UM10527

LPC11Axx User manual

修订版：1 — 2012 年 3 月 22 日 用户手册

文档信息

项目	内容
关键字	LPC11Axx
摘要	LPC11Axx 用户手册

修订记录

版本	日期	描述
1	20120322	初版 LPC11Axx 用户手册

联系信息

有关详细信息，请访问：**http://www.nxp.com**

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

1.1 简介

LPC11Axx 是基于 ARM Cortex-M0 的低成本 32 位 MCU 系列产品，专用于 8 位或 16 位微控制器应用。它性能高、功耗低，支持简单指令集和内存寻址，与现有的 8 位或 16 位架构相比，代码尺寸更小。

LPC11Axx 以高达 50 MHz 的 CPU 频率工作。

模拟 / 混合信号子系统可通过软件从互连的数字外设和模拟外设进行配置。

LPC11Axx 上的数字外设包括高达 32 kB 的闪存、4 kB EEPROM 数据存储器、8 kB SRAM 数据存储器、一个超快速模式 I^2C 总线接口、一个 RS-485/EIA-485 USART、两个 SSP 控制器、四个通用计数器 / 定时器以及最多 42 个通用 I/O 引脚。

模拟外设则包括一个 10 位 ADC、一个 10 位 DAC、一个模拟比较器、一个温度传感器、内部基准电压和 “欠压锁定” (UVLO) 保护功能。

1.2 特性

- 系统：
  - ARM Cortex-M0 处理器，工作频率高达 50 MHz。
  - ARM Cortex-M0 内置可嵌套中断向量控制器 (NVIC)。
  - 串行调试接口 (SWD)
  - JTAG 边界扫描。
  - 系统节拍定时器。
- 存储器：
  - 最高 32 kB 片内闪存程序存储器。
  - 最高 4 kB 片内 EEPROM 数据存储器；字节可擦除和可编程。
  - 最高 8 kB SRAM 数据存储器。
  - 16 kB Boot ROM。
  - 闪存的在系统编程 (ISP) 以及闪存和 EEPROM 的在应用编程 (IAP) 通过片内启动引导程序软件执行。
  - 其中包括基于 ROM 的 32 位整数除法以及 I^2C 总线驱动程序。
- 数字外设：
  - 多达 42 个通用 I/O (GPIO) 引脚，上拉 / 下拉电阻、中继模式和开漏模式可配置。
  - 最多 16 个引脚可配置数字输入干扰滤波器，用于消除宽度不超过 10 ns 的干扰，两个引脚可配置 50 ns 干扰滤波器。
  - GPIO 引脚可用作边沿和电平敏感中断源。
  - 一个引脚 (PIO0_21) 上的大电流源输出驱动器 (20 mA)。
  - 真开漏引脚 （PIO0_2 和 PIO0_3）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 16 个捕获输入和 14 个匹配输出。
  - 可编程窗口化看门狗定时器 (WWDT)，包含专用的内部低功率看门狗振荡器 (WDOsc)。

- 模拟外设：
  - 10 位 ADC，输入在 8 个引脚中多路复用。
  - 10 位 DAC，具有灵活的转换触发。
  - 高度灵活的模拟比较器，具有可编程基准电压。
  - 集成式温度传感器。
  - 内部基准电压。
  - 欠压锁定 (UVLO) 保护，防止电源电压低于 2.4 V。
- 串行接口：
  - USART，生成小数波特率，内部 FIFO，支持 RS-485/9 位模式以及同步模式。
  - 两个 SSP 控制器，搭载 FIFO 和多协议功能。支持高达 25 Mbit/s 的数据速率。
  - 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达到 1 Mbit/s）和监控模式。
- 时钟生成：
  - 晶体振荡器 (SysOsc)，工作频率范围为 1 MHz 至 25 MHz。
  - 12 MHz 内部 RC 振荡器 (IRC)，可精确到 1%，可选择性地用作系统时钟。
  - 内部低功率低频振荡器 (LFOsc)，具有可编程频率输出。
  - 外部系统时钟的时钟输入 （典型值为 25 MHz）。
  - IRC、外部时钟或 SysOsc 作为时钟源时， PLL 允许 CPU 以最大 CPU 速率运行。
  - 时钟输出功能，其中分频器可反映 SysOsc、 IRC、主时钟或 LFOsc。
- 功率控制：
  - 支持一个低功耗模式：ARM 睡眠模式。
  - 功率模型驻留在 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。
  - 使用任何中断从低功耗模式中唤醒处理器。
  - 上电复位 (POR)。
  - 掉电检测 (BOD)，具有两个针对中断的可编程阀值以及一个硬件控制复位跳变点。
  - POR 和 BOD 始终处于使能状态，用于快速 UVLO 保护，防止电源电压低于 2.4 V。
- 识别器件的唯一序列号。
- 3.3 V 单电源 （2.6 V 至 3.6 V）。
- 温度范围 –40 °C 至 +85 °C。
- 提供 LQFP48 封装，HVQFN33(7x7) 和 HVQFN33(5X5) 封装，以及 WLCSP20 超小封装结构。

1.3 订购信息

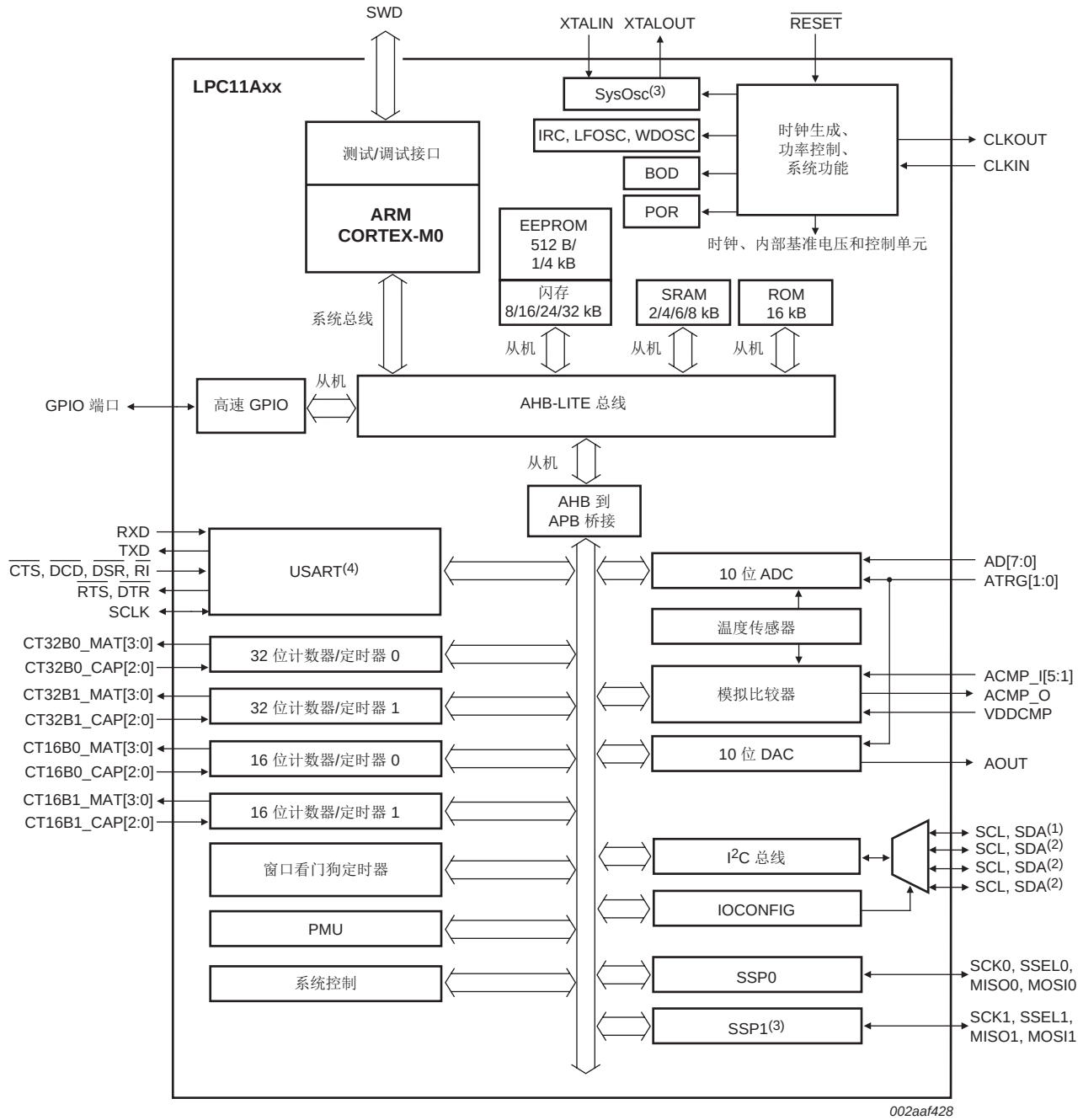
表 1. 订购信息

产品型号	封装		
	名称	描述	版本
LPC11A02UK	WLCSP20	晶圆级芯片尺寸封装； 20 凸块； 2.5 × 2.5 × 0.6 mm	-
LPC11A04UK	WLCSP20	晶圆级芯片尺寸封装； 20 凸块； 2.5 × 2.5 × 0.6 mm	-
LPC11A11FHN33/001	HVQFN33	HVQFN：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
LPC11A12FHN33/101	HVQFN33	HVQFN：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
LPC11A13FHI33/201	HVQFN33	HVQFN：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 5 × 5 × 0.85 mm	不适用
LPC11A14FHN33/301	HVQFN33	HVQFN：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
LPC11A12FBD48/101	LQFP48	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
LPC11A14FBD48/301	LQFP48	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

表 2. 订购选项

产品型号	闪存	SRAM	EEPROM	10 位 ADC 通道	10 位 DAC	温度传感器	模拟比较器	USART	SSP/SPI	I^2C	GPIO	封装
LPC11A02UK	16 kB	4 kB	2 kB	8	1	1	1	1	1	1	18	WLCSP20
LPC11A04UK	32 kB	8 kB	4 kB	8	1	1	1	1	1	1	18	WLCSP20
LPC11A11FHN33/001	8 kB	2 kB	512 B	8	1	1	1	1	2	1	28	HVQFN33
LPC11A12FHN33/101	16 kB	4 kB	1 kB	8	1	1	1	1	2	1	28	HVQFN33
LPC11A12FBD48/101	16 kB	4 kB	1 kB	8	1	1	1	1	2	1	42	LQFP48
LPC11A13FHI33/201	24 kB	6 kB	2 kB	8	1	1	1	1	2	1	28	HVQFN33
LPC11A14FHN33/301	32 kB	8 kB	4 kB	8	1	1	1	1	2	1	28	HVQFN33
LPC11A14FBD48/301	32 kB	8 kB	4 kB	8	1	1	1	1	2	1	42	LQFP48

1.4 功能框图



(1) 开漏引脚。

(2) 标准 I/O 引脚。

(3) 不适用于 WLCSP 封装。

(4) 调制解调器控制引脚不适用于 WLCSP 封装。

图 1. LPC11Axx 框图

1.5 可配置模拟/混合信号子系统

多个模拟/混合信号子系统可通过软件从互连的数字外设和模拟外设进行配置。参见图 2。

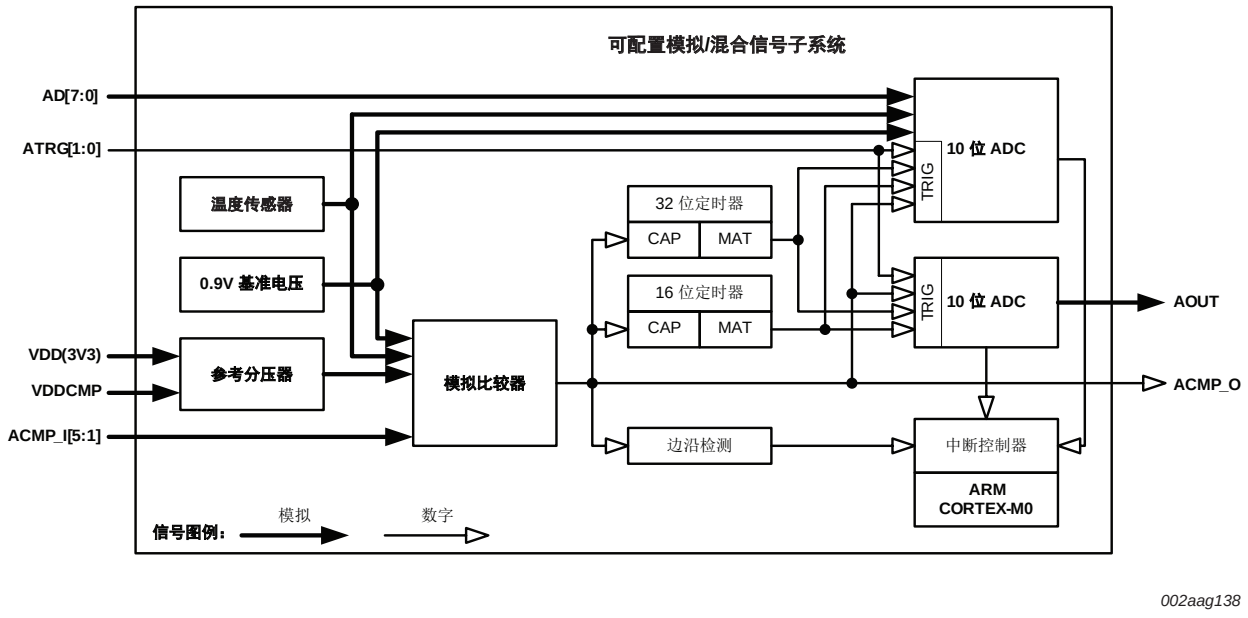


图 2. 可配置模拟/混合信号子系统框图

1.6 ARM Cortex-M0 处理器

第 25.2 节 “关于 Cortex-M0 处理器和内核外设”中详细介绍了 ARM Cortex-M0 处理器。对于 LPC11Axx， ARM Cortex-M0 处理器内核的配置如下所示：

- 系统选项：
  - 可嵌套中断向量控制器 (NVIC) 支持最多 32 个中断。
  - 系统节拍定时器。
- 调试选项：
  - JTAG 调试 / 工具接口。
  - 带两个观察点和四个断点的串行调试接口。

2.1 本章导读

有关 LPC11Axx 器件的存储器配置，请参见表 3。

表 3. LPC11Axx 存储器配置

器件	闪存	SRAM	EEPROM
LPC11A02	16 kB	4 kB	2 kB
LPC11A04	32 kB	8 kB	4 kB
LPC11A11/001	8 kB	2 kB	512 B
LPC11A12/101	16 kB	4 kB	1 kB
LPC11A13/201	24 kB	6 kB	2 kB
LPC11A14/301	32 kB	8 kB	4 kB

2.2 存储器映射

图 1 显示了 LPC11Axx 的存储器和外设地址空间。

AHB 外设区的大小为 2 Mb，可分配多达 128 个外设。在 LPC11Axx 上，GPIO 端口是唯一的 AHB 外设。APB 外设区的大小为 512 kB，可分配多达 32 个外设。每一个 AHB 或 APB 外设空间的大小均为 16 kB。

所有 APB 寄存器地址无论规格大小，均为 32 位字对齐。所有对 APB 寄存器的访问都被视为由外设进行的字访问。无法分别写入 APB 寄存器的单独字节或半字。

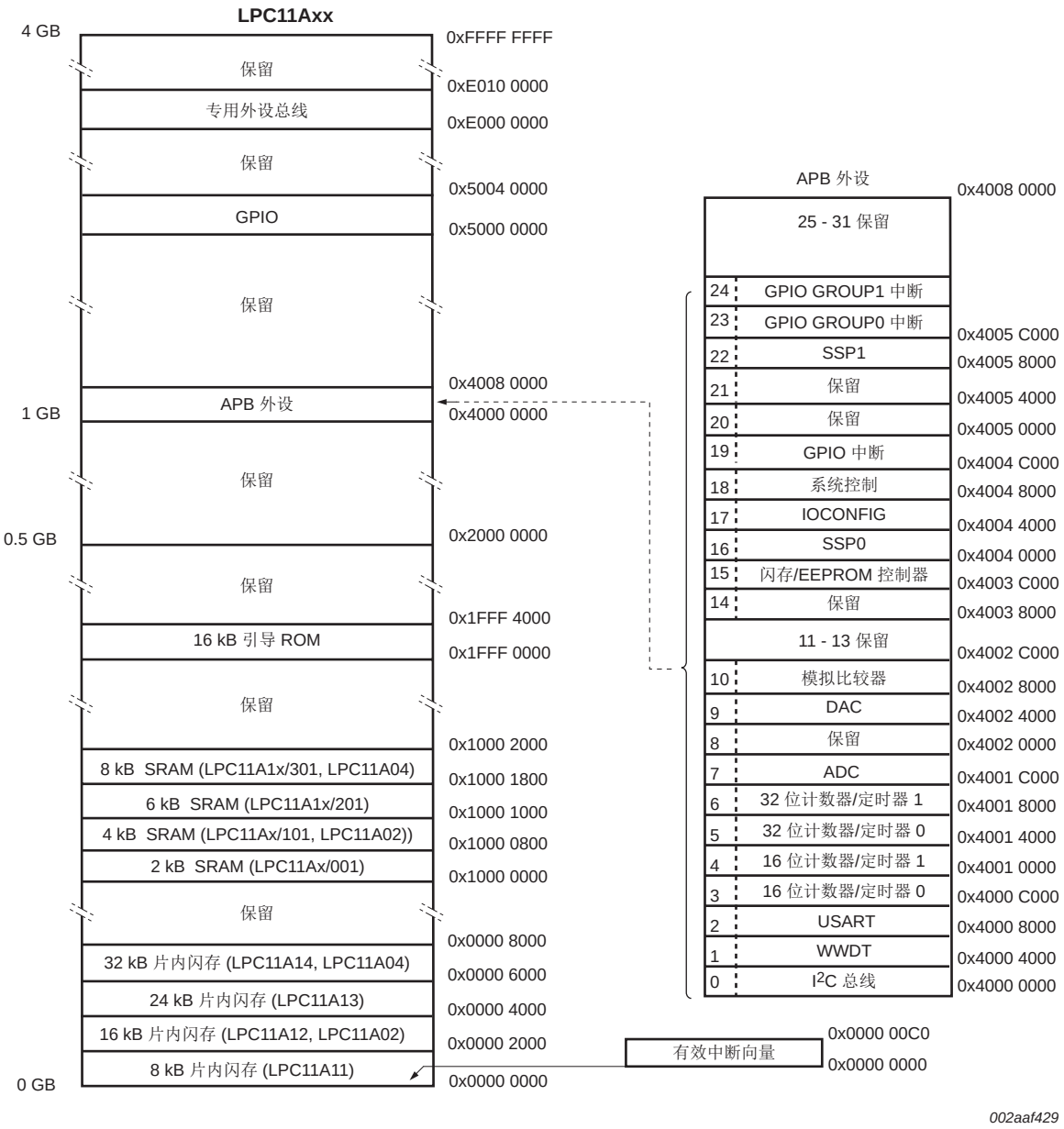


图 3. LPC11Axx 存储器映射

3.1 本章导读

所有 LPC11Axx 器件的系统控制模块都相同。

SPI1 接口不适用于 WLCSP20 器件。

3.2 简介

系统配置模块可控制 LPC11Axx 的振荡器和时钟产生。该模块中还包含用于设置 AHB 访问优先级的寄存器和用于重映射闪存、 SRAM 和 ROM 存储区域的寄存器。

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
CLKIN	I	时钟输入引脚
CLKOUT	O	时钟输出引脚

3.4 时钟和电源控制

3.4.1 时钟

有关 LPC11Axx 时钟产生单元 (CGU) 的概述，参见图 4。

LPC11Axx 包括 4 个振荡器：1 个晶体振荡器 (XTAL)、1 个 12 MHz 高频振荡器 (IRC) 以及可编程低频振荡器和看门狗振荡器（LFOSC、WDOSC）。XTAL、IRC 以及 LFOSC 可用于特定应用中的多种用途。

根据 SYSPLLCLKSEL 寄存器的选择，PLL 的时钟源可以是晶体振荡器、IRC 或 CLKIN 引脚上的外部时钟。

在复位之后， LPC11Axx 将从 IRC 运行，除非通过软件进行切换。

SYSAHBCLKCTRL 寄存器可开启各种外设和存储器的系统时钟。 UART 和 SSP 接口以及看门狗和 SysTick 定时器具有独立的时钟分频器，以从主时钟获得外设时钟。

任何时钟都可在 CLKOUT 引脚上直接输出或由 2 和 255 间的系数分频。

术语 “PCLK”是外设时钟的缩略语，在后续各章中用来表示提供给 APB 外设的时钟。图 4 显示几种从外设角度而言可被称为 PCLK 的信号。

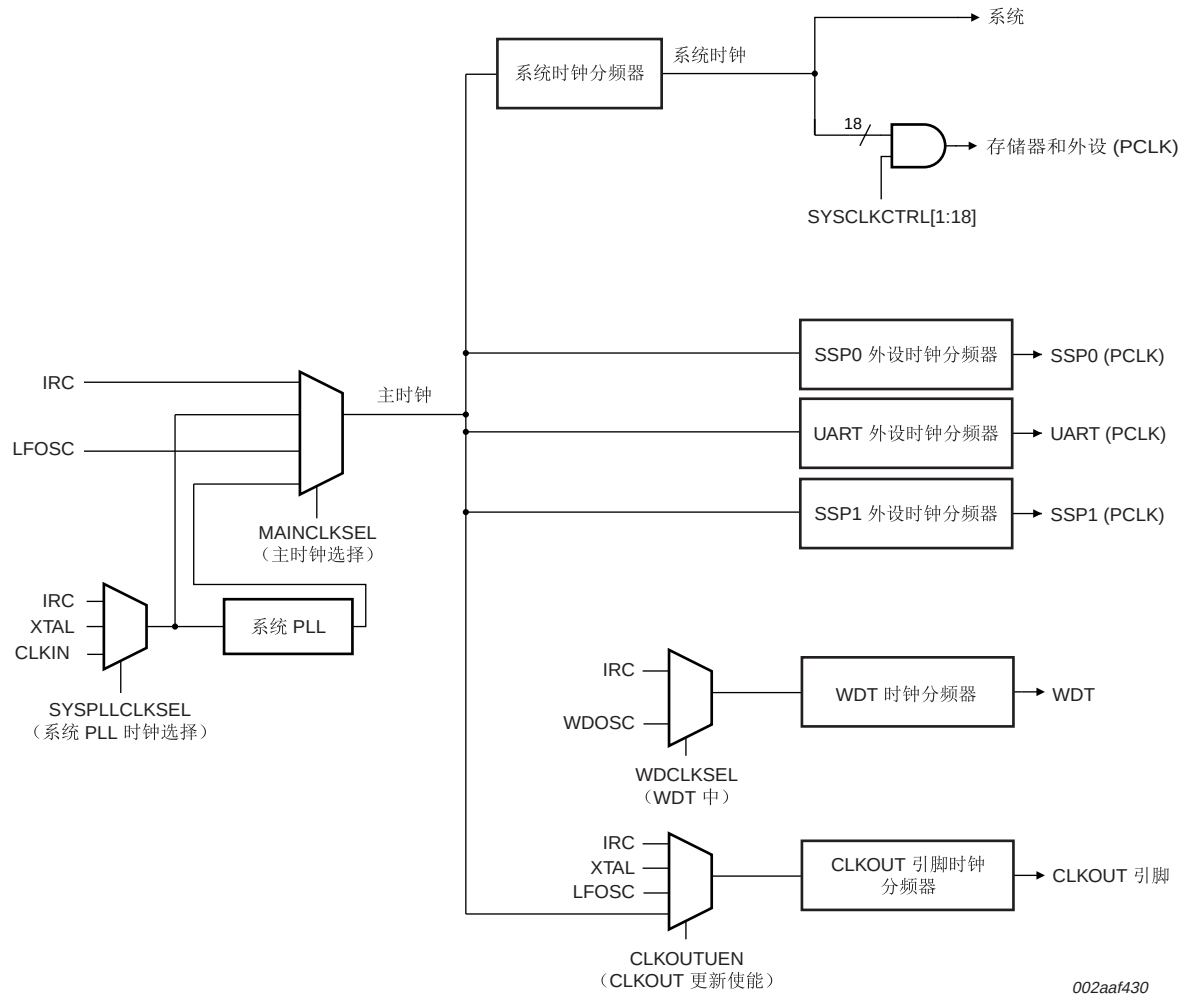


图 4. LPC11Axx CGU 框图

3.4.2 电源控制

LPC11Axx 支持多种电源控制功能。LPC11Axx 的选定模块的电源和时钟经过优化后可在芯片运行时最大限度降低功耗。

3.5 寄存器描述

所有 Syscon 寄存器都按字地址边界对齐。寄存器的详细信息都出现在每种功能的描述中。

有关可重新配置为系统设置一部分的闪存访问定时寄存器，请参见第 3.10 节。此寄存器不属于系统配置模块的一部分。

表 5 中未显示的全部地址偏移均保留，且不得向其写入。

表 5. 寄存器概述：系统控制模块 （基址 0x4004 8000）

名称	访问类型	偏移	描述	复位值	参考
SYSMEMREMAP	R/W	0x0	系统存储器重映射	0x02	表 6
PRESETCTRL	R/W	0x004	外设复位控制	0	表 7
SYSPLLCTRL	R/W	0x008	系统 PLL 控制	0	表 8
SYSPLLSTAT	R	0x00C	系统 PLL 状态	0	表 9
WDTOSCCTRL	R/W	0x024	看门狗振荡器控制	0x0A0	表 10
IRCCTRL	R/W	0x028	IRC 振荡器控制	0x0xx	表 11
LFOSCCTRL	R/W	0x02C	LF 振荡器控制	0x0A0	表 12
SYSRSTSTAT	R/W	0x030	系统复位状态寄存器	0	表 13
SYSPLLCLKSEL	R/W	0x040	系统 PLL 时钟源选择	0	表 14
SYSPLLCLKUEN	R/W	0x044	系统 PLL 时钟源更新使能	0	表 15
MAINCLKSEL	R/W	0x070	主时钟源选择	0	表 16
MAINCLKUEN	R/W	0x074	主时钟源更新使能	0	表 17
SYSAHBCLKDIV	R/W	0x078	系统时钟分频器	0x001	表 18
SYSAHBCLKCTRL	R/W	0x080	系统时钟控制	0x01F	表 19
SSP0CLKDIV	R/W	0x094	SSP0 时钟分频器	0	表 20
UARTCLKDIV	R/W	0x098	UART 时钟分频器	0	表 21
SSP1CLKDIV	R/W	0x09C	SSP1 时钟分频器	0	表 22
-	-	0x0A0 至 0x0DC	保留	-	-
CLKOUTSEL	R/W	0x0E0	CLKOUT 时钟源选择	0	表 23
CLKOUTUEN	R/W	0x0E4	CLKOUT 时钟源更新使能	0	表 24
CLKOUTDIV	R/W	0x0E8	CLKOUT 时钟分频器	0	表 25
PIOPORCAP0	R	0x100	POR 捕获 PIO 状态 0	由用户决定	表 26
PIOPORCAP1	R	0x104	POR 捕获 PIO 状态 1	由用户决定	表 27
BODR	R/W	0x150	掉电检测	0	表 28
SYSTCKCAL	R/W	0x158	系统节拍计数器校准	0x004	表 29
NMISRC	R/W	0x174	NMI 源控制	0	表 30
PINTSEL0	R/W	0x178	GPIO 引脚中断选择寄存器 0	0	表 31
PINTSEL1	R/W	0x17C	GPIO 引脚中断选择寄存器 1	0	表 31
PINTSEL2	R/W	0x180	GPIO 引脚中断选择寄存器 2	0	表 31
PINTSEL3	R/W	0x184	GPIO 引脚中断选择寄存器 3	0	表 31
PINTSEL4	R/W	0x188	GPIO 引脚中断选择寄存器 4	0	表 31
PINTSEL5	R/W	0x18C	GPIO 引脚中断选择寄存器 5	0	表 31
PINTSEL6	R/W	0x190	GPIO 引脚中断选择寄存器 6	0	表 31
PINTSEL7	R/W	0x194	GPIO 引脚中断选择寄存器 7	0	表 31
-	R/W	0x198 0x234	- 保留	-	-
PDRUNCFG	R/W	0x238	电源配置寄存器	0x0000 EDF0	表 32
DEVICE_ID	R	0x3F4	器件 ID	取决于零部件	表 33

3.5.1 系统存储器重映射寄存器

系统存储器重映射寄存器选择是从 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的底部 512 字节。

表 6. 系统存储器重映射寄存器 （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	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。	
		0x3	保留	
31:2	-	-	保留	-

3.5.2 外设复位控制寄存器

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。

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

位	符号	值	描述	复位值
0	SSP0_RST_N		SSP0 复位控制	1
		0	复位 SSP0 外设。	
		1	SSP0 复位已取消。	
1	I2C_RST_N		I2C 复位控制	1
		0	复位 I2C 外设。	
		1	I2C 复位已取消。	
2	SSP1_RST_N		SSP1 复位控制	1
		0	复位 SSP1 外设。	
		1	SSP1 复位已取消。	
3	-		保留	-
4	UART_RST_N		UART 复位控制	1
		0	复位 UART 外设。	
		1	UART 复位已取消。	
5	CT16B0_RST_N		CT16B0 复位控制	1
		0	复位 CT16B0 外设。	
		1	CT16B0 复位已取消。	
6	CT16B1_RST_N		CT16B1 复位控制	1
		0	复位 CT16B1 外设。	
		1	CT16B1 复位已取消。	
7	CT32B0_RST_N		CT32B0 复位控制	1
		0	复位 CT32B0 外设。	
		1	CT32B0 复位已取消。	

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

位	符号	值	描述	复位值
8	CT32B1_RST_N		CT32B1 复位控制	1
		0	复位 CT32B1 外设。	
		1	CT32B1 复位已取消。	
9	ACOMP_RST_N		模拟比较器复位控制	1
		0	复位模拟比较器外设。	
		1	模拟比较器复位已取消。	
10	DAC_RST_N		DAC 复位控制	1
		0	复位 DAC 外设。	
		1	DAC 复位已取消。	
11	ADC_RST_N		ADC 复位控制	1
		0	复位 ADC 外设。	
		1	ADC 复位已取消。	
31:12	-	-	保留	-

3.5.3 系统 PLL 控制寄存器

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。

PLL 工作模式由 DIRECT 和 BYPASS 位设置 （参见表 35）。

表 8. 系统 PLL 控制寄存器 （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 × 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

该寄存器是只读寄存器，提供 PLL 锁定状态 （参见第 3.9.1 节）。

表 9. 系统 PLL 状态寄存器 （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 看门狗振荡器控制寄存器

该寄存器可配置看门狗振荡器。该振荡器包含模拟和数字两个部分。模拟部分含有振荡器功能，可以生成模拟时钟 (F_{clkana})。FREQSEL 字段选择 0.5 MHz 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：
xxx_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 10. 看门狗振荡器控制寄存器 （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 × (1 + DIVSEL)) 00000：2 × (1 + DIVSEL) = 2 00001：2 × (1 + DIVSEL) = 4 至 11111：2 × (1 + DIVSEL) = 64	0
8:5	FREQSEL		选择看门狗振荡器模拟输出频率 (Fclkana)。	0
		0	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。	
		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	-

3.5.6 高频振荡器控制寄存器

该寄存器用于对片内 12 MHz 振荡器进行调整。该调整值为出厂预设值，在启动时由启动代码写入。

表 11. 高频振荡器控制寄存器 （IRCCTRL，地址 0x4004 8028）位描述

位	符号	描述	复位值
7:0	TRIM	调整值	制造过程中进行编程
31:9	-	保留	-

3.5.7 LF 振荡器控制寄存器

该寄存器可配置 LF 振荡器。 LF 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 MHz 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。

LF 振荡器输出时钟频率的计算如下：
xxx_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 12. LF 振荡器控制寄存器 （LFOSCCTRL，地址 0x4004 802C）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 × (1 + DIVSEL)) 00000：2 × (1 + DIVSEL) = 2 00001：2 × (1 + DIVSEL) = 4 至 11111：2 × (1 + DIVSEL) = 64	0
8:5	FREQSEL		选择看门狗振荡器模拟输出频率 (Fclkana)。	0
		0	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。	
		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	-

3.5.8 系统复位状态寄存器

如果取消 POR 信号后，其他复位信号（例如，外部 RESET 引脚）仍然有效，则其位将设置为“检测到”。写入 1 将清除复位。

表 13 中给定的复位值适用于 POR 复位。

表 13. 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		外部复位状态	0
		0	未检测到 $\overline{\text{RESET}}$ 事件	
		1	检测到 $\overline{\text{RESET}}$	
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	-

3.5.9 系统 PLL 时钟源选择寄存器

该寄存器为系统 PLL 选择时钟源。SYSPLLCLKUEN 寄存器（参见第 3.5.10 节）必须从低电平转换到高电平，才能使更新生效。

表 14. 系统 PLL 时钟源选择寄存器（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 (XTAL)	
		0x2	CLKIN 引脚	
		0x3	保留	
31:2	-	-	保留	-

3.5.10 系统 PLL 时钟源更新寄存器

写入 SYSPLLCLKSEL 寄存器后，该寄存器可以使用新输入时钟更新系统 PLL 的时钟源。必须先向 SYSPLLUEN 寄存器写入 0，再向 SYSPLLUEN 写入 1，才能使更新生效。

表 15. 系统 PLL 时钟源更新使能寄存器 （SYSPLLCLKUEN、地址 0x4004 8044）位描述

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

3.5.11 主时钟源选择寄存器

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

MAINCLKUEN 寄存器的位 0 （参见第 3.5.12 节）必须从 0 转换到 1，才能使更新生效。

表 16. 主时钟源选择寄存器 （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	LF 振荡器	
		0x3	PLL 输出	
31:2	-	-	保留	-

3.5.12 主时钟源更新使能寄存器

写到 MAINCLKSEL 寄存器后，该寄存器可以使用新输入时钟更新主时钟的时钟源。必须先向此寄存器的位 0 写入 0，再写入 1，才能使更新生效。

表 17. 主时钟源更新使能寄存器 （MAINCLKUEN、地址 0x4004 8074）位描述

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

3.5.13 系统时钟分频寄存器

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. 系统时钟分频寄存器 （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 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-M0、 SYSCON 模块和 PMU 提供时钟。无法禁用该时钟。

表 19. 系统时钟控制寄存器 （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-M0 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	RAM		使能 RAM 时钟。	1
		0	禁用	
		1	使能	
3	FLASHREG		使能闪存 /EEPROM 寄存器接口的时钟。	1
		0	已禁用	
		1	使能	
4	FLASHARRAY		使能闪存 /EEPROM 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 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. 系统时钟控制寄存器（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	UART		使能 UART 的时钟。请注意，使能 UART 时钟之前必须在 IOCON 模块中对 UART 引脚进行配置。	0
		0	禁用	
		1	使能	
13	ADC		使能 ADC 时钟。	0
		0	禁用	
		1	使能	
14	-		保留	0
15	WDT		使能 WDT 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		GPIO 引脚中断	0
		0	禁用	
		1	使能	
20	ACOMP		使能 ACOMP 的时钟。	0
		0	禁用	
		1	使能	
21	DAC		使能 DAC 的时钟。	0
		0	禁用	
		1	使能	
22	-		保留	-

表 19. 系统时钟控制寄存器（SYSAHBCLKCTRL、地址 0x4004 8080）位描述 *（续）*

位	符号	值	描述	复位值
23	P0INT		GPIO 端口 0 中断	0
		0	禁用	
		1	使能	
24	P1INT		GPIO 端口 1 中断	0
		0	禁用	
		1	使能	
31:25	-	-	保留	-

3.5.15 SSP0 时钟分频寄存器

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 时钟分频寄存器（SSP0CLKDIV，地址 0x4004 8094）位描述

位	符号	描述	复位值
7:0	DIV	SSP0_PCLK 时钟分频器值 0：禁用 SSP1_PCLK。 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

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

注：请注意，使能 UART 时钟之前必须在 IOCON 模块中对 UART 引脚进行配置。

表 21. UART 时钟分频寄存器（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

该寄存器可配置SSP1外设时钟SPI1_PCLK。可以通过将DIV字段设置为0来关闭SPI1_PCLK。

表 22. SSP1 时钟分频寄存器 （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。	0
		禁用 SPI1_PCLK。	
		1 分频。	
		...	
		255 分频。	
31:8	-	保留	-

3.5.18 CLKOUT 时钟源选择寄存器

该寄存器选择作为 CLKOUT 引脚输出的信号。可选择任意振荡器或主时钟。

CLKOUTUEN 寄存器的位 0 （参见第 3.5.19 节）必须从 0 转换到 1，才能使更新生效。

表 23. CLKOUT 时钟源选择寄存器 （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 (XTAL)	
		0x2	LF 振荡器	
		0x3	主时钟	
31:2	-	-	保留	0

3.5.19 CLKOUT 时钟源更新使能寄存器

写入 CLKOUTSEL 寄存器后，该寄存器可以使用新时钟更新 CLKOUT 引脚的时钟源。必须先向此寄存器的位 0 写入 0，再写入 1，才能使更新在 CLKOUT 引脚输入时生效。

表 24. CLKOUT 时钟源更新使能寄存器 （CLKOUTUEN、地址 0x4004 80E4）位描述

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

3.5.20 CLKOUT 时钟分频寄存器

该寄存器可确定 CLKOUT 引脚上信号的分频器值。

表 25. CLKOUT 时钟分频寄存器 （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	-

3.5.21 POR 捕获 PIO 状态寄存器 0

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 26. POR 捕获 PIO 状态寄存器 0 （PIOPORCAP0，地址 0x4004 8100）位描述

位	符号	描述	复位值
31:0	PIOSTAT	上电复位时 PIO0_31 到 PIO0_0 的状态	通过其他复位 NC

3.5.22 POR 捕获 PIO 状态寄存器 1

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 27. POR 捕获 PIO 状态寄存器 1 （PIOPORCAP1，地址 0x4004 8104）位描述

位	符号	描述	复位值
9:0	PIOSTAT	上电复位时 PIO1_9 到 PIO1_0 的状态	通过其他复位 NC
31:10	-	保留	-

3.5.23 掉电检测寄存器

掉电检测电路监控 VDD 电源上的电压。如果它低于某个阀值，则该电路会请求一个中断。BOD寄存器选择中断的阀值。表28中显示的阀值为典型值（参见《LPC11Axx数据手册》）。

使用与表 37 中调出的 BOD 的 IRQ 编号相对应的位号， BOD 中断可在第 25.5.2.2 节 “中断设置 - 使能寄存器” 中使能，在第 25.5.2.3 节 “中断清除 - 使能寄存器” 中禁用。

表 28. 掉电检测寄存器 （BODR，地址 0x4004 8150）位描述

位	符号	值	描述	复位值
1:0	-		保留	0
3:2	BODINTVAL		BOD 中断阀值	0x2
		0x2	2 级：能产生中断的阈值电压为 2.52 V ；能取消中断的阈值电压为 2.66 V。	
		0x3	3 级：能产生中断的阈值电压为 2.80 V ；能取消中断的阈值电压为 2.90 V。	
4	-		保留	0
5	-		保留	0
6	BODINT		如果 BOD 正请求一个中断，该位为 1。	0
31:7	-	-	保留	-

3.5.24 系统节拍计数器校准寄存器

表 29. 系统节拍定时器校准寄存器 （SYSTCKCAL、地址 0x4004 8158）位描述

位	符号	描述	复位值
25:0	CAL	系统节拍定时器校准值	0x4
31:26	-	保留	-

3.5.25 NMI 源选择寄存器

表 30. NMI 源选择寄存器 （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，请参见第 4.3 节。	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，如第 25.5.2 节中所述。

3.5.26 引脚中断选择寄存器

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。

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 37）。

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（表 39）。

表 31. 引脚中断选择寄存器 （PINTSEL0 至 7，地址 0x4004 8178 至 0x4004 8194）位描述

位	符号	描述	复位值
4:0	INTPIN	由 INTPORT 字段识别的端口内的引脚号。	0
5	INTPORT	选择端口： 0 = P0 引脚。 1 = P1 引脚。	
31:6	-	保留	-

3.5.27 电源配置寄存器

PDRUNCFG 寄存器控制各种模拟模块的电源。芯片运行时的任何时刻都可以写入该寄存器，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 32. 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	-		保留	1
4	ADC_PD		ADC 掉电	1
		1	掉电	
		0	上电	
5	XTAL_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	-		保留	1
9	-		保留。**在运行模式下进行正常操作时，该位必须设置为 0。**	0
10	-		保留	1
11	-		保留。在运行模式中，必须设置此位为 1。	1
12	-		保留。	0
13	LFOSC_PD		低频振荡器掉电	1
		1	掉电	
		0	上电	
14	DAC_PD		DAC 掉电	1
		1	掉电	
		0	上电	

表 32. 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
15	TS_PD		温度传感器掉电	1
		1	掉电	
		0	上电	
16	ACOMP_PD		模拟比较器掉电	1
		1	掉电	
		0	上电	
31:17	-	-	保留	-

3.5.28 器件 ID 寄存器

该器件 ID 寄存器为只读寄存器，包含每个 LPC11Axx 器件的器件 ID。该寄存器也可以通过 ISP/IAP 命令读取 （参见第 20.7 节和第 20.8 节）。

表 33. 器件 ID 寄存器 （DEVICE_ID、地址 0x4004 83F4）位描述

位	符号	描述	复位值
31:0	DEVICEID	LPC11Axx 器件的器件 ID 编号 LPC11A14FBD48/301 = 0x35A0 002B LPC11A14FHN33/301 = 0x35A0 002B	-

3.6 复位

LPC11Axx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 Cortex-M0 CPU 外部复位源（POR、BOD 复位、外部复位和看门狗复位）产生时，IRC 会启动。 IRC 启动 （上电后最多 6 µs）以后， IRC 就可以提供稳定的时钟输出。

1. 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。
2. 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 掉电检测

LPC11Axx 包括用于监控 V_{DD} 引脚上电压的掉电检测电路。如果该电压低于所选电平，则 BOD 会断言发送至 NVIC 的中断信号。可针对 NVIC 中断使能寄存器中的中断来使能该信号；否则软件可通过读取专用状态寄存器来监控信号。如果 V_{DD} 低于阀值，则可选择和使能独立阀值来复位芯片。参见表 28。

3.8 电源管理

第 3.4.2 节描述了 LPC11Axx 的低功耗模式。

CPU 时钟速率可根据需要通过更改时钟源、重新配置 PLL 值以及 / 或改变系统时钟分频器值来控制。这就根据应用要求在功率和处理速度之间实现了平衡。

实时电源控制允许关闭各个片内外设的时钟，这样就消除了特定应用中所不需要的任何外设的所有动态电源使用，从而实现功耗的微调。选定的外设（UART、SysTick 定时器、看门狗定时器）具有自身的时钟分频器，用于电源控制。

注：请注意，调试模式在任意低功耗模式下均不受支持。

表 34. LPC11Axx 电源和时钟控制寄存器

寄存器		电源 / 时钟控制功能	适用的模式
电源控制			
PDRUNCFG	表 32	控制模拟模块的电源（振荡器、PLL、ADC、DAC、模拟比较器、闪存和 BOD）。 **注：**该寄存器中的位 9 和位 12 必须为 0，以便确保在运行模式下的正确操作。	运行
时钟控制			
SYSAHBCLKCTRL	表 19	控制 ARM Cortex-M0 CPU、存储器和单独的 APB 外设的时钟。	运行
SYSAHBCLKDIV	表 18	禁用或配置系统时钟。	运行
UARTCLKDIV	表 21	禁用或配置 UART 外设时钟。	运行
CLKOUTDIV	表 25	禁用或配置 CLKOUT 引脚上的时钟。	运行

3.8.1 运行模式

运行模式下，ARM Cortex-M0 内核、存储器和外设由系统时钟计时。SYSAHBCLKCTRL 寄存器控制所运行的存储器和外设。系统时钟的频率可通过 SYSAHBCLKDIV 寄存器来选择。

除系统时钟外，选定的外设（UART、WDT 和 Systick 定时器）具有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来关闭。

各模块的电源（PLL、振荡器、ADC、DAC、模拟比较器、BOD 电路和闪存模块）的电源可以通过 PDRUNCFG 寄存器单独控制。

注：在运行模式下，确保 PDRUNCFG 寄存器的位 9 为 0。

3.8.2 睡眠模式

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

软件执行下列步骤时进入睡眠模式：

1. 确保 Cortex-M0 SCR 寄存器 （表 300）中的 SLEEPDEEP 位为 0。
2. 执行等待中断 (WFI) 指令。

中断到达处理器时，会自动退出睡眠模式。

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 系统 PLL 的功能描述

LPC11Axx 利用系统 PLL 为内核和外设创建时钟。

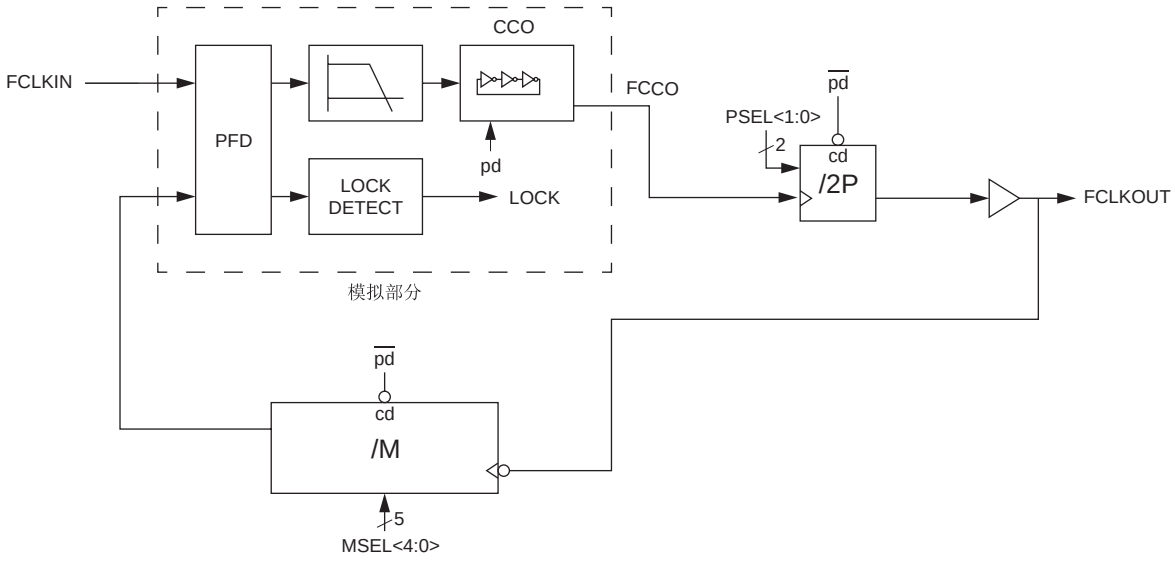


图 5. 系统 PLL 功能框图

此 PLL 的框图如图 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。CCO 输出通过可编程的后分频器按 2×P 分频来创建输出时钟。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。

3.9.1 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。

3.9.2 直接输出模式

在正常工作模式下，CCO 时钟将按 2、4、8 或 16 分频（取决于 PSEL 字段），并给定占空比为 50% 的输出时钟。

3.9.3 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SYS_PLL_PD位设置为 1 来使能该模式（表 32）。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 SYS_PLL_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。

3.9.4 工作模式

表 35. PLL 工作模式

模式	描述	PD 位	BYPASS 位	DIRECT 位
1	正常模式	0	0	0
3	掉电模式	1	0	x

3.9.5 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 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 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。

3.9.6 频率选择

PLL 频率的计算公式使用下列参数 （另请参见图 4）：

表 36. PLL 频率参数

参数	系统 PLL
FCLKIN	来自 XTAL、 IRC 或 CLKIN 的 sys_pllclkin （PLL 的输入时钟）的频率。
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 的后置分频器比率； SYSPLLCTRL 中的 PSEL 位（参见第 3.5.3 节）。
M	系统 PLL 的反馈分频寄存器； SYSPLLCTRL 中的 MSEL 位（参见第 3.5.3 节）。

3.9.6.1 模式 1 （正常模式）

该模式下，占空比时钟为 50%，频率关系如下：

$$Fclkout = M \times Fclkin = \frac{FCCO}{2 \times P}$$

按如下所示选择整数 M 和整数 P：

1. 假设 F_{clkin} 来自 IRC，为 12 MHz，在范围 1 至 32 中选择 M 来产生预期的输出频率
$$Fclkout = M \times 12MHz$$
2. 检验对于 P = 1、 2、 4 或 8 中的某个值，
$$FCCO = Fclkout \times 2 \times P$$
在范围 156MHz 至 320 MHz 中。
3. 如果在主时钟源选择寄存器（表 16）中选择了 PLL 输出，则检验由系统时钟分频寄存器（表 18）分频的 F_{clkout} 是否不超过最大处理器时钟频率（第 1.2 节或 LPC11Axx 数据手册）。
4. 用 MSEL = M-1 和 PSEL = P-1 对系统 PLL 控制寄存器 （表 8）进行编程。

3.9.6.2 模式 3 （掉电模式）

在此模式下，将关闭内部基准电流，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。当通过将 pd 设为低电平结束掉电模式时， PLL 将恢复正常工作，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 闪存访问

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。此寄存器属于闪存配置模块的一部分。

注：此寄存器设置不当会导致 LPC11Axx 闪存无法正常工作。

4.1 简介

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

4.2 特性

- ARM Cortex-M0 的重要组成部分
- 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
- 在 LPC11Axx 中， NVIC 支持 24 个向量中断
- 4 个可编程的中断优先级 （带硬件优先级屏蔽功能）
- 浮动向量表
- 软件中断生成

4.3 中断源

表 37 列出了 LPC11Axx 中的中断源。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。

表 37. 中断源与中断向量控制器的连接

异常号	IRQ 号	向量地址	源	标志
16-23	0-7	0x40-0x5C	GPIO	引脚中断 0-7
24-25	8-9	0x60-64	GPIO	端口中断 0-1
26	10	0x68	模拟比较器	边沿检测 比较器电平
27	11	0x6C	DAC	触发的事件
28	12-1	0x70-0x7	-	保留
30	14	0x78	SSP1	Tx FIFO 半空 Rx FIFO 半满 Rx 超时 Rx 溢出
31	15	0x7C	I^2C	SI （状态更改）
32	16	0x80	CT16B0	匹配 0-3 捕获 0-3
33	17	0x84	CT16B1	匹配 0-3 捕获 0-3
34	18	0x88	CT32B0	匹配 0-3 捕获 0-3
35	19	0x8C	CT32B1	匹配 0-3 捕获 0-3

表 37. 中断源与中断向量控制器的连接 *（续）*

异常号	IRQ 号	向量地址	源	标志
36	20	0x90	SSP0	Tx FIFO 半空 Rx FIFO 半满 Rx 超时 Rx 溢出
37	21	0x94	USART	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)
38-39	22-23	0x98-0x9C	-	保留
40	24	0xA0	ADC	转换结束
41	25	0xA4	WDT	看门狗超时
42	26	0xA8	BOD	掉电检测
43	27	0xAC	闪存接口	
44-47	28-31	0xB0-0xBC	-	保留

表 37 中的 IRQ 编号对应于 Cortex-M0 附录中所述的下列寄存器的位号：

- 第 25.5.2.2 节 “中断设置 - 使能寄存器”
- 第 25.5.2.3 节 “中断清除 - 使能寄存器”
- 第 25.5.2.4 节 “中断设置 - 挂起寄存器”
- 第 25.5.2.5 节 “中断清除 - 挂起寄存器”

IRQ 编号还映射到第 25.5.2.6 节 “中断优先级寄存器”中的字段。

5.1 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 38）。

表 38. GPIO 引脚可用

封装	GPIO 端口 0	GPIO 端口 1
WLCSP20	PIO0_0 至 PIO0_17	-
HVQFN33	PIO0_0 至 PIO0_27	-
LQFPN48	PIO0_0 至 PIO0_31	PIO1_0 至 PIO1_9

5.2 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 31），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）。
- 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）。
- 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）。

5.3 特性

5.3.1 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。

5.3.2 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 分组中断可将器件从睡眠模式唤醒。

5.3.3 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。

5.4 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。

5.4.1 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 31）。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。

5.4.2 GPIO 分组中断

对于连接至两个 GPIO 分组中断模块（组 0 和组 1）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，以及所有使能输入是基于 OR 操作还是 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。

5.4.3 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。

5.5 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 31）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 39 和第 5.5.1 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 40 和第 5.5.2 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 41 和第 5.5.2 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 42 和第 5.5.3 节。

注：在所有 GPIO 寄存器中，未显示的位为**保留位**。

表 39. 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 43
IENR	R/W	0x004	引脚中断电平 （上升沿）中断使能寄存器	0	表 44
SIENR	WO	0x008	引脚中断电平 （上升沿）中断设置寄存器	不适用	表 45
CIENR	WO	0x00C	引脚中断电平 （上升沿中断）清除寄存器	不适用	表 46
IENF	R/W	0x010	引脚中断有效电平（下降沿）中断使能寄存器	0	表 47
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 48
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 49
RISE	R/W	0x01C	引脚中断上升沿寄存器	0	表 50
FALL	R/W	0x020	引脚中断下降沿寄存器	0	表 51
IST	R/W	0x024	引脚中断状态寄存器	0	表 52

表 40. 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 53
PORT_POL0	R/W	0x020	GPIO 分组中断端口 0 极性寄存器	0xFFFF FFFF	表 54
PORT_POL1	R/W	0x024	GPIO 分组中断端口 1 极性寄存器	0xFFFF FFFF	表 55
PORT_ENA0	R/W	0x040	GPIO 分组中断端口 0 使能寄存器	0	表 56
PORT_ENA1	R/W	0x044	GPIO 分组中断端口 1 使能寄存器	0	表 57

表 41. 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 53
PORT_POL0	R/W	0x020	GPIO 分组中断端口 0 极性寄存器	0xFFFF FFFF	表 54
PORT_POL1	R/W	0x024	GPIO 分组中断端口 1 极性寄存器	0xFFFF FFFF	表 55
PORT_ENA0	R/W	0x040	GPIO 分组中断端口 0 使能寄存器	0	表 56
PORT_ENA1	R/W	0x044	GPIO 分组中断端口 1 使能寄存器	0	表 57

GPIO 端口地址可以字节、半字或字进行读写。

表 42. 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 58
B32 至 B42	R/W	0x0020 至 0x0029	字节引脚寄存器端口 1	ext[1]	字节 （8 位）	表 59
W0 至 W31	R/W	0x1000 至 0x107C	字引脚寄存器端口 0	ext[1]	字 （32 位）	表 60
W32 至 W42	R/W	0x1080 至 0x10A4	字引脚寄存器端口 1	ext[1]	字 （32 位）	表 61
DIR0	R/W	0x2000	方向寄存器端口 0	0	字 （32 位）	表 62
DIR1	R/W	0x2004	方向寄存器端口 1	0	字 （32 位）	表 63
MASK0	R/W	0x2080	屏蔽寄存器端口 0	0	字 （32 位）	表 64
MASK1	R/W	0x2084	屏蔽寄存器端口 1	0	字 （32 位）	表 65
PIN0	R/W	0x2100	端口引脚寄存器端口 0	ext[1]	字 （32 位）	表 66
PIN1	R/W	0x2104	端口引脚寄存器端口 1	ext[1]	字 （32 位）	表 67
MPIN0	R/W	0x2180	屏蔽端口寄存器端口 0	ext[1]	字 （32 位）	表 68
MPIN1	R/W	0x2184	屏蔽端口寄存器端口 1	ext[1]	字 （32 位）	表 69
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 70
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 71
CLR0	WO	0x2280	清除端口 0	不适用	字 （32 位）	表 72
CLR1	WO	0x2284	清除端口 1	不适用	字 （32 位）	表 73
NOT0	WO	0x2300	切换端口 0	不适用	字 （32 位）	表 74
NOT1	WO	0x2304	切换端口 1	不适用	字 （32 位）	表 75

[1] 此表和后续表中的 ext 是指复位后的数据读取取决于引脚的状态，这可能进一步又取决于外部源。

5.5.1 GPIO 引脚中断寄存器描述

5.5.1.1 引脚中断模式寄存器

对于 PINTSELn 寄存器 （参见表 31）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 43. 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。	-	-

5.5.1.2 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 31）， 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 44. 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。	-	-

5.5.1.3 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 31），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 45. 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。	-	-

5.5.1.4 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 31），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 46. 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。	-	-

5.5.1.5 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 31）， 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 47. 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。	-	-

5.5.1.6 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 31），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 48. 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。	-	-

5.5.1.7 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 31），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 49. 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。	-	-

5.5.1.8 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 31）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 发送一个中断请求。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 50. 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：复位或上一次向该位写 1 后，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。	-	-

5.5.1.9 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 31）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 发送一个中断请求。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 51. 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：复位或上一次向该位写 1 后，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。	-	-

5.5.1.10 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 52. 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。	-	-

5.5.2 GPIO 组 0/ 组 1 中断寄存器描述

5.5.2.1 分组中断控制寄存器

表 53. 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

5.5.2.2 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 54. 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 PIO0_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 55. 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 PIO1_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	-

5.5.2.3 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 56. 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 PIO0_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 57. 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 PIO1_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	-

5.5.3 GPIO 端口寄存器描述

5.5.3.1 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 58. 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 PIO0_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 59. GPIO 端口 1 字节引脚寄存器 （B32 至 B42，地址 0x5000 0020 至 0x5000 0029）位描述

位	符号	描述	复位值	访问类型
0	PBYTE	读：引脚 PIO1_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	-

5.5.3.2 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 60. 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 61. 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

5.5.3.3 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 62. 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 PIO0_n 的引脚方向（位 0 = PIO0_0，位 1 = PIO0_1，...，位 31 = PIO0_31）。 0 = 输入。 1 = 输出。	0	R/W

表 63. 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 PIO1_n 的引脚方向（位 0 = PIO1_0，位 1 = PIO1_1，...，位 31 = PIO1_31）。 0 = 输入。 1 = 输出。	0	R/W

5.5.3.4 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 64. 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 PIO0_n 的位是有效的（位 0 = PIO0_0，位 1 = PIO0_1， ...，位 31 = PIO0_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 65. 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 PIO1_n 的位是有效的（位 0 = PIO1_0，位 1 = PIO1_1， ...，位 31 = PIO1_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

5.5.3.5 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 66. 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 = PIO0_0，位 1 = PIO0_1，...，位 31 = PIO0_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 67. 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 = PIO1_0，位 1 = PIO1_1，...，位 31 = PIO1_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

5.5.3.6 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 68. 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 = PIO0_0，位 1 = PIO0_1，...，位 31 = PIO0_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 69. 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 = PIO1_0，位 1 = PIO1_1，...，位 31 = PIO1_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

5.5.3.7 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 70. 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 71. 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

5.5.3.8 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 72. 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 73. 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

5.5.3.9 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 74. 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 75. 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

5.6 功能说明

5.6.1 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。

5.6.2 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 5.6.1 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。

5.6.3 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。

5.6.4 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。

5.6.4.1 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 76 中的描述。

表 76. 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。	写入以使能电平中断。
CIENR	写入禁用上升沿中断。	写入以禁用电平中断。
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。

5.6.4.2 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（参见表 37）。

5.6.5 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。

6.1 本章导读

VDDCMP 是由针对不同封装的不同 IOCON 寄存器所控制的功能。

表 77. 特定于封装的 IOCON 寄存器

封装	寄存器 TCK_PIO5 位 FUNC = 0x2	寄存器 PIO0_14 位 FUNC = 0x5
WLCSP	VDDCMP	保留
LQFP	保留	VDDCMP
HVQFN	保留	VDDCMP

6.2 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
- 滞回
- 带模拟功能的焊盘的模拟 / 数字模式
- 开漏输出选项

6.3 简介

IOCON 寄存器控制 GPIO 或外设功能、输入模式以及所有具有 GPIO 功能的引脚的滞回。此外，具有 I^2C 功能的引脚可针对不同的 I^2C 总线模式进行配置。如果引脚具有模拟功能，则可选择模拟模式。

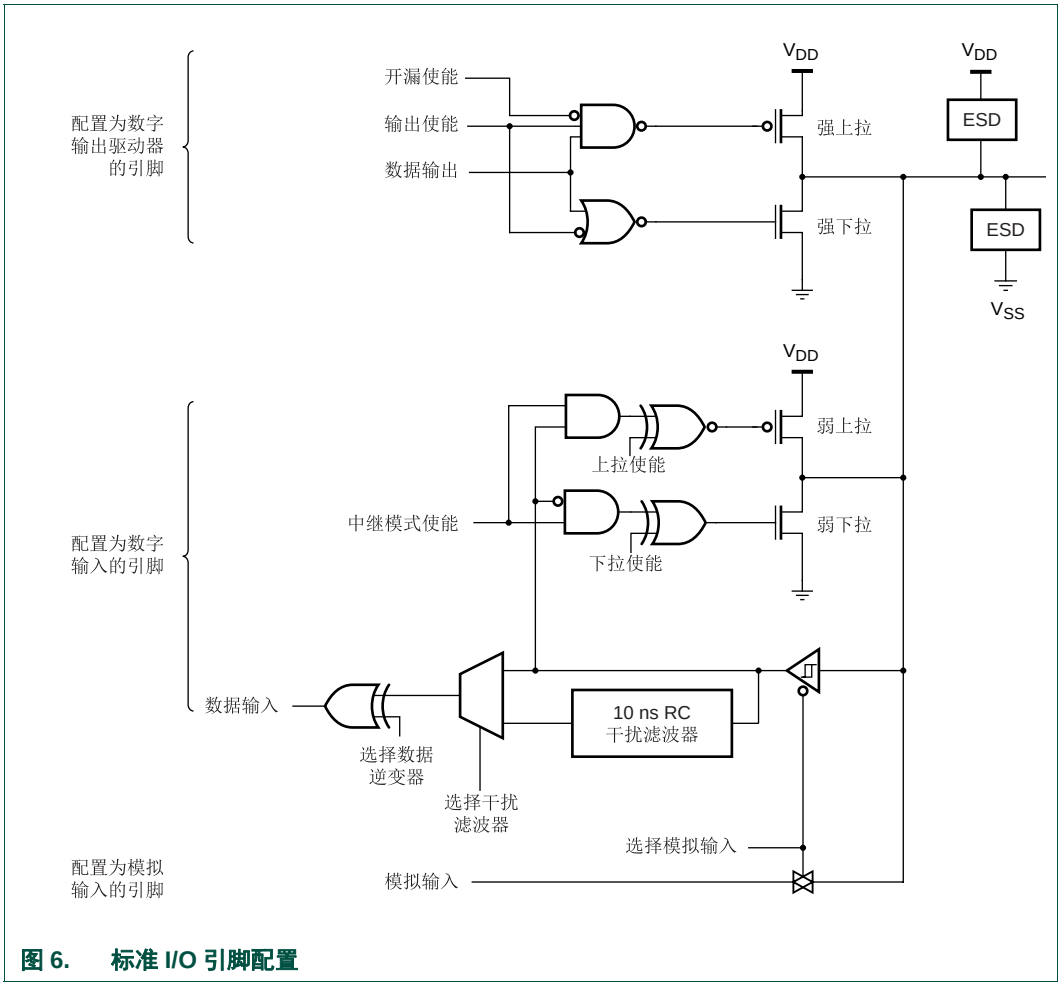


图 6. 标准 I/O 引脚配置

6.3.1 引脚功能

IOCON 寄存器中的 FUNC 位可设为 GPIO（通常为 000）或特殊功能。如果引脚设置为 GPIO，则 DIR 寄存器决定引脚是配置为输入还是输出（参见表 62）。对于任何特殊功能，引脚方向根据功能自动控制。 DIR 寄存器对特殊功能没有影响。

6.3.2 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。

当引脚处于高电平时，中继模式会使能上拉电阻；当引脚处于低电平时，则使能下拉电阻。这样，当引脚配置为输入且由内部驱动时，引脚可以保持上一个已知状态。如果暂时不驱动引脚，通常可用中继模式来防止引脚悬空（当引脚处于不确定状态时，可能会使用大量电量）。

6.3.3 滞回

可将数字功能的输入缓冲配置为有滞回或无滞回。

6.3.4 输入反转

包含该选项的目的是为了使用户无需在输入上包括一个仅可从外部源在相反极性提供的外部逆变器。切勿对 GPIO 输出设置该选项。这么做可能会使相同端口中其他引脚上的操作导致无意切换选择了输入反转的某个输出。举例来说，如果软件读取 GPIO 端口寄存器、修改值中的其他位 / 输出并将结果写回至端口寄存器，则选择了输入反转的端口中的任意输出都会更改状态。

6.3.5 模拟 / 数字模式

在模拟模式下，数字接收器会被断开以消除其对模拟功能的影响。如果选择了模拟模式，则 MODE 字段应当为“非工作”(00)； HYS、INV、FILTR、SLEW 以及 OD 等设置不起作用。

对于具有模拟功能的未连接引脚，则保持 ADMODE 位设为“数字”，MODE 字段设为非零。

6.3.6 I^2C 模式

提供寄存器 PIO0_2 和 PIO0_3（表 81）的 HS 和 HIDRIVE 位，因为这些引脚为器件上的主 I^2C 连接，但是这些选项也可用于其他应用。

- 对于标准模式或快速模式 I^2C 操作，则清除 HS 位以使能输入干扰滤波器。
- 对于超快速模式 I^2C 操作，则清除 HS 以使能输入干扰滤波器并设置 HIDRIVE 位以选择 20 mA 吸收电流。
- 在非 I^2C 操作中，无论 HS 和 HIDRIVE 如何设置，这些引脚都保持开漏并且只能驱动至 V_{SS}。使 HS 保持为 1、HIDRIVE 保持为 0 来实现与其他引脚的最大兼容性。如果清除 HS 则会使能可最多抑制 50 nS 宽脉冲的干扰滤波器。

6.3.7 输出转换速率

无须快速切换状态的数字输出的 SLEW 位应设置为“慢”。该设置可在不降低器件电源 / 接地分配性能的情况下同时切换更多输出，并且对信号跳变时间的影响很小。如果模拟精度对于该应用有效，则就尤为重要了。

6.3.8 开漏模式

选择了输出时（通过选择 FUNC 字段中的特殊功能，或通过选择在 PnDIR 寄存器中具有 1 的引脚的 GPIO 功能）， OD 位中的 1 会选择开漏操作，也即是说 1 会禁用高驱动晶体管。该选项对 I^2C 引脚无影响。

6.3.9 RESET （引脚 RESET/PIO0_0）

关于复位焊盘配置，请参见图 7。此复位引脚包括一个固定 20 ns 干扰滤波器。

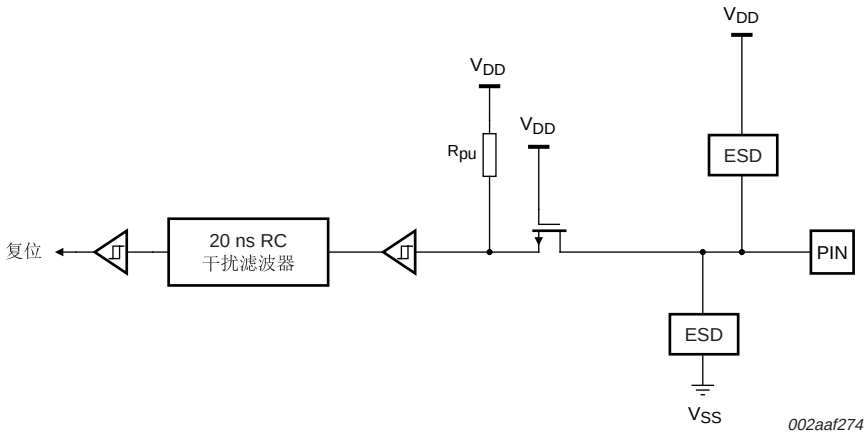


图 7. 复位焊盘配置

6.4 寄存器描述

I/O 配置寄存器控制 GPIO 端口引脚以及所有外设和功能模块的输入和输出。

每个 GPIO 引脚都会分配一个 IOCON 寄存器，用于控制引脚的功能和电气特性。

注：IOCON 寄存器按其端口号和地址在表格 78、80、82 以及 84 中列出。

表 78. 寄存器简介：I/O 配置 （基址 0x4004 4000）

名称	访问类型	地址偏移	描述	类型	复位值	参考
RESET_PIO0_0	R/W	0x000	引脚 RESET/PIO0_0 的 I/O 配置	D	0x90	表 79、80
PIO0_1	R/W	0x004	引脚 PIO0_1/RXD/CLKOUT/ CT32B0_MAT2/SSEL0/CLKIN 的 I/O 配置	D	0x90	表 79、80
PIO0_2	R/W	0x008	引脚 PIO0_2/SCL/ACMP_O/TCK/SWCLK/ 的 I/O 配置 CT16B0_CAP0	I	0x80	表 81、82
PIO0_3	R/W	0x00C	引脚 PIO0_3/SDA/ACMP_O/SWDIO/CT16B1_CAP0 的 I/O 配置	I	0x80	表 81、82
PIO0_4	R/W	0x010	引脚 PIO0_4/AOUT/CT16B0_MAT1/MOSI0 的 I/O 配置	A	0x90	表 83、84
TCK_PIO0_5	R/W	0x014	引脚 TCK/SWCLK/PIO0_5/ VDDCMP/CT16B0_MAT2/SCK0 的 I/O 配置 （参见第 6.1 节）	A	0x90	表 83、84
TDI_PIO0_6	R/W	0x018	引脚 TDI/PIO0_6/AD0/CT32B0_MAT3/MISO0 的 I/O 配置	A	0x90	表 83、84
TMS_PIO0_7	R/W	0x01C	引脚 TMS/PIO0_7/AD1/CT32B1_CAP0/ CT16B0_MAT0 的 I/O 配置	A	0x90	表 83、84
TDO_PIO0_8	R/W	0x020	引脚 TDO/PIO0_8/AD2/CT32B1_MAT0/SCK1 的 I/O 配置	A	0x90	表 83、84
TRST_PIO0_9	R/W	0x024	引脚 TRST/PIO0_9/AD3/ CT32B1_MAT1/CT16B0_MAT1/CTS 的 I/O 配置	A	0x90	表 83、84

表 78. 寄存器简介：I/O 配置 （基址 0x4004 4000）*（续）*

名称	访问类型	地址偏移	描述	类型	复位值	参考
SWDIO_PIO0_10	R/W	0x028	引脚 SWDIO/PIO0_10/AD4/ CT32B1_MAT2/CT16B0_MAT2/$\overline{\text{RTS}}$ 的 I/O 配置	A	0x90	表 83、84
PIO0_11	R/W	0x02C	引脚 PIO0_11/SCLK/AD5/ CT32B1_MAT3/CT32B0_CAP0 的 I/O 配置	A	0x90	表 83、84
PIO0_12	R/W	0x030	引脚 PIO0_12/RXD/ACMP_O/ CT32B0_MAT0/SCL/CLKIN 的 I/O 配置	D	0x90	表 79、80
PIO0_13	R/W	0x034	引脚 PIO0_13/TXD/ACMP_I2/ CT32B0_MAT1/SDA 的 I/O 配置	A	0x90	表 83、84
PIO0_14	R/W	0x038	引脚 PIO0_14/MISO1/AD6/ CT32B0_CAP1/CT16B1_MAT1/VDDCMP 的 I/O 配置 （参见第 6.1 节）	A	0x90	表 83、84
PIO0_15	R/W	0x03C	引脚 PIO0_15/TXD/AD7/CT32B0_CAP2/SDA 的 I/O 配置	A	0x90	表 83、84
PIO0_16	R/W	0x040	引脚 PIO0_16/ATRG0/ACMP_I3/CT16B0_CAP1/SCL 的 I/O 配置	A	0x90	表 83、84
PIO0_17	R/W	0x044	引脚 PIO0_17/ATRG1/ACMP_I4/ CT16B0_CAP2/CT16B0_MAT0 的 I/O 配置	A	0x90	表 83、84
PIO0_18	R/W	0x048	引脚 PIO0_18/SSEL0/ CT16B0_CAP0/CT16B1_CAP1 的 I/O 配置	D	0x90	表 79、80
PIO0_19	R/W	0x04C	引脚 PIO0_19/CLKIN/CLKOUT/ MOSI0/CT16B1_MAT0 的 I/O 配置	D	0x90	表 79、80
PIO0_20	R/W	0x050	引脚 PIO0_20/SCK0/ CT32B1_CAP0/CT16B1_MAT2 的 I/O 配置	D	0x90	表 79、80
PIO0_21	R/W	0x054	引脚 PIO0_21/$\overline{\text{CTS}}$/ACMP_O/ CT32B1_CAP1/SCLK 的 I/O 配置	D	0x90	表 79、80
PIO0_22	R/W	0x058	引脚 PIO0_22/MISO0/ACMP_I5/ CT32B1_MAT2/CT32B1_CAP2 的 I/O 配置	A	0x90	表 83、84
PIO0_23	R/W	0x05C	引脚 PIO0_23/$\overline{\text{RTS}}$/ACMP_O/ CT32B0_CAP0/SCLK 的 I/O 配置	D	0x90	表 79、80
PIO0_24	R/W	0x060	引脚 PIO0_24/SCL/CLKIN/ CT16B1_CAP0 的 I/O 配置	D	0x90	表 79、80
PIO0_25	R/W	0x064	引脚 PIO0_25/SDA/SSEL1/ CT16B1_MAT0 的 I/O 配置	D	0x90	表 79、80
PIO0_26	R/W	0x068	引脚 PIO0_26/TXD/MISO1/ CT16B1_CAP1/CT32B0_CAP2 的 I/O 配置	D	0x90	表 79、80
PIO0_27	R/W	0x06C	引脚 PIO0_27/MOSI1/ACMP_I1/ CT32B1_MAT1/CT16B1_CAP2 的 I/O 配置	A	0x90	表 83、84
PIO0_28	R/W	0x070	引脚 PIO0_28/$\overline{\text{DTR}}$/SSEL1/ CT32B0_CAP0 的 I/O 配置	D	0x90	表 79、80
PIO0_29	R/W	0x074	引脚 PIO0_29/$\overline{\text{DSR}}$/SCK1/ CT32B0_CAP1 的 I/O 配置	D	0x90	表 79、80
PIO0_30	R/W	0x078	引脚 PIO0_30/$\overline{\text{RI}}$/MOSI1/ CT32B0_MAT0/CT16B0_CAP0 的 I/O 配置	D	0x90	表 79、80
PIO0_31	R/W	0x07C	引脚 PIO0_31/$\overline{\text{RI}}$/MOSI1/ CT32B1_MAT0/CT16B1_CAP1 的 I/O 配置	D	0x90	表 79、80
PIO1_0	R/W	0x080	引脚 PIO1_0/$\overline{\text{DCD}}$/SCK0/ CT32B1_MAT3/CT16B0_MAT1 的 I/O 配置	D	0x90	表 79、80

表 78. 寄存器简介：I/O 配置 （基址 0x4004 4000）*（续）*

名称	访问类型	地址偏移	描述	类型	复位值	参考
PIO1_1	R/W	0x084	引脚 PIO1_1/DTR/SSEL0/ CT32B1_MAT3/CT16B1_MAT0 的 I/O 配置	D	0x90	表 79、80
PIO1_2	R/W	0x088	引脚 PIO1_2/DSR/MISO0/ CT16B1_MAT2/CT16B1_MAT1 的 I/O 配置	D	0x90	表 79、80
PIO1_3	R/W	0x08C	引脚 PIO1_3/RI/MOSI0/ CT16B1_CAP0 的 I/O 配置	D	0x90	表 79、80
PIO1_4	R/W	0x090	引脚 PIO1_4/RXD/SSEL1/ CT32B0_MAT1/CT32B1_CAP0/CT16B0_CAP1 的 I/O 配置	D	0x90	表 79、80
PIO1_5	R/W	0x094	引脚 PIO1_5/TXD/SCK1/ CT32B0_MAT2/CT32B1_CAP1/CT16B0_CAP2 的 I/O 配置	D	0x90	表 79、80
PIO1_6	R/W	0x098	引脚 PIO1_6/RTS/MOSI1/ CT32B0_MAT3/CT32B1_CAP2/CT16B0_MAT0 的 I/O 配置	D	0x90	表 79、80
PIO1_7	R/W	0x09C	引脚 PIO1_7/CTS/MOSI0/ CT32B1_MAT1/CT16B0_MAT2/CT16B1_CAP2 的 I/O 配置	D	0x90	表 79、80
PIO1_8	R/W	0x0A0	引脚 PIO1_8/RXD/MISO1/ CT32B1_MAT0/CT16B1_MAT1 的 I/O 配置	D	0x90	表 79、80
PIO1_9	R/W	0x0A4	引脚 PIO1_9/DCD/ CT32B1_MAT2/CT16B1_MAT2 的 I/O 配置	D	0x90	表 79、80

6.4.1 I/O 配置寄存器

LPC11Axx IOCON 寄存器有 3 种类型，分别为 D 型、I 型和 A 型。有关 I/O 配置设置的详情，请参见第 6.3 节。

6.4.1.1 D 型 IOCON 寄存器

表 79. D 型 IOCON 寄存器位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。有关特定值的信息，请参见表 80。	000
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）。	10
		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[1]
		0	禁用。	
		1	使能。	
6	INV		输入极性	0
		0	非反相（引脚高电平 = 1）	
		1	反相（引脚高电平 = 0）	
8:7	-	-	保留。	01
9	SLEW		驱动器转换速率	0
		0	慢速（可同时切换更多输入）	
		1	快速	
10	OD		控制开漏模式	0
		0	图腾柱极输出	
		1	开漏输出（高电平驱动禁用）	
31:11	-	-	保留。	0

[1] HYS 复位至 1 的 RESET/PIO0_0 除外（使能滞回）

表 80. D 型 IOCON 寄存器：FUNC 值和引脚功能

该表中的阴影区域表示保留值，不应对其进行编程。

	FUNC 字段的值							
寄存器	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
RESET_PIO0_0	RESET	PIO0_0						
PIO0_1	PIO0_1	RXD	CLKOUT	CT32B0_MAT2	SSEL0	CLKIN		
PIO0_12	PIO0_12	RXD	ACMP_O	CT32B0_MAT0	SCL	CLKIN		
PIO0_18	PIO0_18		SSEL0	CT16B0_CAP0	CT16B1_CAP1			
PIO0_19	PIO0_19	CLKIN	CLKOUT	MOSI0	CT16B1_MAT0			
PIO0_20	PIO0_20		SCK0	CT32B1_CAP0	CT16B1_MAT2			
PIO0_21	PIO0_21	$\overline{\text{CTS}}$	ACMP_O	CT32B1_CAP1	SCLK			
PIO0_23	PIO0_23	$\overline{\text{RTS}}$	ACMP_O	CT32B0_CAP0	SCLK			
PIO0_24	PIO0_24	SCL	CLKIN	CT16B1_CAP0				
PIO0_25	PIO0_25	SDA	SSEL1	CT16B1_MAT0				
PIO0_26	PIO0_26	TXD	MISO1	CT16B1_CAP1	CT32B0_CAP2			
PIO0_28	PIO0_28	$\overline{\text{DTR}}$	SSEL1	CT32B0_CAP0				
PIO0_29	PIO0_29	$\overline{\text{DSR}}$	SCK1	CT32B0_CAP1				
PIO0_30	PIO0_30	$\overline{\text{RI}}$	MOSI1	CT32B0_MAT0	CT16B0_CAP0			

表 80. D 型 IOCON 寄存器：FUNC 值和引脚功能 *（续）*

该表中的阴影区域表示保留值，不应对其进行编程。

	FUNC 字段的值							
寄存器	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
PIO0_31	PIO0_31	$\overline{RI}$	MOSI1	CT32B1_MAT0	CT16B1_CAP1			
PIO1_0	PIO1_0	$\overline{DCD}$	SCK0	CT32B1_MAT3	CT16B0_MAT1			
PIO1_1	PIO1_1	$\overline{DTR}$	SSEL0	CT32B1_MAT3	CT16B1_MAT0			
PIO1_2	PIO1_2	$\overline{DSR}$	MISO0	CT16B1_MAT2	CT16B1_MAT1			
PIO1_3	PIO1_3	$\overline{RI}$	MOSI0	CT16B1_CAP0				
PIO1_4	PIO1_4	RXD	SSEL1	CT32B0_MAT1	CT32B1_CAP0	CT16B0_CAP1		
PIO1_5	PIO1_5	TXD	SCK1	CT32B0_MAT2	CT32B1_CAP1	CT16B0_CAP2		
PIO1_6	PIO1_6	$\overline{RTS}$	MOSI1	CT32B0_MAT3	CT32B1_CAP2	CT16B0_MAT0		
PIO1_7	PIO1_7	$\overline{CTS}$	MOSI0	CT32B1_MAT1	CT16B0_MAT2	CT16B1_CAP2		
PIO1_8	PIO1_8	RXD	MISO1	CT32B1_MAT0	CT16B1_MAT1			
PIO1_9	PIO1_9	$\overline{DCD}$		CT32B1_MAT2	CT16B1_MAT2			

6.4.1.2 I 型 IOCON 寄存器

表 81. I 型 IOCON 寄存器位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。有关特定值的信息，请参见表 82。	0
7:3		-	保留。	10000
8	HS		禁用 I^2C 功能以实现更快速的操作。	0
		0	使能 I^2C 干扰滤波器和转换速率。	
		1	禁用 I^2C 干扰滤波器和转换速率。	
9	HIDRIVE	0	吸收电流为标准的 4 mA。	0
		1	吸收电流为 20 mA。	
31:10	-	-	保留。	-

表 82. I 型 IOCON 寄存器：FUNC 值和引脚功能

该表中的阴影区域表示保留值，不应对其进行编程。

	FUNC 字段的值							
寄存器	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
PIO0_2	PIO0_2	SCL	ACMP_O	TCK/ SWCLK	CT16B0_CAP0			
PIO0_3	PIO0_3	SDA	ACMP_O	SWDIO	CT16B1_CAP0			

6.4.1.3 A 型 IOCON 寄存器

表 83. A 型 IOCON 寄存器位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能。有关特定值的信息，请参见表 84。	0

表 83. A 型 IOCON 寄存器位描述 *（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）。	10
		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		输入极性	0
		0	非反相 （引脚高电平 = 1）	
		1	反相 （引脚高电平 = 0）	
7	ADMODE		选择模拟 / 数字模式。	1
		0	模拟模式。	
		1	数字模式。	
8	FILTR		控制干扰滤波器	1
		0	噪声脉冲被滤波	
		1	未进行任何滤波	
9	SLEW		驱动器转换速率	0
		0	慢速 （可同时切换更多输入）	
		1	快速	
10	OD		控制开漏模式	0
		0	图腾柱极输出	
		1	开漏输出 （高电平驱动禁用）	
31:11	-	-	保留。	0

表 84. A 型 IOCON 寄存器：FUNC 值和引脚功能

该表中的阴影区域表示保留值，不应对其进行编程。

	FUNC 字段的值							
寄存器	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
PIO0_4	PIO0_4		AOUT	CT16B0_MAT1	MOSI0			
TCK_PIO0_5	TCK/SWCLK	PIO0_5	VDDCMP [1]	CT16B0_MAT2	SCK0			
TDI_PIO0_6	TDI	PIO0_6	AD0	CT32B0_MAT3	MISO0			
TMS_PIO0_7	TMS	PIO0_7	AD1	CT32B1_CAP0	CT16B0_MAT0			
TDO_PIO0_8	TDO	PIO0_8	AD2	CT32B1_MAT0	SCK1			
TRST_PIO0_9	$\overline{\text{TRST}}$	PIO0_9	AD3	CT32B1_MAT1	CT16B0_MAT1	$\overline{\text{CTS}}$		
SWDIO_PIO0_10	SWDIO	PIO0_10	AD4	CT32B1_MAT2	CT16B0_MAT2	$\overline{\text{RTS}}$		
PIO0_11	PIO0_11	SCLK	AD5	CT32B1_MAT3	CT32B0_CAP0			
PIO0_13	PIO0_13	TXD	ACMP_I2	CT32B0_MAT1	SDA			
PIO0_14	PIO0_14	MISO1	AD6	CT32B0_CAP1	CT16B1_MAT1	VDDCMP [1]		
PIO0_15	PIO0_15	TXD	AD7	CT32B0_CAP2	SDA			
PIO0_16	PIO0_16	ATRG0	ACMP_I3	CT16B0_CAP1	SCL			
PIO0_17	PIO0_17	ATRG1	ACMP_I4	CT16B0_CAP2	CT16B0_MAT0			
PIO0_22	PIO0_22	MISO0	ACMP_I5	CT32B1_MAT2	CT32B1_CAP2			
PIO0_27	PIO0_27	MOSI1	ACMP_I1	CT32B1_MAT1	CT16B1_CAP2			

[1] 参见第 6.1 节。

7.1 引脚配置

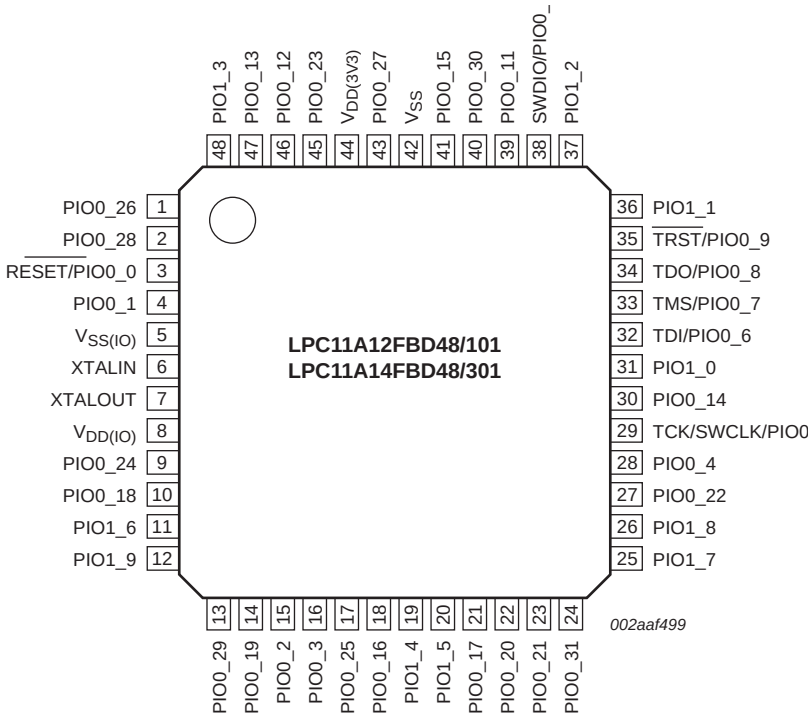


图 8. 引脚配置 LQFP48 封装

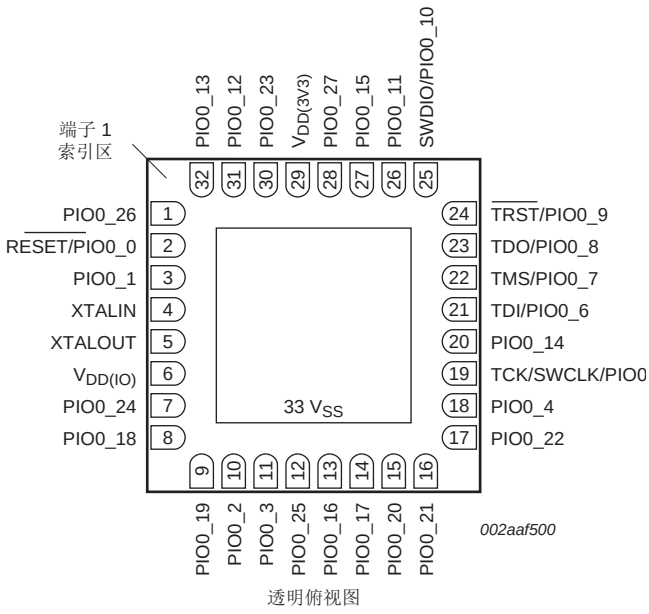


图 9. 引脚配置 HVQFN 33 封装

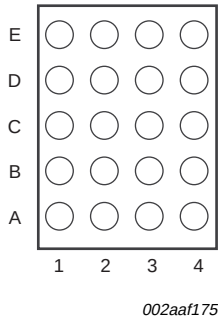


图 10. 引脚配置 WLCSP20 封装

7.2 引脚说明

UM10527 上的所有功能引脚映射到 GPIO 端口 0 和端口 1（参见表 86）。端口引脚经多路复用后可包含多个功能 （参见表 85）。

引脚功能由引脚的 IOCON 寄存器进行控制 （《参见 LPC11A1x 用户手册》）。标准 I/O 焊盘配置如图 6 所示，详细引脚描述见表 86。

表 85. 引脚复用

功能	类型	端口	干扰滤波器	LQFP48 引脚	HVQFN33 引脚	WCSP20 引脚
系统时钟、复位和唤醒						
CLKIN	I	PIO0_1	否	4	3	B2
		PIO0_12	否	46	31	E1
		PIO0_19	否	14	9	-
		PIO0_24	否	9	7	-
CLKOUT	O	PIO0_1	否	4	3	B2
		PIO0_19	否	14	9	-
XTALIN	I（模拟）	-	-	6	4	-
XTALOUT	O（模拟）	-	-	7	5	-
$\overline{\text{RESET}}$	I	PIO0_0	20 ns[1]	3	2	C1
串行调试接口 (SWD) 和 JTAG						
$\overline{\text{TRST}}$	I	PIO0_9	10 ns[2]	35	24	D4
TCK	I	PIO0_5	10 ns[2]	29	19	B3
TDI	I	PIO0_6	10 ns[2]	32	21	C3
TDO	O	PIO0_8	否	34	23	C2
TMS	I	PIO0_7	10 ns[2]	33	22	C4
SWCLK	I	PIO0_2	50 ns[2]	15	10	A1
		PIO0_5	10 ns[2]	29	19	B3
SWDIO	I/O	PIO0_3	50 ns[2]	16	11	B1
		PIO0_10	10 ns[2]	38	25	D3

表 85. 引脚复用 *（续）*

功能	类型	端口	干扰滤波器	LQFP48 引脚	HVQFN33 引脚	WCSP20 引脚
模拟外设 （ADC、 DAC、比较器）						
ACMP_I1	I （模拟）	PIO0_27	否	43	28	-
ACMP_I2	I （模拟）	PIO0_13	否	47	32	D1
ACMP_I3	I （模拟）	PIO0_16	否	18	13	A2
ACMP_I4	I （模拟）	PIO0_17	否	21	14	A3
ACMP_I5	I （模拟）	PIO0_22	否	27	17	-
ACMP_O	O （数字）	PIO0_2	否	15	10	A1
		PIO0_3	否	16	11	B1
		PIO0_12	否	46	31	E1
		PIO0_21	否	23	16	-
		PIO0_23	否	45	30	-
AD0	I （模拟）	PIO0_6	否	32	21	C3
AD1	I （模拟）	PIO0_7	否	33	22	C4
AD2	I （模拟）	PIO0_8	否	34	23	C2
AD3	I （模拟）	PIO0_9	否	35	24	D4
AD4	I （模拟）	PIO0_10	否	38	25	D3
AD5	I （模拟）	PIO0_11	否	39	26	D2
AD6	I （模拟）	PIO0_14	否	30	20	B4
AD7	I （模拟）	PIO0_15	否	41	27	E4
AOUT	O （模拟）	PIO0_4	否	28	18	A4
ATRG0	I	PIO0_16	10 ns[2]	18	13	A2
ATRG1	I	PIO0_17	10 ns[2]	21	14	A3
VDDCMP	I （模拟）	PIO0_14	否	30	20	-
		PIO0_5	否	-	-	B3
I^2C 总线接口						
SCL	I/O	PIO0_2	50 ns[2]	15	10	A1
		PIO0_12	否	46	31	E1
		PIO0_16	10 ns[2]	18	13	A2
		PIO0_24	否	9	7	-
SDA	I/O	PIO0_3	50 ns[2]	16	11	B1
		PIO0_13	10 ns[2]	47	32	D1
		PIO0_15	10 ns[2]	41	27	E4
		PIO0_25	否	17	12	-
SSP0 控制器						
MISO0	I/O	PIO0_6	10 ns[2]	32	21	C3
		PIO0_22	10 ns[2]	27	17	-
		PIO1_2	否	37	-	-

表 85. 引脚复用 *（续）*

功能	类型	端口	干扰滤波器	LQFP48 引脚	HVQFN33 引脚	WCSP20 引脚
MOSI0	I/O	PIO0_4	10 ns[2]	28	18	A4
		PIO0_19	否	14	9	-
		PIO1_3	否	48	-	-
		PIO1_7	否	25	-	-
SCK0	I/O	PIO0_5	10 ns[2]	29	19	B3
		PIO0_20	否	22	15	-
		PIO1_0	否	31	-	-
SSEL0	I/O	PIO0_1	否	4	3	B2
		PIO0_18	否	10	8	-
		PIO1_1	否	36	-	-
SSP1 控制器						
MISO1	I/O	PIO0_14	10 ns[2]	30	20	-
		PIO0_26	否	1	1	-
		PIO1_8	否	26	-	-
MOSI1	I/O	PIO0_27	10 ns[2]	43	28	-
		PIO0_31	否	24	-	-
		PIO0_30	否	40	-	-
		PIO1_6	否	11	-	-
	I/O	PIO0_8	10 ns[2]	34	23	-
		PIO1_5	否	20	-	-
		PIO0_29	否	13	-	-
SSEL1	I/O	PIO0_25	否	17	12	-
		PIO1_4	否	19	-	-
		PIO0_28	否	2	-	-
USART						
RXD	I	PIO0_1	否	4	3	B2
		PIO0_12	否	46	31	E1
		PIO1_4	否	19	-	-
		PIO1_8	否	26	-	-
TXD	O	PIO0_13	否	47	32	D1
		PIO0_15	否	41	27	E4
		PIO0_26	否	1	1	-
		PIO1_5	否	20	-	-
SCLK	I/O	PIO0_11	10 ns[2]	39	26	D2
		PIO0_21	否	23	16	-
		PIO0_23	否	45	30	-
$\overline{\text{CTS}}$	I	PIO0_9	10 ns[2]	35	24	D4
		PIO0_21	否	23	16	-
		PIO1_7	否	25	-	-

表 85. 引脚复用 *（续）*

功能	类型	端口	干扰滤波器	LQFP48 引脚	HVQFN33 引脚	WCSP20 引脚
$\overline{\text{RTS}}$	O	PIO0_10	否	38	25	D3
		PIO0_23	否	45	30	-
		PIO1_6	否	11	-	-
$\overline{\text{DCD}}$	I	PIO1_9	否	12	-	-
		PIO1_0	否	31	-	-
$\overline{\text{DSR}}$	I	PIO0_29	否	13	-	-
		PIO1_2	否	37	-	-
$\overline{\text{DTR}}$	O	PIO0_28	否	2	-	-
		PIO1_1	否	36	-	-
$\overline{\text{RI}}$	I	PIO0_30	否	40	-	-
		PIO0_31	否	24	-	-
		PIO1_3	否	48	-	-
16 位计数器 / 定时器 CT16B0						
CT16B0_CAP0	I	PIO0_2	50 ns[2]	15	10	A1
		PIO0_18	否	10	8	-
		PIO0_30	否	40	-	-
CT16B0_CAP1	I	PIO0_16	10 ns[2]	18	13	A2
		PIO1_4	否	19	-	-
CT16B0_CAP2	I	PIO0_17	10 ns[2]	21	14	A3
		PIO1_5	否	20	-	-
CT16B0_MAT0	O	PIO0_7	否	33	22	C4
		PIO0_17	否	21	14	A3
		PIO1_6	否	11	-	-
CT16B0_MAT1	O	PIO0_4	否	28	18	A4
		PIO0_9	否	35	24	D4
		PIO1_0	否	31	-	-
CT16B0_MAT2	O	PIO0_5	否	29	19	B3
		PIO0_10	否	38	25	D3
		PIO1_7	否	25	-	-
16 位计数器 / 定时器 CT16B1						
CT16B1_CAP0	I	PIO0_3	50 ns[2]	16	11	B1
		PIO0_24	否	9	7	-
		PIO1_3	否	48	-	-
CT16B1_CAP1	I	PIO0_18	否	10	8	-
		PIO0_26	否	1	1	-
		PIO0_31	否	24	-	-
CT16B1_CAP2	I	PIO0_27	10 ns[2]	43	28	-
		PIO1_7	否	25	-	-

表 85. 引脚复用 *（续）*

功能	类型	端口	干扰滤波器	LQFP48 引脚	HVQFN33 引脚	WCSP20 引脚
CT16B1_MAT0	O	PIO0_19	否	14	9	-
		PIO0_25	否	17	12	-
		PIO1_1	否	36	-	-
CT16B1_MAT1	O	PIO0_14	否	30	20	B4
		PIO1_2	否	37	-	-
		PIO1_8	否	26	-	-
CT16B1_MAT2	O	PIO0_20	否	22	15	-
		PIO1_2	否	37	-	-
		PIO1_9	否	12	-	-
32 位计数器 / 定时器 CT32B0						
CT32B0_CAP0	I	PIO0_11	10 ns[2]	39	26	D2
		PIO0_23	否	45	30	-
		PIO0_28	否	2	-	-
CT32B0_CAP1	I	PIO0_14	10 ns[2]	30	20	B4
		PIO0_29	否	13	-	-
CT32B0_CAP2	I	PIO0_15	10 ns[2]	41	27	E4
		PIO0_26	否	1	1	-
CT32B0_MAT0	O	PIO0_12	否	46	31	E1
		PIO0_30	否	40	-	-
CT32B0_MAT1	O	PIO0_13	否	47	32	D1
		PIO1_4	否	19	-	-
CT32B0_MAT2	O	PIO0_1	否	4	3	B2
		PIO1_5	否	20	-	-
CT32B0_MAT3	O	PIO0_6	否	32	21	C3
		PIO1_6	否	11	-	-
32 位计数器 / 定时器 CT32B1						
CT32B1_CAP0	I	PIO0_7	10 ns[2]	33	22	C4
		PIO0_20	否	22	15	-
		PIO1_4	否	19	-	-
CT32B1_CAP1	I	PIO0_21	否	23	16	-
		PIO1_5	否	20	-	-
CT32B1_CAP2	I	PIO0_22	10 ns[2]	27	17	-
		PIO1_6	否	11	-	-
CT32B1_MAT0	O	PIO0_8	否	34	23	C2
		PIO0_31	否	24	-	-
		PIO1_8	否	26	-	-
CT32B1_MAT1	O	PIO0_9	否	35	24	D4
		PIO0_27	否	43	28	-
		PIO1_7	否	25	-	-

表 85. 引脚复用 *（续）*

功能	类型	端口	干扰滤波器	LQFP48 引脚	HVQFN33 引脚	WCSP20 引脚
CT32B1_MAT2	O	PIO0_10	否	38	25	D3
		PIO0_22	否	27	17	-
		PIO1_9	否	12	-	-
CT32B1_MAT3	O	PIO0_11	否	39	26	D2
		PIO1_1	否	36	-	-
		PIO1_0	否	31	-	-
电源和接地引脚						
$V_{DD(IO)}$	供电	-	-	8	6	E2
$V_{DD(3V3)}$	供电	-	-	44	29	E2
V_{SS}	接地	-	-	42	33	E3
$V_{SS(IO)}$	接地	-	-	5	33	E3

[1] 始终打开。

[2] 可编程开 / 关。默认情况下，禁用干扰滤波器。

表 86 按端口号顺序显示所有引脚。首先列出的是复位后的默认功能。 PIO0_0 到 PIO1_9 之间的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_2 和 PIO0_3 除外。

可通过每个端口引脚的IOCON寄存器来对上拉/下拉配置、中继模式和开漏模式进行编程。

表 86. LPC11A1x 引脚描述表

符号	引脚 LQFP48	HVQFN33	WLCSP20	类型	复位状态 [1]	描述
$\overline{\text{RESET}}$/PIO0_0	3	2	C1 [2]	I	I; PU	**$\overline{\text{RESET}}$** — 外部复位输入，具有固定 20 ns 干扰滤波器：此引脚为低电平时将复位设备，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。
				I/O	-	**PIO0_0** — 通用数字输入 / 输出引脚。
PIO0_1/RXD/CLKOUT/ CT32B0_MAT2/SSEL0/ CLKIN	4	3	B2 [3]	I/O	I; PU	**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。
				I	-	**RXD** — USART 的接收器数据输入。
				O	-	**CLKOUT** — 时钟输出。
				O	-	**CT32B0_MAT2** — 32 位定时器 0 的匹配输出 2。
				I/O	-	**SSEL0** — SSP0 的从机选择。
				I	-	**CLKIN** — 外部时钟输入。

表 86. LPC11A1x 引脚描述表 *（续）*

符号	引脚				类型	复位状态 [1]	描述
	LQFP48	HVQFN33	WLCSP20				
PIO0_2/SCL/ACMP_O/ TCK/SWCLK/ CT16B0_CAP0	15	10	A1	[4]	I/O	I; IA	**PIO0_2** — 通用数字输入 / 输出引脚。大电流吸收 (20 mA) 或标准电流吸收 (4 mA) 可编程；用于所有引脚功能的真正开漏。支持输入干扰滤波器 (50 ns)。
					I/O	-	**SCL** — I^2C 总线时钟（真正的开漏）输入 / 输出。支持输入干扰滤波器 (50 ns)。
					O	-	**ACMP_O** — 模拟比较器输出。
					I	-	**TCK/SWCLK** — 串行调试接口时钟 （LQFP 封装和 VQFN 封装的次级时钟）。支持输入干扰滤波器 (50 ns)。仅对于 WLCSP20 封装，该引脚在复位后由启动引导程序配置为 SWCLK 功能。
					I	-	**CT16B0_CAP0** — 16 位定时器 0 的捕获输入 0。支持输入干扰滤波器 (50 ns)。
PIO0_3/SDA/ACMP_O/ SWDIO/CT16B1_CAP0	16	11	B1	[4]	I/O	I; IA	**PIO0_3** — 通用数字输入 / 输出引脚。大电流吸收 (20 mA) 或标准电流吸收 (4 mA) 可编程；用于所有引脚功能的真正开漏。支持输入干扰滤波器 (50 ns)。
					I/O	-	**SDA** — I^2C 总线数据（真正的开漏）输入 / 输出。支持输入干扰滤波器 (50 ns)。
					O	-	**ACMP_O** — 模拟比较器输出。
					I/O	-	**SWDIO** — 串行调试接口 I/O （LQFP 封装和 HVQFN 封装的次级 I/O）。支持输入干扰滤波器 (50 ns)。仅对于 WLCSP20 封装，该引脚在复位后由启动引导程序配置为 SWDIO 功能。
					I	-	**CT16B1_CAP0** — 16 位定时器 1 的捕获输入 0。支持输入干扰滤波器 (50 ns)。
PIO0_4/R/AOUT/ CT16B0_MAT1/MOSI0	28	18	A4	[5]	I/O	I; PU	**PIO0_4** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					-	-	**R** — 保留。
					O	-	**AOUT** — D/A 转换器输出。
					O	-	**CT16B0_MAT1** — 16 位定时器 0 的匹配输出 1。
					I/O	-	**MOSI0** — SSP0 的主机输出从机输入。支持输入干扰滤波器 (10 ns)。
TCK/SWCLK/PIO0_5/ R/CT16B0_MAT2/ SCK0	29	19	-	[7]	I	I; PU	**TCK/SWCLK** — JTAG 接口的测试时钟 TCK 以及初级 （默认）串行调试接口时钟。支持输入干扰滤波器 (10 ns)。
					I/O	-	**PIO0_5** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					-	-	**R** — 保留。
					O	-	**CT16B0_MAT2** — 16 位定时器 0 的匹配输出 2。
					I/O	-	**SCK0** — SSP0 的串行时钟。支持输入干扰滤波器 (10 ns)。

表 86. LPC11A1x 引脚描述表 *（续）*

符号	引脚				类型	复位状态 [1]	描述
	LQFP48	HVQFN33	WLCSP20				
TCK/SWCLK/PIO0_5/ VDDCMP/ CT16B0_MAT2/ SCK0	-	-	B3	[5] [6]	I	I; PU	**TCK/SWCLK** — JTAG 接口的测试时钟 TCK 以及次级串行调试接口时钟。默认 TCK/SWCLK 功能使用 PIO0_2。支持输入干扰滤波器 (10 ns)。
					I/O	-	**PIO0_5** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					I	-	**VDDCMP** — 模拟比较器交替基准电压。
					O	-	**CT16B0_MAT2** — 16 位定时器 0 的匹配输出 2。
					I/O	-	**SCK0** — SSP0 的串行时钟。支持输入干扰滤波器 (10 ns)。
TDI/PIO0_6/AD0/ CT32B0_MAT3/MISO0	32	21	C3	[7]	I	I; PU	**TDI** — JTAG 接口的测试数据输入。支持输入干扰滤波器 (10 ns)。
					I/O	-	**PIO0_6** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					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。
					I/O	-	**MISO0** — SSP0 的主机输入从机输出。支持输入干扰滤波器 (10 ns)。
TMS/PIO0_7/AD1/ CT32B1_CAP0/ CT16B0_MAT0	33	22	C4	[7]	I	I; PU	**TMS** — JTAG 接口的测试模式选择。支持输入干扰滤波器 (10 ns)。
					I/O	-	**PIO0_7** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					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。支持输入干扰滤波器 (10 ns)。
					O	-	**CT16B0_MAT0** — 16 位定时器 0 的匹配输出 2。
TDO/PIO0_8/AD2/ CT32B1_MAT0/SCK1	34	23	C2	[7]	O	I; PU	**TDO** — JTAG 接口的测试数据输出。
					I/O	-	**PIO0_8** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					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。
					I/O	-	**SCK1** — SSP1 的串行时钟。支持输入干扰滤波器 (10 ns)。
$\overline{\text{TRST}}$/PIO0_9/AD3/ CT32B1_MAT1/ $\overline{\text{CT}}$ CT16B0_MAT1/CTS	35	24	D4	[7]	I	I; PU	**$\overline{\text{TRST}}$** — JTAG 接口的测试重置。支持输入干扰滤波器 (10 ns)。
					I/O	-	**PIO0_9** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					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。
					O	-	**CT16B0_MAT1** — 16 位定时器 0 的匹配输出 1。
					I	-	**$\overline{\text{CTS}}$** — USART 的“准许发送”输入。支持输入干扰滤波器 (10 ns)。

表 86. LPC11A1x 引脚描述表 *（续）*

符号	引脚 LQFP48	HVQFN33	WLCSP20		类型	复位状态 [1]	描述
SWDIO/PIO0_10/AD4/ CT32B1_MAT2/ CT16B0_MAT2/RTS	38	25	D3	[7]	I/O	I; PU	**SWDIO** — LQFP48 封装和 HVQFN33 封装的初级（默认）串行调试接口 I/O。对于 WLCSP20 封装，则使用 PIO0_3。支持输入干扰滤波器 (10 ns)。
					I/O	-	**PIO0_10** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					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。
					O	-	**CT16B0_MAT2** — 16 位定时器 0 的匹配输出 2。
					O	-	**$\overline{\text{RTS}}$** — USART 的 “请求发送” 输出。
PIO0_11/SCLK/ AD5/CT32B1_MAT3/ CT32B0_CAP0	39	26	D2	[7]	I/O	I; PU	**PIO0_11** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					I/O	-	**SCLK** — USART 的串行时钟。支持输入干扰滤波器 (10 ns)。
					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	-	**CT32B0_CAP0** — 32 位定时器 0 的捕获输入 0。支持输入干扰滤波器 (10 ns)。
PIO0_12/RXD/ ACMP_O/ CT32B0_MAT0/SCL/ CLKIN	46	31	E1	[3]	I/O	I; PU	**PIO0_12** — 通用数字输入 / 输出引脚。
					I	-	**RXD** — USART 的接收器数据输入。该引脚用于 ISP 通信。
					O	-	**ACMP_O** — 模拟比较器输出。
					O	-	**CT32B0_MAT0** — 32 位定时器 0 的匹配输出 0。
					I/O	-	**SCL** — I^2C总线时钟输入/输出。这不是一个I^2C总线开漏引脚[8]。
					I	-	**CLKIN** — 外部时钟输入。
PIO0_13/TXD/ ACMP_I2/ CT32B0_MAT1/SDA	47	32	D1	[7]	I/O	I; PU	**PIO0_13** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					O	-	**TXD** — USART 的发送器数据输出。该引脚用于 ISP 通信。
					I	-	**ACMP_I2** — 模拟比较器输入 2。
					O	-	**CT32B0_MAT1** — 32 位定时器 0 的匹配输出 1。
					I/O	-	**SDA** — I^2C 总线数据输入 / 输出。这不是一个 I^2C 总线开漏引脚 [8]。支持输入干扰滤波器 (10 ns)。
PIO0_14/MISO1/AD6/ CT32B0_CAP1/ CT16B1_MAT1/ VDDCMP	30	20	-	[5]	I/O	I; PU	**PIO0_14** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					I/O	-	**MISO1** — SSP1 的主机输入从机输出。支持输入干扰滤波器 (10 ns)。
					I	-	**AD6** — A/D 转换器，输入 6。
					I	-	**CT32B0_CAP1** — 32 位定时器 0 的捕获输入 1。支持输入干扰滤波器 (10 ns)。
					O	-	**CT16B1_MAT1** — 16 位定时器 1 的匹配输出 1。
					I	-	**VDDCMP** — 模拟比较器交替基准电压。

表 86. LPC11A1x 引脚描述表 *（续）*

符号	引脚				类型	复位状态 [1]	描述
	LQFP48	HVQFN33	WLCSP20				
PIO0_14/MISO1/AD6/ CT32B0_CAP1/ CT16B1_MAT1	-	-	B4	[7]	I/O	I; PU	**PIO0_14** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					I/O	-	**MISO1** — SSP1 的主机输入从机输出。支持输入干扰滤波器 (10 ns)。
					I	-	**AD6** — A/D 转换器，输入 6。
					I	-	**CT32B0_CAP1** — 32 位定时器 0 的捕获输入 1。支持输入干扰滤波器 (10 ns)。
					O	-	**CT16B1_MAT1** — 16 位定时器 1 的匹配输出 1。
PIO0_15/TXD/AD7/ CT32B0_CAP2/SDA	41	27	E4	[7]	I/O	I; PU	**PIO0_15** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					O	-	**TXD** — USART 的发送器数据输出。
					I	-	**AD7** — A/D 转换器，输入 7。
					I	-	**CT32B0_CAP2** — 32 位定时器 0 的捕获输入 2。支持输入干扰滤波器 (10 ns)。
					I/O	-	**SDA** — I^2C 总线数据输入 / 输出。这不是一个 I^2C 总线开漏引脚 [8]。支持输入干扰滤波器 (10 ns)。
PIO0_16/ ATRG0/ACMP_I3/ CT16B0_CAP1/SCL	18	13	A2	[7]	I/O	I; PU	**PIO0_16** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					I	-	**ATRG0** — ADC 或 DAC 的转换触发 0。支持输入干扰滤波器 (10 ns)。
					I	-	**ACMP_I3** — 模拟比较器输入 3。
					I	-	**CT16B0_CAP1** — 16 位定时器 0 的捕获输入 1。支持输入干扰滤波器 (10 ns)。
					I/O	-	**SCL** — I^2C 总线时钟输入 / 输出。这不是一个 I^2C 总线开漏引脚 [8]。支持输入干扰滤波器 (10 ns)。
PIO0_17/ ATRG1/ACMP_I4/ CT16B0_CAP2/ CT16B0_MAT0	21	14	A3	[7]	I/O	I; PU	**PIO0_17** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					I	-	**ATRG1** — ADC 或 DAC 的转换触发 1。支持输入干扰滤波器 (10 ns)。
					I	-	**ACMP_I4** — 模拟比较器输入 4。
					I	-	**CT16B0_CAP2** — 16 位定时器 0 的捕获输入 2。支持输入干扰滤波器 (10 ns)。
					O	-	**CT16B0_MAT0** — 16 位定时器 0 的匹配输出 0。
PIO0_18/R/SSEL0/ CT16B0_CAP0/ CT16B1_CAP1	10	8	-	[3]	I/O	I; PU	**PIO0_18** — 通用数字输入 / 输出引脚。
					-	-	**R** — 保留。
					I/O	-	**SSEL0** — SSP0 的从机选择。
					I	-	**CT16B0_CAP0** — 16 位定时器 0 的捕获输入 0。
					I	-	**CT16B1_CAP1** — 16 位定时器 1 的捕获输入 1。

表 86. LPC11A1x 引脚描述表 *（续）*

符号	引脚				类型	复位状态 [1]	描述
	LQFP48	HVQFN33	WLCSP20				
PIO0_19/CLKIN/ CLKOUT/ MOSI0/CT16B1_MAT0	14	9	-	[3]	I/O	I; PU	**PIO0_19** — 通用数字输入 / 输出引脚。
					I	-	**CLKIN** — 外部时钟输入。
					O	-	**CLKOUT** — 时钟输出。
					I/O	-	**MOSI0** — SSP0 的主机输出从机输入。
					O	-	**CT16B1_MAT0** — 16 位定时器 1 的匹配输出 0。
PIO0_20/R/SCK0/ CT32B1_CAP0/ CT16B1_MAT2	22	15	-	[3]	I/O	I; PU	**PIO0_20** — 通用数字输入 / 输出引脚。
					-	-	**R** — 保留。
					I/O	-	**SCK0** — SSP0 的串行时钟。
					I	-	**CT32B1_CAP0** — 32 位定时器 1 的捕获输入 0。
					O	-	**CT16B1_MAT2** — 16 位定时器 1 的匹配输出 2。
PIO0_21/$\overline{\text{CTS}}$/ ACMP_O/ CT32B1_CAP1/SCLK	23	16	-	[3]	I/O	I; PU	**PIO0_21** — 通用数字输入 / 输出引脚。如果配置为输出，则该引脚为大电流源输出驱动器 (20 mA)。
					I	-	**$\overline{\text{CTS}}$** — USART 的 “准许发送” 输入。
					O	-	**ACMP_O** — 模拟比较器输出。
					I	-	**CT32B1_CAP1** — 32 位定时器 1 的捕获输入 1。
					I/O	-	**SCLK** — USART 的串行时钟。
PIO0_22/MISO0/ ACMP_I5/ CT32B1_MAT2/ CT32B1_CAP2	27	17	-	[3]	I/O	I; PU	**PIO0_22** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					I/O	-	**MISO0** — SSP0 的主机输入从机输出。支持输入干扰滤波器 (10 ns)。
					I	-	**ACMP_I5** — 模拟比较器输入 5。
					O	-	**CT32B1_MAT2** — 32 位定时器 1 的匹配输出 2。
					I	-	**CT32B1_CAP2** — 32 位定时器 1 的捕获输入 2。支持输入干扰滤波器 (10 ns)。
PIO0_23/$\overline{\text{RTS}}$/ ACMP_O/ CT32B0_CAP0/SCLK	45	30	-	[3]	I/O	I; PU	**PIO0_23** — 通用数字输入 / 输出引脚。
					O	-	**$\overline{\text{RTS}}$** — USART 的 “请求发送” 输出。
					O	-	**ACMP_O** — 模拟比较器输出。
					I	-	**CT32B0_CAP0** — 32 位定时器 0 的捕获输入 0。
					I/O	-	**SCLK** — USART 的串行时钟。
PIO0_24/SCL/CLKIN/ CT16B1_CAP0	9	7	-	[3]	I/O	I; PU	**PIO0_24** — 通用数字输入 / 输出引脚。
					I/O	-	**SCL** — I^2C总线时钟输入/输出。这不是一个I^2C总线开漏引脚[8]。
					I	-	**CLKIN** — 外部时钟输入。
					I	-	**CT16B1_CAP0** — 16 位定时器 1 的捕获输入 0。
PIO0_25/SDA/SSEL1/ CT16B1_MAT0	17	12	-	[3]	I/O	I; PU	**PIO0_25** — 通用数字输入 / 输出引脚。
					I/O	-	**SDA** — I^2C总线数据输入/输出。这不是一个I^2C总线开漏引脚[8]。
					I/O	-	**SSEL1** — SSP1 的从机选择。
					O	-	**CT16B1_MAT0** — 16 位定时器 1 的匹配输出 0。

表 86. LPC11A1x 引脚描述表 *（续）*

符号	引脚				类型	复位状态 [1]	描述
	LQFP48	HVQFN33	WLCSP20				
PIO0_26/TXD/MISO1/ CT16B1_CAP1/ CT32B0_CAP2	1	1	-	[3]	I/O	I; PU	**PIO0_26** — 通用数字输入 / 输出引脚。
					O	-	**TXD** — USART 的发送器数据输出。
					I/O	-	**MISO1** — SSP1 的主机输入从机输出。
					I	-	**CT16B1_CAP1** — 16 位定时器 1 的捕获输入 1。
					I	-	**CT32B0_CAP2** — 32 位定时器 0 的捕获输入 2。
PIO0_27/MOSI1/ ACMP_I1/ CT32B1_MAT1/ CT16B1_CAP2	43	28	-	[3]	I/O	I; PU	**PIO0_27** — 通用数字输入 / 输出引脚。支持输入干扰滤波器 (10 ns)。
					I/O	-	**MOSI1** — SSP1 的主机输出从机输入。支持输入干扰滤波器 (10 ns)。
					I	-	**ACMP_I1** — 模拟比较器输入 1。
					O	-	**CT32B1_MAT1** — 32 位定时器 1 的匹配输出 1。
					I	-	**CT16B1_CAP2** — 16 位定时器 1 的捕获输入 2。支持输入干扰滤波器 (10 ns)。
PIO0_28/$\overline{\text{DTR}}$/SSEL1/ CT32B0_CAP0	2	-	-	[3]	I/O	I; PU	**PIO0_28** — 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。
					I	-	**CT32B0_CAP0** — 32 位定时器 0 的捕获输入 0。
PIO0_29/$\overline{\text{DSR}}$/SCK1/ CT32B0_CAP1	13	-	-	[3]	I/O	I; PU	**PIO0_29** — 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。
					I	-	**CT32B0_CAP1** — 32 位定时器 0 的捕获输入 1。
PIO0_30/$\overline{\text{RI}}$/MOSI1/ CT32B0_MAT0/ CT16B0_CAP0	40	-	-	[3]	I/O	I; PU	**PIO0_30** — 通用数字输入 / 输出引脚。
					I	-	**$\overline{\text{RI}}$** — USART 的振铃指示器输入。
					I/O	-	**MOSI1** — SSP1 的主机输出从机输入。
					O	-	**CT32B0_MAT0** — 32 位定时器 0 的匹配输出 0。
					I	-	**CT16B0_CAP0** — 16 位定时器 0 的捕获输入 0。
PIO0_31/$\overline{\text{RI}}$/MOSI1/ CT32B1_MAT0/ CT16B1_CAP1	24	-	-	[3]	I/O	I; PU	**PIO0_31** — 通用数字输入 / 输出引脚。
					I	-	**$\overline{\text{RI}}$** — USART 的振铃指示器输入。
					I/O	-	**MOSI1** — SSP1 的主机输出从机输入。
					O	-	**CT32B1_MAT0** — 32 位定时器 1 的匹配输出 0。
					I	-	**CT16B1_CAP1** — 16 位定时器 1 的捕获输入 1。
PIO1_0/$\overline{\text{DCD}}$/SCK0/ CT32B1_MAT3/ CT16B0_MAT1	31	-	-	[3]	I/O	I; PU	**PIO1_0** — 通用数字输入 / 输出引脚。
					I	-	**$\overline{\text{DCD}}$** — USART 的 “数据载波检测” 输入。
					I/O	-	**SCK0** — SSP0 的串行时钟。
					O	-	**CT32B1_MAT3** — 32 位定时器 1 的匹配输出 3。
					O	-	**CT16B0_MAT1** — 16 位定时器 0 的匹配输出 1。

表 86. LPC11A1x 引脚描述表 *（续）*

符号	引脚 LQFP48	HVQFN33	WLCSP20		类型	复位状态 [1]	描述
PIO1_1/DTR/SSEL0/ CT32B1_MAT3/ CT16B1_MAT0	36	-	-	[3]	I/O	I; PU	**PIO1_1** — 通用数字输入 / 输出引脚。
					O	-	**DTR** — USART 的 “数据终端就绪” 输出。
					I/O	-	**SSEL0** — SSP0 的从机选择。
					O	-	**CT32B1_MAT3** — 32 位定时器 1 的匹配输出 3。
					O	-	**CT16B1_MAT0** — 16 位定时器 1 的匹配输出 0。
PIO1_2/DSR/MISO0/ CT16B1_MAT2/ CT16B1_MAT1	37	-	-	[3]	I/O	I; PU	**PIO1_2** — 通用数字输入 / 输出引脚。
					I	-	**DSR** — USART 的 “数据设置就绪” 输入。
					I/O	-	**MISO0** — SSP0 的主机输入从机输出。
					O	-	**CT16B1_MAT2** — 16 位定时器 1 的匹配输出 2。
					O	-	**CT16B1_MAT1** — 16 位定时器 1 的匹配输出 1。
PIO1_3/RI/MOSI0/ CT16B1_CAP0	48	-	-	[3]	I/O	I; PU	**PIO1_3** — 通用数字输入 / 输出引脚。
					I	-	**RI** — USART 的振铃指示器输入。
					I/O	-	**MOSI0** — SSP0 的主机输出从机输入。
					I	-	**CT16B1_CAP0** — 16 位定时器 1 的捕获输入 0。
PIO1_4/RXD/SSEL1/ CT32B0_MAT1/ CT32B1_CAP0/ CT16B0_CAP1	19	-	-	[3]	I/O	I; PU	**PIO1_4** — 通用数字输入 / 输出引脚。
					I	-	**RXD** — USART 的接收器数据输入。
					I/O	-	**SSEL1** — SSP1 的从机选择。
					O	-	**CT32B0_MAT1** — 32 位定时器 0 的匹配输出 1。
					I	-	**CT32B1_CAP0** — 32 位定时器 1 的捕获输入 0。
					I	-	**CT16B0_CAP1** — 16 位定时器 0 的捕获输入 1。
PIO1_5/TXD/SCK1/ CT32B0_MAT2/ CT32B1_CAP1/ CT16B0_CAP2	20	-	-	[3]	I/O	I; PU	**PIO1_5** — 通用数字输入 / 输出引脚。
					O	-	**TXD** — USART 的发送器数据输出。
					I/O	-	**SCK1** — SSP1 的串行时钟。
					O	-	**CT32B0_MAT2** — 32 位定时器 0 的匹配输出 2。
					I	-	**CT32B1_CAP1** — 32 位定时器 1 的捕获输入 1。
					I	-	**CT16B0_CAP2** — 16 位定时器 0 的捕获输入 2。
PIO1_6/RTS/MOSI1/ CT32B0_MAT3/ CT32B1_CAP2/ CT16B0_MAT0	11	-	-	[3]	I/O	I; PU	**PIO1_6** — 通用数字输入 / 输出引脚。
					O	-	**RTS** — USART 的 “请求发送” 输出。
					I/O	-	**MOSI1** — SSP1 的主机输出从机输入。
					O	-	**CT32B0_MAT3** — 32 位定时器 0 的匹配输出 3。
					I	-	**CT32B1_CAP2** — 32 位定时器 1 的捕获输入 2。
					O	-	**CT16B0_MAT0** — 16 位定时器 0 的匹配输出 0。

表 86. LPC11A1x 引脚描述表 *（续）*

符号	引脚				类型	复位状态 [1]	描述
	LQFP48	HVQFN33	WLCSP20				
PIO1_7/CTS/MOSI0/ CT32B1_MAT1/ CT16B0_MAT2/ CT16B1_CAP2	25	-	-	[3]	I/O	I; PU	**PIO1_7** — 通用数字输入 / 输出引脚。
					I	-	**CTS** — USART 的 “准许发送” 输入。
					I/O	-	**MOSI0** — SSP0 的主机输出从机输入。
					O	-	**CT32B1_MAT1** — 32 位定时器 1 的匹配输出 1。
					O	-	**CT16B0_MAT2** — 16 位定时器 0 的匹配输出 2。
					I	-	**CT16B1_CAP2** — 16 位定时器 1 的捕获输入 2。
PIO1_8/RXD / MISO1/ CT32B1_MAT0/ CT16B1_MAT1	26	-	-	[3]	I/O	I; PU	**PIO1_8** — 通用数字输入 / 输出引脚。
					I	-	**RXD** — USART 的接收器数据输入。
					I/O	-	**MISO1** — SSP1 的主机输入从机输出。
					O	-	**CT32B1_MAT0** — 32 位定时器 1 的匹配输出 0。
					O	-	**CT16B1_MAT1** — 16 位定时器 1 的匹配输出 1。
PIO1_9/DCD/R/ CT32B1_MAT2 / CT16B1_MAT2	12	-	-	[3]	I/O	I; PU	**PIO1_9** — 通用数字输入 / 输出引脚。
					I	-	**DCD** — USART 的 “数据载波检测” 输入。
					-	-	**R** — 保留。
					O	-	**CT32B1_MAT2** — 32 位定时器 1 的匹配输出 2。
					O	-	**CT16B1_MAT2** — 16 位定时器 1 的匹配输出 2。
XTALIN	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	7	5	-	[9]	-	-	振荡器放大器的输出。
$V_{DD(IO)}$	8	6	E2	[10] [11]	-	-	3.3 V 输入 / 输出电源电压。
$V_{SS(IO)}$	5	33	E3	[12]	-	-	接地。
$V_{DD(3V3)}$	44	29	E2	[10] [11]	-	-	模拟模块、内部调压器和内部时钟生成器电路的3.3 V电源电压。也用作 ADC 基准电压。
V_{SS}	42	33	E3	[12]	-	-	接地。

[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。

[2] 有关复位配置，请参见图 7。

[3] 容压为 5V 的引脚，提供带可配置模式和可配置滞回的标准数字 I/O 功能 （图 6）。

[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] 引脚具有特殊模拟功能，因此容压并非 5V。引脚提供具有可配置模式、可配置滞回以及模拟 I/O 的标准数字 I/O 功能。配置为模拟 I/O 时，引脚的数字部分会被禁用 （图 6）。

[6] 如果该引脚针对其 VDDCMP 功能进行配置，则该器件在线路板上时无法用于 SWCLK。电源的旁通滤波器滤出 SWCLK 时钟输入信号。

[7] 容压为 5V 的引脚，提供具有可配置模式、可配置滞回以及模拟 I/O 的标准数字 I/O 功能。配置为模拟 I/O 时，引脚的数字部分会被禁用且该引脚的容压并非 5 V （图 6）。

[8] I^2C 总线引脚为标准数字 I/O 引脚，与 I^2C 总线完整规范相比，性能和电气特性受限。引脚可配置为具有一个片内上拉电阻（pMOS 器件）和开漏模式。在该模式下，如果使能内部上拉电阻，则支持负载为 20 pF 时高达 100 kbit/s 的典型比特率。通过外部电阻可实现更高的比特率。

[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。

[10] 如果独立的电源用于 $V_{DD(3V3)}$ 和 $V_{DD(IO)}$，则确保根据相应的接地电压 V_{SS} 和 $V_{SS(IO)}$ 对电源引脚进行噪声滤波（LQFP48 封装）。使用独立的滤波电源可降低模拟模块的噪声。

[11] 如果独立的电源用于 $V_{DD(3V3)}$ 和 $V_{DD(IO)}$，则确保两个电源之间的电压差不超过 0.5 V。

[12] 热焊盘 （HVQFN33 引脚封装）。接地。

8.1 本章导读

所有 LPC11Axx 器件中的 USART 模块都是一样的。调制解调器控制信号是否可用取决于封装和 I/O 配置模块的设置。参见表 79 到表 84 范围内的适用表。

8.2 基本配置

USART 按如下方式配置：

- 引脚：在使能USART时钟之前，必须在相应的IOCON寄存器中对USART引脚进行配置（参见 第 6.4 节）。
- 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。

8.3 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。
- 可选 $\overline{\text{RTS}}$/$\overline{\text{CTS}}$ 流控制和其他调制解调器控制信号。
- 可选 1X- 时钟发送或接收。
- 符合 ISO 7816-3 标准的可选智能卡接口。

8.4 引脚说明

以下部分引脚不适用某些封装。

表 87. 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。
$\overline{\text{CTS}}$	输入	**准许发送。**
$\overline{\text{DTR}}$	输出	**数据终端就绪。**
$\overline{\text{DSR}}$	输入	**数据设置就绪。**

表 87. USART 引脚描述 *（续）*

引脚	类型	描述
$\overline{\text{DCD}}$	输入	**数据载波检测。**
$\overline{\text{RI}}$	输入	**振铃指示器。**
SCLK	I/O	**串行时钟。**

8.5 寄存器描述

USART 所包含的寄存器，其结构如表 88 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 88 中未列出的偏移 / 地址**保留**。

表 88. 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]
RBR	RO	0x0	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
THR	WO	0x0	发送保持寄存器。下一个待发送字符写入在此。 (DLAB=0)	不适用
DLL	R/W	0x0	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
DLM	R/W	0x4	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
IER	R/W	0x4	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
IIR	RO	0x8	中断 ID 寄存器，识别要挂起的中断。	0x01
FCR	WO	0x8	FIFO 控制寄存器，控制 USART FIFO 的使用和模式。	0
LCR	R/W	0xC	线路控制寄存器。包含对帧格式化和断点产生的控制。	0
MCR	R/W	0x10	调制解调器控制寄存器。	0
LSR	RO	0x14	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
MSR	RO	0x18	调制解调器状态寄存器。	0
SCR	R/W	0x1C	暂存寄存器，软件的八位临时存储器。	0
ACR	R/W	0x20	自动波特率控制寄存器，包含自动波特率功能的控制。	0
ICR	R/W	0x24	IrDA 控制寄存器。使能和配置 IrDA （远程控制）模式。	0
FDR	R/W	0x28	小数分频寄存器，为波特率分频器生成时钟输入。	0x10
OSR	R/W	0x2C	过采样寄存器。控制每个位时间内的过采样度。	0xF0
TER	R/W	0x30	发送使能寄存器。关闭 USART 发送器，以使用软件流控制。	0x80
SCICTRL	R/W	0x48	智能卡接口控制寄存器。使能和配置智能卡接口功能。	0
RS485CTRL	R/W	0x4C	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
ADRMATCH	R/W	0x50	RS-485/EIA-485 地址匹配，包含 RS-485/EIA-485 模式的地址匹配值。	0
RS485DLY	R/W	0x54	RS-485/EIA-485 方向控制延迟。	0
SYNCCTRL	R/W	0x58	同步模式控制寄存器。	0

[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。

8.5.1 USART 接收器缓冲寄存器 （DLAB = 0，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 100）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 89. USART 接收器缓冲寄存器 （当 DLAB = 0 时）（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	-

8.5.2 USART 发送器保持寄存器 （DLAB = 0，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 90. USART 发送器保持寄存器 （当 DLAB = 0 时）（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	-

8.5.3 USART 除数锁存器 LSB 和 MSB 寄存器 (DLAB = 1)

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 8.5.16 节。

表 91. 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 92. USART 除数锁存器 MSB 寄存器 （DLM - 地址 0x4000 8004，当 DLAB = 1 时）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	-

8.5.4 USART 中断使能寄存器 (DLAB = 0)

IER 用于使能不同 USART 中断源。

表 93. 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	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	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

8.5.5 USART 中断识别寄存器

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 94. USART 中断识别寄存器 （IIR - 地址 0x4004 8008，只读）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] 由自动波特率函数设置，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 95 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 95. 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 + [（触发级别 - 字符数）× 8 + 1] RCLK	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 8.5.9 节 “USART 线路状态寄存器 （LSR - 0x4000 8014，只读）”。

[3] 有关详情，请参阅 第 8.5.1 节 “USART 接收器缓冲寄存器 （DLAB = 0，只读）”。

[4] 有关详情，请参阅 第 8.5.5 节 “USART 中断识别寄存器” 和 第 8.5.2 节 “USART 发送器保持寄存器（DLAB = 0，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。

8.5.6 USART FIFO 控制寄存器

FCR 控制 USART RX 和 TX FIFO 的操作。

表 96. USART FIFO 控制寄存器 （FCR - 地址 0x4000 8008，只写）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 必须接收多少接收器的 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	-

8.5.7 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 97. 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	DTRCON		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	RTSCON		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。$\overline{\text{DSR}}$、$\overline{\text{CTS}}$、$\overline{\text{DCD}}$ 和 $\overline{\text{RI}}$ 引脚均被忽略。从外部看来，$\overline{\text{DTR}}$ 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。	

表 97. USART 调制解调器控制寄存器 （MCR - 地址 0x4000 8010）位描述 *（续）*

位	符号	值	描述	复位值
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	-

8.5.7.1 自动流控制

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

8.5.7.1.1 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。如果使能自动 RTS 模式，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，那么如果使能自动 RTS，则接收 FIFO 只要包含 8 个字节（第 82 页的表 96），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。

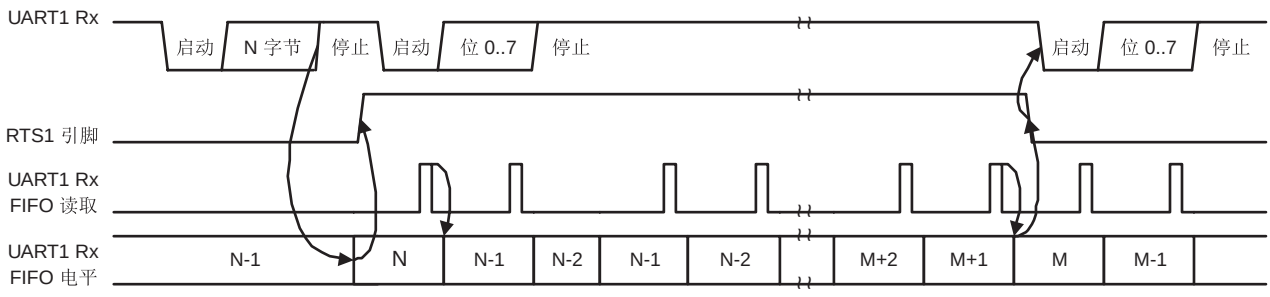


图 11. 自动 RTS 功能时序

8.5.7.1.2 自动 CTS

通过设置 CTSen 位可以使能自动 CTS 功能。如果使能自动 CTS，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 98 列出了生成调制解调器状态中断的条件。

表 98. 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 12 展示了自动 CTS 功能时序。

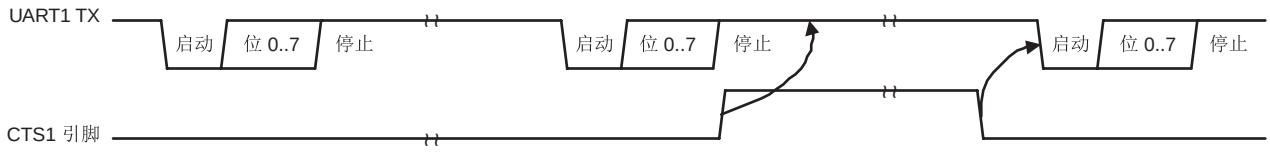


图 12. 自动 CTS 功能时序

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 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。

8.5.8 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 99. 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。	
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	-

8.5.9 USART 线路状态寄存器 （LSR - 0x4000 8014，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 100. USART 线路状态寄存器 （LSR - 地址 0x4000 8014，只读）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		溢出错误。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	溢出错误状态未激活。	
		1	溢出错误状态激活。	
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，出现成帧处理错误。 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。	

表 100. USART 线路状态寄存器 （LSR - 地址 0x4000 8014，只读）位描述 *（续）*

位	符号	值	描述	复位值
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	-

8.5.10 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 101. 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 的补值。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

8.5.11 USART 暂存寄存器 （SCR - 0x4000 801C）

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 102. 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	-

8.5.12 USART 自动波特率控制寄存器 (ACR - 0x4000 8020)

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 103. 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

8.5.13 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。

自动波特率函数可产生两种中断。

- 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。

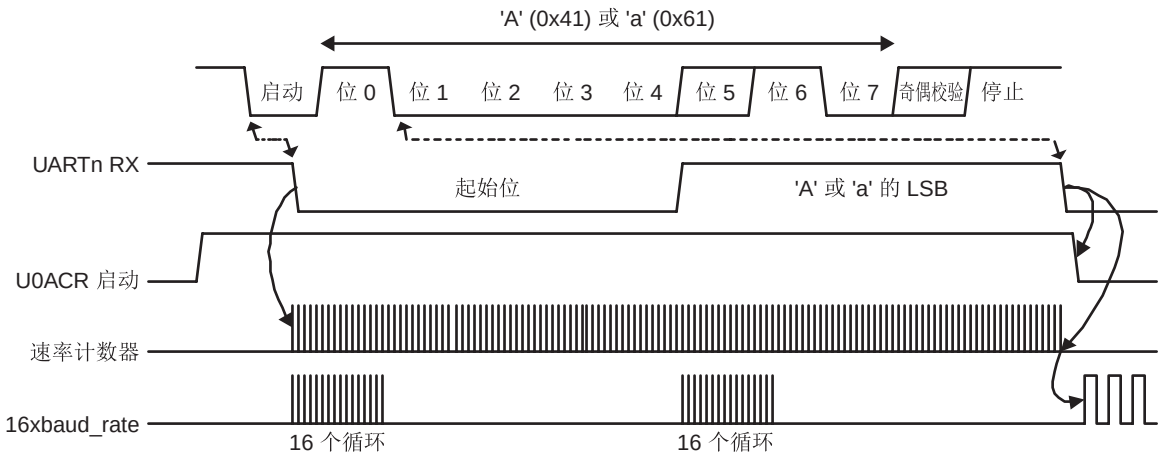
小数波特率发生器在自动波特率期间必须禁用 (DIVADDVAL = 0)。同样，使用自动波特率时，对 DLM 和 DLL 寄存器的任何写操作都必须在 ACR 寄存器写操作前完成。 USART 支持的最小和最大波特率是 UART_PCLK、数据位、停止 位和奇偶校验位的数量的函数。

$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax \quad (1)$$

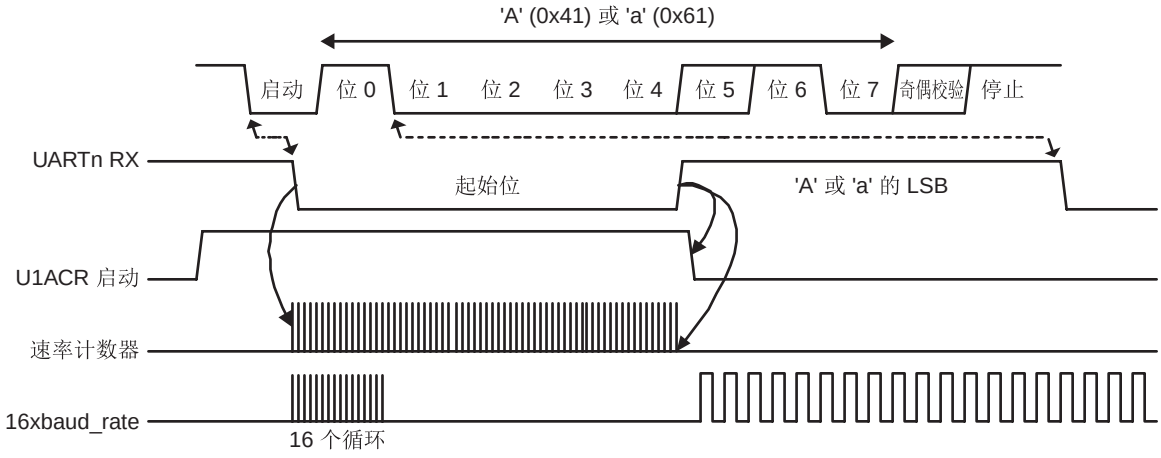
8.5.14 自动波特率模式

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 13. 自动波特率 a) 模式 0 和 b) 模式 1 的波形图

8.5.15 IrDA 控制寄存器 (ICR – 0x4000 8024)

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 104. 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	USART 上的 IrDA 模式禁用，USART 用作标准的 USART。	
		1	USARTn 上的 IrDA 模式使能。	
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 105 显示了可能的脉冲宽度。

表 105. 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	$256 \times T_{PCLK}$

8.5.16 USART 小数分频寄存器 (FDR - 0x4000 8028)

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 106. USART 小数分频寄存器 （FDR - 地址 0x4000 8028）位描述

位	函数	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：

$$UART_{baudrate} = \frac{PCLK}{16 \times (256 \times U0DLM + U0DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)} \quad (2)$$

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。

8.5.16.1 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颇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%。

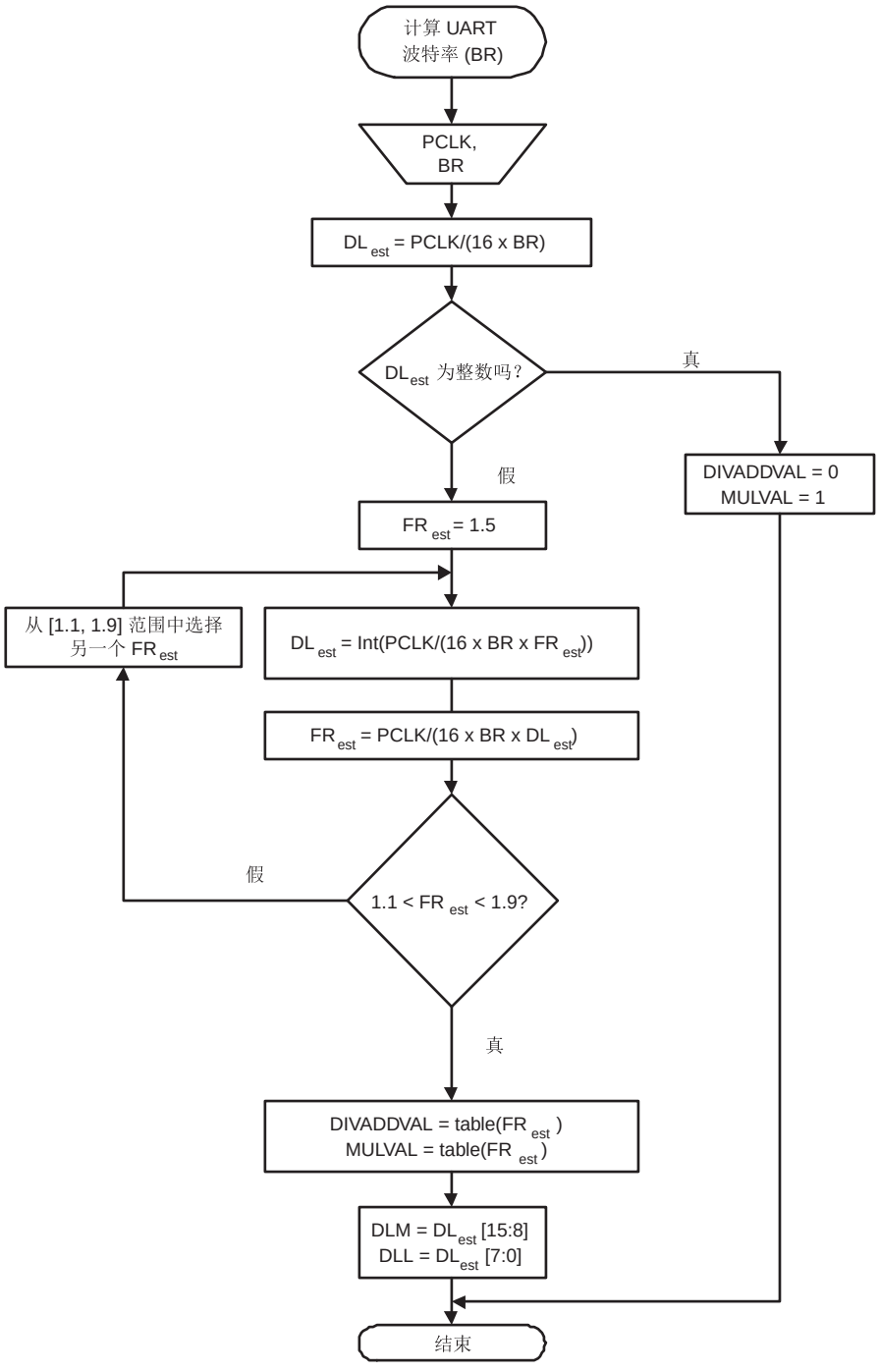


图 14. 设置 USART 分频器的算法

表 107. 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

8.5.16.1.1 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。

8.5.16.1.2 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 107 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 2， 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。

8.5.17 USART 过采样寄存器 (OSR - 0x4000 802C)

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 108. 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）。

8.5.18 USART 发送使能寄存器

该寄存器可实现软件流控制。 TXEN = 1 时，只要数据可用， USART 发送器将连续发送数据。 TXEN 变为 0 时， USART 发送将立即在当前字符的末尾停止。

尽管表 109 描述了如何利用 TXEN 位来实现硬件流控制，但如果信号可用，则最好使用 USART 硬件所实现的自动流控制功能，并且仅在进行软件流控制时使用 TXEN。

表 109. USART 发送使能寄存器 （TER - 地址 0x4000 805C）位描述

位	符号	描述	复位值
6:0	-	保留	-
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。当检测到硬件信号交换的 Tx-permit 信号 （通常为 CTS）变为假时，或者在接收到 XOFF 字符 (DC3) 时，软件通过执行软件信号交换可将该位清除。当检测到 Tx-permit 信号变为真时，或者在接收到 XON (DC1) 字符时，软件可以重新设置该位。	1
31:1	-	保留	-

8.5.19 智能卡接口控制寄存器 (SCICTRL - 0x4000 8048)

该寄存器允许在异步智能卡应用中使用 USART。

表 110. 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	
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

8.5.19.1 智能卡连接

SCICTRL 寄存器中的 SCIEN 位按上述进行设置后， USART 在 TXD 引脚上提供双向串行数据。 SCIEN 为 1 时不使用 RXD 引脚。如果在 I/O 配置模块中使能 USART SCLK 功能，则会在引脚上输出串行时钟：对于智能卡来说可选择性使用此类时钟。软件必须使用定时器来实现字符和时钟等待时间（LPC11Axx 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。

8.5.19.2 智能卡设置

必须在智能卡应用中进行以下设置：

- 如果需要，可编程 PRESETCTRL （第 3.5.2 节），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（或可能）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 （第 3.5.16 节）。
- 为 372x 过采样编程 OSR （第 8.5.17 节）。
- 必要时，分别将 DLM 和 DLL（第 8.5.3 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 8.5.8 节）。
- 编程与智能卡相关的 GPIO 信号，以便 （按以下顺序）：
  - 复位为低电平。

- 向卡提供 VCC （GPIO 引脚不含所需的 200 mA 驱动）。
- VPP （如果提供给卡）位于 “空闲” 状态。

- 编程 SCICTRL （第 8.5.19 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 （第 3.5.14 节）以使能 USART 模块。

此后，软件应监控 USART 和定时器的状态，以便与 ISO 7816 3.2.b 及之后所述的智能卡进行交互。

8.5.20 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 111. 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 RTS 用于方向控制。	
		1	如果方向控制已使能 （位 DCTRL = 1），则引脚 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

8.5.21 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 112. 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	-

8.5.22 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 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 113. USART RS-485 延迟值寄存器 （RS485DLY - 地址 0x4000 80454）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

8.5.23 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。

8.5.24 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 114. 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		发送同步寄存器旁通。	0
		0	< 待定 >	
		1	< 待定 >	
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。	
31:6	-		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。）

LPC11Axx USART 为时钟源时 (CSRC=1)，该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 LPC11Axx 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。

8.6 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 UART_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。

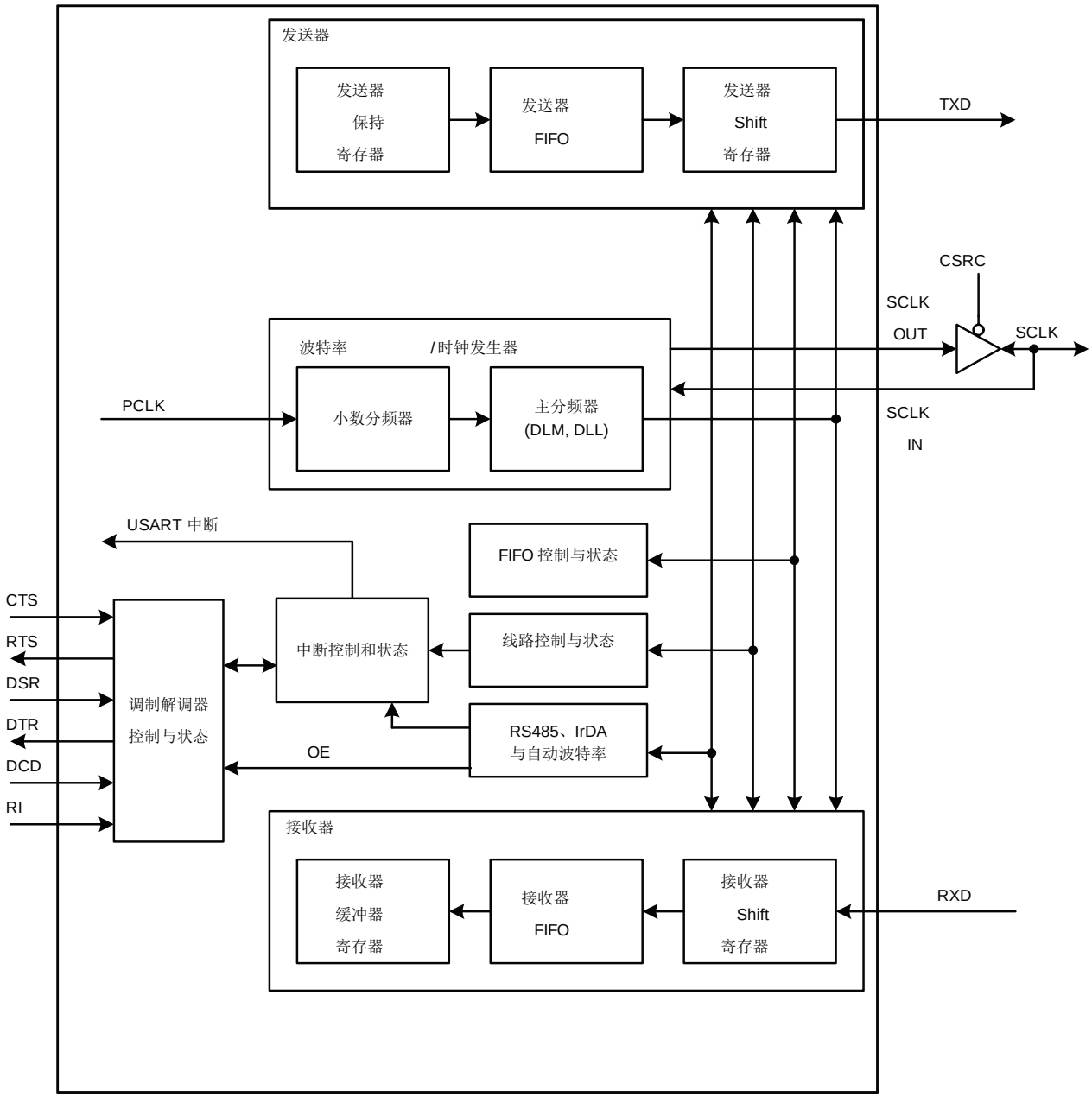


图 15. USART 框图

9.1 本章导读

所有 LPC11Axx 器件中的 I²C 总线模块都是一样的。

9.2 基本配置

I²C 总线接口是使用以下寄存器进行配置的：

1. 主 I2C 引脚 （PIO0_2 和 PIO0_3）：在 IOCONFIG 寄存器模块中配置主 I2C 引脚的 I2C 引脚功能和 I2C 模式 （表 81）。
2. 备用 I2C 引脚：在 IOCONFIG 寄存器模块中配置备用 I2C 引脚的 I2C 引脚功能和 I2C 模式 （表 79）。
3. 电源和外设时钟：在 SYSAHBCLKCTRL 寄存器中，设置位 5 （表 19）。

9.3 特性

- 标准 I²C 兼容总线接口，可配置为主机、从机或主 / 从机。
- 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 I²C 传输速率。
- 主从接口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。
- 支持超快速模式。
- 可以选择识别多达四个不同的从机地址。
- 监控模式可观察所有的 I²C 总线流量，而不用考虑从机地址。
- I²C 总线可用于测试和诊断。
- 两个引脚包含符合 I²C 标准的收发器。
- 时钟和数据功能各自都具有 3 个带伪开漏驱动器的备用引脚，可最大限度将 I²C 与其他必要功能相结合。

9.4 应用

到外部 I²C 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。

9.5 简介

典型的 I²C 总线配置如图 16 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机地址，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 （从机地址）由主机发送，从机返回确认位。接下来，数据字节自从机发送到主机。主机在接收到每个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”。主器件生成串行时钟脉冲以及 “启动”和 “停止”条件。传输在 “停止”条件或在 “重复起动”条件下结束。 I²C 总线在 “停止”条件下被释放，但在 “重复起动”条件 （开始另一个串行传输）下不释放。

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。

通过表 115 中所示的 “主”引脚连接 I²C 时，电气特性完全符合 I²C 规范，其中包括了在不干扰 I²C 总线上其他器件的情况下使 LPC11Axx 掉电的功能。（备用引脚则无该功能。）

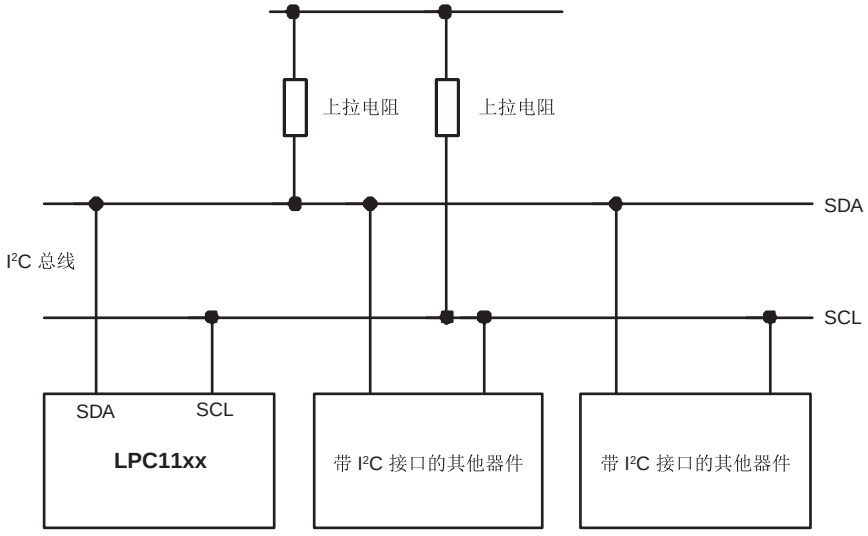


图 16. I²C 总线配置

9.5.1 I²C 超快速模式

超快速模式支持高达 1 Mbit/ 秒的传输。要使用超快速模式，则必须使用主 I²C 引脚并在 IOCONFIG 寄存器模块中进行正确配置。

9.6 引脚说明

如果主引脚用于 I²C，则应按照第 6.3.6 节所述对其 I/O 配置寄存器（表 81 和表 82）的 HS 字段和 HD 字段进行编程。如果使用备用引脚，则应设置其 I/O 配置寄存器（表 79 和表 83）的 OD 位，并可选择性地在 MODE 字段中选择内部上拉。使用内部上拉可能会导致上升时间与预期速度不相符，在该情况下，可以使用外部电阻，同时将 MODE 字段设置为“非工作”状态。

表 115. I²C 总线引脚描述

函数	主引脚	备用引脚	类型	描述
SCL	PIO0_2	PIO0_12、PIO0_16、PIO0_24	输入 / 输出	I²C 串行时钟
SDA	PIO0_3	PIO0_13、PIO0_15、PIO0_25	输入 / 输出	I²C 串行数据

9.6.1 外部上拉

备用引脚上的外部上拉电阻不应与高于 V_{DD} 的电压连接。该限制不适用于主引脚。

9.7 寄存器描述

表 116. 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]
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
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
DAT	R/W	0x008	**I2C 数据寄存器。**在主 / 从发送模式期间，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
ADR0	R/W	0x00C	**I2C 从机地址寄存器 0。**包含 7 位从机地址，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
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
SCLL	R/W	0x014	**SCL 占空比寄存器低半字。**决定 I²C 时钟低电平时间。 SCLL 和 SCLH 共同决定 I²C 主机产生的时钟频率及从机模式下所用的时间。	0x04
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
MMCTRL	R/W	0x01C	**监控模式控制寄存器。**	0x00
ADR1	R/W	0x020	**I2C 从机地址寄存器 1。**包含 7 位从机地址，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
ADR2	R/W	0x024	**I2C 从机地址寄存器 2。**包含 7 位从机地址，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
ADR3	R/W	0x028	**I2C 从机地址寄存器 3。**包含 7 位从机地址，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
DATA_BUFFER	RO	0x02C	**数据缓冲寄存器。**从总线每接收 9 位（8 位数据和 ACK 或 NACK）后，DAT 移位寄存器的 8 MSB 的内容将自动传输到 DATA_BUFFER。	0x00
MASK0	R/W	0x030	**I2C 从机地址屏蔽寄存器 0。**该屏蔽寄存器与 ADR0 共同决定地址匹配。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 116. 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]
MASK1	R/W	0x034	**I2C 从机地址屏蔽寄存器 1**。该屏蔽寄存器与 ADR0 共同决定地址匹配。与通用调用地址（‘0000000’）相比时，屏蔽寄存器不产生任何影响。	0x00
MASK2	R/W	0x038	**I2C 从机地址屏蔽寄存器 2**。该屏蔽寄存器与 ADR0 共同决定地址匹配。与通用调用地址（‘0000000’）相比时，屏蔽寄存器不产生任何影响。	0x00
MASK3	R/W	0x03C	**I2C 从机地址屏蔽寄存器 3**。该屏蔽寄存器与 ADR0 共同决定地址匹配。与通用调用地址（‘0000000’）相比时，屏蔽寄存器不产生任何影响。	0x00

[1] 重置值仅反映存储在所用位中的数据，不包括保留位的内容。

9.7.1 I²C 控制设置寄存器 (CONSET)

CONSET 寄存器控制 CON 寄存器中位的设置，该寄存器控制 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 117. 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
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

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 “重复起动”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 “不可寻址”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，且串行传输被挂起。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 是断言确认标志。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. 已接收到从机地址寄存器中的地址。
2. 在设置 ADR 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。

9.7.2 I²C 状态寄存器 (STAT)

I²C 状态寄存器反映 I²C 接口的情况。 I²C 状态寄存器为只读。

表 118. 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

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 133 ～表 138。

9.7.3 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 位进行了置位， DAT 中的数据就保持稳定不变。 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。

表 119. 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

9.7.4 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 的 LSB 为通用调用位。在此位设置后，会识别通用调用地址 (0x00)。

这些寄存器中，包含位 00x 的将被禁用，不会与总线上任何地址相匹配。复位时，从机地址寄存器将被清除为这一禁用状态。另请参见表 126。

表 120. I²C 从机地址寄存器 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

9.7.5 I²C SCL 高电平占空比寄存器和低电平占空比寄存器（SCLH - 0x4000 0010 和 SCLL- 0x4000 0014）

表 121. 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

表 122. 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

9.7.5.1 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）：

$$I^2C_{bitfrequency} = \frac{I2CPCLK}{SCLH + SCLL} \quad (3)$$

SCLL 和 SCLH 的值必须确保数据速率在适当的 I²C 数据速率范围内。各寄存器的值必须大于或等于 4。表 123 给出了根据 I2C_PCLK 频率和 SCLL 及 SCLH 值计算出来的 I²C 总线速率的示例。

表 123. 用于所选 I²C 时钟值的 SCLL + SCLH 值

I²C 模式	I²C 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，I²C 总线规范定义在快速模式和超快速模式 I²C 下不同值对应的 SCL 低电平时间和高电平时间。

9.7.6 I²C 控制清除寄存器 (CONCLR)

CONCLR 寄存器控制 CON 寄存器中位的清除，该寄存器控制 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清零。写入 0 无效。

表 124. I²C 控制清除寄存器 (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	I²C 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	起始标志清零位。	0
6	I2ENC	I²C 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

AAC 是断言确认清除位。将 1 写入此位会清除 CONSET 寄存器中的 AA 位。写入 0 无效。

SIC 是 I²C 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 I²C 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。

9.7.7 I²C 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 I²C 块在不需实际参与通信或干扰 I²C 总线的情况下监控 I²C 总线上的流量。

表 125. 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]	

表 125. I²C 监控模式控制寄存器 (MMCTRL - 0x4000 001C) 位描述 *（续）*

位	符号	值	描述	复位值
3	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。	

[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 9.7.9 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。

9.7.7.1 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。

9.7.7.2 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。

9.7.8 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 的 LSB 为通用调用位。在此位设置后，会识别通用调用地址 (0x00)。

这些寄存器中，包含位 00x 的将被禁用，不会与总线上任何地址相匹配。复位时，全部四个寄存器将被清除为这一禁用状态。

表 126. I²C 从机地址寄存器 (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

9.7.9 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，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 127. 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

9.7.10 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。这些位将始终回读为 0。

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 128. 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。此位始终回读为 0。	0
7:1	MASK	屏蔽位。	0x00
31:8	-	保留。用户软件不应向保留位写入 1。这些位始终回读为 0。	0

9.8 功能说明

图 17 所示为片内 I²C 总线接口的执行流程，下面章节将对图中各模块进行描述。

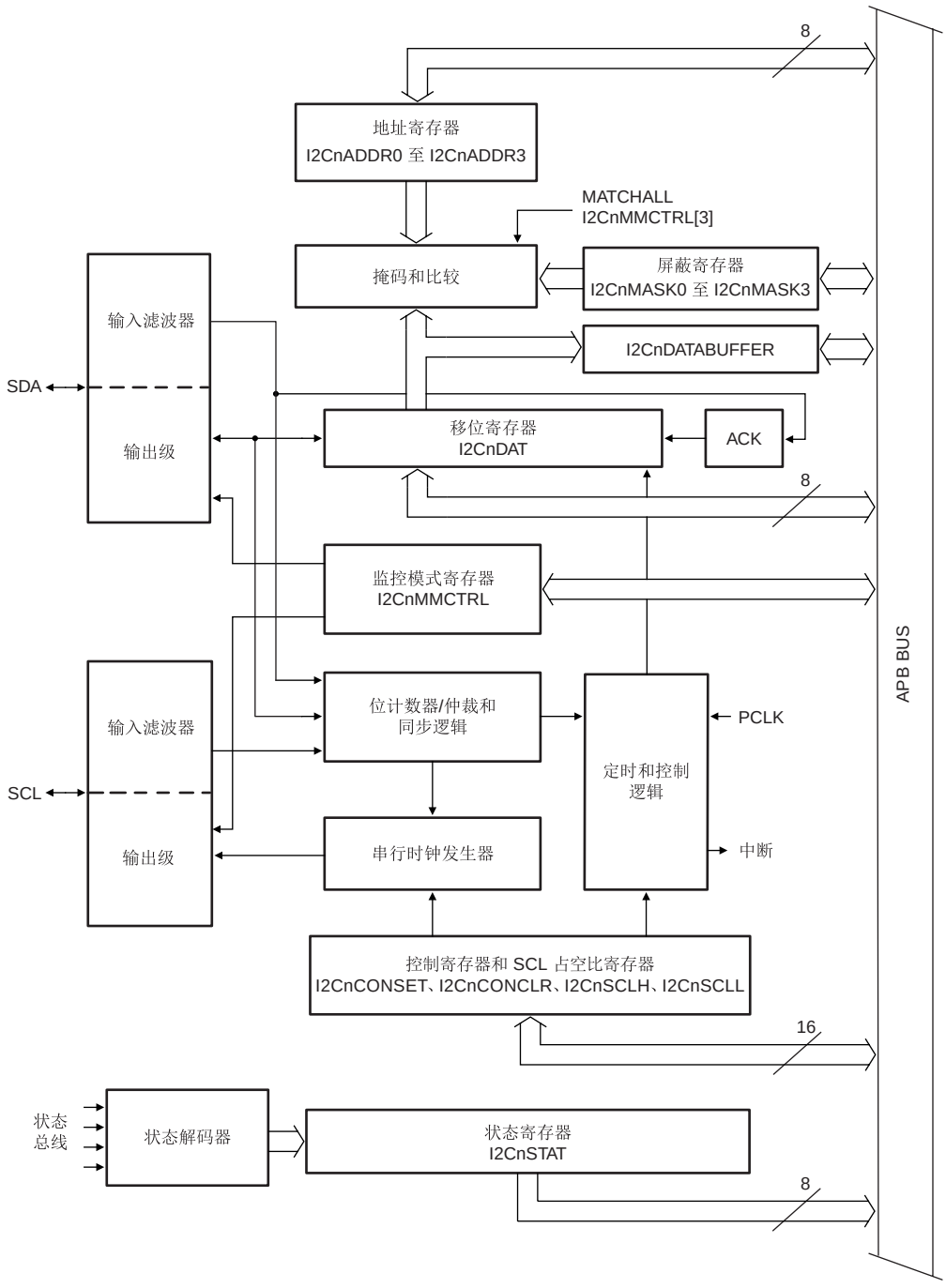


图 17. I²C 串行接口功能框图

9.8.1 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。

9.8.2 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 位从机地址（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。

9.8.3 地址屏蔽， MASK0 至 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。

如 ADRn 位 0 （GC 使能位）处于设置状态，且 (7:1) 各位全为零，无论相关屏蔽寄存器处于何种状态，器件都将响应接收到的地址 = “0000000”。

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。

9.8.4 比较器

比较器将接收到的 7 位从机地址与其自身的从机地址（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。

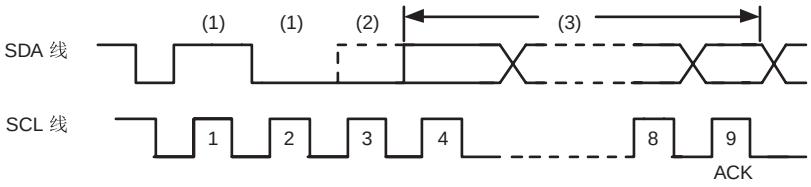
9.8.5 移位寄存器， DAT

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。

9.8.6 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 18 所示为仲裁过程。



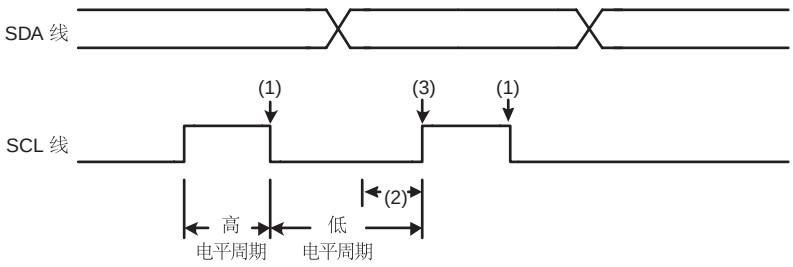
(1) 另一设备发送串行数据。

(2) 另一个设备通过将 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 18. 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 19 所示为同步过程。



(1) 另一个设备在 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) 另一个设备在 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 19. 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。

9.8.7 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。详情请参见关于 SCLL 和 SCLH 寄存器的描述。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。

9.8.8 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。

9.8.9 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。相反，写入 CONCLR 将清除 I²C 控制寄存器中的位，这些位与写入值中的位相对应。

9.8.10 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 I²C 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 I²C 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）。

9.9 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 个从机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。

9.9.1 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 129 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。通过向 CONCLR 寄存器中的 SIC 位写 1 来清零 SI 位。写入从机地址后，应清除 STA 位。

表 129. 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机地址 （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机地址和 “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，I^2C 接口将进入主发送模式。一旦总线空闲，I^2C 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，该例程将从机地址和 “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。

当发送完从机地址和 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，或者如果使能了从机模式（将 AA 设置为 1），则为 0x68、0x78 或 0xB0。各状态码的相应操作见表 133 ～表 138。

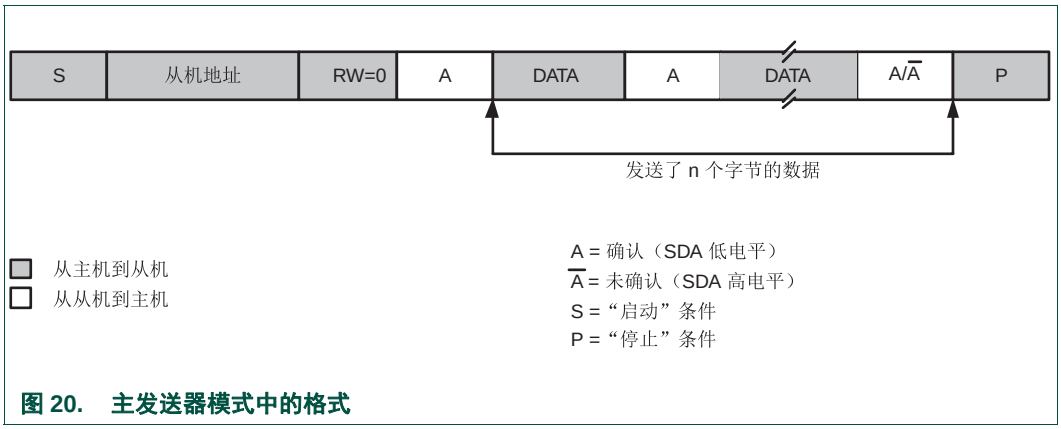


图 20. 主发送器模式中的格式

9.9.2 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机地址和数据方向位装入 I^2C 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”。

发送完从机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 134。

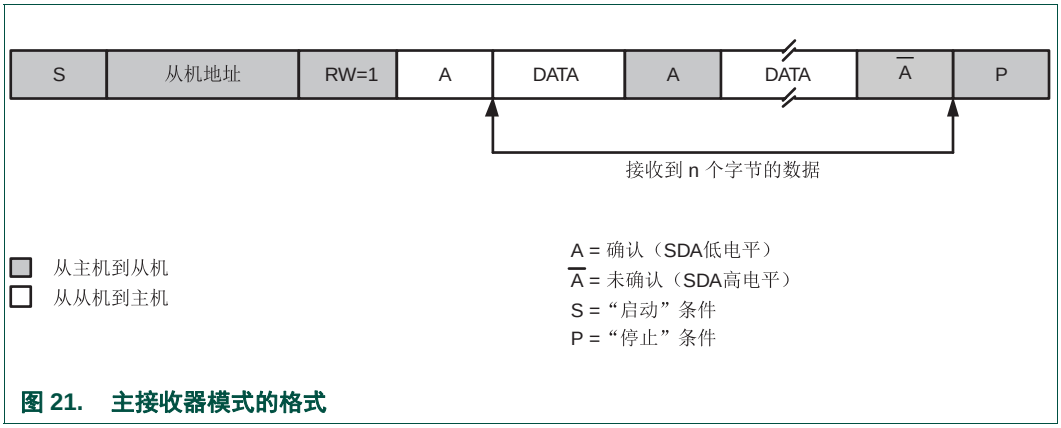


图 21. 主接收器模式的格式

经过一个重复起始条件后， I²C 可切换到主发送模式。

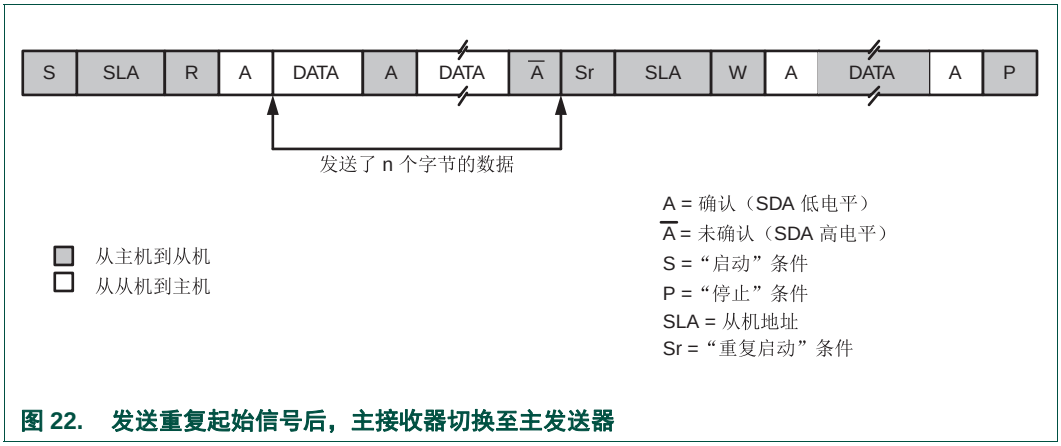


图 22. 发送重复起始信号后，主接收器切换至主发送器

9.9.3 从接收器模式

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 I²C 控制设置寄存器 (CONSET)，如表 130 所示。

表 130. 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 I²C 功能。 AA 位必须设置为 1 以确认其自身的从机地址或通用调用地址。 STA、 STO 和 SI 位都被清零。

初始化 ADR 和 CONSET 后，I²C 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 137。

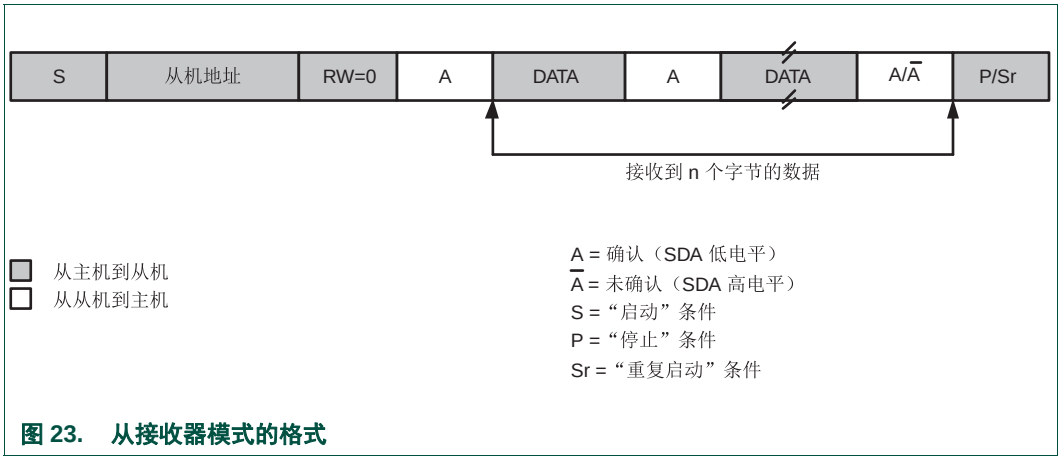


图 23. 从接收器模式的格式

9.9.4 从发送器模式

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。

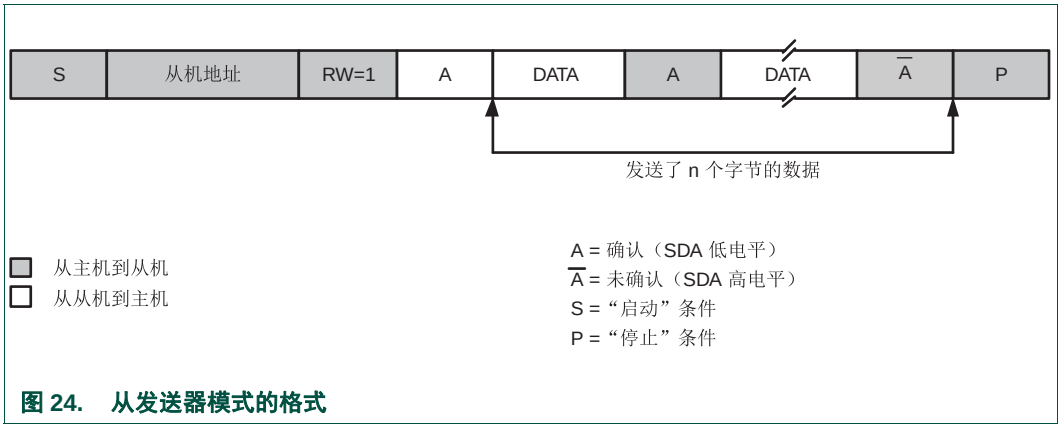


图 24. 从发送器模式的格式

9.10 I²C 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 25、图 26、图 27、图 28 和图 29 所示。表 131 说明了描述 I²C 操作模式的图中所使用缩写的含义。

表 131. 用于描述 I²C 操作的缩写

缩写	描述
S	起始条件
SLA	7 位从机地址
R	读取位 （SDA 为高电平）
W	写入位 （SDA 低电平）
A	确认位 （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 25 ～图 29 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 133 ～表 139。

9.10.1 主发送器模式

在主发送模式中，向从机接收器发送数据字节（见图 25）。CON 必须按如下所述进行初始化，然后才能进入主机发送器模式：

表 132. 用于主发送器模式的 CON

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。 I2EN 必须设置为 1 以使能 I²C 模块。如果 AA 位复位，则当另一个器件正变成总线主机时，I²C 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“起动”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，该例程将从机地址和数据方向位 (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。

从机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 133 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 切换到主接收模式。

表 133. 主发送器模式

状态码 (STAT)	I²C 总线和硬件的状态	应用软件响应 至 / 自 DAT	至 CON STA	STO	SI	AA	I²C 硬件执行的下一个操作
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	“重复起动”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 “重复起动”。
		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 “重复起动”。
		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 “重复起动”。
		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 “重复起动”。
		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”条件。

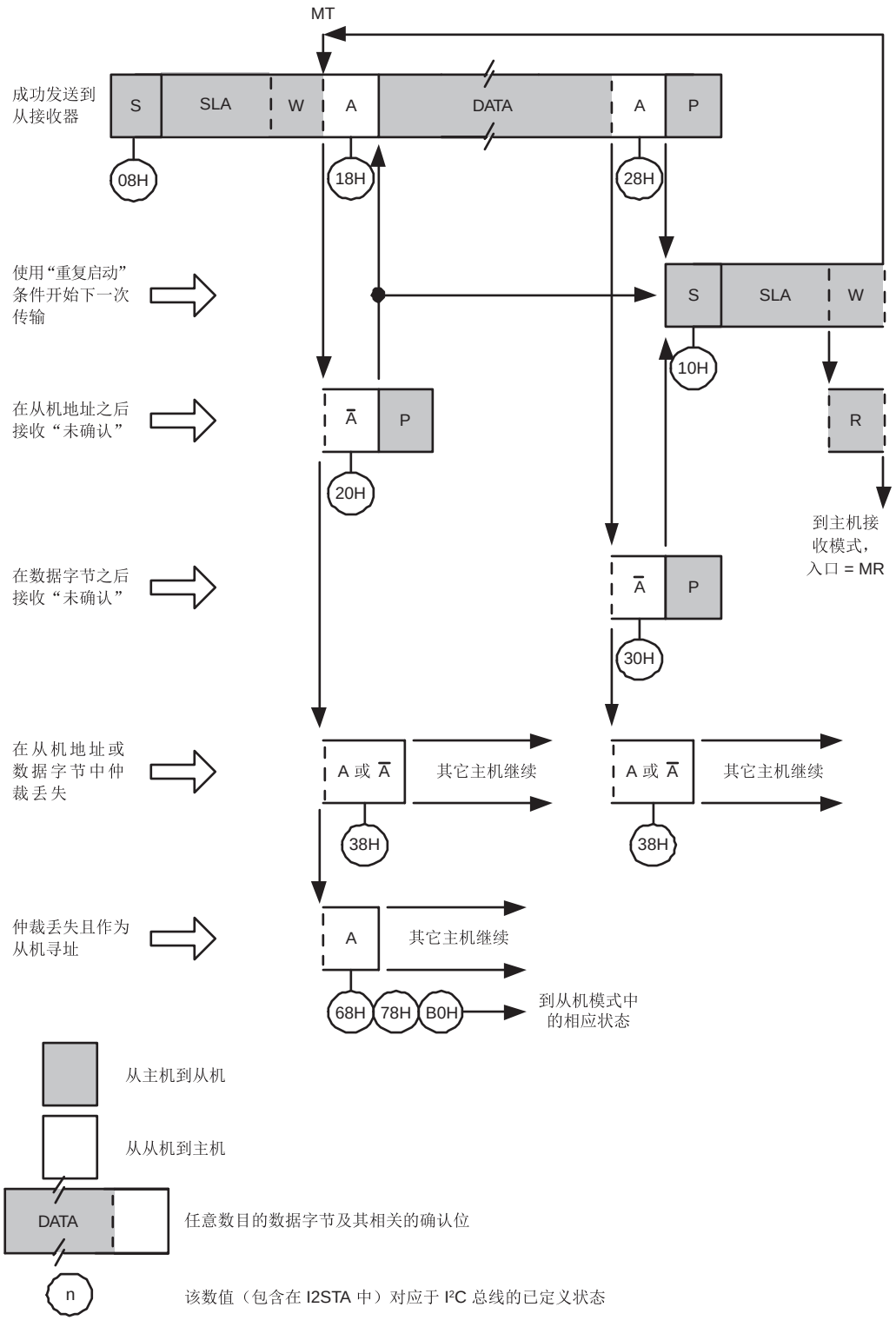


图 25. 主发送器模式下的格式和状态

9.10.2 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 26）。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 位从机地址和数据方向位 (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。

从机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)，则为 0x68、0x78 或 0xB0。表 134 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 134. 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	“重复起动”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 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 “重复起动”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 “重复起动”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。

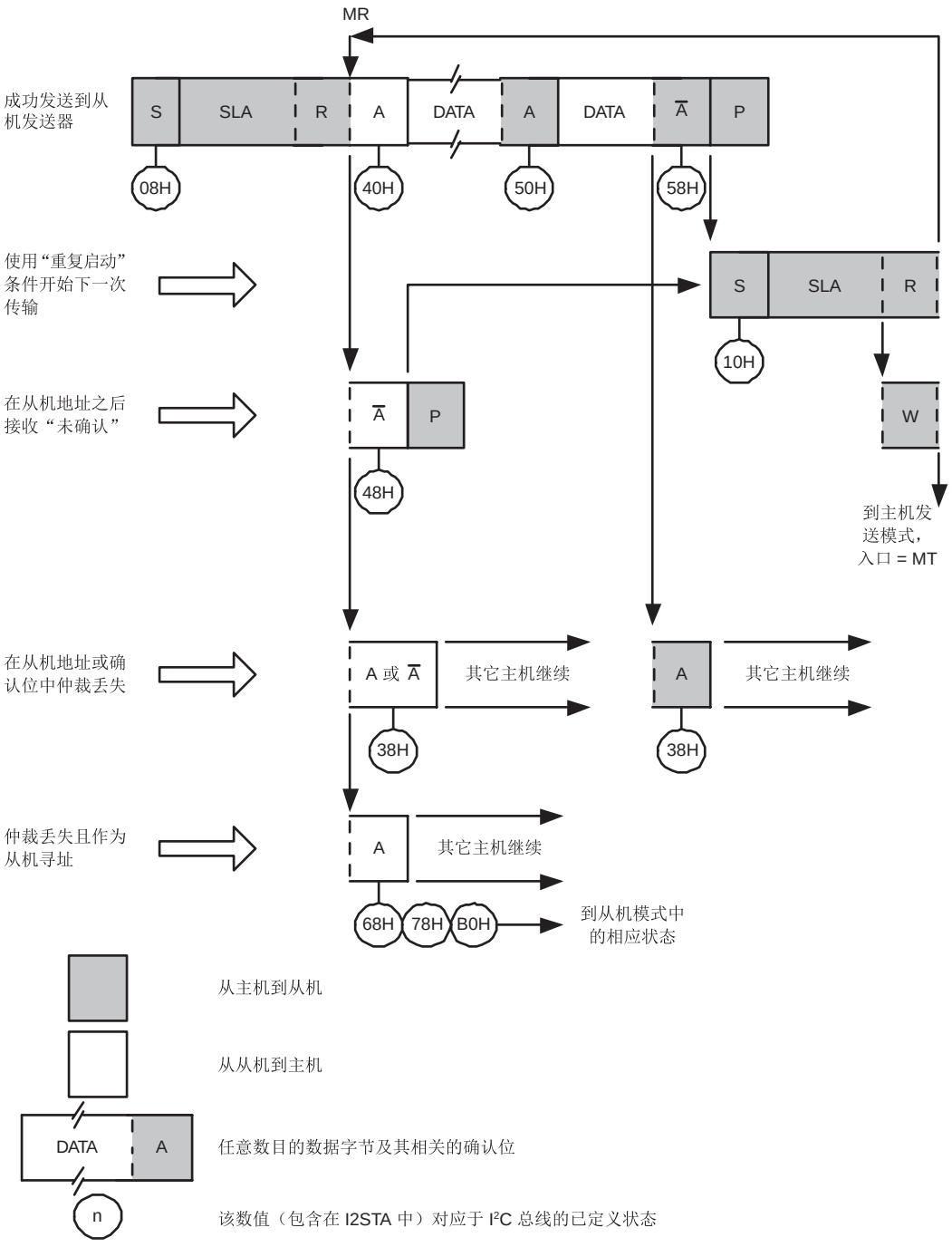


图 26. 主接收器模式下的格式和状态

9.10.3 从接收器模式

在从接收器模式中，从机所接收的数据字节来自主发送器 （见图 27）。要初始化从机接收器模式，必须按如下所示加载 ADR 和 CON：

表 135. 从接收器模式下的 ADR 使用

位	7	6	5	4	3	2	1	0
符号	自身的 7 位从机地址							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 136. 从接收器模式的 CON

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 块来应答其自身从机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 模块在从接收器模式下工作。接收到其自身的从机地址和 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 137 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，也可进入从接收器模式 （见状态 0x68 和 0x78）。

如果 AA 位在传输过程中复位，则在接收完下一个数据字节后 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机地址或通用调用地址。但是，I²C 总线仍被监控，而且，地址识别可随时通过置位 AA 来恢复。这就意味着 AA 位可临时将 I²C 块从 I²C 总线上分离出来。

表 137. 从接收器模式

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。
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	切换到不可寻址 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	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址。当总线空闲时将发送“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	切换到不可寻址 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	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址。当总线空闲时将发送“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。

表 137. 从接收器模式 *（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	
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	切换到不可寻址 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	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址。当总线空闲时将发送“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。

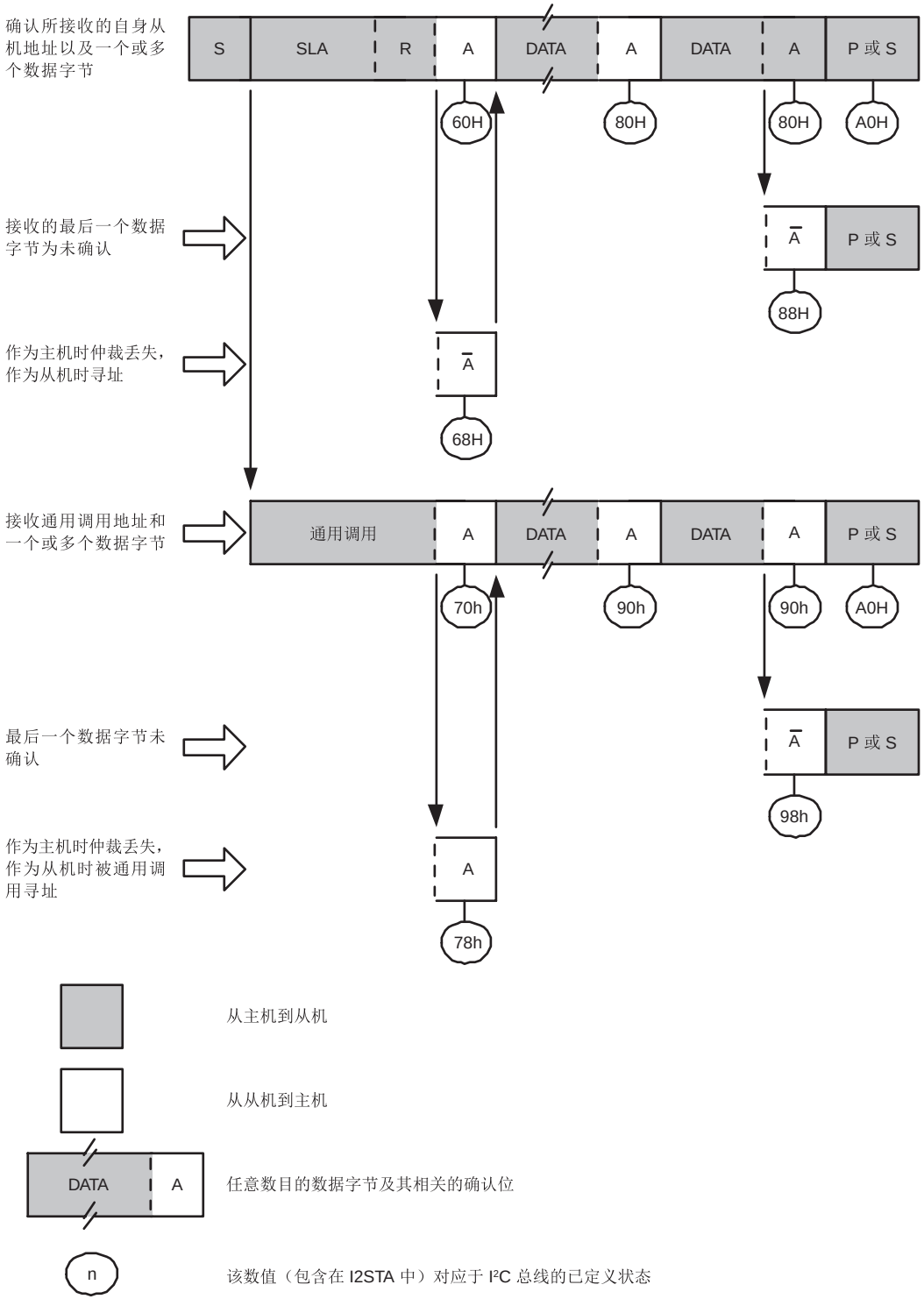


图 27. 从接收器模式下的格式和状态

9.10.4 从发送器模式

在从发送器模式中，向主接收器发送数据字节 （见图 28）。数据传输的初始化与从接收器模式下相同。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 模块在从发送器模式下工作。接收到其自身的从机地址和 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 138。如果当 I^2C 块在主机模式中时仲裁丢失，也可进入从发送器模式 （见状态 0xB0）。

如果 AA 位在传输过程中复位，则 I^2C 块将发送最后一个字节并进入状态 0xC0 或 0xC8。I^2C 块切换到不可寻址从机模式，如果继续传输，它将忽略主接收器。从而主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机地址或通用调用地址。但是，I^2C 总线仍被监控，而且，地址识别可随时通过置位 AA 来恢复。这就意味着 AA 位可临时将 I^2C 块从 I^2C 总线上分离出来。

表 138. 从发送器模式

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	切换到不可寻址 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	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址。当总线空闲时将发送 “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	切换到不可寻址 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	切换到不可寻址 SLV 模式；不识别自身 SLA 或通用调用地址。当总线空闲时将发送 “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。

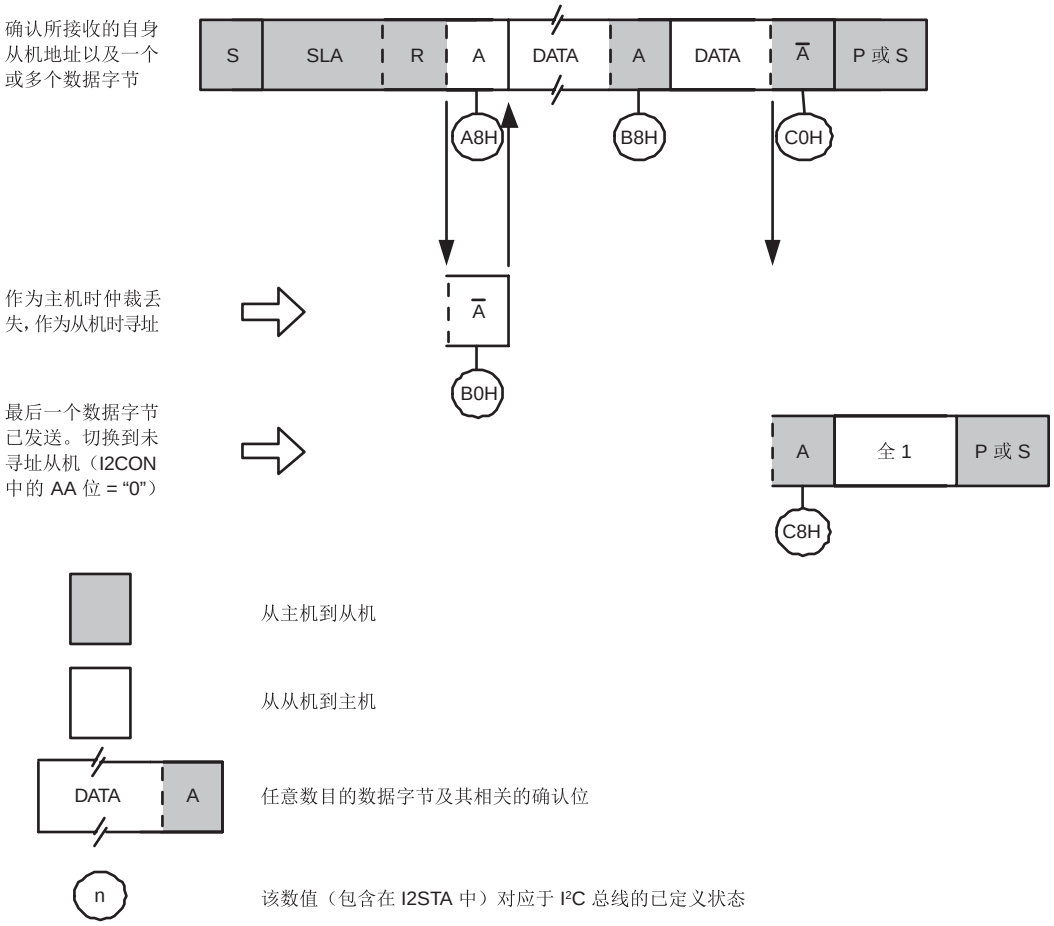


图 28. 从发送器模式下的格式和状态

9.10.5 其它状态

还有两种 STAT 代码与已定义的 I²C 硬件状态不对应 （见表 139）。论述如下。

9.10.5.1 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 I²C 块还未开始执行串行传输之间出现。

9.10.5.2 STAT = 0x00

该状态代码表示在 I²C 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 I²C 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 I²C 模块进入 “不可寻址”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 139. 其它状态

状态码 (STAT)	I²C 总线和硬件的状态	应用软件响应 至 / 自 DAT	至 CON STA	STO	SI	AA	I²C 硬件执行的下一个操作
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。当干扰导致 I²C 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、 I²C 块切换到不可寻址 SLV 模式。 STO 复位。

9.10.6 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 “重复起动”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

9.10.6.1 来自两个主机的同时 “重复起动”条件

在主机发送器模式或主机接收器模式下可以产生 “重复起动”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 29）。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。

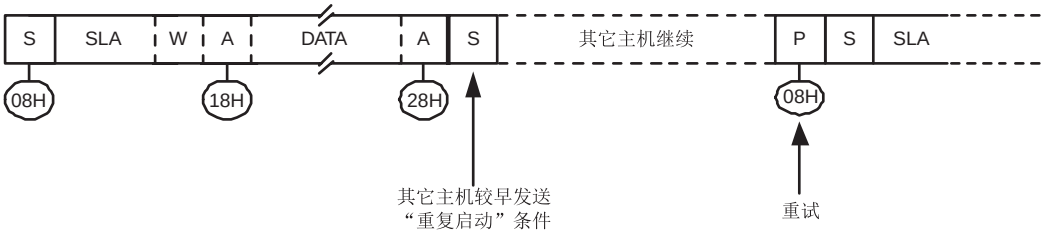


图 29. 来自两个主机的同时 “重复起动”条件

9.10.6.2 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 18）。 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 25 和图 26）。

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。

9.10.6.3 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 30）。

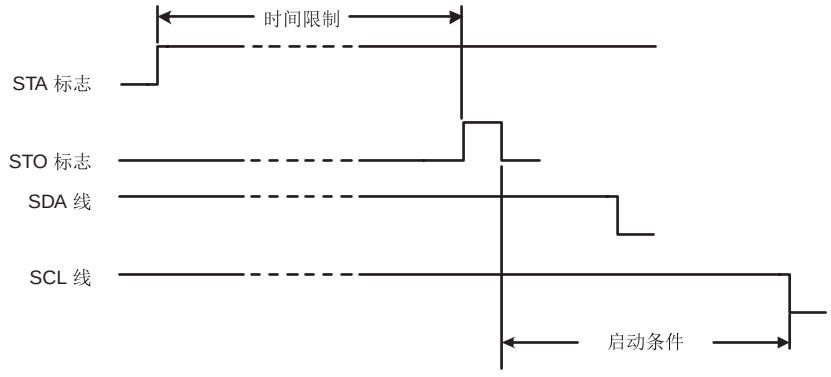
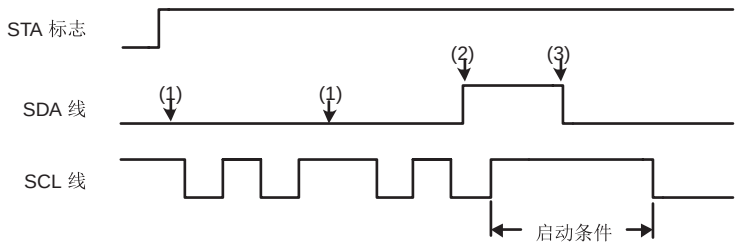


图 30. 强制访问忙 I²C 总线

9.10.6.4 SCL 或 SDA 上的低电平妨碍 I²C 总线的操作

如果 SDA 或 SCL 线被总线上任何一个器件拉低， I²C 总线可能挂起。如果 SCL 线被总线上的器件妨碍 （拉低），不能继续串行传输，这时必须通过拉低 SCL 总线的器件来处理。

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 31）。 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 31. 从因 SDA 低电平而导致总线受阻的状态中恢复

9.10.6.5 总线错误

当格式帧的非法位置上检测到 “启动”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 I²C 硬件作为主机或被寻址的从机进行串行传输时，它才对总线错误有反应。检测到总线错误时， I²C 块会立即切换成不可寻址从机模式，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 139 所示。

9.10.7 I²C 状态服务程序

本节提供了各种 I²C 状态服务程序都必须执行的操作示例。其中包括：

- 复位后 I²C 块的初始化。
- I²C 中断服务
- 支持 4 种 I²C 操作模式的 26 种状态服务程序。

9.10.8 初始化

在初始化示例中，I²C 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：

- 将器件自身的从机地址和通用调用位 (GC) 载入 ADR
- 置位 I²C 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，I²C 硬件开始在 I²C 总线上检查自身的从机地址和通用调用。如果检测到通用调用或自身的从机地址，则请求中断并将相应的状态信息载入 STAT。

9.10.9 I²C 中断服务

当进入I²C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。

9.10.10 状态服务程序

每个状态程序都是 I²C 中断程序的组成部分，分别用来处理 26 种状态。

9.10.11 配合实际应用的状态服务

状态服务示例演示了响应 26 个 I²C 状态代码必须执行的典型操作。如果 4 种 I²C 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I²C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。

9.11 软件示例

9.11.1 初始化程序

将 I²C 接口初始化用作从机和 / 或主机的例子。

1. 将自身的从机地址载入 ADR，如果需要，使能通用调用识别。
2. 使能 I²C 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。

9.11.2 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机地址，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。

9.11.3 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机地址，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。

9.11.4 I²C 中断程序

确定 I²C 的状态和处理该状态的状态程序。

1. 从 STA 中读出 I²C 的状态。
2. 使用状态值分支到 26 个可能状态例程之一。

9.11.5 无指定模式的状态

9.11.5.1 状态：0x00

总线错误。进入不可寻址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。

9.11.5.2 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。

9.11.5.3 状态：0x08

“起动”条件已发送。将发送从机地址 + R/W 位和接收 ACK 位。

1. 将从机地址和 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。

9.11.5.4 状态：0x10

“重复起动”条件已发送。将发送从机地址 + R/W 位和接收 ACK 位。

1. 将从机地址和 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。

9.11.6 主发送状态

9.11.6.1 状态：0x18

之前的状态为状态 8 或状态 10，从机地址 + 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。

9.11.6.2 状态：0x20

已发送从机地址 + 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。

9.11.6.3 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。

9.11.6.4 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。

9.11.6.5 状态：0x38

在从机地址 + 写位或数据期间仲裁已丢失。总线已释放且进入不可寻址从机模式。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。

9.11.7 主接收状态

9.11.7.1 状态：0x40

之前的状态为状态 08 或状态 10。从机地址 + 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。

9.11.7.2 状态：0x48

已发送从机地址 + 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。

9.11.7.3 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。

9.11.7.4 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。

9.11.8 从接收状态

9.11.8.1 状态：0x60

自身的从机地址 + 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。

9.11.8.2 状态：0x68

作为总线主机时，在从机地址和 R/W 位中仲裁已丢失。自身的从机地址 + 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。

9.11.8.3 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。

9.11.8.4 状态：0x78

作为总线主机时，在从机地址 +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。

9.11.8.5 状态：0x80

先前已使用自身的从机地址进行寻址。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。

9.11.8.6 状态：0x88

先前已使用自身的从机地址进行寻址。数据已接收且 NOT ACK 已返回。将不保存接收的数据。进入不可寻址从机模式。

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。

9.11.8.7 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。

9.11.8.8 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。进入不可寻址从机模式。

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。

9.11.8.9 状态：0xA0

已接收 “停止”条件或 “重复启动”条件，但仍作为从机寻址。将不保存数据。进入不可寻址从机模式。

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。

9.11.9 从发送器状态

9.11.9.1 状态：0xA8

自身的从机地址 + 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。

9.11.9.2 状态：0xB0

作为总线主机时，在从机地址和 R/W 位中仲裁丢失。自身的从机地址 + 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。

9.11.9.3 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。

9.11.9.4 状态：0xC0

已发送数据，已接收 NOT ACK。进入不可寻址从机模式。

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。

9.11.9.5 状态：0xC8

最后一个数据字节已发送， ACK 已接收。进入不可寻址从机模式。

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。

10.1 本章导读

所有 LPC11Axx 器件的 SPI 接口都相同。 SPI1 接口不适用于 WLCSP20 封装。

10.2 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。

10.3 简介

SPI/SSP 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。

LPC11Axx 具有 SPI/ 同步串行端口控制器。

10.4 引脚说明

表 140. SSP 引脚说明

引脚名称	类型	接口引脚名称 / 功能			引脚说明
		SPI	SSI	Microwire	
SCK0	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 SPI/SSP 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，SPI/SSP 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	I/O	SSEL	FS	CS	**帧同步 / 从机选择。**当 SPI/SSP 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 SPI/SSP 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	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 SPI/SSP 是从机时，从该信号上输出串行数据。当 SPI/SSP 为主机时，它记录从该信号发出的串行数据。当 SPI/SSP 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	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 SPI/SSP 为主机时，从该信号上输出串行数据。当 SPI/SSP 为从机时，它记录从该信号发出的串行数据。

注：这些信号中的大多数或全部可分配给各种引脚。要使用 SPI/SSP 控制器，则其每个信号都必须在一个并且只能在一个 IOCON 寄存器中选择。参见第 6.4.1 节。

10.5 时钟和电源控制

SSP 模块通过 SYSAHBCLKCTRL 寄存器 （参见表 19）进行门控。 SSP 时钟分频器和预分频器使用的外设 SSP 时钟由 SSP0CLKDIV 寄存器 （参见第 3.5.15 节）控制。

SPI0/1_PCLK 时钟可在 SSP0/1CLKDIV 寄存器（参见第 3.5.15 节）中禁用，SSP 模块可在 SYSAHBCLKCTRL 寄存器 （表 19）中禁用以省电。

10.6 寄存器描述

SSP 控制器的寄存器地址如表 141 所示。

表 141. 寄存器概述：SPI0 （基址 0x4004 0000）

名称	访问类型	地址偏移	描述	复位值 [1]
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
CR1	R/W	0x004	控制寄存器 1。选择主机 / 从机和其他模式。	0
DR	R/W	0x008	数据寄存器。写满发送 FIFO，读空接收 FIFO。	0
SR	RO	0x00C	状态寄存器	-
CPSR	R/W	0x010	时钟预分频寄存器	0
IMSC	R/W	0x014	中断屏蔽设置和清除寄存器	0
RIS	R/W	0x018	原始中断状态寄存器	-
MIS	R/W	0x01C	屏蔽中断状态寄存器	0
ICR	R/W	0x020	中断清除寄存器	不适用

[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。

10.6.1 SPI/SSP 控制寄存器 0

该寄存器控制 SPI/SSP 控制器的基本运行。

表 142. SPI/SSP 控制寄存器 0 （CR0 - 地址 0x4004 0000 (SSP0)）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	
		0x4	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。总线上的每位预定标器输出时钟数目，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
31:9	-		保留	-

10.6.2 SPI/SSP0 控制寄存器 1

该寄存器控制 SPI/SSP 控制器运行的某些方面。

表 143. SPI/SSP 控制寄存器 1 （CR1 - 地址 0x4004 0004 (SSP0) (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		SSP 使能。	0
		0	SSP 控制器被禁用。	
		1	SSP 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 SPI/SSP 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	SSP 控制器作为总线上的主机，驱动 SCK、MOSI 和 SSEL 线路，接收 MISO 线路。	
		1	SSP控制器作为总线上的从机，驱动MISO线路，接收SCK、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 SSP 控制器驱动发送数据线 (MISO)。	0
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
31:8	-		保留	-

10.6.3 SPI/SSP 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 144. SPI/SSP 数据寄存器 （DR - 地址 0x4004 0008 (SSP0)）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 SSP 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，SSP 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	-

10.6.4 SPI/SSP 状态寄存器

该只读寄存器反映 SSP 控制器的当前状态。

表 145. SPI/SSP 状态寄存器 （SR - 地址 0x4004 000C (SSP0)）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 SSP 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

10.6.5 SPI/SSP 时钟预分频寄存器

该寄存器控制预分频器分频 SSP 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 146. SPI/SSP 时钟预分频寄存器 （CPSR - 地址 0x4004 0010 (SSP0) (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，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

注意事项：SSPnCPSR 值必须适当初始化，否则 SSP 控制器不能正确发送数据。

在从机模式下，主机提供的 SSP 时钟速率不能超过第 3.5.15 节或第 3.5.17 节中所选 SSP 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）。

10.6.6 SPI/SSP 中断屏蔽设置 / 清除寄存器

该寄存器控制是否使能 SSP 控制器中 4 个可能的中断条件。请注意， ARM （此外设的设计者）使用的“屏蔽”一词与传统计算机术语含义相反，对于后者，“屏蔽”表示“禁用”。ARM 使用 “屏蔽”一词来表示 “使能”。为了避免混淆，本章不使用 “屏蔽”一词。

表 147. SPI/SSP 中断屏蔽设置 / 清除寄存器 （IMSC - 地址 0x4004 0014 (SSP0)）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，软件应将该位设置为使能中断。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。	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

10.6.7 SPI/SSP 原始中断状态寄存器

不管 SSPIMSC 寄存器中是否使能中断，只要产生中断条件，该只读寄存器便会在相应的位置包含 1。

表 148. SPI/SSP 原始中断状态寄存器 （RIS - 地址 0x4004 0018 (SSP0) (SSP1)）位描述

位	符号	描述	复位值
0	RORRIS	如果在 RxFIFO 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。	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

10.6.8 SPI/SSP 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 SSPIMSC 寄存器中使能时，该寄存器中对应的位为 1。当出现 SSP 中断时，中断服务例程会读取该寄存器以确定中断的原因。

表 149. SPI/SSP 屏蔽中断状态寄存器 （MIS - 地址 0x4004 001C (SSP0)）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。	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

10.6.9 SPI/SSP 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 SSP 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。

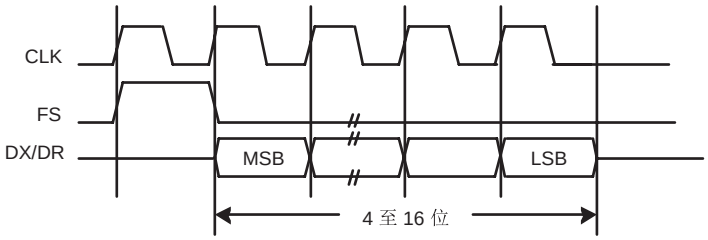
表 150. SPI/SSP 中断清除寄存器 （ICR - 地址 0x4004 0020 (SSP0)）位描述

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。	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

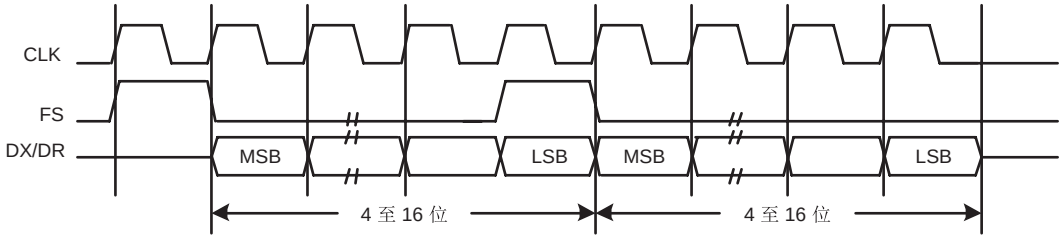
10.7 功能说明

10.7.1 德州仪器同步串行帧格式

图 32 显示了 SSP 模块支持的 4 线德州仪器同步串行帧格式。



a. 单帧传输



b. 连续 / 背靠背帧传输

图 32. 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。

10.7.2 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。

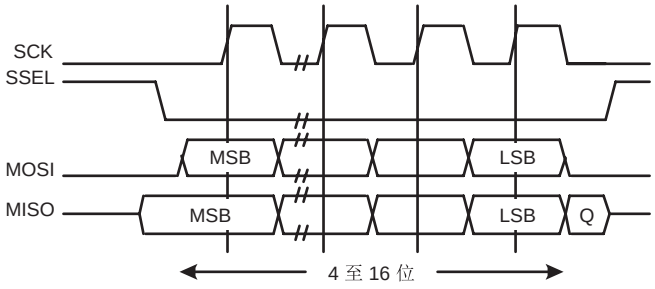
10.7.2.1 时钟极性 (CPOL) 及时钟相位 (CPHA) 控制

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。

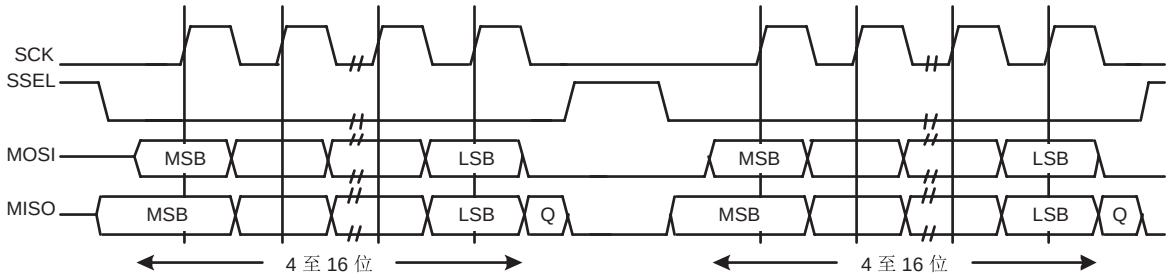
CPHA 控制位选择捕获数据及允许数据更改状态的时钟沿。通过是否在第一个数据捕获沿发生之前允许时钟跳变，该位能最大程度上影响第一个被发送的位。当 CPHA 相位控制位为低电平时，则在第一次出现时钟沿跳变时捕获数据。如果 CPHA 时钟相位控制位为高电平，则在第二次出现时钟沿跳变时捕获数据。

10.7.2.2 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 33 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 33. 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 SPI/SSP，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。

10.7.2.3 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 34 所示，包含单帧和连续帧传输。

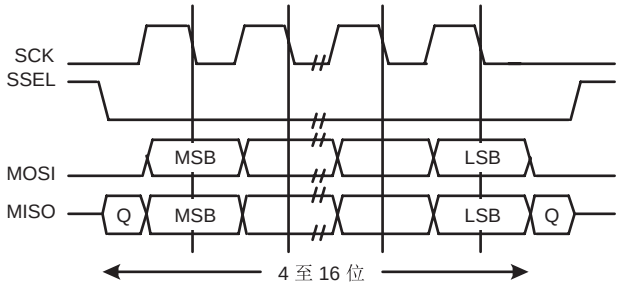


图 34. CPOL=0 且 CPHA=1 时的 SPI 帧格式

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 SPI/SSP，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。

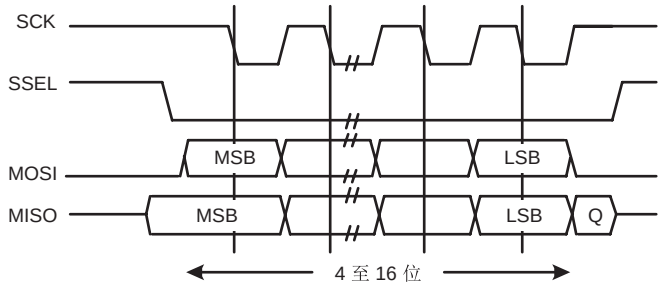
然后数据将在 SCK 信号下降沿捕获，并且在上升沿传播。

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。

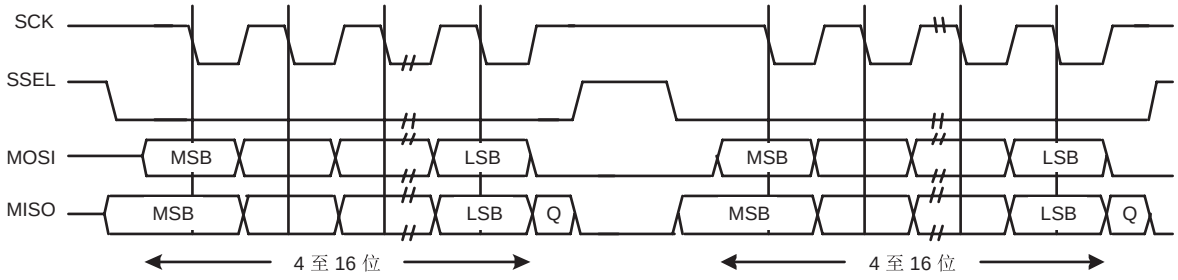
对于连续背靠背传输，SSEL 引脚在连续数据字之间保持为低电平，结束过程与单字传输相同。

10.7.2.4 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 35 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 35. 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 SPI/SSP，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。

10.7.2.5 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 36 所示，包含单帧和连续帧传输。

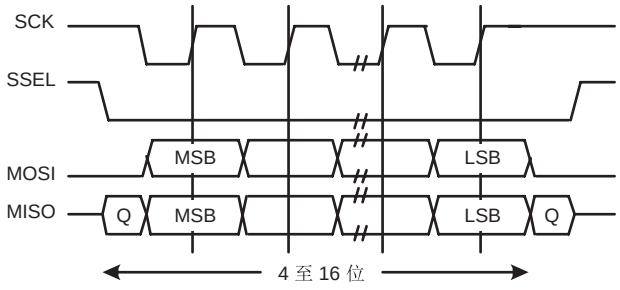


图 36. CPOL=1 且 CPHA=1 时的 SPI 帧格式

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 SPI/SSP，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。

10.7.3 国家半导体 Microwire 格式

图37显示了单帧的Microwire帧格式。图38显示了进行背靠背帧传输时的Microwire帧格式。

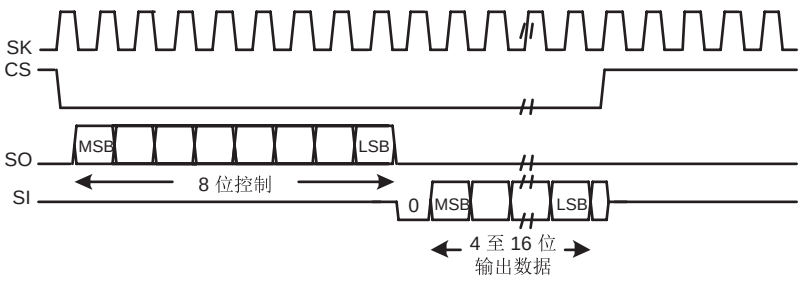


图 37. Microwire 帧格式 （单帧传输）

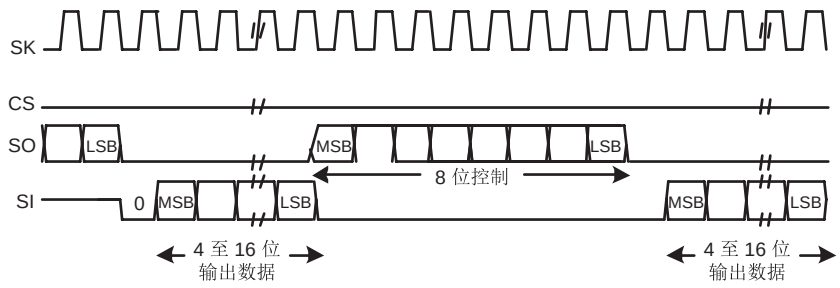


图 38. Microwire 帧格式 （连续帧传输）

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 SPI/SSP 发送到片外从机设备。在此发送过程中， SPI/SSP 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 引脚在此传输过程中保持三态。

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 SPI/SSP 来进行响应。每个位在 SK 的下降沿被驱动到 SI 线。 SPI/SSP 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 SPI/SSP 后，每个收到的值将在 SK 的下降沿从接收移位器传输。

10.7.3.1 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，SPI/SSP 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 39 描绘了这些建立和保持时间的要求。相对于 SK 上升沿（SPI/SSP 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 （SPI/SSP 在 SK 上运行）周期的两倍。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。

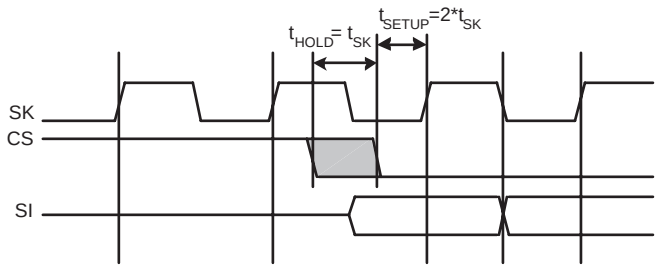


图 39. Microwire 帧格式建立及保持时间

11.1 本章导读

尽管输入数和输出数不同，但所有 PC11Axx 器件中的 16 位定时器模块都是一样的。

11.2 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：在 IOCON 寄存器模块中配置 CT16B0/1 引脚 （表 80）。
- 电源：在 SYSAHBCLKCTRL 寄存器中，设置表 19 中的位 7 和 8。
- 外设时钟由系统时钟确定 （参见表 18）。

11.3 特性

- 两个带有可编程 16 位预分频器的 16 位计数器 / 定时器。
- 计数器或定时器操作。
- 16位捕获通道，可在输入信号跳变时快速捕获定时器值。此类捕获事件也可选择性产生一个中断。
- 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。
- 由匹配寄存器控制的外部输出，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，最多有 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。
- 发生指定捕获事件时，可将定时器和预分频器清除。这样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。

11.4 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）

11.5 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 / 定时器还包括捕获输入，用于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，可使用匹配寄存器向匹配输出引脚提供单边沿控制的 PWM 输出。最好使用未引出的匹配寄存器来控制 PWM 周期长度。

下列各页中描述了 4 个 MATj 输出以及 3 个 CAPj 输入，但这些可能并非都与引脚连接。

注：16 位计数器 / 定时器 0 (CT16B0) 和 16 位计数器 / 定时器 1 (CT16B1) 除外设基址和 I/O 多路复用不同外，其他功能相似。

11.6 引脚说明

表 151 简要总结了每个计数器 / 定时器相关的引脚。

表 151. 计数器 / 定时器引脚描述

引脚	类型	描述
CT16B0_CAPj CT16B1_CAPj	输入	捕获信号： 可以配置引脚或片内信号上的跳变，用计数器 / 定时器中的值载入对应的捕获寄存器，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 11.7.11 节。
CT16B0_MATj CT16B1_MATj	输出	CT16B0/1 的外部匹配输出： 当匹配寄存器 (MRj) 的值与定时器计数器值 (TC) 相等时，输出 “j” 可以进行切换、进入低电平、进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) 控制这些输出的功能。

11.7 寄存器描述

16 位计数器 / 定时器都包含寄存器，如表 152 所示。“16Bi” 中的 “i” 可为 0 或 1，而 “MATj” 或 “CAPj” 中的 “j” 可为 0 至 3。未显示的地址偏移则被保留并且不应被写入。详细描述如下。

表 152. 寄存器简介：16 位计数器 / 定时器 CT16B0 （基址 0x4000 C000）和 CT16B1 （基址 0x4001 0000）

名称	访问类型	地址偏移	描述	复位值 [1]
IR	R/W	0x000	中断寄存器 (IR)。可以对 IR 执行写入操作来清除中断。可以通过读取 IR 来确定挂起 5 个可能中断源中的哪几个。	0
TCR	R/W	0x004	定时器控制寄存器 (TCR)。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
TC	R/W	0x008	定时器计数器 (TC)。16 位 TC 每隔 PR ＋ 1 个 PCLK 周期递增一次。TC 将通过 TCR 来控制。	0
PR	R/W	0x00C	预分频寄存器 (PR)。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
PC	R/W	0x010	预分频计数器 (PC)。 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
MCR	R/W	0x014	匹配控制寄存器 (MCR)。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
MR0	R/W	0x018	匹配寄存器。TC 与 MRj 匹配时，在 MCR 内可使能每个匹配来复位 TC、停止 TC 和 PC、生成中断和 / 或切换 CT16i_MATj 输出 （若有）。	0
MR1	R/W	0x01C	匹配寄存器。TC 与 MRj 匹配时，在 MCR 内可使能每个匹配来复位 TC、停止 TC 和 PC、生成中断和 / 或切换 CT16i_MATj 输出 （若有）。	0
MR2	R/W	0x020	匹配寄存器。TC 与 MRj 匹配时，在 MCR 内可使能每个匹配来复位 TC、停止 TC 和 PC、生成中断和 / 或切换 CT16i_MATj 输出 （若有）。	0
MR3	R/W	0x024	匹配寄存器。TC 与 MRj 匹配时，在 MCR 内可使能每个匹配来复位 TC、停止 TC 和 PC、生成中断和 / 或切换 CT16i_MATj 输出 （若有）。	0
CCR	R/W	0x028	捕获控制寄存器 (CCR)。 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
CR0	RO	0x02C	捕获寄存器。每个 CRj 寄存器的 CT16Bi_CAPj 输入或内部信号上有事件时，可使用 TC 的值来加载该寄存器。	0
CR1	RO	0x030	捕获寄存器。每个 CRj 寄存器的 CT16Bi_CAPj 输入或内部信号上有事件时，可使用 TC 的值来加载该寄存器。	0
CR2	RO	0x034	捕获寄存器。每个 CRj 寄存器的 CT16Bi_CAPj 输入或内部信号上有事件时，可使用 TC 的值来加载该寄存器。	0
CR3	RO	0x038	捕获寄存器。每个 CRj 寄存器的 CT16Bi_CAPj 输入或内部信号上有事件时，可使用 TC 的值来加载该寄存器。	0
EMR	R/W	0x03C	外部匹配寄存器 (EMR)。 EMR 控制匹配功能及外部匹配引脚 CT16Bi_MATj。	0
CTCR	R/W	0x070	计数控制寄存器 (CTCR)。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
PWMC	R/W	0x074	PWM 控制寄存器 (PWMCON)。 PWMCON 使能用于外部匹配引脚 CT16Bi_MATj 的 PWM 模式。	0

[1] 重置值仅反映存储在所用位中的数据，不包括保留位的内容。

11.7.1 中断寄存器 (IR)

中断寄存器 (IR) 最多包含用于匹配中断的 4 个位和用于捕获中断的 4 个位。如果生成了中断，则 IR 内的对应位会变为 1。否则，该位会变为 0。将 1 写入 IR 位会先清除该位，然后再清除中断。写入 0 无效。

表 153. 中断寄存器 （IR - 地址 0x4000 C000 (CT16B0) 和 0x4001 0000）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
6	CR2INT	捕获通道 2 事件的中断标志。	0
7	CR3INT	捕获通道 3 事件的中断标志。	0
31:8	-	保留	-

11.7.2 定时器控制寄存器 (TCR)

定时器控制寄存器 (TCR) 用于控制计数器 / 定时器的操作。

表 154. 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	C EN		计数器使能。	0
		0	计数器被禁能。	
		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

11.7.3 定时器计数器寄存器 (TC)

当预分频器计数器达到其最终计数时或在输入信号上发生选定的边沿时， 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 0xFFFF，然后翻转到 0x0000。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 155. 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。	-

11.7.4 预分频寄存器 (PR)

16 位预分频寄存器指定了预分频计数器的最大计数值。

表 156. 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。	-

11.7.5 预分频计数器寄存器 (PC)

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 157. 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。	-

11.7.6 匹配控制寄存器 (MCR)

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 158 所示。

表 158. 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 158. 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

11.7.7 匹配寄存器 (MR0/1/2/3)

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 159. 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。	-

11.7.8 捕获控制寄存器 (CCR)

捕获控制寄存器用于控制当相应的捕获事件发生时，是否将计数器 / 定时器中的值装入每个捕获寄存器，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，“i”表示定时器编号， 0 或 1。

表 160. 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		CT16Bi_CAP0上升沿捕获：CT16Bi_CAP0上的从0至1的序列，将使CR0载入TC内容。	0
		1	使能	
		0	已禁用	
1	CAP0FE		CT16Bi_CAP0下降沿捕获：CT16Bi_CAP0上的从1至0的序列，将使CR0载入TC内容。	0
		1	使能	
		0	已禁用	
2	CAP0I		CT16Bi_CAP0 事件中断：CT16Bi_CAP0 事件所导致的 CR0 装载将产生一个中断。	0
		1	使能	
		0	已禁用	
3	CAP1RE		CT16Bi_CAP1上升沿捕获：CT16Bi_CAP1上的从0至1的序列，将使CR1载入TC内容。	0
		1	使能	
		0	已禁用	
4	CAP1FE		CT16Bi_CAP1下降沿捕获：CT16Bi_CAP1上的从1至0的序列，将使CR1载入TC内容。	0
		1	使能	
		0	已禁用	
5	CAP1I		CT16Bi_CAP1 事件中断：CT16Bi_CAP1 事件所导致的 CR1 装载将产生一个中断。	0
		1	使能	
		0	已禁用	
6	CAP2RE		CT16Bi_CAP2上升沿捕获：CT16Bi_CAP2上的从0至1的序列，将使CR2载入TC内容。	0
		1	使能	
		0	已禁用	
7	CAP2FE		CT16Bi_CAP2下降沿捕获：CT16Bi_CAP2上的从1至0的序列，将使CR2载入TC内容。	0
		1	使能	
		0	已禁用	

表 160. 捕获控制寄存器 （CCR，地址 0x4000 C028 (CT16B0) 和 0x4001 0028 (CT16B1)）位描述 *（续）*

位	符号	值	描述	复位值
8	CAP2I		CT16Bi_CAP2 事件中断：CT16Bi_CAP2 事件所导致的 CR2 装载将产生一个中断。	0
		1	使能	
		0	已禁用	
9	CAP3RE		比较器输出上升沿捕获：比较器输出上的从 0 至 1 的序列，将使 CR3 载入 TC 内容。	0
		1	使能	
		0	已禁用	
10	CAP3FE		比较器输出下降沿捕获：比较器输出上的从 1 至 0 的序列，将使 CR3 载入 TC 内容。	0
		1	使能	
		0	已禁用	
11	CAP3I		比较器输出事件中断：比较器输出事件所导致的 CR3 装载将产生一个中断。	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

11.7.9 捕获寄存器 (CR0/1/2/3)

每个捕获寄存器都与信号相关，并且在该信号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 161. 捕获寄存器 （CR0 至 3，地址 0x4000 C02C 至 0x4000 C034 (CT16B0) ； CR0 至 3，地址 0x4001 002C 至 0x4001 C034 (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。	-

11.7.10 外部匹配寄存器 (EMR)

外部匹配寄存器为外部匹配通道和外部匹配引脚 CT16Bi_MATj 提供控制和状态。

如果匹配输出在 PWMCON 寄存器 （第 11.7.12 节）中被配置为 PWM 输出，则外部匹配寄存器的功能由 PWM 规则决定（第 168 页上的章节 11.7.13 “单边沿控制 PWM 的规则”）。

表 162. 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 CT16Bi_MAT0 的状态，无论该输出是否连接到其引脚。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 CT16Bi_MAT0 引脚上。	0
1	EM1		外部匹配 1。该位反映输出 CT16Bi_MAT1 的状态，无论该输出是否连接到其引脚。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 CT16Bi_MAT1 引脚上。	0
2	EM2		外部匹配 2。该位反映输出匹配通道 2 的状态，无论该输出是否连接到其引脚。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。请注意，在计数器 / 定时器 0 上，该匹配通道未引出。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 CT16Bi_MAT2 引脚上。	0
3	EM3		外部匹配 3。该位反映输出 CT16Bi_MAT3 的状态，无论该输出是否连接到某个引脚。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 CT16Bi_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 CT16Bi_MATj 引脚处于输出状态，则为低电平）。	
		0x2	将相应的外部匹配位 / 输出设为 1（如果 CT16Bi_MATj 引脚处于输出状态，则为高电平）。	
		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 CT16Bi_MATj 引脚处于输出状态，则为低电平）。	
		0x2	将相应的外部匹配位 / 输出设为 1（如果 CT16Bi_MATj 引脚处于输出状态，则为高电平）。	
		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 CT16Bi_MATj 引脚处于输出状态，则为低电平）。	
		0x2	将相应的外部匹配位 / 输出设为 1（如果 CT16Bi_MATj 引脚处于输出状态，则为高电平）。	
		0x3	切换相应的外部匹配位 / 输出。	
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 CT16Bi_MATj 引脚处于输出状态，则为低电平）。	
		0x2	将相应的外部匹配位 / 输出设为 1（如果 CT16Bi_MATj 引脚处于输出状态，则为高电平）。	
		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 163. 外部匹配控制

EMR[11:10]、EMR[9:8]、EMR[7:6] 或 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 CT16Bi_MATj 引脚处于输出状态，则为低电平）。
10	将相应的外部匹配位 / 输出设为 1（如果 CT16Bi_MATj 引脚处于输出状态，则为高电平）。
11	切换相应的外部匹配位 / 输出。

11.7.11 计数控制寄存器 (CTCR)

计数控制寄存器 (CTCR) 用于选择定时器模式和计数器模式，并在计数器模式中选择要进行计数的信号和边沿。

如果在该寄存器中选择计数器模式，则在 PCLK 时钟的每个上升沿上会对位 3:2 所选的信号进行采样。通过比较该信号的两个连续采样，可检测到信号上的上升沿和下降沿。定时器计数器寄存器根据该寄存器位 1:0 所选择的边沿递增。

由于选定的时钟信号由 PCLK 采样来确定选定时钟信号上的各边沿，因此时钟信号上的高低脉冲宽度不得小于 PCLK 周期。

该寄存器的位 7:4 控制捕获 - 清除 - 定时器功能。该功能允许特定 CAP 输入的选定边沿将定时器和预分频器复位为零。因此，定时器可在输入脉冲的前沿上清除，在后沿上捕获。所捕获的值之后直接反映脉冲宽度，无需在软件内进行减法运算。

表 164. 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。该字段选择定时器 / 计数器是由 PCLK 还是另一个信号计时。	00
		0x0	定时器模式：PC 在每个 PCLK 上升沿递增。	
		0x1	计数器模式：TC 在位 3:2 选定的信号的上升沿递增。	
		0x2	计数器模式：TC 在位 3:2 选定的信号的下降沿递增。	
		0x3	计数器模式：TC 在位 3:2 选定的信号的两个边沿递增。	
3:2	CIS		计数输入选择。在计数器模式中（当该寄存器中的位 1:0 不为 00 时），这些位选择使计数器递增的信号：	00
		0x0	CT16Bi_CAP0（捕获控制寄存器 (CCR) 中的位 2:0 必须为 000）	
		0x1	CT16Bi_CAP1（CCR 中的位 5:3 必须为 000）	
		0x2	CT16Bi_CAP2（CCR 中的位 8:6 必须为 000）	
		0x3	模拟比较器输出（CCR 中的位 11:9 必须为 000）	
4	EnCC		使能“捕获 - 清除”。该位中的 1 可在发生位 7:5 指定的事件时清除定时器和预分频器。	0
7:5	SelCC		“捕获 - 清除”选择。当位 4 为 1 时，这些位选择清除定时器和预分频器的捕获输入边沿。	000
		0x0	CT16Bi_CAP0 的上升沿	
		0x1	CT16Bi_CAP0 的下降沿	
		0x2	CT16Bi_CAP1 的上升沿	
		0x3	CT16Bi_CAP1 的下降沿	
		0x4	CT16Bi_CAP2 的上升沿	
		0x5	CT16Bi_CAP2 的下降沿	
		0x6	模拟比较器输出的上升沿	
		0x7	模拟比较器输出的下降沿	
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

11.7.12 PWM 控制寄存器 (PWMC)

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。

可用的单边沿控制 PWM 输出数在定时器 0 和 1 之间以及该系列的各器件之间可能有所变化。一个匹配寄存器决定 PWM 的周期长度。当任何其他匹配寄存器中出现匹配时， PWM 输出设为高电平。用于设置 PWM 周期长度的匹配寄存器负责将定时器复位。当定时器复位为 0 时，配置为 PWM 输出的匹配输出进入低电平。

表 165. PWM 控制寄存器 （PWMC，地址 0x4000 C074 (CT16B0) 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_MAT1 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 EM03 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。	不适用

11.7.13 单边沿控制 PWM 的规则

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：如果选择匹配输出作为 PWM 输出，则除控制 PWM 周期长度的匹配寄存器之外，匹配控制寄存器 MCR 中的定时器复位 (MRiR) 和定时器停止 (MRiS) 位必须清零。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 MRiR 位设为 1 来使能定时器复位。

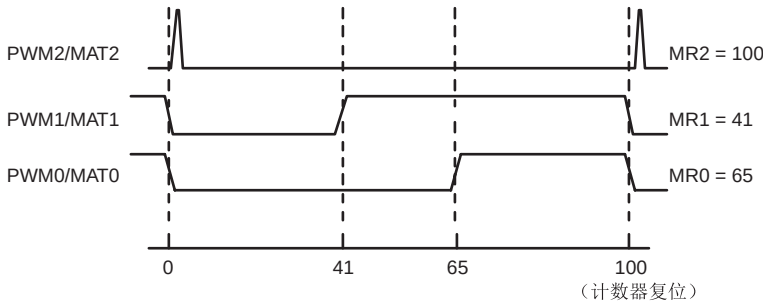


图 40. 采样 PWM 波形，其中 PWM 周期长度为 100（MR3 选择），MAT3:0 被 PWCON 寄存器使能为 PWM 输出。

11.8 定时器操作示例

如图 41 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 42 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。

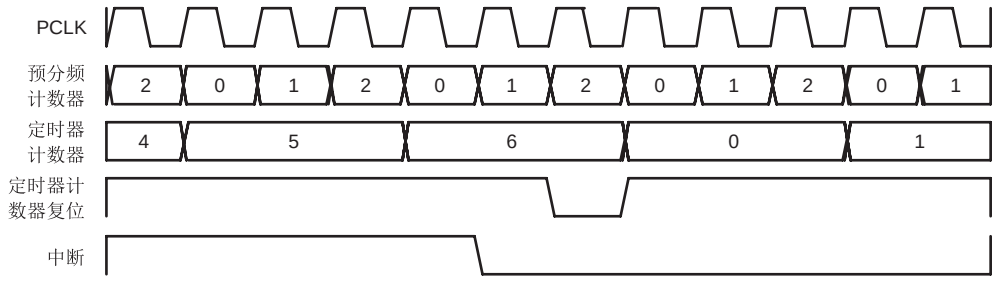


图 41. 定时器周期，其中 PR = 2、 MRx = 6，并且使能了在匹配时同时执行中断和复位操作

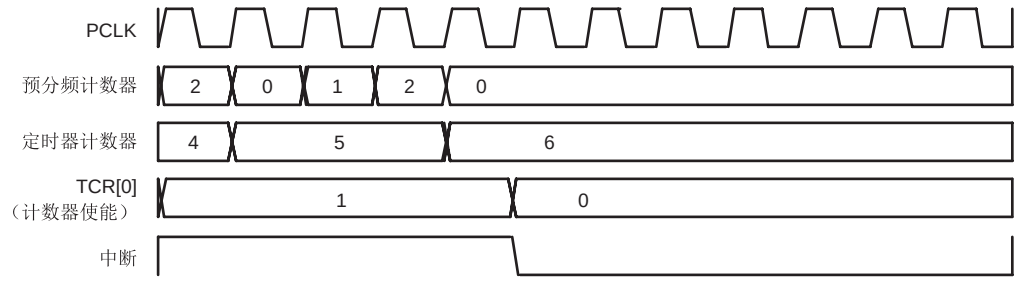


图 42. 定时器周期，其中 PR = 2、 MRx = 6，并且使能了在匹配时同时执行中断和停止操作

11.9 架构

16 位计数器 / 定时器框图如图 43 所示。

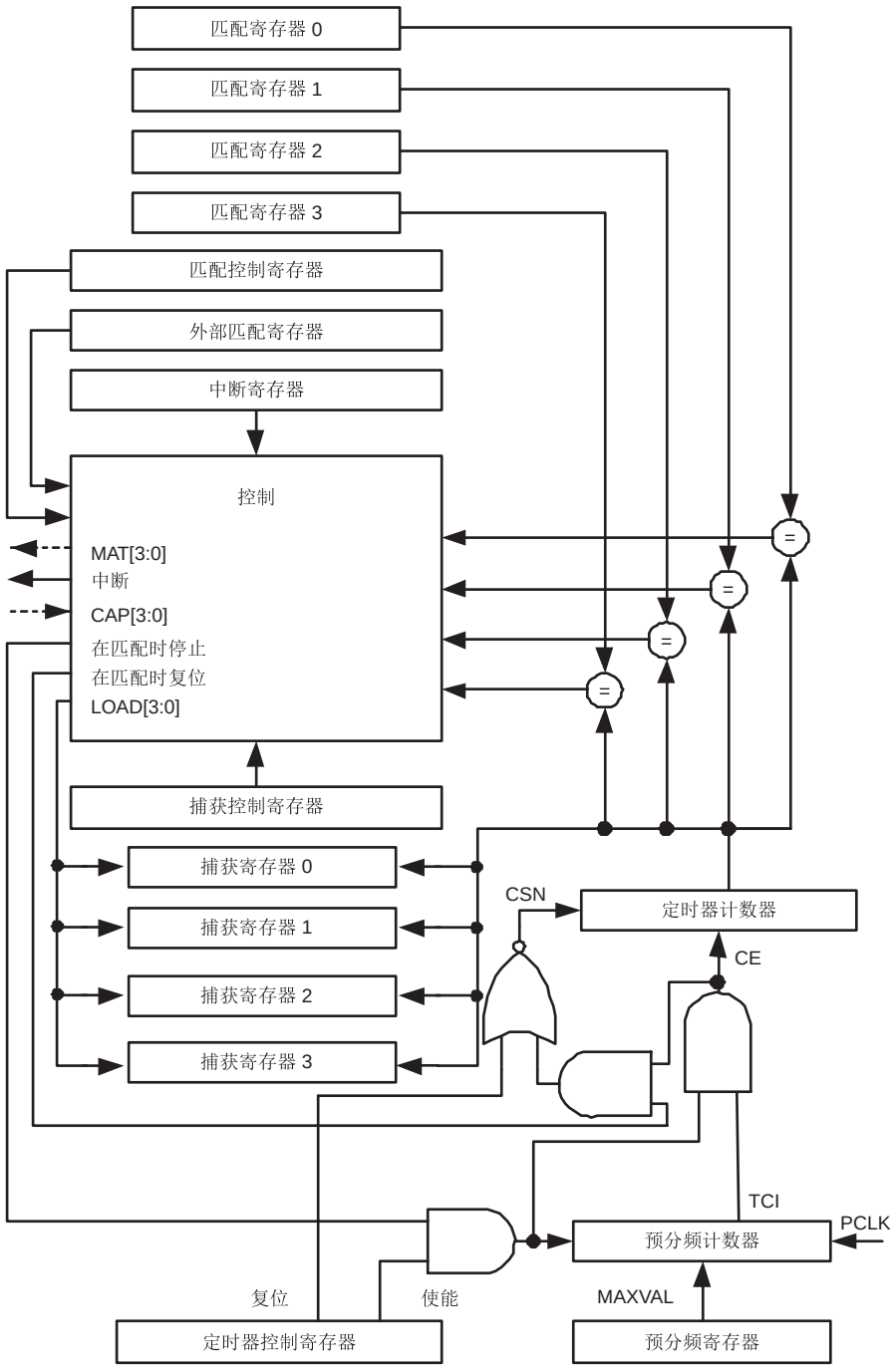


图 43. 16 位计数器 / 定时器功能框图

12.1 本章导读

尽管引脚分配数可能不同，但所有 PC11Axx 器件中的 32 位定时器模块都是一样的。

12.2 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：在 IOCON 寄存器模块中配置 CT32B0/1 引脚 （表 80）。
- 电源：在 SYSAHBCLKCTRL 寄存器中，设置表 19 中的位 9 和 10。
- 外设时钟由系统时钟确定 （参见表 18）。

12.3 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。
- 32位捕获通道，可在输入信号跳变时快速捕获定时器值。此类捕获事件也可选择性产生一个中断。
- 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。
- 由匹配寄存器控制的外部输出，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，最多有 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。
- 发生指定捕获事件时，可将定时器和预分频器清除。这样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。

12.4 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）

12.5 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，可使用匹配寄存器向匹配输出引脚提供单边沿控制的 PWM 输出。最好使用未引出的匹配寄存器来控制 PWM 周期长度。

下列各页中描述了 4 个 MATj 输出以及 3 个 CAPj 输入，但这些可能并非都与引脚连接。

注：32 位计数器 / 定时器 0 (CT32B0) 和 32 位计数器 / 定时器 1 (CT32B1) 除外设基址和 I/O 多路复用不同外，其他功能相似。

12.6 引脚说明

表 166 简要总结了每个计数器 / 定时器相关的引脚。

表 166. 计数器 / 定时器引脚描述

引脚	类型	描述
CT32B0_CAPj CT32B1_CAPj	输入	捕获信号： 可以配置引脚或片内信号上的跳变，用计数器 / 定时器中的值载入对应的捕获寄存器，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 180 页上的章节 12.7.11 “计数控制寄存器 (CTCR)”。
CT32B0_MATj CT32B1_MATj	输出	外部匹配输出： 当匹配寄存器 (MRj) 的值与定时器计数器值 (TC) 相等时， MATj 引脚可以进行切换、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) 控制这些输出的功能。

12.7 寄存器描述

32 位计数器 / 定时器都包含寄存器，如表 167 所示。“32Bi”中的“i”可为 0 或 1。“MATj”或“CAPj”中的“j”可为 0 至 3。未显示的地址偏移则被保留并且不应被写入。详细描述如下。

表 167. 寄存器简介：32 位计数器 / 定时器 CT32B0 （基址 0x4001 4000）和 CT32B1 （基址 0x4001 8000）

名称	访问类型	地址偏移	描述	复位值 [1]
IR	R/W	0x000	中断寄存器 (IR)。可以对 IR 执行写入操作来清除中断。可以通过读取 IR 来确定挂起 5 个可能中断源中的哪几个。	0
TCR	R/W	0x004	定时器控制寄存器 (TCR)。 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
TC	R/W	0x008	定时器计数器 (TC)。32 位 TC 每隔 PR + 1 个 PCLK 周期递增一次。TC 将通过 TCR 来控制。	0
PR	R/W	0x00C	预分频寄存器 (PR)。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
PC	R/W	0x010	预分频计数器 (PC)。 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
MCR	R/W	0x014	匹配控制寄存器 (MCR)。 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
MR0	R/W	0x018	匹配寄存器。TC 与 MRj 匹配时，在 MCR 内可使能每个匹配来复位 TC、停止 TC 和 PC、生成中断和 / 或切换 CT32i_MATj 输出 （若有）。	0
MR1	R/W	0x01C	匹配寄存器。TC 与 MRj 匹配时，在 MCR 内可使能每个匹配来复位 TC、停止 TC 和 PC、生成中断和 / 或切换 CT32i_MATj 输出 （若有）。	0
MR2	R/W	0x020	匹配寄存器。TC 与 MRj 匹配时，在 MCR 内可使能每个匹配来复位 TC、停止 TC 和 PC、生成中断和 / 或切换 CT32i_MATj 输出 （若有）。	0
MR3	R/W	0x024	匹配寄存器。TC 与 MRj 匹配时，在 MCR 内可使能每个匹配来复位 TC、停止 TC 和 PC、生成中断和 / 或切换 CT32i_MATj 输出 （若有）。	0
CCR	R/W	0x028	捕获控制寄存器 (CCR)。 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
CR0	RO	0x02C	捕获寄存器。每个 CRj 寄存器的 CT32Bi_CAPj 输入或内部信号上有事件时，可使用 TC 的值来加载该寄存器。	0
CR1	RO	0x030	捕获寄存器。每个 CRj 寄存器的 CT32Bi_CAPj 输入或内部信号上有事件时，可使用 TC 的值来加载该寄存器。	0
CR2	RO	0x034	捕获寄存器。每个 CRj 寄存器的 CT32Bi_CAPj 输入或内部信号上有事件时，可使用 TC 的值来加载该寄存器。	0
CR3	RO	0x038	捕获寄存器。每个 CRj 寄存器的 CT32Bi_CAPj 输入或内部信号上有事件时，可使用 TC 的值来加载该寄存器。	0
EMR	R/W	0x03C	外部匹配寄存器 (EMR)。EMR 控制匹配功能及外部匹配引脚 CT32B0_MATj。	0
CTCR	R/W	0x070	计数控制寄存器 (CTCR)。 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
PWMC	R/W	0x074	PWM 控制寄存器 (PWMCON)。 PWMCON 使能用于外部匹配引脚 CT32B0_MATj 的 PWM 模式。	0

[1] 重置值仅反映存储在所用位中的数据，不包括保留位的内容。

12.7.1 中断寄存器 (IR)

中断寄存器最多包含用于匹配中断的 4 个位和用于捕获中断的 4 个位。如果生成了中断，则 IR 内的对应位会变为 1。否则，该位会变为 0。将 1 写入 IR 位会先清除该位，然后清除中断。写入 0 无效。

表 168. 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
6	CR2INT	捕获通道 2 事件的中断标志。	0
7	CR3INT	捕获通道 3 事件的中断标志。	0
31:8	-	保留	-

12.7.2 定时器控制寄存器 (TCR)

定时器控制寄存器 (TCR) 用于控制计数器 / 定时器的操作。

表 169. 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	计数器被禁能。	
		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

12.7.3 定时器计数器 (TC)

当预分频器计数器达到其最终计数时或在输入信号上发生选定的边沿时， 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 170. 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

12.7.4 预分频寄存器 (PR)

32 位预分频寄存器指定了预分频计数器的最大计数值。

表 171. 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	预分频值。	0

12.7.5 预分频计数器寄存器 (PC)

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 172. 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

12.7.6 匹配控制寄存器 (MCR)

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 173 所示。

表 173. 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 173. 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

12.7.7 匹配寄存器 (MR0/1/2/3)

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 174. 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

12.7.8 捕获控制寄存器 (CCR)

捕获控制寄存器用于控制当相应的捕获事件发生时，是否将定时器计数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，“i”表示定时器编号， 0 或 1。

表 175. 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		CT32Bi_CAP0上升沿捕获. CT32Bi_CAP0上的从0至1的序列，将使CR0载入TC内容。	0
		1	使能	
		0	已禁用	
1	CAP0FE		CT32Bi_CAP0下降沿捕获. CT32Bi_CAP0上的从1至0的序列，将使CR0载入TC内容。	0
		1	使能	
		0	已禁用	
2	CAP0I		CT32Bi_CAP0 事件中断：CT32Bi_CAP0 事件所导致的 CR0 装载将产生一个中断。	0
		1	使能	
		0	已禁用	
3	CAP1RE		CT32Bi_CAP1上升沿捕获. CT32Bi_CAP1上的从0至1的序列，将使CR1载入TC内容。	0
		1	使能	
		0	已禁用	
4	CAP1FE		CT32Bi_CAP1下降沿捕获. CT32Bi_CAP1上的从1至0的序列，将使CR1载入TC内容。	0
		1	使能	
		0	已禁用	
5	CAP1I		CT32Bi_CAP1 事件中断：CT32Bi_CAP1 事件所导致的 CR1 装载将产生一个中断。	0
		1	使能	
		0	已禁用	
6	CAP2RE		CT32Bi_CAP2上升沿捕获. CT32Bi_CAP2上的从0至1的序列，将使CR2载入TC内容。	0
		1	使能	
		0	已禁用	
7	CAP2FE		CT32Bi_CAP2下降沿捕获. CT32Bi_CAP2上的从1至0的序列，将使CR2载入TC内容。	0
		1	使能	
		0	已禁用	
8	CAP2I		CT32Bi_CAP2 事件中断：CT32Bi_CAP2 事件所导致的 CR2 装载将产生一个中断。	0
		1	使能	
		0	已禁用	
9	CAP3RE		比较器输出上升沿捕获：比较器输出上的从 0 至 1 的序列，将使 CR3 载入 TC 内容。	0
		1	使能	
		0	已禁用	
10	CAP3FE		比较器输出下降沿捕获：比较器输出上的从 1 至 0 的序列，将使 CR3 载入 TC 内容。	0
		1	使能	
		0	已禁用	

表 175. 捕获控制寄存器（CCR，地址 0x4001 4028 (CT32B0) 和 0x4001 8028 (CT32B1)）位描述 *（续）*

位	符号	值	描述	复位值
11	CAP3I		比较器输出事件中断：比较器输出事件所导致的 CR3 装载将产生一个中断。	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

12.7.9 捕获寄存器 (CR)

每个捕获寄存器都与信号相关，并且在该信号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 176. 捕获寄存器（CR0 至 3，地址 0x4001 402C 至 0x4001 4038 (CT16B0) 和 CR0 至 3，地址 0x4001 802C 至 0x4001 8038 (CT16B1)）位描述

位	符号	描述	复位值
31:0	CAP	定时器计数器捕获值。	0

12.7.10 外部匹配寄存器 (EMR)

外部匹配寄存器为外部匹配通道和外部匹配引脚 CT32Bi_MATj 提供控制和状态。

如果匹配输出在 PWMCON 寄存器 （第 12.7.12 节）中被配置为 PWM 输出，则外部匹配寄存器的功能由 PWM 规则决定（第 182 页上的章节 12.7.13 “单边沿控制 PWM 的规则”）。

表 177. 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 CT32Bi_MAT0 的状态，无论该输出是否连接到其引脚。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 CT32Bi_MAT0 引脚上。	0
1	EM1		外部匹配 1。该位反映输出 CT32Bi_MAT1 的状态，无论该输出是否连接到其引脚。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 CT32Bi_MAT1 引脚上。	0
2	EM2		外部匹配 2。该位反映输出 CT32Bi_MAT2 的状态，无论该输出是否连接到其引脚。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 CT32Bi_MAT2 引脚上。	0
3	EM3		外部匹配 3。该位反映输出 CT32Bi_MAT3 的状态，无论该输出是否连接到其引脚。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 CT32Bi_MAT3i 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。	

表 177. 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
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。	
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 178. 外部匹配控制

EMR[11:10]、 EMR[9:8]、 EMR[7:6] 或 EMR[5:4]	函数
0x0	不执行任何操作。
0x1	将相应的外部匹配位 / 输出清零 （如果 CT32Bi_MATj 引脚处于输出状态，则为低电平）。
0x2	将相应的外部匹配位 / 输出设为 1（如果 CT32Bi_MATj 引脚处于输出状态，则为高电平）。
0x3	切换相应的外部匹配位 / 输出。

12.7.11 计数控制寄存器 (CTCR)

计数控制寄存器 (CTCR) 用于选择定时器模式和计数器模式，并在计数器模式中选择要进行计数的信号和边沿。

如果在该寄存器中选择计数器模式，则在 PCLK 时钟的每个上升沿上会对位 3:2 所选的信号进行采样。通过比较该信号的两个连续采样，可检测到信号上的上升沿和下降沿。定时器计数器寄存器根据该寄存器位 1:0 所选择的边沿递增。

由于选定的时钟信号由 PCLK 采样来确定选定时钟信号上的各边沿，因此时钟信号上的高低脉冲宽度不得小于 PCLK 周期。

该寄存器的位 7:4 控制捕获 - 清除 - 定时器功能。该功能允许特定 CAP 输入的选定边沿将定时器和预分频器复位为零。因此，定时器可在输入脉冲的前沿上清除，在后沿上捕获。所捕获的值之后直接反映脉冲宽度，无需在软件内进行减法运算。

表 179. 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。该字段选择定时器 / 计数器是由 PCLK 还是另一个信号计时。	00
		0x0	定时器模式：PC 在每个 PCLK 上升沿递增	
		0x1	计数器模式：TC 在位 3:2 选定的信号的上升沿递增。	
		0x2	计数器模式：TC 在位 3:2 选定的信号的下降沿递增。	
		0x3	计数器模式：TC 在位 3:2 选定的信号的两个边沿递增。	
3:2	CIS		在计数器模式中 （当该寄存器中的位 1:0 不为 00 时），这些位选择使计数器递增的信号：	00
		0x0	CT32Bi_CAP0 （捕获控制寄存器 (CCR) 中的位 2:0 必须为 000）	
		0x1	CT32Bi_CAP1 （CCR 中的位 5:3 必须为 000）	
		0x2	CT32Bi_CAP2 （CCR 中的位 8:6 必须为 000）	
		0x3	模拟比较器输出 （CCR 中的位 11:9 必须为 000）	
4	ENCC		使能 “捕获 - 清除”。该位中的 1 可在发生位 7:5 指定的事件时清除定时器和预分频器。	0
7:5	SELCC		“捕获 - 清除”选择。当位 4 为 1 时，这些位选择清除定时器和预分频器的捕获输入边沿。	000
		0x0	CT32Bi_CAP0 的上升沿	
		0x1	CT32Bi_CAP0 的下降沿	
		0x2	CT32Bi_CAP1 的上升沿	
		0x3	CT32Bi_CAP1 的下降沿	
		0x7	CT32Bi_CAP2 的上升沿	
		0x5	CT32Bi_CAP2 的下降沿	
		0x6	模拟比较器输出的上升沿	
		0x7	模拟比较器输出的下降沿	
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

12.7.12 PWM 控制寄存器 (PWMC)

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。

可用的单边沿控制 PWM 输出数在定时器之间以及该系列的各器件之间可能有所变化。一个匹配寄存器决定 PWM 的周期长度。当任何其他匹配寄存器中出现匹配时，PWM 输出设为高电平。用于设置 PWM 周期长度的匹配寄存器负责将定时器复位。当定时器复位为 0 时，配置为 PWM 输出的匹配输出进入低电平。

表 180. 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_MAT0 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_MAT0 受 EM2 控制。	
		1	CT32Bi_MAT2 的 PWM 模式使能。	
3	PWMEN3		通道 3 的 PWM 模式使能。	0
		0	CT32Bi_MAT0 受 EM03 控制。	
		1	CT32Bi_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

12.7.13 单边沿控制 PWM 的规则

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：如果选择匹配输出作为 PWM 输出，则除控制 PWM 周期长度的匹配寄存器之外，匹配控制寄存器 MCR 中的定时器复位 (MRiR) 和定时器停止 (MRiS) 位必须清零。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 MRiR 位设为 1 来使能定时器复位。

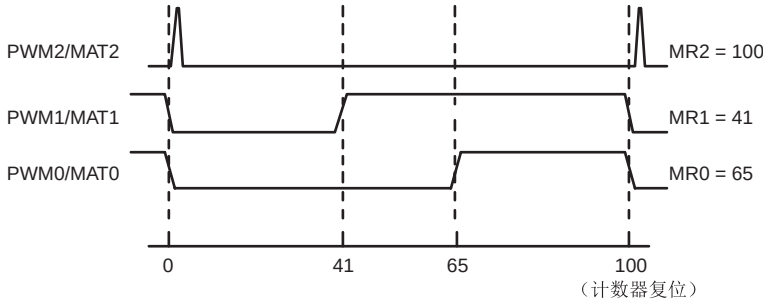


图 44. 采样 PWM 波形，其中 PWM 周期长度为 100（MR3 选择），MAT3:0 被 PWCON 寄存器使能为 PWM 输出。

12.8 定时器操作示例

如图 45 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 46 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。

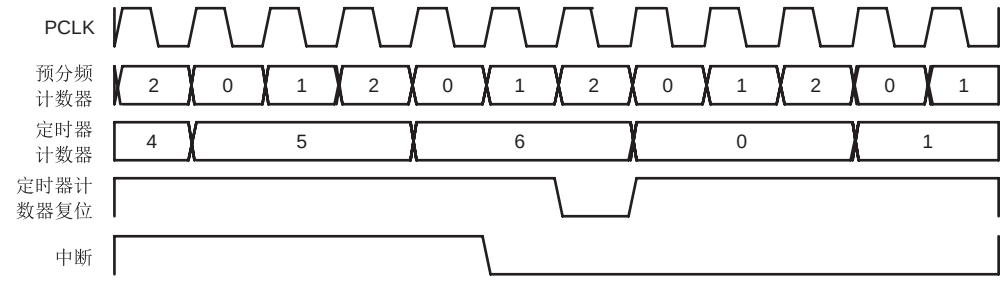


图 45. 定时器周期，其中 PR = 2、 MRx = 6，并且使能了在匹配时同时执行中断和复位操作

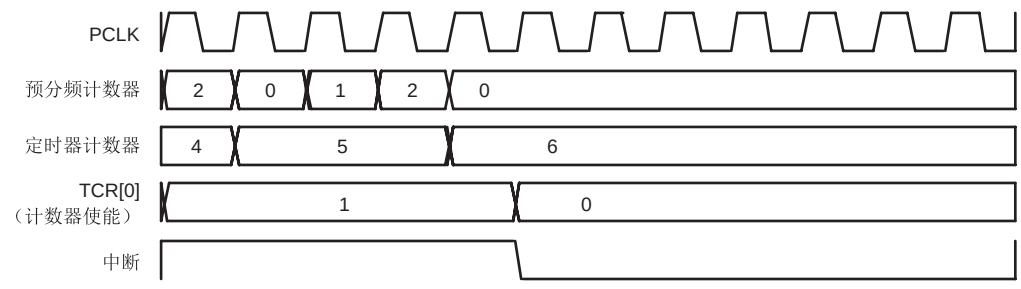


图 46. 定时器周期，其中 PR = 2、 MRx = 6，并且使能了在匹配时同时执行中断和停止操作

12.9 架构

32 位计数器 / 定时器框图如图 47 所示。

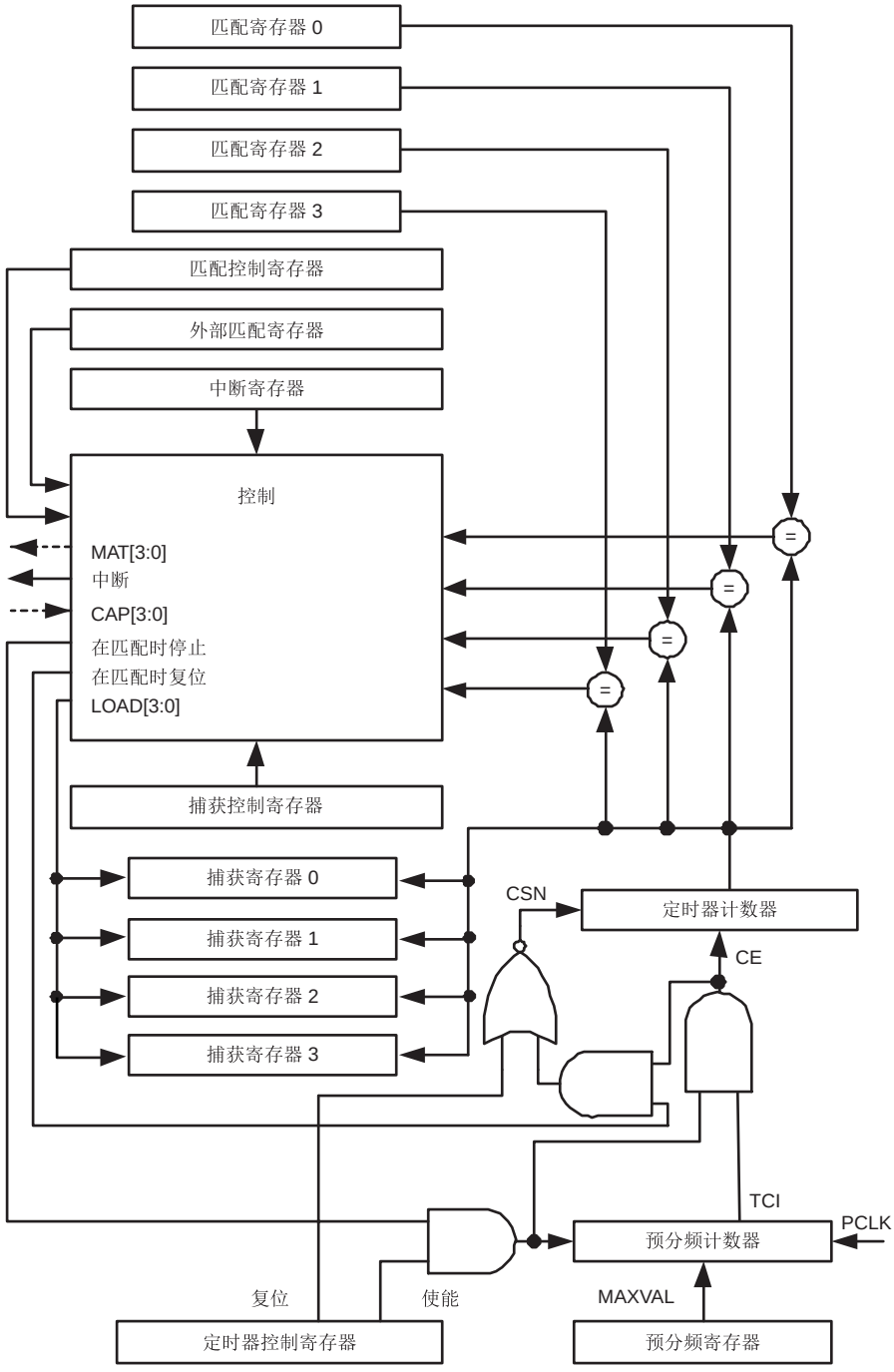


图 47. 32 位计数器 / 定时器功能框图

13.1 本章导读

系统节拍定时器（SysTick 定时器）是 ARM Cortex-M0 内核的一部分，而且对于所有 LPC11Axx 器件都是相同的。

13.2 特性

- 时间间隔为 10 毫秒。
- 使用专用的异常向量。
- 由专用系统节拍定时器时钟从内部计时。

13.3 描述

SysTick 定时器是 Cortex-M0 的主要组成部分。SysTick 定时器为操作系统或其它系统管理软件提供固定 10 毫秒的中断。

由于 SysTick 定时器是 Cortex-M0 的一部分，它为所有基于 Cortex-M0 的器件提供一个标准定时器，有助于软件的移植。

详情请参考 *Cortex-M0 用户指南*。

13.4 操作

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 定时器从 CPU 时钟计时。要以特定的间隔生成循环中断，就必须使用所需间隔的正确值初始化 STRELOAD 寄存器。默认值保存在 STCALIB 寄存器中，可通过软件进行修改。如果 CPU 时钟设置正确，则默认值为 10 毫秒的中断速率。

SysTick 定时器的功能框图如下面的图 48 所示。

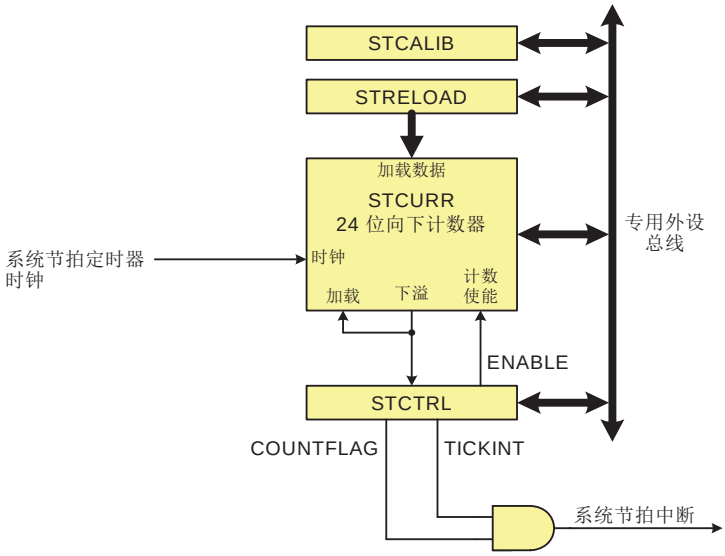


图 48. 系统节拍定时器功能框图

13.5 寄存器描述

表 181. SysTick 定时器寄存器映射 （基址 0xE000 E000）

名称	访问类型	地址偏移	描述	复位值 [1]
STCTRL	R/W	0x010	系统定时器控制和状态寄存器	0x4
STRELOAD	R/W	0x014	系统定时器重载值寄存器	0
STCURR	R/W	0x018	系统定时器当前值寄存器	0
STCALIB	R/W	0x01C	系统定时器校准值寄存器	0x4

[1] 复位值仅反映已使用位中保存的数据，不包括保留位的内容。

13.5.1 系统定时器控制和状态寄存器 (STCTRL - 0xE000 E010)

STCTRL 寄存器包含 SysTick 定时器的控制信息，并提供状态标志。

表 182. 系统定时器控制和状态寄存器 (STCTRL - 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	-	保留	1
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	系统节拍计数器标志。系统节拍计数器向下计数到 0 时会设置该标志，并通过对该寄存器进行读操作清除该标志。	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

13.5.2 系统定时器重载值寄存器 (STRELOAD - 0xE000 E014)

STRELOAD 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 STCALIB 值，则可对 STCALIB 寄存器进行读取，并将其用作 STRELOAD 寄存器的值。

表 183. 系统定时器重载值寄存器 (STRELOAD - 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

13.5.3 系统定时器当前值寄存器 (STCURR - 0xE000 E018)

当软件读取 STCURR 寄存器时，它将从系统节拍计数器返回当前计数。

表 184. 系统定时器当前值寄存器 (STCURR - 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

13.5.4 系统定时器校准值寄存器 (STCALIB - 0xE000 E01C)

有关详情，请参阅表 309。

表 185. 系统定时器校准值寄存器 (STCALIB - 0xE000 E01C) 位描述

位	符号	值	描述	复位值
23:0	TENMS		系统节拍定时器校准值	0x04
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 309。	0
31	NOREF		参见表 309。	0

14.1 本章导读

所有器件中的 WWDT 都是一样的。

14.2 特性

- 如果不能在可编程超时期限内重新载入，则在内部重置芯片。
- 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，要求禁用硬件重置或看门狗重置。
- 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。
- 指示看门狗重置的标志。
- 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。
- 具有调试模式。

14.3 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。

14.4 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 WDTC 寄存器中设置看门狗定时器固定的重新载入值。
- 在 WDMOD 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 WDWINDOW 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 WDWARNINT 寄存器中设置看门狗警告中断的值。
- 向 WDFEED 寄存器写入 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重置条件。 WDTOF 标志必须通过软件清除。

如果看门狗定时器配置为生成警告中断，则当计数器与 WDWARNINT 寄存器定义的值匹配时，会发生中断。

14.5 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 4），供 APB 访问看门狗寄存器使用。 WDCLK 由图 4 中的 wdt_clk 生成，供看门狗定时器计数使用。 IRC 或看门狗振荡器可用作 wdt_clk。

这两个时钟域之间有一些同步逻辑。当 WDMOD 和 WDTC 寄存器通过 APB 操作更新时，新值将在 WDCLK 时钟域逻辑的 3 个 WDCLK 周期后生效。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。

如果没有使用看门狗振荡器，则可在 PDRUNCFG 寄存器（第 3.5.27 节）中将其关闭。为了节能，可在 SYSAHBCLKCTRL 寄存器（第 3.5.14 节）将输入到看门狗寄存器模块的时钟 (PCLK) 禁用。

14.6 寄存器描述

看门狗定时器包含表 186 所示的寄存器。

表 186. 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]
WDMOD	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
WDTC	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
WDFEED	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
WDTV	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
WDCLKSEL	R/W	0x010	看门狗时钟选择寄存器。	0
WDWARNINT	R/W	0x014	看门狗警告中断比较值。	0
WDWINDOW	R/W	0x018	看门狗窗口比较值。	0xFF FFFF

[1] 复位值仅反映已使用位中保存的数据，不包括保留位的内容。

14.6.1 看门狗模式寄存器 (WDMOD)

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 187. 看门狗模式寄存器 （WDMOD - 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		看门狗重置使能位。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
4	WDPROTECT		看门狗更新模式。此位一旦写入 1，便不能重新写入 0。	0
		0	看门狗重新载入值 (WDTC) 可随时改变。	
		1	看门狗重新载入值 (WDTC) 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，还将阻止切换此位到已禁用或已掉电的时钟源。	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重置或看门狗定时器重置来清除。

WDTOF 在看门狗超时、发生输入错误，或当 WDPROTECT =1 且试图写入 WDTC 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 WDWARNINT 规定的值时，设置该看门狗中断标志。此标志在发生重置时清除，并通过软件向此位写入 0 来清除。

在看门狗运行并具有操作时钟源时会发生看门狗复位或中断。任何时钟源都在睡眠模式下工作。

表 188. 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，但不会生成看门狗重置。 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	看门狗重置模式：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。

14.6.2 看门狗定时器常量寄存器 (WDTC)

此 WDTC 寄存器确定超时值。每次发生输入序列时，WDTC 的值就会载入看门狗定时器。WDTC 复位为 0x00 00FF。写入值低于 0xFF 将导致 0x00 00FF 被载入 WDTC。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 WDTC 的值，将导致看门狗复位并设置 WDTOF 标志。

表 189. 看门狗定时器常量寄存器 (WDTC - 0x4000 4004) 位描述

位	符号	描述	复位值
23:0	计数	看门狗超时间隔。	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

14.6.3 看门狗输入寄存器 (WDFEED)

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重置。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，重置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 190. 看门狗输入寄存器 (WDFEED - 0x4000 4008) 位描述

位	符号	描述	复位值
7:0	输入	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

14.6.4 看门狗定时器值寄存器 (WDTV)

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 191. 看门狗定时器值寄存器 (WDTV - 0x4000 400C) 位描述

位	符号	描述	复位值
23:0	计数	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

14.6.5 看门狗时钟选择寄存器 (WDCLKSEL)

表 192. 看门狗时钟选择寄存器 (WDCLKSEL - 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		如果该位为 1，则对该寄存器进行写操作不会影响 CLKSEL 位，因此也无法更改时钟源。一旦设置了 LOCK 位，该位就无法被清除。	0

14.6.6 看门狗定时器警告中断寄存器 (WDWARNINT)

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 193. 看门狗定时器警告中断寄存器 (WDWARNINT - 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

14.6.7 看门狗定时器窗口寄存器 (WDWINDOW)

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 重置到可能的最大 WDTV 值，因此窗口操作无效。

表 194. 看门狗定时器窗口寄存器 (WDWINDOW - 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

14.7 功能框图

看门狗的功能框图如下面的图 49 所示。功能框图中未显示同步逻辑 (PCLK - WDCLK)。

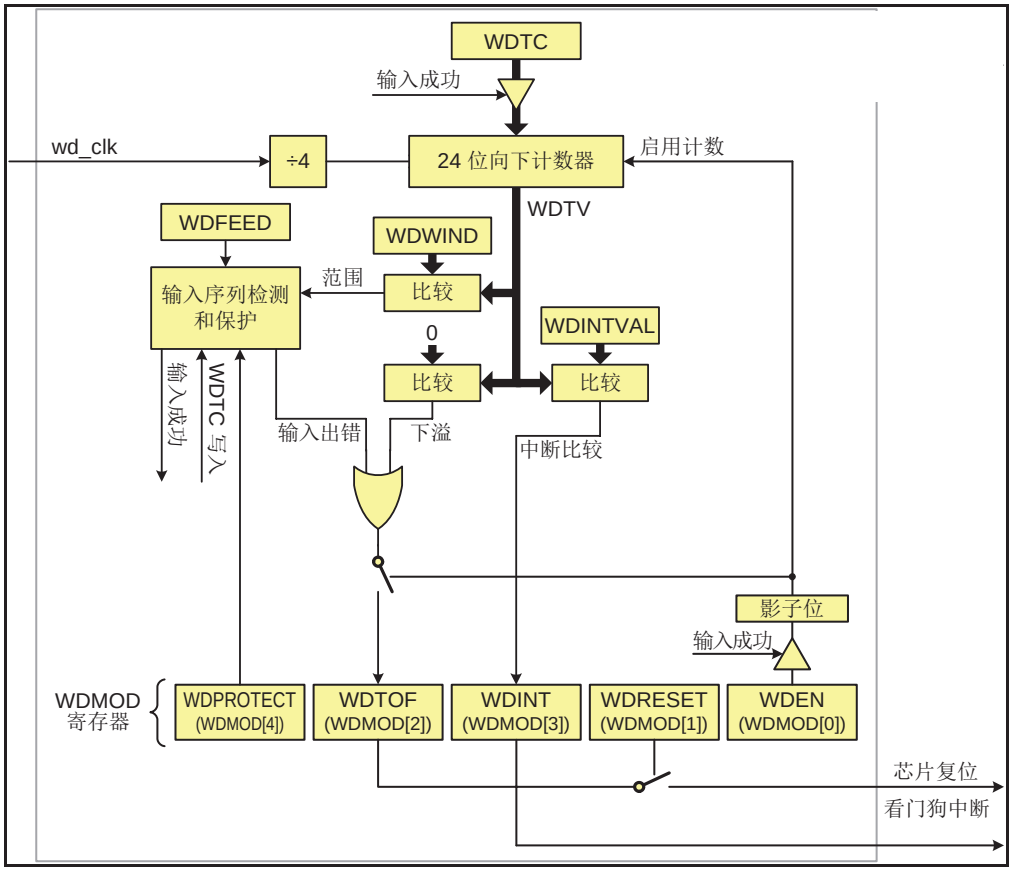


图 49. 看门狗功能框图

14.8 看门狗定时示例

下图描绘了看门狗定时器工作 （如下面的图 50 所示）的几个方面。

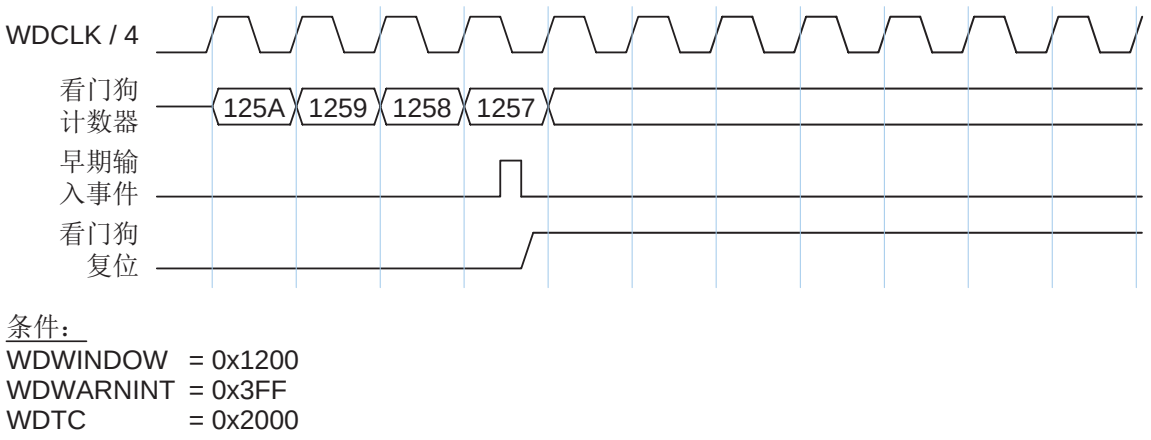


图 50. 使能窗口模式时提前看门狗输入

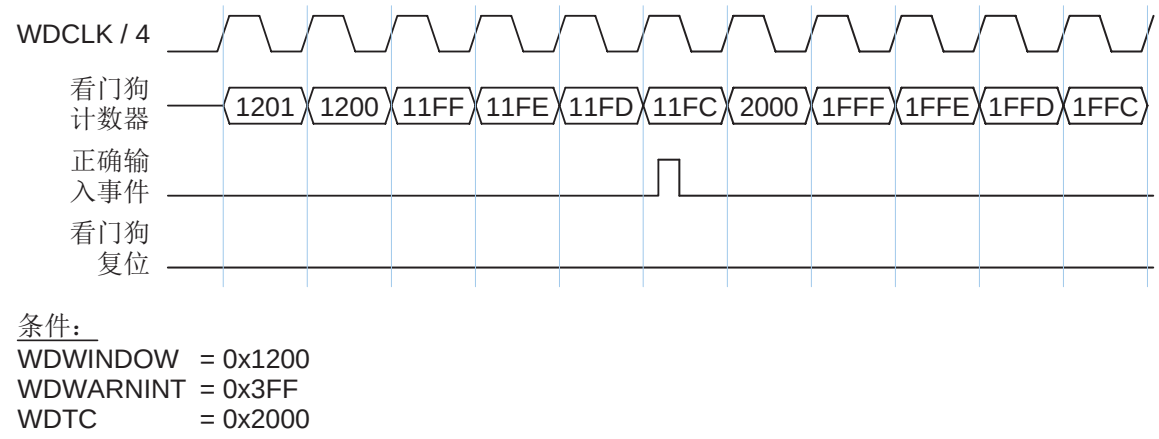


图 51. 使能窗口模式时正确的看门狗输入

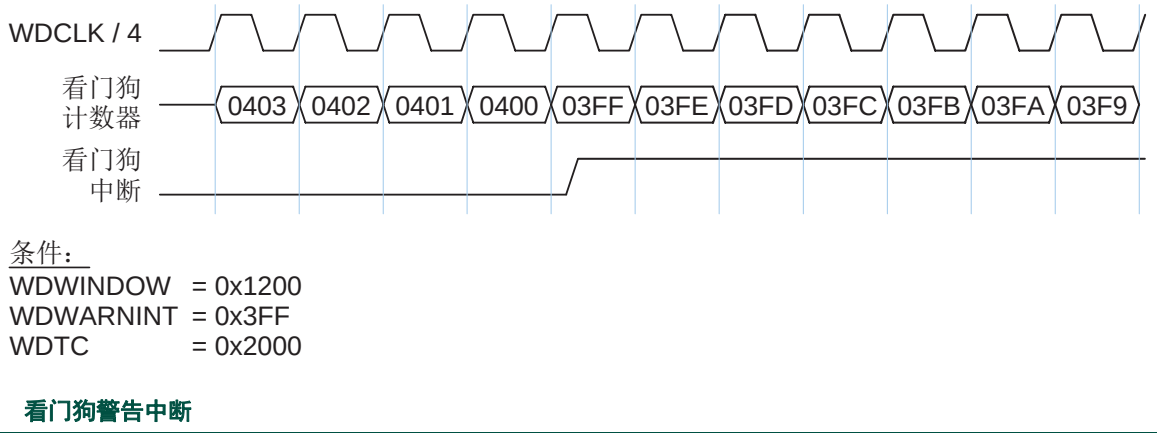


图 52. 看门狗警告中断

15.1 本章导读

所有器件中的温度传感器都是一样的。

15.2 基本配置

通过将 PDRUNCFG 寄存器 （表 32）中的 TS_PD 位设置为 0 来使能温度传感器的电源。

15.3 特性

- 在器件温度范围内， ± 2.3 mV 内的线性输出
- 功耗约 3 uA
- 在器件电压范围内几乎与 V_{DD} 独立

15.4 简介

温度传感器在允许范围内输出与器件温度成反比变化的模拟电压。

15.5 操作

与温度传感器有关的唯一控制位是下文和 “系统控制”章节中描述的电源控制位。 A/D 转换器和模拟比较器都可使用该温度传感器输出。

要用 ADC 来精确测量温度传感器，则必须在单通道连发模式下对 ADC 进行配置。 9 转换（或更多）连发的最后一个值提供精确结果。

温度传感器上电后需要一些时间来稳定并输出一个正确表示器件温度的电压。将 A/D 转换器或模拟比较器切换为使用传感器输出后，适用更短的建立时间。软件可通过 A/D 转换器反复转换和读取温度传感器输出，或简单读取模拟比较器，直到获得恒定的结果来处理这两种因素。

15.6 寄存器描述

该温度传感器无用户可配置寄存器，但系统控制模块中的电源控制除外。参见表 26。

16.1 本章导读

所有 LPC11Axx 器件中的 ADC 模块都是一样的。

16.2 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 83）。
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 32）。

注：A/D 转换器的时钟由 APB 时钟确定 (PCLK)。A/D 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 4.5 MHz（最大）时钟。准确的转换要求有 11 个时钟周期。

16.3 特性

- 10 位逐次逼近型模数转换器 (ADC)。
- 输入在 8 个引脚和 3 个内部源中多路复用。
- 掉电模式。
- 测量范围：0 至 3.6 V。不要超过 V_{DD} 电压电平。
- 10 位转换时间 ≥ 2.44 µs。
- 用于单个或多个输入的连发转换模式。
- 输入引脚 ATRG0 或 ATRG1、定时器匹配信号或比较器输出跳变的选择性转换。
- 每个 A/D 通道的独立结果寄存器可减少中断开销。

16.4 引脚说明

表 195. 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}。
V_{DD}/$V_{DD(3V3)}$	输入	V_{REF}; 基准电压。
ATRG0/1	输入	启动转换的输入触发。 **注：**这些输入信号以及所有其他转换触发信号的最短保持时间必须为三个系统时钟周期。

必须通过 IOCON 寄存器选择 ADC 功能，以获得准确的电压读数。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，就会有一个内部电路将 ADC 硬件与相关引脚断开。

16.5 功能框图

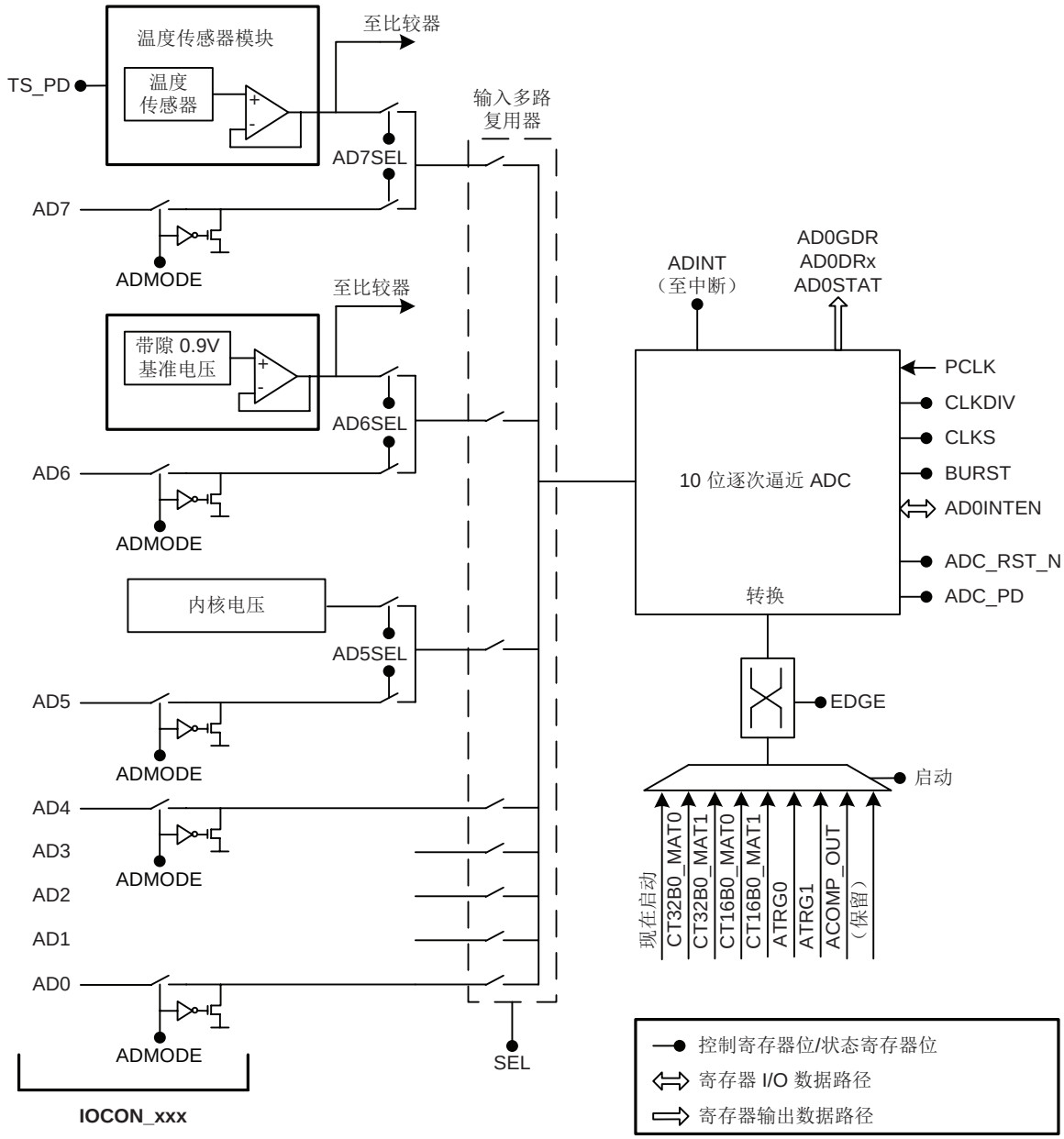


图 53. ADC 框图

16.6 寄存器描述

ADC 所包含的寄存器，其结构如表 196 所示。

表 196. 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
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。	0
GDR	RO	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
SEL	R/W	0x008	A/D 选择寄存器。在外部引脚和内部源之间选择。	0
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
DR0	RO	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
DR1	RO	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
DR2	RO	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
DR3	RO	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
DR4	RO	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
DR5	RO	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
DR6	RO	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
DR7	RO	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
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

[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。

16.6.1 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 197. 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。 如果全部位都为 0 （如复位后的情况），则会自动选择通道 0。	0
15:8	CLKDIV		将 APB 时钟 (PCLK) 进行 CLKDIV +1 分频，得到 ADC 的时钟，该时钟必须小于或等于 4.5 MHz。通常软件应编程该域中的最小值来得到 4.5 MHz 或稍低于 4.5 MHz 的时钟，但某些情况下 （例如高阻抗模拟电源）可能需要更低的时钟。	0
16	BURST		连发模式	0
		0	软件控制模式：转换由软件控制并要求有 11 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，1s内扫描（如果有必要）SEL 字段中被选的引脚。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。	
19:17	CLKS		此字段选择连发模式下每个转换使用的时钟数量，以及 ADDR 的 LS 位中的决定结果准确性的位的数量，其数量介于 11 个时钟 （10 位）和 4 个时钟 （3 位）之间。	000
		0x0	11 个时钟 /10 位	
		0x1	10 个时钟 /9 位	
		0x2	9 个时钟 /8 位	
		0x3	8 个时钟 /7 位	
		0x4	7 个时钟 /6 位	
		0x5	6 个时钟 /5 位	
		0x6	5 个时钟 /4 位	
		0x7	4 个时钟 /3 位	
22:20	-		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
26:23	START		当 BURST 位为 0 时，这些位控制 A/D 转换是否启动及何时启动。保留所有其它值。	0
		0x0	未开始 （将 PDN 清除为 0 时应使用该值）。	
		0x2	立即开始转换。	
		0x4	当位 27 选择的边沿出现在 ATRG0 时启动转换。	
		0x5	当位 27 选择的边沿出现在模拟比较器输出时启动转换。	
		0x6	当位 27 选择的边沿出现在 ATRG1 时启动转换。	
		0x8	当位 27 选择的边沿出现在 CT32B0_MAT0 时启动转换 [1]。	
		0xA	当位 27 选择的边沿出现在 CT32B0_MAT1 时启动转换 [1]。	
		0xC	当位 27 选择的边沿出现在 CT16B0_MAT0 时启动转换 [1]。	
		0xE	当位 27 选择的边沿出现在 CT16B0_MAT1 时启动转换 [1]。	
27	EDGE		该位只有在 START 字段包含 0100-1110 时有效。在以下情况下：	0
		0	在选择信号的上升沿开始转换。	
		1	在选择信号的下降沿开始转换。	
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。

16.6.2 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 198. 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
5:0	-	保留。这些位始终读为零。	0
15:6	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。该字段为 0 表示 ADn 引脚处的电压小于、等于或接近于 V_{SS}，而 0x3FF 表明 ADn 引脚处的电压接近于、等于或大于 V_{REF}。	X
23:16	-	保留。这些位始终读为零。它们允许在无 AND 屏蔽的情况下累加连续的 A/D 值 （最少 256 个值），而不会溢出到 CHN 字段。	0
26:24	CHN	这些位包含 LS 位转换的源通道。	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 LS 位结果的转换之前，一个或多个转换的结果丢失或被覆写，则该位为 1。在非 FIFO 操作中，读取该寄存器时，该位清除。	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

16.6.3 A/D 选择寄存器 (SEL)

A/D 选择寄存器将 A/D 转换器的范围扩展到了内部信号以及外部输入，控制着在两者之间进行选择的模拟开关。

表 199. A/D 选择寄存器 （SEL - 地址 0x4001 C008）位描述

位	符号	值	描述	复位值
9:0	-		保留。始终将 0 写入这些位。	X
11:10	AD5SEL		该字段选择通道 5 的源信号。	00
		0x0	AD5 引脚	
		0x1	无连接或负载	
		0x2	内核调压器输出 （1.2V 至 1.8V）	
		0x3	保留。请不要编写该值。	
13:12	AD6SEL		该字段选择通道 6 的源信号。	00
		0x0	AD6 引脚	
		0x1	无连接或负载	
		0x2	内部基准电压	
		0x3	保留。请不要编写该值。	
15:14	AD7SEL		该字段选择通道 7 的源信号。	00
		0x0	AD7 引脚	
		0x1	无连接或负载	
		0x2	温度传感器	
		0x3	保留。请不要编写该值。	
31:16	-		保留。始终将 0 写入这些位。	X

16.6.4 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。每个 A/D 通道的 ADDRn 寄存器中的 DONE 和 OVERRUN 标志都反映在 ADSTAT 中。中断标志（所有 DONE 标志的逻辑 OR）也可在 ADSTAT 中找到。

表 200. 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 通道的结果寄存器中的 OVERRUN 状态标志。读取 ADSTAT 允许同时检查所用 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。始终为 0。	0

16.6.5 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 201. 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。	1
31:9	-	保留。始终为 0。	0

16.6.6 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 202. 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
5:0	-	保留。这些位始终读为零。	0
15:6	V_VREF	当 DONE 为 1 时，此字段包含一个二进制小数，表示的是 ADn 引脚的电压除以 V_{REF} 引脚上的电压。该字段为 0 表示 ADn 引脚处的电压小于、等于或接近于 V_{REF}，而 0x3FF 表明 AD 输入处的电压接近于、等于或大于 V_{REF}。	不适用
29:16	-	保留。这些位始终读为零。它们允许在无 AND 屏蔽的情况下累加连续的 A/D 值 （最少 256 个值），而不会溢出到 CHN 字段。	0
30	OVERRUN	在连发模式下，如果在产生 LS 位结果转换之前一个或多个转换结果丢失或被覆盖，则该位为 1。读取该寄存器可清除该位。	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

16.7 功能说明

16.7.1 硬件触发转换

如果 ADCR0 中的 BURST 位为 0，且 START 字段包含 0100-1110， 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。

16.7.2 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。

16.7.3 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。在引脚的 IOCON 寄存器中清除 ADMODE 位，可通过禁用引脚的数字接收器，提高转换精度 （参见第 6.3.5 节）。

16.7.4 采样 / 转换频率

对于使用触发信号来启动转换并且需要精确采样频率 / 转换的应用来说，应确保触发信号周期是 ADC 时钟周期的整数倍。

17.1 本章导读

尽管可用的触发输入数各有不同，但所有 PC11Axx 器件中的 DAC 模块都是一样的。

17.2 基本配置

DAC 是使用以下寄存器进行配置的：

1. 引脚：DAC 输出引脚是在 IOCON 寄存器模块中配置的 （表 83）。
2. 电源和外设时钟：在 SYSAHBCLKCTRL 寄存器中，设置位 21（表 19）。DAC 电源由 PDRUNCFG 寄存器控制 （表 32）。

17.3 特性

- 10 位数模转换器
- 电阻串结构
- 缓冲输出
- 掉电模式
- 可选择速度与功率
- 最大更新率为 1 MHz
- 转换可由内部信号或外部信号上的选定边沿来触发
- 边沿触发转换时的中断选项

17.4 引脚说明

表 203 描述了 DAC 输出。该输出可在片内使用或向外驱动至第 6.4.1 节 “I/O 配置寄存器” 中所述的选定引脚上。

表 203. D/A 引脚描述

引脚	类型	描述
AOUT	输出	**模拟输出。**转换启动后，经过所选的建立时间，该信号上的电压 （相对于 V_{SS}）为 VALUE × (V_{DD}/1024)。

17.5 寄存器描述

表 204. 寄存器简介：DAC （基址 0x4002 4000）

名称	访问类型	地址偏移	描述	复位值
CR	R/W	0x000	D/A 控制寄存器	0

[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。

17.5.1 D/A 控制寄存器 (CR)

此读 / 写寄存器包括要转换为模拟的数字值以及一个权衡性能与功耗的位。位 5:0 保留，用于今后的高分辨率 D/A 转换器。

表 205. D/A 转换器寄存器 （CR - 地址 0x4002 4000）位描述

位	符号	值	描述	复位值
5:0	-		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
15:6	VALUE		转换开始后，经过所选的建立时间，AOUT 引脚上的电压（相对于 V_{SS}）为 VALUE × (V_{DD}/1024)。	0
16	BIAS		建立时间。 BIAS 位描述中所注明的建立时间对于 AOUT 引脚上不超过 100 pF 的电容负载有效。大于该值的负载阻抗值会使建立时间长于指定时间。	0
		0	DAC 的最大建立时间为 1 µs[2]，最大电流为 700 µA。这样最大更新速率可达 1 MHz[2]。	
		1	DAC 的建立时间为 2.5 µs[2]，最大电流为 350 µA。这样最大更新速率可达 400 kHz[2]。	
19:17	TRIG		写入该字段的值确定是在对寄存器进行写操作后立即开始转换，还是将转换延迟，直到发生选定的事件。	000
		0x0	对寄存器进行写操作时开始转换， AOUT 立即开始转变为新电压。对于该字段中的所有其他值，AOUT 保持其之前电压，直到发生选定的事件。	
		0x1	转换由模拟比较器 （电平）输出上选定的边沿触发。	
		0x2	转换由 ATRG0 上选定的边沿触发。	
		0x3	转换由 ATRG1 上选定的边沿触发。	
		0x4	转换由 CT32B1_MAT0 上选定的边沿触发 [1]。	
		0x5	转换由 CT32B1_MAT1 上选定的边沿触发 [1]。	
		0x5	转换由 CT16B1_MAT0 上选定的边沿触发 [1]。	
		0x7	转换由 CT16B1_MAT1 上选定的边沿触发 [1]。	
20	-		保留。	不适用
22:21	EDGESEL		对于 TRIG 的非零值，该字段选择触发转换的时间：	00
		0x0	下降沿	
		0x1	上升沿	
		0x2	上升沿和下降沿	
		0x3	上升沿和下降沿	
23	TRIGERD		如果 TRIG 字段 （上述）为非零，则在触发转换时会设置该位，在对该寄存器进行任何写操作会清除该位。	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

[1] 请注意，该操作无需在设备引脚上出现定时器匹配功能。

[2] 如果 VDD >= 2.4V，则这些时间 / 频率可得到保证。如果低于该电压，请查阅 LPC11Axx 数据手册了解降低定额值的信息。

17.5.2 掉电控制

DAC 的电源控制在系统控制模块中实现。参见表 32。

17.6 操作

17.6.1 硬件触发转换

如果 TRIG 字段为非零，D/A 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。

17.6.2 中断

当 DACR 中的 TRIGERD 位为 1 时，会向中断控制器发出中断请求。软件可以使用中断控制器中与 DAC 对应的中断使能位来控制是否产生中断。软件可向 DACR 进行写操作来清除该请求。

17.7 功能框图

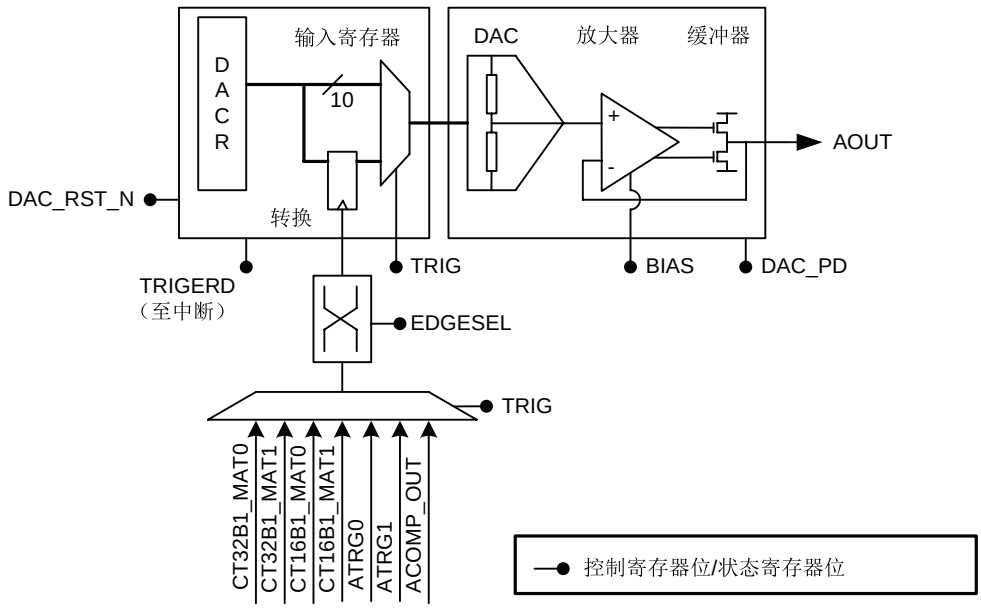


图 54. DAC 框图

18.1 本章导读

所有 LPC11Axx 器件中的模拟比较器模块都是一样的。

VDDCMP 引脚在 WLCSP 封装和 HVQFN、 LQFP 封装中的引脚分配不同。参见表 206。

表 206. VDDCMP 引脚位置

封装	VDDCMP 引脚位置
WLCSP	PIO0_5
LQFP	PIO0_14
HVQFN	PIO0_14

18.2 基本配置

模拟比较器是使用以下寄存器进行配置的：

1. 引脚：模拟比较器引脚功能是在 IOCON 寄存器模块中配置的 （表 84）。
2. 电源和外围设备时钟：在 SYSAHBCLKCTRL 寄存器中，设置位 20 （表 19）。模拟比较器电源由 PDRUNCFG 寄存器控制 （表 32）。

18.3 特性

- 可选择的外部输入可用作比较器的正输入或负输入。
- 内部基准电压和温度传感器可用作比较器的正输入或负输入。
- 32 级电压阶梯可用作比较器的正输入或负输入。
- 电压阶梯源可在 V_{DD}/$V_{DD(3V3)}$ 或 VDDCMP 引脚之间选择。
- 不需要时，电压阶梯可单独掉电。
- 中断功能

18.4 简介

模拟比较器可比较外部引脚和内部电压上的电压电平。

18.5 引脚说明

表 207. 比较器引脚描述

引脚	类型	描述
ACMP_I[5:1]	输入	比较器输入源
ACMP_O	输出	比较器输出
VDDCMP	输入	32 级电压阶梯的外部基准电压源。有关特定于封装的引脚分配，参见第 18.1 节。

18.6 寄存器描述

表 208. 寄存器简介：模拟比较器 （基址 0x4002 8000）

名称	访问类型	地址偏移	描述	复位值
CTL	R/W	0x000	比较器控制寄存器	0
LAD	R/W	0x004	电压阶梯寄存器	0

18.6.1 比较器控制寄存器 (CTL)

该寄存器使能比较器，配置中断，并控制比较器两侧的输入多路复用器。表 209 中未显示的所有位均保留并且应当写为 0。

表 209. 比较器控制寄存器 （CTL，地址 0x4002 8000）位描述

位	符号	值	描述	复位值
2:0	-		保留。写为 0。	0
4:3	EDGESEL		该字段控制比较器输出上的哪些边沿设置 COMPEDGE 位 （下面的位 23）： 00 = 下降沿 01 = 上升沿 1x = 上升沿和下降沿	0
		0x0	下降沿	
		0x1	上升沿	
		0x2	上升沿和下降沿	
		0x3	上升沿和下降沿	
5	-		保留。写为 0。	0
6	COMPSA		比较器输出控制	0
		0	直接使用比较器输出。	
		1	比较器输出与总线时钟同步以输出到其他模块。	
7	-		保留。写为 0。	0

表 209. 比较器控制寄存器 （CTL，地址 0x4002 8000）位描述 *（续）*

位	符号	值	描述	复位值
10:8	COMP_VP_SEL		选择正电压输入	0
		0x0	电压阶梯输出	
		0x1	ACMP_I1	
		0x2	ACMP_I2	
		0x3	ACMP_I3	
		0x4	ACMP_I4	
		0x5	ACMP_I5	
		0x6	内部基准电压	
		0x7	温度传感器	
13:11	COMP_VM_SEL		选择负电压输入	0
		0x0	电压阶梯输出	
		0x1	ACMP_I1	
		0x2	ACMP_I2	
		0x3	ACMP_I3	
		0x4	ACMP_I4	
		0x5	ACMP_I5	
		0x6	内部基准电压	
19:14	-		保留。写为 0。	0
20	EDGECLR		中断清除位。要清除 COMPEDGE 位并取消中断请求，可通过先写入 1 后写入 0 来切换 EDGECLR 位。	0
21	COMPSTAT		比较器状态。该位反映比较器输出的状态。	0
22	-		保留。写为 0。	0
23	COMPEDGE		比较器边沿检测状态。	0
24	-		保留。写为 0。	0
26:25	HYS		控制比较器的滞回。比较器输出某个状态时，该滞回为选定信号之间的差值，方向与输出状态相反，会切换输出。	0
		0x0	无 （该输出会在电压交叉时切换）	
		0x1	5 mV	
		0x2	10 mV	
		0x3	20 mV	
31:27	-		保留	-

18.6.2 电压阶梯寄存器 (LAD)

该寄存器使能并控制电压阶梯。可按 1/31 的步级对该阶梯产生的基准电压的小数部分进行编程。

表 210. 电压阶梯寄存器 （LAD，地址 0x4002 8004）位描述

位	符号	值	描述	复位值
0	LADEN		电压阶梯使能	0
5:1	LADSEL		电压阶梯值。基准电压 Vref 取决于下面的 LADREF 位。 00000 = V_{SS} 00001 = 1 × Vref/31 00010 = 2 × Vref/31 ... 11111 = Vref	0
6	LADREF		选择电压阶梯的基准电压 Vref：	0
		0	V_{DD}/$V_{DD(3V3)}$ 引脚	
		1	VDDCMP 引脚	
31:7	-		未使用	0

18.7 功能说明

18.7.1 功能框图

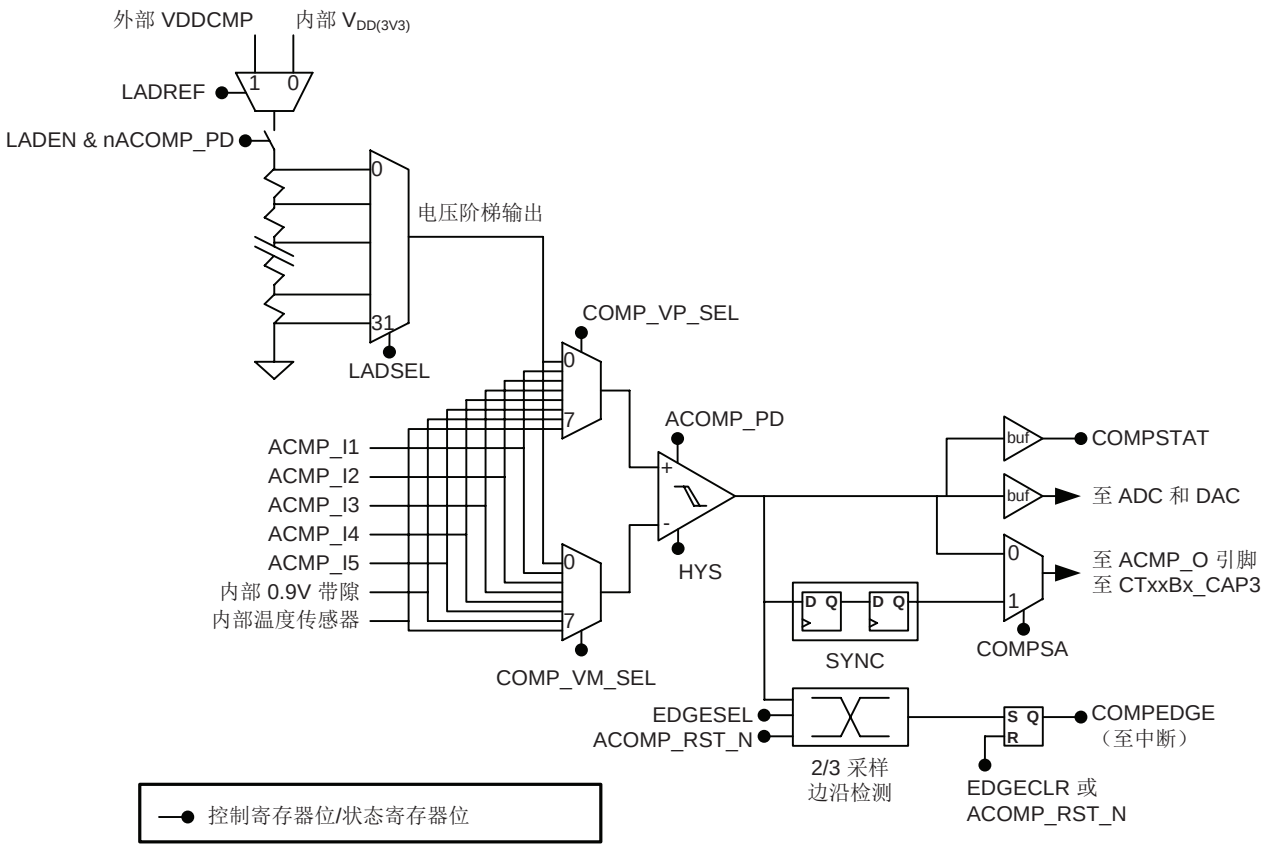


图 55. 比较器功能框图

该比较器具有 8 个单独复用到其正输入和负输入的输入。多路复用器由比较器寄存器 CTL 控制 （参见图 55 和表 209）。

多路复用器的输入 0 为可编程电压阶梯输出。

位 5:1 来自外部输入 ACMP_I[5:1]。

多路复用器的位 6 和位 7 分别为内部基准电压和温度传感器。

18.7.2 基准电压

电压阶梯可使用来自 VDDCMP 或 $V_{DD}/V_{DD(3V3)}$ 引脚的两个基准电压。电压阶梯在引脚电压和 V_{SS}（含）之间选择 32 个步级中的一个。VDDCMP 上的电压不应超过 $V_{DD}/V_{DD(3V3)}$ 上的电压。

18.7.3 建立时间

电压阶梯上电后，在使用该阶梯的比较值达到精确前需经过稳定时间。 LADSEL 值发生变化后并且两个电压源中的其中一个或全部发生变化后，适用更短的建立时间。软件可通过反复读取比较器输出，直到大量读数产生相同结果来处理这些因素。有关定量建立时间，请参见 LPC11Axx 数据手册。

18.7.4 中断

中断输出来自该模块中的边沿检测电路。上升沿、下降沿或这两种边沿都可设置COMPEDGE 位并中断请求。软件将 1 写入 EDGECLR 时， COMPEDGE 和中断请求会被清除。

18.7.5 比较器输出

比较器输出（由 COMPSA 位调节）可传送至某个外部引脚。COMPSA 为 0 且比较器中断被禁用时，如果无需对控制寄存器进行写操作，则可在总线时钟被禁用（第 3.5.14 节 “系统时钟控制寄存器”）的情况下使用比较器来省电。

可通过比较器状态寄存器位观察比较器输出的状态。

比较器输出传送至每个定时器 / 计数器的 CAP3 输入，可捕获某个电压交叉的时间或在某个方向或两个方向上对交叉进行计数。

19.1 本章导读

所有器件均采用内部基准电压。

19.2 特性

- 用于掉电检测
- 可用作 A/D 转换器和模拟比较器的输入

19.3 简介

内部基准电压是生成约 0.9 V 基准电压的带隙电路。该电路是掉电检测 (BOD) 模块的一部分，通过 A/D 转换器和模拟比较器模块，可用于相互校准。

19.4 寄存器描述

内部基准电压无可配置寄存器。

19.5 功能说明

内部基准电压上电后需要一些时间来稳定并输出恒定的电压。将 A/D 转换器或模拟比较器切换为使用基准电压输出后，适用更短的建立时间。软件可通过反复转换和读取 A/D 转换器输出，或简单读取电压比较器，直到获得恒定的结果来处理这两种因素。有关稳定方面的定量细节，请参见 LPC11Axx 数据手册。

20.1 本章导读

有关 LPC11Axx 闪存配置，请参见表 211。

表 211. LPC11Axx 的闪存配置

器件	闪存
LPC11A02UK	16 kB
LPC11A04UK	32 kB
LPC11A11	8 kB
LPC11A12	16 kB
LPC11A13	24 kB
LPC11A14	32 kB

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，见表 248。

20.2 bootloader

启动引导程序控制复位后的初始操作，并提供完成闪存编程的方法。这可能是清空设备的初始编程，已编程设备的擦除和重编程，或者是通过运行系统中的应用程序对闪存编程。

20.3 特性

- 在线系统编程：在系统编程(ISP)是使用bootloader软件和UART串行端口对片内Flash存储器的编程或重新编程。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和EEPROM存储器进行擦除及写入操作。
- 闪存访问时间可以由闪存控制器模块中的寄存器配置。

20.4 应用

启动引导程序提供了在系统和在应用编程的接口，用于对片内闪存和 EEPROM 存储器进行编程。

20.5 描述

每当部件上电或复位时， bootloader 代码即被执行一次。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。PIO0_1 引脚复位后的低电平被视为启动 ISP 命令处理程序的外部硬件请求。假设在 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。

用作 ISP 硬件请求的 PIO0_1 引脚无需特别注意。由于 PIO0_1 复位后处于上拉模式，因此默认操作是检查有效用户代码。

20.5.1 复位后的存储器映射

引导模块的大小为 16 kB。引导模块在存储器区域中的起始地址为 0x1FFF 0000。bootloader 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。RAM 的使用情况将在本章稍后作介绍。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。

20.5.1.1 有效用户代码的判定标准

有效用户代码的判定标准：保留的 Cortex-M0 异常向量位置 7（向量表中偏移量 0x 0000 001C）应包含表中条目 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 0 与主机进行同步。主机应当发送一个同步字符 ?(0x3F) 并等待响应。主机侧的串口设置应是 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。如果同步通过验证，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>” 字符串。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。

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。安全起见，在执行将导致闪存擦除 / 写入操作的命令和 “运行” 命令之前必须使用 “解锁” 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 220 页上的章节 20.7 “ISP 命令”中进行了说明。

20.5.2 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。

20.5.2.1 连接

LPC11Axx 的 PIO0_12 上的 RXD 功能以及 PIO0_13 上的 TXD 功能用于 ISP 通信。

20.5.2.2 ISP 命令格式

"Command Parameter_0 Parameter_1 ...Parameter_n<CR><LF>" "Data"（Data 只适用于写命令）。

20.5.2.3 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）。

20.5.2.4 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。

有关 UU 编码的描述，可访问 wikipedia.org/wiki/Uuencoding。

20.5.2.5 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。

20.5.2.6 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。

20.5.2.7 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。

20.5.2.8 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。

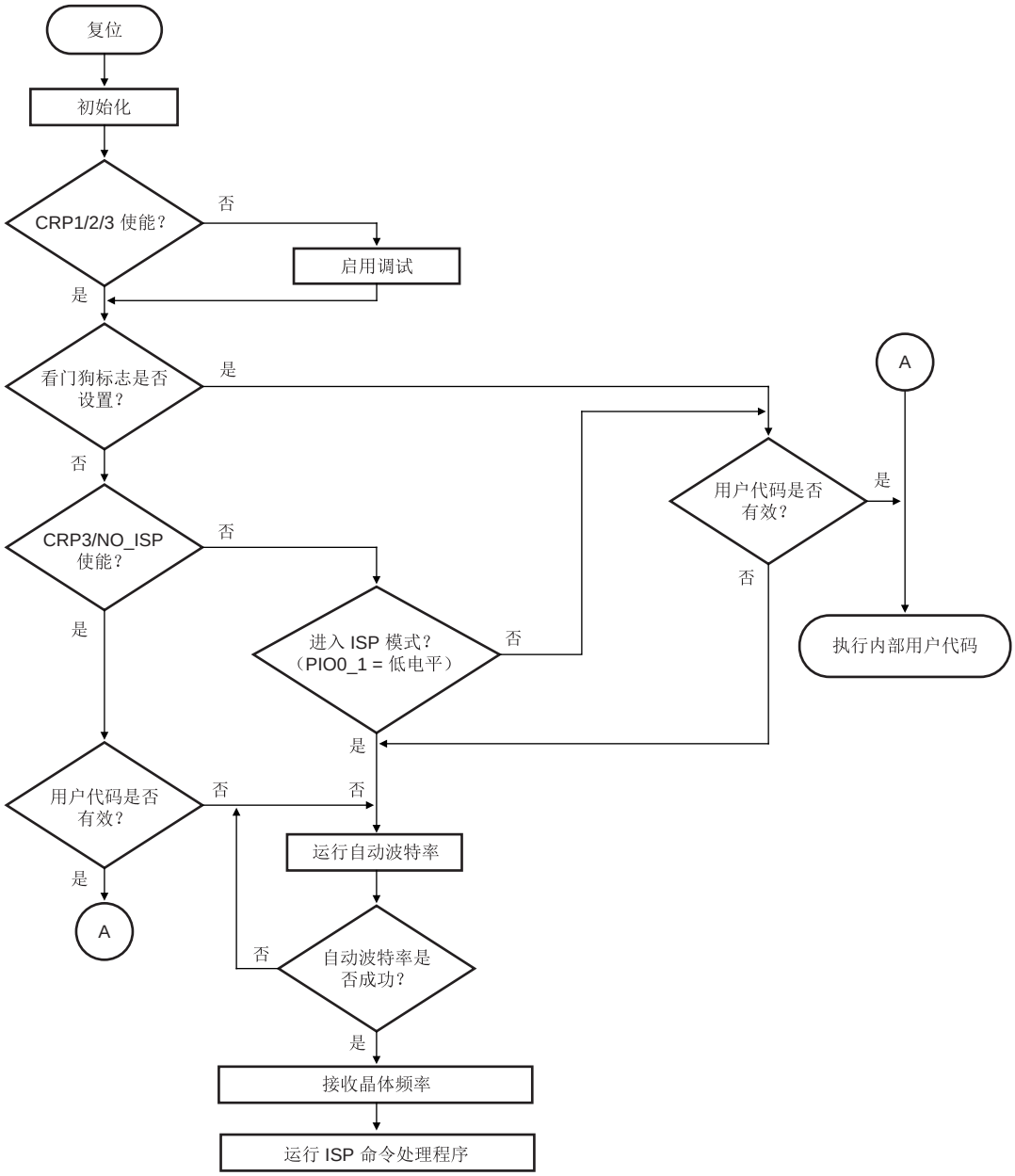
20.5.2.9 ISP 命令处理程序使用的 RAM

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 顶端减去 32 字节。可使用的最大协议栈为 256 字节，协议栈是向下递增的。

20.5.2.10 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。用户分配的协议栈空间中可使用的最大协议栈为 128 字节，协议栈是向下递增的。

20.5.3 Boot 处理流程图



(1) 有关晶频处理的详情，请参见 第 232 页上的章节 20.8.8 “重新调用 ISP”

图 56. Boot 处理流程图

20.5.4 扇区号

一些 IAP 和 ISP 命令在 “扇区” 操作，需指定扇区号。下表显示了 LPC11Axx 设备的扇区号和存储器地址间的对应关系。

表 212. LPC11Axx 器件的扇区

扇区号	扇区大小	地址范围	LPC11A02UK	LPC11A04UK	LPC11A11	LPC11A12	LPC11A13	LPC11A14
0	4 kB	0x0000 0000 - 0x0000 0FFF	是	是	是	是	是	是
1	4 kB	0x0000 1000 - 0x0000 1FFF	是	是	是	是	是	是
2	4 kB	0x0000 2000 - 0x0000 2FFF	是	是	-	是	是	是
3	4 kB	0x0000 3000 - 0x0000 3FFF	是	是	-	是	是	是
4	4 kB	0x0000 4000 - 0x0000 4FFF	-	是	-	-	是	是
5	4 kB	0x0000 5000 - 0x0000 5FFF	-	是	-	-	是	是
6	4 kB	0x0000 6000 - 0x0000 6FFF	-	是	-	-	-	是
7	4 kB	0x0000 7000 - 0x0000 7FFF	-	是	-	-	-	是

20.6 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。

重要事项：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 213. 代码读保护选项

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	禁止通过 JTAG 引脚或串行线引脚进行调试。JTAG 边界扫描有效。此模式允许使用下列 ISP 命令和限制更新部分闪存： • 写入 RAM 命令不应访问在 0x1000 0300 以下的 RAM。 0x1000 0200 以下的访问被禁用。 • “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。 • 空白检查命令仅报告空白扇区。非空白扇区的偏移和值始终被报告为 0。 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	禁止通过 JTAG 引脚或串行线引脚进行调试。JTAG 边界扫描有效。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 • 空白检查命令仅报告空白扇区。非空白扇区的偏移和值始终被报告为 0。 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	禁止通过 JTAG 引脚或串行线引脚进行调试。JTAG 边界扫描有效。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 UART 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 214. 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	是否使能 JTAG/SW 调试	器件是否进入 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 215. 各 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	是；只报告空白扇区	是；只报告空白扇区	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。

20.6.1 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。

20.7 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。“设置波特率”、“写入 RAM”、“读取存储器”和 “运行”命令不遵守此规则。

表 216. ISP 命令总结

ISP 命令	用法	说明见：
解锁	U < 解锁代码 >	表 217
设置波特率	B < 波特率 > < 停止位 >	表 218
回标	A < 设定 >	表 219
写入 RAM	W < 起始地址 > < 字节数 >	表 220
读取内存	R < 地址 > < 字节数 >	表 221
准备写操作扇区	P < 起始扇区号 > < 结束扇区号 >	表 222
将 RAM 内容复制到 Flash	C <Flash 地址 > <RAM 地址 > < 字节数 >	表 223
运行	G < 地址 > < 模式 >	表 224
擦除扇区	E < 起始扇区号 > < 结束扇区号 >	表 225
空白检查扇区	I < 起始扇区号 > < 结束扇区号 >	表 226
读取部件 ID	J	表 227
读 Boot 代码版本	K	表 229
比较	M < 地址 1> < 地址 2> < 字节数 >	表 230
读 UID	N	表 231

20.7.1 解锁 < 解锁代码 >

表 217. 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。

20.7.2 设置波特率 < 波特率 > < 停止位 >

表 218. ISP 设置波特率命令

命令	B
输入	波特率：9600 | 19200 | 38400 | 57600 | 115200 | 230400 停止位：1 | 2
返回码	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。

20.7.3 回应 < 设定 >

表 219. 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。

20.7.4 写入 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 220. 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。

20.7.5 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 221. 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。

20.7.6 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 222. 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。

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

表 223. 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。在此命令前应执行 “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。
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。

20.7.8 运行 < 地址 > < 模式 >

表 224. 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。该命令必须配合 0x0000 0200 或以上的地址使用。
示例	"G 512 T<CR><LF>" 跳转到 Thumb 模式下的地址 0x0000 0200 处。

20.7.9 擦除扇区 < 起始扇区号 > < 结束扇区号 >

表 225. 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。

20.7.10 扇区查空 < 起始扇区号 > < 结束扇区号 >

表 226. 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。**
示例	“I 2 3<CR><LF>”对 Flash 扇区 2 和 3 进行查空。

20.7.11 读器件标识号

表 227. 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 228）。
描述	此命令用于读取部件识别号。

表 228. LPC11Axx 器件标识号

设备	Hex 代码
LPC11A14FBD48/301	0x35A0 002B
LPC11A14FHN33/301	0x35A0 002B

20.7.12 读 Boot 代码版本号

表 229. ISP 读取启动代码版本命令

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。

20.7.13 比较 < 地址 1> < 地址 2> < 字节数 >

表 230. 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。

20.7.14 读 UID

表 231. 读 UID 命令

命令	N
输入	无
返回码	CMD_SUCCESS 后跟 E 类测试信息的 4 个 32 位字 （ASCII 格式）。先发送低位地址的字。
描述	该命令用于读唯一的 ID。

20.7.15 ISP 返回代码

表 232. 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。

表 232. ISP 返回代码总览 *（续）*

返回码	助记符	描述
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。

20.8 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 57。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 内容复制到闪存” 命令。结果的最大数目为 4，由 “ReadUID” 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 位置的第 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。其它参数在协议栈上传递。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 233. IAP 命令总览

IAP 命令	命令代码	说明见：
准备写操作扇区	50_{10}	表 234
将 RAM 内容复制到 Flash	51_{10}	表 235
擦除扇区	52_{10}	表 236
空白检查扇区	53_{10}	表 237
读取部件 ID	54_{10}	表 238
读 Boot 代码版本	55_{10}	表 239
比较	56_{10}	表 240
重新调用 ISP	57_{10}	表 241
读 UID	58_{10}	表 242
EEPROM 写	61_{10}	表 243
EEPROM 读	62_{10}	表 244

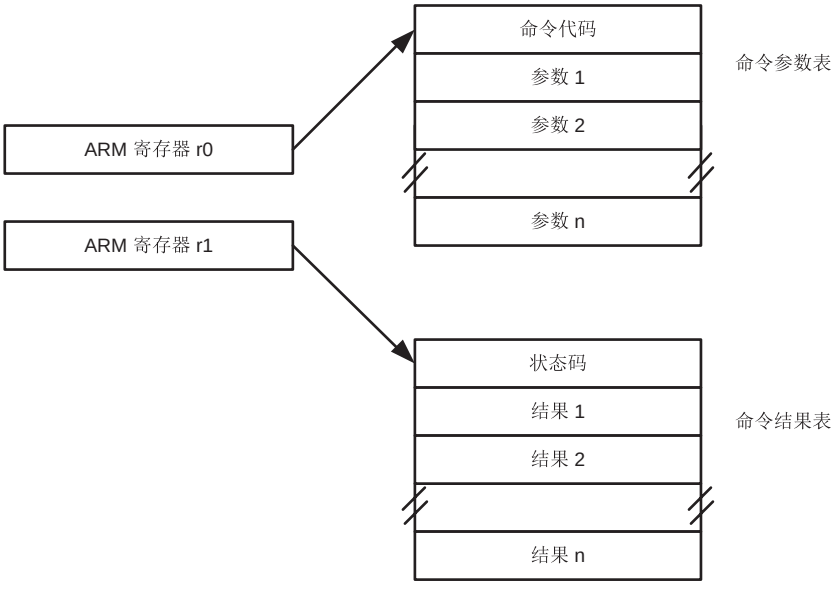


图 57. IAP 参数传递

20.8.1 准备写操作扇区

此命令使闪存写 / 擦除操作分为两步。

表 234. 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_{10}** **参数 0：**起始扇区号 **参数 1：**结束扇区号 （应当大于或等于起始扇区号）。
返回码	CMD_SUCCESS | BUSY | INVALID_SECTOR
结果	无
描述	该命令必须在执行 “将 RAM 内容复制到闪存”或 “擦除扇区”命令之前执行。“将 RAM 内容复制到闪存”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。

20.8.2 将 RAM 内容复制到 Flash

表 235. 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_{10}** **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。

20.8.3 擦除扇区

表 236. IAP 擦除扇区命令

命令	擦除扇区
输入	**命令代码：52_{10}** **参数 0：**起始扇区号 **参数 1：**结束扇区号 （应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。

20.8.4 空白检查扇区

表 237. IAP 扇区查空命令

命令	空白检查扇区
输入	**命令代码：53_{10}** **参数 0：**起始扇区号 **参数 1：**结束扇区号 （应当大于或等于起始扇区号）。
返回码	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。

20.8.5 读器件标识号

表 238. IAP 读器件标识命令

命令	读取部件识别号
输入	**命令代码：54_{10}** **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。

20.8.6 读 Boot 代码版本号

表 239. IAP 读取启动代码版本命令

命令	读 Boot 代码版本号
输入	**命令代码：55_{10}** **参数：**无
返回码	CMD_SUCCESS |
结果	**结果 0：**2 字节 Boot 代码版本号 （ASCII 格式）。将其解释为 < 字节 1 （主） >.< 字节 0 （次） >
描述	此命令用于读取引导代码版本号。

20.8.7 比较 < 地址 1> < 地址 2> < 字节数 >

表 240. IAP 比较命令

命令	比较
输入	**命令代码：56_{10}** **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。**

20.8.8 重新调用 ISP

表 241. 重新调用 ISP

命令	比较
输入	**命令代码：57_{10}**
返回码	无
结果	**无。**
描述	该命令用来调用 ISP 模式中的引导装载程序。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（见表 106）。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。

20.8.9 读 UID

表 242. IAP 读 UID 命令

命令	比较
输入	**命令代码：58_{10}**
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。

20.8.10 写 EEPROM

表 243. IAP 写 EEPROM 命令

命令	比较
输入	**命令代码：61_{10}**
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。

20.8.11 读 EEPROM

表 244. IAP 读 EEPROM 命令

命令	比较
输入	**命令代码：62_{10}**
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。

20.8.12 IAP 状态代码

表 245. 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。

20.9 调试注意事项

20.9.1 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-M0 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 246. 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

20.9.2 串行线调试 (SWD)Flash 编程接口

调试工具可将一部分闪存镜像写入 RAM，然后按照正确的偏移地址重复调用 IAP 命令“将 RAM 内容复制到闪存”。

20.10 闪存控制器寄存器

表 247. 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 248
FMSSTART	R/W	0x020	签名起始地址寄存器	0	表 249
FMSSTOP	R/W	0x024	签名停止地址寄存器	0	表 250
FMSW0	R	0x02C	字 0 [31:0]	-	表 251
FMSW1	R	0x030	字 1 [63:32]	-	表 252
FMSW2	R	0x034	字 2 [95:64]	-	表 253
FMSW3	R	0x038	字 3 [127:96]	-	表 254
EEMSSTART	R/W	0x9C	EEPROM BIST 起始地址寄存器	0x0	表 255
EEMSSTOP	R/W	0xA0	EEPROM BIST 停止地址寄存器	0x0	表 256
EEMSSIG	R	0xA4	EEPROM 24 位 BIST 签名寄存器	0x0	表 257
FMSTAT	R	0xFE0	签名生成状态寄存器	0	表 258
FMSTATCLR	W	0xFE8	签名生成状态清除寄存器	-	表 259

20.10.1 闪存访问寄存器

根据系统时钟频率，通过写入 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 LPC11U1x 闪存无法正常工作。

表 248. 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。**	-

20.10.2 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。

20.10.3 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 249 和表 250 分别显示了 FMSSTART 和 FMSSTOP 寄存器中的位分配。

表 249. 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 250. 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

20.10.4 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 20.10.10 节中说明。

表 254 分别显示了寄存器 FMSW0、 FMSW1、 FMSW2 和 FMSW3 的位分配。

表 251. 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 252. 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 253. 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 254. 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）。	-

20.10.5 EEPROM BIST 起始地址寄存器

EEPROM BIST 起始地址寄存器用于编程 BIST 的起始地址。BIST 期间，通过 16 位读操作访问 EEPROM 设备，从而将地址的 LSB 固定为 0。

表 255. EEPROM BIST 起始地址寄存器 （EEMSSTART - 地址 0x4003 C09C）位描述

位	符号	描述	复位值	访问类型
13:0	STARTA	BIST 起始地址： 位 0 固定为 0，因为只允许偶数地址。	0x0	R/W
31:14	-	保留	0x0	-

20.10.6 EEPROM BIST 停止地址寄存器

EEPROM BIST 停止地址寄存器用于编程 BIST 的停止地址，以及启动 BIST。BIST 期间，通过 16 位读操作访问 EEPROM 设备，从而将地址的 LSB 固定为 0。

表 256. EEPROM BIST 停止地址寄存器 （EEMSSTOP - 地址 0x4003 C0A0）位描述

位	符号	描述	复位值	访问类型
13:0	STOPA	BIST 停止地址： 位 0 固定为 0，因为只允许偶数地址。	0x0	R/W
29:14	-	保留	0x0	-
30	DEVSEL	BIST 设备选择位 0：在总存储器空间范围内生成 BIST 签名。由于使用多个设备时，页在EEPROM设备上交错并联，因此会在多个设备的存储器范围内生成签名。 1：仅在位于单个EEPROM设备的存储器范围内生成BIST签名。因此，生成内部地址时，要使地址的 CS 位保持稳定，以便只选择相同的设备。地址的 MSB 和 LSB 位用于按步通过起始和停止地址字段指定的存储器范围。 注：如果设置该位，起始和停止地址字段的编程必须确保它们都寻址同一EEPROM设备。因此，起始和停止地址处的地址 CS 位必须相同。	0x0	R/W
31	STRTBIST	BIST 起始位 设置该位将启动 BIST。此位是自清除的。	0x0	R/W

20.10.7 EEPROM 签名寄存器

EEPROM BIST 签名寄存器返回内置签名生成器生成的签名。

表 257. EEPROM BIST 签名寄存器 （EEMSSIG - 地址 0x4003 C0A4）位描述

位	字段名称	描述	复位值	访问类型
15:0	DATA_SIG	仅从数据字节求得的 BIST 16 位签名	0x0	R
31:16	PARITY_SIG	仅从数据字节的奇偶校验位求得的 BIST 16 位签名	0x0	R

20.10.8 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 258. 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

20.10.9 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 259. 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

20.10.10 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 58 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 58. 生成 128 位签名的算法

21.1 本章导读

所有 LPC11Axx 器件都提供整数除法程序。

21.2 特性

- 性能优化的有符号 / 无符号整数除法。
- 性能优化的有符号 / 无符号整数带余数除法。
- 基于 ROM 的程序，可减少代码大小。
- 支持最多 32 位的整数。
- ROM 调用可轻松添加到 EABI 兼容函数，以使 C 语言中的 “/” 和 “%” 算符过载。

21.3 简介

整数除法程序执行算术整数除法运算，并可通过简单的 API 调用在应用程序代码中调用。

使用以下函数原型：

```
typedef struct { int quot; int rem; } idiv_return;

typedef struct { unsigned quot; unsigned rem; } uidiv_return;

typedef struct {
     /* Signed integer division */
     int             (*sidiv)(int numerator, int denominator);
     /* Unsigned integer division */
     unsigned(*uidiv) (unsigned numerator, unsigned denominator);
     /* Signed integer division with remainder */
     idiv_return(*sidivmod)(int numerator, int denominator);
     /* Unsigned integer division with remainder */
     uidiv_return (*uidivmod)(unsigned numerator, unsigned denominator);
} LPC_ROM_DIV_STRUCT;
```

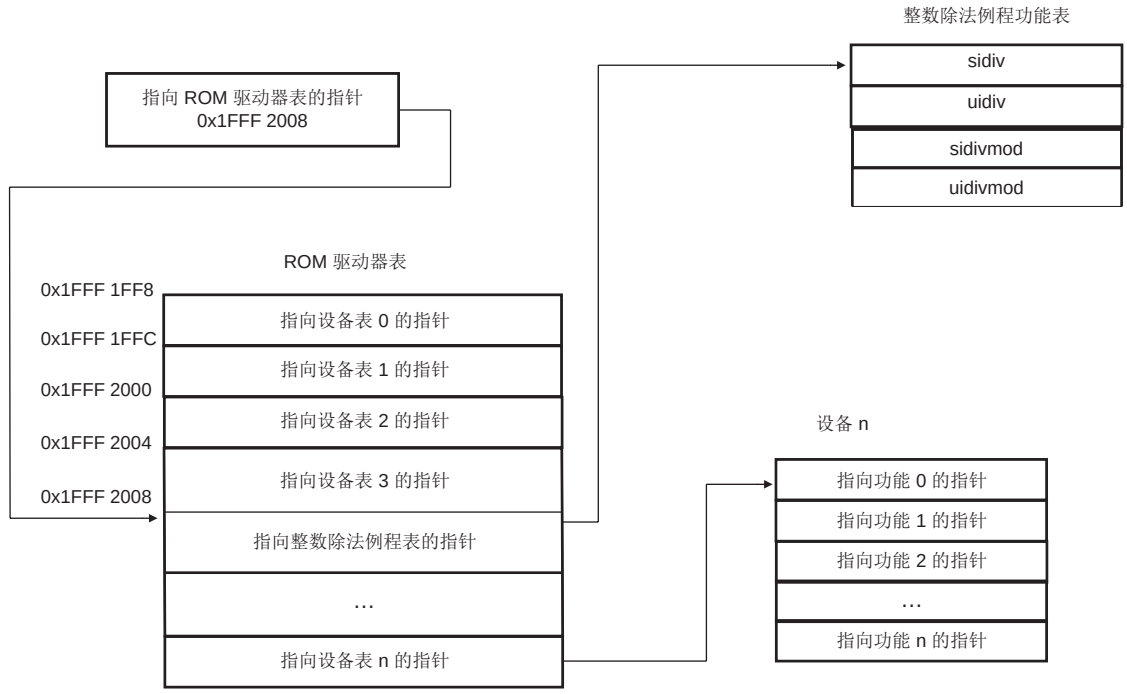


图 59. 整数除法程序指针结构

21.4 API 描述

21.4.1 示例

21.4.1.1 初始化

下面列出的示例 C 代码显示了如何初始化 API 的 ROM 表指针。

```
typedef struct _ROM {
      const unsigned p_dev1;
      const unsigned p_dev2;
      const unsigned p_dev3;
      const PWRD *pPWRD;
      const LPC_ROM_DIV_STUCT * pROMDiv;
      const unsigned p_dev4;
      const unsigned p_dev5;
      const unsigned p_dev6;
ROM ** rom = (ROM **) 0x1FFF1FF8;
      pDivROM = (*rom)->pROMDiv;

} ROM;
```

21.4.1.2 有符号除法

下面列出的示例 C 代码显示了如何通过 ROM API 执行有符号整数除法。

```
/* Divide (-99) by (+6) */
int32_t result;
result = pDivROM->sidiv(-99, 6);
/* result now contains (-16) */
```

21.4.1.3 带余数的无符号除法

下面列出的示例 C 代码显示了如何通过 ROM API 执行带余数的无符号整数除法。

```
/* Modulus Divide (+99) by (+4) */
uidiv_return result;
result = pDivROM-> uidivmod (+99, 4);
/* result.quot contains (+24) */
/* result.rem contains (+3) */
```

22.1 本章导读

所有 LPC11Axx 器件都提供 I2C 总线驱动程序。

22.2 特性

- 简单的 I2C 驱动器，用于在 I2C 总线上发送和接收数据。
- 轮询和中断驱动的接收和发送功能，用于主机和从机模式。

22.3 简介

这些驱动器是可调用的，供应用程序用来在 I2C 总线上发送或接收数据。这些驱动器使应用程序员能快速、方便地使用 I2C 接口来生成工作项目。

用户通过 ROM 例程可将 I2C 接口作为主机或从机来操作。在传输过程中，软件例程不会执行主机切换到从机模式的仲裁

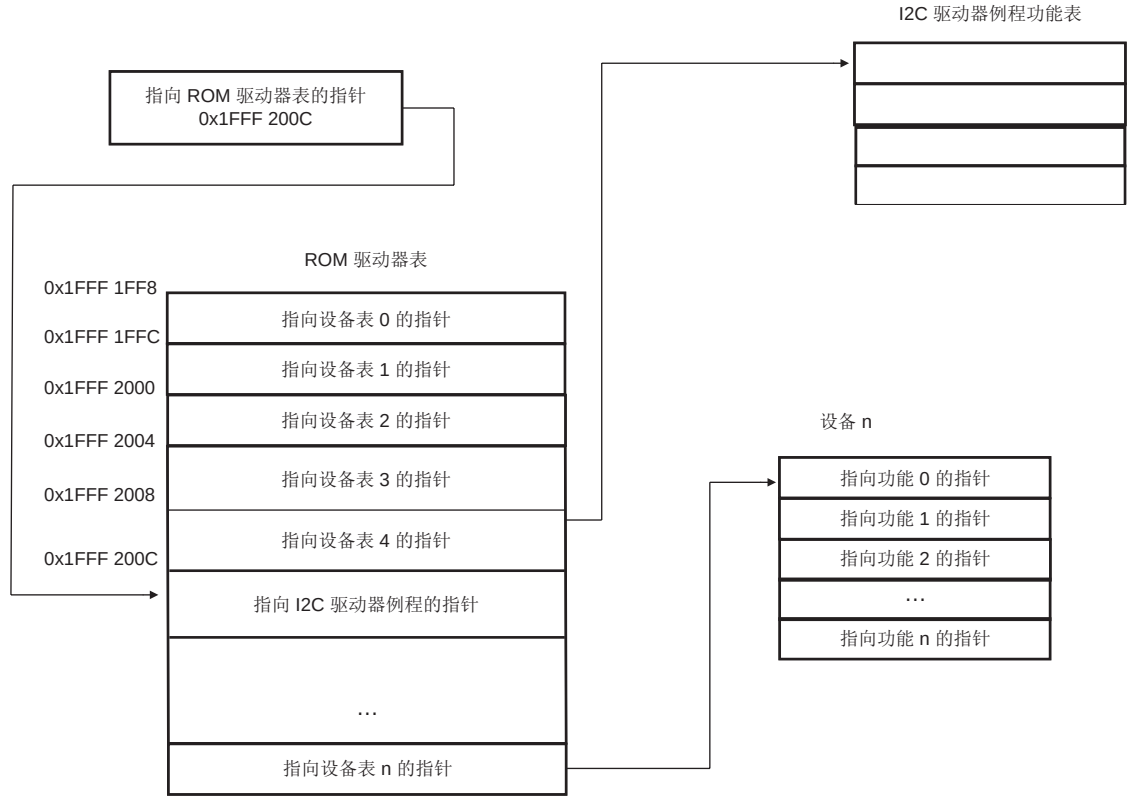


图 60. I2C 总线驱动程序指针结构

22.4 API 描述

表 260. I2C 驱动器函数

函数	偏移	描述
主机模式 - 轮询		
i2c_master_transmit_poll()	< 待定 >	将发送缓冲区中的字节发送到从机。 R/W 位 =0 的从机地址位于发送缓冲区的第 1 个字节中。除非 stop_flag =0，否则 “停止”条件在结束时发送。任务完成后，在调用后的下一行代码返回程序控制。 参数：num_bytes_send、*buffer_ptr_send、stop_flag（以及发送缓冲区第 1 个字节中的从机地址）。
i2c_master_receive_poll()	< 待定 >	接收来自从机的字节，并将其放入缓冲区。 R/W 位 =1 的从机地址位于接收缓冲区的第 1 个字节中。除非 stop_flag =0，否则 “停止”条件在结束时发送。任务完成后，在调用后的下一行代码返回程序控制。 参数：num_bytes_rec、 *buffer_ptr_rec、 stop_flag （以及接收缓冲区第 1 个字节中的从机地址）。

表 260. I2C 驱动器函数 *（续）*

函数	偏移	描述
i2c_master_tx_rx_poll()	< 待定 >	将发送缓冲区中的字节发送到从机（从机地址必须为发送缓冲区中的第 1 个字节）。然后，发出“重复起动”并接收来自从机的字节（从机地址必须为接收缓冲区中的第 1 个字节）。对于发送缓冲区，从机地址字节的最低有效位必须设置为 0；对于接收缓冲区，必须设置为 1。除非 stop_flag =0，否则“停止”条件在结束时发送。任务完成时，在调用后的下一行代码返回程序控制。 参数：num_bytes_send、num_bytes_rec、*buffer_ptr_send、*buffer_ptr_rec、stop_flag（发送缓冲区第 1 个字节中 R/W =0 的从机地址，接收缓冲区第 1 个字节中 R/W =1 的从机地址。）
主机模式 - 中断		
i2c_master_transmit_intr()	< 待定 >	将发送缓冲区中的字节发送到从机。 R/W 位 =0 的从机地址位于发送缓冲区的第 1 个字节中。除非 stop_flag =0，否则“停止”条件在结束时发送。程序控制会被立即返回，并且任务以中断驱动的方式完成。任务完成时会调用回调函数。 参数：num_bytes_send、*buffer_ptr_send、stop_flag、*callback_fp（以及发送缓冲区第 1 个字节中的从机地址）。
i2c_master_receive_intr()	< 待定 >	接收来自从机的字节，并将其放入缓冲区。 R/W 位 =1 的从机地址位于接收缓冲区的第 1 个字节中。除非 stop_flag =0，否则“停止”条件在结束时发送。程序控制会被立即返回，并且任务以中断驱动的方式完成。任务完成时会调用回调函数。 参数：num_bytes_rec、*buffer_ptr_rec、stop_flag、*callback_fp（以及接收缓冲区第 1 个字节中的从机地址）。
i2c_master_tx_rx_intr()	< 待定 >	将发送缓冲区中的字节发送到从机（从机地址必须为发送缓冲区中的第 1 个字节）。然后，发出“重复起动”并接收来自从机的字节（从机地址必须为接收缓冲区中的第 1 个字节）。对于发送缓冲区，从机地址字节的最低有效位必须设置为 0；对于接收缓冲区，必须设置为 1。除非 stop_flag =0，否则“停止”条件在结束时发送。程序控制会被立即返回，并且任务以中断驱动的方式完成。任务完成时会调用回调函数。 参数：num_bytes_send、num_bytes_rec、*buffer_ptr_send、*buffer_ptr_rec、*callback_fp、stop_flag（发送缓冲区第 1 个字节中 R/W =0 的从机地址，接收缓冲区第 1 个字节中 R/W =1 的从机地址。）
从机模式 - 轮询		
i2c_slave_receive_poll()	< 待定 >	首先调用 i2c_set_slave_addr() 函数来设置“我的从机地址”。发生下列某一情况时会成功返回： (1) 接收到“接收我的从机地址”且 R/W 位 =1，表示主机要读取来自从机的数据。 (2) 接收到关于“我的从机地址”的消息，结尾包含“停止”或“重复起动”。 应用程序必须使用“字节数”参数和接收缓冲区中的数据来确定接下来要采取的操作。任务完成时，在调用后的下一行代码返回程序控制。 参数：num_bytes_rec、*buffer_ptr_rec
i2c_slave_transmit_poll()	< 待定 >	发生以下情况后调用该函数：数据从 i2c_slave_receive_poll() 返回，并对数据进行分析以确定哪个“我的从机地址”和数据需要被发送到主机。将数据字节放入发送缓冲区并将参数“字节数”设置为发送。任务完成时，在调用后的下一行代码返回程序控制。 参数：PARMS：num_bytes_send、*buffer_ptr_send

表 260. I2C 驱动器函数 *（续）*

函数	偏移	描述
从机模式 - 中断		
i2c_slave_receive_intr()	< 待定 >	首先调用 i2c_set_slave_addr() 函数来设置 “我的从机地址”。发生下列某一情况时会成功返回： (1) 接收到 “接收我的从机地址” 且 R/W 位 =1，表示主机要读取来自从机的数据。 (2) 接收到关于 “我的从机地址” 的消息，结尾包含 “停止” 或 “重复起动”。 应用程序必须使用 “字节数” 参数和接收缓冲区中的数据来确定接下来要采取的操作。（有关使用中断驱动功能的从机模式的更多信息，另请参见 “在从机模式下操作”。）程序控制会被立即返回，并且任务以中断驱动的方式完成。任务完成时会调用回调函数。 参数：num_bytes_rec、 *buffer_ptr_rec、 *callback_fp
i2c_slave_transmit_intr()	< 待定 >	发生以下情况后调用该函数：数据从 i2c_slave_receive_poll () 返回，并对数据进行分析以确定哪个 “我的从机地址” 和数据需要被发送到主机。将数据字节放入发送缓冲区并将参数 “字节数” 设置为发送。程序控制会被立即返回，并且任务以中断驱动的方式完成。任务完成时会调用回调函数。 参数：num_bytes_send、 *buffer_ptr_send、 *callback_fp

表 261. 错误代码

错误代码	描述
0x0006 0001	ERR_I2C_NAK
0x0006 0002	ERR_I2C_BUFFER_OVERFLOW
0x0006 0003	ERR_I2C_BYTE_COUNT_ERR
0x0006 0004	ERR_I2C_LOSS_OF_ARBRITRATION
0x0006 0005	ERR_I2C_SLAVE_NOT_ADDRESSED
0x0006 0006	ERR_I2C_LOSS_OF_ARBRITRATION_NAK_BIT
0x0006 0007	ERR_I2C_GENERAL_FAILURE
0x0006 0008	ERR_I2C_REGS_SET_TO_DEFAULT

23.1 本章导读

所有 LPC11Axx 器件都提供功率模型。

23.2 特性

- 电源管理服务
- 时钟服务

23.3 简介

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 61 显示用于调用功率模型 API 的指针结构。

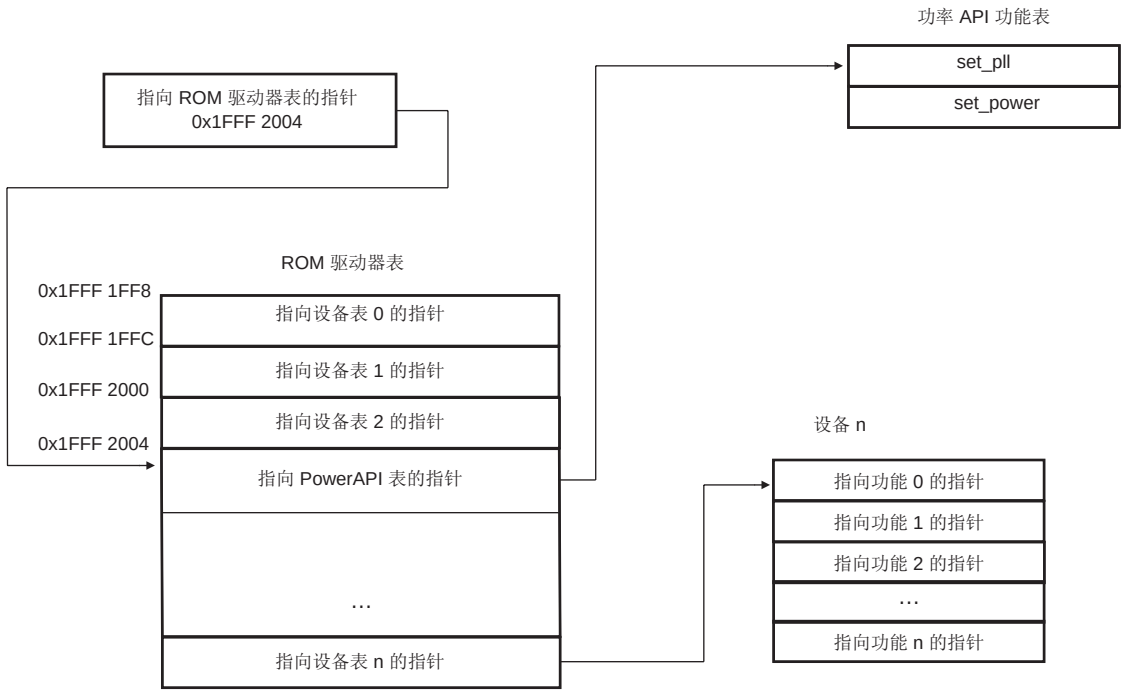


图 61. 功率模型指针结构

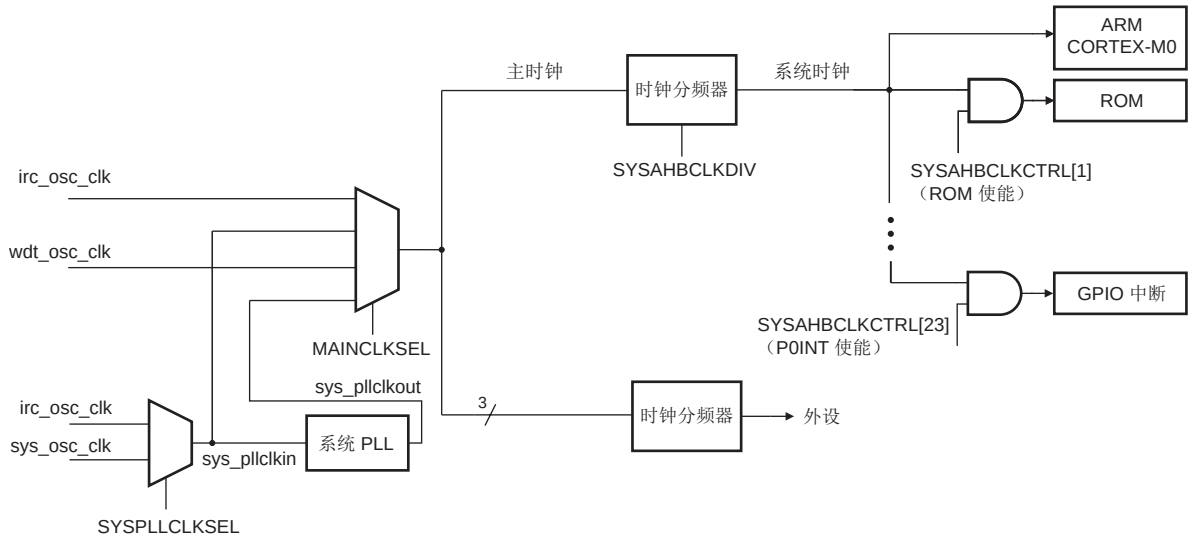


图 62. 用于功率 API 的时钟配置

23.4 定义

必须在使用功率模型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 unsigned p_dev1;
     const unsigned p_dev2;
     const unsigned p_dev3;
     const PWRD *pPWRD;
     const LPC_ROM_DIV_STUCT * pROMDiv;
     const unsigned p_dev4;
     const unsigned p_dev5;
     const unsigned p_dev6;

} ROM;
ROM ** rom = (ROM **) 0x1FFF1FF8;
unsigned int command[4], result[2];
```

23.5 时钟例程

23.5.1 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：调用此例程之前，必须选择 PLL 时钟源 （IRC/ 系统振荡器）（表 11），将主时钟源设为系统 PLL 的输入时钟 （表 13），并且必须将系统 /AHB 时钟分频器设为 1 （表 15）。

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外设时钟。

表 262. 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
```

有关简化的时钟配置方案，请参见图 62。有关更多详情，请参阅图 4。

23.5.1.1 参数 0：系统 PLL 输入频率和参数 1：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。

23.5.1.2 参数 2：mode

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。

23.5.1.3 参数 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。非零值表示返回 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。

注：PLL 锁定时间取决于所选的 PLL 输入时钟源（IRC/ 系统振荡器）及其特性。所选时钟源的抖动程度取决于操作条件，如电源和 / 或环境温度。因此，建议在使用已知良好时钟源并接收到 PLL_NOT_LOCKED 响应时，应先多次调用 set_pll 例程，再声明所选 PLL 时钟源无效。

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。

23.5.1.4 代码示例

以下示例说明了上文讨论的 *set_pll* 的部分功能。

23.5.1.4.1 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。

23.5.1.4.2 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 MHz 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。

23.5.1.4.3 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。

23.5.1.4.4 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。

23.5.1.4.5 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。

23.5.1.4.6 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。

23.6 功率例程

23.6.1 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。

注：set_power 例程专为采用 SYSAHBCLKDIV = 1 这一配置的系统设计（系统时钟分频器寄存器，参见表 15 和图 62）。将此例程用于系统时钟分频器不等于 1 的应用时，微控制器的性能改善程度可能不及主时钟和系统时钟以相同频率运行的设置。

set_power 返回一个结果代码，报告功率设置是否成功改变。

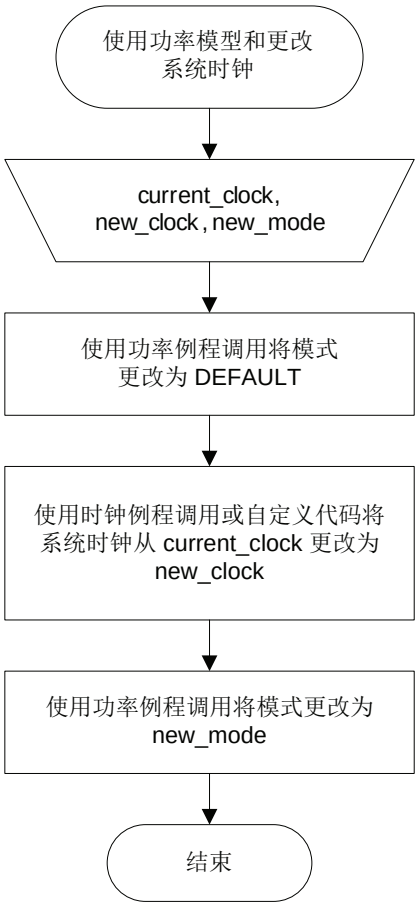


图 63. 功率模型的使用

表 263. set_power 例程

例程	**set_power**
输入	**参数 0**：主时钟 （单位：MHz） **参数 1**：模式（PWR_DEFAULT、PWR_CPU_PERFORMANCE、PWR_ EFFICIENCY、PWR_LOW_CURRENT） **参数 2**：系统时钟 （单位：MHz）
结果	**结果 0**：PWR_CMD_SUCCESS | PWR_INVALID_FREQ | PWR_INVALID_MODE

调用 set_power 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
```

有关简化的时钟配置方案，请参见图 62。有关更多详情，请参阅图 4。

23.6.1.1 参数 0：主时钟

主时钟是微控制器用来寻找系统和外设时钟源的时钟频率。成功执行时钟例程调用或用户提供的相似代码均可配置主时钟。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。

23.6.1.2 参数 1：mode

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。

23.6.1.3 参数 2：系统时钟

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

23.6.1.4 代码示例

以下示例说明了上文讨论的部分 *set_power* 功能。

23.6.1.4.1 无效频率 （超过设备的最大时钟频率）

```
command[0] = 60;
command[1] = PWR_CPU_PERFORMANCE;
command[2] = 60;
(*rom)->pWRD->set_power(command, result);
```

以上设置将用于以 60 MHz 的主时钟和系统时钟运行的系统，需要最大 CPU 处理能力。由于指定的60 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。

23.6.1.4.2 适用的功率设置

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

以上代码指定了某个应用以 24 MHz 主时钟和系统时钟运行，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。

24.1 本章导读

所有 LPC11Axx 器件的调试功能都相同。主 SWD 引脚的位置在 WLCSP 封装和 HVQFN、LQFP 封装中不同。

表 264. SWD 主引脚位置

封装	SWD 主引脚	
	SWCLK	SWDIO
WLCSP	PIO0_2	PIO0_3
LQFP	PIO0_5	PIO0_10
HVQFN	PIO0_5	PIO0_10

24.2 特性

- 支持 ARM 串行调试接口模式。
- 可直接对所有存储器、寄存器和外设进行调试。
- 调试阶段不需要目标资源。
- 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。

24.3 简介

ARM Cortex-M0 集成了调试功能。支持串行调试接口功能。ARM Cortex-M0 的配置可支持最多四个断点和两个观察点。

24.4 描述

LPC11Axx 使用串行线调试模式进行调试。

24.5 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。要最大限度提高 I/O 的灵活性，这些信号中的每一个均可在两个引脚中的任意一个之上。有关引脚配置的信息，请参见表 82 和表 84。要进行调试，则必须准确选择 1 个引脚作为 SWCLK，并选择 1 个引脚作为 SWDIO。

表 265. 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调试逻辑的时钟 (SWDCLK)。
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 LPC11Axx 进行通信并控制它。

24.6 调试注意事项

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。

25.1 本章导读

下面的参考材料以 ARM *《Cortex-M0 用户指南》*为蓝本。只针对 LPC11Axx Cortex-M0 的具体实现做了以下细微的改动：

- 未实现深度睡眠模式

25.2 关于 Cortex-M0 处理器和内核外设

Cortex-M0 处理器是入门级 32 位 ARM Cortex 处理器，适用于多种嵌入式应用。该处理器包含以下特性，给开发者提供了极大的便利：

- 结构简单，容易学习和编程
- 功耗极低，运算效率高
- 出色的代码密度
- 确定、高性能的中断处理
- 向上兼容 Cortex-M 处理器系列。

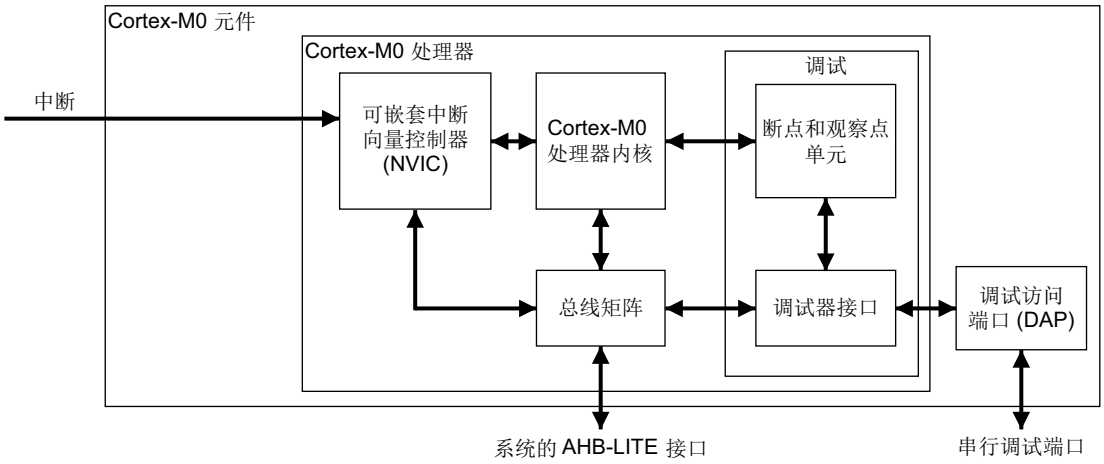


图 64. Cortex-M0 的具体实现

Cortex-M0 处理器基于一个高集成度、低功耗的 32 位处理器内核，采用了 3 级流水线冯·诺伊曼结构 (Von Neumann architecture)。通过简单、功能强大的指令集以及全面优化的设计 （提供包括一个单周期乘法器在内的高端处理硬件）， Cortex-M0 处理器可实现极高的能效。

Cortex-M0 处理器采用 ARMv6-M 架构，基于 16 位的 Thumb 指令集，并包含 Thumb-2 技术。提供了一个现代 32 位结构所希望的出色性能，代码密度比其他 8 位和 16 位微控制器都要高。

Cortex-M0 处理器紧密集成了一个可配置的**可嵌套中断向量控制器** (NVIC)，提供业界领先的中断性能。 NVIC 具有以下功能：

- 提供零抖动中断选项
- 提供四个中断优先级。

处理器内核与 NVIC 的紧密集成使得**中断服务程序** (ISR) 可以快速执行，极大地缩短了中断延迟。这是通过寄存器的硬件协议栈以及加载 - 乘和存储 - 乘操作的停止和重启来获得的。中断处理程序不需要任何汇编程序封装代码，不用消耗任何 ISR 代码。末尾连锁优化还极大地降低了一个 ISR 切换到另一个 ISR 时的开销。

为了优化低功耗设计，NVIC 还与睡眠模式相结合，提供深度睡眠功能，可使整个器件迅速掉电。

25.2.1 系统级接口

Cortex-M0 处理器提供一个简单的系统级接口，使用 AMBA 技术来提供高速、低延迟的存储器访问。

25.2.2 集成的可配置调试

Cortex-M0 处理器执行一个完整的硬件调试解决方案，带有大量硬件断点和观察点选项。通过一个非常适合微控制器和其他小型封装器件的 2 针**串行调试接口** (SWD)，提供了高系统透明度的处理器、存储器和外设执行。

25.2.3 Cortex-M0 处理器的特性小结

- 高代码密度，具有 32 位性能
- 工具和二进制代码与 Cortex-M 处理器系列向上兼容
- 集成了极低功耗的睡眠模式
- 高效的代码执行允许处理器时钟更低，或者延长睡眠模式的时间
- 单周期的 32 位硬件乘法器
- 零抖动中断处理
- 丰富的调试功能。

25.2.4 Cortex-M0 内核外设

Cortex-M0 内核外设有：

NVIC — NVIC 是一个嵌入式中断控制器，支持低延迟的中断处理。

系统控制块 — 系统控制块 (SCB) 是到处理器的编程模型接口。它提供系统实现信息及系统控制，包括配置、控制以及系统异常的报告。

系统定时器 — 系统定时器 SysTick 是一个 24 位的递减定时器。可以将其用作一个实时操作系统 (RTOS) 的节拍定时器，或者用作一个简单的计数器。

25.3 处理器

25.3.1 编程模型

本节描述了 Cortex-M0 的编程模型。除了个别内核寄存器的描述之外，本节还包含处理器模式和协议栈的相关信息。

25.3.1.1 处理器模式

处理器**模式**有：

线程模式 — 用于执行应用软件。处理器在退出复位后进入线程模式。

处理程序模式 — 用于处理异常。处理器在完成所有异常处理后即返回到线程模式。

25.3.1.2 协议栈

处理器使用一个满递减协议栈。这就意味着协议栈指针指向协议栈存储器中的最后一个协议栈项。当处理器将一个新的项压入协议栈时，协议栈指针递减，然后将该项写入新的存储器单元。处理器执行两个协议栈，主协议栈和进程协议栈，两个协议栈有自己独立的协议栈指针副本，见第 25.3.1.3.2 节。

在线程模式下， CONTROL 寄存器控制处理器使用主协议栈还是进程协议栈，见第 25 – 25.3.1.3.7 节。在处理程序模式下，处理器始终使用主协议栈。处理器操作的选择如下：

表 266. 处理器模式和协议栈使用选项小结

处理器模式	用于执行	使用的协议栈
线程模式	应用	主协议栈或进程协议栈 见第 25 – 25.3.1.3.7 节
处理程序	异常处理程序	主协议栈

25.3.1.3 内核寄存器

处理器内核寄存器有：

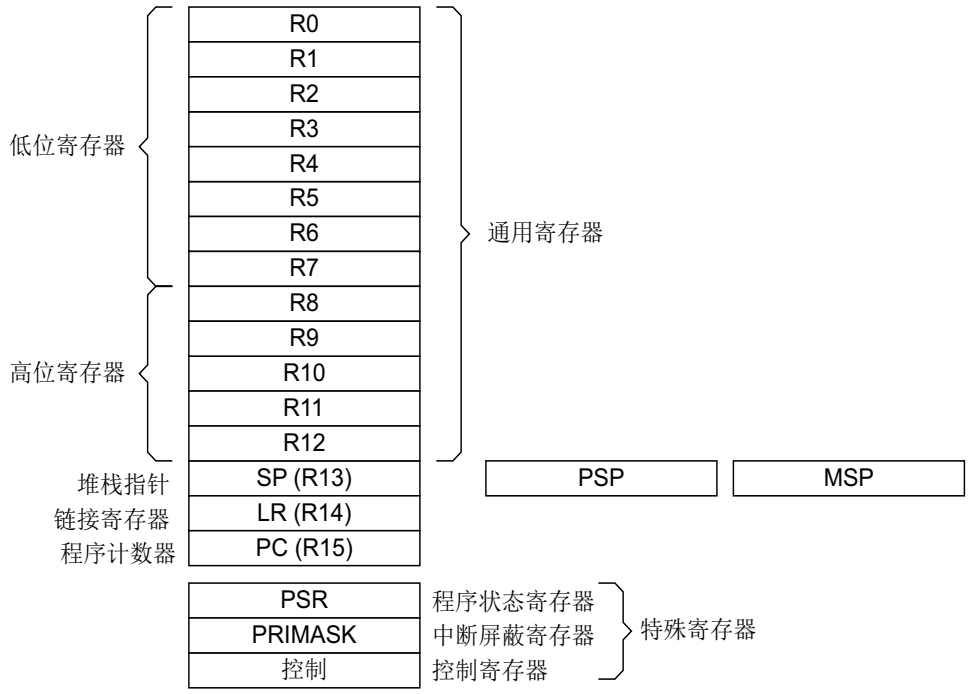


图 65. 处理器内核寄存器组

表 267. 内核寄存器组小结

名称	类型[1]	复位值	描述
R0-R12	RW	不可知	第 25 – 25.3.1.3.1 节
MSP	RW	见文中描述	第 25 – 25.3.1.3.2 节
PSP	RW	不可知	第 25 – 25.3.1.3.2 节
LR	RW	不可知	第 25 – 25.3.1.3.3 节
PC	RW	见文中描述	第 25 – 25.3.1.3.4 节
PSR	RW	不可知 [2]	表 25 – 268
APSR	RW	不可知	表 25 – 269
IPSR	RO	0x00000000	表 270
EPSR	RO	不可知[2]	表 25 – 271
PRIMASK	RW	0x00000000	表 25 – 272
控制	RW	0x00000000	表 25 – 273

[1] 描述在线程模式和处理程序模式下程序执行期间的访问类型。调试访问可以不同。

[2] 位 [24] 为 T 位，从复位向量的位 [0] 加载。

25.3.1.3.1 通用寄存器

R0-R12 是供数据操作使用的 32 位通用寄存器。

25.3.1.3.2 协议栈指针

协议栈指针 (SP) 是寄存器 R13。在线程模式中， CONTROL 寄存器的位 [1] 指示了要使用的协议栈指针：

- 0 = **主协议栈指针** (MSP)。这是复位值。
- 1 = **进程协议栈指针** (PSP)。

复位时，处理器将地址 0x00000000 的值加载到 MSP 中。

25.3.1.3.3 链接寄存器

链接寄存器 (LR) 是寄存器 R14。它存储子程序、函数调用以及异常的返回信息。复位时，LR 值不可知。

25.3.1.3.4 程序计数器

程序计数器 (PC) 是寄存器 R15。它包含当前程序地址。复位时，处理器将复位向量（地址：0x00000004）的值加载到 PC。值的位 [0] 复位时被加载到 EPSR 的 T 位，必须为 1。

25.3.1.3.5 程序状态寄存器

程序状态寄存器 (PSR) 由下列 3 种寄存器组合而成：

- **应用程序状态寄存器** (APSR)
- **中断程序状态寄存器** (IPSR)
- **执行程序状态寄存器** (EPSR)。

在 32 位的 PSR 中，这 3 个寄存器的位域分配互斥。 PSR 的位分配如下所示：

图 66. APSR、 IPSR、 EPSR 寄存器的位分配

这 3 个寄存器可以单独访问，也可以 2 个一组或 3 个一组进行访问，访问时，将寄存器名称作为 MSR 或 MRS 指令的一个自变量。例如：

- 使用寄存器名称 PSR，用 MRS 指令来读所有寄存器
- 使用寄存器名称 APSR，用 MSR 指令来写 APSR。

PSR 的组合和属性如下所示：

表 268. PSR 寄存器组合

寄存器	类型	组合
PSR	RW[1][2]	APSR、 EPSR 和 IPSR
IEPSR	RO	EPSR 和 IPSR
IAPSR	RW[1]	APSR 和 IPSR
EAPSR	RW[2]	APSR 和 EPSR

[1] 处理器忽略对 IPSR 位的写操作。

[2] 读取 EPSR 位将返回零，处理器忽略对这些位的写操作。

有关如何访问程序状态寄存器的更多信息，请参考指令描述第 25 – 25.4.7.6 节和第 25 – 25.4.7.7 节。

应用程序状态寄存器：APSR 包含执行完前面的指令后条件标志的当前状态。有关其属性，请见表 25 – 267 中的寄存器汇总。寄存器的位分配如下所示：

表 269. APSR 位分配

位	名称	函数
[31]	N	负值标志
[30]	Z	零值标志
[29]	C	进位或借位标志
[28]	V	溢出标志
[27:0]	-	保留

有关 APSR 的负值、零值、进位或借位以及溢出标志的更多信息，请参考第 25.4.4.1.4 节。

中断程序状态寄存器：IPSR 包含当前**中断服务程序** (ISR) 的异常编号。有关其属性，请见表 25 – 267 中的寄存器汇总。寄存器的位分配如下所示：

表 270. IPSR 的位分配

位	名称	函数
[31:6]	-	保留
[5:0]	异常编号	这是当前异常的编号： 0 = 线程模式 1 = 保留 2 = NMI 3 = HardFault 4-10 = 保留 11 = SVCall 12、 13 = 保留 14 = PendSV 15 = SysTick 16 = IRQ0 . . . 47 = IRQ31 48-63 = 保留。 更多信息请见第 25 – 25.3.3.2 节。

异常程序状态寄存器：EPSR 包含 Thumb 状态位。

有关 EPSR 属性，请见表 25 – 267 中的寄存器汇总。寄存器的位分配如下所示：

表 271. EPSR 位分配

位	名称	函数
[31:25]	-	保留
[24]	T	Thumb 状态位
[23:0]	-	保留

应用软件尝试使用 MRS 指令直接读取 EPSR 时始终返回零。使用 MSR 指令尝试写入 EPSR 时将被忽略。故障处理程序可检查协议栈 PSR 中的 EPSR 值以确定故障原因。参见第 25 – 25.3.3.6 节。下列情况可将 T 位清零：

- BLX、BX 和 POP{PC} 指令
- 异常返回时恢复被压入栈中的 xPSR 值
- 进入异常时向量值的位 [0]。

在T位为0时尝试执行指令将导致HardFault异常或锁定。更多信息请见第25 – 25.3.4.1节。

可中断 - 可重启指令：可中断 - 可重启指令有 LDM 和 STM。如果在执行这两条中的其中一条指令的过程中出现中断，处理器就终止指令的执行。

在处理完中断后，处理器再从头开始重新执行指令。

25.3.1.3.6 异常屏蔽寄存器

异常屏蔽寄存器禁止处理器处理异常。当异常可能影响到实时任务或要求连续执行的代码序列时，异常就会被禁用。

可以使用 MSR 和 MRS 指令、或 CPS 指令改变 PRIMASK 的值来禁用或重新使能异常。更多信息请见第 25 – 25.4.7.6 节、第 25 – 25.4.7.7 节和第 25 – 25.4.7.2 节。

优先级屏蔽寄存器：PRIMASK 寄存器阻止优先级可配置的所有异常被激活。有关其属性，请见表 25 – 267 中的寄存器汇总。寄存器的位分配如下所示：

表 272. PRIMASK 寄存器的位分配

位	名称	函数
[31:1]	-	保留
[0]	PRIMASK	0 = 无影响 1 = 阻止优先级可配置的所有异常被激活。

25.3.1.3.7 CONTROL 寄存器

CONTROL 寄存器控制着处理器处于线程模式时所使用的协议栈。有关其属性，请见表 25 – 267 中的寄存器汇总。寄存器的位分配如下所示：

表 273. CONTROL 寄存器的位分配

位	名称	函数
[31:2]	-	保留
[1]	有效协议栈指针	定义当前的协议栈： 0 = MSP 是当前的协议栈指针 1 = PSP 是当前的协议栈指针。 在处理程序模式中，这个位读出为零，写操作被忽略。
[0]	-	保留。

处理程序模式始终使用 MSP，因此，在处理程序模式下，处理器忽略对 CONTROL 寄存器的有效协议栈指针位执行的明确的写操作。异常进入和返回机制会将 CONTROL 寄存器更新。

在一个 OS 环境中，推荐运行在线程模式中的线程使用进程协议栈，内核和异常处理器使用主协议栈。

默认情况下，线程模式使用 MSP。要将线程模式中使用的协议栈指针切换为 PSP，请使用 `MSR` 指令将有效协议栈指针位置为 1，见第 25 – 25.4.7.6 节。

注：当更改协议栈指针时，软件必须在 `MSR` 指令后立即使用一个 `ISB` 指令。这将确保 ISB 之后的指令执行时使用新的协议栈指针，参见第 25 – 25.4.7.5 节。

25.3.1.4 异常和中断

Cortex-M0 处理器支持中断和系统异常。处理器和**可嵌套中断向量控制器** (NVIC) 划分所有异常的优先级，并对所有异常进行处理。一个中断或异常会改变软件控制的正常流程。处理器使用处理程序模式来处理除复位之外的所有异常，更多信息请见第 25 – 25.3.3.6.1 节和第 25 – 25.3.3.6.2 节。

NVIC 寄存器控制中断处理。更多信息请见第 25 – 25.5.2 节。

25.3.1.5 数据类型

处理器：

- 支持下列数据类型：
  - 32 位字
  - 16 位半字
  - 8 位字节
- 管理所有数据存储器访问都采用小端模式。指令存储器和**专用外设总线** (PPB) 访问始终是小端模式。更多信息请见第 25 – 25.3.2.1 节。

25.3.1.6 Cortex 微控制器软件接口标准

ARM 为编程 Cortex-M0 微控制器编程提供了 **Cortex 微控制器软件接口标准** (CMSIS)。CMSIS 是器件驱动程序库的一个组成部分。

CMSIS 为 Cortex-M0 微控制器系统定义了：

- 一种通用方式来：
  - 访问外设寄存器
  - 定义异常向量
- 以下两项的名称：
  - 内核外设寄存器
  - 内核异常向量
- 一个 RTOS 内核的器件独立的接口。

CMSIS 包含 Cortex-M0 处理器中内核外设的地址定义和数据结构。还包含有组成 TCP/IP 协议栈和 Flash 文件系统的中间件元件的可选接口。

通过支持模板代码的重复使用以及将不同中间件厂商提供的符合 CMSIS 的软件组件组合起来，CMSIS 大大简化了整个软件开发过程。软件厂商可以扩展 CMSIS，使其包含各个厂商的外设定义以及这些外设的存取函数。

本文档包含了 CMSIS 定义的寄存器名称，并对处理器内核和内核外设相关的 CMSIS 函数进行了简单描述。

注：本文档使用 CMSIS 定义的寄存器缩略名称。在某些情况下，这些名称与其他文档中可能用到的架构缩略名称不同。

下面各节给出了有关 CMSIS 的更多信息：

- 第 25.3.5.3 节 “电源管理编程提示”
- 第 25.4.2 节 “内部函数”
- 第 25.5.2.1 节 “使用 CMSIS 访问 Cortex-M0 NVIC 寄存器”
- 第 25.5.2.8.1 节 “NVIC 编程提示”。

25.3.2 存储器模型

本节描述处理器存储器映射以及存储器访问的行为。处理器有一个固定的存储器映射，提供有高达 4GB 的可寻址存储空间。存储器映射如 HIDDEN 页 图 1 8 中所示。

25.3.2.1 存储区、类型和属性

存储器映射分成多个区域。每个区域有一个定义好的存储器类型，某些区域还有附加的存储器属性。存储器类型和属性决定了各个区域的访问行为。

存储器类型有：

普通存储器 — 处理器为了提高效率，可以重新对传送进行排序，或者刻意地进行读取。

Device 存储器 — 处理器保护与 Device 或强秩序存储器的其他传送相关的传送秩序。

强秩序存储器 — 处理器保护与所有其他传送相关的传送秩序。

Device 存储器和强秩序存储器的不同秩序要求意味着，存储器系统可以缓冲一个对 Device 存储器的写操作，但不得缓冲对强秩序存储器的写操作。

附加的存储器属性包括：

永不执行 (XN) — 表示处理器阻止指令访问。当执行从存储器的 XN 区提取出来的指令时，产生一个 HardFault 异常。

25.3.2.2 系统的存储器访问秩序

对于大多数由明确的存储器访问指令引发的存储器访问，存储器系统都不保证访问秩序与指令的编写顺序完全一致，只要所有访问秩序的重新安排不影响指令序列的操作就行。一般情况下，如果两个存储器访问的顺序必须与两条存储器访问指令编写的顺序完全一致程序才能正确执行，软件就必须在两条存储器访问指令之间插入一条内存屏障指令，请见第 25 – 25.3.2.4 节。

但是，存储器系统保证 Device 存储器和强秩序存储器的一些访问秩序。对于两条存储器访问指令 A1 和 A2，如果 A1 的编写顺序在前，两条指令所引发的存储器访问顺序为：

A1 \ A2	正常访问	器件访问		强秩序访问
		非共享	可共享	
正常访问	-	-	-	-
器件访问，非共享	-	<	-	<
器件访问，可共享	-	-	<	<
强秩序访问	-	<	<	<

图 67. 存储器秩序限制

其中：

- — 表示存储器系统不保证访问秩序。

< — 表示观察到访问顺序与指令编写顺序一致，即， A1 总是在 A2 之前。

25.3.2.3 存储器访问行为

存储器映射中每个区域的访问行为如下：

表 274. 存储器访问行为

地址范围	存储器区域	存储器类型[1]	XN[1]	描述
0x00000000-0x0FFFFFFF	代码	普通	-	程序代码的可执行区域。也可以把数据保存到这里。
0x10000000-0x3FFFFFFF	SRAM	普通	-	数据的可执行区域。也可以把代码保存到这里。
0x40000000-0x5FFFFFFF	外设	设备	XN	外部设备存储器。
0x60000000-0x9FFFFFFF	外部 RAM	普通	-	数据的可执行区域。
0xA0000000-0xDFFFFFFF	外部设备	设备	XN	外部设备存储器。
0xE0000000-0xE00FFFFF	专用外设总线	强秩序存储器	XN	这个区域包括 NVIC、系统定时器和系统控制块。这个区域只能使用字访问。
0xE0100000-0xFFFFFFFF	设备	设备	XN	厂商提供的特定存储器。

[1] 更多信息请见第 25 – 25.3.2.1 节。

Code、 SRAM 和外部 RAM 区域可以保存程序。

25.3.2.4 软件的存储器访问秩序

程序流程的指令顺序不能担保相应的存储器传送顺序。这是因为：

- 为了提高效率，处理器可以将一些存储器访问的顺序重新安排，只要不影响指令序列的操作就行
- 存储器映射中的存储器或设备可能有不同的等待状态
- 某些存储器访问被缓冲，或者是刻意为之的。

第 25 – 25.3.2.2 节节描述了存储器系统在哪些情况下能保证存储器访问的秩序。否则，如果存储器访问的秩序十分重要，软件就必须插入一些内存屏障指令来强制保持存储器访问的秩序。处理器提供了以下内存屏障指令：

DMB — 数据存储器屏障 (DMB) 指令保证先完成未处理的存储器传送，再执行后面的存储器传送，参见第 25 – 25.4.7.3 节。

DSB — 数据同步屏障 (DSB) 指令保证先完成未处理的存储器传送，再执行后面的指令，参见第 25 – 25.4.7.4 节。

ISB — 指令同步屏障 (ISB) 保证所有已完成的存储器传送的结果都能被后面的指令辨认出来。参见第 25 – 25.4.7.5 节。

下面是内存屏障指令使用的一些例子：

向量表 — 如果程序改变了向量表中的一项，然后又使能了相应的异常，那么就在两个操作之间插入一条 DMB 指令。这就确保了，如果异常在被使能后立刻被采纳，处理器能使用新的异常向量。

自修改代码 — 如果一个程序包含自修改代码，代码修改之后在程序中立刻使用一条 ISB 指令。这就确保了后面的指令执行使用的是更新后的程序。

存储器映射切换 — 如果系统包含一个存储器映射切换机制，在切换存储器映射之后使用一条 DSB 指令。这就确保了后面的指令执行使用的是更新后的存储器映射。

对强秩序存储器 （例如，系统控制块）执行的存储器访问不需要使用 DMB 指令。

处理器保护与所有其他传送相关的传送秩序。

25.3.2.5 存储器的字节存储顺序

处理器看到的存储器是一个从零开始、编号逐次递增的字节集合。例如，字节 0-3 存放第一个保存的字，字节 4-7 存放第二个保存的字。第 25 – 25.3.2.5.1 节描述了数据的字在存储器中是如何存放的。

25.3.2.5.1 小端格式

在小端格式中，处理器将字的**最低有效字节** (lsbyte) 保存在编号最小的字节中，**最高有效字节** (msbyte) 保存在编号最大的字节中。例如：

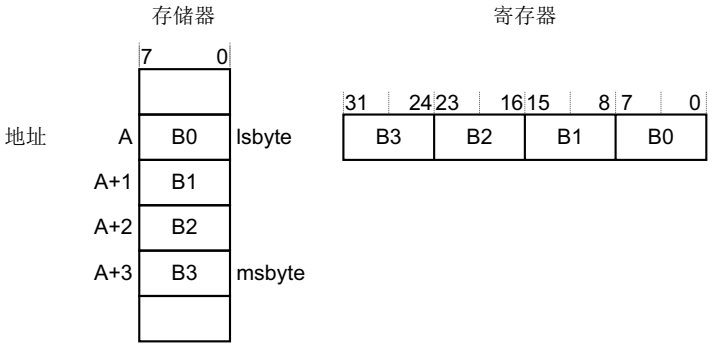


图 68. 小端格式

25.3.3 异常模型

本节描述异常模型。

25.3.3.1 异常状态

每个异常都处于下面其中一种状态：

无效 — 异常无效，也未挂起。

挂起 — 异常正在等待处理器处理。

一个外设或软件的中断请求可以改变相应的挂起中断的状态。

活动 — 一个异常正在被处理器处理，但处理尚未结束。

一个异常处理程序可以中断另一个异常处理程序的执行。在这种情况下，两个异常都处于有效状态。

有效和挂起 — 异常正在被处理器处理，而且有一个来自同一个异常源的异常正在等待处理。

25.3.3.2 异常类型

异常类型有：

复位 — 复位在上电或热复位时启动。异常模型将复位当做一种特殊形式的异常来对待。当复位产生时，处理器的操作停止（可能停止在一条指令的任何一点上）。当复位撤销时，从向量表中复位项提供的地址处重新起动执行。执行在线程模式下重新起动。

NMI — 不可屏蔽中断 (NMI) 可由软件触发。这是除复位之外优先级最高的异常。NMI 永远使能，优先级固定为 –2。 NMI 不能：

- 被屏蔽，它的执行也不能被其他任何异常中止；
- 被除复位之外的任何异常抢占。

HardFault — HardFault 是由于在正常操作过程中或在异常处理过程中出错而出现的一个异常。 HardFault 的优先级固定为 -1，表明它的优先级要高于任何优先级可配置的异常。

SVCall — 超级用户调用 (SVC) 异常是一个由 SVC 指令触发的异常。在 OS 环境下，应用程序可以使用 SVC 指令来访问 OS 内核函数和器件驱动程序。

PendSV — PendSV 是一个中断驱动的系统级服务请求。在 OS 环境下，当没有其他异常有效时，使用 PendSV 来进行任务切换。

SysTick — SysTick 是一个系统定时器到达零时产生的异常。软件也可以产生一个 SysTick 异常。在 OS 环境下，处理器可以将这个异常用作系统节拍。

中断 (IRQ) — 中断（或 IRQ）是外设发出的一个异常，或者是由软件请求产生的一个异常。所有中断都与指令执行不同步。在系统中，外设使用中断来与处理器通信。

表 275. 各种异常类型的特性

异常编号[1]	IRQ 编号[1]	异常类型	优先级	向量地址[2]
1	-	复位	-3，优先级最高	0x00000004
2	-14	NMI	-2	0x00000008
3	-13	HardFault	-1	0x0000000C
4-10	-	保留	-	-
11	-5	SVCall	可配置[3]	0x0000002C
12-13	-	保留	-	-
14	-2	PendSV	可配置[3]	0x00000038
15	-1	SysTick	可配置[3]	0x0000003C
16 和大于 16 的值	0 和大于 0 的值	中断 (IRQ)	可配置[3]	0x00000040 以及更高的地址[4]

[1] 为了简化软件层， CMSIS 只使用 IRQ 编号，因此，对除中断外的其他异常都使用负值。 IPSR 返回异常编号，见表 25 – 270。

[2] 更多信息请见第 25.3.3.4 节。

[3] 参见第 25 – 25.5.2.6 节。

[4] 地址值以 4 为步长，逐次递增。

对于除复位之外的异步异常，在异常被触发和处理器进入异常处理程序之间，处理器可以执行条件指令。

被特许的软件可以将表25 – 275中列出的优先级可配置的异常禁用，见第25 – 25.5.2.3节。

有关 HardFault 的更多信息，请见第 25 – 25.3.4 节。

25.3.3.3 异常处理程序

处理器使用以下处理程序来处理异常：

中断服务程序 (ISR) — 中断 IRQ0-IRQ31 是由 ISR 来处理的异常。

故障处理程序 — HardFault 是唯一一个由故障处理程序来处理的异常。

系统处理程序 — NMI、PendSV、SVCall、SysTick 和 HardFault 是由系统处理程序来处理的全部系统异常。

25.3.3.4 向量表

向量表包含协议栈指针的复位值以及所有异常处理程序的起始地址（也称为异常向量）。图 25 – 69 显示了异常向量在向量表中的放置顺序。每个向量的最低有效位必须为 1，表明异常处理程序都是用 Thumb 代码编写的。

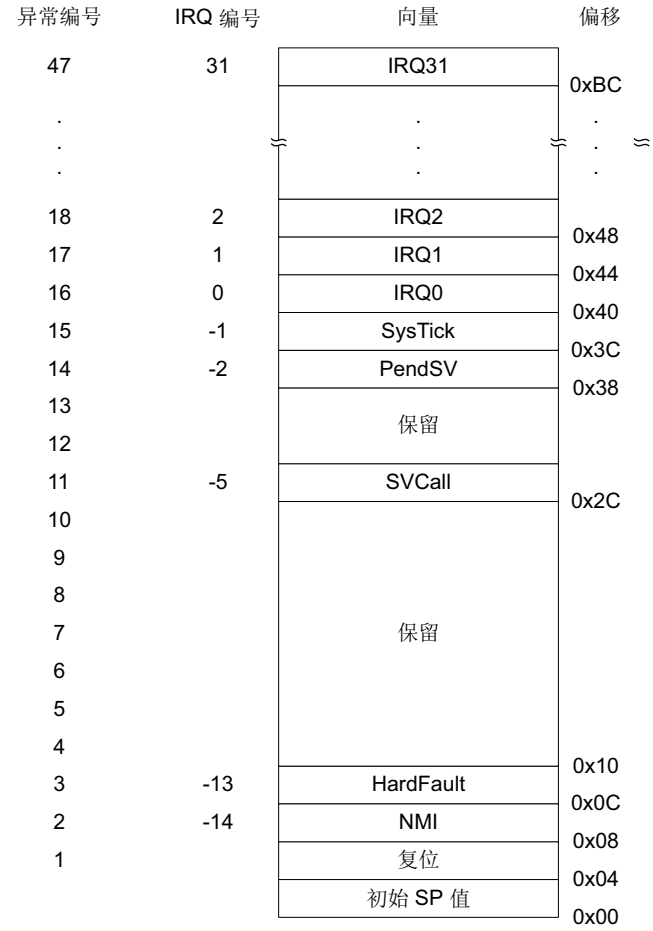


图 69. 向量表

向量表的地址固定为 0x00000000。

25.3.3.5 异常优先级

如表 25 – 275 所示，每个异常都有对应的优先级：

- 优先级值越小表示优先级越高
- 除复位、 HardFault 和 NMI 之外，所有异常的优先级都是可配置的。

如果软件不配置任何优先级，那么，所有优先级可配置的异常的优先级就都为 0。有关配置异常优先级的信息，请见

- 第 25 – 25.5.3.7 节
- 第 25 – 25.5.2.6 节。

注：优先级值的配置范围为 0-3。复位、 HardFault 和 NMI 这些有固定负优先级值的异常，其优先级高于任何其他异常。

给 IRQ[0] 分配一个高优先级值、给 IRQ[1] 分配一个低优先级值就意味着 IRQ[1] 的优先级高于 IRQ[0]。如果 IRQ[1] 和 IRQ[0] 都有效，先处理 IRQ[1]。

如果多个挂起的异常具有相同的优先级，异常编号越小的挂起异常优先处理。例如，如果 IRQ[0] 和 IRQ[1] 均挂起，并且两者的优先级相同，那么先处理 IRQ[0]。

当处理器正在执行一个异常处理程序时，如果出现一个更高优先级的异常，那么这个异常就被抢占。如果出现的异常的优先级和正在处理的异常的优先级相同，这个异常就不会被抢占，与异常的编号大小无关。但是，新中断的状态就变为挂起。

25.3.3.6 异常的进入和返回

描述异常处理时使用了下列术语：

抢占 — 当处理器正在执行一个异常处理程序时，如果一个异常的优先级比正在处理的异常的优先级更高，那么低优先级的异常就被抢占。

当一个异常抢占另一个异常时，这些异常就被称为嵌套异常。更多信息请见第 25 – 25.3.3.6.1 节。

返回 — 当异常处理程序结束，并且满足以下条件时，异常就返回：

- 没有优先级足够高的挂起异常要处理
- 已结束的异常处理程序没有在处理一个迟来的异常。

处理器弹出协议栈，处理器状态恢复到中断出现之前的状态，更多信息请见第 25 – 25.3.3.6.2 节。

末尾连锁 — 这个机制加速了异常的处理。当一个异常处理程序结束时，如果一个挂起的异常满足异常进入的要求，就跳过协议栈弹出，控制权移交给新的异常处理程序。

迟来 — 这个机制加速了抢占的处理。如果一个高优先级的异常在前一个异常正在保存状态的过程中出现，处理器就转去处理更高优先级的异常，开始提取这个异常的向量。状态保存不受迟来异常的影响，因为两个异常保存的状态相同。从迟来异常的异常处理程序返回时，要遵守正常的末尾连锁规则。

25.3.3.6.1 异常的进入

当有一个优先级足够高的挂起异常存在，并且满足下面的任何一个条件，就进入异常处理：

- 处理器处于线程模式
- 新异常的优先级高于正在处理的异常，这时，新异常就抢占了正在处理的异常。

当一个异常抢占另一个异常时，异常就被嵌套。

优先级足够高的意思是该异常的优先级比屏蔽寄存器中所限制的任何一个异常组的优先级都要高，见第 25 – 25.3.1.3.6 节。优先级比这个异常要低的异常被挂起，但不被处理器处理。

当处理器处理异常时，除非异常是一个末尾连锁异常或迟来的异常，否则，处理器都把信息压入到当前的协议栈中。这个操作被称为**入栈**，8 个数据字的结构被称为**栈帧**。栈帧包含以下信息：

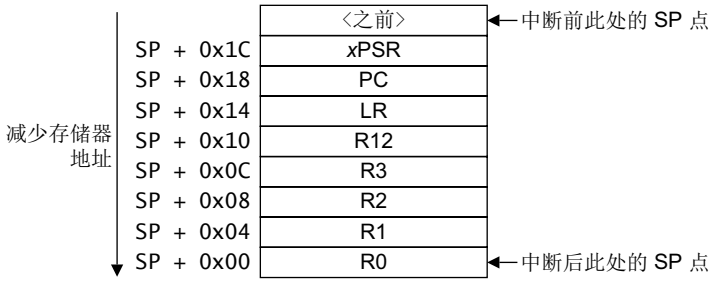


图 70. 异常进入时协议栈的内容

入栈后，协议栈指针立刻指向栈帧的最低地址单元。栈帧按照双字地址对齐。

栈帧包含返回地址。这是被中止的程序中下条指令的地址。这个值在异常返回时返还给 PC，使被中止的程序恢复执行。

处理器执行一次向量提取，从向量表中读出异常处理程序的起始地址。当入栈结束时，处理器开始执行异常处理程序。同时，处理器向 LR 写入一个 EXC_RETURN 值。这个值指示了栈帧对应哪个协议栈指针以及在异常出现之前处理器处于什么工作模式。

如果在异常进入的过程中没有更高优先级的异常出现，处理器就开始执行异常处理程序，并自动将相应的挂起中断的状态变为有效。

如果在异常进入的过程中有另一个优先级更高的异常出现，处理器就开始执行这个高优先级异常的异常处理程序，不改变前一个异常的挂起状态。这是一个迟来异常的情况。

25.3.3.6.2 异常返回

当处理器处于处理程序模式并且执行下面其中一条指令尝试将 PC 设为 EXC_RETURN 值时，出现异常返回：

- POP 指令，用来加载 PC
- BX 指令，用来使用任意的寄存器。

在异常进入时处理器将一个 EXC_RETURN 值保存到 LR 中。异常机制依靠这个值来检测处理器何时执行完一个异常处理程序。 EXC_RETURN 值的位 [31:4] 为 0xFFFFFFF。当处理器将一个相应的这种形式的值加载到 PC 时，它将检测到这个操作并不是一个正常的分支操作，而是异常已经结束。因此，处理器启动异常返回序列。 EXC_RETURN 值的位 [3:0] 指出了所需的返回协议栈和处理器模式，如表 25 – 276 所示。

表 276. 异常返回的行为

EXC_RETURN	描述
0xFFFFFFF1	返回到处理程序模式。 异常返回获得主协议栈的状态。 返回后执行使用 MSP。
0xFFFFFFF9	返回到线程模式。 异常返回获得 MSP 的状态。 返回后执行使用 MSP。
0xFFFFFFFD	返回到线程模式。 异常返回获得 PSP 的状态。 返回后执行使用 PSP。
所有其他值	保留。

25.3.4 故障处理

故障是异常的一个子集，见第 25 – 25.3.3 节。所有的故障都导致 HardFault 异常被处理，或者，如果故障在 NMI 或 HardFault 处理程序中出现，会导致锁定。发生以下情况会导致出现故障：

- 在一个优先级等于或高于 SVCall 的地方执行 SVC 指令
- 在没有调试器的情况下执行 BKPT 指令
- 在加载或存储时出现一个系统产生的总线错误
- 执行一个 XN 存储器地址中的指令
- 从系统产生了一个总线故障的地址单元中执行指令
- 在提取向量时出现了一个系统产生的总线错误
- 执行一个未定义的指令
- 由于 T 位之前被清零而导致不再处于 Thumb 状态的情况下执行一条指令
- 尝试对一个不对齐的地址执行加载或存储操作。

只有复位和 NMI 可以抢占优先级固定的 HardFault 处理程序。HardFault 可以抢占除复位、NMI 或其他硬故障之外的任何异常。

25.3.4.1 锁定

如果在执行 NMI 或 HardFault 处理程序时出现故障，或者，在一个使用 MSP 的异常返回时出栈的却是 PSR 的时候系统产生一个总线错误，处理器进入锁定状态。当处理器处于锁定状态时，它不执行任何指令。处理器保持处于锁定状态，直到下面任何一种情况出现：

- 出现复位
- 调试器将锁定状态终止
- 出现一个 NMI，以及当前的锁定处于 HardFault 处理程序中。

如果锁定状态出现在 NMI 处理程序中，后面的 NMI 就无法使处理器离开锁定状态。

25.3.5 电源管理

Cortex-M0 处理器的睡眠模式可以降低功耗，睡眠模式包含 2 种：

- 睡眠模式：停止处理器时钟
- 深度睡眠模式。 LPC11Axx 中未实现深度睡眠模式。

SCR 的 SLEEPDEEP 位选择使用哪种睡眠模式，见第 25 – 25.5.3.5 节。

本节描述了进入睡眠模式的机制和将器件从睡眠模式唤醒的条件。

25.3.5.1 进入睡眠模式

本节描述了软件可以用来使处理器进入睡眠模式的一种机制。

系统可以产生伪唤醒事件，例如，一个调试操作唤醒处理器。因此，软件必须能够在这样的事件之后使处理器重新回到睡眠模式。程序中可以有一个空闲锁相环使得处理器回到睡眠模式。

25.3.5.1.1 等待中断

等待中断指令 (WFI) 使器件立刻进入睡眠模式。当执行一个 WFI 指令时，处理器停止执行指令，进入睡眠模式。更多信息请见第 25 – 25.4.7.12 节。

25.3.5.1.2 等待事件

注：WFE 指令不能在 LPC11Axx 上使用。

等待事件指令 (WFE) 根据一个一位的事件寄存器的值来进入睡眠模式。处理器执行一个 WFE 指令时检查事件寄存器的值：

0 — 处理器停止执行指令，进入睡眠模式

1 — 处理器将寄存器的值设为 0，并继续执行指令，不进入睡眠模式。

更多信息请见第 25 – 25.4.7.11 节。

如果事件寄存器为 1，表示处理器在执行 WFE 指令时不得进入睡眠模式。通常的原因是出现了一个外部事件，或者系统中的另一个处理器已经执行了 SEV 指令，见第 25 – 25.4.7.9 节。软件不能直接访问这个寄存器。

25.3.5.1.3 Sleep-on-exit

如果 SCR 的 SLEEPONEXIT 位被设为 1，当处理器完成一个异常处理程序的执行并返回到线程模式时，处理器立刻进入睡眠模式。如果应用只要求处理器在中断出现时运行，就可以使用这种机制。

25.3.5.2 从睡眠模式唤醒

处理器的唤醒条件取决于使处理器进入睡眠模式所采用的机制。

25.3.5.2.1 从 WFI 或退出时进入睡眠唤醒

通常，只有当检测到一个优先级足够高的异常导致进入异常处理时，处理器才唤醒。

某些嵌入式系统在处理器唤醒之后可能必须先执行系统恢复任务，然后再执行中断处理程序。通过将 PRIMASK 位设为 1 来实现这个操作。如果到来的中断被使能，并且优先级高于当前的异常优先级，处理器就唤醒，但不执行中断处理程序，直至处理器将 PRIMASK 设为 0。有关 PRIMASK 的更多信息，请见第 25 – 25.3.1.3.6 节。

25.3.5.2.2 从 WFE 唤醒

注：WFE 指令不能在 LPC11Axx 上使用。

如果出现以下情况，处理器就唤醒：

- 处理器检测到一个优先级足够高的异常导致进入异常处理
- 在一个多处理器的系统中，系统中的另一个处理器执行了 SEV 指令。

另外，如果 SCR 的 SEVONPEND 位被设为 1，那么任何新的挂起中断都能触发一个事件和唤醒处理器，即使这个中断被禁用，或者这个中断的优先级不够高而导致无法进入异常处理。有关 SCR 的更多信息，请见第 25 – 25.5.3.5 节。

25.3.5.3 电源管理编程提示

ISO/IEC C 不能直接产生 WFI、WFE 和 SEV 指令。 CMSIS 为这些指令提供了以下内在函数：

```
void __WFE(void) // 等待事件

void __WFI(void) // 等待中断

void __SEV(void) // 发送事件
```

25.4 指令集

25.4.1 指令集汇总

处理器执行 Thumb 版本的指令集。表 278 列出了所支持的指令。

注：输入表 278

- 尖括号 <> 括着操作数的备用格式
- 大括号 {} 括着可选的操作数和助记符部分
- 操作数列所列出的操作数不完全。

有关指令和操作数的信息，详见指令描述。

表 277. Cortex-M0 指令

助记符	操作数	简述	标志	参考
ADCS	*{Rd,} Rn, Rm*	进位加法	N,Z,C,V	第 25 – 25.4.5.1 节
ADD{S}	*{Rd,} Rn, <Rm|#imm>*	加法	N,Z,C,V	第 25 – 25.4.5.1 节
ADR	*Rd, label*	将基于 PC 相对偏移的地址读到寄存器	-	第 25 – 25.4.4.1 节
ANDS	*{Rd,} Rn, Rm*	位与操作	N,Z	第 25 – 25.4.5.1 节
ASRS	*{Rd,} Rm, <Rs|#imm>*	算术右移	N,Z,C	第 25 – 25.4.5.3 节
B{cc}	*label*	跳转 { 有条件 }	-	第 25 – 25.4.6.1 节
BICS	*{Rd,} Rn, Rm*	位清除	N,Z	第 25 – 25.4.5.2 节
BKPT	*#imm*	断点	-	第 25 – 25.4.7.1 节
BL	*label*	带链接的跳转	-	第 25 – 25.4.6.1 节
BLX	*Rm*	带链接的间接跳转	-	第 25 – 25.4.6.1 节
BX	*Rm*	间接跳转	-	第 25 – 25.4.6.1 节
CMN	*Rn, Rm*	比较负值	N,Z,C,V	第 25 – 25.4.5.4 节
CMP	*Rn, <Rm|#imm>*	比较	N,Z,C,V	第 25 – 25.4.5.4 节
CPSID	*i*	更改处理器状态，禁用中断	-	第 25 – 25.4.7.2 节
CPSIE	*i*	更改处理器状态，使能中断	-	第 25 – 25.4.7.2 节
DMB	-	数据内存屏障	-	第 25 – 25.4.7.3 节
DSB	-	数据同步屏障	-	第 25 – 25.4.7.4 节
EORS	*{Rd,} Rn, Rm*	异或	N,Z	第 25 – 25.4.5.2 节
ISB	-	指令同步屏障	-	第 25 – 25.4.7.5 节
LDM	*Rn{!}, reglist*	加载多个寄存器，访问之后会递增地址	-	第 25 – 25.4.4.5 节
LDR	*Rt, label*	从基于 PC 相对偏移的地址加载寄存器	-	第 25 – 25.4.4 节
LDR	*Rt, [Rn, <Rm|#imm>]*	用字加载寄存器	-	第 25 – 25.4.4 节
LDRB	*Rt, [Rn, <Rm|#imm>]*	用字节加载寄存器	-	第 25 – 25.4.4 节
LDRH	*Rt, [Rn, <Rm|#imm>]*	用半字加载寄存器	-	第 25 – 25.4.4 节
LDRSB	*Rt, [Rn, <Rm|#imm>]*	用有符号的字节加载寄存器	-	第 25 – 25.4.4 节
LDRSH	*Rt, [Rn, <Rm|#imm>]*	用有符号的半字加载寄存器	-	第 25 – 25.4.4 节
LSLS	*{Rd,} Rn, <Rs|#imm>*	逻辑左移	N,Z,C	第 25 – 25.4.5.3 节
U	*{Rd,} Rn, <Rs|#imm>*	逻辑右移	N,Z,C	第 25 – 25.4.5.3 节
MOV{S}	*Rd, Rm*	传输	N,Z	第 25 – 25.4.5.5 节
MRS	*Rd, spec_reg*	从特别寄存器传输到通用寄存器	-	第 25 – 25.4.7.6 节
MSR	*spec_reg, Rm*	从通用寄存器传输到特别寄存器	N,Z,C,V	第 25 – 25.4.7.7 节
MULS	*Rd, Rn, Rm*	乘法， 32 位结果值	N,Z	第 25 – 25.4.5.6 节
MVNS	*Rd, Rm*	位非	N,Z	第 25 – 25.4.5.5 节
NOP	-	无操作	-	第 25 – 25.4.7.8 节
ORRS	*{Rd,} Rn, Rm*	逻辑或	N,Z	第 25 – 25.4.5.2 节
POP	*reglist*	出栈，将协议栈的内容放入寄存器	-	第 25 – 25.4.4.6 节

表 277. Cortex-M0 指令 *（续）*

助记符	操作数	简述	标志	参考
PUSH	*reglist*	压栈，将寄存器的内容压入协议栈	-	第 25 - 25.4.4.6 节
REV	*Rd, Rm*	反转字里面的字节顺序	-	第 25 - 25.4.5.7 节
REV16	*Rd, Rm*	反转打包半字里面的字节顺序	-	第 25 - 25.4.5.7 节
REVSH	*Rd, Rm*	反转有符号半字里面的字节顺序	-	第 25 - 25.4.5.7 节
RORS	*{Rd,} Rn, Rs*	循环右移	N,Z,C	第 25 - 25.4.5.3 节
RSBS	*{Rd,} Rn, #0*	反向减法	N,Z,C,V	第 25 - 25.4.5.1 节
SBCS	*{Rd,} Rn, Rm*	进位减法	N,Z,C,V	第 25 - 25.4.5.1 节
SEV	-	发送事件	-	第 25 - 25.4.7.9 节
STM	*Rn!, reglist*	存储多个寄存器，在访问后地址递增	-	第 25 - 25.4.4.5 节
STR	*Rt, [Rn, <Rm|#imm>]*	将寄存器作为字来存储	-	第 25 - 25.4.4 节
STRB	*Rt, [Rn, <Rm|#imm>]*	将寄存器作为字节来存储	-	第 25 - 25.4.4 节
STRH	*Rt, [Rn, <Rm|#imm>]*	将寄存器作为半字来存储	-	第 25 - 25.4.4 节
SUB{S}	*{Rd,} Rn, <Rm|#imm>*	减法	N,Z,C,V	第 25 - 25.4.5.1 节
SVC	*#imm*	超级用户调用	-	第 25 - 25.4.7.10 节
SXTB	*Rd, Rm*	符号扩展字节	-	第 25 - 25.4.5.8 节
SXTH	*Rd, Rm*	符号扩展半字	-	第 25 - 25.4.5.8 节
TST	*Rn, Rm*	基于逻辑与的测试	N,Z	第 25 - 25.4.5.9 节
UXTB	*Rd, Rm*	0 扩展字节	-	第 25 - 25.4.5.8 节
UXTH	*Rd, Rm*	0 扩展半字	-	第 25 - 25.4.5.8 节
WFE	-	等待事件	-	第 25 - 25.4.7.11 节
WFI	-	等待中断	-	第 25 - 25.4.7.12 节

25.4.2 内部函数

ISO/IEC C 代码不能直接访问某些 Cortex-M0 指令。本章节对可以产生这些指令的内部函数进行了描述，内部函数可由 CMSIS 和有可能由 C 编译器提供。若 C 编译器不支持相关的内部函数，则用户可能需要使用内联汇编程序来访问相关的指令。

CMSIS 提供下列内部函数来产生 ISO/IEC C 代码不能直接访问的指令：

表 278. 产生某些 Cortex-M0 指令的 CMSIS 内部函数

指令	CMSIS 内部函数
CPSIE i	void __enable_irq(void)
CPSID i	void __disable_irq(void)
ISB	void __ISB(void)
DSB	void __DSB(void)
DMB	void __DMB(void)
NOP	void __NOP(void)
REV	uint32_t __REV(uint32_t int value)

表 278. 产生某些 Cortex-M0 指令的 CMSIS 内部函数 *（续）*

指令	CMSIS 内部函数
REV16	uint32_t __REV16(uint32_t int value)
REVSH	uint32_t __REVSH(uint32_t int value)
SEV	void __SEV(void)
WFE	void __WFE(void)
WFI	void __WFI(void)

CMSIS 还提供使用 MRS 和 MSR 指令来访问特别寄存器的函数：

表 279. 通过 MRS/MSR 访问特殊寄存器的内部函数

特别寄存器	访问类型	CMSIS 函数
PRIMASK	读	uint32_t __get_PRIMASK (void)
	写	void __set_PRIMASK (uint32_t value)
CONTROL	读	uint32_t __get_CONTROL (void)
	写	void __set_CONTROL (uint32_t value)
MSP	读	uint32_t __get_MSP (void)
	写	void __set_MSP (uint32_t TopOfMainStack)
PSP	读	uint32_t __get_PSP (void)
	写	void __set_PSP (uint32_t TopOfProcStack)

25.4.3 关于指令描述

下列小节对如何使用指令进行了更为详细的描述：

- 第 25.4.3.1 节 “操作数”
- 第 25.4.3.2 节 “使用 PC 或 SP 时的限制”
- 第 25.4.3.3 节 “移位操作”
- 第 25.4.3.4 节 “地址对齐”
- 第 25.4.3.5 节 “PC 相对表达式”
- 第 25.4.3.6 节 “条件执行”。

25.4.3.1 操作数

指令操作数可以是 ARM 寄存器、常量或其他的指令特定参数。指令在操作数上操作，并经常将结果存放在目标寄存器中。当指令中存在目标寄存器时，它通常会在其他操作数之前被指定。

25.4.3.2 使用 PC 或 SP 时的限制

对于用于操作数或目标寄存器的**程序计数器** (PC) 或**协议栈指针**，许多指令都不能使用它们，或者存在着用户能否使用它们的限制。更多信息，详见指令的描述。

注：当使用 BX、BLX 或 POP 指令来更新 PC 时，为正确执行程序，任何地址的位 [0] 都必须为 1。这是因为该位指示目标指令集，且 Cortex-M0 处理器只支持 Thumb 指令。当 BL 或 BLX 指令将位 [0] 的值写入 LR 时，值 1 会被自动分配。

25.4.3.3 移位操作

寄存器移位操作通过特定的位数（**移位长度**）来实现寄存器位的左右移位操作。寄存器移位可以由指令 ASR、 LSR、 LSL 和 ROR 直接执行，且结果会被写入到目标寄存器中。允许的移位长度由移位类型和指令决定，请见各个指令的描述。若移位长度为 0，则不发生移位操作。寄存器移位操作会更新进位标志，当移位长度被指定为 0 时除外。下列各子节描述了各种移位操作以及它们是如何影响进位标志。在这些描述中，*Rm* 是包含着移位值的寄存器， *n* 是移位长度。

25.4.3.3.1 ASR

算术右移 *n* 位的操作是将 *Rm* 寄存器左边的 32-*n* 个位向右移动 *n* 位，结果是寄存器右边有 32-*n* 个位，然后再将寄存器位 [31] 的原始值复制到结果寄存器最左边的 *n* 个位中。参见图 25 – 71。

用户可以使用 ASR 对寄存器 *Rm* 的符号值进行 2^n 除法操作，得到的结果为负无穷大。

当指令为 ASRS 时，进位标志会被更新为最后移出的位值，即寄存器 *Rm* 的位 [*n*-1]。

注：

- 如果 *n* 为 32 或大于 32，那么结果中的所有位都会被置为 *Rm* 中位 [31] 的值。
- 如果 *n* 为 32 或大于 32，那么进位标志被更新为 *Rm* 位 [31] 的值。

图 71. ASR #3

25.4.3.3.2 LSR

逻辑右移 *n* 位的操作是将 *Rm* 寄存器左边的 32-*n* 个位向右移动 *n* 位，结果是寄存器右边有 32-*n* 个位，然后再将结果寄存器最左边的 *n* 个位设为 0。参见图 72。

用户可以使用 LSR 操作来对寄存器 *Rm* 的值进行 2^n 除法操作，如果值为无符号的整数。

当指令为 LSRS 时，进位标志会被更新为最后移出的位值，即寄存器 *Rm* 的位 [*n*-1]。

注：

- 如果 *n* 为 32 或大于 32，那么结果中的所有位都会被清除为 0。
- 如果 *n* 为 33 或大于 33，那么进位标志被更新为 0。

图 72. LSR #3

25.4.3.3.3 LSL

逻辑左移 *n* 位的操作是将 *Rm* 寄存器右边的 32-*n* 个位向左移动 *n* 位，结果是寄存器左边有 32-*n* 个位，然后再将结果寄存器最右边的 *n* 个位设为 0。参见图 73。

用户可以使用 LSL 操作来对寄存器 *Rm* 的值与 2^n 进行乘法操作，如果值为无符号的整数或是有符号 2 的补码整数值。溢出会在无警告的提示下发生。

当指令为 LSLS 时，进位标志会被更新为最后移出的位值，即寄存器 *Rm* 的位 [32-*n*]。当使用 LSL #0 时，这些指令不会影响进位标志。

注：

- 如果 *n* 为 32 或大于 32，那么结果中的所有位都会被清除为 0。
- 如果 *n* 为 33 或大于 33，那么进位标志被更新为 0。

图 73. LSL #3

25.4.3.3.4 ROR

循环右移 *n* 位的操作是将 *Rm* 寄存器左边的 32-*n* 个位向右移动 *n* 位，结果是寄存器右边有 32-*n* 个位，然后再将寄存器最右边的 *n* 个位移动到结果寄存器的最左边的 *n* 个位中。参见图 25 – 74。

当指令为 RORS 时，进位标志会被更新为最后循环出的位值，即寄存器 *Rm* 的位 [*n*-1]。

注：

- 如果 *n* 为 32，那么结果中的所有位值都相同于 *Rm* 中的值，且如果进位标志被更新，则会被更新为 *Rm* 的位 [31] 的值。
- ROR

  n 大于 32 的

  ROR

与移位长度为 *n*-32 的 ROR 操作得到的结果相同。

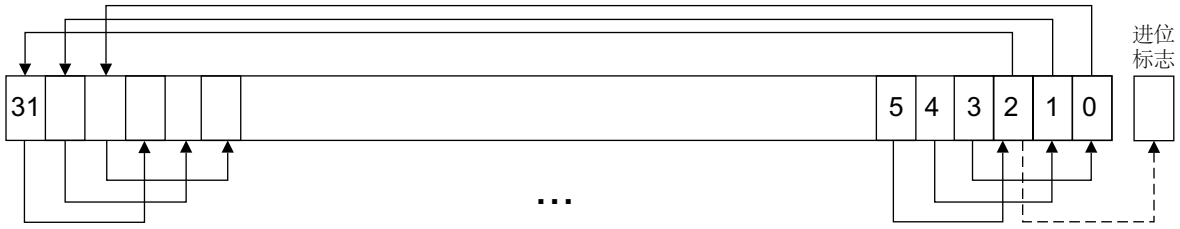


图 74. ROR #3

25.4.3.4 地址对齐

对齐访问是这样的一个操作：字对齐地址是用于字或多字访问，或者半字对齐地址是用于半字访问。字节访问通常是对齐访问的。

Cortex-M0 处理器不支持非对齐地址的访问。任何尝试执行一个非对齐的存储器访问操作都会导致 HardFault 异常。

25.4.3.5 PC 相对表达式

PC 相对表达式或**标签**是一个代表着指令或文字数据的地址的符号。在指令中它被表示为 PC 值加上或减去一个数字偏移量。汇编程序从标签和当前指令的地址中计算出所要求的偏移量。如果偏移量太大，则汇编程序会产生一个错误。

注：

- 对于多大数指令， PC 的值就是当前指令的地址加上 4 个字节。
- 汇编程序可能允许用其他语法来表示 PC 相对表达式，如标签加上或减去一个数值，或者用 [PC, *#imm*] 格式表示。

25.4.3.6 条件执行

大多数数据处理指令依据操作的结果在**应用程序状态寄存器** (APSR) 中更新条件标志，见第节。某些指令更新所有标志，而某些指令则仅更新子集。如果标志不被更新，则原始值被保留。有关指令对标志的影响，请见指令的描述。

用户可以根据另一个指令中设置的条件标志按以下方式执行条件性的跳转指令：

- 在指令更新标志后立即执行
- 在经过任意数量的不会更新标志的间隔数指令后执行。

在 Cortex-M0 处理器上，通过使用条件性的跳转指令，就可以实现条件性的执行操作。

本小节描述了以下内容：

- 第 25.4.3.6.1 节 “条件标志”
- 第 25.4.3.6.2 节 “条件代码后缀”。

25.4.3.6.1 条件标志

APSR 包含下列条件标志：

N — 当操作的结果为负值时置为 1，否则清除为 0。
Z — 当操作的结果为 0 时置为 1，否则清除为 0。
C — 当操作的结果导致要进位时置为 1，否则清除为 0。
V — 当操作引发溢出时置为 1，否则清除为 0。

有关 APSR 的更多信息，请见第 25 – 25.3.1.3.5 节。

当出现下列情况时，会发生进位操作：

- 如果加法的结果大于或等于 2^{32}
- 如果减法的结果为正或等于 0
- 由于移位指令或循环指令而发生的进位操作。

当位 [31] 中结果的符号值不与在无穷精度中所执行操作的结果符号值匹配时，溢出发生，例如：

- 如果二个负值相加得出一个正值
- 如果二个正值相加得出一个负值
- 如果一个负值减去一个正值得到一个正值
- 如果一个正值减去一个负值得到一个负值。

对于 CMP，比较操作与减法操作相同，对于 CMN，则加法操作相同，结果值会被丢弃除外。更多信息，详见指令的描述。

25.4.3.6.2 条件代码后缀

条件性跳转在语法描述显示为 B{*cond*}。只有 APSR 的条件代码标志符合指定的条件时，才能执行带有条件代码的跳转指令，否则要忽略跳转指令。显示了使用的条件代码。

表 280 同时还显示了条件代码后缀和 N、 Z、 C 和 V 标志之间的联系。

表 280. 条件代码后缀

后缀	标志	意义
EQ	Z = 1	等效，最后标志设置结果为 0
NE	Z = 0	不等效，最后标志设置结果为非 0
CS 或 HS	C = 1	更高或相同，无符号
CC 或 LO	C = 0	更低，无符号
MI	N = 1	负数
PL	N = 0	正数或 0
VS	V = 1	溢出
VC	V = 0	无溢出
HI	C = 1 和 Z = 0	更高，无符号
LS	C = 0 或 Z = 1	更低或相同，无符号
GE	N = V	大于或等效，有符号
LT	N != V	少于，有符号

表 280. 条件代码后缀 *（续）*

后缀	标志	意义
GT	Z = 0 和 N = V	大于，有符号
LE	Z = 1 和 N != V	少于或等于，有符号
AL	可以为任意值	总是。当没有指定后缀时，这是默认的操作。

25.4.4 存储器访问指令

表 281 所示为存储器访问指令：

表 281. 访问指令

助记符	简述	参见
LDR{type}	使用寄存器偏移量来加载寄存器	第 25 – 25.4.4.3 节
LDR	从基于 PC 相对偏移的地址加载寄存器	第 25 – 25.4.4.4 节
POP	出栈，将协议栈的内容放入寄存器	第 25 – 25.4.4.6 节
PUSH	压栈，将寄存器的内容压入协议栈	第 25 – 25.4.4.6 节
STM	存储多个寄存器	第 25 – 25.4.4.5 节
STR{type}	使用立即数偏移量来存储寄存器	第 25 – 25.4.4.2 节
STR{type}	使用寄存器偏移量来存储寄存器	第 25 – 25.4.4.3 节

25.4.4.1 ADR

产生一个 PC 相对地址。

25.4.4.1.1 语法

ADR *Rd*, *label*

其中：

Rd 是目标寄存器。

label 是 PC-相对表达式。参见第 25 – 25.4.3.5 节。

25.4.4.1.2 操作

ADR 通过将立即数值加入到 PC 中来产生一个地址，并将得到的地址结果写入到目标寄存器中。

ADR 产生区域独立代码非常便利，因为地址是 PC 相对地址。

如果用户使用 ADR 来产生 BX 或 BLX 指令的目标地址，为了能正确执行程序，必须要保证将产生的地址的位 [0] 设置为 1。

25.4.4.1.3 限制

在该指令中，*Rd* 必须指定 R0-R7。地址数据值必须是字对齐，且不能超出当前 PC 的 1020 字节。

25.4.4.1.4 条件标志

该指令不会改变标志。

25.4.4.1.5 示例

```
ADR    R1, TextMessage    ; 将标签为
                                ; TextMessage 单元上的地址值写入到 R1 中

ADR    R3, [PC,#996]      ; 将 R3 的值设为 PC + 996。
```

25.4.4.2 LDR 和 STR，立即数偏移量

具有立即数偏移量的加载和存储。

25.4.4.2.1 语法

LDR *Rt*, [<*Rn* | SP> {, #*imm*}]

LDR<B|H> *Rt*, [*Rn* {, #*imm*}]

STR *Rt*, [<*Rn* | SP>, {,#*imm*}]

STR<B|H> *Rt*, [*Rn* {,#*imm*}]

其中：

Rt *是加载或存储的寄存器。*

Rn *是基于存储器地址上的寄存器。*

imm 是 *Rn* 的偏移量。如果 *imm* 被省略，则假设它为 0。

25.4.4.2.2 操作

LDR、LDRB 和 LDRH 指令将存储器中的字、字节或半字数据值加载到 *Rt* 指定的寄存器中。在将数据写入 *Rt* 指定的寄存器之前，长度少于字的数据要用 0 扩充到 32 位的长度。

STR、STRB 和 STRH 指令将 *Rt* 寄存器指定的单个寄存器中所包含的字、最低有效位字节或低半字存放到存储器中。从加载的存储器地址或用于存放的存储器地址是 *Rn* 或 SP 所指定的寄存器的值与立即数 *imm* 的和。

25.4.4.2.3 限制

在这些指令中：

- *Rt* 和 *Rn* 必须只能指定 R0-R7。
- *imm* 的值必须要符合下列要求：
  - 0 到 1020 之间，对于 LDR 和 STR 操作，
    在将 SP 用作基址寄存器时，其值必须是 4 的整数倍
  - 0 到 124 之间，对于 LDR 和 STR 操作，
    在将 R0-R7 用作基址寄存器时，其值必须是 4 的整数倍
  - 0 到 62 之间，对于 LDRH 和 STRH 操作，其值必须是 2 的整数倍
  - 0 到 31 之间，对于 LDRB 和 STRB。
- 计算出的地址必须能够被事务中的字节数整除，见第 25 – 25.4.3.4 节。

25.4.4.2.4 条件标志

这些指令不会改变标志。

25.4.4.2.5 示例

```
LDR     R4, [R7                 ; 将 R7 的地址载入到 R4。
STR     R2, [R0,#const-struc]   ; const-struc 是评估
                                ; 处于 0-1020 范围内的常量的表达式。
```

25.4.4.3 LDR 和 STR，寄存器偏移量

带寄存器偏移量的加载和存储。

25.4.4.3.1 语法

LDR *Rt*, [*Rn*, *Rm*]

LDR<B|H> *Rt*, [*Rn*, *Rm*]

LDR<SB|SH> *Rt*, [*Rn*, *Rm*]

STR *Rt*, [*Rn*, *Rm*]

STR<B|H> *Rt*, [*Rn*, *Rm*]

其中：

Rt 是加载或存储的寄存器。

Rn 是基于存储器地址上的寄存器。

Rm 是含有用作偏移量的值的寄存器。

25.4.4.3.2 操作

LDR、LDRB、U、LDRSB 和 LDRSH 将存储器中的字、0 扩展字节、0 扩展半字、符号扩展字节或符号扩展半字值加载到 *Rt* 指定的寄存器中。

STR、STRB 和 STRH 指令将 *Rt* 寄存器指定的单个寄存器中所包含的字，最低有效位字节或低半字存放到存储器中。

从中加载的存储器地址或用于存放的存储器地址是 *Rn* 和 *Rm* 所指定的寄存器中的值之和。

25.4.4.3.3 限制

在这些指令中：

- *Rt*、 *Rn* 和 *Rm* 必须只能指定 R0-R7。
- 计算出的地址必须能够被加载或存储的字节数整除，见第 25 – 25.4.3.4 节。

25.4.4.3.4 条件标志

这些指令不会改变标志。

25.4.4.3.5 示例

```
STR     R0, [R5, R1]          ; 将 R0 的值存储到
                              ; R5 加 R1 得出的地址中

LDRSH   R1, [R2, R3]          ; 从 (R2 + R3) 所指定的存储器地址中加载半字数据，
                              ; 符号扩展到 32 位并

                              ; 将其写入到 R1 中。
```

25.4.4.4 LDR， PC-相对

从存储器中加载寄存器 （文字数据）。

25.4.4.4.1 语法

LDR *Rt*, *label*

其中：

Rt 是加载的寄存器。

label 是 PC-相对表达式。参见第 25 – 25.4.3.5 节。

25.4.4.4.2 操作

将 *label* 所指定的存储器中的字加载到 *Rt* 所指定的寄存器中。

25.4.4.4.3 限制

在这些指令中， *label* 的大小必须位于当前 PC 的 1020 字节范围之内，且是字对齐的。

25.4.4.4.4 条件标志

这些指令不会改变标志。

25.4.4.4.5 示例

```
LDR     R0, LookUpTable   ; 将标签为 LookUpTable 的地址中的字数据
                          ; 加载到 R0 中。

LDR     R3, [PC, #100]    ; 将 (PC + 100) 上的存储器字加载到 R3 中。
```

25.4.4.5 LDM 和 STM

加载和存储多个寄存器。

25.4.4.5.1 语法

LDM *Rn*{!}, reglist

STM *Rn*!, reglist

其中：

Rn 是存储器地址所基于的寄存器。

! 回写后缀。

reglist 是被加载或存储的一个或多个寄存器的列表，用大括号括住。它可包含寄存器范围。若它包含多于一个的寄存器或寄存器范围，必须要将其用逗号隔开，见第 25 – 25.4.4.5.5 节。

对于 LDM，LDMIA 和 LDMFD 相近。LDMIA 为每次访问后都会递增的基址寄存器。LDMFD 用法是将数据从满的递减协议栈中移出。

对于 STM，STMIA 和 STMEA 相近。STMIA 为每次访问后都会递增的基址寄存器。STMEA 用法是将数据压入空的递增协议栈中。

25.4.4.5.2 操作

LDM 指令将基于 *Rn* 上的存储器地址的字值加载到 *reglist* 的寄存器中。

STM 指令将 *reglist* 中的寄存器的字值存放到基于 *Rn* 的存储器地址中。

用于访问的存储器地址为 4 字节间隔，其范围为 *Rn* 所指定的寄存器的值至 *Rn* + 4 * (*n*-1) 所指定的寄存器的值，这里的 *n* 是 *reglist* 中的寄存器数量。访问的顺序是按照寄存器的编号从低到高发生，最低编号的寄存器使用最低的存储器地址，最高编号的寄存器使用最高的存储器地址。如果写回后缀被指定，则 *Rn* + 4 **n* 所指定的寄存器的值会被写回到 *Rn* 所指定的寄存器中。

25.4.4.5.3 限制

在这些指令中：

- *reglist* 和 *Rn* 限制为 R0-R7。
- 必须始终使用写回后缀，除非指令是LDM指令，在LDM里，reglist也含有*Rn*，在这种情况下，不得使用写回后缀。
- *Rn* 所指定的寄存器的值必须是字对齐的。更多信息请见第 25 – 25.4.3.4 节。
- 对于 STM，如果 *reglist* 中存在着 *Rn*，那么 Rn 必须是列表中的第一个寄存器。

25.4.4.5.4 条件标志

这些指令不会改变标志。

25.4.4.5.5 示例

```
LDM     R0,{R0,R3,R4}       ; LDMIA 相近于 LDM
      STMIA   R1!,{R2-R4,R6}
```

25.4.4.5.6 错误示例

```
STM     R5!,{R4,R5,R6} ; 存放于 R5 的值是不可预测的
      LDM     R2,{}           ; 在列表中至少要存在着一个寄存器
```

25.4.4.6 PUSH 和 POP

将寄存器压入满递减协议栈和将满递减协议栈中的内容移入寄存器。

25.4.4.6.1 语法

PUSH *reglist*

POP *reglist*

其中：

reglist 是非空的寄存器列表，用大括号括着。它可包含寄存器范围。若它包含多于一个的寄存器或寄存器范围，必须要将其用逗号隔开。

25.4.4.6.2 操作

PUSH 将寄存器存放到协议栈中，最低编号的寄存器使用最低存储器地址，最高编号的寄存器使用最高存储器地址。

POP 将协议栈中的内容加载到寄存器中，最低编号的寄存器使用最低存储器地址，最高编号的寄存器使用最高存储器地址。

PUSH 将 SP 寄存器的值减去 4 所得的值用作最高存储器地址，

POP 将 SP 寄存器的值用作最低的存储器地址来执行满递减协议栈操作。当操作完成时，

PUSH 会更新 SP 寄存器来指向最低存储值的单元，

而 POP 则会更新 SP 寄存器来指向高于所加载的最高单元的单元。

如果 POP 指令在它的 *reglist* 中包含了 PC，则当 POP 指令完成时，会在该单元上执行一个跳转操作。为 PC 所读出的 Bit[0] 值用来更新 APSR T- 位。该位必须为 1，以确保能正确执行程序。

25.4.4.6.3 限制

在这些指令中：

- *reglist* 必须只为 R0-R7。
- 对于 PUSH 和 POP，异常情况分别是 LR 和 PC。

25.4.4.6.4 条件标志

这些指令不会改变标志。

25.4.4.6.5 示例

```
PUSH    {R0,R4-R7}      ; 将 R0、R4、R5、R6、R7 压入协议栈
        PUSH    {R2,LR}         ; 将 R2 和链接寄存器压入协议栈
        POP     {R0,R6,PC}      ; 令 r0、r6 和 PC 出栈，然后跳转到
                                ; 新的 PC 值。
```

25.4.5 通用数据处理指令

表 282 显示了数据处理指令：

表 282. 数据处理指令

助记符	简述	参见
ADCS	进位加法	第 25 – 25.4.5.1 节
ADD{S}	加法	第 25 – 25.4.5.1 节
ANDS	逻辑与	第 25 – 25.4.5.2 节
ASRS	算术右移	第 25 – 25.4.5.3 节
BICS	位清除	第 25 – 25.4.5.2 节
CMN	比较负值	第 25 – 25.4.5.4 节
CMP	比较	第 25 – 25.4.5.4 节
EORS	异或	第 25 – 25.4.5.2 节
LSLS	逻辑左移	第 25 – 25.4.5.3 节
LSRS	逻辑右移	第 25 – 25.4.5.3 节
MOV{S}	传输	第 25 – 25.4.5.5 节
MULS	乘法	第 25 – 25.4.5.6 节
MVNS	取反传输	第 25 – 25.4.5.5 节
ORRS	逻辑或	第 25 – 25.4.5.2 节
REV	反转字里面的字节顺序	第 25 – 25.4.5.7 节
REV16	反转每半字里面的字节顺序	第 25 – 25.4.5.7 节
REVSH	反转低半字中的字节顺序，并进行符号扩展	第 25 – 25.4.5.7 节
RORS	循环右移	第 25 – 25.4.5.3 节
RSBS	反向减法	第 25 – 25.4.5.1 节
SBCS	进位减法	第 25 – 25.4.5.1 节
SUBS	减法	第 25 – 25.4.5.1 节
SXTB	符号扩展字节	第 25 – 25.4.5.8 节
SXTH	符号扩展半字	第 25 – 25.4.5.8 节
UXTB	0 扩展字节	第 25 – 25.4.5.8 节
UXTH	0 扩展半字	第 25 – 25.4.5.8 节
TST	测试	第 25 – 25.4.5.9 节

25.4.5.1 ADC、 ADD、 RSB 和 SUB

进位加法、加法、反向减法、进位减法、减法。

25.4.5.1.1 语法

ADCS {*Rd*,} *Rn*, *Rm*

ADD{S} {*Rd*,} *Rn*, <*Rm*|*#imm*>

RSBS {*Rd*,} *Rn*, *Rm*, *#0*

SBCS {*Rd*,} *Rn*, *Rm*

SUB{S} {*Rd*,} *Rn*,

<*Rm*|*#imm*>

其中：

S 会令 ADD 或 SUB 指令更新标志

Rd 指定结果寄存器

Rn 指定首个源寄存器

Rm 指定第二个源寄存器

imm 指定一个常量立即数值。

当省略了可选的 *Rd* 寄存器限定符时，会假定其值与 *Rn* 相同，例如， ADDS R1、 R2 与 ADDS R1、 R1、 R2 相同。

25.4.5.1.2 操作

ADCS 指令将 *Rn* 中的值加到 *Rm* 的值中，如果进位标志被置位，则加多一个 1，并将结果存放在 *Rd* 所指定寄存器里，同时更新 N、 Z、 C 和 V 标志。

ADD 指令将 *Rn* 的值加到 *Rm* 的值或 *imm* 指定的立即数中，并将结果存放到 *Rd* 所指定的寄存器中。

ADDS 指令执行的操作与 ADD 相同，并还可以更新 N、 Z、 C 和 V 标志。

RSBS 指令是用 0 减去 *Rn* 中的值，得到一个负数，然后将结果值存放在 Rd 所指定的寄存器中，并更新 N、 Z、 C 和 V 标志。

SBCS 指令是用 *Rm* 的值减去 *Rm* 的值，如果进位标志置位，则减去一个 1。指令会将结果值存放到 Rd 所指定的寄存器中，并更新 N、 Z、 C 和 V 标志。

SUB 指令会减去 *Rm* 的值或 *imm* 所指定的立即数。指令把结果值存放到 *Rd* 所指定的寄存器中。

SUBS 指令执行的操作与 SUB 相同，同时它还可以更新 N、 Z、 C 和 V 标志。

有关如何使用 ADC 和 SBC 来综合处理多字算术，请见第 25.4.5.1.4 节。

另请参见第 25 – 25.4.4.1 节。

25.4.5.1.3 限制

表 283 列出了寄存器指示符的合法组合和每一个指令可以使用的立即数。

表 283. ADC、 ADD、 RSB、 SBC 和 SUB 操作数限制

指令	Rd	Rn	Rm	imm	限制
ADCS	R0-R7	R0-R7	R0-R7	-	*Rd* 和 *Rn* 必须指定相同的寄存器。
ADD	R0-R15	R0-R15	R0-PC	-	*Rd* 和 *Rn* 必须指定相同的寄存器。 *Rn* 和 *Rm* 必须不能同时指定 PC。
	R0-R7	SP 或 PC	-	0-1020	立即数必须为 4 的整数倍。
	SP	SP	-	0-508	立即数必须为 4 的整数倍。
ADDS	R0-R7	R0-R7	-	0-7	-
	R0-R7	R0-R7	-	0-255	*Rd* 和 *Rn* 必须指定相同的寄存器。
	R0-R7	R0-R7	R0-R7	-	-
RSBS	R0-R7	R0-R7	-	-	-
SBCS	R0-R7	R0-R7	R0-R7	-	*Rd* 和 *Rn* 必须指定相同的寄存器。
SUB	SP	SP	-	0-508	立即数必须为 4 的整数倍。
SUBS	R0-R7	R0-R7	-	0-7	-
	R0-R7	R0-R7	-	0-255	Rd 和 *Rn* 必须指定相同的寄存器。
	R0-R7	R0-R7	R0-R7	-	-

25.4.5.1.4 示例

下例所示为二个指令将 R0 和 R1 所包含的 64 位整数值加到 R2 和 R3 所包含的另一个 64 位整数值中，并将结果存放到 R0 和 R1 中。

64 位加法：

```
ADDS    R0, R0, R2    ; 加上最低有效位的字
ADCS    R1, R1, R3    ; 加上最高有效位的字，带进位
```

多字的值不需要使用连续的寄存器。下面示例为指令会令 R4、R5 和 R6 所包含的 96 位整数值减去 R1、 R2 和 R3 所包含的 96 位整数值。该例将结果值存放在 R4、 R5 和 R6 中。

96 位减法：

```
SUBS    R4, R4, R1    ; 减去最低有效位的字
SBCS    R5, R5, R2    ; 减去中间的字，带进位
SBCS    R6, R6, R3    ; 减去最高有效位的字，带进位
```

下列所示的 RSBS 指令是用来执行单个寄存器 1 的补码的操作。

算术负值运算： `RSBS R7, R7, #0 ; 用 0 减去 R7`

25.4.5.2 AND、 ORR、 EOR 和 BIC

逻辑 AND、 OR、异或和位清除。

25.4.5.2.1 语法

ANDS {*Rd,*} *Rn*, *Rm*

ORRS {*Rd,*} *Rn*, *Rm*

EORS {*Rd,*} *Rn*, *Rm*

BICS {*Rd,*} *Rn*, *Rm*

其中：

Rd 是目标寄存器。

Rn 是保存第一个操作数的寄存器，且还是与目标寄存器相同的寄存器。

Rm 是第二个寄存器。

25.4.5.2.2 操作

AND、 EOR 和 ORR 对 *Rn* 和 *Rm* 的值按位执行 AND、异或、或操作。

BIC 指令对 *Rn* 上的位执行 AND 操作，对 *Rm* 值上的相应位执行逻辑非操作。

条件代码标志会根据操作的结果被更新，见第 25.4.3.6.1 节。

25.4.5.2.3 限制

在这些指令中， *Rd*、 *Rn* 和 *Rm* 必须只能指定 R0-R7。

25.4.5.2.4 条件标志

这些指令：

- 根据结果值来更新 N 和 Z 标志
- 不影响 C 或 V 标志。

25.4.5.2.5 示例

```
ANDS    R2, R2, R1
      ORRS    R2, R2, R5
      ANDS    R5, R5, R8
      EORS    R7, R7, R6
      BICS    R0, R0, R1
```

25.4.5.3 ASR、 LSL、 LSR 和 ROR

算术右移、逻辑左移、逻辑右移、循环右移。

25.4.5.3.1 语法

ASRS {Rd,} *Rm*, *Rs*

ASRS {Rd,} *Rm*, *#imm*

LSLS {Rd,} *Rm*, *Rs*

LSLS {Rd,} *Rm*, *#imm*

LSRS {Rd,} *Rm*, *Rs*

LSRS {Rd,} *Rm*, *#imm*

RORS {Rd,} *Rm*, *Rs*

其中：

Rd 是目标寄存器。如果 *Rd* 被省略，则假定它的值与 *Rm* 相同。

Rm 是保存要移位的值的寄存器。

Rs 是保存着移位长度 （该长度要应用到 *Rm* 中的值）的寄存器。

imm 是移位长度。

移位长度要由指令来决定：

ASR — 移位长度为 1 到 32
LSL — 移位长度为 0 到 31
LSR — 移位长度为 1 到 32。

注：MOVS Rd, *Rm* 是 LSLS Rd, *Rm*, #0 的假名。

25.4.5.3.2 操作

ASR、LSL、LSR 和 ROR 按照立即数 *imm* 所指定的或者 *Rs* 所指定寄存器的最低有效位字节值所指定的长度对 *Rm* 寄存器的位执行算术左移、逻辑左移、逻辑右移或循环右移。

关于不同的指令会产生什么样的结果，详情请见第 25 – 25.4.3.3 节。

25.4.5.3.3 限制

在这些指令中，*Rd*、*Rm* 和 *Rs* 必须只能指定 R0-R7。对于非立即数指令，*Rd* 和 *Rm* 必须指定相同的寄存器。

25.4.5.3.4 条件标志

这些指令根据结果值来更新 N 和 Z 标志。

C 标志被更新为最后移出的位，移位长度为 0 时除外，见第 25 – 25.4.3.3 节。V 标志不变。

25.4.5.3.5 示例

```
ASRS    R7, R5, #9  ; 算术右移 9 位
        LSLS    R1, R2, #3  ; 逻辑左移 3 位，并更新标志
        LSRS    R4, R5, #6  ; 逻辑右移 6 位
        RORS    R4, R4, R6  ; 循环右移 R6 低字节中的值。
```

25.4.5.4 CMP 和 CMN

比较和比较负值。

25.4.5.4.1 语法

CMN *Rn*, *Rm*

CMP *Rn*, *#imm*

CMP *Rn*, *Rm*

其中：

Rn 是保存第一个操作数的寄存器。

Rm 是用于比较的寄存器。

imm 是用于比较的立即数值。

25.4.5.4.2 操作

这些指令将一个寄存器中的值与另一个寄存器中的值或立即数进行比较。指令会根据结果值来更新条件标志，但不会将结果写入寄存器。

CMP 指令将 *Rn* 的值减去 *Rm* 所指定的寄存器值或立即数 *imm*，并更新标志。这操作与 SUBS 指令相同，不同的是结果值会被丢弃。

CMN 指令将 *Rm* 的值加到 *Rn* 的值中，并更新标志。这操作与 ADDS 指令相同，不同的是结果值会被丢弃。

25.4.5.4.3 限制

对于：

- CMN

  指令 *Rn* 和 *Rm* 必须只能指定 R0-R7。

- CMP 指令：
  - *Rn* 和 *Rm* 可以指定 R0-R14
  - 立即数的范围为 0-255。

25.4.5.4.4 条件标志

这些指令根据结果值来更新 N、 Z、 C 和 V 标志。

25.4.5.4.5 示例

```
CMP     R2, R9
     CMN     R0, R2
```

25.4.5.5 MOV 和 MVN

传输和取反传输。

25.4.5.5.1 语法

MOV{S} *Rd*, *Rm*

MOVS *Rd*, *#imm*

MVNS *Rd*, *Rm*

其中：

S 是可选后缀。如果指定了 *S*，则会根据操作的结果值来更新条件代码标志，见第 25 – 25.4.3.6 节。

Rd 是目标寄存器。

Rm 是寄存器。

imm 可以是 0-255 范围内的任何一个值。

25.4.5.5.2 操作

MOV 指令将 *Rm* 的值复制到 *Rd* 中。

MOVS 指令执行的操作与 MOV 指令相同，但是它会更新 N 和 Z 标志。

MVNS 指令采用 *Rm* 的值，对该值执行按位的逻辑取反操作，并将结果存放到 *Rd* 中。

25.4.5.5.3 限制

在这些指令中， *Rd* 和 *Rm* 必须只能指定 R0-R7。

当在 MOV 指令里 *Rd* 是 PC 时：

- 结果值的位 [0] 被丢弃。
- 在通过将结果值的位 [0] 强制为 0 来创造的地址上执行跳转操作。 T- 位保持不变。

注：尽管可以将 MOV 用作跳转指令，但是为了软件的可移植性，ARM 强烈建推荐使用 BX 或 BLX 指令来执行跳转操作。

25.4.5.5.4 条件标志

如果 *S* 被指定，则这些指令：

- 根据结果值来更新 N 和 Z 标志
- 不影响 C 或 V 标志。

25.4.5.5.5 示例

```
MOVS  R0, #0x000B     ; 将 0x000B 写入 R0，更新标志
      MOVS  R1, #0x0        ; 将 0 写入 R1，更新标志
      MOV   R10, R12        ; 将 R12 的值写入 R10，不更新标志
      MOVS  R3, #23         ; 将 23 写入 R3
      MOV   R8, SP          ; 将协议栈指针的值写入 R8
      MVNS  R2, R0          ; 将 R0 取反写入 R2 并更新标志
```

25.4.5.6 MULS

使用 32 位操作数的乘法，产生 32 位的结果值。

25.4.5.6.1 语法

MULS Rd, *Rn*, *Rm*

其中：

Rd 是目标寄存器。

Rn、 *Rm* 是保存进行乘法操作值的寄存器。

25.4.5.6.2 操作

MUL 指令将 *Rn* 和 *Rm* 所指定的寄存器的值进行乘法操作，并将结果值的最低有效 32 位存放在 *Rd* 中。条件代码标志会根据操作的结果被更新，见第 25 – 25.4.3.6 节。

该指令的结果并不是由操作数是有符号还是无符号来决定。

25.4.5.6.3 限制

在该指令中：

- *Rd*、 *Rn* 和 *Rm* 必须只能指定 R0-R7
- *Rd* 必须要和 *Rm* 相同。

25.4.5.6.4 条件标志

该指令：

- 根据结果值来更新 N 和 Z 标志
- 不影响 C 或 V 标志。

25.4.5.6.5 示例

```
MULS    R0, R2, R0      ; 乘法操作，标志被更新，R0 = R0 x R2
```

25.4.5.7 REV、 REV16 和 REVSH

反转字节。

25.4.5.7.1 语法

REV Rd, *Rn*

REV16 Rd, *Rn*

REVSH Rd, *Rn*

其中：

Rd 是目标寄存器。

Rn 是源寄存器。

25.4.5.7.2 操作

使用这些指令来改变数据的端点排序：

REV — 将 32 位大端数据转换成小端的数据或将 32 位小端的数据转换成大端数据。

REV16 — 将二个打包的 16 位大端数据转换成小端的数据或将二个打包的 16 位小端的数据转换成大端数据。

REVSH — 将 16 位有符号的大端数据转换成 32 位有符号小端数据或将 16 位有符号小端数据转换成 32 位有符号大端数据。

25.4.5.7.3 限制

在这些指令中， *Rd* 和 *Rn* 必须只能指定 R0-R7。

25.4.5.7.4 条件标志

这些指令不会改变标志。

25.4.5.7.5 示例

```
REV    R3, R7  ; 反转 R7 值的字节顺序，并将其写入 R3
REV16  R0, R0  ; 反转 R0 中的每一个 16 位半字的字节顺序
REVSH  R0, R5  ; 反转有符号的半字
```

25.4.5.8 SXT 和 UXT

符号扩展和 0 扩展。

25.4.5.8.1 语法

SXTB Rd, *Rm*

SXTH Rd, *Rm*

UXTB Rd, *Rm*

UXTH Rd, *Rm*

其中：

Rd 是目标寄存器。

Rm 是寄存器，其保存的值会被扩展。

25.4.5.8.2 操作

这些指令从结果值中提取位：

- SXTB 提取位 [7:0] 并将值进行符号扩展到 32 位
- UXTB 提取位 [7:0] 并将值用 0 扩展到 32 位
- SXTH 提取位 [15:0] 并将值进行符号扩展到 32 位
- UXTH 提取位 [15:0] 并将值用 0 扩展到 32 位。

25.4.5.8.3 限制

在这些指令中， *Rd* 和 *Rm* 必须只能指定 R0-R7。

25.4.5.8.4 条件标志

这些指令不会影响标志。

25.4.5.8.5 示例

```
SXTH   R4, R6           ; 获取 R6 的低半字，
                        ; 然后将其进行符号扩展到 32 位，
                        ; 并将结果写入 R4。
       UXTB   R3, R1    ; 获取 R10 最低位字节，并用 0
                        ; 扩展，最后将结果写入 R3
```

25.4.5.9 TST

测试位。

25.4.5.9.1 语法

TST *Rn*, *Rm*

其中：

Rn 是保存第一个操作数的寄存器。

Rm 是测试的寄存器。

25.4.5.9.2 操作

该指令将一个寄存器的值与别一个寄存器中的值进行测试。它会根据结果值来更新条件标志，但是不会将结果值写入寄存器。

TST 指令对 *Rn* 中的值和 *Rm* 中的值执行位与操作。这是与 ANDS 指令相同的操作，不同的是它会丢弃结果值。

为了测试 *Rn* 中的某个位是 0 还是 1，要使用 TST 指令，且寄存器的该位要设为 1，其他所有位被清除为 0。

25.4.5.9.3 限制

在这些指令中， *Rn* 和 *Rm* 必须只能指定 R0-R7。

25.4.5.9.4 条件标志

该指令：

- 根据结果值来更新 N 和 Z 标志
- 不影响 C 或 V 标志。

25.4.5.9.5 示例

```
TST     R0, R1  ; 对 R0 值和 R1 值执行位与操作，
                ; 更新条件代码标志，但结果值会被丢弃
```

25.4.6 跳转和控制指令

表 284 所示为跳转和控制指令：

表 284. 跳转和控制指令

助记符	简述	参见
B{cc}	跳转 { 有条件 }	第 25 – 25.4.6.1 节
BL	带链接的跳转	第 25 – 25.4.6.1 节
BLX	带链接的间接跳转	第 25 – 25.4.6.1 节
BX	间接跳转	第 25 – 25.4.6.1 节

25.4.6.1 B、 BL、 BX 和 BLX

跳转指令。

25.4.6.1.1 语法

B{*cond*} *label*

BL *label*

BX *Rm*

BLX *Rm*

其中：

cond 是可选的条件代码，见第 25 – 25.4.3.6 节。

label 是 PC-相对表达式。参见第 25 – 25.4.3.5 节。

Rm 是提供跳转地址的寄存器。

25.4.6.1.2 操作

所有这些指令都会对 *label* 所指示的地址或在 *Rm* 所指定的寄存器中包含地址上执行跳转操作。另外：

- BL 和 BLX 指令将下一个指令的地址写入 LR，链接寄存器 R14。
- 如果 *Rm* 的位 [0] 是 0，则 BX 和 BLX 指令会导致 HardFault 异常。

BL 和 BLX 指令还会将 LR 的位 [0] 设置为 1。这就确保了该值适合由后续 POP {PC} 或 BX 指令使用其来执行成功的返回跳转操作。

表 285 所示为适用于各种跳转指令的跳转范围。

表 285. 跳转范围

指令	跳转范围
B *label*	–2 KB 至 +2 KB
B*cond label*	–256 字节至 +254 字节
BL *label*	–16 MB 至 +16 MB
BX *Rm*	寄存器中的任意值
BLX *Rm*	寄存器中的任意值

25.4.6.1.3 限制

在这些指令中：

- 不要在 BX 或 BLX 指令里使用 SP 或 PC。
- 对于BX和BLX，为实现正确的执行操作，*Rm*的位[0]必须为1。位[0]用于更新EPSR T-位，并会被从目标地址上丢弃。

注：B*cond* 是在 Cortex-M0 处理器上唯一的条件指令。

25.4.6.1.4 条件标志

这些指令不会改变标志。

25.4.6.1.5 示例

```
B       loopA  ; 跳转到 loopA
        BL      funC    ; 对函数 funC 进行带链接的跳转（调用），返回存放在
                        ; LR 的地址
        BX      LR      ; 从函数调用中返回
        BLX     R0      ; 带链接的跳转，并从（调用）中更改为存放在
                        ; R0 所存放的地址

BEQ     labelD ; 条件跳转到 labelD，如果最后的标志设置
                ; 指令设置 Z 标志，否则不执行跳转。
```

25.4.7 其他指令

表 286 所示为余下的 Cortex-M0 指令：

表 286. 其他指令

助记符	简述	参见
BKPT	断点	第 25 – 25.4.7.1 节
CPSID	更改处理器状态，禁用中断	第 25 – 25.4.7.2 节
CPSIE	更改处理器状态，使能中断	第 25 – 25.4.7.2 节
DMB	数据内存屏障	第 25 – 25.4.7.3 节
DSB	数据同步屏障	第 25 – 25.4.7.4 节
ISB	指令同步屏障	第 25 – 25.4.7.5 节
MRS	从特别寄存器传输到寄存器	第 25 – 25.4.7.6 节
MSR	从寄存器传输到特别寄存器	第 25 – 25.4.7.7 节
NOP	无操作	第 25 – 25.4.7.8 节
SEV	发送事件	第 25 – 25.4.7.9 节
SVC	超级用户调用	第 25 – 25.4.7.10 节
WFE	等待事件	第 25 – 25.4.7.11 节
WFI	等待中断	第 25 – 25.4.7.12 节

25.4.7.1 BKPT

断点。

25.4.7.1.1 语法

BKPT #*imm*

其中：

imm 是 0-255 范围内的整数。

25.4.7.1.2 操作

BKPT 指令会令处理器进入调试状态。当指令到达特定的地址时，调试工具可以使用该指令来调查系统状态。处理器会忽略 *imm*。如有需要，调试器可以使用它来存放断点的其他信息。

如果在执行 BKPT 指令时调试器没有连接上，那么处理器还有可能会产生 HardFault 或进入锁定状态。更多信息请见第 25 – 25.3.4.1 节。

25.4.7.1.3 限制

该指令没有限制。

25.4.7.1.4 条件标志

该指令不会改变标志。

25.4.7.1.5 示例

```
BKPT #0      ; 立即数值设为 0x0 的断点。
```

25.4.7.2 CPS

更改处理器状态。

25.4.7.2.1 语法

CPSID i

CPSIE i

25.4.7.2.2 操作

CPS 更改 PRIMASK 特别寄存器值。通过设置 PRIMASK，CPSID 可禁用中断。而通过清除 PRIMASK，CPSIE 则可使能中断。关于这些寄存器的详细信息，请见第 25 – 25.3.1.3.6 节。

25.4.7.2.3 限制

该指令没有限制。

25.4.7.2.4 条件标志

该指令不会更改条件标志。

25.4.7.2.5 示例

```
CPSID i ; 禁用所有的中断，NMI 除外 ( 设置 PRIMASK)

CPSIE i ; 使能中断 ( 清除 PRIMASK)
```

25.4.7.3 DMB

数据内存屏障。

25.4.7.3.1 语法

DMB

25.4.7.3.2 操作

DMB 用作数据内存屏障。它可确保会先检测到程序中位于 DMB 指令前的所有显式内存访问指令，然后再检测到程序中位于 DMB 指令后的显式内存访问指令。 DMB 不影响其他指令（不访问内存的指令）在处理器上的执行顺序。

25.4.7.3.3 限制

该指令没有限制。

25.4.7.3.4 条件标志

该指令不会改变标志。

25.4.7.3.5 示例

```
DMB  ; 数据内存屏障
```

25.4.7.4 DSB

数据同步屏障。

25.4.7.4.1 语法

DSB

25.4.7.4.2 操作

DSB 用作特别数据同步内存屏障。只有当此指令执行完毕后，才会执行程序中位于此指令后的指令。位于此指令前的所有显式内存访问均完成时， DSB 指令才会完成。

25.4.7.4.3 限制

该指令没有限制。

25.4.7.4.4 条件标志

该指令不会改变标志。

25.4.7.4.5 示例

```
DSB ; 数据同步屏障
```

25.4.7.5 ISB

指令同步屏障。

25.4.7.5.1 语法

ISB

25.4.7.5.2 操作

ISB 用作指令同步屏障。它会刷新处理器的管道，因此在完成了 ISB 指令后，需要再次将 ISB 之后的所有指令从高速缓存或内存中提取出来。

25.4.7.5.3 限制

该指令没有限制。

25.4.7.5.4 条件标志

该指令不会改变标志。

25.4.7.5.5 示例

```
ISB  ; 指令同步屏障
```

25.4.7.6 MRS

将特别寄存器的内容移动到通用寄存器中。

25.4.7.6.1 语法

MRS *Rd*, *spec_reg*

其中：

Rd 是通用目标寄存器。

spec_reg 是其中一个特别寄存器：APSR、IPSR、EPSR、IEPSR、IAPSR、EAPSR、PSR、 MSP、 PSP、 PRIMASK 或 CONTROL。

25.4.7.6.2 操作

MRS 将特别寄存器的内容存放到通用寄存器中。MRS 指令可以结合 MR 指令来产生读 - 修改 - 写序列，这适用于在 PSR 中修改特别标志。

参见第 25 – 25.4.7.7 节。

25.4.7.6.3 限制

在该指令中， *Rd* 必须不能是 SP 或 PC。

25.4.7.6.4 条件标志

该指令不会改变标志。

25.4.7.6.5 示例

```
MRS  R0, PRIMASK ; 读取 PRIMASK 值并将其写入 R0
```

25.4.7.7 MSR

将通用寄存器的内容传移到指定的特别寄存器中。

25.4.7.7.1 语法

MSR *spec_reg*, *Rn*

其中：

Rn 是通用源寄存器。

spec_reg 是特别目的寄存器：APSR、IPSR、EPSR、IEPSR、IAPSR、EAPSR、PSR、MSP、 PSP、 PRIMASK 或 CONTROL。

25.4.7.7.2 操作

MSR 使用 *Rn* 所指定的寄存器的值来更新其中一个特别寄存器。

参见第 25 – 25.4.7.6 节。

25.4.7.7.3 限制

在该指令中， *Rn* 必须不能是 SP，且不能是 PC。

25.4.7.7.4 条件标志

该指令明确地根据 *Rn* 中的值来更新标志。

25.4.7.7.5 示例

```
MSR  CONTROL, R1 ; 读取 R1 的值，并将其写入 CONTROL 寄存器
```

25.4.7.8 NOP

无操作。

25.4.7.8.1 语法

NOP

25.4.7.8.2 操作

NOP 执行的是无操作，且不能保证会占用指令时间。处理器可在它到达执行阶段之前将其从管道中移除。

使用 NOP 指令来进行填充，例如，在 64 位边界上放置后续指令。

25.4.7.8.3 限制

该指令没有限制。

25.4.7.8.4 条件标志

该指令不会改变标志。

25.4.7.8.5 示例

```
NOP  ; 无操作
```

25.4.7.9 SEV

注：SEV 指令不能在 LPC11Axx 上使用。

发送事件。

25.4.7.9.1 语法

SEV

25.4.7.9.2 操作

SEV将事件信号发送到一个多处理器系统内的所有处理器中。它还可设置局部事件寄存器，见第 25 – 25.3.5 节。

另请参见第 25 – 25.4.7.11 节。

25.4.7.9.3 限制

该指令没有限制。

25.4.7.9.4 条件标志

该指令不会改变标志。

25.4.7.9.5 示例

```
SEV ; 发送事件
```

25.4.7.10 SVC

超级用户调用。

25.4.7.10.1 语法

SVC *#imm*

其中：

imm 是 0-255 范围内的整数。

25.4.7.10.2 操作

SVC 指令会引发 SVC 异常。

处理器会忽略 *imm*。如果有需要，可以通过异常处理程序获取 imm 来决定要请求什么样的服务程序。

25.4.7.10.3 限制

该指令没有限制。

25.4.7.10.4 条件标志

该指令不会改变标志。

25.4.7.10.5 示例

```
SVC  #0x32 ; 超级用户调用（SVC 处理程度使用协议栈的 PC 来锁定立即数的位置，
           ; 然后将其提取出来）
```

25.4.7.11 WFE

等待事件。

注：WFE 指令不能在 LPC11Axx 上使用。

25.4.7.11.1 语法

WFE

25.4.7.11.2 操作

如果事件寄存器为 0，则 WFE 挂起执行，直至发生下列其中的一个事件：

- 出现异常，除非异常屏蔽寄存器或当前优先级级别将其屏蔽
- 异常进入挂起状态，如果系统控制寄存器的 SEVONPEND 置位
- 存在调试进入请求，如果调试使能的话
- 外设或多处理器系统里另一个处理器通过使用 SEV 指令来发出事件信号。

如果事件寄存器为 1，则 WFE 将其清除为 0 并立即完成操作。

更多信息请见第 25 – 25.3.5 节。

注：WFE 的目的只是用于节约功率。当写软件时，假定 WFE 作为 NOP 运行。

25.4.7.11.3 限制

该指令没有限制。

25.4.7.11.4 条件标志

该指令不会改变标志。

25.4.7.11.5 示例

```
WFE  ; 等待事件
```

25.4.7.12 WFI

等待中断。

25.4.7.12.1 语法

WFI

25.4.7.12.2 操作

WFI

挂起执行，直至发生下列其中的一个事件：

- 异常
- 中断变为挂起状态，如果 PRIMASK 被清除，则该中断占用优先权
- 存在调试进入请求，无论调试是否使能。

注：WFI 的目的只是用于节约功率。当写软件时，假定 WFI 作为 NOP 运行。

25.4.7.12.3 限制

该指令没有限制。

25.4.7.12.4 条件标志

该指令不会改变标志。

25.4.7.12.5 示例

```
WFI ; 等待中断
```

25.5 外设

25.5.1 关于 ARM Cortex-M0

专用外设总线 (PPB) 的地址映射为：

表 287. 内核外设寄存器区

地址	内核外设	描述
0xE000E008-0xE000E00F	系统控制块	表 25 – 296
0xE000E010-0xE000E01F	系统定时器	表 25 – 305
0xE000E100-0xE000E4EF	可嵌套中断向量控制器	表 25 – 288
0xE000ED00-0xE000ED3F	系统控制块	表 25 – 296
0xE000EF00-0xE000EF03	可嵌套中断向量控制器	表 25 – 288

在寄存器描述中，寄存器的**类型**有以下几种：

RW — 可读和可写。
RO — 只读。
WO — 只写。

25.5.2 可嵌套中断向量控制器

本节描述**可嵌套中断向量控制器** (NVIC) 以及它使用的寄存器。 NVIC 支持：

- 32 个中断。
- 每个中断的优先级可编程为0-3四种级别。级别越高对应的优先级越低。因此，级别0是最高的中断优先级。
- 中断信号的电平和脉冲检测。
- 中断末尾连锁。
- 一个外部**不可屏蔽中断** (NMI)。

处理器在异常进入时自动使它的状态入栈，在异常退出时自动使它的状态出栈，无需采用任何指令。这就实现了低延迟的异常处理。 NVIC 的硬件寄存器有：

表 288. NVIC 寄存器汇总

地址	名称	类型	复位值	描述
0xE000E100	ISER	RW	0x00000000	第 25 – 25.5.2.2 节
0xE000E180	ICER	RW	0x00000000	第 25 – 25.5.2.3 节
0xE000E200	ISPR	RW	0x00000000	第 25 – 25.5.2.4 节
0xE000E280	ICPR	RW	0x00000000	第 25 – 25.5.2.5 节
0xE000E400-0xE000E41C	IPR0-7	RW	0x00000000	第 25 – 25.5.2.6 节

25.5.2.1 使用 CMSIS 访问 Cortex-M0 NVIC 寄存器

CMSIS 函数允许在不同的 Cortex-M 系列处理器之间进行软件移植。

当利用 CMSIS 来访问 NVIC 寄存器时要用到以下函数：

表 289. 访问 NVIC 的 CMSIS 函数

CMSIS 函数	描述
void NVIC_EnableIRQ(IRQn_Type IRQn)[1]	使能一个中断或异常。
void NVIC_DisableIRQ(IRQn_Type IRQn)[1]	禁用一个中断或异常。
void NVIC_SetPendingIRQ(IRQn_Type IRQn)[1]	将中断或异常的挂起状态设为 1。
void NVIC_ClearPendingIRQ(IRQn_Type IRQn)[1]	将中断或异常的挂起状态清零。
uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)[1]	读取中断或异常的挂起状态。 如果挂起状态被设为 1，这个函数就返回非零值。
void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)[1]	将一个优先级可配置的中断或异常的优先级设置为级别 1。
uint32_t NVIC_GetPriority(IRQn_Type IRQn)[1]	读取一个优先级可配置的中断或异常的优先级。这个函数返回当前的优先级级别。

[1] 输入参数 IRQn 是 IRQ 编号，更多信息请见表 275。

25.5.2.2 中断设置 - 使能寄存器

ISER 使能中断，并显示哪些中断被使能。有关寄存器属性请见表 288 中的寄存器汇总。

根据表 37 中的 IRQ 编号进行位的分配。

表 290. ISER 的位分配

位	名称	函数
[31:0]	SETENA	中断设置 - 使能位。 写入： 0 = 无影响 1 = 使能中断。 读： 0 = 中断被禁用 1 = 中断被使能。

如果一个挂起中断被使能， NVIC 就根据它的优先级来激活该中断。如果一个中断未被使能，使该中断的中断信号有效可将中断的状态变成挂起，但是，不管这个中断的优先级如何， NVIC 都不会激活该中断。

25.5.2.3 中断清除 - 使能寄存器

ICER禁用中断，并显示哪些中断被使能。有关寄存器属性请见表25 – 288中的寄存器汇总。

根据表 37 中的 IRQ 编号进行位的分配。

表 291. ICER 的位分配

位	名称	函数
[31:0]	CLRENA	中断清除 - 使能位。 写入： 0 = 无影响 1 = 禁用中断。 读： 0 = 中断被禁用 1 = 中断被使能。

25.5.2.4 中断设置 - 挂起寄存器

ISPR 强制中断进入挂起状态，并显示哪些中断正在挂起。有关寄存器属性请见表 25 – 288 中的寄存器汇总。

根据表 37 中的 IRQ 编号进行位的分配。

表 292. ISPR 的位分配

位	名称	函数
[31:0]	SETPEND	中断设置 - 挂起位。 写入： 0 = 无影响 1 = 将中断状态变为挂起。 读： 0 = 中断没有挂起 1 = 中断正在挂起。

注：向 ISPR 位写 1 相当于下面两种情况：

- 正在挂起的中断不会有任何影响
- 被禁用的中断会将中断的状态设置成挂起。

25.5.2.5 中断清除 - 挂起寄存器

ICPR 使中断离开挂起状态，并显示哪些中断正在挂起。有关寄存器属性请见表 25 – 288 中的寄存器汇总。

根据表 37 中的 IRQ 编号进行位的分配。

表 293. ICPR 的位分配

位	名称	函数
[31:0]	CLRPEND	中断清除 - 挂起位。 写入： 0 = 无影响 1 = 清除中断的挂起状态。 读： 0 = 中断没有挂起 1 = 中断正在挂起。

注：向 ICPR 位写 1 不影响相应中断的有效状态。

25.5.2.6 中断优先级寄存器

IPR0-IPR7 寄存器为每个中断提供了一个优先级域。这些寄存器只能字访问。关于它们的属性请见表 25 – 288 中的寄存器汇总。每个寄存器包含 4 个优先级域，如下所示，并且根据表 37 中的 IRQ 编号进行排列。

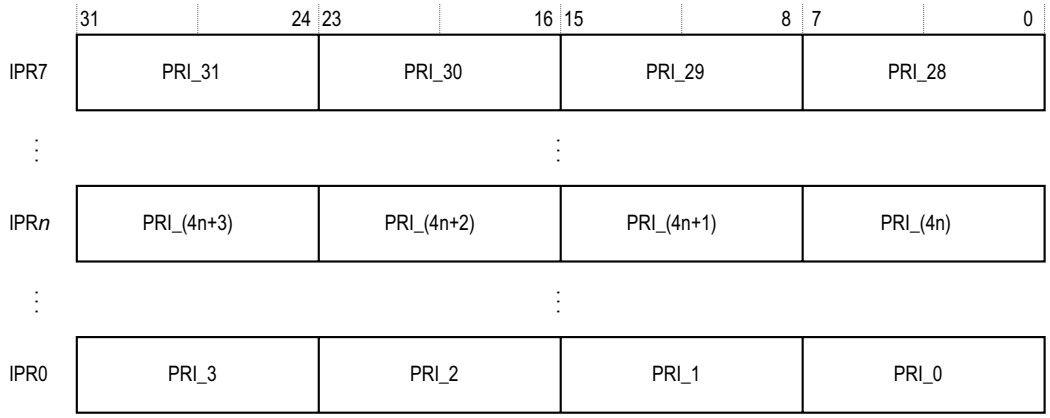


图 75. IPR 寄存器

表 294. IPR 的位分配

位	名称	函数
[31:24]	优先级，字节偏移量 3	每个优先级域在位 [7:6] 中保持优先级值 0-3。值越小，对应中断的优先级越高。每个域的位 [5:0] 读作 0，在写入时被忽略。
[23:16]	优先级，字节偏移量 2	
[15:8]	优先级，字节偏移量 1	
[7:0]	优先级，字节偏移量 0	

有关中断优先级阵列（提供了中断优先级的软件视角）访问的更多信息，请见第 25 – 25.5.2.1 节。

使用下面的方法为中断 **M** 找出 IPR 编号和字节偏移量：

- 相应的 IPR 编号 **N**，通过等式 **N** = **M** DIV 4 得出
- 这个寄存器中所需优先级域的字节偏移量是 **M** MOD 4 （M 除以 4 取余），在这里：
  - 字节偏移量 0 指的是寄存器位 [7:0]
  - 字节偏移量 1 指的是寄存器位 [15:8]

- 字节偏移量 2 指的是寄存器位 [23:16]
- 字节偏移量 3 指的是寄存器位 [31:24]。

25.5.2.7 电平有效的中断和脉冲中断

处理器支持电平有效的中断和脉冲中断。脉冲中断也被描述成边沿触发的中断。

一个电平有效的中断一直保持有效，直至外设将中断信号撤销。通常，发生这种情况的原因是 ISR 访问外设导致外设将中断请求清除。脉冲中断是在处理器时钟的上升沿同时采样得到的一个中断信号。为了确保 NVIC 检测到中断，外设必须使中断信号至少在一个时钟周期内保持有效，在这段时间内 NVIC 检测脉冲并锁存中断。

当处理器进入 ISR 时，它自动消除中断的挂起状态，见第 25.5.2.7.1 节。对于一个电平有效的中断，如果在处理器从 ISR 返回之前中断信号未被撤销，中断就再次变成挂起，处理器必须再次执行 ISR。这就表示，外设可以一直使中断信号保持有效，直到它不再需要服务为止。

25.5.2.7.1 中断的硬件和软件控制

Cortex-M0 锁存所有的中断。外设中断会由于下面的其中一个原因而变为挂起：

- NVIC 检测到中断信号有效，而相应的中断无效
- NVIC 检测到中断信号的一个上升沿
- 软件向相应的中断设置 - 挂起寄存器位写入值，见第 25 – 25.5.2.4 节。

挂起的中断一直保持挂起，直到出现以下其中一种情况：

- 处理器进入中断的 ISR。这就使中断的状态从挂起变为有效。而且：
  - 对于电平有效的中断，当处理器从 ISR 返回时，NVIC 采样中断信号。如果中断信号有效，中断的状态变回挂起，这可能使得处理器立刻再次进入 ISR。否则，中断的状态变为无效。
  - 对于脉冲中断，NVIC 继续监测中断信号，如果这个中断信号一直处于脉冲状态，中断的状态就变成挂起和有效。在这种情况下，当处理器从 ISR 返回时，中断的状态变为挂起，这可能使得处理器立刻重新进入 ISR。

    如果当处理器在处理 ISR 时中断信号的脉冲就不存在了，那么，当处理器从 ISR 返回时中断的状态变为无效。
- 软件向相应的中断清除 - 挂起寄存器位写入值。

  对于电平有效的中断，如果中断信号仍然有效，中断的状态不改变。否则，中断的状态变为无效。

  对于脉冲中断，中断的状态变为：
  - 无效 （如果中断之前的状态是挂起）
  - 有效 （如果中断之前的状态是有效和挂起）。

25.5.2.8 NVIC 的使用提示和技巧

保证软件正确使用对齐的寄存器访问。处理器不支持非对齐的 NVIC 寄存器访问。

中断即使被禁用也可以进入挂起状态。禁用一个中断只阻止处理器处理中断。

25.5.2.8.1 NVIC 编程提示

软件使用 CPSIE i 和指令来使能和禁用中断。 CMSIS 为这些指令提供了以下内在函数：

```
void __disable_irq(void) // 禁用中断

void __enable_irq(void) // 使能中断
```

另外， CMSIS 提供了许多 NVIC 控制函数，包括：

表 295. CMSIS 的 NVIC 控制函数

CMSIS 中断控制函数	描述
void NVIC_EnableIRQ(IRQn_t IRQn)	使能 IRQn
void NVIC_DisableIRQ(IRQn_t IRQn)	禁用 IRQn
uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn)	如果 IRQn 正在挂起，返回真 (1)
void NVIC_SetPendingIRQ (IRQn_t IRQn)	设置 IRQn 挂起状态
void NVIC_ClearPendingIRQ (IRQn_t IRQn)	清除 IRQn 挂起状态
void NVIC_SetPriority (IRQn_t IRQn, uint32_t priority)	设置 IRQn 的优先级
uint32_t NVIC_GetPriority (IRQn_t IRQn)	读取 IRQn 的优先级
void NVIC_SystemReset (void)	复位系统

输入参数 IRQn 是 IRQ 编号，更多信息请见表 25 – 275。有关这些函数的更多信息，请见 CMSIS 的资料。

25.5.3 系统控制块

系统控制块 (SCB) 提供了系统执行信息和系统控制，包括配置、控制和系统异常的报告。 SCB 寄存器有：

表 296. SCB 寄存器汇总

地址	名称	类型	复位值	描述
0xE000ED00	CPUID	RO	0x410CC200	第 25.5.3.2 节
0xE000ED04	ICSR	RW[1]	0x00000000	第 25 – 25.5.3.3 节
0xE000ED0C	AIRCR	RW[1]	0xFA050000	第 25 – 25.5.3.4 节
0xE000ED10	SCR	RW	0x00000000	第 25 – 25.5.3.5 节
0xE000ED14	CCR	RO	0x00000204	第 25 – 25.5.3.6 节
0xE000ED1C	SHPR2	RW	0x00000000	第 25 – 25.5.3.7.1 节
0xE000ED20	SHPR3	RW	0x00000000	第 25 – 25.5.3.7.2 节

[1] 更多信息请见寄存器描述。

25.5.3.1 Cortex-M0 SCB 寄存器的 CMSIS 映射

为了提高软件效率， CMSIS 简化了 SCB 寄存器的表现形式。在 CMSIS 中，阵列 SHP[1] 对应寄存器 SHPR2-SHPR3。

25.5.3.2 CPUID 寄存器

CPUID 寄存器包含处理器的部件号、版本和实现信息。有关其属性，请见中的寄存器汇总。寄存器的位分配如下所示：

表 297. CPUID 寄存器的位分配

位	名称	函数
[31:24]	Implementer	实现代码： 0x41 = ARM
[23:20]	Variant	更新编号，产品版本标识符 r**n**p**n** 中 r 的值： 0x0 = 版本 0
[19:16]	Constant	定义处理器架构的常量；读取的结果是： 0xC = ARMv6-M 架构
[15:4]	Partno	处理器的部件号： 0xC20 = Cortex-M0
[3:0]	Revision	修订编号，产品版本标识符 r**n**p**n** 中 p 的值： 0x0 = Patch 0

25.5.3.3 中断控制和状态寄存器

ICSR：

- 提供了：
  - 为**不可屏蔽中断** (NMI) 异常提供了一个设置 - 挂起位
  - 为 PendSV 和 SysTick 异常提供了设置 - 挂起位和清除 - 挂起位
- 指明了：
  - 正在处理的异常的异常编号
  - 是否有被抢占的有效异常
  - 最高优先级挂起异常的异常编号
  - 是否有任何异常正在挂起。

有关 ICSR 的属性请见表 25 – 296 中的寄存器汇总。寄存器的位分配如下所示：

表 298. ICSR 的位分配

位	名称	类型	函数
[31]	NMIPENDSET	RW	NMI 设置 - 挂起位。 写入： 0 = 无影响 1 = 将 NMI 异常的状态变为挂起。 读： 0 = NMI 异常未挂起 1 = NMI 异常正在挂起。 由于 NMI 是优先级最高的异常，因此，一般情况下，处理器一旦检测到向该位写 1 就立刻进入 NMI 异常处理程序。处理器进入处理程序后将该位清零。这就表示，只有当 NMI 信号在处理器正在执行 NMI 异常处理程序的过程中再次有效，通过异常处理程序读取这个位才返回 1。
[30:29]	-	-	保留。
[28]	PENDSVSET	RW	PendSV 设置 - 挂起位。 写入： 0 = 无影响 1 = 将 PendSV 异常的状态变为挂起。 读： 0 = PendSV 异常未挂起 1 = PendSV 异常正在挂起。 向该位写 1 是将 PendSV 异常状态设为挂起的唯一方法。
[27]	PENDSVCLR	WO	PendSV 清除 - 挂起位。 写入： 0 = 无影响 1 = 撤销 PendSV 异常的挂起状态。
[26]	PENDSTSET	RW	SysTick 异常设置 - 挂起位。 写入： 0 = 无影响 1 = 将 SysTick 异常的状态变为挂起。 读： 0 = SysTick 异常未挂起 1 = SysTick 异常正在挂起。
[25]	PENDSTCLR	WO	SysTick 异常清除 - 挂起位。 写入： 0 = 无影响 1 = 撤销 SysTick 异常的挂起状态。 该位只可写。当对这个寄存器执行读操作时，该位读出的值不可知。
[24:23]	-	-	保留。
[22]	ISRPENDING	RO	除 NMI 和故障之外的中断的挂起标志： 0 = 中断未挂起 1 = 中断正在挂起。
[21:18]	-	-	保留。

表 298. ICSR 的位分配 *（续）*

位	名称	类型	函数
[17:12]	VECTPENDING	RO	指示优先级最高的、正在挂起的并且使能的异常的异常编号： 0 = 没有正在挂起的异常 非零 = 优先级最高的、正在挂起的并且使能的异常的异常编号。
[11:6]	-	-	保留。
[5:0]	VECTACTIVE[1]	RO	包含有效的异常编号： 0 = 线程模式 非零 = 当前有效异常的异常编号 [1]。 **注：**这个值减去 16 得到 CMSIS IRQ 编号，编号标识出对应在中断清除 - 使能、设置 - 使能、清除 - 挂起、设置 - 挂起以及优先级寄存器中的位，请见表 25 – 270。

[1] 这个值与 IPSR 位 [5:0] 的值相同，请见表 25 – 270。

写 ICSR 时，如果执行下列操作，结果将不可知：

- 写 1 到 PENDSVSET 位和写 1 到 PENDSVCLR 位
- 写 1 到 PENDSTSET 位和写 1 到 PENDSTCLR 位。

25.5.3.4 应用中断 / 复位控制寄存器

AIRCR 提供了数据访问的字节顺序状态和系统的复位控制信息。有关其属性，请见表 25 – 296 和表 25 – 299 中的寄存器汇总。

如果要写这个寄存器，必须先向 VECTKEY 域写入 0x05FA，否则，处理器会将写操作忽略。

寄存器的位分配如下所示：

表 299. AIRCR 的位分配

位	名称	类型	函数
[31:16]	读：保留 写入：VECTKEY	RW	寄存器码： 读出的值不可知 执行写操作时将 0x05FA 写入 VECTKEY，否则写操作被忽略。
[15]	ENDIANESS	RO	采用的数据字节存储顺序： 0 = 小端 1 = 大端。
[14:3]	-	-	保留
[2]	SYSRESETREQ	WO	系统复位请求： 0 = 无影响 1 = 请求一个系统级复位。 这个位读出为 0。
[1]	VECTCLRACTIVE	WO	保留供调试使用。这个位读出为 0。当写这个寄存器时，必须向这个位写 0，否则操作将不可预知。
[0]	-	-	保留。

25.5.3.5 系统控制寄存器

SCR 控制着低功耗状态的进入和退出特性。有关其属性，请见表 25 – 296 中的寄存器汇总。寄存器的位分配如下所示：

表 300. SCR 的位分配

位	名称	函数
[31:5]	-	保留。
[4]	SEVONPEND	挂起时发送事件位： 0 = 只有使能的中断或事件能够唤醒处理器，不接受禁用中断的唤醒 1 = 使能的事件和包括禁用中断在内的所有中断都能唤醒处理器。 当一个事件或中断进入挂起状态时，事件信号将处理器从 WFE 唤醒。如果处理器并未在等待一个事件，事件被记录，影响下个 WFE。 处理器也可以在执行 SEV 指令时唤醒。
[3]	-	保留。
[2]	SLEEPDEEP	控制处理器是将睡眠模式还是深度睡眠模式作为低功耗模式（深度睡眠模式在 LPC11Axx 上不受支持）： 0 = 睡眠 1 = 深度睡眠 （在 LPC11Axx 上不受支持；切勿使用该设置）
[1]	SLEEPONEXIT	指示当从处理程序模式返回到线程模式时 sleep-on-exit（退出时进入睡眠）： 0 = 处理器返回到线程模式时不进入睡眠。 1 = 处理器从 ISR 返回到线程模式时进入睡眠或深度睡眠 （深度睡眠在 LPC11Axx 上不受支持）。 将该位设为 1 允许一个中断驱动的应用程序避免返回到一个空的主应用程序。
[0]	-	保留。

25.5.3.6 配置和控制寄存器

CCR 是一个只读寄存器，指出了 Cortex-M0 处理器行为的一些情况。有关 CCR 属性，请见表 25 – 296 中的寄存器汇总。

寄存器的位分配如下所示：

表 301. CCR 的位分配

位	名称	函数
[31:10]	-	保留。
[9]	STKALIGN	该位读出总是为 0，指示进入异常时协议栈按 8 字节对齐。 进入异常时，处理器使用入栈的 PSR 的位 [9] 来指示栈对齐。从异常中返回时，处理器使用这个入栈的位来恢复正确的栈对齐。
[8:4]	-	保留。
[3]	UNALIGN_TRP	该位读出总是为 0，指示所有未对齐的访问产生一个 HardFault。
[2:0]	-	保留。

25.5.3.7 系统处理程序优先级寄存器

SHPR2-SHPR3 寄存器设置优先级可配置的异常处理程序的优先级级别 (0-3)。

SHPR2-SHPR3 是字可访问的。关于它们的属性请见表 25 – 296 中的寄存器汇总。

利用 CMSIS 访问系统异常的优先级级别要用到以下 CMSIS 函数：

- `uint32_t NVIC_GetPriority(IRQn_Type IRQn)`
- `void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)`

输入参数 `IRQn` 是 IRQ 编号，更多信息请见表 25 – 275。

系统故障处理程序、优先级域以及每个处理程序的寄存器如下所示：

表 302. 系统故障处理程序优先级域

处理程序	域	寄存器描述
SVCall	PRI_11	第 25 – 25.5.3.7.1 节
PendSV	PRI_14	第 25 – 25.5.3.7.2 节
SysTick	PRI_15	

每个PRI_N域为8位宽，但处理器只使用每个域的位[7:6]，位[5:0]读出为0，写操作被忽略。

25.5.3.7.1 系统处理程序优先级寄存器 2

寄存器的位分配如下所示：

表 303. SHPR2 寄存器的位分配

位	名称	函数
[31:24]	PRI_11	系统处理程序 11 (SVCall) 的优先级
[23:0]	-	保留

25.5.3.7.2 系统处理程序优先级寄存器 3

寄存器的位分配如下所示：

表 304. SHPR3 寄存器的位分配

位	名称	函数
[31:24]	PRI_15	系统处理程序 15 （SysTick 异常）的优先级
[23:16]	PRI_14	系统处理程序 14 (PendSV) 的优先级
[15:0]	-	保留

25.5.3.8 SCB 使用的提示和技巧

保证软件使用对齐的 32 位字大小传输来访问所有的 SCB 寄存器。

25.5.4 系统定时器， SysTick

当系统定时器被使能时，定时器从重新载入值开始递减计数到零，下个时钟周期再将 SYST_RVR 的值重新加载到定时器（一个回合），然后在后面的时钟周期下继续开始递减计数。向 SYST_RVR 写零会使计数器在下个回合禁用。当计数器跳变到零时， COUNTFLAG 状态位被设为 1。读 SYST_CSR 将 COUNTFLAG 位清零。

写 SYST_CVR 会将该寄存器和 COUNTFLAG 状态位都清零。这个写操作不触发 SysTick 异常逻辑。读取 SYST_CVR 寄存器返回的是读取操作执行当下的寄存器值。

注：当寄存器由于调试而被终止时，计数器不递减计数。

系统定时器寄存器有：

表 305. 系统定时器寄存器汇总

地址	名称	类型	复位值	描述
0xE000E010	SYST_CSR	RW	0x00000000	第 25.5.4.1 节
0xE000E014	SYST_RVR	RW	不可知	第 25 – 25.5.4.2 节
0xE000E018	SYST_CVR	RW	不可知	第 25 – 25.5.4.3 节
0xE000E01C	SYST_CALIB	RO	0xC0000000 [1]	第 25 – 25.5.4.4 节

[1] SysTick 的校准值。

25.5.4.1 SysTick 控制和状态寄存器

SYST_CSR 使能 SysTick 特性。有关其属性，请见中的寄存器汇总。寄存器的位分配如下所示：

表 306. SYST_CSR 的位分配

位	名称	函数
[31:17]	-	保留。
[16]	COUNTFLAG	如果自从上次读这个寄存器之后定时器计数到 0，该位就返回 1。
[15:3]	-	保留。
[2]	CLKSOURCE	选择 SysTick 定时器的时钟源： 0 = 外部基准时钟 1 = 处理器时钟。
[1]	TICKINT	使能 SysTick 异常请求： 0 = 计数到零不提交 SysTick 异常请求 1 = 计数到零提交 SysTick 异常请求。
[0]	ENABLE	使能计数器： 0 = 计数器被禁用 1 = 计数器被使能。

25.5.4.2 SysTick 重新载入值寄存器

SYST_RVR 设定了加载到 SYST_CVR 的起始值。有关其属性，请见表 25 – 305 中的寄存器汇总。寄存器的位分配如下所示：

表 307. SYST_RVR 的位分配

位	名称	函数
[31:24]	-	保留。
[23:0]	RELOAD	当计数器被使能且计数值到达 0 时加载到 SYST_CVR 的值，请见第 25.5.4.2.1 节。

25.5.4.2.1 计算 RELOAD 值

RELOAD 值可以是 0x00000001-0x00FFFFFF 范围内的任何值。您可以将 RELOAD 的值设为 0，这不会产生任何影响，因为计数值从 1 变为 0 时 SysTick 异常请求和 COUNTFLAG 都被激活了。

如果要产生一个周期为 N 个处理器时钟周期的多次触发定时器，就可以将 RELOAD 值设为 N-1。例如，如果要求每隔 100 个时钟脉冲就触发一次 SysTick 中断，可将 RELOAD 设为 99。

25.5.4.3 SysTick 当前值寄存器

SYST_CVR 包含 SysTick 计数器的当前值。有关其属性，请见表 25 – 305 中的寄存器汇总。寄存器的位分配如下所示：

表 308. SYST_CVR 的位分配

位	名称	函数
[31:24]	-	保留。
[23:0]	CURRENT	读取时返回 SysTick 计数器的当前值。 向这个域写入任何值都会将该域清零，还会清零 SYST_CSR 的 COUNTFLAG 位。

25.5.4.4 SysTick 校准值寄存器

SYST_CALIB 寄存器指明了 SysTick 的校准特性。有关其属性，请见表 25 – 305 中的寄存器汇总。寄存器的位分配如下所示：

表 309. SYST_CALIB 寄存器的位分配

位	名称	函数
[31]	NOREF	该位读出为 0。该位指明不提供独立的基准时钟。
[30]	SKEW	该位读出为 0。由于 TENMS 不可知，因此，10ms 不精确计时的校准值不能确定。这会影响 SysTick 作为软件实时时钟的适用性。
[29:24]	-	保留。
[23:0]	TENMS	读取为零。该域指明校准值不可知。

如果校准信息不可知，就通过处理器时钟或外部时钟的频率来计算所需的校准值。

25.5.4.5 SysTick 的使用提示和技巧

利用中断控制器时钟来更新 SysTick 计数器。如果该时钟信号因低功耗模式而停止，则 SysTick 计数器会停止。

确保软件使用字访问来访问 SysTick 寄存器。

如果在复位时没有定义 SysTick 计数器的重新载入值和当前值，正确的 SysTick 计数器初始化序列如下：

1. 设置重新载入值。
2. 清除当前值。
3. 设置控制和状态寄存器。

25.6 Cortex-M0 指令汇总

表 310. Cortex-M0 指令汇总

操作	描述	汇编程序	周期
传输	8 位立即数	MOVS Rd, #<imm>	1
	Lo 至 Lo	MOVS Rd, Rm	1
	任意至任意	MOV Rd, Rm	1
	任意至 PC	MOV PC, Rm	3
加法	3 位立即数	ADDS Rd, Rn, #<imm>	1
	全部寄存器 Lo	ADDS Rd, Rn, Rm	1
	任意至任意	ADD Rd, Rd, Rm	1
	任意至 PC	ADD PC, PC, Rm	3
加法	8 位立即数	ADDS Rd, Rd, #<imm>	1
	带进位	ADCS Rd, Rd, Rm	1
	立即数到 SP	ADD SP, SP, #<imm>	1
	来自 SP 的格式地址	ADD Rd, SP, #<imm>	1
	来自 PC 的格式地址	ADR Rd, <label>	1
减法	Lo 和 Lo	SUBS Rd, Rn, Rm	1
	3 位立即数	SUBS Rd, Rn, #<imm>	1
	8 位立即数	SUBS Rd, Rd, #<imm>	1
	带进位	SBCS Rd, Rd, Rm	1
	SP 中的立即数	SUB SP, SP, #<imm>	1
	求补	RSBS Rd, Rn, #0	1
乘法	乘法	MULS Rd, Rm, Rd	1
比较	比较	CMP Rn, Rm	1
	负数	CMN Rn, Rm	1
	立即数	CMP Rn, #<imm>	1
逻辑	AND	ANDS Rd, Rd, Rm	1
	异或	EORS Rd, Rd, Rm	1
	OR	ORRS Rd, Rd, Rm	1
	位清除	BICS Rd, Rd, Rm	1
	取反传输	MVNS Rd, Rm	1
	与测试	TST Rn, Rm	1
移位	逻辑左移立即数	LSLS Rd, Rm, #<shift>	1
	逻辑左移寄存器	LSLS Rd, Rd, Rs	1
	逻辑右移立即数	LSRS Rd, Rm, #<shift>	1
	逻辑右移寄存器	LSRS Rd, Rd, Rs	1
	算术右移	ASRS Rd, Rm, #<shift>	1
	算术右移寄存器	ASRS Rd, Rd, Rs	1
循环	循环右移寄存器	RORS Rd, Rd, Rs	1

表 310. Cortex-M0 指令汇总 *（续）*

操作	描述	汇编程序	周期
负载	字，立即数偏移量	LDR Rd, [Rn, #<imm>]	2
	半字，立即数偏移量	LDRH Rd, [Rn, #<imm>]	2
	字节，立即数偏移量	LDRB Rd, [Rn, #<imm>]	2
	字，寄存器偏移量	LDR Rd, [Rn, Rm]	2
	半字，寄存器偏移量	LDRH Rd, [Rn, Rm]	2
	带符号半字，寄存器偏移量	LDRSH Rd, [Rn, Rm]	2
	字节，寄存器偏移量	LDRB Rd, [Rn, Rm]	2
	带符号字节，寄存器偏移量	LDRSB Rd, [Rn, Rm]	2
	相对 PC	LDR Rd, <label>	2
	相对 SP	LDR Rd, [SP, #<imm>]	2
	多重，不包括基数	LDM Rn!, {<loreglist>}	1 + N[1]
	多重，包括基数	LDM Rn, {<loreglist>}	1 + N[1]
存储	字，立即数偏移量	STR Rd, [Rn, #<imm>]	2
存储	半字，立即数偏移量	STRH Rd, [Rn, #<imm>]	2
	字节，立即数偏移量	STRB Rd, [Rn, #<imm>]	2
	字，寄存器偏移量	STR Rd, [Rn, Rm]	2
	半字，寄存器偏移量	STRH Rd, [Rn, Rm]	2
	字节，寄存器偏移量	STRB Rd, [Rn, Rm]	2
	相对 SP	STR Rd, [SP, #<imm>]	2
	多重	STM Rn!, {<loreglist>}	1 + N[1]
压栈	压栈	PUSH {<loreglist>}	1 + N[1]
	用链路寄存器压栈	PUSH {<loreglist>, LR}	1 + N[1]
出栈	出栈	POP {<loreglist>}	1 + N[1]
	出栈和返回	POP {<loreglist>, PC}	4 + N[2]
分支	条件性	B<cc> <label>	1 或 3[3]
	非条件性	B <label>	3
	带链路	BL <label>	4
	带交换	BX Rm	3
	带链路和交换	BLX Rm	3
延长	带符号半字到字	SXTH Rd, Rm	1
	带符号字节到字	SXTB Rd, Rm	1
	不带符号半字	UXTH Rd, Rm	1
	不带符号字节	UXTB Rd, Rm	1
反向	字中的字节	REV Rd, Rm	1
	两个半字中的字节	REV16 Rd, Rm	1
	带符号的低位半字	REVSH Rd, Rm	1
状态变化	超级用户调用	SVC <imm>	-[4]
	禁用中断	CPSID i	1
	使能中断	CPSIE i	1
	对特殊寄存器进行读操作	MRS Rd, <specreg>	4
	对特殊寄存器进行写操作	MSR <specreg>, Rn	4

表 310. Cortex-M0 指令汇总 *（续）*

操作	描述	汇编程序	周期
提示	发送事件	SEV	1
	等待中断	WFI	2[5]
	产生	YIELD[6]	1
	无操作	NOP	1
屏障	指令同步	ISB	4
	数据存储器	DMB	4
	数据同步	DSB	4

[1] N 是元素数量。

[2] N 是包括 PC 在内的出栈列表中的元素数量，假定加载或存储不会产生 HardFault 异常。

[3] 若采用，为 3，若不采用，为 1。

[4] 周期计数取决于内核和调试配置。

[5] 不包括等待中断或事件所花费的时间。

[6] 作为 NOP 运行。

26.1 缩略词

表 311. 缩略词

首字母缩略词	描述
ADC	模数转换器
AHB	高级高性能总线
AMBA	高级微控制器总线架构
APB	高级外设总线
BOD	掉电检测
DAC	数模转换器
GPIO	通用输入 / 输出
I^2C 或 IIC	集成电路间的总线
I/O 或 IO	输入 / 输出
IAP	在应用编程
ISP	在系统编程
ISR	中断服务例程
PIO	并行输入 / 输出
PLL	锁相环
POR	上电复位
PWM	脉冲宽度调制
SPI	串行外设接口
SSI	串行同步接口
TTL	晶体管 - 晶体管逻辑
UART	通用异步收发器

26.2 法律信息

26.2.1 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。

26.2.2 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。恩智浦半导体不对本文中非源自恩智浦半导体的信息内容负责。

在任何情况下，对于任何间接、意外、惩罚性、特殊或衍生性损害（包括但不限于利润损失、积蓄损失、业务中断、因拆卸或更换任何产品而产生的开支或返工费用），无论此等损害是否基于侵权行为（包括过失）、担保、违约或任何其他法理，恩智浦半导体均不承担任何责任。

对于因任何原因给客户带来的任何损害，恩智浦半导体对本文所述产品的总计责任和累积责任仅限于恩智浦*商业销售条款和条件*所规定的范围。

修改权利—恩智浦半导体保留对本文所发布的信息（包括但不限于规格和产品说明）随时进行修改的权利，恕不另行通知。本文件将取代并替换之前就此提供的所有信息。

适宜使用—恩智浦半导体产品并非设计、授权或担保适合用于生命保障、生命关键或安全关键系统或设备，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。恩智浦半导体及其供应商对在此类设备或应用中加入和 / 或使用恩智浦半导体产品不承担任何责任，客户需自行承担因加入和 / 或使用恩智浦半导体产品而带来的风险。

应用—本文件所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主管部门批准。

26.2.3 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。

I²C — 是恩智浦的商标。

26.3 表

26.4 图

26.5 内容

第 1 章：简介信息

第 2 章：LPC11Axx 存储器映射

第 3 章：LPC11Axx 系统配置

第 4 章：LPC11Axx 中断控制器

第 5 章：LPC11Axx 通用 I/O (GPIO)

第 6 章：LPC11Axx I/O 配置

第 7 章：LPC11Axx 引脚配置

第 8 章：LPC11Axx 通用同步 / 异步接收器 / 发送器 (USART)

第 9 章：LPC11Axx I2C 总线接口

第 10 章：LPC11Axx SPI/SSP

第 11 章：LPC11Axx 16 位计数器 / 定时器 (CT16B0/1)

第 12 章：LPC11Axx 32 位计数器 / 定时器 (CT32B0/1)

第 13 章：LPC11Axx 系统节拍定时器

第 14 章：LPC11Axx 看门狗定时器 (WDT)

第 15 章：LPC11Axx 温度传感器

第 16 章：LPC11Axx 模数转换器 (ADC)

第 17 章：LPC11Axx 数模转换器 (DAC)

第 18 章：LPC11Axx 模拟比较器

第 19 章：LPC11Axx 内部基准电压

第 20 章：LPC11Axx 闪存 /EEPROM 编程固件

第 21 章：LPC11Axx 整数除法程序

第 22 章：LPC11Axx I2C 总线驱动程序

第 23 章：LPC11Axx 功率模型

第 24 章：LPC11Axx 串行线调试

第 25 章：附录：LPC11Axx ARM Cortex-M0 参考

第 26 章：补充信息

续 >>

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 3 月 22 日
文档号：UM10527