEDIV		当 FixPulseEn = 1 时，对脉冲宽度进行配置。	0
		0x0	3 /（16 × 波特率）	
		0x1	$2 \times T_{PCLK}$	
		0x2	$4 \times T_{PCLK}$	
		0x3	$8 \times T_{PCLK}$	
		0x4	$16 \times T_{PCLK}$	
		0x5	$32 \times T_{PCLK}$	
		0x6	$64 \times T_{PCLK}$	
		0x7	$128 \times T_{PCLK}$	
31:6	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	0

如果在 IrDA 模式下使用了固定脉冲宽度模式（IrDAEn = 1 且 FixPulseEn = 1），ICR 中的 PulseDiv 位用于选择脉冲宽度。用户必须对这些位的值进行设置，以便得到的脉冲宽度至少为 1.63 µs。表 220 显示了可能的脉冲宽度。

表 220. IrDA 脉冲宽度

FixPulseEn	PulseDiv	IrDA 发送器脉冲宽度 (µs)
0	x	3 /（16 × 波特率）
1	0	$2 \times T_{PCLK}$
1	1	$4 \times T_{PCLK}$
1	2	$8 \times T_{PCLK}$
1	3	$16 \times T_{PCLK}$
1	4	$32 \times T_{PCLK}$
1	5	$64 \times T_{PCLK}$
1	6	$128 \times T_{PCLK}$
1	7	$256 \times T_{PCLK}$

12.5.14 USART 小数分频寄存器

USART 小数分频器寄存器 (FDR) 控制波特率生成时钟预分频器，可由用户决定读取或写入。预换算器采用 APB 时钟并根据指定的小数要求生成输出时钟。

注意事项：如果小数分频器有效 (DIVADDVAL > 0)，且 DLM = 0，则 DLL 寄存器的值必须大于或等于 3。

表 221. USART 小数分频寄存器 （FDR - 地址 0x4000 8028）位描述

位	功能	描述	复位值
3:0	DIVADDVAL	波特率发生预换算器分频器的值。如果此字段为 0，则小数波特率发生器将不会影响 USART 波特率。	0
7:4	MULVAL	波特率预换算器乘数的值。此字段必须大于等于 1，USART 才能正常工作，无论是否使用小数波特率发生器。	1
31:8	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	0

此寄存器控制波特率发生的时钟预换算器。寄存器的复位值使 USART 的小数功能保持禁用，以确保 USART 在软件和硬件上，都与没有配备此功能的 USART 完全兼容。

USART 波特率的计算如下：

$$UART_{baudrate} = \frac{PCLK}{16 \times (256 \times U0DLM + U0DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)} \quad (3)$$

其中，UART_PCLK 是外设时钟，DLM 和 DLL 是标准 USART 波特率分频器寄存器，DIVADDVAL 和 MULVAL 是 USART 小数波特率发生器的特定参数。

MULVAL 和 DIVADDVAL 的值应符合以下条件：

1. 1 ≤ MULVAL ≤ 15
2. 0 ≤ DIVADDVAL ≤ 14
3. DIVADDVAL< MULVAL

FDR 的值在发送 / 接收数据时不能修改，否则数据可能丢失或破坏。

如果 FDR 寄存器值不符合这两项要求，则小数分频器的输出是未定义的。如果 DIVADDVAL 为零，则小数分频器禁用，时钟不会被分频。

12.5.14.1 波特率计算

操作 USART 时可使用也可不使用小数分频器。在实际应用中，使用几个不同的小数分频器设置就可能获得所需的波特率。以下算法展示了一种查找一组 DLM、DLL、MULVAL 和 DIVADDVAL 值的方法。这组参数产生的波特率相对于所需波特率的误差小于 1.1%。

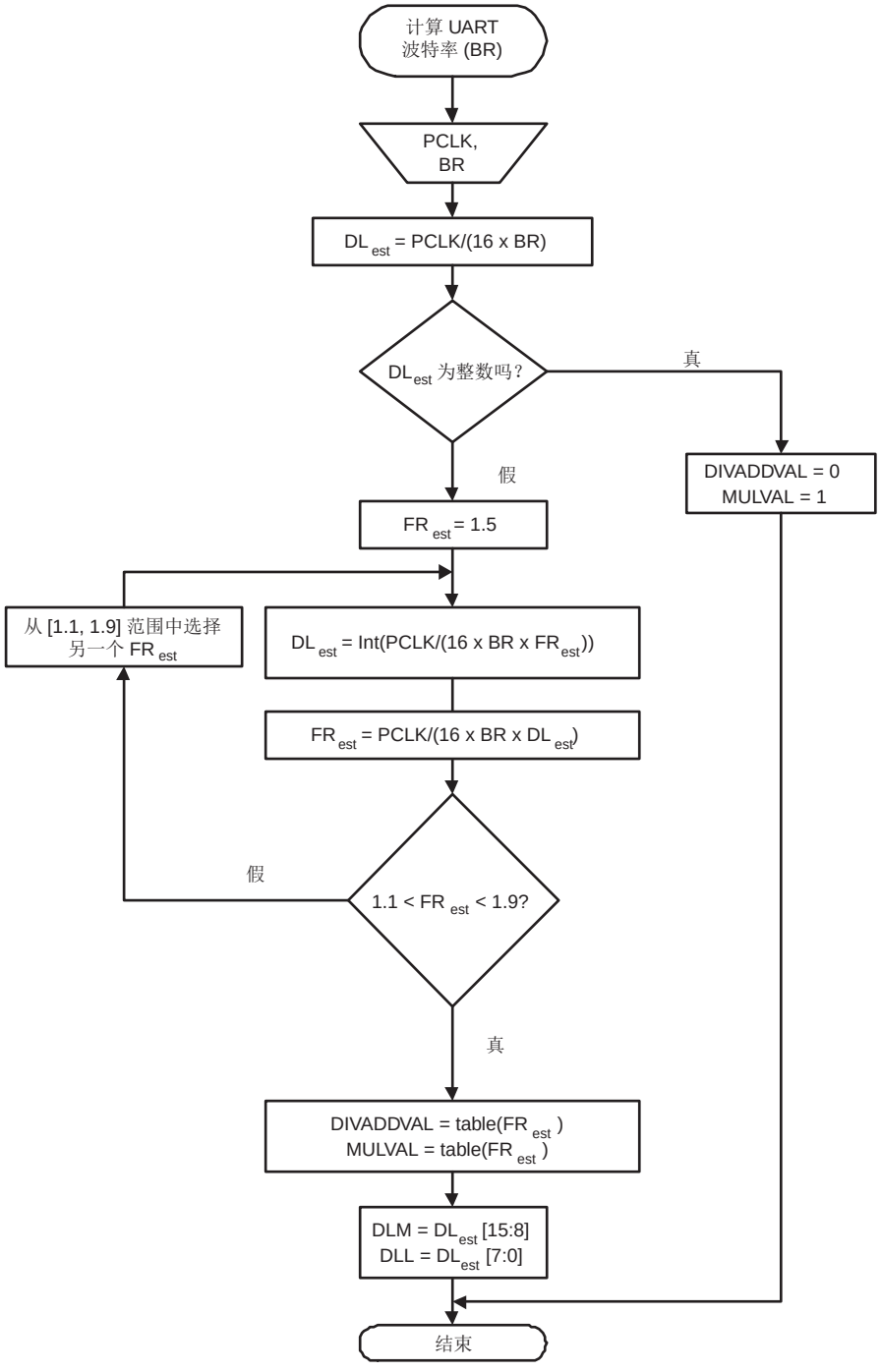


图 19. 设置 USART 分频器的算法

表 222. 小数分频器设置查找表

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

12.5.14.1.1 示例 1：UART_PCLK = 14.7456 MHz， BR = 9600

根据所提供的算法，DL_{est} = PCLK/(16 x BR) = 14.7456 MHz / (16 x 9600) = 96。因为这里的 DL_{est} 是一个整数，所以 DIVADDVAL = 0， MULVAL = 1， DLM = 0，且 DLL = 96。

12.5.14.1.2 示例 2：UART_PCLK = 12.0 MHz， BR = 115200

根据所提供的算法，DL_{est} = PCLK/(16 x BR) = 12 MHz / (16 x 115200) = 6.51。此处 DL_{est} 不是整数，下一步可以估测 FR 参数。使用 FR_{est} = 1.5 进行首次估算，得到新的 DL_{est} = 4，然后使用 FR_{est} = 1.628 再进行计算。由于 FRest = 1.628 是在 1.1 到 1.9 的指定范围之内，因此 DIVADDVAL 和 MULVAL 的值可通过附带的查找表获得。

在查找表表 222 中，最接近 FRest = 1.628 的值为 FR = 1.625。也就是说 DIVADDVAL = 5 而 MULVAL = 8。

根据这些查到的值，建议的 USART 设置应为：DLM = 0、 DLL = 4、 DIVADDVAL = 5 和 MULVAL = 8。根据方程 3， USART 的波特率为 115384。该速率与原来指定的 115200 之间存在 0.16% 的相对误差。

12.5.15 USART 过采样寄存器

在多数应用中，USART 在每个名义位时间对已接收数据进行 16 次采样，并发送 16 个输入时钟宽的位。该寄存器允许软件控制输入时钟和位时钟的比值。这是智能卡模式所必需的，并为其他模式的小数分频提供替代方式。

表 223. USART 过采样寄存器 （OSR，地址 0x4000 802C）位描述

位	符号	描述	复位值
0	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
3:1	OSFRAC	过采样率的小数部分，以输入时钟周期的八分之一为单位。(001 = 0.125, ..., 111 = 0.875)	0
7:4	OSINT	过采样率的整数部分，减去 1。复位值等于每个位时间 16 个输入时钟的正常操作模式。	0xF
14:8	FDINT	在智能卡模式中，这些位可以作为 OSint 字段的最有效扩展，支持 ISO7816-3 要求的最高 2048 的过采样率。在智能卡模式中，位 14:4 最初应设置为 371，产生 372 的过采样率。	0
31:15	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

示例：对于 24 MHz USART 时钟频率的 3.25 Mbps 波特率，理想的过采样率为 24/3.25 或 7.3846。为 7 时钟 / 位设置 OSINT 为 0110，为 0.375 时钟 / 位设置 OSFrac 为 011，得到 7.375 的过采样率。

在智能卡模式中， OSInt 由 FDINT 扩展。这将让可能的采样扩展到 2048 （符合支持 ISO 7816-3 的要求）。请注意，当 D<0 时，该值可能被超过，但不会得到 USART 支持。使能智能卡模式时， OSINT 和 FDINT 的初始值应编程为 “00101110011”（372 减去 1）。

12.5.16 USART 发送使能寄存器

除配备全硬件流控制（上述的 auto-cts 和 auto-rts 机制）以外，TER 还可实现软件流控制。TxEn = 1 时，USART 发送器将连续发送数据，只要这些数据可用。TxEn 变为 0 时，USART 发送将立即停止。

虽然表 224 描述了如何利用 TXEn 位来实现硬件流控制，但我们强烈建议用户采用 USART 硬件所实现的自动流控制特性处理硬件流控制，并限制 TXEn 对软件流控制的范围。

表 224 描述了如何利用 TXEn 位来实现软件流控制。

表 224. USART 发送使能寄存器 （TER - 地址 0x4000 8030）位描述

位	符号	描述	复位值
6:0	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
7	TXEN	该位为 1 时 （复位后），一旦先前的数据都被发送后，写入 THR 的数据就会在 TXD 引脚上输出。如果字符发送时该位被清 0，将完成该字符的发送，但是在重新设置该位之前，将不会再发送字符。也就是说，该位的 0 值阻止了 THR 或者 TX FIFO 传送字符至发送移位寄存器。当检测到硬件信号握手 TX-permit 信号 (CTS) 变为假时，或者在接收到 XOFF 字符 (DC3) 时，软件通过执行软件握手可以将该位清零。当检测到 TX-permit 信号变为真时，或者在接收到 XON (DC1) 字符时，软件可以将该位重新置位。	1
31:8	-	保留	-

12.5.17 USART 半双工使能寄存器

注：在智能卡模式或 IrDA 模式下，应禁用 HDEN 寄存器 （默认情况下，智能卡和 IrDA 在半双工模式下运行）。

复位后， USART 将处于全双工模式，意味着 TX 和 RX 都独立工作。设置 HDEN 位后，USART 将处于半双工模式。在此模式下， USART 确保接收器在空闲时锁定，或在接收完正在进行的完整字符后进入锁定状态。线路冲突必须在软件中处理。正在发送数据时，如果出现要接收的数据，则 USART 的行为是不可预测的。

因此， HDEN 寄存器的值在发送或接收数据时不能修改，否则数据可能丢失或破坏。

表 225. USART 半双工使能寄存器 （HDEN - 地址 0x4000 8040）位描述

位	符号	值	描述	复位值
0	HDEN		半双工模式使能	0
		0	禁用半双工模式。	
		1	使能半双工模式。	
31:1	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	-

12.5.18 USART 智能卡接口控制寄存器

该寄存器允许在异步智能卡应用中使用 USART。

表 226. USART 智能卡接口控制寄存器 （SCICTRL - 地址 0x4000 8048）位描述

位	符号	值	描述	复位值
0	SCIEN		智能卡接口使能。	0
		0	智能卡接口禁用。	
		1	异步半双工智能卡接口使能。	
1	NACKDIS		NACK 响应禁用。仅在 T=0 时可用。	0
		0	NACK 响应使能。	
		1	NACK 响应禁止。	
2	PROTSEL		按 ISO7816-3 标准的定义进行协议选择。	0
		0	T = 0	
		1	T = 1	
4:3	-	-	保留。	-
7:5	TXRETRY	-	协议选择 T 位（上文）为 0 时，如果远程设备发出 NACK 信号，此字段将控制 USART 尝试的最大重发数。NACK 发生过这些次数加上 1 时，设置 LSR 中的 Tx 错误位，请求中断 （若使能），并锁定 USART 直到清除 FIFO。	-
15:8	XTRAGUARD	-	协议选择 T 位（上文）为 0 时，此字段指示 USART 发送字符后，保护时间应超过名义 2 位时间的位时间数 (ETU)。此字段中的 0xFF 可能表示一个字符或 11 位时间 / 字符后只有一位。	
31:16	-	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

12.5.19 USART RS485 控制寄存器

RS485CTRL 寄存器控制 RS-485/EIA-485 模式下 USART 的配置。

表 227. USART RS485 控制寄存器 （RS485CTRL - 地址 0x4000 804C）位描述

位	符号	值	描述	复位值
0	NMMEN		NMM 使能。	0
		0	RS-485/EIA-485 正常多点模式 (NMM) 禁用。	
		1	RS-485/EIA-485 正常多点模式 (NMM) 使能。在该模式下，当收到的字节导致 USART 设置奇偶校验错误并生成中断时，检测到地址。	
1	RXDIS		接收器使能。	0
		0	接收器使能。	
		1	接收器禁用。	
2	AADEN		AAD 使能。	0
		0	自动地址检测 (AAD) 禁用。	
		1	自动地址检测 (AAD) 使能。	
3	SEL		选择方向控制引脚	0
		0	如果方向控制已使能 （位 DCTRL = 1），则引脚 $\overline{\text{RTS}}$ 用于方向控制。	
		1	如果方向控制已使能 （位 DCTRL = 1），则引脚 $\overline{\text{DTR}}$ 用于方向控制。	
4	DCTRL		自动方向控制使能。	0
		0	禁用自动方向控制。	
		1	使能自动方向控制。	
5	OINV		极性控制。该位完全改变了 $\overline{\text{RTS}}$ （或 $\overline{\text{DTR}}$）引脚上方向控制信号的极性。	0
		0	当发送器有数据要发送时，方向控制引脚会被驱动为逻辑 0。在最后一个数据位被发送后，该位就会被驱动为逻辑 1。	
		1	当发送器有数据要发送时，方向控制引脚会被驱动为逻辑 1。在最后一个数据位被发送后，该位就会被驱动为逻辑 0。	
31:6	-	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

12.5.20 USART RS-485 地址匹配寄存器

RS485ADRMATCH 寄存器包含 RS-485/EIA-485 模式的地址匹配值。

表 228. USART RS-485 地址匹配寄存器 （RS485ADRMATCH - 地址 0x4000 8050）位描述

位	符号	描述	复位值
7:0	ADRMATCH	包含地址匹配值。	0x00
31:8	-	保留	-

12.5.21 USART RS-485 延迟值寄存器

用户可通过对 8 位 RS485DLY 寄存器编程来设置：最后一个停止位离开 TXFIFO 到使 $\overline{\text{RTS}}$（或 $\overline{\text{DTR}}$）无效之间的延迟。此延迟时间以波特率时钟为周期。可以编程 0 到 255 位倍数的任何延迟时间。

表 229. USART RS-485 延迟值寄存器 （RS485DLY - 地址 0x4000 8054）位描述

位	符号	描述	复位值
7:0	DLY	包含方向控制 （RTS 或 DTR）延迟值。此寄存器与一个 8 位计数器共同工作。	0x00
31:8	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

12.5.22 USART 同步模式控制寄存器

SYNCCTRL 寄存器控制同步模式。此模式有效时， USART 在 SCLK 引脚上生成或接收位时钟并将其应用于发送和接收移位寄存器。同步模式不应与智能卡模式一起使用。

表 230. USART 同步模式控制寄存器 （SYNCCTRL - 地址 0x4000 8058）位描述

位	符号	值	描述	复位值
0	SYNC		使能同步模式。	0
		0	已禁用	
		1	使能	
1	CSRC		时钟源选择。	0
		0	同步从模式 （SCLK 进）	
		1	同步主模式 （SCLK 出）	
2	FES		下降沿采样。	0
		0	RxD 在 SCLK 的上升沿采样。	
		1	RxD 在 SCLK 的下降沿采样。	
3	TSBYPASS		同步从机模式中的发送同步旁通。	0
		0	用于时钟边沿检测逻辑前，输入时钟同步。	
		1	用于时钟边沿检测逻辑前，输入时钟不同步。这将带来更高的输入时钟速率并影响潜在的亚稳定性。	
4	CSCEN		连续主时钟使能 （仅在 CSRC 为 1 时使用）	0
		0	SCLK 仅在字符正在 TxD 上发送时循环	
		1	SCLK 连续运行（RxD 上的字符接收可独立于 TxD 上的传输）	
5	SSDIS		启动 / 停止位	0
		0	发送起动和停止位，如其他模式。	
		1	不发送起动 / 停止位。	
6	CCCLR		连续时钟清除	0
		0	CSCEN 受软件控制。	
		1	收到每个字符后硬件清除 CSCEN。	
31:7	-		保留。未定义从保留位读取的值。	不适用

复位后，同步模式禁用。通过设置 SYNC 位使能同步模式。SYNC 为 1 时，USART 按以下步骤操作：

1. CSRC 位控制 USART 在 SCLK 引脚上是发送 （主机模式）还是接收 （从机模式）串行位时钟。
2. CSRC 为 1（选择主机模式）时，CSCEN 位选择 USART 是在 SCLK 上连续生成时钟 (CSCEN=1) 还是仅当在 TxD 上发出发送数据时生成时钟 (CSCEN=0)。
3. SSDIS 位控制是否使用起始位和停止位。SSDIS 为 0 时，USART 发送起始位和停止位并进行采样，如其他模式中一样。SSDIS 为 1 时， USART 既不发送起始位或停止位，也不进行采样， SCLK 上的每个下降沿都将 RxD 上的一个数据位采样到接收移位寄存器，同时对发送移位寄存器进行移位。

本节其余内容将提供 SYNC 为 1 时的进一步操作详情。

数据从 SCLK 上的下降沿开始在 TxD 上发生变化。SSDIS 为 0 时，FES 位控制 USART 在 SCLK 的上升沿还是下降沿采样 RxD 上的串行数据。SSDIS 为 1 时，USART 将忽略 FES 并总是在 SCLK 的下降沿采样 RxD。

SYNC=1、 CSRC=1、 CSCEN=1 和 SSDIS=1 的组合是一种困难的操作模式，这是因为 SCLK 适用于两个方向的数据流，并且当 TxD 或 RxD 上存在有效数据时，没有定义的机制来向接收器发送信号。

缺少这种机制时，在与 SPI 协议相似的模式中， SSDIS=1 可与 CSCEN=0 或 CSRC=0 搭配使用。在此模式中，对于 SCLK 上的每组 8 个时钟周期，字符 （至少在概念上）会在 USART 和远程设备之间交换。此项操作可称为全双工，但同样的硬件模式可用于高层协议控制下的半双工方式，其中，每当数据要从任一方向发出， SCLK 源都将以 N 个周期为一组进行切换。（N 为位 / 字符的数量。）

LPC1315/16/17/45/46/47 USART 为时钟源时 (CSRC=1)，此类半双工操作可能引发非常看似人工的要求：向发送器保持寄存器写入虚拟字符以生成 8 个时钟，从而接收字符。 CCCLR 为编程半双工接收提供一种更加自然的方式。高层协议指示 LPC1315/16/17/45/46/47 USART 应接收字符，软件应向 SYNCCTRL 寄存器写入 CSCEN=1 和 CCCLR=1。USART 发送 N 个时钟周期并因此接收一个字符后，将清除 CSCEN 位。如果之后需要接收更多字符，软件可重复设置 CSCEN 和 CCCLR。

除了此类半双工操作外，CSCEN=1 主要和 SSDIS=0 搭配使用，这样起始位可以指示每个字符在各个方向的发送。

12.6 功能说明

12.6.1 RS-485/EIA-485 模式的操作

RS-485/EIA-485 功能可将 USART 配置为可寻址从机。可寻址从机是由单个主机控制的多个从机之一。

USART 主机发送器将设置奇偶位（第 9 位）为 1 以标识一个地址字符。对于数据字符，将奇偶校验位设置为 0。

可为每个 USART 从机接收器分配一个唯一的地址。可以对从机进行编程，使其手动或自动拒绝不是其自身地址后的数据。

RS-485/EIA-485 正常多点模式

将 RS485CTRL 位设置为 0 可使能此模式。在该模式下，当收到的字节导致 USART 设置奇偶校验错误并生成中断时，检测到地址。

如果接收器被禁用（RS485CTRL 位 1=1），任何接收到的数据字节都将被忽略，且不会存入 RXFIFO。检测到地址字节 （奇偶校验位 =1）时，会将其放入 RXFIFO，并生成一个“Rx 数据就绪中断”。处理器然后即可读取地址字节，并决定是否使能接收器来接收随后的数据。

当接收器被使能 （RS485CTRL 位 1=0），所有接收到的字节将会被接受并存入 RXFIFO，不论是数据还是地址。收到地址字符时，将生成一个奇偶校验错误中断，然后处理器会决定是否禁用接收器。

RS-485/EIA-485 自动地址检测 (AAD) 模式

RS485CTRL 寄存器位 0（9 位模式使能）和 2（AAD 模式使能）都设置时， USART 处于自动地址检测模式。

在该模式下，接收器会将任何收到的地址字节（奇偶校验 = 1）与编程到 RS485ADRMATCH 寄存器中的 8 位值进行比较。

如果接收器被禁用 （RS485CTRL 位 1=1），任何接收到的与 RS485ADRMATCH 值不匹配的数据字节或者地址字节都将被丢弃。

检测到匹配的地址字符时，会将其连同奇偶校验位一起推入 RXFIFO，然后接收器将自动使能 （RS485CTRL 位 1 将由硬件清除）。接收器也将生成一个 “Rx 数据就绪中断”。

接收器使能时 （RS485CTRL 位 1 = 0），将接受收到的所有字节并存储在 RXFIFO 中，直至收到与 RS485ADRMATCH 值不匹配的地址字节。出现这种情况时，接收器将在硬件中自动禁用（RS485CTRL 位 1 将被设置），收到的不匹配地址字符将不会存储在 RXFIFO 中。

RS-485/EIA-485 自动方向控制

RS485/EIA-485 模式包含允许发送器以方向控制输出信号的方式自动控制 DIR 引脚状态的选项。

设置 RS485CTRL 位 4=1 将使能该功能。

将 RS485CTRL 位 3 保持为 0，这样，如果已使能方向控制，就会使用 $\overline{\text{RTS}}$ 引脚。

使能“自动方向控制”后，CPU 将数据写入 TXFIFO 时，选定的引脚将被断言（驱动低电平）。数据的最后一位发送后，引脚将被取消（驱动高电平）。请参见 RS485CTRL 寄存器中的位 4 和 5。

除了环回模式之外， RS485CTRL 位 4 优于所有其他控制方向引脚的机制。

RS485/EIA-485 驱动器延迟时间

驱动器延迟时间是指最后一个停止位离开 TXFIFO 到 $\overline{\text{RTS}}$ 失效这一段时间。此延迟时间可在 8 位 RS485DLY 寄存器中编程，延迟时间以波特率时钟为周期。可以使用 0 到 255 位倍数的任何延迟时间。

RS485/EIA-485 输出反转

引脚 $\overline{\text{RTS}}$ （或 $\overline{\text{DTR}}$）方向控制信号的极性可通过对寄存器 RS485CTRL 第 5 位编程来反转。此位设置后，发送器有要发送的数据时，方向控制引脚将驱动到逻辑 1。数据的最后一位发送后，方向控制引脚将被驱动到逻辑 0。

12.6.2 智能卡模式

图 20 所示为一个典型的异步智能卡应用。

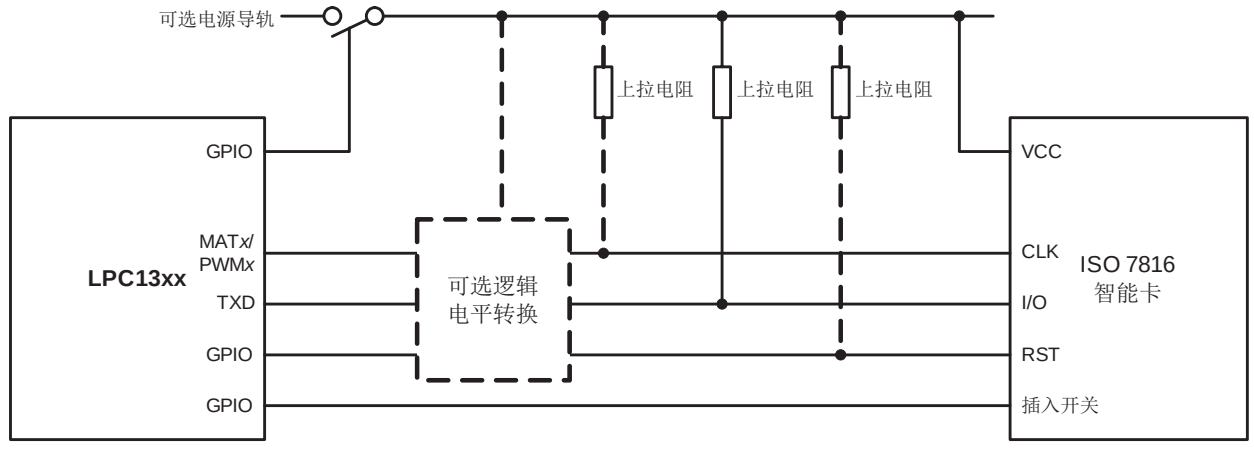


图 20. 典型智能卡应用

SCICTRL 寄存器中的 SCIEN 位 （表 226）按说明设置后， USART 在开漏 TXD 引脚上提供双向串行数据。SCIEN 为 1 时不使用 RXD 引脚。如果需要将时钟源作为智能卡的振荡器源，在需要与数据比特率不同步的更高频率时钟时，可使用定时器匹配或 PWM 输出。USART SCLK 引脚将以数据比特率同步输出数据，对于大多数异步卡，这可能还不够。软件必须使用定时器来实现字符和时钟等待时间（LPC1315/16/17/45/46/47 上不提供通过触发信号支持的硬件）。GPIO 引脚可用于控制智能卡复位和电源引脚。任何提供给卡的电源都必须采用外部开关，因为卡的电源要求通常超过了 LPC1315/16/17/45/46/47 上可能的电源电流。由于具体应用可能采用任何可用的 ISO 7816 的 A、B 或 C 类电源要求，因此如果卡使用与 LPC1315/16/17/45/46/47 不同的电源导轨，进行通信或供应电源时，要注意逻辑电平公差和要求。

12.6.2.1 智能卡设置步骤

T = 0 协议传输包含一个 8 位数据、一个偶数奇偶校验位和两个保护位，其中保护位允许特定传输的接收器通过 NACK 响应标记奇偶校验错误（参见图 21）。也可根据卡的要求添加额外保护位。如果没有发送 NACK（假定接口在 SCICTRL 中接受它们），则下一个字节会在最后一个保护位后立即发送。如果已发送 NACK，则发送器将重试发送字节，直至被成功收到或达到 SCICTRL 重试限制。

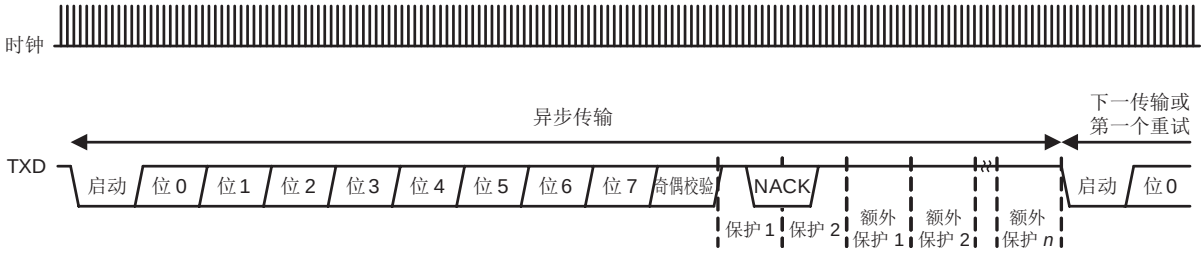


图 21. 智能卡 T = 0 波形

设置智能卡时必须考虑以下事项：

- 如果需要，可编程 PRESETCTRL （表 7），以便不会连续复位 USART。
- 编程一个 IOCON 寄存器以使能 USART TXD 功能。
- 如果要与之通信的智能卡需要一个时钟，则编程一个 IOCON 寄存器以使用 USART SCLK 功能。 USART 将使用它作为输出。
- 为 3.58 MHz 的初始 USART 频率编程 UARTCLKDIV （表 21）。
- 为 372x 过采样编程 OSR （第 12.5.15 节）。
- 必要时，分别将 DLM 和 DLL（第 12.5.3 节）编程为 00 和 01，以便不使用分频直接通过 USART 时钟。
- 为 8 位字符、已使能的奇偶校验和偶数奇偶校验编程 LCR （第 12.5.7 节）。
- 编程与智能卡相关的 GPIO 信号，以便 （按以下顺序）：
  a. 复位为低电平。
  b. 向卡提供 VCC （GPIO 引脚不含所需的 200 mA 驱动）。
  c. VPP （如果提供给卡）位于 “空闲” 状态。
- 编程 SCICTRL （第 12.5.18 节）以使能带所需选项的智能卡功能。
- 设置一个或多个定时器，提供 ISO 7816 启动所需的计时功能。
- 编程 SYSAHBCLKCTRL （表 19）以使能 USART 模块。

此后，软件应监控卡插入、处理激活、等待回答，以按 ISO7816-3 所述进行复位。

12.7 架构

USART 的架构如以下框图所示。

APB 接口提供了 CPU 或主机与 USART 之间的通信链接。

USART 接收器模块 RX 监控串行输入线路， RXD 用于有效输入。 USART RX 移位寄存器 (RSR) 通过 RXD 接受有效字符。有效字符汇编到 RSR 中后，被传递到 USART RX 缓冲寄存器 FIFO，等待 CPU 或主机通过通用主机接口进行访问。

USART 发送器模块 TX 接受 CPU 或主机写入的数据，并将数据缓冲在 USART TX 保持寄存器 FIFO (THR) 中。USART TX 移位寄存器 (TSR) 读取 THR 中存储的数据，并汇编数据以通过串行输出引脚 TXD1 发送。

USART 波特率发生器模块 BRG 生成由 USART TX 模块使用的定时。 BRG 时钟输入源是 USART_PCLK。主时钟被 DLL 和 DLM 寄存器中指定的除数分频。此分频后的时钟是 16 倍过采样时钟， NBAUDOUT。

中断接口包含寄存器 IER 和 IIR。中断接口接收从 TX 和 RX 块使能的多个一时钟宽度。

TX 和 RX 的状态信息存储在 LSR 中。 TX 和 RX 的控制信息存储在 LCR 中。

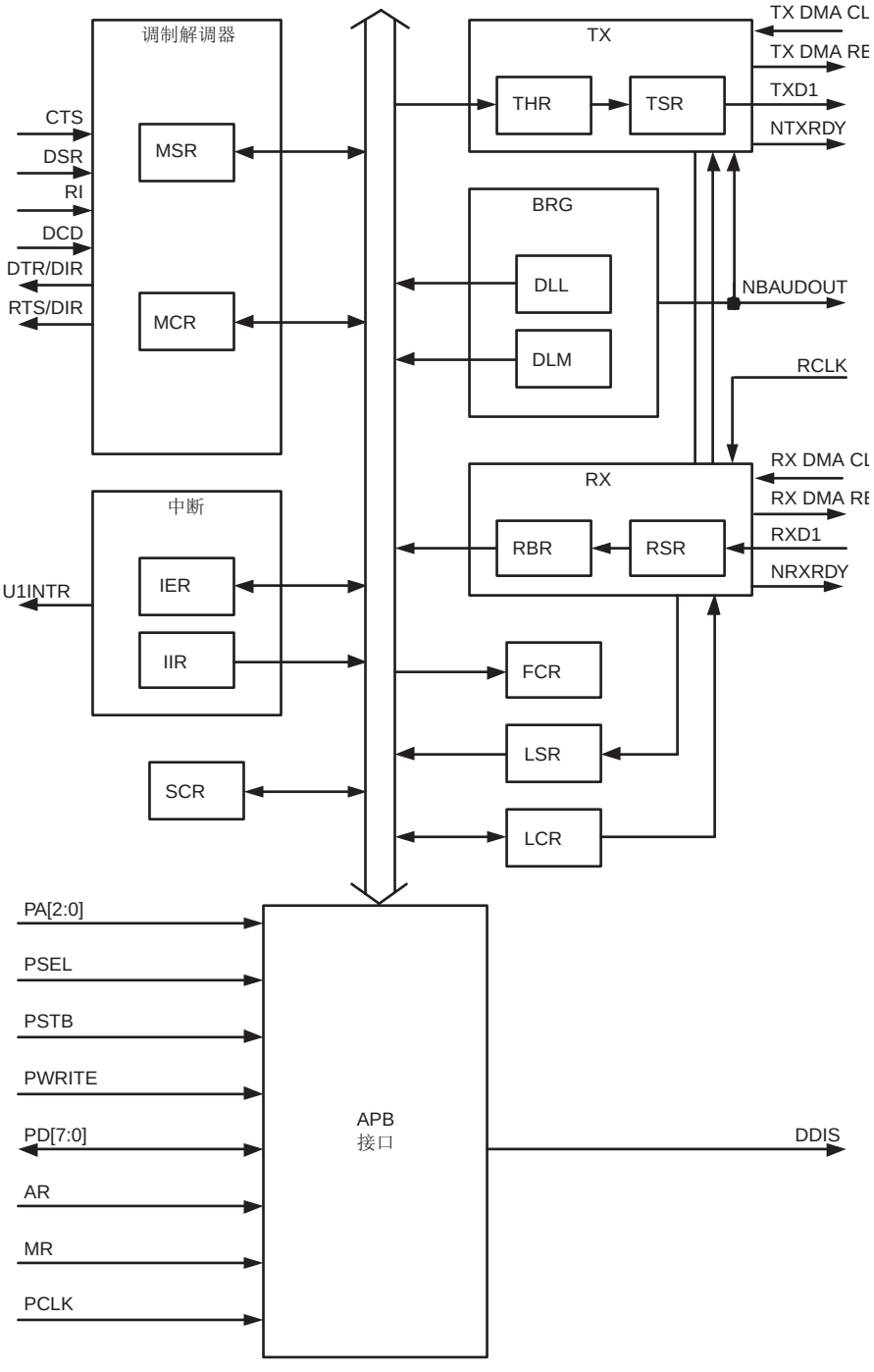


图 22. USART 框图

13.1 本章导读

所有 LPC1315/16/17/45/46/47 器件上均配有两个 SSP/SPI 接口。

13.2 基本配置

SSP0/1 是使用以下寄存器进行配置的：

1. 引脚：SSP/SPI 引脚必须在 IOCON 寄存器模块中配置。
2. 电源：在 SYSAHBCLKCTRL 寄存器中，设置 SSP0 的位 11 和 SSP1 的位 18（表 19）。
3. 外设时钟：通过写入 SSP0/1CLKDIV 寄存器使能 SSP0/SSP1 外围设备时钟 （表 20/表 22）。
4. 复位：访问 SSP/SPI 模块前，请确保将 PRESETCTRL 寄存器中的 SSP0/1_RST_N 位（位 0 和位 2）（表 7）设为 1。这将取消 SSP/SPI 模块的复位信号。

13.3 特性

- 兼容摩托罗拉 SPI、 4 线德州仪器 SSI 和国家半导体 Microwire 总线。
- 同步串行通信。
- 支持主机或从机操作。
- 同时适用于发送与接收的 8 帧 FIFO。
- 4 位至 16 位帧。

13.4 简介

SSP/SPI 是同步串行端口 (SSP) 控制器，可控制 SPI、4 线 SSI 或 Microwire 总线的操作。它可以与总线上的多个主机和从机进行交互。在指定数据传输中，总线上只有一个主机和一个从机进行通信。数据传输原则上为全双工方式， 4 位到 16 位数据帧由主机发送到从机或由从机发送到主机。实际上，通常情况下只有一个方向上的数据流包含有意义的数据。

13.5 引脚说明

表 231. SSP/SPI 引脚描述

引脚名称	类型	接口引脚名称 / 功能			引脚说明
		SPI	SSI	Microwire	
SCK0/1	I/O	SCK	CLK	SK	**串行时钟。**SCK/CLK/SK是用于同步数据传输的时钟信号。由主机驱动，从机接收。当使用 SSP/SPI 接口时，可将时钟编程为高电平有效或低电平有效，否则，它一直是高电平有效。 SCK 只在数据传输期间跳变。在其它时间，SSP/SPI 接口使其保持非工作状态或不驱动它 （使其处于高阻抗状态）。
SSEL0/1	I/O	SSEL	FS	CS	**帧同步 / 从机选择。**当 SSP/SPI 接口为总线主机时，它在串行数据发起前将该信号驱动到工作状态，再在发送数据后将信号释放到非工作状态。该信号为高电平有效还是低电平有效取决于所选择的总线和模式。当 SSP/SPI 接口为总线从机时，该信号根据使用的协议限定从主机发出的数据。 当只有一个总线主机和一个总线从机时，来自主机的帧同步或从机选择信号可直接连接到从机的相应输入中。当总线上有多个从机时，通常必需进一步限制其帧选择 / 从机选择输入，以避免多个从机对传输作出响应。
MISO0/1	I/O	MISO	DR(M) DX(S)	SI(M) SO(S)	**主机输入从机输出。**MISO 信号将串行数据由从机传输到主机。当 SSP/SPI 是从机时，从该信号上输出串行数据。当 SSP/SPI 为主机时，它记录从该信号发出的串行数据。当 SSP/SPI 为从机，且未被 FS/SSEL 选择时，它不会驱动该信号（使其处于高阻抗状态）。
MOSI0/1	I/O	MOSI	DX(M) DR(S)	SO(M) SI(S)	**主机输出从机输入。**MOSI 信号将串行数据从主机传输到从机。当 SSP/SPI 为主机时，从该信号上输出串行数据。当 SSP/SPI 为从机时，它记录从该信号发出的串行数据。

13.6 寄存器描述

SPI 控制器的寄存器地址如表 232 所示。

复位值仅反映已使用位中所存储的数据。不包含保留位中的内容。

注：寄存器名称使用 SSP 前缀表示 SPI 控制器拥有全部 SSP 功能。

表 232. 寄存器简介：SSP/SPI0 （基址 0x4004 0000）

名称	访问类型	地址偏移	描述	复位值	参考
CR0	R/W	0x000	控制寄存器 0。选择串行时钟速率、总线类型和数据大小。	0	表 234
CR1	R/W	0x004	控制寄存器 1。选择主机 / 从机和其他模式。	0	表 235
DR	R/W	0x008	数据寄存器。写满发送 FIFO，读空接收 FIFO。	0	表 236
SR	RO	0x00C	状态寄存器	0x0000 0003	表 237
CPSR	R/W	0x010	时钟预定标寄存器	0	表 238
IMSC	R/W	0x014	中断屏蔽设置和清除寄存器	0	表 239
RIS	RO	0x018	原始中断状态寄存器	0x0000 0008	表 240
MIS	RO	0x01C	屏蔽中断状态寄存器	0	表 241
ICR	WO	0x020	SSPICR 中断清除寄存器	不适用	表 242

表 233. 寄存器简介：SSP/SPI1 （基址 0x4005 8000）

名称	访问类型	地址偏移	描述	复位值	参考
CR0	R/W	0x000	控制寄存器 0。选择串行时钟速率、总线类型和数据大小。	0	表 234
CR1	R/W	0x004	控制寄存器 1。选择主机 / 从机和其他模式。	0	表 235
DR	R/W	0x008	数据寄存器。写满发送 FIFO，读空接收 FIFO。	0	表 236
SR	RO	0x00C	状态寄存器	0x0000 0003	表 237
CPSR	R/W	0x010	时钟预定标寄存器	0	表 238
IMSC	R/W	0x014	中断屏蔽设置和清除寄存器	0	表 239
RIS	RO	0x018	原始中断状态寄存器	0x0000 0008	表 240
MIS	RO	0x01C	屏蔽中断状态寄存器	0	表 241
ICR	WO	0x020	SSPICR 中断清除寄存器	不适用	表 242

13.6.1 SSP/SPI 控制寄存器 0

该寄存器控制 SSP/SPI 控制器的基本运行。

表 234. SSP/SPI 控制寄存器 0（CR0 - 地址 0x4004 0000 (SSP0)、0x4005 8000 (SSP1)）位描述

位	符号	值	描述	复位值
3:0	DSS		数据大小选择。该域控制每帧中传输的位数。不支持且不使用值 0000-0010。	0000
		0x3	4 位传输	
		0x4	5 位传输	
		0x5	6 位传输	
		0x6	7 位传输	
		0x7	8 位传输	
		0x8	9 位传输	
		0x9	10 位传输	
		0xA	11 位传输	
		0xB	12 位传输	
		0xC	13 位传输	
		0xD	14 位传输	
		0xE	15 位传输	
		0xF	16 位传输	
5:4	FRF		帧格式。	00
		0x0	SPI	
		0x1	TI	
		0x2	Microwire	
		0x3	此组合不受支持，因此不应使用。	
6	CPOL		时钟输出极性。该位只用于 SPI 模式。	0
		0	在帧与帧之间由 SPI 控制器将总线时钟维持在低电平状态。	
		1	在帧与帧之间由 SPI 控制器将总线时钟维持在高电平状态。	
7	CPHA		时钟输出相位。该位只用于 SPI 模式。	0
		0	SPI 控制器在帧传输的第一次时钟跳变时捕获串行数据，也就是说跳变**远离**时钟线的帧间状态。	
		1	SPI 控制器在帧传输的第二次时钟跳变时捕获串行数据，也就是说跳变**回到**时钟线的帧间状态。	
15:8	SCR		串行时钟速率。总线上的每位预分频器输出时钟数目，减去 1。假设 CPSDVSR 为预分频器，APB 时钟 PCLK 计时预分频器，则位频率为 PCLK/(CPSDVSR × [SCR+1])。	0x00
31:16	-	-	保留	-

13.6.2 SSP/SPI 控制寄存器 1

该寄存器控制 SSP/SPI 控制器运行的某些方面。

表 235. SSP/SPI 控制寄存器 1（CR1 - 地址 0x4004 0004 (SSP0)、0x4005 8004 (SSP1)）位描述

位	符号	值	描述	复位值
0	LBM		环回模式。	0
		0	正常运行期间。	
		1	串行输入取自串行输出 （MOSI 或 MISO），而不是串行输入引脚 （分别为 MISO 或 MOSI）。	
1	SSE		SPI 使能。	0
		0	SPI 控制器被禁用。	
		1	SPI 控制器将在串行总线上与其他装置进行交互。在设置该位之前，软件应该向其他 SSP/SPI 寄存器和中断控制器寄存器写入合适的控制信息。	
2	MS		主机 / 从机模式。只有在 SSE 位为 0 时，才能对该位执行写入操作。	0
		0	SPI 控制器作为总线上的主机，驱动 SCLK、MOSI 和 SSEL 线路，接收 MISO 线路。	
		1	SPI 控制器作为总线上的从机，驱动 MISO 线路，接收 SCLK、 MOSI 和 SSEL 线路。	
3	SOD		从机输出禁用。只有在从机模式下才与此位有关 (MS = 1)。如果值为 1，则禁止此 SPI 控制器驱动发送数据线 (MISO)。	0
31:4	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

13.6.3 SSP/SPI 数据寄存器

软件可将需要发送的数据写入该寄存器，并从中读取接收到的数据。

表 236. SSP/SPI 数据寄存器 （DR - 地址 0x4004 0008 (SSP0)、 0x4005 8008 (SSP1)）位描述

位	符号	描述	复位值
15:0	数据	**写入：**只要状态寄存器中的 TNF 位为 1，表示 Tx FIFO 不处于满状态，软件就可将即将在未来帧中发送的数据写入该寄存器。如果 Tx FIFO 之前为空且总线上的 SPI 控制器不在忙碌状态，可以立即开始数据发送。否则写入该寄存器的数据将在之前所有数据发送（接收）完毕后立即发送出去。如果数据长度小于 16 位，则软件必须使写入该寄存器的数据向右对齐。 **读：**只要状态寄存器的 RNE 位为 1，表示 Rx FIFO 不为空，软件就可从该寄存器中读取数据。当软件从该寄存器中读取数据时，SPI 控制器将返回 Rx FIFO 中最近帧中的数据。如果数据长度小于 16 位，则使此字段的数据向右对齐，更高阶位用 0 填充。	0x0000
31:16	-	保留。	-

13.6.4 SSP/SPI 状态寄存器

该只读寄存器反映 SPI 控制器的当前状态。

表 237. SSP/SPI 状态寄存器 （SR - 地址 0x4004 000C (SSP0)、 0x4005 800C (SSP1)）位描述

位	符号	描述	复位值
0	TFE	发送 FIFO 为空。当发送 FIFO 为空时该位为 1，否则为 0。	1
1	TNF	发送 FIFO 未满。如果 Tx FIFO 已满，则该位为 0；反之为 1。	1
2	RNE	接收 FIFO 未空。如果接收 FIFO 为空，则该位为 0；反之为 1。	0
3	RFF	接收 FIFO 满。如果接收 FIFO 已满，则该位为 1；反之为 0。	0
4	BSY	忙。如果 SPI 控制器空闲，则该位为 0；如果当前正在发送 / 接收一个帧和 / 或 Tx FIFO 未空，则该位为 1。	0
31:5	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

13.6.5 SSP/SPI 时钟预分频寄存器

该寄存器控制预分频器分频 SPI 外设时钟 SPI_PCLK 以产生预分频器时钟的系数，反过来，再在 SSPCR0 寄存器中被 SCR 系数分频，决定位时钟。

表 238. SSP/SPI 时钟预分频寄存器 （CPSR - 地址 0x4004 0010 (SSP0)、 0x4005 8010 (SSP1)）位描述

位	符号	描述	复位值
7:0	CPSDVSR	此偶数值介于 2 至 254 之间，SPI_PCLK 通过该值进行分频以产生预分频器输出时钟。位 0 始终读取为 0。	0
31:8	-	保留。	-

注意事项：SSPnCPSR 值必须适当初始化，否则 SPI 控制器不能正确发送数据。

在从机模式下，主机提供的 SPI 时钟速率不能超过表 20/ 表 22 中所选 SPI 外设时钟的 1/12。SSPnCPSR 寄存器的内容与此无关。

在主机模式下， $CPSDVSR_{min}$ = 2 或更大的值 （只能为偶数）。

13.6.6 SSP/SPI 中断屏蔽设置 / 清除寄存器

该寄存器控制是否使能 SPI 控制器中 4 个可能的中断条件。

表 239. SSP/SPI 中断屏蔽设置 / 清除寄存器 （IMSC - 地址 0x4004 0014 (SSP0)、 0x4005 8014 (SSP1)）位描述

位	符号	描述	复位值
0	RORIM	出现接收溢出（即当 Rx FIFO 已满且另一个帧完全接收）时，软件应将此位设置为使能中断。 ARM 规范指明，出现这种情况时，前面的帧数据会被新的帧数据覆盖。	0
1	RTIM	出现接收超时条件时，软件应将此位设置为使能中断。当Rx FIFO 不为空且超过超时周期没有被读取时，将发生接收超时。对于主机和从机模式来说超时周期是相同的，由 SSP 比特率决定：在 PCLK/(CPSDVSR × [SCR+1]) 时为 32 位。	0
2	RXIM	当 Rx FIFO 为至少半满状态时，软件应该设置此位以允许中断。	0
3	TXIM	当 Tx FIFO 为至少半空状态时，软件应该设置此位以允许中断。	0
31:4	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

13.6.7 SSP/SPI 原始中断状态寄存器

不管 IMSC 寄存器中是否使能中断，只要出现断言的中断条件，该只读寄存器便会在相应的位置包含 1。

表 240. SSP/SPI 原始中断状态寄存器 （RIS - 地址 0x4004 0018 (SSP0)、 0x4005 8018 (SSP1)）位描述

位	符号	描述	复位值
0	RORRIS	如果在 Rx FIFO 已满时完整接收到另一个帧，则该位为 1。ARM 规范指明，出现这种情况时，前面的帧数据会被新的帧数据覆盖。	0
1	RTRIS	当 Rx FIFO 不为空且超过超时周期没有被读取时，该位为 1。对于主机和从机模式来说超时周期是相同的，由 SSP 比特率决定：在 PCLK/(CPSDVSR × [SCR+1]) 时为 32 位。	0
2	RXRIS	如果 Rx FIFO 为至少半满状态该位为 1。	0
3	TXRIS	如果 Tx FIFO 为至少半空状态该位为 1。	1
31:4	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

13.6.8 SSP/SPI 屏蔽中断状态寄存器

该寄存器是一个只读寄存器，当中断条件出现且相应的中断在 IMSC 寄存器中使能时，该寄存器中对应的位为 1。当出现 SSP/SPI 中断时，中断服务例程会读取此寄存器以确定中断的原因。

表 241. SSP/SPI 屏蔽中断状态寄存器（MIS - 地址 0x4004 001C (SSP0)、0x4005 801C (SSP1)）位描述

位	符号	描述	复位值
0	RORMIS	如果当 RxFIFO 状态为满时另外一帧被完全接收了，该位为 1 且该中断被使能。	0
1	RTMIS	当 Rx FIFO 不为空且超过超时周期没有被读取时，该位为 1 且该中断被使能。对于主机和从机模式来说超时周期是相同的，由 SSP 比特率决定：在 PCLK/(CPSDVSR × [SCR+1]) 时为 32 位。	0
2	RXMIS	如果 Rx FIFO 为至少半满状态，该位为 1 且该中断被使能。	0
3	TXMIS	如果 Tx FIFO 为至少半空状态，该位为 1 且该中断被使能。	0
31:4	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

13.6.9 SSP/SPI 中断清除寄存器

软件可向该只写寄存器写入 1 个或多个 1 以清除 SPI 控制器中相应的中断条件。注意其他两个中断条件可以通过写入或者读取正确的 FIFO 来清除，或者通过清除 SSPIMSC 寄存器中对应的位值来禁用。

表 242. SSP/SPI 中断清除寄存器（ICR - 地址 0x4004 0020 (SSP0)、0x4005 8020 (SSP1)）位描述

位	符号	描述	复位值
0	RORIC	对该位写入 1 将清除 “当 RxFIFO 为满状态时帧已接收”中断。	不适用
1	RTIC	对该位写入 1 将清除 “Rx FIFO 不为空且超过超时周期没有被读取”中断。对于主机和从机模式来说超时周期是相同的，由 SSP 比特率决定：在 PCLK/(CPSDVSR × [SCR+1]) 时为 32 位。	不适用
31:2	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

13.7 功能说明

13.7.1 德州仪器同步串行帧格式

图 23 显示了 SPI 模块支持的 4 线德州仪器同步串行帧格式。

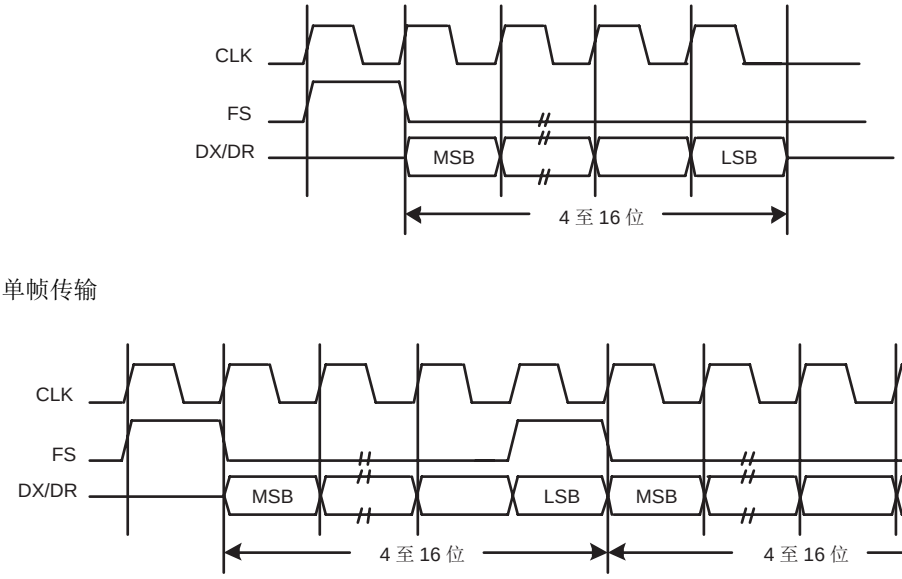


图 23. 德州仪器同步串行帧格式：a) 单帧和 b) 连续 / 背靠背 2 帧传输

对于在该模式下配置为主机的设备，CLK 和 FS 被强制为低电平，且只要 SSP 空闲，发送数据线 DX 便会处于 3 态模式。一旦发送 FIFO 的底部入口包含数据， FS 将被脉冲触发为一个 CLK 周期的高电平。将要发送的值也被从发送 FIFO 传输到发送逻辑中的串行移位寄存器。在下一个 CLK 的上升沿， 4 至 16 位数据帧的 MSB 被移出至 DX 引脚。同样，接收数据的 MSB 由片外串行从设备传送到 DR 引脚。

在每个 CLK 的下降沿，SSP 和片外串行从设备将各个数据位放入其串行移位器。LSB 被锁存后，在 CLK 的第一个上升沿，接收的数据从串行移位器传输到接收 FIFO。

13.7.2 SPI 帧格式

SPI 接口是 4 线接口，其中 SSEL 信号用作从机选择。 SPI 格式的主要特性是 SCK 信号的非工作状态和相位可通过对 SSPCR0 控制寄存器内的 CPOL 和 CPHA 位编程设定。

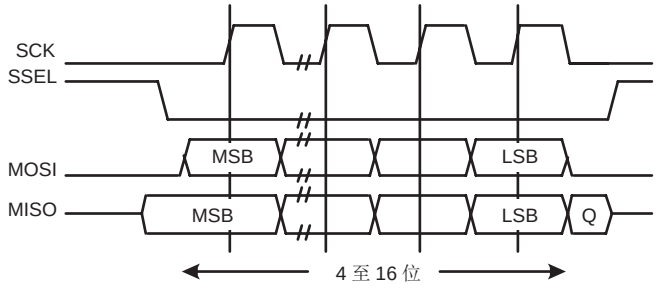
13.7.2.1 时钟极性 (CPOL) 及相位 (CPHA) 控制

当 CPOL 时钟极性控制位为低电平时，它会在 SCK 引脚产生一个稳定状态的低电平值。如果 CPOL 时钟极性控制位为高电平，则在没有传输数据时，它会在 CLK 引脚上产生一个稳态高电平值。

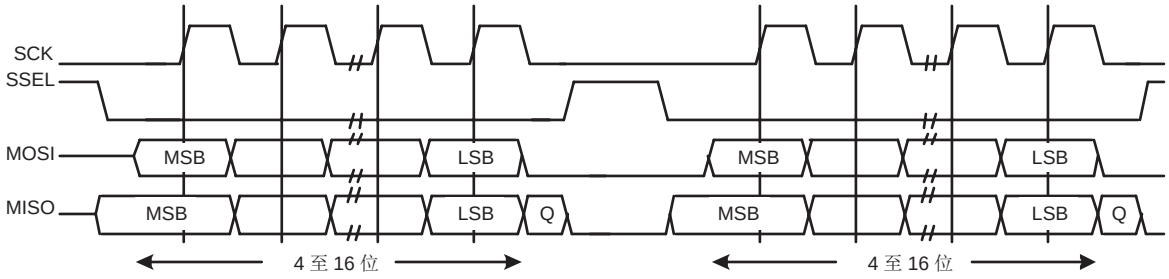
CPHA 控制位选择捕获数据及允许数据更改状态的时钟沿。通过是否在第一个数据捕获沿发生之前允许时钟跳变，该位能最大程度上影响第一个被发送的位。当 CPHA 相位控制位为低电平时，则在第一次出现时钟沿跳变时捕获数据。如果 CPHA 时钟相位控制位为高电平，则在第二次出现时钟沿跳变时捕获数据。

13.7.2.2 CPOL=0， CPHA=0 时的 SPI 格式

CPOL = 0 且 CPHA = 0 时 SPI 格式的单帧和连续帧传输信号序列如图 24 所示。



a. CPOL=0 且 CPHA=0 时的单帧传输



b. CPOL=0 且 CPHA=0 时的连续帧传输

图 24. CPOL=0 且 CPHA=0 时的 SPI 帧格式 （a) 单帧和 b) 连续帧传输）

在这种配置下，空闲期间：

- CLK 信号被强制为低电平。
- SSEL 被强制为高电平。
- 发送 MOSI/MISO 垫片处于高阻态。

如果使能 SSP/SPI，且在发送 FIFO 中存在有效数据，则 SSEL 主机信号被驱动为低电平，指示数据传输开始。这将使得从机数据被连接至主机的MISO输入线上。使能主机的MOSI。

1/2 个 SCK 周期后，有效的主机数据将被传输到 MOSI 引脚。由于主机和从机数据均已设定，因此再过 1/2 个 SCK 周期， SCK 主时钟引脚就会变为高电平。

在 SCK 信号的上升沿捕获数据，并在 SCK 信号的下降沿传播数据。

在单字发送的情况下，当数据字的所有位都传输完毕以后，SSEL 线在最后一位被捕获后的一个 SCK 周期后回到空闲高电平状态。

但是，在进行连续的背靠背传输时，传输各数据字之间的 SSEL 信号必须为高电平。这是因为如果 CPHA 位为逻辑值 0，从机选择引脚会冻结其串行外设寄存器中的数据，并且不允许其被更改。因此主机装置必须在每个数据传输之间升高从机装置的 SSEL 引脚上的电平以使能串行外设数据写入。连续传输完成后，SSEL 引脚在捕获到最后一位后的一个 SCK 周期内将返回至空闲状态。

13.7.2.3 CPOL=0， CPHA=1 时的 SPI 格式

CPOL = 0 且 CPHA = 1 时 SPI 格式的传输信号序列如图 25 所示，包含单帧和连续帧传输。

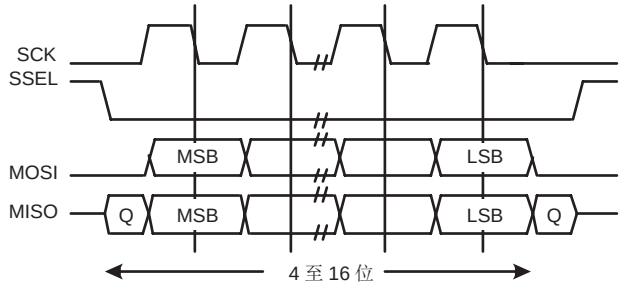


图 25. CPOL=0 且 CPHA=1 时的 SPI 帧格式

在这种配置下，空闲期间：

- CLK 信号被强制为低电平。
- SSEL 被强制为高电平。
- 发送 MOSI/MISO 垫片处于高阻态。

如果使能 SSP/SPI，且在发送 FIFO 中存在有效数据，则 SSEL 主机信号被驱动为低电平，指示数据传输开始。使能主机的 MOSI 引脚。在又过了半个 SCK 周期以后，主机和从机的有效数据都被连接至各自的发送线上。同时， SCK 在上升沿跳变时使能。

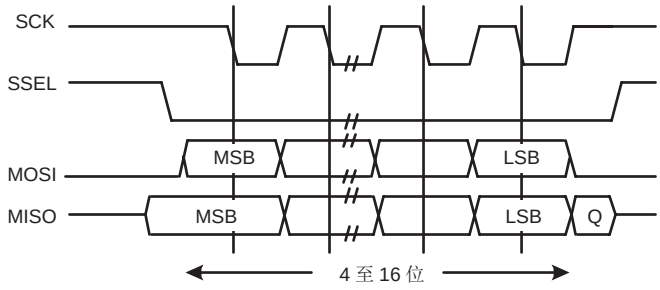
然后数据将在 SCK 信号下降沿捕获，并且在上升沿传播。

在单字传输的情况下，当所有位都传输完毕以后，SSEL 线在最后一位被捕获后的一个 SCK 周期后回到空闲高电平状态。

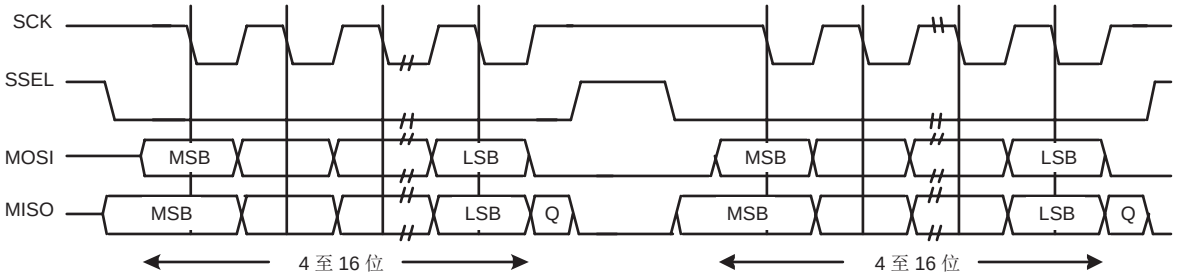
对于连续背靠背传输，SSEL 引脚在连续数据字之间保持为低电平，结束过程与单字传输相同。

13.7.2.4 CPOL = 1 且 CPHA = 0 时的 SPI 格式

CPOL=1 且 CPHA=0 时， SPI 格式的单帧和连续帧传输信号序列如图 26 所示。



a. CPOL=1 且 CPHA=0 时的单帧传输



b. CPOL=1 且 CPHA=0 时的连续帧传输

图 26. CPOL = 1 且 CPHA = 0 时的 SPI 帧格式 （(a) 单帧和 b) 连续帧传输）

在这种配置下，空闲期间：

- CLK 信号被强制为高电平。
- SSEL 被强制为高电平。
- 发送 MOSI/MISO 垫片处于高阻态。

如果使能 SSP/SPI，且在发送 FIFO 中存在有效数据，则 SSEL 主机信号被驱动为低电平，指示数据传输开始，这会使从机数据立即传输到主机的MISO线上。使能主机的MOSI引脚。

1/2 个 SCK 周期后，有效的主机数据将被传输到 MOSI 线。现在主机和从机的数据都被设置好了，再过半个 SCK 周期 SCK 主机时钟引脚转至低电平。这意味着，在 SCK 信号的下降沿捕获数据并在 SCK 信号的上升沿传播数据。

在单字发送的情况下，当数据字的所有位都传输完毕以后，SSEL 线在最后一位被捕获后的一个 SCK 周期后回到空闲高电平状态。

但是，在进行连续的背靠背传输时，传输各数据字之间的 SSEL 信号必须为高电平。这是因为如果 CPHA 位为逻辑值 0，从机选择引脚会冻结其串行外设寄存器中的数据，并且不允许其被更改。因此主机装置必须在每个数据传输之间升高从机装置的 SSEL 引脚上的电平以使能串行外设数据写入。连续传输完成后，SSEL 引脚在捕获到最后一位后的一个 SCK 周期内将返回至空闲状态。

13.7.2.5 CPOL = 1 且 CPHA = 1 时的 SPI 格式

CPOL = 1 且 CPHA = 1 时 SPI 格式的传输信号序列如图 27 所示，包含单帧和连续帧传输。

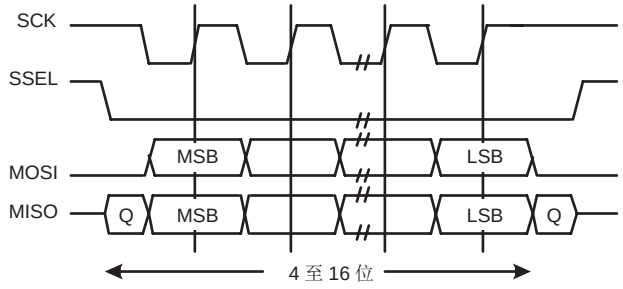


图 27. CPOL = 1 且 CPHA = 1 时的 SPI 帧格式

在这种配置下，空闲期间：

- CLK 信号被强制为高电平。
- SSEL 被强制为高电平。
- 发送 MOSI/MISO 垫片处于高阻态。

如果使能 SSP/SPI，且在发送 FIFO 中存在有效数据，则 SSEL 主机信号被驱动为低电平，指示数据传输开始。使能主机的 MOSI。在又过了半个 SCK 周期以后，主机和从机的数据都被连接至各自的发送线上。同时 SCK 被一个下降沿跳变使能。然后，在 SCK 信号的上升沿捕获数据，并在 SCK 信号的下降沿传播数据。

发送单个字时，在传输完所有位后，在捕获最后一位后的一个 SCK 周期内， SSEL 线返回到其空闲高电平状态。对于连续背靠背发送，SSEL 引脚保持有效低电平状态，直到最后一个字的最后一位被捕获，然后如上所述回到其空闲状态。通常，当进行连续的背靠背传输时，连续的数据字之间 SSEL 引脚保持为低电平，终止条件与传送单个字时相同。

13.7.3 半导体 Microwire 帧格式

图28显示了单帧的Microwire帧格式。图29显示了进行背靠背帧传输时的Microwire帧格式。

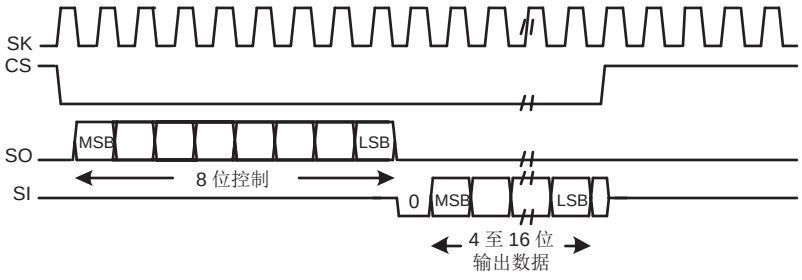


图 28. Microwire 帧格式 （单帧传输）

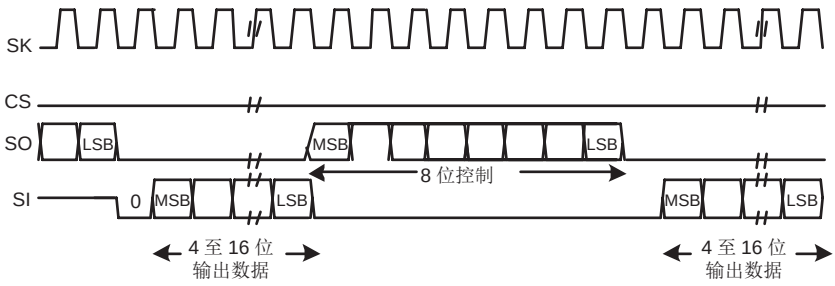


图 29. Microwire 帧格式 （连续传送）

Microwire 格式与 SPI 格式非常相似，都是使用主 - 从消息传递技术，但它采用的传输方式是半双工方式而不是全双工方式。每次串行传输均从一个 8 位控制字开始，由 SSP/SPI 发送到片外从机设备。在此发送过程中， SSP/SPI 不会接收输入的数据。发送完消息后，片外从机对其进行解码，然后在 8 位控制消息的最后一位发送结束后再等待一个串行时钟，以所需的数据进行响应。返回的数据长度为 4 到 16 位，使整个帧长度范围为 13 到 25 位。

在这种配置下，空闲期间：

- SK 信号被强制为低电平。
- CS 被强制为高电平。
- 发送数据线 SO 被任意强制为低电平。

通过向发送 FIFO 写入控制字节来触发传送。 CS 下降沿会使包含在发送 FIFO 底端的值传输到发送逻辑的串行移位寄存器，并使 8 位控制帧的 MSB 输出到 SO 引脚。 CS 在帧传输期间保持低电平。 SI 引脚在此发送过程中保持三态。

片外串行从设备在每个 SK 的上升沿将各控制位锁存到串行移位器中。当从机设备将最后一位锁存后，控制字节会在一个时钟等待状态期间被解码，而从机则通过将数据传回至 SSP/SPI 来进行响应。每个位在 SK 的下降沿被驱动到 SI 线。 SSP/SPI 依次在 SK 的上升沿锁存每个位。对于单帧传输，在帧的末端，在最后一位已锁存到接收串行移位器后的一个时钟周期内， CS 信号会被置为高电平，这会导致数据被传输到接收 FIFO。

注： LSB 被接收移位器锁存后或当 CS 引脚变为高电平时，片外从机设备的接收线在 SK 下降沿呈现三态。

对于连续帧传输，数据传输开始和结束的方式与单帧传输相同。然而， CS 线持续有效（保持低电平），且数据以背靠背方式进行传输。当前数据帧的 LSB 被接收后，下一帧的控制字节立即发送。在帧数据的 LSB 被锁存到 SSP/SPI 后，每个收到的值将在 SK 的下降沿从接收移位器传输。

13.7.3.1 Microwire 模式下 CS 相对于 SK 的建立和保持时间要求

在 Microwire 模式下，CS 变为低电平后，SSP/SPI 从机在 SK 上升沿时对接收数据的第一位进行采样。主机驱动 SK 自由运行时必须确保 CS 信号相对于 SK 上升沿有充足的建立和保持时间。

图 30 描绘了这些建立和保持时间的要求。相对于 SK 上升沿（SSP/SPI 从机在该上升沿对接收数据的第一位进行采样）， CS 的建立时间必须至少为 SK （SSP/SPI 在 SK 上运行）周期的 2 倍。相对于该边沿之前的 SK 上升沿， CS 必须保持至少一个 SK 周期。

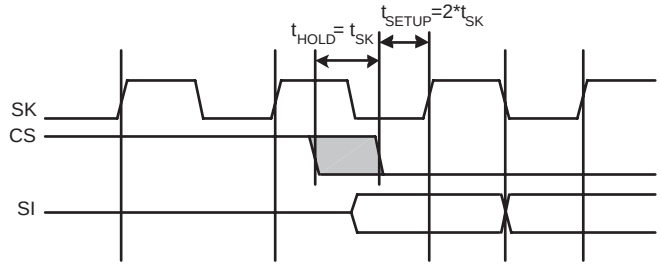


图 30. Microwire 帧格式建立及保持时间

14.1 本章导读

所有 LPC1315/16/17/45/46/47 器件中的 I^2C 总线模块都是一样的。

14.2 基本配置

I^2C 总线接口是使用以下寄存器进行配置的：

1. 引脚：I2C 引脚功能和 I2C 模式是在 IOCON 寄存器模块中配置的 （表 60 和表 61）。
2. 电源和外围设备时钟：在 SYSAHBCLKCTRL 寄存器中，设置位 5 （表 19）。
3. 复位：访问 I2C 模块前，请确保将 PRESETCTRL 寄存器中的 I2C_RST_N 位 （位 1）（表 7）设为 1。这将取消 I2C 模块的复位信号。

14.3 特性

- I^2C 总线包含一个带有 2 个引脚的标准 I^2C 兼容总线接口。
- I^2C 总线接口，可配置为主机、从机或主 / 从机。
- 支持超快速模式。
- 在同时发送的主机之间处理仲裁，不会破坏总线上的串行数据。
- 可编程时钟允许调整 I^2C 传输速率。
- 主从机之间的数据传输是双向的。
- 串行时钟同步允许具有不同位率的设备通过一个串行总线通信。
- 串行时钟同步用作信号交换机制，以挂起并恢复串行传输。
- 可以选择识别多达四个不同的从属地址。
- 监控模式可观察所有的 I^2C 总线流量，而不用考虑从属地址。
- I^2C 总线可用于测试和诊断。

14.4 应用

到外部 I^2C 标准零件的接口，如串行 RAM、 LCD、音频发生器和其它微控制器等。

14.5 简介

典型的 I²C 总线配置如图 31 所示。根据方向位的状态 (R/W)， I²C 总线上可能存在以下两种类型的数据传输方式：

- 从主机发送器到从机接收器的数据传输。主机发送的第一个字节是从属地址，后面跟着许多数据字节。从机在接收到每个字节后返回一个确认位。
- 从从机发送器到主机接收器的数据传输。第一个字节 （从属地址）由主机发送，从机随后返回确认位。后面跟随从机发送到主机的数据字节。主机在接收到所有字节而非最后一个字节后返回确认位。在最后接收的字节末尾返回一个“未确认”。主设备生成所有串行时钟脉冲以及 “起动”和 “停止”条件。传输在 “停止”条件或在 “重复起始”条件下结束。由于重复起始条件也是下一次串行传输的开始，因此不释放 I²C 总线。

I²C 接口是字节导向型，有 4 个操作模式：主机发送器模式、主机接收器模式、从机发送器模式和从机接收器模式。

I²C 接口遵循整个 I²C 规范，支持在不影响同一 I²C 总线上其它器件的情况下关闭 ARM Cortex-M3。

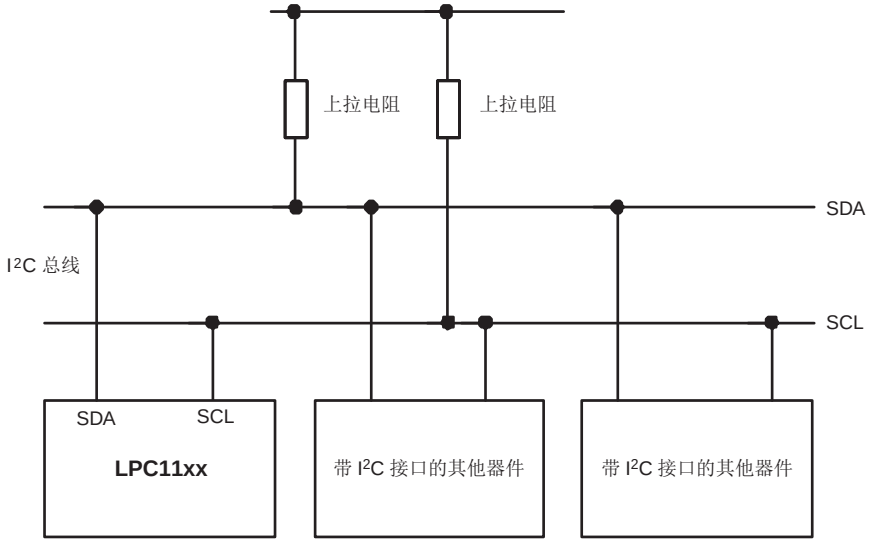


图 31. I²C 总线配置

14.5.1 I²C 超快速模式

超快速模式支持以 1 Mbit/ 秒的传输速率与恩智浦半导体现在所提供的 I²C 总线产品通信。

14.6 引脚说明

表 243. I²C 总线引脚描述

引脚	类型	描述
SDA	输入 / 输出	I²C 串行数据
SCL	输入 / 输出	I²C 串行时钟

I²C 总线引脚必须通过 IOCON_PIO0_4 （表 60）和 IOCON_PIO0_5 寄存器 （表 61）配置，以用于标准 / 快速模式或超快速模式。在超快速模式中，可选择的速率在 400 kHz 以上，高达 1MHz。 I²C 总线引脚为开漏输出并且完全兼容 I²C 总线规范。

14.7 寄存器描述

表 244. 寄存器简介：I²C （基址 0x4000 0000）

名称	访问类型	地址偏移	描述	复位值 [1]	参考
CONSET	R/W	0x000	**I2C 控制置位寄存器。**当向该寄存器的位写 1 时， I²C 控制寄存器中的相应位置位。写 0 时对 I²C 控制寄存器的相应位没有影响。	0x00	表 245
STAT	RO	0x004	**I2C 状态寄存器。**在 I²C 工作期间，该寄存器提供详细的状态码，允许软件决定需要执行的下一步操作。	0xF8	表 246
DAT	R/W	0x008	**I2C 数据寄存器。**在主机或从机发送模式期间，要发送的数据写入该寄存器。在主机或从机接收模式期间，可从此寄存器读取已经接收的数据。	0x00	表 247
ADR0	R/W	0x00C	**I2C 从属地址寄存器 0。**包含 7 位从属地址，用于从机模式下 I²C 接口操作，不用于主机模式下。最低有效位确定从机是否响应通用调用地址。	0x00	表 248
SCLH	R/W	0x010	**SCH 占空比寄存器高半字。**决定 I²C 时钟的高电平时间。	0x04	表 249
SCLL	R/W	0x014	**SCL 占空比寄存器低半字。**决定 I²C 时钟低电平时间。I2nSCLL 和 I2nSCLH 一起决定 I²C 主机产生的时钟频率及从机模式下所用的时间。	0x04	表 250
CONCLR	WO	0x018	**I2C 控制清除寄存器。**当向该寄存器的位写 1 时， I²C 控制寄存器中的相应位清零。写 0 时对 I²C 控制寄存器的相应位没有影响。	不适用	表 252
MMCTRL	R/W	0x01C	**监控模式控制寄存器。**	0x00	表 253
ADR1	R/W	0x020	**I2C 从属地址寄存器 1。**包含 7 位从属地址，用于从机模式下 I²C 接口操作，不用于主机模式下。最低有效位确定从机是否响应通用调用地址。	0x00	表 254
ADR2	R/W	0x024	**I2C 从属地址寄存器 2。**包含 7 位从属地址，用于从机模式下 I²C 接口操作，不用于主机模式下。最低有效位确定从机是否响应通用调用地址。	0x00	表 254
ADR3	R/W	0x028	**I2C 从属地址寄存器 3。**包含 7 位从属地址，用于从机模式下 I²C 接口操作，不用于主机模式下。最低有效位确定从机是否响应通用调用地址。	0x00	表 254
DATA_BUFFER	RO	0x02C	**数据缓冲寄存器。**从总线每接收 9 位（8 位数据加 ACK 或 NACK）， I2DAT 移位寄存器中 8 个 MSB 的内容，便将被自动传送到 DATA_BUFFER。	0x00	表 255
MASK0	R/W	0x030	**I2C 从属地址屏蔽寄存器 0**。该屏蔽寄存器与 I2ADR0 一起决定地址匹配。与通用调用地址 （‘0000000’）相比时，屏蔽寄存器不产生任何影响。	0x00	表 256

表 244. 寄存器简介：I²C （基址 0x4000 0000）（续）

名称	访问类型	地址偏移	描述	复位值 [1]	参考
MASK1	R/W	0x034	**I2C 从属地址屏蔽寄存器 1**。该屏蔽寄存器与 I2ADR0 一起决定地址匹配。与通用调用地址 （‘0000000’）相比时，屏蔽寄存器不产生任何影响。	0x00	表 256
MASK2	R/W	0x038	**I2C 从属地址屏蔽寄存器 2**。该屏蔽寄存器与 I2ADR0 一起决定地址匹配。与通用调用地址 （‘0000000’）相比时，屏蔽寄存器不产生任何影响。	0x00	表 256
MASK3	R/W	0x03C	**I2C 从属地址屏蔽寄存器 3**。该屏蔽寄存器与 I2ADR0 一起决定地址匹配。与通用调用地址 （‘0000000’）相比时，屏蔽寄存器不产生任何影响。	0x00	表 256

[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。

14.7.1 I²C 控制设置寄存器 (CONSET)

CONSET 寄存器控制 CON 寄存器中位的设置，这些位控制 I²C 接口的操作。向该寄存器的位写 1 会使 I²C 控制寄存器中的相应位置位。写入 0 无效。

表 245. I²C 控制置位寄存器 （CONSET - 地址 0x4000 0000）位描述

位	符号	描述	复位值
1:0	-	保留。用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
2	AA	断言确认标志。	
3	SI	I²C 中断标志。	0
4	STO	停止标志。	0
5	STA	起动标志。	0
6	I2EN	I²C 接口使能。	0
31:7	-	保留。未定义从保留位读取的值。	-

I2EN I²C 接口使能。当 I2EN 为 1 时， I²C 接口使能。可通过将 1 写入 CONCLR 寄存器中的 I2ENC 位将 I2EN 清除。当 I2EN 为 0 时， I²C 接口禁用。

当 I2EN 为“0”时，忽略 SDA 和 SCL 输入信号，I²C 模块处于“不可寻址”的从机状态，STO 位强制为 “0”。

I2EN 不应用于暂时释放 I²C 总线，因为当 I2EN 复位时，I²C 总线状态丢失。应使用 AA 标志代替。

STA 是起始标志。该位置位时，I²C 接口进入主机模式并发送一个起始条件，如果已经处于主机模式，则发送一个重复起始条件。

当 STA 为 1 且 I²C 接口没有处于主机模式时，它将进入主机模式，校验总线并在总线空闲时产生一个起始条件。如果总线不空闲，则其等待 “停止”条件 （这将释放总线），并在延迟内置时钟发生器的半个时钟周期后产生 “起动”条件。当 I²C 接口已经处于主机模式且已发送或接收了数据时，它会发送一个重复起始条件。STA 可在任意时间置位，包括 I²C 接口处于可寻址的从机模式时也可置位。

可通过将 1 写入 CONCLR 寄存器中的 STAC 位将 STA 清除。当 STA 为 0 时，将不会生成任何 “起动”条件或 “重复起始”条件。

STA 和 STO 都置位时，如果接口处于主机模式下，则向 I²C 总线发送一个停止条件，然后再发送一个起始条件。如果 I²C 接口处于从机模式，则产生内部停止条件，但不发送到总线上。

STO 是停止标志。在主机模式下，该位置位会使 I²C 接口发送一个停止条件，或在从机模式下从错误状态中恢复。当主机模式下 STO=1 时，向 I²C 总线发送停止条件。当总线检测到停止条件时， STO 自动清零。

在从机模式下，设置此位可从错误状态中恢复。在此情况下，没有 “停止” 条件传送到总线。硬件的行为就好像已经接收到 “停止” 条件并切换到 “未寻址的” 从机接收器模式。STO 标志由硬件自动清除。

SI 是 I²C 中断标志。当 I²C 状态改变时该位置位。不过由于进入状态 F8 时，无需中断服务程序处理任何事宜，便不会设置 SI。

SI 处于设置状态时，SCL 线路串行时钟的低电平时间段有延长，且会中止串行传送。当 SCL 为 “高电平” 时，其不受 SI 标志状态的影响。SI 必须通过软件复位，通过将 1 写入 CONCLR 寄存器的 SIC 位实现。

AA 是应答标志位。当设置为 1 时，在以下情况下，SCL 线上的确认时钟脉冲期间会返回确认 （SDA 为低电平）：

1. 已接收到从属地址寄存器中的地址。
2. 在设置 ADR 寄存器中的通用调用位 (GC) 时，已经接收到通用调用地址。
3. 当 I²C 处于主接收模式时，接收到一个数据字节。
4. 当 I²C 处于可寻址的从机接收模式时，接收到一个数据字节。

可通过将 1 写入 CONCLR 寄存器中的 AAC 位清除 AA 位。当 AA 为 0 时，在以下情况下，SCL 线上的确认时钟脉冲期间会返回确认 （SDA 为高电平）：

1. 当 I²C 处于主接收模式时，接收到一个数据字节。
2. 当 I²C 处于可寻址的从机接收模式时，接收到一个数据字节。

14.7.2 I²C 状态寄存器 (STAT)

每个 I²C 状态寄存器反映相应 I²C 接口的情况。 I²C 状态寄存器为只读。

表 246. I²C 状态寄存器 (STAT - 0x4000 0004) 位描述

位	符号	描述	复位值
2:0	-	这些位未使用且始终为 0。	0
7:3	状态	这些位提供关于 I²C 接口的实际状态信息。	0x1F
31:8	-	保留。未定义从保留位读取的值。	-

三个最低有效位始终为 0。作为字节时，状态寄存器内容表示状态码。有 26 种可能的状态码。当状态码为 0xF8 时，没有相关信息可用且 SI 位未置位。其它所有 25 个状态码符合定义的 I²C 状态。当进入这些状态中的任一状态时，SI 位将被设置。关于状态码的完整列表，参见表表 261 ～表 266。

14.7.3 I²C 数据寄存器 (DAT)

此寄存器包含要发送的数据或刚接收的数据。 SI 位置位后，仅在此寄存器没有进行字节移位时 CPU 才可以对其进行读写操作。只要 SI 位进行了设置， DAT 寄存器中的数据就保持稳定不变。 DAT 寄存器中的数据始终从右向左移位：要发送的第一位是 MSB （位 7），在收到一个字节后，已接收数据的第一位放在 DAT 寄存器的 MSB 上。

表 247. I²C 数据寄存器 (DAT - 0x4000 0008) 位描述

位	符号	描述	复位值
7:0	数据	此寄存器保留已接收或要发送的数据值。	0
31:8	-	保留。未定义从保留位读取的值。	-

14.7.4 I²C 从机地址寄存器 0 (ADR0)

该寄存器可读 / 写，只有在 I²C 接口设置为从机模式时才可用。在主机模式下，此寄存器无效。 ADR 寄存器的 LSB 为通用调用位。在此位设置后，会识别通用调用地址 (0x00)。

如果此寄存器包含0x00，则I²C将不确认总线上的任何地址。复位时全部四个寄存器（ADR0 至 ADR3）将被清除到此禁用状态。另请参见表 254。

表 248. I²C 从属地址寄存器 0 (ADR0- 0x4000 000C) 位描述

位	符号	描述	复位值
0	GC	通用调用使能位。	0
7:1	地址	从机模式的 I²C 器件地址。	0x00
31:8	-	保留。未定义从保留位读取的值。	-

14.7.5 I²C SCL 高、低占空比寄存器 （SCLH 和 SCLL）

表 249. I²C SCL 高电平占空比寄存器 （SCLH - 地址 0x4000 0010）位描述

位	符号	描述	复位值
15:0	SCLH	SCL 高电平时段选择的计数。	0x0004
31:16	-	保留。未定义从保留位读取的值。	-

表 250. I²C SCL 低电平占空比寄存器 (SCLL - 0x4000 0014) 位描述

位	符号	描述	复位值
15:0	SCLL	SCL 低电平时段选择的计数。	0x0004
31:16	-	保留。未定义从保留位读取的值。	-

14.7.5.1 选择适当的 I²C 数据速率和占空比

软件必须为寄存器 SCLH 和 SCLL 设置值，以选择适当的数据速率和占空比。SCLH 为 SCL 高电平时间定义 I2C_PCLK 周期数。SCLL 为 SCL 低电平时间定义 I2C_PCLK 周期数。频率由下面公式确定 （I2C_PCLK 是外围 I2C 时钟的频率）：

(4)

$$I^2C_{bitfrequency} = \frac{I2CPCLK}{SCLH + SCLL}$$

SCLL 和 SCLH 的值必须确保数据速率在适当的 I^2C 数据速率范围内。各寄存器的值必须大于或等于 4。表 251 给出了根据 I2C_PCLK 频率和 SCLL 及 SCLH 值计算出来的 I^2C 总线速率的示例。

表 251. 用于所选 I^2C 时钟值的 SCLL + SCLH 值

I^2C 模式	I^2C 位频率	I2C_PCLK (MHz)								
		6	8	10	12	16	20	30	40	50
		SCLH + SCLL								
标准模式	100 kHz	60	80	100	120	160	200	300	400	500
快速模式	400 kHz	15	20	25	30	40	50	75	100	125
超快速模式	1 MHz	-	8	10	12	16	20	30	40	50

SCLL 和 SCLH 值不必相同，软件可通过设置这两个寄存器设置 SCL 的不同占空比。例如，I^2C 总线规范定义在快速模式和超快速模式 I^2C 下不同值对应的 SCL 低电平时间和高电平时间。

14.7.6 I^2C 控制清除寄存器 (CONCLR)

CONCLR 寄存器控制 CON 寄存器中位的清除，这些位控制 I^2C 接口的操作。向该寄存器的位写 1 会使 I^2C 控制寄存器中的相应位清零。写入 0 无效。

表 252. I^2C 控制清零寄存器 (CONCLR - 0x4000 0018) 位描述

位	符号	描述	复位值
1:0	-	保留。用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
2	AAC	断言确认清除位。	
3	SIC	I^2C 中断清零位。	0
4	-	保留。用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
5	STAC	START 标志清除位。	0
6	I2ENC	I^2C 接口禁用位。	0
7	-	保留。用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
31:8	-	保留。未定义从保留位读取的值。	-

AAC 是声明应答清零位。将 1 写入此位会清除 CONSET 寄存器中的 AA 位。写入 0 无效。

SIC 是 I^2C 中断清零位。将 1 写入此位会清除 CONSET 寄存器中的 SI 位。写入 0 无效。

STAC 是起始标志清零位。将 1 写入此位会清除 CONSET 寄存器中的 STA 位。写入 0 无效。

I2ENC 是 I^2C 接口禁用位。将 1 写入此位会清除 CONSET 寄存器中的 I2EN 位。写入 0 无效。

14.7.7 I^2C 监控模式控制寄存器 (MMCTRL)

该寄存器控制监控模式，监控模式允许 I^2C 块在不需实际参与通信或干扰 I^2C 总线的情况下监控 I^2C 总线上的流量。

表 253. I²C 监控模式控制寄存器 (MMCTRL - 0x4000 001C) 位描述

位	符号	值	描述	复位值
0	MM_ENA		监控模式使能。	0
		0	监控模式禁用。	
		1	I²C 块将进入监控模式。在此模式下， SDA 输出将强制为高电平。这可防止 I²C 块向 I²C 数据总线输出任何类型的数据（包括 ACK）。 根据 ENA_SCL 位状态，也可以将输出强制为高电平，以防止模块控制 I²C 时钟线。	
1	ENA_SCL		SCL 输出使能。	0
		0	如果此位被清除为 ‘0’ ，则当模块处于监控模式时将强制 SCL 输出为高电平。如上所述，这可防止模块控制 I²C 时钟线。	
		1	该位置位时， I²C 块将以与正常操作中相同的方法控制时钟线。这意味着，作为从机设备， I²C 模块可 “延长” 时钟线（使其为低电平），直到它有时间响应 I²C 中断为止。[1]	
2	MATCH_ALL		选择中断寄存器匹配。	0
		0	如果此位被清除，则只有当四个 （最多）上述地址寄存器中之一出现匹配时，才会生成中断。也就是说，就地址识别而言模块会作为普通从机响应。	
		1	当该位设为 1 且 I²C 处于监控模式时，可在任意接收的地址上产生中断。这将使该部件可以监控总线上的所有流量。	
31:3	-	-	保留。未定义从保留位读取的值。	

[1] 当 ENA_SCL 位清除且 I²C 不能再延迟总线时，中断响应时间就变得很重要。为了使器件在这些情况下能有更多时间对 I²C 中断作出响应，就需要使用 DATA_BUFFER 寄存器 （第 14.7.9 节）来保存接收到的数据，保存时间为发送完一个 9 位字的时间。

注： 如果 MM_ENA 为 ‘0’（即，如果模块不处于监控模式），则 ENA_SCL 和 MATCH_ALL 位无效。

14.7.7.1 监控模式的中断

模块处于监控模式时，所有中断都将正常发生。这意味着第一个中断会在检测到地址匹配时发生 （如果已设置 MATCH_ALL 位，则接收的任何地址都会发生中断，否则只有在地址与四个地址寄存器中之一匹配时才会发生中断）。

地址匹配检测后，对于从机写传输，在接收每个字节后都会产生中断，对于从机读传输，则在模块 “认为” 已传输每个字节后产生中断。在此第二种情况下，数据寄存器实际上包含了总线上其它从机发送的数据，该从机实际上是由主机寻址的。

在所有这些中断发生后，处理器可读取数据寄存器以查看总线上实际传送的数据。

14.7.7.2 监控模式中的仲裁丢失

在监控模式下，I²C 块不能响应总线主机的信息请求或发布应答，而是由总线上的其它从机响应。就我们的模块而言，这很可能会导致丢失仲裁的状态。

软件应当意识到，模块处于监控模式且不应响应检测到的任何仲裁状态丢失。另外，模块中还可设计一个硬件以阻止一些 / 所有的仲裁丢失状态发生 （如果这些状态会阻止产生想要的中断或产生不想要的中断）。是否需要此类硬件仍待定。

14.7.8 I²C 从机地址寄存器 (ADR[1, 2, 3])

这些寄存器可读 / 写，只有在 I²C 接口设置为从机模式时才可用。在主机模式下，此寄存器无效。 ADR 寄存器的 LSB 为通用调用位。在此位设置后，会识别通用调用地址 (0x00)。

如果这些寄存器包含 0x00，则 I²C 将不确认总线上的任何地址。复位时所有四个寄存器都将清除到此禁用状态 （另请参见表 248）。

表 254. I²C 从属地址寄存器 (ADR[1, 2, 3]- 0x4000 00[20, 24, 28]) 位描述

位	符号	描述	复位值
0	GC	通用调用使能位。	0
7:1	地址	从机模式的 I²C 器件地址。	0x00
31:8	-	保留。未定义从保留位读取的值。	0

14.7.9 I²C 数据缓冲寄存器 (DATA_BUFFER)

在监控模式下，如果 ENA_SCL 没有置位，则 I²C 块就不能延长时钟（使总线延迟）。这意味着处理器读取总线接收数据内容的时间有限。如果处理器读 DAT 移位寄存器，则在接收的数据被新数据覆写之前，它如平常一样只有一位时间响应中断。

为了给处理器提供更多时间来响应，将增加一个新的 8 位只读 DATA_BUFFER 寄存器。在总线上每接收九个位（8 位数据加 ACK 或 NACK）后，会自动将 DAT 移位寄存器的 8 MSB 内容传输到 DATA_BUFFER，这意味着处理器有九位传输时间响应中断并在数据被覆写前读取。

处理器仍能像往常一样直接读 DAT 寄存器，并且 DAT 的行为不会有任何改变。

尽管 DATA_BUFFER 寄存器主要用于 ENA_SCL 位 = ‘0’ 的监控模式，但它也可用于在任何操作模式下随时读取。

表 255. I²C 数据缓冲寄存器 (DATA_BUFFER - 0x4000 002C) 位描述

位	符号	描述	复位值
7:0	数据	此寄存器保留 DAT 移位寄存器的 8 MSB 内容。	0
31:8	-	保留。未定义从保留位读取的值。	0

14.7.10 I²C 屏蔽寄存器 (MASK[0, 1, 2, 3])

四个屏蔽寄存器每个都包含七个有效位 (7:1)。这些屏蔽寄存器与之关联的 ADRn 寄存器相比，其设置为 ‘1’ 的任何位都会导致已接收地址的相应位上的自动比较。换句话说，决定地址匹配时不考虑 ADRn 寄存器中被屏蔽的位。

重置时，所有屏蔽寄存器位都被清除为 ‘0’。

屏蔽寄存器对与通用调用地址 （“0000000”）的比较不产生任何影响。

屏蔽寄存器的位 (31:8) 和位 (0) 未使用且不应写入。读取这些位时将始终返回零。

发生地址匹配中断时，处理器必须读数据寄存器 (DAT) 以确定实际导致匹配的已接收地址。

表 256. I²C 屏蔽寄存器 (MASK[0, 1, 2, 3] - 0x4000 00[30, 34, 38, 3C]) 位描述

位	符号	描述	复位值
0	-	保留。用户软件不应向保留位写入 1。读取此位时始终返回零。	0
7:1	MASK	屏蔽位。	0x00
31:8	-	保留。未定义从保留位读取的值。	0

14.8 I²C 操作模式

在给定的应用中，I²C 块可作为主机、从机或同时作主机和从机。在从机模式，I²C 硬件查找其 4 个从属地址中的任何一个地址及通用调用地址。如果检测到这些地址之一，则请求中断。如果处理器希望成为总线主机，则在进入主机模式前，硬件将一直等待，直到总线空闲，这样就不会中断可能的从机操作。如果在主机模式下丢失总线仲裁，则 I²C 模块将立即切换到从机模式并在同一串行传输中检测自身的从机地址。

14.8.1 主发送器模式

此模式下，数据从主机发送到从机。在进入主发送模式前，必须按表 257 所示初始化 CONSET 寄存器。I2EN 必须置 1 以使能 I²C 功能。如果 AA 位为 0，则当另一个器件为总线上的主机时，I²C 接口不会对任何地址作出应答，因此不能进入从机模式。STA、STO 和 SI 位必须为 0。通过将 1 写入 CONCLR 寄存器中的 SIC 位将 SI 位清除。写入从属地址后，应清除 STA 位。

表 257. 用于配置主机模式的 CONSET

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

发送的第一个字节包含接收设备的从属地址（7 位）和数据方向位。在此模式下，数据方向位 (R/W) 应为 0，表示“写入”。发送的第一个字节包含从属地址和“写”位。一次发送 8 位数据。每发送完一个字节后，接收一个确认位。输出“起动”和“停止”条件指示串行传输的开始和结束。

软件置位 STA 位时，I²C 接口将进入主发送模式。一旦总线空闲，I²C 逻辑就会发送起始条件。发送“起动”条件后，SI 位置位，STAT 寄存器中的状态码为 0x08。此状态码用于引导状态服务例程，该例程将从属地址和“写”位加载到 DAT 寄存器，然后清除 SI 位。通过将 1 写入到 CONCLR 寄存器中的 SIC 位清除 SI。

当发送完从属地址和 R/W 位并接收确认位后，SI 位再次设置，此时主机模式下可能的状态码为 0x18、0x20 或 0x38，或者如果使能了从属模式（将 AA 设置为 1），则为 0x68、0x78 或 0xB0。各状态码的相应操作见表 261 ～表 266。

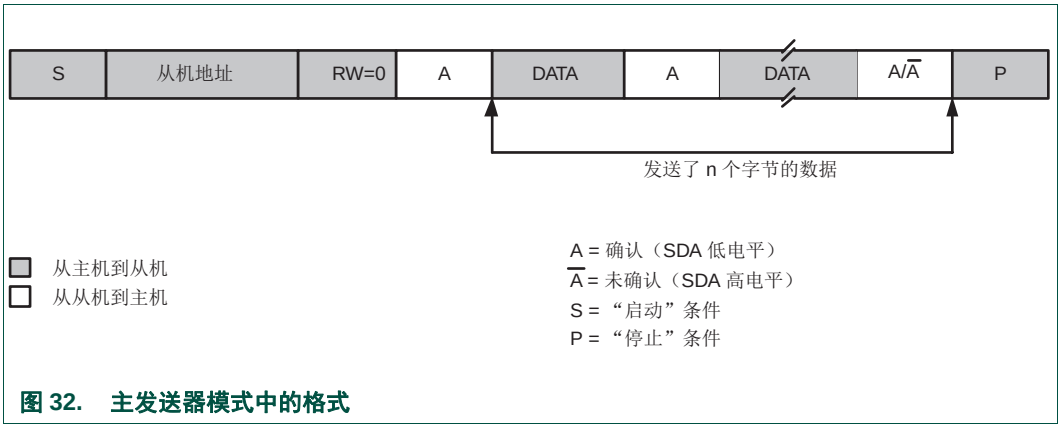


图 32. 主发送器模式中的格式

14.8.2 主接收器模式

在主机接收器模式下，从从机发送器接收数据。传输的发起方式与在主机发送器模式下相同。发送完起始条件后，中断服务程序必须将从属地址和数据方向位装入 I²C 数据寄存器 (DAT)，然后清零 SI 位。在此情况下，数据方向位 (R/W) 应为 1 以表示 “读取”。

发送完从属地址和数据方向位并接收到确认位后，SI 位被设置，状态寄存器将显示状态码。对于主机模式，可能的状态码为 0x40、 0x48 或 0x38。对于从机模式，可能的状态码为 0x68、 0x78 或 0xB0。更多详细信息，请参阅表 262。

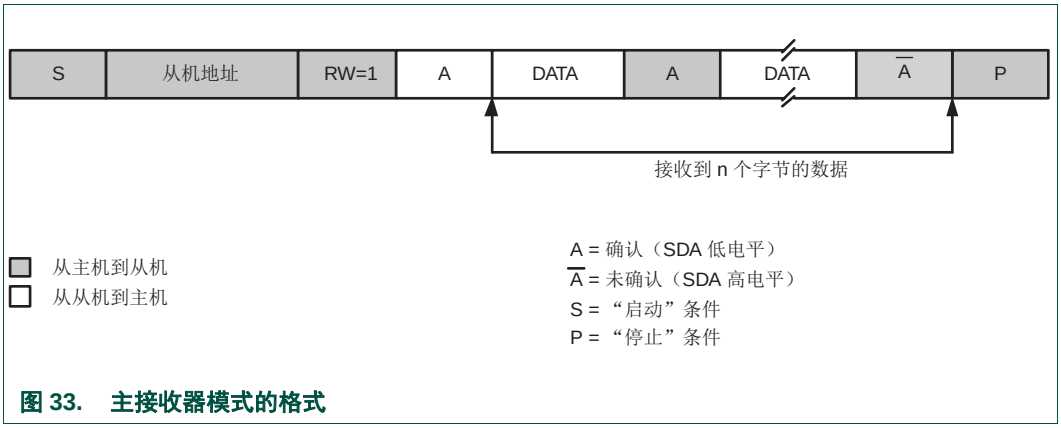


图 33. 主接收器模式的格式

经过一个重复起始条件后， I²C 可切换到主发送模式。

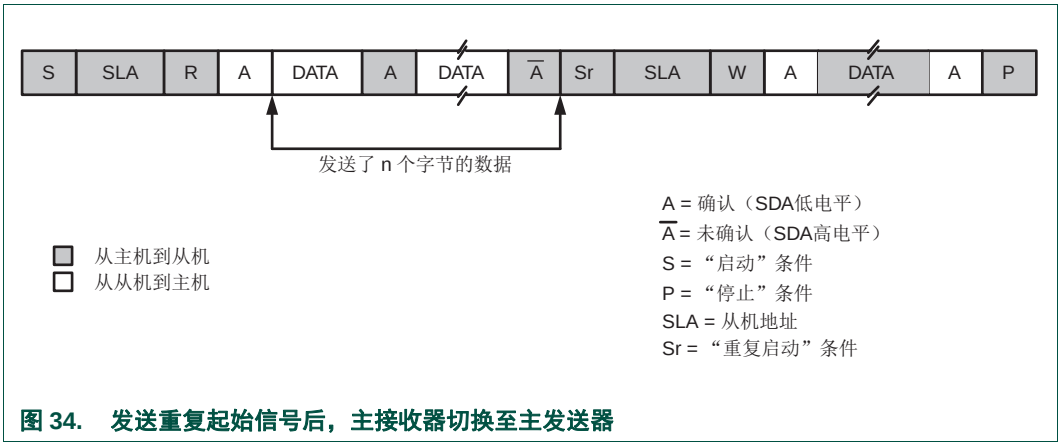


图 34. 发送重复起始信号后，主接收器切换至主发送器

14.8.3 从机接收器模式

在从机接收器模式下，从主机发送器接收数据字节。要初始化从机接收器模式，对任一从机地址寄存器 (ADR0-3) 进行写操作并写 I^2C 控制设置寄存器 (CONSET)，如表 258 所示。

表 258. 用于配置从机模式的 CONSET

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

I2EN 必须置 1 以使能 I^2C 功能。 AA 位必须设置为 1 以确认其自身的从属地址或通用调用地址。 STA、 STO 和 SI 位都设置为 0。

初始化 ADR 和 CONSET 后，I^2C 接口开始等待，直到被其自身地址或后跟数据方向位的通用地址寻址为止。如果方向位为 0 (W)，则其进入从机接收器模式。如果方向位为 1 (R)，则其进入从机发送器模式。接收到地址和方向位后， SI 位设置，并可从状态寄存器 (STAT) 读取一个有效状态码。关于状态码和操作请见表 265。

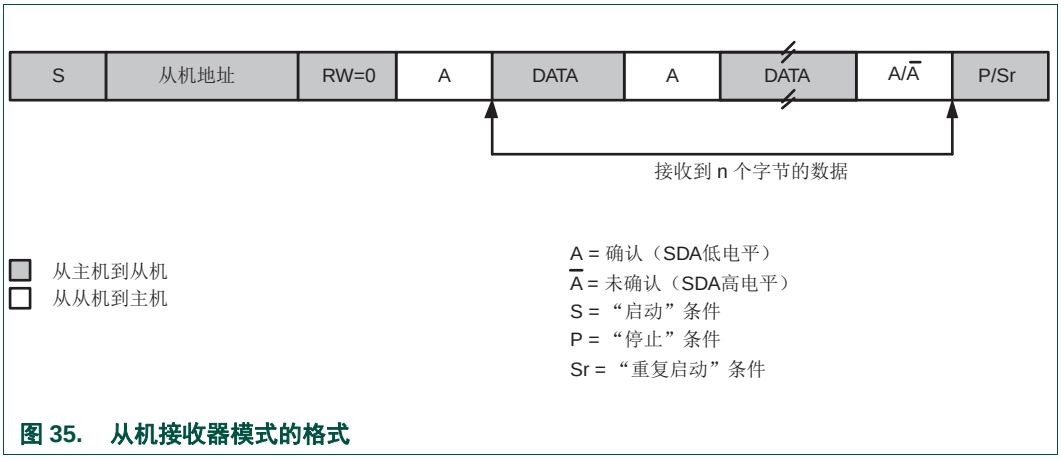


图 35. 从机接收器模式的格式

14.8.4 从机发送器模式

接收和处理第一个字节的方式与从机接收器模式下相同。但在此模式下，方向位为 1，指示读操作。串行数据通过 SDA 发送，同时通过 SCL 输入串行时钟。“起动”和“停止”条件被识别为串行传输的开始和结束。在特定应用中， I^2C 可作为主机 / 从机。在从机模式下，I^2C 硬件查找其自身的从机地址和通用调用地址。如果检测到这些地址之一，则请求中断。当微控制器希望成为总线主机时，在进入主机模式前，硬件将一直等待，直到总线空闲，这样就不会中断可能的从机操作。如果在主机模式下丢失总线仲裁，则 I^2C 接口将立即切换到从机模式并在同一串行传输中检测自身的从机地址。

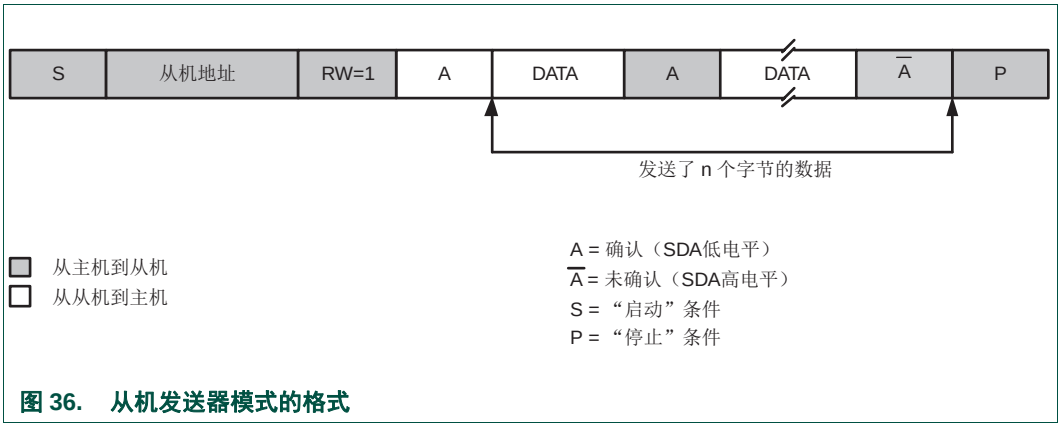


图 36. 从机发送器模式的格式

14.9 I^2C 执行和操作

图 37 所示为片内 I^2C 总线接口的执行流程，下面章节将对图中各模块进行描述。

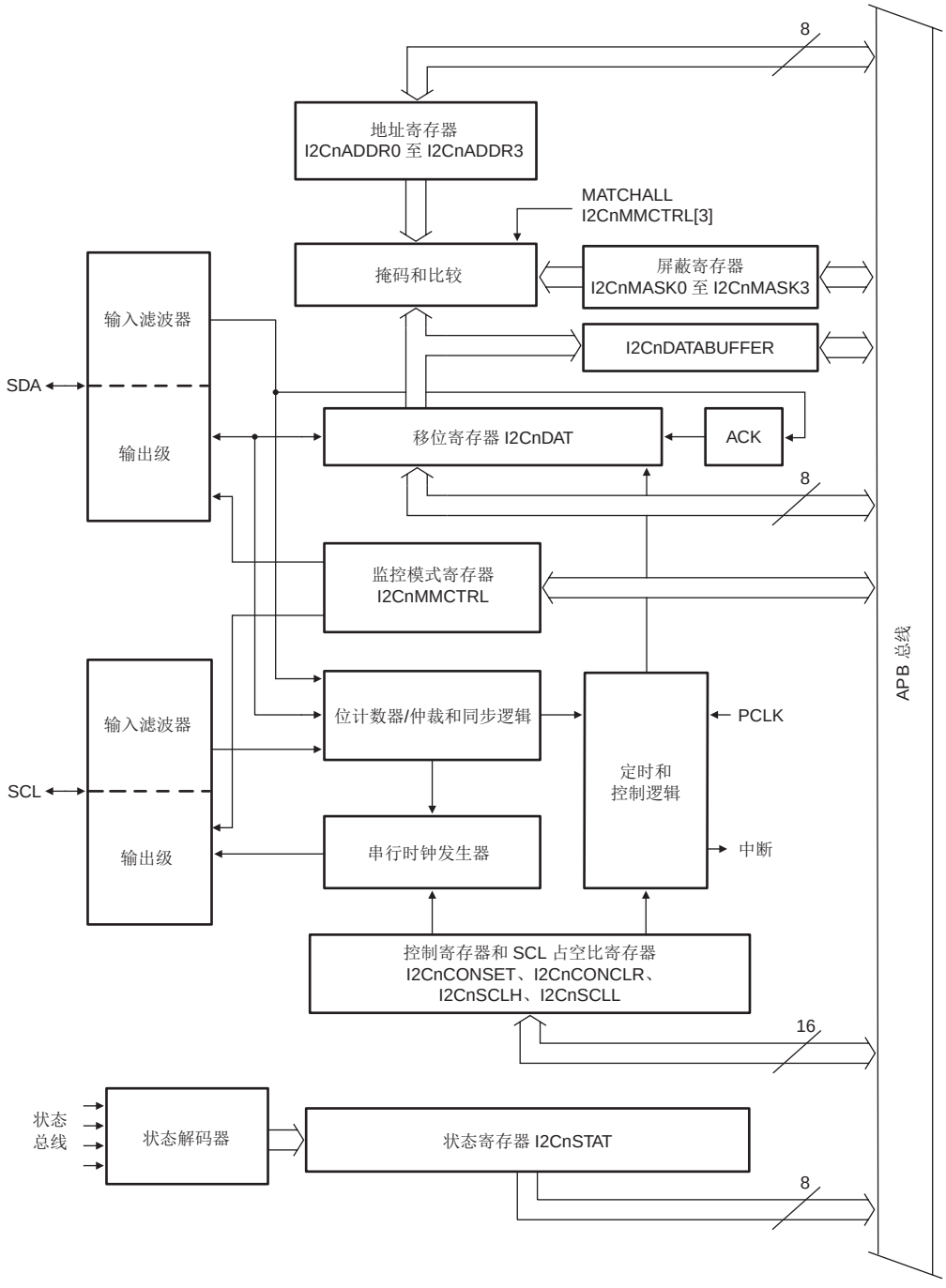


图 37. I²C 串行接口功能框图

14.9.1 输入滤波器与输出级

输入信号与内部时钟同步，小于三个时钟的峰值将被滤出。

I²C 输出是一个特殊焊盘，是为符合 I²C 规范而设计的。

14.9.2 地址寄存器 ADR0 ～ ADR3

当作为从发送器或接收器时，这些寄存器可装入 7 位从属地址（7 个最高有效位），I²C 块将对这些地址作出响应。LSB (GC) 用于使能通用调用地址 (0x00) 识别。使能了多个从机地址后，且已接收到自己的从机地址时，接收的实际地址可从 DAT 寄存器读取。

14.9.3 地址屏蔽寄存器， MASK0 ～ MASK3

四个屏蔽寄存器每个都包含七个有效位 (7:1)。这些屏蔽寄存器与之关联的 ADRn 寄存器相比，其设置为‘1’的任何位都会导致已接收地址的相应位上的自动比较。换句话说，决定地址匹配时不考虑 ADRn 寄存器中被屏蔽的位。

发生地址匹配中断时，处理器必须读数据寄存器 (DAT) 以确定实际导致匹配的已接收地址。

14.9.4 比较器

比较器将接收到的 7 位从属地址与其自身的从属地址（ADR 中的 7 个最高有效位）进行比较，它还将第一次收到的 8 位字节与通用调用地址 (0x00) 进行比较。如果发现相等，则设置相应的状态位并请求中断。

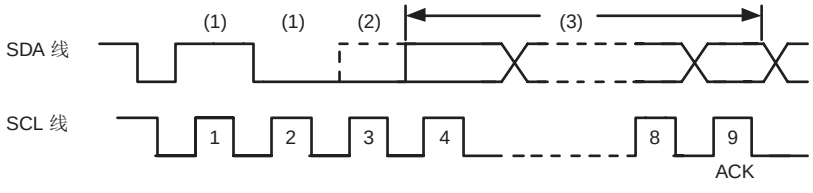
14.9.5 移位寄存器， DAT

此 8 位寄存器包含一个要发送的串行数据字节或一个刚接收到的字节。 DAT 中的数据始终从右向左移位。要发送的第一位是 MSB （位 7），在收到一个字节后，已接收数据的第一位放在 DAT 的 MSB 上。当数据被移出时，总线上的数据同时移入； DAT 始终包含总线上出现的最后一个字节。因此，在仲裁丢失的情况下，会用 DAT 中的正确数据进行从主机发送器到从机接收器的跳变。

14.9.6 仲裁与同步逻辑

在主发送模式下，仲裁逻辑校验每个发送的逻辑 1 在 I²C 总线上是否真正以逻辑 1 出现。如果总线上另一个器件否定逻辑 1 并将 SDA 线拉低，则仲裁丢失， I²C 块立即由主发送器转换成从接收器。I²C 块将继续输出时钟脉冲（在 SCL 上），直到发送完当前串行字节为止。

主机接收器模式下也可能丢失仲裁。在该模式下，只有在 I²C 模块向总线返回一个 “无应答”：（逻辑 1）才会发生仲裁丢失。当总线上另一设备将此信号拉低时，仲裁丢失。由于这只会在串行字节结束时出现，因此 I²C 块不再产生时钟脉冲。图 38 所示为仲裁过程。



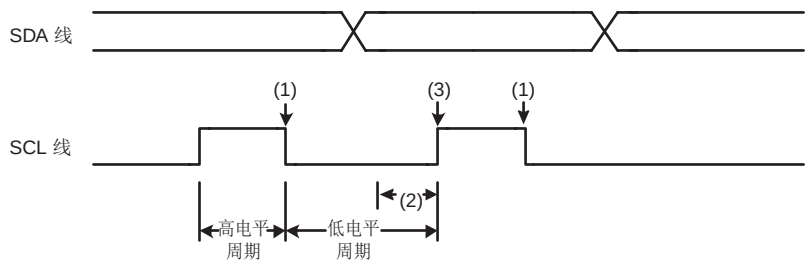
(1) 另一设备发送串行数据。

(2) 另一个器件通过将 SDA 线拉低撤消了该 I²C 主机发送的一个逻辑 （虚线）。仲裁丢失，该 I²C 进入从接收模式。

(3) 该 I²C 为从接收模式，但仍产生时钟脉冲，直到当前字节发送完为止。该 I²C 不会为下一个字节产生时钟脉冲。赢得仲裁后， SDA 上的数据由新主机产生。

图 38. 仲裁过程

同步逻辑将使串行时钟发生器与来自另一设备的 SCL 线上的时钟脉冲同步。如果两个或多个主机设备产生时钟脉冲，则“标记”持续时间由产生最短“标记”的设备确定，“间隔”持续时间由产生最长 “间隔”的设备确定。图 39 所示为同步过程。



(1) 另一个器件在 I²C 定时一个完整的高电平时间前将 SCL 线拉低。其它设备有效地确定了（较短）高电平期间。

(2) 另一个器件在 I²C 定时一个完整的低电平时间并释放 SCL 后继续将 SCL 线拉低。I²C 时钟发生器必须等待，直到 SCL 变为高电平。其它设备有效地确定了 （较长）低电平期间。

(3) SCL 线被释放，时钟发生器开始定时高电平时间。

图 39. 串行时钟同步

从机可延长间隔持续时间以减慢总线主机，也可以延长间隔持续时间以实现信号交换目的。此操作可在每位或一个完整字节传输后进行。在发送或接收完一个字节且确认位已传输后，I²C 块将延长 SCL 间隔持续时间。串行中断标志 (SI) 被设置，且延长继续进行直到串行中断标志被清除。

14.9.7 串行时钟生成器

当 I²C 块处于主发送或主接收模式时，可编程时钟脉冲发生器提供 SCL 时钟脉冲。当 I²C 块处于从机模式时，时钟脉冲发生器关闭。I²C 输出时钟频率和占空比可通过 I²C 时钟控制寄存器编程。详情请参见 I2CSCLL、I2CSCLH 寄存器描述。除非总线与上述其它 SCL 时钟源同步，输出时钟脉冲的占空比均会按照预先编程确定的情况出现。

14.9.8 定时与控制

定时和控制逻辑为串行字节处理产生定时和控制信号。该逻辑块为 DAT 提供移位脉冲，可使能比较器、产生并检测起始和停止条件、接收并发送应答位、控制主 / 从机模式，还包含中断请求逻辑并监控 I²C 总线状态。

14.9.9 控制寄存器 CONSET 和 CONCLR

I²C 控制寄存器包含用于控制以下 I²C 块功能的位：串行传输的起动和重启、串行传输终止、比特率、地址识别及确认。

I²C 控制寄存器的内容可能读出为 CONSET。写入 CONSET 将设置 I²C 控制寄存器中的位，这些位与写入值中的位相对应。反之，写入 CONCLR 将清零 I²C 控制寄存器中的位，这些位与写入值中的位相对应。

14.9.10 状态解码器与状态寄存器

状态解码器获取所有内部状态位并将其压缩成 5 位代码，该代码与各 I^2C 总线状态一一对应。该 5 位代码可用于产生向量地址，以快速处理各种服务例程。每个服务例程处理一个特定的总线状态。如果使用 I^2C 块的所有 4 种模式，则存在 26 种可能的总线状态。串行中断标志设置 （通过硬件）后，该 5 位状态码被锁存到状态寄存器的 5 个最高有效位并保持稳定，直到中断标志被软件清除。状态寄存器的三个最低有效位始终为零。如果状态码用作服务例程的向量，则这些例程由八个地址位置替代。对于大多数的服务例程，八个字节的代码足够 （参见本节的软件示例）。

14.10 I^2C 操作模式详解

四种操作模式为：

- 主机发送器
- 主机接收器
- 从机接收器
- 从机发送器

各模式下数据传输操作如图 40、图 41、图 42、图 43 和图 44 所示。表 259 说明了介绍 I^2C 操作模式的图中所使用缩写的含义。

表 259. 用于描述 I^2C 操作的缩写

缩写	描述
S	起始条件
SLA	7 位从属地址
R	读取位 （SDA 为高电平）
W	写入位 （SDA 低电平）
A	应答位 （SDA 低电平）
$\overline{A}$	未确认位 （SDA 为高电平）
数据	8 位数据字节
P	“停止”条件

在图 40 ～图 44 中，圆圈用来指示串行中断标志何时被置位。圆圈中的数字显示 STAT 寄存器中保留的状态码。在这些点，必须执行服务例程以继续或完成串行传输。这些服务例程不是至关重要的，因为串行传输被挂起，直到串行中断标志被软件清除。

当进入串行中断例程时， STAT 中的状态码用于分支到相应的服务例程。对于每个状态代码，需要的软件操作以及后面串行传输的详细情况见表 261 ～表 267。

14.10.1 主发送器模式

在主发送器模式中，向从机接收器发送数据字节 （见图 40）。先按下列信息初始化 I2CON 后，才能进入主发送器模式：

表 260. 用于初始化主发送模式的 CONSET

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

I²C 速率也必须在 SCLL 和 SCLH 寄存器中配置。必须将 I2EN 设置为逻辑 1 来使能 I²C 块。如果 AA 位复位，则当另一个器件正变成总线主机时，I²C 模块将不会应答其自身的从机地址或通用调用地址。也就是说，如果 AA 位复位，则 I²C 接口就不能进入从机模式。 STA、STO 和 SI 必须复位。

现在即可通过设置 STA 位进入主机发送器模式。一旦总线空闲，I²C 逻辑会立即测试 I²C 总线并产生一个起始条件。当 “起始” 条件发送后，串行中断标志 (SI) 设置，状态寄存器 (STAT) 中的状态码为 0x08。中断服务例程使用此状态码进入相应的状态服务例程，该例程将从属地址和数据方向位 (SLA+W) 载入 DAT。随后必须复位 CON 中的 SI 位，之后才能继续串行传输。

从属地址和方向位发送完且接收到确认位后，串行中断标志 (SI) 再次置位，STAT 中可能存在许多状态码。主机模式下有 0x18、0x20 或 0x38，如果使能了从机模式（AA = 逻辑 1），则有 0x68、0x78 或 0xB0。表 261 中详细介绍了每个状态代码对应的操作。在发送完重复起始条件 （状态 0x10）后， I²C 块通过将 SLA+R 装入 DAT 切换到主接收器模式。

表 261. 主发送器模式

状态代码 (I2CSTAT)	I²C 总线和硬件的状态	应用软件响应					I²C 硬件执行的下一个操作
		至 / 自 DAT	至 CON				
			STA	STO	SI	AA	
0x08	“起动” 条件已发送。	加载 SLA+W；清除 STA	X	0	0	X	将发送 SLA+W；将接收 ACK 位。
0x10	“重复起始” 条件已发送。	加载 SLA+W 或	X	0	0	X	同上。
		加载 SLA+R；清除 STA	X	0	0	X	将发送 SLA+R； I²C 模块将切换为 MST/REC 模式。
0x18	SLA+W 已发送；ACK 已接收。	加载数据字节或	0	0	0	X	将发送数据字节；将接收 ACK 位。
		无 DAT 操作或	1	0	0	X	将发送 “重复起始”。
		无 DAT 操作或	0	1	0	X	将发送“停止”条件； STO 标志将复位。
		无 DAT 操作	1	1	0	X	“停止” 条件之后将发送 “起动” 条件；STO 标志将复位。
0x20	已发送 SLA+W；已接收 NOT ACK。	加载数据字节或	0	0	0	X	将发送数据字节；将接收 ACK 位。
		无 DAT 操作或	1	0	0	X	将发送 “重复起始”。
		无 DAT 操作或	0	1	0	X	将发送“停止”条件； STO 标志将复位。
		无 DAT 操作	1	1	0	X	“停止” 条件之后将发送 “起动” 条件；STO 标志将复位。
0x28	DAT 中的数据字节已发送； ACK 已接收。	加载数据字节或	0	0	0	X	将发送数据字节；将接收 ACK 位。
		无 DAT 操作或	1	0	0	X	将发送 “重复起始”。
		无 DAT 操作或	0	1	0	X	将发送“停止”条件； STO 标志将复位。
		无 DAT 操作	1	1	0	X	“停止” 条件之后将发送 “起动” 条件；STO 标志将复位。
0x30	DAT 中的数据字节已发送； NOT ACK 已接收。	加载数据字节或	0	0	0	X	将发送数据字节；将接收 ACK 位。
		无 DAT 操作或	1	0	0	X	将发送 “重复起始”。
		无 DAT 操作或	0	1	0	X	将发送“停止”条件； STO 标志将复位。
		无 DAT 操作	1	1	0	X	“停止” 条件之后将发送 “起动” 条件；STO 标志将复位。
0x38	SLA+R/W 或数据字节操作中仲裁丢失。	无 DAT 操作或	0	0	0	X	I²C 总线将被释放；进入不可寻址从机模式。
		无 DAT 操作	1	0	0	X	当总线空闲时将发送 “起动”。

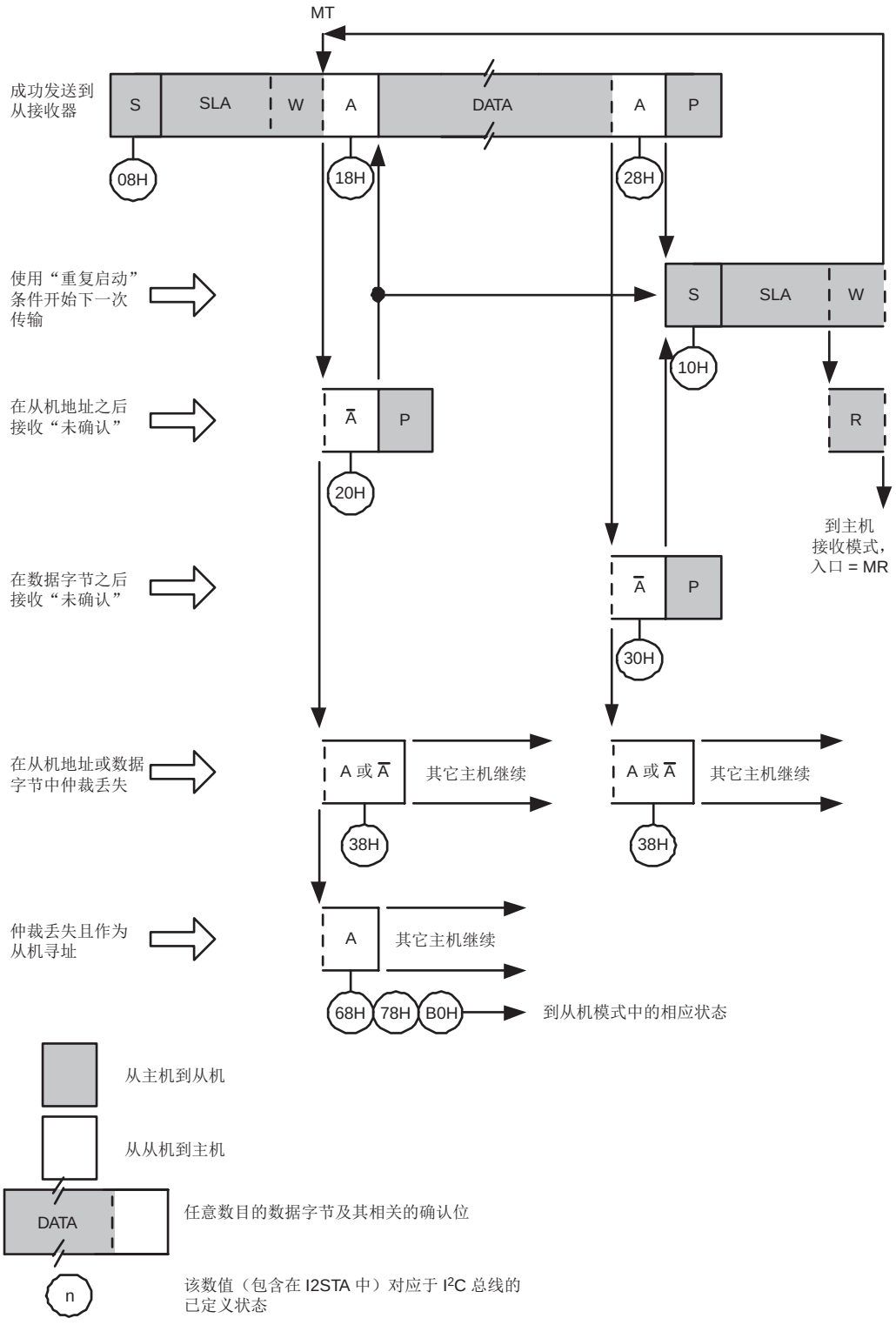


图 40. 主发送器模式下的格式和状态

14.10.2 主接收器模式

在主接收器模式中，主机所接收的数据字节来自从机发送器 （见图 41）。传输的初始化与主机发送器模式下相同。发送完 “起动”条件后，中断服务例程必须将 7 位从属地址和数据方向位 (SLA+R) 载入 DAT 中。随后必须清除 CON 中的 SI 位，之后才能继续串行传输。

从属地址和数据方向位发送完且接收到确认位后，串行中断标志 (SI) 再次置位，STAT 中可能存在许多状态码。主机模式下为 0x40、 0x48 或 0x38，如果使能了从机模式 (AA = 1)，为 0x68、0x78 或 0xB0。表 262 中详细介绍了每个状态代码对应的操作。在发送完重复起始条件 （状态 0x10）后， I^2C 模块通过将 SLA+W 装入 DAT 切换到主机发送器模式。

表 262. 主接收器模式

状态码 (STAT)	I^2C 总线和硬件的状态	应用软件响应					I^2C 硬件执行的下一个操作
		至 / 自 DAT	至 CON				
			STA	STO	SI	AA	
0x08	“起动”条件已发送。	加载 SLA+R	X	0	0	X	将发送 SLA+R ；将接收 ACK 位。
0x10	“重复起始”条件已发送。	加载 SLA+R 或	X	0	0	X	同上。
		加载 SLA+W	X	0	0	X	将发送 SLA+W ； I^2C 块将切换为 MST/TRX 模式。
0x38	在 NOT ACK 位中丢失仲裁。	无 DAT 操作或	0	0	0	X	I^2C 总线将被释放； I^2C 模块进入从机模式。
		无 DAT 操作	1	0	0	X	当总线空闲时将发送 “起动”条件。
0x40	SLA+R 已发送；ACK 已接收。	无 DAT 操作或	0	0	0	0	将接收数据字节；将返回 NOT ACK 位。
		无 DAT 操作	0	0	0	1	将接收数据字节；将返回 ACK 位。
0x48	SLA+R 已发送；NOT ACK 已接收。	无 DAT 操作或	1	0	0	X	将发送 “重复起始”条件。
		无 DAT 操作或	0	1	0	X	将发送“停止”条件； STO 标志将复位。
		无 DAT 操作	1	1	0	X	“停止”条件之后将发送“起动”条件；STO 标志将复位。
0x50	数据字节已接收；ACK 已返回。	读取数据字节或	0	0	0	0	将接收数据字节；将返回 NOT ACK 位。
		读取数据字节	0	0	0	1	将接收数据字节；将返回 ACK 位。
0x58	数据字节已接收；NOT ACK 已返回。	读取数据字节或	1	0	0	X	将发送 “重复起始”条件。
		读取数据字节或	0	1	0	X	将发送“停止”条件； STO 标志将复位。
		读取数据字节	1	1	0	X	“停止”条件之后将发送“起动”条件；STO 标志将复位。

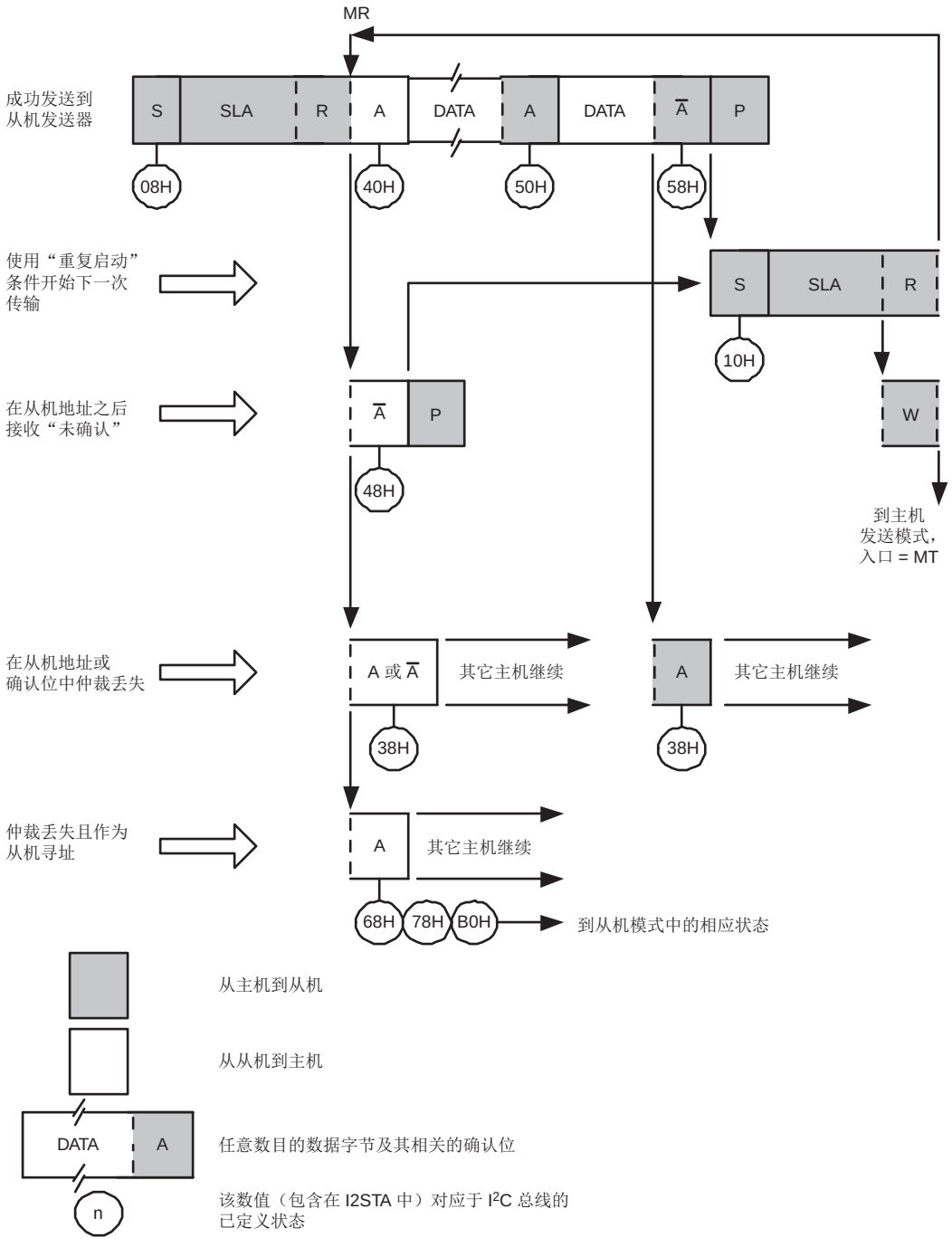


图 41. 主接收器模式下的格式和状态

14.10.3 从机接收器模式

在从机接收器模式中，从机所接收的数据字节来自主发送器 （见图 42）。要初始化从机接收器模式，必须按如下所示加载 ADR 和 CON：

表 263. 从机接收器模式下的 ADR 使用

位	7	6	5	4	3	2	1	0
符号	自身的 7 位从属地址							GC

高 7 位是主机寻址时 I²C 模块将要响应的地址。如果设置了 LSB (GC)，则 I²C 模块将响应通用调用地址 (0x00)；否则它将忽略该通用调用地址。

表 264. 用于初始化从机接收器模式的 CONSET

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

I²C 总线速率的设置不影响从机模式中的 I²C 块。必须将 I2EN 设置为逻辑 1 来使能 I²C 块。AA 位必须置位以使能 I²C 块来应答其自身从属地址或通用调用地址。 STA、 STO 和 SI 必须复位。

当初始化 ADR 和 CON 后，I²C 模块一直等待，直至被自身的从机地址寻址，之后是数据方向位，该数据方向位必须为 “0” (W)，以便 I²C 模块在从机接收器模式下工作。接收到其自身的从属地址和 W 位后，将设置串行中断标记 (SI)，并可从 STAT 读取一个有效状态码。此状态码用作状态服务例程的向量。表 265 中详细介绍了每个状态代码对应的操作。如果当 I²C 块在主机模式中时仲裁丢失，也可进入从接收模式 （见状态 0x68 和 0x78）。

如果 AA 位在传输过程中复位，则在接收完下一个数据字节后 I²C 块将向 SDA 返回一个非应答 （逻辑 1）。当 AA 复位时， I²C 块不响应其自身的从属地址或通用调用地址。但是，I²C 总线仍被监控，而且，地址识别可随时通过置位 AA 来恢复。这就意味着 AA 位可临时将 I²C 块从 I²C 总线上分离出来。

表 265. 从机接收器模式

状态码 (STAT)	I²C 总线和硬件的状态	应用软件响应					I²C 硬件执行的下一个操作
		至 / 自 DAT	至 CON				
			STA	STO	SI	AA	
0x60	已接收自身的 SLA+W，已返回 ACK。	无 DAT 操作或	X	0	0	0	将接收数据字节并返回 NOT ACK。
		无 DAT 操作	X	0	0	1	将接收数据字节并返回 ACK。
0x68	作为主机，在 SLA+R/W 中仲裁丢失；自身的 SLA+W 已接收，ACK 已返回。	无 DAT 操作或	X	0	0	0	将接收数据字节并返回 NOT ACK。
		无 DAT 操作	X	0	0	1	将接收数据字节并返回 ACK。
0x70	通用调用地址 (0x00) 已接收； ACK 已返回。	无 DAT 操作或	X	0	0	0	将接收数据字节并返回 NOT ACK。
		无 DAT 操作	X	0	0	1	将接收数据字节并返回 ACK。
0x78	作为主机，在 SLA+R/W 中仲裁丢失；通用调用地址已接收， ACK 已返回。	无 DAT 操作或	X	0	0	0	将接收数据字节并返回 NOT ACK。
		无 DAT 操作	X	0	0	1	将接收数据字节并返回 ACK。
0x80	先前已使用自身的 SLV 地址进行寻址；DATA已接收； ACK 已返回。	读取数据字节或	X	0	0	0	将接收数据字节并返回 NOT ACK。
		读取数据字节	X	0	0	1	将接收数据字节并返回 ACK。

表 265. 从机接收器模式 *（续）*

状态码 (STAT)	I²C 总线和硬件的状态	应用软件响应					I²C 硬件执行的下一个操作
		至 / 自 DAT	至 CON				
			STA	STO	SI	AA	
0x88	先前已使用自身的SLA进行寻址； DATA 字节已接收； NOT ACK 已返回。	读取数据字节或	0	0	0	0	切换到未寻址的 SLV 模式；不识别自身 SLA 或通用调用地址。
		读取数据字节或	0	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA ；如果 ADR[0] = 逻辑 1，将识别通用调用地址。
		读取数据字节或	1	0	0	0	切换到未寻址的 SLV 模式；不识别自身 SLA 或通用调用地址。当总线空闲时将发送 “起动”条件。
		读取数据字节	1	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果ADR[0] = 逻辑 1，将识别通用调用地址；当总线空闲后发送起始条件。
0x90	先前已使用通用调用进行寻址；DATA字节已接收； ACK 已返回。	读取数据字节或	X	0	0	0	将接收数据字节并返回 NOT ACK。
		读取数据字节	X	0	0	1	将接收数据字节并返回 ACK。
0x98	先前已使用通用调用进行寻址； DATA 字节已接收； NOT ACK 已返回。	读取数据字节或	0	0	0	0	切换到未寻址的 SLV 模式；不识别自身 SLA 或通用调用地址。
		读取数据字节或	0	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA ；如果 ADR[0] = 逻辑 1，将识别通用调用地址。
		读取数据字节或	1	0	0	0	切换到未寻址的 SLV 模式；不识别自身 SLA 或通用调用地址。当总线空闲时将发送 “起动”条件。
		读取数据字节	1	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果ADR[0] = 逻辑 1，将识别通用调用地址；当总线空闲后发送起始条件。
0xA0	已接收 “停止”条件或 “重复起始”条件，但仍作为 SLV/REC 或 SLV/TRX 寻址。	无 STDAT 操作或	0	0	0	0	切换到未寻址的 SLV 模式；不识别自身 SLA 或通用调用地址。
		无 STDAT 操作或	0	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果ADR[0] = 逻辑 1，将识别通用调用地址。
		无 STDAT 操作或	1	0	0	0	切换到未寻址的 SLV 模式；不识别自身 SLA 或通用调用地址。当总线空闲时将发送 “起动”条件。
		无 STDAT 操作	1	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA；如果ADR[0] = 逻辑 1，将识别通用调用地址；当总线空闲后发送起始条件。

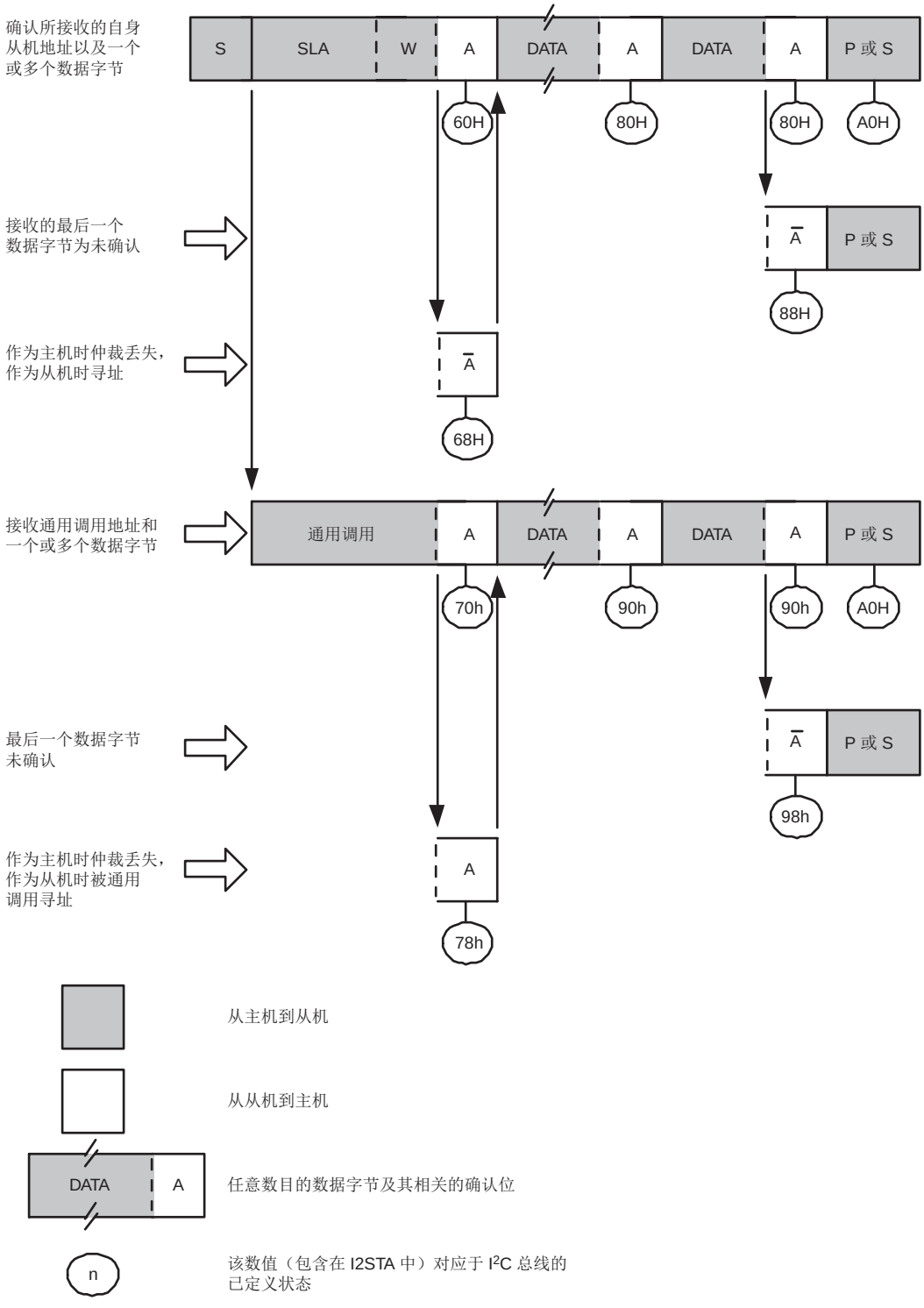


图 42. 从机接收模式下的格式和状态

14.10.4 从机发送器模式

在从机发送模式中，向主接收器发送数据字节 （见图 43）。数据传输的初始化与从机接收器模式下相同。当初始化 ADR 和 CON 后，I^2C 块一直等待，直至被自身的从机地址寻址，之后是数据方向位，该数据方向位必须为 “1” (R)，以便 I^2C 模块在从机发送器模式下工作。接收到其自身的从属地址和 R 位后，将设置串行中断标记 (SI)，并可从 STAT 读取一个有效状态码。该状态代码用作状态服务程序的向量，每个状态代码的对应操作详见表 266。如果当 I^2C 块在主机模式中时仲裁丢失，也可进入从发送模式 （见状态 0xB0）。

如果 AA 位在传输过程中复位，则 I^2C 块将发送最后一个字节并进入状态 0xC0 或 0xC8。I^2C 块切换到非寻址的从机模式，如果继续传输，它将忽略主接收器。从而主机接收器作为串行数据接收所有 1。当 AA 复位时， I^2C 块不响应其自身的从属地址或通用调用地址。但是， I^2C 总线仍被监控，而且，地址识别可随时通过置位 AA 来恢复。这就意味着 AA 位可临时将 I^2C 块从 I^2C 总线上分离出来。

表 266. 从机发送器模式

状态码 (STAT)	I^2C 总线和硬件的状态	应用软件响应					I^2C 硬件执行的下一个操作
		至 / 自 DAT	至 CON				
			STA	STO	SI	AA	
0xA8	自身的SLA+R已接收；ACK 已返回。	加载数据字节或	X	0	0	0	将发送最后一个数据字节并接收ACK位。
		加载数据字节	X	0	0	1	将发送数据字节；将接收 ACK。
0xB0	作为主机，在SLA+R/W中仲裁丢失；自身的SLA+R已接收，ACK已返回。	加载数据字节或	X	0	0	0	将发送最后一个数据字节并接收ACK位。
		加载数据字节	X	0	0	1	将发送数据字节；将接收 ACK 位。
0xB8	DAT 中的数据字节已发送； ACK 已接收。	加载数据字节或	X	0	0	0	将发送最后一个数据字节并接收ACK位。
		加载数据字节	X	0	0	1	将发送数据字节；将接收 ACK 位。
0xC0	DAT 中的数据字节已发送；NOT ACK已接收。	无 DAT 操作或	0	0	0	0	切换到未寻址的 SLV 模式；不识别自身 SLA 或通用调用地址。
		无 DAT 操作或	0	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA ；如果 ADR[0] = 逻辑 1，将识别通用调用地址。
		无 DAT 操作或	1	0	0	0	切换到未寻址的 SLV 模式；不识别自身 SLA 或通用调用地址。当总线空闲时将发送 “起动”条件。
		无 DAT 操作	1	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA ；如果 ADR[0] = 逻辑 1，将识别通用调用地址；当总线空闲后发送起始条件。
0xC8	DAT 中最后一个数据字节已发送 (AA = 0) ；ACK 已接收。	无 DAT 操作或	0	0	0	0	切换到未寻址的 SLV 模式；不识别自身 SLA 或通用调用地址。
		无 DAT 操作或	0	0	0	1	切换到不可寻址 SLV 模式；识别自身 SLA ；如果 ADR[0] = 逻辑 1，将识别通用调用地址。
		无 DAT 操作或	1	0	0	0	切换到未寻址的 SLV 模式；不识别自身 SLA 或通用调用地址。当总线空闲时将发送 “起动”条件。
		无 DAT 操作	1	0	0	01	切换到不可寻址 SLV 模式；识别自身 SLA ；如果 ADR.0 = 逻辑 1，将识别通用调用地址；当总线空闲后发送起始条件。

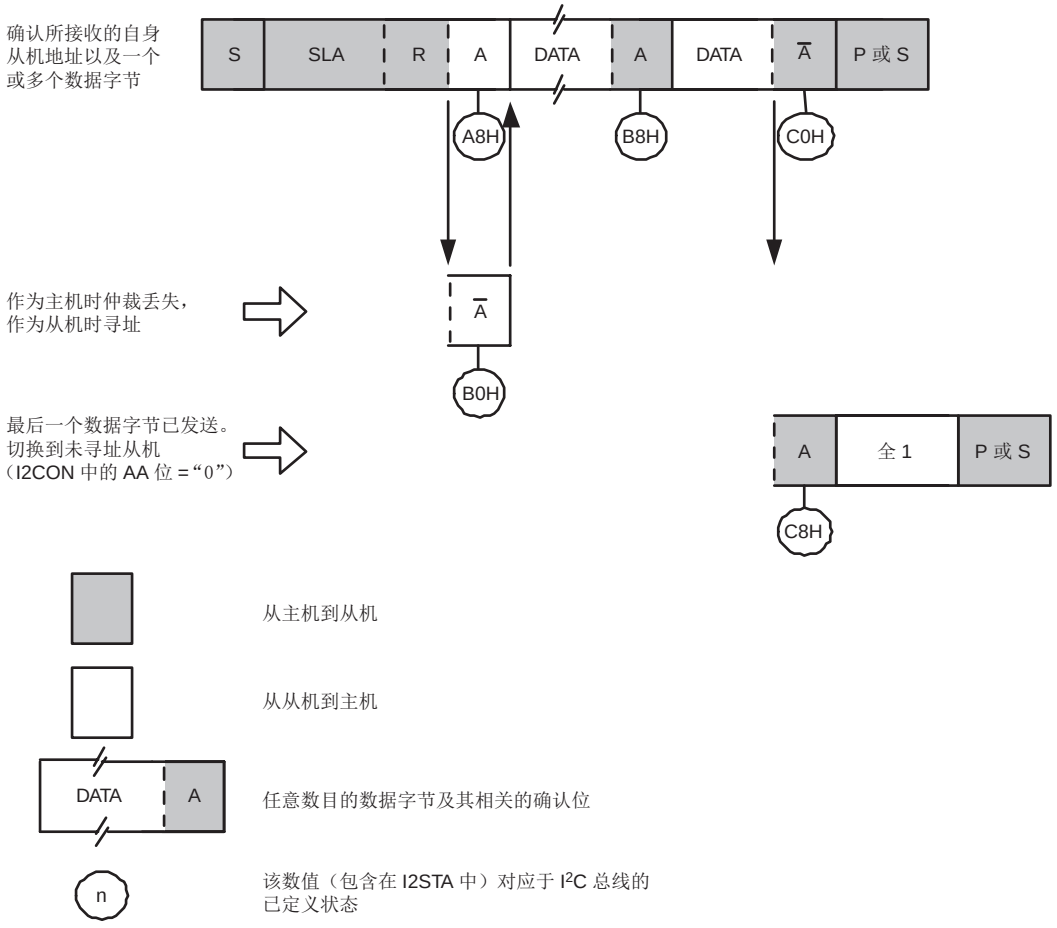


图 43. 从发送模式下的格式和状态

14.10.5 其它状态

还有两种 STAT 代码与已定义的 I^2C 硬件状态不对应 （见表 267）。论述如下。

14.10.5.1 STAT = 0xF8

此状态码表示没有任何可用的相关信息，因为串行中断标记 SI 尚未置位。这种情况在其它状态和 I^2C 块还未开始执行串行传输之间出现。

14.10.5.2 STAT = 0x00

该状态代码表示在 I^2C 串行传输过程中出现了总线错误。当 “起动”或 “停止”条件发生在格式帧的非法位置上时会导致总线错误。此类非法位置的示例有在地址字节、数据字节或确认位的串行传输期间。当外部干扰影响到内部 I^2C 块信号时也会产生总线错误。发生总线错误时 SI 设置。要从总线错误中恢复， STO 标志必须设置且 SI 必须清除。这使得 I^2C 模块进入 “非寻址”的从机模式（已定义的状态）并清除 STO 标志（CON 中的其他位不受影响）。 SDA 和 SCL 线被释放 （不发送 “停止”条件）。

表 267. 其它状态

状态码 (STAT)	I^2C 总线和硬件的状态	应用软件响应					I^2C 硬件执行的下一个操作
		至 / 自 DAT	至 CON				
			STA	STO	SI	AA	
0xF8	无可用的相关状态信息； SI = 0。	无 DAT 操作	无 CON 操作				等待或继续进行当前传输。
0x00	由于非法的 “起动”或 “停止”条件，在 MST 或所选的从机模式期间总线出错。当外部干扰使 I^2C 块进入未定义的状态时也出现 0x00 状态。	无 DAT 操作	0	1	0	X	在 MST 或寻址的 SLV 模式下，仅内部硬件受影响。在所有情况下，总线被释放、 I^2C 块切换到非寻址的 SLV 模式。 STO 复位。

14.10.6 某些特殊情况

I²C 硬件可以处理串行传输过程中出现的以下几种特殊情况：

- 来自两个主机的同时 “重复起始”条件
- 仲裁丢失后进行数据传输
- 强制访问 I²C 总线
- SCL 或 SDA 低电平妨碍 I²C 总线的操作
- 总线错误

14.10.6.1 来自两个主机的同时 “重复起始”条件

在主机发送器模式或主机接收器模式下可以产生 “重复起始”条件。如果此时另一个主机同时产生重复起始条件，就出现特殊情况 （参见图 44）。因为两台主机发送相同的数据，所以任一台主机都不会丢失仲裁，直到出现了这种情况。

如果 I²C 硬件在产生重复起始条件之前在 I²C 总线上检测到重复起始条件，则它将释放总线，并且不产生中断请求。如果另一个主机通过产生停止条件来释放总线，则 I²C 块将发送一个正常的起始条件 （状态 0x08），并开始重新进行完整的串行数据传输。

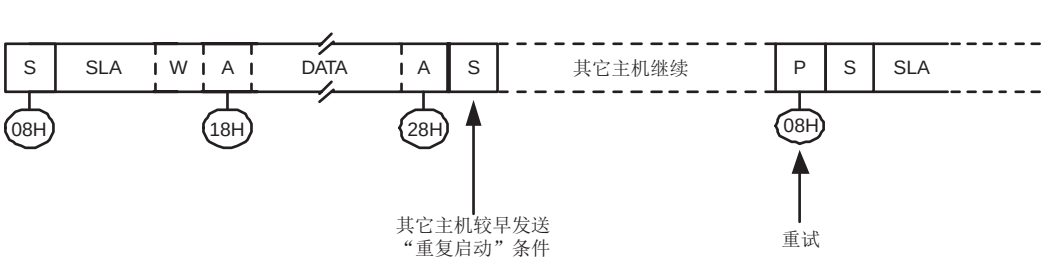


图 44. 来自两个主机的同时 “重复起始”条件

14.10.6.2 仲裁丢失后进行数据传输

在主发送模式和主接收模式中仲裁可能会丢失 （见图 38）。 STAT 寄存器中的状态代码可表示仲裁丢失，代码有：0x38、 0x68、 0x78 和 0xB0 （见图 40 和图 41）。

如果 CON 中的 STA 标志由服务于这些状态的例程设置，则当总线再次空闲时，会在不受 CPU 干扰下发送 “起动”条件 （状态 0x08），并可开始重试整个串行传输。

14.10.6.3 强制访问 I²C 总线

在一些应用中，未控制源可能会导致总线挂起。在这种情况下，干扰、总线临时中断或 SDA 和 SCL 之间的临时短路都可能导致总线挂起。

如果不可控制源产生了一个多余的起始条件或屏蔽了一个停止条件，则 I²C 总线一直保持忙碌状态。如果 STA 标志置位且在相应的时间内未访问总线，那么 I² 总线有可能会被强制访问。这通过在 STA 标志仍被设置的情况下设置 STO 标志来实现。不发送 “停止”条件。I²C 硬件的操作就好像是接收到停止条件一样，可以发送起始条件。 STO 标志由硬件清除（参见图 45）。

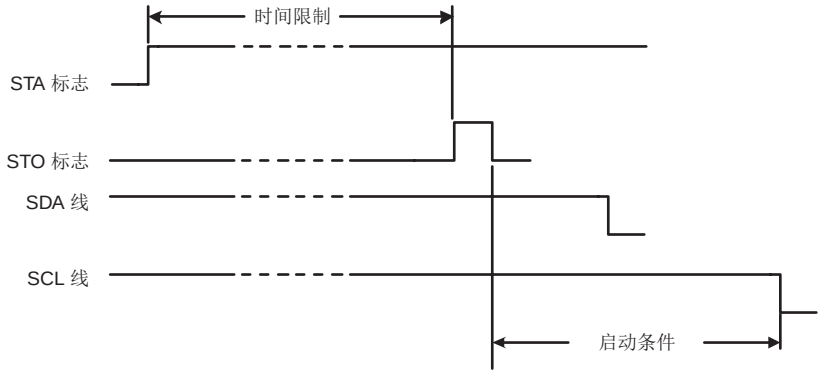
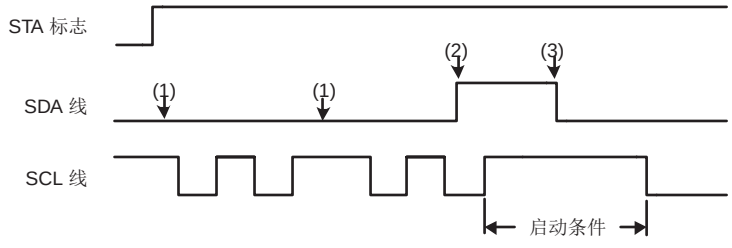


图 45. 强制访问忙 I²C 总线

14.10.6.4 SCL 或 SDA 低电平妨碍 I²C 总线的操作

如果 SDA 或 SCL 线被总线上任何设备保持为低电平，则 I²C 总线会挂起。如果 SCL 线被总线上的设备阻塞 （拉低），则不能继续进行任何串行传输，且必须通过拉低 SCL 总线的设备解决此问题。

通常，SDA 线可能被总线上的另一与当前总线主机不同步的设备（由于丢失时钟或将噪声脉冲感测为时钟）阻塞。在这种情况下，可通过在 SCL 线上发送其它时钟脉冲解决问题（参见图 46）。I²C 接口不包含检测受阻总线的专用超时定时器，但可以使用系统中的另一个定时器来执行。检测时，软件可在 SCL 上强制时钟 （可要求多达 9 个），直到 SDA 被不良设备释放。此时，从机可能还是不同步，所以还要发送一个起始条件以确保所有 I²C 外设同步。



(1) 尝试发送一个起始条件不成功。
(2) SDA 线被释放。
(3) 尝试发送一个起始条件成功。进入状态 08H。

图 46. 从因 SDA 低电平而导致总线受阻的状态中恢复

14.10.6.5 总线错误

当格式帧的非法位置上检测到 “起动”或 “停止”条件时会出现总线错误。非法位置的示例有在地址字节、数据位或确认位的串行传输期间。

仅当 I^2C 硬件作为主机或被寻址的从机进行串行传输时，它才对总线错误有反应。检测到总线错误时， I^2C 块会立即切换成非寻址的从机模式，并释放 SDA 和 SCL 线，设置中断标志，并将 0x00 装入状态寄存器。该状态代码可用作状态服务程序的向量，尝试再次终止串行传输或从错误状态中恢复，如表 267 所示。

14.10.7 I^2C 状态服务程序

本节将介绍不同 I^2C 状态服务程序都必须执行的操作。其中包括：

- 复位后 I^2C 块的初始化。
- I^2C 中断服务
- 支持 4 种 I^2C 操作模式的 26 种状态服务程序。

14.10.8 初始化

在初始化示例中，I^2C 块可在主机模式和从机模式中使能。每种模式都使用缓冲区进行发送和接收。初始化例程执行以下功能：

- 将部件自身的从属地址和通用调用位 (GC) 载入 ADR
- 置位 I^2C 中断使能位和中断优先级位
- 通过同时设置CON中的I2EN和AA位使能从机模式，通过加载SCLH和SCLL寄存器定义串行时钟频率 （对于主机模式）。主机例程必须在主程序中启动。

这时，I^2C 硬件开始在 I^2C 总线上检查自身的从属地址和通用调用。如果检测到通用调用或自身的从属地址，则请求中断并将相应的状态信息载入 STAT。

14.10.9 I^2C 中断服务

当进入I^2C中断时，STAT含有一个状态代码，可识别要执行的26个状态服务中的其中一个。

14.10.10 状态服务程序

每个状态程序都是 I^2C 中断程序的组成部分，分别用来处理 26 种状态。

14.10.11 配合实际应用的状态服务

状态服务示例演示了响应 26 个 I^2C 状态代码必须执行的典型操作。如果 4 种 I^2C 操作模式中有一种或几种没被用到，则模式的相关的状态服务可被忽略，只要小心处理，那些状态就不会出现。

在应用中，可能需要在I^2C操作过程中执行一些超时处理，来限制无效总线或丢失服务程序。

14.11 软件示例

14.11.1 初始化程序

将 I^2C 接口初始化用作从机和 / 或主机的例子。

1. 将自身的从属地址载入 ADR，如果需要，使能通用调用识别。
2. 使能 I^2C 中断。
3. 将 0x44 写入 CONSET 以设置 I2EN 和 AA 位，使能从机功能。对于仅主机功能，将 0x40 写入 CONSET。

14.11.2 启动主机发送功能

通过建立缓冲区、指针和数据计数开始主机发送操作，然后启动 “起动”。

1. 初始化主机数据计数器。
2. 建立数据将发送到的从属地址，并添加写位。
3. 将 0x20 写入 CONSET 以设置 STA 位。
4. 建立要在主机发送缓冲区中发送的数据。
5. 初始化主机数据计数器以匹配正在发送的消息长度。
6. 退出。

14.11.3 启动主机接收功能

通过建立缓冲区、指针和数据计数开始主机接收操作，然后启动 “起动”。

1. 初始化主机数据计数器。
2. 建立数据将发送到的从属地址，并添加读位。
3. 将 0x20 写入 CONSET 以设置 STA 位。
4. 建立主机接收缓冲区。
5. 初始化主机数据计数器以匹配要接收的消息长度。
6. 退出。

14.11.4 I^2C 中断程序

确定 I^2C 的状态和处理该状态的状态程序。

1. 从 STA 中读出 I^2C 的状态。
2. 使用状态值分支到 26 个可能状态例程之一。

14.11.5 非模式特定状态

14.11.5.1 状态：0x00

总线错误。进入未寻址的从机模式并释放总线。

1. 将 0x14 写入 CONSET 以设置 STO 和 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 退出。

14.11.5.2 主机状态

状态 08 和状态 10 用于主机发送模式和主机接收模式。R/W 位决定下一状态是在主机发送模式中还是主机接收模式中。

14.11.5.3 状态：0x08

“起动”条件已发送。将发送从属地址 + R/W 位和接收 ACK 位。

1. 将从属地址和 R/W 位写入 DAT。
2. 将 0x04 写入 CONSET 以设置 AA 位。
3. 将 0x08 写入 CONCLR 以清除 SI 标志。
4. 建立主机发送模式数据缓冲区。
5. 建立主机接收模式数据缓冲区。
6. 初始化主机数据计数器。
7. 退出。

14.11.5.4 状态：0x10

“重复起始”条件已发送。将发送从属地址 + R/W 位和接收 ACK 位。

1. 将从属地址和 R/W 位写入 DAT。
2. 将 0x04 写入 CONSET 以设置 AA 位。
3. 将 0x08 写入 CONCLR 以清除 SI 标志。
4. 建立主机发送模式数据缓冲区。
5. 建立主机接收模式数据缓冲区。
6. 初始化主机数据计数器。
7. 退出。

14.11.6 主发送状态

14.11.6.1 状态：0x18

之前的状态为状态 8 或状态 10，从属地址 + 写位已发送，并接收了 ACK。将发送第一个数据字节并接收 ACK 位。

1. 将来自主机发送缓冲区的第一个数据字节载入 DAT。
2. 将 0x04 写入 CONSET 以设置 AA 位。
3. 将 0x08 写入 CONCLR 以清除 SI 标志。
4. 主机发送缓冲区指针加递增。
5. 退出。

14.11.6.2 状态：0x20

已发送从属地址 + 写位，已接收 NOT ACK。将发送 “停止”条件。

1. 将 0x14 写入 CONSET 以设置 STO 和 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 退出。

14.11.6.3 状态：0x28

已发送数据，已接收 ACK。如果发送的数据是最后一个数据字节则发送一个“停止”条件，否则发送下一个数据字节。

1. 主机数据计数器递减，如果发送的不是最后一个数据字节则跳至第 5 步。
2. 将 0x14 写入 CONSET 以设置 STO 和 AA 位。
3. 将 0x08 写入 CONCLR 以清除 SI 标志。
4. 退出。
5. 将来自主机发送缓冲区的下一个数据字节载入 DAT。
6. 将 0x04 写入 CONSET 以设置 AA 位。
7. 将 0x08 写入 CONCLR 以清除 SI 标志。
8. 主机发送缓冲区指针加递增。
9. 退出。

14.11.6.4 状态：0x30

已发送数据，已接收 NOT ACK。将发送 “停止”条件。

1. 将 0x14 写入 CONSET 以设置 STO 和 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 退出。

14.11.6.5 状态：0x38

在从属地址 + 写位或数据期间仲裁已丢失。总线已释放且进入未寻址的从机模式。当总线再次空闲时将发送一个新的 “起动”条件。

1. 将 0x24 写入 CONSET 以设置 STA 和 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 退出。

14.11.7 主接收状态

14.11.7.1 状态：0x40

之前的状态为状态 08 或状态 10。从属地址 + 读位已发送，并接收了 ACK。将接收数据并返回 ACK。

1. 将 0x04 写入 CONSET 以设置 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 退出。

14.11.7.2 状态：0x48

已发送从属地址 + 读位，已接收 NOT ACK。将发送 “停止”条件。

1. 将 0x14 写入 CONSET 以设置 STO 和 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 退出。

14.11.7.3 状态：0x50

数据已接收，ACK 已返回。将从 DAT 读取数据。将接收其它数据。如果这是最后一个数据字节，则返回 NOT ACK，否则返回 ACK。

1. 从 DAT 读取数据字节到主机接收缓冲区中。
2. 主机数据计数器递减，如果发送的不是最后一个数据字节则跳至第 5 步。
3. 将 0x0C 写入 CONCLR 以清除 SI 标志和 AA 位。
4. 退出。
5. 将 0x04 写入 CONSET 以设置 AA 位。
6. 将 0x08 写入 CONCLR 以清除 SI 标志。
7. 主机接收缓冲区指针加递增
8. 退出。

14.11.7.4 状态：0x58

数据已接收， NOT ACK 已返回。将从 DAT 读取数据。将发送 “停止”条件。

1. 从 DAT 读取数据字节到主机接收缓冲区中。
2. 将 0x14 写入 CONSET 以设置 STO 和 AA 位。
3. 将 0x08 写入 CONCLR 以清除 SI 标志。
4. 退出。

14.11.8 从接收状态

14.11.8.1 状态：0x60

自身的从属地址 + 写位已接收， ACK 已返回。将接收数据并返回 ACK。

1. 将 0x04 写入 CONSET 以设置 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 建立从机接收模式数据缓冲区。
4. 初始化从机数据计数器。
5. 退出。

14.11.8.2 状态：0x68

作为总线主机时，在从属地址和 R/W 位中仲裁已丢失。自身的从属地址 + 写位已接收，ACK 已返回。将接收数据并返回 ACK。在总线再次空闲后设置 STA 以重启主机模式。

1. 将 0x24 写入 CONSET 以设置 STA 和 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 建立从机接收模式数据缓冲区。
4. 初始化从机数据计数器。
5. 退出。

14.11.8.3 状态：0x70

通用调用已接收， ACK 已返回。将接收数据并返回 ACK。

1. 将 0x04 写入 CONSET 以设置 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 建立从机接收模式数据缓冲区。
4. 初始化从机数据计数器。
5. 退出。

14.11.8.4 状态：0x78

作为总线主机时，在从属地址 +R/W 位中仲裁已丢失。通用调用已接收且 ACK 已返回。将接收数据并返回 ACK。在总线再次空闲后设置 STA 以重启主机模式。

1. 将 0x24 写入 CONSET 以设置 STA 和 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 建立从机接收模式数据缓冲区。
4. 初始化从机数据计数器。
5. 退出。

14.11.8.5 状态：0x80

先前已使用自身的从属地址进行寻址。数据已接收且 ACK 已返回。将读取其它数据。

1. 从 DAT 读取数据字节到从机接收缓冲区中。
2. 从机数据计数器递减，如果发送的不是最后一个数据字节则跳至第 5 步。
3. 将 0x0C 写入 CONCLR 以清除 SI 标志和 AA 位。
4. 退出。
5. 将 0x04 写入 CONSET 以设置 AA 位。
6. 将 0x08 写入 CONCLR 以清除 SI 标志。
7. 从机接收缓冲区指针加递增。
8. 退出。

14.11.8.6 状态：0x88

先前已使用自身的从属地址进行寻址。数据已接收且 NOT ACK 已返回。将不保存接收的数据。进入未寻址的从机模式。

1. 将 0x04 写入 CONSET 以设置 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 退出。

14.11.8.7 状态：0x90

先前已使用通用调用进行寻址。数据已接收， ACK 已返回。将保存接收的数据。将只接收第一个数据字节和 ACK。将接收其它数据和 NOT ACK。

1. 从 DAT 读取数据字节到从机接收缓冲区中。
2. 将 0x0C 写入 CONCLR 以清除 SI 标志和 AA 位。
3. 退出。

14.11.8.8 状态：0x98

先前已使用通用调用进行寻址。数据已接收， NOT ACK 已返回。将不保存接收的数据。进入未寻址的从机模式。

1. 将 0x04 写入 CONSET 以设置 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 退出。

14.11.8.9 状态：0xA0

已接收 “停止”条件或 “重复起始”条件，但仍作为从机寻址。将不保存数据。进入未寻址的从机模式。

1. 将 0x04 写入 CONSET 以设置 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 退出。

14.11.9 从发送状态

14.11.9.1 状态：0xA8

自身的从属地址 + 读位已接收， ACK 已返回。将发送数据，将接收 ACK 位。

1. 将来自从机发送缓冲区的第一个数据字节载入 DAT。
2. 将 0x04 写入 CONSET 以设置 AA 位。
3. 将 0x08 写入 CONCLR 以清除 SI 标志。
4. 建立从机发送模式数据缓冲区。
5. 从机发送缓冲区指针加递增。
6. 退出。

14.11.9.2 状态：0xB0

作为总线主机时，在从属地址和 R/W 位中仲裁丢失。自身的从属地址 + 读位已接收， ACK 已返回。将发送数据，将接收 ACK 位。在总线再次空闲后设置 STA 以重启主机模式。

1. 将来自从机发送缓冲区的第一个数据字节载入 DAT。
2. 将 0x24 写入 CONSET 以设置 STA 和 AA 位。
3. 将 0x08 写入 CONCLR 以清除 SI 标志。
4. 建立从机发送模式数据缓冲区。
5. 从机发送缓冲区指针加递增。
6. 退出。

14.11.9.3 状态：0xB8

已发送数据，已接收 ACK。将发送数据，将接收 ACK 位。

1. 将来自从机发送缓冲区的数据字节载入 DAT。
2. 将 0x04 写入 CONSET 以设置 AA 位。
3. 将 0x08 写入 CONCLR 以清除 SI 标志。
4. 从机发送缓冲区指针加递增。
5. 退出。

14.11.9.4 状态：0xC0

已发送数据，已接收 NOT ACK。进入未寻址的从机模式。

1. 将 0x04 写入 CONSET 以设置 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 退出。

14.11.9.5 状态：0xC8

最后一个数据字节已发送， ACK 已接收。进入未寻址的从机模式。

1. 将 0x04 写入 CONSET 以设置 AA 位。
2. 将 0x08 写入 CONCLR 以清除 SI 标志。
3. 退出。

15.1 本章导读

所有 LPC1315/16/17/45/46/47 器件上均配有 CT16B0/1。

以下引脚仅在 LQFP64 封装上可用：

- CT16B0_CAP1、 CT16B1_CAP1、 CT32B1_CAP1

以下引脚仅在 LQFP64 和 LQFP48 封装上可用：

- CT32B0_CAP1

15.2 基本配置

CT16B0/1 计数器 / 定时器通过以下寄存器进行配置：

- 引脚：CT16B0/1 引脚必须在 IOCON 寄存器模块中配置。
- 电源：在 SYSAHBCLKCTRL 寄存器中，设置表 19 中的位 7 和 8。
- 外围设备时钟由系统时钟确定 （表 18）。

15.3 特性

- 两个带有可编程 16 位预分频器的 16 位计数器 / 定时器。
- 计数器 / 定时器操作
- 两个16位捕获通道，可在输入信号跳变时快速捕获定时器值。捕获事件还可以有选择性地生成中断。
- 可配置定时器和预分频器在指定捕获事件清零。此特性通过在输入脉冲前沿清零定时器并捕获定时器在后沿的值，方便进行脉冲宽度测量。
- 四个 16 位匹配寄存器允许：
  - 连续操作，可选择在匹配时产生中断。
  - 在与可选中断生成相匹配时停止定时器运行。
  - 在与可选中断生成相匹配时进行定时器复位。
- 两个对应于匹配寄存器的外部输出，具备以下功能：
  - 匹配时设置低电平。
  - 匹配时设置高电平。
  - 匹配时切换。
  - 匹配时不执行任何操作。
- 对于各定时器，最多4个匹配寄存器可配置为PWM，允许使用最多2个匹配输出作为单独边沿控制的 PWM 输出。

15.4 应用

- 时间间隔定时器，用于对内部事件进行计数
- 脉冲宽度解调器 （经捕获输入）
- 自由运行的定时器
- 脉冲宽度调制器 （经匹配输出）

15.5 描述

每个计数器 / 定时器都设计用来计算外设时钟 (PCLK) 或外部供电时钟的周期，并且可根据 4 个匹配寄存器的值在指定定时器值处产生中断或执行其他操作。每个计数器 / 定时器还包括 1 个捕获输入，用来在输入信号跳变时捕获定时器值，同时可根据需要产生一个中断。

在 PWM 模式下，可使用其中 2 个匹配寄存器向匹配输出引脚提供单边沿控制的 PWM 输出。建议使用不用于输出的匹配寄存器来控制 PWM 周期长度。

注： 16 位计数器 / 定时器 0 (CT16B0) 和 16 位计数器 / 定时器 1 (CT16B1) 除外设基址不同外，其他功能相似。

15.6 引脚说明

表 268 简要总结了每个计数器 / 定时器相关的引脚。

表 268. 计数器 / 定时器引脚描述

引脚	类型	描述
CT16B0_CAP[1:0] CT16B1_CAP[1:0]	输入	捕获信号： 可以配置捕获引脚上的跳变，用计数器 / 定时器中的值载入捕获寄存器，并且可以有选择性地产生中断。 计数器 / 定时器模块可选择捕获信号作为时钟源来代替 PCLK 衍生时钟。有关更多详情，请参阅第 15.7.11 节。
CT16B0_MAT[1:0] CT16B1_MAT[1:0]	输出	CT16B0/1 的外部匹配输出： 当 CT16B0/1 (MR1:0) 的匹配寄存器与定时器计数器 (TC) 相等时，该输出可以进行切换、转入低电平、转入高电平或不执行任何操作。外部匹配寄存器 (EMR) 和 PWM 控制寄存器 (PWMCON) 控制该输出的功能。

15.7 寄存器描述

16 位计数器 / 定时器 0 包含的寄存器如表 269 所示，16 位计数器 / 定时器 1 包含的寄存器如表 270 所示。详细描述如下。

表 269. 寄存器简介：16 位计数器 / 定时器 0 CT16B0 （基址 0x4000 C000）

名称	访问类型	地址偏移	描述	复位值 [1]	参考
IR	R/W	0x000	中断寄存器。可以对 IR 执行写入操作来清除中断。可以对 IR 执行读取操作，以确定可能使用哪个中断源 （共 8 个）。	0	表 271
TCR	R/W	0x004	定时器控制寄存器。 TCR 用于控制定时器计数器功能。通过 TCR 可以对定时器计数器执行禁用或复位操作。	0	表 272
TC	R/W	0x008	定时器计数器。16 位 TC 每隔 PR+1 个 PCLK 周期递增一次。TC 将通过 TCR 来控制。	0	表 273
PR	R/W	0x00C	预分频寄存器。当预分频计数器（见下）与该值相等时，下个时钟 TC 加 1， PC 清零。	0	表 274
PC	R/W	0x010	预分频计数器。16 位 PC 是一个计数器，它会增加到与 PR 中存放的值相等。当计数到达 PR 中的值时， TC 将递增计数，并且会清除 PC 值。通过总线接口可以观察和控制 PC。	0	表 275
MCR	R/W	0x014	匹配控制寄存器。MCR 用于控制在发生匹配时是否生成中断，以及是否进行 TC 复位。	0	表 276
MR0	R/W	0x018	匹配寄存器 0。通过 MCR 可以使能 MR0，以便在每次 MR0 与 TC 相匹配时执行 TC 复位、停止 TC 和 PC 运行和 / 或生成中断。	0	表 277
MR1	R/W	0x01C	匹配寄存器 1。请参见 MR0 描述。	0	表 277
MR2	R/W	0x020	匹配寄存器 2。请参见 MR0 描述。	0	表 277
MR3	R/W	0x024	匹配寄存器 3。请参见 MR0 描述。	0	表 277
CCR	R/W	0x028	捕获控制寄存器。CCR 用于控制将使用哪些捕获输入边缘来加载捕获寄存器，并且在出现捕获时是否生成中断。	0	表 278
CR0	RO	0x02C	捕获寄存器 0(CR0)。当 CT16B0_CAP0 输入上产生事件时， CR0 载入 TC 值。	0	表 279
CR1	RO	0x030	捕获寄存器 1(CR1)。当 CT16B0_CAP1 输入上产生事件时， CR1 载入 TC 值。	0	表 279
-	-	0x034 - 0x38	保留。	-	
EMR	R/W	0x03C	外部匹配寄存器。 EMR 控制匹配功能及外部匹配引脚 CT16B0_MAT[1:0] 和 CT16B1_MAT[1:0]。	0	表 280
-	-	0x040 - 0x06C	保留。	-	
CTCR	R/W	0x070	计数控制寄存器。 CTCR 用于选择定时器模式和计数器模式。并且在计数器模式中，会选择要进行计数的信号和边缘。	0	表 282
PWMC	R/W	0x074	PWM 控制寄存器 (PWMCON)。PWMCON 使能 PWM 模式，用于外部匹配引脚 CT16B0_MAT[1:0] 和 CT16B1_MAT[1:0]。	0	表 283

[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。

表 270. 寄存器简介：16 位计数器 / 定时器 1 CT16B1 （基址 0x4001 0000）

名称	访问类型	地址	描述	复位值[1]	参考
IR	R/W	0x000	中断寄存器。可以对 IR 执行写入操作来清除中断。可以对 IR 执行读取操作，以确定可能使用哪个中断源 （共 8 个）。	0	表 271
TCR	R/W	0x004	定时器控制寄存器。 TCR 用于控制定时器计数器功能。通过 TCR 可以对定时器计数器执行禁用或复位操作。	0	表 272
TC	R/W	0x008	定时器计数器。 16 位 TC 每隔 PR+1 个 PCLK 周期递增一次。 TC 将通过 TCR 来控制。	0	表 273
PR	R/W	0x00C	预分频寄存器。当预分频计数器（见下）与该值相等时，下个时钟 TC 加 1， PC 清零。	0	表 274
PC	R/W	0x010	预分频计数器。 16 位 PC 是一个计数器，它会增加到与 PR 中存放的值相等。当计数到达 PR 中的值时， TC 将递增计数，并且会清除 PC 值。通过总线接口可以观察和控制 PC。	0	表 275
MCR	R/W	0x014	匹配控制寄存器。MCR 用于控制在发生匹配时是否生成中断，以及是否进行 TC 复位。	0	表 276
MR0	R/W	0x018	匹配寄存器 0。通过 MCR 可以使能 MR0，以便在每次 MR0 与 TC 相匹配时执行 TC 复位、停止 TC 和 PC 运行和 / 或生成中断。	0	表 277
MR1	R/W	0x01C	匹配寄存器 1。请参见 MR0 描述。	0	表 277
MR2	R/W	0x020	匹配寄存器 2。请参见 MR0 描述。	0	表 277
MR3	R/W	0x024	匹配寄存器 3。请参见 MR0 描述。	0	表 277
CCR	R/W	0x028	捕获控制寄存器。CCR 用于控制将使用哪些捕获输入边缘来加载捕获寄存器，并且在出现捕获时是否生成中断。	0	表 278
CR0	RO	0x02C	捕获寄存器 0(CR0)。当 CT16B1_CAP0 输入上产生事件时， CR0 载入 TC 值。	0	表 279
CR1	RO	0x030	捕获寄存器 1(CR1)。当 CT16B1_CAP1 输入上产生事件时， CR1 载入 TC 值。	0	表 279
-	-	0x034 - 0x038	保留。	-	
EMR	R/W	0x03C	外部匹配寄存器。 EMR 控制匹配功能及外部匹配引脚 CT16B0_MAT[1:0] 和 CT16B1_MAT[1:0]。	0	表 280
-	-	0x040 - 0x06C	保留	-	
CTCR	R/W	0x070	计数控制寄存器。 CTCR 用于选择定时器模式和计数器模式。并且在计数器模式中，会选择要进行计数的信号和边缘。	0	表 282
PWMC	R/W	0x074	PWM 控制寄存器 (PWMCON)。 PWMCON 使能 PWM 模式，用于外部匹配引脚 CT16B0_MAT[1:0] 和 CT16B1_MAT[1:0]。	0	表 283

[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。

15.7.1 中断寄存器

中断寄存器包含 4 个用于匹配中断的位及 2 个用于捕获中断的位。如果有中断产生， IR 中的相应位为高电平。否则，该位为低电平。将逻辑 1 写入到相应的 IR 位中，将执行中断复位。写入 0 无效。

表 271. 中断寄存器 （IR，地址 0x4000 C000 (CT16B0) 和 0x4001 0000 (CT16B1)）位描述

位	符号	描述	复位值
0	MR0INT	匹配通道 0 的中断标志。	0
1	MR1INT	匹配通道 1 的中断标志。	0
2	MR2INT	匹配通道 2 的中断标志。	0
3	MR3INT	匹配通道 3 的中断标志。	0
4	CR0INT	捕获通道 0 事件的中断标志。	0
5	CR1INT	捕获通道 1 事件的中断标志。	0
31:6	-	保留	-

15.7.2 定时器控制寄存器

定时器控制寄存器 (TCR) 用于控制计数器 / 定时器的操作。

表 272. 定时器控制寄存器（TCR，地址 0x4000 C004 (CT16B0) 和 0x4001 0004 (CT16B1)）位描述

位	符号	值	描述	复位值
0	CEN		计数器使能。	0
		0	计数器被禁用。	
		1	定时器 / 计数器和预分频计数器使能计数。	
1	CRST		计数器复位。	0
		0	不执行任何操作。	
		1	定时器计数器和预分频计数器在 PCLK 的下一个上升沿同步复位。计数器将保留为复位状态，直至 TCR[1] 归零为止。	
31:2	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

15.7.3 定时器计数器

当预分频器计数器达到其最终计数时， 16 位定时器计数器会递增计数。如果 TC 在到达计数器上限之前没有复位，它将一直计数到 0x0000 FFFF 然后翻转到 0x0000 0000。该事件不会产生中断，如果需要，可使用匹配寄存器检测溢出。

表 273. 定时器计数器寄存器（TC，地址 0x4000 C008 (CT16B0) 和 0x4001 0008 (CT16B1)）位描述

位	符号	描述	复位值
15:0	TC	定时器计数器值。	0
31:16	-	保留。	-

15.7.4 预分频寄存器

16 位预分频寄存器指定了预分频计数器的最大计数值。

表 274. 预分频寄存器 （PR，地址 0x4000 C00C (CT16B0) 和 0x4001 000C (CT16B1)）位描述

位	符号	描述	复位值
15:0	PCVAL	预分频值。	0
31:16	-	保留。	-

15.7.5 预分频计数器寄存器

16 位预分频计数器将在 PCLK 应用于定时器计数器之前，使用某一常数值对 PCLK 执行分频控制。它所控制的是定时器分辨率与最大时间之间的关系，从而能防止定时器溢流。预分频计数器会在每个 PCLK 时钟上递增计数。当预分频计数器的计数达到预分频寄存器中存储的值时，定时器计数器将递增计数，并且在下一个 PCLK 时钟上进行预分频计数器复位。这将使得 TC 当 PR = 0 时在每个 PCLK 上递增计数，当 PR = 1 时，在每 2 个 PCLK 上递增计数，依次类推。

表 275. 预分频计数器寄存器（PC，地址0x4000 C010 (CT16B0)和0x4001 0010 (CT16B1)）位描述

位	符号	描述	复位值
15:0	PC	预分频计数器值。	0
31:16	-	保留。	-

15.7.6 匹配控制寄存器

匹配控制寄存器用于控制在某个匹配寄存器与定时器计数器相匹配时将执行的操作。匹配控制寄存器各位的功能如表 276 所示。

表 276. 匹配控制寄存器 （MCR，地址 0x4000 C014 (CT16B0) 和 0x4001 0014 (CT16B1)）位描述

位	符号	值	描述	复位值
0	MR0I		MR0 上的中断：当 MR0 与 TC 值匹配时产生中断。	0
		1	使能	
		0	已禁用	
1	MR0R		MR0 上的复位：MR0 与 TC 匹配将使 TC 复位。	0
		1	使能	
		0	已禁用	
2	MR0S		MR0 上的停止：MR0 与 TC 匹配时将使 TC 和 PC 停止， TCR[0] 置 0。	0
		1	使能	
		0	已禁用	
3	MR1I		MR1 上的中断：当 MR1 与 TC 值匹配时产生中断。	0
		1	使能	
		0	已禁用	
4	MR1R		MR1 上的复位：MR1 与 TC 匹配将使 TC 复位。	0
		1	使能	
		0	已禁用	
5	MR1S		MR1 上的停止：MR1 与 TC 匹配时将使 TC 和 PC 停止， TCR[0] 置 0。	0
		1	使能	
		0	已禁用	

表 276. 匹配控制寄存器（MCR，地址 0x4000 C014 (CT16B0) 和 0x4001 0014 (CT16B1)）位描述 *（续）*

位	符号	值	描述	复位值
6	MR2I		MR2 上的中断：当 MR2 与 TC 值匹配时产生中断。	0
		1	使能	
		0	已禁用	
7	MR2R		MR2 上的复位：MR2 与 TC 匹配将使 TC 复位。	0
		1	使能	
		0	已禁用	
8	MR2S		MR2 上的停止：MR2 与 TC 匹配时将使 TC 和 PC 停止， TCR[0] 置 0。	0
		1	使能	
		0	已禁用	
9	MR3I		MR3 上的中断：当 MR3 与 TC 值匹配时产生中断。	0
		1	使能	
		0	已禁用	
10	MR3R		MR3 上的复位：MR3 与 TC 匹配将使 TC 复位。	0
		1	使能	
		0	已禁用	
11	MR3S		MR3 上的停止：MR3 与 TC 匹配时将使 TC 和 PC 停止， TCR[0] 置 0。	0
		1	使能	
		0	已禁用	
31:12	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

15.7.7 匹配寄存器

匹配寄存器值将不断与定时器计数器值进行比较。直至两个值相等时，会自动触发各种操作。可能产生的操作有生成中断、定时器计数器复位或停止定时器运行。这些操作将由 MCR 寄存器中的设置来控制。

表 277. 匹配寄存器 （MR0 到 3，地址 0x4000 C018 到 24 (CT16B0) 以及 0x4001 0018 到 24 (CT16B1)）位描述

位	符号	描述	复位值
15:0	MATCH	定时器计数器匹配值。	0
31:16	-	保留。	-

15.7.8 捕获控制寄存器

捕获控制寄存器用于控制当捕获事件发生时，是否将计数器 / 定时器中的值装入捕获寄存器，以及捕获事件是否产生中断。同时设置上升位和下降位是一种有效的配置，这会在两个边沿产生捕获事件。在下面描述中， n 表示定时器编号， 0 或 1。

表 278. 捕获控制寄存器 （CCR，地址 0x4000 C028 (CT16B0) 和 0x4001 0028 (CT16B1)）位描述

位	符号	值	描述	复位值
0	CAP0RE		CT16Bn_CAP0上升沿捕获. CT16Bn_CAP0上的从0至1的序列，将使CR0载入TC内容。	0
		1	使能。	
		0	禁用。	
1	CAP0FE		CT16Bn_CAP0下降沿捕获. CT16Bn_CAP0上的从1至0的序列，将使CR0载入TC内容。	0
		1	使能。	
		0	禁用。	
2	CAP0I		CT16Bn_CAP0 事件中断：CT16Bn_CAP0 事件所导致的 CR0 装载将产生一个中断。	0
		1	使能。	
		0	禁用。	
3	CAP1RE		CT16Bn_CAP1上升沿捕获. CT16Bn_CAP1上的从0至1的序列，将使CR0载入TC内容。	0
		1	使能。	
		0	禁用。	
4	CAP1FE		CT16Bn_CAP1下降沿捕获. CT16Bn_CAP1上的从1至0的序列，将使CR0载入TC内容。	0
		1	使能。	
		0	禁用。	
5	CAP1I		CT16Bn_CAP1 事件中断：CT16Bn_CAP1 事件所导致的 CR0 装载将产生一个中断。	0
		1	使能。	
		0	禁用。	
31:3	-	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

15.7.9 捕获寄存器

各捕获寄存器与器件引脚相关联，当引脚发生特定的事件时，可将计数器 / 定时器的值装入该捕获寄存器。捕获控制寄存器中的设置用于确定是否使能捕获功能，以及是在相关引脚的上升沿，还是下降沿或两者之上发生捕获事件。

表 279. 捕获寄存器 （CR，地址 0x4000 C02C (CR0) 至 0x4000 C030 (CR1)(CT16B0) 和 0x4001 002C (CR0) 至 0x4001 0030 (CR1) (CT16B1)）位描述

位	符号	描述	复位值
15:0	CAP	定时器计数器捕获值。	0
31:16	-	保留。	-

15.7.10 外部匹配寄存器

外部匹配寄存器为外部匹配引脚 CT16Bn_MAT[1:0] 提供控制和状态。

如果匹配输出被配置为 PWM 输出，则外部匹配寄存器的功能由 PWM 规则决定 （第 274 页上的章节 15.7.13 “单边沿控制的 PWM 输出规则”）。

表 280. 外部匹配寄存器 （EMR，地址 0x4000 C03C (CT16B0) 和 0x4001 003C (CT16B1)）位描述

位	符号	值	描述	复位值
0	EM0		外部匹配 0。该位反映输出 CT16B0_MAT0/CT16B1_MAT0 的状态，不管该输出是否连接到此引脚。当 TC 与 MR0 相匹配时，该位可以进行切换、转入低电平、转入高电平或不执行任何操作。位 EMR[5:4] 控制该输出的功能。如果选用了 IOCON 寄存器的匹配功能（0= 低电平， 1= 高电平），该位就会被驱动到 CT16B0_MAT0/CT16B1_MAT0 引脚上。	0
1	EM1		外部匹配 1。该位反映输出 CT16B0_MAT1/CT16B1_MAT1 的状态，不管该输出是否连接到此引脚。当 TC 与 MR1 相匹配时，该位可以进行切换、转入低电平、转入高电平或不执行任何操作。位 EMR[7:6] 控制该输出的功能。如果选用了 IOCON 寄存器的匹配功能（0= 低电平， 1= 高电平），该位就会被驱动到 CT16B0_MAT0/CT16B1_MAT0 引脚上。	0
2	EM2		外部匹配 2。该位反映匹配通道 2 的状态。当 TC 与 MR2 相匹配时，该位可以进行切换、转入低电平、转入高电平或不执行任何操作。位 EMR[9:8] 控制该输出的功能。	0
3	EM3		外部匹配 3。该位反映匹配通道 3 的输出的状态。当 TC 与 MR3 相匹配时，该位可以进行切换、转入低电平、转入高电平或不执行任何操作。位 EMR[11:10] 控制该输出的功能。	0
5:4	EMC0		外部匹配控制 0。决定外部匹配 0 的功能。这些位的编码如表 281 所示。	00
		0x0	不执行任何操作。	
		0x1	将相应的外部匹配位 / 输出清零 （如果引脚处于输出状态，则 CT16Bi_MAT0 引脚为低电平）。	
		0x2	将相应的外部匹配位 / 输出置 1 （如果引脚处于输出状态，则 CT16Bi_MAT0 引脚为高电平）。	
		0x3	切换相应的外部匹配位 / 输出。	
7:6	EMC1		外部匹配控制 1。用于确定外部匹配 1 的功能。	00
		0x0	不执行任何操作。	
		0x1	将相应的外部匹配位 / 输出清零 （如果引脚处于输出状态，则 CT16Bi_MAT1 引脚为低电平）。	
		0x2	将相应的外部匹配位 / 输出置 1 （如果引脚处于输出状态，则 CT16Bi_MAT1 引脚为高电平）。	
		0x3	切换相应的外部匹配位 / 输出。	
9:8	EMC2		外部匹配控制 2。用于确定外部匹配 2 的功能。	00
		0x0	不执行任何操作。	
		0x1	将相应的外部匹配位 / 输出清零 （如果引脚处于输出状态，则 CT16Bi_MAT2 引脚为低电平）。	
		0x2	将相应的外部匹配位 / 输出置 1 （如果引脚处于输出状态，则 CT16Bi_MAT2 引脚为高电平）。	
		0x3	切换相应的外部匹配位 / 输出。	

表 280. 外部匹配寄存器（EMR，地址 0x4000 C03C (CT16B0) 和 0x4001 003C (CT16B1)）位描述 *（续）*

位	符号	值	描述	复位值
11:10	EMC3		外部匹配控制 3。决定外部匹配 3 的功能。这些位的编码如表 281 所示。	00
		0x0	不执行任何操作。	
		0x1	将相应的外部匹配位 / 输出清零（如果引脚处于输出状态，则 CT16Bi_MAT3 引脚为低电平）。	
		0x2	将相应的外部匹配位 / 输出置 1（如果引脚处于输出状态，则 CT16Bi_MAT3 引脚为高电平）。	
		0x3	切换相应的外部匹配位 / 输出。	
31:12	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	-

表 281. 外部匹配控制

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	函数
00	不执行任何操作。
01	将相应的外部匹配位 / 输出清零（如果引脚处于输出状态，则 CT16Bn_MATm 引脚为低电平）。
10	将相应的外部匹配位 / 输出置 1（如果引脚处于输出状态，则 CT16Bn_MATm 引脚为高电平）。
11	切换相应的外部匹配位 / 输出。

15.7.11 计数控制寄存器

计数控制寄存器 (CTCR) 用于选择定时器模式或者计数器模式，并在计数器模式中选择要计数的引脚和边沿。

如果将计数器模式选为操作模式，则在 PCLK 时钟的每个上升沿上会对（CTCR 位 3:2 所选的）CAP 输入进行采样。在比较该 CAP 输入的两个连续样本后，将会确认下列四个事件之一：所选 CAP 输入电平处于上升沿、下降沿，或上升下降沿或无变化。仅当所标识的事件与 CTCR 寄存器中的位 1:0 所选的内容相匹配时，定时器计数器寄存器才会递增计数。

计数器的外部提供时钟的处理能力具有一定的局限性，因此不能执行有效的处理。由于 PCLK 时钟的两个连续上升沿仅用于标识 CAP 所选输入的一个边缘，因此 CAP 输入的频率不能超过 PCLK 时钟的一半。因此在这种情况下，相同的 CAP 输入上的高电平 / 低电平的持续时间不能少于 1/PCLK。

该寄存器的位 7:4 还用于使能和配置捕获 - 清除 - 定时器特性。该特性允许特定 CAP 输入的指定边沿将定时器全部清零。使用该机制在输入脉冲前沿清除定时器然后在后沿执行捕获，可以使用单捕获输入来直接测量脉冲宽度，而无需在软件中执行减法操作。

表 282. 计数控制寄存器（CTCR，地址 0x4000 C070 (CT16B0) 和 0x4001 0070 (CT16B1)）位描述

位	符号	值	描述	复位值
1:0	CTM		计数器 / 定时器模式。该字段用于选择可使用哪个 PCLK 上升沿，使定时器预分频计数器 （PC）递增计数，或清除 PC 并使定时器计数器 （TC）递增计数。 **注：**如果在 CTCR 中选择计数器模式，必须将捕获控制寄存器 (CCR) 中的位 2:0 设置为 000。	0
		0x0	定时器模式：每个 PCLK 的上升沿	
		0x1	计数器模式：将在位 3:2 所选的 CAP 输入的上升沿上执行 TC 递增计数。	
		0x2	计数器模式：将在位 3:2 所选的 CAP 输入的下降沿上执行 TC 递增计数。	
		0x3	计数器模式：将在位 3:2 所选的 CAP 输入的上升沿和下降沿上执行 TC 递增计数。	
3:2	CIS		计数输入选择。在计数器模式下 （当该寄存器中位 1:0 不为 00 时），这些位将选择为时钟提供哪个 CAP 引脚或比较器输出。值 0x1 至 0x3 保留。	0
		0x0	CT16Bn_CAP0	
		0x1	CT16Bn_CAP1	
4	ENCC		将此位置 1 可在发生位 7:5 指定的捕获 - 边沿事件时清零定时器和预分频器。	0
7:5	SELCC		当位 4 为 1 时，这两位选择哪个捕获输入边沿将导致定时器和预分频器被清零。当位 4 为低电平时，这两位无效。值 0x4 至 0x7 保留。	0
		0x0	CAP0 的上升沿清零定时器 （如果位 4 被置位）	
		0x1	CAP0 的下降沿清零定时器 （如果位 4 被置位）	
		0x2	CAP1 的上升沿清零定时器 （如果位 4 被置位）	
		0x3	CAP1 的下降沿清零定时器 （如果位 4 被置位）	
31:8	-	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	-

15.7.12 PWM 控制寄存器

PWM 控制寄存器用于将匹配输出配置为 PWM 输出。每个匹配输出均可分别设置，以决定匹配输出是作为 PWM 输出还是作为功能受外部匹配寄存器 (EMR) 控制的匹配输出。

对于定时器，CT16Bn_MAT[1:0] 输出最多可选择 3 个单边沿控制的 PWM 输出。一个附加的匹配寄存器决定 PWM 的周期长度。当任何其他匹配寄存器出现匹配时，PWM 输出置为高电平。用于设置 PWM 周期长度的匹配寄存器负责将定时器复位。当定时器复位到 0 时，所有当前配置为 PWM 输出的高电平匹配输出清零。

表 283. PWM 控制寄存器 （PWMC，地址 0x4000 C074 和 0x4001 0074 (CT16B1)）位描述

位	符号	值	描述	复位值
0	PWMEN0		通道 0 的 PWM 模式使能。	0
		0	CT16Bi_MAT0 受 EM0 控制。	
		1	CT16Bi_MAT0 的 PWM 模式使能。	
1	PWMEN1		通道 1 的 PWM 模式使能。	0
		0	CT16Bi_MAT01 受 EM1 控制。	
		1	CT16Bi_MAT1 的 PWM 模式使能。	
2	PWMEN2		通道 2 的 PWM 模式使能。	0
		0	CT16Bi_MAT2 受 EM2 控制。	
		1	CT16Bi_MAT2 的 PWM 模式使能。	
3	PWMEN3		通道 3 的 PWM 模式使能。	0
		0	CT16Bi_MAT3 受 EM3 控制。	
		1	CT16Bi_MAT3 的 PWM 模式使能。	
31:4	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	-

15.7.13 单边沿控制的 PWM 输出规则

1. 所有单边沿控制的 PWM 输出在 PWM 周期开始时都为低电平 （定时器置为 0），除非它们的匹配值等于 0。
2. 每个 PWM 输出在达到其匹配值时将转入高电平。如果没有发生匹配 （即匹配值大于 PWM 周期长度），则 PWM 输出将继续保持低电平。
3. 如果将大于 PWM 周期长度的匹配值写入到匹配寄存器，且 PWM 信号已经为高电平，则在下一个 PWM 周期开始时 PWM 信号将被清零。
4. 如果匹配寄存器中包含与定时器复位值 （PWM 周期长度）相同的值，则在下一个时钟节拍时 PWM 输出将复位到低电平。因此，PWM 输出总是包含一个时钟节拍宽度的正脉冲，周期由 PWM 周期长度决定 （即定时器重载入值）。
5. 如果匹配寄存器置 0，则 PWM 输出将在定时器第一次返回 0 时变为高电平，并继续保持高电平。

注：当选择匹配输出作为 PWM 输出来执行时，除匹配寄存器设置 PWM 周期长度外，匹配控制寄存器 MCR 中的定时器复位 (MRnR) 和定时器停止 (MRnS) 位必须置为 0。对于该寄存器，当定时器值与相应的匹配寄存器值匹配时，将 MRnR 位置 1 以使能定时器复位。

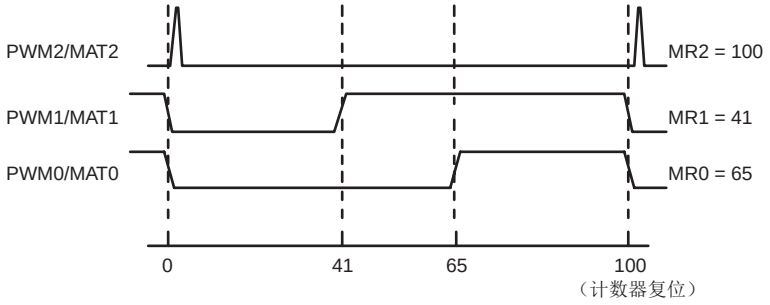


图 47. 采样 PWM 波形，其中 PWM 周期长度为 100（MR3 选择），MAT3:0 被 PWCON 寄存器使能为 PWM 输出。

15.8 定时器操作示例

如图 48 所示，定时器配置为在匹配时复位计数并产生中断。将预分频器设置为 2，匹配寄存器设置为 6。在发生匹配的定时器周期结束时，会进行定时器计数复位。这就为匹配值提供了一个完整周期。当定时器计数到达匹配值后，将在下一个时钟内生成中断以指示出现了匹配。

如图 49 所示，定时器配置为在匹配时停止计数并产生中断。将预分频器再次设置为 2，匹配寄存器设置为 6。当定时器计数到达匹配值后，将在下一个时钟内清除 TCR 中的定时器使能位，并且会生成中断以指示出现了匹配。

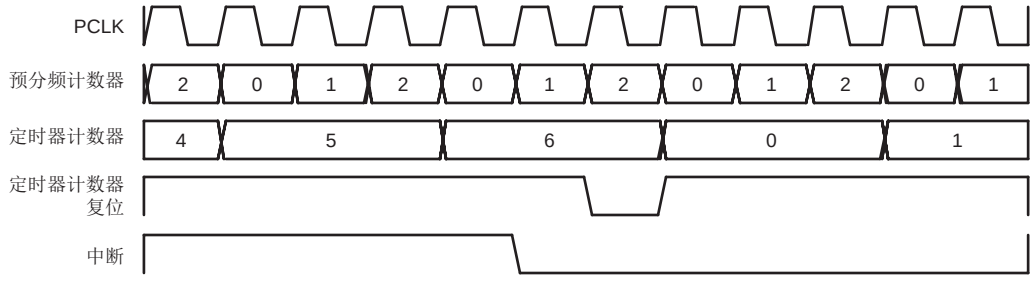


图 48. 定时器周期，其中 PR = 2、 MRx = 6，并且使能了在匹配时同时执行中断和复位操作

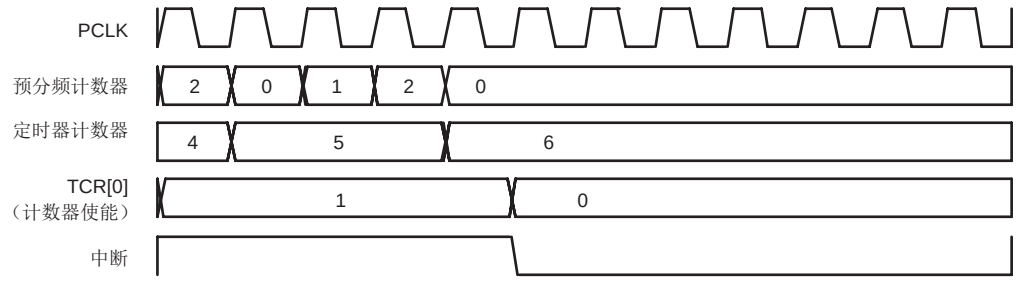


图 49. 定时器周期，其中 PR = 2、 MRx = 6，并且使能了在匹配时同时执行中断和停止操作

15.9 架构

计数器 / 定时器 0 和计数器 / 定时器 1 的功能框图如图 50 所示。

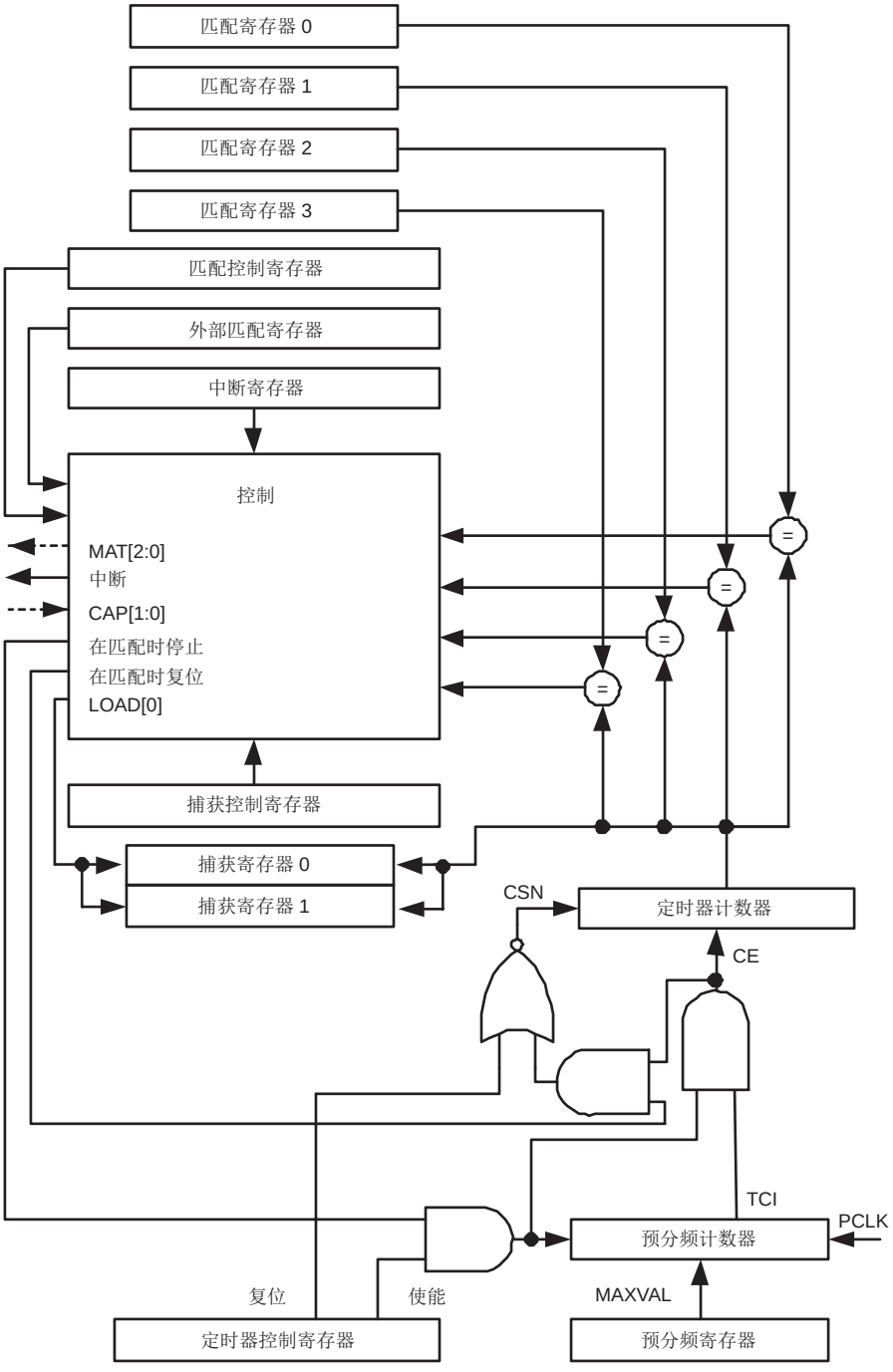


图 50. 16 位计数器 / 定时器功能框图

16.1 本章导读

所有 LPC1315/16/17/45/46/47 器件上均配有 CT32B0/1。

以下引脚仅在 LQFP64 封装上可用：

- CT32B1_CAP1

以下引脚仅在 LQFP64 和 LQFP48 封装上可用：

- CT32B0_CAP1

16.2 基本配置

CT32B0/1 计数器 / 定时器通过以下寄存器进行配置：

- 引脚：CT32B0/1 引脚必须在 IOCON 寄存器模块中配置。
- 电源：在 SYSAHBCLKCTRL 寄存器中，设置表 19 中的位 9 和 10。
- 外围设备时钟由系统时钟确定 （参见表 18）。

16.3 特性

- 两个带有可编程 32 位预分频器的 32 位计数器 / 定时器。
- 计数器或定时器操作。
- 四个32位捕获通道，可在输入信号跳变时快速捕获定时器值。捕获事件还可以有选择性地生成中断。
- 可配置定时器和预分频器在指定捕获事件清零。此特性通过在输入脉冲前沿清零定时器并捕获定时器在后沿的值，方便进行脉冲宽度测量。
- 四个 32 位匹配寄存器允许：
  - 连续操作，可选择在匹配时产生中断。
  - 在与可选中断生成相匹配时停止定时器运行。
  - 在与可选中断生成相匹配时进行定时器复位。
- 四个对应于匹配寄存器的外部输出，具备以下功能：
  - 匹配时设置低电平。
  - 匹配时设置高电平。
  - 匹配时切换。
  - 匹配时不执行任何操作。
- 对于各定时器，最多4个匹配寄存器可配置为PWM，允许使用最多3个匹配输出作为单独边沿控制的 PWM 输出。

16.4 应用

- 时间间隔定时器，用于对内部事件进行计数
- 脉冲宽度解调器 （经捕获输入）
- 自由运行的定时器
- 脉冲宽度调制器 （经匹配输出）

16.5 简介

每个计数器 / 定时器都设计用来计算外设时钟 (PCLK) 或外部供电时钟的周期，并且可根据 4 个匹配寄存器的值在指定定时器值处产生中断或执行其他操作。每个计数器 / 定时器还包括 1 个捕获输入，用来在输入信号跳变时捕获定时器值，同时可根据需要产生一个中断。

在 PWM 模式下，可使用其中 3 个匹配寄存器向匹配输出引脚提供单边沿控制的 PWM 输出。而剩下的那个匹配寄存器则用于控制 PWM 周期长度。

16.6 引脚说明

表 284 简要总结了每个计数器 / 定时器相关的引脚。

表 284. 计数器 / 定时器引脚描述

引脚	类型	描述
CT32B0_CAP[1:0] CT32B1_CAP[1:0]	输入	捕获信号： 可以配置捕获引脚上的跳变，用定时器计数器中的值载入其中一个捕获寄存器，并且可以有选择性地产生一个中断。 计数器 / 定时器模块可选择捕获信号作为时钟源来代替 PCLK 衍生时钟。有关更多详情，请参阅第 286 页上的章节 16.7.11 “计数控制寄存器”。
CT32B0_MAT[3:0] CT32B1_MAT[3:0]	输出	CT32B0/1 的外部匹配输出： 当匹配寄存器 MR3:0 的值与定时器计数器值 (TC) 相等时，该输出可以进行切换、转入低电平、转入高电平或不执行任何操作。外部匹配寄存器 (EMR) 和 PWM 控制寄存器 (PWMCON) 控制该输出的功能。

16.7 寄存器描述

32 位计数器 / 定时器 0 包含的寄存器如表 285 所示，32 位计数器 / 定时器 1 包含的寄存器如表 286 所示。详细描述如下。

表 285. 寄存器简介：32 位计数器 / 定时器 0 CT32B0 （基址 0x4001 4000）

名称	访问类型	地址偏移	描述	复位值[1]	参考
IR	R/W	0x000	中断寄存器。可以对 IR 执行写入操作来清除中断。可以对 IR 执行读取操作，以确定可能使用哪个中断源 （共 8 个）。	0	表 287
TCR	R/W	0x004	定时器控制寄存器。TCR 用于控制定时器计数器功能。通过 TCR 可以对定时器计数器执行禁用或复位操作。	0	表 288
TC	R/W	0x008	定时器计数器。32 位 TC 每隔 PR+1 个 PCLK 周期递增一次。TC 将通过 TCR 来控制。	0	表 289
PR	R/W	0x00C	预分频寄存器。当预分频计数器 （见下）与该值相等时，下个时钟 TC 加 1， PC 清零。	0	表 290
PC	R/W	0x010	预分频计数器。32 位 PC 是一个计数器，它会增加到与 PR 中存放的值相等。当计数到达 PR 中的值时，TC 将递增计数，并且会清除 PC 值。通过总线接口可以观察和控制 PC。	0	表 291
MCR	R/W	0x014	匹配控制寄存器。 MCR 用于控制在发生匹配时是否生成中断，以及是否进行 TC 复位。	0	表 292
MR0	R/W	0x018	匹配寄存器 0。通过 MCR 可以使能 MR0，以便在每次 MR0 与 TC 相匹配时执行 TC 复位、停止 TC 和 PC 运行和 / 或生成中断。	0	表 293
MR1	R/W	0x01C	匹配寄存器 1。请参见 MR0 描述。	0	表 293
MR2	R/W	0x020	匹配寄存器 2。请参见 MR0 描述。	0	表 293
MR3	R/W	0x024	匹配寄存器 3。请参见 MR0 描述。	0	表 293
CCR	R/W	0x028	捕获控制寄存器。 CCR 用于控制将使用哪些捕获输入边缘来加载捕获寄存器，并且在出现捕获时是否生成中断。	0	表 294
CR0	RO	0x02C	捕获寄存器 0(CR0)。当 CT32B0_CAP0 输入上产生事件时，CR0 载入 TC 值。	0	表 295
CR1	RO	0x02C	捕获寄存器 1(CR1)。当 CT32B0_CAP1 输入上产生事件时，CR1 载入 TC 值。	0	表 295
-	-	0x034 - 0x038	保留。	-	-
EMR	R/W	0x03C	外部匹配寄存器。 EMR 控制匹配功能及外部匹配引脚 CT32Bn_MAT[3:0]。	0	表 296
-	-	0x040 - 0x06C	保留。	-	
CTCR	R/W	0x070	计数控制寄存器。CTCR 用于选择定时器模式和计数器模式。并且在计数器模式中，会选择要进行计数的信号和边缘。	0	表 298
PWMC	R/W	0x074	PWM 控制寄存器 (PWMCON)。PWMCON 使能 PWM 模式，用于外部匹配引脚 CT32Bn_MAT[3:0]。	0	表 299

[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。

表 286. 寄存器简介：32 位计数器 / 定时器 1 CT32B1 （基址 0x4001 8000）

名称	访问类型	地址偏移	描述	复位值[1]	参考
IR	R/W	0x000	中断寄存器。可以对 IR 执行写入操作来清除中断。可以对 IR 执行读取操作，以确定可能使用哪个中断源 （共 8 个）。	0	表 287
TCR	R/W	0x004	定时器控制寄存器。TCR 用于控制定时器计数器功能。通过 TCR 可以对定时器计数器执行禁用或复位操作。	0	表 288
TC	R/W	0x008	定时器计数器。32 位 TC 每隔 PR+1 个 PCLK 周期递增一次。TC 将通过 TCR 来控制。	0	表 289
PR	R/W	0x00C	预分频寄存器。当预分频计数器 （见下）与该值相等时，下个时钟 TC 加 1， PC 清零。	0	表 290
PC	R/W	0x010	预分频计数器。32 位 PC 是一个计数器，它会增加到与 PR 中存放的值相等。当计数到达 PR 中的值时，TC 将递增计数，并且会清除 PC 值。通过总线接口可以观察和控制 PC。	0	表 291
MCR	R/W	0x014	匹配控制寄存器。 MCR 用于控制在发生匹配时是否生成中断，以及是否进行 TC 复位。	0	表 292
MR0	R/W	0x018	匹配寄存器 0。通过 MCR 可以使能 MR0，以便在每次 MR0 与 TC 相匹配时执行 TC 复位、停止 TC 和 PC 运行和 / 或生成中断。	0	表 293
MR1	R/W	0x01C	匹配寄存器 1。请参见 MR0 描述。	0	表 293
MR2	R/W	0x020	匹配寄存器 2。请参见 MR0 描述。	0	表 293
MR3	R/W	0x024	匹配寄存器 3。请参见 MR0 描述。	0	表 293
CCR	R/W	0x028	捕获控制寄存器。 CCR 用于控制将使用哪些捕获输入边缘来加载捕获寄存器，并且在出现捕获时是否生成中断。	0	表 294
CR0	RO	0x02C	捕获寄存器 0(CR0)。当 CT32B1_CAP0 输入上产生事件时，CR0 载入 TC 值。	0	表 295
CR1	RO	0x030	捕获寄存器 1(CR1)。当 CT32B1_CAP1 输入上产生事件时，CR1 载入 TC 值。	0	表 295
-	-	0x034 - 0x038	保留。	-	-
EMR	R/W	0x03C	外部匹配寄存器。 EMR 控制匹配功能及外部匹配引脚 CT32Bn_MAT[3:0]。	0	表 296
-	-	0x040 - 0x06C	保留。	-	
CTCR	R/W	0x070	计数控制寄存器。CTCR 用于选择定时器模式和计数器模式。并且在计数器模式中，会选择要进行计数的信号和边缘。	0	表 298
PWMC	R/W	0x074	PWM 控制寄存器 (PWMCON)。PWMCON 使能 PWM 模式，用于外部匹配引脚 CT32Bn_MAT[3:0]。	0	表 299

[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。

16.7.1 中断寄存器

中断寄存器由匹配中断的四个位以及捕获中断的四个位组成。如果有中断产生， IR 中的相应位为高电平。否则，该位为低电平。将逻辑 1 写入到相应的 IR 位中，将执行中断复位。写入 0 无效。

表 287. 中断寄存器 （IR，地址 0x4001 4000 (CT32B0) ；和 IR，地址 0x4001 8000）位描述

位	符号	描述	复位值
0	MR0INT	匹配通道 0 的中断标志。	0
1	MR1INT	匹配通道 1 的中断标志。	0
2	MR2INT	匹配通道 2 的中断标志。	0
3	MR3INT	匹配通道 3 的中断标志。	0
4	CR0INT	捕获通道 0 事件的中断标志。	0
5	CR1INT	捕获通道 1 事件的中断标志。	0
31:6	-	保留	-

16.7.2 定时器控制寄存器

定时器控制寄存器 (TCR) 用于控制计数器 / 定时器的操作。

表 288. 定时器控制寄存器（TCR，地址 0x4001 4004 (CT32B0) 和 0x4001 8004 (CT32B1)）位描述

位	符号	值	描述	复位值
0	CEN		计数器使能。	0
		0	计数器被禁用。	
		1	定时器 / 计数器和预分频计数器使能计数。	
1	CRST		计数器复位。	0
		0	不执行任何操作。	
		1	定时器计数器和预分频计数器在 PCLK 的下一个上升沿同步复位。计数器将保留为复位状态，直至 TCR[1] 归零为止。	
31:2	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

16.7.3 定时器计数器寄存器

当预分频器计数器达到其最终计数时， 32 位定时器计数器会递增计数。如果 TC 在到达计数器上限之前没有复位，它将一直计数到 0xFFFF FFFF 然后翻转到 0x0000 0000。该事件不会产生中断，如果需要，可使用匹配寄存器检测溢出。

表 289. 定时器计数器寄存器（TC，地址0x4001 4008 (CT32B0)和0x4001 8008 (CT32B1)）位描述

位	符号	描述	复位值
31:0	TC	定时器计数器值。	0

16.7.4 预分频寄存器

32 位预分频寄存器指定了预分频计数器的最大计数值。

表 290. 预分频寄存器 （PR，地址 0x4001 400C (CT32B0) 和 0x4001 800C (CT32B1)）位描述

位	符号	描述	复位值
31:0	PCVAL	预分频器值。	0

16.7.5 预分频计数器寄存器

32 位预分频计数器将在 PCLK 应用于定时器计数器之前，使用某一常数值对 PCLK 执行分频控制。它所控制的是定时器分辨率与最大时间之间的关系，从而能防止定时器溢流。预分频计数器会在每个 PCLK 时钟上递增计数。当预分频计数器的计数达到预分频寄存器中存储的值时，定时器计数器将递增计数，并且在下一个 PCLK 时钟上进行预分频计数器复位。这将使得 TC 当 PR = 0 时在每个 PCLK 上递增计数，当 PR = 1 时，在每 2 个 PCLK 上递增计数，依次类推。

表 291. 预分频寄存器 （PC，地址 0x4001 4010 (CT32B0) 和 0x4001 8010 (CT32B1)）位描述

位	符号	描述	复位值
31:0	PC	预分频计数器值。	0

16.7.6 匹配控制寄存器

匹配控制寄存器用于控制在某个匹配寄存器与定时器计数器相匹配时将执行的操作。匹配控制寄存器各位的功能如表 292 所示。

表 292. 匹配控制寄存器 （MCR，地址 0x4001 4014 (CT32B0) 和 0x4001 8014 (CT32B1)）位描述

位	符号	值	描述	复位值
0	MR0I		MR0 上的中断：当 MR0 与 TC 值匹配时产生中断。	0
		1	使能	
		0	已禁用	
1	MR0R		MR0 上的复位：MR0 与 TC 匹配将使 TC 复位。	0
		1	使能	
		0	已禁用	
2	MR0S		MR0 上的停止：MR0 与 TC 匹配时将使 TC 和 PC 停止， TCR[0] 置 0。	0
		1	使能	
		0	已禁用	
3	MR1I		MR1 上的中断：当 MR1 与 TC 值匹配时产生中断。	0
		1	使能	
		0	已禁用	
4	MR1R		MR1 上的复位：MR1 与 TC 匹配将使 TC 复位。	0
		1	使能	
		0	已禁用	
5	MR1S		MR1 上的停止：MR1 与 TC 匹配时将使 TC 和 PC 停止， TCR[0] 置 0。	0
		1	使能	
		0	已禁用	

表 292. 匹配控制寄存器（MCR，地址 0x4001 4014 (CT32B0) 和 0x4001 8014 (CT32B1)）位描述 *（续）*

位	符号	值	描述	复位值
6	MR2I		MR2 上的中断：当 MR2 与 TC 值匹配时产生中断。	0
		1	使能	
		0	已禁用	
7	MR2R		MR2 上的复位：MR2 与 TC 匹配将使 TC 复位。	0
		1	使能	
		0	已禁用	
8	MR2S		MR2 上的停止：MR2 与 TC 匹配时将使 TC 和 PC 停止， TCR[0] 置 0。	0
		1	使能	
		0	已禁用	
9	MR3I		MR3 上的中断：当 MR3 与 TC 值匹配时产生中断。	0
		1	使能	
		0	已禁用	
10	MR3R		MR3 上的复位：MR3 与 TC 匹配将使 TC 复位。	0
		1	使能	
		0	已禁用	
11	MR3S		MR3 上的停止：MR3 与 TC 匹配时将使 TC 和 PC 停止， TCR[0] 置 0。	0
		1	使能	
		0	已禁用	
31:12	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

16.7.7 匹配寄存器

匹配寄存器值将不断与定时器计数器值进行比较。直至两个值相等时，会自动触发各种操作。可能产生的操作有生成中断、定时器计数器复位或停止定时器运行。这些操作将由 MCR 寄存器中的设置来控制。

表 293. 匹配寄存器 （MR0 到 3，地址 0x4001 4018 到 24 (CT32B0) 以及 0x4001 8018 到 24 (CT32B1)）位描述

位	符号	描述	复位值
31:0	MATCH	定时器计数器匹配值。	0

16.7.8 捕获控制寄存器

捕获控制寄存器用于控制在发生捕获事件时，四个捕获寄存器之一是否会载入定时器计数器中的值，并且控制是否通过捕获事件来生成中断。同时设置上升位和下降位是一种有效的配置，这会在两个边沿产生捕获事件。在下面描述中，“n”表示定时器编号， 0 或 1。

表 294. 捕获控制寄存器 （CCR，地址 0x4001 4028 (CT32B0) 和 0x4001 8028 (CT32B1)）位描述

位	符号	值	描述	复位值
0	CAP0RE		CT32Bn_CAP0 上升沿捕获：CT32Bn_CAP0 上 “0” 到 “1” 的跳变将使 TC 的内容装入 CR0。	0
		1	使能。	
		0	禁用。	
1	CAP0FE		CT32Bn_CAP0 下降沿捕获：CT32Bn_CAP0 上 “1” 到 “0” 的跳变将使 TC 的内容装入 CR0。	0
		1	使能。	
		0	禁用。	
2	CAP0I		CT32Bn_CAP0 事件中断：CT32Bn_CAP0 事件所导致的 CR0 装载将产生一个中断。	0
		1	使能。	
		0	禁用。	
3	CAP1RE		CT32Bn_CAP1 上升沿捕获：CT32Bn_CAP1 上 “0” 到 “1” 的跳变将使 TC 的内容装入 CR1。	0
		1	使能。	
		0	禁用。	
4	CAP1FE		CT32Bn_CAP1 下降沿捕获：CT32Bn_CAP1 上 “1” 到 “0” 的跳变将使 TC 的内容装入 CR1。	0
		1	使能。	
		0	禁用。	
5	CAP1I		CT32Bn_CAP1 事件中断：CT32Bn_CAP1 事件所导致的 CR1 装载将产生一个中断。	0
		1	使能。	
		0	禁用。	
316	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

16.7.9 捕获寄存器

每个捕获寄存器都与设备引脚相关，并且在该引脚上发生指定的事件时会载入定时器计数器值。捕获控制寄存器中的设置用于确定是否使能捕获功能，以及是在相关引脚的上升沿，还是下降沿或两者之上发生捕获事件。

表 295. 捕获寄存器 （CR，地址 0x4001 402C (CR0) 至 0x4001 4030 (CR1) (CT32B0) 和 0x4001 802C (CR0) 至 0x4001 4030 (CR1)(CT32B1)）位描述

位	符号	描述	复位值
31:0	CAP	定时器计数器捕获值。	0

16.7.10 外部匹配寄存器

外部匹配寄存器为外部匹配引脚 CAP32Bn_MAT[3:0] 提供控制和状态。

如果匹配输出被配置为 PWM 输出，则外部匹配寄存器的功能由 PWM 规则决定（第 289 页上的章节 16.7.13 “单边沿控制的 PWM 输出规则”）。

表 296. 外部匹配寄存器（EMR，地址 0x4001 403C (CT32B0) 和 0x4001 803C (CT32B1)）位描述

位	符号	值	描述	复位值
0	EM0		外部匹配 0。该位反映输出 CT32Bn_MAT0 的状态，不管该输出是否连接到此引脚。当 TC 与 MR0 相匹配时，该位可以进行切换、转入低电平、转入高电平或不执行任何操作。位 EMR[5:4] 控制该输出的功能。如果选用了 IOCON 寄存器的匹配功能（0= 低电平，1= 高电平），该位就会被驱动到 CT32B0_MAT0/CT32B1_MAT0 引脚上。	0
1	EM1		外部匹配 1。该位反映输出 CT32Bn_MAT1 的状态，不管该输出是否连接到此引脚。当 TC 与 MR1 相匹配时，该位可以进行切换、转入低电平、转入高电平或不执行任何操作。位 EMR[7:6] 控制该输出的功能。如果选用了 IOCON 寄存器的匹配功能（0= 低电平，1= 高电平），该位就会被驱动到 CT32B0_MAT1/CT32B1_MAT1 引脚上。	0
2	EM2		外部匹配 2。该位反映输出 CT32Bn_MAT2 的状态，不管该输出是否连接到此引脚。当 TC 与 MR2 相匹配时，该位可以进行切换、转入低电平、转入高电平或不执行任何操作。位 EMR[9:8] 控制该输出的功能。如果选用了 IOCON 寄存器的匹配功能（0= 低电平，1= 高电平），该位就会被驱动到 CT32B0_MAT2/CT32B1_MAT2 引脚上。	0
3	EM3		外部匹配 3。该位反映输出 CT32Bn_MAT3 的状态，不管该输出是否连接到此引脚。当 TC 与 MR3 相匹配时，该位可以进行切换、转入低电平、转入高电平或不执行任何操作。位 EMR[11:10] 控制该输出的功能。如果选用了 IOCON 寄存器的匹配功能（0= 低电平，1= 高电平），该位就会被驱动到 CT32B3_MAT0/CT32B1_MAT3 引脚上。	0
5:4	EMC0		外部匹配控制 0。用于确定外部匹配 0 的功能。	00
		0x0	不执行任何操作。	
		0x1	将相应的外部匹配位 / 输出清零（如果引脚处于输出状态，则 CT32Bi_MAT0 引脚为低电平）。	
		0x2	将相应的外部匹配位 / 输出置 1（如果引脚处于输出状态，则 CT32Bi_MAT0 引脚为高电平）。	
		0x3	切换相应的外部匹配位 / 输出。	
7:6	EMC1		外部匹配控制 1。用于确定外部匹配 1 的功能。	00
		0x0	不执行任何操作。	
		0x1	将相应的外部匹配位 / 输出清零（如果引脚处于输出状态，则 CT32Bi_MAT1 引脚为低电平）。	
		0x2	将相应的外部匹配位 / 输出置 1（如果引脚处于输出状态，则 CT32Bi_MAT1 引脚为高电平）。	
		0x3	切换相应的外部匹配位 / 输出。	
9:8	EMC2		外部匹配控制 2。用于确定外部匹配 2 的功能。	00
		0x0	不执行任何操作。	
		0x1	将相应的外部匹配位 / 输出清零（如果引脚处于输出状态，则 CT32Bi_MAT2 引脚为低电平）。	
		0x2	将相应的外部匹配位 / 输出置 1（如果引脚处于输出状态，则 CT32Bi_MAT2 引脚为高电平）。	
		0x3	切换相应的外部匹配位 / 输出。	

表 296. 外部匹配寄存器（EMR，地址 0x4001 403C (CT32B0) 和 0x4001 803C (CT32B1)）位描述 *（续）*

位	符号	值	描述	复位值
11:10	EMC3		外部匹配控制 3。用于确定外部匹配 3 的功能。	00
		0x0	不执行任何操作。	
		0x1	将相应的外部匹配位 / 输出清零（如果引脚处于输出状态，则 CT32Bi_MAT3 引脚为低电平）。	
		0x2	将相应的外部匹配位 / 输出置 1（如果引脚处于输出状态，则 CT32Bi_MAT3 引脚为高电平）。	
		0x3	切换相应的外部匹配位 / 输出。	
31:12	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

表 297. 外部匹配控制

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	函数
00	不执行任何操作。
01	将相应的外部匹配位 / 输出清零（如果引脚处于输出状态，则 CT32Bn_MATm 引脚为低电平）。
10	将相应的外部匹配位 / 输出置 1（如果引脚处于输出状态，则 CT32Bn_MATm 引脚为高电平）。
11	切换相应的外部匹配位 / 输出。

16.7.11 计数控制寄存器

计数控制寄存器 (CTCR) 用于选择定时器模式或者计数器模式，并在计数器模式中选择要计数的引脚和边沿。

如果将计数器模式选为操作模式，则在 PCLK 时钟的每个上升沿上会对（CTCR 位 3:2 所选的）CAP 输入进行采样。在比较该 CAP 输入的两个连续样本后，将会确认下列四个事件之一：所选 CAP 输入电平处于上升沿、下降沿，或上升下降沿或无变化。仅当所标识的事件与 CTCR 寄存器中的位 1:0 所选的内容相匹配时，定时器计数器寄存器才会递增计数。

计数器的外部提供时钟的处理能力具有一定的局限性，因此不能执行有效的处理。由于 PCLK 时钟的两个连续上升沿仅用于标识 CAP 所选输入的一个边缘，因此 CAP 输入的频率不能超过 PCLK 时钟的一半。因此在这种情况下，相同的 CAP 输入上的高电平 / 低电平的持续时间不能少于 1/PCLK。

该寄存器的位 7:4 还用于使能和配置捕获 - 清除 - 定时器特性。该特性允许特定 CAP 输入的指定边沿将定时器全部清零。使用该机制在输入脉冲前沿清除定时器然后在后沿执行捕获，可以使用单捕获输入来直接测量脉冲宽度，而无需在软件中执行减法操作。

表 298. 计数控制寄存器（CTCR，地址 0x4001 4070 (CT32B0) 和 0x4001 8070 (CT32B1)）位描述

位	符号	值	描述	复位值
1:0	CTM		计数器 / 定时器模式。该字段用于选择可使用哪个 PCLK 上升沿，使定时器预分频计数器 （PC）递增计数，或清除 PC 并使定时器计数器 （TC）递增计数。 **注：** 如果在 CTCR 中选择计数器模式，必须将捕获控制寄存器 (CCR) 中的位 2:0 设置为 000。	00
		0x0	定时器模式：每个 PCLK 的上升沿	
		0x1	计数器模式：将在位 3:2 所选的 CAP 输入的上升沿上执行 TC 递增计数。	
		0x2	计数器模式：将在位 3:2 所选的 CAP 输入的下降沿上执行 TC 递增计数。	
		0x3	计数器模式：将在位 3:2 所选的 CAP 输入的上升沿和下降沿上执行 TC 递增计数。	
3:2	CIS		计数输入选择。在计数器模式下 （当该寄存器中位 1:0 不为 00 时），这些位将选择为时钟提供哪个 CAP 引脚或比较器输出。 **注：** 如果在 CTCR 中选择计数器模式，必须将捕获控制寄存器 (CCR) 中与该输入对应的 3 位设置为 000。值 0x2 至 0x3 保留。	00
		0x0	CT32Bn_CAP0	
		0x1	CT32Bn_CAP1	
4	ENCC		将此位置 1 可在发生位 7:5 指定的捕获 - 边沿事件时清零定时器和预分频器。	0
7:5	SEICC		当位 4 为 1 时，这两位选择哪个捕获输入边沿将导致定时器和预分频器被清零。当位 4 为低电平时，这两位无效。值 0x3 至 0x7 保留。	
		0x0	CAP0 的上升沿清零定时器 （如果位 4 被置位）	
		0x1	CAP0 的下降沿清零定时器 （如果位 4 被置位）	
		0x2	CAP1 的上升沿清零定时器 （如果位 4 被置位）	
		0x3	CAP1 的下降沿清零定时器 （如果位 4 被置位）	
31:8	-	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	-

16.7.12 PWM 控制寄存器

PWM 控制寄存器用于将匹配输出配置为 PWM 输出。每个匹配输出均可分别设置，以决定匹配输出是作为 PWM 输出还是作为功能受外部匹配寄存器 (EMR) 控制的匹配输出。

对于定时器， MATn.2:0 输出最多可选择 3 个单边沿控制的 PWM 输出。一个附加的匹配寄存器决定 PWM 的周期长度。当任何其他匹配寄存器出现匹配时， PWM 输出置为高电平。用于设置 PWM 周期长度的匹配寄存器负责将定时器复位。当定时器复位到 0 时，所有当前配置为 PWM 输出的高电平匹配输出清零。

表 299. PWM 控制寄存器 （PWMC， 0x4001 4074 (CT32B0) 和 0x4001 8074 (CT32B1)）位描述

位	符号	值	描述	复位值
0	PWMEN0		通道 0 的 PWM 模式使能。	0
		0	CT32Bi_MAT0 受 EM0 控制。	
		1	CT32Bi_MAT0 的 PWM 模式使能。	
1	PWMEN1		通道 1 的 PWM 模式使能。	0
		0	CT32Bi_MAT01 受 EM1 控制。	
		1	CT32Bi_MAT1 的 PWM 模式使能。	
2	PWMEN2		通道 2 的 PWM 模式使能。	0
		0	CT32Bi_MAT2 受 EM2 控制。	
		1	CT32Bi_MAT2 的 PWM 模式使能。	
3	PWMEN3		通道 3 的 PWM 模式使能。**注：**建议使用匹配通道 3 设置 PWM 周期。	0
		0	CT32Bi_MAT3 受 EM3 控制。	
		1	CT132Bi_MAT3 的 PWM 模式使能。	
31:4	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

16.7.13 单边沿控制的 PWM 输出规则

1. 所有单边沿控制的 PWM 输出在 PWM 周期开始时都为低电平 （定时器置为 0），除非它们的匹配值等于 0。
2. 每个 PWM 输出在达到其匹配值时将转入高电平。如果没有发生匹配 （即匹配值大于 PWM 周期长度），则 PWM 输出将继续保持低电平。
3. 如果将大于 PWM 周期长度的匹配值写入到匹配寄存器，且 PWM 信号已经为高电平，则在下一个 PWM 周期开始时 PWM 信号将被清零。
4. 如果匹配寄存器中包含与定时器复位值 （PWM 周期长度）相同的值，则在定时器达到匹配值后的下一个时钟节拍时 PWM 输出将复位到低电平。因此， PWM 输出总是包含一个时钟节拍宽度的正脉冲，周期由 PWM 周期长度决定 （即定时器重载入值）。
5. 如果匹配寄存器置 0，则 PWM 输出将在定时器第一次返回 0 时变为高电平，并继续保持高电平。

注：当选择匹配输出作为 PWM 输出来执行时，除匹配寄存器设置 PWM 周期长度外，匹配控制寄存器 MCR 中的定时器复位 (MRnR) 和定时器停止 (MRnS) 位必须置为 0。对于该寄存器，当定时器值与相应的匹配寄存器值匹配时，将 MRnR 位置 1 以使能定时器复位。

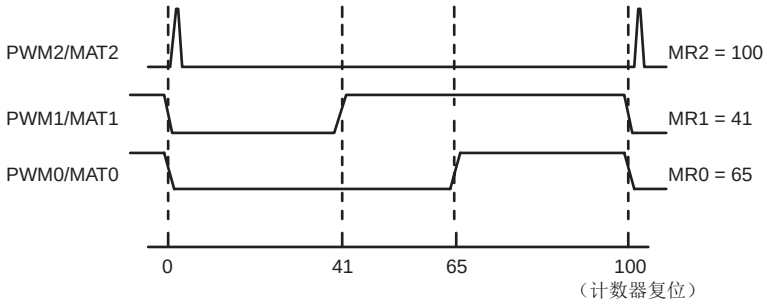


图 51. 采样 PWM 波形，其中 PWM 周期长度为 100（MR3 选择），MAT3:0 被 PWCON 寄存器使能为 PWM 输出。

16.8 定时器操作示例

如图 52 所示，定时器配置为在匹配时复位计数并产生中断。将预分频器设置为 2，匹配寄存器设置为 6。在发生匹配的定时器周期结束时，会进行定时器计数复位。这就为匹配值提供了一个完整周期。当定时器计数到达匹配值后，将在下一个时钟内生成中断以指示出现了匹配。

如图 53 所示，定时器配置为在匹配时停止计数并产生中断。将预分频器再次设置为 2，匹配寄存器设置为 6。当定时器计数到达匹配值后，将在下一个时钟内清除 TCR 中的定时器使能位，并且会生成中断以指示出现了匹配。

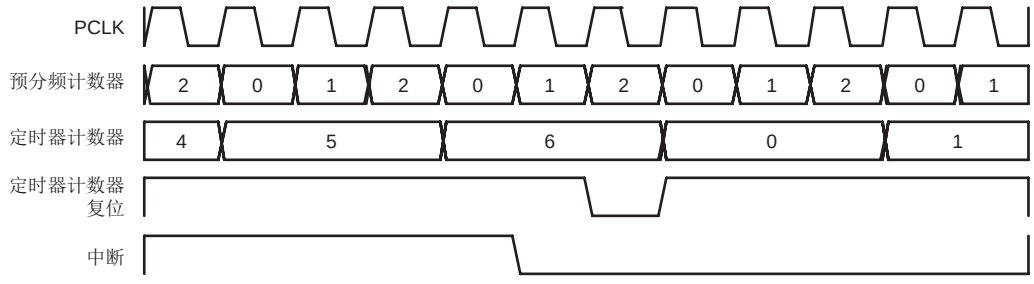


图 52. 定时器周期，其中 PR = 2、 MRx = 6，并且使能了在匹配时同时执行中断和复位操作

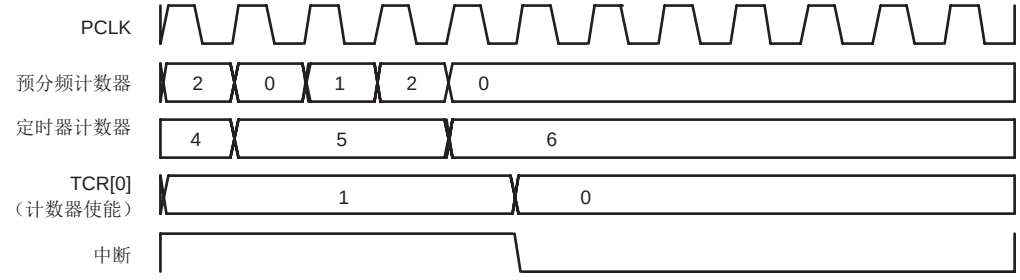


图 53. 定时器周期，其中 PR = 2、 MRx = 6，并且使能了在匹配时同时执行中断和停止操作

16.9 架构

32 位计数器 / 定时器 0 和 32 位计数器 / 定时器 1 的功能框图如图 54 所示。

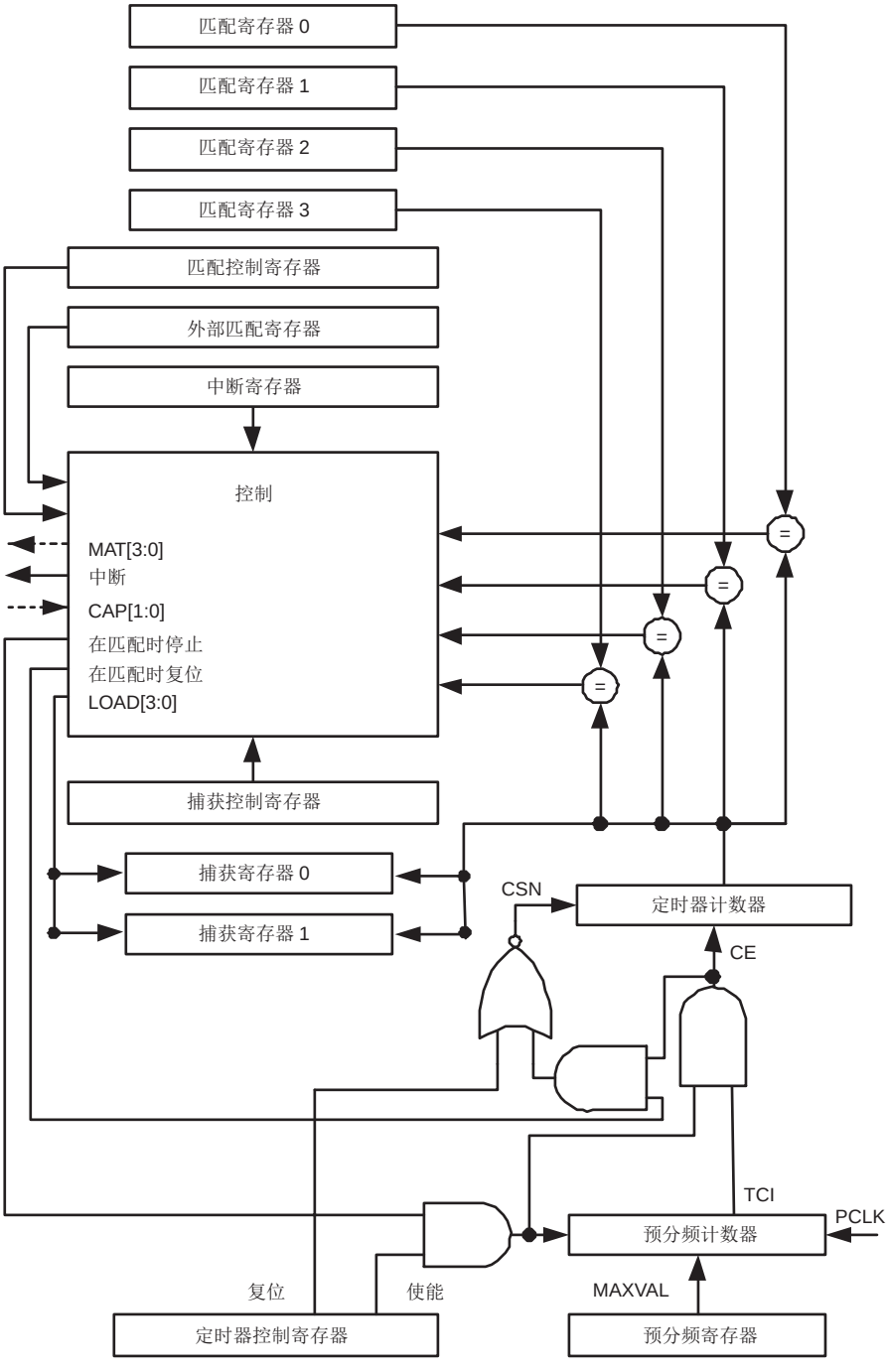


图 54. 32 位计数器 / 定时器功能框图

17.1 本章导读

所有 LPC1315/16/17/45/46/47 器件中的 WWDT 都是一样的。

17.2 基本配置

WWDT 是通过以下寄存器进行配置的：

- 寄存器接口电源（WWDT PCLK 时钟）：在 SYSAHBCLKCTRL 寄存器中，设置表 19 中的位 15。
- 使能 PDRUNCFG 寄存器 （表 42）中的 WWDT 时钟源 （看门狗振荡器或 IRC）。
- 需要时，在 STARTERP1 寄存器中使能用于唤醒的看门狗中断 （表 39）。

17.3 特性

- 如果不能在可编程超时期限内重新载入，则在内部复位芯片。
- 可选的窗口化操作，要求在最大和最小超时期限之间重新载入，两个时限都可编程。
- 可在看门狗超时之前的可编程时间生成可选的警告中断。
- 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 ($T_{WDCLK} \times 2^{24} \times 4$)，递增量为 4 个看门狗时钟。
- “安全”看门狗操作。如果使能，要求禁用硬件复位或看门狗复位。
- 不当的输入序列会导致立即发生看门狗事件 （如已使能）。
- 可选择保护看门狗重新载入值，使其只能在 “警告中断”时间后才能改变。
- 指示看门狗复位的标志。
- 看门狗时钟 (WDCLK) 源可选作内部高频振荡器 (IRC) 或看门狗振荡器。
- 将看门狗振荡器用作时钟源时，可以将看门狗定时器配置为在深度睡眠或掉电模式下运行。
- 具有调试模式。

17.4 应用

看门狗定时器的目的是，如果进入错误状态，在可编程时间内复位或中断微控制器。使能后，如果用户程序不能在预定时间内输入（重新载入）看门狗，则将复位和 / 或生成看门狗。

编程看门狗窗口时，早期的看门狗输入也被视为看门狗事件。这可防止有故障的系统仍然输入看门狗的情况。例如，应用程序代码可能在包含看门狗输入的中断服务中阻塞。将窗口设置为使这种情况导致早期输入，这将生成看门狗事件，允许系统恢复。

17.5 描述

看门狗包括一个固定的 （4 分频）预分频器和一个 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。看门狗应以下列方式使用：

- 在 TC 寄存器中设置看门狗定时器固定的重新载入值。
- 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。
- 向 FEED 寄存器中写入 0xAA，然后写入 0x55，使能看门狗。
- 在看门狗计数器达到零之前必须重新输入看门狗，以防止看门狗事件。如果窗口值是编程的，则还必须在看门狗计数器超过该值后才输入。

如果 “看门狗定时器” 配置为看门狗事件导致复位，则当计数器达到零时， CPU 将复位，从向量表中载入协议栈指针和程序计数器，如同外部复位一样。可检查看门狗超时标志 (WDTOF)，以确定看门狗是否导致了复位条件。 WDTOF 标志必须通过软件清除。

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

17.5.1 功能框图

看门狗的功能框图如下面的图 55 所示。功能框图中未显示同步逻辑 (PCLK - WDCLK)。

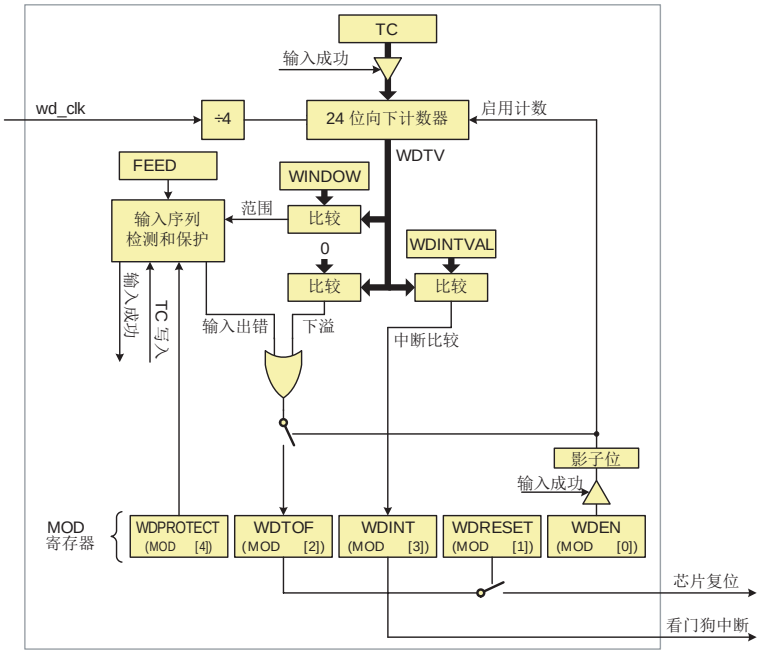


图 55. 看门狗功能框图

17.6 时钟和电源控制

看门狗定时器块使用两个时钟：PCLK 和 WDCLK。 PCLK 由系统时钟生成 （见图 3），供 APB 访问看门狗寄存器使用。 WDCLK 由图 3 中的 wdt_clk 生成，供看门狗定时器计数使用。在工作模式或睡眠模式中， IRC 或看门狗振荡器可用作 wdt_clk，但在深度睡眠模式或掉电模式中，只能使用看门狗振荡器。

注：如果设置 MOD 寄存器 （表 301）中的 LOCK 位，并选择 IRC 作为 WWDT 时钟源，则在深度睡眠模式和掉电模式中， IRC 强制开启，导致功耗增加。

这两个时钟域之间有一些同步逻辑。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，以便 CPU 将其作为 WDTV 寄存器读取。

17.7 使用 WWDT 锁定功能

WWDT 支持几种锁定功能，可使能这些功能以确保 WWDT 始终运行：

- 意外覆盖 WWDT 时钟源
- 更改 WWDT 时钟源
- 更改 WWDT 重新载入值

17.7.1 意外覆盖 WWDT 时钟

如果设置了 WWDT CLKSEL 寄存器（表 306）的位 31，将忽略对 CLKSEL 寄存器的位 0（时钟源选择位）的写入，并且时钟源不会更改。

时钟源覆盖锁定机制可通过简单地清除 CLKSEL 寄存器的位 31 来禁用。

17.7.2 更改 WWDT 时钟源

如果设置了 WWDT MOD 寄存器的位 5，在 CLKSEL 寄存器中选择的当前时钟源会被锁定，并且在进入睡眠模式、深度睡眠模式或掉电模式时，不会由软件或硬件更改。因此，用户必须确保在设置 MOD 寄存器的位 5 **之前**，选择合适的 WWDT 时钟源用于每个功率模式：

- 工作模式或睡眠模式：允许 IRC 或看门狗振荡器。
- 深度睡眠模式：允许 IRC 和看门狗振荡器。但是，在深度睡眠模式中使用 IRC 时会增加功耗。要最大限度降低功耗，应使用看门狗振荡器作为时钟源。
- 掉电模式：只有看门狗振荡器可以用作 WWDT 的时钟源。因此，在设置位 5 和锁定时钟源之前，必须将 WWDT 时钟源设为看门狗振荡器。否则，器件可能无法进入掉电模式。
- 深度掉电模式：由于 WWDT 和所有时钟都停止运行，时钟锁定机制均无效。但是，可以设置 PMU 中的附加锁定位，以阻止器件进入深度掉电模式（参见表 48）。

时钟源锁定机制只能通过任何类型的复位禁用。

17.7.3 更改 WWDT 重新载入值

如果设置了 WWDT MOD 寄存器的位 4，看门狗超时值 (TC) 仅在计数器低于 WDWARNINT 和 WDWINDOW 的值时才能改变。

重新载入覆盖锁定机制只能通过任何类型的复位禁用。

17.8 寄存器描述

看门狗定时器包含表 300 所示的寄存器。

表 300. 寄存器简介：看门狗定时器 （基址 0x4000 4000）

名称	访问类型	地址偏移	描述	复位值 [1]	参考
MOD	R/W	0x000	看门狗模式寄存器，此寄存器包含 “看门狗定时器” 的基本模式和状态。	0	表 301
TC	R/W	0x004	看门狗定时器常数寄存器，此 24 位寄存器确定超时值。	0xFF	表 303
FEED	WO	0x008	看门狗输入序列寄存器，向此寄存器写入 0xAA，然后写入 0x55，将使用 WDTC 中包含的值重新载入看门狗定时器。	不适用	表 304
TV	RO	0x00C	看门狗定时器值寄存器，此 24 位寄存器读取看门狗定时器的当前值。	0xFF	表 305
CLKSEL	R/W	0x010	看门狗时钟选择寄存器。	0	表 305
WARNINT	R/W	0x014	看门狗警告中断比较值。	0	表 307
WINDOW	R/W	0x018	看门狗窗口比较值。	0xFF FFFF	表 308

[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。

17.8.1 看门狗模式寄存器

WDMOD 寄存器控制着看门狗的操作。注意，看门狗输入必须在 WDMOD 寄存器的任何改变生效之前执行。

表 301. 看门狗模式寄存器 （MOD - 0x4000 4000）位描述

位	符号	值	描述	复位值
0	WDEN		看门狗使能位，此位一旦写入 1，便不能重新写入 0。	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。此位一旦写入 1，便不能重新写入 0。	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。在看门狗定时器由于输入错误或与 WDPROTECT 关联的事件而超时时设置。由软件清除。如果 WDRESET=1，将引起芯片复位。	0（仅在外部复位之后）
3	WDINT		警告中断标志。在定时器达到 WDWARNINT 寄存器中的值时设置。由软件清除。	0

表 301. 看门狗模式寄存器 （MOD - 0x4000 4000）位描述 *（续）*

位	符号	值	描述	复位值
4	WDPROTECT		看门狗更新模式。此位可由软件设置一次，并且只能通过复位清除。	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		此位中的 1 将阻止禁用或掉电 WDCLKSRC 寄存器位 0 选择的时钟源，还将阻止切换到已禁用或已掉电的时钟源。此位可由软件设置一次，并且只能通过任何复位清除。 **注：**如果在进入深度睡眠或掉电模式时，此位为 1 且 WWDT 时钟源为 IRC，则 IRC 将保持运行，从而增加了深度睡眠模式中的功耗并可能阻止器件正确进入掉电模式 （参见第 17.7 节）。	0
31:6	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

WDEN、 **WDPROTECT** 或 **WDRESET** 位设置后，不能用软件清除。两个标志都须利用外部复位或看门狗定时器复位来清除。

WDTOF 在看门狗超时、发生输入错误，或当 PROTECT =1 且试图写入 TC 寄存器时，设置该看门狗超时标志。此标志通过软件向此位写入 0 来清除。

WDINT 当看门狗计数器达到 WARNINT 规定的值时，设置该看门狗中断标志。此标志在发生复位时清除，并通过软件向此位写入 0 来清除。

在除深度掉电模式外的所有功率模式中，看门狗运行并具有操作时钟源时可发生看门狗复位或中断。可选择看门狗振荡器或 IRC，保持在睡眠模式和深度睡眠模式下运行。在掉电模式中，只能使用看门狗振荡器。如果看门狗中断发生在睡眠、深度睡眠或掉电模式中，且在 NVIC 中使能 WWDT 中断，则会唤醒设备。请注意，在深度睡眠和掉电模式中，除在 NVIC 中以外，还必须在 STARTERP1 寄存器中使能 WWDT 中断。

表 302. 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：将生成看门狗警告中断，但不会生成看门狗复位。 选择此模式后，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：使能看门狗中断和看门狗复位。 选择此模式后，在看门狗计数器达到 WDWARNINT 规定的值时将设置 WDINT 标志，生成看门狗中断请求，并且看门狗计数器到达 0 时将复位微控制器。达到 WDWINDOW 的值之前的看门狗输入也将导致看门狗复位。

17.8.2 看门狗定时器常量寄存器

此 TC 寄存器确定超时值。每次发生输入序列时，TC 的值就会载入看门狗定时器。TC 复位为 0x00 00FF。写入值低于 0xFF 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 303. 看门狗定时器常量寄存器 （TC - 0x4000 4004）位描述

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

17.8.3 看门狗输入寄存器

向该寄存器依序写入 0xAA 和 0x55 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，则此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 后必须完成有效的输入序列，才能使看门狗能够生成复位。此时，看门狗将忽略输入错误。

向 WDFEED 写入 0xAA 之后，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将会立即产生复位 / 中断，并设置 WDTOF 标志。在输入序列期间，复位将在错误访问看门狗寄存器后的第二个 PCLK 期间生成。

如果是这样的应用：某个 / 任何中断可能导致重新计划处理器控制 （脱离输入过程中的当前任务），然后在控制返回到中断任务之前引发对 WDT 的某种其他访问，则最好禁用输入序列周围的中断。

表 304. 看门狗输入寄存器 (FEED - 0x4000 4008) 位描述

位	符号	描述	复位值
7:0	FEED	输入值应是 0xAA，后跟 0x55。	不适用
31:8	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

17.8.4 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。

在读取 24 位计数器的值时，锁定和同步步骤会占用 6 个 WDCLK 周期加 6 个 PCLK 周期，因此， WDTV 的值比 CPU 读取它时定时器的实际值早。

表 305. 看门狗定时器值寄存器 (TV - 0x4000 400C) 位描述

位	符号	描述	复位值
23:0	COUNT	计数器定时器值。	0x00 00FF
31:24	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

17.8.5 看门狗时钟选择寄存器

表 306. 看门狗时钟选择寄存器 (CLKSEL - 0x4000 4010) 位描述

位	符号	值	描述	复位值
0	CLKSEL		选择 WDT 时钟源	0
		0	IRC	
		1	看门狗振荡器 (WDOSC)	
30:1	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
31	LOCK		如果将此位设为 1，写入此寄存器不会影响位 0。只能通过先清除该位，再向位 0 写入新值来更改时钟源。	0

17.8.6 看门狗定时器警告中断寄存器

WDWARNINT 寄存器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 WDWARNINT 规定的值匹配时，会在后续的 WDCLK 后生成中断。

如果计数器后 10 位与 WARNINT 的 10 位值相同，而计数器前面其余位的值都是 0，则看门狗定时器计数器与 WDWARNINT 匹配，这使得看门狗事件之前发生中断的最大时间是 1,023 看门狗定时器计数 （4,096 个看门狗时钟）。如果 WARNINT 为 0，则将在看门狗事件的同时发生中断。

表 307. 看门狗定时器警告中断寄存器 （WARNINT - 0x4000 4014）位描述

位	符号	描述	复位值
9:0	WARNINT	看门狗警告中断比较值。	0
31:10	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

17.8.7 看门狗定时器窗口寄存器

WDWINDOW 寄存器确定在执行看门狗输入时允许的最大 WDTV 值。如果输入序列发生在 WDTV 大于 WDWINDOW 中的值时，则将发生看门狗事件。

WDWINDOW 复位到可能的最大 WDTV 值，因此窗口操作无效。

表 308. 看门狗定时器窗口寄存器 (WINDOW - 0x4000 4018) 位描述

位	符号	描述	复位值
23:0	WINDOW	看门狗窗口值。	0xFF FFFF
31:24	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

17.9 看门狗定时示例

下图描绘了看门狗定时器工作的几个方面。

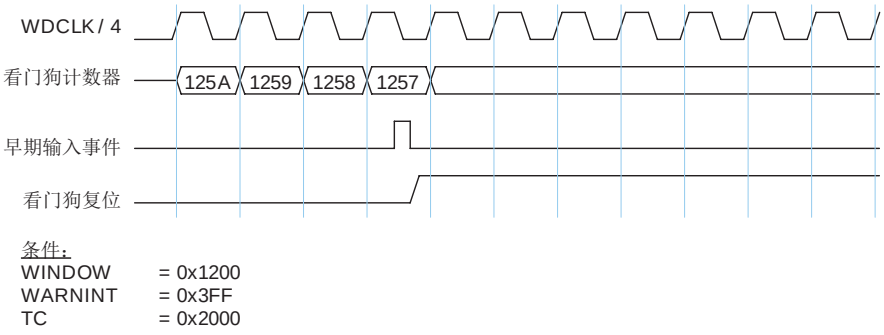


图 56. 使能窗口模式时提前看门狗输入

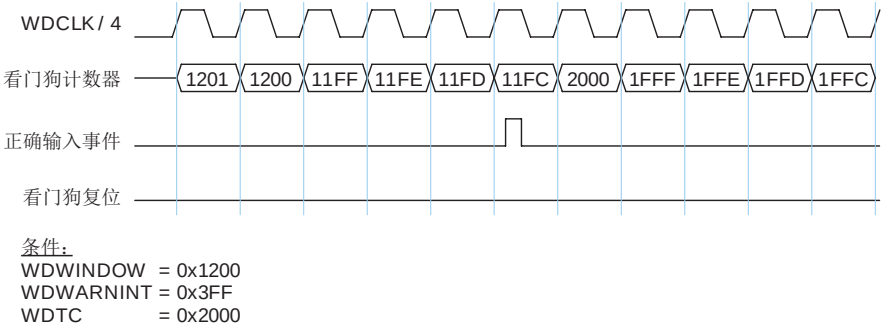


图 57. 使能窗口模式时正确的看门狗输入

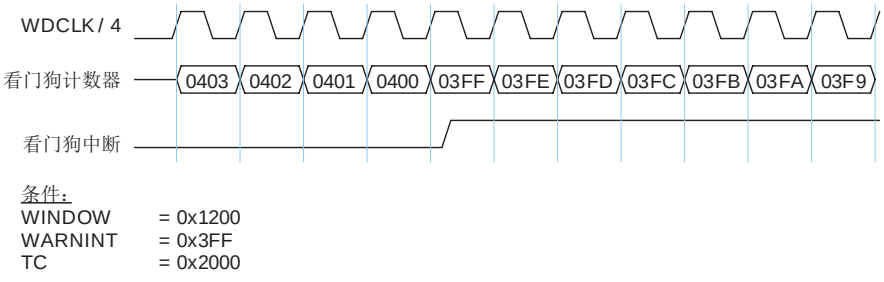


图 58. 看门狗警告中断

18.1 本章导读

系统节拍定时器（SysTick 定时器）是 ARM Cortex-M3 内核的一部分，而且对于所有 LPC1315/16/17/45/46/47 器件都是相同的。

18.2 基本配置

系统节拍定时器是使用以下寄存器进行配置的：

1. 引脚：系统节拍定时器不使用外部引脚。
2. 电源：系统节拍定时器通过 SysTick 控制寄存器（表 364）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。
3. 在 SYST_CSR 寄存器（表 364）中使能 SysTick 定时器的时钟源。

18.3 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 SYSTICKCLK 从内部定时。

18.4 简介

SysTick 定时器的功能框图如下面的图 59 所示。

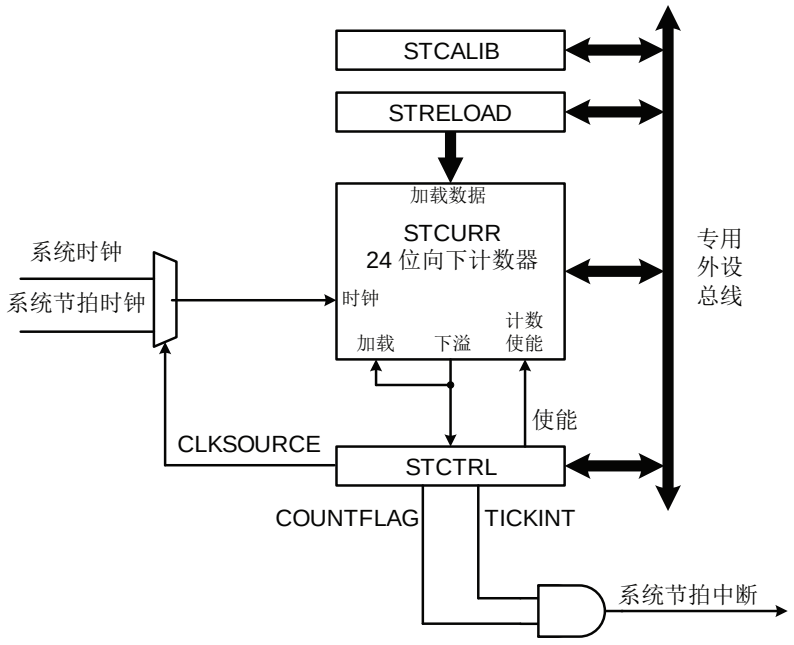


图 59. 系统节拍定时器功能框图

SysTick 定时器是 Cortex-M3 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-M3 的一部分，它为基于 Cortex-M3 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。软件可使用它测量时间 （如：完成任务所需时间、已使用的时间）。
- 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，作为动态时钟管理锁相环的一部分。

详情请参考 《Cortex-M3 用户指南》。

18.5 寄存器描述

systick定时器寄存器位于ARM Cortex-M3专用外设总线上（参见图2），是ARM Cortex-M3内核外设的一部分。有关详情，请参阅第 21.5.4 节。

表 309. 寄存器简介：SysTick 定时器 （基址 0xE000 E000）

名称	访问类型	地址偏移	描述	复位值 [1]	参考
SYST_CSR	R/W	0x010	系统定时器控制和状态寄存器	0x000 0000	表 310
SYST_RVR	R/W	0x014	系统定时器重载值寄存器	0	表 311
SYST_CVR	R/W	0x018	系统定时器当前值寄存器	0	表 312
SYST_CALIB	R/W	0x01C	系统定时器校准值寄存器	0x4	表 313

[1] 复位值仅反映用到的位中保存的数据，不包括保留位的内容。

18.5.1 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-M3 内核系统定时器寄存器模块的一部分。有关该寄存器的位描述，请参阅第 21.5.4 节。

该寄存器决定系统节拍定时器的时钟源。

表 310. SysTick 定时器控制和状态寄存器 (SYST_CSR - 0xE000 E010) 位描述

位	符号	描述	复位值
0	ENABLE	系统节拍计数器使能。为 1 时，计数器使能。为 0 时，计数器禁用。	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) 时钟。为 0 时，选择来自系统节拍时钟分频器 (SYSTICKDIV) 的输出时钟作为基准时钟。这种情况下，内核时钟必须至少比基准时钟快 2.5 倍，否则，计数值将不可预知。	0
15:3	-	保留，用户软件不应向保留位写入 1。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取以来 SysTick 定时器数到 0，则返回 1。	0
31:17	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

18.5.2 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 寄存器进行读取，并将其用作 SYST_RVR 寄存器的值。

表 311. 系统定时器重载值寄存器 (SYST_RVR - 0xE000 E014) 位描述

位	符号	描述	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，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

18.5.3 系统定时器当前值寄存器

当软件读取 SYST_CVR 寄存器时，它将从系统节拍计数器返回当前计数。

表 312. 系统定时器当前值寄存器 (SYST_CVR - 0xE000 E018) 位描述

位	符号	描述	复位值
23:0	CURRENT	读该寄存器会返回系统节拍计数器的当前值。写任意值都可将系统节拍计数器和 STCTRL 中的 COUNTFLAG 位清零。	0
31:24	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

18.5.4 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中 SYSTCKCAL 寄存器的值驱动（参见表 24）。

表 313. 系统定时器校准值寄存器 (SYST_CALIB - 0xE000 E01C) 位描述

位	符号	值	描述	复位值
23:0	TENMS		参见表 367。	0x4
29:24	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
30	SKEW		参见表 367。	0
31	NOREF		参见表 367。	0

18.6 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 2）提供，也可由参考时钟 （固定为 CPU 时钟频率的二分之一）提供。若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 50 MHz，则默认值为 10 毫秒中断速率。

18.7 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 LOAD 寄存器进行编程，以获得所需的时间间隔。
2. 通过写操作将 VAL 寄存器清零。这样能确保定时器使能后可以从 LOAD 值开始计数，而不是从任意值开始计数。

以下示例说明了选择用于不同系统配置的 SysTick 定时器重载值。所有示例都计算 10 ms 的中断间隔，因为旨在使用 SysTick 定时器，并且没有舍入误差。

系统时钟 = 72 MHz

取值 0x7 对 CTRL 寄存器进行编程，即可选择系统时钟作为时钟源并使能 SysTick 定时器和 SysTick 定时器中断。

RELOAD = （系统时钟频率 × 10 ms） –1 = (72 MHz × 10 ms) –1 = 720000 –1 = 719999 = 0x000AFC7F

系统节拍定时器时钟 = 24 MHz

取值 0x3 对 CTRL 寄存器进行编程，即可选择来自系统节拍时钟分频器的时钟 （使用 DIV = 3）作为时钟源并使能 SysTick 定时器和 SysTick 定时器中断。

RELOAD =（系统节拍定时器时钟频率 × 10 ms）–1 = (24 MHz × 10 ms) –1 = 240000 –1 = 239999 = 0x0003A97F

系统时钟 = 12 MHz

取值 0x7 对 CTRL 寄存器进行编程，即可选择系统时钟作为时钟源并使能 SysTick 定时器和 SysTick 定时器中断。

这种情况下，系统时钟源自 IRC 时钟。

RELOAD = （系统时钟频率 × 10 ms） –1 = (12 MHz × 10 ms) –1 = 120000 –1 = 119999 = 0x0001D4BF

19.1 本章导读

所有 LPC1315/16/17/45/46/47 器件上均配有 RI 定时器。

19.2 基本配置

RI 定时器是通过以下寄存器进行配置的：

- 寄存器接口电源（RI 定时器时钟）：在 SYSAHBCLKCTRL 寄存器中，设置表 19 中的位 25。

19.3 特性

- 根据主时钟运行的 48 位计数器。计数器可自由运行，或通过一个已产生的中断来复位。
- 48 位比较值。
- 48 位比较掩码。计数器值等于比较值时，会在遮蔽后生成中断。这样可实现简单比较无法实现的组合。

19.4 简介

“重复性中断定时器” (RIT) 提供了以指定时间间隔生成中断的通用方法，无需使用标准定时器。其目的是重复与 “操作系统中断”不相关的中断。如果有不同的系统要求， RIT 也可用作 Cortex-M3 系统节拍定时器的替代装置。

RI 定时器可搭配 ARM Cortex-M3; v. r2p1) 的嵌入式跟踪宏单元 (ETM) 用于 ETM 时间戳。RI 定时器可根据特定事件 （异常、从异常返回、跟踪 FIFO 刷新）将时间值定期插入 ETM 的跟踪数据流。使用 ETM 时间戳功能，可以关联多个跟踪数据流，从而粗略衡量代码性能。

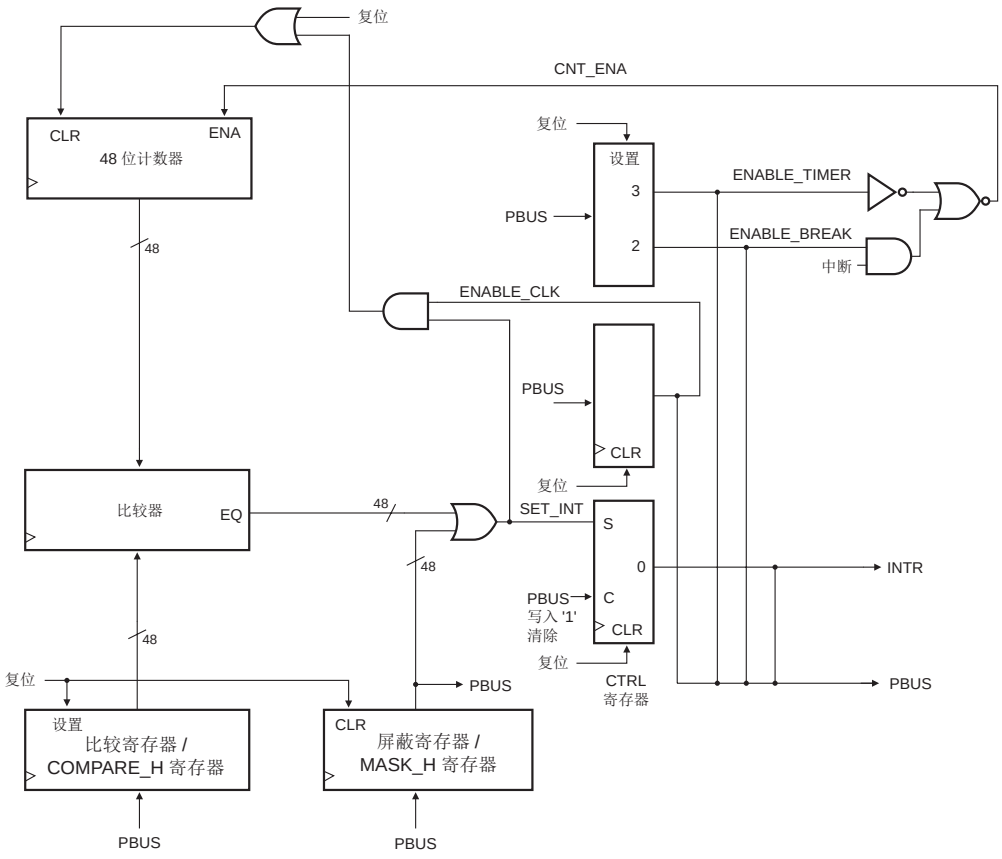


图 60. 重复性中断定时器 （RI 定时器）框图

19.5 寄存器描述

表 314. 寄存器简介：重复性中断定时器 (RIT) （基址 0x4006 4000）

名称	访问类型	地址	描述	复位值 [1]	参考
COMPVAL	R/W	0x000	比较值 LSB 寄存器。保存比较值的 32 个 LSB。	0xFFFF FFFF	表 315
MASK	R/W	0x004	屏蔽 LSB 寄存器。此寄存器保存屏蔽值的 32 个 LSB。在任何位写入 ‘1’ 都将强制计数器的相应位与比较寄存器进行比较。	0	表 316
CTRL	R/W	0x008	控制寄存器。	0xC	表 317
计数器	R/W	0x00C	计数器 LSB 寄存器。计数器的 32 个 LSB。	0	表 318
COMPVAL_H	R/W	0x010	比较值 MSB 寄存器。保存比较值的 16 个 MSB。	0x0000 FFFF	表 315
MASK_H	R/W	0x014	屏蔽 MSB 寄存器。此寄存器保存屏蔽值的 16 个 MSB。在任何位写入 ‘1’ 都将强制计数器的相应位与比较寄存器进行比较。	0	表 316
COUNTER_H	R/W	0x018	计数器 MSB 寄存器。计数器的 16 个 MSB。	0	表 318

[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。

19.5.1 RI 比较值 LSB 寄存器

表 315. RI 比较值 LSB 寄存器 （COMPVAL – 地址 0x4006 4000）位描述

位	符号	描述	复位值
31:0	RICOMP	比较寄存器。保存与计数器比较的比较值的 32 个 LSB。	0xFFFF FFFF

19.5.2 RI 屏蔽 LSB 寄存器

表 316. RI 屏蔽 LSB 寄存器 （MASK - 地址 0x4006 4004）位描述

位	符号	描述	复位值
31:0	RIMASK	掩码寄存器。此寄存器保存屏蔽值的 32 个 LSB。在任何位写入 1 将会覆盖计数器相应位和比较寄存器的比较结果 （导致寄存器位的比较始终为 True）。	0

19.5.3 RI 控制寄存器

表 317. RI 控制寄存器 （CTRL - 地址 0x4006 4008）位描述

位	符号	值	描述	复位值
0	RITINT		中断标志	0
		1	只要计数器值等于由 RICOMPVAL 和 RIMASK 寄存器的内容指定的已遮蔽比较值，此位就会被硬件设置为 1。 在此位写入 1 将清除至 0。写入 0 无效。	
		0	计数器值与已遮蔽的比较值不等。	
1	RITENCLR		定时器使能清除	
		1	只要计数器值等于由 COMPVAL/COMPVAL_H 和 MASK/MASK_H 寄存器的内容指定的已屏蔽比较值，定时器就将被清除为 0。这将发生在设置中断标志的同一时钟上。	0
		0	定时器不能清除至 0。	
2	RITENBR		为调试使能定时器	1
		1	当处理器因调试而停止时，定时器停止。	
		0	调试对定时器操作无影响。	
3	RITEN		定时器使能。	1
		1	定时器已使能。 **注：** 如果已在位 2 使能，则此定时器可能因调试停止而被否决。	
		0	定时器已禁用。	
31:4	-	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

19.5.4 RI 计数器 LSB 寄存器

表 318. RI 计数器寄存器 （COUNTER - 地址 0x4006 400C）位描述

位	符号	描述	复位值
31:0	RICOUNTER	正计数器的 32 个 LSB。连续计数，除非 CTRL 寄存器内的 RITEN 位被清除或进入调试模式（如果已被 RICTRL 内的 RITNEBR 位使能）。可在软件中载入任何值。	0

19.5.5 RI 比较值 MSB 寄存器

表 319. RI 比较值 MSB 寄存器 （COMPVAL_H – 地址 0x4006 4010）位描述

位	符号	描述	复位值
15:0	RICOMP	比较值MSB寄存器。保存与计数器比较的比较值的16个MSB。	0x0000 FFFF
31:16	-	保留。	-

19.5.6 RI 屏蔽 MSB 寄存器

表 320. RI 屏蔽 MSB 寄存器 （MASK_H - 地址 0x4006 4014）位描述

位	符号	描述	复位值
15:0	RIMASK	掩码寄存器。此寄存器保存屏蔽值的 16 个 MSB。在任何位写入 1 将会覆盖计数器相应位和比较寄存器的比较结果（导致寄存器位的比较始终为 True）。	0
31:16	-	保留。	-

19.5.7 RI 计数器 MSB 寄存器

表 321. RI 计数器 MSB 寄存器 （COUNTER_H - 地址 0x4006 4018）位描述

位	符号	描述	复位值
15:0	RICOUNTER	正计数器的 16 个 LSB。连续计数，除非 RICTRL 寄存器内的 RITEN 位被清除或输入了调试模式（如果已被 RICTRL 内的 RITNEBR 使能）。可在软件中载入任何值。	0
31:16	-	保留。	-

19.6 RI 定时器操作

复位后，计数器从0开始正计数。无论何时计数器值等于编程到COMPVAL和COMPVAL_H寄存器中的 48 位值，都将设置中断标志。通过向 MASK 和 MASK_H 寄存器中的相应位写入 1，可把任何位或位组合从此比较 （即，强制比较）中移除。如果 RITENCLR 位为低电平（默认状态），则有效比较仅导致设置中断标志。对计数顺序无影响，计数如常继续。计数器达到 0xFFFF FFFF FFFF 后，则在下一时钟翻转至 0，然后继续计数。如果 RITENCLR 位设置为 1，则有效比较也将导致计数器复位为零。计数将在下一时钟边沿从此处恢复。

可在软件中停止计数，方法为向 RITEN 位写入 ‘0’。如果已设置 RITENBR 位，则处理器因调试而停止时，计数也将停止。 RITEN 和 RITENBR 位都在复位时进行设置。

可在软件中清除中断标志，方法是向 RITINT 位写入 1。

在为计数器重新载入新值之前，软件必须先停止计数器。

计数器 (COUNTER/COUNTER_H)、 COMPVAL/COMPVAL_H 寄存器、 MASK/MASK_H 寄存器和 CTRL 寄存器都可由软件随时读取。

20.1 本章导读

所有 LPC1315/16/17/45/46/47 器件中的 ADC 模块都是一样的。

为获得更好的抗噪声性，在 LQFP64 封装上引出 VREFN/P 和 V_{DDA}/V_{SSA}。对于 LQFP48 和 HVQFN33 引脚封装，将 V_{DD} 和 V_{SS} 用于 ADC 基准引脚。

20.2 基本配置

ADC 是使用以下寄存器进行配置的：

1. 引脚：ADC 引脚功能是在 IOCON 寄存器模块中配置的 （表 55）。
2. 电源和外围设备时钟：在 SYSAHBCLKCTRL 寄存器中，设置位 13（表 19）。ADC 电源由 PDRUNCFG 寄存器控制 （表 42）。

注：APB 时钟提供 A/D 转换器的基本计时。每个转换器包括一个可编程分频器，将此时钟调整到逐次逼近所需的时钟（12 位模式中最大为 15.5 MHz； 10 位模式中最大为 31 MHz（BURST 位 = 0））。一个完全准确的转换要求有 31 个这样的时钟。

20.3 特性

- 12 位逐次逼近型模数转换器 (ADC)。
- 输入在 8 个引脚中多路复用。
- 掉电模式 （参见 SYSCON 模块的 PDRUNCFG 寄存器）（表 42）。
- 低功耗模式。
- 测量范围为 VREFN 至 VREFP（对于没有 VREFP 和 VREFN 引脚的引脚封装，为 0 V 至 V_{DD}）。不要超过 V_{DD} 电压电平。
- 12 位转换速率为 500 kSamples/s。
- 10 位，双转换速率模式 （高达 1 Msamples/s）。
- 用于单个或多个输入的连发转换模式。
- 输入引脚或定时器匹配信号跳变的选择性转换。

20.4 引脚说明

表 322 简要总结了 ADC 相关引脚。

表 322. ADC 引脚说明

引脚	类型	描述
AD[7:0]	输入	**模拟输入。**A/D 转换器元件可测量这些输入信号的电压。 **注：**尽管在数字模式下，引脚为 5 V 容限，但当引脚配置为模拟输入时，最大输入电压不得超过 V_{DD}。
VREFP、VREFN	参考	电压参考。这些引脚为 ADC 提供电压参考水平。注：不使用 ADC 时，VREFP 应连接到 VDD(3V3)， VREFN 应连接到 VSS。
V_{DD}、V_{DDA}	电源	模拟电源和接地。这些电压通常应与 VDD 和 VSS 相同，但应将其隔离以最大限度降低噪声和错误。注：不使用 ADC 时， V_{DDA} 应连接到 VDD，VSSA 应连接到 VSS。

必须通过 IOCON 寄存器选择 ADC 功能，以获得对监视引脚的准确电压读数。对于接受 ADC 输入的引脚，不可能选择数字功能而仍获得有效的 ADC 读数。只要在该引脚上选择数字功能，有一个内部电路就会将 ADC 硬件与相关引脚断开。

20.5 寄存器描述

ADC 所包含的寄存器，其结构如表 323 所示。

表 323. 寄存器简介：ADC （基址 0x4001 C000）

名称	访问类型	地址偏移	描述	复位值 [1]	参考
CR	R/W	0x000	A/D 控制寄存器。必须写入 CR 寄存器以选择操作模式，然后才会发生 A/D 转换。	0x0000 0000	表 324
GDR	R/W	0x004	A/D 全局数据寄存器。包含最新的 A/D 转换的结果。	不适用	表 325
-	-	0x008	保留。	-	-
INTEN	R/W	0x00C	A/D 中断使能寄存器。此寄存器包含使能位，该使能位使得每个 A/D 通道的 DONE 标志被包含或排除在生成 A/D 中断的过程中。	0x0000 0100	表 326
DR0	R/W	0x010	A/D 通道 0 数据寄存器。此寄存器包含通道 0 中最新完成的转换的结果。	不适用	表 327
DR1	R/W	0x014	A/D 通道 1 数据寄存器。该寄存器包含通道 1 中最新完成的转换的结果。	不适用	表 327
DR2	R/W	0x018	A/D 通道 2 数据寄存器。该寄存器包含通道 2 中最新完成的转换的结果。	不适用	表 327
DR3	R/W	0x01C	A/D 通道 3 数据寄存器。该寄存器包含通道 3 中最新完成的转换的结果。	不适用	表 327
DR4	R/W	0x020	A/D 通道 4 数据寄存器。该寄存器包含通道 4 中最新完成的转换的结果。	不适用	表 327
DR5	R/W	0x024	A/D 通道 5 数据寄存器。该寄存器包含通道 5 中最新完成的转换的结果。	不适用	表 327
DR6	R/W	0x028	A/D 通道 6 数据寄存器。该寄存器包含通道 6 中最新完成的转换的结果。	不适用	表 327
DR7	R/W	0x02C	A/D 通道 7 数据寄存器。该寄存器包含通道 7 中最新完成的转换的结果。	不适用	表 327
STAT	RO	0x030	A/D 状态寄存器。此寄存器包含全部 A/D 通道的 DONE 和 OVERRUN 标志，以及 A/D 中断标志。	0	表 328
TRM	R/W	0x034	A/D 微调寄存器	0xF00	表 329

[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。

20.5.1 A/D 控制寄存器 (CR)

A/D 控制寄存器提供的位可用于选择要转换的 A/D 通道、 A/D 计时、 A/D 模式和 A/D 起动触发。

表 324. A/D 控制寄存器 （CR - 地址 0x4001 C000）位描述

位	符号	值	描述	复位值
7:0	SEL		选择要采样和转换的 AD7:0 引脚。位 0 选择引脚 AD0，位 1 选择引脚 AD1，…，位 7 选择引脚 AD7。 在软件控制模式中 (BURST = 0)，只可选择一个通道，即这些位中只应有一位为 1。 在硬件扫描模式中 (BURST = 1)，可以选择任何数量的通道，即任何或全部位都可设为 1。如果将全部位设为 0，则会自动选择通道 0 (SEL = 0x01)。	0x00
15:8	CLKDIV		将主时钟 (PCLK_ADC) 进行分频 （CLKDIV 值 +1）得到 A/D 转换器的时钟。在软件控制模式中 （BURST 位 = 0），该时钟必须小于或等于 15.5 MHz （12 位模式）或者 31 MHz （10 位模式） 。一般来说，软件应编程该字段的最小值，该值会产生 15.5 MHz 或稍小频率的时钟，但是在某些情况下 （比如高阻抗模拟源），可能需要频率较低的时钟。	0
16	BURST		连发模式 **注：**如果将 BURST 设为 1，必须将 INTEN 寄存器的 ADGINTEN 位（表 326）设为 0。	0
		0	软件控制模式：转换由软件控制并要求有 31 个时钟。	
		1	硬件扫描模式：AD 转换器以 CLKS 字段选择的速率来重复转换，扫描（如果有必要）SEL 字段中的 1 选择的引脚。起动后进行的第一次转换对应于 SEL 字段中设为 1 的最低有效位，然后扫描设为 1 的下一更高位 （引脚）（适用情况下）。清除此位可终止重复转换，但是清除此位时正在执行的转换仍将完成。 **注意事项：**当 BURST = 1 时， START 位必须为 000，否则转换无法启动。	
20:17	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	-
21	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	-
22	LPWRMODE		低功耗模式	0
		0	禁用低功耗 ADC 模式。没有请求转换时，模拟电路仍保持激活。	
		1	使能低功耗 ADC 模式。 没有转换发生时，模拟电路将自动掉电。检测到任何 （硬件或软件）触发事件时，使能模拟电路。所需的启动时间过后，将启动请求的转换。一旦转换完成，如果没有进一步的转换挂起，模拟电路将再次掉电。 **注：**如果 ADC 掉电 （PDRUNCFG 寄存器的 ADC_PD 位处于高电平）或者器件处于深度睡眠模式、掉电模式或深度掉电模式，此模式不会为 A/D 上电。	
23	MODE10BIT		10 位转换速率模式	
		0	禁用 10 位转换速率模式。	
		1	使能高转换速率的 10 位转换速率模式。A/D 分辨率减少至 10 位（转换结果中的两个 LSB 将被强制为 0）。时钟速率（通过 CLKDIV 字段设置）可翻倍至最高 31 MHz，获得每秒高达 100 万次采样的翻转速率。	

表 324. A/D 控制寄存器 （CR - 地址 0x4001 C000）位描述 *（续）*

位	符号	值	描述	复位值
26:24	START		当 BURST 位为 0 时，这些位控制 A/D 转换是否开始，何时开始。	0
		0x0	未开始 （将 PDN 清除为 0 时应使用该值）。	
		0x1	立即开始转换。	
		0x2	当位 27 选择的边沿出现在 PIO0_2/SSEL/CT16B0_CAP0 时启动转换。	
		0x3	当位 27 选择的边沿出现在 PIO1_5/DIR/CT32B0_CAP0 时启动转换。	
		0x4	当位 27 选择的边沿出现在 CT32B0_MAT0 时启动转换 [1]。	
		0x5	当位 27 选择的边沿出现在 CT32B0_MAT1 时启动转换 [1]。	
		0x6	当位 27 选择的边沿出现在 CT16B0_MAT0 时启动转换 [1]。	
		0x7	当位 27 选择的边沿出现在 CT16B0_MAT1 时启动转换 [1]。	
27	EDGE		边沿控制。该位只有在 START 字段包含 010-111 时有效。	0
		0	在所选 CAP/MAT 信号的上升沿启动转换。	
		1	在所选 CAP/MAT 信号的下降沿启动转换。	
31:28	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

[1] 请注意，该操作无需在设备引脚上出现定时器匹配功能。

20.5.2 A/D 全局数据寄存器 (GDR)

A/D 全局寄存器包含最新的 A/D 转换的结果。这包括数据、 DONE 和溢出标志以及与数据相关的 A/D 通道的数量。

表 325. A/D 全局数据寄存器 （GDR - 地址 0x4001 C004）位描述

位	符号	描述	复位值
3:0	-	保留。这些位始终读为零。	0
15:4	V_VREF	当 DONE 为 1 时，该字段包含的是一个二进制小数，表示的是 SEL 字段所选定的 ADn 引脚的电压除以 V_{DD} 引脚上的电压，或者在 VREFP 至 VREFN 范围内。该字段为 0 表示 ADn 引脚处的电压小于、等于或接近于 V_{SS}/VREFN，而 0xFFF 表明 ADn 引脚处的电压接近于、等于或大于 V_{DD}/VREFP。	X
23:16	-	保留。这些位始终读为零。	0
26:24	CHN	这些位包含结果位 V_VREF 的转换通道。	X
29:27	-	保留。这些位始终读为零。	0
30	OVERRUN	在连发模式中，如果在产生 V_VREF 位结果的转换之前，一个或多个转换的结果丢失或被覆写，则此位为 1。	0
31	DONE	当 A/D 转换完成后，此位被设置为 1。该位在读取该寄存器和写 ADCR 时清零。如果在转换过程中写 ADCR，则该位置位并启动新的转换。	0

20.5.3 A/D 中断使能寄存器 (INTEN)

使用此寄存器可控制在转换完成时哪些 A/D 通道产生中断。例如，可能需要使用一些 A/D 通道不断地对传感器执行转换来监视它们。无论何时需要，都可通过应用程序来读取最新的结果。在这种情况下，不希望在某些 A/D 通道的每次转换结束时都有中断。

表 326. A/D 中断使能寄存器 （INTEN - 地址 0x4001 C00C）位描述

位	符号	描述	复位值
7:0	ADINTEN	使用这些位可控制在转换完成时哪些 A/D 通道产生中断。当位 0 为 1 是，A/D 通道 0 中的转换完成将产生中断，当位 1 为 1，A/D 通道 1 中的转换完成将产生中断，等等。	0x00
8	ADGINTEN	当该位为 1 时，使能 ADDR 中的全局 DONE 标志来产生中断。当该位为 0 时，只有 ADINTEN 7:0 使能的单个 A/D 通道将产生中断。 **注：**在连发模式中，该位必须设为 0（BURST = CR 寄存器中的 1）。	1
31:9	-	保留。未使用，始终为 0。	0

20.5.4 A/D 数据寄存器 （DR0 至 DR7）

A/D 数据寄存器保存 A/D 转换完成时的结果，也包括表明转换何时完成及转换超限何时发生的标志。

表 327. A/D 数据寄存器 （DR0 到 DR7 - 地址 0x4001 C010 到 0x4001 C02C）位描述

位	符号	描述	复位值
3:0	-	保留。	0
15:4	V_VREF	当 DONE 为 1 时，此字段包含一个二进制小数，表示的是 ADn 引脚的电压 （在 VREFP 至 VREFN 范围内）。该字段为 0 表示 ADn 引脚处的电压小于、等于或接近于 VREFN/V_{SS}，而 0xFFF 表明 AD 输入处的电压接近于、等于或大于 VREFP/VDD。	不适用
29:16	-	保留。	0
30	OVERRUN	在连发模式下，如果在产生 V_VREF 位结果转换之前一个或多个转换结果丢失或被覆盖，则该位置 1。读取该寄存器可清除此位。	0
31	DONE	当 A/D 转换完成后，此位被设置为 1。读取寄存器时将清除此位。	0

20.5.5 A/D 状态寄存器 (STAT)

利用 A/D 状态寄存器可以同时检查所有 A/D 通道的状态。DRn 寄存器中出现的每个 A/D 通道的 DONE 和 OVERRUN 标志都镜像在 ADSTAT 中。中断标志（所有 DONE 标志的逻辑 OR）也可在 ADSTAT 中找到。

表 328. A/D 状态寄存器 （STAT - 地址 0x4001 C030）位描述

位	符号	描述	复位值
7:0	DONE	这些位镜像了出现在每个 A/D 通道 n 的结果寄存器中的 DONE 状态标志。	0
15:8	OVERRUN	这些位镜像了出现在每个 A/D 通道 n 的结果寄存器中的 OVERRRUN 状态标志。读取 ADSTAT 允许同时检查所有 A/D 通道的状态。	0
16	ADINT	此位为 A/D 中断标志。当任何单个 A/D 通道 Done 标志被断言并使能，以通过 ADINTEN 寄存器促成 A/D 中断时，该位为 1。	0
31:17	-	保留。未使用，始终为 0。	0

20.5.6 A/D 微调寄存器 (TRM)

该寄存器将由引导代码在起动时设置。它包含 DAC 和 ADC 的微调值。ADC 的偏移微调值可由用户覆盖。读取该寄存器时全部 12 位都可见。

表 329. A/D 微调寄存器 （TRM - 地址 0x4001 C034）位描述

位	符号	描述	复位值
3:0	-	保留。	不适用
7:4	ADCOFFS	ADC 操作的偏移微调位。由引导代码初始化。可由用户覆盖。	0
11:8	TRIM	由引导代码写入。不能由用户覆盖。写入引导代码后锁定这些位。	1111
31:12	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	-

20.6 操作

20.6.1 硬件触发转换

如果 ADCR0 中的 BURST 位为 0，且 START 字段包含 010-111，A/D 转换器将在选定引脚或定时器匹配信号发生跳变时启动转换。

20.6.2 中断

当 ADSTAT 寄存器中的 ADINT 位为 1 时，有中断请求送至中断控制器。当已使能中断（通过 ADINTEN 寄存器）的 A/D 通道的任何 DONE 位为 1 时，ADINT 位为 1。软件可以使用中断控制器中与 ADC 相应的中断使能位来控制是否产生中断。要清除相应的 DONE 标志，必须读取产生中断的 A/D 通道的结果寄存器。

20.6.3 精度与数字接收器

A/D 转换器可用于测量任意 ADC 输入引脚处的电压，无论 IOCON 模块中的引脚设置如何。在 IOCON 寄存器中选择 ADC 功能，可通过禁用引脚的数字接收器，提高转换精度（另请参见第 7.3.7 节）。

20.6.4 可选操作模式

CR 寄存器中可以选择两种可选 A/D 操作模式：

1. 10 位模式。在该模式下，将损失两位的 ADC 精确度以使转换速率翻倍。选择该模式时，最大 ADC 时钟速率将增加至 31 MHz （BURST 位 = 0）。使能该模式时，转换结果中的两个 LSB 将被强制为 00。
2. 低功耗模式。选择该模式时，如果没有转换正在进行， ADC 的模拟部分将自动关闭。发生任何硬件或软件触发事件时， ADC 都会自动重启 （只要 ADC 未在 PDRUNCFG 寄存器中掉电或者器件处于深度睡眠、掉电或深度掉电模式）。请求的转换完成时，如果没有新的转换挂起，模拟 ADC 电路将返回掉电状态。设置 BURST 位将覆盖低功耗模式并防止 ADC 自动掉电。

当 ADC 不在连续使用状态时，低功耗模式可节约可观的电量，但在触发事件和开始采样及转换之间会有延迟。

这两个可选模式并不互斥。

21.1 本章导读

各种 Flash 配置请参见表 330。

表 330. LPC1315/16/17/45/46/47 闪存配置

产品型号	闪存	EEPROM	通过UART进行ISP	通过USB进行ISP
LPC1345FHN33	32	2	是	是
LPC1345FBD48	32	2	是	是
LPC1346FHN33	48	4	是	是
LPC1346FBD48	48	4	是	是
LPC1347FHN33	64	4	是	是
LPC1347FBD48	64	4	是	是
LPC1347FBD64	64	4	是	是
LPC1315FHN33	32	2	是	否
LPC1315FBD48	32	2	是	否
LPC1316FHN33	48	4	是	否
LPC1316FBD48	48	4	是	否
LPC1317FHN33	64	4	是	否
LPC1317FBD48	64	4	是	否
LPC1317FBD64	64	4	是	否

注：除了 ISP 和 IAP 命令，还可在闪存控制器模块中访问寄存器以配置闪存访问时间，见第 21.16.1 节。

21.2 启动引导程序

启动引导程序控制复位后的初始操作，并提供闪存编程的方法。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。

ISP/IAP 调用可以读取启动引导程序版本 （见第 21.13.12 节或第 21.14.6 节）。

21.3 特性

- 在线系统编程：在系统编程(ISP)是使用启动引导程序软件和UART串行端口对片闪存的编程或重新编程。当器件位于最终用户端时，可执行此操作。
- 在应用编程：在应用编程 (IAP) 的功能是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。
- 小尺寸 (256 B) 页擦除编程。
- 闪存访问时间可以由闪存控制器模块中的寄存器配置。
- 一个扇区的擦除时间为 100 ms ± 5%。一个 256 字节模块的编程时间为 1 ms ± 5%。

21.4 描述

每当部件上电或复位时，启动引导程序代码即被执行一次 （见图 61）。加载程序还可以执行 ISP 命令处理程序或用户应用程序代码。复位时引脚 PIO0_1 的低电平被视为启动 ISP 命令处理程序 （或在 LPC1345/46/47 上，如果引脚 PIO0_3 位于高电平，则为 USB 设备处理程序）的外部硬件请求，无需先检查有效用户代码。

假设在 $\overline{\text{RESET}}$ 引脚上产生上升沿时，电源引脚处于标称的电平，那么在最多 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。如果找到有效的用户程序，则会将执行控制转交给它。如果没找到有效的用户程序，将调用自动波特率例程。

对于 LPC1345/46/47 器件， PIO0_3 的状态决定将使用 UART 还是 USB 接口：

- 如果PIO0_3的采样结果为高电平，启动引导程序将把LPC1345/46/47当作MSC USB设备连接到 PC 主机。LPC1345/46/47 闪存空间表示为主机的 Windows 操作系统中的驱动器。
- 如果PIO0_3的采样结果为低电平，启动引导程序将引脚PIO0_18和PIO0_19用于RXD和 TXD 以配置 UART 串行端口，并调用 ISP 命令处理程序。

注：引脚 PIO0_1 的采样可通过编程闪存位置 0x0000 02FC 来禁用 （见第 21.12.1 节）。

21.5 复位后的存储器映射

引导模块大小为 16 kB，在存储器区域中的起始地址为 0x1FFF 0000。启动引导程序被设计为从该存储器区域运行，但 ISP 和 IAP 软件都会占用部分的片内 RAM。本章后面的部分描述了 RAM 的使用情况。复位后，驻留在片内 Flash 存储器的引导模块中的中断向量也变为有效。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 闪存内容保护机制

LPC1315/16/17/45/46/47 具备支持纠错代码 (ECC) 的闪存。纠错模块有双重目的。首先，它将从存储器读取的数据字解码为输出数据字。其次，它对要被写入存储器的数据字进行编码。纠错功能由利用汉明码的单位纠错组成。

ECC 的操作对运行的应用程序是透明的。ECC 内容本身存储在一个闪存中，该闪存无法通过用户代码自行读取或写入。一个 ECC 字节相当于用户可访问的闪存的每 128 个连续位。因此，从0x0000 0000到0x0000 000F的闪存字节由第一个ECC字节保护，从0x0000 0010到 0x0000 001F 的闪存字节由第二个 ECC 字节保护，以此类推。

每当 CPU 请求从用户闪存读取时，将评估包含指定存储器位置的 128 位原始数据和匹配的 ECC 字节。如果 ECC 机制在获取的数据中检测到一个错误，则在将数据提供给 CPU 之前会应用纠错。请求写入用户闪存时，在写入用户指定内容的同时将写入在 ECC 存储器中计算和存储的 ECC 匹配值。

擦除闪存的一个扇区时，相应的 ECC 字节也被擦除。一旦写入 ECC 字节后，除非先行擦除，否则无法进行更新。因此，要让实施的 ECC 机制正确运行，必须将数据以 16 字节（或 16 的倍数）一组写入闪存，如上所述对齐。

21.7 有效用户代码的判定标准

保留的 ARM Cortex-M3 异常向量位置 7（向量表中偏移量 0x0000 001C）应此包含表中条目 0 到 6 的校验和的二补数。这将使表中头 8 个条目的校验和为 0。启动引导程序代码将求闪存扇区 0 的前 8 个位置的校验和。如果结果为 0，则执行控制将被转交到用户代码。

如果签名无效，则自动波特率例程通过串行端口 (UART) 与主机进行同步。

如果选定 UART，主机应当发送一个同步字符 “？ ” (0x3F) 并等待响应。主机侧的串口设置应是 8 个数据位，1 个停止位，没有奇偶校验。自动波特率例程按其自身的频率测量接收到的同步字符的位时间，并对串口的波特率发生器进行编程。它还向主机发送一个 ASCII 字符串 （“Synchronized<CR><LF>”）。作为响应，主机应当发送同一字符串 （“Synchronized<CR><LF>”）。自动波特率例程检查接收到的字符以验证同步。如果通过验证，则向主机发送 “OK<CR><LF>” 字符串。主机应当通过发送正在运行部分的晶频 （单位为 KHz）进行响应。例如，如果器件以 10 MHz 运行，主机的响应当为 “10000<CR><LF>”。在接收到晶频后再向主机发送 “OK<CR><LF>”。如果同步未得到验证，则自动波特率例程再次等待同步字符。在用户调用 ISP 的情况下要使自动波特率正确工作，CCLK 频率应当大于或等于 10 MHz。在 USART ISP 模式下，LPC1315/16/17/45/46/47 通过 IRC 来计时，并忽略晶频。

接收到晶体频率后，将立即初始化部件并调用 ISP 命令处理程序。因为安全方面的原因，在执行将导致闪存擦除 / 写入操作的命令和 “Go” 命令之前必需有 “Unlock” 命令。其他命令可直接执行，不需要先执行解锁命令。每个 ISP 会话都要求执行一次 Unlock 命令。解锁命令在 第 326 页上的章节 21.13 “ISP 命令” 中进行了说明。

21.8 ISP/IAP 通信协议

所有 ISP 命令都应作为单一 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，多余的 <CR> 和 <LF> 将被忽略。所有 ISP 响应都是以 <CR><LF> 终止的 ASCII 字符串形式发送。数据以 UU 编码格式发送和接收。

21.8.1 ISP 命令格式

"Command Parameter_0 Parameter_1 ...Parameter_n<CR><LF> " "Data" （Data 只适用于写命令）。

21.8.2 ISP 响应格式

"Return_Code<CR><LF>Response_0<CR><LF>Response_1<CR><LF> ...Response_n<CR><LF>" "Data" （Data 只适用于读命令）。

21.8.3 ISP 数据格式

数据流采用 UU 编码格式。 UU 编码算法将 3 字节的二进制数据转换成 4 字节的可打印的 ASCII 字符集。它比十六进制格式更有效，十六进制格式将 1 字节的二进制数据转换成 2 字节的 ASCII 十六进制。发送 20 个 UU 编码行后，发送端应发送校验和。 UU 编码行的长度不应超过 61 个字符 （字节），也就是它可容纳 45 个数据字节。接收端应将其与已接收字节的校验和比较。如果校验和匹配，接收器应当响应 “OK<CR><LF>” 来继续下一次发送。如果校验和不匹配，则接收器应当响应 “RESEND<CR><LF>” 。作为响应，发送端应重新发送字节。

21.8.4 ISP 流量控制

为了防止由缓冲区超限造成的数据丢失，使用软件 XON/XOFF 流控制模式。当数据快速到达时，发送 ASCII 控制字符 DC3 （停止）使数据流停止。发送 ASCII 控制字符 DC1 （启动）恢复数据流。主机也应支持同样的流控制模式。

21.8.5 ISP 命令中止

命令可通过发送 ASCII 控制字符 “ESC” 来中止。此功能在 “ISP 命令” 部分没有作为命令被记录。接收到转义码后， ISP 命令处理程序将等待新的命令。

21.8.6 ISP 过程中的中断

在任何复位后，位于 Flash 引导块内的引导块中断向量都有效。

21.8.7 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。用户应在使用闪存擦除 / 写入 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。

21.8.8 ISP 命令处理程序使用的 RAM

ISP 命令使用片内地址 0x1000 017C 到 0x1000 025B 范围内的 RAM。用户可以使用此区，但其内容在重置时可能会丢失。 Flash 编程命令使用片内 RAM 最顶端的 32 字节。协议栈位于 RAM 的前 32 字节，可使用的最大协议栈为 256 字节，向下增长。

21.8.9 IAP 命令处理程序使用的 RAM

Flash 编程命令使用片内 RAM 最顶端的 32 字节。在分配给用户的堆栈空间中，可使用的最大堆栈为 128 字节，向下增长。

21.9 USB 通信协议

LPC1345/46/47 被枚举为 PC 或其他嵌入式系统的海量存储设备类 (MSC) 设备。要通过 USB 接口进行连接，器件必须使用频率为 12 MHz 的外部晶体。MSC 设备提供与 PC Windows 操作系统的简易集成。闪存空间表示为主机文件系统中的驱动器。整个可用的用户闪存都映射在主机文件夹中默认名称为“firmware.bin”的文件中，其大小与 LPC1345/46/47 闪存相同。可删除“firmware.bin”文件并将新文件复制到目录，从而更新闪存中的用户代码。请注意，新闪存镜像文件的文件名并不重要。复位或在一个电源周期之后，新文件将以默认名称 “firmware.bin”出现在主机文件系统中。

注：仅 Windows 操作系统支持 USB ISP 命令。

代码读保护 （CRP，见表 331）等级决定如何重新编程闪存：

如果使能 CRP1 或 CRP2，则在删除文件时会擦除用户闪存。

如果使能 CRP1 或未选择 CRP，则在复制新文件时，会擦除并重新编程用户闪存。但是，只会擦除和重新编程新文件占用的区域。

注：对 LPC1345/46/47 闪存镜像文件夹支持的 Windows 命令只有复制和删除。

可以为通过 USB 更新的闪存镜像使能三个代码读保护 (CRP) 等级 （有关详情，请参见第 21.12 节）。 MSCD 上的卷标指示 CRP 状态。

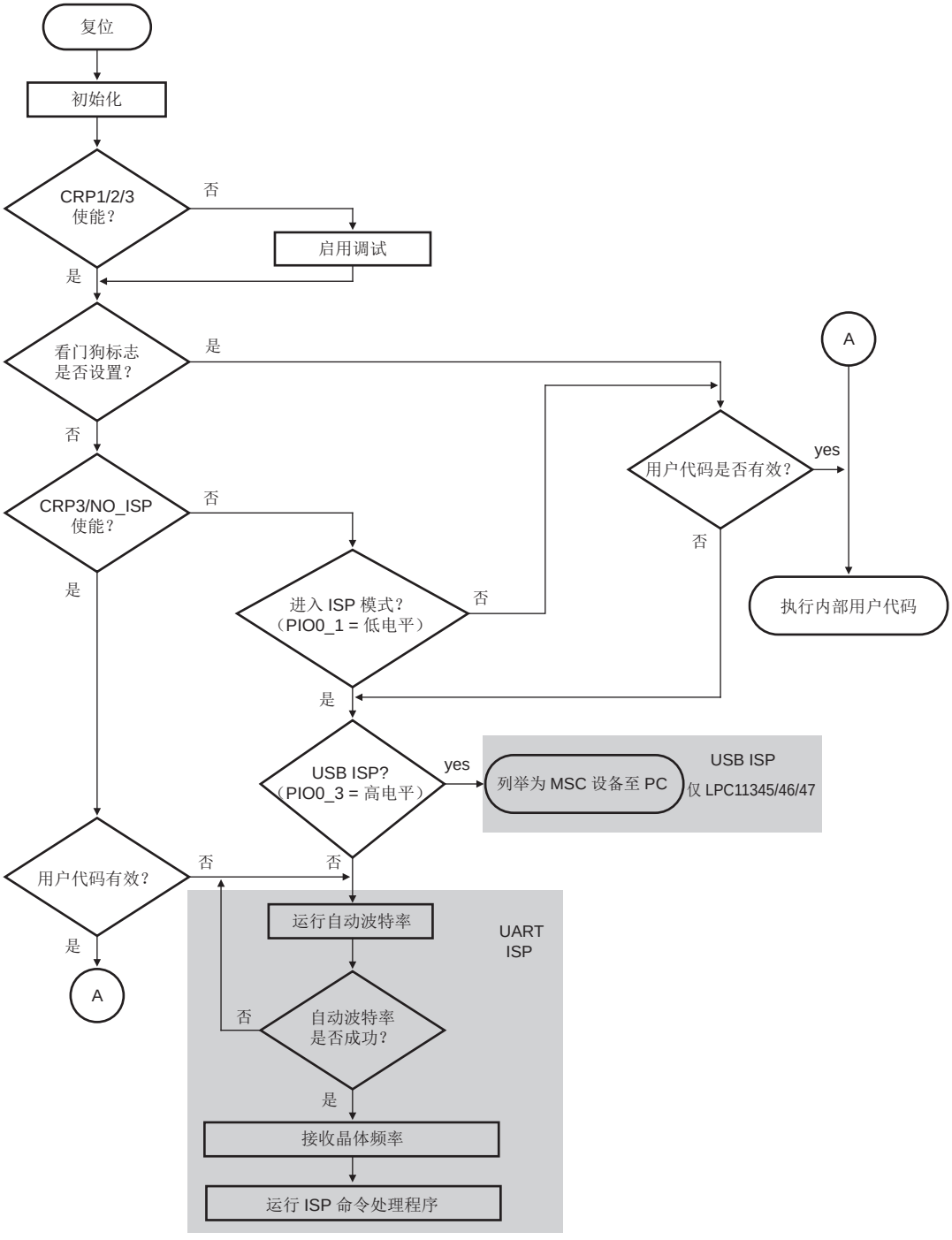
表 331. 用于 USB 启动镜像的 CRP 等级

CRP 状态	卷标	描述
无 CRP	CRP DISABLD	可读写用户闪存。
CRP1	CRP1 ENABLD	无法读取用户闪存内容，但可进行更新。闪存扇区根据新的固件镜像进行更新。
CRP2	CRP2 ENABLD	无法读取用户闪存内容，但可进行更新。写入新固件镜像之前，擦除整个用户闪存。
CRP3	CRP3 ENABLD	无法读取或更新用户闪存内容。启动引导程序总是执行用户应用程序 （如果有效）。

21.9.1 使用说明

通过 Flash Magic 或 Serial Wire Debugger (SWD) 编程闪存镜像时，编程工具将自动插入用户代码有效签名。使用 USB ISP 时，用户代码有效签名必须为向量表的一部分，或者必须对 axf 或二进制文件进行后处理以插入校验和。

21.10 Boot 处理流程图



(1) 有关晶频处理的详情，请参见第 21.14.8 节。

(2) 有关基于 CRP 设置的可用 ISP 命令的更多详细信息，参见第 21.12 节。

图 61. Boot 处理流程图

21.11 扇区号

一些 IAP 和 ISP 命令在扇区操作，需指定扇区号。此外，LPCLPC1315/16/17/45/46/47 支持一个页擦除命令。下表显示了 LPC1315/16/17/45/46/47 设备的页号、扇区号和存储器地址间的对应关系。

扇区的大小为 4 kB，页的大小为 256 字节。一个扇区包含 16 页。

表 332. LPC1315/16/17/45/46/47 闪存扇区

扇区号	扇区大小 [kB]	页号	地址范围	LPC1345/ LPC1315	LPC1346/ LPC1316	LPC1347/ LPC1317
0	4	0 -15	0x0000 0000 - 0x0000 0FFF	是	是	是
1	4	16 - 31	0x0000 1000 - 0x0000 1FFF	是	是	是
2	4	32 - 47	0x0000 2000 - 0x0000 2FFF	是	是	是
3	4	48 - 63	0x0000 3000 - 0x0000 3FFF	是	是	是
4	4	64 - 79	0x0000 4000 - 0x0000 4FFF	是	是	是
5	4	80 - 95	0x0000 5000 - 0x0000 5FFF	是	是	是
6	4	96 - 111	0x0000 6000 - 0x0000 6FFF	是	是	是
7	4	112 - 127	0x0000 7000 - 0x0000 7FFF	是	是	是
8	4	128 - 143	0x0000 8000 - 0x0000 8FFF	否	是	是
9	4	144 - 159	0x0000 9000 - 0x0000 9FFF	否	是	是
10	4	160 - 175	0x0000 A000 - 0x0000 AFFF	否	是	是
11	4	176 - 191	0x0000 B000 - 0x0000 BFFF	否	是	是
12	4	192 - 207	0x0000 C000 - 0x0000 CFFF	否	否	是
13	4	208 - 223	0x0000 D000 - 0x0000 DFFF	否	否	是
14	4	224 - 239	0x0000 E000 - 0x0000 EFFF	否	否	是
15	4	240 - 255	0x0000 F000 - 0x0000 FFFF	否	否	是

21.12 代码读保护 (CRP)

代码读保护是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 Flash 的访问和 ISP 的使用的一种机制。需要时，可通过在 Flash 地址单元 0x0000 02FC 编程特定的格式来调用 CRP。 IAP 命令不受代码读取保护影响。

注意事项：CRP 所作出的任何改变只有在器件经过一个电源周期之后才会生效。

表 333. 代码读保护 (CRP) 选项

名称	在 0x0000 02FC 处编程的格式	描述
NO_ISP	0x4E69 7370	阻止对 PIO0_1 引脚进行采样而进入 ISP 模式。 PIO0_1 引脚用作其他用途。
CRP1	0x12345678	通过 SWD 引脚访问芯片被禁用。此模式允许使用下列 ISP 命令和限制更新部分闪存： • 写入 RAM 命令不能访问在 0x1000 0300 以下的 RAM。 • “将 RAM 内容复制到 Flash”命令不能写入扇区 0。 • 只有在选择擦除所有扇区时，擦除命令才能擦除扇区 0。 • 比较命令被禁用。 • 读取存储器命令被禁用。 此模式在要求 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 Flash 部分更新的情况下比较命令被禁用，因此辅助加载程序应执行校验和机制来验证 Flash 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： • 读取内存 • 写入 RAM • 运行 • 将 RAM 内容复制到 Flash • 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，则禁止通过拉低 PIO0_1 来进入 ISP。 该模式有效禁止了通过 PIO0_1 引脚来强行进入 ISP 的行为。用户的应用程序可决定是调用 IAP 来进行闪存更新还是通过 UART0 重新调用 ISP 命令来进行闪存更新。 **注意：如果选择了 CRP3，此后该器件将无法执行厂商测试。**

表 334. 代码读保护硬件 / 软件的相互作用

CRP 选件	用户代码有效性	复位时 PIO0_1 引脚电平	SWD 是否使能	LPC1315/16/17/45/46/47 进入 ISP 模式	ISP模式下部分闪存更新
无	无	x	有	有	有
无	有	高电平	有	无	不适用
无	有	低电平	有	有	有
CRP1	有	高电平	无	无	不适用
CRP1	有	低电平	无	有	有
CRP2	有	高电平	无	无	不适用
CRP2	有	低电平	无	有	无
CRP3	有	x	无	无	不适用
CRP1	无	x	无	有	有
CRP2	无	x	无	有	无
CRP3	无	x	无	有	无

表 335. 不同 CRP 等级下允许使用的 ISP 命令

ISP 命令	CRP1	CRP2	CRP3 （不允许在 ISP 模式下进入）
解锁	是	是	不适用
设置波特率	是	是	不适用
回标	是	是	不适用
写入 RAM	是；只在 0x1000 0300 以上	否	不适用
读取内存	否	否	不适用
准备写操作扇区	是	是	不适用
将 RAM 内容复制到 Flash	是；不到扇区 0	否	不适用
运行	否	否	不适用
擦除扇区	是；只有在擦除所有扇区时才能擦除扇区 0。	是；只有所有扇区	不适用
空白检查扇区	否	否	不适用
读取部件 ID	是	是	不适用
读 Boot 代码版本	是	是	不适用
比较	否	否	不适用
读 UID	是	是	不适用

如果已使能了任何 CRP 模式且通过 ISP 允许访问芯片，不受支持或受限的 ISP 命令将被终止，同时返回代码 CODE_READ_PROTECTION_ENABLED。

21.12.1 ISP 入口保护

除了 3 种 CRP 模式外，用户可防止对 PIO0_1 引脚采样而进入 ISP 模式，从而释放 PIO0_1 引脚，使其用于其它方面。这称为 NO_ISP 模式。可通过对 0x0000 02FC 位置的 0x4E69 7370 模式进行编程进入 NO_ISP 模式。

NO_ISP 模式和 CRP3 模式完全一样，除 SWD 访问之外 （在 NO_ISP 模式中允许，但在 CRP3 模式中禁用）。 NO_ISP 模式不提供任何代码保护。

21.13 ISP 命令

下列命令被 ISP 命令处理程序接受。对于每条命令都支持详细的状态码。当接收到未定义的命令时，命令处理程序发送返回码 INVALID_COMMAND。命令和返回码都采用 ASCII 格式。

只有当接收到的 ISP 命令执行完毕且可通过主机给出新的 ISP 命令时， ISP 命令处理程序才会发送 CMD_SUCCESS。“Set Baud Rate”、“Write to RAM”、“Read Memory”和 "Go”不遵守此规则。

表 336. ISP 命令总结

ISP 命令	用法	说明见：
解锁	U < 解锁代码 >	表 337
设置波特率	B < 波特率 > < 停止位 >	表 338
回标	A < 设定 >	表 339
写入 RAM	W < 起始地址 > < 字节数 >	表 340
读取内存	R < 地址 > < 字节数 >	表 341
准备写操作扇区	P < 起始扇区号 > < 结束扇区号 >	表 342
将RAM内容复制到Flash	C <Flash 地址 > <RAM 地址 > < 字节数 >	表 343
运行	G < 地址 > < 模式 >	表 344
擦除扇区	E < 起始扇区号 > < 结束扇区号 >	表 345
空白检查扇区	I < 起始扇区号 > < 结束扇区号 >	表 346
读取部件 ID	J	表 347
读 Boot 代码版本	K	表 349
比较	M < 地址 1> < 地址 2> < 字节数 >	表 350
读 UID	N	表 351

21.13.1 解锁 < 解锁代码 >

表 337. 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
描述	此命令用于解锁闪存 “写入”、“擦除”和 “运行”命令。
示例	"U 23130<CR><LF>" 解锁闪存写入 / 擦除以及运行命令。

21.13.2 设置波特率 < 波特率 > < 停止位 >

表 338. ISP 设置波特率命令

命令	B
输入	波特率：9600 | 19200 | 38400 | 57600 | 115200 停止位：1 | 2
返回码	CMD_SUCCESS | INVALID_BAUD_RATE | INVALID_STOP_BIT | PARAM_ERROR
描述	此命令用于更改波特率。命令处理程序发送 CMD_SUCCESS 返回码后，新的波特率生效。
示例	“B 57600 1<CR><LF>”设置串口波特率 57600bps 和 1 个停止位。

21.13.3 回应 < 设定 >

表 339. ISP 回应命令

命令	A
输入	设置：打开 = 1 | 关闭 = 0
返回码	CMD_SUCCESS | PARAM_ERROR
描述	回标命令的默认设置是 ON。当其为 ON 时， ISP 命令处理程序将接收到的串行数据发送回主机。
示例	“A 0<CR><LF>”将回应关闭。

21.13.4 写 RAM < 起始地址 > < 字节数 >

只有接收到 CMD_SUCCESS 返回码后，主机才应发送数据。当发送完 20 个 UU 编码行之后主机应当发送校验和。校验和是通过把原始数据 （UU 编码前）字节的值叠加起来产生的，当发送完 20 个 UU 编码行后该值会重置。任何 UU 编码行的长度不应超过 61 个字符（字节），也就是说，它可以携带 45 个数据字节。当数据少于 20 个 UU 编码行时，校验和应当按照实际发送的字节数进行计算。 ISP 命令处理程序将其与已接收字节的校验和作比较。如果校验和匹配，那么 ISP 命令处理程序响应 “OK<CR><LF>”来继续下一次发送。如果校验和不匹配，那么 ISP 命令处理程序响应 “RESEND<CR><LF>”。作为响应，主机应当重新发送字节。

表 340. ISP 写 RAM 命令

命令	W
输入	**起始地址：**要写入数据字节的 RAM 地址。此地址应是字边界。 **字节数：**写入的字节数。此数字应是 4 的倍数。
返回码	CMD_SUCCESS | ADDR_ERROR （地址不在字边界）| ADDR_NOT_MAPPED | COUNT_ERROR （字节数不是 4 的倍数） | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
描述	此命令用于下载数据到 RAM。数据应为 UU 编码格式。当代码读保护使能时该命令被禁止。
示例	“W 268436224 4<CR><LF>”向地址 0x1000 0300 写入 4 个字节数据。

21.13.5 读存储器 < 地址 > < 字节数 >

数据流后是命令成功返回代码。发送完 20 个 UU 编码行之后发送校验和。校验和是通过把原始数据 （UU 编码前）字节的值叠加起来产生的，当发送完 20 个 UU 编码行后该值会重置。任何 UU 编码行的长度不应超过 61 个字符 （字节），也就是说，它可以携带 45 个数据字节。当数据少于 20 个 UU 编码行时，校验和按照实际发送的字节数进行计算。主机应将其与已接收字节的校验和比较。如果校验和匹配，那么主机应当响应 “OK<CR><LF>” 来继续下一次发送。如果校验和不匹配，主机应当响应 “RESEND<CR><LF>” 。作为响应， ISP 命令处理程序再次发送数据。

表 341. ISP 读存储器命令

命令	R
输入	**起始地址：**被读出数据字节的地址。此地址应是字边界。 **字节数：**读出的字节数。此数字应是 4 的倍数。
返回码	CMD_SUCCESS，后面是 < 实际数据 （UU 编码） > | ADDR_ERROR （地址不在字边界）| ADDR_NOT_MAPPED | COUNT_ERROR （字节数不是 4 的倍数）| PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
描述	此命令用于从 RAM 或闪存中读取数据。当代码读保护使能时该命令被禁止。
示例	“R 268435456 4<CR><LF>”从地址 0x1000 0000 读出 4 个字节数据。

21.13.6 准备写操作的扇区 < 起始扇区号 > < 结束扇区号 >

此命令使闪存写 / 擦除操作分为两步。

表 342. ISP 准备写操作命令的扇区

命令	P
输入	**起始扇区号** **结束扇区号：**应当大于或等于起始扇区号。
返回码	CMD_SUCCESS | BUSY | INVALID_SECTOR | PARAM_ERROR
描述	该命令必须在执行 “将 RAM 内容复制到 Flash”或 “擦除扇区”命令之前执行。“将 RAM 内容复制到 Flash”或“擦除扇区”命令的成功执行会使相关的扇区再次被保护。不能使用该命令准备引导块。要准备单一的扇区，使用相同的 “起始”和 “结束”扇区号。
示例	“P 0 0<CR><LF>”准备 Flash 扇区 0。

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

写入闪存时，适用以下限制：

1. 可通过复制 RAM 到闪存命令写入闪存的最小数据量为 256 字节 （等于一页）。
2. 一页包含 16 个闪存字 （行），每个闪存写操作可修改的最小数据量为一个闪存字 （一行）。从 ECC 的应用到闪存写操作，均遵循此限制，见第 21.6 节。
3. 要防止写干扰 （闪存的内部机制），应在同一页内执行 16 个连续写操作后，再执行擦除。请注意，擦除操作随后将擦除整个扇区。

   注：一页被写入 16 次后，仍可写入同一扇区内的其他页，无需执行扇区擦除 （假设这些页之前已被擦除）。

表 343. ISP 复制命令

命令	C
输入	**闪存地址 (DST)：**要写入数据字节的目标 Flash 地址。目标地址应为 256 字节边界。 **RAM 地址 (SRC)：**读出数据字节的源 RAM 地址。 **字节数：**写入的字节数。应为 256 | 512 | 1024 | 4096。
返回码	CMD_SUCCESS | SRC_ADDR_ERROR （地址不在字边界） | DST_ADDR_ERROR （地址不在正确的边界） | 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 | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
描述	此命令用于闪存编程。在此命令前执行应 “准备写操作扇区”命令。一旦复制命令成功执行，受影响的扇区将自动被重新保护。不能使用该命令写入引导模块。当代码读保护使能时该命令被禁止。有关可写入的字节数，另请参见第 21.6 节。
示例	“C 0 268467504 512<CR><LF>”将 RAM 地址 0x1000 0800 开始的 512 字节复制到 Flash 地址 0。

21.13.8 运行 < 地址 > < 模式 >

表 344. ISP 运行命令

命令	G
输入	**地址：**代码执行起始的 Flash 或 RAM 地址。此地址应在字边界。 **模式：**T （执行 Thumb 模式下的程序） | A （执行 ARM 模式下的程序）。
返回码	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
描述	此命令用于执行驻留在 RAM 或闪存中的程序。一旦此命令执行成功，则可能不能返回 ISP 命令处理程序。当代码读保护使能时该命令被禁止。
示例	“G 0 A<CR><LF>”跳转到 ARM 模式下的地址 0x0000 0000 处。

21.13.9 擦除扇区 < 起始扇区号 > < 结束扇区号 >

表 345. ISP 擦除扇区命令

命令	E
输入	**起始扇区号** **结束扇区号：**应当大于或等于起始扇区号。
返回码	CMD_SUCCESS | BUSY | INVALID_SECTOR | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
描述	此命令用于擦除片内闪存的一个或多个扇区。引导块不能使用该命令来擦除。当代码读保护使能时，该命令只允许擦除所有用户扇区的内容。
示例	“E 2 3<CR><LF>”擦除 Flash 扇区 2 和 3。

21.13.10 扇区查空 < 起始扇区号 > < 结束扇区号 >

表 346. ISP 扇区查空命令

命令	I
输入	**起始扇区号：** **结束扇区号：**应当大于或等于起始扇区号。
返回码	CMD_SUCCESS | SECTOR_NOT_BLANK （后跟 < 第一个非空字的偏移量 > < 非空字的内容 >） | INVALID_SECTOR | PARAM_ERROR
描述	此命令用于空白检查片内闪存的一个或多个扇区。 **对扇区 0 查空总是失败，这是由于前 64 字节重新映射到闪存引导模块。** 使能 CRP 时，对于非空扇区的偏移和值，查空命令返回 0。无论 CRP 设置为何，都会正确报告空扇区。
示例	“I 2 3<CR><LF>”对 Flash 扇区 2 和 3 进行查空。

21.13.11 读器件标识号

表 347. ISP 读器件标识命令

命令	J
输入	无。
返回码	CMD_SUCCESS 后跟 ASCII 格式的器件标识号 （见 表 348 “LPC1315/16/17/45/46/47 设备标识号”）。
描述	此命令用于读取部件识别号。

表 348. LPC1315/16/17/45/46/47 设备标识号

设备	Hex 代码
LPC1345FHN33	0x2801 0541
LPC1345FBD48	0x2801 0541
LPC1346FHN33	0x0801 8542
LPC1346FBD48	0x0801 8542
LPC1347FHN33	0x0802 0543
LPC1347FBD48	0x0802 0543
LPC1347FBD64	0x0802 0543
LPC1315FHN33	0x3A01 0523
LPC1315FBD48	0x3A01 0523
LPC1316FHN33	0x1A01 8524
LPC1316FBD48	0x1A01 8524
LPC1317FHN33	0x1A02 0525
LPC1317FBD48	0x1A02 0525
LPC1317FBD64	0x1A02 0525

21.13.12 读 Boot 代码版本号

表 349. ISP 读 Boot 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。将其解释为 < 字节 1（主）>.< 字节 0（次）>。
描述	此命令用于读取引导代码版本号。

21.13.13 比较 < 地址 1> < 地址 2> < 字节数 >

表 350. ISP 比较命令

命令	M
输入	**地址 1(DST)**：要比较的数据字节的起始 Flash 或 RAM 地址。此地址应是字边界。 **地址 2(SRC)**：要比较的数据字节的起始 Flash 或 RAM 地址。此地址应是字边界。 **字节数**：待比较的字节数；计数值应当为 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 |
描述	此命令用于比较两个位置的内存内容。 **当源或目的地址包含从地址 0 起的前 512 个字节时，比较结果可能不正确。前 512 个字节重新映射到 boot ROM**
示例	“M 8192 268468224 4<CR><LF>”将 RAM 地址 0x1000 8000 开始的 4 个字节与 Flash 地址 0x2000 开始的 4 个字节进行比较。

21.13.14 读 UID

表 351. 读 UID 命令

命令	N
输入	无
返回码	CMD_SUCCESS 后跟唯一序列号的 4 个 32 位字 （ASCII 格式）。先发送低位地址的字。
描述	该命令用于读唯一的 ID。

21.13.15 ISP 返回代码

表 352. ISP 返回代码总览

返回码	助记符	描述
0	CMD_SUCCESS	命令执行成功。只有当主机给出的命令成功执行完成时，才由 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。
3	DST_ADDR_ERROR	目标地址不在正确的边界。
4	SRC_ADDR_NOT_MAPPED	源地址没有映射在内存映像图中。适用时应考虑计数值。
5	DST_ADDR_NOT_MAPPED	目标地址没有映射在内存映像图中。适用时应考虑计数值。
6	COUNT_ERROR	计数的字节不是 4 的倍数或允许值。
7	INVALID_SECTOR	扇区号无效或结束扇区号比起始扇区号大。
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没被执行。
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。
14	ADDR_NOT_MAPPED	地址没有映射在内存映像图中。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	无效的停止位设置。
19	CODE_READ_PROTECTION_ENABLED	代码读取保护使能。

21.14 IAP 命令

对于应用编程， IAP 例程应通过寄存器 r0 中的字指针来调用，此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。用户可以把寄存器 r0 和 r1 中的指针赋予相同的值，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。参数传递见图 62。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，传送到“将 RAM 内容复制到 Flash”命令。结果的最大数目为 4，由 “读 UID”命令返回。当接收到未定义的命令时，命令处理程序发送状态码 INVALID_COMMAND。IAP 程序是 Thumb 代码，驻留在地址 0x1FFF 1FF0 处。

IAP 函数可使用 C 语言通过下列方法调用。

定义 IAP 位置入口点。由于 IAP 地址的第 0 位被置位，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，你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），最多 4 个参数可在寄存器 r0、r1、r2 和 r3 中分别传递。其它参数在堆栈上传递。最多 4 个参数可在寄存器 r0、r1、r2 和 r3 中分别返回。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降低了这种风险。

写操作和擦除操作期间，无法访问闪存。会导致闪存写 / 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此空间。

表 353. IAP 命令总览

IAP 命令	命令代码	说明见：
准备写操作扇区	50（十进制）	表 354
将 RAM 内容复制到 Flash	51（十进制）	表 355
擦除扇区	52（十进制）	表 356
空白检查扇区	53（十进制）	表 357
读取部件 ID	54（十进制）	表 358
读 Boot 代码版本	55（十进制）	表 359
比较	56（十进制）	表 360
重新调用 ISP	57（十进制）	表 361
读 UID	58（十进制）	表 362
擦除页	59（十进制）	表 363
EEPROM 写	61（十进制）	表 364
EEPROM 读	62（十进制）	表 365

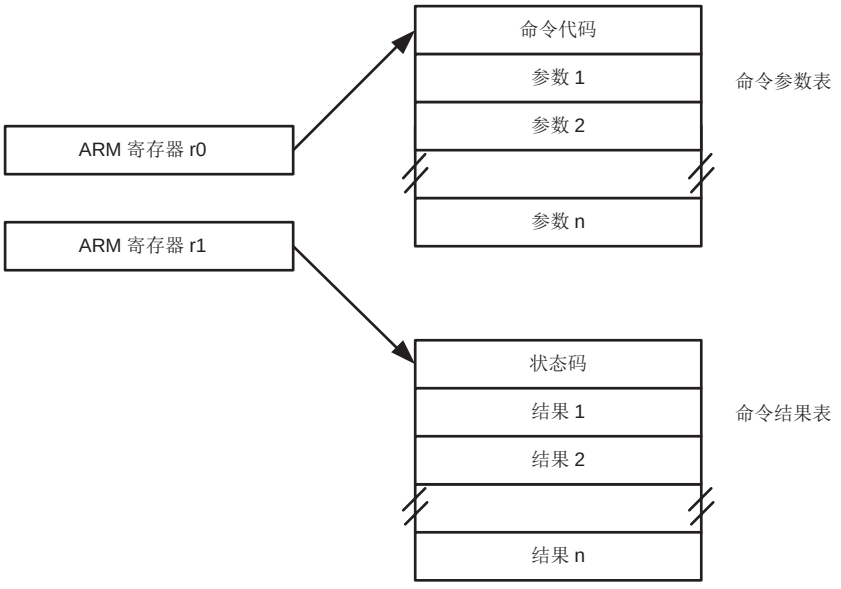


图 62. IAP 参数传递

21.14.1 准备写操作扇区

此命令使闪存写 / 擦除操作分为两步。

表 354. 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 （应当大于或等于起始扇区号）。
返回码	CMD_SUCCESS | BUSY | INVALID_SECTOR
结果	无
描述	该命令必须在执行 “将 RAM 内容复制到 Flash”或 “擦除扇区”命令之前执行。“将 RAM 内容复制到 Flash”或“擦除扇区”命令的成功执行会使相关的扇区再次被保护。不能使用该命令准备引导扇区。要准备单一的扇区，使用相同的 “起始”和 “结束”扇区号。

21.14.2 将 RAM 内容复制到 Flash

关于写入闪存过程的限制，请参见第 21.13.7 节。

表 355. IAP 将 RAM 内容复制到 Flash 命令

命令	将 RAM 内容复制到 Flash
输入	**命令代码：51 （十进制）** **参数 0(DST)：**要写入数据字节的目标 Flash 地址。该地址应当为 256 字节边界。 **参数 1(SRC)：**从中读取数据字节的源 RAM 地址。此地址应是字边界。 **参数 2：**写入的字节数。应为 256 | 512 | 1024 | 4096。 **参数 3：**系统时钟频率 (CCLK) （单位：kHz）。
返回码	CMD_SUCCESS | SRC_ADDR_ERROR （地址不以字为边界） | DST_ADDR_ERROR （地址不在正确的边界） | 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
描述	此命令用于闪存编程。受影响的扇区应先通过调用 “准备写操作扇区”命令准备好。一旦复制命令成功执行，受影响的扇区将自动被重新保护。不能使用该命令写入引导扇区。有关可写入的字节数，另请参见第 21.6 节。

21.14.3 擦除扇区

表 356. 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 （应当大于或等于起始扇区号）。 **参数 2：**系统时钟频率 (CCLK) （单位：kHz）。
返回码	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
结果	无
描述	此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。要擦除单一的扇区，使用相同的 “起始” 和 “结束” 扇区号。

21.14.4 空白检查扇区

表 357. IAP 扇区查空命令

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 （应当大于或等于起始扇区号）。
返回码	CMD_SUCCESS | BUSY | SECTOR_NOT_BLANK | INVALID_SECTOR
结果	**结果 0：**状态代码为 SECTOR_NOT_BLANK 时第一个非空字位置的偏移量。 **结果 1：**非空字位置的内容。
描述	此命令用于空白检查片内闪存的一个或多个扇区。要空白检查单一的扇区，使用相同的 “起始” 和 “结束” 扇区号。

21.14.5 读器件标识号

表 358. IAP 读器件标识命令

命令	读器件标识号
输入	**命令代码：54 （十进制）** **参数：**无
返回码	CMD_SUCCESS
结果	**结果 0：**器件标识号。
描述	此命令用于读取部件识别号。

21.14.6 读 Boot 代码版本号

表 359. IAP 读 Boot 代码版本号命令

命令	读 Boot 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。读为 < 字节 1 （主） >.< 字节 0 （次） >
描述	此命令用于读取引导代码版本号。

21.14.7 比较 < 地址 1> < 地址 2> < 字节数 >

表 360. IAP 比较命令

命令	比较
输入	**命令代码：56 （十进制）** **参数 0(DST)：**要比较的数据字节的起始 Flash 或 RAM 地址。此地址应是字边界。 **参数 1(SRC)：**要比较的数据字节的起始 Flash 或 RAM 地址。此地址应是字边界。 **参数 2：**待比较的字节数；计数值应当为 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。
描述	此命令用于比较两个位置的内存内容。 **当源或目标地址包含从地址 0 开始的前 512 字节中的任意一个地址时，比较的结果可能不正确。因为前 512 字节是可以重新映射到 RAM 中的。**

21.14.8 重新调用 ISP

表 361. 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。**
描述	该命令用来调用 ISP 模式中的启动引导程序。它会映射引导向量，设定 PCLK = CCLK，配置 UART 引脚 RXD 和 TXD，复位计数器 / 定时器 CT32B1 和 FDR 寄存器 （见表 221）。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。

21.14.9 读 UID

表 362. IAP 读 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。
描述	该命令用于读唯一的 ID。

21.14.10 擦除页

表 363. 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 （应大于或等于起始页） **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
描述	该命令用于擦除片内 Flash 存储器的一页或多页。要擦除单页，可使用相同的“起始”和“结束”页号。

21.14.11 写 EEPROM

表 364. IAP 写 EEPROM 命令

命令	比较
输入	**命令代码：61 （十进制）**
返回码	CMD_SUCCESS | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED
结果	**参数 0：**EEPROM 地址。 **参数 1：**RAM 地址。 **参数 2：**写入的字节数。 **参数 3：**系统时钟频率 (CCLK) （单位：kHz）。
描述	从 RAM 地址复制数据到 EEPROM 地址。 **注：**EEPROM 存储器的前 64 字节保留，且不能写入到其中。

21.14.12 读 EEPROM

表 365. IAP 读 EEPROM 命令

命令	比较
输入	**命令代码：62 （十进制）**
返回码	CMD_SUCCESS | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED
结果	**参数 0：**EEPROM 地址。 **参数 1：**RAM 地址。 **参数 2：**读出的字节数。 **参数 3：**系统时钟频率 (CCLK) （单位：kHz）。
描述	从 EEPROM 地址复制数据到 RAM 地址。

21.14.13 IAP 状态代码

表 366. IAP 状态代码小结

状态码	助记符	描述
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。
3	DST_ADDR_ERROR	目标地址不在正确的边界。
4	SRC_ADDR_NOT_MAPPED	源地址没有映射在内存映像图中。适用时应考虑计数值。
5	DST_ADDR_NOT_MAPPED	目标地址没有映射在内存映像图中。适用时应考虑计数值。
6	COUNT_ERROR	计数的字节不是 4 的倍数或允许值。
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没被执行。
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。

21.15 调试注意事项

21.15.1 比较闪存镜像

根据使用的调试器和 IDE 调试设置，调试器连接时可见的存储器可能是 Boot ROM、内部 SRAM 或闪存。为了帮助确定当前调试环境中存在的存储器，可检查闪存地址 0x0000 0004 处包含的值。该地址包含 ARM Cortex-M3 向量表中代码的入口点，分别为 Boot ROM、内部 SRAM 或闪存的底部。

表 367. 调试模式中的存储器映射

存储器映射模式	存储器起始地址在 0x0000 0004 处可见
启动引导程序模式	0x1FFF 0000
用户闪存模式	0x0000 0000
用户 SRAM 模式	0x1000 0000

21.15.2 串行线调试 (SWD) Flash 编程接口

调试工具可将一部分 Flash 映像写入 RAM，然后按照正确的偏移地址重复调用 IAP 命令“将 RAM 内容复制到 Flash”。

21.16 闪存控制器寄存器

表 368. 寄存器简介：FMC （基址 0x4003 C000）

名称	访问类型	地址偏移	描述	复位值	参考
FLASHCFG	R/W	0x010	闪存访问时间配置寄存器	-	表 369
FMSSTART	R/W	0x020	签名起始地址寄存器	0	表 370
FMSSTOP	R/W	0x024	签名停止地址寄存器	0	表 371
FMSW0	R	0x02C	字 0 [31:0]	-	表 372
FMSW1	R	0x030	字 1 [63:32]	-	表 373
FMSW2	R	0x034	字 2 [95:64]	-	表 374
FMSW3	R	0x038	字 3 [127:96]	-	表 375
FMSTAT	R	0xFE0	签名生成状态寄存器	0	第 21.16.5 节
FMSTATCLR	W	0xFE8	签名生成状态清除寄存器	-	第 21.16.6 节

21.16.1 闪存访问寄存器

根据系统时钟频率，通过在地址 0x4003 C010 写入 FLASHCFG 寄存器，为访问闪存配置不同的访问时间。

注：此寄存器设置不当会导致 LPC1315/16/17/45/46/47 闪存无法正常工作。

表 369. 闪存配置寄存器 （FLASHCFG，地址 0x4003 C010）位描述

位	符号	值	描述	复位值
1:0	FLASHTIM		闪存访问时间。 FLASHTIM +1 等于用于闪存访问的系统时钟数。	0x2
		0x1	1 个系统时钟的闪存访问时间 （对于高达 20 MHz 的系统时钟频率）。	
		0x2	2 个系统时钟的闪存访问时间 （对于高达 40 MHz 的系统时钟频率）。	
		0x3	3 个系统时钟的闪存访问时间 （对于高达 50 MHz 的系统时钟频率）。	
		0x4	保留。	
31:2	-	-	保留。**用户软件不得更改这些位的值。位 31:2 必须完全按照读取的值写回**。	-

21.16.2 闪存签名生成

闪存模块包含内置的签名生成器。此生成器可从一系列闪存中生成 128 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对齐，例如，128 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断言等待状态，直至签名生成完成。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。发起签名生成的代码也应放在闪存外面。

21.16.3 签名生成地址和控制寄存器

这些寄存器控制自动签名生成，可为闪存内容的任何部分生成签名。用于生成签名的地址范围通过向签名起始地址寄存器 (FMSSTART) 写入起始地址，以及向签名停止地址寄存器 (FMSSTOP) 写入停止地址来定义。起始地址和停止地址必须与 128 位边界对齐，而且可通过将字节地址除以 16 得到。

签名生成通过设置 FMSSTOP 寄存器中的 SIG_START 位开始。在单一写操作中，SIG_START 位的设置通常与签名停止地址相结合。

表 370 和表 371 分别显示了 FMSSTART 和 FMSSTOP 寄存器中的位分配。

表 370. 闪存模块签名起始寄存器 (FMSSTART - 0x4003 C020) 位描述

位	符号	描述	复位值
16:0	START	签名生成起始地址 （对应于 AHB 字节地址位 [20:4]）。	0
31:17	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

表 371. 闪存模块签名停止寄存器 (FMSSTOP - 0x4003 C024) 位描述

位	符号	值	描述	复位值
16:0	STOP		除以16的BIST停止地址（对应于AHB字节地址[20:4]）。	0
17	SIG_START		签名生成起始控制位。	0
		0	签名生成停止。	
		1	发起签名生成。	
31:18	-		保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

21.16.4 签名生成结果寄存器

签名生成结果寄存器返回内置生成器生成的闪存签名。此 128 位签名由四个寄存器 FMSW0、 FMSW1、 FMSW2 和 FMSW3 反映。

生成的闪存签名可用于验证闪存内容。可将生成的签名与预期签名相比，从而节省时间和代码空间。生成签名的方法在第 21.16.2 节中说明。

表 375 分别显示了寄存器 FMSW0、 FMSW1、 FMSW2 和 FMSW3 的位分配。

表 372. FMSW0 寄存器 （FMSW0，地址：0x4003 C02C） 位描述

位	符号	描述	复位值
31:0	SW0[31:0]	128 位签名的字 0 （位 31 到 0）。	-

表 373. FMSW1 寄存器 （FMSW1，地址：0x4003 C030） 位描述

位	符号	描述	复位值
31:0	SW1[63:32]	128 位签名的字 1 （位 63 到 32）。	-

表 374. FMSW2 寄存器 （FMSW2，地址：0x4003 C034） 位描述

位	符号	描述	复位值
31:0	SW2[95:64]	128 位签名的字 2 （位 95 到 64）。	-

表 375. FMSW3 寄存器 （FMSW3，地址：0x4003 40C8） 位描述

位	符号	描述	复位值
31:0	SW3[127:96]	128 位签名的字 3 （位 127 到 96）。	-

21.16.5 闪存模块状态寄存器

FMSTAT 只读寄存器提供了一种确定签名生成是否完成的方法。签名生成的完成情况可通过轮询 FMSTAT 寄存器中的 SIG_DONE 位来检查。在开始签名生成操作前， SIG_DONE 应通过 FMSTATCLR 寄存器清除，否则其状态可能表示先前操作已完成。

表 376. 闪存模块状态寄存器 (FMSTAT - 0x4003 CFE0) 位描述

位	符号	描述	复位值
1:0	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
2	SIG_DONE	其值为 1 时，先前启动的签名生成已完成。有关清除此标志的信息，参见 FMSTATCLR 寄存器说明。	0
31:3	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

21.16.6 闪存模块状态清除寄存器

FMSTATCLR 寄存器用于清除签名生成完成标志。

表 377. 闪存模块状态清除寄存器 (FMSTATCLR - 0x0x4003 CFE8) 位描述

位	符号	描述	复位值
1:0	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用
2	SIG_DONE_CLR	向此位写入 1 将清除 FMSTAT 寄存器中的签名生成完成标志 (SIG_DONE)。	0
31:3	-	保留，用户软件不应向保留位写入 1。未定义从保留位读取的值。	不适用

21.16.7 签名生成的算法和步骤

签名生成

可为闪存内容的任何部分生成签名。用于生成签名的地址范围通过向 FMSSTART 寄存器写入起始地址，以及向 FMSSTOP 寄存器写入停止地址来定义。

签名生成通过向 FMSSTOP 寄存器中的 SIG_START 位写入 “1” 开始。签名生成的开始通常与停止地址的定义相结合，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。为签名生成而读取闪存时，使用了自定时读取机制且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：

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，单位为 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 到 FMSW3 中读取 128 位的签名。此 128 位签名反映从闪存中读取的正确数据。此 128 位签名反映闪存奇偶校验位并检查位值。

内容验证

从寄存器 FMSW0 到 FMSW3 读取的签名必须与参考签名相等。图 63 给出了推算参考签名的算法。

```
int128 signature = 0
int128 nextSignature
FOR address = flashpage 0 TO address = flashpage max
{
        FOR i = 0 TO 126 {
                nextSignature[i] = flashword[i] XOR signature[i+1] }
        nextSignature[127] = flashword[127] XOR signature[0] XOR signature[2]
                XOR signature[27] XOR signature[29]
        signature = nextSignature
}
return signature
```

图 63. 生成 128 位签名的算法

22.1 本章导读

所有 LPC1315/16/17/45/46/47 器件的调试功能都相同。

22.2 特性

- 支持 ARM 串行调试接口模式。
- 跟踪端口提供 CPU 指令跟踪功能。通过串行线浏览器输出。
- 可直接对所有存储器、寄存器和外设进行调试。
- 调试阶段不需要目标资源。
- 4 个断点。四个指令断点，也可用于重新映射代码补丁的指令地址。两个数据比较器，可用于重新映射文字值补丁的地址。
- 两个数据观察点，也可用作触发器。
- 支持 JTAG 边界扫描。
- “仪表跟踪宏单元”允许额外的软件控制跟踪。

22.3 简介

ARM Cortex-M3 集成了调试功能。支持串行调试接口功能。ARM Cortex-M3 的配置可支持最多四个断点和两个观察点。

22.4 描述

LPC1315/16/17/45/46/47 使用串行调试接口模式进行调试。可支持边界扫描。

可使用串行线输出进行跟踪。使用串行线输出时，可跟踪的数据较少，但不使用应用相关引脚。请注意，ARM Cortex-M3 可用的跟踪功能与先前基于 ARM7 的设备可用的跟踪功能有很大不同。

22.5 引脚说明

下表显示了与调试有关的各种引脚功能。其中有些功能与其它功能共享引脚，因此不能同时使用。使用串行线输出的跟踪的带宽有限。

表 378. 串行调试接口引脚说明

引脚名称	类型	描述
SWCLK	输入	**串行接口时钟。**此引脚在串行调试接口模式下是 SWD 调试逻辑的时钟 (SWD)。此引脚从内部上拉。
SWDIO	输入/输出	**串行调试接口数据输入 / 输出。**外部调试工具使用 SWDIO 引脚与 LPC1315/16/17/45/46/47 进行通信并控制它。此引脚从内部上拉。
SWO	输出	**串行线输出。**SWO 引脚也可提供 ITM 和 / 或 ETM 数据，供外部调试工具评估。

表 379. JTAG 边界扫描引脚描述

引脚名称	类型	描述
TCK	输入	**JTAG 测试时钟。**$\overline{\text{RESET}}$ 引脚位于低电平时，此引脚是 JTAG 边界扫描的时钟。
TMS	输入	**JTAG 测试模式选择。** TMS 引脚选择 TAP 状态机中的下一状态。$\overline{\text{RESET}}$ 引脚位于低电平时，此引脚包含一个内部上拉，用于 JTAG 边界扫描。
TDI	输入	**JTAG 测试数据输入。**这是移位寄存器的串行数据输入。$\overline{\text{RESET}}$ 引脚位于低电平时，此引脚包含一个内部上拉，用于 JTAG 边界扫描。
TDO	输出	**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。$\overline{\text{RESET}}$ 引脚位于低电平时，此引脚用于 JTAG 边界扫描。
$\overline{\text{TRST}}$	输入	**JTAG 测试复位。**$\overline{\text{TRST}}$ 引脚可用于复位调试逻辑内的测试逻辑。$\overline{\text{RESET}}$ 引脚位于低电平时，此引脚包含一个内部上拉，用于 JTAG 边界扫描。

22.6 功能说明

22.6.1 调试限制

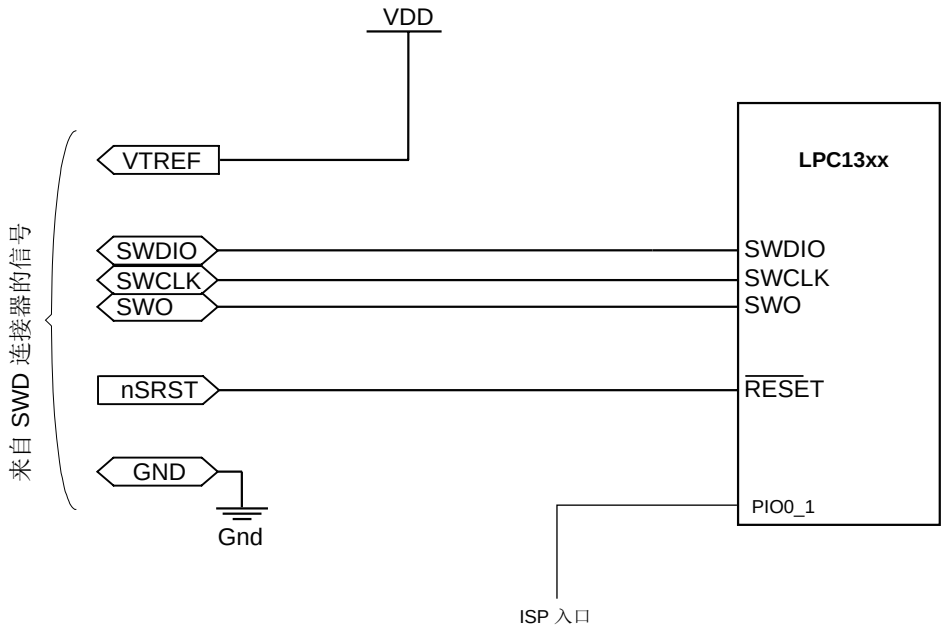
注意事项：由于 ARM Cortex-M3 集成的限制， LPC1315/16/17/45/46/47 无法通过常规方式从深度睡眠模式中唤醒。建议在调试期间不要使用此模式。

另一个问题是调试模式会改变低功耗模式在 ARM Cortex-M3 CPU 内部的工作方式，并波及到整个系统。这些差异意味着在调试过程中不应该进行功率测量，其结果会高于正常运行的情况。

在调试会话过程中，当 CPU 停止时系统节拍定时器将会自动停止，其他外设不受影响。

22.6.2 SWD 的调试连接

就调试而言，需要访问 ISP 入口引脚 PIO0_1。此引脚可用于使器件从将禁用 SWD 端口的配置中恢复，如不当的 PLL 配置或重新配置 SWD 引脚作为 ADC 输入、复位后进入深度掉电模式等。此引脚可用于 GPIO 等其他功能，但在上电或复位时不应保持低电平。



SWD 连接器上的 VTREF 引脚可使调试连接器匹配目标电压。

图 64. 将 SWD 引脚连接到标准的 SWD 连接器

22.6.3 边界扫描

RESET 引脚在用于 JTAG 边界扫描的检测 TAP 控制器（RESET = 低电平）和 ARM SWD 调试端口 TAP 控制器 （RESET = 高电平）之间选择。 LPC1315/16/17/45/46/47 复位时，禁用 ARM SWD 调试端口。 TRST 引脚低电平可复位检测 TAP 控制器。

注：POR 之后 250 µs 内不应启动边界扫描操作。边界扫描后必须将检测 TAP 复位并将其置于 TLR 或 RTO 状态。边界扫描不受代码读保护的影响。

注：POR、BOD 复位或 TRST 引脚上的低电平会将检测 TAP 控制器置于测试 - 逻辑复位状态。 RESET = 高时的第一个 TCK 时钟会将检测 TAP 置于运行 - 测试空闲模式。

23.1 缩略词

表 380. 缩略词

首字母缩略词	描述
A/D	模拟到数字
ADC	模数转换器
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
JTAG	联合测试行动小组
PLL	锁相环
RC	电阻 - 电容
SPI	串行外设接口
SSI	串行同步接口
SSP	同步串口
TAP	测试访问端口
UART	通用异步收发器
USART	通用同步 / 异步收发器

23.2 法律信息

23.2.1 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。

23.2.2 免责声明

有限担保和责任 — 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。

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，军事、飞机、太空或生命保障设备，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会导致人员受伤、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 — 本文件所述任何产品的应用仅限于例证目的。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。客户须自行负责检验恩智浦半导体的产品是否适用于其规划的应用和产品，以及是否适用于其第三方客户的规划应用和使用。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国家主管部门批准。

23.2.3 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。

23.3 表

23.4 图

23.5 内容

第 1 章：LPC1315/16/17/45/46/47 简介信息

第 2 章：LPC1315/16/17/45/46/47 存储器映射

第 3 章：LPC1315/16/17/45/46/47 系统控制模块

第 4 章：LPC1315/16/17/45/46/47 电源管理单元 (PMU)

第 5 章：LPC1315/16/17/45/46/47 功率配置

第 6 章：LPC1315/16/17/45/46/47 NVIC

第 7 章：LPC1315/16/17/45/46/47 I/O 配置

第 8 章：LPC1315/16/17/45/46/47 引脚配置

第 9 章：LPC1315/16/17/45/46/47 GPIO

第 10 章：LPC1345/46/47 USB2.0 设备控制器

第 11 章：LPC1345/46/47 USB 片上驱动器

第 12 章：LPC1315/16/17/45/46/47 USART

第 13 章：LPC1315/16/17/45/46/47 SSP/SPI

第 14 章：LPC1315/16/17/45/46/47 I2C 总线控制器

第 15 章：LPC1315/16/17/45/46/47 16 位计数器 / 定时器 CT16B0/1

第 16 章：LPC1315/16/17/45/46/47 32 位计数器 / 定时器 CT32B0/1

第 17 章：LPC1315/16/17/45/46/47 窗口化看门狗定时器 (WWDT)

第 18 章：LPC1315/16/17/45/46/47 系统节拍定时器

第 19 章：LPC1315/16/17/45/46/47 重复中断定时器 （RI 定时器）

第 20 章：LPC1315/16/17/45/46/47 ADC

第 21 章：LPC1315/16/17/45/46/47 闪存 /EEPRPOM 编程固件

注意：关于本文及相关产品的重要说明详见“法律信息”一节。

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

发布日期：2012 年 2 月 17 日
文档号： UM10524