



UM10398

LPC111x/LPC11Cxx 用户手册

Rev. 5 — 2011 年 6 月 21 日

用户手册

文档信息

Info	Content
Keywords	ARM Cortex-M0, LPC1111, LPC1112, LPC1113, LPC1114, LPC11C12, LPC11C14, LPC1100, LPC1100L, LPC11C00, LPC11C22, LPC11C24
Abstract	LPC111x/LPC11Cxx 用户手册

This translated version is for reference only, and the English version shall prevail in case of any discrepancy between the translated and English versions.

版权所有 2012恩智浦有限公司 未经许可，禁止转载



修改记录

版本	日期	描述
5	20110621	LPC111x/LPC11C1x/LPC11C2x 用户手册
更新:		<ul style="list-style-type: none"> 如下更新仅仅适用于 LPC111x/102/202/302 系列的用户手册: <ul style="list-style-type: none"> 使用 IRC 进入深度掉电模式, 见 Section 3.9.4.2。 允许 UART 串行时钟, 见 Section 10.2。 更新了 IOCON 寄存器的 Chapter 7。 增加了 Chapter 17 “LPC111x/LPC11Cxx Windowed WatchDog Timer (WDT)”。 内部上拉电阻的上拉水平见 Section 7.1。 更新了对 WDEN 位的描述, 见 Table 247 和 Table 255 (WDMOD 寄存器)。 更新了 GPIO 章节 Table 109。
4	20110304	
更新:		<ul style="list-style-type: none"> 更新了 LPC111FHN33/102 和 LPC11FHN33/201 设备 ID, 见表 43 和表 289。 增加了 3.7 章节的启动动作。 更新了图 6 的 功耗配置指着结构。
3	20110114	LPC111x/LPC11C1x/LPC11C2x 用户手册
更新:		<ul style="list-style-type: none"> 增加了 LPC11C22 和 LPC11C24 。 增加了片上 CAN 驱动器的描述。 更新了对系统滴答定时器的描述 (18 章) 和表 351。 更新了第 12 章。 指定了单周期硬件乘法, 见表 355。
2	20101102	LPC111x/LPC11C1x 用户手册
更新:		<ul style="list-style-type: none"> 增加了 LPC111x/102/202/302 (LPC1100L 系列) (表 1, 表 280)。 更新了 PLL 输出频率 (< 100 MHz), 见 3.10 节 “系统 PLL 功能描述”。 更新了对深度睡眠模式和深度掉电模式的描述, 见 3.8 节 “电源管理”。 增加了第五章 “LPC111x/LPC11Cxx 功耗配置”。 WDT 更改为 24 位定时器, 见 17 章 “LPC111x/LPC11Cxx 看门狗定时器 (WDT)”。 21.6.2 章节的 “调试连接”。 SYSTCKCAL 寄存器的地址改为 0x4004 8154 (表 6 和表 33)。 添加了比较 flash 映像的注解 (20.10 章节的 “flash 映像的注解”)。 在 20.5 部分 – “UART ISP 命令” 中, ISP 的波特率被严格限制 (230400 bps 除外)。 Flash 写操作的干扰影响 (20.5.7 节的 “复制 RAM 到 flash<Flash address> <RAM address> <no of bytes> (UART ISP)。 时钟和电源控制部分被删除了, 并结合了每章部分基本配置内容。
1	20100721	LPC111x/LPC11C1x 用户手册

Contact information

For more information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: salesaddresses@nxp.com

1.1 简介

基于 ARM Cortex-M0 的 LPC111x/LPC11Cxx 系列微控制器是低功耗，32 位微控制器家族中的一员，面向 8、16 位微处理应用，具有高性能，低功耗，简单指令集，统一编址寻址等优点，而且，相对于现在市场上存在的 8/16 位架构来说，它有效的降低了代码长度。

LPC111x/LPC11Cxx 系列微控制器的工作频率可高达 50MHz。

LPC111x/LPC11Cxx 系列微控制器加入的外围组件包括：高达 32KB 的 flash 存储器，8KB 的数据存储器，一个增强快速模式（FM+）I2C 接口，一个 RS-485/EIA-485 标准的通用异步串行收发器，两个具有 SSP 特性的 SPI 接口，四个通用定时器，一个 10 位 ADC 和 42 个 GPIO 引脚。

片上 C_CAN 驱动器和闪存的系统编程工具通过 C_CAN 连接在 LPC11Cxx 里，此外 LPC11C2x 还包含一个片上 CAN 收发器。

备注：此用户手册适涵盖 LPC111x/LPC11Cxx，包括 LPC1100 系列（LPC111x/101/201/301），LPC1100L 系列（LPC111x/102/202/302），和 LPC11C00 系列（LPC11C1x/301 和 LPC11C2x/301）。LPC1100L 包含功耗配置，LPC11C00 包含 C_CAN 控制器和片上 CAN 驱动器。见 [Table 1](#)。

Table 1. LPC111x/LPC11Cxx 增强功能

系列	特征预览
LPC1100 系列	<ul style="list-style-type: none"> • I2C、SSP、UART、GPIO • 计数器和看门狗计数器 • 10-bit ADC • Flash/SRAM 存储器 • 详细信息见 Section 1.2.
LPC1100L 系列	在 LPC1100 系列的基础上加上一下特征： <ul style="list-style-type: none"> • 在运行和睡眠模式下具有更低的功耗 • 内部上拉电阻阻止引脚达到 V_{DD} 的电压上限 • GPIO 引脚的可编程伪开漏模式 • WWDTC 具有时钟源锁定功能
LPC11C00 系列	在 LPC1100 系列的基础上增加一下功能： <ul style="list-style-type: none"> • CAN 控制器 • 片上 CAN 驱动器 • 片上 CAN 发送器 (LPC11C2x) • WDT (非可视化) 具有锁定时钟源的功能

1.2 特征

- 系统：
 - ARM Cortex-M0 处理器，工作频率高达 50MHz 的。
 - ARM Cortex-M0 内嵌向量中断控制器（NVIC）。
 - 串行线调试。
 - 系统滴答定时器。
- 存储器：
 - 32 kB (LPC1114/LPC11C14)，24 kB (LPC1113)，16 kB (LPC1112/LPC11C12)，或者 8 kB (LPC1111) 片上可编程存储器。
 - 8 kB，4 kB，或者 2 kB 的 SRAM。
 - 通过片上引导（bootloader）软件实现现场编程（ISP）和在线编程（IAP）。
- 数字外设：
 - 多达 42 个带有可配置上拉 / 下拉电阻的 GPIO 引脚，在 LPC11C22/C24 中 GPIO 引脚的数目有所减少是为了更好地包装。
 - GPIO 引脚可以用作边沿和电平触发。
 - 每个引脚有大电流（20mA）驱动输出。
 - 两个 I2C 总线引脚在增强快速模式时，为大电流灌入驱动（20mA）。
 - 四个通用定时器 / 计数器（共 4 个捕获输入和 13 个比较输出）。
 - 可编程的看门狗定时器（WDT）。
- 模拟外设：
 - 8 通道 10 位 ADC。
- 串行接口：
 - 有分数波特率发生器，内部 FIFO，支持 RS-485 总线和 modem 控制的 UART。
 - 最多可有两个具有 SSP 特性的、带 FIFO 的 SPI 控制器，具有很好的多协议的兼容性，（第二个 SPI 只在 LQFP48 和 PLCC44 封装中存在）。
 - I2C 总线接口支持全速 I2C 总线规格和增强快速模式（速率可达到 1Mbit/s，具有多地址 识别监听模式）。
 - C_CAN 控制器（仅仅支持 LPC11Cxx）。片上 CAN 和 CAN 驱动器。
 - 片上告诉 CAN 发送器（仅仅支持 LPC11C22/C24）。
- 时钟产生：
 - 12 MHz 12MHz 内部 RC 振荡器精度误差不超过 1%，可以选择作为系统时钟使用。
 - 晶体振荡器的工作范围为 1MHz ~ 25MHz。
 - 频率范围在 7.8 kHz 和 1.8 MHz 之间的可编程的看门狗振荡器。
 - PLL 允许 CPU 达到最高工作频率而不需要外部高频振荡器。可以使用主振荡器和内部 RC 振荡器。

- 带分频器的时钟输出功能可以反映主振荡器时钟 IRC 时钟、CPU 时钟和看门狗时钟的状态。
- 电源控制：
 - 集成 PMU（电源管理单元），自动调整其内部的电压调节器，以最小化睡眠、深度睡眠和深度掉电模式期间的功耗。
 - 在引导 ROM 中的功耗配置允许最大限度地减少任何给定应用程序的功耗，此电源配置可以通过一个简单的函数来调用（仅仅适用 LPC111x/102/202/302 only）。
 - 三种低功耗模式：睡眠，深度睡眠，深度掉电。
 - 多达 13 个功能引脚可通过一个专门的启动逻辑来将处理器从深度睡眠模式中唤醒。
 - 上电复位（POR）。
 - 掉点检测有四个独立的中断和强迫复位阈值。
- 具有用于识别的唯一设备序列号。
- 单一 3.3V 供电（1.8V ~ 3.6V）。
- 可以采用 LQFP48，PLCC44 和 HVQFN33 封装。

1.3 分类信息

Table 2. 分类信息

类型编号	封装		
	名称	描述	版本
LPC1111FHN33/101	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装，无引脚，33 个接线端子，尺寸 7 x 7 x 0.85 mm	n/a
LPC1111FHN33/102	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装，无引脚，33 个接线端子，尺寸：7 x 7 x 0.85 mm	n/a
LPC1111FHN33/201	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装，无引脚，33 个接线端子，尺寸：7 x 7 x 0.85 mm	n/a
LPC1111FHN33/202	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装，无引脚，33 个接线端子，尺寸：7 x 7 x 0.85 mm	n/a
LPC1112FHN33/101	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装，无引脚，33 个接线端子，尺寸：7 x 7 x 0.85 mm	n/a
LPC1112FHN33/102	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装，无引脚，33 个接线端子，尺寸：7 x 7 x 0.85 mm	n/a
LPC1112FHN33/201	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装，无引脚，33 个接线端子，尺寸：7 x 7 x 0.85 mm	n/a
LPC1112FHN33/202	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装，无引脚，33 个接线端子，尺寸：7 x 7 x 0.85 mm	n/a
LPC1113FHN33/201	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装，无引脚，33 个接线端子，尺寸：7 x 7 x 0.85 mm	n/a
LPC1113FHN33/202	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装，无引脚，33 个接线端子，尺寸：7 x 7 x 0.85 mm	n/a
LPC1113FHN33/301	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装，无引脚，33 个接线端子，尺寸：7 x 7 x 0.85 mm	n/a

Table 2. 分类信息 ...continued

类型编号	封装		
	名称	描述	版本
LPC1113FHN33/302	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装, 无引脚, 33 个接线端子, 尺寸: 7 x 7 x 0.85 mm	n/a
LPC1114FHN33/201	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装, 无引脚, 33 个接线端子, 尺寸: 7 x 7 x 0.85 mm	n/a
LPC1114FHN33/202	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装, 无引脚, 33 个接线端子, 尺寸: 7 x 7 x 0.85 mm	n/a
LPC1114FHN33/301	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装, 无引脚, 33 个接线端子, 尺寸: 7 x 7 x 0.85 mm	n/a
LPC1114FHN33/302	HVQFN33	HVQFN: 塑封高耐热型超薄四侧引脚扁平封装, 无引脚, 33 个接线端子, 尺寸: 7 x 7 x 0.85 mm	n/a
LPC1113FBD48/301	LQFP48	LQFP48: 塑封薄小型方块平面封装, 48 个引脚, 尺寸: 7 x 7 x 1.4 mm	sot313-2
LPC1113FBD48/302	LQFP48	LQFP48: 塑封薄小型方块平面封装, 48 个引脚, 尺寸: 7 x 7 x 1.4 mm	sot313-2
LPC1114FBD48/301	LQFP48	LQFP48: 塑封薄小型方块平面封装, 48 个引脚, 尺寸: 7 x 7 x 1.4 mm	sot313-2
LPC1114FBD48/302	LQFP48	LQFP48: 塑封薄小型方块平面封装, 48 个引脚, 尺寸: 7 x 7 x 1.4 mm	sot313-2
LPC1114FA44/301	PLCC44	PLCC44: 塑封有引线芯片载体; 44 个引脚	sot187-2
LPC1114FA44/302	PLCC44	PLCC44: 塑封有引线芯片载体; 44 个引脚	sot187-2
LPC11C12FBD48/301	LQFP48	LQFP48: 塑封薄小型方块平面封装, 48 个引脚, 尺寸: 7 x 7 x 1.4 mm	sot313-2
LPC11C14FBD48/301	LQFP48	LQFP48: 塑封薄小型方块平面封装, 48 个引脚, 尺寸: 7 x 7 x 1.4 mm	sot313-2
LPC11C22FBD48/301	LQFP48	LQFP48: 塑封薄小型方块平面封装, 48 个引脚, 尺寸: 7 x 7 x 1.4 mm	sot313-2
LPC11C24FBD48/301	LQFP48	LQFP48: 塑封薄小型方块平面封装, 48 个引脚, 尺寸: 7 x 7 x 1.4 mm	sot313-2

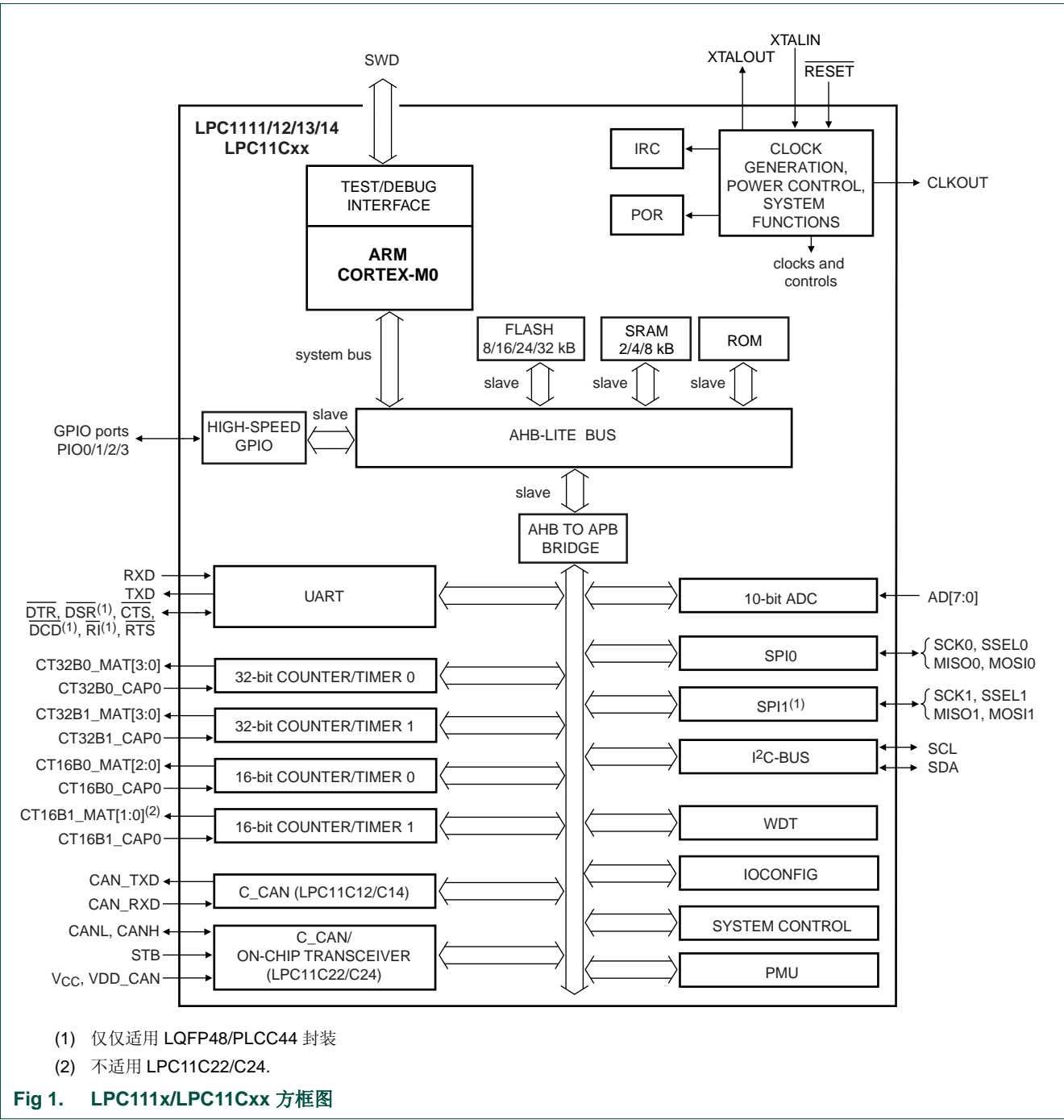
Table 3. 分类选项

类型编号	系列	Flash	全部 SRAM	UART RS-485	I ² C/ Fm+	SPI	C_CAN	电源配 置	ADC 通道 数	封装
LPC1111										
LPC1111FHN33/101	LPC1100	8 kB	2 kB	1	1	1	-	no	8	HVQFN33
LPC1111FHN33/102	LPC1100L	8 kB	2 kB	1	1	1	-	yes	8	HVQFN33
LPC1111FHN33/201	LPC1100	8 kB	4 kB	1	1	1	-	no	8	HVQFN33
LPC1111FHN33/202	LPC1100L	8 kB	4 kB	1	1	1	-	yes	8	HVQFN33
LPC1112										
LPC1112FHN33/101	LPC1100	16 kB	2 kB	1	1	1	-	no	8	HVQFN33
LPC1112FHN33/102	LPC1100L	16 kB	2 kB	1	1	1	-	yes	8	HVQFN33
LPC1112FHN33/201	LPC1100	16 kB	4 kB	1	1	1	-	no	8	HVQFN33
LPC1112FHN33/202	LPC1100L	16 kB	4 kB	1	1	1	-	yes	8	HVQFN33

Table 3. 分类选项 ...continued

类型编号	系列	Flash	全部 SRAM	UART RS-485	I ² C/ Fm+	SPI	C_CAN	电源配 置	ADC 通道 数	封装
LPC1113										
LPC1113FHN33/201	LPC1100	24 kB	4 kB	1	1	1	-	no	8	HVQFN33
LPC1113FHN33/202	LPC1100L	24 kB	4 kB	1	1	1	-	yes	8	HVQFN33
LPC1113FHN33/301	LPC1100	24 kB	8 kB	1	1	1	-	no	8	HVQFN33
LPC1113FHN33/302	LPC1100L	24 kB	8 kB	1	1	1	-	yes	8	HVQFN33
LPC1113FBD48/301	LPC1100	24 kB	8 kB	1	1	2	-	no	8	LQFP48
LPC1113FBD48/302	LPC1100L	24 kB	8 kB	1	1	2	-	yes	8	LQFP48
LPC1114										
LPC1114FHN33/201	LPC1100	32 kB	4 kB	1	1	1	-	no	8	HVQFN33
LPC1114FHN33/202	LPC1100L	32 kB	4 kB	1	1	1	-	yes	8	HVQFN33
LPC1114FHN33/301	LPC1100	32 kB	8 kB	1	1	1	-	no	8	HVQFN33
LPC1114FHN33/302	LPC1100L	32 kB	8 kB	1	1	1	-	yes	8	HVQFN33
LPC1114FBD48/301	LPC1100	32 kB	8 kB	1	1	2	-	no	8	LQFP48
LPC1114FBD48/302	LPC1100L	32 kB	8 kB	1	1	2	-	yes	8	LQFP48
LPC1114FA44/301	LPC1100	32 kB	8 kB	1	1	2	-	no	8	PLCC44
LPC1114FA44/302	LPC1100L	32 kB	8 kB	1	1	2	-	yes	8	PLCC44
LPC11C12/LPC11C14										
LPC11C12FBD48/301	LPC11C00	16 kB	8 kB	1	1	2	1	no	8	LQFP48
LPC11C14FBD48/301	LPC11C00	32 kB	8 kB	1	1	2	1	no	8	LQFP48
LPC11C22/LPC11C24 内置片上快速 CAN 收发器										
LPC11C22FBD48/301	LPC11C00	16 kB	8 kB	1	1	2	1	no	8	LQFP48
LPC11C24FBD48/301	LPC11C00	32 kB	8 kB	1	1	2	1	no	8	LQFP48

1.4 方框图



1.5 ARM Cortex-M0 处理器

ARM Cortex-M0 处理器的详细描述请查看 [Section 23.2 “About the Cortex-M0 processor and core peripherals”](#)，LPC111x/LPC11Cxx ARM Cortex-M0 的处理器内核配置如下：

- 系统选项：
 - 包括内嵌向量中断控制器（NVIC），支持多达 32 个中断。
 - 包括系统嘀嗒定时器。
- 调试选项： 串行线调试（SWD）可设置两个观察点和四个断点。

2.1 如何阅读本章

[Table 4](#) 和 [Table 5](#) 描述了不同型号的 LPC111x/LPC11Cxx 处理器的存储器的配置。

Table 4. LPC111x 存储器配置

编号 前缀	Flash	SRAM /101; /102	/201; /202	/301; /302
LPC1111	8 kB	2 kB	4 KB	-
LPC1112	16 kB	2 kB	4 KB	-
LPC1113	24 kB	-	4 KB	8 kB
LPC1114	32 kB	-	4 KB	8 kB

Table 5. LPC11Cxx 存储器配置

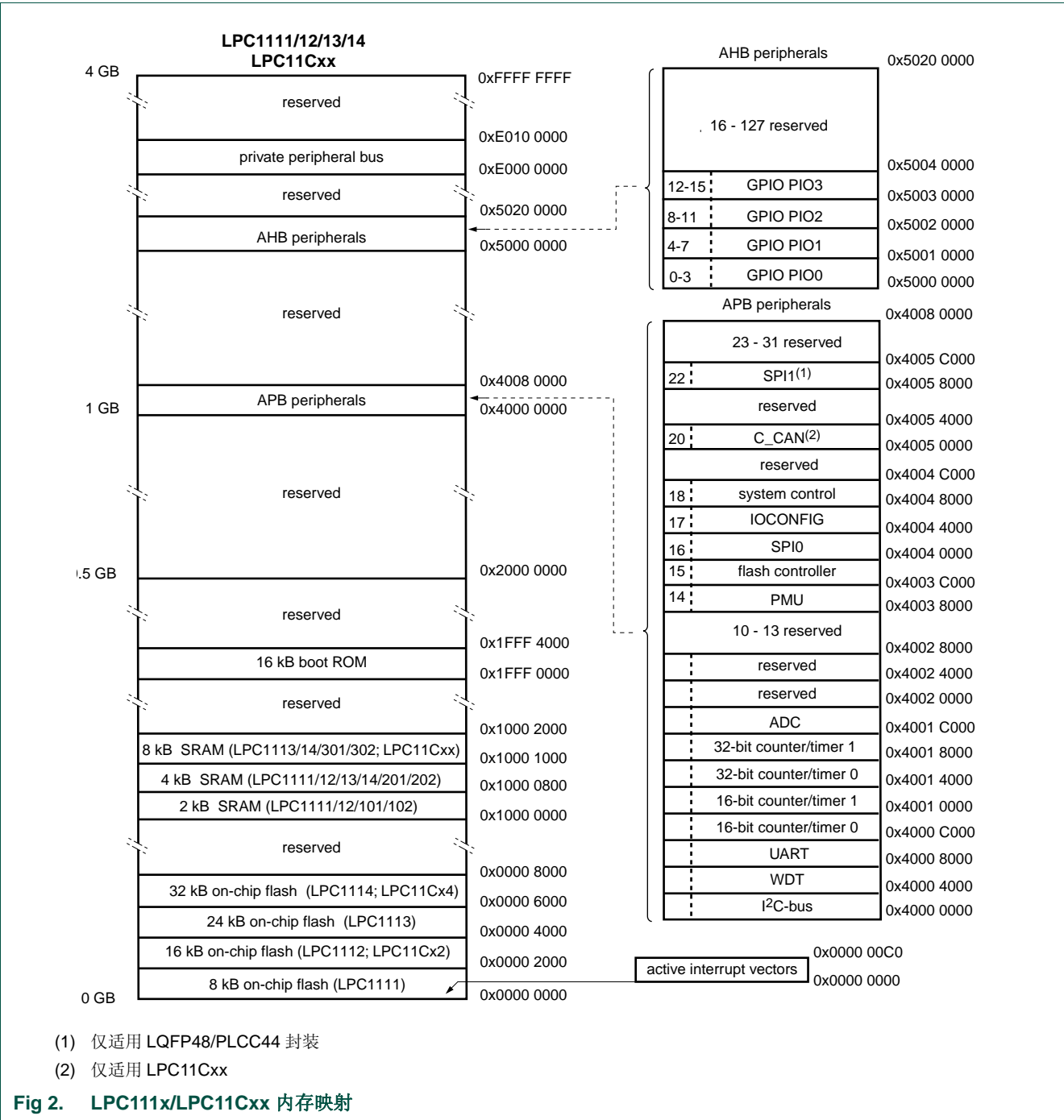
编号	Flash	SRAM
LPC11C12/301	16 kB	8 kB
LPC11C14/301	32 kB	8 kB
LPC11C22/301	16 kB	8 kB
LPC11C24/301	32 kB	8 kB

2.2 内存映射

[Figure 2](#) 描述了 LPC111x/LPC11Cxx 存储器和外设的地址空间。

AHB 外设区域大小为 2 MB，可以被划分成最多支持 128 个外设使用。在 LPC111x/LPC11Cxx 上，GPIO 端口是唯一的 AHB 外设。APB 外设区域大小为 512KB，可以被划分成最多支持 32 个外设划分使用。任意一种类型的外设实际都分配 16KB 的空间，以简化每个外设的地址解码。

所有的外设寄存器地址，无论大小，都是 32 位字对齐。这表示访问字寄存器和半字寄存器都需要一个周期。例如，无法单独读写一个字的高字节。



3.1 如何阅读本章

系统配置模块的一些功能取决于特定的产品系列号。

C_CAN 控制器

C_CAN 的位于 SYSAHBCLKCTRL 寄存器 ([Table 21](#)) 里的时钟控制位 17 和在 PRESETCTRL 寄存器 ([Table 9](#)) 中的复位控制位 3 仅仅在 LPC11Cxx/101/201/301 中起作用。。

进入深度睡眠模式

在进入深度睡眠模式之前 IRC 的状态 (见 [Section 3.9.4.2](#)):

- IRC 必须被使能, 适用的产品系列是 LPC111x/101/201/301 和 LPC11Cxx/101/201/301
- IRC 的状态没有影响, 适用的产品系列是 LPC111x/102/202/302。

使能 UART 时钟序列

使能 UART 外围时钟的要求:

- 对于 LPC111X/101/201/301, 在 UART 时钟通过设置 SYSAHBCLKCTRL 寄存器 ([Table 21](#)) 被使能之前, UART 引脚必须在 IOCON 模块中被配置。
- UART 引脚的配置顺序和 UART 时钟对于 LPC111x/102/202/302 和 LPC11Cxx/101/201/301 没有影响。

3.2 概述

系统配置模块控制 LPC111x/LPC11Cxx 的振荡器、启动逻辑和时钟发生器。同时, 该模块还包含一些设置 AHB 优先级的寄存器和一个重映射 flash、SRAM 和 ROM 存储器区域的寄存器。

3.3 引脚描述

[Table 6](#) 描述了与系统配置模块相关的引脚

Table 6. 引脚描述

引脚名称	引脚方向	引脚描述
CLKOUT	输出	时钟输出引脚
PIO0_0 to PIO0_11	输入	通过 PIO0 启动逻辑唤醒
PIO1_0	输入	通过 PIO1 启动逻辑唤醒

3.4 时钟产生

[Figure 3](#) 展示了 LPC111x/LPC11Cxx 的时钟产生单元 (CGU)。

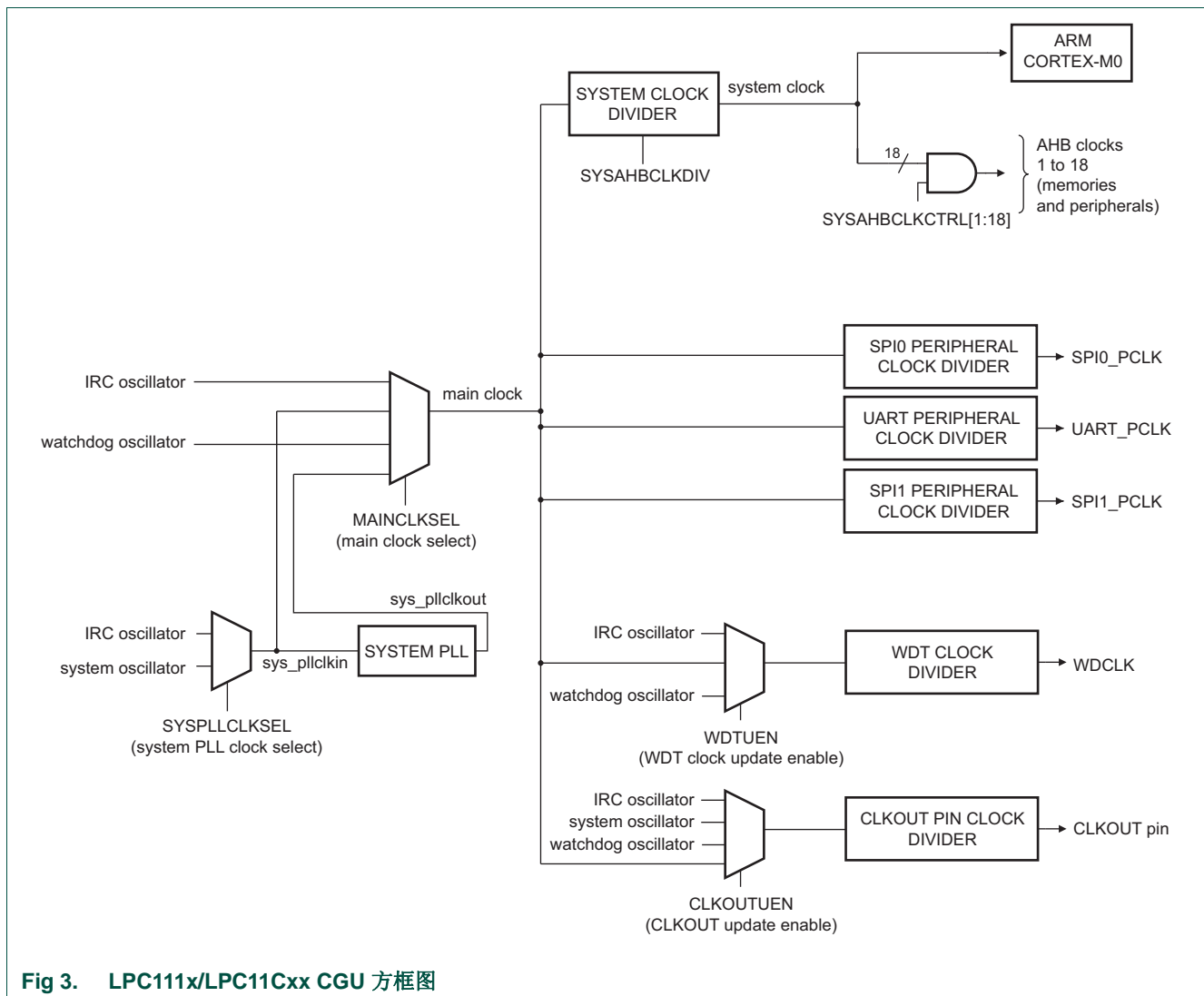
LPC111x/LPC11Cxx 包含三个独立的振荡器。分别是系统振荡器、内部 RC 振荡器 (IRC) 和看门狗振荡器。在具体应用中，每一个振荡器都可以有不止一个用途。

复位之后，LPC111x/LPC11Cxx 会在内部 RC 振荡器下工作，直到通过软件进行切换。这就使得系统 bootloader 工作在一个已知的频率下而不会受任何外部晶振的影响。

SYSAHBCLKCTRL 寄存器用于分配系统时钟给不同的外部设备和存储器。UART，WDT 和 SPI0/1 都有自己的时钟分频器，用以从系统时钟得到自己的时钟。

主时钟、IRC 输出时钟、系统振荡器和看门狗振荡器可以直接从 CLKOUT 引脚观察到。

电源控制的详细信息见 [Section 3.9](#)。



3.5 寄存器描述

所有的寄存器，不管大小，都占有一个字的地址宽度。各个寄存器的详细信息见各寄存器的功能描述。

flash 访问定时寄存器 [Section 3.12](#) 可以被重置为系统设置的一部分，但该寄存器并不属于系统配置模块。

Table 7. 寄存器概览：系统控制模块 (基址 0x4004 8000)

名称	访问类型	偏移地址	描述	复位值	参考
SYSMEMREMAP	R/W	0x000	系统内存重映射	0x002	Table 8
PRESETCTRL	R/W	0x004	外设复位控制	0x000	Table 9
SYSPLLCTRL	R/W	0x008	系统 PLL 控制	0x000	Table 10
SYSPLLSTAT	R	0x00C	系统 PLL 状态	0x000	Table 11
-	-	0x010 - 0x01C	保留	-	-
SYSOSCCTRL	R/W	0x020	系统振荡器控制	0x000	Table 12
WDTOSCCTRL	R/W	0x024	看门狗振荡器控制	0x000	Table 13
IRCCTRL	R/W	0x028	IRC 控制	0x080	Table 14
-	-	0x02C	保留	-	-
SYSRSTSTAT	R	0x030	系统复位状态寄存器	0x000	Table 15
-	-	0x034 - 0x03C	保留	-	-
SYSPLLCLKSEL	R/W	0x040	系统 PLL 时钟源选择	0x000	Table 16
SYSPLLCLKUEN	R/W	0x044	系统 PLL 时钟源更新允许	0x000	Table 17
-	-	0x048 - 0x06C	保留	-	-
MAINCLKSEL	R/W	0x070	主时钟源选择	0x000	Table 18
MAINCLKUEN	R/W	0x074	主时钟源更新允许	0x000	Table 19
SysAHBCLKDIV	R/W	0x078	系统 AHB 时钟分频器	0x001	Table 20
-	-	0x07C	保留	-	-
SysAHBCLKCTRL	R/W	0x080	系统 AHB 时钟控制	0x85F	Table 21
-	-	0x084 - 0x090	保留	-	-
SSP0CLKDIV	R/W	0x094	SPI0 时钟分频器	0x000	Table 22
UARTCLKDIV	R/W	0x098	UART 时钟分频器	0x000	Table 23
SSP1CLKDIV	R/W	0x09C	SPI1 时钟分频器	0x000	Table 24
-	-	0x0A0-0x0CC	保留	-	-
WDTCLKSEL	R/W	0x0D0	WDT 时钟源选择	0x000	Table 25
WDTCLKUEN	R/W	0x0D4	WDT 时钟源更新允许	0x000	Table 26
WDTCLKDIV	R/W	0x0D8	WDT 时钟分频器	0x000	Table 27
-	-	0x0DC	保留	-	-
CLKOUTCLKSEL	R/W	0x0E0	CLKOUT 时钟源选择	0x000	Table 28
CLKOUTUEN	R/W	0x0E4	CLKOUT 时钟源更新允许	0x000	Table 29
CLKOUTCLKDIV	R/W	0x0E8	CLKOUT 时钟分频器	0x000	Table 30
-	-	0x0EC - 0x0FC	保留	-	-
PIOPORCAP0	R	0x100	POR 捕获 PIO 状态 0	用户自定义	Table 31
PIOPORCAP1	R	0x104	POR 捕获 PIO 状态 1	用户自定义	Table 32

Table 7. 寄存器概览：系统控制模块 (基址 0x4004 8000) ...continued

名称	访问类型	偏移地址	描述	复位值	参考
-	R	0x108 - 0x14C	保留	-	-
BODCTRL	R/W	0x150	BOD 控制	0x000	Table 33
SYSTCKCAL	R/W	0x154	系统滴答计数器的校准	0x004	Table 34
-	-	0x158 - 0x1FC	保留	-	-
STARTAPRP0	R/W	0x200	启动逻辑边沿控制寄存器 0		Table 35
STARTERP0	R/W	0x204	启动逻辑信号允许寄存器 1		Table 36
STARTRSRP0CLR	W	0x208	启动逻辑复位寄存器 0	n/a	Table 37
STARTSRP0	R	0x20C	启动逻辑状态寄存器 0	n/a	Table 38
-	-	0x210 - 0x22C	保留	-	-
PDSLEEP_CFG	R/W	0x230	深度睡眠模式的电源关闭状态	0x0000 0000	Table 40
PDAWAKE_CFG	R/W	0x234	深度睡眠模式唤醒后的电源掉电状态	0x0000 EDF0	Table 41
PDRUN_CFG	R/W	0x238	掉电配置寄存器	0x0000 EDF0	Table 42
-	-	0x23C - 0x3F0	保留	-	-
DEVICE_ID	R	0x3F4	设备 ID	依赖设备	Table 43

3.5.1 系统内存重映射寄存器

系统内存重映射寄存器用于选择是从 ROM, flash, 还是 SRAM 中读取 ARM 中断向量。

Table 8. 系统内存重映射寄存器 (SYSMEMREMAP, 地址 0x4004 8000) 位描述

位	符号	值	描述	复位值
1:0	MAP		系统内存重映射	10
		0x0	Boot Loader 模式，中断向量被重映射到 Boot ROM 中。	
		0x1	User RAM 模式，中断向量被重映射到 SRAM 中。	
		0x2	User Flash 模式，中断向量没有被重映射，仍然保留在 flash 中。	
31:2	-	-	保留	0x00

3.5.2 外设复位控制寄存器

该寄存器允许软件复位 SPI 和 I2C 外设。写 0 到 SSP0/1_RST_N 或者 I2C_RST_N 位将会复位 SPI0/1 或者 I2C 外设。写 1 使复位无效。

备注：在访问 SPI 和 I2C 外设之前，要写 1 到相应位中用以确保相应外设复位无效。

Table 9. 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		SPI0 复位控制	0
		0	复位 SPI0 外设	
		1	SPI0 复位无效	
1	I2C_RST_N		I2C 复位控制	0
		0	复位 I2C 外设	
		1	I2C 复位无效	
2	SSP1_RST_N		SPI1 复位控制	0
		0	复位 SPI1 外设	
		1	SPI1 复位无效	
3	CAN_RST_N		C_CAN 复位控制。详细信息见 Section 3.1	0
		0	复位 C_CAN 外设	
		1	C_CAN 复位无效	
31:4	-	-	保留	0x00

3.5.3 系统 PLL 控制寄存器

该寄存器用于连接和允许系统 PLL 并且配置系统 PLL 的倍频器和分频器的值。PLL 可以接受来自不同时钟源的频率介于 10MHz 和 25MHz 之间的时钟输入信号。输入的时钟信号先倍频到一个较高的频率，然后再分频提供给 CPU、外设、存储器。PLL 可以产生 CPU 允许的最大时钟频率。

Table 10. 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		反馈分频器的值。分频器的值 M 是 MSEL 的值 + 1。	0x000
		00000	分频器的值 M = 1	
		11111	分频器的值 M = 32.	
6:5	PSEL		后分频器值 P。分频器的值是 2 × P.	0x00
		0x0	P = 1	
		0x1	P = 2	
		0x2	P = 4	
		0x3	P = 8	
31:7	-	-	保留，不要将 1 写入这些保留位中	0x0

3.5.4 系统 PLL 状态寄存器

这个寄存器是一个只读寄存器，它提供 PLL 的时钟锁定状态（见 [Section 3.11.1](#)）。

Table 11. 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	0x0
		0	PLL 没有被锁定	
		1	PLL 被锁定	
31:1	-	-	保留	0x00

3.5.5 系统振荡器控制寄存器

该寄存器配置系统振荡器的频率范围。

Table 12. 系统振荡器控制寄存器 (SYSOSCCTRL, 地址 0x4004 8020) 位描述

位	符号	值	描述	复位值
0	BYPASS		旁路系统振荡器	0x0
		0	系统振荡器不被旁路	
		1	旁路被允许。PLL 输入 (sys_osc_clk) 由 XTALIN 和 XTALOUT 引脚直接提供。	
1	FREQRANGE		位低功耗振荡器确定频率范围。	0x0
		0	1 - 20 MHz 频率范围	
		1	15 - 25 MHz 频率范围	
31:2	-	-	保留	0x00

3.5.6 看门狗振荡器控制寄存器

此寄存器用于配置看门狗振荡器。看门狗振荡器包含一个模拟部分和数字部分。模拟部分包括振荡器 功能和产生模拟时钟 (Fclkana)。通过数字部分将模拟时钟 (Fclkana) 分频为所需的输出时钟频率 wdt_osc_clk。模拟输出频率 (Fclkana) 可以根据 FREQSEL 位在 500 kHz 和 3.4 MHz 之间进行调整，通过数字部分使用 DIVSEL 位将 Fclkana 分频以达到 wdt_osc_clk 所设置的频率表（分频器的值 = 2, 4, ..., 64）。

看门狗振荡器的输出频率可以根据下面的公式计算出：
$$\text{wdt_osc_clk} = \text{Fclkana} / (2^{\text{DIVSEL}} (1 + \text{DIVSEL})) = 7.8 \text{ kHz to } 1.7 \text{ MHz (标准值)}。$$

备注：对 FREQSEL 位的任何设置都将产生一个列出频率值 ±40% 误差的 Fclkana。看门狗振荡器是一个功耗最低的时钟源， 如果需要精确地时钟， 要选择 IRC 或者系统振荡器。

备注：看门狗振荡器的频率在复位之后是一个未知值。在使用看门狗振荡器之前，看门狗振荡器的频率必须被编程写入 WDTOSCCTRL 寄存器。

Table 13. 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		为 Fclkana 选择分频器。 $\text{wdt_osc_clk} = \text{Fclkana} / (2 \times (1 + \text{DIVSEL}))$ 00000: $2 \times (1 + \text{DIVSEL}) = 2$ 00001: $2 \times (1 + \text{DIVSEL}) = 4$ 至 11111: $2 \times (1 + \text{DIVSEL}) = 64$	0
8:5	FREQSEL		选择看门狗振荡器模拟输出频率 (Fclkana).	0x00
		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	0x00

3.5.7 内部谐振控制寄存器

该寄存器用来调整偏上 12 MHz 振荡器，调整值是厂家预设的并且在启动时被引导写入。

Table 14. 内部谐振控制寄存器 (IRCCTRL, 地址 0x4004 8028) 位描述

位	符号	描述	复位值
7:0	TRIM	调整值	0x1000 0000, flash 将会重编程
31:9	-	保留	0x00

3.5.8 系统复位状态寄存器

如果另外一个复位信号 - 例如 EXTRST - 在 POR 信号消失之后仍然有效, 那么它的相应位将会被设置为已被检测到状态。

Table 15. 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	0x0
		0	未检测到 POR	
		1	已检测到 POR	
1	EXTRST			0x0
		0	未检测到 $\overline{\text{RESET}}$	
		1	已检测到 $\overline{\text{RESET}}$	
2	WDT		看门狗复位状态	0x0
		0	未检测到 WDT 复位	
		1	已检测到 WDT 复位	
3	BOD		掉点检测 Brown-out 复位状态	0x0
		0	未检测到 BOD	
		1	已检测到 BOD	
4	SYSRST		软件系统复位状态	0x0
		0	未检测到系统复位	
		1	已检测到系统复位	
31:5	-	-	保留	0x0

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 (见 [Section 3.5.10](#)) 必须从低切换到高, 更新方能生效。

备注: 当更改时钟源时, 在时钟源更新之前两个时钟源都要运行。

备注: 当使用 C_CAN 控制器波特率大于 100 kbit/s 时, 系统振荡器必须被选择。

Table 16. 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	0x00
		0x0	IRC 振荡器	
		0x1	系统振荡器	
		0x2	保留	
		0x3	保留	
31:2	-	-	保留	0x00

3.5.10 系统 PLL 时钟源更新允许寄存器

该寄存器在 SYSPLLCLKSEL 寄存器被写入以后使用新的输入时钟来更新系统 PLL 时钟源。为了使更新生效，需要先往 SYSPLLUEN 寄存器中先写 0 再写 1。

备注：当切换时钟源时，在时钟被更新之前两个时钟都要运行。

Table 17. 系统 PLL 时钟源更新允许寄存器 (SYSPLLCLKUEN, 地址 0x4004 8044) 位描述

位	符号	值	描述	复位值
0	ENA		允许系统 PLL 时钟源更新	0x0
		0	无变化	
		1	更新时钟源	
31:1	-	-	保留	0x00

3.5.11 主时钟源选择寄存器

该寄存器用来选择系统主时钟源，它可以是系统 PLL 的任何输入、系统 PLL 的输出 (sys_pllclkout)、看门狗 或者 IRC 振荡器。主系统时钟为内核、外设和存储器提供时钟。

.

MAINCLKUEN 寄存器（见 [Section 3.5.12](#)）必须从低切换到高才能使更新生效。

备注：当切换时钟源时，在时钟源被更新之前两个时钟都要运行。

备注：当使用 C_CAN 控制器波特率在 100 kbit/s 以上时，必须选择为系统振荡器。

Table 18. 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	0x00
		0x0	IRC 振荡器	
		0x1	系统 PLL 的输入时钟	
		0x2	WDT 振荡器	
		0x3	系统 PLL 的输出时钟	
31:2	-	-	保留	0x00

3.5.12 主时钟源更新允许寄存器

该寄存器在 MAINCLKSEL 寄存器被写入以后用新的输入时钟来更新主时钟源。为了使更新生效，需要往寄存器 MAINCLKUEN 中先写 0 再写 1。

备注：当切换时钟源时，在时钟源被更新之前两个时钟必须都在运行。

Table 19. 主时钟源更新允许寄存器 (MAINCLKUEN, 地址 0x4004 8074) 位描述

位	符号	值	描述	复位值
0	ENA		允许主时钟源更新	0x0
		0	无变化	
		1	更新时钟源	
31:1	-	-	保留	0x00

3. 5. 13 系统 AHB 时钟分频寄存器

该寄存器对主时钟进行分频，用以为内核、存储器和外设提供系统时钟。可以通过设置 DIV 位为 0 完全关闭系统时钟。

Table 20. 系统 AHB 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: 禁止系统时钟 1: 分频值为 1. 至 255: 分频值为 255.	0x01
31:8	-	保留	0x00

3. 5. 14 系统 AHB 时钟控制寄存器

AHBCLKCTRL 寄存器用于允许时钟提供给独立的系统和外设模块。系统时钟 (sys_ahb_clk[0], AHBCLKCTRL 寄存器的 0 位) 提供给 AHB 到 APB 桥, AHB 矩阵, ARM Cortex-M0 核, 系统模块和 PMU.。这个时钟不能被禁止。

Table 21. 系统 AHB 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		允许 AHB 到 APB 桥、AHB 矩阵、Cortex-M0 FCLK 和 HCLK, SysCon 和 PMU 时钟。改为只读	1
		0	保留	
		1	允许	
1	ROM		允许 ROM 的时钟	1
		0	禁止	
		1	允许	
2	RAM		允许 RAM 的时钟	1
		0	禁止	
		1	允许	
3	FLASHREG		允许 flash 寄存器接口的时钟	1
		0	禁止	
		1	允许	

Table 21. 系统 AHB 时钟控制寄存器 (SYSAHBCLKCTRL, 地址 0x4004 8080) 位描述 ...continued

位	符号	值	描述	复 位 值
4	FLASHARRAY		允许 flash 阵列存取的时钟	1
		0	禁止	
		1	允许	
5	I2C		允许 I2C 的时钟	0
		0	禁止	
		1	允许	
6	GPIO		允许 GPIO 的时钟	1
		0	禁止	
		1	允许	
7	CT16B0		允许 16 位 计数器 / 定时器 0 的时钟	0
		0	禁止	
		1	允许	
8	CT16B1		允许 16 位 计数器 / 定时器 1 的时钟	0
		0	禁止	
		1	允许	
9	CT32B0		允许 32 位 计数器 / 定时器 0 的时钟	0
		0	禁止	
		1	允许	
10	CT32B1		允许 32 位计数器 / 定时器 1 的时钟	0
		0	禁止	
		1	允许	
11	SSP0		允许 SPI0 的时钟	1
		0	禁止	
		1	允许	
12	UART		允许 UART 的时钟。详细信息见 Section 3.1	0
		0	禁止	
		1	允许	
13	ADC		允许 ADC 的时钟	0
		0	禁止	
		1	允许	
14	-		保留	0
15	WDT		允许 WDT 的时钟	0
		0	禁止	
		1	允许	
16	IOCON		允许 I/O 配置模块的时钟	0
		0	禁止	
		1	允许	

Table 21. 系统 AHB 时钟控制寄存器 (SYSAHBCLKCTRL, 地址 0x4004 8080) 位描述 ...continued

位	符号	值	描述	复位值
17	CAN		允许 C_CAN 的时钟。相信信息见 Section 3.1	0
		0	禁止	
		1	允许	
18	SSP1		允许 SPI1 的时钟	0
		0	禁止	
		1	允许	
31:19	-	-	保留	0x00

3.5.15 SPI0 时钟分频器寄存器

该寄存器配置 SPI0 的外部时钟 SPI0_PCLK。设置 DIV 位为 0x0，可以关闭 SPI0_PCLK。

Table 22. SPI0 时钟分频器寄存器 (SSP0CLKDIV, 地址 0x4004 8094) 位描述

位	符号	描述	复位值
7:0	DIV	SPI0_PCLK 时钟分频器值 0: 禁止 SPI0_PCLK 1: 分频值为 1 至 255: 分频值为 255.	0x00
31:8	-	保留	0x00

3.5.16 UART 时钟分频器寄存器

该寄存器配置 UART 外设时钟 UART_PCLK.，设置 DIV 的值为 0x0 可以关闭 UART_PCLK。

备注：在允许 UART 时钟之前一定要在 IOCON 模块中配置 UART 引脚。详细信息见 [Section 3.1](#)。

Table 23. UART 时钟分频器寄存器 (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: 禁止 UART_PCLK. 1: 分频值为 1. 至 255: 分频值为 255.	0x00
31:8	-	保留	0x00

3.5.17 SPI1 时钟分频器寄存器

此寄存器配置 SPI1 的外围时钟 SPI1_PCLK.。通过设置 DIV 位为 0x0 可以关闭 SPI1_PCLK。

Table 24. SPI1 时钟分频器寄存器 (SSP1CLKDIV, 地址 0x4004 809C) 位描述

位	符号	描述	复位值
7:0	DIV	SPI1_PCLK 时钟分频器值 0: 禁止 SPI1_PCLK. 1: 分频值为 1. 至 255: 分频值为 255.	0x00
31:8	-	保留	0x00

3.5.18 WDT 时钟源选择寄存器

该寄存器选择看门狗定时器的时钟源。WDTCLKUEN 寄存器（见 [Section 3.5.19](#)）必须从低切换到高方能使设置生效。

备注：当切换时钟源时，在时钟源被更新之前两个时钟必须都在运行。

Table 25. WDT 时钟源选择寄存器 (WDTCLKSEL, 地址 0x4004 80D0) 位描述

位	符号	值	描述	复位值
1:0	SEL		WDT 时钟源	0x00
		0x0	IRC 振荡器	
		0x1	主时钟	
		0x2	看门狗振荡器	
		0x3	保留	
31:2	-	-	保留	0x00

3.5.19 WDT 时钟源更新允许寄存器

在写 WDTCLKSEL 寄存器之后，该寄存器允许使用新的输入时钟源来更新看门狗定时器的时钟源。为了使更新有效必须先往 WDTCLKUEN 寄存器中写 0 之后再写 1

备注：当切换时钟源时，在时钟源被更新之前两个时钟都要在运行。

Table 26. WDT 时钟源更新允许 (WDTCLKUEN, 地址 0x4004 80D4) 位描述

位	符号	值	描述	复位值
0	ENA		允许 WDT 时钟源更新	0x0
		0	无变化	
		1	更新时钟源	
31:1	-	-	保留	0x00

3.5.20 WDT 时钟分频器定时器

该寄存器决定了看门狗时钟 wdt_clk 的分频器的值。

Table 27. WDT 时钟分频器寄存器 (WDTCLKDIV, 地址 0x4004 80D8) 位描述

位	符号	描述	复位值
7:0	DIV	WDT 时钟分频器值 0: 关闭 WDCLK. 1: 分频值为 1. 至 255: 分频值为 255.	0x00
31:8	-	保留	0x00

3. 5. 21 CLKOUT 时钟源选择寄存器

该寄存器配置 clkout_clk 信号并通过 CLKOUT 引脚输出。所有的振荡器和主时钟都可以被选择为 clkout_clk clock 时钟。

CLKOUTCLKUEN 寄存器（见 [Section 3. 5. 22](#)）必须从低切换到高方能使更新生效。

备注：当切换时钟源时，在时钟源被更新之前两个时钟都要在运行。

Table 28. CLKOUT 时钟源选择寄存器 (CLKOUTCLKSEL, 地址 0x4004 80E0) 位描述

位	符号	值	描述	复位值
1:0	SEL		CLKOUT 时钟源	0x00
		0x0	IRC 振荡器	
		0x1	系统振荡器	
		0x2	看门狗振荡器	
		0x3	主时钟	
31:2	-	-	保留	0x00

3. 5. 22 CLKOUT 时钟源更新允许寄存器

将值写入 CLKOUTCLKSEL 寄存器以后，该寄存器用于允许使用新的时钟来更新 CLKOUT 引脚输出的时钟源。为了使在 CLKOUT 引脚的输入更新生效，需要先往 CLKCLKUEN 寄存器中写 0，然后再写 1。

备注：当切换时钟源时，在时钟源被更新之前两个时钟都要在运行。

Table 29. CLKOUT 时钟源更新允许寄存器 (CLKOUTUEN, 地址 0x4004 80E4) 位描述

位	符号	值	描述	复位值
0	ENA		允许 CLKOUT 时钟源更新	0x0
		0	无变化	
		1	更新时钟源	
31:1	-	-	保留	0x00

3. 5. 23 CLKOUT 时钟分频器寄存器

该寄存器决定了 clkout 在 CLKOUT 引脚上的 clk 信号的分频值。。

Table 30. CLKOUT 时钟分频器寄存器 (CLKOUTCLKDIV, 地址 0x4004 80E8) 位描述

位	符号	值	描述	复位值
7:0	DIV		输出时钟分频器值 0: 关闭 CLKOUT. 1: 分频值为 1. 至 255: 分频值为 255.	0x00
31:8	-	-	保留	0x00

3.5.24 POR 捕获 PIO 状态寄存器 0

PIOPORCAP0 寄存器捕获在上电复位时端口 0、1、2（PIO2_0 到 PIO2_7 引脚）的 PIO 引脚的状态（高或低）。每个位代表一个 GPIO 引脚的复位状态。该寄存器是一个只读状态寄存器。

Table 31. 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
11:0	CAPPIO0_n	PIO0_n 原始复位输入状态： PIO0_11 至 PIO0_0	依赖用户实现
23:12	CAPPIO1_n	PIO1_n 原始复位输入状态： PIO1_11 至 PIO1_0	依赖用户实现
31:24	CAPPIO2_n	PIO2_n 原始复位输入状态： PIO2_11 至 PIO2_0	依赖用户实现

3.5.25 POR 捕获 PIO 状态寄存器 1

PIOPORCAP1 寄存器捕获在上电复位时端口 2（PIO2_8 到 PIO2_11 引脚）和端口 3 的 PIO 引脚的状态（高或低）。每个位代表一个 PIO 引脚的复位状态。该寄存器是一个只读状态寄存器。

Table 32. POR 捕获 PIO 状态寄存器 1 (PIOPORCAP1, 地址 0x4004 8104) 位描述

Bit	Symbol	Description	Reset value
0	CAPPIO2_8	PIO2_8 原始复位输入状态	依赖用户实现
1	CAPPIO2_9	PIO2_9 原始复位输入状态	依赖用户实现
2	CAPPIO2_10	PIO2_10 原始复位输入状态	依赖用户实现
3	CAPPIO2_11	PIO2_11 原始复位输入状态	依赖用户实现
4	CAPPIO3_0	PIO3_0 原始复位输入状态	依赖用户实现
5	CAPPIO3_1	PIO3_1 原始复位输入状态	依赖用户实现
6	CAPPIO3_2	PIO3_2 原始复位输入状态	依赖用户实现
7	CAPPIO3_3	PIO3_3 原始复位输入状态	依赖用户实现
8	CAPPIO3_4	PIO3_4 原始复位输入状态	依赖用户实现
9	CAPPIO3_5	PIO3_5 原始复位输入状态	依赖用户实现
31:10	-	保留	-

3. 5. 26 BOD 控制寄存器

掉电检测 (BOD) 控制寄存器选择使得掉电检测器向 NVIC 发出 BOD 中断并强制复位的四个独立阈值 in [Table 33](#) 列出的复位和中断阈值是典型值。

Table 33. BOD 控制寄存器 (BODCTRL, 地址 0x4004 8150) 位描述

位	符号	值	描述	复位值
1:0	BODRSTLEV		BOD 复位电平	00
		0x0	电平 1: 复位的有效阈值电压 1.46 V; 复位的无效阈值电压 1.63V。	
		0x1	电平 2: 复位的有效阈值电压 2.06V; 复位的无效阈值电压 2.15 V。	
		0x2	电平 3: 复位的有效阈值电压 2.35 V; 复位的无效阈值电压 2.43 V。	
		0x3	电平 4: 复位的有效阈值电压 2.63 V; 复位的无效阈值电压 2.71V。	
3:2	BODINTVAL		BOD 中断电平	00
		0x0	电平 0: 中断有效阈值电压 1.65 V; 中断无效阈值电压 1.80 V	
		0x1	电平 1: 中断有效阈值电压 2.22 V; 中断无效阈值电压 2.35 V	
		0x2	电平 2: 中断有效阈值电压 2.52 V; 中断无效阈值电压 2.66V	
		0x3	电平 3: 中断有效阈值电压 2.80 V; 中断无效阈值电压 2.90V	
4	BODRSTENA		允许 BOD 复位	0
		0	禁止复位功能	
		1	允许复位功能	
31:5	-	-	保留	0x00

3. 5. 27 系统滴答定时器校准寄存器

该寄存器决定了 SYST_CALIB 寄存器 (见 [Table 264](#)) 的值。

Table 34. 系统滴答定时器校准寄存器 (SYSTCKCAL, 地址 0x4004 8154) 位描述

位	符号	描述	复位值
25:0	CAL	系统时钟计时器校准值	0x04
31:26	-	保留	0x00

3. 5. 28 启动逻辑边沿控制寄存器 0

STARTAPRP0 寄存器控制端口 0 (PI00_0~ PI00_11) 和端口 1 (PI01_0) 的启动逻辑输入。该寄存器为对应的 PIO 输入选择一个下降沿或上升沿，来为启动逻辑 (见 [Section 3.10.2](#)) 产生一个下降或上升时钟沿。

STARTAPRP0 寄存器的每个位控制一个端口的输入，并与 NVIC 的一个唤醒中断相连接。STARTAPRP0 寄存器的第 0 位对应中断 0，第 1 位对应中断 1，依此类推（见 [Table 53](#)），共计 13 个中断。

备注：如果对应的 PIO 引脚用来将处理器从深度睡眠模式中唤醒，那么每一个连接到启动逻辑输入的中断必须在 NVIC 中被允许。

Table 35. 启动逻辑边沿控制寄存器 0 (STARTAPRP0, 地址 0x4004 8200) 位描述

位	符号	描述	复位值
11:0	APRPIO0_n	选择启动逻辑输入 PIO0_n: PIO0_11 至 PIO0_0 是上升沿还是下降沿 0 = 下降沿 1 = 上升沿	0x0
12	APRPIO1_0	选择启动逻辑输入 PIO0_n: PIO0_11 至 PIO0_0 是上升沿还是下降沿 0 = 下降沿 1 = 上升沿	0x0
31:13	-	保留	0x0

3.5.29 启动逻辑信号允许寄存器 0

STARTERP0 寄存器用于允许或禁止启动逻辑中的启动信号位。每个位的功能和 [Table 35](#) 中描述的相同。

Table 36. 启动逻辑信号允许寄存器 0 (STARTERP0, 地址 0x4004 8204) 位描述

位	符号	描述	复位值
11:0	ERPIO0_n	允许启动逻辑输入 PIO0_n: PIO0_11 至 PIO0_0 的起始信号 0 = 禁止 1 = 允许	0x0
12	ERPIO1_0	允许启动逻辑输入 PIO1_0 的起始信号 0 = 禁止 1 = 允许	0x0
31:13	-	保留	0x0

3.5.30 启动逻辑复位寄存器 0

往 STARTRSRPOCLR 寄存器中某一位写 0 将复位启动逻辑。位的功能与 [Table 35](#) 一致。启动逻辑使用输入信号产生一个记录启动信号的时钟边沿。这个时钟边沿（上升沿或下降沿）用于设置从深度睡眠模式唤醒的中断。因此，在使用之前必须清楚启动逻辑的状态。

Table 37. 启动逻辑复位寄存器 0 (STARTSRP0CLR, 地址 0x4004 8208) 位描述

位	符号	描述	复位值
11:0	RSRPIO0_n	启动逻辑输入 PIO0_n:PIO0_11 至 PIO0_0 的起始信号复位 0 = 无影响 1 = 复位起始信号	n/a
12	RSRPIO1_0	启动逻辑输入 PIO1_0 的起始信号复位 0 = 无影响 1 = 复位起始信号	n/a
31:13	-	保留	n/a

3.5.31 启动逻辑状态寄存器 0

该寄存器反映了启动信号位的允许状态。位分配见 Table 35，每个位（如果允许）反映了启动逻辑的状态，也就是被指定的引脚是否接收到唤醒信号。

Table 38. 启动逻辑状态寄存器 0 (STARTSRP0, 地址 0x4004 820C) 位描述

位	符号	描述	复位值
11:0	SRPIO0_n	启动逻辑输入 PIO0_n: PIO0_11 ~PIO0_0 的启动信号状态 0 = 未收到启动信号 1 = 启动信号被挂起	n/a
12	SRPIO1_0	启动逻辑输入 PIO1_0 的启动信号状态 0 = 未收到启动信号 1 = 启动信号被挂起	n/a
31:13	-	保留	n/a

3.5.32 深度睡眠模式配置寄存器

访寄存器的各位可以通过软件进行编程，以指示芯片在 ARM Cortex-M0 进入深度睡眠模式时要进入的状态。进入睡眠模式后 PDSLEEPCFG 寄存器的值将会自动加载到 PDRUNCFG 寄存器中。

在进入深度睡眠模式之前，该寄存器必须要进行至少一次的初始化为 Table 39 所示的四个可选值中的一个。

Table 39. PDSLEEPCFG 寄存器的可选值

配置	WD 振荡器开	WD 振荡器关
BOD 开	PDSLEEPCFG = 0x0000 18B7	PDSLEEPCFG = 0x0000 18F7
BOD 关	PDSLEEPCFG = 0x0000 18BF	PDSLEEPCFG = 0x0000 18FF

备注：初始化或者软件编程该寄存器的失败将会导致微处理器产生意想不到的结果。只有 Table 39 列出的值是 PDSLEEPCFG 寄存器所支持的值。

为了给深度掉电模式选择适当的电源配置，请阅读下页。

- BOD: 保留 BOD 循环将会在深度睡眠模式下避免一个低电压事件的产生。然而，BOD 循环会产生额外的功率消耗。
 - WD 振荡器：在深度掉电模式下如果需要为一个唤醒事件设置时间，那么看门狗定时器振荡器可以被保留用以位看门狗定时器或者通用定时器产生时钟（详细信息见 [Section 3.10.3](#)）。在这种情况下，看门狗振荡器的模拟输出频率必须设置为它的最低值 (WDTOSCCTRL 寄存器中的 FREQSEL 位，见 [Table 13](#)) 并且在进入深度睡眠模式之前除了定时器时钟的所有其它外设时钟必须在 SYSAHBCLKCTRL 寄存器（见 [Table 21](#)）中被禁止。
- 如果运行的话，看门狗定时器也会产生额外的功耗。

备注：该寄存器的保留位的写入值必须不下表所示，并且在进入深度掉电模式之前给寄存器必须被正确地初始化。

Table 40. 深度睡眠配置寄存器 (PDSLEEPCFG, 地址 0x4004 8230) 位描述

位	符号	值	描述	复位值
2:0	NOTUSED		保留。一般情况下为 111	0
3	BOD_PD		在深度睡眠模式下的 BOD 掉电控制，见 Table 39 .	0
		0	上电	
		1	掉电	
5:4	NOTUSED		保留。一般情况为 11	0
6	WDTOSC_PD		在深度睡眠模式下的看门狗振荡器电源控制，见 Table 39 .	0
		0	上电	
		1	掉电	
7	NOTUSED		保留。一般情况为 1	0
10:8	NOTUSED		保留。一般情况为 000	0
12:11	NOTUSED		保留。一般情况为 11	0
31:13	-	0	保留	0

3.5.33
唤醒配置寄存器

该寄存器的各位指示芯片从深度睡眠模式中唤醒时需要进入的状态。

默认情况下，在从深度睡眠模式下唤醒之后，IRC、flash 存储器被上电并且运行，同时 BOD 也被允许。

备注：保留位必须按下表设置

Table 41. 唤醒配置寄存器 (PDAWAKECFG, 地址 0x4004 8234) 位描述

位	符号	值	描述	复位值
0	IRCOUT_PD		IRC 振荡器输出唤醒配置	0
		0	上电	
		1	掉电	
1	IRC_PD		IRC 振荡器掉电唤醒配置	0
		0	上电	
		1	掉电	
2	FLASH_PD		Flash 唤醒配置	0
		0	有点	
		1	掉电	
3	BOD_PD		BOD 唤醒配置	0
		0	上电	
		1	掉电	
4	ADC_PD		ADC 唤醒配置	1
		0	上电	
		1	掉电	
5	SYSOSC_PD		系统振荡器唤醒配置	1
		0	上电	
		1	掉电	
6	WDTOSC_PD		看门狗振荡器唤醒配置	1
		0	上电	
		1	掉电	
7	SYSPLL_PD		系统 PLL 唤醒配置	1
		0	上电	
		1	掉电	
8	-		保留。该位在运行模式下正常运行时必须置 1	1
9	-		保留。该位在运行模式下正常运行时必须置 0	0
10	-		保留。该位在运行模式下正常运行时必须置 1	1
11	-		保留。该位在运行模式下正常运行时必须置 1	1
12	-		保留。该位在运行模式下正常运行时必须置 0	0
15:13	-		保留。该位在运行模式下正常运行时必须置 111	111
31:16	-	-	保留	-

3. 5. 34 掉电配置寄存器

PDRUNCFG 寄存器中的各位控制着不同模拟模块中的供电。该寄存器可在处理器运行的时随时写入；除了给 IRC 的掉电信号之外，每一次写入都将会立即生效。

为了避免 IRC 掉电时产生抖动，IRC 时钟会在一个稳定点自动关闭。因此，对 IRC 来讲，在掉电生效之前可能会产生一个延时。

默认情况下，在从深度睡眠模式下唤醒之后，IRC、flash 存储器被上电并且运行，同时 BOD 也被允许。

备注：保留位必须按下表设置

Table 42. 掉电配置寄存器 (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
		0	上电	
		1	掉电	
1	IRC_PD		IRC 振荡器掉电	0
		0	上电	
		1	掉电	
2	FLASH_PD		Flash 掉电	0
		0	上电	
		1	掉电	
3	BOD_PD		BOD 掉电	0
		0	上电	
		1	掉电	
4	ADC_PD		ADC 掉电	1
		0	上电	
		1	掉电	
5	SYSOSC_PD		系统振荡器掉电	1
		0	上电	
		1	掉电	
6	WDTOSC_PD		看门狗振荡器掉电	1
		0	上电	
		1	掉电	
7	SYSPLL_PD		系统 PLL 掉电	1
		0	上电	
		1	掉电	
8	-		保留。该位在运行模式下正常运行时必须置 1。	1
9	-		保留。该位在运行模式下正常运行时必须置 0。	0
10	-		保留。该位在运行模式下正常运行时必须置 1。	1
11	-		保留。该位在运行模式下正常运行时必须置 1。	1
12	-		保留。该位在运行模式下正常运行时必须置 0。	0
15:13	-		保留。该位在运行模式下正常运行时必须置 111。	111
31:16	-	-	保留	-

3. 5. 35 设备 ID 寄存器

设备 ID 寄存器是一个只读寄存器，它包含 LPC111x 每个元件的元件编号。开发人员也可通过 ISP/IAP 指令来读取该寄存器 ([Section 21.5.11](#))。

Table 43. 设备 ID 寄存器 (DEVICE_ID, 地址 0x4004 83F4) 位描述

位	符号	描述	复位值
31:0	DEVICEID	LPC111x/LPC11Cxx 原件的部分 ID 号 0x041E 502B = LPC1111FHN33/101 0x2516 D02B = LPC1111FHN33/102 0x0416 502B = LPC1111FHN33/201 0x2516 902B = LPC1111FHN33/202 0x042D 502B = LPC1112FHN33/101 0x2524 D02B = LPC1112FHN33/102 0x0425 502B = LPC1112FHN33/201 0x2524 902B = LPC1112FHN33/202 0x0434 502B = LPC1113FHN33/201 0x2532 902B = LPC1113FHN33/202 0x0434 102B = LPC1113FHN33/301 0x2532 102B = LPC1113FHN33/302 0x0434 102B = LPC1113FBD48/301 0x2532 102B = LPC1113FBD48/302 0x0444 502B = LPC1114FHN33/201 0x2540 902B = LPC1114FHN33/202 0x0444 102B = LPC1114FHN33/301 0x2540 102B = LPC1114FHN33/302 0x0444 102B = LPC1114FBD48/301 0x2540 102B = LPC1114FBD48/302 0x0444 102B = LPC1114FA44/301 0x2540 102B = LPC1114FA44/302 0x1421 102B = LPC11C12/FBD48/301 0x1440 102B = LPC11C14/FBD48/301 0x1431 102B = LPC11C22/FBD48/301 0X1430 102B = LPC11C24/FBD48/301	依赖原件

3.6 复位

LPC111x/LPC11Cxx: 有四个复位源：RESET 引脚、看门狗复位、上电复位（POR）和掉电检测（BOD）。此外，还有一个 ARM 软件复位。

RESET 引脚是一个施密特触发器输入引脚。任何源都可以使复位生效，一旦运行电压达到一个可用电平，开启 IRC 将使复位一直保持有效直到外部复位失效，而振荡器将一直在运行，flash 控制器也已完成其初始化

当 Cortex-M0 处理器之外的外部中断源（POR、OBD 复位、外部复位、看门狗复位）有效时，IRC 开启。在 IRC 开始计时后（上电后最多 6 μs），IRC 提供一个稳定的时钟输出。

- 1. IRC 启动。经过启动时间（上电后至多 6 ms）后，IRC 提供一个稳定的时钟输出。
- 2. ROM 启动处的引导代码。引导代码扮演着引导的任务，并可能跳转到 flash 中。。

3. flash 上电。这大约需要 100 μ s。然后 flash 初始化序列开始，这大约需要 250 个周期。
当内部复位被移除之后，处理器从地址 0 处（即最初的从引导块映射的复位向量）开始执行。在那里，所有的处理器和外设的寄存器被初始为预定值。

3.7 Start-up 行为

复位之后的启动时间如 Figure 4。复位时，IRC 是默认时钟。在供给电压到达阈值 1.8V 之后，它提供一个完整的系统时钟。

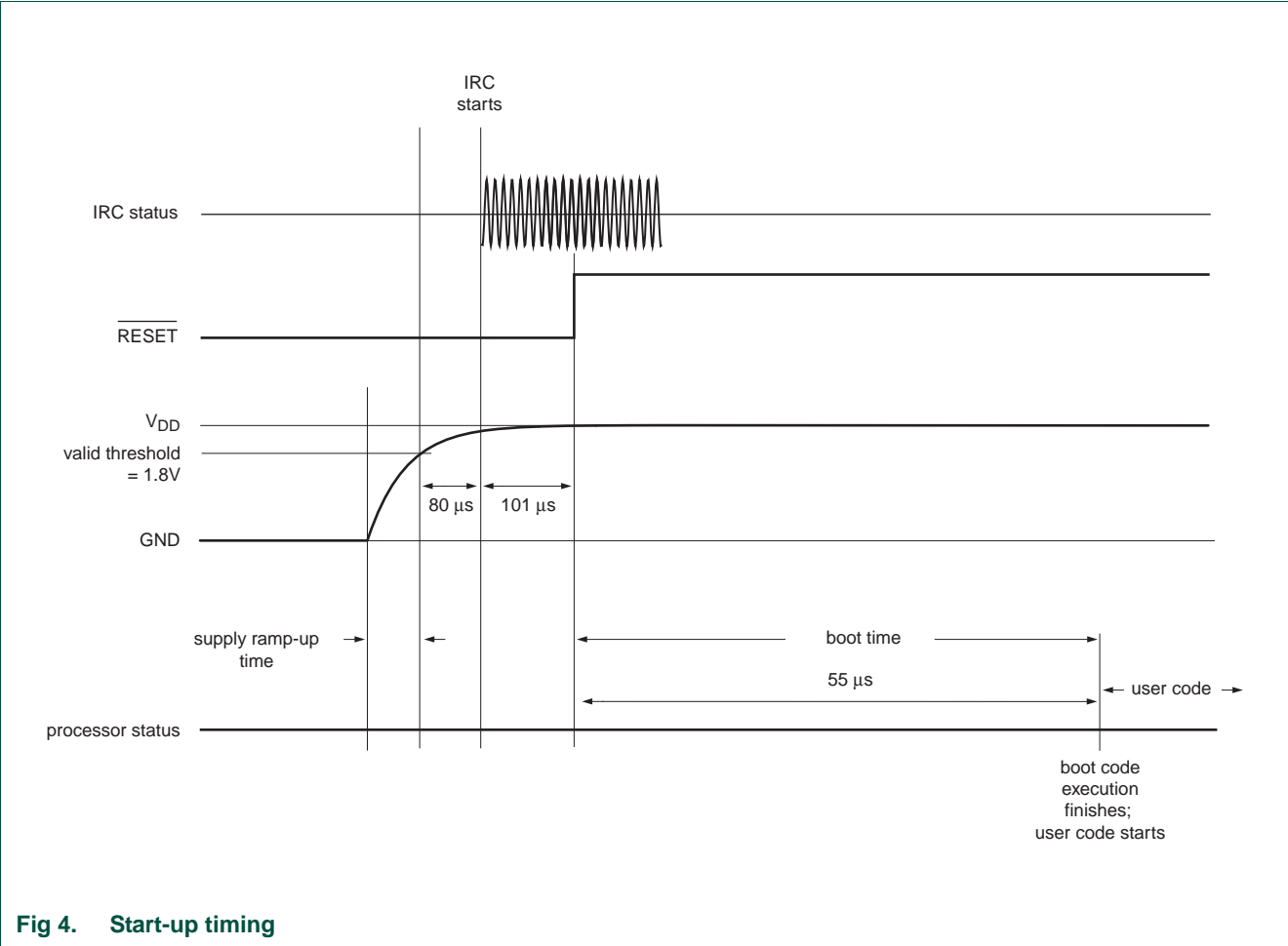


Fig 4. Start-up timing

3.8 掉电检测

LPC111x/LPC11Cxx 有四个电平用于监视 VDD(3V3) 引脚电压。如果电压下降到选定的电平（这四个电平之一），那么 BOD 会产生一个中断信号到 NVIC。若该信号被 NVIC 的中断允许寄存器允许，则会引发一个 CPU 中断；若未被允许，则软件可以通过读 NVIC 状态寄存器（见 Table 53）来监视这个信号。此外还可在四个阈值电平之中选择一个，当电压下降到该电平时处理器将被强制复位（见 Table 33）。

3.9 电源管理

LPC111x/LPC11Cxx 支持功耗控制的多项特性。当芯片运行时，通过选择模块的电源和时钟，可以优化处理器运行时的电源消耗。此外，还有三个专门的省电模式：睡眠模式，深度睡眠模式和深度掉电模式。

备注：在睡眠模式、深度睡眠模式和深度掉电模式下不支持调试模式。

3.9.1 运行模式

在运行模式下，ARM Cortex-M0 核和存储器的时钟由系统时钟提供，外设时钟可以由系统或者由专门的外设时钟提供。

复位之后芯片进入运行模式，由 PDRUNCFG 和 SYSAHBCLKCTRL 寄存器的值决定默认的电配置。电源配置也可以在运行时进行更改。

3.9.1.1 运行模式下的电源配置

运行模式下的电源配置由以下选项控制：

- SYSAHBCLKCTRL 寄存器决定哪个存储器和外设运行 ([Table 21](#))。
- 各种模拟模块的时钟可以由 PDRUNCFG 寄存器在任何时候单独地控制 ([Table 42](#))。
- 系统时钟可以来自 IRC（默认），系统振荡器或者看门狗振荡器（见 [Figure 3](#) 和相关寄存器）。
- 系统时钟的频率可以由 SYSPLLCTRL ([Table 10](#)) 和 SYSAHBCLKDIV 寄存器 ([Table 20](#)) 进行配置。
- 已被选择的外设 (UART, SPI0/1, WDT) 通过各自的分频器使用各自的时钟。外设时钟可以通过各自的分频器寄存器来关闭 ([Table 22](#) 至 [Table 24](#))。

3.9.2 睡眠模式

在睡眠模式下，ARM Cortex-M0 的系统时钟已经停止工作，指令执行被挂起，直到一个复位或中断发生为止。

如果 AHBCLKCTRL 寄存器选择提供时钟的话，外设功能在睡眠模式下仍然可以执行，而且可以产生中断来引起处理器恢复运行。睡眠模式减少处理器自身、存储系统和相关控制器及内部总线使用的动态功耗。处理器状态和寄存器、外设寄存器、内部 SRAM 的值被保持，引脚的逻辑电平不变。

3.9.2.1 睡眠模式下的电源配置

睡眠模式下的电源配置与运行模式下的电源配置相似：

- 时钟保持运行。

- 系统时钟的频率和运行模式时相同，但是处理器是没有被锁定的。
- 模拟的和数字的外围设备的选择同运行模式。

3.9.2.2 编程进入睡眠模式

按照下列步骤进入睡眠模式：

1. PCON 寄存器中的 DPDEN 位必须写为 0 ([Table 48](#)).
2. ARM Cortex-M0 SCR 寄存器中的 DPDEN 位必须写为 0，见 ([Table 354](#)).
3. 使用 ARM Cortex-M0 的等待中断指令 (WFI)。

3.9.2.3 从睡眠模式下唤醒

当一个被 NVIC 允许的中断或者一个复位信号到达时，处理器立即从睡眠模式下唤醒。如果是因为中断而唤醒，处理器将会恢复到由 PDRUNCFG 和 SYSAHBCLKDIV 寄存器预设值的电源配置状态。如果是复位唤醒，微处理器将会进入运行模式下的默认配置。

3.9.3 深度睡眠模式

在深度睡眠模式下，系统时钟不可用。除了 BOD 和看门狗振荡器（要在 PDSLEEPCFG 寄存器中选择）以外的所有模拟模块都被掉电。

在深度睡眠模式下，所有的 flash 和模拟时钟都被关闭。而且，处理器状态和寄存器、外设寄存器、内部 SRAM 值被保持，引脚的逻辑电平不变。

3.9.3.1 深度睡眠模式下的电源配置

深度睡眠模式下的电源配置由 PDSLEEPCFG ([Table 40](#)) 寄存器中的深度睡眠电源配置位决定：

- 深度睡眠模式下唯一可用的时钟源是看门狗振荡器，如果需要定时唤醒处理器就可以设置看门狗振荡器在此模式下运行（见 [Section 3.10.3](#)）。其它的所有时钟源（IRC 和系统振荡器）以及系统 PLL 都被关闭。看门狗振荡器的模拟输出频率必须被设置为最小值 (WDTOSCCTRL = 0001，见 [Table 13](#))。
- 如果实际需要 BOD 也可以在深度睡眠模式下运行。
- 如果看门狗振荡器在此模式下被选择运行，只有看门狗定时器或者一个通用定时器可以在 SYSAHBCLKCTRL 寄存器中被允许，这是为了降低功耗。

3.9.3.2 编程进入深度睡眠模式

按照下列步骤进入深度睡眠模式：

1. 写 0 到 PCON 寄存器的 DPDEN 位 ([Table 48](#))。

2. 通过 PDSLEEPCFG ([Table 40](#)) 寄存器选择深度睡眠模式下的电源配置。
 - a. 如果需要一个定时唤醒，e 确保在 PDRUNCFG 寄存器中对看门狗振荡器进行上电，并且在 MAINCLKSEL 寄存器中讲系统时钟源设置为看门狗时钟 ([Table 18](#))。
 - b. 如果不需要定时唤醒并且看门狗振荡器被关闭，要确保在 PDRUNCFG 寄存器中对 IRC 进行上电并且在 MAINCLKSEL 寄存器 register ([Table 18](#)) 中将时钟源设置为 IRC。这是为了确保系统时钟无障碍的关闭。
3. 通过 PDAWAKECFG ([Table 41](#)) 寄存器选择唤醒后的电源配置。
4. 如果用外部引脚产生唤醒，在启动逻辑寄存器 ([Table 35](#) to [Table 38](#)) 中使能并清零唤醒引脚，并且要在 NVIC 中使能启动逻辑中断。
5. 在 SYSAHBCLKCTRL 寄存器 ([Table 21](#)) 中禁能所有的外设，如果需要 WDT 定时器 / 计数器可以使能他们。
6. 写 1 到 ARM Cortex-M0 SCR 寄存器 ([Table 354](#)) 中的 SLEEPDEEP 位。
7. 使用 ARM WFI 指令。

3.9.3.3 从深度睡眠模式下唤醒

按照下列步骤从深度睡眠模式下唤醒：

- 通过外部引脚的信号。为此 PI00_0 至 PI00_11 和 PI01_0 引脚要在启动逻辑中被使能，启动逻辑不需要任何时钟 但如果 NVIC 允许它可以产生一个中断从而从深度睡眠模式下唤醒处理器。
- 启动逻辑的输入信号可以由一个通用定时器的外部匹配产生 承担匹配定时器功能的引脚必须在 NVIC 中被设置为启动逻辑引脚，相应的定时器必须在 SYSAHBCLKCTRL 寄存器中被使能，同时看门狗振荡器必须在深度睡眠模式下运行（详细信息见 [Section 3.10.3](#)）。
- BOD 产生的复位。在这种情况下，BOD 循环必须在 PDSLEEPCFG 寄存器中被使能，同时 BOD 复位必须在 BODCTRL 寄存器 ([Table 33](#)) 中被使能。
- 看门狗定时器产生复位。这种情况下，看门狗振荡器必须保持运行（见 PDSLEEPCFG 寄存器），同时 WDT 必须在 SYSAHBCLKCTRL 寄存器中被使能。
- 通过外部 RESET 引脚产生复位。

备注：如果看门狗振荡器在深度睡眠模式下运行，它的频率决定了由看门狗定时器产生的复位时间是否长于由 IRC 产生的复位时间。

3.9.4 深度掉电模式

在深度掉电模式下，除 WAKEUP 引脚之外，整个芯片上的电源和时钟都关闭。

在深度掉电模式下 SRAM 和相关寄存器中的内容不能被保持，除了少量数据可以保存在 PMU 模块中的五个 32 寄存器中。

在深度掉电模式下只有 WAKEUP 引脚可以使用。

3.9.4.1 深度掉电模式下的电源配置 e

深度掉电模式没有可以选择的配置选项，所有的时钟、内核、外设都被关闭。只有 WAKEUP 引脚可以使用。

3.9.4.2 编程进入深度掉电模式

按照下列步骤进入深度掉电模式：

1. 写 1 到 PCON 寄存器（见 [Table 48](#)）的 DPDEN 位。
2. 将有用数据保存到通用寄存器中（[Table 49](#)）。
3. 写 1 到 ARM Cortex-M0 SCR 寄存器（[Table 354](#)）的 SLEEPDEEP 位。
4. 再计入深度掉电模式之前将 PDRUNCFG 寄存器的 IRCOUT_PD 和 IRC_PD 位置 0，确保 IRC 上电（默认设置）。

备注：这一步是独立的一部分详细信息见 [Section 3.1](#)。

5. 使用 ARM WFI 指令。

备注：进图此模式之前必须将 WAKEUP 引脚拉高。。

3.9.4.3 从深度掉电模式下唤醒

将 WAKEUP 引脚拉低可以将 LPC111x/LPC11Cxx 从深度睡眠模式下唤醒，芯片会经历整个复位过程（[Section 3.6](#)）。WAKEUP 引脚从高到低的跳变至少要经过 50ns。。

按照下列步骤从深度掉电模式下唤醒：

1. 在深度掉电模式下，一个持续 50ns 的 WAKEUP 引脚从高到低的跳变可以产生一个复位信号。
 - PMU 将会打开片上电压调节器，当内核电压达到复位电压（POR）时，一个系统复位就会触发然后芯片复位。
 - 除了 GPREG0 至 GPREG4 寄存器之外的所有寄存器都将处于复位状态。
2. 一旦芯片重新运行，读 PCON 寄存器（[Table 48](#)）的深度掉电标志位可以判断上一个复位是由什么引起的。
3. 清除 PCON 寄存器（[Table 48](#)）的深度掉电标志位。
4. （可选）读通用寄存器（[Table 49](#) 和 [Table 50](#)）中存储的数据。
5. 为下一个深度掉电循环设置 PMU。

备注：在深度掉电模式下 $\overline{\text{RESET}}$ 引脚无效。

3.10 深度睡眠模式细节

3.10.1 IRC 振荡器

IRC 是 LPC111x 上唯一一个可以总是无故障关闭的振荡器。因此在芯片进入深度睡眠模式前，如果没有选择其他时钟源保持电源，建议用户选择 12MHz 的 IRC 作为时钟源

3.10.2 启动逻辑

深度睡眠模式会在启动逻辑给 ARM 核发出中断时退出。PI00_0 至 PI00_11 以及 PI01_0 都连接了启动逻辑，并且保留为启动引脚。用户必须对启动逻辑寄存器的每个输入编程，为对应的唤醒事件设置合适的边沿极性。而且，必须允许 NVIC 中对应的输入中断。NVIC 中的 0 至 12 个中断相对应到这 13 个引脚上（见 [Section 3.5.28](#)）。

启动逻辑不需要时钟就能运行，因为它在被允许后使用 PIO 输入信号来产生时钟边沿。因此，启动逻辑信号在使用前应先清除（见 [Table 37](#)）。

启动逻辑也能用在正常运行模式下（即不在睡眠模式或深度睡眠模式），通过 LPC111x/LPC11Cxx' 的输入引脚来提供一个向量中断。

3.10.3 使用通用计数器 / 定时器产生一个自己的唤醒事件

如果在 SYSAHBCLKCFG 寄存器中允许深度掉电模式，定时器 / 计数器就开始计数循环，当它等于一个与设值时就产生一个匹配事件。匹配事件引起匹配输出引脚为高，低或者翻转。匹配输出引脚的状态也被启动逻辑所监测，如果该引脚被 NVIC 使能同时启动逻辑在启动逻辑控制寄存器（见 [Table 35](#)）中被做了相应的配置该引脚就可以产生一个唤醒中断。

如果要产生一个定时的深度睡眠模式唤醒事件，必须按如下步骤进行定时器 / 计数器的配置：

1. 在 IOCONFIG 模块中配置引脚端口位匹配输出。从 PI00_1 或者 PI00_8 至 PI00_11 中选择，这些都是启动逻辑输入引脚并且具有匹配输出功能。
2. 在相应的计数器 / 定时器中，设置匹配值，并且为已选择的引脚设置匹配值。
3. 在 PDSLEEPCFG 寄存器中设置看门狗振荡器在深度睡眠模式下运行。
4. 在 MAINCLKSEL 寄存器 register ([Table 18](#)) 中将时钟源设置为看门狗振荡器并且在寄存器总确保看门狗振荡器上电。
5. 使能引脚并配置边沿特性，在启动逻辑寄存器 ([Table 35](#) to [Table 38](#)) 中复位启动逻辑，在 NVIC 中使能中断。

6. 在 SYSAHBCLKCTRL 寄存器中关闭其它所有外设。
7. 确保 PCON 寄存器中的 DPDEN 位为 0 ([Table 48](#))。
8. 写 1 到 ARM Cortex-M0 SCR 寄存器 ([Table 354](#)) 中的 SLEEPDEEP 位。
9. 启动定时器 / 计数器。
10. 使用 ARM WFI 指令进入深度睡眠模式。

3.11 系统 PLL 功能描述

LPC111x/LPC11Cxx 使用系统 PLL 来为内核和外设产生时钟

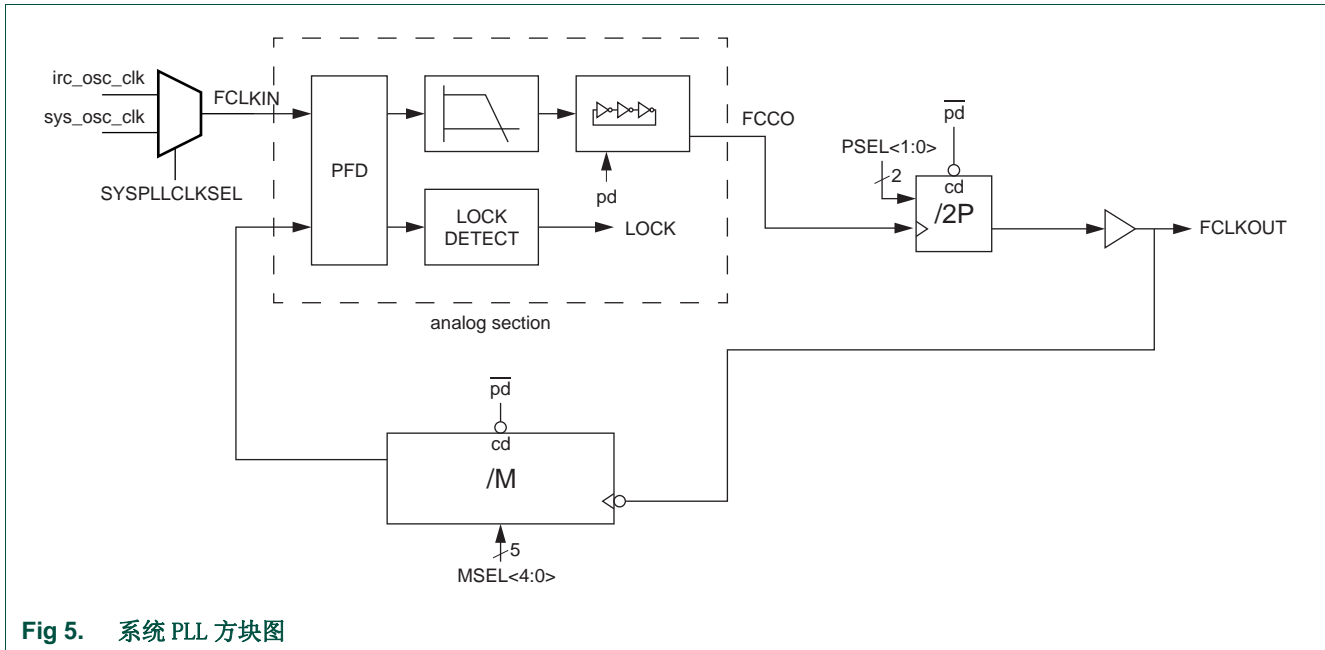


Fig 5. 系统 PLL 方块图

PLL 方框图如 [Figure 5](#)，输入频率介于 10 MHz 和 25 MHz 之间。输入时钟直接供给相位频率检测器 (PFD)，这个模块比较输入时钟的相位和频率，如果不符合要求就产生一个控制信号。回路滤波器过滤控制信号并且对目前振荡频率进行分频 (CCO)，这将会产生主时钟和两个额外的选择波段。CCO 频率介于 156 MHz 和 320 MHz 之间，这个时钟要被 2^P 分频以产生输出时钟要么直接作为输出时钟 (s)。主输出时钟然后被反馈分频器以 M 至进行分频继而产生反馈时钟。输出信号的相位频率检测器同时被锁定检测器检测，当 PLL 被锁定是就产生信号。

备注： P 和 M 的必须选择恰当，以确保 PLL 输出时钟频率 FCLKOUT 低于 100MHz。

3.11.1 锁定检测器

锁定检测器测量输入和反馈时钟上升沿的相位的差别。只有在差别小于“锁定标准”并连续超过 8 个输入时钟周期时，锁定输出才从低切换到高。单独一个很大的相位差会立即复位计数器，并引起锁定信号下降（如果为高的话）。要求测量连续的 8 个相位低于某一定值的，可确保锁定检测器在输入和反馈时钟的相位和频率没有很好的对齐之前不会指示锁定。这可有效地防止了错误锁定标志，从而确保了无干扰信号锁定。

3.11.2 掉电控制

为了降低不需要 PLL 时钟时的功耗，可以采用一种掉电模式。通过设置掉电配置寄存器 (Table 42) 中的 SYS_PLL_PD 位为 1 来允许该模式。在这种模式下，内部电流基准将被关闭，振荡器和相频检测器将停止工作，分频器进入复位状态。在掉电模式下，锁定输出为低以表示 PLL 没有锁定。通过设置 SYS_PLL_PD 位为 0 来终止掉电模式时，PLL 将恢复正常运行，一旦重新获得输入时钟锁定，PLL 将会把锁定信号置高。

3.11.3 分频系数编程

后分频器

后分频器的分频系数受 PSEL 位的控制。该分频系数是 Table 10 中 PSEL 所选择的 P 值的两倍。这确保输出一个具有 50% 占空比的输出时钟。

反馈分频器

反馈分频器的分频系数由 MSEL 位控制。如在 Table 10 中所规定的，PLL 的输出时钟与输入时钟之间的分频系数是 MSEL 位的十进制的值加 1。

改变分频值

不建议在 PLL 运行时改变分频器的值。由于无法和分频器同步改变 MSEL 和 PSEL 的值，从而可能产生计数器读取一个未定义值的风险，这将导致多余的毛刺和输出时钟频率的下降。建议在修改分频值之前先将 PLL 关电，调整分频器之后再 将 PLL 上电。

3.11.4 频率选择

PLL 频率方程使用下列参数（亦如 Figure 3）：

Table 44. PLL 频率参数

参数	系统 PLL
FCLKIN	来自 SYSPLLCLKSEL 倍频器 (见 Section 3.5.9) 的 sys_pllclk _{in} （系统 PLL 输入时钟）频率。
FCCO	电流控振荡器 (CCO) 的频率； 156 ~ 320 MHz。
FCLKOUT	sys_pllclk _{out} 的频率。
P	系统 PLL 后分频器值；SYSPLLCTRL 中的 PSEL 位域 (见 Section 3.5.3)。
M	系统 PLL 反馈分频器寄存器；SYSPLLCTRL 的 MSEL 位域 (见 Section 3.5.3)。

3.11.4.1 正常模式

在该模式下后分频器被允许，产生一个 50% 占空比的满足以下频率关系的周期时钟：

(1)

$$FCLKOUT = M \times FCLKIN = (FCCO)/(2 \times P)$$

为了给 M 和 P 选择合适的值，建议遵循下面步骤：

- 1. 指定输入时钟频率。
- 2. 根据 $M = Fclkout / Fclkin$ 计算 M，以获取所需的输出频率。
- 3. 根据 $FCCO = 2 \times P \times Fclkout$ 确定 FCCO 的值。
- 4. 根据 [Table 10](#) 中规定的限制，验证所有的频率和分频器的值。
- 5. 确保 $FCLKOUT < 100\text{ MHz}$.

[Table 45](#) 展示了如何使用 SYSPLLCTRL 寄存器 ([Table 10](#)) 为一个 12MHz 的振荡器配置 PLL。如果系统时钟分频器 SYSAHBCLKDIV 被设置为 1 (见 [Table 20](#))，那么主时钟就等于系统时钟。

Table 45. PLL 配置示例

PLL 输入时钟 c sys_pllclkin (FCLKIN)	主时钟 (FCLKOUT)	MSEL 位 Table 10	M 分频器 值	PSEL 位 Table 10	P 分频器 值	FCCO 频率
12 MHz	48 MHz	00011	4	01	2	192 MHz
12 MHz	36 MHz	00010	3	10	4	288 MHz
12 MHz	24 MHz	00001	2	10	4	192 MHz

3.11.4.2 掉电模式

在这种模式下，内部电流基准将被关闭，振荡器和相位频率检测器将被停止，分频器将进入一个复位状态。而在掉电模式下，锁定输出将被置低电平，表明 PLL 未锁定。当将 pd 位置低中止掉电模式时 ([Table 42](#))，PLL 将恢复正常运行，一旦重新获得输入时钟锁定，PLL 将会把锁定信号置高。

3.12 访问 Flash 存储器

根据系统时钟频率，通过写地址为 0x4003 C010 的 FLASHCFG 寄存器来为 Flash 存储器配置不同的存取时间。这个寄存器是 flash 配置块的一部分 (见 [Figure 2](#))。

备注：该寄存器设置不当可能导致 LPC111x/LPC11Cxx flash 的错误操作。

Table 46. Flash 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		Flash 存储器存取时间 . FLASHTIM +1 等于系统时钟用于存取 flash 所用的次数。	10
		00	1 系统时钟的 flash 存取时间 (系统时钟频率为 20MHz)。	
		01	2 系统时钟的 flash 存取时间 (系统时钟频率为 40MHz)。	
		10	3 系统时钟的 flash 存取时间 (系统时钟频率为 50MHz)。	
		11	保留	
31:2	-	-	保留。用户软件不能改变此位域的值。 31:2 位写回的值必须和读取的一样。	-

4.1 如何阅读本章

备注：对于 LPC111x/102/202/302 的电源管理，同样可以参考 [Chapter 5](#)。

4.2 概述

PMU 控制深度掉电模式。四个通用寄存器可用于在深度掉电模式下保存数据。

4.3 寄存器描述

Table 47. 寄存器概览：PMU (基址 0x4003 8000)

名称	访问	便宜地址	描述	复位值
PCON	R/W	0x000	电源控制寄存器	0x0
GPREG0	R/W	0x004	通用寄存器 0	0x0
GPREG1	R/W	0x008	通用寄存器 1	0x0
GPREG2	R/W	0x00C	通用寄存器 2	0x0
GPREG3	R/W	0x010	通用寄存器 3	0x0
GPREG4	R/W	0x014	通用寄存器 4	0x0

4.3.1 电源控制寄存器

电源控制寄存器用来选择 ARM Cortex-M0 是否进入掉电模式 (睡眠模式或者深度睡眠模式) 或者深度掉电模式并且分别给各个模式提供标志。怎样进入掉电模式见 [Section 3.9](#)。

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

位	符号	值	描述	复位值
0	-	-	保留，此位不能写 1	0x0
1	DPDEN		深度掉电模式允许	0
		0	WFI 指令将进入睡眠或者深度睡眠模式 (ARM Cortex-M0 核的时钟关闭)	
		1	WFI 指令将进入深度掉电模式 (ARM Cortex-M0 核掉电)。	
7:2	-	-	保留，此位不能写 1	0x0

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

位	符号	值	描述	复位值
8	SLEEPFLAG		睡眠模式标志	0
		0	读：未进入掉电模式， LPC111x/LPC11Cxx 处于运行状态。 写：无效。	
		1	读：睡眠 / 深度睡眠或者深度掉电模式。 写：写 1 将清除 SLEEPFLAG 位为 0。	
10:9	-	-	保留，此位不能写 1。	0x0
11	DPDFLAG		深度掉电标志	0x0
		0	读：未进入深度掉电模式。 写：无效。	0x0
		1	读：深度掉电模式。 写：清除深度掉电标志。	0x0
31:12	-	-	保留，此位不能写 1。	0x0

4.3.2 通用寄存器 0~3

在深度掉电模式下，只要给 VDD(3V3) 引脚供电，通用寄存器就能保留数据。只有在冷启动时，所有电源都从处理器上移除，通用寄存器才复位。

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

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

4.3.3 通用寄存器 4

在处理器进入深度睡眠后，只要 VDD 引脚不掉电，通用寄存器 4 仍可以保留数据。只有在冷启动，移除芯片上所有电源供电时才会复位该通用寄存器。

备注：I 如果 VDD 上的外部供电电压降低到低于某个电压值（2.2V），为了让处理器能从深度掉电模式下唤醒，需要消除 WAKEUP 输入引脚的滞后。

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

位	符号	值	描述	复位值
9:0	-	-	保留，不能往这些位写 1。	0x0

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

位	符号	值	描述	复位值
10	WAKEUPHYS		许 WAKEUP 引脚滞后	0x0
		1	允许 WAKEUP 引脚滞后	
		0	禁止 WAKEUP 引脚滞后	
31:11	GPDATA		深度掉电模式下保持数据	0x0

4.4 功能描述

进入或者退出深度掉电模式的详细信息见 [Section 3.9.4](#).

5.1 如何阅读本章

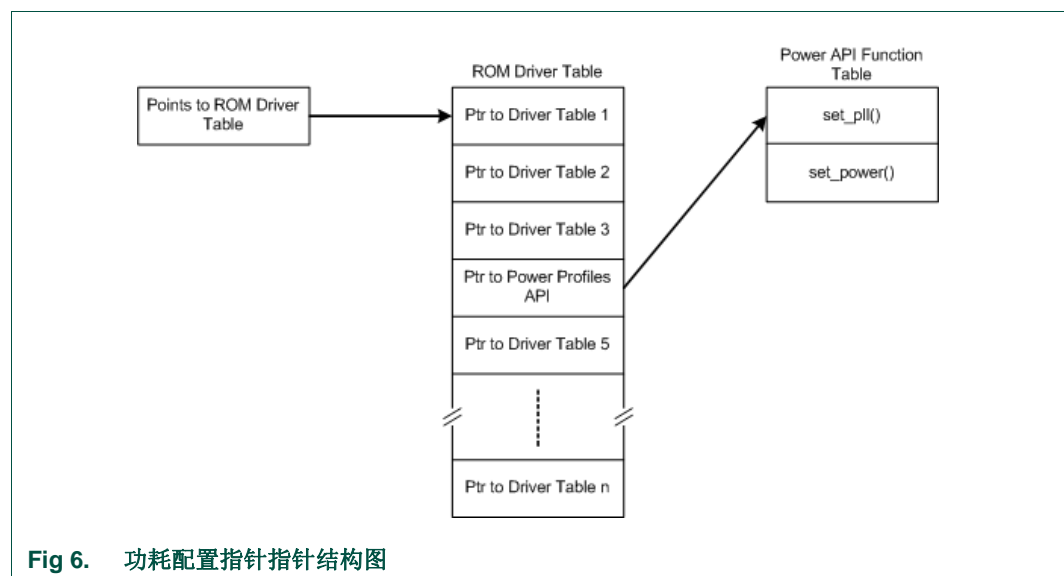
本功耗管理模块仅适用于处理器 LPC111x/102/202/302(LPC1100L 系列)。

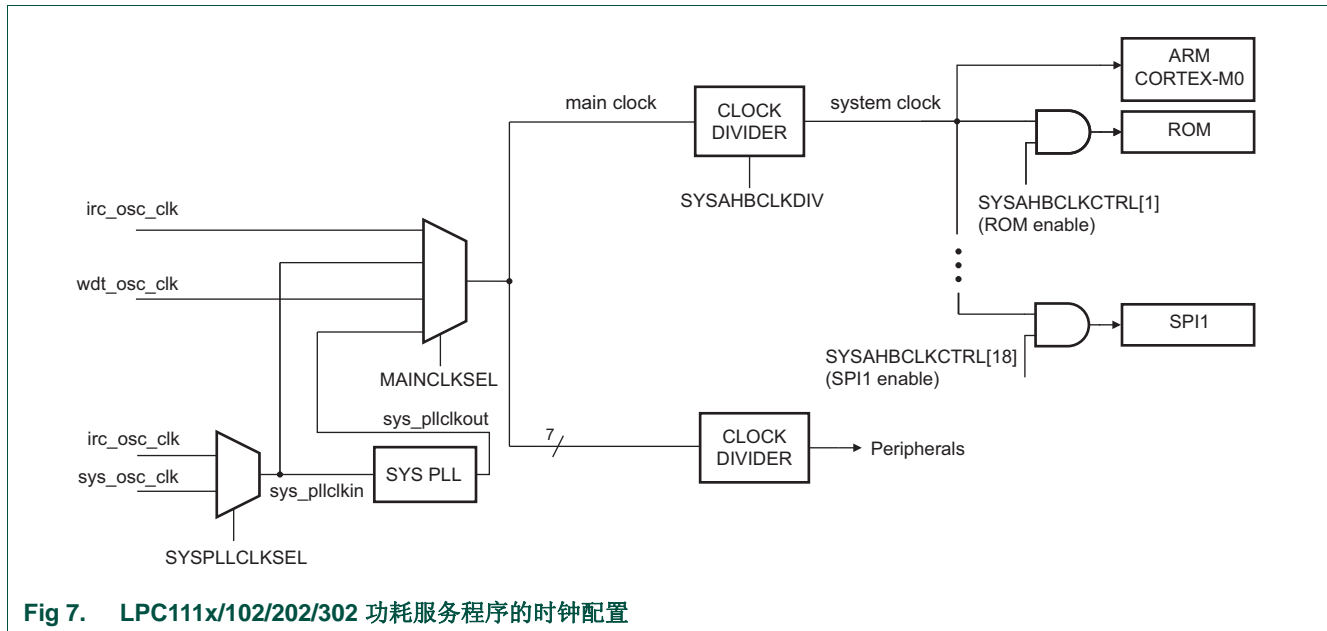
5.2 特征

- 基于 ROM 的应用服务程序
- 电源管理服务程序
- 时钟服务程序

5.3 概述

应用程序访问 ROM 是通过执行函数来完成的，每个函数被一个存放在 ROM 驱动表中的指针所指。[Figure 6](#) 展示应用于功耗管理应用程序接口的指针结构。





5.4 定义

在一个功耗管理的应用程序中必须包含以下的内容定义：

```
typedef struct _PWRD {
    void (*set_pll)(unsigned int cmd[], unsigned int resp[]);
    void (*set_power)(unsigned int cmd[], unsigned int resp[]);
} PWRD;
typedef struct _ROM {
    const PWRD * pWRD;
} ROM;
ROM ** rom = (ROM **) 0x1FFF1FF8;
unsigned int command[4], result[2];
```

5.5 时钟子程序

5.5.1 set_pll

这个子程序根据调用的参数来配置系统 PLL，如果希望得到的时钟可以通过简单地对 PLL 输入时钟进行分频得到，那么 set_pll 就会旁路 PLL 以降低系统功耗。

备注：在时钟子程序被调用之前，PLL 时钟源 (IRC/ 系统振荡器) 必须已经选择好 (Table 16)，主时钟源必须被设置为系统 PLL 输入时钟 (Table 18)，系统 AHB 时钟分频值必须置为 1 (Table 20)。

set_pll 试图找到一个 PLL 设置可以匹配调用的参数，一旦找到一个正确的反馈分频器值 (SYSPLLCTRL,M)、后分频器值 (SYSPLLCTRL,P)、和系统 AHB 分频器值的一个组合，set_pll 就会加载这些值并且把系统时钟设置为系统 PLL 的输出时钟（如果需要）。

时钟子程序返回一串结果代码用来表示系统 PLL 是成功设置 (PLL_CMD_SUCCESS)，还是出错（这种情况下结果代码标明出错信息），当前系统频率值也会返回，应用程序应该根据返回的信息来调整其它的设备时钟 (SSP,UATR,WDT, 和 / 或输出时钟)。

Table 51. set_pll 子程序

功能	set_pll
输入	<p>参数 0: 系统 PLL 输入时钟频率 (kHz)。</p> <p>参数 1: 希望得到的系统时钟 (kHz)。</p> <p>参数 2: 模式 (CPU_FREQ_EQU, CPU_FREQ_LTE, CPU_FREQ_GTE, CPU_FREQ_APPROX)</p> <p>参数 3: 系统 PLL 锁定时间。</p>
输出	<p>结果 0: PLL_CMD_SUCCESS PLL_INVALID_FREQ PLL_INVALID_MODE PLL_FREQ_NOT_FOUND PLL_NOT_LOCKED</p> <p>结果 1: 系统时钟 (kHz)。</p>

当要调用 set_pll 电源子程序时必须要做以下声明：

```
/* 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
```

简化的时钟配置请参考 [Figure 7](#)，详细信息参考 [Figure 3](#)。

5.5.1.1 参数 0: 系统 PLL 输入频率 和 参数 1: 希望得到的系统时钟

set_pll 要计算出一个系统PLL时钟不大于50MHz的设置，当希望得到的系统时钟和系统PLL输入时钟的比率是一个整数时很容易实现，但在其它情况下也能找到答案。

系统 PLL 输入频率（参数 0）必须在 10MHz 和 25MHz 之间，希望得到的系统时钟（参数 1）必须在 1Hz 和 50MHz 之间。这两个条件中的任一个不满足， set_pll 返回 PLL_INVALID_FREQ 并且将参数 0 作为结果 1 返回，因为 PLL 设置未 发生改变。

5.5.1.2 参数 2: 模式

set_pll 的首要任务是在参数 1 指定的速率下找到一个生成系统时钟的设置。如果这个精确匹配值不能找到，就要输入模式参数 (参数 2)。如果实际的系统时钟小于等于、大于等于或者和预期的系统时钟（参数 1）相近时参数 2 就可以用以匹配时钟。

一个指定 CPU_FREQ_EQU 的调用只有在 PLL 可以输出精确符合参数 1 要求的频率的情况下才能成功。

CPU_FREQ_LTE 可以在需要的频率不会被超过的情况下使用 (比如因为全部电流消耗和功率预算原因)。

CPU_FREQ_GTE 适用于要求 CPU 的处理能力不低于某一下限值的应用程序中。

CPU_FREQ_APPROX 将产生一个与需要的值尽可能接近的系统时钟 (它可以略大于或者略小于需要的值)。

如果一个非法的模式被指定, `set_pll` 返回 PLL_INVALID_MODE, 如果期望的系统时钟超过了此子程序所支持的范围, `set_pll` 返回 PLL_FREQ_NOT_FOUND, 在这些情况下, 当前的 PLL 设置不会被改变, 参数 0 作为结果 1 返回。

5.5.1.3 参数 3: 系统 PLL 锁定时间

如果一个有效的配置被选择, 系统 PLL 的锁定时间将不会超过 100us。如果参数 3 为 0, `set_pll` 将会无限期地等待系统 PLL 被锁定。如果参数 3 是一个非零值, 则此值就是在 PLL 返回之前对 PLL 进行锁定检测的次数, 在这种情况下, 系统 PLL 设置将不会被改变并且参数 0 作为结果 1 返回。

备注: PLL 锁定所用的时间取决于 PLL 的输入时钟源 (IRC/ 系统振荡器) 和 PLL 自身的特性。被选定的时钟源可能会因实际的工作条件比如电源供给, 环境温度而产生微小的震动。所以, 当一个理论上很好的时钟源被应用但一个 PLL_NOT_LOCKED 请求却被允许的情况下, `set_pll` 子程序会在声明 PLL 时钟源无效之前唤醒多次。

提示: 将参数 3 设置为系统 PLL 频率 [Hz] 的 10000 分之一时, 将会产生足够多的设置 PLL 锁定轮询周期。

5.5.1.4 示例代码

下面的例子将会阐明以上所讨论的 `set_pll` 的一些特点。

5.5.1.4.1 无效频率 (设备最大时钟频率)

```
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 PLL 输入时钟为 12MHz, 系统时钟为 60MHz, 应用程序准备无限期地等待 PLL 被锁定。但是, 预期的系统时钟是 60MHz, 它超过了最大时钟频率 (50MHz)。因此, `set_pll` 将 PLL_INVALID_FREQ 作为结果 0 返回, 将 12000 作为结果 1 返回并且不改变 PLL 设置。

5.5.1.4.2 无效的时钟选择 (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 PLL 输入时钟是 12MHz，系统时钟仅仅是 40KHz，没有等待 PLL 锁定时间。因为最大的分频值是 255，而 40KHz 要求的分频值是 300，所以 set_pll 将 PLL_INVALID_FREQ 作为结果 1 返回，将 12000 作为结果 2 返回并且不改变 PLL 设置。

5.5.1.4.3 找不到精确解（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 PLL 输入时钟为 12MHz，系统时钟是 25MHz，应用程序无限期地等待 PLL 被锁定。由于在目前的限制条件下没有有效地 PLL 设置，因此，set_pll 将 PLL_FREQ_NOT_FOUND 作为结果 0 返回，将 12000 作为结果 1 返回并且不改变 PLL 的设置。

5.5.1.4.4 系统时钟小于或者等于预期值

```
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 PLL 的输入时钟是 12MHz，系统时钟是 25MHz，没有等待锁定时间。Set_pll 将 PLL_CMD_SUCCESS 作为结果 0 返回，将 24000 作为结果 1 返回，新的系统时钟是 24MHz。

5.5.1.4.5 系统时钟大于或等于预期值

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

以上代码指定 PLL 输入时钟为 12MHz，系统时钟是 25MHz，没有等待锁定时间。Set_pll 将 PLL_CMD_SUCCESS 作为结果 0 返回，将 36000 作为结果 1 返回，新的系统时钟是 36MHz。

5.5.1.4.6 系统时钟约等于预期值

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

以上代码指定 PLL 输入时钟为 12MHz，系统时钟约为 16.5MHz，没有等待锁定时间。Set_pll 将 PLL_CMD_SUCCESS 作为结果 0 返回，将 16000 作为结果 1 返回，新的系统时钟是 16MHz。

5.6 电源子程序

5.6.1 set_power

此子程序根据调用的参数对设备的内在消耗功率进行配置，其目的是在确保应用程序运行在其最佳状态的情况下尽可能的降低功耗。

备注: set_power子程序是为系统配置为SYSAHBCLKDIV =1 的情况而设计的(系统时钟分频寄存器, 参考 [Table 20](#) 和 [Figure 7](#))。当系统时钟分频寄存器的值不为 1 时，在应用程序中调用 set_power 子程序将不会太多地提高微处理器的性能 ---- 与把主时钟和系统时钟配置为相同的值相比。

set_power 返回结果代码，用以表明电源设置是否成功地改变。

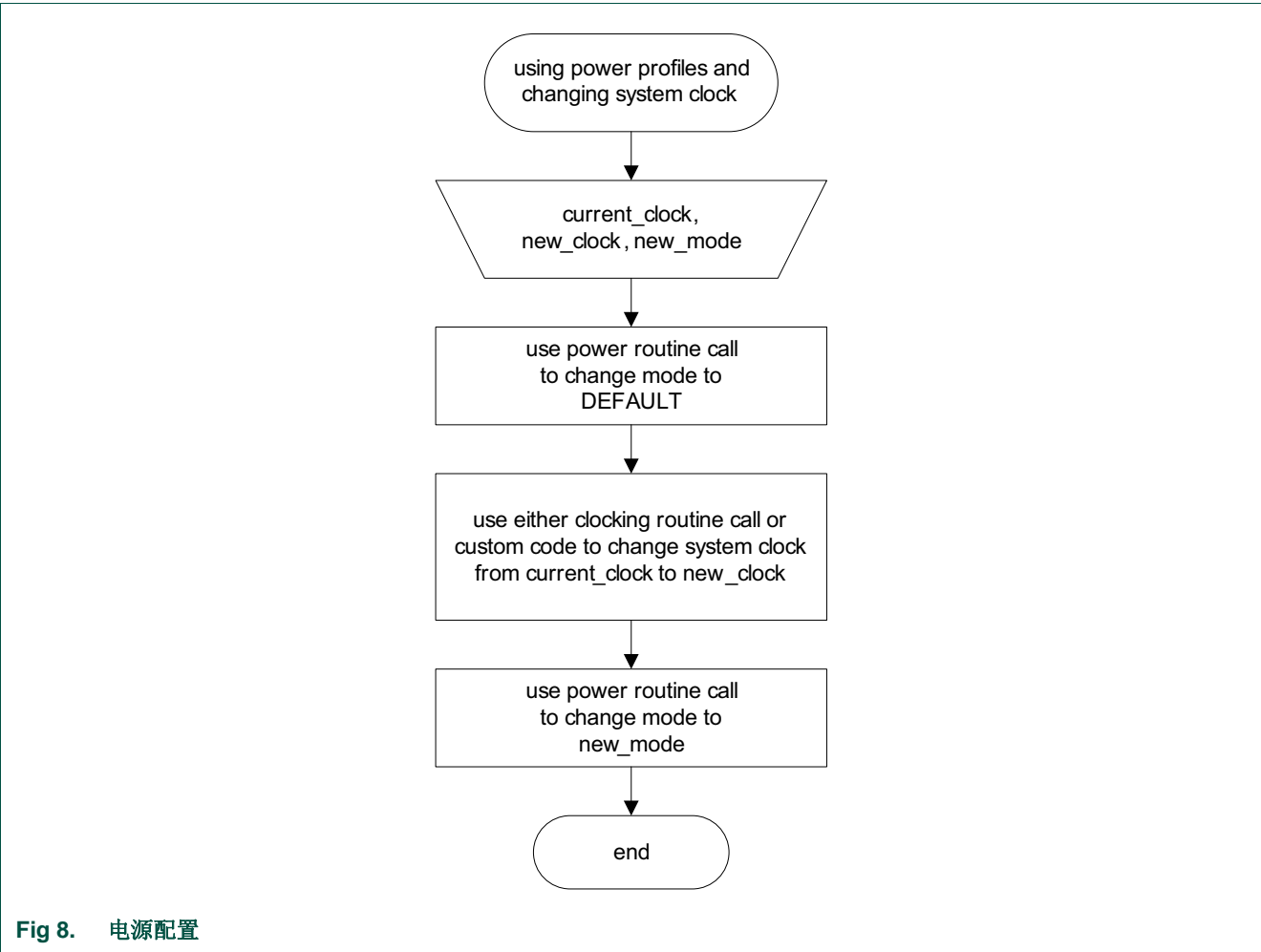


Table 52. set_power 子程序

功能	set_power
输入	参数 0: 主时钟 (MHz) 参数 1: 模式 (PWR_DEFAULT, PWR_CPU_PERFORMANCE, PWR_EFFICIENCY, PWR_LOW_CURRENT) 参数 2: 系统时钟 (MHz)
输出	结果 0: PWR_CMD_SUCCESS PWR_INVALID_FREQ PWR_INVALID_MODE

在调用 set_power 子程序时以下内容要被定义:

```
/* 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
```

简化的时钟配置请参考 [Figure 7](#), 详细信息参考 [Figure 3](#)。

5.6.1.1 参数 0: 主时钟

主时钟是为微控制器提供系统时钟和外围设备时钟的。主时钟可以通过调用时钟子程序或者用户自定义的代码来进行配置。这个参数必须是一个介于 1Hz 和 50MHz 之间的一个整数。如果参数超出了这个范围, 那么 set_power 将会返回 PWR_INVALID_FREQ 并且不改变电源控制系统。

5.6.1.2 参数 1: 模式

这个输入模式参数 (参数 1) 可以指定四个可选的电源设置之一。如果这个参数是一个非法值, set_power 返回 PWR_INVALID_MODE 并且不改变电源控制系统。

PWR_DEFAULT 使设备保持在基准电源设置 (类似于复位状态) 的情况下。

PWR_CPU_PERFORMANCE 配置微控制器以便它可以有更强的处理应用程序的能力, 这样 CPU 的性能比默认情况下高 30%。

PWR_EFFICIENCY 设置是为了在实际电流和 CPU 运行代码、处理数据的能力之间找一个平衡点。在这种模式下, 不管是在提高处理器的性能上, 还是在降低实际电流方便上都超过默认设置。

PWR_LOW_CURRENT 侧重于降低功耗而不是提高处理器的性能。

5.6.1.3 参数 2: 系统时钟

系统时钟是在调用 set_power 时微控制器核运行的时钟。这个参数必须是介于 1 到 50MHz 之间的一个整数。

5.6.1.4 示例代码

下面的例子将会阐明以上所讨论的 `set_power` 的一些特点。

5.6.1.4.1 无效的频率 (设备最大时钟频率)

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

以上设置将应用于主时钟和系统时钟都为 60MHz 的情况下，需要最大的 CPU 处理能力。由于指定的 60MHz 超过了最大时钟频率 (50MHz)，`set_power` 将 `PWR_INVALID_FREQ` 作为结果 0 返回，并且不会改变现有的电源设置。

5.6.1.4.2 一个可用的电源设置

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

以上设置应用于主时钟和系统时钟都为 24MHz 的情况下，着重强调效率。`set_power` 在配置微控制器的内部功耗控制特性之后将 `PWR_CMD_SUCCESS` 作为结果 0 返回。

6.1 如何阅读本章

C_CAN 控制器中断仅仅适用于 LPC11Cxx 。

6.2 概述

内嵌向量中断控制器 (NVIC) 是 Cortex-M0 的一个重要组成部分。它与 CPU 处理器内核紧密耦合，实现低中断延迟以及对新到中断的有效处理。

6.3 特征

- NVIC 是 ARM Cortex-M0 的一个集成部分。
- 紧耦合方式使中断延迟大大缩短。
- 可控制系统的异常及外设中断。
- NVIC 支持 32 路向量中断。
- 4 个带硬件优先级屏蔽的可编程中断优先级。
- 可产生软件中断。

6.4 中断源

[Table 53](#) 列出了每一个外设中断源的功能。每一个外设可能有一个或多个到中断向量控制器的中断线。每一条中断线可代表一个以上的中断源，除了 ARM 公司确定的标准，连接线的位置没有优先级和重要性的区别。

NVIC 寄存器位描述见 [Section 23.5.2](#) 。

Table 53. 中断源到向量中断控制器的链接

异常编号	向量便宜	功能	标志
12 至 0		启动逻辑唤醒中断	每一个中断都连接到一个端口（PIO）上，作为输入将 MCU 从深度睡眠中唤醒；中断 0 到 11 对应 PIO0_0~11 引脚，中断 12 对应 PIO1 的第 0 脚。见 Section 3.5.28 .
13		C_CAN	C_CAN 中断
14		SPI/SSP1	Tx FIFO 半空 Rx FIFO 半满 Rx 超时 Rx 溢出
15		I ² C	SI (状态更改)

Table 53. 中断源到向量中断控制器的链接

异常编号	向量便宜	功能	标志
16		CT16B0	匹配 0 - 2 捕获 0
17		CT16B1	匹配 0 - 1 捕获 0
18		CT32B0	匹配 0 - 3 捕获 0
19		CT32B1	匹配 0 - 3 捕获 0
20		SPI/SSP0	Tx FIFO 半空 Rx FIFO 半满 Rx 超时 Rx 溢出
21		UART	Rx 线状态 (RLS) 发送保持寄存器空 (THRE) Rx 数据可用 (RDA) 字符超时指示 (CTI) 自动波特率结束 (ABEO) 自动波特率超时 (ABTO)
22		-	保留
23		-	保留
24		ADC	A/D 转换结束
25		WDT	看门狗中断 (WDINT)
26		BOD	掉电检测
27		-	保留
28		PIO_3	GPIO 端口 3 中断状态
29		PIO_2	GPIO 端口 2 中断状态
30		PIO_1	GPIO 端口 1 中断状态
31		PIO_0	GPIO 端口 0 中断状态

7.1 如何阅读本章

对于不同的 LPC111x/LPC11Cxx 型号和封装，I/O 配置寄存器的实现各不相同。[Table 55](#) 列出了在不同的封装上使用了哪些 IOCON 寄存器。

IOCON 块几个功能的实现依赖于特定的型号：

C_CAN 引脚

- 对于 LPC11C12/C14，PI03_4 和 PI03_5 的功能不可用，相反，这两个引脚专门用于 C_CAN 的接收和发送（见 [Table 55](#)），无上拉或下拉电阻。C_CAN 引脚没有可编程的引脚配置。
- 对于 LPC11C22/C24，引脚 PI01_9，PI02_4，PI02_5 和 PI02_9 不可用，并被片上 CAN 接收器引脚代替，CAN 接收器引脚没有可编程的引脚配置。

伪开漏功能

对于 LPC111x/102/202/302，在 IOCON 寄存器可以对每个除了 I2C 引脚的数字引脚选择一个伪开漏模式，在 LPC111x/101/201/301 中开漏模块不可用。

上拉水平

如果上拉寄存器允许（默认），对 LPC111x/101/201/301 所有非 I2C 引脚上拉至 2.6V，对 LPC11Cxx 和 LPC111x/102/202/302 上拉至 3.3V ($V_{DD} = 3.3\text{ V}$)。

7.2 特性

I/O 配置寄存器控制着每个端口的电气特性，可编程设置以下特性：

- 引脚功能
- 内部上拉 / 下拉电阻或总线保持功能
- 滞后
- 引脚控制 ADC 输入的模拟输入或数字模式
- 引脚控制 I2C 总线功能的 I2C 模式
- 非 I2C 引脚为伪开漏模式（详见 [Section 7.1](#)）

7.3 概述

IOCON 寄存器控制着功能（GPIO 或外设功能）、输入模式和所有 $PIO_n.m$ 管脚的滞后。另外，可以为不同的 I2C 总线模式配置 I2C 总线管脚。如果管脚用作 ADC 的输入，那么可以选择模拟输入模式。

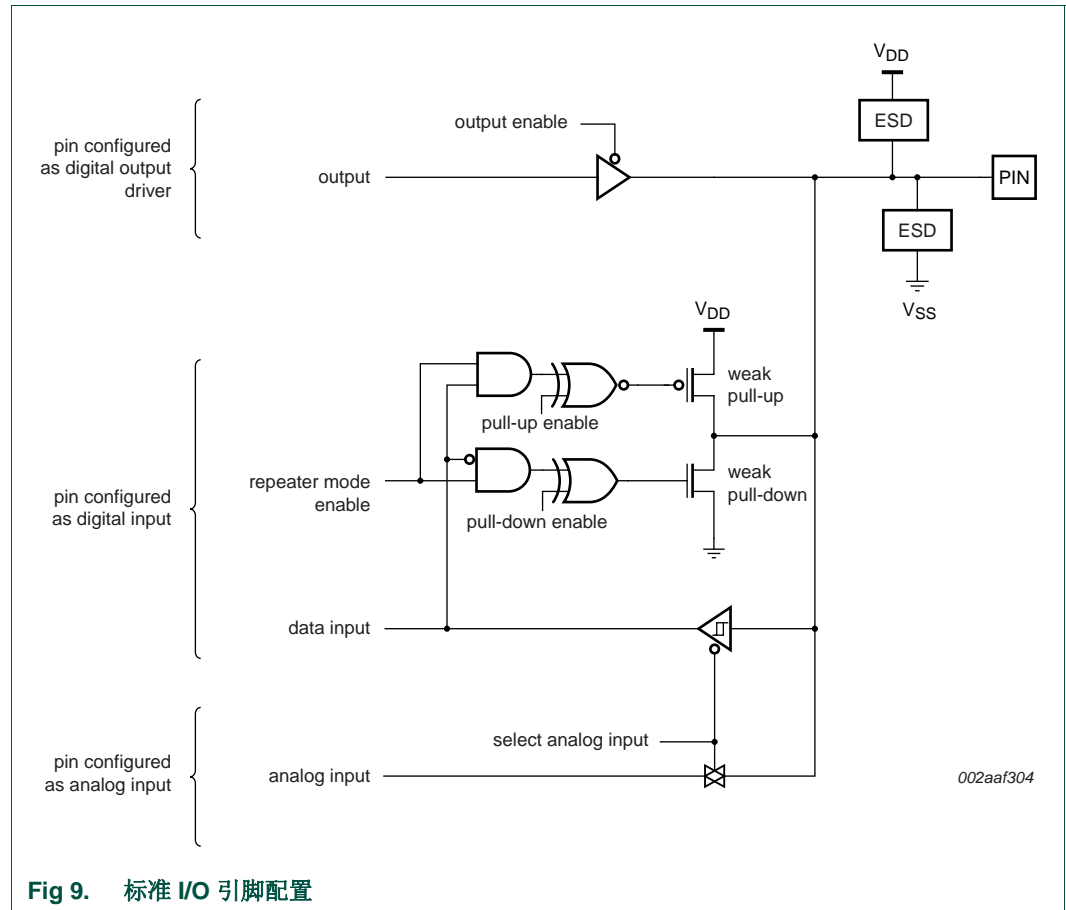


Fig 9. 标准 I/O 引脚配置

7.3.1 引脚功能

IOCON 寄存器的 FUNC 位可以设为 GPIO (FUNC=000) 或者一种外设功能。如果管脚用作 GPIO 管脚，那么 GPIO_nDIR 寄存器确定哪个管脚配置为输入或输出（见 [Section 9.3.2](#)）。如果配置为其它外设功能，引脚方向根据引脚的功能特性自动决定，此时 GPIO_nDIR 寄存器设置无效。

7.3.2 引脚模式

IOCON 寄存器的 MODE 位允许为每个管脚选择片内上拉或下拉寄存器或者选择中继模式。

片内电阻配置有上拉使能、下拉使能或无上拉 / 下拉。默认的值是上拉。详见 [Section 7.1](#)。

如果引脚处于逻辑高电平那么中继模式将允许上拉电阻，如果为低电平则允许下拉电阻。如果引脚被作为输入且没有外部驱动，那么引脚将会保留上次的已知状态。深度掉电模式将不保持这种状态。如果引脚暂时没有被驱动，通常为中继模式，以防止引脚处于浮空状态（如果引脚处于一种不确定的状态，系统将潜在地启用一个有效电源）。

7.3.3 滞后作用

通过 IOCON 寄存器，可以将数字功能的输入缓冲区配置为带滞后功能或普通缓冲区（详情见 LPC111X 数据手册）。

如果外部引脚提供的电压 VDD(IO) 在 2.5 V 和 3.6 V 之间，滞后缓冲区可以被允许或禁止；如果 VDD(IO) 低于 2.5 V，那么在引脚处于输入模式的情况下必须禁止滞后缓冲区。

7.3.4 A/D 模式

在 A/D 模式中，数字接收器断开连接，用来为模数转换获取精确的输入电压。在控制带模拟功能管脚的 IOCON 寄存器中都可以选择该模式。如果选择了 A/D 模式，那么滞后和管脚模式设置都无效。

对于没有模拟功能的管脚，A/D 模式设置无效。

7.3.5 I²C 模式

如果寄存器 IOCON_PIO0_4 ([Table 66](#)) 和 IOCON_PIO0_5 ([Table 67](#)) 的 FUNC 位选择 I2C 功能，则 I2C 总线管脚可以配置为不同的 I2C 模式：

- 标准模式或快速 I2C 模式：具有输入干扰滤波器的特性（根据 I2C 总线的规范包括了一个开漏输出）。
- FM+（Fast-mode Plus）模式：具有输入干扰滤波器的特性（根据 I2C 总线的标准包括了一个开漏输出）。在该模式下，引脚吸收大电流。
- 标准开漏 I/O 功能没有输入干扰滤波器。

注释：当引脚作为 GPIO 引脚时，该引脚要么被选择为 I2C 标准模式 / 快速 I2C 模式，要么被选择为标准 I/O 模式。

7.3.6 开漏模式

当选择输出时，要么通过在 FUNC 位域选择特定的功能，要么通过为引脚配置 GPIO 功能以在 GPIO_DIR 寄存器中置 1 和将 OD 位置 1，来选择开漏操作，也就是说，1 禁止高驱动晶体管。该操作对主 I2C 引脚无影响。

注意：开漏模式并不是对所有部分都可用（见 [Section 7.1](#)）。

7.4 寄存器描述

I/O 配置寄存器控制 PIO 的端口引脚、所有外设和各功能模块的输入输出、I2C 总线引脚和 ADC 输入引脚。

每一个端口引脚 PIO_n_m 都由一个 IOCON 寄存器去控制该引脚的功能特性和电气特性。

一些输入功能 (SCK0、DSR0、DCD0 和 RI0) 复用几个物理引脚，通过 IOCON_LOC 寄存器来为这些功能选择引脚位置。

注意：Table 54 按内存中的位置顺序列出 IOCON 寄存器，IOCON 寄存器在内存中的位置顺序。对应于 LQFP48 封装中的物理引脚编号顺序，从左上角引脚 1 (PIO2-6) 开始。Table 55 则按端口号顺序列出了 IOCON 寄存器。

IOCON 位置配置寄存器选择一个物理引脚来实现复用功能。

备注：一旦选择了引脚的位置，必须在相应的 IOCON 寄存器配置该引脚功能来允许该引脚的功能。

Table 54. 寄存器概览：I/O 配置 (基址 0x4004 4000)

寄存器名	访问方式	偏移地址	描述	复位值	参考
IOCON_PIO2_6	R/W	0x000	为以下引脚配置 I/O: PIO2_6	0xD0	Table 56
-	R/W	0x004	保留	-	-
IOCON_PIO2_0	R/W	0x008	为以下引脚配置 I/O: PIO2_0/DTR/SSEL1	0xD0	Table 57
IOCON_RESET_PIO0_0	R/W	0x00C	为以下引脚配置 I/O: RESET/PIO0_0	0xD0	Table 58
IOCON_PIO0_1	R/W	0x010	为以下引脚配置 I/O: PIO0_1/CLKOUT/CT32B0_MAT2	0xD0	Table 56
IOCON_PIO1_8	R/W	0x014	为以下引脚配置 I/O: PIO1_8/CT16B1_CAP0	0xD0	Table 60
-	R/W	0x018	保留	-	-
IOCON_PIO0_2	R/W	0x01C	为以下引脚配置 I/O: PIO0_2/SSEL0/CT16B0_CAP0	0xD0	Table 61
IOCON_PIO2_7	R/W	0x020	为以下引脚配置 I/O: PIO2_7	0xD0	Table 62
IOCON_PIO2_8	R/W	0x024	为以下引脚配置 I/O: PIO2_8	0xD0	Table 63
IOCON_PIO2_1	R/W	0x028	为以下引脚配置 I/O: PIO2_1/DSR/SCK1	0xD0	Table 64
IOCON_PIO0_3	R/W	0x02C	为以下引脚配置 I/O: PIO0_3	0xD0	Table 65
IOCON_PIO0_4	R/W	0x030	为以下引脚配置 I/O: PIO0_4/SCL	0x00	Table 66
IOCON_PIO0_5	R/W	0x034	为以下引脚配置 I/O: PIO0_5/SDA	0x00	Table 67
IOCON_PIO1_9	R/W	0x038	为以下引脚配置 I/O: PIO1_9/CT16B1_MAT0	0xD0	Table 68
IOCON_PIO3_4	R/W	0x03C	为以下引脚配置 I/O: PIO3_4	0xD0	Table 69
IOCON_PIO2_4	R/W	0x040	为以下引脚配置 I/O: PIO2_4	0xD0	Table 70
IOCON_PIO2_5	R/W	0x044	为以下引脚配置 I/O: PIO2_5	0xD0	Table 71
IOCON_PIO3_5	R/W	0x048	为以下引脚配置 I/O: PIO3_5	0xD0	Table 72
IOCON_PIO0_6	R/W	0x04C	为以下引脚配置 I/O: PIO0_6/SCK0	0xD0	Table 73
IOCON_PIO0_7	R/W	0x050	为以下引脚配置 I/O: PIO0_7/CTS	0xD0	Table 74
IOCON_PIO2_9	R/W	0x054	为以下引脚配置 I/O: PIO2_9	0xD0	Table 75
IOCON_PIO2_10	R/W	0x058	为以下引脚配置 I/O: PIO2_10	0xD0	Table 76

Table 54. 寄存器概览 : I/O 配置 (基址 0x4004 4000)

寄存器名	访问方式	偏移地址	描述	复位值	参考
IOCON_PIO2_2	R/W	0x05C	为以下引脚配置 I/O: PIO2_2/ $\overline{\text{DCD}}$ /MISO1	0xD0	Table 77
IOCON_PIO0_8	R/W	0x060	为以下引脚配置 I/O: PIO0_8/MISO0/CT16B0_MAT0	0xD0	Table 78
IOCON_PIO0_9	R/W	0x064	为以下引脚配置 I/O: PIO0_9/MOSI0/CT16B0_MAT1	0xD0	Table 79
IOCON_SWCLK_PIO0_10	R/W	0x068	为以下引脚配置 I/O: SWCLK/PIO0_10/ SCK0/CT16B0_MAT2	0xD0	Table 80
IOCON_PIO1_10	R/W	0x06C	为以下引脚配置 I/O: PIO1_10/AD6/CT16B1_MAT1	0xD0	Table 81
IOCON_PIO2_11	R/W	0x070	为以下引脚配置 I/O: PIO2_11/SCK0	0xD0	Table 82
IOCON_R_PIO0_11	R/W	0x074	为以下引脚配置 I/O: R/PIO0_11/AD0/CT32B0_MAT3	0xD0	Table 83
IOCON_R_PIO1_0	R/W	0x078	为以下引脚配置 I/O: R/PIO1_0/AD1/CT32B1_CAP0	0xD0	Table 84
IOCON_R_PIO1_1	R/W	0x07C	为以下引脚配置 I/O: R/PIO1_1/AD2/CT32B1_MAT0	0xD0	Table 85
IOCON_R_PIO1_2	R/W	0x080	为以下引脚配置 I/O: R/PIO1_2/AD3/CT32B1_MAT1	0xD0	Table 86
IOCON_PIO3_0	R/W	0x084	为以下引脚配置 I/O: PIO3_0/ $\overline{\text{DTR}}$	0xD0	Table 87
IOCON_PIO3_1	R/W	0x088	为以下引脚配置 I/O: PIO3_1/ $\overline{\text{DSR}}$	0xD0	Table 88
IOCON_PIO2_3	R/W	0x08C	为以下引脚配置 I/O: PIO2_3/ $\overline{\text{RI}}$ /MOSI1	0xD0	Table 89
IOCON_SWDIO_PIO1_3	R/W	0x090	为以下引脚配置 I/O: SWDIO/PIO1_3/AD4/CT32B1_MAT2	0xD0	Table 90
IOCON_PIO1_4	R/W	0x094	为以下引脚配置 I/O: PIO1_4/AD5/CT32B1_MAT3	0xD0	Table 91
IOCON_PIO1_11	R/W	0x098	为以下引脚配置 I/O: PIO1_11/AD7	0xD0	Table 92
IOCON_PIO3_2	R/W	0x09C	为以下引脚配置 I/O: PIO3_2/ $\overline{\text{DCD}}$	0xD0	Table 93
IOCON_PIO1_5	R/W	0x0A0	为以下引脚配置 I/O: PIO1_5/RTS/CT32B0_CAP0	0xD0	Table 94
IOCON_PIO1_6	R/W	0x0A4	为以下引脚配置 I/O: PIO1_6/RXD/CT32B0_MAT0	0xD0	Table 95
IOCON_PIO1_7	R/W	0x0A8	为以下引脚配置 I/O: PIO1_7/TXD/CT32B0_MAT1	0xD0	Table 96
IOCON_PIO3_3	R/W	0x0AC	为以下引脚配置 I/O: PIO3_3/ $\overline{\text{RI}}$	0xD0	Table 97
IOCON_SCK_LOC	R/W	0x0B0	SCK 引脚位置选择寄存器	0x00	Table 98
IOCON_DSR_LOC	R/W	0x0B4	$\overline{\text{DSR}}$ 引脚位置选择寄存器	0x00	Table 99
IOCON_DCD_LOC	R/W	0x0B8	$\overline{\text{DCD}}$ 引脚位置选择寄存器	0x00	Table 100
IOCON_RI_LOC	R/W	0x0BC	$\overline{\text{RI}}$ 引脚位置寄存器	0x00	Table 101

Table 55. I/O configuration registers ordered by port number

端口引脚	寄存器名	LPC1111/ 12/13/14 HVQFN33	LPC1114 PLCC44	LPC1113/14 LQFP48	LPC11C12/ C14 LQFP48	LPC11C22/ C24 LQFP48	Reference
PIO0_0	IOCON_RESET_PIO0_0	是	是	是	是	是	Table 58
PIO0_1	IOCON_PIO0_1	是	是	是	是	是	Table 56
PIO0_2	IOCON_PIO0_2	是	是	是	是	是	Table 61
PIO0_3	IOCON_PIO0_3	是	是	是	是	是	Table 65
PIO0_4	IOCON_PIO0_4	是	是	是	是	是	Table 66
PIO0_5	IOCON_PIO0_5	是	是	是	是	是	Table 67
PIO0_6	IOCON_PIO0_6	是	是	是	是	是	Table 73
PIO0_7	IOCON_PIO0_7	是	是	是	是	是	Table 74
PIO0_8	IOCON_PIO0_8	是	是	是	是	是	Table 78
PIO0_9	IOCON_PIO0_9	是	是	是	是	是	Table 79
PIO0_10	IOCON_SWCLK_PIO0_10	是	是	是	是	是	Table 80
PIO0_11	IOCON_R_PIO0_11	是	是	是	是	是	Table 83
PIO1_0	IOCON_R_PIO1_0	是	是	是	是	是	Table 84
PIO1_1	IOCON_R_PIO1_1	是	是	是	是	是	Table 85
PIO1_2	IOCON_R_PIO1_2	是	是	是	是	是	Table 86
PIO1_3	IOCON_SWDIO_PIO1_3	是	是	是	是	是	Table 90
PIO1_4	IOCON_PIO1_4	是	是	是	是	是	Table 91
PIO1_5	IOCON_PIO1_5	是	是	是	是	是	Table 94
PIO1_6	IOCON_PIO1_6	是	是	是	是	是	Table 95
PIO1_7	IOCON_PIO1_7	是	是	是	是	是	Table 96
PIO1_8	IOCON_PIO1_8	是	是	是	是	是	Table 60
PIO1_9	IOCON_PIO1_9	是	是	是	是	否	Table 68
PIO1_10	IOCON_PIO1_10	是	是	是	是	是	Table 81
PIO1_11	IOCON_PIO1_11	是	是	是	是	是	Table 92
PIO2_0	IOCON_PIO2_0	是	是	是	是	是	Table 57
PIO2_1	IOCON_PIO2_1	否	是	是	是	是	Table 64
PIO2_2	IOCON_PIO2_2	否	是	是	是	是	Table 77
PIO2_3	IOCON_PIO2_3	否	是	是	是	是	Table 89
PIO2_4	IOCON_PIO2_4	否	是	是	是	否	Table 70
PIO2_5	IOCON_PIO2_5	否	是	是	是	否	Table 71
PIO2_6	IOCON_PIO2_6	否	是	是	是	是	Table 56
PIO2_7	IOCON_PIO2_7	否	是	是	是	是	Table 62
PIO2_8	IOCON_PIO2_8	否	是	是	是	是	Table 63
PIO2_9	IOCON_PIO2_9	否	是	是	是	否	Table 75
PIO2_10	IOCON_PIO2_10	否	是	是	是	是	Table 76
PIO2_11	IOCON_PIO2_11	否	是	是	是	是	Table 82
PIO3_0	IOCON_PIO3_0	否	否	是	是	是	Table 87
PIO3_1	IOCON_PIO3_1	否	否	是	是	是	Table 88
PIO3_2	IOCON_PIO3_2	是	否	是	是	是	Table 93

Table 55. I/O configuration registers ordered by port number

端口引脚	寄存器名	LPC1111/ 12/13/14	LPC1114	LPC1113/14	LPC11C12/ C14	LPC11C22/ C24	Reference
		HVQFN33	PLCC44	LQFP48	LQFP48	LQFP48	
PIO3_3	IOCON_PIO3_3	否	否	是	是	是	Table 97
PIO3_4	IOCON_PIO3_4	是	是	是	否	否	Table 69
PIO3_5	IOCON_PIO3_5	是	是	是	否	否	Table 72
-	IOCON_SCK_LOC	是 (SCKLOC = 01 保留)	是	是	是	是	Table 98
-	IOCON_DSR_LOC	否	否	是	是	是	Table 99
-	IOCON_DCD_LOC	否	否	是	是	是	Table 100
-	IOCON_RI_LOC	否	否	是	是	是	Table 101

7.4.1 IOCON_PIO2_6

Table 56. IOCON_PIO2_6 寄存器 (IOCON_PIO2_6, 地址 0x4004 4000) 位域描述

位	符号	值	描述	复位 值
2:0	FUNC		选择引脚功能，其他所有值保留	000
		0x0	选择功能 PIO2_6	
4:3	MODE		选择功能模式 (片内上拉 / 下拉电阻控制)	10
		0x0	无效 (无下拉 / 上拉电阻使能)	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.2 IOCON_PIO2_0

Table 57. IOCON_PIO2_0 寄存器 (IOCON_PIO2_0, 地址 0x4004 4008) 为描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能，其他所有值保留	000
		0x0	选择功能 PIO2_0	
		0x1	选择功能 $\overline{\text{DTR}}$.	
		0x2	选择功能 SSEL1	
4:3	MODE		选择功能模式 (片内上拉 / 下拉电阻控制)	10
		0x0	无效 (无下拉 / 上拉电阻允许)	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.3 IOCON_PIO_RESET_PIO0_0

Table 58. IOCON_RESET_PIO0_0 寄存器 (IOCON_RESET_PIO0_0, 地址 0x4004 400C) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能，其他所有值保留	000
		0x0	选择功能 $\overline{\text{RESET}}$	
		0x1	选择功能 PIO0_0	
4:3	MODE		选择功能模式 (片内上拉 / 下拉电阻控制)	10
		0x0	无效 (无下拉 / 上拉电阻允许)	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011

Table 58. IOCON_RESET_PIO0_0 寄存器 (IOCON_RESET_PIO0_0, 地址 0x4004 400C) 位描述

位	符号	值	描述	复位值
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.4 IOCON_PIO0_1

Table 59. IOCON_PIO0_1 寄存器 (IOCON_PIO0_1, 地址 0x4004 4010) 位描述

位	复位	值	描述	复位值
2:0	FUNC		选择引脚功能. 其他所有值保留	000
		0x0	选择功能 PIO0_1	
		0x1	选择功能 CLKOUT	
		0x2	选择功能 CT32B0_MAT2	
4:3	MODE		选择功能模式 (片内上拉 / 下拉电阻控制)	10
		0x0	无效 (无下拉 / 上拉电阻允许)	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.5 IOCON_PIO1_8

Table 60. IOCON_PIO1_8 寄存器 (IOCON_PIO1_8, 地址 0x4004 4014) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能. 其他所有值保留	000
		0x0	选择功能 PIO1_8	
		0x1	选择功能 CT16B1_CAP0	

Table 60. IOCON_PIO1_8 寄存器 (IOCON_PIO1_8, 地址 0x4004 4014) 位描述 ...continued

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
		0x0	无效 (无下拉 / 上拉电阻允许)	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后模式	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1 .	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.6 IOCON_PIO0_2

Table 61. IOCON_PIO0_2 寄存器 (IOCON_PIO0_2, 地址 0x4004 401C) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能, 其他所有值保留	000
		0x0	选择功能 PIO0_2	
		0x1	选择功能 SSEL0	
		0x2	选择功能 CT16B0_CAP0	
4:3	MODE		选择功能模式 (片内上拉 / 下拉电阻控制)	10
		0x0	无效 (无下拉 / 上拉电阻允许)	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.7 IOCON_PIO2_7

Table 62. IOCON_PIO2_7 寄存器 (IOCON_PIO2_7, 地址 0x4004 4020) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO2_7	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.8 IOCON_PIO2_8

Table 63. IOCON_PIO2_8 寄存器 (IOCON_PIO2_8, 地址 0x4004 4024) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO2_8	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.9 IOCON_PIO2_1

Table 64. IOCON_PIO2_1 寄存器 (IOCON_PIO2_1, 地址 0x4004 4028) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO2_1	
		0x1	选择功能 $\overline{\text{DSR}}$.	
		0x2	选择功能 SCK1	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.10 IOCON_PIO0_3

Table 65. IOCON_PIO0_3 寄存器 (IOCON_PIO0_3, 地址 0x4004 402C) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO0_3	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011

Table 65. IOCON_PIO0_3 寄存器 (IOCON_PIO0_3, 地址 0x4004 402C) 位描述

位	符号	值	描述	复位值
10	OD		选择伪开漏模式	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.11 IOCON_PIO0_4

Table 66. IOCON_PIO0_4 寄存器 (IOCON_PIO0_4, 地址 0x4004 4030) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能, 其他所有值保留	000
		0x0	选择功能 PIO0_4 (开漏引脚)	
		0x1	选择 I2C 功能 SCL (开漏引脚)	
7:3	-	-	保留	00000
9:8	I2CMODE		选择 I2C 模式。选择标准模式 (I2CMODE = 00, 默认值), 或者标准 I/O 功能 (I2CMODE = 01), 如果引脚功能是 GPIO (FUNC = 000)	00
		0x0	标准模式 / 快速 I2C 模式	
		0x1	标准 I/O 功能	
		0x2	快速模式 Plus 的 I2C	
		0x3	保留	
31:10	-	-	保留	-

7.4.12 IOCON_PIO0_5

Table 67. IOCON_PIO0_5 寄存器 (IOCON_PIO0_5, 地址 0x4004 4034) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能, 其他所有值保留	000
		0x0	选择功能 PIO0_5 (开漏引脚)	
		0x1	选择 I2C 功能 SDA (开漏引脚)	
7:3	-	-	保留	00000
9:8	I2CMODE		选择 I2C 模式。选择标准模式 (I2CMODE = 00, 默认值), 或者标准 I/O 功能 (I2CMODE = 01), 如果引脚功能是 GPIO (FUNC = 000)	00
		0x0	标准模式 / 快速 I2C 模式	
		0x1	标准 I/O 功能	
		0x2	快速模式 Plus 的 I2C	
		0x3	保留	
31:10	-	-	保留	-

7.4.13 IOCON_PIO1_9

备注：详见 [Section 7.1](#)

Table 68. IOCON_PIO1_9 寄存器 (IOCON_PIO1_9, 地址 0x4004 4038) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO1_9	
		0x1	选择功能 CT16B1_MAT0	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.14 IOCON_PIO3_4

备注：详见 [Section 7.1](#)

Table 69. IOCON_PIO3_4 寄存器 (IOCON_PIO3_4, 地址 0x4004 403C) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO3_4	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.15 IOCON_PIO2_4

备注：详见 [Section 7.1](#)

Table 70. IOCON_PIO2_4 寄存器 (IOCON_PIO2_4, 地址 0x4004 4040) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO2_4	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.16 IOCON_PIO2_5

备注：详见 [Section 7.1](#)

Table 71. IOCON_PIO2_5 寄存器 (IOCON_PIO2_5, 地址 0x4004 4044) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO2_5	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011

Table 71. IOCON_PIO2_5 寄存器 (IOCON_PIO2_5, 地址 0x4004 4044) 位描述

位	符号	值	描述	复位值
10	OD		选择伪开漏模式	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.17 IOCON_PIO3_5

备注：详见 [Section 7.1](#)**Table 72. IOCON_PIO3_5 寄存器 (IOCON_PIO3_5, 地址 0x4004 4048) 位描述**

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO3_5	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.18 IOCON_PIO0_6

Table 73. IOCON_PIO0_6 寄存器 (IOCON_PIO0_6, 地址 0x4004 404C) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO0_6	
		0x1	保留	
		0x2	选择功能 SCK0 (只有当按 Table 98 选择 PIO0_6/SCK0 引脚时)	

Table 73. IOCON_PIO0_6 寄存器 (IOCON_PIO0_6, 地址 0x4004 404C) 位描述

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
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7. 4. 19 IOCON_PIO0_7

Table 74. IOCON_PIO0_7 寄存器 (IOCON_PIO0_7, 地址 0x4004 4050) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO0_7	
		0x1	选择功能 CTS	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7. 4. 20 IOCON_PIO2_9

备注：详见 [Section 7.1](#)

Table 75. IOCON_PIO2_9 寄存器 (IOCON_PIO2_9, 地址 0x4004 4054) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO2_9	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.21 IOCON_PIO2_10

Table 76. IOCON_PIO2_10 寄存器 (IOCON_PIO2_10, 地址 0x4004 4058) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO2_10	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.22 IOCON_PIO2_2

Table 77. IOCON_PIO2_2 寄存器 (IOCON_PIO2_2, 地址 0x4004 405C) 位描述

位	符号	值	位描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO2_2	
		0x1	选择功能 \overline{DCD}	
		0x2	选择功能 MISO1	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.23 IOCON_PIO0_8

Table 78. IOCON_PIO0_8 寄存器 (IOCON_PIO0_8, 地址 0x4004 4060) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO0_8	
		0x1	选择功能 MISO0	
		0x2	选择功能 CT16B0_MAT0	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011

Table 78. IOCON_PIO0_8 寄存器 (IOCON_PIO0_8, 地址 0x4004 4060) 位描述

位	符号	值	描述	复位值
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7. 4. 24 IOCON_PIO0_9

Table 79. IOCON_PIO0_9 寄存器 (IOCON_PIO0_9, 地址 0x4004 4064) 位描述

位	符号	值	位描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO0_9	
		0x1	选择功能 MOSI0	
		0x2	选择功能 CT16B0_MAT1	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
5	HYS	0x3	中继模式	0
			滞后作用	
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7. 4. 25 IOCON_SWCLK_PIO0_10

Table 80. IOCON_SWCLK_PIO0_10 寄存器 (IOCON_SWCLK_PIO0_10, 地址 0x4004 4068) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 SWCLK	
		0x1	选择功能 PIO0_10	
		0x2	选择功能 SCK0 (只有当按 Table 98 选择引脚 SWCLK/PIO0_10/SCK0/CT16B0_MAT2 时)	
		0x3	选择功能 CT16B0_MAT2	

Table 80. IOCON_SWCLK_PIO0_10 寄存器 (IOCON_SWCLK_PIO0_10, 地址 0x4004 4068) 位描述 ...continued

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
		0x0	无效 (无下拉 / 上拉电阻允许)	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7. 4. 26 IOCON_PIO1_10

Table 81. IOCON_PIO1_10 寄存器 (IOCON_PIO1_10, 地址 0x4004 406C) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能, 其他所有值保留	000
		0x0	选择功能 PIO1_10	
		0x1	选择功能 AD6	
		0x2	选择功能 CT16B1_MAT1	
4:3	MODE		选择功能模式 (片内上拉 / 下拉电阻控制)	10
		0x0	无效 (无下拉 / 上拉电阻允许)	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
6	-	-	保留	1
7	ADMODE		选择 模拟 / 数字 模式	1
		0	模拟输入模式	
		1	数字输入模式	
9:8	-	-	保留	00

Table 81. IOCON_PIO1_10 寄存器 (IOCON_PIO1_10, 地址 0x4004 406C) 位描述

位	符号	值	描述	复位值
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.27 IOCON_PIO2_11

Table 82. IOCON_PIO2_11 寄存器 (IOCON_PIO2_11, 地址 0x4004 4070) bit description

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO2_11	
		0x1	选择功能 SCK0 (只有当按照 Table 98 选择 PIO2_11/SCK0 引脚时)	
4:3	MODE		选择功能模式 (片内上拉 / 下拉电阻控制)	10
		0x0	无效 (无下拉 / 上拉电阻允许)	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.28 IOCON_R_PIO0_11

Table 83. IOCON_R_PIO0_11 寄存器 (IOCON_R_PIO0_11, 地址 0x4004 4074) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 R. 此功能保留 选择以下功能的其中一个	
		0x1	选择功能 PIO0_11	
		0x2	选择功能 AD0	
		0x3	选择功能 CT32B0_MAT3	

Table 83. IOCON_R_PIO0_11 寄存器 (IOCON_R_PIO0_11, 地址 0x4004 4074) 位描述

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
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
6	-	-	保留	1
7	ADMODE		选择模拟 / 数字模式	1
		0	模拟输入模式	
		1	数字功能模式	
9:8	-	-	保留	00
10	OD		选择伪开漏模式	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7. 4. 29 IOCON_R_PIO1_0

Table 84. IOCON_R_PIO1_0 寄存器 (IOCON_R_PIO1_0, 地址 0x4004 4078) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 R，此功能保留 选择以下功能的其中一个	
		0x1	选择功能 PIO1_0	
		0x2	选择功能 AD1	
		0x3	选择功能 CT32B1_CAP0	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
6	-	-	保留	1

Table 84. IOCON_R_PIO1_0 寄存器 (IOCON_R_PIO1_0, 地址 0x4004 4078) 位描述

位	符号	值	描述	复位值
7	ADMODE		选择模拟 / 数字模式	1
		0	模拟输入模式	
		1	数字功能模式	
9:8	-	-	保留	00
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7. 4. 30 IOCON_R_PIO1_1

Table 85. IOCON_R_PIO1_1 寄存器 (IOCON_R_PIO1_1, 地址 0x4004 407C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 R，此功能保留 选择以下功能的其中一个	
		0x1	选择功能 PIO1_1	
		0x2	选择功能 AD2	
		0x3	选择功能 CT32B1_MAT0	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
6	-	-	保留	1
7	ADMODE		选择模拟 / 数字模式	1
		0	模拟输入模式	
		1	数字功能模式	
9:8	-	-	保留	00
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.31 IOCON_R_PIO1_2

Table 86. IOCON_R_PIO1_2 寄存器 (IOCON_R_PIO1_2, 地址 0x4004 4080) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 R，此功能保留 选择以下功能的其中一个	
		0x1	选择功能 PIO1_2	
		0x2	选择功能 AD3	
		0x3	选择功能 CT32B1_MAT1	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁能	
		1	使能	
6	-	-	保留	1
7	ADMODE		选择模拟 / 数字模式	1
		0	模拟输入模式	
		1	数字功能模式	
9:8	-	-	保留	00
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.32 IOCON_PIO3_0

Table 87. IOCON_PIO3_0 寄存器 (IOCON_PIO3_0, 地址 0x4004 4084) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO3_0	
		0x1	选择功能 $\overline{\text{DTR}}$	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	

Table 87. IOCON_PIO3_0 寄存器 (IOCON_PIO3_0, 地址 0x4004 4084) 位描述 ...continued

位	符号	值	描述	复位值
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.33 IOCON_PIO3_1

Table 88. IOCON_PIO3_1 寄存器 (IOCON_PIO3_1, 地址 0x4004 4088) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO3_1	
		0x1	选择功能 $\overline{\text{DSR}}$	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.34 IOCON_PIO2_3

Table 89. IOCON_PIO2_3 寄存器 (IOCON_PIO2_3, 地址 0x4004 408C) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 PIO2_3	
		0x1	选择功能 $\overline{\text{Ri}}$	
		0x2	选择功能 MOSI1	

Table 89. IOCON_PIO2_3 寄存器 (IOCON_PIO2_3, 地址 0x4004 408C) 位描述

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
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.35 IOCON_SWDIO_PIO1_3

Table 90. IOCON_SWDIO_PIO1_3 寄存器 (IOCON_SWDIO_PIO1_3, 地址 0x4004 4090) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择管脚功能，其他所有值保留	000
		0x0	选择功能 SWDIO	
		0x1	选择功能 PIO1_3	
		0x2	选择功能 AD4	
		0x3	选择功能 CT32B1_MAT2	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
6	-	-	保留	1
7	ADMODE		选择模拟 / 数字模式	1
		0	模拟输入模式	
		1	数字功能模式	
9:8	-	-	保留	00

Table 90. IOCON_SWDIO_PIO1_3 寄存器(IOCON_SWDIO_PIO1_3, 地址 0x4004 4090) 位描述
...continued

位	符号	值	描述	复位值
10	OD		选择伪开漏模式	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7. 4. 36 IOCON_PIO1_4

Table 91. IOCON_PIO1_4 寄存器 (IOCON_PIO1_4, 地址 0x4004 4094) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能，不管 FUNC 的值如何，如果 LPC111x/LPC11Cxx 处于深度掉电模式该引脚功能作为 WAKEUP 引脚，其他所有值保留	000
		0x0	选择功能 PIO1_4	
		0x1	选择功能 AD5	
		0x2	选择功能 CT32B1_MAT3	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
5	HYS		中继模式	0
		0	禁止	
		1	允许	
6	-	-	保留	1
7	ADMODE		选择模拟 / 数字模式	1
		0	模拟输入模式	
		1	数字功能模式	
9:8	-	-	保留	00
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.37 IOCON_PIO1_11

Table 92. IOCON_PIO1_11 寄存器 (IOCON_PIO1_11 地址 0x4004 4098) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能，其他所有值保留	000
		0x0	选择功能 PIO1_11	
		0x1	选择功能 AD7	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
6	-	-	保留	1
7	ADMODE		选择模拟 / 数字模式	1
		0	模拟输入模式	
		1	数字功能模式	
9:8	-	-	保留	00
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.38 IOCON_PIO3_2

Table 93. IOCON_PIO3_2 寄存器 (IOCON_PIO3_2, 地址 0x4004 409C) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能，其他所有值保留	000
		0x0	选择功能 PIO3_2	
		0x1	选择功能 \overline{DCD}	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011

Table 93. IOCON_PIO3_2 寄存器 (IOCON_PIO3_2, 地址 0x4004 409C) 位描述

位	符号	值	描述	复位值
10	OD		选择伪开漏模式	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.39 IOCON_PIO1_5

Table 94. IOCON_PIO1_5 寄存器 (IOCON_PIO1_5, a 地址 0x4004 40A0) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能，其他所有值保留	000
		0x0	选择功能 PIO1_5	
		0x1	选择功能 $\overline{\text{RTS}}$	
		0x2	选择功能 CT32B0_CAP0	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.40 IOCON_PIO1_6

Table 95. IOCON_PIO1_6 寄存器 (IOCON_PIO1_6, 地址 0x4004 40A4) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能，其他所有值保留	000
		0x0	选择功能 PIO1_6	
		0x1	选择功能 RXD	
		0x2	选择功能 CT32B0_MAT0	

Table 95. IOCON_PIO1_6 寄存器 (IOCON_PIO1_6, 地址 0x4004 40A4) 位描述 ...continued

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
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.41 IOCON_PIO1_7

Table 96. IOCON_PIO1_7 寄存器 (IOCON_PIO1_7, 地址 0x4004 40A8) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能，其他所有值保留	000
		0x0	选择功能 PIO1_7	
		0x1	选择功能 TXD	
		0x2	选择功能 CT32B0_MAT1	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.42 IOCON_PIO3_3

Table 97. IOCON_PIO3_3 寄存器 (IOCON_PIO3_3, 地址 0x4004 40AC) 位描述

位	符号	值	描述	复位值
2:0	FUNC		选择引脚功能，其他所有值保留	000
		0x0	选择功能 PIO3_3	
		0x1	选择功能 $\overline{R}I$	
4:3	MODE		选择功能模式（片内上拉 / 下拉电阻控制）	10
		0x0	无效（无下拉 / 上拉电阻允许）	
		0x1	下拉电阻允许	
		0x2	上拉电阻允许	
		0x3	中继模式	
5	HYS		滞后作用	0
		0	禁止	
		1	允许	
9:6	-	-	保留	0011
10	OD		选择伪开漏模式 详见 Section 7.1	0
		0	标准 GPIO 输出	
		1	开漏输出	
31:11	-	-	保留	-

7.4.43 IOCON_SCK_LOC

Table 98. IOCON_SCK 位置寄存器 (IOCON_SCK_LOC, 地址 0x4004 40B0) 位描述

位	符号	值	描述	复位值
1:0	SCKLOC		选择 SCK0 引脚的位置	00
		0x0	在管脚位置 SWCLK/PIO0_10/SCK0/CT16B0_MAT2 选择 SCK0 功能 (见 Table 80)	
		0x1	在管脚位置 PIO2_11/SCK0 选择 SCK0 功能 (见 Table 82)	
		0x2	在管脚位置 PIO0_6/SCK0 选择 SCK0 功能 (见 Table 73)	
		0x3	保留	
31:2	-	-	保留	-

7.4.44 IOCON_DSR_LOC

Table 99. IOCON $\overline{\text{DSR}}$ 位置寄存器 (IOCON_DSR_LOC, 地址 0x4004 40B4) 位描述

位	符号	值	描述	复位值
1:0	DSRLOC		选择 DSR0 管脚的位置	00
		0x0	在引脚位置 PIO2_1/ $\overline{\text{DSR}}$ /SCK1 选择 $\overline{\text{DSR}}$ 功能	
		0x1	在引脚位置 PIO3_1/ $\overline{\text{DSR}}$ 选择 $\overline{\text{DSR}}$ 功能	
		0x2	保留	
		0x3	保留	
31:2	-	-	保留	-

7.4.45 IOCON_DCD_LOC

Table 100. IOCON $\overline{\text{DCD}}$ 位置寄存器 (IOCON_DCD_LOC, 地址 0x4004 40B8) 位描述

位	符号	值	描述	复位值
1:0	DCDLOC		选择 DCD 管脚的位置	00
		0x0	在引脚位置 PIO2_2/ $\overline{\text{DCD}}$ /MISO1 选择 $\overline{\text{DCD}}$ 功能	
		0x1	在引脚位置 PIO3_2/ $\overline{\text{DCD}}$ 选择 $\overline{\text{DCD}}$ 功能	
		0x2	保留	
		0x3	保留	
31:2	-	-	保留	-

7.4.46 IOCON_RI_LOC

Table 101. IOCON $\overline{\text{RI}}$ 位置寄存器 (IOCON_RI_LOC, 地址 0x4004 40BC) 位描述

位	符号	值	描述	复位值
1:0	RILOC		选择 RI 管脚的位置	00
		0x0	在引脚位置 PIO2_3/ $\overline{\text{RI}}$ /MOSI1 选择 $\overline{\text{RI}}$ 功能	
		0x1	在引脚位置 PIO3_3/ $\overline{\text{RI}}$ 选择 $\overline{\text{RI}}$ 功能	
		0x2	保留	
		0x3	保留	
31:2	-	-	保留	-

8.1 如何阅读本章

LPC111x 有三种封装: LQFP48 (LPC1113, LPC1114), PLCC44 (LPC1114) 和 HVQFN33 (LPC1111, LPC1112, LPC1113, LPC1114)。

LPC11Cxx 为 LQFP48 封装。

Table 102. LPC111x/LPC11Cxx 引脚配置

Part		LQFP48	PLCC44	HVQFN33
LPC1111	引脚配置	-	-	Figure 12
	引脚描述	-	-	Table 105
LPC1112	引脚配置	-	-	Figure 12
	引脚描述	-	-	Table 105
LPC1113	引脚配置	Figure 10	-	Figure 12
	引脚描述	Table 103	-	Table 105
LPC1114	引脚配置	Figure 10	Figure 11	Figure 12
	引脚描述	Table 103	Table 104	Table 105
LPC11C12	引脚配置	Figure 13	-	-
	引脚描述	Table 103	-	-
LPC11C14	引脚配置	Figure 13	-	-
	引脚描述	Table 103	-	-
LPC11C22	引脚配置	Figure 14	-	-
	引脚描述	Table 106	-	-
LPC11C24	引脚配置	Figure 14	-	-
	引脚描述	Table 106	-	-

8.2 LPC111x 引脚配置

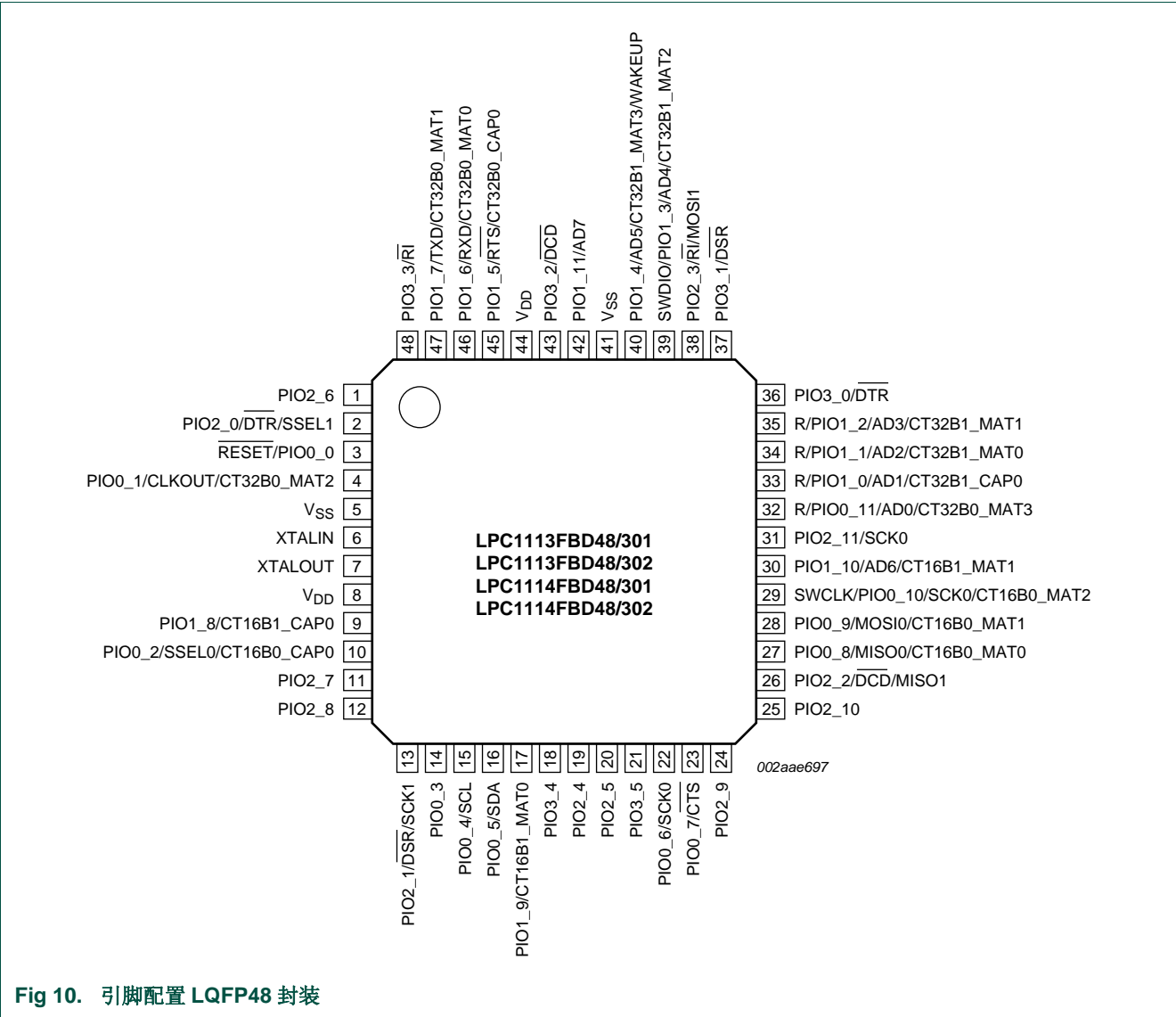


Fig 10. 引脚配置 LQFP48 封装

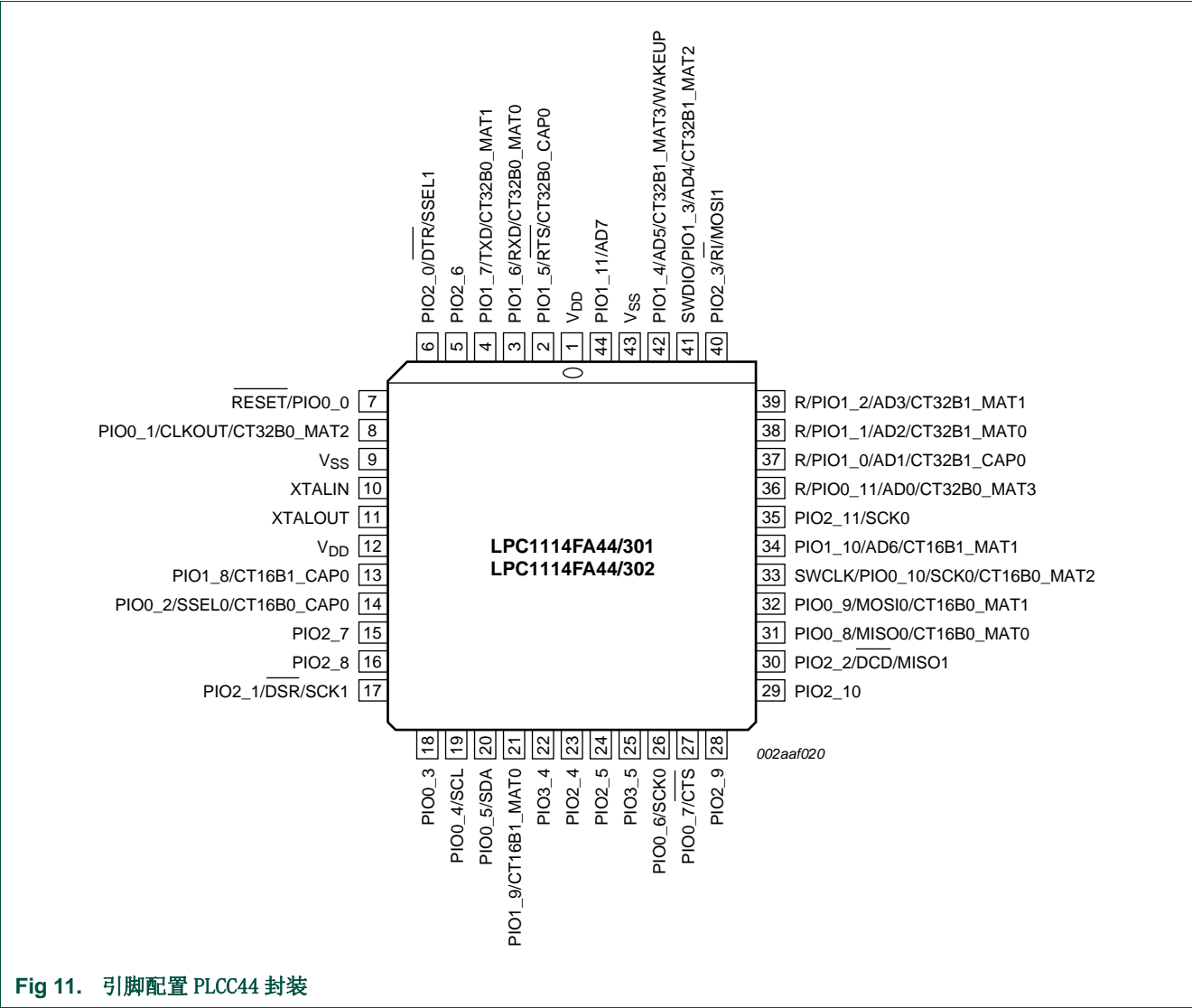


Fig 11. 引脚配置 PLCC44 封装

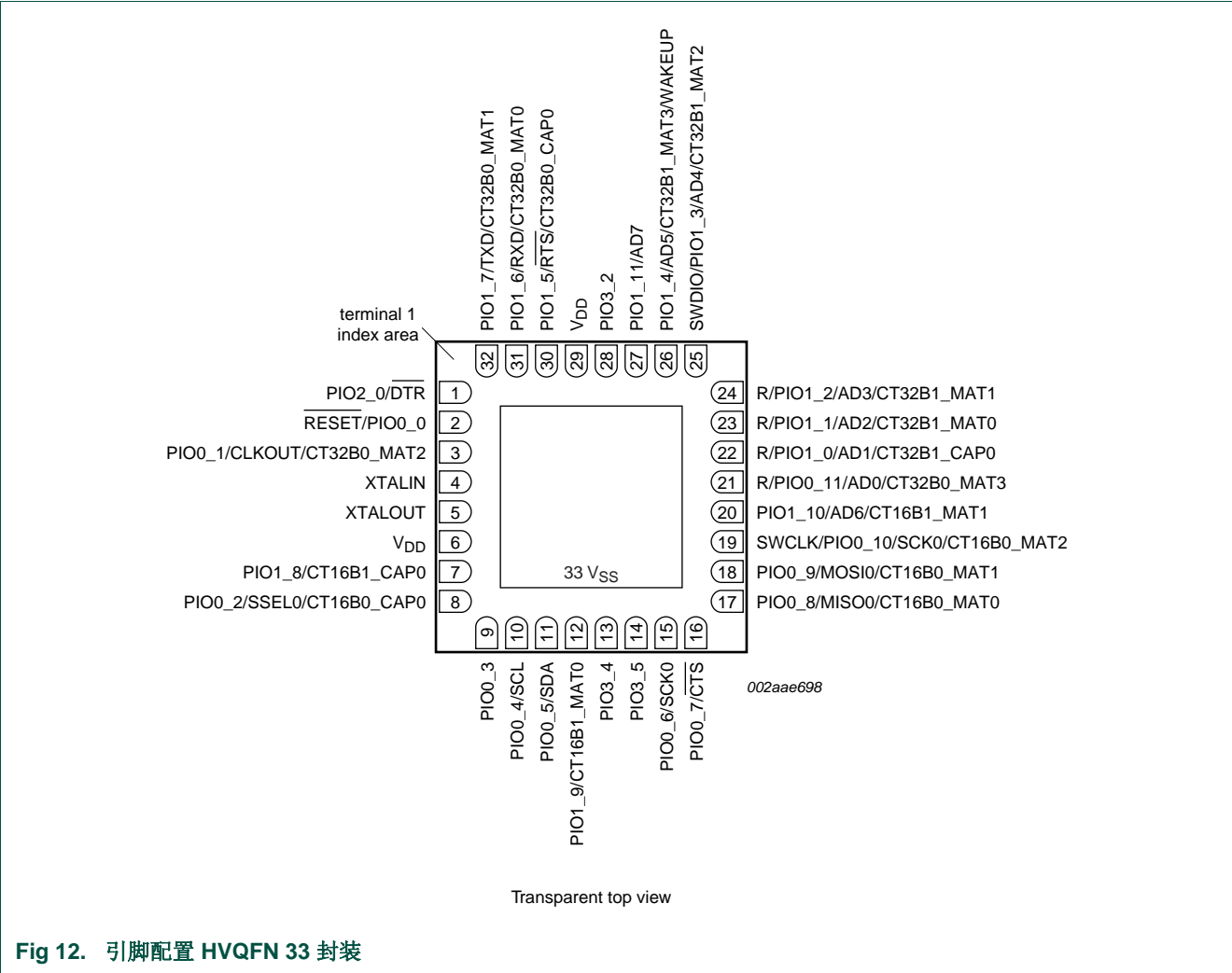


Fig 12. 引脚配置 HVQFN 33 封装

8.3 LPC11Cxx 引脚配置

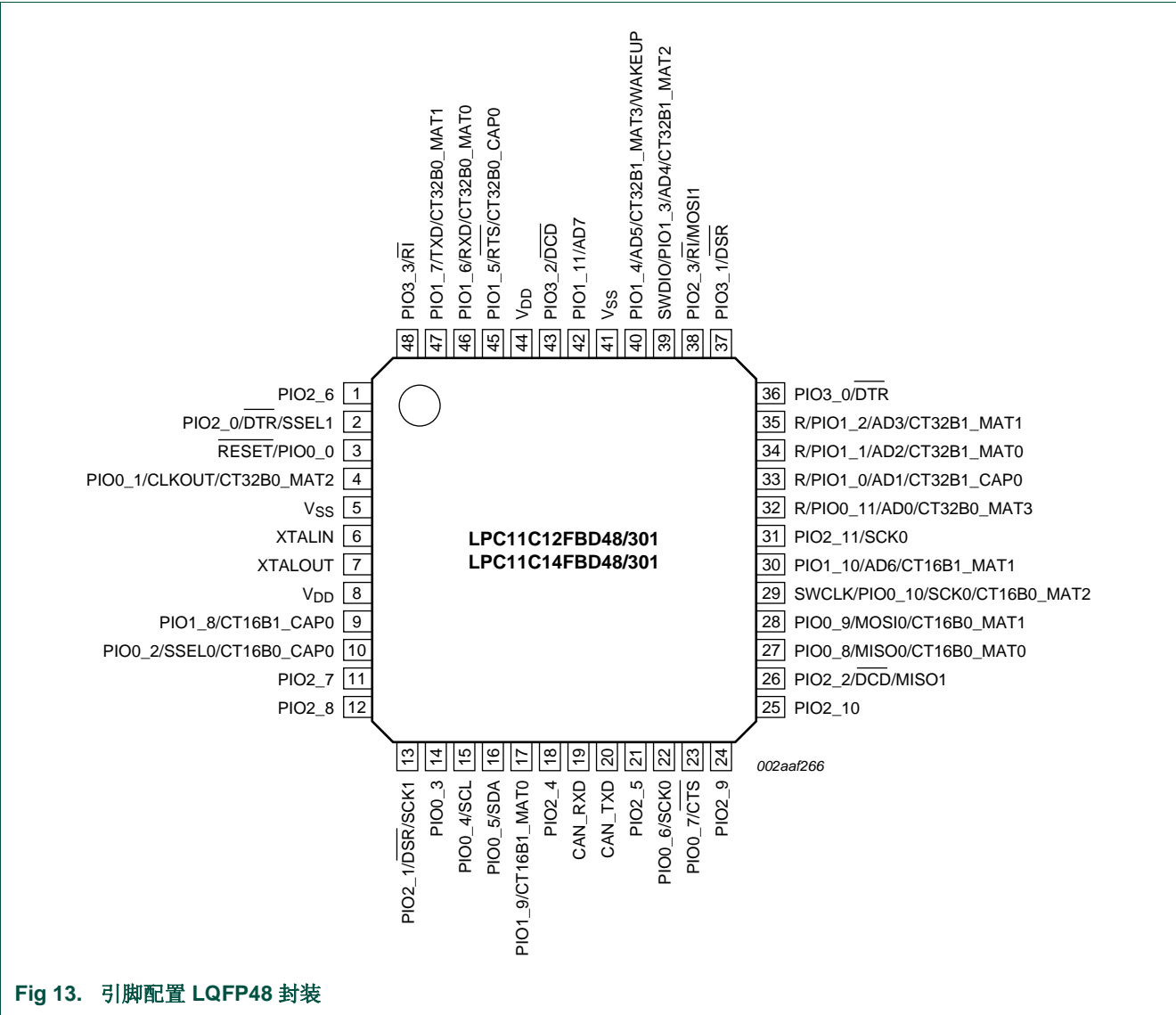
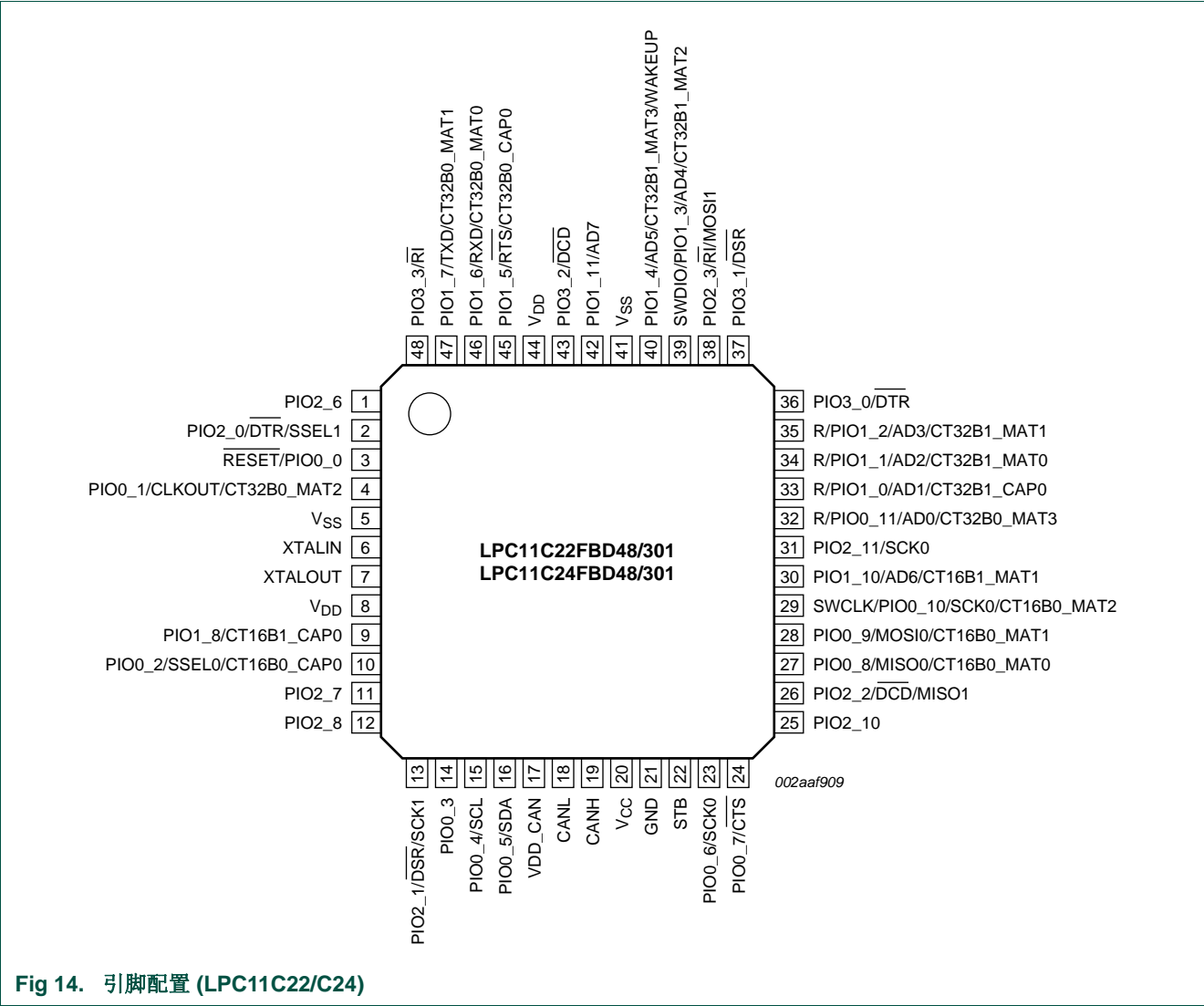


Fig 13. 引脚配置 LQFP48 封装



8.4 LPC111x/LPC11Cxx 引脚描述

Table 103. LPC1113/14 and LPC11C12/C14 引脚描述表 (LQFP48 封装)

符号	引脚	类型	描述
PIO0_0 to PIO0_11		I/O	Port 0 — Port 0 口是一个12位的端口，每一位都有独立的方向和功能控制。port 0 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。
RESET/PIO0_0	3 ^{[1][2]}	I	RESET — 外部复位输入：此引脚上的低电平会使设备复位，I/O 端口和外设复位为初始的默认状态，并使处理器从 0 地址开始执行。 —
		I/O	PIO0_0 — 通用数字输入 / 输出引脚。
PIO0_1/CLKOUT/ CT32B0_MAT2	4 ^{[3][2]}	I/O	PIO0_1 — 通用数字输入 / 输出引脚。复位时该引脚上有低电平将通过 UART (如果 PIO0_3 为高电位) 或 C_CAN (如果 PIO0_3 为低电位) 启动在线系统编程 (ISP) 命令处理程序。
		O	CLKOUT — 时钟输出引脚。
		O	CT32B0_MAT2 — 用于 32 位定时器 0 的匹配输出 2。

Table 103. LPC1113/14 and LPC11C12/C14 引脚描述表 (LQFP48 封装) ...continued

符号	引脚	类型	描述
PIO0_2/SSEL0/ CT16B0_CAP0	10 ^{[3][2]}	I/O	PIO0_2 — 通用数字输入 / 输出引脚。
		O	SSEL0 — SPI0 从机选择引脚。
		I	CT16B0_CAP0 — 用于 16 位定时器 0 的捕获输入 0。
PIO0_3	14 ^{[3][2]}	I/O	PIO0_3 — 通用数字输入 / 输出引脚，复位时该引脚连同 PIO0_1 引脚一起被检测。低电平可以通过 C_CAN 启动 ISP 命令处理程序，高电平可以通过 UART 启动 ISP 命令处理程序。
PIO0_4/SCL	15 ^{[4][2]}	I/O	PIO0_4 — 通用数字输入 / 输出引脚 (开漏)。
		I/O	SCL — I2C 总线时钟输入 / 输出 (开漏)。仅在 IO 配置寄存器选择 I2C 为增强型快速模式 FM+ 时，该引脚为大电流灌入引脚。
PIO0_5/SDA	16 ^{[4][2]}	I/O	PIO0_5 — 通用数字输入 / 输出引脚 (开漏)。
		I/O	SDA — I2C 总线，数据输入 / 输出 (开漏)。仅在 IO 配置寄存器选择 I2C 为增强型快速模式 FM+ 时，该引脚为大电流灌入引脚。
PIO0_6/SCK0	22 ^{[3][2]}	I/O	PIO0_6 — 通用数字输入 / 输出引脚。
		I/O	SCK0 — SPI0 串行时钟。
PIO0_7/ $\overline{\text{CTS}}$	23 ^{[3][2]}	I/O	PIO0_7 — 通用数字输入 / 输出引脚 (大电流输出驱动)。
		I	$\overline{\text{CTS}}$ — UART 清除发送。—
PIO0_8/MISO0/ CT16B0_MAT0	27 ^{[3][2]}	I/O	PIO0_8 — 通用数字输入 / 输出引脚。
		I/O	MISO0 — SPI0 主器件数据输出、从器件数据输入。
		O	CT16B0_MAT0 — 用于 16 位定时器 0 的匹配输出 0。
PIO0_9/MOSI0/ CT16B0_MAT1	28 ^{[3][2]}	I/O	PIO0_9 — 通用数字输入 / 输出引脚。
		I/O	MOSI0 — SPI0 主器件数据输出、从器件数据输入。
		O	CT16B0_MAT1 — 用于 16 位定时器 0 的匹配输出 0。
SWCLK/PIO0_10/ SCK0/CT16B0_MAT2	29 ^{[3][2]}	I	SWCLK — 串行线时钟。
		I/O	PIO0_10 — SPI0 主器件数据输出、从器件数据输入。
		I/O	SCK0 — SPI0 串行时钟。
		O	CT16B0_MAT2 — 用于 16 位定时器 0 的匹配输出 0。
R/PIO0_11/ AD0/CT32B0_MAT3	32 ^{[5][2]}	I	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO0_11 — 通用数字输入 / 输出引脚。
		I	AD0 — A/D 转换器输入 0。
		O	CT32B0_MAT3 — 用于 32 位定时器 0 的匹配输出 0。
PIO1_0 to PIO1_11		I/O	Port 1 — Port 1 口是一个 12 位的端口，每一位都有独立的方向和功能控制。port 1 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。
R/PIO1_0/ AD1/CT32B1_CAP0	33 ^{[5][2]}	I	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO1_0 — 通用数字输入 / 输出引脚。
		I	AD1 — A/D 转换器输入 1。
		I	CT32B1_CAP0 — 用于 32 位定时器 1 的捕获输入 0。
R/PIO1_1/ AD2/CT32B1_MAT0	34 ^[5]	O	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO1_1 — 通用数字输入 / 输出引脚。
		I	AD2 — A/D 转换器输入 2。
		O	CT32B1_MAT0 — 用于 32 位定时器 1 匹配输出 0。

Table 103. LPC1113/14 and LPC11C12/C14 引脚描述表 (LQFP48 封装) ...continued

符号	引脚	类型	描述
R/PIO1_2/ AD3/CT32B1_MAT1	35 ^[5]	I	R — 保留, IOCONFIG 模块进行复用功能的配置。
		I/O	PIO1_2 — 通用数字输入 / 输出引脚。
		I	AD3 — A/D 转换器输入 3。
		O	CT32B1_MAT1 — 用于 32 位定时器 1 的匹配输出 1。
SWDIO/PIO1_3/AD4/ CT32B1_MAT2	39 ^[5]	I/O	SWDIO — 串行线调试输入输出脚。
		I/O	PIO1_3 — 通用数字输入 / 输出引脚。
		I	AD4 — A/D 转换器输入 4。
		O	CT32B1_MAT2 — 用于 32 位定时器 1 匹配输出 2。
PIO1_4/AD5/ CT32B1_MAT3/WAKEUP	40 ^[5]	I/O	PIO1_4 — 通用数字输入 / 输出引脚。
		I	AD5 — A/D 转换器输入 5。
		O	CT32B1_MAT3 — 用于 32 位定时器 1 的匹配输出 3。
		I	WAKEUP — 深度掉电模式唤醒引脚。要进入深度掉电模式必须从外部将该引脚拉高, 拉低则退出深度掉电模式。
PIO1_5/RTS/ CT32B0_CAP0	45 ^[3]	I/O	PIO1_5 — 通用数字输入 / 输出引脚。
		O	RTS — UART 请求发送。
		I	CT32B0_CAP0 — 用于 32 位定时器 0 的捕获输入 0。
PIO1_6/RXD/ CT32B0_MAT0	46 ^[3]	I/O	PIO1_6 — 通用数字输入 / 输出引脚。
		I	RXD — UART 数据接收。
		O	CT32B0_MAT0 — 用于 32 位定时器 0 的匹配输出 0。
PIO1_7/TXD/ CT32B0_MAT1	47 ^[3]	I/O	PIO1_7 — 通用数字输入 / 输出引脚。
		O	TXD — UART 数据发送。
		O	CT32B0_MAT1 — 用于 32 位定时器 0 的匹配输出 1。
PIO1_8/CT16B1_CAP0	9 ^[3]	I/O	PIO1_8 — 通用数字输入 / 输出引脚。
		I	CT16B1_CAP0 — 用于 16 位定时器 1 的捕获输入 0。
PIO1_9/CT16B1_MAT0	17 ^[3]	I/O	PIO1_9 — 通用数字输入 / 输出引脚。
		O	CT16B1_MAT0 — 用于 16 位定时器 1 的匹配输出 0。
PIO1_10/AD6/ CT16B1_MAT1	30 ^[5]	I/O	PIO1_10 — 通用数字输入 / 输出引脚。
		I	AD6 — A/D 转换器输入 6。
		O	CT16B1_MAT1 — 用于 16 位定时器 1 的匹配输出 1。
PIO1_11/AD7	42 ^[5]	I/O	PIO1_11 — 通用数字输入 / 输出引脚。
		I	AD7 — A/D 转换器输入 7。
PIO2_0 to PIO2_11		I/O	Port 2 — Port 2 口是一个 12 位的端口, 每一位都有独立的方向和功能控制。port 2 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。
PIO2_0/DTR/SSEL1	2 ^[3]	I/O	PIO2_0 — 通用数字输入 / 输出引脚。
		O	DTR — UART 数据终端就绪输出。
		O	SSEL1 — SPI1 从设备选择。
PIO2_1/DSR/SCK1	13 ^[3]	I/O	PIO2_1 — 通用数字输入 / 输出引脚。
		I	DSR — UART 数据输入就绪。
		I/O	SCK1 — SPI1 串行时钟。

Table 103. LPC1113/14 and LPC11C12/C14 引脚描述表 (LQFP48 封装) ...continued

符号	引脚	类型	描述
PIO2_2/ $\overline{\text{DCD}}$ /MISO1	26 ^[3]	I/O	PIO2_2 — 通用数字输入 / 输出引脚。
		I	$\overline{\text{DCD}}$ — UART 数据载波侦测输入。
		I/O	MISO1 — SPI1 主器件数据输入，从器件数据输出。
PIO2_3/ $\overline{\text{RI}}$ /MOSI1	38 ^[3]	I/O	PIO2_3 — 通用数字输入 / 输出引脚。
		I	$\overline{\text{RI}}$ — UART 振铃指示输入。
		I/O	MOSI1 — SPI1 主器件数据输出，从器件数据输入。
PIO2_4	19 ^[3]	I/O	PIO2_4 — 通用数字输入 / 输出引脚 (LPC1113/14 only)。
PIO2_4	18 ^[3]	I/O	PIO2_4 — 通用数字输入 / 输出引脚 (LPC11C12/C14 only)。
PIO2_5	20 ^[3]	I/O	PIO2_5 — 通用数字输入 / 输出引脚 (LPC1113/14 only)。
PIO2_5	21 ^[3]	I/O	PIO2_5 — 通用数字输入 / 输出引脚 (LPC11C12/C14 only)。
PIO2_6	1 ^[3]	I/O	PIO2_6 — 通用数字输入 / 输出引脚。
PIO2_7	11 ^[3]	I/O	PIO2_7 — 通用数字输入 / 输出引脚。
PIO2_8	12 ^[3]	I/O	PIO2_8 — 通用数字输入 / 输出引脚。
PIO2_9	24 ^[3]	I/O	PIO2_9 — 通用数字输入 / 输出引脚。
PIO2_10	25 ^[3]	I/O	PIO2_10 — 通用数字输入 / 输出引脚。
PIO2_11/SCK0	31 ^[3]	I/O	PIO2_11 — 通用数字输入 / 输出引脚。
		I/O	SCK0 — SPI0 串行时钟。
PIO3_0 to PIO3_5		I/O	Port 3 — Port 3 口是一个 12 位的端口，每一位都有独立的方向和功能控制。port 3 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。
PIO3_0/ $\overline{\text{DTR}}$	36 ^[3]	I/O	PIO3_0 — 通用数字输入 / 输出引脚。
		O	$\overline{\text{DTR}}$ — UART 数据终端就绪输出。
PIO3_1/ $\overline{\text{DSR}}$	37 ^[3]	I/O	PIO3_1 — 通用数字输入 / 输出引脚。
		I	$\overline{\text{DSR}}$ — UART 数据设备就绪。
PIO3_2/ $\overline{\text{DCD}}$	43 ^[3]	I/O	PIO3_2 — 通用数字输入 / 输出引脚。
		I	$\overline{\text{DCD}}$ — UART 数据载波侦测输入。
PIO3_3/ $\overline{\text{RI}}$	48 ^[3]	I/O	PIO3_3 — 通用数字输入 / 输出引脚。
		I	$\overline{\text{RI}}$ — UART 振铃指示输入。
PIO3_4	18 ^[3]	I/O	PIO3_4 — 通用数字输入 / 输出引脚 (LPC1113/14 only)。
PIO3_5	21 ^[3]	I/O	PIO3_5 — 通用数字输入 / 输出引脚 (LPC1113/14 only)。
CAN_RXD	19 ^[6]	I	CAN_RXD — C_CAN 接收数据输入 (LPC11C12/14 only)。
CAN_TXD	20 ^[6]	O	CAN_TXD — C_CAN 发送数据输出 (LPC11C12/14 only)。
V _{DD}	8; 44	I	用于内部电压调节器和 ADC 的 3.3 V 电压输入 也用作 ADC 参考电压。
XTALIN	6 ^[7]	I	振荡器电路和内部时钟发生器的输入 输入电压不得超过 1.8V。
XTALOUT	7 ^[7]	O	振荡器放大输出。
V _{SS}	5; 41	I	地。

[1] 端口可接受 5V 电压， $\overline{\text{RESET}}$ 功能在深度掉电模式下无效。使用 WAKEUP 引脚复位芯片，并将其从深度掉电模式下唤醒。

[2] 作为深度睡眠引脚启动 启动独立选择的引脚功能 (见 *LPC111x/11C1x 用户手册*)。

[3] 端口可接受 5V 电压，可配置上拉 / 下拉电阻和滞后控制的数字 I/O 功能。

- [4] I2C 总线设备遵从 I2C 标准模式和 I2C 快速模式规范。
- [5] 端口可接受 5V 电压，可配置上拉/下拉电阻和滞后控制的数字 I/O 功能。当配置为 ADC 转换器的输入时，该设备的数字部分将被禁用且相应引脚不再是 5V 逻辑电平。
- [6] 端口可接受 5V 电压，无上拉 / 下拉电阻。
- [7] 当不使用系统振荡器时，XTALIN 与 XTALOUT 应这样连接：XTALIN 悬空或者接地（接地应作首选以减少对噪声的敏感度）。XTALOUT 悬空。

Table 104. LPC1114 引脚描述表 (PLCC44 封装)

符号	引脚	类型	描述
PIO0_0 to PIO0_11		I/O	Port 0 — Port 0 口是一个 12 位的端口，每一位都有独立的方向和功能控制。port 0 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。
RESET/PIO0_0	7 ^{[1][2]}	I	RESET — 外部复位输入：此引脚上的低电平会使设备复位，I/O 端口和外设复位成初始的默认状态，并使处理器从 0 地址开始执行。
		I/O	PIO0_0 — 通用数字输入 / 输出引脚。
PIO0_1/CLKOUT/ CT32B0_MAT2	8 ^{[3][2]}	I/O	PIO0_1 — 通用数字输入 / 输出引脚。复位时该引脚为低电平，将启动在线系统编程命令处理程序。
		O	CLKOUT — 时钟输出脚。
		O	CT32B0_MAT2 — 用于 32 位定时器 0 的匹配输出 2。
PIO0_2/SSEL0/ CT16B0_CAP0	14 ^{[3][2]}	I/O	PIO0_2 — 通用数字输入 / 输出引脚。
		O	SSEL0 — SPI0 从设备选择。
		I	CT16B0_CAP0 — 用于 16 位定时器 0 的捕获输入 0。
PIO0_3	18 ^{[3][2]}	I/O	PIO0_3 — 通用数字输入 / 输出引脚。
PIO0_4/SCL	19 ^{[4][2]}	I/O	PIO0_4 — 通用数字输入 / 输出引脚 (开漏)。
		I/O	SCL — I2C 总线时钟输入 / 输出引脚，开漏。只有在 IO 配置寄存器选择 I2C 为增强型快速模式 FM+ 时，为大电流吸吸引脚。
PIO0_5/SDA	20 ^{[4][2]}	I/O	PIO0_5 — 通用数字输入 / 输出引脚 (开漏)。
		I/O	SDA — I2C 总线时钟输入 / 输出引脚，开漏。只有在 IO 配置寄存器选择 I2C 为增强型快速模式 FM+ 时，为大电流吸吸引脚。
PIO0_6/SCK0	26 ^{[3][2]}	I/O	PIO0_6 — 通用数字输入 / 输出引脚。
		I/O	SCK0 — SPI0 串行时钟。
PIO0_7/CTS	27 ^{[3][2]}	I/O	PIO0_7 — 通用数字输入 / 输出引脚 (high-current output driver)。
		I	CTS — UART 清除发送输入。
PIO0_8/MISO0/ CT16B0_MAT0	31 ^{[3][2]}	I/O	PIO0_8 — 通用数字输入 / 输出引脚。
		I/O	MISO0 — SPI0 主器件数据输入，从器件数据输出。
		O	CT16B0_MAT0 — 用于 16 位定时器 0 的匹配输出 0。
PIO0_9/MOSI0/ CT16B0_MAT1	32 ^{[3][2]}	I/O	PIO0_9 — 通用数字输入 / 输出引脚。
		I/O	MOSI0 — SPI0 主器件数据输出，从器件数据输入。
		O	CT16B0_MAT1 — 用于 16 位定时器 0 的匹配输出 1。
SWCLK/PIO0_10/ SCK0/CT16B0_MAT2	33 ^{[3][2]}	I	SWCLK — 串行线时钟。
		I/O	PIO0_10 — 通用数字输入 / 输出引脚。
		I/O	SCK0 — SPI0 串行时钟。
		O	CT16B0_MAT2 — 用于 16 位定时器 0 的匹配输出 2。

Table 104. LPC1114 引脚描述表 (PLCC44 封装) ...continued

符号	引脚	类型	描述
R/PIO0_11/ AD0/CT32B0_MAT3	36 ^{[5][2]}	I	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO0_11 — 通用数字输入 / 输出引脚。
		I	AD0 — A/D 转换器输入 0。
		O	CT32B0_MAT3 — 用于 32 位定时器 0 的匹配输出 3。
PIO1_0 to PIO1_11		I/O	Port 1 — Port 1 口是一个 12 位的端口，每一位都有独立的方向和功能控制。port 1 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。
R/PIO1_0/ AD1/CT32B1_CAP0	37 ^{[5][2]}	I	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO1_0 — 通用数字输入 / 输出引脚。
		I	AD1 — A/D 转换器输入 1。
		I	CT32B1_CAP0 — 用于 32 位定时器 1 的捕获输入 0。
R/PIO1_1/ AD2/CT32B1_MAT0	38 ^[5]	O	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO1_1 — 通用数字输入 / 输出引脚。
		I	AD2 — A/D 转换器输入 2。
		O	CT32B1_MAT0 — 用于 32 位定时器 1 的匹配输出 0。
R/PIO1_2/ AD3/CT32B1_MAT1	39 ^[5]	I	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO1_2 — 通用数字输入 / 输出引脚。
		I	AD3 — A/D 转换器输入 3。
		O	CT32B1_MAT1 — 用于 32 位定时器 1 的匹配输出 1。
SWDIO/PIO1_3/AD4/ CT32B1_MAT2	41 ^[5]	I/O	SWDIO — 串行线调试输入输出引脚。
		I/O	PIO1_3 — 通用数字输入 / 输出引脚。
		I	AD4 — A/D 转换器输入 4。
		O	CT32B1_MAT2 — 用于 32 位定时器 1 的匹配输出 2。
PIO1_4/AD5/ CT32B1_MAT3/WAKEUP	42 ^[5]	I/O	PIO1_4 — 通用数字输入 / 输出引脚。
		I	AD5 — A/D 转换器输入 5。
		O	CT32B1_MAT3 — 用于 32 位定时器 1 的匹配输出 3。
		I	WAKEUP — 深度掉电模式唤醒引脚。该引脚必须在外部被拉高进入深度掉电模式，拉低则退出。
PIO1_5/RTS/ CT32B0_CAP0	2 ^[3]	I/O	PIO1_5 — 通用数字输入 / 输出引脚。
		O	RTS — UART 请求发送输出。
		I	CT32B0_CAP0 — 用于 32 位定时器 0 的捕获输入 0。
PIO1_6/RXD/ CT32B0_MAT0	3 ^[3]	I/O	PIO1_6 — 通用数字输入 / 输出引脚。
		I	RXD — UART 数据接收。
		O	CT32B0_MAT0 — 用于 32 位定时器 0 的匹配输出 0。
PIO1_7/TXD/ CT32B0_MAT1	4 ^[3]	I/O	PIO1_7 — 通用数字输入 / 输出引脚。
		O	TXD — UART 数据发送。
		O	CT32B0_MAT1 — 用于 32 位定时器 0 的匹配输出 1。
PIO1_8/CT16B1_CAP0	13 ^[3]	I/O	PIO1_8 — 通用数字输入 / 输出引脚。
		I	CT16B1_CAP0 — 用于 16 位定时器 1 的捕获输入 0。
PIO1_9/CT16B1_MAT0	21 ^[3]	I/O	PIO1_9 — 通用数字输入 / 输出引脚。
		O	CT16B1_MAT0 — 用于 16 位定时器 1 匹配输出 0。

Table 104. LPC1114 引脚描述表 (PLCC44 封装) ...continued

符号	引脚	类型	描述
PIO1_10/AD6/ CT16B1_MAT1	34 ^[5]	I/O	PIO1_10 — 通用数字输入 / 输出引脚。
		I	AD6 — A/D 转换器输入 6。
		O	CT16B1_MAT1 — 用于 16 位定时器 1 的匹配输出 1。
PIO1_11/AD7	44 ^[5]	I/O	PIO1_11 — 通用数字输入 / 输出引脚。
		I	AD7 — A/D 转换器输入 7。
PIO2_0 to PIO2_11		I/O	Port 2 — Port 2 口是一个 12 位的端口，每一位都有独立的方向和功能控制。port 2 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。
PIO2_0/ $\overline{\text{DTR}}$ /SSEL1	6 ^[3]	I/O	PIO2_0 — 通用数字输入 / 输出引脚。
		O	$\overline{\text{DTR}}$ — UART 数据终端就绪输出。
		O	SSEL1 — SPI1 从设备选择。
PIO2_1/ $\overline{\text{DSR}}$ /SCK1	17 ^[3]	I/O	PIO2_1 — 通用数字输入 / 输出引脚。
		I	$\overline{\text{DSR}}$ — UART 数据就绪输入。
		I/O	SCK1 — SPI1 串行时钟。
PIO2_2/ $\overline{\text{DCD}}$ /MISO1	30 ^[3]	I/O	PIO2_2 — 通用数字输入 / 输出引脚。
		I	$\overline{\text{DCD}}$ — UART 数据载波侦测输入。
		I/O	MISO1 — SPI1 主器件数据输入，从器件数据输出。
PIO2_3/ $\overline{\text{RI}}$ /MOSI1	40 ^[3]	I/O	PIO2_3 — 通用数字输入 / 输出引脚。
		I	$\overline{\text{RI}}$ — UART 振铃指示输入。
		I/O	MOSI1 — SPI1 主器件数据输出，从器件数据输入。
PIO2_4	23 ^[3]	I/O	PIO2_4 — 通用数字输入 / 输出引脚。
PIO2_5	24 ^[3]	I/O	PIO2_5 — 通用数字输入 / 输出引脚。
PIO2_6	5 ^[3]	I/O	PIO2_6 — 通用数字输入 / 输出引脚。
PIO2_7	15 ^[3]	I/O	PIO2_7 — 通用数字输入 / 输出引脚。
PIO2_8	16 ^[3]	I/O	PIO2_8 — 通用数字输入 / 输出引脚。
PIO2_9	28 ^[3]	I/O	PIO2_9 — 通用数字输入 / 输出引脚。
PIO2_10	29 ^[3]	I/O	PIO2_10 — 通用数字输入 / 输出引脚。
PIO2_11/SCK0	35 ^[3]	I/O	PIO2_11 — 通用数字输入 / 输出引脚。
		I/O	SCK0 — SPI0 串行时钟。
PIO3_0 to PIO3_5		I/O	Port 3 — Port 3 口是一个 12 位的端口，每一位都有独立的方向和功能控制。port 3 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。引脚从 PIO3_0 到 PIO3_3 以及从 PIO3_6 到 PIO3_11 都不可用。
PIO3_4	22 ^[3]	I/O	PIO3_4 — 通用数字输入 / 输出引脚。
PIO3_5	25 ^[3]	I/O	PIO3_5 — 通用数字输入 / 输出引脚。
V _{DD}	1; 12	I	用于内部电压调节器和 ADC 的 3.3 V 电源，也用于 ADC 参考电压。
XTALIN	10 ^[7]	I	I 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 1.8V。
XTALOUT	11 ^[7]	O	振荡器放大输出。
V _{SS}	9; 43	I	地。

[1] 端口可接受 5V 电压， $\overline{\text{RESET}}$ 功能在深度掉电模式下无效。使用 WAKEUP 引脚复位芯片，并将其从深度掉电模式下唤醒。

[2] 作为深度睡眠引脚启动 启动独立选择的引脚功能 (见 *LPC111x/11C1x 用户手册*)。

[3] 端口可接受 5V 电压，可配置上拉 / 下拉电阻和滞后控制的数字 I/O 功能。

- [4] I2C 总线设备遵从 I2C 标准模式和 I2C 快速模式规范。
- [5] 端口可接受 5V 电压，可配置上拉/下拉电阻和滞后控制的数字 I/O 功能。当配置为 ADC 转换器的输入时，该设备的数字部分将被禁用且相应引脚不再是 5V 逻辑电平。
- [6] 当不使用系统振荡器时，XTALIN 与 XTALOUT 应这样连接：XTALIN 悬空或者接地（接地应作首选以减少对噪声的敏感度）。XTALOUT 悬空。

Table 105. LPC1111/12/13/14 引脚描述表 (HVQFN33 封装)

符号	引脚	类型	描述
PIO0_0 to PIO0_11		I/O	Port 0 — Port 0 口是一个 12 位的端口，每一位都有独立的方向和功能控制。port 0 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。
RESET/PIO0_0	2 ^{[1][2]}	I	RESET — 外部复位输入：此引脚上的低电平会使设备复位，I/O 端口和外设复位为初始的默认状态，并使处理器从 0 地址开始执行。
		I/O	PIO0_0 — 通用数字输入 / 输出引脚。
PIO0_1/CLKOUT/ CT32B0_MAT2	3 ^{[3][2]}	I/O	PIO0_1 — 通用数字输入 / 输出引脚。复位时该引脚为低电平启动在线系统编程命令处理程序。
		O	CLKOUT — 时钟输出引脚。
		O	CT32B0_MAT2 — 用于 32 位定时器 0 的匹配输出 2。
PIO0_2/SSEL0/ CT16B0_CAP0	8 ^{[3][2]}	I/O	PIO0_2 — 通用数字输入 / 输出引脚。
		O	SSEL0 — SPI0 从设备选择。
		I	CT16B0_CAP0 — 用于 16 位定时器 0 的捕获输入 0。
PIO0_3	9 ^{[3][2]}	I/O	PIO0_3 — 通用数字输入 / 输出引脚。
PIO0_4/SCL	10 ^{[4][2]}	I/O	PIO0_4 — 通用数字输入 / 输出引脚 (开漏)。
		I/O	SCL — I2C 总线时钟输入 / 输出引脚，开漏。只有在 IO 配置寄存器选择 I2C 为增强型快速模式 FM+ 时，为大电流吸收引脚。
PIO0_5/SDA	11 ^{[4][2]}	I/O	PIO0_5 — 通用数字输入 / 输出引脚 (开漏)。
		I/O	SDA — I2C 总线数据输入 / 输出引脚，开漏。只有在 IO 配置寄存器选择 I2C 为增强型快速模式 FM+ 时，为大电流吸收引脚。
PIO0_6/SCK0	15 ^{[3][2]}	I/O	PIO0_6 — 通用数字输入 / 输出引脚。
		I/O	SCK0 — SPI0 的串行时钟。
PIO0_7/CTS	16 ^{[3][2]}	I/O	PIO0_7 — 通用数字输入 / 输出引脚 (大电流输出驱动器)。
		I	CTS — UART 清除发送输入。
PIO0_8/MISO0/ CT16B0_MAT0	17 ^{[3][2]}	I/O	PIO0_8 — 通用数字输入 / 输出引脚。
		I/O	MISO0 — SPI0 主器件数据输入，从器件数据输出。
		O	CT16B0_MAT0 — 用于 16 位定时器 0 的匹配输出 0。
PIO0_9/MOSI0/ CT16B0_MAT1	18 ^{[3][2]}	I/O	PIO0_9 — 通用数字输入 / 输出引脚。
		I/O	MOSI0 — SPI0 主器件数据输出，从器件数据输入。
		O	CT16B0_MAT1 — 用于 16 位定时器 0 的匹配输出 1。
SWCLK/PIO0_10/SCK0/ CT16B0_MAT2	19 ^{[3][2]}	I	SWCLK — 串行线。
		I/O	PIO0_10 — 通用数字输入 / 输出引脚。
		I/O	SCK0 — SPI0 的串行时钟。
		O	CT16B0_MAT2 — 用于 16 位定时器 0 的匹配输出。
R/PIO0_11/AD0/ CT32B0_MAT3	21 ^{[5][2]}	I	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO0_11 — 通用数字输入 / 输出引脚。
		I	AD0 — A/D 转换器输入 0。
		O	CT32B0_MAT3 — 用于 32 位定时器 0 的匹配输出 3。
PIO1_0 to PIO1_11		I/O	Port 1 — Port 1 口是一个 12 位的端口，每一位都有独立的方向和功能控制。port 1 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。

Table 105. LPC1111/12/13/14 引脚描述表 (HVQFN33 封装) ...continued

符号	引脚	类型	描述
R/PIO1_0/AD1/ CT32B1_CAP0	22 ^{[5][2]}	I	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO1_0 — 通用数字输入 / 输出引脚。
		I	AD1 — A/D 转换器输入 1。
		I	CT32B1_CAP0 — 用于 32 位定时器 1 的捕获输入 0。
R/PIO1_1/AD2/ CT32B1_MAT0	23 ^[5]	O	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO1_1 — 通用数字输入 / 输出引脚。
		I	AD2 — A/D 转换器输入 2。
		O	CT32B1_MAT0 — 用于 32 位定时器 1 的匹配输出 0。
R/PIO1_2/AD3/ CT32B1_MAT1	24 ^[5]	I	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO1_2 — 通用数字输入 / 输出引脚。
		I	AD3 — A/D 转换器输入 3。
		O	CT32B1_MAT1 — 用于 32 位定时器 1 的匹配输出 1。
SWDIO/PIO1_3/AD4/ CT32B1_MAT2	25 ^[5]	I/O	SWDIO — 串行线调试输入输出引脚。
		I/O	PIO1_3 — 通用数字输入 / 输出引脚。
		I	AD4 — A/D 转换器输入 4。
		O	CT32B1_MAT2 — 用于 32 位定时器 1 的匹配输出 2。
PIO1_4/AD5/ CT32B1_MAT3/WAKEUP	26 ^[5]	I/O	PIO1_4 — 通用数字输入 / 输出引脚。
		I	AD5 — A/D 转换器输入 5。
		O	CT32B1_MAT3 — 用于 32 位定时器 1 的匹配输出 3。
		I	WAKEUP — 深度掉电模式唤醒引脚。该引脚必须在外部被拉高进入深度掉电模式，拉低则退出。
PIO1_5/RTS/ CT32B0_CAP0	30 ^[3]	I/O	PIO1_5 — 通用数字输入 / 输出引脚。
		O	RTS — UART 请求发送输出。
		I	CT32B0_CAP0 — 用于 32 位定时器 0 的捕获输入 0。
PIO1_6/RXD/ CT32B0_MAT0	31 ^[3]	I/O	PIO1_6 — 通用数字输入 / 输出引脚。
		I	RXD — UART 数据接收。
		O	CT32B0_MAT0 — 用于 32 位定时器 0 的匹配输出 0。
PIO1_7/TXD/ CT32B0_MAT1	32 ^[3]	I/O	PIO1_7 — 通用数字输入 / 输出引脚。
		O	TXD — UART 数据发送。
		O	CT32B0_MAT1 — 用于 32 位定时器 0 的匹配输出 1。
PIO1_8/CT16B1_CAP0	7 ^[3]	I/O	PIO1_8 — 通用数字输入 / 输出引脚。
		I	CT16B1_CAP0 — 用于 16 位定时器 1 的捕获输入 0。
PIO1_9/CT16B1_MAT0	12 ^[3]	I/O	PIO1_9 — 通用数字输入 / 输出引脚。
		O	CT16B1_MAT0 — 用于 16 位定时器 1 的匹配输出 0。
PIO1_10/AD6/ CT16B1_MAT1	20 ^[5]	I/O	PIO1_10 — 通用数字输入 / 输出引脚。
		I	AD6 — A/D 转换器输入 6。
		O	CT16B1_MAT1 — 用于 16 位定时器 1 的匹配输出 1。
PIO1_11/AD7	27 ^[5]	I/O	PIO1_11 — 通用数字输入 / 输出引脚。
		I	AD7 — A/D 转换器输入 7。
PIO2_0		I/O	Port 2 — Port 2 口是一个 12 位的端口，每一位都有独立的方向和功能控制，port 2 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。引脚从 PIO2_1 到 PIO2_11 不可用。

Table 105. LPC1111/12/13/14 引脚描述表 (HVQFN33 封装) ...continued

符号	引脚	类型	描述
PIO2_0/ $\overline{\text{DTR}}$	1 ^[3]	I/O	PIO2_0 — 通用数字输入 / 输出引脚。
		O	$\overline{\text{DTR}}$ — UART 数据终端就绪输出。
PIO3_0 to PIO3_5		I/O	Port 3 — Port 3 口是一个 12 位的端口，每一位都有独立的方向和功能控制，port 3 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。引脚从 PIO3_0, PIO3_1, PIO3_3 以及从 PIO3_6 到 PIO3_11 都不可用。
PIO3_2	28 ^[3]	I/O	PIO3_2 — 通用数字输入 / 输出引脚。
PIO3_4	13 ^[3]	I/O	PIO3_4 — 通用数字输入 / 输出引脚。
PIO3_5	14 ^[3]	I/O	PIO3_5 — 通用数字输入 / 输出引脚。
V _{DD}	6; 29	I	用于内部电压调节器和 ADC 的 3.3 V 电源，也用于 ADC 参考电压。
XTALIN	4 ^[6]	I	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 1.8V。
XTALOUT	5 ^[6]	O	振荡放大器输出。
V _{SS}	33	-	导热引脚，接地。

- [1] 端口可接受 5V 电压， $\overline{\text{RESET}}$ 功能在深度掉电模式下无效。使用 WAKEUP 引脚复位芯片，并将其从深度掉电模式下唤醒。
- [2] 作为深度睡眠引脚启动 启动独立选择的引脚功能 (见 *LPC111x/11C1x 用户手册*)。
- [3] 端口可接受 5V 电压，可配置上拉 / 下拉电阻和滞后控制的数字 I/O 功能。
- [4] I2C 总线设备遵从 I2C 标准模式和 I2C 快速模式规范。
- [5] 端口可接受 5V 电压，可配置上拉 / 下拉电阻和滞后控制的数字 I/O 功能。当配置为 ADC 转换器的输入时，该设备的数字部分将被禁用且相应引脚不再是 5V 逻辑电。
- [6] 当不使用系统振荡器时，XTALIN 与 XTALOUT 应这样连接：XTALIN 悬空或者接地（接地应作首选以减少对噪声的敏感度）。XTALOUT 悬空。

Table 106. LPC11C24/C22 引脚配置表 (LQFP48 封装)

符号	引脚	类型	描述
PIO0_0 to PIO0_11			Port 0 — Port 0 口是一个 12 位的端口，每一位都有独立的方向和功能控制，port 0 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。
$\overline{\text{RESET}}$ /PIO0_0	3 ^[1]	I	$\overline{\text{RESET}}$ — 外部复位输入：此引脚上的低电平会使设备复位，I/O 端口和外设复位为初始的默认状态，并使处理器从 0 地址开始执行。
		I/O	PIO0_0 — 通用数字输入 / 输出引脚，带有 10ns 干扰滤波器。
PIO0_1/CLKOUT/ CT32B0_MAT2	4 ^[3]	I/O	PIO0_1 — 通用数字输入/输出引脚。复位时该引脚上有低电平将通过 UART (如果 PIO0_3 为高电位) 或 C_CAN (如果 PIO0_3 为低电位) 启动在线系统编程 (ISP) 命令处理程序)。
		O	CLKOUT — 时钟输出引脚。
		O	CT32B0_MAT2 — 用于 32 位定时器 0 的匹配输出 2。
PIO0_2/SSEL0/ CT16B0_CAP0	10 ^[3]	I/O	PIO0_2 — 通用数字输入 / 输出引脚。
		I/O	SSEL0 — SPI0 从设备选择。
		I	CT16B0_CAP0 — 用于 16 位定时器 0 的捕获输入 0。
PIO0_3	14 ^[3]	I/O	PIO0_3 — 通用数字输入/输出引脚。复位时该引脚连同 PIO0_1 引脚一起被检测，低电平可以通过 C_CAN 启动 ISP 命令处理程序，高电平可以通过 UART 启动 ISP 命令处理程序。
PIO0_4/SCL	15 ^[4]	I/O	PIO0_4 — 通用数字输入 / 输出引脚 (开漏)。
		I/O	SCL — I2C 总线时钟输入 / 输出引脚，开漏。只有在 IO 配置寄存器选择 I2C 为增强型快速模式 FM+ 时，为大电流吸收引脚。

Table 106. LPC11C24/C22 引脚配置表 (LQFP48 封装)

符号	引脚	类型	描述
PIO0_5/SDA	16 ^[4]	I/O	PIO0_5 — 通用数字输入 / 输出引脚 (开漏)。
		I/O	SDA — I2C 总线数据输入 / 输出引脚, 开漏。只有在 IO 配置寄存器选择 I2C 为增强型快速模式 FM+ 时, 为大电流吸收引脚。
PIO0_6/SCK0	23 ^[3]	I/O	PIO0_6 — 通用数字输入 / 输出引脚。
		I/O	SCK0 — SPI0 的串行时钟。
PIO0_7/CTS	24 ^[3]	I/O	PIO0_7 — 通用数字输入 / 输出引脚 (高电流输出驱动器)。
		I	CTS — UART 清除发送输入。
PIO0_8/MISO0/ CT16B0_MAT0	27 ^[3]	I/O	PIO0_8 — 通用数字输入 / 输出引脚。
		I/O	MISO0 — SPI0 主器件数据输入, 从器件数据输出。
		O	CT16B0_MAT0 — 用于 16 位定时器 0 的匹配输出 0。
PIO0_9/MOSI0/ CT16B0_MAT1	28 ^[3]	I/O	PIO0_9 — 通用数字输入 / 输出引脚。
		I/O	MOSI0 — SPI0 主器件数据输出, 从器件数据输入。
		O	CT16B0_MAT1 — 用于 16 位定时器 0 的匹配输出 1。
SWCLK/PIO0_10/ SCK0/ CT16B0_MAT2	29 ^[3]	I	SWCLK — 串行线时钟。
		I/O	PIO0_10 — 通用数字输入 / 输出引脚。
		I/O	SCK0 — SSPI0 的串行时钟。
		O	CT16B0_MAT2 — 用于 16 位定时器 0 的匹配输出 2。
R/PIO0_11/ AD0/ CT32B0_MAT3	32 ^[5]	-	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO0_11 — 通用数字输入 / 输出引脚。
		I	AD0 — A/D 转换器输入 0。
		O	CT32B0_MAT3 — 用于 32 位定时器 0 的匹配输出 3。
PIO1_0 to PIO1_11			Port 1 — Port 1 口是一个 12 位的端口, 每一位都有独立的方向和功能控制, port 1 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。
R/PIO1_0/AD1/ CT32B1_CAP0	33 ^[5]	-	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO1_0 — 通用数字输入 / 输出引脚。
		I	AD1 — A/D 转换器输入 1。
		I	CT32B1_CAP0 — 用于 32 位定时器 1 的捕获输入 0。
R/PIO1_1/AD2/ CT32B1_MAT0	34 ^[5]	-	R — 保留。IOCONFIG 模块进行复用功能的配置。
		I/O	PIO1_1 — 通用数字输入 / 输出引脚。
		I	AD2 — A/D 转换器输入 2。
		O	CT32B1_MAT0 — 用于 32 位定时器 1 的匹配输出 0。
R/PIO1_2/AD3/ CT32B1_MAT1	35 ^[5]	-	R — 保留, IOCONFIG 模块进行复用功能的配置。
		I/O	PIO1_2 — 通用数字输入 / 输出引脚。
		I	AD3 — A/D 转换器输入 3。
		O	CT32B1_MAT1 — 用于 32 位定时器 1 的匹配输出 1。
SWDIO/PIO1_3/ AD4/ CT32B1_MAT2	39 ^[5]	I/O	SWDIO — 串行线调试输入输出引脚。
		I/O	PIO1_3 — 通用数字输入 / 输出引脚。
		I	AD4 — A/D 转换器输入 4。
		O	CT32B1_MAT2 — 用于 32 位定时器 1 的匹配输出 2。

Table 106. LPC11C24/C22 引脚配置表 (LQFP48 封装)

符号	引脚	类型	描述
PIO1_4/AD5/ CT32B1_MAT3/ WAKEUP	40 ^[5]	I/O	PIO1_4 — 通用数字输入 / 输出引脚，带有 10ns 干扰滤波器。
		I	AD5 — A/D 转换器输入 5。
		O	CT32B1_MAT3 — 用于 32 位定时器 1 的匹配输出 3。
		I	WAKEUP — 深度掉电模式在 20 ns 的毛刺滤波器下唤醒引脚。该引脚必须在外部被拉高进入深度掉电模式，拉低则退出，一个 50 ns 的短持续低脉冲唤醒端口。
PIO1_5/RTS/ CT32B0_CAP0	45 ^[3]	I/O	PIO1_5 — 通用数字输入 / 输出引脚。
		O	RTS — UART 请求发送输出。
		I	CT32B0_CAP0 — 用于 32 位定时器 0 的捕获输入 0。
PIO1_6/RXD/ CT32B0_MAT0	46 ^[3]	I/O	PIO1_6 — 通用数字输入 / 输出引脚。
		I	RXD — UART 数据接收。
		O	CT32B0_MAT0 — 用于 32 位定时器 0 的匹配输出 0。
PIO1_7/TXD/ CT32B0_MAT1	47 ^[3]	I/O	PIO1_7 — 通用数字输入 / 输出引脚。
		O	TXD — UART 数据发送。
		O	CT32B0_MAT1 — 用于 32 位定时器 0 的匹配输出 1。
PIO1_8/ CT16B1_CAP0	9 ^[3]	I/O	PIO1_8 — 通用数字输入 / 输出引脚。
		I	CT16B1_CAP0 — 用于 16 位定时器 1 的捕获输入 0。
PIO1_10/AD6/ CT16B1_MAT1	30 ^[5]	I/O	PIO1_10 — 通用数字输入 / 输出引脚。
		I	AD6 — A/D 转换器输入 6。
		O	CT16B1_MAT1 — 用于 16 位定时器 1 的匹配输出 1。
PIO1_11/AD7	42 ^[5]	I/O	PIO1_11 — 通用数字输入 / 输出引脚。
		I	AD7 — A/D 转换器输入 7。
PIO2_0 to PIO2_11			Port 2 — Port 2 口是一个 12 位的端口，每一位都有独立的方向和功能控制，port 2 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能。
PIO2_0/DTR/ SSEL1	2 ^[3]	I/O	PIO2_0 — 通用数字输入 / 输出引脚。
		I/O	DTR — UART 数据终端就绪输出。
		I/O	SSEL1 — SPI1 从设备选择。
PIO2_1/DSR/SCK1	13 ^[3]	I/O	PIO2_1 — 通用数字输入 / 输出引脚。
		I	DSR — UART 数据终端就绪输出。
		I/O	SCK1 — SPI1 串行时钟。
PIO2_2/DCD/ MISO1	26 ^[3]	I/O	PIO2_2 — 通用数字输入 / 输出引脚。
		I	DCD — UART 数据载波检测输入。
		I/O	MISO1 — SPI1 主器件数据输入，从器件数据输出。
PIO2_3/RI/MOSI1	38 ^[3]	I/O	PIO2_3 — 通用数字输入 / 输出引脚。
		I	RI — UART 振铃指示输入。
		I/O	MOSI1 — SPI1 主器件数据输出，从器件数据输入。
PIO2_6	1 ^[3]	I/O	PIO2_6 — 通用数字输入 / 输出引脚。
PIO2_7	11 ^[3]	I/O	PIO2_7 — 通用数字输入 / 输出引脚。
PIO2_8	12 ^[3]	I/O	PIO2_8 — 通用数字输入 / 输出引脚。
PIO2_10	25 ^[3]	I/O	PIO2_10 — 通用数字输入 / 输出引脚。
PIO2_11/SCK0	31 ^[3]	I/O	PIO2_11 — 通用数字输入 / 输出引脚。
		I/O	SCK0 — SPI0 串行时钟。

Table 106. LPC11C24/C22 引脚配置表 (LQFP48 封装)

符号	引脚	类型	描述
PIO3_0 to PIO3_3			Port 3 — Port 3 口是一个 12 位的端口，每一位都有独立的方向和功能控制，port 3 引脚的操作取决于通过 IOCONFIG 寄存器块选择的功能，引脚从 PIO3_4 到 PIO3_11 不可用。
PIO3_0/DTR	36 ^[3]	I/O	PIO3_0 — 通用数字输入 / 输出引脚。 DTR — UART 数据终端就绪输出。
PIO3_1/DSR	37 ^[3]	I/O	PIO3_1 — 通用数字输入 / 输出引脚。 DSR — UART 数据设备就绪输入。
PIO3_2/DCD	43 ^[3]	I/O	PIO3_2 — 通用数字输入 / 输出引脚。 DCD — UART 数据载波侦测输入。
PIO3_3/RI	48 ^[3]	I/O	PIO3_3 — 通用数字输入 / 输出引脚。 RI — UART 振铃指示输入。
CANL	18	I/O	低电平 CAN 总线。
CANH	19	I/O	高电平 CAN 总线。
STB	22	I	沉寂状态控制 CAN 收发器的输入 (低电平 = 正常模式，高电平 = 沉寂状态)。
VDD_CAN	17	-	为 CAN 收发器的 I/O 等级提供电源电压。
VCC	20	-	为 CAN 收发器提供电源电压。
GND	21	-	CAN 收发器接地。
VDD	8;44	I	用于内部电压调节器和 ADC 的 3.3 V 电压输入，也用作 ADC 参考电压。
XTALIN	6 ^[7]	I	振荡器电路和内部时钟发生器的输入，输入电压不得超过 1.8V。
XTALOUT	7 ^[7]	O	振荡器放大输出。
VSS	5; 41	I	接地。

- [1] 端口可接受 5V 电压，RESET 功能在深度掉电模式下无效。使用 WAKEUP 引脚复位芯片，并将其从深度掉电模式下唤醒。
- [2] 作为深度睡眠引脚启动 启动独立选择的引脚功能 (见 LPC111x/11C1x 用户手册)。
- [3] 端口可接受 5V 电压，可配置上拉 / 下拉电阻和滞后控制的数字 I/O 功能。
- [4] 端口可接受 5V 电压，可配置上拉 / 下拉电阻和滞后控制的数字 I/O 功能。当配置为 ADC 转换器的输入时，该设备的数字部分将被禁用，且相应引脚不再是 5V 电压。
- [5] 端口可接受 5V 电压，无上拉 / 下拉电阻。
- [6] 当不使用系统振荡器时，XTALIN 与 XTALOUT 应这样连接：XTALIN 悬空或者接地（接地应作首选以减少对噪声的敏感度）。XTALOUT 悬空。

9.1 如何阅读本章

每个端口上可用的 GPIO 引脚数量与 LPC111x 元件和封装有关，可用的 GPIO 引脚见 [Table 107](#)

Table 107. GPIO configuration

型号	封装	GPIO 端口 0	GPIO 端口 1	GPIO 端口 2	GPIO 端口 3	GPIO 引脚总数
LPC1111	HVQFN33	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0	PIO3_2; PIO3_4; PIO3_5	28
LPC1112	HVQFN33	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0	PIO3_2; PIO3_4; PIO3_5	28
LPC1113	HVQFN33	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0	PIO3_2; PIO3_4; PIO3_5	28
	LQFP48	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0 to PIO2_11	PIO3_0 to PIO3_5	42
LPC1114	HVQFN33	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0	PIO3_2; PIO3_4; PIO3_5	28
	PLCC44	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0 to PIO2_11	PIO3_4 and PIO3_5	38
	LQFP48	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0 to PIO2_11	PIO3_0 to PIO3_5	42
LPC11C12	LQFP48	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0 to PIO2_11	PIO3_0 to PIO3_3	40
LPC11C14	LQFP48	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11	PIO2_0 to PIO2_11	PIO3_0 to PIO3_3	40
LPC11C22	LQFP48	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11 except PIO1_9	PIO2_0 to PIO2_11 except PIO2_4, PIO2_5, PIO2_9	PIO3_0 to PIO3_3	36
LPC11C24	LQFP48	PIO0_0 to PIO0_11	PIO1_0 to PIO1_11 except PIO1_9	PIO2_0 to PIO2_11 except PIO2_4, PIO2_5, PIO2_9	PIO3_0 to PIO3_3	36

无效 PION_m 管脚所对应的寄存器位预留。

9.2 概述

9.2.1 特性

- 可通过软件配置 GPIO 引脚为输入或输出
- 每个独立的端口引脚均可作为外部边沿或电平触发中断的输入引脚
- 边沿触发中断可配置为上升沿触发、下降沿触发以及双边沿触发
- 电平触发中断引脚可以配置为高电平或低电平触发
- 所有 GPIO 引脚默认情况下均为输入
- 从端口读取和写入数据操作可以通过地址位 13:2 屏蔽

9.3 Register 寄存器描述

每个 GPIO 寄存器都支持多达 12 位的宽度，可以在字地址已半字或字的形式进行访问

Table 108. 寄存器概览 : GPIO (端口寄存器基址 端口 0: 0x5000 0000; port 1: 0x5001 0000, port 2: 0x5002 0000; port 3: 0x5003 0000)

寄存器名	访问	便宜地址	描述	复位值
GPIO_nDATA	R/W	0x0000 to 0x3FF8	端口 n 数据地址屏蔽寄存器的引脚位置从 PION_0 到 PION_11 (见 Section 9.4.1).	n/a
GPIO_nDATA	R/W	0x3FFC	从引脚 PION_0 至 PION_11 的端口 n 数据寄存器	n/a
-	-	0x4000 to 0x7FFC	保留	-
GPIO_nDIR	R/W	0x8000	端口 n 的数据方向寄存器	0x00
GPIO_nIS	R/W	0x8004	端口 n 的中断感应寄存器	0x00
GPIO_nIBE	R/W	0x8008	端口 n 的双边沿触发寄存器	0x00
GPIO_nIEV	R/W	0x800C	端口 n 的中断事件寄存器	0x00
GPIO_nIE	R/W	0x8010	端口 n 的中断屏蔽寄存器	0x00
GPIO_nRIS	R	0x8014	端口 n 的原始中断状态寄存器	0x00
GPIO_nMIS	R	0x8018	端口 n 的被屏蔽中断状态寄存器	0x00
GPIO_nIC	W	0x801C	端口 n 的中断清除寄存器	0x00
-	-	0x8020 - 0xFFFF	保留	0x00

9.3.1 GPIO 数据寄存器

GPIO_nDATA 寄存器独立保持当前引脚的逻辑状态 (高电平或低电平), 无论引脚被配置成 GPIO 输入或输出, 亦或其他数字功能。如果引脚被配置成 GPIO 输出, 那么引脚驱动 GPIO_nDATA 寄存器的当前值。

Table 109. GPIO_nDATA 寄存器 (GPIO0DATA, 地址 0x5000 0000 至 0x5000 3FFC; GPIO1DATA, 地址 0x5001 0000 至 0x5001 3FFC; GPIO2DATA, 地址 0x5002 0000 至 0x5002 3FFC; GPIO3DATA, 地址 0x5003 0000 至 0x5003 3FFC) 位描述

位	符号	描述	复位值	访问值
11:0	DATA	引脚 PION_0 到 PION_11 的逻辑电平, 高电平 = 1, 低电平 = 0。	n/a	R/W
31:12	-	保留	-	-

读 GPIO_nDATA 寄存器总会返回它独立配置的当前引脚的逻辑电平, 因为有一个单一的数据寄存器同时存放输出的驱动值和引脚的输入状态, 根据引脚的配置写操作会有不同影响。

- 如果引脚被配置为 GPIO 输入, 向 GPIO_nDATA 寄存器写操作对引脚电平无效, 读操作会返回当前引脚的当前状态。
- 如果引脚被配置为 GPIO 输出, 引脚驱动 GPIO_nDATA 寄存器的当前值, 该值可以是想 GPIO_nDATA 寄存器写入的结果, 或者反映引脚以前的状态, 如果该引脚从 GPIO 输入切换到 GPIO 输出或者其他数字功能, 读操作会返回输出锁存器的当前值。

- 如果引脚被配置成其他的数字功能 (输入或输出), 向 GPIOnDATA 寄存器写操作对引脚电平无效, 读操作会返回引脚的当前状态即使它被配置成输出, 也就是说通过读 GPIOnDATA 寄存器, 函数的输入输出值而不是引脚上的 GPIO 会被监测。

以下规则适应于当引脚从输入切换到输出的时候:

- 引脚被配置为高电平输入
 - 改变引脚输出: 引脚驱动高电平。
- 引脚被配置为低电平输入
 - 改变引脚输出: 引脚驱动低电平。

结果表面引脚监测当前的逻辑电平, 因此浮动引脚可以驱动当从输入到输出切换时不可预测的电平。

9.3.2 GPIO 数据方向寄存器

Table 110. GPIOnDIR 寄存器 (GPIO0DIR, 地址 0x5000 8000 至 GPIO3DIR, 地址 0x5003 8000) 位描述

位	符号	描述	复位值	访问
11:0	IO	选择引脚 x 作为输入或输出 (x = 0 to 11) 0 = 引脚 PION_x 配置为输入 1 = 引脚 PION_x 配置为输出	0x00	R/W
31:12	-	保留	-	-

9.3.3 GPIO 中断感应寄存器

Table 111. GPIOnIS 寄存器 (GPIO0IS, 地址 0x5000 8004 至 GPIO3IS, 地址 0x5003 8004) 位描述

位	符号	描述	复位值	访问
11:0	ISENSE	选择中断引脚 x 对电平或边沿触发 (x = 0 to 11) 0 = 中断引脚 PION_x 配置为边沿触发 1 = 中断引脚 PION_x 配置为电平触发	0x00	R/W
31:12	-	保留	-	-

9.3.4 GPIO 中断双边沿感应寄存器

Table 112. GPIOnIBE 寄存器 (GPIO0IBE, 地址 0x5000 8008 至 GPIO3IBE, 地址 0x5003 8008) 位描述

位	符号	描述	复位值	访问
11:0	IBE	选择中断引脚 x 作为双边沿触发 (x = 0 to 11) 0 = 通过 GPIOnIEV 寄存器控制 PION_x 引脚中断 1 = 双边沿 PION_x 触发中断	0x00	R/W
31:12	-	保留	-	-

9.3.5 GPIO 中断事件寄存器

Table 113. GPIOnIEV 寄存器 (GPIO0IEV, 地址 0x5000 800C 至 GPIO3IEV, 地址 0x5003 800C) 位描述

位	符号	描述	复位值	访问
11:0	IEV	选择引脚 x 上是上升沿还是下降沿触发来中断 (x = 0 - 11)。PIOn_x 引脚上是下降沿触发中断还是低电平触发中断, 与 GPIOnIS 寄存器的设置 (见 Table 111) 有关。PIOn_x 引脚上是上升沿触发中断, 还是高电平触发中断与 GPIOnIS 寄存器的设置 (见 Table 111) 有关。	0x00	R/W
31:12	-	保留	-	-

9.3.6 GPIO 中断屏蔽寄存器

通过设置 GPIOnIE 的相应位为高来允许某对应的引脚触发中断以及组合的 GPIOnINTR 线。清除该寄存器相应位来禁止相应的引脚触发中。

Table 114. GPIOnIE 寄存器 (GPIO0IE, 地址 0x5000 8010 至 GPIO3IE, 地址 0x5003 8010) 位描述

位	符号	描述	复位值	访问
11:0	MASK	所选择引脚 pin x 中中断被屏蔽 (x = 0 - 11) 0 = 引脚 PIOn_x 中断被屏蔽 1 = 引脚 PIOn_x 中断未被屏蔽	0x00	R/W
31:12	-	保留	-	-

9.3.7 GPIO 原始中断状态寄存器

读取 GPIOnIRS 寄存器, 如果某位为高, 反映其在被允许触发 GPIOIE 之前, 原始 (屏蔽前) 中断状态所对应的引脚已经满足了所有的条件; 位读取为 0 则表明相应的输入引脚还没有启动中断, 该寄存器为只读寄存器。

Table 115. GPIOnRIS 寄存器 (GPIO0RIS, 地址 0x5000 8014 至 GPIO3RIS, 地址 0x5003 8014) 位描述

位	符号	描述	复位值	访问
11:0	RAWST	原始中断状态 (x = 0 to 11) 0 = 引脚 PIOn_x 没有中断请求 1 = 引脚 PIOn_x 发生中断事件	0x00	R
31:12	-	保留	-	-

9.3.8 GPIO 中断屏蔽状态寄存器

读取 GPIOnMIS 寄存器, 如果某位为高, 则反映了相应外部输入引脚触发了一次中断。读取某位为低, 则表明相应的输入引脚没有发生中断或者中断被屏蔽。GPIOMIS 是屏蔽后的中断状态。该寄存器为只读寄存器。

Table 116. GPIOnMIS 寄存器 (GPIO0MIS, 地址 0x5000 8018 至 GPIO3MIS, 地址 0x5003 8018) 位描述

Bit	Symbol	Description	Reset value	Access
11:0	MASK	所选择中断引脚 pin x 被屏蔽 (x = 0 to 11) 0 = 引脚 PION_x 没有中断或被中断所屏蔽 1 = 引脚 PION_x 发生中断	0x00	R
31:12	-	保留	-	-

9.3.9 GPIO 中断清除寄存器

使用该寄存器，可软件清除被设置为边沿触发的端口位的值。如果某端口位是电平触发的，则无效。

Table 117. GPIOnIC 寄存器 (GPIO0IC, 地址 0x5000 801C 至 GPIO3IC, 地址 0x5003 801C) 位域描述

位	符号	描述	复位值	访问值
11:0	CLR	所选择引脚 pin x 中断被清零 (x = 0 to 11). 清除中断边沿检测逻辑；寄存器为只写。 Remark: 在 GPIO 和 NVIC 之间的同步器产生 2 个时钟的延迟。在未退出中断服务程序时，建议在清除中断边沿检测逻辑后增加 2 条 NOP 指令。 0 = 无影响 1 = 清除引脚 PION_x 边沿检测逻辑	0x00	W
31:12	-	保留	-	-

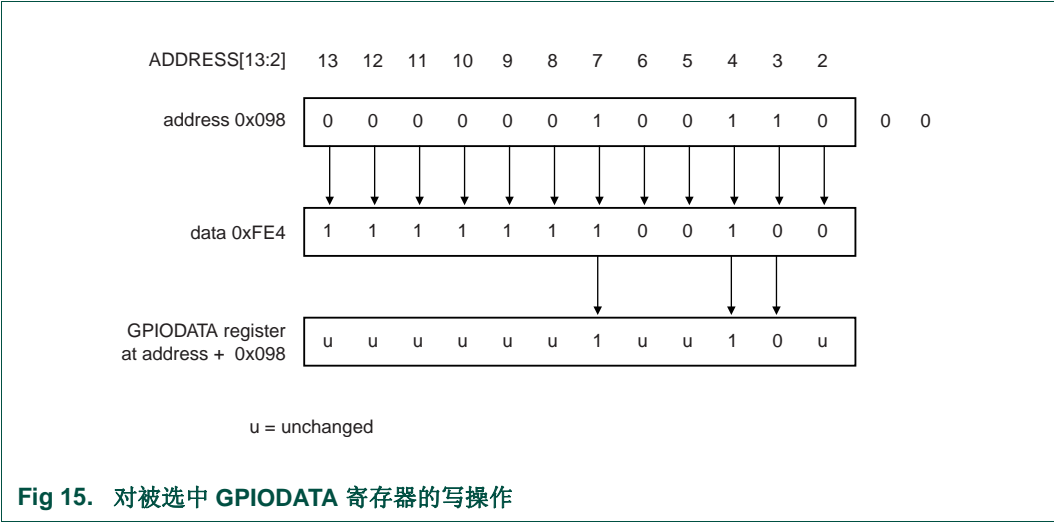
9.4 Functional description

9.4.1 读 / 写数据操作

为了使软件能在一个单独的写操作内设置 GPIO 位而不受其它引脚的影响，14 位地址中的 13:2 位用来为每个端口的 12 个 GPIO 引脚的读写操作产生一个 12 位宽的掩码。被选中的 GPIODATA 寄存器可以定位到 GPIOn 地址空间偏移地址 0x0000 ~ 0x3FFC 之间的任何位置。

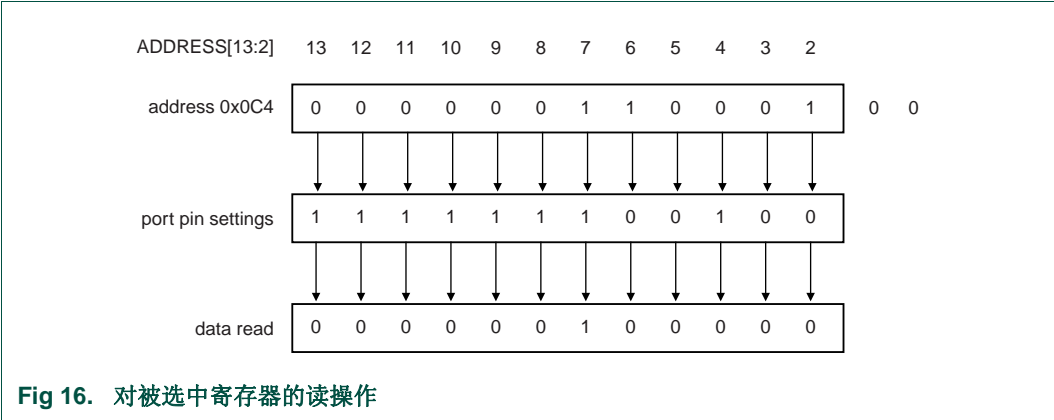
写操作

如果与 GPIO 端口位 i (i=0 -11) 相关的地址位 (i+2) 被设置为高，GPIODATA 寄存器位 i 的值将被更新。如果地址位 (i+2) 为低电平，那么相应的 GPIODATA 寄存器位 i 将保持不变。



读操作

如果与 GPIO 数据位相关的地址位为高，那么数值将被读取；如果地址位为低，读取 GPIO数据位为0。读端口数据寄存器将获得端口引脚11:0 的状态与地址位 13:2相 “与” 的结果。



10.1 如何阅读本章

所有 LPC111x 系列 UART 模块都相同。 \overline{DSR} 、 \overline{DCD} 和 \overline{RI} 三个调制信号只在 LQFP48 和 PLCC44 封装下配置有引脚。

注意对部分 LPC111x/101/201/301, UART 引脚必须在 UART 时钟使能之前配置好, 但对部分 LPC11Cxx 和部分 LPC111x/102/202/302 则没有这一要求。

10.2 基本配置

使用以下寄存器配置 UART:

1. 引脚: 对于 LPC111x/101/201/301, 必须在 SYSAHBCLKCTRL 寄存器使能 UART 时钟之前在 IOCONFIG 寄存器模块 ([Section 7.4](#)) 中配置 UART 引脚。对于 LPC111x/102/202/302 和 LPC11Cxx/101/201/301, 没有特定使能顺序的要求。

注: 如果调制解调器输入引脚在使用, 调制解调器功能位置必须在 UART 位置寄存器中选定 ([Section 7.4](#))。

2. 电源: 在 SYSAHBCLKCTRL 寄存器, 设置第 12 位 ([Table 21](#))。
3. 外设时钟: 通过写 UARTCLKDIV 寄存器使能 UART 外设时钟 ([Table 23](#))。

10.3 特性

- 16 字节收发 FIFO
- 寄存器位置符合 '550 工业标准
- 接收器 FIFO 触发点可为 1、4、8 和 14 字节
- 内置波特率发生器
- UART 允许软件或硬件实现流控制
- 支持 RS-485/EIA-485 9 位模式
- 调制解调器控制

10.4 引脚描述

Table 118. UART 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
RTS	输出	请求发送 . RS-485 方向控制引脚
DTR	输出	数据终端就绪
DSR ^[1]	输入	数据设备就绪
CTS	输入	清除发送
DCD ^[1]	输入	数据载波检测
RI ^[1]	输入	振铃指示

[1] 只针对 LQFP48 封装

调制解调器输入 $\overline{\text{DSR}}$, $\overline{\text{DCD}}$ 和 $\overline{\text{RI}}$ 复用在两个不同的引脚上。通过使用 IOCON_LOC 寄存器 (见 [Section 7.4](#)) 可为每个功能选择 LQFP48 封装上引脚的位置，通过 IOCON 寄存器选择功能。

$\overline{\text{DTR}}$ 输出对于 2 个引脚位置都是可用的。引脚 $\overline{\text{DTR}}$ 的值同时受两个位置分别驱动， $\overline{\text{DTR}}$ 功能在哪个引脚上可以通过 IOCON 寄存器为引脚选择功能来确定。

10.5 寄存器描述

UART 所包含的寄存器如 [Table 119](#). 所列。除数锁存访问位（DLAB）位于 U0LCR[7]，它用于允许对除数锁存器的访问。

复位值仅反映已使用位中存储的数值，它不包括保留位的内容。

Table 119. 寄存器概览：UART (基地址：0x4000 8000)

寄存器名	访问	偏移地址	描述	复位值
U0RBR	RO	0x000	接收器缓存寄存器，存放着下一个要读的已接受字节 (DLAB=0)	NA
U0THR	WO	0x000	发送保持寄存器。存放下一个将要被发送的字符 (DLAB=0)	NA
U0DLL	R/W	0x000	除数锁存 LSB。波特率除数值的最低字节。完整除数用于分数分频器，来生成波特率 (DLAB=1)	0x01
U0DLM	R/W	0x004	除数锁存 MSB。波特率除数值的最高字节。完整除数用于分数分频器，来生成波特率 (DLAB=1)	0x00
U0IER	R/W	0x004	中断允许寄存器。包含有 7 个独立的潜在的 UART 中断的允许位 (DLAB=0)	0x00
U0IIR	RO	0x008	中断 ID 寄存器。识别出哪个中断正被挂起	0x01

Table 119. 寄存器概览 : UART (基地址 : 0x4000 8000)

寄存器名	访问	偏移地址	描述	复位值
U0FCR	WO	0x008	FIFO 控制寄存器。控制 UART FIFO 的用法和模式	0x00
U0LCR	R/W	0x00C	线控制寄存器。包含有帧格式和中断生成的控制	0x00
U0MCR	R/W	0x010	Modem 控制寄存器	0x00
U0LSR	RO	0x014	线状态寄存器。包含接收和发送状态，以及线错误的标志	0x60
U0MSR	RO	0x018	Modem 状态寄存器	0x00
U0SCR	R/W	0x01C	暂存寄存器。软件可以使用这 8 位临时存储空间	0x00
U0ACR	R/W	0x020	自动波特率控制寄存器。包含自动波特率特性的控制	0x00
-	-	0x024	保留	-
U0FDR	R/W	0x028	分数分频寄存器。为波特率分频器产生时钟输入	0x10
-	-	0x02C	保留	-
U0TER	R/W	0x030	发送允许寄存器。软件流控制时，使用该寄存器关掉 UART 发送器	0x80
-	-	0x034 - 0x048	保留	-
U0RS485CTRL	R/W	0x04C	RS-485/EIA-485 控制。包含配置 RS-485/EIA-485 模式各个方面的控制	0x00
U0RS485ADR MATCH	R/W	0x050	RS-485/EIA-485 地址匹配。包含 RS-485/EIA-485 模式下的地址匹配值	0x00
U0RS485DLY	R/W	0x054	RS-485/EIA-485 直接控制延迟	0x00

10.5.1 UART 接收缓存寄存器 (U0RBR - 0x4000 8000, 当 DLAB = 0, 只读)

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

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

由于 PE、FE 和 BI 位 (见 [Table 131](#)) 与 RBR FIFO 顶端的字节相对应（即下次读 RBR 时，读出的字节），因此，将接收的字节及其状态位成对读出的正确方法是先读 U0LSR、再读 U0RBR。

Table 120. UART 接收器缓存寄存器 (U0RBR – 地址 0x4000 8000 当 DLAB = 0, 只读) 位域描述

位	符号	描述	复位值
7:0	RBR	UART接收器缓存寄存器包含UART Rx FIFO当中最早接收到的字节	未定义
31:8	-	保留	-

10.5.2 UART 发送器保持寄存器（U0THR - 0x4000 8000，DLAB=0，只写）

U0THR 是 UART TX(发送) FIFO 的最高字节。该最高字节是 TX FIFO 中最新的字符，可通过总线接口写入。LSB 代表最先要发送的位。

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

Table 121. UART 发送器保持寄存器 (U0THR – 地址 0x4000 8000 当 DLAB = 0, 只写) 位域描述

位	符号	描述	复位值
7:0	THR	写 UART 发送器保持寄存器将使数据保存到 UART 发送 FIFO 当中，当字节到达 FIFO 的最底部并且发送器就绪时，该字节将被发送。	NA
31:8	-	保留	-

10.5.3 UART 除数锁存 LSB 和 MSB 寄存器 (U0DLL - 0x4000 8000 和 U0DLM - 0x4000 8004, 当 DLAB = 1)

UART 除数锁存是 UART 波特率发生器的一部分，并且保持使用的值，与分数分频器一起对 UART_PCLK 时钟分频来产生波特率时钟，波特率时钟是波特率的 16 倍。U0DLL 和 U0DLM 寄存器一起构成一个 16 位除数，其中 U0DLL 包含除数的低 8 位，U0DLM 包含除数的高 8 位；值 0x0000 被看作是 0x0001，因为除数是不允许为 0。当访问 UART 除数锁存寄存器时，U0LCR 中的除数锁存访问位（DLAB）必须为 1。如何选择正确的 U0DLL 和 U0DLM 值，详见 [Section 10.5.15](#)。

Table 122. UART 除数锁存 LSB 寄存器 (U0DLL – 地址 0x4000 8000 当 DLAB = 1) 位域描述

位	符号	描述	复位值
7:0	DLLSB	UART 除数锁存 LSB 寄存器与 U0DLM 寄存器一起决定 UART 的波特率	0x01
31:8	-	保留	-

Table 123. UART 除数锁存 MSB 寄存器 (U0DLM – 地址 0x4000 8004 当 DLAB = 1) 位域描述

位	符号	描述	复位值
7:0	DLMSB	UART 除数锁存 MSB 寄存器与 U0DLL 寄存器一起决定 UART 的波特率	0x00
31:8	-	保留	-

10.5.4 UART 中断允许寄存器 (U0IER - 0x4000 8004, 当 DLAB = 0)

U0IER 用于允许 UART 四个中断源。

Table 124. UART 中断允许寄存器 (U0IER – 地址 0x4000 8004 当 DLAB = 0) 位域描述

位	符号	值	描述	复位值
0	RBRIE		RBR 中断使能。允许 UART 接收数据有效中断。它还控制字符接收超时中断	0
		0	禁止 RDA 中断	
		1	允许 RDA 中断	
1	THREIE		THRE 中断使能，允许 UART THRE 中断。该中断的状态可从 U0LSR[5] 读出	0
		0	禁止 THRE 中断	
		1	允许 THRE 中断	
2	RXLIE		RX 线中断使能，允许 UART Rx 线状态中断。该中断的状态可从 U0LSR[4:1] 中读出	0
		0	禁止 Rx 线状态中断	
		1	允许 Rx 线状态中断	
3	-	-	保留	-
6:4	-		保留，用户软件不要向保留位写入 1，从保留位读出的值未被定义	NA
7	-	-	保留	0
8	ABEOIntEn		允许自动波特率结束中断	0
		0	禁止自动波特率结束中断	
		1	允许自动波特率结束中断	

Table 124. UART 中断允许寄存器 (U0IER – 地址 0x4000 8004 当 DLAB = 0) 位域描述 ...continued

位	符号	值	描述	复位值
9	ABTOIntEn		允许自动波特率超时中断	0
		0	禁止自动波特率超时中断	
		1	允许自动波特率超时中断	
31:10	-		保留，用户软件不要向保留位写入 1，从保留位读出的值未被定义	NA

10.5.5 UART 中断标识寄存器 (U0IIR - 0x4004 8008, 只读)

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

Table 125. UART 中断标识寄存器 (U0IIR – 地址 0x4004 8008, 只读) 位域描述

位	符号	值	描述	复位值
0	IntStatus		中断状态。U0IIR[0] 为低有效。挂起的中断可通过 U1IIR[3:1] 确定	1
		0	至少有 1 个中断被挂起	
		1	没有挂起的中断	
3:1	IntId		中断标识。U0IER[3:1] 指示对应于 UART Rx FIFO 的中断。下面未列出的 U0IER[3:1] 的其它组合都为保留值（100, 101, 111）	0
		0x3	1 - 接收线状态（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 - Modem 中断	
5:4	-		保留，用户软件不要向保留位写入 1，从保留位读出的值未被定义	NA
7:6	FIFOEnable		这些位等效于 U0FCR[0]	0
8	ABEOInt		自动波特率结束中断。如果自动波特率成功完成且中断被允许，则 ABEOInt 为 1	0
9	ABTOInt		自动波特率超时中断。如果波特率未超时且中断被允许，则 ABTOInt 为 1	0
31:10	-		保留，用户软件不要向保留位写入 1，从保留位读出的值未被定义	NA

位 U0IIR[9:8] 通过自动波特率功能设置，指示超时或自动波特率结束。设置自动波特率控制寄存器的 Clear 位，将清除自动波特率中断条件。

如果 `IntStatus` 位是 1 且没有中断挂起，则 `IntId` 位会是 0。如果 `IntStatus` 是 0，且一个非自动波特率中断被挂起，在这种情况下 `IntId` 位标识中断类型，其处理过程如 [Table 126](#) 所示。通过 `U0IIR[3:0]` 的状态，中断处理程序可以确定中断产生的原因和如何清除激活的中断。为了清除中断优先级，在退出中断服务程序之前必须读 `U0IIR`。

UART RLS 中断（`U0IIR[3:1]=011`）是最高优先级的中断。任何时候，只要 UART Rx 输入产生四个错误条件（溢出错误（OE）、奇偶错误（PE）、帧错误（FE）和间隔中断（BI））中的任意一个，该中断标志将被置位。产生该中断的 UART Rx 错误条件可通过查看 `U0LSR[4:1]` 得到。当读取 `U0LSR` 时清除该中断。

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

CTI 中断（`U0IIR[3:1]=110`）为第二优先级中断。当 UART Rx FIFO 包含至少 1 个字符并且在接收 3.5 到 4.5 字符的时间内没有发生 UART Rx FIFO 动作时，该中断置位。UART Rx FIFO 的任何动作（读或写 UART RSR）都将清除该中断。在接收到的信息不是触发等级值的倍数时，CTI 中断将会清空 UART RBR。例如，如果一个外设想要发送一个 105 个字符的信息，触发等级值为 10 个字符；那么 CPU 将接收 10 个 RDA 中断，完成 100 个字符的传输；之后 CPU 将收到 1-5 个 CTI 中断（取决于服务程序），完成剩下 5 个字符的传输。

Table 126. UART 中断处理

U0IIR[3:0] 值 [1]	优先级	中断类型	中断源	中断复位
0001	-	无	无	-
0110	最高	RX 线状态 / 错误	OE [2] or PE [2] or FE [2] or BI [2]	读 U0LSR [2]

Table 126. UART 中断处理

U0IIR[3:0] 值 [1]	优先级	中断类型	中断源	中断复位
0100	第二	RX 数据有效	Rx 数据有效或 FIFO 达到触发等级 (U0FCR0=1)	读 U0RBR [3] 或 UART FIFO 达到触发等级
1100	第二	字符超时指示	Rx FIFO 包含至少 1 个字符且在一段时间内无字符输入或移出，该时间的长短取决于 FIFO 中的字符数以及设置的触发等级值（在 3.5 到 4.5 字符的时间之间）。 实际的时间为： [(字长度)*7- 2]*8+ [(触发值-字符数)*8 + 1] RCLKs	读 U0RBR [3]
0010	第三	THRE	THRE[2]	读 U0IIR[4] (如果是中断源) 或 THR 写操作

[1] 值 “0000”, “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”, “1101”, “1110”, “1111” are reserved.

[2] 详见 [Section 10.5.9 “UART Line Status Register \(U0LSR - 0x4000 8014, Read Only\)”](#)

[3] 详见 [Section 10.5.1 “UART Receiver Buffer Register \(U0RBR - 0x4000 8000, when DLAB = 0, Read Only\)”](#)

[4] 详见 [Section 10.5.5 “UART Interrupt Identification Register \(U0IIR - 0x4004 8008, Read Only\)”](#) 的 [Section 10.5.2 “UART Transmitter Holding Register \(U0THR - 0x4000 8000 when DLAB = 0, Write Only\)”](#)

UART THRE 中断 (U0IIR[3:1] = 001) 是第 3 优先级中断。当 UART THR FIFO 为空并且满足特定的初始化条件时，该中断激活。这些初始化条件将使 UART THR FIFO 被数据填充，以免在系统启动时产生许多 THRE 中断。在上一次 THRE=1 事件之后，若 U0THR 中没有至少 2 个字符，当 THRE=1 时，并在延时一个字符减去停止位之后触发 THRE 中断。在没有译码和 THRE 中断服务时，该延迟为 CPU 提供了将数据写入 U0THR 的时间。如果当前或曾经在 UART THR FIFO 中有两个或更多字符，而当前 U0THR 为空时，THRE 中断立即被设置。当 U0THR 写操作或 U0IIR 读操作发生，且 THRE 为最高优先级中断 (U0IIR[3:1]=001) 时，THRE 中断复位。

10.5.6 UART FIFO 控制寄存器 (U0FCR - 0x4000 8008, 只写)

U0FCR 控制 UART0 RX 和 TXFIFOs 的操作。

Table 127. UART FIFO 控制寄存器 (U0FCR – 地址 0x4000 8008, 只写) 位域描述

位	符号	值	描述	复位值
0	FIFOEn		FIFO 使能	0
		0	UART FIFO 被禁止。在应用中必须不能被使用。	
		1	高电允许对 UART Rx 和 Tx FIFO 以及 U0FCR[7:1] 的访问。该位必须设置以实现正确的 UART 操作。该位的任何变化都将使 UART FIFOs 自动清空。	
1	RXFIFO Res		RX FIFO 复位。	0
		0	对两个 UART FIFO 都无影响。	
		1	写 1 到 U0FCR[1]，将清零 UART Rx FIFO 中的所有字节并复位指针逻辑。该位自动清零。	
2	TXFIFO Res		TX FIFO 复位。	0
		0	对两个 UART FIFO 都无影响。	
		1	写 1 到 U0FCR[2]，将清零 UART Tx FIFO 中的所有字节并复位指针逻辑。该位自动清零。	
3	-	-	保留	0
5:4	-	-	保留，用户软件不要向保留位写入 1 从保留位读出的值未被定义	NA
7:6	RXTL		这两个位决定在激活中断之前，接收器 UART FIFO 必须写入多少个字符。	0
		0x0	触发点 0（1 个字符或 0x01）	
		0x1	触发点 1（4 个字符或 0x04）	
		0x2	触发点 2（8 个字符或 0x08）	
		0x3	触发点 3（14 个字符或 0x0E）	
31:8	-	-	保留	-

10.5.7 UART 线控制寄存器 (U0LCR - 0x4000 800C)

U0LCR 决定发送和接收数据字符的格式。

Table 128. 线控制寄存器 (U0LCR - 地址 0x4000 800C) 位域描述

位	符号	值	描述	复位值
1:0	WLS		字长选择	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 个停止位	
		1	2 个停止位 (如果 U0LCR[1:0]=00 则为 1.5 个)	

Table 128. 线控制寄存器 (U0LCR - 地址 0x4000 800C) 位域描述 ...continued

位	符号	值	描述	复位值
3	PE		奇偶校验使能	0
		0	禁止奇偶产生和校验	
		1	允许奇偶产生和校验	
5:4	PS		奇偶校验选择	0
		0x0	奇校验。发送字符和附带的校验位中 1 的个数为奇数	
		0x1	偶校验。发送字符和附带的校验位中 1 的个数为偶数	
		0x2	校验位强制为 “1”	
		0x3	校验位强制为 “0”	
6	BC		间隔控制	0
		0	禁止间隔发送	
		1	允许间隔发送。当 U0LCR[6] 为高电平有效时，输出引脚 UART TxD 强制为逻辑 0	
7	DLAB		除数锁存访问位	0
		0	禁止访问除数锁存	
		1	允许访问除数锁存	
31:8	-	-	保留	-

10.5.8 UART Modem 控制寄存器

U0MCR 允许 modem 的回环模式并控制 modem 的输出信号

Table 129. UART0 Modem 控制寄存器 (U0MCR - 地址 0x4000 8010) 位域描述

位	符号	值	描述	复位值
0	DTRC		modem 输出引脚 DTR 的源，该位在回环模式激活时读出为 0	0
1	RTSC		modem 输出引脚 RTS 的源，该位在回环模式激活时读出为 0	0
3:2	-		保留，用户软件不要向保留位写入 1，从保留位读出的值未定义	0

Table 129. UART0 Modem 控制寄存器 (U0MCR - 地址 0x4000 8010) 位域描述

位	符号	值	描述	复位值
4	LMS		回环模式选择。modem 回环模式提供了一个执行回环测试的诊断机制。0 发送器输出的串行数据在内部连接到接收器的串行输入端。输入脚 RxD 对回环模式无影响，输出脚 TxD 总保持为标记状态。4 个 modem 输入 (CTS, DSR, RI 和 DCD) 与外部断开。从外部来看，modem 的输出端 (RTS, DTR) 无效。在内部，4 个 modem 输出连接到 4 个 modem 输入。这样连接的结果是 U0MSR 的高 4 位由 U0MCR 的低 4 位驱动，而不是在正常模式下由 4 个 modem 输入驱动。这样在回环模式下，就可通过写 U0MCR 的低 4 位来允许 modem 状态中断的产生。	0
		0	禁止 modem 回环模式	
		1	允许 modem 回环模式	
5	-		保留，用户软件不要向保留位写入 1，从保留位读出的值未定义	0
6	RTSen		RTS 流控制	0
		0	禁止自动 RTS 流控制	
		1	允许自动 RTS 流控制	
7	CTSen		CTS 流控制	0
		0	禁止自动 CTS 流控制	
		1	允许自动 CTS 流控制	
31:8	-	-	保留	-

10.5.8.1 自动流控制

如果自动 RTS 模式被允许，那么 UART 的接收器 FIFO 硬件控制 UART 的 $\overline{\text{RTS}}$ 输出。如果自动 CTS 模式被允许，若 CTS 输入信号有效，那么 UART 的 U0TSR 硬件将只启动发送。

10.5.8.1.1 自动 RTS

设置 CTSen 位来允许自动 RTS 功能。自动 RTS 数据流控制在 U0RBR 模块中产生，并与已编程的接收 FIFO 触发等级相链接。如果自动 RTS 被允许，数据流按以下方式控制：

当接收 FIFO 的字节数到达设置的触发等级时， $\overline{\text{RTS}}$ 当接收 FIFO 的字节数到达设置的触发等级时， $\overline{\text{RTS}}$ （变为高）。发送 UART 可在达到触发等级后发送一个额外的字节（假设发送的 UART 有另一个字节要发送），因为它可能不知道 $\overline{\text{RTS}}$ 的无效直至它开始发送额外的字节之后。一旦接收 FIFO 到达原来的触发等级，RTS 就自动重新有效（变低）。RTS 的重新有效将指示正处在发送的 UART 继续发送数据。

如果自动 RTS 模式被禁止，那么 RTSen 位控制的 UART 的 $\overline{\text{RTS}}$ 输出。如果自动 RTS 模式被允许，那么硬件控制 $\overline{\text{RTS}}$ 输出且 $\overline{\text{RTS}}$ 的实际值将在 UART 的 RTSen 位中被复制。只要自动 RTS 被允许，则对于软件 RTSen 位的值只可读。

例如：假设 UART 工作于 type550 模式，在 U0FCR 中设置触发等级为 0x2，那么若自动 $\overline{\text{RTS}}$ 被允许，一旦接收 FIFO 包含 8 字节 (Table 127 on page 125). UART 将使 $\overline{\text{RTS}}$ 输出无效。一旦接收 FIFO 到达原来的触发等级：4 个字节，那么 $\overline{\text{RTS}}$ 输出就将重新有效。

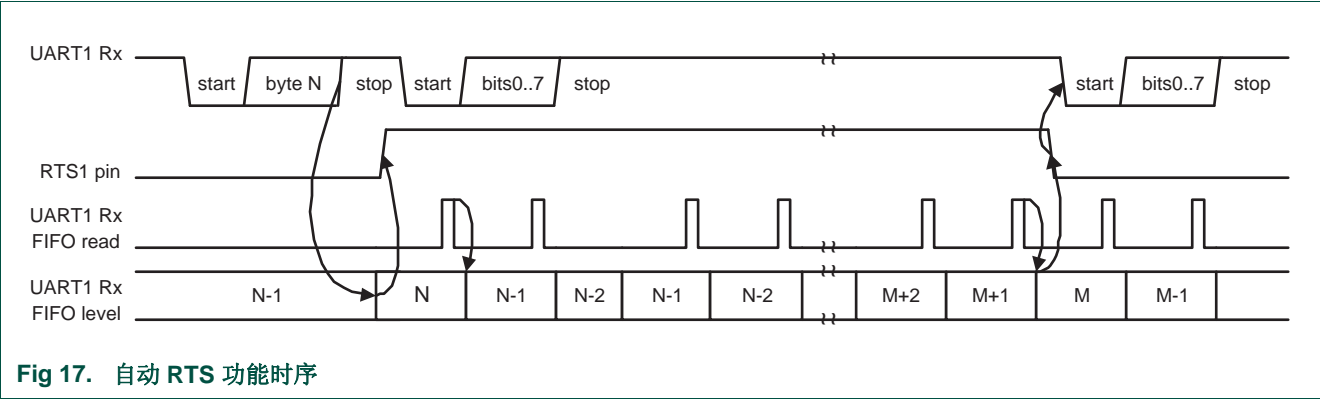


Fig 17. 自动 RTS 功能时序

10.5.8.1.2 自动 CTS

自动 CTS 功能通过设置 CTSen 位来允许。如果自动 CTS 被允许，则 U0TSR 模块中的发送器电路在发送下一个数据字节之前检查 $\overline{\text{CTS}}$ S 输入。当 $\overline{\text{CTS}}$ 有效（低电平）时，发送器发送下一个字节。为了停止发送器发送后面的字节， $\overline{\text{CTS}}$ 必须在当前发送的最后一个停止位的中间之前被释放。在自动 CTS 模式中 $\overline{\text{CTS}}$ 信号的变化不会触发 modem 状态中断，除非 CTS 中断允许位置位，但 U0MSR 中的 Delta CTS 位将被置位。Table 130 列出产生 modem 状态中断的条件。

Table 130. Modem 状态中断产生

允许 modem 状态中断 (U0IER[3])	CTSen (U0MCR[7])	CTS 中断允许 (U0IER[7])	Delta CTS (U0MSR[0])	Delta DCD 或后沿 RI 或 Delta DSR(U0MSR[3] 或 U0MSR[2] 或 U0MSR[1])	Modem 状态中断
0	x	x	x	x	否
1	0	x	0	0	否
1	0	x	1	x	是
1	0	x	x	1	是
1	1	0	x	0	否
1	1	0	x	1	是
1	1	1	0	0	否
1	1	1	1	x	是
1	1	1	x	1	是

自动 CTS 功能减少了主机系统的中断。当流控制允许时， $\overline{\text{CTS}}$ 状态变化不触发主机中断，因为设备自动控制其自身的发送器。没有自动 CTS，将导致发送器发送任何出现在 TX FIFO 中的数据，以及接收器溢出错误。Figure 18 为自动 CTS 功能的时序。

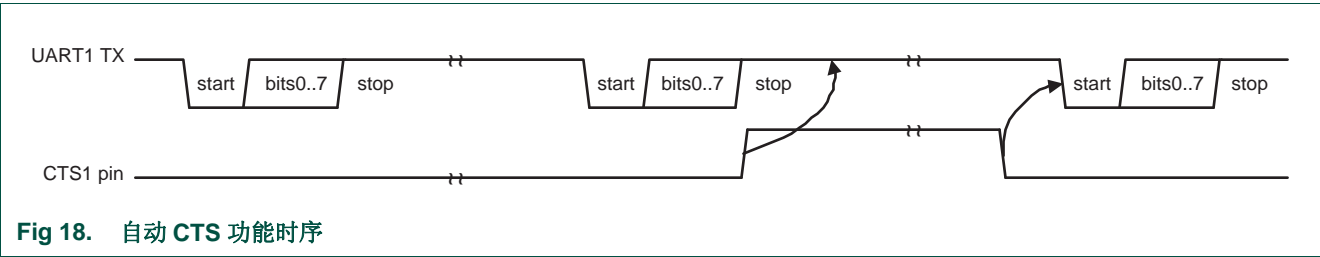


Fig 18. 自动 CTS 功能时序

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

10.5.9 UART 线状态寄存器 (U0LSR - 0x4000 8014, 只读)

U0LSR 为只读寄存器，提供 UART Tx 和 Rx 模块的状态信息。

Table 131. UART 线状态寄存器 (U0LSR - 地址 0x4000 8014, 只读) 位域描述

位	符号	值	描述	复位值
0	RDR		当 U0RBR 包含未读取的字符时，U0LSR[0] 置位；当 UART RBR FIFO 为空时，U0LSR[0] 清零。	0
		0	U0RBR 为空	
		1	U0RBR 包含有效数据	
1	OE		溢出错误。T 溢出错误条件在溢出错误发生后立即设置。读 U0LSR 操作将清零 U0LSR[1]。当 UART RSR 已经有新的字符组合而 UART RBR FIFO 已满时，U0LSR[1] 置位。此时 UART RBR FIFO 不会被覆盖，UART RSR 中的字符将丢失。	0
		0	溢出错误状态未被激活	
		1	溢出错误状态被激活	
2	PE		奇偶校验错误。当接收字符的校验位为错误状态时产生一个奇偶错误。读 U0LSR 操作将清零 U0LSR[2] 位。奇偶错误检测时间取决于 U0FCR[0]。 注：奇偶错误与 UART RBR FIFO 中顶部的字符相关	0
		0	奇偶错误状态未被激活	
		1	奇偶错误状态被激活	

Table 131. UART 线状态寄存器 (U0LSR - 地址 0x4000 8014, 只读) 位域描述 ...continued

位	符号	值	描述	复位值
3	FE		帧错误. 当接收字符的停止位为 0 时, 产生帧错误。读 U0LSR 操作将清零 U0LSR[3]。帧错误检测时间取决于 U0FCR0。当检测到一个帧错误时, Rx 将试图与数据重新同步, 并假设错误的停止位实际是一个超前的起始位。但是, 即使没有出现帧错误, 它也不能假设下一个接收到的字节是正确的。 注: 帧错误与 UART RBR FIFO 中顶部的字符相关	0
		0	帧错误状态未被激活	
		1	帧错误状态被激活	
4	BI		在发送整个字符 (起始位、数据位、校验位和停止位) 过程中 Rx D1 如果都保持逻辑 0, 则产生间隔中断。当检测到间隔条件时, 接收器立即进入空闲状态直到 Rx D1 变为全 1 状态。读 U0LSR 操作将清零该状态位。间隔检测的时间取决于 U0FCR[0]。 注: 间隔中断与 UART RBR FIFO 中顶部的字符相关	0
		0	间隔中断状态未被激活	
		1	间隔中断状态被激活	
5	THRE		当检测到 UART THR 空时, THRE 置位, U0THR 写操作将清零该位	1
		0	U0THR 包含有效数据	
		1	U0THR 为空	
6	TEMT		当 U0THR 和 U0TSR 为空时, THRE 置位; U0THR 或 U0TSR 包含有效数据将清零该位	1
		0	U0THR 和 / 或 U0TSR 包含有效数据	
		1	U0THR 和 U0TSR 为空	
7	RXFE		接收 FIFO 中的错误. 当一个带有 Rx 错误 (例如帧错误、奇偶错误或间隔中断) 的字符装入 U0RBR 时, U0LSR[7] 被置位。当读取 U0LSR 寄存器并且 UART FIFO 中不再有错误时, U0LSR[7] 被清零。	0
		0	U0RBR 中没有 UART Rx 错误, 或 U0FCR[0]=0	
		1	UART RBR 包含至少一个 UART Rx 错误 0	
31: 8	-	-	保留	-

10.5.10 UART Modem 状态寄存器

U0MSR 是一个只读寄存器，它提供 modem 输入信号的状态信息。U0MSR[3:0] 在读取 U0MSR 时被清零。需要注意的是，modem 信号对 UART 的操作没有直接影响。该寄存器用于帮助 Modem 信号的软件实现。

Table 132. UART Modem 状态寄存器 (U0MSR - 地址 0x4000 8018) 位域描述

位	符号	值	描述	复位值
0	D CTS		当输入 CTS 状态发生变化时，该位置位。读 U0MSR 时清零该位	0
		0	没有检测到 modem 输入 $\overline{\text{CTS}}$ 上的状态变化	
		1	检测到 modem 输入 $\overline{\text{CTS}}$ 上的状态变化	
1	DDSR		当输入 $\overline{\text{DSR}}$ 状态发生变化时，该位被置位。读 U0MSR 时清零该位。	0
		0	没有检测到 modem 输入 $\overline{\text{DSR}}$ 上的状态变化	
		1	检测到 modem 输入 $\overline{\text{DSR}}$ 上的状态变化	
2	TERI		当输入 $\overline{\text{RI}}$ 发生低到高的跳变时，该位置位。读取 U0MSR 时清零	0
		0	没有检测到 modem 输入 $\overline{\text{RI}}$ 上的状态变化	
		1	检测到 modem 输入 $\overline{\text{RI}}$ 上低到高的跳变。	
3	DDCD		当输入 $\overline{\text{DCD}}$ 的状态发生变化时，该位置位。读取 U1MSR 时清零。	0
		0	没有检测到 modem 输入 $\overline{\text{DCD}}$ 上的状态变化	
		1	检测到 modem 输入 $\overline{\text{DCD}}$ 上的状态变化。	
4	CTS		清除发送状态。输入信号 $\overline{\text{CTS}}$ 的补。在回环模式下，该位连接到 U0MCR[1]	0
5	DSR		数据设备就绪状态。输入信号 $\overline{\text{DSR}}$ 的补。在回环模式下，该位连 0 接到 U0MCR[0]	0
6	RI		响铃指示状态。输入信号 $\overline{\text{RI}}$ 的补。在回环模式下，该位连接到 0 U0MCR[2].	0
7	DCD		数据载波检测状态。输入信号 $\overline{\text{DCD}}$ 的补。在回环模式下，该位连 0 接到 U0MCR[3]	0
31: 8	-	-	保留	-

10.5.11 UART 暂存寄存器 (U0SCR - 0x4000 801C)

在 UART 操作时 U0SCR 无效。用户可自由对该寄存器进行读或写。中断接口不向主机提供 USCR 所发生的读或写操作的指示。

Table 133. UART UART 暂存寄存器 (U0SCR - 地址 0x4000 801C) 位域描述

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	-

10.5.12 UART 自动波特率控制寄存器 (U0ACR - 0x4000 8020)

为生成波特率，而对输入时钟 / 数据率进行测量的过程，由 UART 自动波特率控制寄存器 U0ACR 进行控制。用户可自由对该寄存器进行读或写。

Table 134. 自动波特率控制寄存器 (U0ACR - 地址 0x4000 8020) 位域描述

位	符号	值	描述	复位值
0	Start		开始位，自动波特率结束后该位自动清零	0
		0	自动波特率停止（自动波特率未运行）	
		1	自动波特率启动（自动波特率正在运行）。自动波特率运行位。该位在自动波特率结束后自动清零	
1	Mode		自动波特率模式选择位	0
		0	模式 0	
		1	模式 1	
2	AutoRestart		复位使能	0
		0	不复位	
		1	如果超时则复位（计数器在下一个 UARTRx 下降沿复位）	
7:3	-		保留，用户软件不要向保留位写入 1，从保留位读出的值未被定义	0
8	ABEOIntClr		动波特率中断结束清零位（仅可写访问）	0
		0	写 0 无影响	
		1	写 1 将在 U1IIR 中清除相应的中断	
9	ABTOIntClr		自动波特率超时中断清零位（仅可写访问）	0
		0	写 0 无影响	
		1	写 1 将在 U0IIR 中清除相应的中断	
31:10	-		保留，用户软件不要向保留位写入 1，从保留位读出的值未被定义	0

10.5.13 自动波特率

UART 自动波特率功能可用于测量基于“AT”协议 (Hayes 命令) 的输入波特率。如果允许，那么自动波特率特性将测量接收数据流的位时间，并设置除数锁存寄存器 U0DLM 和 U0DLL。

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

可通过 U0ACR 模式位来选择两种自动波特率测量模式。在模式 0 中，波特率在 UART Rx 引脚的两个连续的下降沿上测量（起始位的下降沿和最低位的下降沿）。在模式 1 中，波特率在 UART Rx 引脚的下降沿和后续的上升沿之间测量（起始位的长度）。

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

自动波特率功能可产生两种中断。

- 如果中断允许，那么将设置 U0IIR ABTOInt 中断（U0IER ABTOIntEn 被置位，且自动波特率测量计数器溢出）。
- 如果中断允许，那么将设置 U0IIR ABEOInt 中断（U0IER ABEOIntEn 被置位，且自动波特率成功完成）。

通过设置 U0ACR 相应的 ABTOIntClr 和 ABEOIntClr 位来清除自动波特率中断。

在自动波特率期间，分数波特率发生器被禁止 (DIVADDVAL=0)。也就是，当自动波特率被使用时，任何写 U0DLM 和 U0DLL 寄存器的操作都应在写 U0ACR 寄存器前完成。UART0 所支持的波特率最小和最大值由 UART_PCLK、数据位的个数、停止位和校验位决定。

(2)

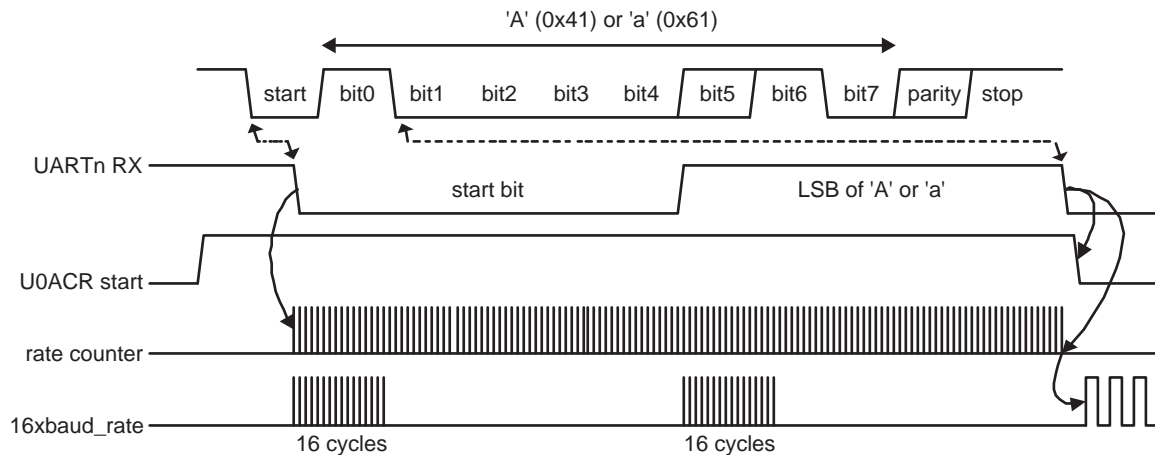
$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

10.5.14 自动波特率模式

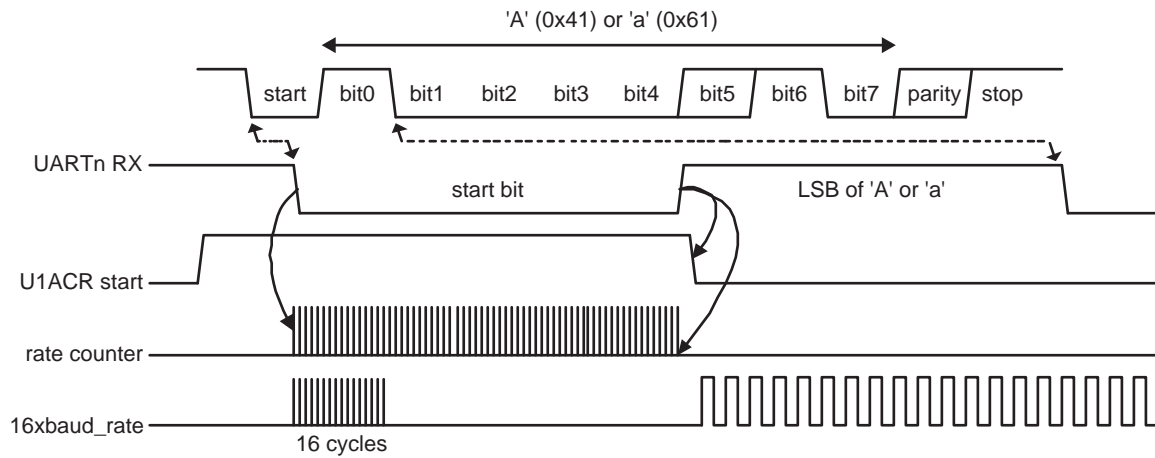
当软件正在期望“AT”命令时，用期望的字符格式配置 UART 并设置 U0ACR 起始位。这里可以不用关心除数锁存器 U0DLL 和 U0DLM 的初始值。由于“A”或“a”的 ASCII 编码是“A”=0x41 或“a”=0x61，因此 UART Rx 引脚可通过检测起始位以及所期望字符的 LSB 是两个下降沿来确定 AT 命令。当 U0ACR 起始位置位时，自动波特率协议将执行以下过程：

1. 在 U0ACR 起始位置位时，速率测量计数器复位且 UART U0RSR 复位。U0RSR 波特率切换为最高的速率。
2. UART Rx 引脚下降沿触发起始位的开始。速率测量计数器将开始对 PCLK 进行计数。
3. 在起始位的接收过程中，RSR 的波特率输入端将根据 UART 输入时钟频率产生 16 个脉冲，保证起始位存储在 U0RSR 中。

4. 在接收起始位（和模式 0 的字符 LSB）的过程中，速率计数器将按照预分频的 UART 输入时钟（UART_PCLK）增加。
5. 如果 Mode=0，那么速率计数器将在 UART Rx 引脚的下一个下降沿停止。如果 Mode=1，那么速率计数器将在 UART Rx 引脚的下一个上升沿停止。
6. 速率计数器被装入到 U0DLM/U0DLL，且波特率将自动切换为正常操作模式。设置完 U0DLM/U0DLL 后，如果使能，U0IIR ABEOInt(自动波特率结束中断) 将被置位。U0RSR 将继续接收“A/a”字符剩下的位。



a. 模式 0 (起始位和最低位用于自动波特率)



b. 模式 1 (仅起始位用于自动波特率)

Fig 19. 自动波特率 a) 模式 0、b) 模式 1 波形

10.5.15 UART 分数分频寄存器 (U0FDR - 0x4000 8028)

UART 分数分频寄存器 (U0FDR) 控制用于产生波特率的时钟预分频器，并可被用户读写。该预分频器根据指定的分数，使用 APB 时钟来产生一个输出时钟。

注意：如果分数分频被激活 (DIVADDVAL > 0) 且 DLM = 0, DLL 寄存器的值必须大于或等于 3。

Table 135. UART 分数分频寄存器 (U0FDR - address 0x4000 8028) 位域描述

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

该寄存器控制用于波特率生成的时钟预分频器。该寄存器的复位值保持为 UART 分数功能被禁止时的值，以确保在软件和硬件上与此特性的 UART 完全兼容。

可用下式计算 UART 波特率：

(3)

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

其中 UART_PCLK 为外围时钟，U0DLM 和 U0DLL 为标准 UART 波特率除数寄存器，DIVADDVAL 和 MULVAL 为 UART 分数波特率发生器特定参数。

MULVAL 和 DIVADDVAL 的值应遵循以下的条件：

- 1. 1 ≤ MULVAL ≤ 15
- 2. 0 ≤ DIVADDVAL ≤ 14
- 3. DIVADDVAL < MULVAL

正在发送 / 接收数据时，不应修改 U0FDR 的值，否则数据可能丢失或被破坏。

I 如果 U0FDR 寄存器值不遵循这两个要求，那么分数分频输出是未定义的。如果 DIVADDVAL 为 0，那么分数分频禁止且时钟将不会被分频。

10.5.15.1 波特率计算

无论是否有分数分频器，UART 都可以工作。在现实的应用程序中，可能有若干种分数分频器设置可实现要求的波特率。以下描述一种设置 DLM、DLL、MULVAL 和 DIVADDDVAL 值的方法，和要求的值相比，如下设置参数所得到的波特率相对误差小于 1.1%。

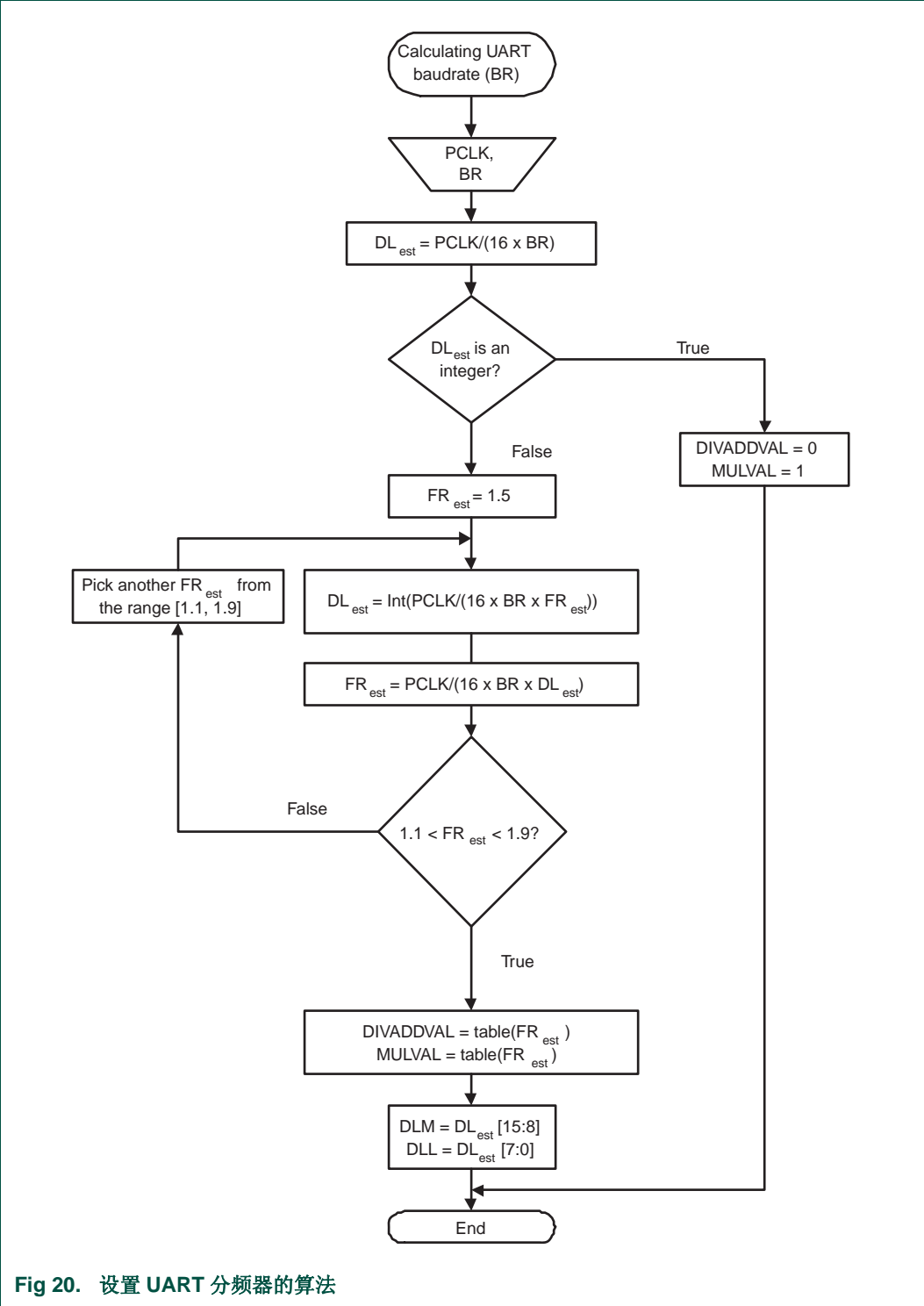


Fig 20. 设置 UART 分频器的算法

Table 136. 分数分频器设置查询表

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

10.5.15.1.1 例 1: UART_PCLK = 14.7456 MHz, BR = 9600

根据提供的算法 $DL_{est} = PCLK / (16 \times BR) = 14.7456 \text{ MHz} / (16 \times 9600) = 96$ 。而这个 DL_{est} 是一个整数， $DIVADDVAL = 0$ ， $MULVAL = 1$ ， $DLM = 0$ ，且 $DLL = 96$ 。

10.5.15.1.2 例 2: UART_PCLK = 12 MHz, BR = 115200

根据提供的算法 $DL_{est} = PCLK / (16 \times BR) = 12 \text{ MHz} / (16 \times 115200) = 6.51$ 。这个 DL_{est} 不是一个整数，下一步要估计 FR 的参数。使用初始估计 $FR_{est} = 1.5$ ，新的 $DL_{est} = 4$ 被计算出来，同时 FR_{est} 被计算出来为 $FR_{est} = 1.628$ 。因为 $FR_{est} = 1.628$ 是在 1.1 到 1.9 范围之内， $DIVADDVAL$ 和 $MULVAL$ 的值可以查表获取。

在 [Table 136](#) 中最接近 $FR_{est}=1.628$ 的值是 $FR=1.625$ 。它相当于 $DIVADDVAL=5$ 和 $MULVAL=8$ 。

根据这些发现，建议 UART 的设置如下： $DLM=0$ 、 $DLL=4$ 、 $DIVADDVAL=5$ 和 $MULVAL=8$ 。根据 [Equation 3](#)，UART 的波特率应该是 115384。这个波特率和原始指定的 115200 相对误差为 0.16%。

10.5.16 UART UART 发送允许寄存器 (U0TER - 0x4000 8030)

除配备了完整的硬件流控制 (如前描述的 auto-cts 和 auto-rts 机制) 之外，通过 U0TER 可实现软件流控制。当 $TxE_n=1$ 时，如果 UART 发送器有效将会持续发送数据。当 TxE_n 变成 0，则会停止 UART 发送。

虽然 [Table 137](#) 描述了如何使用 TxEn 位实现硬件流，但是还是强烈建议让 UART 硬件实现自动控制流，并限制 TxEn 实现软件控制流的范围。

[Table 137](#) 描述了如何使用 TXEn 位去实现软件控制流。

Table 137. UART 发送允许寄存器 (U0TER – 地址 0x4000 8030) 位域描述

位	符号	描述	复位值
6:0	-	保留，用户软件不要向该位写入 1。从保留位读出的值未被定义	NA
7	TXEN	该位为 1（复位值）时，如果以前的数据都被发送后，写入 THR 1 的数据被输出到 TxD 引脚。如果一个字符正在发送时该位清零则结束这个字符的发送，后面的字符也不再发送，直到该位重新被置位。换言之，该位为 0 将终止 THR 或 Tx FIFO 到发送移位寄存器的字符传输。当检测到硬件握手 Tx 允许信号 CTS 出错或利用软件握手接收到一个 XOFF 字符（DC3）时，软件将该位清零。当检测到正确的 Tx 允许信号或接收到 XON 字符（DC1）时软件又能将该位重新置位。	1
31:8	-	保留	-

10.5.17 UART RS485 控制寄存器 (U0RS485CTRL - 0x4000 804C)

U0RS485CTRL 寄存器控制 UART RS-485/EIA-485 模式的配置。

Table 138. UART RS485 控制寄存器 (U0RS485CTRL – 地址 0x4000 804C) 位域描述

位	符号	值	描述	复位值
0	NMMEN		NMM 使能	0
		0	RS-485/EIA-485 普通多点模式 (NMM) 被禁止	
		1	RS-485/EIA-485 普通多点模式 (NMM) 允许。在该模式下，当接收到的字节是地址时，将会导致 UART 设置校验错误并产生一个中断。	
1	RXDIS		接收器使能。	0
		0	接收器被允许	
		1	接收器被禁止	
2	AADEN		AAD 使能。	0
		0	自动地址检测 (AAD) 被禁止	
		1	自动地址检测 (AAD) 被允许	
3	SEL		选择方向控制引脚	0
		0	如果方向控制被允许 (位 DCTRL = 1)，引脚 $\overline{\text{RTS}}$ 被用于方向控制。	
		1	如果方向控制被允许 (位 DCTRL = 1)，引脚 $\overline{\text{DTR}}$ 被用于方向控制。	
4	DCTRL		自动方向控制使能	0
		0	禁止自动方向控制	
		1	允许自动方向控制	

Table 138. UART RS485 控制寄存器 (U0RS485CTRL – 地址 0x4000 804C) 位域描述 ...continued

位	符号	值	描述	复位值
5	OINV		极性控制。该位使在引脚 $\overline{\text{RTS}}$ (or $\overline{\text{DTR}}$) 上的方向控制极性反转。	0
		0	当发送器已将数据发送出去时，方向控制引脚会被置为逻辑 '0'。当数据的最后一位也被发送出去之后它会被置为逻辑 '1'	
		1	当发送器已将数据发送出去时，方向控制引脚会被置为逻辑 '1'。当数据的最后一位也被发送出去之后它会被置为逻辑 '0'	
31:6	-	-	保留，用户软件不要向保留位写入 1。从保留位读出的值为被定义。	NA

10.5.18 UART RS-485 地址匹配寄存器 (U0RS485ADRMATCH - 0x4000 8050)

U0RS485ADRMATCH 寄存器包含用于 RS-485/EIA-485 模式的地址匹配值。

Table 139. UART RS-485 地址匹配寄存器 (U0RS485ADRMATCH – 地址 0x4000 8050) 位域描述

位	符号	描述	复位值
7:0	ADRMATCH	包含地址匹配值	0x00
31:8	-	保留	-

10.5.19 UART1 RS-485 延迟值寄存器 (U0RS485DLY - 0x4000 8054)

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 $\overline{\text{RTS}}$ (or $\overline{\text{DTR}}$) 有效之间的延迟。这段延迟是在波特率时钟周期之内的。延迟可编程设置为 0-255 个位时间。

Table 140. UART RS-485 延迟值寄存器 (U0RS485DLY – 地址 0x4000 8045A) 位域描述

位	符号	描述	复位值
7:0	DLY	包含方向控制 (RTS 或 DTR) 延迟值。该寄存器与一个 8 位计数器连接进行工作	0x00
31:8	-	保留，用户软件不要向保留位写入 1。从保留位读出的值为被定义	NA

10.5.20 RS-485/EIA-485 操作模式

RS-485/EIA-485 的特性允许将 UART 配置成可寻址的从设备。可寻址的从设备是众多受控于主设备的从设备中的一个。

UART 主设备发送器通过检查校验位（第九位）是否设置为 1，来识别地址字符。对于数据字符，校验位是 0。

每个 UART 从设备接收器都能够被分配到一个不同的地址。从设备可以通过编程设置为人工或自动的方式拒绝不是它们自己地址的数据。

RS-485/EIA-485 普通多点模式 (NMM)

通过将 RS485CTRL 位设置为 0，来允许该模式。在该模式下，当一个接收的字符被检测为地址时，将会导致 UART 设置校验错误并产生一个中断。

如果接收器被禁止 (RS485CTRL 位 1 = '1')，任何接收到的数据字节都会被忽略且不会存储到 RXFIFO。当一个地址字节被检测到了 (parity 位 = '1')，它会被放置在 RXFIFO 同时生成一个 Rx 数据准备中断。处理器可以读地址字节，决定是否允许接收器去接受后续数据。

当接收器被允许时 (RS485CTRL 位 1 = '0')，所有接收到的数据都会被接受并存储到 RXFIFO，无论它们是数据或地址。当一个地址字符被接收时，将会产生一个校验错误中断，由处理器决定是否禁止接收器。

RS-485/EIA-485 自动地址检测 (AAD) 模式

当 RS485CTRL 寄存器位 0(9 位模式允许) 和 2(AAD 模式允许) 都被设置时，UART 处在自动地址检测模式。

在该模式，接收器将会把它接收到的任何地址字节 (parity = '1') 和 RS485ADRMATCH 寄存器中的 8 位值（可编程设置）进行比较。

若接收器被禁止 (RS485CTRL 位 1 = '1')，如果接收到字节是数据字节或是与 RS485ADRMATCH 值不匹配的地址字节，则都会被丢弃。

若检测到一个匹配的地址字符，它会带着校验位一起被压入 RXFIFO 中，同时接收器会自动被允许 (RS485CTRL 位 1 会被硬件清除)。接收器也会生成一个 Rx 数据准备中断。

当接收器被允许 (RS485CTRL 位 1 = '0') 的时候，所有接收的字节都会被接受并存储到 RXFIFO，直到收到一个与 RS485ADRMATCH 值不匹配的地址字节。当这个情况发生的时候，接收器会自动被硬件禁止 (RS485CTRL 位 1 会被设置)，不匹配的地址字符将不会被存储在 RXFIFO 中。

RS-485/EIA-485 自动方向控制

RS485/EIA-485 模式允许发送器自动控制 DIR 引脚的状态，它可以作为方向控制的输出信号。

通过设置 RS485CTRL 位 4 = '1' 允许这个特性。

如果允许方向控制，当 RS485CTRL 位 3 = '0' 时会使用 $\overline{\text{RTS}}$ 引脚，在 RS485CTRL 位 3 = '1' 时则使用 $\overline{\text{DTR}}$ 引脚。

当允许自动方向控制的时候，在 CPU 写数据进入 TXFIFO 时候被选择的引脚会有效 (驱动为低)。当数据最后一位被发送出去之后，引脚变为无效 (驱动为高)。见 RS485CTRL 寄存器的位 4 和 5。

除了回环模式之外，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 $\overline{\text{RTS}}$ 信号失效这一段时间。该段延迟时间可在 8 位 RS485DLY 寄存器中进行编程设置。延迟时间是在波特率时钟周期中的一部分，其值可以设置为 0-255 个位时间。

RS485/EIA-485 输出反转

引脚 $\overline{\text{RTS}}$ (or $\overline{\text{DTR}}$) 方向控制信号的极性可以通过对寄存器 U0RS485CTRL 第 5 位编程来反转。该位被设置，则当发送器中有数据正在等待被发送时，方向控制引脚会被驱动为逻辑 1。当数据的最后一位已被发送出去时，方向控制引脚会被设置为逻辑 0。

10.6 结构

UART 的结构框图如下所示。

APB 接口提供 CPU 或主机与 UART 之间的通信连接。

UART 接收器模块 U0RX 监视串行输入线 RxD0 的有效输入。UART RX 移位寄存器 (U0RSR) 通过 RXD0 接受有效的字符。当 U0RSR 接收到一个有效字符时，它将该字符传送到 UART RX 缓存寄存器 FIFO 中，等待 CPU 或主机通过主机接口进行访问。

UART 发送器模块 U0TX 接受 CPU 或主机写入的数据并将数据缓存到 UART TX 保持寄存器 FIFO (U0THR) 中。UART TX 移位寄存器 (U0TSR) 读取 U0THR 中的数据，并将数据通过串行输出引脚 TXD0 发出。

UART 波特率发生器模块 U0BRG 产生 UART TX 模块所使用的定时。U0BRG 模块时钟源为 UART_PCLK。主时钟是通过 U0DLL 和 U0DLM 寄存器所设定的除数来进行分频的。分频所得的时钟为 16 倍的采样时钟 NBAUDOUT。

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

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

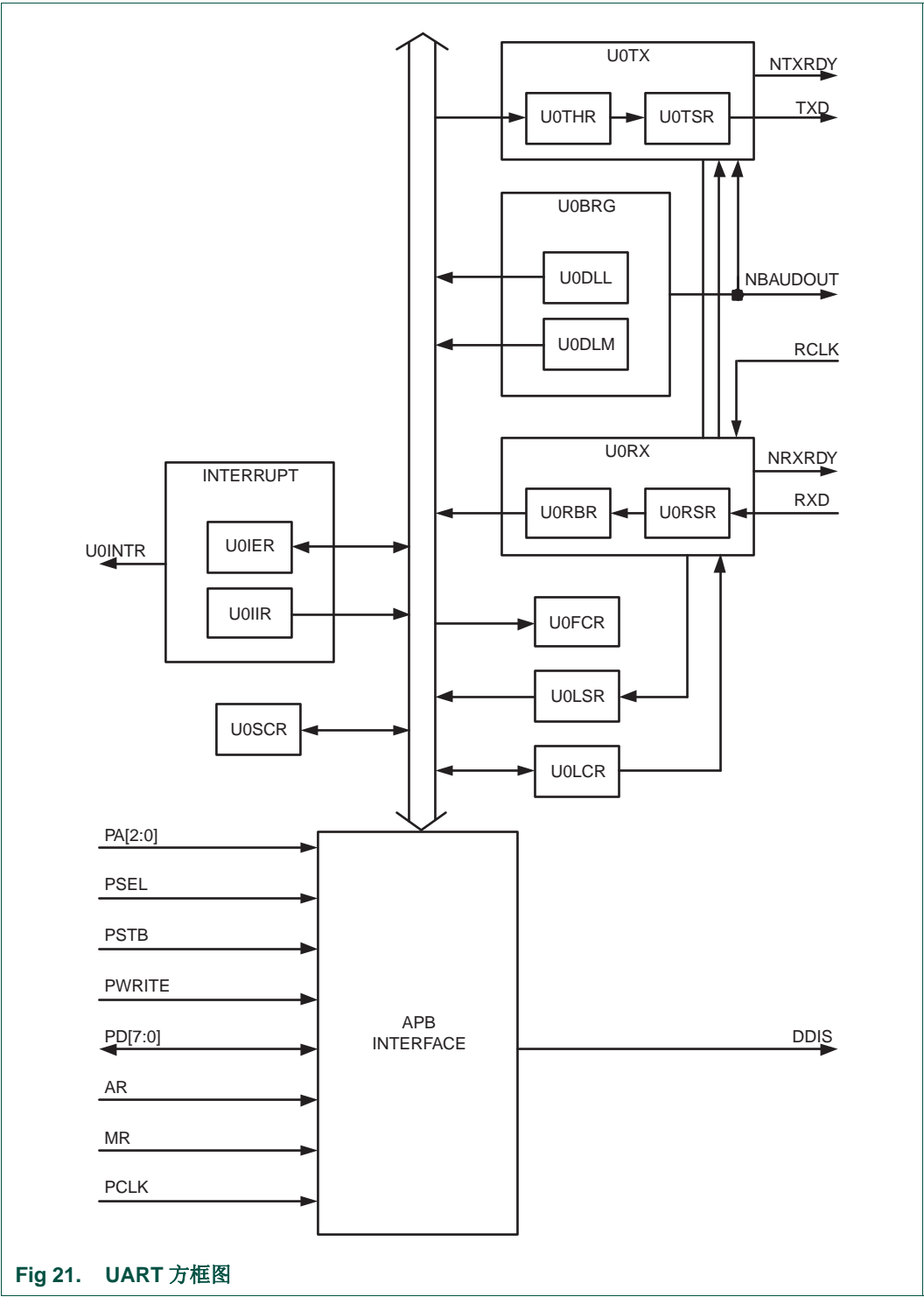


Fig 21. UART 方框图

11.1 如何阅读本章

所有 LPC111x 系列处理器的 SPI 模块均相同。第二个 SPI 模块 SPI1，只存在于 LQFP48 和 PLCC44 封装中，在 HVQFN33 封装中没有。

备注：两个 SPI 模块都包含全部的 SSP 特征集，所有相关寄存器都使用 SSP 前缀命名。

11.2 基本配置

SPI0/1 通过以下寄存器进行配置：

1. 引脚：SPI 的引脚必须在 IOCONFIG 寄存器中进行配置。另外，通过 IOCON_LOC 寄存器来选择为 SCK0 功能。（见 [Section 7.4](#)）。
2. 电源：SYSAHBCLKCTRL 寄存器中 11 位和 18 位。（[Table 21](#)）。
3. 外设时钟：可以允许软件写入 SSP0/1CLKDIV 寄存器来配置 SPI0/1 外设时钟。（[Section 3.5.15](#) 和 [Section 3.5.17](#)）。
4. 复位：在访问 SPI 模块之前，要确保 PRESETCTRL 寄存器中 SSP_RST_N 位被设置位 1。（[Table 9](#)）这将拉高 SPI 模块的复位信号。

11.3 特征

- 兼容 Motorola SPI、4 线 TI SSI 和美国国家半导体公司的 Microwire 总线。
- 同步串行通信。
- 支持主机和从机操作。
- 收发均有 8 帧 FIFO。
- 每帧有 4-16 位数据。

11.4 基本描述

SPI/SSP 是一个同步串行端口（SSP）控制器，可控制 SPI、4 线 SSI 和 Microwire 总线。它可以与总线上的多个主机和从机相互作用。在数据传输过程中，总线上只能有一个主机与一个从机进行通信。原则上数据传输是全双工的，4 ~ 16 位帧的数据由主机发送到从机或由从机发送到主机。但实际上，大多数情况下只有一个方向上的数据流包含有意义的数

LPC111x 系列处理器有两个 SPI/ 同步串行端口控制器。

11.5 引脚描述

Table 141. SPI 引脚描述

引脚名称	类型	接口引脚名称 / 功能			引脚描述
		SPI	SSI	Microwire	
SCK0/1	I/O	SCK	CLK	SK	串行时钟 .SCK/CLK/SK 是用来同步数据传输的时钟信号。它由主机驱动，从机接收。当使用 SPI 接口时，时钟可编程为高电平有效或低电平有效，否则总是高电平有效。SCK 仅在数据传输过程中切换。在其它时间里，SPI/SSP 接口保持无效状态或不驱动它（使其处于高阻态）。
SSEL0/1	I/O	SSEL	FS	CS	帧同步 / 从机选择 当 SPI/SSP 接口是总线主机时，它在串行数据启动前驱动该信号为有效状态。在数据发送出去之后又将该信号恢复为无效状态。该信号的有效状态根据所选的总线和模式可以是高或低。当 SPI/SSP 接口作为总线从机时，该信号根据使用的协议来判断主机数据的存在。 当只有一个总线主机和一个总线从机时，来至主机的帧同步信号或从机选择信号直接与从机相应的输入相连。当总线上接有多个从机时，需要管理好这些从机的帧选择 / 从机选择输入，以免一次传输有多个从机响应。
MISO0/1	I/O	MISO	DR(M) DX(S)	SI(M) SO(S)	主机输入从机输出 . MISO 信号线从从机传送串行数据传送到主机。当 SPI/SSP 作为从机，串行数据从该信号输出。当 SPI/SSP 作为主机，从该信号得到串行数据时钟。当 SPI/SSP 作为从机但未被 FS/SSEL 选定，它不会驱动该信号（保持高阻态）。
MOSI0/1	I/O	MOSI	DX(M) DR(S)	SO(M) SI(S)	主机输出从机输入 MOSI 信号线从主机传送串行数据到从机。当 SPI/SSP 作为主机，串行数据从该信号输出。当 SPI/SSP 作为从机，从该信号得到串行数据时钟。

备注： SCK0 功能在三个不同的引脚上复用（在 HVQFN 封装上有两个引脚）。通过设置 IOCON_L0C 寄存器（见 [Section 7.4](#)）来选择一个引脚作为 SCK0 功能，另外在 IOCON 寄存器中设置功能。SCK1 引脚没有复用。

11.6 寄存器描述

SPI 控制器寄存器地址如 [Table 142](#) 和 [Table 143](#)。

复位值已经使用位的数据，并不包括保留位的内容。

备注： 使用 SSP 前缀的寄存器名称，表示 SPI 控制器完全兼容 SSP 功能。

Table 142. 寄存器概览：SPI0 (基址 0x4004 0000)

名字	访问	地址偏移量	描述	复位值
SSP0CR0	R/W	0x000	控制寄存器 0。选择串行时钟频率，总线类型和数据长度。	0
SSP0CR1	R/W	0x004	控制寄存器 1。选择主机 / 从机和其他模式。	0
SSP0DR	R/W	0x008	数据寄存器。写满将发送 FIFO，读空将接收 FIFO。	0
SSP0SR	RO	0x00C	状态寄存器	0x0000 0003
SSP0CPSR	R/W	0x010	时钟预分频寄存器。	0
SSP0IMSC	R/W	0x014	中断屏蔽设置和清零寄存器。	0
SSP0RIS	RO	0x018	原始中断状态寄存器。	0x0000 0008
SSP0MIS	RO	0x01C	屏蔽中断状态寄存器。	0
SSP0ICR	WO	0x020	SSPICR 中断清零寄存器。	NA

Table 143. 寄存器概览：SPI1 (基址 0x4005 8000)

名字	访问	地址偏移量	描述	复位值
SSP1CR0	R/W	0x000	控制寄存器 0。选择串行时钟频率，总线类型和数据长度。	0
SSP1CR1	R/W	0x004	控制寄存器 1。选择主机 / 从机和其他模式。	0
SSP1DR	R/W	0x008	数据寄存器。写满将发送 FIFO，读空将接收 FIFO。	0
SSP1SR	RO	0x00C	状态寄存器	0x0000 0003
SSP1CPSR	R/W	0x010	时钟预分频寄存器。	0
SSP1IMSC	R/W	0x014	中断屏蔽设置和清零寄存器。	0
SSP1RIS	RO	0x018	原始中断状态寄存器。	0x0000 0008
SSP1MIS	RO	0x01C	屏蔽中断状态寄存器。	0
SSP1ICR	WO	0x020	SSPICR 中断清零寄存器。	NA

11.6.1 SPI/SSP 控制寄存器 0

该寄存器控制 SPI/SSP 控制器的基本操作。

Table 144: SPI/SSP 控制寄存器 0 (SSP0CR0 - 地址 0x4004 0000, SSP1CR0 - 地址 0x4005 8000)
位描述

位	符号	值	描述	复位值
3:0	DSS		数据长度选择。该字段控制着每帧传输的位数目。不支持且不使用值 0000-0010。	0000
		0x3	4 位传输	
		0x4	5 位传输	
		0x5	6 位传输	
		0x6	7 位传输	
		0x7	8 位传输	
		0x8	9- 位传输	
		0x9	10 位传输	
		0xA	11 位传输	
		0xB	12 位传输	
		0xC	13 位传输	
		0xD	14- 位传输	
		0xE	15- 位传输	
		0xF	16 位传输	
5:4	FRF		帧格式	00
		0x0	SPI	
		0x1	TI	
		0x2	Microwire	
		0x3	不支持且不应使用这个组合。	
6	CPOL		时钟输出极性。该位只用于 SPI 模式。	0
		0	SPI 控制器使总线时钟在两帧传输之间保持低电平。	
		1	SPI 控制器使总线时钟在两帧传输之间保持高电平。	
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个时钟跳变沿捕获串行数据，也就是说，传输远离时钟线的帧间状态。	
		1	SPI 控制器在帧传输的第二个时钟跳变沿捕获串行数据，也就是说，传输紧邻时钟线的帧间状态。	
15:8	SCR		串行时钟频率。SCR 之值为总线上每传输一个数据位所对应的预分频时钟数减 1。假设 CPSDVSR 为预分频器的分频值，APB 时钟 PCLK 为预分频器的时钟，则位频率为 PCLK / (CPSDVSR × [SCR+1])。	0x00
		-	保留	
31:16	-	-	保留	-

11.6.2 SPI/SSP0 Control Register 1

该寄存器控制着 SPI/SSP 控制器的工作方式的某些方面。

Table 145: SPI/SSP 控制寄存器 1 (SSP0CR1 - 地址 0x4004 0004, SSP1CR1 - 地址 0x4005 8004) 位描述

位	符号	值	描述	复位值
0	LBM		回环模式	0
		0	正常操作模式	
		1	串行输入脚同时也是串行输出脚（MOSI 或 MISO），而不是仅作为串行输入脚（MISO 或 MOSI 分别起作用）。	
1	SSE		SPI 使能	0
		0	禁止 SPI 控制器	
		1	SPI 控制器可与串行总线上的其它器件相互通信。在设置该位之前，软件应将合适的控制信息写入其它 SPI/SSP 寄存器和中断控制器寄存器。	
2	MS		主机 / 从机模式。该位只能在 SSE 位为 0 时写入。	0
		0	SPI 控制器作为总线主机，驱动 SCLK、MOSI 和 SSEL 线并接收 MISO 线。	
		1	SPI 控制器作为总线从机，驱动 MISO 线并接收 SCLK、MOSI 和 SSEL 线。	
3	SOD		从机输出禁止。该位只与从机模式有关（MS=1）。如果该位为 1，将阻塞 SSI 控制器驱动发送数据线（MISO）。	0
31:4	-		保留，用户软件不要向保留位写入 1。从保留位读出的值未定义。	NA

11.6.3 SPI/SSP 数据寄存器

软件可将要发送的数据写入该寄存器，或从该寄存器读出接收到的数据。

Table 146: SPI/SSP 数据寄存器 (SSP0DR - 地址 0x4004 0008, SSP1DR - 地址 0x4005 8008) 位描述

位	符号	描述	复位值
15:0	DATA	<p>写：当状态寄存器的 TNF 位为 1 时，指示 Tx FIFO 未滿，软件可将要发送的帧数据写入该寄存器。如果 Tx FIFO 以前为空，且总线上的 SPI 控制器不忙，则立刻开始发送数据；否则，写入该寄存器的数据要等到所有数据发送（或接收）完后才能发送。如果数据长度小于 16 位，软件必须对数据进行右对齐后再写入该寄存器。</p> <p>当状态寄存器的 RNE 位为 1 时，指示 Rx FIFO 不为空，软件可读取该寄存器。软件读取该寄存器时，SSP 控制器将返回 Rx FIFO 中最早收到的一帧数据。如果数据长度小于 16 位，该字段的数据必须进行右对齐，高位补零。</p>	0x0000
31:16	-	保留	-

11.6.4 SPI/SSP 状态寄存器

该只读寄存器反映了 SSP 控制器的当前状态。

Table 147: SPI/SSP 状态寄存器 (SSP0SR - 地址 0x4004 000C, SSP1SR - 地址 0x4005 800C) 位描述

位	符号	描述	复位值
0	TFE	发送 FIFO 空。发送 FIFO 为空时该位为 1，反之为 0	1
1	TNF	发送 FIFO 未滿。Tx FIFO 满时该位为 0，反之为 1。	1
2	RNE	接收 FIFO 非空。接收 FIFO 为空时该位为 0，反之为 1。	0
3	RFF	接收 FIFO 满。接收 FIFO 满时该位为 1，反之为 0。	0
4	BSY	忙。SPI 控制器空闲时该位为 0，当前正在发送 / 接收一帧数据和 / 或 Tx FIFO 非空时该位为 1。	0
31:5	-	保留。用户软件不要向保留位写入 1。从保留位读出的值未定义。	NA

11.6.5 SPI/SSP 时钟预分频寄存器

SPI_PCLK 到 SPI 预分频器时钟的分频系数，由该寄存器控制。换言之，SSPCR0 寄存器中 SCR 系数，决定位时钟。

Table 148: SPI/SSP 时钟预分频寄存器 (SSP0CPSR - 地址 0x4004 0010, SSP1CPSR - 地址 0x4005 8010) 位描述

位	符号	描述	复位值
7:0	CPSDVS	这是 2 到 254 之间的一个偶数，SPI_PCLK 经分频后得到预分频器输出时钟。位 0 读总是为 0。	0
31:8	-	保留	-

重要提示：必须正确初始化 SSPnCPSR，否则 SPI 控制器将不可能正常发送数据。

在从机模式时，SPI 时钟频率由主机提供，不能超过 [Section 3.5.15](#) 中选定的 SPI 外设时钟频率的 1/12。寄存器 SSPnCPSR 中的值与时钟频率无关。

主机模式，CPSDVS_{min} = 2 或更高（仅限偶数）。

11.6.6 SPI/SSP 中断屏蔽 设置 / 清除寄存器

该寄存器控制 SPI 控制器中 4 个可能的中断条件是否已经被允许。注意 ARM 使用 “masked” 在经典计算机技术中的意思相反，在经典计算机技术中 “masked” 指 “disabled”，而 ARM 使用 “masked” 表示 “enabled”。为避免引起歧义这里将不使用 “masked” 一词。

Table 149: SPI/SSP 中断屏蔽设置/清除寄存器 (SSP0IMSC - 地址 0x4004 0014, SSP1IMSC - 地址 0x4005 8014) 位描述

位	符号	描述	复位值
0	RORIM	软件设置该位来允许接收溢出中断，当 Rx FIFO 满时又完成另一帧的接收时该位置位。ARM 特别指出发生接收溢出时，新数据帧会将前面的数据帧覆盖。	0
1	RTIM	软件设置该位来允许接收超时中断，当 Rx FIFO 非空且在“超时周期”之内没有接收到任何数据，就会产生接收超时。	0
2	RXIM	软件置位该位，使得当 Rx FIFO 至少有一半为满时触发中断。	0
3	TXIM	软件置位该位，使得当 Tx FIFO 至少有一半为空时触发中断。	0
31:4	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未定义。	NA

11.6.7 SPI/SSP 原始中断状态寄存器

该只读寄存器的每个位在相应中断条件出现后产生 1，而不管该中断是否在 SSPIMSC 寄存器中被允许。

Table 150: SPI/SSP 原始中断状态寄存器 (SSP0RIS – 地址 0x4004 0018, SSP1RIS- 地址 0x4005 8018) 位域描述

位	符号	描述	复位值
0	RORRIS	当 Rx FIFO 满、又接收到另一帧数据时该位置位。ARM 特别指出，此时接收到的新数据帧会将前面的数据帧覆盖。	0
1	RTRIS	如果 Rx FIFO 不为空，且在“超时周期”之内没有被读时，该位置位。主从模式的超时期限是一样的，它由 SSP 的波特率：32 bits at PCLK / (CPSDVSR × [SCR+1])。	0
2	RXRIS	当 Rx FIFO 至少一半为满时该位置位。	0
3	TXRIS	当 Tx FIFO 至少一半为空时该位置位。	1
31:4	-	保留，用户软件不要向保留位写入 1。从保留位读出的值未定义。	NA

11.6.8 SPI/SSP 屏蔽中断状态寄存器

当一个中断条件出现且相应的中断在 SSPIMSC 中被允许时，该只读寄存器中对应位置位。当产生 SPI 中断时，中断服务程序可通过读该寄存器来判断中断源。

Table 151: SPI/SSP 屏蔽中断状态寄存器 (SSP0MIS – 地址 0x4004 001C, SSP1MIS – 地址 0x4005 801C) 位域描述

位	符号	描述	复位值
0	RORMIS	当 Rx FIFO 满时又接收到另一帧数据，并且中断被允许 时该位 为 1。	0
1	RTMIS	Rx FIFO 非空，在 “ 超时周期 ” 之内未被读，且中断被允许时， 该位为 1。	0
2	RXMIS	当 Rx FIFO 至少一半为满且该中断被允许时， 该位为 1。	0
3	TXMIS	当 Tx FIFO 至少一半为空且该中断被允许时， 该位为 1。	0
31:4	-	保留，用户软件不要向保留位写入 1。从保留位读出的的值未 定义。	NA

11.6.9 SPI/SSP 中断清除寄存器

软件可以写一个或多个 1 到该只写寄存器，以清除 SPI 控制器中相应的中断条件 。注 意，另外两个中断条件可以通过写或读适当的 FIFO 清除，也可以通过清除 SSPIMSC 寄存器 中相应的位来禁止。

Table 152: SPI/SSP中断清除寄存器 (SSP0ICR - 地址0x4004 0020, SSP1ICR - 地址0x4005 8020) 位描述

位	符号	描述	复位值
0	RORIC	写 1 到该位清除 “RxFIFO 满时，接收到帧 ” 中断	NA
1	RTIC	写 1 到该位清除 "Rx FIFO 非空，超时周期内未读到数据 ” 中断 NA 主从模式的超时期限是一样的，它由 SSP 的波特率：32 bits at PCLK / (CPSDVSR × [SCR+1])	
31:2	-	保留，用户软件不要向保留位写入 1。从保留位读出的的值未定 义。	NA

11.7 功能描述

11.7.1 T I 同步串行帧格式

Figure 22 所示为 SPI 模块支持的 4 线 TI 同步串行帧格式 。

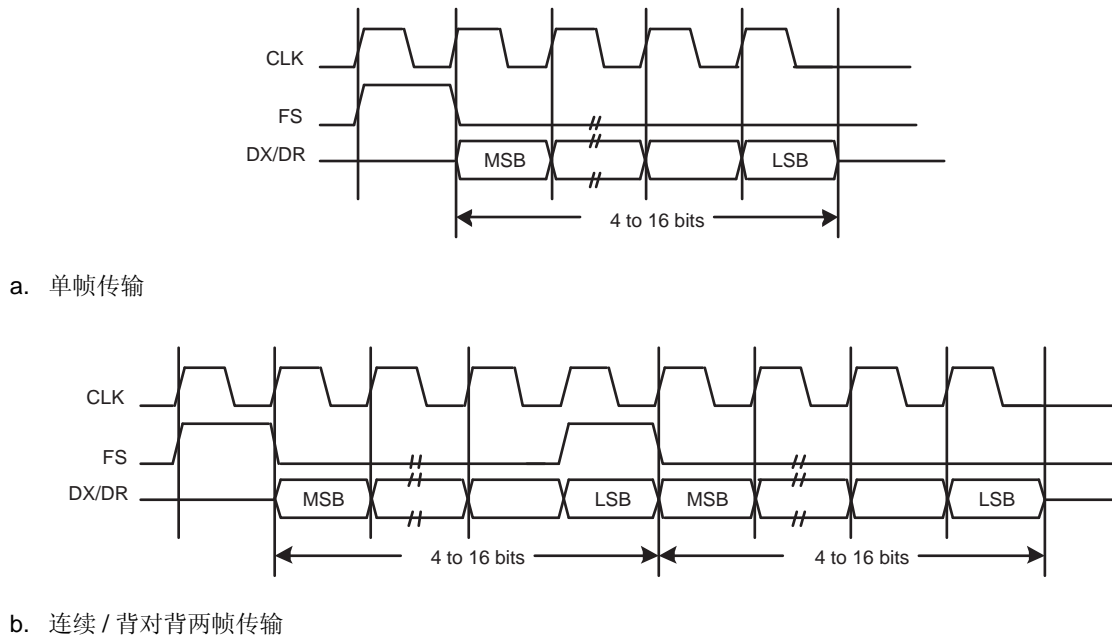


Fig 22. TI 同步串行数据帧格式: a) 单帧 和 b) 连续 / 背对背两帧传输

设备若在该模式下被配置为主机，每当 SSP 空闲时，CLK 和 FS 强制为低，发送数据线 DX 为三态模式。一旦发送 FIFO 的底部装入了数据，FS 立即变为高电平，并维持一个 CLK 周期。需发送的数据位从发送 FIFO 转移到串行移位寄存器。下一个 CLK 上升沿到来时，4 位到 16 位数据帧的最高位从 DX 引脚移出。同样，接收数据最高位也由片外串行从设备从 DR 引脚移入。

SSP 和片外串行从器件在每个 CLK 的下降沿将数据移位到它们的串行移位寄存器中。当最低位 (LSB) 被锁存后，当 CLK 上升沿到来时，接收到的数据从串行移位寄存器传送到接收 FIFO。

11.7.2 SPI frame format

当 SSEL 信号作为从机选择信号时，SPI 接口是一个 4 线接口。SPI 格式的主要特征是：SCK 信号的无效状态和相位可以通过设置 SSPCR0 寄存器中的 CPOL 和 CPHA 位来改变。

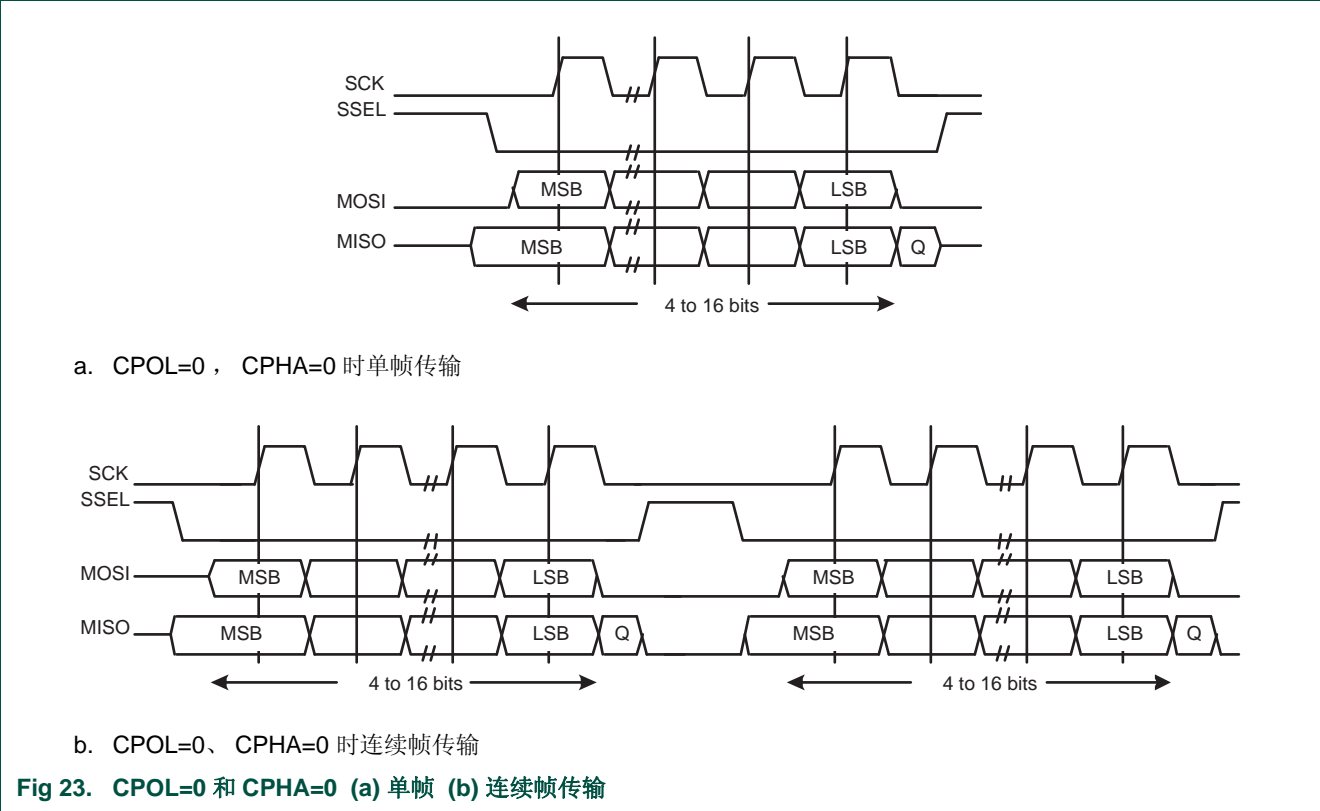
11.7.2.1 Clock Polarity (CPOL) and Phase (CPHA) control

当 CPOL 时钟极性控制位为低，SCK 引脚产生稳定的低电平。如果 CPOL 时钟极性控制位为高，数据未发送时，CLK 引脚产生一个稳定的高电平。

CPHA 控制位选择采样数据的时钟边沿，并允许它改变状态。通过允许和不允许在第一个数据捕获时钟边沿采样，它将对通讯时所传输的第一个位产生重要影响。若 CPHA 相位控制位为低，数据在第一个时钟跳变沿被采样；若 CPHA 时钟相位控制位为高，数据在第二个时钟跳变沿被采样。

11.7.2.2 CPOL=0, CPHA=0 的帧格式

CPOL = 0, CPHA = 0 时单 SPI 帧和连续 SPI 帧传输的时序如 [Figure 23](#). 所示。



该配置下，在空闲周期内：

- CLK 信号强制为低。
- SSEL 强制为高。
- 发送引脚 MOSI/MISO 处于高阻态。

如果 SPI/SSP 被允许并且发送 FIFO 中有效数据，SSEL 主信号驱动为低指示数据发送开始。这使主机的 MOSI 被允许，从机的数据也发送到主机的 MISO 引脚上。

半个 SCK 周期后，主机的有效数据传输到 MOSI 引脚。这时主机和从机数据都已经设置好，再经过半个 SCK 周期，SCK 引脚信号变为高。

数据在 SCK 信号的上升沿被捕获，保持到 SCK 的下降沿。

发送单个帧时，当数据帧的所有位发送完，最后一个数据位被捕获后的一个 SCK 周期内，SSEL 恢复空闲高电平状态。

但是，在连续帧的发送过程中，每个数据帧之间 SSEL 信号必须为高。这是因为当 CPHA 位为 0 时，从机选择引脚将数据冻结在串行外围寄存器中，不允许其改变。因此，在每次数据帧传输之间，主设备必须拉高从器件的 SSEL 引脚来允许串行外设数据的写操作。当连续帧传输结束，最后一位被捕获后一个 SCK 周期内，SSEL 返回到空闲状态。

11.7.2.3 CPOL=0, CPHA=1 的 SPI 帧格式

CPOL = 0, CPHA = 1 时 SPI 帧传输时序如 [Figure 24](#) 所示，包括单帧传输和连续传输两种模式。

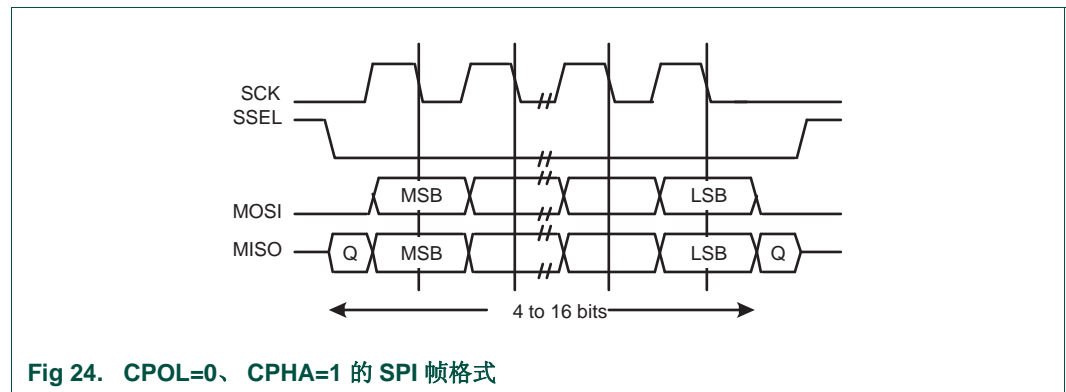


Fig 24. CPOL=0、CPHA=1 的 SPI 帧格式

该配置下，在空闲周期内：

- CLK 信号强制为低。
- SSEL 强制为高。
- 发送引脚 MOSI/MISO 处于高阻态。

如果 SPI/SSP 被允许并且发送 FIFO 中的有效数据，则 SSEL 被拉为低表示开始发送数据。主机 MOSI 引脚被允许。在半个 SCK 周期之后，主机和从机中的有效数据分别被允许输出到各自的发送线上。同时，SCK 出现上升沿跳变。

然后，数据在 SCK 信号的下降沿被捕获并保持到 SCK 信号的下一个上升沿。

在单个字的传输过程中，当所有位传输结束后，最后一位被捕获后的一个 SCK 周期之后，SSEL 引脚返回到空闲的高电平状态。

对于连续背对背帧的传输，SSEL 在两个连续的数据字传输之间保持低电平，传输终止与单字传输一样。

11.7.2.4 CPOL = 1, CPHA = 0 的帧格式

CPOL=1, CPHA=0 时单 SPI 帧和连续 SPI 帧传输时序如 [Figure 25](#) 所示。

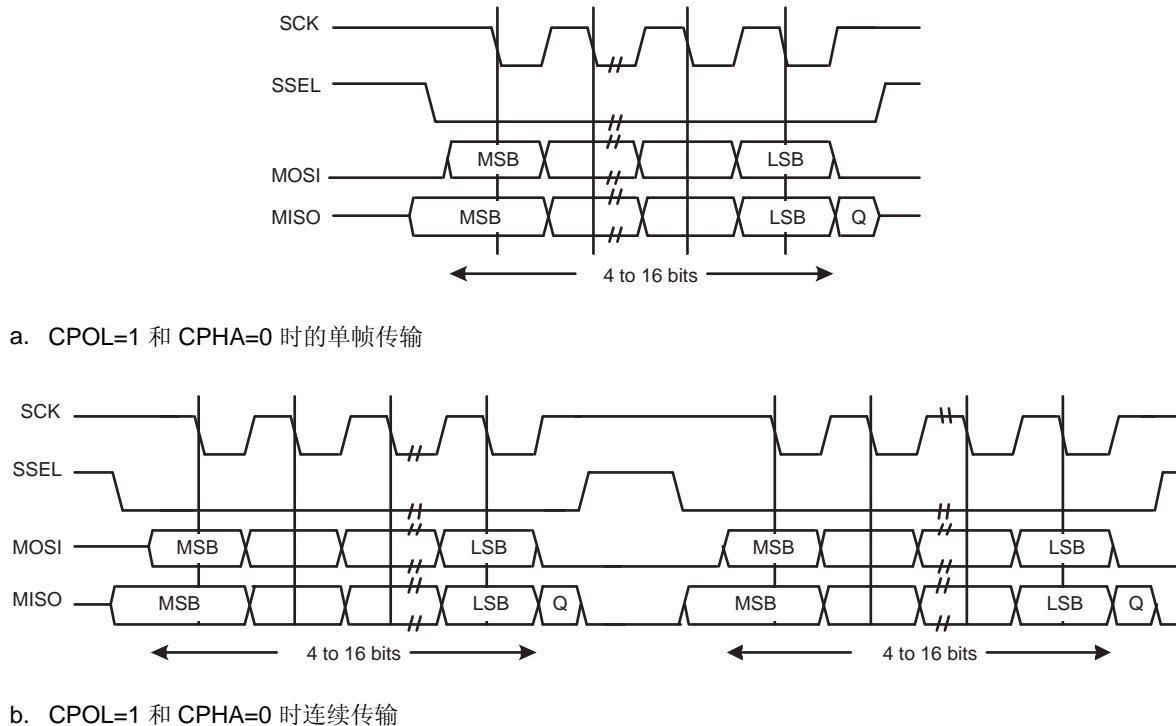


Fig 25. CPOL = 1 和 CPHA = 0 的 (a) 单帧 (b) 连续帧传输

该配置下，在空闲周期内：

- CLK 信号强制为高。
- SSEL 强制为高。
- 发送 MOSI/MISO 引脚处于高阻态。

如果 SPI/SSP 被允许，且发送 FIFO 中有有效数据，则 SSEL 被拉为低表示开始发送数据，这使得从机数据立即被传输到主机的主 MOSI 线上。主机的主 MOSI 引脚被允许。

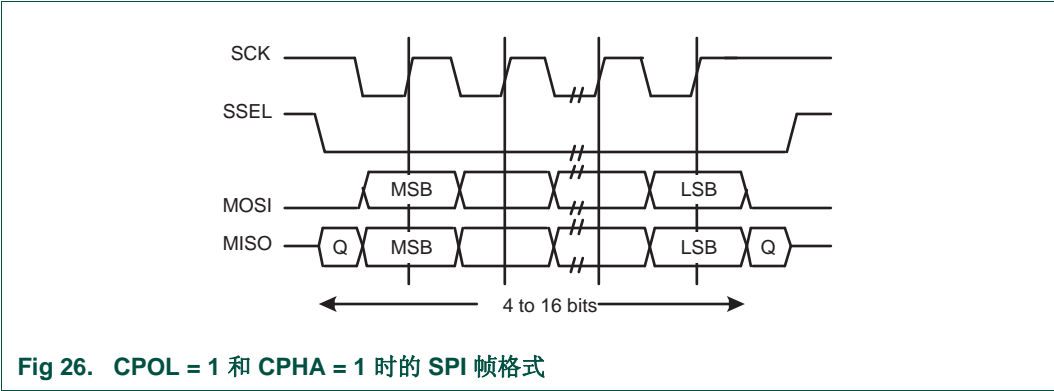
半个 SCK 周期后，有效的主机数据被传输到 MOSI 线。由于主机和从机数据都被设置，再过半个 SCK 周期后 SCK 引脚将变低。这意味着数据在 SCK 信号的下降沿被捕获，并保持到 SCK 的下一个上升沿。

I 在发送单个字时，当数据字的所有位发送完，最后一位被捕获后的一个 SCK 周期之后，SSEL 引脚返回到高电平状态。

但是，在连续帧的发送过程中，在每个数据字传输之间 SSEL 信号必须为高。这是因为当 CPHA 位为逻辑 0 时，从机选择引脚冻结了串行外围寄存器中的数据，不允许其改变。因此，在每次数据传输之间主设备必须拉高从设备的 SSEL 引脚，来允许对串行外设数据的写操作。当连续传输结束，最后一位被捕获后一个 SCK 周期内，SSEL 引脚返回到空闲状态。

11.7.2.5 CPOL = 1, CPHA = 1 的 SPI 帧格式

CPOL = 1, CPHA = 1 的 SPI 帧传输时序如 [Figure 26](#) 所示，包含单帧和连续传输两种方式。



该配置下，在空闲周期内：

- CLK 信号强制为高。
- SSEL 强制为高。
- 发送 MOSI/MISO 引脚处于高阻态。

如果 SPI/SSP 被允许，且发送 FIFO 中有有效数据，则 SSEL 主机信号被拉为低，指示数据发送开始。主机的 MOSI 引脚被允许。再过半个 SCK 周期，主机和从机的有效数据都被允许输出到各自的发送线上。同时，通过下降沿跳变允许 SCK。然后，数据在 SCK 信号的上升沿被捕获并保持到 SCK 信号的下降沿。

A 在单个字的传输过程中，当所有位传输结束，最后一位被捕获的后一个 SCK 周期内，SSEL 返回到高电平状态。对于连续帧的连续传输，SSEL 引脚则仍保持有效的低电平状态，直到最后一个字的最后一位捕获之后，它再返回到空闲状态。总的说来，SSEL 引脚在两个连续的数据字传输之间保持低电平，传输终止的方法与单个字传输相同。

11.7.3 半导体 Microwire 帧格式

[Figure 27](#) 所示为 Microwire 帧格式的单帧传输，[Figure 28](#) 所示则为连续帧传输。

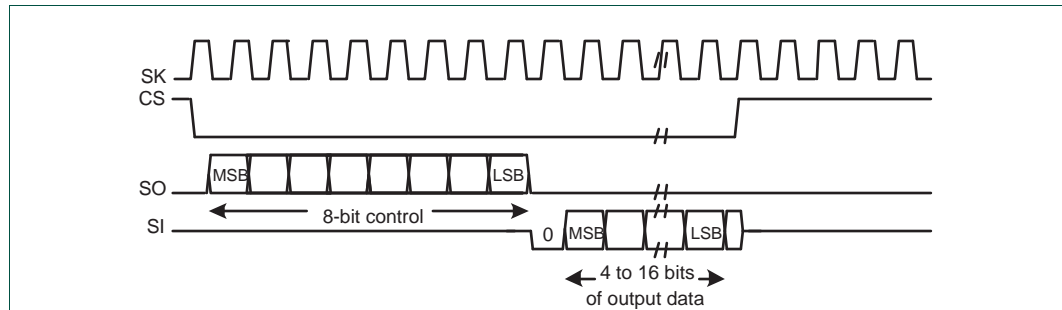


Fig 27. Microwire 帧格式 (单帧传输)

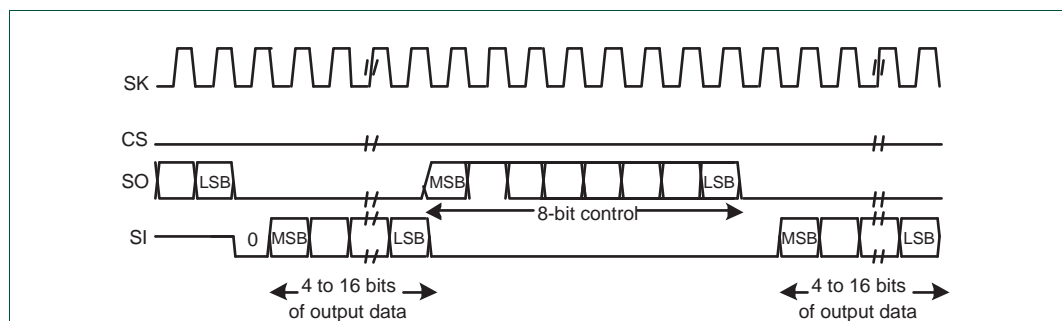


Fig 28. Microwire 帧格式 (连续帧传输)

Microwire 格式与 SPI 格式类似，但它的发送是半双工而非全双工模式，数据从主机传输到从机。每次串行发送以一个 8 位控制字开始，从 SPI/SSP 传输到片外从设备。在发送控制字的过程中，SPI/SSP 不接收数据。控制字发送结束后，片外从设备对其进行译码，在 8 位控制信息的最后一位发送结束后的一个串行时钟之后，才返回主机所需的数据。返回的数据长度为 4 到 16 位，使得总的帧长度在 13 到 25 位之间。

该配置下，在空闲周期内：

- SK 信号强制为低。
- CS 强制为高。
- 发送数据线 SO 可强制为低。

发送过程由写一个控制字节到发送 FIFO 来触发。CS 的下降沿使发送 FIFO 底端的数据传输到串行移位寄存器，8 位控制帧的最高位被移位到 SO 引脚。CS 在帧发送过程中保持低电平。SI 在帧发送过程中保持三态。

片外串行从设备在每个 SK 的上升沿将每个控制位锁存到其串行移位器。当从设备完成最后一位的锁存后，再用一个 SK 周期对控制字节进行译码，然后从设备将数据发回给 SPI/SSP。每一个数据位在 SK 的下降沿驱动到 SI 上。SPI/SSP 在 SK 的上升沿锁存每位数据。在帧的结尾，对单帧传输来说，在最后一位被锁存到接收串行移位器后的一个 SK 周期内，CS 信号被拉高，使数据传输到接收 FIFO。

注意：在最低位被接收移位器锁存后，或当 CS 变为高电平时，片外从设备的接收线在 SK 下降沿时刻为三态。

对于连续传输过程的数据发送，开始和结束的方法都与单帧传输相同。所不同的是，在连续传输过程中，CS 持续有效（保持低电平），数据连续发送。当前数据帧的最低位被接收后，下一帧的控制字节立刻直接发送。在一帧数据的最低位被锁存到 SPI/SSP 后，来自接收移位器的每个接收到的数据在 SK 的下降沿被传送。

11.7.3.1 Microwire 模式中，相对 SK 的 CS 建立和保持时间的要求

在 Microwire 模式中，当 CS 变低后，SPI/SSP 从机在 SK 上升沿对接收数据的首位进行采样。主机可自由驱动 SK，以确保相对 SK 的上升沿 CS 信号有足够的建立和保持时间。

[Figure 29](#) 给出了对 CS 的建立和保持时间的要求。相对 SPI/SSP 从机采样接收数据的首个位的 SK 上升沿，CS 的建立时间至少为 SPI/SSP 操作的 SK 周期的 2 倍。相对于之前的 SK 上升沿，CS 的保持时间至少为一个 SK 周期。

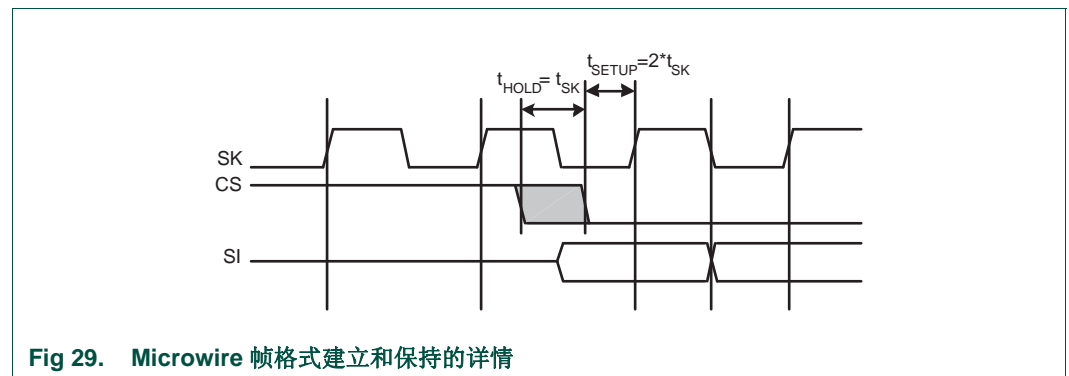


Fig 29. Microwire 帧格式建立和保持的详情

12.1 如何阅读本章

该 I²C 总线模块的介绍适用于所有 LPC111x 和 LPC11Cxx 系列芯片。

12.2 基本配置

I2C 总线接口通过以下寄存器进行配置：

1. 引脚：I2C 引脚功能和 I2C 模式都通过 IOCONFIG 寄存器进行配置。（[Section 7.4](#), [Table 66](#) 和 [Table 67](#)）。
2. 电源和外设时钟：通过 SYSAHBCLKCTRL 寄存器中第 5 位 进行设置 ([Table 21](#))。
3. 复位：再访问 I2C 模块之前，确保 PRESETCTRL 寄存器中 I2C_RST_N 位被设置为 1。这将拉高 I2C 模块的复位信号 ([Table 9](#)) 。

12.3 特征

- 标准的 I²C 兼容总线接口可以配置为主模式、从模式或主 / 从模式。
- 当总线上无串行数据时，多个主设备同时传输则会进行仲裁。
- 允许对时钟编程，以调整 I²C 的传输速率。
- 数据在主设备和从设备之间的传输是双向的。
- 串行时钟同步使得不同位速率的设备可通过一个串行总线进行通信。
- 串行时钟同步作为握手机制来挂起和恢复串行传输。
- 支持增强型快速模式。
- 从设备可多达四个不同的地址。
- 监测模式允许观察 I²C 总线上所有的通讯，忽视从机地址。
- I²C 总线可用于测试和诊断。
- 在 I²C 总线上包含一个带两个引脚的标准 I²C 兼容总线接口。

12.4 应用

该接口用于与外部 I²C 标准器件连接，如串行存储器、LCD、音调发生器和其他微控制器等。

12.5 基本描述

典型的 I²C 总线配置如 [Figure 30](#) 所示。根据方向位的状态（读 / 写），在 I²C 总线上可能有两种类型的数据传输：

- 数据传输从一个主设备的发送器到一个从设备的接收器。主设备发送的第一个字节是从设备的地址。紧接着的是大量的数据字节。从设备在每个字节接收完成后返回一个应答位。

- 数据传输从一个从设备的发送器到一个主设备的接收器。由主设备发送的第一个字节（从设备的地址），然后返回一个应答位，紧接着的数据字节由从设备传输到主设备。除了最后一个字节之外，主设备在其他每个字节接收完成之后返回一个应答位。在最后一个字节接收完成后，返回一个“非应答”。主设备产生的所有的串行时钟脉冲与起始（START）和停止（STOP）条件。一次传输由 STOP 条件或重复 START 条件来结束。由于重复 START 条件将开始下一个串行传输，故 I2C 总线不会被释放。

I2C 接口是面向字节的，有四个工作模式：主发送模式、主接收模式、从发送模式和从接收模式。

I2C 接口符合完全 I2C 规范，可以将 ARM Cortex - M0 处理器断电而不干扰同一个 I2C 总线上的其他设备。

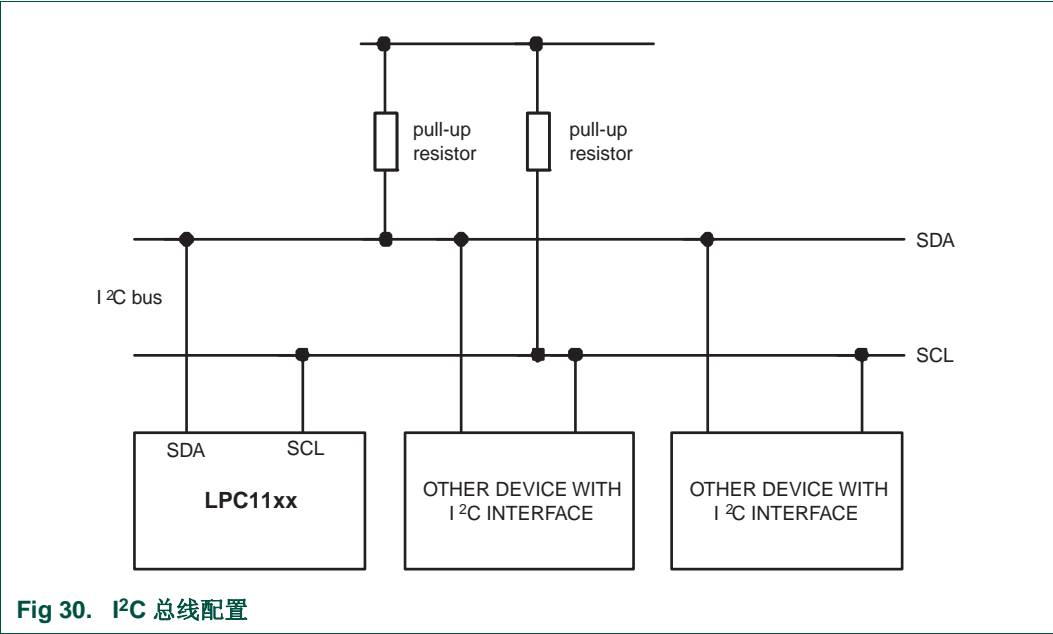


Fig 30. I2C 总线配置

12.5.1 I2C 增强型快速模式（Fast-mode Plus）

增强型快速模式支持 1 兆位 / 秒的传输速率，用于与恩智浦半导体现在提供的 I2C 总线设备进行通信。

12.6 引脚描述

Table 153. I2C- 总线引脚描述

引脚	类型	描述
SDA	输入 / 输出	I2C 串行数据
SCL	输入 / 输出	I2C 串行时钟

I2C 总线引脚必须通过 IOCON_PIO0_4 寄存器 (Table 66) 和 IOCON_PIO0_5 (Table 67) 寄存器来将其配置为标准 / 快速模式或增强型快速模式。在这些模式中，I2C 总线引脚是开漏输出，完全兼容 I2C 总线规范。

12.7 寄存器描述

Table 154. 寄存器概览：I2C (基地址 0x4000 0000)

名字	访问	地址偏移量	描述	复位值 [1]
I2C0CONSET	R/W	0x000	I2C 控制设置寄存器。 当写入 1 到该寄存器的某位，则 I2C 控制寄存器相应的位被设置。写 0 对 I2C 控制寄存器相应的位没有影响。	0x00
I2C0STAT	RO	0x004	I2C 状态寄存器。 在 I2C 工作过程中，这寄存器提供了详细的状态码，以帮助软件确定下一步的动作。	0xF8
I2C0DAT	R/W	0x008	I2C 数据寄存器。 在主或从发送模式下，数据在写入这个寄存器后被传送。在主或从接收模式，已收到的数据可从该寄存器读取	0x00
I2C0ADR0	R/W	0x00C	I2C 从地址寄存器 0。 当 I2C 接口处于从模式时，该寄存器包含 7 位的从地址；在主模式时无用。最后一个有效位决定从设备是否响应广播地址。	0x00
I2C0SCLH	R/W	0x010	SCH 占空比寄存器高半字。 决定 I2C 时钟高电平时间。	0x04
I2C0SCLL	R/W	0x014	SCL 占空比寄存器低半字。 决定 I2C 时钟低电平时间。I2nSCLL 和 I2nSCLH 共同决定由 I2C 主设备产生的时钟频率，某些时候用于从设备。	0x04
I2C0CONCLR	WO	0x018	I2C 控制清零寄存器。 当写入 1 到这个寄存器的某位，相应的 I2C 控制寄存器的位被清零。写一个 0 对相应的 I2C 控制寄存器位没有影响。	NA
I2C0MMCTRL	R/W	0x01C	监测模式控制寄存器。	0x00
I2C0ADR1	R/W	0x020	I2C 从地址寄存器 1。 当 I2C 接口处于从模式时，该寄存器包含 7 位的从地址；在主模式时无用。最后一个有效位决定从设备是否响应广播地址。	0x00
I2C0ADR2	R/W	0x024	I2C 从地址寄存器 2。 当 I2C 接口处于从模式时，该寄存器包含 7 位的从地址；在主模式时无用。最后一个有效位决定从设备是否响应广播地址。	0x00
I2C0ADR3	R/W	0x028	I2C 从地址寄存器 3。 当 I2C 接口处于从模式时，该寄存器包含 7 位的从地址；在主模式时无用。最后一个有效位决定从设备是否响应广播地址。	0x00
I2C0DATA_BUFFER	RO	0x02C	数据缓冲寄存器。 在每次从总线上接收 9 位数据（8 位数据加 ACK 或 NACK）后，自动将移位寄存器 I2DAT 的前八位的内容传送到 DATA_BUFFER 寄存器。	0x00
I2C0MASK0	R/W	0x030	I2C 从地址屏蔽寄存器 0。 这个屏蔽寄存器与 I2ADR0 共同确定一个地址匹配。该寄存器对于地址与广播地址 ('0000000') 比较时没有影响。	0x00
I2C0MASK1	R/W	0x034	I2C 从地址屏蔽寄存器 1。 这个屏蔽寄存器与 I2ADR0 共同确定一个地址匹配。该寄存器对于地址与广播地址 ('0000000') 比较时没有影响。	0x00
I2C0MASK2	R/W	0x038	I2C 从地址屏蔽寄存器 2。 这个屏蔽寄存器与 I2ADR0 共同确定一个地址匹配。该寄存器对于地址与广播地址 ('0000000') 比较时没有影响。	0x00
I2C0MASK3	R/W	0x03C	I2C 从地址屏蔽寄存器 3。 这个屏蔽寄存器与 I2ADR0 共同确定一个地址匹配。该寄存器对于地址与广播地址 ('0000000') 比较时没有影响。	0x00

[1] 复位值只反映被使用位的数据存储情况，不包括保留位的内容。

12.7.1 I2C 控制设置寄存器 (I2C0CONSET - 0x4000 0000)

CONSET 寄存器控制 I2CON 寄存器（用于控制 I2C 接口运作）的位设置。写入 1 到该寄存器的某位，将导致 I2C 控制寄存器的相应位被设置。写 0 则没有影响。

Table 155. I2C 控制设置寄存器 (I2C0CONSET - address 0x4000 0000) 位域描述

位	符号	描述	复位值
1:0	-	保留，用户软件不要对保留位写‘1’。从保留位读取的值没有定义。	NA
2	AA	生效应答标志位。	
3	SI	I2C 中断标志	0
4	STO	STOP 标志位	0
5	STA	START 标志位	0
6	I2EN	I2C 接口允许	0
31:7	-	保留，从保留位读取的值没有定义。	-

I2EN I2C 接口允许。当 I2EN 为 1 时，I2C 接口被允许。写“1”到 CONCLR 寄存器的 I2ENC 位能清除 I2EN 位。当 I2EN 为“0”时，I2C 接口被禁止。

当 I2EN 为“0”时，SDA 和 SCL 的输入信号被忽略，I2C 处在“不可寻址”的从设备状态，且 STO 位被强制为“0”。

I2EN 不应被用来暂时释放 I2C 总线，因为当 I2EN 被复位时 I2C 总线的状态会丢失。应当使用 AA 标志位来代替。

STA 是 START 标志位。设置这个位导致 I2C 接口进入主设备模式，并传输一个 START 条件，如果 I2C 接口已经是主模式则传送一个重复 START 条件。

当 STA 为 1 而且 I2C 接口还不是主模式，它将进入从模式。然后检查总线，如果总线空闲则产生一个 START 条件；如果总线忙，它将等待 STOP 条件（它将使总线空闲）并在内部时钟发生器延时半个时钟周期后产生一个 START 条件。如果 I2C 已经是主模式而且数据已经发出或接收到，它将发送一个重复 START 条件。可以在任何时候设置 STA，包括当 I2C 接口处在一个可寻址的从模式。

写“1”到 CONCLR 寄存器的 STAC 位能清除 STA 位。当 STA 为 0 时，将不会产生 START 条件或重复 START 条件。

在 STA 和 STO 都被设置时，如果接口在主模式，则在 I2C 总线上传输一个 STOP 条件，然后传输一个 START 条件；如果 I2C 接口在从模式，则会产生一个内部 STOP 条件但不会在总线上传输。

STO 是 STOP 标志位。设置这个位将导致 I2C 接口在主模式时传输一个 STOP 条件，如果在从模式时则从错误条件中恢复。在主模式下，当 STO 为 1 时，在 I2C 总线上传输一个 STOP 条件。当总线检测到 STOP 条件时 STO 将被自动清除。

I 在从模式时，设置此位可以从错误条件中恢复。在这种情况下，将没有 STOP 条件传输到总线。硬件行为就像总线接收到了一个 STOP 条件一样，接口将转到“不可寻址”从接收模式。STO 标志位被硬件自动清除。

SI 是 I2C 中断标志位。在 I2C 状态变化时该位被设置。但是在进入 F8 状态时不会设置 SI 位，因为在那种情况下中断服务例程没有什么事情可做。

当 SI 被设置时，SCL 线上串行时钟的低电平时间被延长，串行传输暂停。当 SCL 为高电平时，它将不会影响 SI 标志位。SI 必须由软件通过写 1 到 CONCLR 寄存器的 SIC 位来复位。

AA 是生效应答标志位。当被设置为 1 时，在 SCL 线上的应答时钟脉冲期间，如果有下面的情况下之一，将会返回一个应答信号（SDA 的低电平）：

- 1. 从地址寄存器的地址已经收到。
- 2. 当 I2ADR 中的广播位 (GC 位) 被设置时, 接收到广播地址。
- 3. 在 I2C 为主接收模式时, 接收到一个数据字节。
- 4. 在 I2C 为可寻址的从接受模式时, 接收到一个数据字节。

通过对 CONCLR 寄存器的 AAC 位写 1 可以清除 AA 位。当 AA 为 0 时, 如果有下面的情况下之一, 将会返回一个非应答信号 (SDA 的高电平) :

- 1. 在 I2C 为主接收模式时, 接收到一个数据字节。
- 2. 在 I2C 为可寻址的从接受模式时, 接收到一个数据字节。

12.7.2 I2C 状态寄存器 (I2C0STAT - 0x4000 0004)

每个 I2C 状态寄存器反映了相应 I2C 接口的情况。I2C 状态寄存器只读。

Table 156. I2C 状态 寄存器 (I2C0STAT - 0x4000 0004) 位域描述

位	符号	描述	复位值
2:0	-	这些位没有用到并且始终为 0。	0
7:3	Status	这些位给出了 I2C 接口的实际状态信息。	0x1F
31:8	-	保留。从保留位读出的值未定义。	-

这三个最低有效位总是为 0。作为一个字节, 这个状态寄存器的内容表示一个状态码。一共有 26 种可能的状态码。当状态码为 0xF8 时, 没有任何相关有用信息且不会对 SI 位置位。所有其他 25 个状态码与标准的 I2C 状态相符。当进入这些状态中的任何一个时, 将对 SI 位置位。如需状态码的完整列表, 请参阅 Table 171 到 Table 176。

12.7.3 I2C 数据寄存器 (I2C0DAT - 0x4000 0008)

这个寄存器包含将要发送的数据和刚接收的数据。当 SI 为 1 时, 该寄存器不处于移位过程时, CPU 将读取和写入这个寄存器。只要 SI 位为 1, I2DAT 寄存器中的数据就会保持稳定。I2DAT 寄存器中的数据总是从右移到左: 传送时, MSB (第 7 位) 是被传送的第 1 位; 接收时, 收到数据的第 1 位位于 I2DAT 的 MSB 位。

Table 157. I2C 数据寄存器 (I2C0DAT - 0x4000 0008) 位域描述

位	符号	描述	复位值
7:0	Data	这个寄存器保存要发出的数据或者已接收的数据。	0
31:8	-	保留。从保留位读出的值未定义。	-

12.7.4 I2C 从地址寄存器 0 (I2C0ADR0- 0x4000 000C)

该寄存器可读可写, 且仅用在 I2C 接口设置为从模式时。主模式时, 该寄存器无效。I2ADR 的 LSB 位是广播位。当该位为 1 时, 可识别广播地址 (0x00)。

这些寄存器中, 任何包含 00x 的寄存器将被禁止, 且不会匹配总线上的任何地址。在复位时, 将清除其的禁用状态。参见 Table 164。

Table 158. I2C 从地址寄存器 0 (I2C0ADR0- 0x4000 000C) 位域描述

位	符号	描述	复位值
0	GC	广播允许位	0
7:1	Address	I2C 从模式的设备地址。	0x00
31:8	-	保留。从保留位读出的值未定义。	-

12.7.5 I²C SCL HIGH 和 LOW 占空比寄存器 (I2C0SCLH - 0x4000 0010 和 I2C0SCLL- 0x4000 0014)

Table 159. I²C SCL HIGH 占空比寄存器 (I2C0SCLH - 地址 0x4000 0010) 位域描述

位	符号	描述	复位值
15:0	SCLH	SCL 高电平时间计数值。	0x0004
31:16	-	保留。从保留位读出的值未定义。	-

Table 160. I²C SCL Low 占空比寄存器 (I2C0SCLL - 0x4000 0014) 位域描述

位	符号	描述	复位值
15:0	SCLL	SCL 低电平时间计数值。	0x0004
31:16	-	保留。从保留位读出的值未定义	-

12.7.5.1 选择适当的 I2C 数据率和占空比

必须通过软件设置寄存器 I2SCLH 和 I2SCLL 的值来选择适当的数据率和占空比。I2SCLH 定义了 SCL 高电平持续时间的 I2C_PCLK 周期数，I2SCLL 定义了 SCL 低电平持续时间的 I2C_PCLK 周期数。频率由下式决定 (I2C_PCLK 是外围设备 I2C 的时钟频率)：

(4)

$$I^2C_{bitfrequency} = \frac{I2CPCLK}{SCLH + SCLL}$$

I2SCLL 和 I2SCLH 的值必须确保数据的传输速率在适合的 I2C 数据速率范围内。每个寄存器的值必须大于或等于 4。[Table 161](#) 给了一些基于 I2C_PCLK 频率和 I2SCLL 和 I2SCLH 的值来确定 I2C- 总线速率的例子。

Table 161. 根据 I2C 时钟频率选择 I2SCLL + I2SCLH 值

I ² C 模式	I ² C 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
增强型快速模式	1 MHz	-	8	10	12	16	20	30	40	50

I2SCLL 和 I2SCLH 没有必要是相同的。可以通过软件设置这两个寄存器的值来设置不同的 SCL 占空比。I2C 总线规范定义中，在快速模式和增强型快速模式时 SCL 低电平时间和高电平时间就可不同。

12.7.6 I2C 控制清除寄存器 (I2C0CONCLR - 0x4000 0018)

CONCLR 寄存器控制 I2CON 寄存器（用于控制 I2C 接口）中位的清除。写入 1 到这个寄存器的某位，将导致 I2C 控制寄存器相应的位被清除。写一个 0，则没有影响。

Table 162. I2C 控制清除寄存器 (I2C0CONCLR - 0x4000 0018) 位域描述

位	符号	描述	复位值
1:0	-	保留，用户软件不要对保留位写‘1’。从保留位读取的值未定义。	NA
2	AAC	有效应答清除位	
3	SIC	I2C 中断清除位。	0
4	-	保留，用户软件不要对保留位写‘1’。从保留位读取的值未定义。	NA
5	STAC	START 标志位清除位。	0
6	I2ENC	I2C 接口禁用位。	0
7	-	保留，用户软件不要对保留位写‘1’。从保留位读取的值未定义。	NA
31:8	-	保留。从保留位读取的值未定义。	-

AAC 是有效应答清除位。在 CONSET 寄存器中写 1 将清除 AA 位，写 0 无效。

SIC 是 I2C 中断清除位。在 CONSET 寄存器中写 1 将清除 SI 位，写 0 无效。

STAC 是 START 标志清除位。在 CONSET 寄存器中写 1 将清除 STA 位，写 0 无效。

I2ENC 是 I2C 接口禁用位。在 CONSET 寄存器中写 1 将清除 I2EN 位，写 0 无效。

12.7.7 I2C 监测模式控制寄存器 (I2C0MMCTRL - 0x4000 001C)

该寄存器控制监测模式，监测模式允许 I2C 模块在其并不真正参与 I2C 总线通讯的情况下，仍能检测 I2C 总线上的通讯。

Table 163. I2C 监测模式控制寄存器 (I2CMMCTRL0 - 0x4000 001C) 位域描述

位	符号	值	描述	复位值
0	MM_ENA		允许检测描述	0
		0	禁用监测模式。	
		1	I2C 模块将进入 监测模式。在该模式中 SDA 的输出将被强制为高电平。这将阻止 I2C 模块输出任何种类的数据 (包括 ACK) 到 I2C 数据总线上。 根据 ENA_SCL 位的状态，输出也可能被强制为高电平，以阻止模块对 I2C 时钟线的控制。	

Table 163. I²C 监测模式控制寄存器 (I2CMMCTRL0 - 0x4000 001C) 位域描述

位	符号	值	描述	复位值
1	ENA_SCL		SCL 输出允许	0
		0	当该位被清为 ‘0’ 时，在模块为监测模式时 SCL 的输出将被强制为高电平。正如上面描述的，阻止模块对 I2C 时钟线的任何控制。	
		1	当此位被设置，I2C 模块对时钟线可行使与正常工作时相同的控制。这意味着，作为一个从外设，I2C 模块可以“扩展”时钟线（保持低电平），直到它有时间响应一个 I2C 中断。 [1]	
2	MATCH_ALL		选择中断寄存器匹配。	0
		0	当此位被清零时，只有当一个地址与上述描述的 4 个地址寄存器之一匹配时，才会产生一个中断。也就是说，只要识别出的地址是相关的，模块将作为一个正常的从模式响应。	
		1	当该位被置为 ‘1’ 并且 I2C 处于监测模式时，接收到任何地址都会产生一个中断。这将使模块可监测总线上的所有通讯。	
31:3	-	-	保留。从保留位读取的值未定义。	

[1] 当 ENA_SCL 被清除并且 I2C 不再有阻塞总线的能力时，中断响应时间变得很重要。为了在这这些情况下，让模块有时间来响应一个 I2C 中断。DATA_BUFFER 寄存器 ([Section 12.7.9](#)) 被用来保存接收的数据，保存时间为 9 位字的传输时间。

备注：如果 MM_ENA 为 ‘0’ (即不是监测模式)，ENA_SCL 和 MATCH_ALL 位是无效的。

12.7.7.1 监测模式中的中断

在监测模式时，模块所有的中断将正常产生。这意味着检测到一个地址匹配 (MATCH_ALL 为 1 时接收到任何地址将会产生中断，否则就是接收到与四个地址寄存器其中任一相匹配的地址) 将会产生第一个中断。

在一个地址匹配检测后，中断将会在从设备写传输的每个数据字节接收之后产生，或在从设备读传输的每个字节发送成功之后产生。在第二种情况下，数据寄存器实际包含了在总线上被其他从设备传输的数据，这些从设备都被主设备寻址过。

跟着所有这些中断，处理器可以读取数据寄存器，看看总线上实际传输了什么。

12.7.7.2 监测模式下的仲裁丢失

在监测模式下，I2C 将不能响应总线上主机发出的信息请求或者发出的一个 ACK。而总线上的一些其他的从设备将会响应。只要我们的模块是相关的，这将极有可能导致仲裁状态丢失。

软件应该知道一个事实，即模块在监测模式下不会对被检测到的任何仲裁状态丢失作出响应。此外，可以在模块中硬件实现，阻止一些或所有的仲裁状态（如果这些状态阻止了一个期望发生的中断或产生一个不想要的中断）丢失。然而，是否添加这种硬件实现仍有待确定。

12.7.8 I²C 从地址寄存器 (I2C0ADR[1, 2, 3]- 0x4000 00[20, 24, 28])

该寄存器可读可写，且仅在 I2C 接口被设置为从模式时使用。在主模式时，该寄存器无效。I2ADR 的 LSB 位是广播位。当该位为 1 时，广播地址 (0x00) 被识别。

这些寄存器中，任何包含位 00x 的寄存器将被禁止，且不会与总线上的任何地址匹配。在复位时，四个从地址寄存器都将被清零到这个禁用状态。(也可以参见 [Table 158](#)).

Table 164. I²C 从地址寄存器 (I2C0ADR[1, 2, 3]- 0x4000 00[20, 24, 28]) 位域描述

位	符号	描述	复位值
0	GC	广播允许位	0
7:1	Address	从模式时 I2C 的设备地址。	0x00
31:8	-	保留。从保留位读取的值未定义。	0

12.7.9 I²C 数据缓冲寄存器 (I2C0DATA_BUFFER - 0x4000 002C)

在监测模式下，如果 ENA_SCL 位为 0，I2C 模块将没有扩展时钟 (停止总线) 的能力。这意味着该处理器将必须在一个有限的时间之内去读取总线上收到的数据的内容。如果处理器和平常一样读取 I2DAT 移位寄存器，那么在接收数据被新的数据覆盖之前，它只能有一个位的时间来响应中断。

为了让处理器有更多的响应时间，添加一个新的 8 位只读寄存器 DATA_BUFFER。在每次从总线上接收 9 位数据（8 位加 ACK 或 NACK）后，将 I2DAT 的前 8 位的内容传送到 DATA_BUFFER 寄存器中。这意味着处理器在数据被覆盖之前，将有 9 位传输时间响应中断并读取数据。

像平时一样，处理器将仍然有能力直接读取 I2DAT，且 I2DAT 的行为将不会有任何方式改变。

虽然 DATA_BUFFER 寄存器主要在监测模式和 ENA_SCL 位 = '0' 时使用，但是它可在任何运行模式下的任何时间都可以被读取。

Table 165. I²C 数据缓冲寄存器 (I2CDATA_BUFFER - 0x4000 002C) 位域描述

位	符号	描述	复位值
7:0	Data	该寄存器保存 I2DAT 移位寄存器前 8 位（8 MSBs）的值	0
31:8	-	保留。从保留位读取的值未定义。	0

12.7.10 I²C 屏蔽寄存器 (I2C0MASK[0, 1, 2, 3] - 0x4000 00[30, 34, 38, 3C])

四个屏蔽寄存器每个包含七个有效位 (7:1)。若些寄存器中任何一位设置为 ‘1’ 时，当接收到的地址和 I2ADDRn 寄存器相比较时，将会导致接收到的地址的相应位和那个屏蔽寄存器做自动比较。换句话说， I2ADDRn 寄存器中被屏蔽的位将不会参与决定地址的匹配。

在复位时，所有的屏蔽寄存器都清 ‘0’。

屏蔽寄存器在比较广播地址 (“0000000”) 时没有用。

屏蔽寄存器的位 (31:8) 位 (0) 没有使用，且不能写入。读这些位总是返回 0。

当地址匹配中断发生时，处理器将读取数据寄存器（I2DAT），以确定实际上是那个地址导致了地址匹配。

Table 166. I²C 屏蔽寄存器 (I2C0MASK[0, 1, 2, 3] - 0x4000 00[30, 34, 38, 3C]) 位域描述

位	符号	描述	复位值
0	-	保留，用户软件不要对保留位写 ‘1’。读取这些位总是返回 ‘0’。	0
7:1	MASK	屏蔽位。	0x00
31:8	-	保留。从保留位读取的值未定义。	0

12.8 I²C 操作模式

在一个给定的应用程序中， I²C 模块可作为一个主设备，一个从设备，或两者兼有。在从模式时， I²C 硬件将监听总线，是否有四个从设备地址之一和广播地址，如果找到其中的任何一个就会发出中断请求。如果处理器希望成为总线主机，在进入主模式之前必须等待直到总线空闲，这样可以保证一个可能正在进行的从操作不会被中断。如果在主模式时总线仲裁丢失， I²C 模块会立即切换到从模式，并可以在同一个串行传输中检测到其自己的从设备地址。

12.8.1 主发送器模式

在这种模式下，数据从主设备传输到从设备。在可以进入主发送模式之前，必须如 [Table 167](#). 所示初始化 CONSET 寄存器。I2EN 必须设置为 1，以允许 I2C 功能。如果 AA 位为 0，当其他设备是总线主机时， I²C 接口将不识别任何地址，所以它不能进入从模式。STA、STO 和 SI 位必须为 0，其中 SI 位可以通过写 ‘1’ 到 CONCLR 寄存器的 SIC 位来清除；STA 位必须在写从地址之后清除。

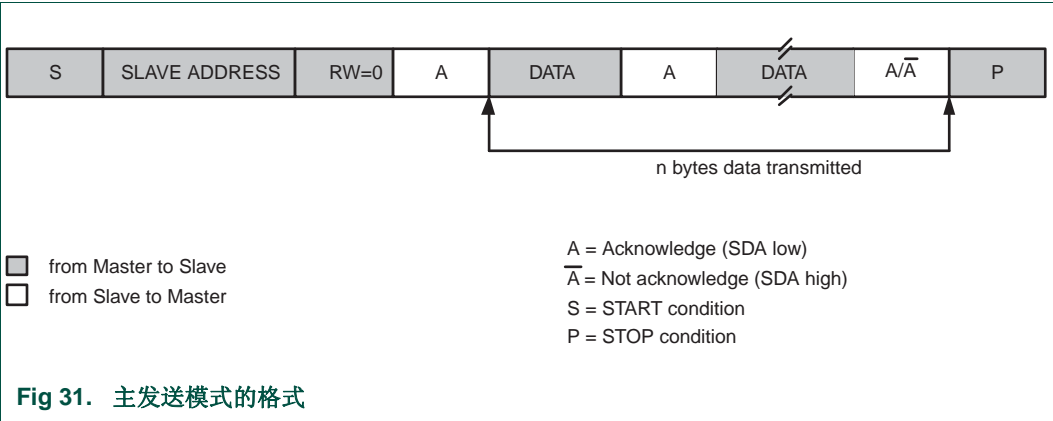
Table 167. 用于配置主模式的 I2C0CONSET 和 I2C1CONSET

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

传送的第一个字节包含接收设备的从地址（7 位）和数据方向位。在此模式下，数据方向位（读 / 写）应该是 0，这意味着写入。传送的第一个字节包含接收设备的从地址和数据写入位。每次传输 8 位数据，在每个字节传输完成后会接收到一个应答位。START 条件和 STOP 条件的输出表明一个串行传输的开始和结束。

当软件置位 STA 位时，I2C 接口将进入主发送模式。总线一旦空闲，I2C 逻辑将发送 START 条件；START 条件发送后，SI 位将被置 1；STAT 寄存器中的状态码是 0x08，这个状态码将会作为一个状态服务程序的向量；状态服务程序将会加载从设备地址和写 I2DAT 寄存器，并清除 SI 位。SI 位可以通过写 '1' 到 CONCLR 寄存器的 SIC 位清除。

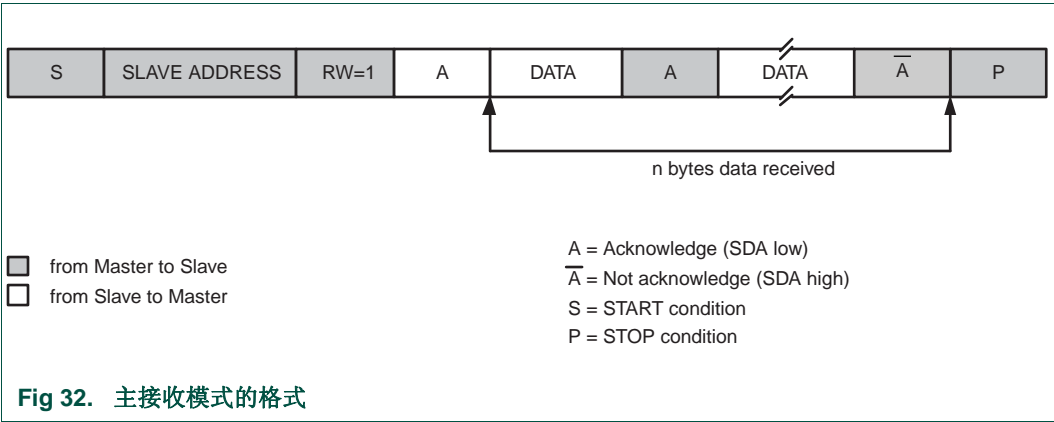
当从地址和读 / 写位已经发送并且也收到了应答位，SI 位将会再次被置 1，对于主模式现在可能的状态码是 0x18、0x20 或 0x38，而对于从模式允许（通过设置 AA 为 1）时则可能是 0x68、0x78 或 0xB0。这些状态码中的每一个状态都会有一个相应的动作，具体动作如 Table 171 到 Table 176. 所示。



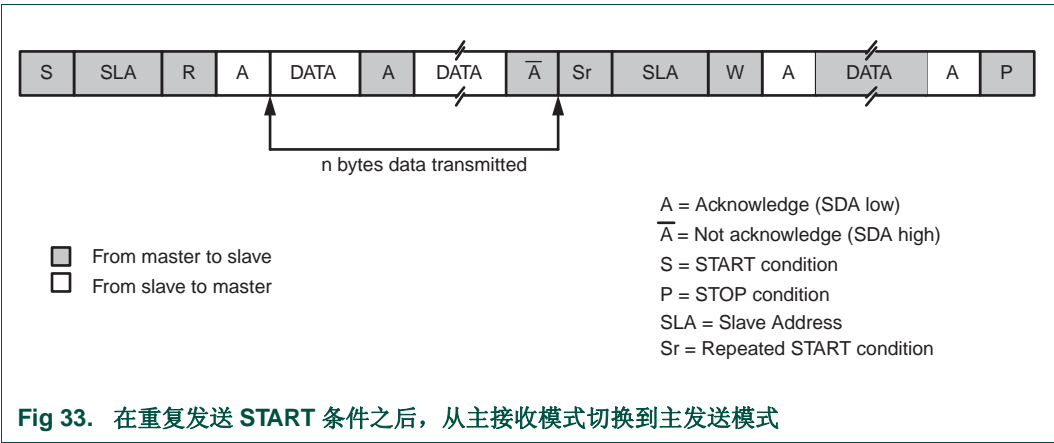
12.8.2 主接收模式

在主接收模式，从一个从发送器接收数据。传输以与主发送模式同样的方式启动。当 START 条件已经传送，中断服务程序必须加载从地址和数据方向位到 I2C 数据寄存器（I2DAT）中，然后清除的 SI 位。在这种情况下，数据方向位（读 / 写）应为 1，表示读。

当从地址和数据方向位已发送，且已收到一个应答位时，SI 位也被设置为 1，状态寄存器将显示状态代码。对于主模式，可能的状态代码是 0x40、0x48 或 0x38；对于从模式，可能的状态代码是 0x68、0x78 或 0xB0。有关详情，请参阅 Table 172。



在一个重复的起始条件后，I2C 会转到主发送模式。



12.8.3 从接收模式

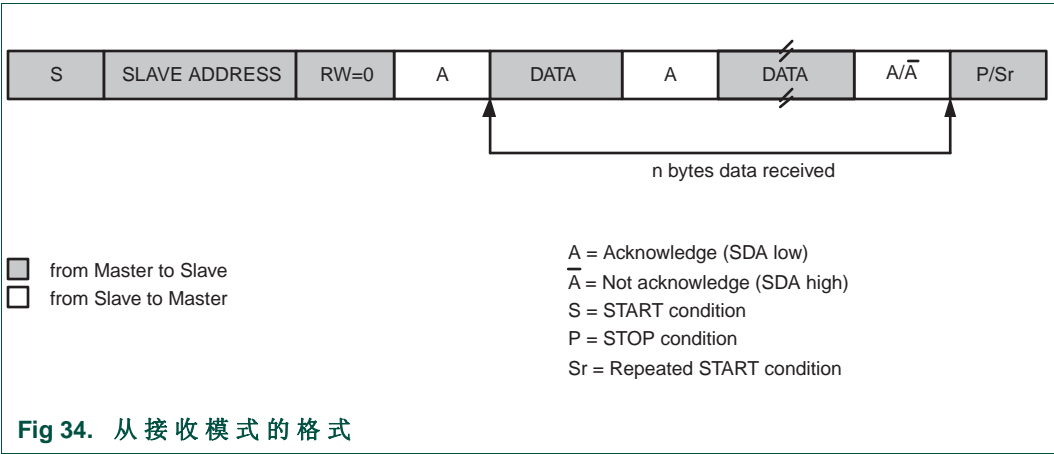
在从接收模式中，从一个主发送器接收数据字节。为了初始化从接收器模式，需要对任意一个地址寄存器（I2ADR0 - 3）写从地址，还要按 [Table 168](#) 所示去写 I2C 控制设置寄存器（CONSET）。

Table 168. 用于配置从模式的 I2C0CONSET 和 I2C1CONSET

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

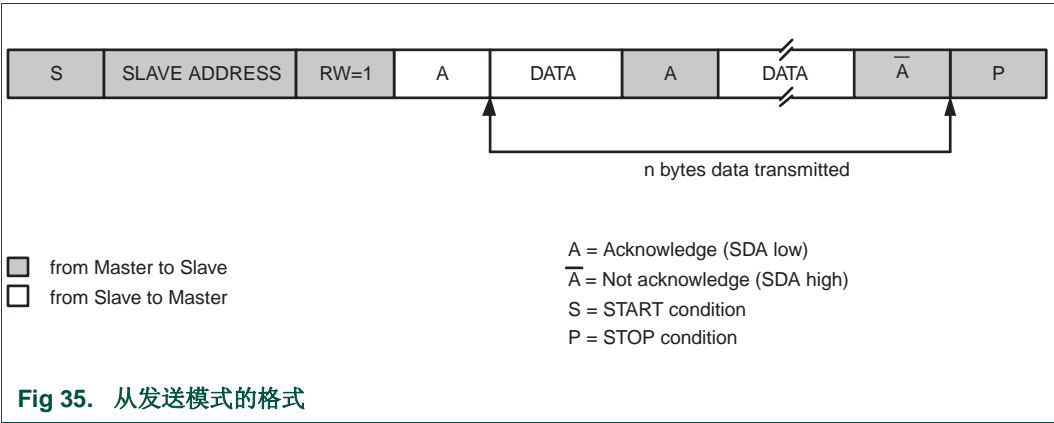
I2EN 位必须设置为 1，以允许 I2C 的功能。AA 位必须设置为 1，以识别自己从地址或广播地址。STA、STO 和 SI 位均要设置为 0。

在 I2ADR 和 CONSET 被初始化之后，I2C 接口将等待接收其自身地址或广播地址，地址后面带有数据方向位。如果方向位 0（写），它进入从接收器模式；如果方向位为 1（读），则进入从发射模式。在从地址和方向位均已收到之后，SI 位将设置为 1，可以从状态寄存器（STAT）中读取有效状态代码。状态码和相应动作可参考 [Table 175](#)。



12.8.4 从发送模式

第一个字节的接收和处理与从接收模式一样，但该模式下方向位为 1，表示读操作。串行数据通过 SDA 传输，串行时钟由 SCL 输入。START 条件和 STOP 条件用于识别一个串行传输的开始和结束。在一个给定的应用程序中，I2C 可用于主模式或从模式。在从模式中，I2C 硬件寻找其自己的从地址和广播地址。如果检测到这些地址之一，将会产生一个中断请求。如果处理器希望成为总线主机，那么在进入主模式之前必须等待直到总线空闲，这样可能正在进行的从操作不会被中断。如果在主模式下总线仲裁被丢失，I2C 接口会立即切换到从模式，并可以在同一个串行传输中检测到其自己的从设备地址。



12.9 I2C 的实现与操作

Figure 36 所示为如何片上实现 I2C 总线接口的，其中每个模块的功能在后面介绍。

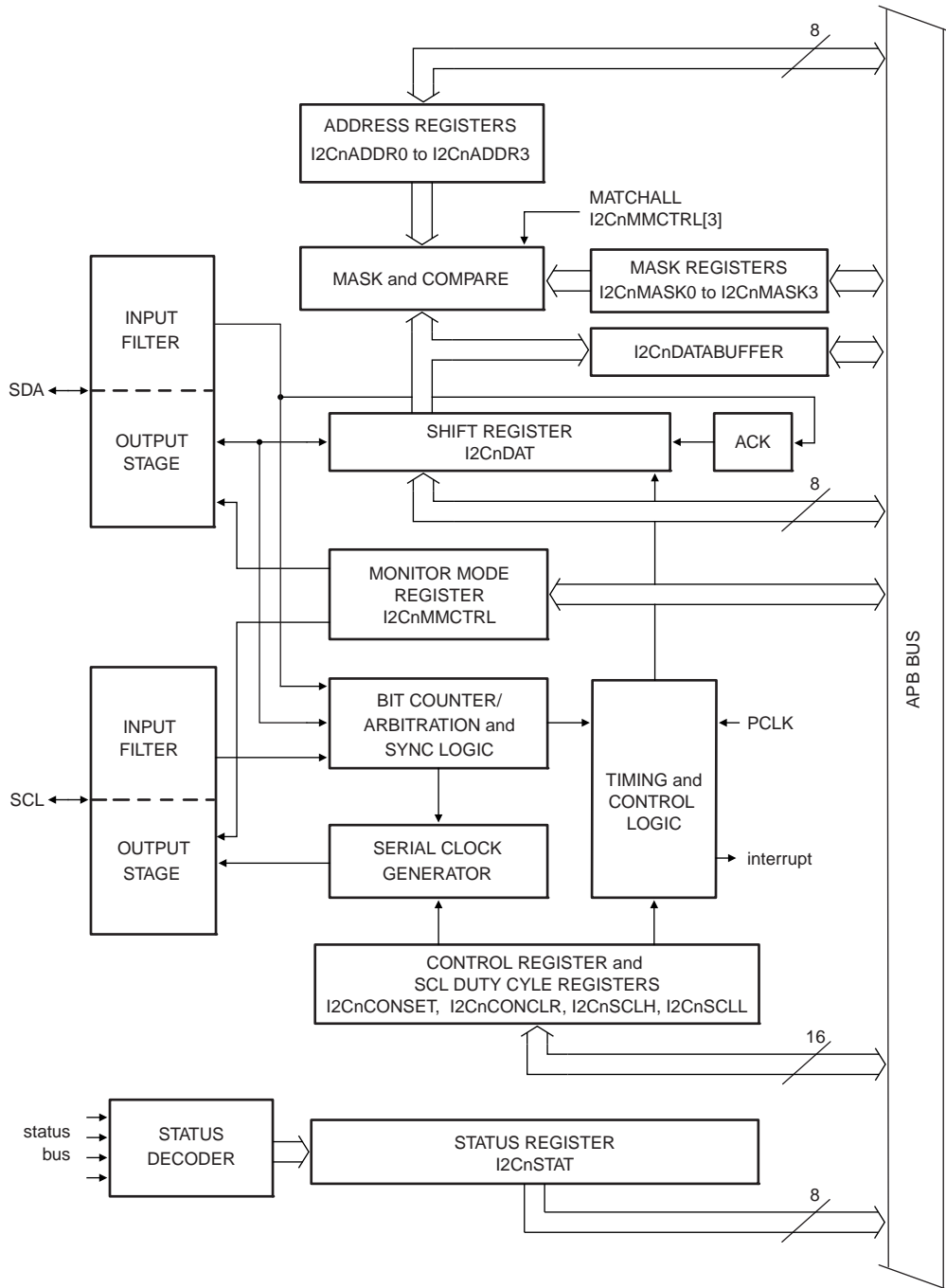


Fig 36. I2C serial interface block diagram

12.9.1 输入滤波器（Input filter）和输出段（output stage）

输入信号与内部时钟同步，少于 3 时钟周期的毛刺将被过滤掉。

为了符合 I2C 规范，I2C 的输出设计为一个特殊的引脚。

12.9.2 地址寄存器, ADDR0 to ADDR3

这些寄存器可装载 7 位从地址（最高 7 位），在 I2C 模块被编程配置为从发送或从接收模式时，会对这些地址做出响应。最低位 (GC) 被用来允许广播地址 (0x00) 被识别。允许多个从地址时，如果已收到自身从地址，可以从 I2DAT 寄存器中读出实际接收到的地址。

12.9.3 地址屏蔽寄存器, I2MASK0 - I2MASK3

四个屏蔽寄存器每个包含七个有效位 (7:1)。若些寄存器中任何一位设置为 '1' 时，当接收到的地址和 ADDRn 寄存器相比较时，将会导致接收到的地址的相应位和那个屏蔽寄存器做自动比较。换句话说，在 ADDRn 寄存器中被屏蔽的位将不会参与决定地址的匹配。

当地址匹配中断发生时，处理器将读取数据寄存器 (DAT)，以确定实际上是哪个地址导致了地址匹配。

12.9.4 比较器

比较器会将收到的 7 位从设备地址与自身的从设备地址 (ADR 中最高 7 位) 比较。它也会将首先收到的 8 位与广播地址 (0x00) 进行比较。一旦发现相等的情况，就会设置一个合适的状态码并产生一个中断请求。

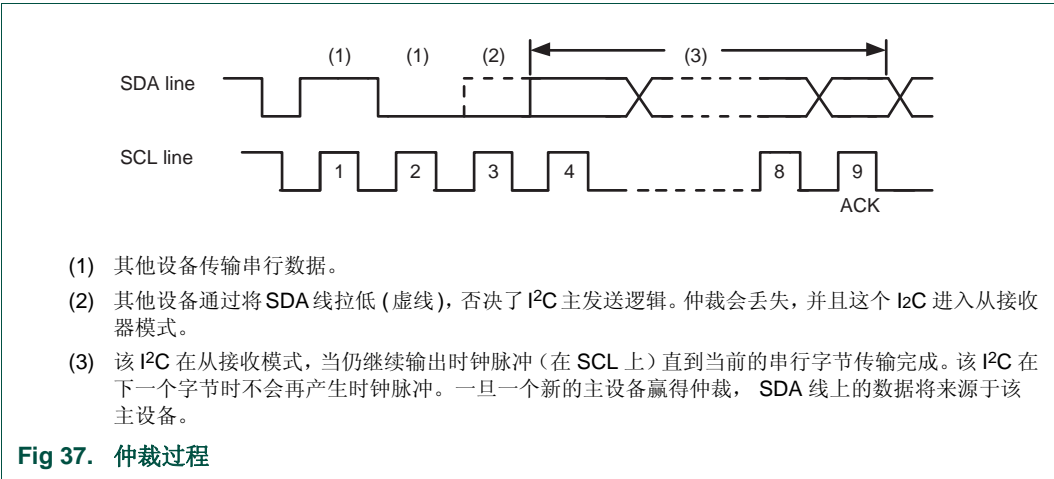
12.9.5 移位寄存器, DAT

这个 8 位寄存器包含了一个字节的串行数据，该数据是将要发送的或是刚接收的。DAT 的数据总是从右移到左，第一个被发出的位是最高位（第 7 位），在一个字节已经接收完成后，最先接收到的数据放在 DAT 的最高位。当数据移出的同时，总线上的数据被移入。DAT 总是保存总线上最后一个字节的数据。因此，在仲裁丢失的情况下，主发送器会将 DAT 中正确的数据传输到从接收器中。

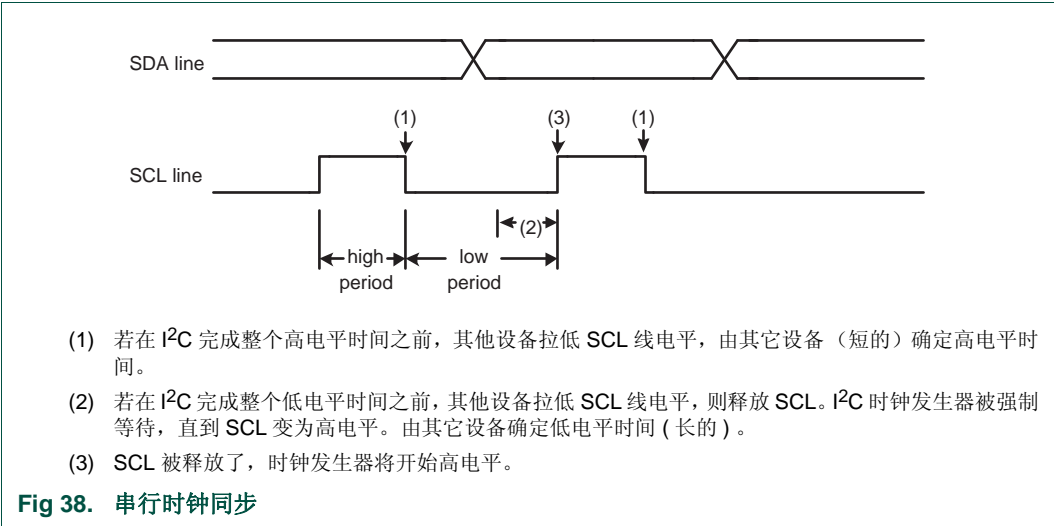
12.9.6 仲裁和同步逻辑

在主控发送模式下，仲裁逻辑检查每个传输逻辑 1，也就是实际在 I2C 总线出现的逻辑 1。如果总线上的另一个设备否决了逻辑 1，把 SDA 拉低，仲裁丢失，而且 I2C 模块立即从主发送变为从接收。I2C 模块将继续输出时钟脉冲（在 SCL 上）直到当前的串行字节传输完成。

在主接收模式下，仲裁也可能丢失。在此模式下，只有当 I2C 模块返回一个“非应答”（逻辑 1）到总线上时才发生仲裁丢失。当总线上其他设备拉低这个信号时仲裁丢失。因为这只可能发生在在一个串行字节结束时，故 I2C 模块不再产生时钟脉冲。[Figure 37](#) 所示为仲裁过程。



同步逻辑将串行时钟发生器与其他设备 SCL 线的时钟脉冲同步。如果两个或两个以上的主设备产生时钟脉冲, 高电平的持续时间取决于设备生成最短高电平, 低电平持续时间则取决于设备生成最长的低电平。Figure 38 为同步过程。



从设备可以拉伸低电平时间以将总线主机速度放慢。另外, 为了握手, 低电平时间也可能被拉伸。这可以在每个位或一个完整的字节传输之后发生。在完成一个字节发送或完成一个字节接收且传输一个应答位之后, I²C 模块将拉伸 SCL 的低电平时间。串行中断标志 (SI) 被设置为 1, 低电平时间将拉伸, 直到串行中断标志被清除。

12.9.7 串行时钟发生器

当 I²C 模块在主发送或主接收模式时, 这种可编程时钟脉冲发生器提供 SCL 时钟脉冲。当 I²C 模块在从模式时, 其将关闭。通过 I²C 时钟控制寄存器, 设置 I²C 输出时钟频率和占空比。详见 I2CSCLL 和 I2CSC LH 寄存器描述。输出时钟脉冲按编程设置产生占空比, 除非该总线与其他 SCL 时钟源同步 (如上所述)。

12.9.8 定时与控制

定时和控制逻辑产生用于串行字节处理的定时和控制信号。该逻辑模块为 DAT 提供移位脉冲，允许比较器，产生和检测 START 和 STOP 条件，接收和发送应答位，控制主从模式，包含了中断请求逻辑，还监测 I2C 总线状态。

12.9.9 控制寄存器，CONSET 和 CONCLR

I2C 控制寄存器位用于控制 I2C 模块的以下功能：启动和重新启动串行传输、终止串行传输、比特率、地址识别和确认。

I2C 控制寄存器的内容可通过 CONSET 读出。对 CONSET 寄存器相应位写 1，将会设置 I2C 寄存器的相应位。对 CONCLR 寄存器相应位写 1，则将会清除 I2C 寄存器的相应位。

12.9.10 状态解码器和状态寄存器

状态解码器获取所有内部状态位，并将它们压缩成了一个 5 位的代码。每个 I2C 总线状态都有唯一的代码。5 位的编码，可用于生成各种服务程序快速处理的向量地址。每个服务程序处理一个特定总线状态。如果 I2C 使用了所有的四种模式，共有 26 种可能的总线状态。在串行中断标志被硬件设置时，5 位的状态代码锁存到状态寄存器的最高 5 位中，并保持稳定直到中断标志被软件清除。这个状态寄存器的最低 3 位始终为零。如果状态码用作服务程序的向量地址，则程序被 8 个地址位置代替。对大多数服务程序来说，8 个字节的代码是足够的（见本节软件举例）。

12.10 I2C 操作模式详细介绍

四个操作模式：

- 主发送
- 主接收
- 从接收
- 从发送

每种模式的数据传输如 [Figure 39](#), [Figure 40](#), [Figure 41](#), [Figure 42](#), 和 [Figure 43](#). [Table 169](#) 列出了这些图在描述 I2C 操作模式时所用到的缩写。

Table 169. 描述 I2C 操作时用到的缩写

缩写	说明
S	START 条件
SLA	7 位地址
R	读位 (SDA 上高电平)
W	写位 (SDA 上低电平)
A	应答位 (SDA 上低电平)
\overline{A}	非应答位 (SDA 上高电平)
Data	8 位数据字节
P	STOP 条件

在 [Figure 39](#) 到 [Figure 43](#) 中，圆圈用于指示当串行中断标志被设置。圆圈中的数字为保存在 STAT 寄存器中的状态代码。在这些点上，服务程序必须执行以继续或完成串行传输。这些服务例程并不紧急，因为直至串行中断标志由软件清除之前，串行传输是被挂起的。

当进入一个串行中断程序时，STAT 中的状态代码用于跳转到相应的服务程序中。对于每一个状态代码，所需软件动作和下一个串行传输传输的详情见 [Table 171](#) 到 [Table 177](#)。

12.10.1 主发送模式

在主机发送模式，大量的数据字节数被传送到一个从接收器 (见 [Figure 39](#)). 在进入主发送器模式之前，I2CON 必须做如下初始化：

Table 170. I2CONSET 用于初始化主发送器模式

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

必须通过 SCLL 和 SCLH 寄存器配置 I2C 速率。I2EN 必须设置为逻辑 1，以启用 I2C 模块。如果 AA 位被复位，I2C 模块将不响应其自身从地址或广播地址，即使其他设备成为总线主机。换句话说，当 AA 复位时，I2C 接口不能进入从模式，STA、STO 和 SI 都必须都复位。

现在可通过设置 **STA** 位进入主发送模式。现在，**I2C** 逻辑将会测试 **I2C** 总线，一旦总线空闲将生成一个 **START** 条件。当一个 **START** 条件被传输，串行中断标志位 (**SI**) 被设置，且状态寄存器 (**STAT**) 的状态代码将是 **0x08**。中断服务程序使用此状态代码，以进入一个正确的状态服务程序，该状态服务程序将加载从机地址和数据方向位 (**SLA+W**) 到 **DAT**。在串行传输继续之前，**I2CON** 中的 **SI** 位必须被复位。

当从地址和方向位已传输，且已收到一个应答信号，串行中断标志 (位 **SI**) 将再次被设置，**STAT** 中的状态码有多种可能。主模式下可能有 **0x18**、**0x20** 或 **0x38**；如果从模式被允许 (**AA** = 逻辑 1) 则可能有 **0x68**、**0x78** 或 **0xB0**。这些状态代码所对应的动作详见 [Table 171](#)。经过在重复 **START** 条件之后 (状态 **0x10**)，**I2C** 模块将通过加载 **SLA+R** 到 **DAT** 以转换到主接收模式。

Table 171. 主发送模式

状态码 (I2CSTAT)	I2C 和硬件总线状态	应用软件响应 To/From DAT	To CON				I2C 硬件的下一个动作
			STA	STO	SI	AA	
0x08	已发送 START 条件	加载 SLA+W; 清除 STA	X	0	0	X	将传输 SLA+W, 将会接收到 ACK 位。
0x10	已发重复 START 条件	加载 SLA+W 或	X	0	0	X	同上
		加载 SLA+R; 清除 STA	X	0	0	X	将传输 SLA+W; I2C 模块转换到 MST/REC 模式。
0x18	已传输 SLA+W; 已收到了 ACK。	加载数据字节或	0	0	0	X	将传输 SLA+W; I2C 模块转换到 MST/REC 模式。
		无 DAT 动作或	1	0	0	X	传输重复 START 条件。
		无 DAT 动作或	0	1	0	X	传输 STOP 条件; STO 标志将复位。
		无 DAT 动作	1	1	0	X	STOP 条件之后传输一个 START 条件; STO 标志将被复位。
0x20	SLA+W 已被传输; 非应答位已接收到	加载数据字节或	0	0	0	X	将传输数据字节; 将会接收到 ACK 位。
		无 DAT 动作或	1	0	0	X	传输重复 START 条件。
		无 DAT 动作或	0	1	0	X	将传输 STOP 条件; STO 标志将复位。
		无 DAT 动作	1	1	0	X	将传输 STOP 条件之后传输一个 START 条件; STO 标志将被复位。
0x28	DAT 中数据字节已传输, ACK 已被接收	加载数据字节或	0	0	0	X	将传输数据字节; 将会接收到 ACK 位。
		无 DAT 动作或	1	0	0	X	传输重复 START 条件。
		无 DAT 动作或	0	1	0	X	将传输 STOP 条件; STO 标志将复位。
		无 DAT 动作	1	1	0	X	将传输 STOP 条件之后传输一个 START 条件; STO 标志将被复位。
0x30	DAT 中数据字节已被传输; 非应答位已被接收	加载数据字节或	0	0	0	X	将传输数据字节; 将会接收到 ACK 位。
		无 DAT 动作或	1	0	0	X	传输重复 START 条件。
		无 DAT 动作或	0	1	0	X	将传输 STOP 条件; STO 标志将复位。
		无 DAT 动作	1	1	0	X	将传输 STOP 条件之后传输一个 START 条件; STO 标志将被复位。
0x38	在数据字节或 SLA+R/W 传输阶段, 仲裁丢失了。	无 DAT 动作或	0	0	0	X	I2C 总线将被释放; 将进入不可寻址从模式。
		无 DAT 动作	1	0	0	X	当总线空闲时, 将发生 START 条件。

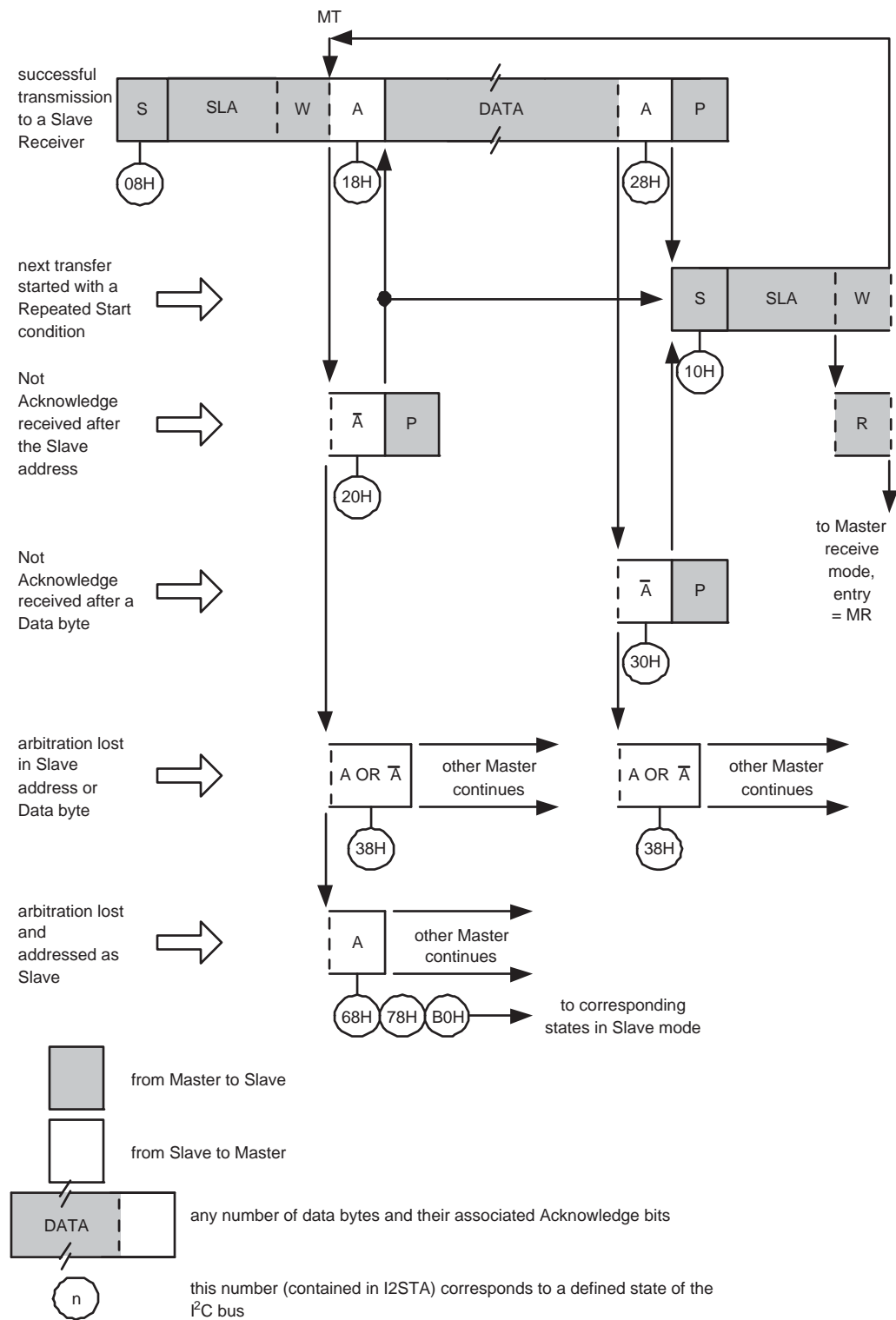


Fig 39. 主传输模式的格式和状态

12.10.2 主接收模式

在主接收模式中，从从发送器接收大量的数据字节 (见 [Figure 40](#)). 传输被初始化为发送模式。当 START 条件已传输时，中断服务程序必须加载 7 位从地址和数据方向位 (SLA+R) 到 DAT。在串行传输继续之前， I2CON 中的 SI 位必须被清零。

当已完成从地址和方向位传输，且已收到一个应答信号时，串行中断标志 (位 SI) 将被再次设置，且 I2STAT 中的状态码有多种可能。主模式可能有 0x40、0x20 或 0x38 ； 如果从模式被允许 (AA = 逻辑 1) 则可能有 0x68、0x78 或 0xB0 。这些状态代码所对应的动作详见 [Table 172](#). 经过重复 START 条件之后 (状态 0x10) 。I2C 模块将通过加载 SLA+W 到 DAT 以转换到主发送器模式。

Table 172. 主接收模式

状态码 (STAT)	I2C 总线及硬件状态	应用软件响应 To/From DAT	To CON				I2C 硬件的下一个动作
			STA	STO	SI	AA	
0x08	已完成 START 条件传输。	加载 SLA+R	X	0	0	X	SLA+R 将被传送；将接收到 ACK 位。
0x10	已完成一个重复 START 条件传输。	加载 SLA+R 或	X	0	0	X	同上
		加载 SLA+W	X	0	0	X	将发送 SLA+W，I2C 模块将切换到 MST/TRX 模式。
0x38	在 NOT ACK 位传输阶段，仲裁丢失	无 DAT 动作 或	0	0	0	X	I2C 将会释放；I2C 模块将进入从模式。
		无 DAT 动作	1	0	0	X	当总线空闲时，将发送 START 条件。
0x40	SLA+R 已传输；已收到 ACK	无 DAT 动作 或	0	0	0	0	将收到数据字节；将返回 NOT ACK 位。
		无 DAT 动作	0	0	0	1	将收到数据字节；将返回 ACK 位。
0x48	SLA+R 已传输；已收到 NOT ACK	无 DAT 动作 或	1	0	0	X	重复 START 条件将会发送。
		无 DAT 动作 或	0	1	0	X	将发送 STOP 条件，STO 标志将复位。
		无 DAT 动作	1	1	0	X	将发送 STOP 条件，之后再传输一个 START 条件；STO 标志将被复位。
0x50	已收到数据字节；已返回 ACK。	读数据字节 或	0	0	0	0	将收到数据字节；将返回 NOT ACK 位。
		读数据字节	0	0	0	1	将收到数据字节；将返回 ACK 位。
0x58	已收到数据字节；已返回 NOT ACK	读数据字节 或	1	0	0	X	重复 START 条件将会发送。
		读数据字节 或	0	1	0	X	将发送 STOP 条件，STO 标志将复位。
		读数据字节	1	1	0	X	将发送 STOP 条件，之后再传输一个 START 条件；STO 标志将被复位。

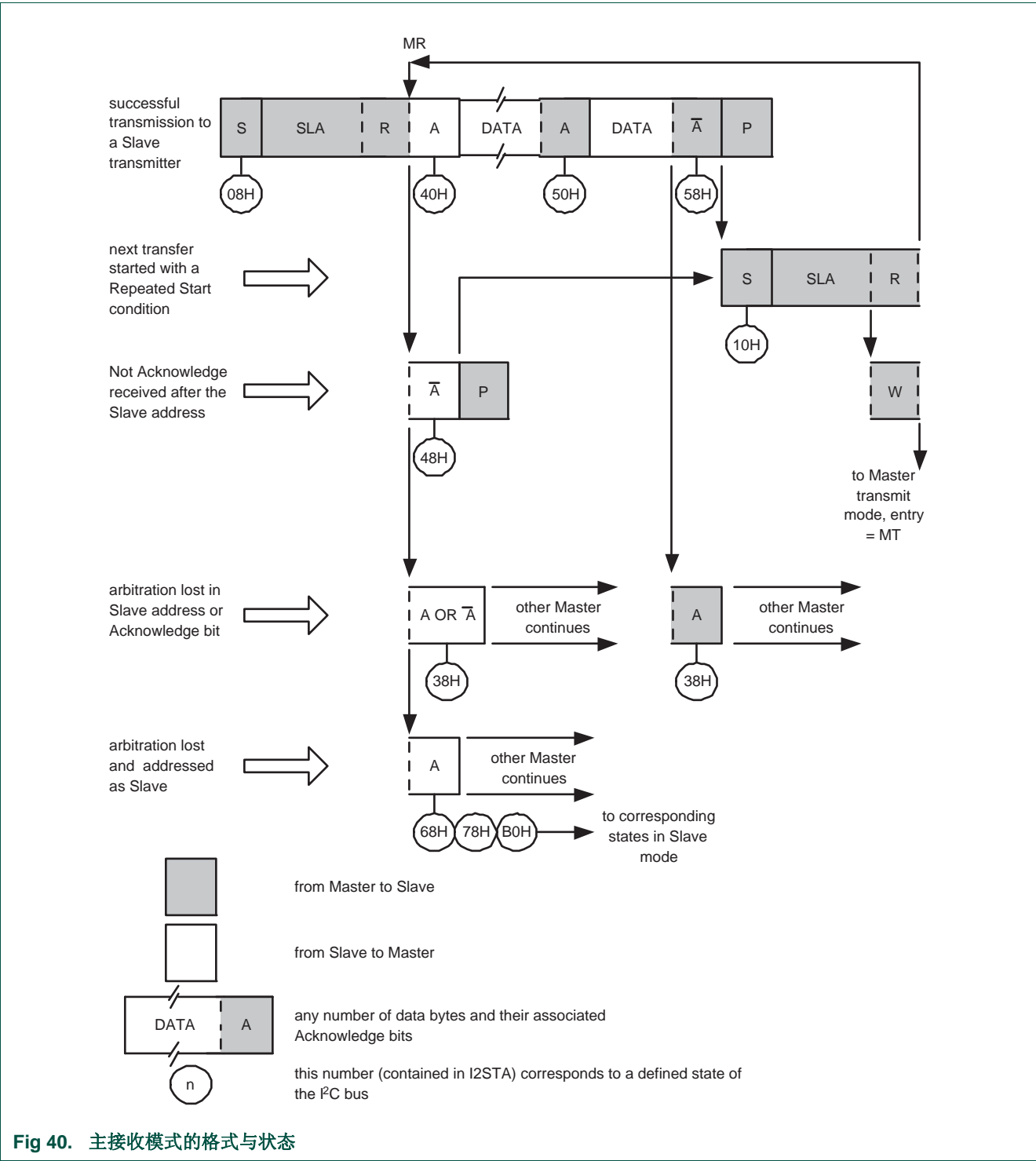


Fig 40. 主接收模式的格式与状态

12.10.3 从接收模式

在从接收模式下，从主发送器接收数据字节 (见 [Figure 41](#)). 为了初始化从接收模式，ADR 和 CON 必须如下加载：

Table 173. 从接收模式中 I2C0ADR 和 I2C1ADR 的用法

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

前面的 7 位 是当主设备所发出地址时，I2C 模块将响应的地址。如果设置了 LSB (GC)，I2C 模块将会响应广播地址 (0x00)； 否则它将忽略广播地址。

Table 174. 用于初始化从接收模式的 I2C0CONSET 和 I2C1CONSET

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

在从模式时，I2C 总线速度的设置不会影响到 I2C 模块。I2EN 必须设置为逻辑 1 以允许 I2C 模块。AA 位必须设置为 1，以使 I2C 块能够应答自身的从地址或广播地址。STA、STO 和 SI 都必须复位。

当 I2ADR 和 I2CON 已被初始化之后，I2C 模块将等待，直到它收到与其从地址相同的地址，并且如果 I2C 模块要工作在从接收模式下，则随后的数据方向位必须是“0”（W）。在已收到自身从机地址和 W 位后，串行中断标志（SI）被设置且可以从 I2STAT 读取有效状态代码。每个状态代码都对应合适的动作，具体动作详见 [Table 175](#). 如果 I2C 模块在主模式时仲裁丢失，也可以进入从接收模式 (见 状态码 0x68 和 0x78)。

如果在传输过程中 AA 位被复位，I2C 模块在接收到下一个数据字节之后，将返回一个非应答（逻辑 1）到 SDA 线上。当 AA 复位时，I2C 块将不会响应自己的从地址或广播地址。然而，I2C 总线仍然被监测，并可在任何时间设置 AA 恢复地址识别。这意味着，AA 位可用于暂将 I2C 模块与 I2C 总线隔离。

Table 175. 从接收模式

状态码 (STAT)	I2C 总线及硬件状态	应用软件响应 To/From DAT	To CON				I2C 硬件的下一个动作
			STA	STO	SI	AA	
0x60	已经收到自身 SLA+W ; 已返回 ACK。	无 DAT 动作 或	X	0	0	0	将接收数据字节, 且将返回 NOT ACK。
		无 DAT 动作	X	0	0	1	将接收数据字节, 且将返回 ACK。
0x68	作为主控器, 在 SLA+R/W 传输阶段, 仲裁丢失; 已收到自 身 SLA+W, 返回 ACK。	无 DAT 动作 或	X	0	0	0	将接收数据字节, 且将返回 NOT ACK。
		无 DAT 动作	X	0	0	1	将接收数据字节, 且将返回 ACK。
0x70	已收到广播地址 (0x00) 已返回 ACK。	无 DAT 动作 或	X	0	0	0	将接收数据字节, 且将返回 NOT ACK。
		无 DAT 动作	X	0	0	1	将接收数据字节, 且将返回 ACK。
0x78	作为主控器, 在 SLA+R/W 传输阶段, 仲裁丢失; 已收到广 播地址, 返回 ACK	无 DAT 动作 或	X	0	0	0	将接收数据字节, 且将返回 NOT ACK。
		无 DAT 动作	X	0	0	1	将接收数据字节, 且将返回 ACK。
0x80	之前用自身 SLV 地址 寻址; 已收到 DATA; 已返回 ACK。	读数据字节 或	X	0	0	0	将接收数据字节, 且将返回 NOT ACK。
		读数据字节	X	0	0	1	将接收数据字节, 且将返回 ACK。
0x88	之前用自身 SLA 地址 寻址; 已收到 DATA; 已返回 NOT ACK。	读数据字节 或	0	0	0	0	转换到不可寻址 SLV 模式; 不能识别自身 SLA 地址或广播地址。
		读数据字节 或	0	0	0	1	转换到不可寻址 SLV 模式; 将识别自身 SLA 地址, 如果 ADR[0] = 1 则识别广播地 址。
		读数据字节 或	1	0	0	0	转换到不可寻址 SLV 模式; 不能识别自身 SLA 地址或广播地址。当总线空闲时, 将 发送一个 START 条件。
		读数据字节	1	0	0	1	转换到不可寻址 SLV 模式; 将识别自身 SLA 地址, 如果 I2ADR[0] = 1 则识别广播 地址。当总线空闲时, 将发送一个 START 条件。
0x90	之前用广播地址寻址; 已收到 DATA; 已返回 ACK。	读数据字节 或	X	0	0	0	将接收数据字节, 且将返回 NOT ACK。
		读数据字节	X	0	0	1	将接收数据字节, 且将返回 ACK。

Table 175. 从接收模式 ...continued

状态码 (STAT)	I2C 总线及硬件状态	应用软件响应 To/From DAT	To CON				I2C 硬件的下一个动作
			STA	STO	SI	AA	
0x98	之前用广播地址寻址；已收到 DATA；已返回 NOT ACK。	读数据字节 或	0	0	0	0	转换到不可寻址 SLV 模式；不能识别自身 SLA 地址或广播地址。
		读数据字节 或	0	0	0	1	转换到不可寻址 SLV 模式；将识别自身 SLA 地址，如果 ADR[0] = 1 则识别广播地址。
		读数据字节 或	1	0	0	0	转换到不可寻址 SLV 模式；不能识别自身 SLA 地址。当总线空闲时，将发送一个 START 条件。
		读数据字节	1	0	0	1	转换到不可寻址 SLV 模式；将识别自身 SLA 地址，如果 ADR[0] = 1 则识别广播地址。当总线空闲时，将发送一个 START 条件。
0xA0	当作为 SLV/REC 或 SLV/TRX 寻址时，已收到 STOP 条件或重复 START 条件。	无 STDAT 动作 或	0	0	0	0	转换到不可寻址 SLV 模式；不能识别自身 SLA 地址或广播地址。
		无 STDAT 动作 或	0	0	0	1	转换到不可寻址 SLV 模式；将识别自身 SLA 地址，如果 I2ADR[0] = 1 则识别广播地址。
		无 STDAT 动作 或	1	0	0	0	转换到不可寻址 SLV 模式；不能识别自身 SLA 地址。当总线空闲时，将发送一个 START 条件。
		无 STDAT 动作 或	1	0	0	1	转换到不可寻址 SLV 模式；将识别自身 SLA 地址，如果 I2ADR[0] = 1 则识别广播地址。当总线空闲时，将发送一个 START 条件。

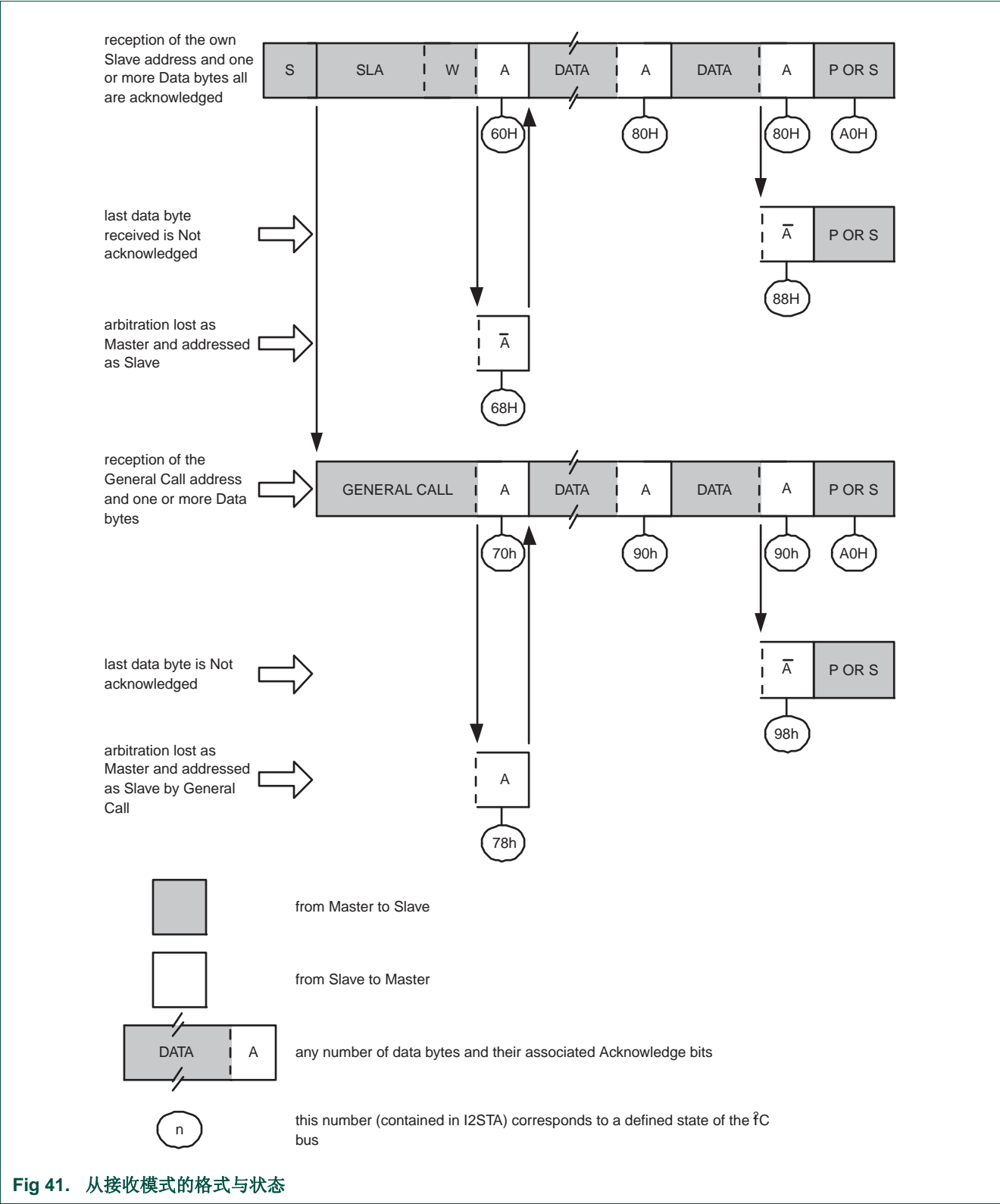


Fig 41. 从接收模式的格式与状态

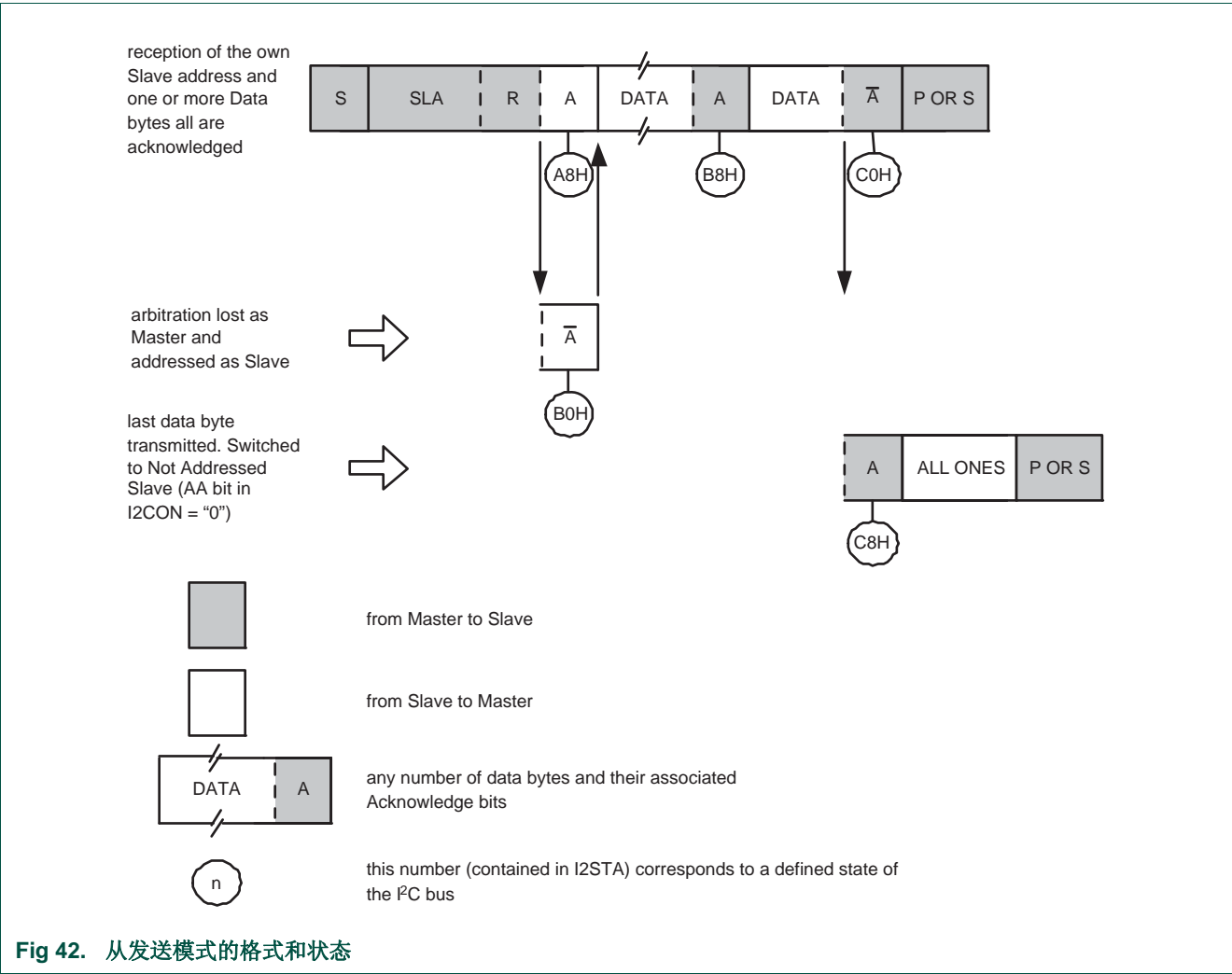
12.10.4 从发送模式

在从发送模式时，数据字节被传送到主接收器 (见 [Figure 42](#)). 数据传输的初始化方法与从接收模式一样。当 **ADR** 和 **CON** 已被初始化之后，**I2C** 模块将等待，直到它收到与其自身从地址相同的地址，且如果 **I2C** 模块要工作在从发送模式下，则随后的数据方向位必须是“1” (**R**)。在其从机地址和 **R** 位已被收到之后，串行中断标志 (**SI**) 被设置，而且可从 **STAT** 读取有效的状态代码。此状态代码是状态服务程序的向量，每一个状态代码对应合适的动作，具体动作详见 [Table 176](#). 如果 **I2C** 模块在主模式时仲裁丢失，也可进入从发送模式 (见状态码 **0xB0**)。

如果 **AA** 位在传输过程中复位，**I2C** 模块将传输完最后一个字节，并进入的状态 **0xC0** 或 **0xC8**。**I2C** 模块切换到不可寻址匹配的从模式，如果它继续传输将忽略主接收器。因此，主接收器接收到的所有串行数据全是 1。虽然 **AA** 复位，**I2C** 模块将不响应自身从地址或广播地址。然而，**I2C** 总线仍然被监测，且在任何时间可设置 **AA** 以恢复地址识别。这意味着，**AA** 位可以被用来暂时将 **I2C** 块与 **I2C** 总线隔离。

Table 176. 从发送模式

状态码 (STAT)	I2C 总线及硬件状态	应用软件响应 To/From DAT	To CON				I2C 硬件的下一个动作
			STA	STO	SI	AA	
0xA8	已接收到自身 SLA+R, 已返回 ACK	加载数据字节 或	X	0	0	0	将传输最后一个数据字节; 将收到 ACK。
		加载数据字节	X	0	0	1	将传输数据字节; 将收到 ACK。
0xB0	作为主控设备, 在 SLA+R/W 传输阶段, 仲裁丢失; 已接收到自 身 SLA+R, 已返回 ACK。	加载数据字节 或	X	0	0	0	将传输最后一个数据字节; 将收到 ACK。
		加载数据字节	X	0	0	1	将传输数据字节; 将收到 ACK。
0xB8	I2DAT 中的数据字节已 发送; 已接收到 ACK。	加载数据字节 或	X	0	0	0	将传输最后一个数据字节; 将收到 ACK。
		加载数据字节	X	0	0	1	将传输数据字节; 将收到 ACK。
0xC0	I2DAT 中的数据字节已 发送; 已接收到 NOT ACK。	无 DAT 动作或	0	0	0	0	转换到不可寻址 SLV 模式; 不能识别自身 SLA 地址或广播地址。
		无 DAT 动作或	0	0	0	1	转换到不可寻址 SLV 模式; 将识别自身 SLA 地址, 如果 ADR[0] = 1 则识别广播地 址。
		无 DAT 动作或	1	0	0	0	转换到不可寻址 SLV 模式; 不能识别自身 SLA 地址或广播地址。当总线空闲时, 将 发送一个 START 条件。
		无 DAT 动作	1	0	0	1	转换到不可寻址 SLV 模式; 将识别自身 SLA 地址, 如果 ADR[0] = 1 则识别广播地 址。当总线空闲时, 将发送一个 START 条件。
0xC8	I2DAT 中的数据字节已 发送 (AA = 0); 已接收 到 ACK。	无 DAT 动作或	0	0	0	0	转换到不可寻址 SLV 模式; 不能识别自身 SLA 地址或广播地址。
		无 DAT 动作或	0	0	0	1	转换到不可寻址 SLV 模式; 将识别自身 SLA 地址, 如果 ADR[0] = 1 则识别广播地 址。
		无 DAT 动作或	1	0	0	0	转换到不可寻址 SLV 模式; 不能识别自身 SLA 地址或广播地址。当总线空闲时, 将 发送一个 START 条件。
		无 DAT 动作	1	0	0	01	转换到不可寻址 SLV 模式; 将识别自身 SLA 地址, 如果 ADR[0] = 1 则识别广播地 址。当总线空闲时, 将发送一个 START 条件。



12.10.5 杂项状态

还有两个 I2STAT 代码，不符合 I2C 硬件定义的状态（见 Table 177）。这些将在下面讨论。

12.10.5.1 STAT = 0xF8

此状态代码表示没有相关信息可用，因为串行中断标志位 SI 没有被设置。当 I2C 模块不参与串行传输时，这种情况出现在其他状态之间。

12.10.5.2 STAT = 0x00

此状态代码表示在 I2C 串行传送期间发生了总线错误。START 或 STOP 条件出现在帧格式中的非法位置，就会产生一个总线错误。此状态代码表示在 I2C 串行传送期间发生了总线错误。START 或 STOP 条件出现在帧格式中的非法位置，就会产生一个总线错误。非常位置可以是在串行传输中的地址字节、数据字节或应答位中。当外部干扰信号干扰内部的 I2C 模块信号时，也可能出现总线错误。当总线错误时，SI 被设置。要恢复总线错误，必须置 STO 标志位为 1、且 SI 位必须被清除。这将导致的 I2C 块进入“不可寻址”的从模式（一个定义的状态），并清除 STO 标志位（不影响 I2CON 其他位）。SDA 和 SCL 线被释放（没有传输 STOP 条件）。

Table 177. 杂项状态

状态码 (STAT)	I2C 总线及硬件状态	应用软件的响应 To/From DAT	To CON				应用软件的响应
			STA	STO	SI	AA	
0xF8	无相关的状态码可用； SI = 0。	无 DAT 动作		无 CON 动作			等待或进行当前传输。
0x00	由于非法的 START 或 TOP 条件，导致总线在 MST 或选定的从模式发生总线错误。在当干扰信号导致的 I2C 模块进入一个未定义状态时，也可能进入 0x00 状态。	无 DAT 动作	0	1	0	X	在 MST 或可寻址 SLV 模式下，仅内部硬件受影响。在所有的情况下，总线被释放，I2C 模块进入不可寻址 SLV 模式，STO 被复位。

12.10.6 一些特殊情况

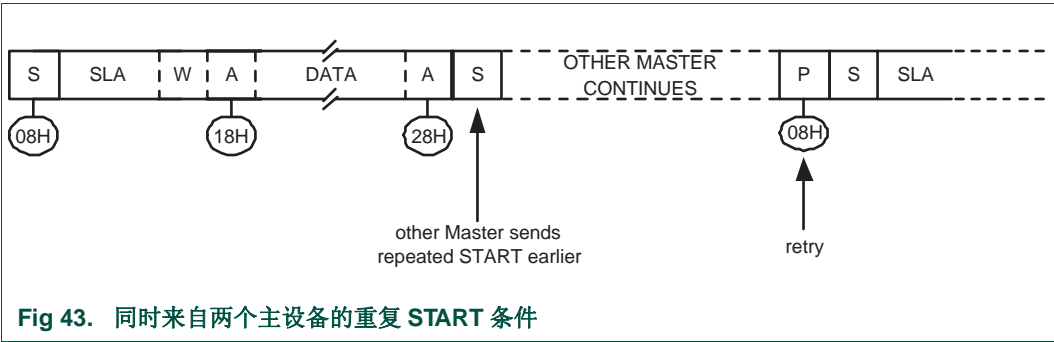
I2C 硬件中有器件能处理在串行传输中可能的发生以下列特殊情况：

- 同时有来自两个主设备的重复 START 条件。
- 仲裁丢失之后的数据传输。
- 强制访问 I2C 总线。
- I2C 总线被 SCL 或 SDA 线上的低电平阻塞。
- 总线错误。

12.10.6.1 同时有来自两个主设备的重复 START 条件

在主发送或主接收模式下，可能生成一个重复 START 条件。如果另一个主设备同时生成一个重复 START 件 (见 [Figure 43](#)). 则将会发生一个特殊的情况。发生这种情况之前，任何一方都没有丢失仲裁，因为它们传输的是相同的数据。

在 I2C 硬件本身产生重复 START 条件之前，如果在 I2C 总线上检测到一个重复 START 条件，它将释放总线，且不产生中断请求。如果另一个主设备通过产生 STOP 条件释放了总线，I2C 模块将传输正常的 START 条件（状态 0x08），并重新开始进行串行数据传输。



12.10.6.2 仲裁丢失后的数据传输

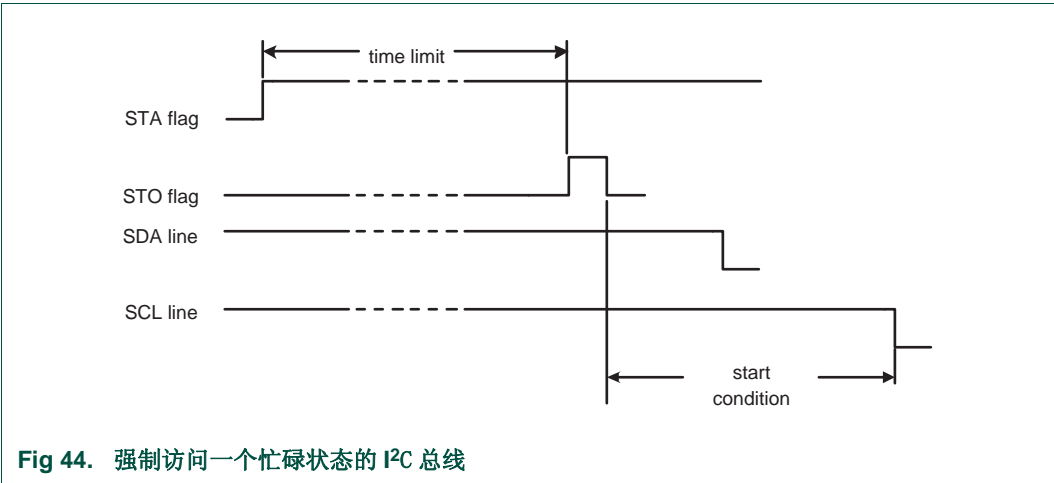
在主发送模式和主接收模式下，仲裁可能丢失 (见 Figure 37). 当 STAT 是以下状态仲裁可能丢失: 0x38、0x68、0x78 和 0xB0 (见 Figure 39 和 Figure 40)。

如果 CON 中的 STA 标志被被这些状态的服务程序设置，那么如果总线再次空闲，在没有 CPU 的干预下，将传输一个 START 条件 (状态 0x08)，并重新开始一个全部串行传送。

12.10.6.3 强制访问 I2C 总线

在一些应用中，可能存在一个不受控制的来源导致总线挂断。在这种情况下，问题可能是由干扰、总线上暂时的中断或在 SDA 和 SCL 之间临时性的短期的电流造成的。

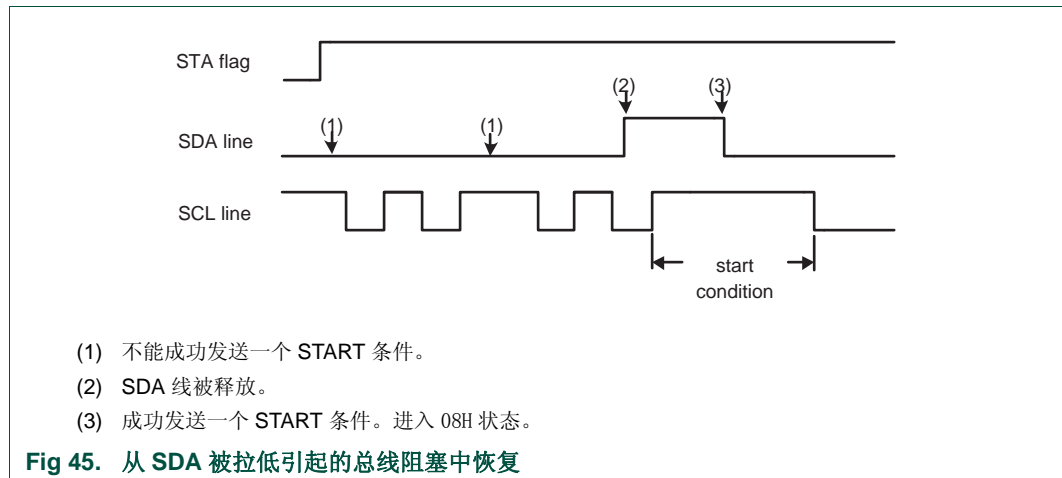
如果不受控制源产生了一个多余的 START 条件或屏蔽了 STOP 条件，那么 I2C 总线将一直保持忙碌状态。如果 STA 的标志被设置，且没有在合理的时间内访问总线，那么可能强制访问 I2C 总线。在 STA 状态标志仍被设置的情况下设置 STO，就会出现强制访问，没有 STOP 条件被传送。此时，I2C 硬件的行为就像收到 STOP 条件一样，将可以发送一个 START 条件。STO 标志由硬件清除 (见 Figure 44)。



12.10.6.4 I²C 总线被 SCL 或 SDA 线上的低电平阻塞

总线上的任何设备将 SDA 和 SCL 线上的电平拉低，就会发生一个 I²C 总线挂断。如果 SCL 线被总线上的设备阻塞（拉低），那么将不可能进行后续的串行传输，并且这个问题必须由拉低 SCL 线的设备解决。

通常情况下，SDA 线在以下情况下可能会被阻塞：总线上的其他设备没有和当前的总线主控设备同步、任何错误的时钟或将噪声脉冲作为时钟。在这种情况下，问题可以通过在 SCL 线上传输一个额外时钟脉冲来解决（见 Figure 45）。I²C 接口中没有一个专门的超时计时器来检测总线受阻，但这可以通过使用系统中另一个计时器来实现。当检测到 SDA 阻塞时，软件可以强制在 SCL 线上产生时钟（最多可达 9 个时钟周期），直到 SDA 被有问题的设备释放。在这一点上，从设备仍可能不同步，因此应产生一个 START 条件以保证所有的 I²C 外设是同步的。



12.10.6.5 总线错误

当 START 或 STOP 条件出现在帧格式中非法的位置上时，就会产生一个总线错误。串行传输中，非法位置有：地址字节、数据字节或应答位。

只有在 I²C 模块作为主设备或从设备并参与串行传输出现总线错误时，I²C 硬件才会作出反应。当检测总线错误后，I²C 模块立即切换到不可寻址从机模式，释放 SDA 和 SCL 线，设置中断标志，并用 0x00 装载状态寄存器。此状态码可用状态服务程序的向量，状态服务程序要么企图中止一次串行传输，要么简单地从错误状态中恢复，见 Table 177。

12.10.7 I²C 状态服务程序

本节提供的操作例子，由不同 I²C 状态服务例程执行。这包括：

- 在复位后，初始化 I²C 块。
- I²C 中断服务。
- 支持所有 4 个 I²C 工作模式的 26 个状态服务程序。

12.10.8 初始化

在初始化的例子中，I2C 模块可以为主机和从机模式。对于每个模式，都要有一个缓冲区用于发送和接收。初始化程序执行以下功能：

- 往 ADR 中加载模块自身的从地址和广播位 (GC)。
- 允许 I2C 中断并设置中断优先级。
- 同时设置 CON 中的 I2EN 和 SI 位，允许从设备模式；通过 SCLH 和 SCLL 寄存器定义串行时钟频率（对主模式）。主设备程序必须在主程序中开始。

I2C 硬件开始在 I2C 总线上检测自身从地址和广播位。如果检测到广播地址或自身从地址被，将产生中断请求，并且在 STAT 中加载了一个合适的状态信息。

12.10.9 I2C 中断服务

当进入 I2C 中断，STAT 包含一个状态代码，该状态代码确定 26 个状态服务中的哪个将被执行。

12.10.10 状态服务程序

每个状态例程是 I2C 中断例程的一部分，并处理 26 个状态之一。

12.10.11 将状态服务改为应用程序

状态服务的例子表明，响应 26 个 I2C 状态代码，必须一个执行典型动作。如果 4 个 I2C 操作模式中一个或更多模式没有被使用，相关的状态服务可被忽略，只要注意确认这些状态永远不会发生。

应用程序中，在 I2C 操作期间，可能需要执行某种超时程序，以处理总线无效或服务程序丢失。

12.11 软件举例

12.11.1 初始化程序

初始化 I2C 接口为主 / 从模式的例子。

1. 将自身从地址加载到 I2ADR，如果需要，允许广播的识别。
2. 允许 I2C 中断。
3. 写 0x44 到 CONSET 以设置 I2EN 和 AA 位，允许从设备功能。仅是主设备功能，则写 0x40 到 CONSET。

12.11.2 开始主设备传输功能

通过设置缓冲区、指针和数据计数，以开始主发送操作，然后启动一个 START 条件

1. 初始化主设备数据计数器。
2. 设置将要发送的从地址，再加上一个写 (W) 位。

3. 写 0x20 到 I2CONSET 以设置 STA 位。
4. 将要发出的数据写到主设备发送缓冲区中。
5. 初始化主数据计数器，以确定将被发送的消息的长度。
6. 退出。

12.11.3 开始主接收功能

通过设立缓冲区、指针和数据计数，以开始主接收操作，然后启动一个 START 条件。

1. 初始化主设备数据计数器。
2. 设置将要发送的从地址，再加上一个读（R）位。
3. 写 0x20 到 CONSET 以设置 STA 位。
4. 设置主接收缓冲区。
5. 初始化主数据计数器，用于确定将被接收消息的长度。
6. 退出。

12.11.4 I²C 中断服务程序

确定 I²C 状态以及将要执行哪一个状态服务程序。

1. 从 STA 中读取 I²C 的状态。
2. 根据状态值，跳转到 26 种可能的状态服务程序之一。

12.11.5 无具体模式的状态

12.11.5.1 状态：0x00

总线错误。进入不可寻址从模式并释放总线。

1. 写 0x14 到 I2CONSET 以设置 STO 和 AA 位。
2. 写 0x08 到 I2CONCLR 以清除 SI 标志。
3. 退出。

12.11.5.2 主设备状态

主接收模式和主发送模式下都会出现状态 08 和状态 10。RW 位决定了下一个状态是主发送模式还是主接收模式。

12.11.5.3 状态：0x08

已经发送一个 START 条件。将要发送从设备地址 + R/W 位，并将接收一个 ACK 位。

1. 写从地址和 R/W 位到 DAT。
2. 写 0x04 到 CONSET 以设置 AA 位。
3. 写 0x08 到 CONCLR 以清除 SI 标志。
4. 设置主发送模式缓冲区。

5. 设置主接收模式缓冲区。
6. 初始化主数据计数器。
7. 退出。

12.11.5.4 状态 : 0x10

已发送一个重复 START 条件。将要发送从设备地址 + R/W 位，并接收一个 ACK 位。

1. 写从地址和 R/W 位 到 I2DAT。
2. 写 0x04 到 CONSET 以设置 AA 位。
3. 写 0x08 到 CONCLR 以清除 SI 标志。
4. 设置主发送模式缓冲区。
5. 设置主接收模式缓冲区。
6. 初始化主数据计数器。
7. 退出。

12.11.6 主发送状态

12.11.6.1 状态 : 0x18

前一次的状态是状态 8 或状态 10，已完成从地址 + Write 位传输，已收到 ACK。将要发送第一个数据，并接收 ACK 位。

1. 从主发送模式缓冲区加载第一个数据到 DAT。
2. 写 0x04 到 CONSET 以设置 AA 位。
3. 写 0x08 到 CONCLR 以清除 SI 标志。
4. 主发送缓冲区指针加 1。
5. 退出。

12.11.6.2 状态 : 0x20

已经完成从地址 + Write 传输，已接收到 ACK。将发送一个 STOP 条件。

1. 写 0x14 到 CONSET 以设置 STO 和 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 退出。

12.11.6.3 状态 : 0x28

数据传输完成，已接收到 ACK。如果传输的数据是最后一个数据字节，那么传输一个 STOP 条件，否则传输下一个数据。

1. 主数据计数器减 1，如果不是最后一个数据字节则跳到步骤 5。
2. 写 0x14 到 CONSET 以设置 STO 和 AA 位。
3. 写 0x08 到 CONCLR 以清除 SI 标志。
4. 退出。
5. 从主发送模式缓冲区加载下一个数据到 DAT。

6. 写 0x04 到 CONSET 以设置 AA 位。
7. 写 0x08 到 CONCLR 以清除 SI 标志。
8. 主发送缓冲区指针加 1。
9. 退出。

12.11.6.4 State: 0x30

数据传输完成，接收到 NOT ACK。将传输一个 STOP 条件。

1. 写 0x14 到 CONSET 以设置 STO 和 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 退出。

12.11.6.5 State: 0x38

在传输从地址 + Write 或数据时仲裁丢失。总线已经释放，并已进入不可寻址的从模式。当总线再次空闲时，将传输一个 START 条件。

1. 写 0x24 到 CONSET 以设置 STA 和 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 退出。

12.11.7 主接收状态

12.11.7.1 状态 : 0x40

前一次状态是状态 08 或状态 10。已经完成从地址 +Read 传输，已返回 ACK。将接收到数据并返回 ACK。

1. 写 0x04 到 CONSET 以设置 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 退出。

12.11.7.2 状态 : 0x48

已完成从地址 +Read 的传输，已接收到 NOT ACK。将传输一个 STOP 条件。

1. 写 0x14 到 CONSET 以设置 STO 和 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 退出。

12.11.7.3 状态 : 0x50

已接收到数据，已返回 ACK。将从 I2DAT 读出数据。将接收到额外的数据。如果这是最后一个数据字节则将返回 NOT ACK，否则将返回 ACK。

1. 将 I2DAT 中的数据读出，存到主接收模式缓冲区。
2. 主数据计数器减 1，如果不是最后一个数据字节跳到步骤 5。
3. 写 0x0C 到 CONCLR 以清除 SI 标志和 AA 位。
4. 退出。

5. 写 0x04 到 CONSET 以设置 AA 位。
6. 写 0x08 到 CONCLR 以清除 SI 标志。
7. 主接收缓冲区的指针加 1。
8. 退出。

12.11.7.4 状态 : 0x58

已收到数据，已返回 NOT ACK。将从 I2DAT 读出数据。将传输一个 STOP 条件。

1. 将 I2DAT 中的数据读出，存到主接收模式缓冲区。
2. 写 0x14 到 CONSET 以设置 STO 和 AA 位。
3. 写 0x08 到 CONCLR 以清除 SI 标志。
4. 退出。

12.11.8 从接收状态

12.11.8.1 状态 : 0x60

已收到自身从地址 + Write 位，已返回 ACK。将接收数据并返回 ACK。

1. 写 0x04 到 CONSET 以设置 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 设置从接收器模式数据缓冲区。
4. 初始化从设备数据计数器。
5. 退出。

12.11.8.2 状态 : 0x68

作为总线主控器时，在自身从地址 +R/W 中阶段仲裁丢失。已收到自身从地址 +Write。将接收到数据并返回 ACK。当总线再次空闲时，设置 STA 位以重新启动主模式。

1. 写 0x24 到 CONSET 以设置 STA 和 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 设置从接收器模式数据缓冲区。
4. 初始化从设备数据计数器。
5. 退出。

12.11.8.3 状态 : 0x70

已经接收到广播信息，ACK 已返回。将接收数据并返回 ACK。

1. 写 0x04 到 CONSET 以设置 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 设置从接收器模式数据缓冲区。
4. 初始化从设备数据计数器。

5. 退出。

12.11.8.4 状态 : 0x78

作为总线主控器时，在自身从地址 +R/W 传输阶段仲裁丢失。已收到广播地址，已返回 ACK。将要接收数据并返回 ACK。在总线再次空闲时，设置 STA 位以重新启动主模式。

1. 写 0x24 到 CONSET 以设置 STA 和 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 设置从接收器模式数据缓冲区。
4. 初始化从设备数据计数器。
5. 退出。

12.11.8.5 状态 : 0x80

前一次使用自身从地址寻址，数据已收到，已返回 ACK。将要读额外的数据。

1. 将 DAT 中的数据读出，存到从接收模式缓冲区。
2. 从数据计数器减 1，如果不是最后一个数据字节跳到步骤 5。
3. 写 0x0C 到 CONCLR，以清除 SI 标志和 AA 位。
4. 退出。
5. 写 0x04 到 CONSET 以设置 AA 位。
6. 写 0x08 到 CONCLR 以清除 SI 标志。
7. 从接收缓冲区指针加 1。
8. 退出。

12.11.8.6 状态 : 0x88

前一次使用自身从地址寻址，数据已接收到，已返回 NOT ACK 位。不保存接收到的数据。将进入不可寻址的从模式。

1. 写 0x04 到 CONSET 以设置 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 退出。

12.11.8.7 状态 : 0x90

前一次寻址使用广播地址。数据已收到，已返回 ACK 位。将保存接收到的数据。只有第一个数据字节接收时返回 ACK，其他数据字节接收将返回 NOT ACK。

1. 将 DAT 中的数据读出，存到从接收模式缓冲区。
2. 写 0x0C 到 CONCLR 以清除 SI 标志和 AA 位。
3. 退出。

12.11.8.8 状态 : 0x98

前一次寻址使用广播地址。数据已收到，已返回 NOT ACK 位。不保存接收到的数据。将进入不可寻址的从模式。

1. 写 0x04 到 CONSET 以设置 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 退出。

12.11.8.9 状态 : 0xA0

当仍作为一个可寻址从模式时，接收到一个 STOP 条件或重复 START 条件。不保存接收到的数据。将进入不可寻址的从模式。

1. 写 0x04 到 CONSET 以设置 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 退出。

12.11.9 从发送模式

12.11.9.1 状态 : 0xA8

已接收到自身从地址 + 读位，已返回 ACK。将要发出数据并将收到 ACK 位。

1. 从从发送模式缓冲区加载第一个数据到 DAT。
2. 写 0x04 到 CONSET 以设置 AA 位。
3. 写 0x08 到 CONCLR 以清除 SI 标志。
4. 设置从发送模式数据缓冲区。
5. 从发送缓冲区指针加 1。
6. 退出。

12.11.9.2 状态 : 0xB0

作为总线主控器时，在自身从地址 +R/W 传输阶段仲裁丢失。已收到自身从地址 +Read，已返回 ACK。将要发送数据并返回 ACK。在总线再次空闲时，设置 STA 位以重新启动主模式。

1. 从从发送模式缓冲区加载第一个数据到 DAT。
2. 写 0x24 到 CONSET 以设置 STA 和 AA 位。
3. 写 0x08 到 CONCLR 以清除 SI 标志。
4. 设置从发送器模式数据缓冲区。
5. 从发送缓冲区指针加 1。
6. 退出。

12.11.9.3 状态 : 0xB8

数据已经传输完成，已经返回 ACK。将要发送数据，并接收 ACK。

1. 从从发送模式缓冲区加载数据字节到 DAT。

2. 写 0x04 到 CONSET 以设置 AA 位。
3. 写 0x08 到 CONCLR 以清除 SI 标志。
4. 从发送缓冲区的指针加 1。
5. 退出。

12.11.9.4 状态 : 0xC0

数据已经传输完成，已经返回 NOT ACK。进入不可寻址的从模式。

1. 写 0x04 到 CONSET 以设置 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 退出。

12.11.9.5 状态 : 0xC8

最后一个数据字节已经传输完成，已返回 ACK。进入不可寻址的从模式。

1. 写 0x04 到 CONSET 以设置 AA 位。
2. 写 0x08 到 CONCLR 以清除 SI 标志。
3. 退出。

13.1 怎样阅读本章

C_CAN 模块只在 LPC11CXX (LPC11C00 系列) 中才有。

LPC11C22 和 LPC11C24 处理器包含一个片上高速收发器。对这些部分来说，CAN_RXD, CAN_TXD 信号内连到片上收发器，而收发器的信号与引脚引出（见 [Table 179](#)）。

13.2 基本配置

C_CAN 通过以下寄存器进行配置：

1. 电源：通过 SYSAHBCLKCTRL 寄存器的第 17 位来配置 ([Table 21](#))。
2. 时钟：为 C_CAN 模块提供一个准确的外设时钟，通常选择系统振荡器作为主时钟 ([Table 18](#)) 或是系统 PLL 的输入时钟 ([Table 16](#))。如果 C_CAN 的波特率要求在 100kbit/s 以上时，不要选择 IRC 振荡器。
3. 复位：在访问 C_CAN 模块之前，确保 PRESETCTRL 寄存器 ([Table 9](#)) 中 CAN_RST_N (位 3) 被设置为 1，这会令 C_CAN 模块的复位信号无效。

C_CAN 外设时钟 (C_CAN 的系统时钟) 和 C_CAN 可编程时钟分频器的时钟 (见 [Table 210](#)) 由系统时钟提供 (见 [Table 21](#))。这个时钟可以通过设置 SYSAHBCLKCTRL 寄存器 17 位来关闭从而节省电力。

注意：如果 C_CAN 的波特率要求在 100kbit/s 以上时，系统振荡器必须选择为系统时钟源。对于低一点的波特率，IRC 也可以作为时钟源。

13.3 特征

- 遵从 2.0 版本协议的 A 和 B 部分
- 支持的位速率可高达 1Mbit/s
- 支持 32 个报文对象
- 每个报文对象具有其自身的标示符屏蔽
- 提供可编程的 FIFO 模式（链接报文对象）
- 提供可屏蔽中断。
- 对于定时触发的 CAN 应用程序，提供被禁止的自动重发（DAR）模式
- 提供可编程的回环模式来进行自测试操作

13.4 概述

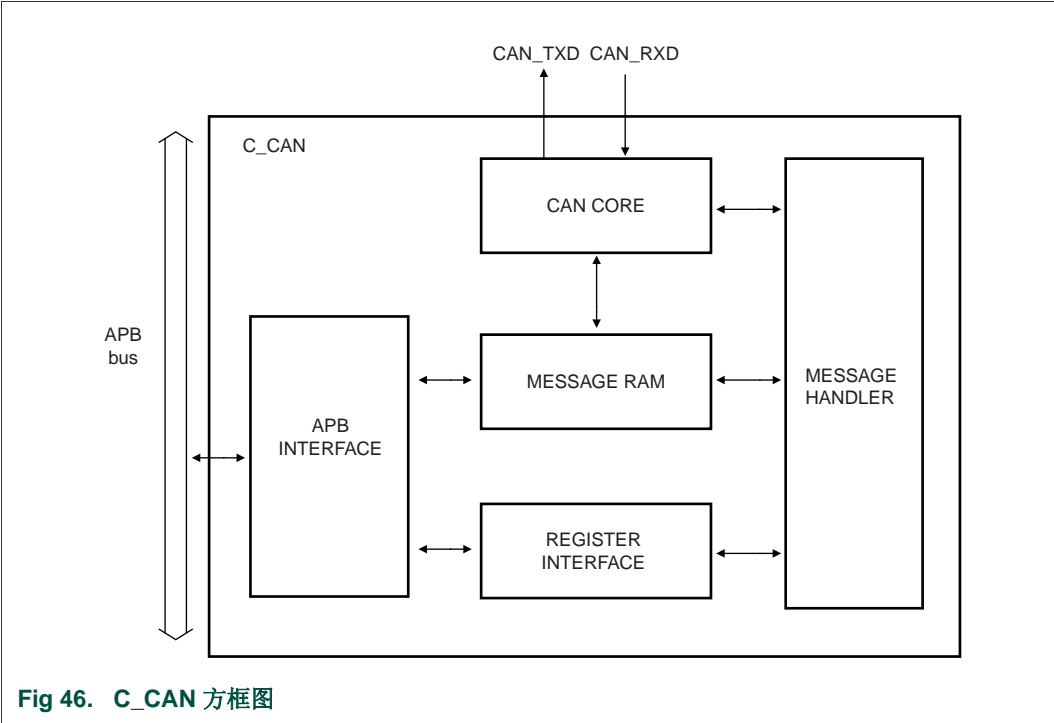
控制器局域网 (CAN) 的定义是高性能的通信协议，用于执行串行数据的通信，依据 2.0B 版本的 CAN 规范，C_CAN 控制器被设计用于完全实现 CAN 协议。C_CAN 控制器可以与低成本多路复用网络建立强大的局域网，这通过分布式实时控制来实现，同时具有极高的安全性。

CAN 控制器由 CAN 内核，报文 RAM，报文处理程序，控制寄存器和 APB 接口组成。

对于 CAN 网络中的通信，要单独配置各个报文对象。所接收到的需要进行过滤的报文对象和标识符屏蔽是存放在报文 RAM 中的。

所有关于报文处理的函数均由报文处理程序执行。这些函数包含接收过滤函数、CAN 内核和报文 RAM 之间的报文传输函数、处理发送请求的函数和产生模块中断的函数。

CAN 控制器的寄存器集可由外部 CPU 通过 APB 总线来进行直接访问。这些寄存器用来控制配置 CAN 内核和报文处理程序，以及访问报文 RAM。



13.5 引脚描述

Table 178. CAN 引脚描述 (LPC11C12/C14)

引脚	类型	描述
CAN_TXD	O	C_CAN 发送输出
CAN_RXD	I	C_CAN 接收输入

Table 179. CAN 引脚描述 (LPC11C22/C24)

引脚	类型	描述
CANL	I/O	低电平 CAN 总线
CANH	I/O	高电平 CAN 总线
STB	I	CAN 收发器安静模式下的控制输入 (低 = 正常模式, 高 = 安静模式)
VDD_CAN	-	为 CAN 收发器的 I/O 等级提供电源
V _{CC}	-	为 CAN 收发器提供电源
GND	-	CAN 收发器地线

13.6 寄存器描述

C_CAN 寄存器是 32 位宽的寄存器。

二组接口寄存器 (IF1 和 IF2) 控制 CPU 访问报文 RAM。它们把发送给 RAM 或接受自 RAM 的数据进行缓冲, 避免在 CPU 访问和报文接收 / 发送之间造成冲突。

Table 180. 寄存器汇总 : CCAN (基址 0x4005 0000)

名称	访问	地址偏移量	描述	复位值
CANCNTL	R/W	0x000	CAN 控制	0x0001
CANSTAT	R/W	0x004	状态寄存器	0x0000
CANEC	RO	0x008	错误计数器	0x0000
CANBT	R/W	0x00C	位定时寄存器	0x2301
CANINT	RO	0x010	中断寄存器	0x0000
CANTEST	R/W	0x014	测试寄存器	-
CANBRPE	R/W	0x018	波特率预分频扩展寄存器	0x0000
-	-	0x01C	保留	-
CANIF1_CMDREQ	R/W	0x020	报文报文接口 1 命令请求	0x0001
CANIF1_CMDMSK	R/W	0x024	报文接口 1 命令屏蔽 (写方向)	0x0000
CANIF1_CMDMSK	R/W	0x024	报文接口 1 命令屏蔽 (读方向)	0x0000
CANIF1_MSK1	R/W	0x028	报文接口 1 屏蔽 1	0xFFFF
CANIF1_MSK2	R/W	0x02C	报文接口 1 屏蔽 2	0xFFFF
CANIF1_ARB1	R/W	0x030	报文接口 1 仲裁 1	0x0000
CANIF1_ARB2	R/W	0x034	报文接口 1 仲裁 2	0x0000
CANIF1_MCTRL	R/W	0x038	报文接口 1 报文控制	0x0000
CANIF1_DA1	R/W	0x03C	报文接口 1 数据 A1	0x0000
CANIF1_DA2	R/W	0x040	报文接口 1 数据 A2	0x0000
CANIF1_DB1	R/W	0x044	报文接口 1 数据 B1	0x0000
CANIF1_DB2	R/W	0x048	报文接口 1 数据 B2	0x0000
-	-	0x04C - 0x07C	保留	-
CANIF2_CMDREQ	R/W	0x080	报文接口 2 命令请求 (写方向)	0x0001

Table 180. 寄存器汇总：CCAN (基址 0x4005 0000)

名称	访问	地址偏移量	描述	复位值
CANIF2_CMDREQ	R/W	0x080	报文接口 2 命令请求 (读方向)	0x0001
CANIF2_CMDMSK	R/W	0x084	报文接口 2 命令屏蔽	0x0000
CANIF2_MSK1	R/W	0x088	报文接口 2 屏蔽 1	0xFFFF
CANIF2_MSK2	R/W	0x08C	报文接口 2 屏蔽 2	0xFFFF
CANIF2_ARB1	R/W	0x090	报文接口 2 仲裁 1	0x0000
CANIF2_ARB2	R/W	0x094	报文接口 2 仲裁 2	0x0000
CANIF2_MCTRL	R/W	0x098	报文接口 2 报文控制	0x0000
CANIF2_DA1	R/W	0x09C	报文接口 2 数据 A1	0x0000
CANIF2_DA2	R/W	0x0A0	报文接口 2 数据 A2	0x0000
CANIF2_DB1	R/W	0x0A4	报文接口 2 数据 B1	0x0000
CANIF2_DB2	R/W	0x0A8	报文接口 2 数据 B2	0x0000
-	-	0x0AC - 0x0FC	保留	-
CANTXREQ1	RO	0x100	发送请求 1	0x0000
CANTXREQ2	RO	0x104	发送请求 2	0x0000
-	-	0x108 - 0x11C	保留	-
CANND1	RO	0x120	新数据 1	0x0000
CANND2	RO	0x124	新数据 2	0x0000
-	-	0x128 - 0x13C	保留	-
CANIR1	RO	0x140	中断挂起 1	0x0000
CANIR2	RO	0x144	中断挂起 2	0x0000
-	-	0x148 - 0x15C	保留	-
CANMSGV1	RO	0x160	报文有效 1	0x0000
CANMSGV2	RO	0x164	报文有效 2	0x0000
-	-	0x168 - 0x17C	保留	-
CANCLKDIV	R/W	0x180	CAN 时钟分频器寄存器	0x0000

13.6.1 CAN 协议寄存器

13.6.1.1 CAN 控制寄存器

CANCTRL 寄存器的复位值 0x0001 允许软件进行初始化（INIT=1）。C_CAN 不会影响 CAN 总线，直至 CPU 将 INIT 位复位为 0。

Table 181. CAN 控制寄存器 (CANCNTL, 地址 0x4005 0000) 位描述

位	符号	值	描述	复位值	访问
0	INIT		初始化	1	R/W
		0	正常操作。		
		1	启动初始化。复位时，软件需要初始化 CAN 控制器。		
1	IE		模块中断允许	0	R/W
		0	禁止 CAN 中断。中断线总是为高电平。		
		1	允许 CAN 中断。中断线被设置为低电平，并保持为低电平，直到所有挂起的中断被清除。		
2	SIE		状态更改中断允许	0	R/W
		0	禁止状态更改中断，不会产生状态更改中断。		
		1	允许状态更改中断。当成功完成报文传输或检测到 CAN 总线错误时，产生状态更改中断。		
3	EIE		错误中断允许	0	R/W
		0	禁止错误中断，不会产生错误中断。		
		1	允许错误中断。CANSTAT 寄存器的 BOFF 或 EWARN 位发生改变时会产生中断。		
4	-	-	保留	0	-
5	DAR		禁止自动重发。	0	R/W
		0	允许被干扰报文的自动重发。		
		1	禁止自动重发。		
6	CCE		配置更改允许	0	R/W
		0	CPU 不对位定时寄存器进行写访问		
		1	CPU 会在 INIT 位为 1 时对 CANBT 寄存器进行写访问。		
7	TEST		测试模式允许	0	R/W
		0	正常模式		
		1	测试模式		
31:8	-		保留	-	-

注意：总线关闭恢复序列（见 2.0 版本的 CAN 规范）不可以通过置位或复位 INIT 位来变短。如果设备进入总线关闭状态，它将会置位 INIT，停止所有的总线活动。一旦 CPU 清除了 INIT，设备将会先等待 129 个总线空闲发生（129×11 个连续高电平 / 隐性位），然后再恢复正常操作。在结束总线关闭恢复序列时，错误管理计数器将会复位。

在复位 INT 后的等待时间内，每次都监控到 11 个高电平 / 隐性位的序列，而 Bit0Error 代码被写入到状态寄存器 CANSTAT 中，使得 CPU 可以监控总线关闭恢复序列的执行状态，从而决定是否令 CAN 总线一直处于低电平 / 显性或连续被干扰。

13.6.1.2 CAN 状态寄存器

BOFF、EWARN、RXOK、TXOK 和 LEC 位可以产生状态中断。BOFF 和 EWARN 产生错误中断。如果 EIE 和 SIE 分别在 CANCTRL 寄存器里被设置为允许，那么 RXOK，TXOK 和 LEC 产生状态更改中断。

EPASS 位发生改变，写 RXOK、TXOK 或 LEC 将永不会产生状态中断。

读 CANSTAT 寄存器将会在 CANIR 寄存器中清除状态中断值（0x8000）。

Table 182. CAN 状态寄存器 (CANSTAT, 地址 0x4005 0004) 位描述

位	符号	值	描述	复位值	访问
2:0	LEC		最近错误代码 出现在 CAN 总线上的最近错误类型。LEC 域保存着表示最近出现在 CAN 总线上的错误类型的代码。当无错误地完成报文传输（接收或发送）时，该域将会被清除为 ‘0’。‘111’ 是没有使用的编码，CPU 可以写入这个编码来检查更新。	000	R/W
		0x0	无错误		
		0x1	Stuff error: 在接受报文里，序列中存在着超过 5 个相同的位，这是不允许发生的。		
		0x2	Form error: 接受帧的固定格式部分的格式错误。		
		0x3	AckError: 该 CAN 内核传送的报文不被应答。		
		0x4	Bit1Error: 在报文传送期间（仲裁域除外），设备想要发送高电平 / 隐性电平（位逻辑值为 ‘1’），但是监控到的总线为低电平 / 显性电平。		
		0x5	Bit0Error: 在报文（或应答位、或有效错误标志、或过载标志）传送期间，设备想要发送低电平 / 显性电平（数据或标识符位逻辑值为 ‘0’），但是监控到的总线值是高电平 / 隐性电平。在总线关闭恢复期间，在每次监控到 11 个高电平 / 隐性位的序列时，要将该状态置位。这可令 CPU 监控总线关闭恢复序列的进程（表示总线不处于低电平 / 显性电平或连续被干扰）。		
		0x6	CRCError: 所接到的报文中的 CRC 校验和不正确。		
		0x7	Unused: 未检测到 CAN 总线事件（由 CPU 写入）。		
3	TXOK		成功发送报文 该位由 CPU 复位。CAN 控制器不能将它复位	0	R/W
		0	由于该位由 CPU 复位，因此报文没有成功发送		
		1	由于该位由 CPU 最后复位，因此报文成功发送（无错误并至少被另一个节点进行应答）。		

Table 182. CAN 状态寄存器 (CANSTAT, 地址 0x4005 0004) 位描述
...continued

位	符号	值	描述	复位值	访问
4	RXOK		成功接受报文 该位由 CPU 复位。CAN 控制器不能将它复位。	0	R/W
		0	由于该位由 CPU 最后复位，因此报文没有成功发送。		
		1	由于该位由 CPU 最后设置为 0，因此报文接收成功，与接收过滤的结果无关。		
5	EPASS		错误消极	0	RO
		0	CAN 控制器处于错误有效状态。		
		1	CAN 控制器处于 CAN2.0 规范中所定义的错误消极状态。		
6	EWARN		警告状态	0	RO
		0	二个错误计数器的错误警告数低于 96 这个限数		
		1	EML 中至少有一个错误计数器的数值达到了 96 个错误警告上限		
7	BOFF		总线关闭状态	0	RO
		0	CAN 模块不处于总线关闭状态		
		1	CAN 控制器处于总线关闭状态		
31:8	-	-	保留		

13.6.1.3 CAN 错误计数器

Table 183. CAN 错误计数器 (CANEC, 地址 0x4005 0008) 位描述

位	符号	值	描述	复位值	访问
7:0	TEC[7:0]		发送错误计数器 发送错误计数器的当前值 (最大值 255)	0	RO
14:8	REC[6:0]		接受错误计数器 (最大值 127)	-	RO
15	RP		接收错误消极	-	RO
		0	接收计数器没有达到错误消极水平		
		1	接收计数器达到了在 CAN2.0 规范中所定义的错误消极水平		
31:16	-	-	保留	-	-

13.6.1.4 CAN 位定时寄存器

Table 184. CAN 位定时寄存器 (CANBT, 地址 0x4005 000C) 位描述

Bit	Symbol	Description	Reset value	Access
5:0	BRP	波特率预分频器 振荡器频率被分频的值是用于产生位时间量子。位时间值是 该量子的整数倍，波特率预分频的有效值为 0 ~ 63. [1]	000001	R/W
7:6	SJW	(重新) 同步跳转宽度有效的编程值为 0 ~ 3. [1]	00	R/W
11:8	TSEG1	T 采样点之前的时间段有效值为 1~15. [1]	0011	R/W
14:12	TSEG2	采样点之后的时间段有效值为 0~7. [1]	010	R/W
31:15	-	保留	-	-

[1] 硬件把往这些位编写入的值理解为位值 +1。

例如，将模块时钟 CAN_CLK LPC11Cx 系列 Cortex-M0 微控制器系统时钟设为 8MHz，复位值将 C_CAN 配置，以进行位速率为 500KBit/s 的操作。

如果 CANCTRL 中的配置更改允许，且软件初始化了控制器（CAN 控制寄存器中的 CCE 和 INIT 位被置位），则寄存器都为只写寄存器。

有关位定时的详细描述，请参考 [Section 13.7.5](#) 和 1.2 版本的 Bosch C_CAN 用户手册

波特率预分频器

位时间量子 t_q 由 BRP 值决定：

$$t_q = \text{BRP} / f_{\text{sys}}$$

(f_{sys} 是 C_CAN 模块的 LPC11Cx 系列 Cortex-M0 微控制器系统时钟)。

时间段 1 和 2

时间段 TSEG1 和 TSEG2 决定每位时间的时间量子数量和采样点的位置：

$$t_{\text{TSEG1/2}} = t_q \cdot (\text{TSEG1/2} + 1)$$

同步跳转宽度

为了对不同总线控制器的时钟振荡器之间的相位漂移进行补偿，任何总线控制器必须在当前传送的相关信号边沿上重新同步。同步跳转宽度 t_{SJW} 定义时钟周期的最大数目，执行一次重新同步操作时，某位的周期可能变短或变长：

$$t_{\text{SJW}} = t_q \cdot (\text{SJW} + 1)$$

13.6.1.5 CAN 中断寄存器

Table 185. CAN 中断寄存器 (CANINT, 地址 0x4005 0010) 位描述

位	符号	描述	复位值	访问
15:0	INTID	0x0000 = 无中断挂起 0x0001 - 0x0020 = 引发中断的报文对象编号 0x0021 - 0x7FFF = 未使用 0x8000 = 状态中断 0x8001 - 0xFFFF = 未使用	0	R
31:16	-	保留	-	-

如果有几个中断被挂起，CAN 中断寄存器则会指向具有最高优先级的挂起中断，而忽略它们的时间顺序。中断一直将保持挂起直到 CPU 将其清除。如果 INTID 不同于 0x0000，而且 IE 被置位，则到 CPU 的中断线有效。中断线保持有效直到 INTID 返回 0x0000 值（原因是中断被复位）或直到 IE 被复位。

状态中断具有最高的优先级。在这些报文中断里，报文对象的中断优先级是随着报文编号的增加而递减。

通过清除报文对象的 INTPND 位，即可以清除报文中断。而状态中断的清除则通过读取状态寄存器来完成。

13.6.1.6 CAN 测试寄存器

通过置位 CAN 控制寄存器的测试位，即可允许对测试寄存器的写访问。

用户可以组合出不同的测试功能，但是，当选择了 TX[1:0] 不等于 “00” 时，报文传输被干扰。

Table 186. CAN 测试寄存器 (CANTEST, 地址 0x4005 0014) 位描述

位	符号	值	描述	复位值	访问
1:0	-	-	保留		-
2	BASIC		基本模式	0	R/W
		0	基本模式禁止		
		1	IF1 寄存器用作 TX 缓冲区，IF2 寄存器用作 RX 缓冲区。		
3	SILENT		安静模式	0	R/W
		0	正常模式		
		1	模块处于安静模式		
4	LBACK		回环模式	0	R/W
		0	禁止回环模式		
		1	允许回环模式		

Table 186. CAN 测试寄存器 (CANTEST, 地址 0x4005 0014) 位描述

位	符号	值	描述	复位值	访问
6:5	TX		CAN_TXD 引脚的控制	00	R/W
		0x0	CAN_TXD 引脚上的电平由 CAN 控制器控制。这是复位时的值。		
		0x1	可在 CAN_TXD 引脚上检测采样点		
		0x2	CAN_TXD 引脚低电平有效 / 显性		
		0x3	CAN_TXD 引脚高电平有效 / 隐形		
7	RX		监测 CAN_RXD 引脚上的实际值	0	R
		0	CAN 总线为隐性 (CAN_RXD = '1')		
		1	CAN 总线为显性 (CAN_RXD = '0')		
31:8	-		R/W		-

13.6.1.7 CAN 波特率预分频器扩充寄存器

Table 187. CAN 波特率预分频器扩充寄存器 (CANBRPE, 地址 0x4005 0018) 位描述

位	符号	描述	复位值	访问
3:0	BRPE	波特率预分频器扩展 通过可编程的 BRPE 波特率预分频器可扩展至 1023，硬件将这个值认为 BRPE（高位）和 BRP（低位）加 1，允许从 0 到 15 的值范围。	0x0000	R/W
31:4	-	保留	-	-

13.6.2 报文接口寄存器

有两组接口寄存器用于控制 CPU 对报文 RAM 的访问。接口寄存器通过将传送的数据进行缓冲来避免 CPU 对 RAM 和 CAN 报文收发之间的访问冲突。在单次传输中，完整的报文对象 (Section 13.6.2.1) 或部分报文对象可以在报文 RAM 和 IFx 报文缓冲寄存器之间传送。

两组接口寄存器的功能相同（测试模式基本配置除外）。一组寄存器可能用来向报文 RAM 中传送数据，另一组寄存器则可能用来从报文 RAM 中传送数据，允许在进程中互相中断。

每组接口寄存器由报文缓冲寄存器组成，这些报文缓冲寄存器由各自的命令寄存器控制。命令屏蔽寄存器指定数据传送的方向，以及要传送哪一部分报文对象。命令请求寄存器在报文 RAM 中选择一个报文对象作为传输的目的对象或源对象，同时它也启动在命令屏蔽寄存器中指定的操作。

Table 188. 报文接口寄存器

IF1 r 寄存器 名称	IF1 寄存器组	IF2 寄存器名称	IF2 寄存器组
CANIF1_CMDREQ	IF1 命令请求	CANIF2_CMDREQ	IF2 命令请求
CANIF1_CMDMASK	IF1 命令屏蔽	CANIF2_CMDMASK	IF2 命令屏蔽
CANIF1_MASK1	IF1 屏蔽 1	CANIF2_MSK1	IF2 屏蔽 1
CANIF1_MASK2	IF1 屏蔽 2	CANIF2_MSK2	IF2 屏蔽 2
CANIF1_ARB1	IF1 仲裁 1	CANIF2_ARB1	IF2 仲裁 1
CANIF1_ARB2	IF1 仲裁 2	CANIF2_ARB2	IF2 仲裁 2
CANIF1_MCTRL	IF1 报文控制	CANIF2_MCTRL	IF2 报文控制
CANIF1_DA1	IF1 数据 A1	CANIF2_DA1	IF2 数据 A1
CANIF1_DA2	IF1 数据 A2	CANIF2_DA2	IF2 数据 A2
CANIF1_DB1	IF1 数据 B1	CANIF2_DB1	IF2 数据 B1
CANIF1_DB2	IF1 数据 B2	CANIF2_DB2	IF2 数据 B2

在报文 RAM 中有 32 个报文对象。为了避免 CPU 对报文 RAM 和 CAN 报文收发之间的访问造成冲突，CPU 不能直接访问报文对象。报文对象可以通过 IFx 接口寄存器访问。

有关报文处理的详细描述，请看 [Section 13.7.3](#)。

13.6.2.1 报文对象

报文对象包含了来自于接口寄存器中各个位的信息。[Table 189](#) 所示为报文对象的结构图示。报文对象和相关的接口寄存器的各个位显示了位被设置或清零的情况。有关位功能的描述，请参考相关的接口寄存器。

Table 189. 报文 RAM 中报文对象的结构

UMASK	MSK[28:0]	MXTD	MDIR	EOB	NEWDAT	MSGLST	RXIE	TXIE	INTPND
IF1/2_MCTRL	IF1/2_MSK1/2			IF1/2_MCTRL					
RMTEN	TXRQST	MSGVAL	ID[28:0]	XTD	DIR	DLC3	DLC2	DLC1	DLC0
IF1/2_MCTRL		IF1/2_ARB1/2				IF1/2_MCTRL			
DATA0	DATA1	DATA2	DATA3	DATA4	DATA5	DATA6	DATA7		
IF1/2_DA1		IF1/2_DA2		IF1/2_DB1		IF1/2_DB2			

13.6.2.2 CAN 报文接口命令请求寄存器

一旦 CPU 将报文编号写入到命令请求寄存器，报文传送就会启动。随着写操作的进行 BUSY 会自动被设置为 ‘1’，信号 CAN_WAIT_B 被拉低来通知 CPU 传输正在进行。在等待 3 到 6 个 CAN_CLK 周期后，接口寄存器和报文 RAM 之间的传输完成。BUSY 位会被设置回 0，而且信号 CAN_WAIT_B 也被设置回原状态。

Table 190. CAN 报文接口命令请求寄存器 (CANIF1_CMDREQ, 地址 0x4005 0020 和 CANIF2_CMDREQ, 地址 0x4005 0080) 位描述

位	符号	值	描述	复位值	访问
5:0	MN		报文编号 0x01 - 0x20 = 有效的报文编号，报文 RAM 中的报文对象被选中用于传输数据 0x00 = 无效的报文编号，该值被看做 0x20. ^[1] 0x21 - 0x3F = 无效的报文编号。该值被看为 0x01 - 0x1F. ^[1]	0x00	R/W
14:6	-		保留	-	-
15	BUSY		BUSY 标志	0	RO
		0	当对该命令请求寄存器的读 / 写操作完成时，硬件将其设置为 0。		
		1	当写该命令寄存器时，硬件将其设置为 1		
31:16	-	-	保留	-	-

[1] 当向命令请求寄存器写入无效的报文编号时，报文编号将会变为一个有效的值，并传送此报文对象

13. 6. 2. 3 CAN 报文接口命令屏蔽寄存器

IFx 命令屏蔽寄存器的控制位指定传输的方向，并且选择出哪一个 IFx 报文缓冲寄存器用作进行数据传送的源寄存器或目的寄存器。寄存器位的功能由传输的方向决定（读或写），方向由该命令屏蔽寄存器的 WR/RD 位（位 7）进行选择。

WR/RD 的选择情况如下：

1 写传输方向（写报文 RAM）

0 读传输方向（读报文 RAM）

Table 191. CAN 报文接口命令屏蔽寄存器 (CANIF1_CMDMSK, 地址 0x4005 0024 和 CANIF2_CMDMSK, 地址 0x4005 0084) 位描述 – 写方向

位	符号	值	描述	复位值	访问
0	DATA_B		访问数据字节 4-7	0	R/W
		0	数据字节 4-7 不变		
		1	将数据字节 4-7 传送给报文对象		
1	DATA_A		访问数据字节 0-3	0	R/W
		0	数据字节 0-3 不变		
		1	将数据字节 0-3 传送给报文对象		

Table 191. CAN 报文接口命令屏蔽寄存器 (CANIF1_CMDMSK, 地址 0x4005 0024 和 CANIF2_CMDMSK, 地址 0x4005 0084) 位描述 – 写方向 ...continued

位	符号	值	描述	复位值	访问
2	TXRQST		访问传送请求位	0	R/W
		0	无传送请求, IF1/2_MCTRL 中的 TXRQSRT 位不变		
		1	请求传输, 将 IF1/2_MCTRL 的 TXRQST 位置位		
3	CLRINTPND	-	该位在写方向的操作里被忽略	0	R/W
4	CTRL		访问控制位	0	R/W
		0	控制位不改变		
		1	将控制位传送到报文对象		
5	ARB		访问仲裁位	0	R/W
		0	仲裁位不变		
		1	将标识位、DIR、XTD 和 MSGVAL 位传送到报文对象		
6	MASK		访问屏蔽位	0	R/W
		0	屏蔽位不变		
		1	将 MASK + MDIR + MXTD 标识符传送到报文对象		
7	WR/RD	1	写传输	0	R/W
			将所选报文缓冲区寄存器的数据传送到命令请求寄存器 CANIFn_CMDREQ 所寻址的报文对象		
31:8	-	-	保留	0	-

Table 192. CAN 报文接口命令屏蔽寄存器 (CANIF1_CMDMSK, 地址 0x4005 0024 和 CANIF2_CMDMSK, 地址 0x4005 0084) 位描述 – 读方向

位	符号	值	描述	复位值	访问
0	DATA_B		访问数据字节 4-7	0	R/W
		0	数据字节 4-7 不变		
		1	传送数据字节 4-7 到 IFx 报文缓冲寄存器		
1	DATA_A		访问数据字节 0-3	0	R/W
		0	数据字节 0-3 不变		
		1	传送数据字节 0-3 到 IFx 报文缓冲区		

Table 192. CAN 报文接口命令屏蔽寄存器 (CANIF1_CMDMSK, 地址 0x4005 0024 和 CANIF2_CMDMSK, 地址 0x4005 0084) 位描述 – 读方向

位	符号	值	描述	复位值	访问
2	NEWDAT		访问新数据位	0	R/W
		0	NEWDAT 位保持不变		
		1	清除报文对象里的 NEWDAT 位		
3	CLRINTPND		清除中断挂起位	0	R/W
		0	INTPND 位保持不变		
		1	清除报文对象里的 INTPND		
4	CTRL		访问控制位	0	R/W
		0	控制位保持不变		
		1	将控制位传送到 IFx 报文缓冲区		
5	ARB		访问仲裁位	0	R/W
		0	仲裁位保持不变		
		1	将标识符、DIR、XTD 和 MSGVAL 位传送到 IFx 报文缓冲寄存器		
6	MASK		访问屏蔽为	0	R/W
		0	屏蔽位保持不变		
		1	将 MASK + MDIR + MXTD 标识符传送到 IFx 报文缓冲寄存器		
7	WR/RD	0	读传输	0	R/W
			将命令请求寄存器所寻址的报文对象的数据送到所选报文缓冲寄存器 CANIFn_CMDREQ		
31:8	-	-	保留	0	-

13.6.2.4 IF1 and IF2 报文缓冲寄存器

报文缓冲区寄存器的位反映报文 RAM 中的报文对象。 .

13.6.2.4.1 CAN 报文接口命令屏蔽 1 寄存器

Table 193. CAN 报文接口命令屏蔽 1 寄存器 (CANIF1_MSK1, 地址 0x4005 0028 和 CANIF2_MASK1, 地址 0x4005 0088) 位描述

位	符号	值	描述	复位值	访问
15:0	MSK[15:0]		标识符屏蔽	0xFFFF	R/W
		0	报文标识符的相应位不能禁止接收过滤里的匹配操作。		
		1	相应的标识符位应用于接收过滤操作。		
31:16	-	-	保留	0	-

13.6.2.4.2 CAN 报文接口命令屏蔽 2 寄存器

Table 194. CAN 报文接口命令屏蔽 2 寄存器 (CANIF1_MSK2, 地址 0x4005 002C 和 CANIF2_MASK2, 地址 0x4005 008C) 位描述

位	符号	值	描述	复位值	访问
12:0	MSK[28:16]		标识符屏蔽	0xFFF	R/W
		0	报文标识符的相应位不能禁止接收过滤里的匹配操作。		
		1	相应的标识符位应用于接收过滤操作		
13	-		保留	1	-
14	MDIR		屏蔽报文方向	1	R/W
		0	报文方向位 (DIR) 不会影响接收过滤操作		
		1	报文方向位 (DIR) 用来接收过滤操作		
15	MXTD		屏蔽扩展标识符	1	R/W
		0	扩展的标识符位 (XTD) 不会影响接收过滤操作		
		1	扩展的标识符位 (XTD) 用来接收过滤操作		
31:16	-	-	保留	0	-

13.6.2.4.3 CAN 报文接口命令仲裁 1 寄存器

Table 195. CAN 报文接口命令仲裁 1 寄存器 (CANIF1_ARB1, 地址 0x4005 0030 和 CANIF2_ARB1, 地址 0x4005 0090) 位描述

位	符号	描述	复位值	访问
15:0	ID[15:0]	报文标识符	0x00	R/W
		29 位标识符 (扩展帧)		
		11 位标识符 (标准帧)		
31:16	-	保留	0	-

13.6.2.4.4 CAN 报文接口命令仲裁2寄存器

Table 196. CAN 报文接口命令仲裁2寄存器(CANIF1_ARB2, 地址0x4005 0034 和CANIF2_ARB2, 地址0x4005 0094) 位描述

位	符号	值	描述	复位值	访问
12:0	ID[28:16]		报文标识符 29 位标识符 (扩展帧) 11 位标识符 (标准帧)	0x00	R/W
13	DIR		报文方向	0x00	R/W
		0	方向 = 接收 在 TXRQST 里, 把带该报文对象标识符的远程帧发送出去。在接收到带匹配标识符的数据帧时, 将此报文存储在该报文对象里。		
		1	方向 = 发送 在 TXRQST 里, 相应的报文对象作为数据帧发送出去。在接收到带匹配标识符的远程帧时, 该报文对象的 TXRQST 位被置位 (如果 RMTEN=1)		
14	XTD		扩展标识符	0x00	R/W
		0	11 位标准标识符用于该报文对象		
		1	29 位扩展标识符用于该报文对象		
15	MSGVAL		报文有效 注意: CPU 在复位 CAN 控制寄存器的 INIT 位之前的初始化进程中, 必须复位所有未被使用的报文对象的 MSGVAL 位。在修改标识符 ID28:0, 控制位 XTD、DIR 或数据长度代码 DLC 3:0 之前, 或不再要求使用报文对象时, 也必须要将该位复位	0	R/W
		0	报文处理程序忽略报文对象		
		1	报文对象被配置, 且由报文处理程序进行识处理		
31:16	-	-	保留	0	-

13.6.2.4.5 CAN 报文接口报文控制寄存器

Table 197. CAN 报文接口报文控制寄存器 (CANIF1_MCTRL, 地址 0x4005 0038 和 CANIF2_MCTRL, 地址 0x4005 0098) 位描述

位	符号	值	描述	复位值	访问
3:0	DLC[3:0]		数据长度代码 注意: 报文对象的数据长度代码的定义必须要与所有相应的对象相同, 在其它的节点上具有相同的标识符。当报文处理程序存放数据帧时, 它将会把 DLC 写入接收到的报文所给定的值。 0000 - 1000 = 数据帧具有 0 – 8 个数据字节 1001 - 1111 = 数据帧具有 8 个数据字节	0000	R/W
6:4	-		保留	-	-
7	EOB		缓冲区结束	0	R/W
		0	报文对象属于 FIFO 缓冲区, 且不会是 FIFO 缓冲区的最后一个报文对象		
		1	FIFO 缓冲区的单个报文对象或最后一个报文对象		

Table 197. CAN 报文接口报文控制寄存器 (CANIF1_MCTRL, 地址 0x4005 0038 和 CANIF2_MCTRL, 地址 0x4005 0098) 位描述 ...continued

位	符号	值	描述	复位值	访问
8	TXRQST		发送请求	0	R/W
		0	该报文对象不是在等待被发送		
		1	请求发送该报文对象，但还没有完成发送		
9	RMTEN		远程允许	0	R/W
		0	在接收到远程帧时，不会更改 TXRQST		
		1	在接收到远程帧时，置位 TXRQST		
10	RXIE		接收中断允许	0	R/W
		0	在成功接收帧后，INTPND 不改变		
		1	在成功接收帧后，INTPND 将置位		
11	TXIE		发送中断允许	0	R/W
		0	在成功发送帧后，INTPND 不改变		
		1	在成功发送帧后，INTPND 将置位		
12	UMASK		使用接收屏蔽	0	R/W
			注意： 如果用户将 UMASK 设置为 1，那么就要在将 MAGVAL 设为 1 之前的初始化进程中，要对报文对象的屏蔽位进行编写。		
		0	忽略屏蔽		
		1	使用屏蔽 (MSK[28:0], MXTD, 和 MDIR) 进行接收过滤		
13	INTPND		中断挂起	0	R/W
		0	该报文对象不是中断源		
		1	该报文对象是中断源。如果不存在更高优先级的中断源，中断寄存器的中断标识符将会指向该报文对象。		
14	MSGLST		报文丢失 (只对接收方向的报文对象有效)	0	R/W
		0	无报文丢失，因为 CPU 最后才将该位复位		
		1	当 NEWDAT 仍然保持置位时，报文处理程序将一个新报文存储进该对象，此时 CPU 已经将报文丢失		
15	NEWDAT		新数据	0	R/W
		0	报文处理程序没有把新数据写入到该报文对象的数据部分，因为该标志由 CPU 最后清零。		
		1	报文处理程序或 CPU 已将新数据写入到该报文对象的数据部分		
31:16	-	-	保留	0	-

13.6.2.4.6 CAN 报文接口数据 A1 寄存器

在 CAN 数据帧中，DATA0 首先被发送或者接收，而 DATA7（在 CAN_IF1B2 和 CAN_IF2B2）则是最后被发送或接收的字节。在 CAN 的串行位流中，首先发送的是每个字节的最高位。

注意： 在接收进程中，字节 DATA0 是最先被移送到 CAN 内核的移位寄存器的数据字节，字节 DATA7 则是最后发送。当报文处理程序存放数据帧时，它将会把所有八个数据字节写入到报文对象中。如果数据长度代码少于 8，则剩余在报文对象的字节将会被非指定的值覆盖。

Table 198. CAN 报文接口数据 A1 寄存器 (CANIF1_DA1, 地址 0x4005 003C 和 CANIF2_DA1, 地址 0x4005 009C) 位描述

位	符号	描述	复位值	访问
7:0	DATA0	数据字节 0	0x00	R/W
15:8	DATA1	数据字节 1	0x00	R/W
31:16	-	保留	-	-

13. 6. 2. 4. 7 CAN 报文接口数据 A2 寄存器

Table 199. CAN 报文接口数据 A2 寄存器 (CANIF1_DA2, 地址 0x4005 0040 和 CANIF2_DA2, 地址 0x4005 00A0) 位描述

位	符号	描述	复位值	访问
7:0	DATA2	数据字节 2	0x00	R/W
15:8	DATA3	数据字节 3	0x00	R/W
31:16	-	保留	-	-

13. 6. 2. 4. 8 CAN 报文接口数据 B1 寄存器

Table 200. CAN 报文接口数据 B1 寄存器 (CANIF1_DB1, 地址 0x4005 0044 和 CANIF2_DB1, 地址 0x4005 00A4) 位描述

位	符号	描述	复位值	访问
7:0	DATA4	数据字节 4	0x00	R/W
15:8	DATA5	数据字节 5	0x00	R/W
31:16	-	保留	-	-

13. 6. 2. 4. 9 CAN 报文接口数据 B2 寄存器

Table 201. CAN 报文接口数据 B2 寄存器 (CANIF1_DB2, 地址 0x4005 0048 和 CANIF2_DB2, 地址 0x4005 00A8) 位描述

位	符号	描述	复位值	访问
7:0	DATA6	数据字节 6	0x00	R/W
15:8	DATA7	数据字节 7	0x00	R/W
31:16	-	保留	-	-

13. 6. 3 报文处理程序寄存器

所有的报文处理程序寄存器都是只读寄存器。它们的内容（每一个报文对象的 TXRQST, NEWDAT, INTPND, 和 MSGVAL 位和中断标识符）是由报文处理程序 FSM 提供的状态信息。

13.6.3.1 CAN 发送请求 1 寄存器

该寄存器包含报文对象（1~16）的 TXRQST 位。通过读出 TXRQST 位，CPU 可以查看出哪一个报文对象的发送请求被挂起。在接收远程帧后，或成功发送后，特定报文对象的 TXRQST 位可由 CPU 通过 IFx 报文接口寄存器来进行置位 / 复位，或可由报文处理程序置位 / 复位。

Table 202. CAN 发送请求 1 寄存器 (CANTXREQ1, 地址 0x4005 0100) 位描述

位	符号	描述	复位值	访问
15:0	TXRQST[16:1]	报文对象 16 ~ 1 的发送请求 0 = 该报文对象不是在等待着被发送 1 = 请求发送该报文对象，但发送未完成	0x00	R
31:16	-	保留	-	-

13.6.3.2 CAN 发送请求 2 寄存器

该寄存器包含报文对象（32~17）的 TXRQST 位。通过读出 TXRQST 位，CPU 可以出哪一个报文对象的发送请求被挂起。在接收远程帧后，或成功发送后，特定报文对象的 TXRQST 位可由 CPU 通过 IFx 报文接口寄存器来进行设置 / 复位，或可由报文处理程序进行设置 / 复位。

Table 203. CAN CAN 发送请求 2 寄存器 (CANTXREQ2, 地址 0x4005 0104) 位描述

位	符号	描述	复位值	访问
15:0	TXRQST[32:17]	报文对象 32~17 的发送请求 0 = 该报文对象不是在等着被发送 1 = 请求发送该报文对象，但发送未完成	0x00	R
31:16	-	保留	-	-

13.6.3.3 CAN 新数据 1 寄存器

该寄存器包含报文对象（16 ~ 1）。通过读出 NEWDAT 位，CPU 可以查看出哪一个报文对象的数据部分被更新。在接收完数据帧，或发送成功后，特定报文对象的位可由 CPU 通过 IFx 报文接口寄存器来进行置位 / 复位，或可由报文处理程序进行置位 / 复位。

Table 204. CAN 新数据 1 寄存器 (CANND1, 地址 0x4005 0120) 位描述

位	符号	描述	复位值	访问
15:0	NEWDAT[16:1]	报文对象 16~1 的新数据位。 0 = 报文处理程序没有把新数据写入到该报文对象的数据部分，因为 CPU 之前已经把该数据为清零。 1 = 报文处理程序或CPU已经把新数据写入该报文对象的数据部分。	0x00	R
31:16	-	保留	-	-

13.6.3.4 CAN 新数据 2 寄存器

该寄存器包含报文对象（32~17）的 NEWDAT 位。通过读 NEWDAT 位，CPU 可以查看出哪一个报文对象的数据部分被更新。在接收完数据帧或者发送成功后，特定报文对象的 NEWDAT 位可由 CPU 通过 Ifx 报文接口寄存器或者由报文处理程序进行置位 / 复位。

Table 205. CAN 新数据 2 寄存器 (CANND2, 地址 0x4005 0124) 位描述

位	符号	描述	复位值	访问
15:0	NEWDAT[32:17]	报文对像 32~17 的新数据位。 0 = 报文处理程序没有把新数据写入到该报文对象的数据部分，因为 CPU 之前已经把该数据为清零。 1 = 报文处理程序或 CPU 已经把新数据写入该报文 对象的数据部分。	0x00	R
31:16	-	保留	-	-

13.6.3.5 CAN 中断挂起 1 寄存器

该寄存器包含报文对象（16~1）的 INTPND 位。通过读取 INTPND 位，CPU 可以查看数哪一个报文对象的中断被挂起。在接收完帧或者成功发送完帧后，特定报文对象的 INTPND 位可由 CPU 通过 Ifx 报文接口寄存器进行置位 / 复位，或者由报文处理程序进行置位 / 复位。这将会影响中断寄存器中的 INTPND 位的值。

Table 206. CAN 中断挂起 1 寄存器 (CANIR1, 地址 0x4005 0140) 位描述

位	符号	描述	复位值	访问
15:0	INTPND[16:1]	报文对象 16~1 的中断挂起位 0 = 报文处理程序忽略该报文对象 1 = 该报文对象是中断源	0x00	R
31:16	-	保留	-	-

13.6.3.6 CAN 中断挂起 2 寄存器

该寄存器包含报文对象的 32~17 的 INTPND 位。通过读出 INTPND 位，CPU 可以查看数哪一个报文对象的中断被挂起。在接收完帧，或者成功发送完帧后，特定报文对象的 INTPND 位可由 CPU 通过 Ifx 报文接口寄存器进行置位 / 复位，或者由报文处理程序进行置位 / 复位。这将会影响中断寄存器 INTPND 位的值。

Table 207. CAN 中断挂起 2 寄存器 (CANIR2, 地址 0x4005 0144) 位描述

位	符号	描述	复位值	访问
15:0	INTPND[32:17]	报文对象 32~17 的中断挂起 0= 报文处理程序忽略该报文对象。 1 = 报文对象是中断源	0x00	R
31:16	-	保留	-	-

13.6.3.7 CAN 报文有效 1 寄存器

该寄存器包含报文对象（16~1）的 MSGVAL 位。通过读 MSGVAL 位，CPU 可以知道哪一个报文对象有效。特定报文对象的 MSGVAL 位可以由 CPU 通过 Ifx 报文接口寄存器进行置位 / 复位。

Table 208. CAN 报文有效 1 寄存器 (CANMSGV1, 地址 0x4005 0160) 位描述

位	符号	描述	复位值	访问
15:0	MSGVAL[16:1]	报文对象 16~1 的报文有效位。 0 = 报文处理程序忽略该报文对象。 1 = 该报文对象被配置，且能被报文处理程序识别。	0x00	R
31:16	-	保留	-	-

13.6.3.8 CAN 报文有效 2 寄存器

该寄存器包含报文对象（32~17）的 MSGVAL 位。通过读 MSGVAL 位，CPU 可以查看出哪一个报文对象有效。特定报文对象的 MSGVAL 位可以由 CPU 通过 Ifx 报文接口寄存器进行置位 / 复位。

Table 209. CAN 报文有效 2 寄存器 (CANMSGV2, 地址 0x4005 0164) 位描述

位	符号	描述	访问	复位值
15:0	MSGVAL[32:17]	报文对象 32~17 的报文有效位。 0 = 报文处理程序忽略该报文对象。 1 = 报文对象被配置而且可以被报文处理程序识别。	R	0x00
31:16	-	保留	-	-

13.6.4 CAN 定时寄存器

13.6.4.1 CAN 时钟分频寄存器

该寄存器决定 CAN 时钟信号。用该寄存器的值对外设时钟 PCLK 进行分频，可以得到 CAN_CLK。

Table 210. CAN 时钟分频寄存器 (CANCLKDIV, 地址 0x4005 0180) 位描述

位	符号	描述	复位值	访问
3:0	CLKDIVVAL	时钟分频器值。 CAN_CLK = PCLK/(CLKDIVVAL +1) 0000: CAN_CLK = PCLK divided by 1 0001: CAN_CLK = PCLK divided by 2 0010: CAN_CLK = PCLK divided by 3 0010: CAN_CLK = PCLK divided by 4 ... 1111: CAN_CLK = PCLK divided by 16	0000	R/W
31:4	-	保留	-	-

13.7 功能描述

13.7.1 复位后 C_CAN 控制器状态

经过硬件复位后，寄存器内的值如 [Table 180](#)。另外，总线关闭状态被复位，输出 CAN_TXD 被设置为隐形（高电平）。CAN 控制寄存器的值 0x0001（INIT= ‘1’）允许软件初始化。CAN 控制器不会与 CAN 总线进行通信，直至 CPU 将 INIT 复位为 ‘0’ 才能通信。

存放在报文 RAM 中的数据不受硬件复位的影响。在上电后，报文 RAM 中的内容未定义。

13.7.2 C_CAN 操作模式

13.7.2.1 软件初始化

软件初始化通过设置 CAN 控制寄存器的 INIT 位来启动，也可通过软件或硬件复位，或进入总线关闭状态来启动。

在软件初始化（INIT 位被置位）期间，要满足下列条件：

- 所有 CAN 总线上的报文传输停止。
- CAN 输出 CAN_TXD 的状态为隐性（高电平）。
- EML 计数器没改变。
- 配置寄存器没改变。
- 如果 CAN 控制寄存器的 CCE 位也被置位，则允许访问位定时寄存器和 BRP 扩展寄存器。

为了初始化 CAN 控制器，软件必须设置位定时寄存器和每一个报文对象。如果不需要报文对象，只需将 MSGVAL 位设置为无效。否则，必须初始化整个报文对象。

复位 INIT 位来完成软件的初始化。在位流处理器 BSP 参与总线活动和启动报文传输之前，可通过等待 11 个连续隐性位出现的序列（总线空闲）的出现，将其与 CAN 总线上的数据传输同步。

注意：报文对象的初始化独立于 INIT，也可以在空闲时完成，但是在 BSP 启动报文传输之前，在执行软件初始化的过程中，应将报文对象配置到特定的标识符中或设置为无效。为了在正常操作的过程中更改报文对象的配置，必须将 MSGVAL 位设置为无效来启动 CPU。当配置完成时，再次将 MSGVAL 设置为有效。

13.7.2.2 CAN 报文传输

一旦 CAN 控制器启动且 INIT 被复位为 0，CAN 内核会把自身同步到 CAN 总线，并启动报文传输。

如果接收到的报文传递了报文处理程序的接收过滤，则它们会被存放在各自相关的报文对象里。整个报文，包括所有仲裁位、DLC 和 8 个数据字节，都存放在报文对象中。如果使用了标识符屏蔽，则被标识为“无关”的仲裁位有可能在报文对象中被写覆盖。

CPU 通过接口寄存器可随时读或写每一个报文。在并发存取的情况下，报文处理程序可以保证数据的一致性。

要被发送的报文由 CPU 进行更新。如果报文有一个永久的报文对象存在（仲裁和控制位在配置过程中设置），则只更新数据字节，然后 TXRQUT 位和 NEWDAT 位被置位来启动传送。如果向同一个报文对象分配几个发送报文（当报文对象的编号不够用时），则在请求发送该报文之前就要把整个报文对象配置好。

任何编号的报文对象的传送可以在同时被请求，其后依照它们的内部优先级来进行发送。报文可以随时被更新或被设为无效，即使它们请求的传送仍在挂起。在启动挂起的传送之前，如果更新了报文，则会丢弃旧的数据。

根据报文对象的配置，在接收到具有匹配标识符的远程帧时，可以自动请求传送报文。

13.7.2.3 禁止的自动重发（DAR）

依照 CAN 规范（ISO11898，6.3.3 恢复管理），CAN 控制器为在传送过程中丢失仲裁的帧或被错误干扰的帧提供了几种自动重发的方法。在传送成功完成之前，用户不可以使用帧发送服务。在默认情况下，当出现丢失仲裁或错误时，自动重发是允许的。可以将其禁止，以便 CAN 控制器可以在定时触发的 CAN（TTCAN，见 ISO11898-1）环境内工作。

将 CAN 控制寄存器的 DAR 位编写为 1，即可将禁止的自动重发模式变为允许状态。在该操作模式下，编程者必须要考虑报文缓冲区里控制寄存器的 TXRQST 和 NEWDAT 位的不同行为：

- 传送启动时，相关报文缓冲区的 TXRQST 位复位，而 NEWDAT 位保持置位。
- 当传送成功完成时，位 NEWDAT 复位。
- 当传送失败时（丢失仲裁或错误），NEWDAT 位保持置位。为重启传送，CPU 必须将 TXRQST 重置为 1。

13.7.2.4 测试模块

在 CAN 控制寄存器中将 TEST 位设置为 1，即可进入测试模式。在测试模式下，可对测试寄存器的 TX1、TX0、LBACK、SILENT 和 BASIC 位进行写操作。位 RX 监控着引脚 RDO、1 的状态，因此它是只读位。所有测试寄存器的功能在位 TEST 被复位为 0 时禁止。

13.7.2.4.1 安静模式

将测试寄存器的 SILENT 位设为 1，即可令 CAN 内核处于安静模式。

在安静模式下，CAN 控制器可以接收有效的数据帧和有效的远程帧。但是它在 CAN 总线上只发送隐性位，并不能启动传送。如果要求 CAN 内核发送显性位（ACK 位，过载标志、有效错误标志），则内部重发该位，以便 CAN 内核监控到该显性位，虽然 CAN 总线可能仍处于隐性状态。通过传送显性位（应答位、错误帧），安静模式可在不影响总线的情况下分析 CAN 总线的流量。

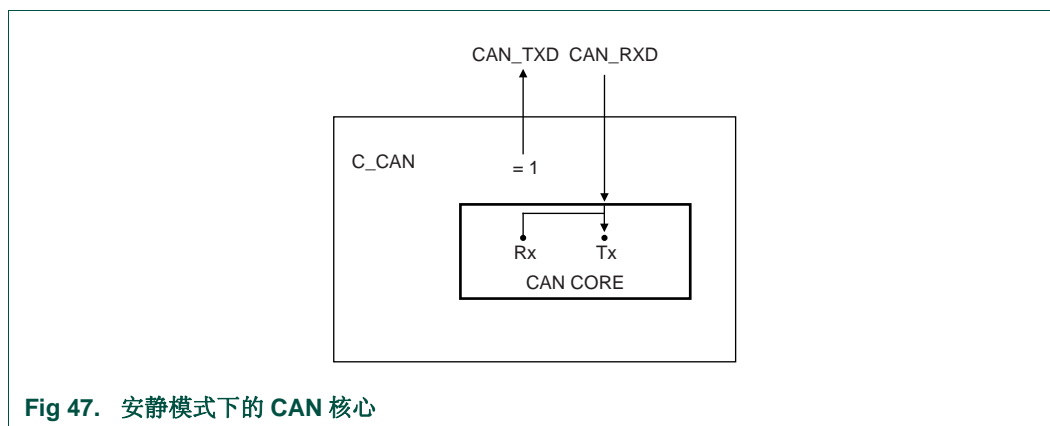


Fig 47. 安静模式下的 CAN 核心

13.7.2.4.2 回环模式

将测试寄存器的 LBACK 位编写为 1，即可令 CAN 内核处于回环模式。在回环模式下，CAN 内核将自身发送的报文看作是接收到的报文，并将它们（如果它们传递接收过滤）存放到接收缓冲区中。

该模式用于自测试功能。为独立于外部仿真，CAN 内核在回环模式下会忽略应答错误（数据 / 远程帧应答间隙里采样到的隐性位）。在该模式中，CAN 内核执行从 CAN_TXD 输出到 CAN_RXD 输入的反馈。CAN_RXD 输入引脚的真实值被 CAN 内核忽略。可以在 CAN_TXD 引脚上监控被发送的报文。

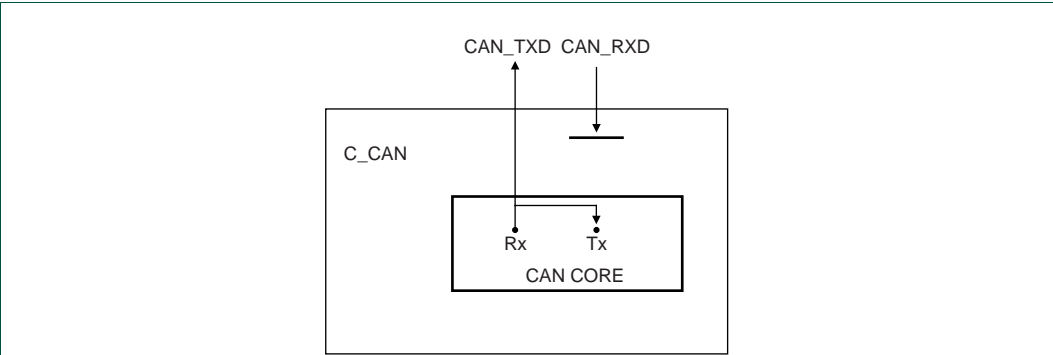


Fig 48. 在还回模式下的 CAN 核

13.7.2.4.3 回环模式和安静模式的结合

将 LBACK 和 SLIENT 位同时编程为 1，可以整合回环模式和安静模式。该模式用于 “Hot Selftest”，表示在不影响正在运行的 CAN 系统（接入 CAN_TXD 和 CAN_RXD 引脚）前提下，可以测试 C_CAN。在该模式下，CAN_RXD 不接入 CAN 内核，且 CAN_TXD 引脚保持为隐性。

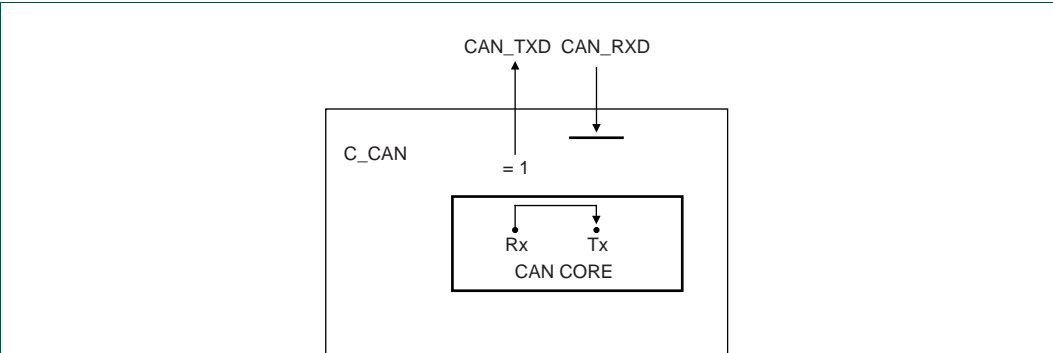


Fig 49. 回环模式和安静模式结合下的 CAN 核

13.7.2.4.4 基本模式

将测试寄存器的 BASIC 位编程为 1，即可令 CAN 内核进入基本模式。在该模式下，CAN 控制器运行时不需要使用报文 RAM。

IF1 寄存器用作发送缓冲区。将 IF1 命令请求寄存器的 BUSY 位写 ‘1’，可请求发送 IF1 寄存器的内容。当 BUSY 置位时，IF1 寄存器被锁定。BUSY 位指明了传送正在挂起。

一旦 CAN 总线空闲，IF1 寄存器被载入到 CAN 内核的移位寄存器中，并启动传送。当传送结束时，BUSY 位复位，且锁定的 IF1 寄存器被释放。

当 IF1 寄存器被锁定时，将 IF1 命令请求寄存器的 BUSY 位复位，可在任何时候中止挂起的传送。如果 CPU 已经复位 BUSY 位，则在丢失仲裁或错误的情况下，禁止可能进行的重新发送操作。

IF2 寄存器用作接收缓冲区。在接收到报文之后，移位寄存器的内容被存放到 IF2 寄存器中，而不用进行任何接收过滤。

另外，可以在报文传输的过程中监控移位寄存器的实际内容。每次向 IF2 命令请求寄存器的 BUSY 位写入 ‘1’，都会启动读报文对象操作，移位寄存器的内容存放在 IF2 寄存器中。

在基本模式下，与控制 and 状态位相关的所有报文对象，以及 IFx 命令屏蔽寄存器控制位的报文对象，对它们的评估被关闭。命令请求寄存器的报文编号不能被评估。IF2 报文控制寄存器的 NWEDAT 和 MSGST 位保留着它们的功能，DLC3 ~ 0 将会显示所接收到的 DLC，其它的控制位被读为 ‘0’。

在基本模式下，就绪输出 CAN_WAIT_B 被禁止（总为 ‘1’）。

13.7.2.4.5 软件控制 CAN_TXD 引脚

在 CAN 发送 CAN_TXD 引脚上，共有 4 种输出功能：

1. 串行数据输出（默认）。
2. 驱动 CAN 采样点信号来监控 CAN 控制器的时序。
3. 驱动隐性常量值。
4. 驱动显性常量值。

最后的二个功能，结合可读的 CAN 接收引脚 CAN_RXD，可用于检查 CAN 总线的物理层。

将测试寄存器位 TX1 和 TX0 按 [Section 13.6.1.6](#) 来设置，就可以选择引脚 CAN_TXD 的输出模式。

注意：CAN_TXD 引脚的三个测试功能涉及所有的 CAN 协议函数。当 CAN 报文传输或选择了从回环模式、安静模式或基本模式中的任何一种模式时，必须令 CAN_TXD 引脚处于其默认功能。

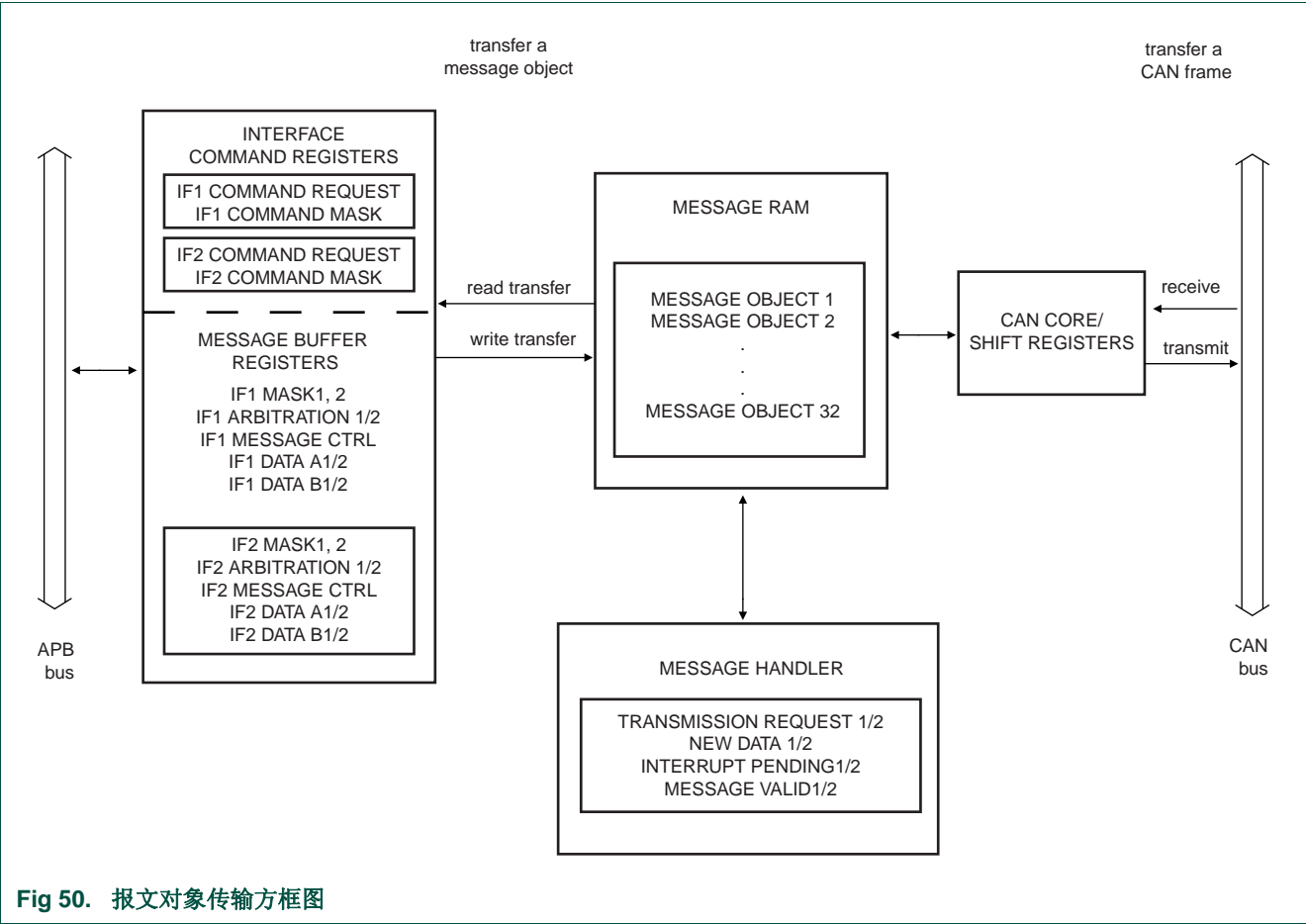
13.7.3 CAN 报文处理程序

报文处理程序控制 CAN 内核 Rx/Tx 移位寄存器、报文 RAM 和 IFx 寄存器之间的数据传输，参考 [Figure 50](#)。

报文处理程序包括以下功能：

- Ifx 寄存器和报文 RAM 之间的数据传输；

- 将移位寄存器的数据传送到报文 RAM 中；
- 将报文 RAM 中的数据传送到移位寄存器中；
- 将移位寄存器的数据传输到接收过滤单元中；
- 扫描报文 RAM，以查找符合匹配的报文对象；
- 处理 TXRQST 标志；
- 处理中断。



13. 7. 3. 1 报文对象的管理

复位芯片时，不会影响到报文 RAM 中报文对象的配置（MSGVAL、NEWDAT、INTPND 和 TXRQST 位除外）。所有的报文对象必须由 CPU 启动，或必须将它们设为无效（MSGVAL= ‘0’ ）。位时序必须在 CPU 清除 CAN 控制寄存器的 INIT 位之前配置好。

将二个接口寄存器中的一个寄存器的屏蔽、仲裁、控制和数据域设为所要求的值，即可完成报文对象的配置。写相应的 IFx 命令请求寄存器，可将 IFx 报文缓冲区寄存器载入到在报文 RAM 中寻址的报文对象中。

当 CAN 控制寄存器的 INIT 位清零时，CAN 内核的 CAN 协议控制器状态机和报文处理程序状态机控制着 CAN 控制器的内部数据流。传递接收过滤的被接收到的报文被存放到报文 RAM 中，带有挂起发送请求的报文被载入到 CAN 内核的移位寄存器中，并通过 CAN 总线来发送。

CPU 读取接收到的报文，并更新通过 IFx 接口寄存器传送的报文。根据配置情况，CPU 会被某些 CAN 报文和 CAN 错误事件中断。

13.7.3.2 IFx 寄存器和报文 RAM 之间的数据传输

当 CPU 启动 IFx 位寄存器和报文 RAM 之间的数据传输时，报文处理程序将相关命令寄存器的 BUSY 位设为 ‘1’。在完成传输后，BUSY 位被设为 ‘0’。

命令屏蔽寄存器指定是传输一个完整的报文对象还是只传输报文对象的一部分。由于报文 RAM 的结构原因，不可能写一个报文对象的单个位 / 字节。软件必须将完整的报文对象写入到报文 RAM 中。因此，将 IFx 寄存器的数据传送到报文 RAM 要求一个读 - 改 - 写周期：

1. 使用命令屏蔽寄存器来读取报文对象的某些部分，而报文对象从报文 RAM 中读取后不改变。
 - 在部分读取报文对象后，不是在命令屏蔽寄存器中选择出的报文缓冲寄存器将会保持不变。
2. 将报文缓冲寄存器的完整内容写入到报文对象中。
 - 在部分写报文对象后，不是在命令屏蔽寄存器中选择出的报文缓冲寄存器将会设为所选的报文对象的实际内容。

13.7.3.3 CAN 内核的移位寄存器和报文缓冲区之间的报文传送

如果 CAN 内核单元的移位寄存器已准备好进行加载，并且，如果 IFx 寄存器和报文 RAM 之间没有数据要传输，则要对报文有效寄存器的 MSGVAL 位和传送请求寄存器的 TXRQST 位进行评估。挂起传送请求的、具有最高优先级的有效报文对象会被报文处理程序载入到移位寄存器中，并启动传送，报文对象的 NEWDAT 位复位。

在成功完成传送后，且如果没有把新数据写入到报文对象 (NEWDAT = ‘0’)，由于启动了传送，所以 TXRQST 位将会复位。如果 TXIE 置位，INTPND 将会在成功完成传输后置位。如果在传送的过程中，CAN 控制器丢失了仲裁或发生错误，一旦 CAN 总线再次空闲，将会重发报文。如果此时请求了要发送一个优先级更高的报文，则会按报文优先级的排序来发送报文。

13.7.3.4 接收到报文的接受过滤

当正在进入报文的仲裁和控制域（标识符 + IDE + RTR + DLC）已完全移入到 CAN 内核的 Rx/Tx 移位寄存器中，报文处理程序状态会启动报文 RAM 的扫描操作，以查找匹配有效的的报文对象。

为了扫描报文 RAM 以查找到匹配的报文对象，要将 CAN 内核移位寄存器的仲裁载入到接收过滤单元中。然后报文对象 1 的仲裁和屏蔽域（包括 MSGVAL、UMASK、NEWDAT 和 EOB）被载入到接收过滤单元中，并将它们与移位寄存器的仲裁域进行比较。对每一个随后的报文对象都进行重复的操作，直至查找到匹配的报文对象，或直至到达报文 RAM 的尽头。

如果存在匹配的报文对象，则扫描停止，报文处理程序状态机会根据所接收到的帧（数据帧或远程帧）类型作出处理。

13.7.3.4.1 接收数据帧

报文处理程序状态机将 CAN 内核移位寄存器上的报文存放到报文 RAM 中相关的报文对象。数据字节、所有仲裁位和数据长度代码被存放在相关的报文对象里。执行这样的操作是为了令数据字节与标识符相连，即使使用了仲裁屏蔽寄存器。

置位 NEWDAT 位来表示已接收到新数据（CPU 没有发现）。当 CPU/ 软件读取报文对象时应要复位 NEWDAT。如果在接收的时候，NEWDAT 位已被置位，则置位 MSGLST 位来表示之前的数据（假定 CPU 没有发现）丢失。如果 RxIE 位被置位，INTPND 位也置位，会令中断寄存器指向该报文对象。

在刚接收完请求的数据帧时，复位该报文对象的 TXRQST 位来阻止传送远程帧。

13.7.3.4.2 接收远程帧

当接收远程帧时，必须要考虑匹配报文对象的三种不同配置：

1. DIR = ‘1’（方向 = 发送），RMTEN = ‘1’，UMASK = ‘1’ 或 ‘0’ 当接收到匹配远程帧时，该报文对象的 TXRQST 位被置位。其余保持不变。
2. DIR = ‘1’（方向 = 发送），RMTEN = ‘0’，UMASK = ‘0’
当接收到匹配远程帧时，该报文对象的 TXRQST 位保持不变，远程帧被忽略。
3. DIR = ‘1’（方向 = 发送），RMTEN = ‘0’，UMASK = ‘1’

在接收到匹配远程帧时，该报文对象的 TXRQST 位被复位。移位寄存器的仲裁和控制域（标识符 + IDE + RTR + DLC）被存放到报文 RAM 的报文对象中，并设置该报文对象的 NEWDAT 位。报文对象的数据域保持不变，处理远程帧的方式与处理接收到的数据帧相似。

13.7.3.5 接收 / 发送优先级

报文对象的接收 / 发送优先级与报文编号相关。报文对象 1 具有最高优先级，而报文对象 32 具有最低的优先级。如果超过一个发送请求被挂起，要按相应的报文对象的优先级来进行处理。

13.7.3.6 传输对象的配置

[Table 211](#) 所示为软件应如何初始化一个发送对象（也可参考 [Table 189](#)）：

Table 211. 发送对象的初始化

MSGVAL	Arbitration bits	Data bits	Mask bits	EOB	DIR	NEWDAT
1	application dependent	application dependent	application dependent	1	1	0
MSGLST	RXIE	TXIE	INTPND	RMTEN	TXRQST	
0	0	application dependent	0	application dependent	0	

仲裁寄存器（ID28:0 和 XTD 位）由应用程序给定。它们定义传送出去的报文的标识符和类型。如果使用 11 位标识符（“标准帧”），可将其编写为 ID28。在这种情况下，可以忽略 ID18、ID17~ID0。

如果 TXIE 位置位，则在成功传送完报文对象之后，置位 INTPND 位。

如果 RMTEN 位置位，匹配接收的远程帧将会令 TXRQST 位置位，且数据帧会自动应答远程帧。

数据寄存器（DLC3:0，数据 0:7）由应用程序给定。在数据有效之前，不可以将 TXRQST 和 TMTEN 置位。

可以使用屏蔽寄存器（Msk28-0、UMASK、MXTD 和 MDIR 位）（UMASK=’1’），以允许用带有相似标识符的远程帧组来设置 TXRQST 位。详情请参考“接收远程帧” [Section 13.7.3.4.2](#)。

13.7.3.7 更新发送对象

CPU 随时可通过 IFx 接口寄存器来更新发送对象的数据字节。在更新之前，不必将 MSGVAL 或 TXRQST 复位。

即使只更新一部分数据字节，在将该寄存器的内容传送到报文对象之前，要令所有相关 IFx 数据 A 寄存器或 IFx 数据 B 寄存器的四个字节有效。在 CPU 写入新数据字节之前，CPU 要将所有四个字节写入到 IFx 数据寄存器中或将报文对象传送到 IFx 数据寄存器中。

当仅仅只更新（8 个）数据字节时，最先将 0x0087 写入命令屏蔽寄存器。然后将报文对象的编号写入命令请求寄存器，同时更新数据字节，并置位 TXRQST。

在更新数据时，为防止在正进行的传送结束时复位 TXRQST 位，必须将 NEWDAT 和 TXRQST 同时置位。详细请参考 [Section 13.7.3.3](#)。

当同时设置 NEWDAT 和 TXRQST 时，一旦新传送启动，NEWDAT 将复位。

13. 7. 3. 8 配置接收对象

Table 212 所示为软件应如何初始化一个发送对象（也可参考 Table 189）

Table 212. 初始化接收对象

MSGVAL	Arbitration bits	Data bits	Mask bits	EOB	DIR	NEWDAT
1	application dependent	application dependent	application dependent	1	0	0
MSGLST	RXIE	TXIE	INTPND	RMTEN	TXRQST	
0	application dependent	0	0	0	0	

仲裁寄存器（ID28~0 和 XTD 位）由应用程序给定。他们定义接收到的接收报文的标识符和类型。如果使用 11 位标识符（“标准帧”），可将其编写为 ID28~ID18。可以忽略 ID17~ID0。当接收到具有 11 位标识符的数据帧时，ID17~ID0 将会被设置为 ‘0’。

如果 RXIE 位置位，则在接受了接收到的数据帧，将其保存到报文对象之后，置位 INTPND 位。

数据长度代码（DLC[3:0] 由应用程序给定。当报文处理程序将数据帧存放到报文对象时，它将会存放接收到的数据长度代码和 8 个数据字节。如果数据长度代码少于 8，则将报文对象的其余字节将会被非指定的值写覆盖。

可以使用屏蔽寄存器（Msk[28:0]、UMASK、MXTD 和 MDIR 位）（UMASK=’ 1’），以允许带有相似标识符的数据帧组被接受。详情请参考 Section 13. 7. 3. 4. 1. 在典型应用程序中，不应屏蔽 DIR 位。

13. 7. 3. 9 处理接收到的报文

CPU 随时可通过 IFx 接口寄存器读取接收到的报文。报文处理程序状态机可保证数据的一致性。

为将报文 RAM 中整个接收到的报文传送到报文缓冲区中，软件首先必须要将 0x007F 写入命令屏蔽寄存器。然后将报文对象的编号写入命令请求寄存器。另外，在报文 RAM（不是在报文缓冲区）中清除 NEWDAT 和 INTPND 位。

如果报文对象使用屏蔽来进行接收过滤操作，仲裁位会显示已接收到哪些匹配的报文。

NEWDAT 的实际值显示自从上一次读取该报文对象之后，是否接收到新的报文。MSGLST 的真实值显示自从上一次读取该报文对象后，是否接收到个数大于 1 的报文。MSGLST 不能自动复位。

使用远程帧时，CPU 可能要求另一个 CAN 节点为接收报文提供新数据。置位接收报文的 TXRQST 位将会使带接收报文对象标识符的远程帧被传送。该远程帧触发其它 CAN 节点来启动匹配数据帧的传送。如果在可以发送远程帧之前接收到匹配数据帧，TXRQST 位就自动复位。

13.7.3.10 配置 FIFO 缓冲区

除 EOB 位之外，属于 FIFO 缓冲区的接收对象的配置与（单个）接收对象的配置相同，见 [Section 13.7.3.8](#)。

为将二个或多个报文对象放入 FIFO 缓冲区，必须将这些报文对象的标识符和屏蔽（如有使用）编程为匹配值。由于报文对象的隐性优先级，具有最低编号的报文对象将会是 FIFO 缓冲区的第一个报文对象。FIFO 缓冲区的所有报文对象的 EOB 位（最后报文对象除外）必须编程为 0。FIFO 缓冲区的最后报文对象 EOB 位被设为 1，将其配置为模块的末端。

13.7.3.10.1 接收 FIFO 缓冲区报文

具有匹配 FIFO 缓冲区标识符的接收报文被存放在该 FIFO 缓冲区的报文对象中，从最低报文编号的报文对象开始执行。

当报文存放在 FIFO 缓冲区的报文对象时，该报文对象的 NEWDAT 位被置位。在 EOB 为 0 时置位 NEWDAT，报文对象被锁定，由报文处理程序进行进一步的写访问操作，直至 CPU 将 NEWDAT 位写为 0。

报文存放在 FIFO 缓冲区中，直至达到该 FIFO 缓冲区的最后一个报文对象。如果将 NEWDAT 写为 0 不会释放进程中的报文对象，则该 FIFO 缓冲区的所有以后报文都会被写入到 FIFO 缓冲区的最后一个报文对象，因此会覆盖之前的报文。

13.7.3.10.2 读取 FIFO 缓冲区

当 CPU 通过将报文编号写入到 IFx 命令请求寄存器来将报文对象的内容传送到 IFx 报文缓冲区寄存器时，相关命令屏蔽寄存器的 NEWDAT 和 INTPND 位应被复位为 0 (TXRQST/NEWDAT = '1' and ClrINTPND = '1')。报文控制寄存器中这些值总是反映这些位复位之前的状态。

为确保 FIFO 缓冲区功能正确，CPU 应在最低报文编号的 FIFO 对象处开始读出报文对象。

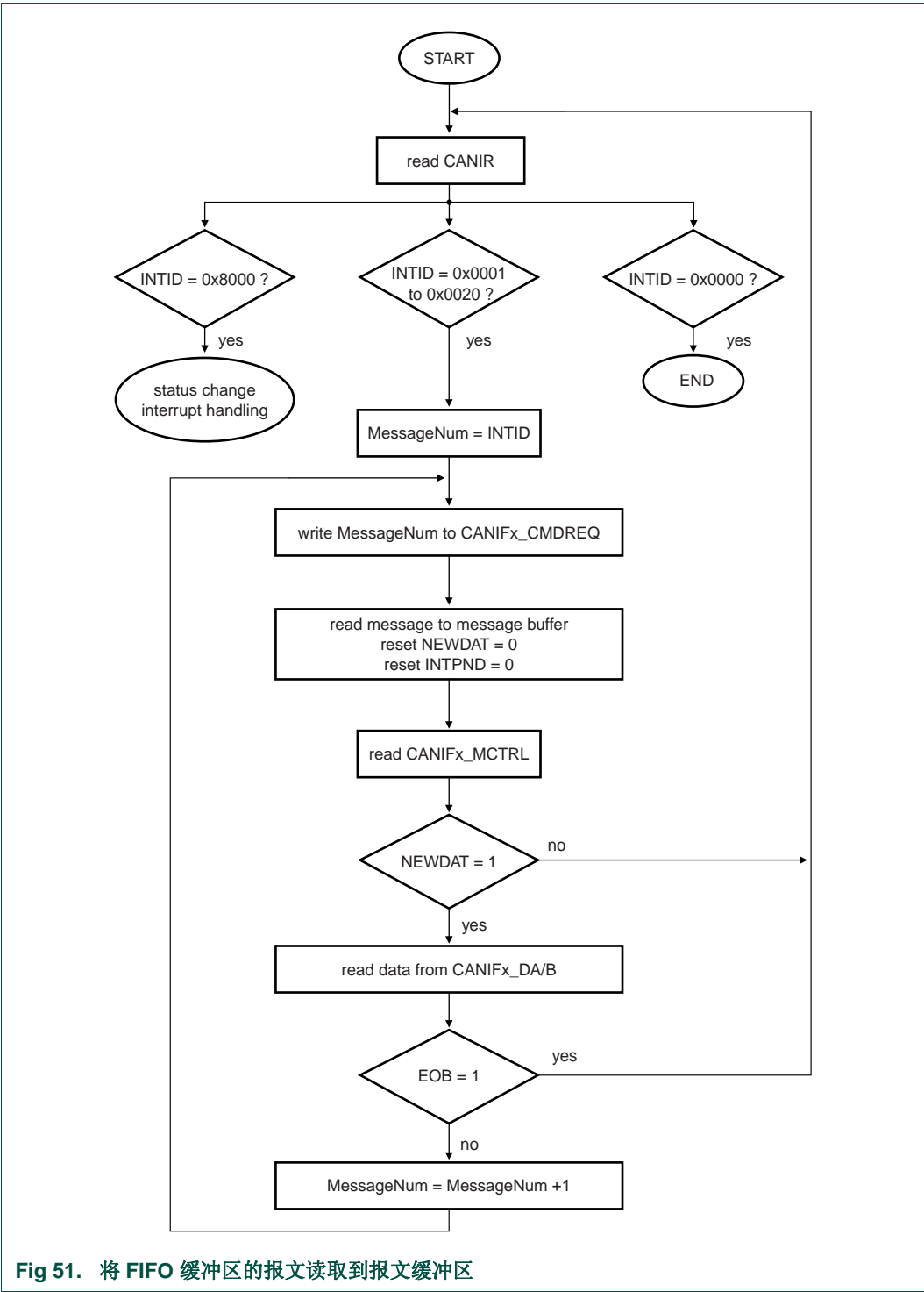


Fig 51. 将 FIFO 缓冲区的报文读取到报文缓冲区

13.7.4 中断处理

如果有几个中断被挂起，CAN 中断寄存器将会指向具有最高优先级的挂起中断，忽略它们时间的先后顺序。中断会保持挂起，直至 CPU 将其清除。

状态中断具有最高优先级。在报文中断里，报文对象的中断优先级随着报文编号的递增而下降。

将报文对象的 INTPND 位清除，即可清除报文中断。状态中断的清除则通过读取状态寄存器来完成。

中断寄存器中的中断标识符 INTID 表示引发中断的原因。当没有中断挂起时，寄存器将会保存 0 值。如果中断寄存器的值不为 0，那么存在中断被挂起，如果 IE 置位，则到 CPU 的中断 IRQ_B 有效。中断线保持有效，直至 IE 复位。

0x8000 值表示中断正在挂起，因为 CAN 内核已更新（没有必要更改）了状态寄存器（错误中断或状态中断）。该中断具有最高优先级。CPU 可以更新（复位）状态位 RXOK、TXOK 和 LEC，但是 CPU 对状态寄存器执行写访问操作将永不会产生或复位中断。

所有其它值表示中断源是其中的一个报文对象，其中 INTID 指向具有最高级中断优先级的挂起报文中断。

当中断寄存器不为 0 时（CAN 控制寄存器 IE 位），CPU 控制着状态寄存器的改变是否会引发中断（CAN 控制寄存器的 EIE 和 SIE 位）和中断线是否有效。即使 IE 复位时，中断寄存器会被更新。

CPU 有二种方式来查找报文中断源：

- 软件可以查看中断寄存器的 INTID 位；。
- 软件可以轮询中断挂起寄存器，见 [Section 13.6.3.5](#)。

读取报文（报文是中断源）的中断服务程序可以同时读取报文并复位报文对象的 INTPND（命令屏蔽寄存器的 ClrINTPND）。当 INTPND 被清除时，中断寄存器将会指向下一个具有挂起中断的报文对象。

13.7.5 位定时

尽管 CAN 位定时配置里的小错误不会导致立即的失败，但却会大大削弱 CAN 网络的性能。在许多情况下，CAN 位同步会将 CAN 位定时的错误配置改良到这样一个程度：只偶尔产生一个错误帧。但是，在仲裁情况下，当二个或多个 CAN 节点同时想要发送帧时，错位的采样点将会令其中一个发送器变为错误消极。

对偶尔错误的分析，要求对 CAN 节点内的 CAN 位同步和 CAN 总线上的 CAN 节点的互动知识有详细的了解。

13.7.5.1 位时间和比特率

CAN 支持的比特率范围低于 1KBit/s，并可高达 1000KBit/s。每一个 CAN 网络上的成员都具有自身的时钟发生器，通常是石英振荡器。可以为每一个 CAN 节点单独配置位时间的定时参数（即比特率的互相关数）以产生共同的比特率，即使 CAN 节点的振荡器周期 (f_{osc}) 可能不同。

这些振荡器的频率并不是绝对稳定的，因为温度的改变或电压和元件的恶化都会令频率发生微小的变化。只是变化量位于指定的振荡器容忍范围 (df) 内，CAN 节点可以通过重新同步到位流来向不同的比特率进行补偿。

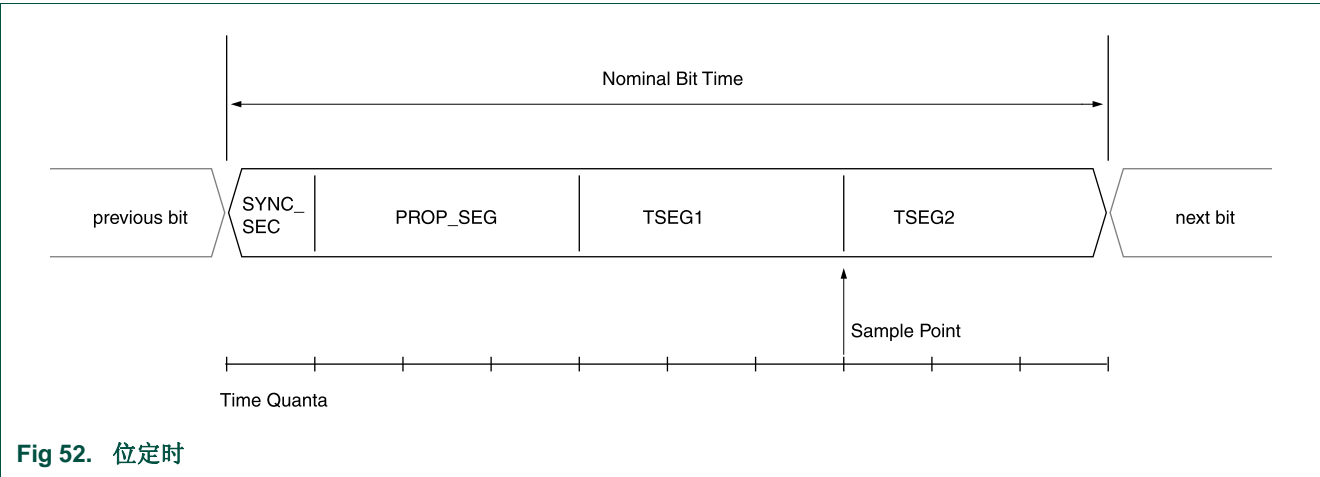
依据 CAN 规范，位时间被分成四个段 (Figure 52)：同步段、传播时间段、相位缓冲区间 1 和相位缓冲区间 2。每一个段包含有一个指定、可编程的时间量子数（见 Table 213），位时间的基本时间单元的时间量子的长度 (t_q) 由 CAN 控制器的系统时钟 f 和波特率预分频器确定 (BRP): $t_q = BRP / f_{sys}$ 。C_CAN 的系统时钟 f_{sys} 是 LPC11Cx 的系统时钟（见 Section 13.2）。

同步段 Sync_Seg 是位时间的一部分，CAN 总线的边沿电平会在这里出现，在 Sync_Seg 外部发生边沿和 Sync_Seg 上发生边沿之间的距离叫做该边沿的相位错误。传播时间段 Prop_Seg 用于对 CAN 网络内的物理延迟时间进行补偿。相位缓冲区间 Phase_Seg1 和 Phase_Seg2 包围着采样点。（重新）同步跳转宽度 (SJW) 定义着重新同步操作在相位缓冲区间所定义的内部限制内可将采样点移动的距离，以实现了对边沿相位错误作出补偿。

Table 213 描述了 CAN 协议所要求的最小编程范围。位时间参数通过 CANBT 寄存器 Table 184. 来进行编程。关于位定时和范例的详细描述，请参考修改版本 1.2 的 C_CAN 用户指南。

Table 213. C_CAN 位时间的参数

参数	范围	功能
BRP	(1...32)	定义时间量子 t_q 的长度。
SYNC_SEG	$1t_q$	同步段。长度固定，将总线输入同步到系统时钟。
PROP_SEG	$(1...8) \times t_q$	传播时间段。对物理延迟时间进行补偿，该参数由 C_CAN 网络里的系统延迟时间决定。
TSEG1	$(1...8) \times t_q$	相位缓冲区间 1。可通过同步操作临时加长时间长度。
TSEG2	$(1...8) \times t_q$	相位缓冲区间 2。可通过同步操作临时缩短时间长度。
SJW	$(1...4) \times t_q$	(重新) 同步跳转宽度。其长度不可以长于相位缓冲区间。



14.1 怎样阅读本章

C_CAN 模块只在 LPC11Cxx (LPC11C00 系列) 中才具有。

14.2 特性

片上驱动程序存储在 boot ROM 中，并通过定义好了的 API 向用户应用程序提供 CAN 和 CANopen 的初始化和通信特性。下面的函数都包含在 API 中：

- CAN 的配置和初始化
- CAN 发送和接收报文
- CAN 状态
- CANopen 对象词典
- CANopen SDO 加速通信
- CANopen SDO 分段通信基元
- CANopen SDO 返回处理

14.3 概述

除了 CAN ISP 之外，boot ROM 还提供了 CAN 和 CANopen API，以简化 CAN 的应用开发。它涵盖了初始化、配置、CAN 的基本发送/接收以及一个 CANopen SDO 接口。此外，回调函数可以用来处理接收事件。

14.3.1 完全兼容 CANopen 差异

虽然 bootloader 使用了 SDO 通信协议和对象词典的数据组织方法，但它仍不是一个完全的 CiA301 标准兼容的 CANopen 节点。尤其要指出以下的功能不可用或者不同于标准。

- 没有网络管理的报文处理。
- 没有心跳消息，没有入口 0x1017。
- 采用专有的 SDO 中止代码显示设备错误。
- 在 SDO 分段下载/写入节点时，空 SDO 响应缩短为一个数据字节，而不是象标准所描述的八个字节，因而加速了通信进程。
- 入口 [1018h,1] 的供应商 ID 从地址 0x0000 0000 读取，而不是 CiA 官方指定的唯一供应商 ID。这主要是因为芯片将被纳入用户设计之中，而用户将成为整个设备的供应商，因此主机不得不使用另一种不同的方法来识别 CAN ISP 器件。

14.4 API description

14.4.1 调用 C_CAN API

ROM 中有一个固定的单元包含了一个指向 ROM 驱动程序表的指针，即 0x1FFF 1FF8。该单元在所有的 LPC11xx 系列器件中都是相同的。ROM 的驱动程序表包含一个 CAN API 表的指针，所有指向不同 CAN API 函数的指针存放在这个表中，并且可以使用 C 结构体来调用 CANAPI 函数。

Figure 53 显示了用于访问片上 CAN API 的指针机制。CAN API 使用了地址从 0x1000 0050 到 0x1000 00B8 的片上 RAM，应用程序不能使用该范围的地址。对于使用片上地址的应用程序，应当修改连接器控制文件，以防止该区域用于应用程序的各种存储操作。

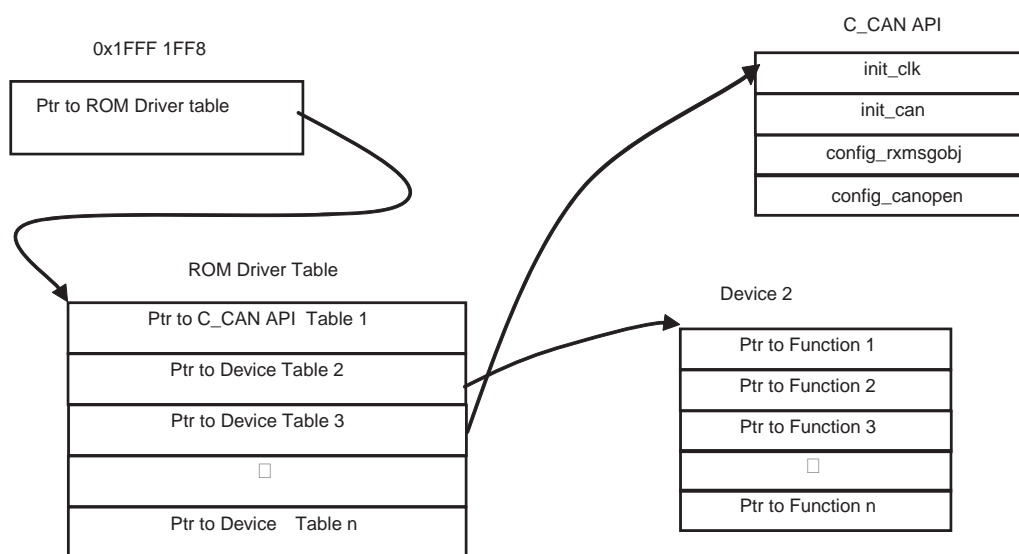


Fig 53. CAN API 指针结构体

在 C 语言中，使用带函数列表的结构体来调用来调用 API 函数，结构体如下所示：

```

typedef struct _CAND {
    void (*init_can) (uint32_t * can_cfg);
    void (*isr) (void);
    void (*config_rxmsgobj) (CAN_MSG_OBJ * msg_obj);
    uint8_t (*can_receive) (CAN_MSG_OBJ * msg_obj);
    void (*can_transmit) (CAN_MSG_OBJ * msg_obj);
    void (*config_canopen) (CAN_CANOPENCFG * canopen_cfg);
    void (*canopen_handler) (void);
    void (*config_calb) (CAN_CALLBACKS * callback_cfg);
} CAND;
  
```

14.4.2 CAN 的初始化

CAN 控制器的时钟分频器和比特率设置，以及初始化 CAN 控制器，都是基于一个存放指针值的数组来实现的，而这些值则通过指针来进行传递。

```
void init_can (uint32_t * can_cfg, uint8_t isr_ena)
```

数组中第一个 32 位值被应用到 CANCLKDIV 寄存器，第二个值则被应用到 CAN_BTR 寄存器。

第二个参数变量允许 CAN 控制器中断，设置为 FALSE 以实现轮询通信。

调用范例：

```
ROM **rom = (ROM **)0x1fff1ff8;

uint32_t CanApiClkInitTable[2] = {
    0x00000000UL, // CANCLKDIV
    0x00004DC5UL // CAN_BTR
};

(*rom)->pCANAPI->init_can(&CanApiCanInitTable[0]);
```

14.4.3 CAN 的中断处理

当用户应用程序有效时，中断处理程序会映射到用户 Flash 空间中。用户应用程序必须为 CAN 中断提供中断处理程序。为了处理 CAN 事件和调用回调函数，应用程序必须从中断处理程序服务中直接调用 CAN API 中断处理程序。CAN API 中断处理程序根据在 CAN 总线上接收到的数据和检测到的状态来采取相关的操作。

```
void isr (void)
```

CAN 的中断处理程序并不处理 CANopen 消息。

调用范例：

```
(*rom)->pCAND->isr();
```

对于轮询通信，中断处理程序根据需求可以被人为调用。而用于发送、接收和报错的回调函数则会按照所描述的方式执行，并且按照相同的等级来调用相应的中断处理程序。

14.4.4 CAN 接收消息的对象设置

CAN API 支持和使用具有 32 个报文对象的完整 CAN 模块。任何报文对象都可以用来接收和发送 11 位或 29 位的 CAN 报文，具有 RTR 位设置（远程发送）的 CAN 报文同样被支持。对于接收对象，报文标识符的屏蔽模式可允许接收的报文范围，可高至接收单个报文对象里总线上的所有 CAN 报文。

参见 [Section 13.7.3.4](#).

发送报文对象在使用时自动被配置。

```
// CAN_MSG_OBJ.mode_id的控制位
#define CAN_MSGOBJ_STD 0x00000000UL // CAN 2.0a 11-bit ID
#define CAN_MSGOBJ_EXT 0x20000000UL // CAN 2.0b 29-bit ID
#define CAN_MSGOBJ_DAT 0x00000000UL // data frame
#define CAN_MSGOBJ_RTR 0x40000000UL // rtr frame

typedef struct _CAN_MSG_OBJ {
    uint32_t mode_id;
    uint32_t mask;
    uint8_t data[8];
    uint8_t dlc;
    uint8_t msgobj;
} CAN_MSG_OBJ;
```

```
void config_rxmsgobj (CAN_MSG_OBJ * msg_obj)
```

调用范例：

```
// 配置报文对象 从 1 到接收所有 11 位报文 0x000-0x00F
msg_obj.msgobj = 1;
msg_obj.mode_id = 0x000;
msg_obj.mask = 0x7F0;
(*rom)->pCAND-> config_rxmsgobj(&msg_obj);
```

14.4.5 CAN 发送

CAN 接收函数允许读取 Rx 报文对象接收到的报文，指向报文对象结构体的指针会被传递到接收函数中。在调用函数之前，必须要在结构体中设置将要被读取的报文对象的编号。

```
void config_rxmsgobj (CAN_MSG_OBJ * msg_obj)
```

调用范例：

```
// 读取接收到的报文
msg_obj.msgobj = 5;
(*rom)->pCAND->can_receive(&msg_obj);
```

14.4.6 CAN 发送

CAN 发送函数允许设置报文对象，并可在总线上触发 CAN 报文的发送。11 位 标准和 29 位扩展报文对象被看作为标准数据和远程发送（RTR）报文。

```
void config_txmsgobj (CAN_MSG_OBJ * msg_obj)
```

调用范例：

```

msg_obj.msgobj = 3;
msg_obj.mode_id = 0x123UL;
msg_obj.mask = 0x0UL;
msg_obj.dlc = 1;
msg_obj.data[0] = 0x00;
(*rom)->pCAND->can_transmit(&msg_obj);

```

14.4.7 CANopen 配置

CAN API 支持对象字典接口和 SDO 协议。为了激活 CANopen 配置函数，必须要用一个指向结构体的指针来调用它，这个结构体包含 CANopen 的结点 ID (1...127)，该 ID 也是用来接收和发送 SDOs 的报文对象编号，同时是一个有 CANopen SDO 处理是自动发生在中断服务函数中，还是通过专用 API 函数处理的标志量。除指向结构体的指针外，调用 CANopen 配置函数还需要两个指向对象字典配置表和它们大小的指针。一个表包含 4 个字节或少于 4 个字节的所有只读、常量入口，另一个表包含所有变量和可写以及 SDO 分段的入口。

```

typedef struct _CAN_ODCONSTENTRY {
    uint16_t index;
    uint8_t subindex;
    uint8_t len;
    uint32_t val;
} CAN_ODCONSTENTRY;

// CAN_ODENTRY.entrytype_len 的高位半字节值
#define OD_NONE 0x00 // Object Dictionary entry doesn't exist
#define OD_EXP_RO 0x10 // Object Dictionary entry expedited, read-only
#define OD_EXP_WO 0x20 // Object Dictionary entry expedited, write-only
#define OD_EXP_RW 0x30 // Object Dictionary entry expedited, read-write
#define OD_SEG_RO 0x40 // Object Dictionary entry segmented, read-only
#define OD_SEG_WO 0x50 // Object Dictionary entry segmented, write-only
#define OD_SEG_RW 0x60 // Object Dictionary entry segmented, read-write

typedef struct _CAN_ODENTRY {
    uint16_t index;
    uint8_t subindex;
    uint8_t entrytype_len;
    uint8_t isr_handled;
    uint8_t *val;
} CAN_ODENTRY;

typedef struct _CAN_CANOPENCFG {
    uint8_t node_id;
    uint8_t msgobj_rx;
    uint8_t msgobj_tx;
    uint32_t od_const_num;
    CAN_ODCONSTENTRY *od_const_table;
    uint32_t od_num;
    CAN_ODENTRY *od_table;
} CAN_CANOPENCFG;

```

OD 表和 CANopen 配置结构体的范例：

```
// 固定的、只读对象字典 (OD) 入口的列表
// 只加速的 SDO, 长度为 1/2/4 字节
const CAN_ODCONSTENTRY myConstOD [] = {
// 索引分类指数的长度值
    { 0x1000, 0x00, 4, 0x54534554UL }, // "TEST"
    { 0x1018, 0x00, 1, 0x00000003UL },
    { 0x1018, 0x01, 4, 0x00000003UL },
    { 0x2000, 0x00, 1, (uint32_t)'M' },
};

// 变量 OD 入口的列表
// 加速的 SDO, 长度为 1/2/4 字节
// 分段 SDO 应用程序处理, 长度和数值指针无关紧要
const CAN_ODENTRY myOD [] = {
// 索引分类指数访问类型 | 长度数值指针
    { 0x1001, 0x00, OD_EXP_RO | 1, (uint8_t *)&error_register },
    { 0x1018, 0x02, OD_EXP_RO | 4, (uint8_t *)&device_id },
    { 0x1018, 0x03, OD_EXP_RO | 4, (uint8_t *)&fw_ver },
    { 0x2001, 0x00, OD_EXP_RW | 2, (uint8_t *)&param },
    { 0x2200, 0x00, OD_SEG_RW, (uint8_t *)NULL },
};

// CANopen 配置结构体
const CAN_CANOPENCFG myCANopen = {
    20, // node_id
    5, // msgobj_rx
    6, // msgobj_tx
    TRUE, // isr_handled
    sizeof(myConstOD)/sizeof(myConstOD[0]), // od_const_num
    (CAN_ODCONSTENTRY *)myConstOD, // od_const_table
    sizeof(myOD)/sizeof(myOD[0]), // od_num
    (CAN_ODENTRY *)myOD, // od_table
};

调用范例：

// 初始化 CANopen 处理程序
(*rom)->pCAND->config_canopen((CAN_CANOPENCFG *)&myCANopen);
```

14.4.8 CANopen 处理程序

CANopen 处理程序通过处理 CANopen SDO 报文来访问对象字典, 并在初始化时调用 CANopen 的回调函数。它既可以被中断处理程序自动调用 (在 CANopen 初始化结构体中 `isr_handled == TRUE`), 也可以通过 CANopen 处理程序的 API 函数人为调用。如果人为调用, 那么只要应用程序有需要, 就必须经常周期性的调用 CANopen 处理程序。

在典型的 CANopen 应用程序中, SDO 处理具有最低的优先权, 并在前台完成, 而不是在中断进程中完成。

调用范例：

```
// 调用 CANopen 中断处理程序
(*rom)->pCAND->canopen_handler();
```

14.4.9 CAN/CANopen 回调函数

CAN API 支持各种事件的回调函数，回调函数通过 API 函数来发布。

```
typedef struct _CAN_CALLBACKS {
    void (*CAN_rx)(uint8_t msg_obj);
    void (*CAN_tx)(uint8_t msg_obj);
    void (*CAN_error)(uint32_t error_info);
    uint32_t (*CANOPEN_sdo_read)(uint16_t index, uint8_t subindex);
    uint32_t (*CANOPEN_sdo_write)(
        uint16_t index, uint8_t subindex, uint8_t *dat_ptr);
    uint32_t (*CANOPEN_sdo_seg_read)(
        uint16_t index, uint8_t subindex, uint8_t openclose,
        uint8_t *length, uint8_t *data, uint8_t *last);
    uint32_t (*CANOPEN_sdo_seg_write)(
        uint16_t index, uint8_t subindex, uint8_t openclose,
        uint8_t length, uint8_t *data, uint8_t *fast_resp);
    uint8_t (*CANOPEN_sdo_req)(
        uint8_t length_req, uint8_t *req_ptr, uint8_t *length_resp,
        uint8_t *resp_ptr);
} CAN_CALLBACKS;
```

回调表定义范例：

```
// 回调函数指针列表
const CAN_CALLBACKS callbacks = {
    CAN_rx,
    CAN_tx,
    CAN_error,
    CANOPEN_sdo_exp_read,
    CANOPEN_sdo_exp_write,
    CANOPEN_sdo_seg_read,
    CANOPEN_sdo_seg_write,
    CANOPEN_sdo_req,
};
```

调用范例：

```
// 发布回调
(*rom)->pCAND->config_calb((CAN_CALLBACKS *)&callbacks);
```

14.4.10 CAN 报文接收的回调

CAN 中断处理程序会按照中断的级别来调用 CAN 报文接收回调函数。

调用范例：

```
// CAN 接收处理程序
void CAN_rx(uint8_t msgobj_num)
{
    // 读取接收的报文
    msg_obj.msgobj = msgobj_num;
    (*rom)->pCAND->can_receive(&msg_obj);

    return;
}
```

注意：如果用户 CANopen 处理程序是为用于 SDO 接收的报文对象激活时，不必调用该回调函数。

14.4.11 CAN 报文发送回调

由 CAN 中断处理程序按照中断级别调用 CAN 报文发送回调函数，在一个报文后已经成功的在总线上发送之后调用。

调用范例：

```
// CAN 发送处理程序
void CAN_tx(uint8_t msgobj_num)
{
    // 应用程序使用复位标识符来等待发送结束
    if (wait_for_tx_finished == msgobj_num)
        wait_for_tx_finished = 0;

    return;
}
```

注意：在用户 CANopen 处理程序已经通过报文对象发送了一个 SDO 响应后就不必再调用该回调函数。

14.4.12 CAN 错误回调

CAN 中断处理程序按照中断级别调用 CAN 错误回调函数。

```
// error status bits
#define CAN_ERROR_NONE 0x00000000UL
#define CAN_ERROR_PASS 0x00000001UL
#define CAN_ERROR_WARN 0x00000002UL
#define CAN_ERROR_BOFF 0x00000004UL
#define CAN_ERROR_STUF 0x00000008UL
#define CAN_ERROR_FORM 0x00000010UL
#define CAN_ERROR_ACK 0x00000020UL
#define CAN_ERROR_BIT1 0x00000040UL
#define CAN_ERROR_BIT0 0x00000080UL
#define CAN_ERROR_CRC 0x00000100UL
```

调用范例：

```
// CAN 错误处理程序
```

```
void CAN_error(uint32_t error_info)
// 如果我们进入总线关闭状态，就要告诉应用程序
// 重新初始化 CAN 控制器
if (error_info & CAN_ERROR_BOFF)
    reset_can = TRUE;

return;
}
```

14.4.13 CANopen SDO 加速读取回调

CANopen SDO 加速读取回调函数由 CANopen 程序调用。在 SDO 产生响应之前就要调用该回调函数，以便允许修改和更新数据。

调用范例：

```
// 用于加速读访问的 CANopen 回调
uint32_t CANOPEN_sdo_exp_read(uint16_t index, uint8_t subindex)
{
    // 每次读 [2001h,0] 都会令 param 加 1
    if ((index == 0x2001) && (subindex==0))
        param++;

    return 0;
}
```

注意：当初始化 CANopen 时，如果 `isr_handled` 标志已经被设置为 TRUE，CAN API 中断处理程序会调用该回调函数，并按照中断的级别执行它。

14.4.14 CANopen SDO 加速写回调

CANopen SDO 加速写回调函数由 CANopen 中断程序调用。回调函数传递新数据，但在新数据写入之前就调用回调函数，以允许拒绝数据或给接收数据施加条件。

调用范例：

```
// 用于加速写访问的 CANopen 回调
uint32_t CANOPEN_sdo_exp_write(uint16_t index, uint8_t subindex, uint8_t
*dat_ptr)
{
    // 写 0xAA55 到入口 [2001h,0] 以把配置表解锁
    if ((index == 0x2001) && (subindex == 0))
        if (*(uint16_t *)dat_ptr == 0xAA55)
        {
            write_config_ena = TRUE;
            return(TRUE);
        }
        else
            return(FALSE); // Reject any other value
}
```

注意：当初始化 CANopen 时，如果 `isr_handled` 标志量已经被设置为 `TRUE`，那么 CAN API 中断处理程序会调用该回调函数，并按照中断级别来执行它。

14.4.15 CANopen SDO 分段读回调

CANopen SDO 分段读回调函数由 CANopen 处理程序调用。回调函数允许以下操作：

- 通知打开读通道
- 可向读取主机提供高达 7 个字节的数据段
- 当读取完所有数据时关闭通道
- 可随时终止传输

```
// 用于 CANOPEN_sdo_seg_read/write() callback 'openclose' 参数的值
#define CAN_SDOSEG_SEGMENT 0 // segment read/write
#define CAN_SDOSEG_OPEN 1 // channel is opened
#define CAN_SDOSEG_CLOSE 2 // channel is closed
```

调用范例 (读缓冲区):

```
uint8_t read_buffer[0x123];

// 用于分段读访问的 CANopen 回调
uint32_t CANOPEN_sdo_seg_read(
    uint16_t index, uint8_t subindex, uint8_t openclose,
    uint8_t *length, uint8_t *data, uint8_t *last)
{
    static uint16_t read_ofs;
    uint16_t i;

    if ((index == 0x2200) && (subindex==0))
    {
        if (openclose == CAN_SDOSEG_OPEN)
        {
            // 用 "something" 初始化读缓冲区
            for (i=0; i<sizeof(read_buffer); i++)
            {
                read_buffer[i] = (i+5) + (i<<2);
            }
            read_ofs = 0;
        }
        else if (openclose == CAN_SDOSEG_SEGMENT)
        {
            i = 7;
            while (i && (read_ofs < sizeof(read_buffer)))
            {
                *data++ = read_buffer[read_ofs++];
                i--;
            }
        }
    }
}
```

```

        *length = 7-i;
        if (read_ofs == sizeof(read_buffer)) // 读取整个缓冲区

                                                    // 这是最后的段区
        {
            *last = TRUE;
        }
    }
    return 0;
}
else
{
    return SDO_ABORT_NOT_EXISTS;
}
}

```

注意：当初始化 CANopen 时，如果 `isr_handled` 标志量已经被设置为 `TRUE`，那么 CAN API 中断处理程序会调用回调函数，并按照中断级别来调用它。

14.4.16 CANopen SDO 分段写回调

CANopen SDO 分段写回调函数由 CANopen 处理程序调用。回调函数允许以下操作：

- 通知打开和关闭写通道
- 可从写主机中传输高达 7 个字节的数据段
- 可随时终止传输，例如，当存在缓冲区溢出时

可以选择 8 字节（符合 CANopen 标准）或 1 字节（更快但并不是所有的 SDO 客户端都支持）响应。

```

// 用于 CANOPEN_sdo_seg_read/write() callback 'openclose' 参数的值
#define CAN_SDOSEG_SEGMENT 0 // segment read/write
#define CAN_SDOSEG_OPEN 1 // channel is opened
#define CAN_SDOSEG_CLOSE 2 // channel is closed

```

调用范例（写缓冲区）：

```

uint8_t write_buffer[0x321];

// 用于分段写访问的 CANopen 回调
uint32_t CANOPEN_sdo_seg_write(
    uint16_t index, uint8_t subindex, uint8_t openclose,
    uint8_t length, uint8_t *data, uint8_t *fast_resp)
{
    static uint16_t write_ofs;
    uint16_t i;

    if ((index == 0x2200) && (subindex==0))
    {
        if (openclose == CAN_SDOSEG_OPEN)
        {

```

```

// 初始化写缓冲区
for (i=0; i<sizeof(write_buffer); i++)
{
    write_buffer[i] = 0;
}
write_ofs = 0;
}
else if (openclose == CAN_SDOSEG_SEGMENT)
{
    *fast_resp = TRUE; // 使用快速1字节分段写响应
    i = length;
    while (i && (write_ofs < sizeof(write_buffer)))
    {
        write_buffer[write_ofs++] = *data++;
        i--;
    }
    if (i && (write_ofs >= sizeof(write_buffer))) // 太多数据要写
    {
        return SDO_ABORT_TRANSFER; // 不能写入的数据
    }
}
else if (openclose == CAN_SDOSEG_CLOSE)
{
    // 写操作完成：标记缓冲区有效等。
}
return 0;
}
else
{
    return SDO_ABORT_NOT_EXISTS;
}
}

```

注意：当初始化 CANopen 时，如果 `isr_handled` 标志量已经被设置为 `TRUE`，那么 CAN API 中断处理程序会调用回调函数，并按照中断级别来调用它。

14.4.17 CANopen fall-back SDO 处理程序回调

CANopen fall-back SDO 处理程序回调函数由 CANopen 中断处理程序调用。只要存在着一个不能被处理的 SDO 请求或者 SDO 请求被终止响应，都要调用该函数。调用时要使用所请求的完整数据缓冲区，并允许产生任何类型的 SDO 响应。这可用于执行自定义的 SDO 处理程序，例如，执行 SDO 块传输方法。

```

// 返回 CANOPEN_sdo_req() 的值
#define CAN_SDOREQ_NOTHANDLED 0 // 正常处理，无影响
#define CAN_SDOREQ_HANDLED_SEND 1 // 回调处理，自动发送
// 返回报文
#define CAN_SDOREQ_HANDLED_NOSEND 2 // 回调处理，不发送
// 响应

```

调用范例 (不执行自定义处理):

```
// 自定义 SDO 请求处理程序的 CANopen 回调
uint8_t CANOPEN_sdo_req (
    uint8_t length, uint8_t *req_ptr, uint8_t *length_resp, uint8_t *resp_ptr)
{
    return CAN_SDOREQ_NOTHANDLED;
}
```

注意：当初始化 CANopen 时，如果 `isr_handled` 标志量已经被设置为 `TRUE`，那么 CAN API 中断处理程序会调用回调函数，并按照中断级别来调用它。

15.1 怎样阅读本章

LPC111x 中的 16 位定时器模块和 LPC11Cxx 中的完全一样。

在 LPC11C22 和 LPC11C24 中，定时器 1 的 MAT0 匹配输出没有与引脚直接相连。

15.2 基本配置

定时器 CT16B0/1 通过以下寄存器进行配置：

1. CT16B0/1 引脚并须在 IOCONFIG 寄存器中配置 ([Section 7.4](#)).
2. 电源和外设时钟：在 SYSAHBCLKCTRL 寄存器中设置第 7,8 位来允许电源时钟 ([Table 21](#))., 外设时钟由系统时钟提供（见 [Table 20](#)）。

15.3 特征

- 2 个带可编程的 16 位预分频器的 16 位定时器 / 计数器。
- 可进行定时或计数操作。
- 具有 1 路 16 位的捕获通道。当输入信号跳变时可捕获定时器当前值。也可以选择使捕获事件产生中断。
- 4 个 16 位的匹配寄存器
 - 匹配时持续比较，可选择产生中断。
 - 匹配时停止定时器，可选择产生中断。
 - 匹配时重启定时器，可选择产生中断。
- CT16B0 具有 3 个外部输出，CT16B1 具有 2 个外部输出，其对应的匹配寄存器带有以下功能：
 - 匹配时设置低电平。。
 - 匹配时设置高电平。
 - 匹配时翻转。
 - 匹配时无操作。
- 对于每个定时器，多达 4 个可配置为 PWM 的匹配寄存器，允许使用 3 个匹配输出作为单边沿控制的 PWM 输出。

15.4 应用

- 为间隔事件计数的限时器。
- 通过捕获输入的脉冲宽度解调器。
- 不同步的定时器。
- 通过匹配输出的脉冲宽度调制器。

15.5 概述

每个计数 / 定时器用于对外设时钟（PLCK）或者外部提供的时钟进行计数。用户可以指定 4 个匹配寄存器的定时器的值来在规定的时间内产生中断或执行其他操作。外设时钟是由系统时钟提供的（见 [Figure 3](#)）。每个计数 / 定时器具有捕获比较功能，用来在输入信号变化时捕获定时器的值和产生中断。

在 PWM 模式下，CT16B0 有 3 个匹配寄存器用于提供一个单边沿的控制 PWM 输出，而 CT16B1 只有两个。推荐使用没有与引脚相连的匹配寄存器控制 PWM 周期长度。

注意：CT16B0 和 CT16B1 除了外设基址不一样，其他的完全一样。

15.6 Pin description

[Table 214](#) 简单总结了一下与定时器 / 计数器相关的引脚。

Table 214. 计数器 / 定时器引脚描述

引脚	类型	描述
CT16B0_CAP0 CT16B1_CAP0	输入	捕获信号： 捕获引脚的跳变可配置为将定时器 / 计数器值装入捕获寄存器，并可选择产生一个中断 计数器 / 定时器可选择一个捕获信号作为时钟源来代替 PCLK 分频时钟，更多细节请 看见 Section 15.7.11 。
CT16B0_MAT[2:0] CT16B1_MAT[1:0]	输出	CT16B0/1 的外部匹配输出： 当 CT16B0/1 匹配寄存器（MR3:0）任何一个等于定时器计数器（TC）时，其输出可 翻转，变为低电平高电平或不变。外部匹配寄存器（EMR）和 PWM 控制寄存器 （PWMCON）控制该输出的功能。

15.7 寄存器描述

CT16B0/0 寄存器如 [Table 215](#) 所示，CT16B0/1 寄存器如 [Table 216](#) 所示 更多细节描述如下：

Table 215. 寄存器列表 : 16-bit counter/timer 0 CT16B0 (基地址 0x4000 C000)

名字	访问	地址偏移量	描述	复位值 [1]
TMR16B0IR	R/W	0x000	中断寄存器 (IR). 写入该寄存器可以用来清除中断, 读此寄存器可以用来判断可能的 五个中断源哪个被挂起。	0
TMR16B0TCR	R/W	0x004	定时器控制寄存器 (TCR): 用来控制定时器计数器功能。通过 TCR 可以禁止或重启定时器计数器。	0
TMR16B0TC	R/W	0x008	定时器计数器 (TC). PCLK 控制预分频计数器值递增, 当它达到预分频寄存器 PR 中的值时, TC 加 1, TC 由 TCR 控制。	0
TMR16B0PR	R/W	0x00C	预分频寄存器 (PR). 当预分频计数器等于这个值时, 在下个时钟, TC 加 1, PC 清 0.	0
TMR16B0PC	R/W	0x010	预分频计数器 (PC). 16 位的 PC 是一个从 0 递增到存储在 PR 中的值。当它等于 PR 中的值时, TC 加 1, PC 清 0, 通过总线接口, PC 是可见的, 并且是可控制的。	0
TMR16B0MCR	R/W	0x014	匹配控制寄存器 (MCR). 当一个中断产生, 或是当一个匹配发生, TC 复位时, MCR 用于控制它们。	0
TMR16B0MR0	R/W	0x018	匹配寄存器 0 (MR0). MR0 通过 MCR 使能去重置 TC, 停止 TC 和 PC, 或当 MR0 与 TC 匹配时产生中断。	0
TMR16B0MR1	R/W	0x01C	匹配寄存器 1 (MR1). 参见 MR0 描述。	0
TMR16B0MR2	R/W	0x020	匹配寄存器 2 (MR2). 参见 MR0 描述。	0
TMR16B0MR3	R/W	0x024	匹配寄存器 3 (MR3). 参见 MR0 描述。	0
TMR16B0CCR	R/W	0x028	捕获控制寄存器 (CCR). CCR 控制捕获事件是发生在上升沿, 下降沿, 还是双边沿是否装载捕获寄存器的值以及是否产生中断。	0
TMR16B0CR0	RO	0x02C	捕获寄存器 0 (CR0). 当有一个事件在 CT16B1_CAP0 输入端发生时, CR0 将会装载 TC 中的值。	0
TMR16B0EMR	R/W	0x03C	外部匹配寄存器 (EMR). EMR 控制匹配功能, 并且控制外部匹配引脚 and CT16B1_MAT[2:0]。	0
-	-	0x040 - 0x06C	保留	-
TMR16B0CTCR	R/W	0x070	计数控制寄存器 (CTCR) CTCR 选择使计数器或是定时器模式, 在计数器模式下, 选择一个单边沿或是多边沿计数。	0
TMR16B0PWMC	R/W	0x074	PWM 控制寄存器 (PWMCON). PWMCON 用于将外部匹配输出引脚 CT16B1_MAT[2:0] 配置为 PWM 输出。	0

[1] 复位值仅仅反映使用位的数据。它不包括保留位的内容。

Table 216. 寄存器列表 : 16-bit counter/timer 1 CT16B1 (基地址 0x4001 0000)

名字	访问	地址偏移量	描述	复位值 ^[1]
TMR16B1IR	R/W	0x000	中断寄存器 (IR). 写入该寄存器可以用来清除中断, 读此寄存器可以用来判断可能的 五个中断源哪个被挂起。	0
TMR16B1TCR	R/W	0x004	定时器控制寄存器 (TCR): 用来控制定时器计数器功能。通过 TCR 可以禁止或重启定时器计数器。	0
TMR16B1TC	R/W	0x008	定时器计数器 (TC).PCLK 控制预分频计数器值递增, 当它达到预分频寄存器 PR 中的值时, TC 加 1, TC 由 TCR 控制。	0
TMR16B1PR	R/W	0x00C	预分频寄存器 (PR). 当预分频计数器等于这个值时, 在下个时钟, TC 加 1, pc 清 0.	0
TMR16B1PC	R/W	0x010	预分频计数器 (PC). 16 位的 PC 是一个从 0 递增到存储在 PR 中的值。当它等于 PR 中的值时, TC 加 1, PC 清 0, 通过总线接口, PC 是可见的, 并且是可控制的。	0
TMR16B1MCR	R/W	0x014	匹配控制寄存器 (MCR). 当一个中断产生, 或是当一个匹配发生, TC 复位时, MCR 用于控制它们。	0
TMR16B1MR0	R/W	0x018	匹配寄存器 0 (MR0). MR0 通过 MCR 使能去重置 TC, 停止 TC 和 PC, 或当 MR0 与 TC 匹配时产生中断。	0
TMR16B1MR1	R/W	0x01C	匹配寄存器 1 (MR1). 参见 MR0 描述。	0
TMR16B1MR2	R/W	0x020	匹配寄存器 2(MR2). 参见 MR0 描述。	0
TMR16B1MR3	R/W	0x024	匹配寄存器 3(MR3). 参见 MR0 描述。	0
TMR16B1CCR	R/W	0x028	捕获控制寄存器 (CCR). CCR 控制捕获事件是发生在上升沿, 下降沿, 还是双边沿是否装载捕获寄存器的值以及是否产生中断。	0
TMR16B1CR0	RO	0x02C	捕获寄存器 0 (CR0). 当有一个事件在 CT16B1_CAP0 输入端发生时, CR0 将会装载 TC 中的值。	0
TMR16B1EMR	R/W	0x03C	外部匹配寄存器 (EMR). EMR 控制匹配功能, 并且控制外部匹配引脚 CT16B1_MAT[1:0]。	0
-	-	0x040 - 0x06C	保留	-
TMR16B1CTCR	R/W	0x070	计数控制寄存器 (CTCR) CTCR 选择使计数器或是定时器模式, 在计数器模式下, 选择一个单边沿或是多边沿计数。	0
TMR16B1PWMC	R/W	0x074	PWM 控制寄存器 (PWMCON). PWMCON 用于将外部匹配输出引脚 CT16B1_MAT[1:0] 配置为 PWM 输出。	0

[1] 复位值仅仅反映使用位的数据。它不包括保留位的内容。

15.7.1 中断寄存器 (TMR16B0IR and TMR16B1IR)

中断寄存器 (IR) 包含 4 位来匹配中断, 一位来捕获中断。如果一个中断发生, IR 相应的位会被置位。否则, 这些位为 0. 写 1 到 IR 中相应的位会复位中断, 写 0 则没有影响。

Table 217. 中断寄存器 (TMR16B0IR – 地址 0x4000 C000 , TMR16B1IR – 地址 0x4001 0000) 位描述

位	符号	描述	复位值
0	MR0 Interrupt	匹配通道 0 的中断标志位	0
1	MR1 Interrupt	匹配通道 1 的中断标志位	0
2	MR2 Interrupt	匹配通道 2 的中断标志位	0
3	MR3 Interrupt	匹配通道 3 的中断标志位	0
4	CR0 Interrupt	捕获通道 0 事件的中断标志位	0
31:5	-	保留	-

15.7.2 定时器控制寄存器 (TMR16B0TCR and TMR16B1TCR)

定时器控制寄存器 (TCR) 用于控制定时器 / 计数器的各种操作。

Table 218. 定时器控制寄存器 (TMR16B0TCR - 地址 0x4000 C004 和 TMR16B1TCR - 地址 0x4001 0004) 位描述

位	符号	描述	复位值
0	CEn	计数器使能: 为 1 时, 定时器, 计数器和福分频器被允许计数, 为 0 时, 则计数器被禁止。	0
1	CRst	计数器复位: 为 1 时, 定时 / 计数器和预分频器在下一个 PCLK 上升沿同步复位。计数器在 TCR[1] 恢复之前保持复位状态。	0
31:2	-	保留, 用户, 软件不应写 1 到保留位。从保留位读出的值未定义。	NA

15.7.3 定时器计数器 (TMR16B0TC – 地址 0x4000 C008 , TMR16B1TC – 地址 0x4001 0008)

当预分频计数器 PC 到达其计数值时, 16 定时器计数器 TC 加 1, 如果 TC 在到达计数上限之前没有被复位, 则它将一直计数到 0x0000FFFF 然后翻转到 0x00000000. 这个事件不会产生一个中断, 但是如果需要, 匹配寄存器用于侦测溢出。

Table 219: 定时器计数器寄存器 (TMR16B0TC, 地址 0x4000 C008 , TMR16B1TC 0x4001 0008) 位描述

位	符号	描述	复位值
15:0	TC	定时器计数器的值	0
31:16	-	保留	-

15.7.4 预分频寄存器 (TMR16B0PR – 地址 0x4000 C00C, TMR16B1PR – 地址 0x4001 000C)

16 位的预分频寄存器为预分频计数器指定了一个最大值。.

Table 220: 预分频寄存器 (TMR16B0PR, 地址 0x4000 C00C 和 TMR16B1PR 0x4001 000C) 位描述

位	符号	描述	复位值
15:0	PR	预分频最大值	0
31:16	-	保留	-

15.7.5 预分频计数器寄存器 (TMR16B0PC - 地址 0x4000 C010 和 TMR16B1PC - 地址 0x4001 0010)

16 位的预分频计数器在控制 PCLK 进行分频之前，一些恒定的值写入到定时器计数器。这允许控制定时器分辨率和定时器溢出之前的最大时间之间的关系。每个 PCLK 时钟预分频计数器都会加 1。当它达到存储在预分频寄存器中的值时，定时器计数器就会加 1，并且在下一个时钟周期，PC 复位。这将导致：当 PR=0 时，每个 PCLK 时钟，TC 加 1，PR=1 时，每两个 PCLK 时钟 TC 加 1。

Table 221: 预分频计数器寄存器 (TMR16B0PC, 地址 0x4001 C010 和 TMR16B1PC 0x4000 0010) 位描述

位	符号	描述	复位值
15:0	PC	预分频计数器的值	0
31:16	-	保留	-

15.7.6 匹配控制寄存器 (TMR16B0MCR 和 TMR16B1MCR)

匹配控制寄存器用来控制当匹配寄存器和定时器计数器想匹配时执行哪些操作。位描述如 [Table 222](#) 所示。

Table 222. 匹配控制寄存器 (TMR16B0MCR - 地址 0x4000 C014 和 TMR16B1MCR - 地址 0x4001 0014) 位描述

位	符号	值	描述	复位值
0	MR0I		MR0 中断：当 MR0 与 TC 中的值相匹配时，产生一个中断。	0
		1	使能	
		0	禁能	
1	MR0R		MR0 复位：如果 MR0 与 TC 中的值相匹配时，TC 复位。	0
		1	使能	
		0	禁能	
2	MR0S		MR0 停止：如果 MR0 与 TC 值匹配时，TC 和 PC 停止，TCR[0] 清 0。	0
		1	使能	
		0	禁能	
3	MR1I		MR1 中断：当 MR1 与 TC 中的值相匹配时，产生一个中断。	0
		1	使能	
		0	禁能	

Table 222. 匹配控制寄存器 (TMR16B0MCR - 地址 0x4000 C014 和 TMR16B1MCR - 地址 0x4001 0014) 位描述 ...continued

位	符号	值	描述	复位值
4	MR1R		MR1 复位 : 如果 MR1 与 TC 中的值相匹配时, TC 复位	0
		1	使能	
		0	禁能	
5	MR1S		MR1 停止 : 如果 MR1 与 TC 值匹配时, TC 和 PC 停止, TCR[0] 清 0	0
		1	使能	
		0	禁能	
6	MR2I		MR2 中断 : 当 MR2 与 TC 中的值相匹配时, 产生一个中断	0
		1	使能	
		0	禁能	
7	MR2R		MR2 复位 : 如果 MR2 与 TC 中的值相匹配时, TC 复位 .	0
		1	使能	
		0	禁能	
8	MR2S		MR2 停止 : 如果 MR2 与 TC 值匹配时, TC 和 PC 停止, TCR[0] 清 0	0
		1	使能	
		0	禁能	
9	MR3I		MR3 中断 : 当 MR3 与 TC 中的值相匹配时, 产生一个中断	0
		1	使能	
		0	禁能	
10	MR3R		MR3 复位 : 如果 MR3 与 TC 中的值相匹配时, TC 复位	0
		1	使能	
		0	禁能	
11	MR3S		MR3 停止 : 如果 MR2 与 TC 值匹配时, TC 和 PC 停止, TCR[0] 清 0	0
		1	使能	
		0	禁能	
31:12	-		保留, 用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

15.7.7 匹配寄存器 (TMR16B0MR0/1/2/3 - 地址 0x4000 C018/1C/20/24 和 TMR16B1MR0/1/2/3 - 地址 0x4001 0018/1C/20/24)

匹配寄存器里的值会连续不断的和定时器计数器里的值比较。当两个值相等时, 会自动触发操作。这些操作可能是产生一个中断, 复位定时器计数器或是停止时钟。操作由 MCR 寄存器控制。

Table 223: 匹配寄存器 (TMR16B0MR0 to 3, 地址 0x4000 C018 到 24 和 TMR16B1MR0 到 3, 地址 0x4001 0018 to 24) 位描述

位	符号	描述	复位值
15:0	MATCH	定时器计数器匹配值	0
31:16	-	保留	-

15.7.8 捕获控制寄存器 (TMR16B0CCR 和 TMR16B1CCR)

捕获控制寄存器用来：当捕获事件发生时，是否装载计数器 / 定时器中的值，是否产生一个中断，捕获事件是发生在上升沿，下降沿还是双边沿。在下面的描述中，“n”代表定时器的标号，0 或 1。

Table 224. 捕获控制寄存器 (TMR16B0CCR - 地址 0x4000 C028 和 TMR16B1CCR - 地址 0x4001 0028) 位描述

位	符号	值	描述	复位值
0	CAP0RE		CT16Bn_CAP0 上升沿捕获: CT16Bn_CAP0 上 0 到 1 的跳变，会导致 TC 的内容被装载到 CR0 中。	0
		1	使能	
		0	禁能	
1	CAP0FE		CT16Bn_CAP0 下降沿捕获: CT16Bn_CAP0 上 1 到 0 的跳变，会导致 TC 的内容被装载到 CR0 中 0。	0
		1	使能	
		0	禁能	
2	CAP0I		CT16Bn_CAP0 事件中断: CT16Bn_CAP0 事件会导致 CR- 被装载，并会产生一个中断。 0	0
		1	使能	
		0	禁能	
31:3	-	-	保留，用软软件不应写 1 到保留位，从保留位读出的值未定义。	NA

15.7.9 捕获寄存器 (CT16B0CR0 - 地址 0x4000 C02C 和 CT16B1CR0 - 地址 0x4001 002C)

每个捕获寄存器都与一个设备引脚相连并且当一个特定事件发生在这个引脚上时，相应的捕获寄存器会装载计数器 / 定时器的值。捕获控制寄存器设置：是否允许捕获功能，捕获事件是发生在上升沿，下降沿，还是双边沿。

Table 225: 捕获寄存器(TMR16B0CR0, 地址0x4000 C02C 和TMR16B1CR0, 地址0x4001 002C) 位描述

位	符号	描述	复位值
15:0	CAP	定时器计数器捕获值	0
31:16	-	保留	-

15.7.10 外部匹配寄存器 (TMR16B0EMR 和 TMR16B1EMR)

外部匹配寄存器提供对外部匹配引脚 CT16B0_MAT[2:0] and CT16B1_MAT[1:0] 的控制，并反映他们的状态。

在 PWMCON 寄存器中，如果匹配输出被匹配成 PWM 输出 (Section 15.7.12), 那么外部匹配寄存器由 PWM 规则决定。(Section 15.7.13 “Rules for single edge controlled PWM outputs” on page 263)。

Table 226. 外部匹配寄存器 (TMR16B0EMR - 地址 0x4000 C03C 和 TMR16B1EMR - 地址 0x4001 003C) 位描述

位	符号	值	描述	复位值
0	EM0		外部匹配 0: 无论这路输出是否关联到其引脚，该位都会反映 CT16B0_MAT0/CT16B1_MAT0 的输出状态. 当 MR0 与 TC 发生匹配时，该位可为低电平，高电平，或不执行任何操作。位 EMR[5:4] 控制该输出功能。如果在 IOCON 寄存器（0= 低，1= 高）选择了匹配功能，那么该位就会驱动 CT16B0_MAT0/CT16B1_MAT0 引脚。	0
1	EM1		外部匹配 1: 无论这路输出是否关联到其引脚，该位都会反映 CT16B0_MAT1/CT16B1_MAT1 的输出状态. 当 MR1 与 TC 发生匹配时，该位可为低电平，高电平，或不执行任何操作。位 EMR[7:6] 控制该输出功能。如果在 IOCON 寄存器（0= 低，1= 高）选择了匹配功能，那么该位就会驱动 CT16B0_MAT1/CT16B1_MAT1 引脚。	0
2	EM2		外部匹配 2: 无论这路输出是否关联到其引脚，该位都会反映 CT16B0_MAT2/CT16B1_MAT2 的输出状态. 当 MR2 与 TC 发生匹配时，该位可为低电平，高电平，或不执行任何操作。位 EMR[9:8] 控制该输出功能。如果在 IOCON 寄存器（0= 低，1= 高）选择了匹配功能，那么该位就会驱动 CT16B0_MAT2/CT16B1_MAT2 引脚。	0
3	EM3		外部匹配 3: 该位反映了匹配通道 3 的状态。当 MR3 与 TC 发生匹配时，该位可为低电平，高电平，或不执行任何操作。位 EMR[11:10] 控制该功能输出。两个定时器中都没有与这个 通道相关联的输出引脚。	0
5:4	EMC0		外部匹配控制 0. 决定外部匹配 0 的功能 :.	00
		0x0	不做任何操作	
		0x1	将对应的外部匹配位 / 输出清 0 （如果连接到引脚，则 CT16Bn_MATm 引脚为低电平）。	
		0x2	将对应的外部匹配位 / 输出置 1 （如果连接到引脚，则 CT16Bn_MATm 引脚为高电平 ）。	
		0x3	将对应的外部匹配位输出翻转	
7:6	EMC1		外部匹配控制 1. 决定外部匹配 1 的功能:	00
		0x0	不做任何操作	
		0x1	将对应的外部匹配位 / 输出清 0 （如果连接到引脚，则 CT16Bn_MATm 引脚为低电平）。	
		0x2	将对应的外部匹配位 / 输出置 1 （如果连接到引脚，则 CT16Bn_MATm 引脚为高电平 ）。	
		0x3	将对应的外部匹配位 / 输出翻转。	

Table 226. 外部匹配寄存器 (TMR16B0EMR - 地址 0x4000 C03C 和 TMR16B1EMR - 地址 0x4001 003C) 位描述

位	符号	值	描述	复位值
9:8	EMC2		外部匹配控制 2. 决定外部匹配 2 的功能:	00
		0x0	不做任何操作	
		0x1	将对应的外部匹配位 / 输出清 0 (如果连接到引脚, 则 CT16Bn_MATm 引脚为低电平)	
		0x2	将对应的外部匹配位 / 输出置 1 (如果连接到引脚, 则 CT16Bn_MATm 引脚为高电平)	
		0x3	将对应的外部匹配位 / 输出翻转	
11:10	EMC3		外部匹配控制 3. 决定外部匹配 3 的功能:	00
		0x0	不做任何操作	
		0x1	将对应的外部匹配位 / 输出清 0 (如果连接到引脚, 则 CT16Bn_MATm 引脚为低电平)	
		0x2	将对应的外部匹配位 / 输出置 1 (如果连接到引脚, 则 CT16Bn_MATm 引脚为高电平)	
		0x3	将对应的外部匹配位输出翻转	
31:12	-		保留, 用户软件不应写 1 到保留位, 从保留位读出的值未定义	NA

Table 227. 外部匹配控制

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	功能
00	不执行任何操作
01	将对应的外部匹配位 / 输出清 0 (如果连接到引脚, 则 CT16Bn_MATm 引脚为低电平)
10	将对应的外部匹配位 / 输出置 1 (如果连接到引脚, 则 CT16Bn_MATm 引脚为高电平).
11	将对应的外部匹配位 / 输出翻转

15.7.11 计数器控制寄存器 (TMR16B0CTCR 和 TMR16B1CTCR)

计数控制寄存器 (CTCR) 用来选择是定时器模式还是计数器模式, 在计数器模式下选择计数的引脚和边沿。

当选择工作在计数器模式下, 在每个 PCLK 时钟上升沿对 CAP 输入 (通过 CTCR 的位 3:2) 进行采样。再完成两个连续的采样后, 可以识别以下 4 个时间之一: 上升沿, 下降沿, 任一边沿或在选定的 CAP 输入上无电平变化。只要识别的事件与 CTCR 寄存器为 [1:0] 选定的事件对应, 定时 / 计数器寄存器将递增 1。

计数器的外部时钟源的有效操作有一些限制。由于 2 个连续上升沿用来是被 CAP 选择输入的一个边沿, 所以 CAP 输入的频率不能大于 1/2 个 PCLK 时钟。因此, 这种情况下同一 CAP 输入的高 / 低电平的持续时间不能小于 1/(2XPCLK)。

Table 228. 计数器控制寄存器 (TMR16B0CTCR - 地址 0x4000 C070 和 TMR16B1CTCR - 地址 0x4001 0070) 位描述

位	符号	值	描述	复位值
1:0	CTM		定时器 / 计数器模式 该位域选择触发定时器的预分频计数器（PC）递增、清除 PC 和定时器（TC）递增的 PCLK 边沿。	00
		0x0	定时器模式：每个 PCLK 的上升沿计数	
		0x1	计数器模式：在位 3:2 所选择的 CAP 输入的上升沿，TC 递增。	
		0x2	计数器模式：在位 3:2 所选择的 CAP 输入的下降沿，TC 递增。	
		0x3	计数器模式：在位 3:2 所选择的 CAP 输入的下降沿，TC 递增。	
3:2	CIS		计数输入选择。在计数器模式下（当为 [1:0] 不是 00），这些位用来选择对哪个 CAP 引脚进行采样计时。 注意： 如果在 CTCR 寄存器中选择了计数器模式，捕获控制寄存器（CCR）的位 [2:0] 必须位 000	00
		0x0	CT16Bn_CAP0	
		0x1	保留	
		0x2	保留	
		0x0	保留	
31:4	-	-	保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

15.7.12 PWM 控制寄存器 (TMR16B0PWMC 和 TMR16B1PWMC)

PWM控制寄存器PWMC用于将匹配输出配置为PWM输出。每个匹配输出都可独立地配置为PWM输出或匹配输出，而这些功能由外部匹配寄存器（EMR）来控制。

对于定时器 0 来说，3 个 CT16B0_MAT[2:0] 输出可以可以被选择为单边沿控制 PWM 输出，而对于定时器 1 来说，只有两个 CT16B1_Mat[1:0] 输出可以被选择位单边沿控制 PWM 输出。另外一个匹配寄存器用来决定 PWM 周期长度。当任意一个匹配寄存器发生匹配时，PWM 输出配置为高电平，定时器通过那个被配置为 PWM 输出的周期长度的匹配寄存器来复位。当定时器被复位位 0 时，所有被配置为 WM 输出的高电平匹配输出被清除。

Table 229. PWM 控制寄存器 (TMR16B0PWMC - 地址 0x4000 C074 和 TMR16B1PWMC- 地址 0x4001 0074) 位描述

位	符号	值	描述	复位值
0	PWMEN0		PWM 通道 0 使能	0
		0	CT16Bn_MAT0 被 EM0 控制	
		1	允许 CT16Bn_MAT0 使用 PWM 模式	

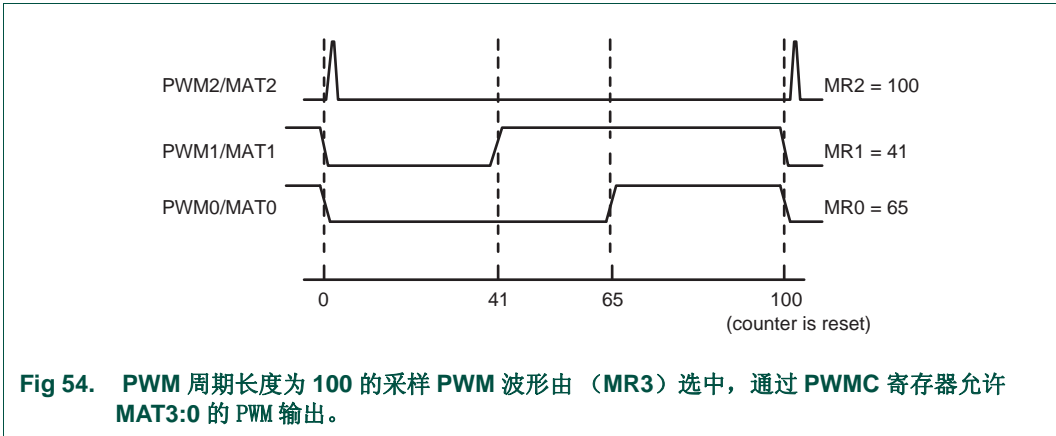
Table 229. PWM 控制寄存器 (TMR16B0PWC - 地址 0x4000 C074 和 TMR16B1PWC- 地址 0x4001 0074) 位描述

位	符号	值	描述	复位值
1	PWMEN1		PWM 通道 1 使能	0
		0	CT16Bn_MAT1 被 EM1 控制	
		1	允许 CT16Bn_MAT1 使用 PWM 模式。	
2	PWMEN2		PWM 通道 2 使能	0
		0	匹配通道 2 或 CT16B0_MAT2 被 EM2 控制。在定时器 1 上，匹配通道 2 没有与引脚相连。	
		1	允许通道 2 或引脚 CT16B0_MAT2 使用 PWM 模式。	
3	PWMEN3		PWM 通道 3 使能	0
			注意： 建议使用匹配通道 3 去设置 PWM 周期，因为它没有引脚输出。	
		0	匹配通道 3 被 EM3 控制	
		1	允许 MAT3 使用 PWM 模式。	
31:4	-		保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

15.7.13 单边沿控制 PWM 输出的规则如下：

- 1. 所有单边沿控制 PWM 输出在每个 PWM 周期开始时都置为低电平（定时器设置为 0），除非他们的匹配值等于 0。
- 2. 每个 PWM 输出都在它的匹配值到达是置为高电平，在没有达到（也就是匹配值大于 PWM 周期的长度）时为低电平。
- 3. 如果匹配值大于写入匹配寄存器中 PWM 周期长度，同时 PWM 信号已经是高电平，那么 PWM 信号会在下一个 PWM 周期开始时被清除。
- 4. 如果匹配寄存器中的值与定时器复位值相同 (PWM周期的长度)，那么 PWM 输出就会在下一个时钟滴答复位为低。因此， PWM 输出一般包括一个时钟滴答宽度的正脉冲，其宽度由 PWM 周期长度决定（也就是定时器的重装载值）。
- 5. 如果匹配寄存器被设置为 0，那么 PWM 输出会在定时器第一次回到 0 时变成高电平，而且持续保持高电平。

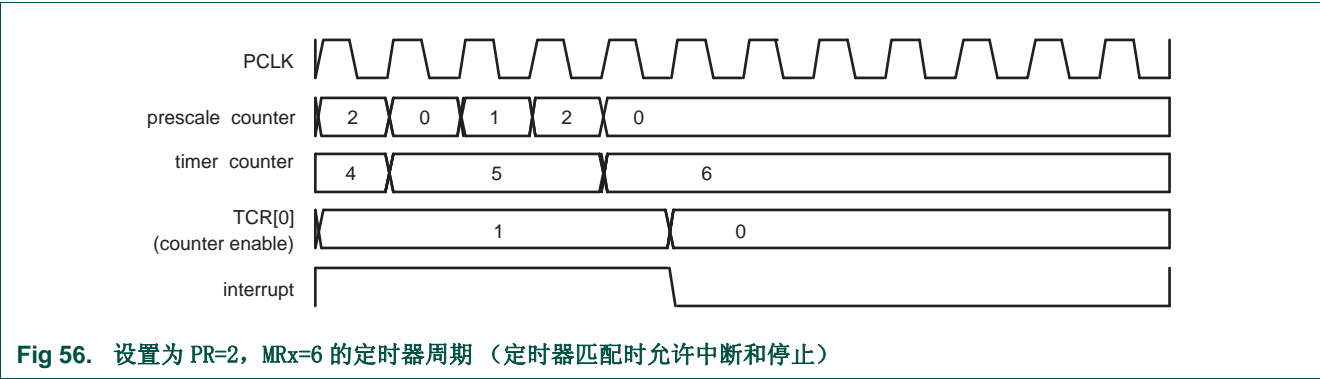
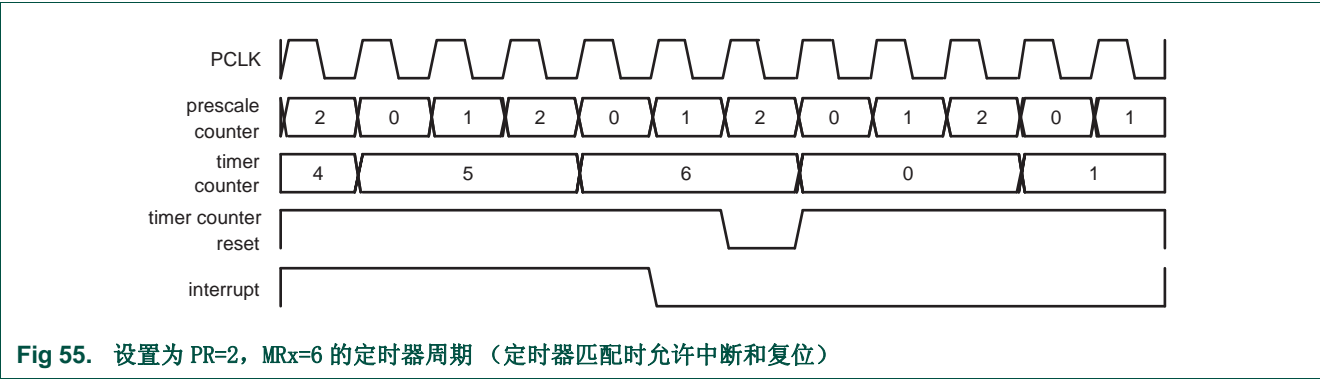
注意：当匹配输出被用于 PWM 输出时，在匹配寄存器中的定时器复位位（MRnR）和定时器停止位（MRnS）就必须设置为 0.，除了用于设置 PWM 周期长度的匹配寄存器。对于该寄存器，如果设这 MRnR 位为 1，当定时器值与相应的匹配寄存器值发生匹配时，定时器允许被复位。



15.8 定时器操作实例

Figure 55 定时器配置为：在匹配发生时，定时器产生中断；预分频器设置为 2，匹配寄存器设置为 6。在匹配发生的定时器周期结束时，定时器计数器值被复位，这样就使匹配值具有完整长度的周期。在定时器达到匹配值下一个时钟周期内，中断产生。

Figure 56 定时器配置为：在匹配发生时，定时器停止并产生中断；预分频器设置为 2，匹配寄存器设置为 6。在定时器达到匹配值后的下一个时钟周期内，TCR 中的定时器允许位被清 0 并产生中断。



15.9 Architecture

计数 / 定时器 0 和计数 / 定时器 1 的方框图 [Figure 57](#).

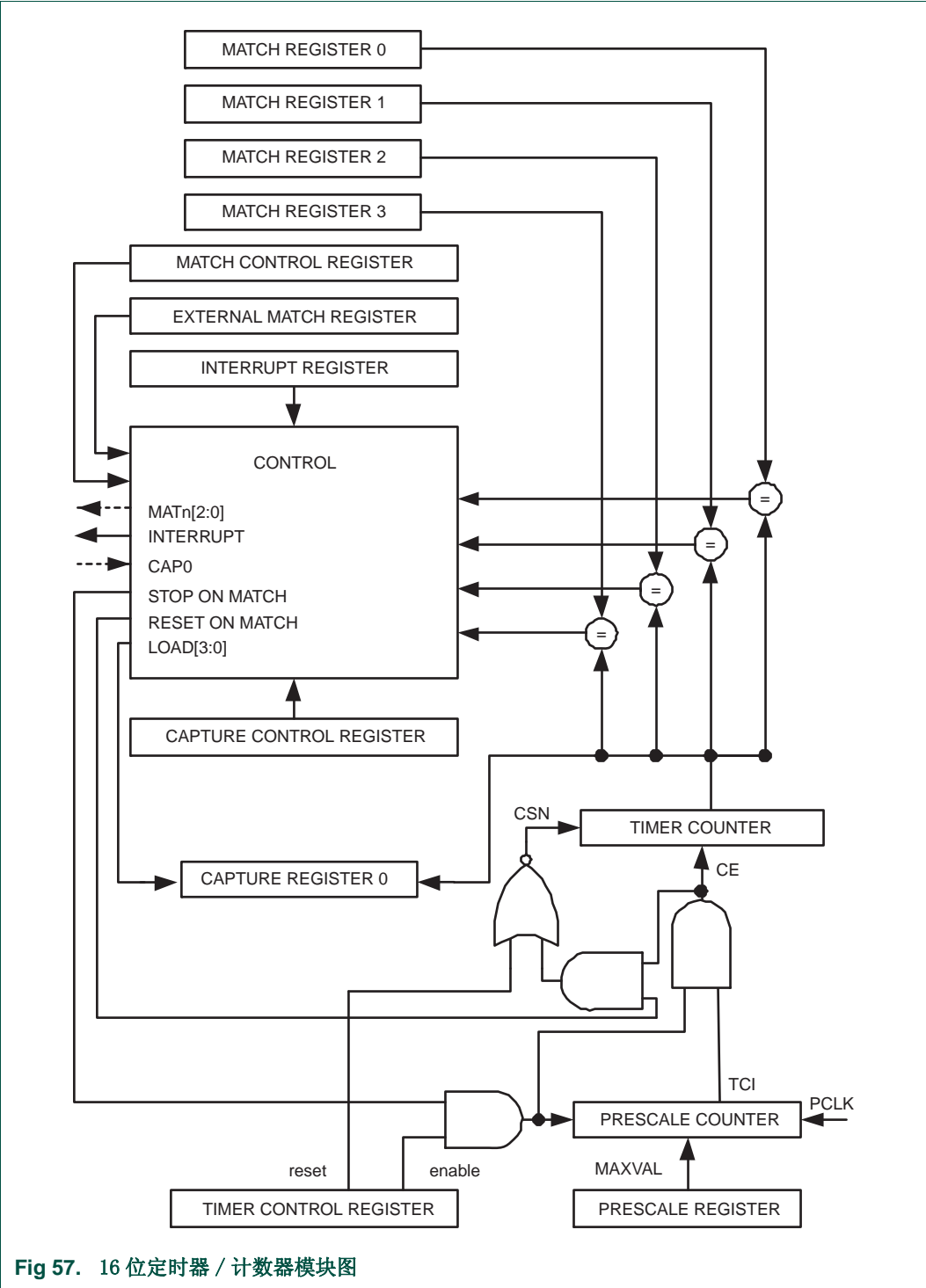


Fig 57. 16 位定时器 / 计数器模块图

16.1 如何阅读本章

所有 LPC111x 和 LPC11Cx 处理器的 32 位定时器完全相同。

16.2 基本配置

通过以下寄存器对 CT32B0/1 进行配置：

1. 引脚：CT32B0/1 引脚必须由 IOCONFIG 寄存器进行配置 ([Section 7.4](#))。
2. 电源和外设时钟：在 SYSAHBCLKCTRL 寄存器，设置第 9 位和第 10 位 ([Table 21](#))。外设时钟 (PCLK) 由系统时钟提供 (见 [Table 20](#))。

16.3 特性

- 2 个 32 位计数器 / 定时器，带可编程 32 位预分频器。
- 计数器或定时器操作
- 具有 1 路 32 位的捕获通道。当输入信号跳变时可捕获定时器的当前值。也可以选择使捕获事件产生中断。
- 4 个 32 位匹配寄存器允许：
 - 匹配时持续比较，可选择产生中断。
 - 匹配时停止定时器，可选择产生中断。
 - 匹配时复位定时器，可选择产生中断。
- 4 个外部输出，其对应的匹配寄存器具有以下功能：
 - 匹配时设置低电平。
 - 匹配时设置高电平。
 - 匹配时翻转。
 - 匹配时无动作。
- 对于每个定时器，多达四个可配置为 PWM 的匹配寄存，允许使用多达 3 个匹配输出作为单边沿控制的 PWM 输出。

16.4 应用

- 用于对内部事件计数的间隔定时器
- 通过捕获输入的脉冲解调
- 自由运行定时器
- 通过匹配输出的脉冲调制

16.5 描述

每个计数器 / 定时器都可以对外设时钟 (PCLK) 或外部提供的时钟周期进行计数，可选择产生中断，或根据 4 个匹配寄存器的设定，在到达指定的定时值时执行其他动作。外设时钟由系统时钟提供 (见 [Figure 3](#)) 。每个计数器 / 定时器都包括 1 个捕获输入，用于在输入信号发生跳变时捕获定时器值，并可选择产生中断。

在 PWM 模式，3 个匹配寄存器可以在匹配输出引脚上提供单边沿控制 PWM 输出，另一个匹配寄存器用作控制 PWM 周期长度。

注意：32 位计数器 / 定时器 0 (CT32B0) 与 32 位计数器 / 定时器 1 (CT32B1)，除了外设基地址不同之外，功能上都是相同的。

16.6 引脚描述

[Table 230](#) 所列每个计数器 / 定时器相关引脚的简要描述。

Table 230. 计数器 / 定时器引脚描述

引脚	类型	描述
CT32B0_CAP0 CT32B1_CAP0	输入	捕获信号： 捕获引脚的跳变可配置为将计数器 / 定时器值装入捕获寄存器，并可选择产生一个中断，计数器 / 定时器可选择一个捕获信号作为时钟源来代替 PCLK 分频时钟。详见 Section 16.7.11 “Count Control Register (TMR32B0CTCR and TMR32B1TCR)” on page 275 。
CT32B0_MAT[3:0] CT32B1_MAT[3:0]	输出	CT32B0/1 的外部匹配输出： 当匹配寄存器 TMR32B0/1 (MR3:0) 等于定时器计数器 (TC)，其输出可翻转，变为低电平、变为高电平或不变。外部匹配寄存器 (EMR) 和 PWM 控制寄存器 (PWMCON) 控制该输出的功能。

16.7 寄存器描述

32 位计数器 / 定时器 0 包含的寄存器如 [Table 231](#) 所示，以及 32 位计数器 / 定时器 1 包含的寄存器如 [Table 232](#) 所示。更多细节如下所示。

Table 231. 寄存器概览：32 位计数器 / 定时器 0 CT32B0 (基地址 0x4001 4000)

名称	访问方式	地址偏移	描述	复位值 U
TMR32B0IR	R/W	0x000	中断寄存器 (IR)。可以写 IR 来清除中断。可读取 IR 来识别五个中哪个中断源被挂起。	0
TMR32B0TCR	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 TCR 寄存器可以禁止或者复位定时器计数器。	0
TMR32B0TC	R/W	0x008	定时器计数器 (TC)。32 位 TC 每经 PR+1 个 PCLK 周期递增。通过 TCR 寄存器控制 TC 。	0

Table 231. 寄存器概览: 32 位计数器 / 定时器 0 CT32B0 (基地址 0x4001 4000) ...continued

名称	访问方式	地址偏移	描述	复位值 [1]
TMR32B0PR	R/W	0x00C	预分频寄存器 (PR)。当预分频计数器等于该寄存器值时候下个时钟周期会递增 TC 并清除 PC 的值。	0
TMR32B0PC	R/W	0x010	预分频寄存器 (PC)。32 位 PC 是一个从 0 到 PR 寄存器值递增的计数器。当到达 PR 寄存器值的时候, TC 的值会递增且 PC 的值会被清除通过总线接口可观测和控制 PC (Prescale Counter)。	0
TMR32B0MCR	R/W	0x014	匹配控制寄存器 (MCR)。MCR 用于控制在匹配发生时是否产生中断或复位 TC。	0
TMR32B0MR0	R/W	0x018	匹配寄存器 0 (MR0)。MR0 可通过 MCR 设定为在每次 MR0 匹配 TC 时复位 TC、停止 TC 和 PC、和 / 或产生中断。	0
TMR32B0MR1	R/W	0x01C	匹配寄存器 1 (MR1)。参阅 MR0 的描述。	0
TMR32B0MR2	R/W	0x020	匹配寄存器 2 (MR2)。参阅 MR0 的描述。	0
TMR32B0MR3	R/W	0x024	匹配寄存器 3 (MR3)。参阅 MR0 的描述。	0
TMR32B0CCR	R/W	0x028	捕获控制寄存器 (CCR)。CCR 控制在哪个捕获输入边沿装在捕获寄存器以及在发生捕获时是否产生中断。	0
TMR32B0CR0	RO	0x02C	捕获寄存器 0 (CR0)。当在 CT32B0_CAP0 输入上有事件时, CR0 装载 TC 值。	0
TMR32B0EMR	R/W	0x03C	外部匹配寄存器 (EMR)。控制匹配功能和外部匹配引脚 CT32B0_MAT[3:0]。	0
-	-	0x040 - 0x06C	保留	-
TMR32B0CTCR	R/W	0x070	计数控制寄存器 (CTCR)。CTCR 选择定时器或计数器模式, 且在计数器模式下选择计数的信号和边沿。	0
TMR32B0PWC	R/W	0x074	PWM 控制寄存器 (PWMCON)。PWMCON 允许外部匹配引脚 CT32B0_MAT[3:0] 的 PWM 模式。	0

[1] 复位值仅指使用位中数据, 不包括保留位中的内容。

Table 232. 寄存器概览: 32 位计数器 / 定时器 1 CT32B1 (基地址 0x4001 8000)

名称	访问方式	地址偏移	描述	复位值 [1]
TMR32B1IR	R/W	0x000	中断寄存器 (IR)。可以写 IR 来清除中断。可读取 IR 来识别五个中哪个中断源被挂起。	0
TMR32B1TCR	R/W	0x004	定时器控制寄存器 (TCR)。TCR 用于控制定时器计数器的功能。通过 TCR 可以禁止或复位定时器计数器。	0
TMR32B1TC	R/W	0x008	定时器计数器 (TC)。32 位 TC 每经 PR+1 个 PCLK 周期递增加 1。TC 受 TCR 的控制。	0
TMR32B1PR	R/W	0x00C	预分频寄存器 (PR)。当 PC 等于 PR 寄存器的值时候, TC 的值会递增且 PC 的值被清除。	0
TMR32B1PC	R/W	0x010	预分频计数器 (PC)。32 位 PC 是一个从 0 到 PR 寄存器值递增的计数器。当到达 PR 寄存器的值时候, TC 的值会递增且 PC 的值会被清除。通过总线接口可观测和控制 PC。	0

Table 232. 寄存器概览: 32 位计数器 / 定时器 1 CT32B1 (基地址 0x4001 8000) ...continued

名称	访问方式	地址偏移	描述	复位值 ^[1]
TMR32B1MCR	R/W	0x014	匹配控制寄存器 (MCR)。MCR 用于控制在匹配发生时是否产生中断或复位 TC。	0
TMR32B1MR0	R/W	0x018	匹配寄存器 0 (MR0)。MR0 可通过 MCR 设定为在每次 MR0 匹配时 TC 时复位 TC、停止 TC 和 PC、和 / 或产生中断。	0
TMR32B1MR1	R/W	0x01C	匹配寄存器 1 (MR1)。参阅 MR0 的描述。	0
TMR32B1MR2	R/W	0x020	匹配寄存器 2 (MR2)。参阅 MR0 的描述。	0
TMR32B1MR3	R/W	0x024	匹配寄存器 3 (MR3)。参阅 MR0 的描述。	0
TMR32B1CCR	R/W	0x028	捕获控制寄存器 (CCR)。CCR 控制在哪个捕获输入边沿装载捕获寄存器以及在发生捕获时是否产生中断。	0
TMR32B1CR0	RO	0x02C	捕获寄存器 0 (CR0)。当 CT32B1_CAP0 输入上有事件时，CR0 装载 TC 值。	0
TMR32B1EMR	R/W	0x03C	外部匹配寄存器 (EMR)。控制匹配功能和外部匹配引脚 CT32B1_MAT[3:0]。	0
-	-	0x040 - 0x06C	保留	-
TMR32B1CTCR	R/W	0x070	计数控制寄存器 (CTCR)。CTCR 选择定时器或计数器模式，且在计数器模式下选择计数的信号和边沿。	0
TMR32B1PWMC	R/W	0x074	PWM 控制寄存器 (PWMCON)。PWMCON 允许外部匹配引脚 CT32B1_MAT[3:0] 的 PWM 模式。	0

[1] 复位值仅指使用位中的数据，不包括保留位中的内容。

16.7.1 中断寄存器 (TMR32B0IR 和 TMR32B1IR)

中断寄存器 (IR) 包含四个中断匹配位和一个中断捕获位。如果一个中断产生，则 IR 中相应的位会被置为高电平，否则就为低电平。写 “1” 到 IR 中相应的位会被复位中断，写 “0” 则没有影响。

Table 233: 中断寄存器 (TMR32B0IR - 地址 0x4001 4000 和 TMR32B1IR - 地址 0x4001 8000) 位域描述

位	符号	描述	复位值
0	MR0 Interrupt	匹配通道 0 的中断标志位。	0
1	MR1 Interrupt	匹配通道 1 的中断标志位。	0
2	MR2 Interrupt	匹配通道 2 的中断标志位。	0
3	MR3 Interrupt	匹配通道 3 的中断标志位。	0
4	CR0 Interrupt	捕获通道 0 事件的中断标志位。	0
31:5	-	保留	-

16.7.2 定时器控制寄存器 (TMR32B0TCR and TMR32B1TCR)

定时器控制寄存器 (TCR) 是用来控制计数器 / 定时器的操作。

Table 234: 定时器控制寄存器 (TMR32B0TCR - 地址 0x4001 4004 和 TMR32B1TCR - 地址 0x4001 8004) 位域描述

位	符号	描述	复位值
0	CEn	为 1 时，定时器计数器和预分频计数器被允许计数。为 0 时则计数器被禁止。	0
1	CRst	为 1 时，定时器计数器和预分频计数器在下一个 PCLK 的上升沿同步复位。计数器在 TCR[1] 回复为 0 之前保持复位状态。	0
31:2	-	保留，用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

16.7.3 定时器计数器 (TMR32B0TC - 地址 0x4001 4008 和 TMR32B1TC - 地址 0x4001 8008)

当预分频计数器到达其计数值时，32 位定时器计数器加 1。如果 TC 在到达计数上限之前没有被复位，它将一直计数到 0xFFFF FFFF 然后翻转到 0x0000 0000 。该事件不会产生中断。如果需要，可用匹配寄存器检测溢出。

Table 235: 定时器计数器寄存器 (TMR32B0TC, 地址 0x4001 4008 和 TMR32B1TC 0x4001 8008) 位域描述

位	符号	描述	复位值
31:0	TC	定时器值。	0

16.7.4 预分频寄存器 (TMR32B0PR - 地址 0x4001 400C 和 TMR32B1PR - 地址 0x4001 800C)

32 位预分频寄存器指定预分频计数器最大值。

Table 236: 预分频寄存器 (TMR32B0PR, 地址 0x4001 400C 和 TMR32B1PR 0x4001 800C) 位域描述

位	符号	描述	复位值
31:0	PR	预分频值	0

16.7.5 预分频计数器寄存器 (TMR32B0PC - 地址 0x4001 4010 和 TMR32B1PC - 地址 0x4001 8010)

32 位预分频计数器控制着 PCLK 分频系数。在被用于定时器计数器之前，分频系数是一些常量决定。它可控制定时器溢出之前最大时间和定时器分辨率之间的关系。每个 PCLK 时钟会让预分频计数器递增。当它到达了存储在预分频寄存器中的值时候，定时器计数器也会递增，且预分频计数器会在下一个 PCLK 时钟复位。当 PR = 0 时，每个 PCLK 会使 TC 递增 1；当 PR = 1 时，每 2 个 PCLK 会使 TC 递增 1，以此类推。

Table 237: 预分频计数器寄存器 (TMR32B0PC, 地址 0x4001 4010 和 TMR32B1PC 0x4001 8010) 位描述

位	符号	描述	复位值
31:0	PC	预分频计数值。	0

16.7.6 匹配控制寄存器 (TMR32B0MCR 和 TMR32B1MCR)

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应匹配时所执行的操作。各位的功能如 [Table 238](#)。

Table 238: 匹配控制寄存器 (TMR32B0MCR - 地址 0x4001 4014 和 TMR32B1MCR - 地址 0x4001 8014) 位域描述

位	符号	值	描述	复位值
0	MR0I		MR0 中断: 当 MR0 与 TC 的值匹配时产生一个中断。	0
		1	使能	
		0	禁止	
1	MR0R		MR0 复位: 如果 MR0 与 TC 的值匹配将使 TC 复位。	0
		1	使能	
		0	禁止	
2	MR0S		MR0 停止: 如果 MR0 与 TC 的值匹配将使 TC 和 PC 停止, TCR[0] 清零。	0
		1	使能	
		0	禁止	
3	MR1I		MR1 中断: 如果 MR1 与 TC 的值匹配将产生中断。	0
		1	使能	
		0	禁止	
4	MR1R		MR1 复位: 如果 MR1 与 TC 的值匹配将使 TC 复位。	0
		1	使能	
		0	禁止	
5	MR1S		MR1 停止: 如果 MR1 与 TC 的值匹配将使 TC 和 PC 停止, TCR[0] 清零。	0
		1	使能	
		0	禁止	
6	MR2I		MR2 中断: 如果 MR1 与 TC 的值匹配将产生中断。	0
		1	使能	
		0	禁止	
7	MR2R		MR2 复位: 如果 MR1 与 TC 的值匹配将使 TC 复位。	0
		1	使能	
		0	禁止	
8	MR2S		MR2 停止: 如果 MR0 与 TC 的值匹配将使 TC 和 PC 停止, TCR[0] 清零。	0
		1	使能	
		0	禁止	
9	MR3I		MR3 中断: 如果 MR1 与 TC 的值匹配将产生中断。	0
		1	使能	
		0	禁止	

Table 238: 匹配控制寄存器 (TMR32B0MCR - 地址 0x4001 4014 和 TMR32B1MCR - 地址 0x4001 8014) 位域描述

位	符号	值	描述	复位值
10	MR3R		MR3 复位: 如果 MR1 与 TC 的值匹配将使 TC 复位。	0
		1	使能	
		0	禁止	
11	MR3S		MR3 停止: 如果 MR0 与 TC 的值匹配将使 TC 和 PC 停止, TCR[0] 清零。	0
		1	使能	
		0	禁止	
31:12	-		保留, 用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

16.7.7 匹配寄存器 (TMR32B0MR0/1/2/3 - 地址 0x4001 4018/1C/20/24 和 TMR32B1MR0/1/2/3 地址 0x4001 8018/1C/20/24)

匹配寄存器值连续与定时器计数值相比较。当两个值相等时自动触发相应动作。这些动作包括产生中断、复位定时器 / 计数器或停止定时器。所执行的动作由 MCR 寄存器的设定来控制。

Table 239: 匹配寄存器 (TMR32B0MR0 to 3, 地址 0x4001 4018 to 24 和 TMR32B1MR0 to 3, 地址 0x4001 8018 to 24) 位域描述

位	符号	描述	复位值
31:0	MATCH	定时器计数器匹配值	0

16.7.8 捕获控制寄存器 (TMR32B0CCR 和 TMR32B1CCR)

当捕获事件发生时, 捕获控制寄存器用来控制捕获寄存器是否装载在计数器 / 定时器中的值, 以及是否产生中断。同时设置下降位和上升位是一个有效配置, 这将导致产生双边沿捕获事件。在下面的描述中 “n” 为定时器号, 0 或 1。

Table 240: 捕获控制寄存器 (TMR32B0CCR - 地址 0x4001 4028 和 TMR32B1CCR - 地址 0x4001 8028) 位域描述

位	符号	值	描述	复位值
0	CAP0RE		CT32Bn_CAP0 上升沿捕获: CT32Bn_CAP0 上 0 到 1 的跳变, 会导致 TC 的内容装载 0 到 CR0 中。	0
		1	使能	
		0	禁止	
1	CAP0FE		CT32Bn_CAP0 上升沿捕获: CT32Bn_CAP0 上 1 到 0 的跳变, 会导致 TC 的内容装载 0 到 CR0 中。	0
		1	使能	
		0	禁止	

Table 240: 捕获控制寄存器 (TMR32B0CCR - 地址 0x4001 4028 和 TMR32B1CCR - 地址 0x4001 8028) 位域描述

位	符号	值	描述	复位值
2	CAP0I		CT32Bn_CAP0 事件中断: CT32Bn_CAP0 的事件所导致的 CR0 装载, 并产生一个中断。	0
		1	使能	
		0	禁止	
31:3	-		保留, 用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

16.7.9 捕获寄存器 (TMR32B0CR0 - 地址 0x4001 402C 和 TMR32B1CR0 - 地址 0x4001 802C)

每个捕获寄存器都与一个设备引脚相联系。当一个特殊事件发生在该引脚上时, 捕获寄存器会转载计数器 / 定时器的值。捕获控制寄存器上的设置决定, 是否允许捕获功能、捕获事件是发生在上升沿, 下降沿还是双边沿。

Table 241: 捕获寄存器 (TMR32B0CR0, 地址 0x4001 402C 和 TMR32B1CR0, 地址 0x4001 802C) 位域描述

位	符号	描述	复位值
31:0	CAP	定时器计数器捕获值	0

16.7.10 外部匹配寄存器 (TMR32B0EMR 和 TMR32B1EMR)

外部匹配寄存器提供对外部匹配引脚 CAP32Bn_MAT[3:0] 的控制, 并反映它们的状态。

如将匹配输出配置成 PWM 输出, 外部匹配寄存器的功能是由 PWM 决定的。
([Section 16.7.13 “Rules for single edge controlled PWM outputs” on page 277](#))。

Table 242: 外部匹配寄存器 (TMR32B0EMR - 地址 0x4001 403C 和 TMR32B1EMR - 地址 0x4001 803C) 位域描述

位	符号	值	描述	复位值
0	EM0		外部匹配 0。无论这路输出是否关联到其引脚，该位都会反映 CT32Bn_MAT0。当 MR0 与 TC 发生匹配时候，该位可翻转，为低电平，为高电平或不执行任何动作。位 EMR[5:4] 控制该输出的功能。如果在 IOCON 寄存器 (0 = 低, 1 = 高) 选择了匹配功能，那么该位就会驱动引脚 CT32B0_MAT0/CT16B1_MAT0。	0
1	EM1		外部匹配 1。无论这路输出是否关联到其引脚，该位都会反映 CT32Bn_MAT1。当 MR0 与 TC 发生匹配时候，该位可翻转，为低电平，为高电平或不执行任何动作。位 EMR[5:4] 控制该输出的功能。如果在 IOCON 寄存器 (0 = 低, 1 = 高) 选择了匹配功能，那么该位就会驱动引脚 CT32B0_MAT0/CT16B1_MAT1。	0
2	EM2		外部匹配 2。无论这路输出是否关联到其引脚，该位都会反映 CT32Bn_MAT2。当 MR0 与 TC 发生匹配时候，该位可翻转，为低电平，为高电平或不执行任何动作。位 EMR[5:4] 控制该输出的功能。如果在 IOCON 寄存器 (0 = 低, 1 = 高) 选择了匹配功能，那么该位就会驱动引脚 CT32B0_MAT0/CT16B1_MAT2。	0
3	EM3		外部匹配 3。无论这路输出是否关联到其引脚，该位都会反映 CT32Bn_MAT3。当 MR0 与 TC 发生匹配时候，该位可翻转，为低电平，为高电平或不执行任何动作。位 EMR[5:4] 控制该输出的功能。如果在 IOCON 寄存器 (0 = 低, 1 = 高) 选择了匹配功能，那么该位就会驱动引脚 CT32B0_MAT0/CT16B1_MAT3。	0
5:4	EMC0		外部匹配控制 0。决定外部匹配 0 的功能。	00
		0x0	不执行任何动作。	
		0x1	将对应的外部匹配位 / 输出清零 (如果链接到引脚，则 CT32Bn_MATm 引脚为低电平)。	
		0x2	将对应的外部匹配位 / 输出清零 (如果链接到引脚，则 CT32Bn_MATm 引脚为高电平)。	
		0x3	切换对应的外部匹配位 / 输出。	
7:6	EMC1		外部匹配控制 1。决定外部匹配 1 的功能。	00
		0x0	不执行任何动作。	
		0x1	将对应的外部匹配位 / 输出清零 (如果链接到引脚，则 CT32Bn_MATm 引脚为低电平)。	
		0x2	将对应的外部匹配位 / 输出清零 (如果链接到引脚，则 CT32Bn_MATm 引脚为高电平)。	
		0x3	切换对应的外部匹配位 / 输出。	
9:8	EMC2		外部匹配控制 2。决定外部匹配 2 的功能。	00
		0x0	不执行任何动作。	
		0x1	将对应的外部匹配位 / 输出清零 (如果链接到引脚，则 CT32Bn_MATm 引脚为低电平)。	
		0x2	将对应的外部匹配位 / 输出清零 (如果链接到引脚，则 CT32Bn_MATm 引脚为高电平)。	
		0x3	切换对应的外部匹配位 / 输出。	

Table 242: 外部匹配寄存器 (TMR32B0EMR - 地址 0x4001 403C 和 TMR32B1EMR - 地址 0x4001 803C) 位域描述

位	符号	值	描述	复位值
11:10	EMC3		外部匹配控制 3。决定外部匹配 3 的功能。	00
		0x0	不执行任何动作。	
		0x1	将对应的外部匹配位 / 输出清零 (如果链接到引脚, 则 CT32Bn_MATm 引脚为低电平)。	
		0x2	将对应的外部匹配位 / 输出清零 (如果链接到引脚, 则 CT32Bn_MATm 引脚为高电平)。	
		0x3	切换对应的外部匹配位 / 输出。	
31:12	-		保留, 用于软件不应写 1 到保留位。从保留位读出的值未定义。	NA

Table 243. 外部匹配控制

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	功能
00	不执行任何动作。
01	将对应的外部匹配位 / 输出清 0 (如果连接到引脚, 则 CT32Bn_MATm 引脚为低电平)。
10	将对应的外部匹配位 / 输出置 1 (如果连接到引脚, 则 CT32Bn_MATm 引脚为高电平)。
11	使对应的外部匹配位 / 输出发生翻转。

16.7.11 计数控制寄存器 (TMR32B0CTCR and TMR32B1TCR)

计数控制寄存器 (CTCR) 用于选择是定时器还是计数器模式, 且在计数器模式下悬在计数的引脚和边沿。

当选择工作在计数器模式时, 在每个 PCLK 时钟上升沿对 CAP 输入 (通过 CTCR 的位 3:2) 进行采样。在比较完 CAP 输入的两个连续采样后, 就可以识别以下四个事件之一: 上升沿、下降沿、任一边沿或在选定的 CAP 输入上无电平变化。只要识别到的事件与 CTCR 寄存器位 1:0 选定的事件对应, 定时器计数器寄存器将递增 1。

计数器外部时钟源有效操作有一些限制。由于 PCLK 时钟的 2 个连续上升沿用来识别 CAP 选择输入的一个边沿, 所以 CAP 输入的频率不能大于 1/2 个 PCLK 时钟。因此, 这种情况下同一 CAP 输入的高 / 低电平持续时间不能小于 1/(2 × PCLK) 。

Table 244: 计数控制寄存器 (TMR32B0CTCR - 地址 0x4001 4070 和 TMR32B1TCR - 地址 0x4001 8070) 位域描述

位	符号	值	描述	复位值
1:0	CTM		计数器 / 定时器模式。该位域选择触发定时器的预分频计数器 (PC) 递增、清除 PC 和定时器计数器 (TC) 递增的 PCLK 边沿。 定时器模式：每个 PCLK 的上升沿。	00
		0x0	定时器模式：每个 PCLK 的上升沿。	
		0x1	计数器模式：在位 3:2 所选择的 CAP 输入的上升沿， TC 递增。	
		0x2	计数器模式：在位 3:2 所选择的 CAP 输入的下降沿， TC 递增。	
		0x3	计数器模式：在位 3:2 所选择的 CAP 输入的双边沿， TC 递增。	
3:2	CIS		计数输入选择：当位 1:0 不是 00，这些位用来选择对哪个 CAP 引脚进行采样计时。	00
		0x0	CT32Bn_CAP0	
		0x1	保留	
		0x2	保留	
		0x3	保留	
			注意： 如果在 TnCTCR 寄存器中选择了计数器模式，捕获控制寄存器 (TnCCR) 的位 2:0 必须为 000。	
31:4	-	-	保留，用于软件不应写 1 到保留位。从保留位读出的值未定义。	NA

16.7.12 PWM 控制寄存器 (TMR32B0PWMC 和 TMR32B1PWMC)

PWM 控制寄存器用于将匹配输出配置为 PWM 输出或匹配输出，而这些功能由外部匹配寄存器 (EMR) 来控制。

对于每个定时器，可选择 MATn[2:0] 这三个输出为单位边沿控制的 PWM 输出。另外一个匹配寄存器决定 PWM 周期长度。当在其它任何一个匹配寄存器中出现匹配时， PWM 输出设置为高电平。定时器通过配置为设定 PWM 周期长度的匹配寄存器来复位。当定时器被复位为 0 时，所有被配置为 PWM 输出的高电平匹配输出被清除。

Table 245: PWM 控制寄存器 (TMR32B0PWMC - 0x4001 4074 和 TMR32B1PWMC - 0x4001 8074) 位域描述

位	符号	值	描述	复位值
0	PWMEN0		PWM 通道 0 使能	0
		0	CT32Bn_MAT0 由 EM0 控制。	
		1	允许 CT32Bn_MAT0 使用 PWM 模式。	

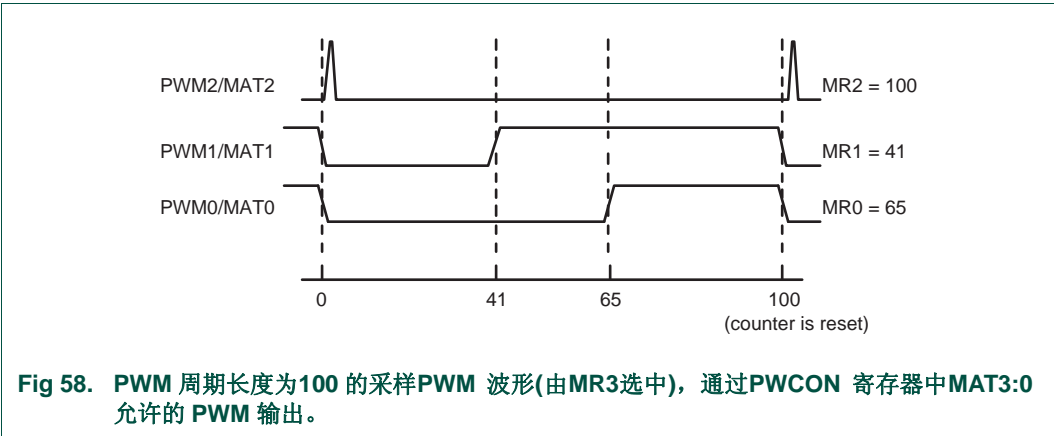
Table 245: PWM 控制寄存器 (TMR32B0PWC - 0x4001 4074 和 TMR32B1PWC - 0x4001 8074) 位域描述

位	符号	值	描述	复位值
1	PWMEN1		PWM 通道 1 使能	0
		0	CT32Bn_MAT1 由 EM1 控制。	
		1	允许 CT32Bn_MAT1 使用 PWM 模式。	
2	PWMEN2		PWM 通道 2 使能	0
		0	CT32Bn_MAT2 由 EM2 控制。	
		1	允许 CT32Bn_MAT2 使用 PWM 模式。	
3	PWMEN3		PWM 通道 3 使能	0
			注意: 建议使用通道 3 设置 PWM 周期, 因为它没有引脚输出。	
		0	CT32Bn_MAT3 由 EM3 控制。	
		1	允许 CT32Bn_MAT3 使用 PWM 模式。	
31:4	-		保留, 用户软件不应写 1 到保留位。从保留位读出的值未定义。	NA

16.7.13 单边沿控制 PWM 输出规则

- 1. 所有被单边沿控制 PWM 输出在每个 PWM 周期开始时都置为低电平（定时器设置为 0），除非它们的匹配值等于 0。
- 2. 每个 PWM 输出都会在它的匹配值到达时置为高电平，在没有到达（也就是匹配值大于 PWM 周期的长度）时为低电平。
- 3. 如果匹配值大于写入匹配寄存器中的 PWM 周期长度，同时 PWM 信号已经是高电平，那么 PWM 信号会在下一个 PWM 周期开始时被清除。
- 4. 如果匹配寄存器中的值与定时器复位值相同（PWM 周期的长度），那么 PWM 输出就会在下一个时钟滴答复位为低。因此，PWM 输出一般包括一个时钟滴答宽度的正脉冲，其宽度由 PWM 周期长度决定（也就是定时器的重装载值）。
- 5. 如果匹配寄存器被设置为 0，那么 PWM 输出会在定时器第一次回到 0 时成为高电平，且持续保持高电平。

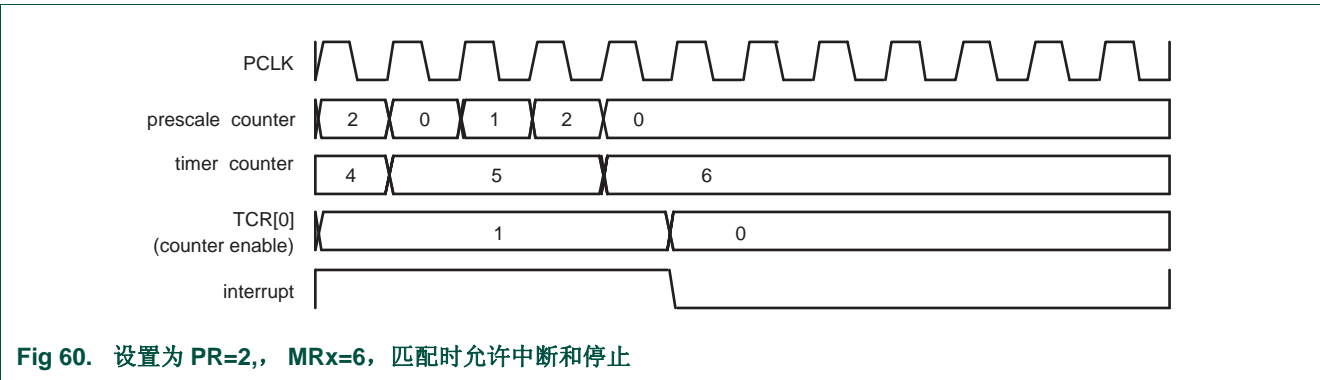
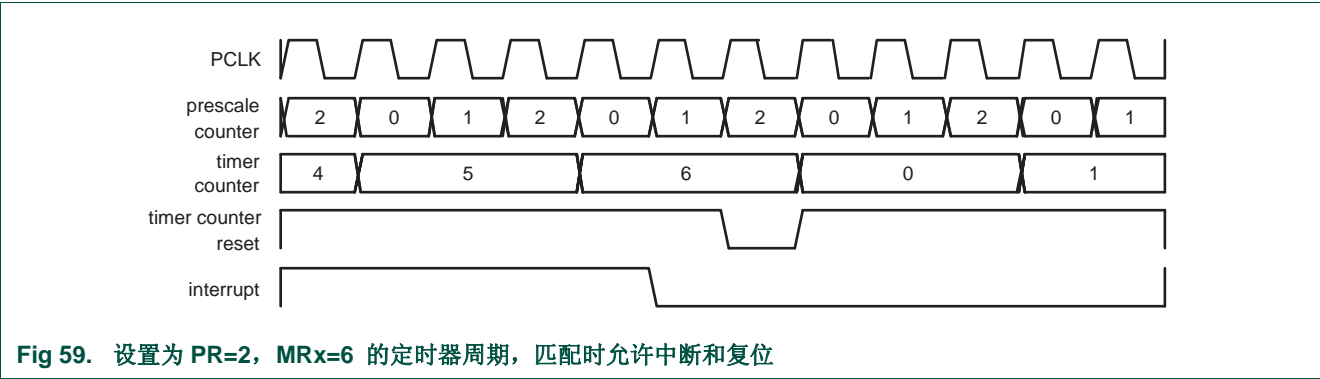
注意: 当匹配输出被用作 PWM 输出时，在匹配寄存器中的定时器复位位 (MRnR) 和定时器停止位 (MRnS) 就必须设置为 0，除了用于设置 PWM 周期长度的匹配寄存器。对于该寄存器，如果设置 MRnR 位为 1，当定时器值与相应的匹配寄存器值发生匹配时，定时器允许被复位。



16.8 定时器操作实例

Figure 59 所示定时器配置，当匹配发生时，定时器复位并产生一个中断。预分频器设置为 2，匹配寄存器设置为 6。在匹配发生的定时器周期结束时，定时器计数值被复位。这样就使匹配值具有完整长度的周期。在定时器到达匹配值后的下一个时钟周期内，中断产生。

Figure 60 所示定时器配置，当匹配发生时，定时器停止并产生一个中断。预分频器设置为 2，匹配寄存器设置为 6。在定时器到达匹配值后的下一个时钟周期内，TCR 中的定时器允许位被清零，并产生一个中断。



16.9 Architecture

32 位计数器 / 定时器 0 和 32 位计数器 / 定时器 1 的结构如图 [Figure 61](#)。

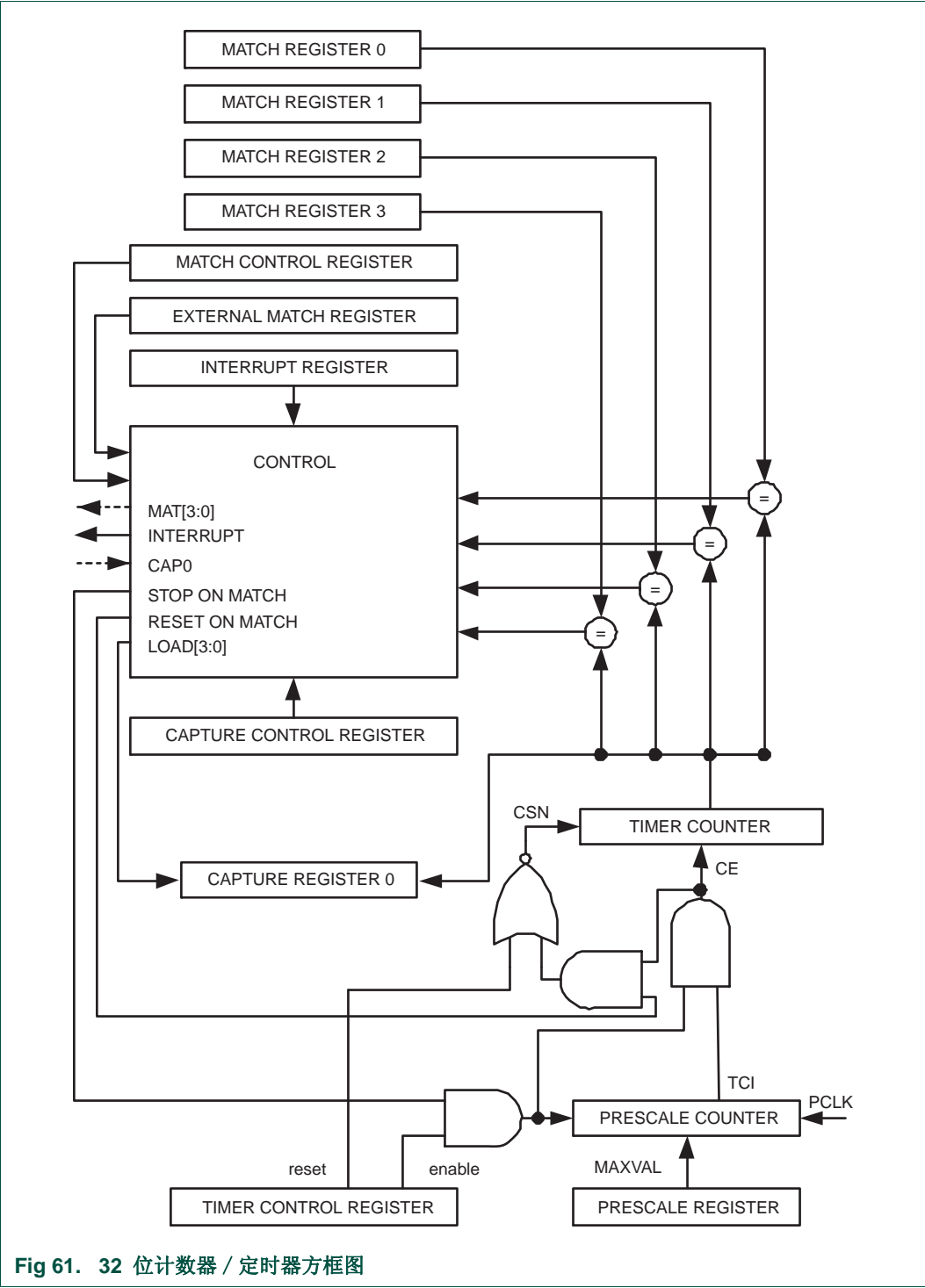


Fig 61. 32 位计数器 / 定时器方框图

17.1 如何阅读本章

本章介绍 LPC111x/102/202/302 LPC1100L 部分的窗口看门狗定时器 WDT。

17.2 基本配置

通过以下寄存器配置 WDT：

1. 引脚：WDT 不使用外部引脚。
2. 电源：在 SYSAHBCLKCTRL 寄存器，设置第 15 位 ([Table 21](#))。
3. 外设时钟：选择看门狗时钟源 ([Table 25](#))，通过写 WDTCLKDIV 寄存器 ([Table 27](#))，使能 WDT 外设时钟。
备注：看门狗振荡器在复位之后时器的频率是不确定的。如果要因为 WDT 而使用看门狗振荡器时，必须将振荡器的频率写到 WDTOSCCTRL 寄存器中。(见 [Table 13](#))
4. 锁定功能：一旦通过设置 WDMOD 寄存器的 WDEN 位启动看门狗定时器，以下锁定功能将有效：
 - a. WDEN 位不能更改为 0，即 WDT 不能被禁用。
 - b. 看门狗时钟源是不能改变的。如果 WDT 需要进入深度睡眠模式，在设置 WDEN 之前，选择看门狗振荡器作为时钟源。

17.3 特性

- 如果没有周期性重装计数值，则产生片内复位。
- 可选的窗口操作重载最小和最大超时期限之间的值，两个超时期限都可编程设置。
- 在可编程看门狗超时之前，可选的警告中断可以产生。
- 可编程的 24 位定时器（带有内部预分频器）。
- 时钟周期可选，可从 $(T_{WDCLK} \times 256 \times 4)$ 到 $(T_{WDCLK} \times 2^{24} \times 4)$ ，取 $(4 \times T_{WDCLK})$ 的整数倍。
- “安全”看门狗操作。一旦使能，需要通过硬件复位或看门狗复位来禁能。
- 一个专用的片上看门狗振荡器提供了可靠地时钟源。当看门狗定时器在运行时，不要停止该时钟源。
- 如果看门狗被允许，不正确 / 不完整的喂狗时序会产生复位 / 中断。
- 看门狗重载值能随意保护，这样只有在警告中断发生时，它的值才能改变。
- 具有看门狗复位标志。

17.4 应用

看门狗定时器的目的是当它进入错误状态时，在一个合理的时间内复位微控制器。当它启用时，如果用户程序没有在预定的时间内“喂”重装看门狗，将会产生一个看门狗事件。如果这个看门狗事件发生，将会导致芯片复位。

当运行一个看门狗窗口时，太早的重装看门狗也将被视为一个看门狗事件。这样可以防止系统出现故障的情况下仍可以重装看门狗。例如：应用程序代码可以停留在一个中断服务，其中就包含一个看门狗的重装。设置窗口将导致在早期的重装将会产生一个看门狗事件，从而使系统恢复。

17.5 描述

看门狗定时器包括一个固定的4分频器和1个24位计数器，定时计数器递减。开始递减的值，最小必须是0xFF，如果设定小于0xFF的值，则默认将0xFF装载到计数器。因此，看门狗最小时间是 $(T_{WDCLK} \times 256 \times 4)$ ，最大时间是 $(T_{WDCLK} \times 2^{24} \times 4)$ ，取 $(T_{WDCLK} \times 4)$ 的倍数。看门狗必须按如下方法使用：

- 在WDTC寄存器中设定看门狗定时器重装载值。
- 在WDMOD寄存器中设定看门狗定时器工作模式。
- 如果需要窗口化的操作，就要在WDWINDOW寄存器中先为看门狗窗口时间设置一个值。
- 如果需要警告中断，就要在WDWARNINT寄存器中设置一个看门狗警告中断的初始值。
- 向WDFEED寄存器先写入0xAA，再写0x55启动看门狗。
- 在计数值下溢之前再次喂狗，以防止看门狗复位/中断。如果一个窗口值被编程设置，当计数器经过预设值时喂狗动作同样需要。

当看门狗处在复位模式且计数器下溢，CPU将被复位，从向量表中读取堆栈指针和程序计数器，与外部复位一样。可检测看门狗超时标志(WDTOF)判断看门狗是否已产生复位条件，要必须对WDTOF通过软件清0。

当看门狗计数器被配置为生成一个警告中断时，当计数器的值达到WDWARNINT寄存器中设置的值时就会产生一个中断。

看门狗的方块图如 [Figure 62](#)，图中没有展示同步逻辑(PCLK - WDCLK)。

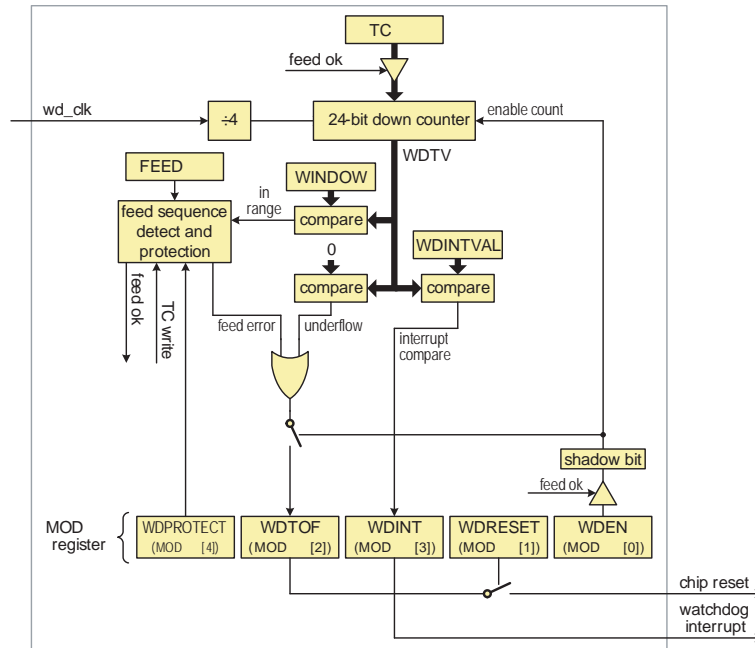


Fig 62. 窗口看门狗定时器 (WWDT) 方框图

17.6 时钟控制

看门狗定时器模块使用2个时钟，PCLK和WDCLK。PCLK是APB访问看门狗寄存器所用的时钟，它来自系统时钟（见 [Figure 3](#)）。WDCLK用于看门狗定时器计数，它来自于看门狗时钟见 [Figure 3](#)。可作为 wdt_clk 时钟源的时钟有：IRC、看门狗振荡器、主时钟。时钟源在系统控制块中选择（见 [Table 25](#)）。WDCLK 拥有他自己的时钟分频器 ([Table 27](#))，该时钟可以通过自己的时钟分频器禁止。

这两个时钟域有一些同步逻辑，当 WDMON 和 WDTIC 寄存器被 APB 操作更新时，WDCLK 时钟域的新值将在 3 个 WDCLK 周期生效。当看门狗定时器按 WDCLK 进行计数，同步逻辑会先锁定 WDCLK 上的计数值，然后再让它与 PCLK 同步，以供 CPU 读取 WDTV 寄存器。

如果不使用看门狗振荡器，可通过 PDRUNCFG 寄存器 ([Table 42](#)) 将其关闭。为了节省功耗，可以在 AHBCLKCTRL 寄存器中 ([Table 21](#)) 禁止到看门狗寄存器模块的时钟。

备注：复位之后看门狗振荡器的频率是未知值。在将看门狗振荡器作为 WDT 的时钟源之前，必须通过写 WDTOSCCTRL 寄存器（见 [Table 13](#)）对看门狗振荡器的时钟源进行编程设定。

17.7 寄存器描述

看门狗包括的寄存器如 [Table 246](#) 所列。

Table 246. 寄存器概览：看门狗定时器 (基地址 0x4000 4000)

名称	访问方式	偏移地址	描述	复位值 ^[1]
WDMOD	R/W	0x000	工作模式寄存器，包含基本模式和看门狗定时器模式。	0
WDTC	R/W	0x004	看门狗定时器常量寄存器，决定看门狗超时时间值。	0xFF
WDFEED	WO	0x008	看门狗喂狗寄存器，对该寄存器先写 0xAA，再写 0x55 以装载 WDTC 中的值到看门狗定时器。	-
WDTV	RO	0x00C	看门狗定时器值寄存器，通过它可以读出看门狗定时器的当前值。	0xFF
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] 复位值仅反映可用位的数据，它不包含保留位的内容。

17.7.1 看门狗模式寄存器

通过看门狗模式寄存器 (WDMOD) 中的 W DEN 和 RESET 位的组合可以控制看门狗操作。注意，WDMOD 寄存器任何改变生效之前，必须执行一次喂狗操作。

Table 247: 看门狗模式寄存器 (WDMOD - 0x4000 4000) 位域描述

位	符号	值	描述	复位值
0	WDEN		看门狗使能位，该位只能设置。	0
		0	看门狗定时器被停止。	
		1	看门狗定时器运行状态。	
1	WDRESET		看门狗复位使能位，该位只能设置。	0
		0	看门狗超时时不会导致芯片复位。	
		1	看门狗超时将导致芯片复位。	
2	WDTOF		看门狗超时标志。当看门狗定时器超时、喂狗错误或 WDPROTECT 相关事件的发生时置位，通过软件清零。当 WDRESET = 1 时，芯片复位。	0 (只在外部复位以后)
3	WDINT		看门狗中断标志。当计时器到达 WDWARNINT 的值时置位。通过软件清零。	0
4	WDPROTECT		看门狗更新模式，该位只能设置。	0
		0	看门狗重载值 (WDTC)，可随时更改。	
		1	只有当计数值低于 WDWARNINT 和 WDWINDOW 的值时，可改变看门狗重载值 (WDTC)。注意：只有当 WDRESET =1 时，才使用这种模式。	
31: 5	-		保留。读取值未定义，只可对该位写入零。	-

一旦 **WDEN**、**WDPROTECT** 或者 **WDRESET** 其中一个置位或都被置位时，它们不能被软件清除。这两个标志在外部复位或看门狗复位时清零。

WDTOF 看门狗超时、喂狗错误或 **WDPROTECT = 1** 时该标志置位，并向 **WDTC** 寄存器写入。可对该位写入 0 清除。

WDINT 当看门狗计数器到达 **WDWARNINT** 所指定的值时看门狗中断标志置位。发生任何复位时该标志位清除，同样可通过软件对该位写 1 进行清零。

看门狗复位或中断可在看门狗运行时的任何时间发生。如果看门狗处于休眠模式，看门狗中断将唤醒该设备。

Table 248. 看门狗运行模式选择

WDEN	WDRESET	运行模式
0	X (0 or 1)	看门狗被禁用。
1	0	看门狗中断模式：带看门狗中断调试，但是不允许看门狗复位。选择该模式，当看门狗计数器到达 WDWARNINT 指定的值将置位 WDINT ，并产生看门狗中断请求。
1	1	看门狗复位模式：带看门狗中断和看门狗复位操作。选择该模式，看门狗计数器到达 WDWARNINT 指定的值将置位 WDINT ，产生看门狗中断请求，看门狗计数值为零并复位微控制器。在到达 WDWINDOW 的值之前喂狗同样会导致看门狗复位。

17.7.2 看门狗定时器常量寄存器

WDTC 寄存器决定超时时间，喂狗操作将使 **WDTC** 中的内容装载到看门狗定时器中。复位后有个预装载值 0x00 00FF。写入小于 0xFF 的数值会默认将 0x0000 00FF 装载到 **WDTC** 中。因此最小下溢时间是 $T_{WDCLK} \times 256 \times 4$ 。

如果 **WDMOD** 寄存器中的位 **WDPROTECT = 1**，若在看门狗计数值低于 **WDWARNINT** 和 **WDWINDOW** 的值时，改变 **WDTC** 的值会导致看门狗复位并置位 **WDTOF** 标志。

Table 249: 看门狗重载值寄存器 (WDTC - 0x4000 4004) 位域描述

位	符号	描述	复位值
23:0	Count	设定看门狗超时时间。	0x00 00FF
31:24	-	保留。读取值未定义，只可对该位写入零。	NA

17.7.3 看门狗喂狗寄存器

对该寄存器先写 0xAA 再写 0x55，将把 WDTC 中的装载值重新装载到看门狗定时器中。如果通过 WDMOD 寄存器允许了看门狗，此操作将启动看门狗。置位 WDMOD 中的 WDEN 并不足以启动看门狗。置位 WDEN 之后，必须完成一个有效的喂狗操作，才能产生看门狗复位。在此之前，看门狗将忽略喂狗错误。写入 0xAA 到 WDFEED 后，必须紧接着写入 0x55，否则如果看门狗被允许，访问任何看门狗的寄存器操作，将会立即产生看门狗复位或中断，并置位 WDTOF 标志。在喂狗期间错误地访问看门狗定时器寄存器，将会在第二个 PCLK 时钟产生复位。

Table 250: 看门狗喂狗寄存器 (WDFEED - 0x4000 4008) 位域描述

位	符号	描述	复位值
7:0	Feed	喂狗必须顺序写入 0xAA 和 0x55。	-
31:8	-	保留	-

17.7.4 看门狗定时器值寄存器

看门狗定时器值 (WDTV) 寄存器 用于读取看门狗当前计数值。

读取 24 位的计数值时，锁定和同步过程需要 6 个 WDCLK 周期加上 6 个 PCLK 周期，因此 CPU 读到值要比实际的要旧。

Table 251: 看门狗定时器值寄存器 (WDTV - 0x4000 400C) 位域描述

位	符号	描述	复位值
23:0	Count	定时器计数值。	0x00 00FF
31:24	-	保留。读取值未定义，只可对该位写入零。	-

17.7.5 看门狗定时器警告中断寄存器

看门狗警告中断寄存器 (WDWARNINT) 决定了可能会产生一个看门狗中断的定时器计数器的值。当看门狗定时器计数器的值与 WDWARNINT 中定义的值相匹配时，将会在 WDCLK 之后产生一个中断。

当定时器计数器的低十位的值和 WARNINT 相同时，那么看门狗定时器计数器的值与 WDWARNINT 发生匹配。定时器计数器的剩下的高位全为 0。这将在看门狗时间之前留出 1023 看门狗定时器计数值 (4096 个看门狗时钟) 的最大值。如果 WARNINT 设置为 0，看门狗时间一发生，将会产生中断。

Table 252: 看门狗定时器警告中断寄存器 (WDWARNINT - 0x4000 4014) 位域描述

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。读取值未定义，只可对该位写入零。	-

17.7.6 看门狗定时器窗口寄存器

WDWINDOW 寄存器决定了执行喂狗序列时 WDTV 允许的最高值。如果在 WDTV 值达到 WDWINDOW 之前完成一个有效的喂狗序列，将导致看门狗事件发生。

WDWINDOW 以最大可能的 WDTV 的值复位，使得窗口功能无效。

Table 253: 看门狗定时器窗口寄存器 (WDWINDOW - 0x4000 4018) 位域描述

位	符号	描述	复位值
23:0	WINDOW	看门狗窗口值。	0xFF FFFF
31:24	-	保留。读取值未定义，只可对该位写入零。	-

17.7.7 看门狗定时器举例

以下几幅图呈现看门狗定时器操作的几个方面。

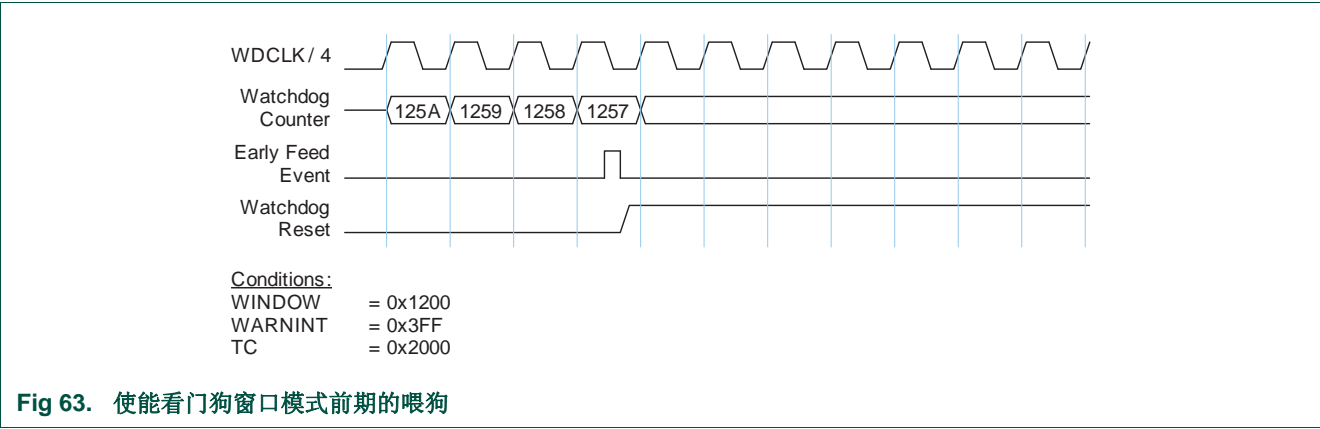
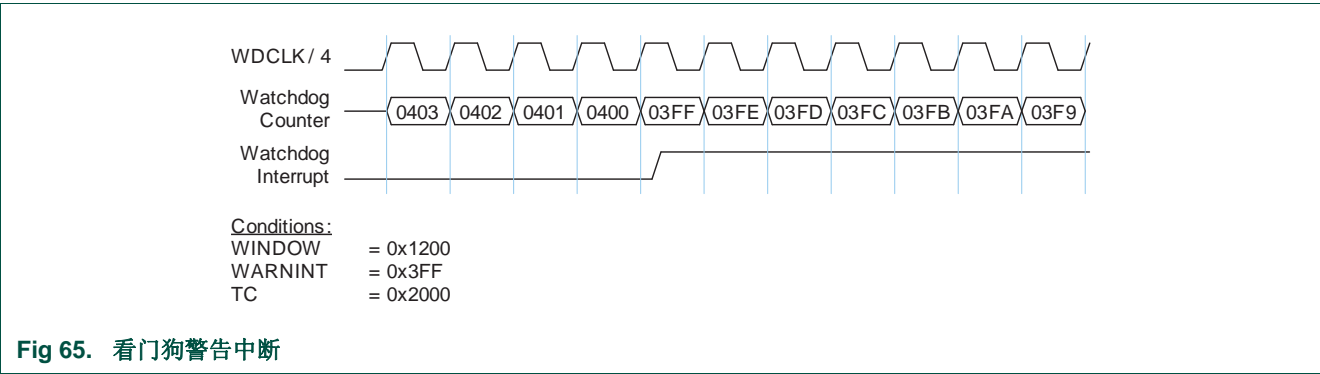
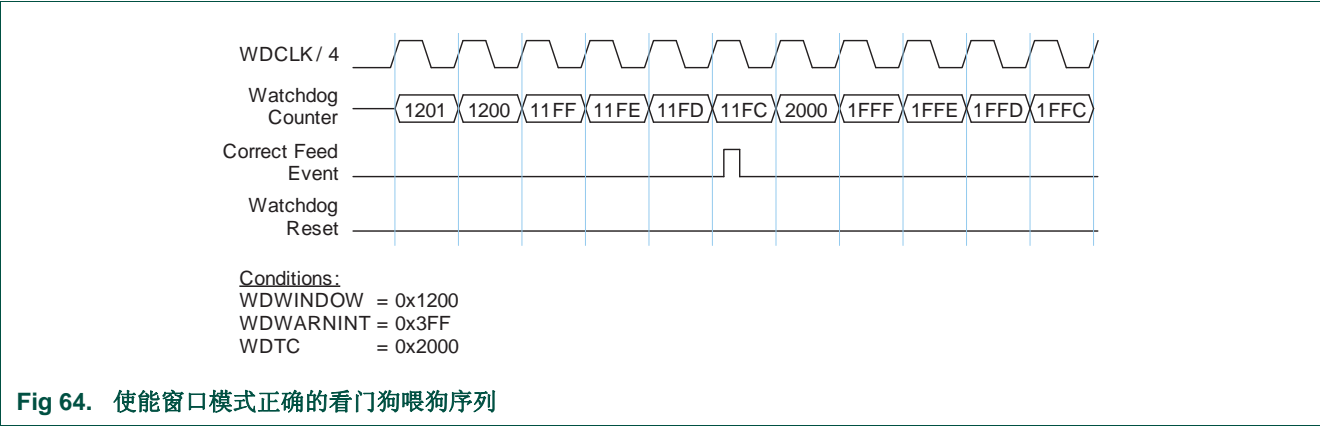


Fig 63. 使能看门狗窗口模式前期的喂狗



18.1 如何阅读本章

所有 LPC111x 和 LPC11Cx 系列处理器的非窗口式看门狗定时器 WDT 模块完全一致。

当 WDT 被使能时，时钟源锁定功能也只能在部分 LPC11Cx 系列处理器上实现。

18.2 基本配置

通过以下寄存器配置 WDT：

1. 引脚：WDT 不使用外部引脚。
2. 电源：在 SYSAHBCLKCTRL 寄存器，设置第 15 位 ([Table 21](#))。
3. 外设时钟：选择看门狗时钟源 ([Table 25](#))，通过写 WDTCLKDIV 寄存器 ([Table 27](#)) 使能 WDT 外设时钟。

备注：看门狗复位后，其振荡器的频率是不确定的。在使用看门狗振荡器之前，必须重新在 WDTOSCCTRL 寄存器 (见 [Table 13](#)) 写入设定值。

4. 锁定功能：一旦通过设置 WDMOD 寄存器的 WDEN 位启动看门狗定时器，以下锁定功能将有效：
 - a. WDEN 位不能更改为 0，即 WDT 不能被禁用 (LPC111x/101/201/301 和 LPC11Cx)。
 - b. 看门狗时钟源是不能改变的。如果 WDT 需要进入深度睡眠模式，在设置 WDEN 之前，选择看门狗振荡器作为时钟源 (仅在 LPC11Cx)。

18.3 特性

- 如果没有周期性重载计数值 (即喂狗)，则产生片内复位。
- 支持调试模式。
- 通过软件允许看门狗，但禁止看门狗硬件复位或看门狗复位 / 中断。
- 如果看门狗被允许，错误 / 不完整的喂狗时序会导致复位 / 中断。
- 具有看门狗复位标志。
- 带内部预分频器的可编程 24 位定时器。
- 时钟周期可选，从 $(T_{WDCLK} \times 256 \times 4)$ 到 $(T_{WDCLK} \times 2^{24} \times 4)$ ，取 $T_{WDCLK} \times 4$ 的倍数。
- 可在系统控制模块中选择内部 RC 振荡器 (IRC)、主时钟或看门狗振荡器，来作为看门狗的时钟 (WDCLK) 源，见 [Table 25](#)。看门狗定时器在不同功耗条件下，有多种可能计时选择。为增加可靠性，同时还提供了一个看门狗定时器专用的完整内部时钟源，它不依赖于外部晶振及其相关元件和线路。

18.4 应用

看门狗的目的是为了使微控制器在程序运行进入错误状态时，使系统在一个合理的时间内复位。当看门狗被允许之后，如果用户程序没有在预定的时间内进行“喂狗”（或重新装载计数值），看门狗将产生一个系统复位。

18.5 描述

看门狗定时器包括一个固定的 4 分频器和一个 24 位计数器，时钟通过预频器送给定时器。每到一个时钟，定时器计数值减 1，开始递减的值，最小必须是 0xFF。如果设定小于 0xFF 的值，则默认将 0xFF 装载到计数器。因此，看门狗最小时间间隔是 $(T_{WDCLK} \times 256 \times 4)$ ，最大时间间隔是 $(T_{WDCLK} \times 2^{24} \times 4)$ ，取 $(T_{WDCLK} \times 4)$ 的倍数。看门狗必须按如下方法使用：

1. 在 WDTC 寄存器中设定看门狗定时器重装值。
2. 在 WDMOD 寄存器中设定看门狗定时器工作模式。
3. 向 WDFEED 寄存器先写入 0xAA，再写 0x55 启动看门狗。
4. 在计数值下溢之前再次喂狗，以防止看门狗复位 / 中断。

当看门狗在复位模式下且计数器下溢，CPU 将被复位，从向量表中读取堆栈指针和程序计数器，与外部复位一样。可检测看门狗超时标志 (WDTOF) 判断看门狗是否已产生复位条件，WDTOF 标志必须由软件清零。

18.6 WDT 时钟

看门狗定时器功能块使用两个时钟 PCLK 和 WDCLK。PCLK 来自系统时钟（见 [Figure 3](#)），APB 访问看门狗寄存器时使用。WDCLK 来自 wdt_clk（见 [Figure 3](#)），用于看门狗定时器计数。可作为 wdt_clk 时钟源的有：IRC、看门狗振荡器、主时钟。时钟源在系统控制块中选择（见 [Table 25](#)）。WDCLK 拥有它自己的时钟分频器（[Section 3.5.20](#)），该时钟可以被禁止。

这两个时钟域有一些同步逻辑，当 WDMOD 和 WDTC 寄存器被 APB 操作更新时，WDCLK 时钟域的新值将在 3 个 WDCLK 周期生效。当看门狗定时器按 WDCLK 进行计数，同步逻辑会先锁定 WDCLK 上的计数值，然后再让它与 PCLK 同步，以供 CPU 读取 WDTV 寄存器。

备注：看门狗复位后，其振荡器的频率是不确定的。在使用看门狗振荡器之前，必须重新在 WDTOSCCTRL 寄存器（见 [Table 13](#)）写入设定值。

18.7 寄存器描述

看门狗包括 4 个寄存器如 [Table 254](#) 所列。

Table 254. 寄存器概览：看门狗定时器 (基地址 0x4000 4000)

名称	访问方式	偏移地址	描述	复位值 [1]
WDMOD	R/W	0x000	工作模式寄存器，包含基本模式和看门狗定时器模式。	0
WDTC	R/W	0x004	看门狗定时器常量寄存器，决定看门狗超时时间值。	0xFF
WDFEED	WO	0x008	看门狗喂狗寄存器，对该寄存器先写 0xAA，再写 0x55 以装载 WDTC 中的值到看门狗定时器。	NA
WDTV	RO	0x00C	看门狗定时器值寄存器，通过它可以读出看门狗定时器的当前值。	0xFF

[1] 复位值仅反映可用位的数据，它不包含保留位的内容。

18.7.1 看门狗模式寄存器 (WDMOD - 0x4000 0000)

通过看门狗模式寄存器 (WDMOD) 中的 WDEN 和 RESET 位的组合可以控制看门狗操作。注意，WDMOD 寄存器任何改变生效之前，必须执行一次喂狗操作。

Table 255. 看门狗模式寄存器 (WDMOD - 地址 0x4000 4000) 位域描述

位	符号	描述	复位值
0	WDEN	启动看门狗（只能设置），为 1 时看门狗定时器运行。 备注： 该位设置也锁定一个看门狗时钟源，当启用了看门狗定时器，看门狗时钟源是不能改变的。如果要使看门狗定时器处于深度睡眠模式，必须改变看门狗之前对此位的设置。并不是所有模块都具有时钟源锁定功能，见 Section 18.1 。	0
1	WDRESET	看门狗复位允许（只能设置），为 1 时，如果看门狗下溢将导致芯片复位。	0
2	WDTOF	下溢标志位，看门狗下溢时被置位，用软件清除。	0 (仅在 POR 和 BOD 复位后)
3	WDINT	看门狗中断标志（只读，不能被软件清除）	0
7:4	-	保留，用于软件不应对这些位写 1，读出值未定义。	NA
31:8	-	保留	-

一旦 **WDEN**、**WDRESET** 其中一个被置位或都被置位，它们不能被软件清除。这两个标志在复位或看门狗定时器下溢时清零。

WDTOF 看门狗下溢时该标志置位，可用软件清除，上电复位或掉电检测复位时也可以清除。

WDINT 看门狗下溢时该标志置位，复位时清除。一旦看门狗中断被服务，它可以在 NVIC 中被禁止，或将不确定地产生看门狗中断请求。引入看门狗的目的是为了在看门狗处于活动状态下允许调试，在看门狗下溢时不复位设备。

当看门狗拥有一个活动时钟且处于运行状态时，看门狗复位和中断可以随时产生。任何时钟源在睡眠模式下，如果在睡眠模式下发生看门狗中断，它将唤醒处理器。

Table 256. 看门狗运行模式选择

WDEN	WDRESET	运行模式
0	X (0 or 1)	看门狗被禁用。
1	0	看门狗中断模式：带看门狗中断调试，但是不允许看门狗复位。 选择该模式，当看门狗计数器下溢会置位 WDINT 标志，并产生看门狗中断请求。 备注： 在中断模式下，检查 WDINT 标志。如果设置了该标志，产生的中断为真，可以由中断服务程序提供服务。如果没有设置该标志，产生的中断会被忽略掉。
1	1	看门狗复位模式：带看门狗中断和看门狗复位操作。 选择该模式，看门狗计数下溢复位芯片，虽然允许了看门狗中断 (WDEN = 1) 但这种情况下它不会被响应，因为看门狗复位将清除 WDINT 标志。

18.7.2 看门狗定时器常量寄存器 (WDTC - 0x4000 4004)

WDTC 寄存器决定超时时间，喂狗操作将使 WDTC 中的内容装载到看门狗定时器中。该寄存器 32 位，复位时最低 8 位为 1，写入小于 0xFF 的数值会默认将 0x0000 00FF 装载到 WDTC 中。因此最小下溢时间是 $T_{WDCLK} \times 256 \times 4$ 。

Table 257. 看门狗重载值寄存器 (WDTC - 地址 0x4000 4004) 位域描述

位	符号	描述	复位值
23:0	Count	设定看门狗超时时间。	0x0000 00FF
31:25	-	保留	-

18.7.3 看门狗喂狗寄存器 (WDFEED - 0x4000 4008)

对该寄存器先写 0xAA 再写 0x55，将把 WDTC 中的装载值重新装载到看门狗定时器中。如果通过 WDMOD 寄存器允许了看门狗，此操作将启动看门狗。置位 WDMOD 中的 WDEN 并不足以启动看门狗。置位 WDEN 之后，必须完成一个有效的喂狗操作，才能产生看门狗复位。在此之前，看门狗将忽略喂狗错误。写入 0xAA 到 WDFEED 后，必须紧接着写入 0x55，否则如果看门狗被允许，访问任何看门狗的寄存器操作，将会立即产生看门狗复位或中断。在喂狗期间错误地访问看门狗定时器，将会在第二个 PCLK 时钟产生复位。

喂狗期间时必须禁止中断，如果喂狗时产生一个中断将会产生中止 (abort) 条件。

Table 258. 看门狗喂狗寄存器 (WDFEED - 地址 0x4000 4008) 位域描述

位	符号	描述	复位值
7:0	Feed	喂狗必须顺序写入 0xAA 和 0x55。	NA
31:8	-	保留	-

18.7.4 看门狗定时器值寄存器 (WDTV - 0x4000 400C)

看门狗定时器值 (WDTV) 寄存器 用于读取看门狗当前计数值。

读取 24 位的计数值时，锁定和同步过程需要 6 个 WDCLK 周期加上 6 个 PCLK 周期，因此 CPU 读到值要比实际的要早。

Table 259. 定时器计数值寄存器 (WDTV - 地址 0x4000 000C) 位域描述

位	符号	描述	复位值
23:0	Count	定时器计数值	0x0000 00FF
31:24	-	保留	-

18.8 方框图

看门狗如 Figure 66 所示，同步逻辑单元 (PCLK/WDCLK) 没有在框图中显示。

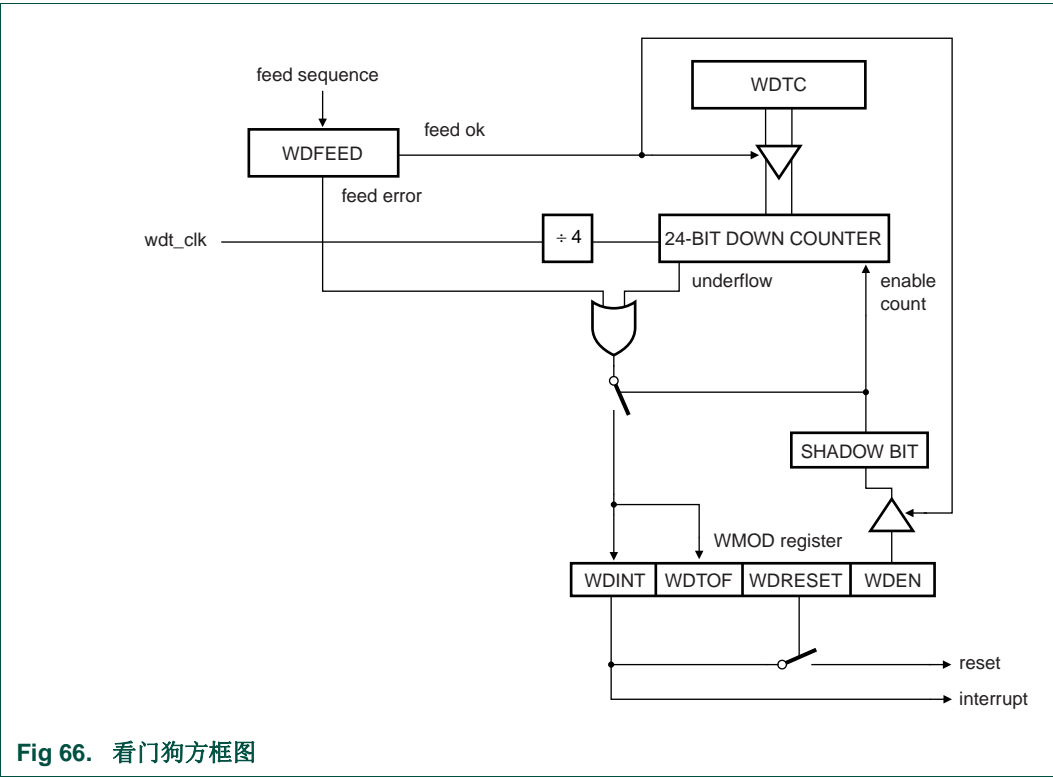


Fig 66. 看门狗方框图

19.1 如何阅读本章

系统滴答定时器 (SysTick timer) 是 ARM Cortex-M0 核的一部分，所有 LPC111x 和 LPC11Cxx 系列处理器的系统滴答定时器均相同。

19.2 基本配置

通过以下寄存器对系统滴答定时器进行配置：

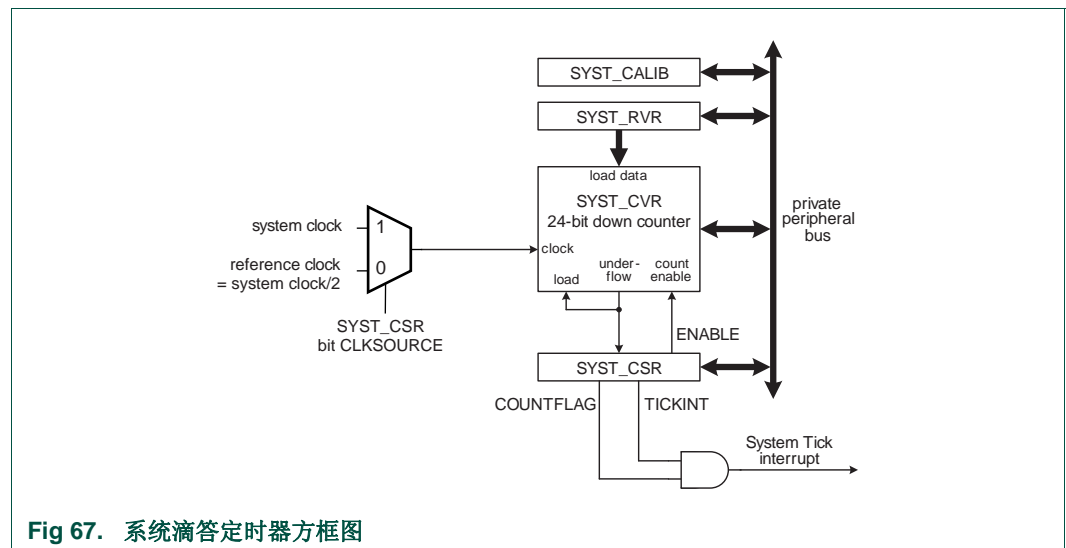
1. 引脚：系统滴答定时器不使用外部引脚。
2. 电源：系统滴答定时器是通过 SysTick 控制寄存器 ([Section 23.5.4.1](#)) 配置允许。系统滴答定时器时钟源是固定的系统时钟频率的一半，即参考时钟。
3. SYST_CSR 寄存器使能 SysTick 定时器时钟源 ([Table 261](#))。

19.3 特性

- 简单 24 位定时器。
- 使用专用的异常向量。
- 内部时钟源由系统时钟或参考时钟提供。

19.4 概述

SysTick 定时器的框图如 [Figure 67](#) 所示。



SysTick 定时器是 Cortex-M0 内部集成的一部分。SysTick 定时器的作用是产生一个固定的 10 毫秒中断，供操作系统或其它系统管理软件使用。

由于 SysTick 定时器是 Cortex-M0 的一部分，它为基于 Cortex-M0 的处理器提供一个标准计时器，有助于软件的移植。SysTick 定时器可用于：

- 可编程设置频率的 RTOS 定时器 (例如 100 Hz)，调用一个 SysTick 服务程序。
- 用于核时钟的高速报警定时器。
- 简单计数器。软件可使用它测量时间（如：完成任务所需时间、已使用时间）。
- 基于丢失 / 命中期限控制的内部时钟源。控制和状态寄存器中的 COUNTFLAG 位域，可用于决定一个动作是否在设定的期限内完成，作为动态时钟管理控制环的一部分。

详情可参考 *Cortex-M0 用户指南*。

19.5 寄存器描述

SysTick 定时器位于 ARM Cortex-M0 专用外设总线上 (见 [Figure 75](#))，是 ARM Cortex-M0 核内外设。详见 [Section 23.5.4](#)。

Table 260. 寄存器概览: SysTick 定时器 (基地址 0xE000 E000)

名称	访问方式	地址偏移	描述	复位值 [1]
SYST_CSR	R/W	0x010	系统定时器控制和状态寄存器	0x000 0000
SYST_RVR	R/W	0x014	系统定时器重载寄存器	0
SYST_CVR	R/W	0x018	系统定时器当前计数值寄存器	0
SYST_CALIB	R/W	0x01C	系统定时器校准值寄存器	0x4

[1] 复位值只反映使用位的值，不包括保留位的内容。

19.5.1 系统定时器控制和状态寄存器

SYST_CSR 寄存器包含 SysTick 定时器的控制信息，并提供一个状态标志。该寄存器是 ARM Cortex-M0 内核系统定时器寄存器模块的一部分。该寄存器的位域描述，见 [Section 23.5.4 “System timer, SysTick”](#)。

该寄存器决定了系统滴答定时器的时钟源。

Table 261. 系统定时器控制和状态寄存器 (SYST_CSR - 0xE000 E010) 位域描述

位	符号	描述	复位值
0	ENABLE	SysTick 计数器使能 1 = 允许计数 0 = 禁止计数	0
1	TICKINT	SysTick 中断请求允许 1 = SysTick 中断允许 0 = SysTick 中断禁止 当中断请求允许，系统时钟计数器递减计数至 0 时产生中断。	0
2	CLKSOURCE	SysTick 时钟的时钟源选择 1 = CPU 时钟 0 = 参考时钟	0
15:3	-	保留，用户软件不应该对保留位写入。从保留位读取的值未定义。	NA
16	COUNTFLAG	自上次读该寄存器后，如果 SysTick 定时器计数为 0，则返回 1。	0
31:17	-	保留，用户软件不应该对保留位写入。从保留位读取的值未定义。	NA

19.5.2 系统定时器重载寄存器

每当 SysTick 定时器计数减少到零时，会将 SYST_RVR 寄存器中设置的值装载到 SysTick 定时器中。软件加载该寄存器可以作为定时器初始化的一部分。如果 CPU 运行时在为了使用 SYST_CALIB 值的频率时，则 SYST_RVR 寄存器可读取并作为 SYST_CALIB 的值。

Table 262. 系统定时器重载寄存器 (SYST_RVR - 0xE000 E014) 位域描述

位	符号	描述	复位值
23:0	RELOAD	每当 System Tick 定时器计数减少到零时，将要到装载这个值到 System Tick 定时器。	0
31:24	-	保留，用户软件不应该对保留位写“1”。从保留位读取的值未定义。	NA

19.5.3 系统定时器当前值寄存器

当软件读 SYST_CVR 寄存器时，返回系统滴答计数器当前的计数值。

Table 263. 系统定时器当前值寄存器 (SYST_CVR - 0xE000 E018) 位域描述

位	符号	描述	复位值
23:0	CURRENT	读该寄存器，将返回系统滴答计数器当前的计数值。对该寄存器写任何值都会清除系统滴答计数器和 STCTRL 寄存器的 COUNTFLAG 位。	0
31:24	-	保留，用户软件不应该对保留位写“1”。从保留位读取的值未定义。	NA

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

SYST_CALIB 寄存器的值是由系统配置模块中 SYSTCKCAL 寄存器进行配置。
(见 [Table 34](#))。

Table 264. 系统定时器校准值寄存器 (SYST_CALIB - 0xE000 E01C) 位域描述

位	符号	值	描述	复位值
23:0	TENMS		见 Table 363 。	0x4
29:24	-		保留，用户软件不应该对保留位写 “1” 。从保留位读取的值未定义。	NA
30	SKEW		见 Table 363 。	0
31	NOREF		见 Table 363 。	0

19.6 功能描述

SysTick 定时器是一个 24 位定时器，向下计数到零就会产生一个中断。这样为的是可以提供一个固定的 10 毫秒间隔的中断。SysTick 定时器的时钟来自 CPU 时钟 (系统时钟，参见 [Figure 3](#)) ，或来自被固定到 CPU 时钟频率的一半的参考时钟。为了连续产生特定时间间隔的中断，SYST_RVR 寄存器必须用正确的值进行初始化，以保证获得所需的时间间隔。在 SYST_CALIB 寄存器中有一个默认值，可以通过软件修改。如果设置为默认设置，则会每 10 毫秒产生一次中断。

19.7 定时器计算举例

按如下步骤使用系统滴答定时器：

1. 用重载值 RELOAD 对 SYST_RVR 寄存器进行编程设置，以获得要求的时间间隔。
2. 通过写 SYST_CVR 寄存器来清除该寄存器。这将确保定时器被允许时，定时器将从 SYST_RVR 的值开始计数，而不是从任意值开始计数。
3. 用值 0x7 设置寄存器 SYST_SCR，以允许 SysTick 定时器和 SysTick 定时器中断。

下面的例子说明了不同系统配置时 SysTick 定时器的重载计数值。所有例子计算的中断间隔都为 10 毫秒，是因为 LPC111x/LPC11Cxx 系统时钟设置为 50 MHz。

例 (system clock = 50 MHz)

The system tick clock = system clock = 50 MHz. Bit CLKSOURCE in the SYST_CSR register set to 1 (system clock)

$$\text{RELOAD} = (\text{system tick clock frequency} \times 10 \text{ ms}) - 1 = (50 \text{ MHz} \times 10 \text{ ms}) - 1 = 500000 - 1 = 499999 = 0x0007A11F$$

20.1 如何阅读本章

ADC 模块是所有 LPC111x 和 LPC11Cxx 处理器中相同的部分。

20.2 基本配置

ADC 使用以下寄存器来配置：

1. 引脚：ADC 引脚功能在 IOCONFIG 寄存器块中配置（见 7.4 节）。
2. 电源和外设时钟：在 SYSAHBCLKCTRL 寄存器中，设置位 13 ([Table 21](#))。在运行时，通过设置 PDRUNCFG 寄存器来给 ADC 时钟供电 ([Table 42](#))。

注意：基本的 A/D 转换时钟由 APB(PCLK) 时钟确定。ADC 内含一个可编程的分频器，可将 APB 时钟调整为逐次逼近转换所需的时钟，最大可达 4.5MHz。一个精确的转换需要 11 个时钟周期。

20.3 特点

- 10 位逐次逼近型模拟数字转换器 (ADC)
- 8 位引脚复用输入
- 支持掉电模式
- 测量范围 0—3.6V。不可超过 VDD 电压水平
- 10 位转换时间 $\geq 2.44 \mu\text{s}$
- 具有单个或多个输入的突发转换模式
- 可选择由输入引脚跳变转换或定时匹配信号来触发转换
- 每个 A/D 通道具有单独的转换结果寄存器

20.4 引脚描述

表 265 给出了一个 ADC 相关管脚的简要介绍。

Table 265. ADC 引脚描述

Pin	Type	Description
AD[7:0]	输入	模拟输入 。A/D 转换器单元可以测量任一引脚输入信号的电压 注意： 虽然在数字模式引脚可承受 5V 电压，但当配置为模拟输入时，输入电压不能超过 V _{DD} (3.3V)
V _{DD}	输入	V _{REF} ；参考电压，3.3V

若要通过监控的管脚获得准确的电压读数，必须事先通过 ICON 寄存器选用 ADC 的功能。对于作为 ADC 输入管脚来说，在选用数字功能的情况下仍能获得 ADC 读取值的情况是不可能存在的。在选用数字功能的情况下，内部电路会切断该管脚与 ADC 硬件的连接。

20.5 寄存器描述

ADC 所包含的寄存器如 [Table 266](#) 所示

Table 266. 寄存器概览：ADC (基地址 0x4001 C000)

名称	访问方式	偏移地址	描述	复位值 ^[1]
AD0CR	R/W	0x000	A/D 控制寄存器。A/D 转换开始前，必须写 AD0CR 寄存器来选择工作模式	0x0000 0000
AD0GDR	R/W	0x004	A/D 全局数据寄存器。包含最近一次 A/D 转换的结果	NA
-	-	0x008	保留	-
AD0INTEN	R/W	0x00C	A/D 中断允许寄存器。该寄存器包含的允许位控制每个 A/D 通道的 DONE 标志是否用于产生 A/D 中断	0x0000 0100
AD0DR0	R/W	0x010	A/D 通道0数据寄存器。该寄存器包含在通道0上完成的最近一次转换的结果	NA
AD0DR1	R/W	0x014	A/D 通道1数据寄存器。该寄存器包含在通道1上完成的最近一次转换的结果	NA
AD0DR2	R/W	0x018	A/D 通道2数据寄存器。该寄存器包含在通道2上完成的最近一次转换的结果	NA
AD0DR3	R/W	0x01C	A/D 通道3数据寄存器。该寄存器包含在通道3上完成的最近一次转换的结果	NA
AD0DR4	R/W	0x020	A/D 通道4数据寄存器。该寄存器包含在通道4上完成的最近一次转换的结果	NA
AD0DR5	R/W	0x024	A/D 通道5数据寄存器。该寄存器包含在通道5上完成的最近一次转换的结果	NA
AD0DR6	R/W	0x028	A/D 通道6数据寄存器。该寄存器包含在通道6上完成的最近一次转换的结果	NA
AD0DR7	R/W	0x02C	A/D 通道7数据寄存器。该寄存器包含在通道7上完成的最近一次转换的结果	NA
AD0STAT	RO	0x030	A/D 状态寄存器。该寄存器包含 A/D 所有通道的 DONE 和 OVERRUN 标志，以及 A/D 中断标志	0

[1] 复位值只反映在使用位中保存的数据，不包括保留位的内容

20.5.1 A/D 控制寄存器 (AD0CR – 0x4001 C000)

A/D 控制寄存器中的位可用于选择要转换的 A/D 通道，A/D 转换时间，A/D 模式 和 A/D 启动触发。

Table 267. A/D 控制寄存器 (AD0CR - 地址 0x4001 C000) 位描述

位域	符号	值	描述	复位值
7:0	SEL		选择 AD[7:0] 中的一个引脚作为输入采样和转换的有效引脚。位 0 选中引脚 AD0，位 1 选中引脚 AD1、...、位 7 选中引脚 AD7。 在软件控制模式 (BURST = 0), 只能选中一个通道, 即这些位中只有一个可以被配置为 1。 在硬件扫描模式 (BURST = 1), 多个通道都可被选中, 即这些位中任意位都可以被置成 1。 如果所有位都设置为 0, 则通道 0 自动被选定 (SEL=0x01)。	0x00
15:8	CLKDIV		APB 时钟 (PCLK) 被 CLKDIV +1 分频, 产生 ADC 时钟, 要求小于或等于 4.5MHz。通常, 在软件编程时应该设置该值为最小值, 以产生 4.5MHz 的时钟或略小的时钟, 然而在某些情况下 (如高阻抗模拟信号源) 可能要求一个较慢的时钟。	0
16	BURST		Burst 模式 注意: 如果 BURST 被置为 1, AD0INTEN 中的 ADGINTEN 位 (Table 270) 必须置为 0。	0
		0	软件控制模式: 转换是由软件控制的, 需要 11 个时钟周期 (默认为 0)。	
		1	硬件扫描模式: A/D 转换器以 CLKS 设定的频率不断地转换, 扫描 (如果需要) 有 SEL 选定的引脚。第一次转换对应 SEL 位域中设置为 1 的最低有效位所对应的引脚, 然后再扫描下一个为 1 位所对应的引脚。清除该位可以终止复位转换, 但即使该位被清除, 正在进行的转换仍然会完成。 注意: 当 BURST=1 时 STRAT 位域必须是 000, 否则转换不会启动。	
19:17	CLKS		该位域选定在 BURST 模式下每次转换需要的时钟数, 和存储在 ADDR 寄存器 LS 位域转换结果的精度位数。可在 11 clock (10bit) 和 4 clock (3bit) 之间取值。	000
		0x0	11 clocks / 10 bits	
		0x1	10 clocks / 9 bits	
		0x2	9 clocks / 8 bits	
		0x3	8 clocks / 7 bits	
		0x4	7 clocks / 6 bits	
		0x5	6 clocks / 5 bits	
		0x6	5 clocks / 4 bits	
		0x7	4 clocks / 3 bits	
23:20	-		保留, 用户程序不能向保留位写 1, 从保留位读取的值未定义。	NA
26:24	START		当 BURST 位域为 0 时, 这些位域控制是否开始以及何时开始 A/D 转换。	0
		0x0	不开始 (当清除 PDN 为 0 时, 使用该值)	
		0x1	现在开始转换	
		0x2	当被第 27 位所选择的边沿发生在 PIO0_2/SSEL/CT16B0_CAP0 时, 启动转换	
		0x3	当被第 27 位所选择的边沿发生在 PIO1_5/DIR/CT32B0_CAP0 时, 启动转换	
		0x4	当被第 27 位所选择的边沿发生在 CT32B0_MAT0 ^[1] 时, 启动转换	
		0x5	当被第 27 位所选择的边沿发生在 CT32B0_MAT1 ^[1] 时, 启动转换	
		0x6	当被第 27 位所选择的边沿发生在 CT16B0_MAT0 ^[1] 时, 启动转换	
		0x7	当被第 27 位所选择的边沿发生在 CT16B0_MAT1 ^[1] 时, 启动转换	
27	EDGE		该位只有在 START 位域包含 010—111 时才有意义。在这种情况下:	0
		0	在选定的 CAP/MAT 信号的上升沿启动转换	
		1	在选定的 CAP/MAT 信号的下降沿启动转换	
31:28	-		保留, 用户程序不能向保留位写 1, 从保留位读取的值未定义	NA

20.5.2 A/D 全局数据寄存器 (AD0GDR - 0x4001 C004)

A/D 全局数据寄存器 保存了最近一次 A/D 转换的结果。内容包括数据、DONE 标志、OVERRUN 标志以及与数据相关的 A/D 通道号。

Table 268. A/D 全局数据寄存器 (AD0GDR - 地址 0x4001 C004) 位描述

位域	符号	描述	复位值
5:0	-	保留，这些位总是 0	0
15:6	V_VREF	当 DONE 标志为 1 时，该位域包含的二进制小数表示 ADn 引脚上的电压除以 V _{REF} 电压值的结果。该位域为 0，表明 ADn 引脚上的电压小于等于或接近 V _{SS} 引脚上的电压；该位域为 0x3FFF，表明 ADn 引脚上的电压接近等于或大于 V _{REF} 引脚上的电压。	X
23:16	-	保留，从这些位读取的数据总是 0	0
26:24	CHN	这些位包含被 LS 位设置的通道	X
29:27	-	保留，从这些位读取的数据总是 0	0
30	OVERRUN	在 BURST 模式下，如果一个或多个转换结果丢失该位被置为 1，并在转换产生的结果影响 LS 位之前被覆盖 在非 FIFO 操作下，对该寄存器的读操作会将该位清 0。	0
31	DONE	当一次 A/D 转换结束时该位被置为 1。对该寄存器的读操作以及对 ADCR 寄存器的写操作会将该位清 0。如果一次转换正在进行时，对 ADCR 进行写操作，该位被置位并开始一次新的转换。	0

[1] 注意：这并不要求定时器匹配功能出现在处理器引脚上。

20.5.3 A/D 状态寄存器 (AD0STAT – 0x4001 C030)

A/D 状态寄存器可以查看所有 A/D 通道的状态。各 A/D 通道在 ADDRn 寄存器中的 DONE 标志和 OVERRUN 标志都镜像到 ADSTAT 寄存器中。在 ADSTAT 寄存器中也可以查看中断标志（所有 DONE 标志的逻辑或）

Table 269. A/D 状态寄存器 (AD0STAT - 地址 0x4001 C030) 位描述

位域	符号	描述	复位值
7:0	DONE	这些位镜像出现在每个通道结果寄存器中的 DONE 状态标志位	0
15:8	OVERRUN	这些位镜像出现在每个通道结果寄存器中的 OVERRUN 状态标志位。读 AD0STAT 寄存器可同时查看所有 A/D 通道的状态	0
16	ADINT	该位是 A/D 中断标志。当任一 A/D 通道 DONE 标志有效时该位为 1，并允许通过 ADINTEN 寄存器引起 A/D 中断	0
31:17	-	未使用，从这些位读取的数据总是 0	0

20.5.4 A/D 中断允许寄存器 (AD0INTEN – 0x4001 C00C)

该寄存器用来控制一次转换完成时哪个 A/D 通道产生中断。例如，可能需要对一些通道进行连续转换来监控传感器。应用程序可根据需要读出最近一次转换的结果。在这种情况下，一些 A/D 通道的各转换结束时都不需要产生中断。

Table 270. A/D 中断允许寄存器 (AD0INTEN - 地址 0x4001 C00C) 位描述

位域	符号	描述	复位值
7:0	ADINTEN	这些位控制哪些 A/D 通道在转换完成后产生中断。第 0 位为 1 时，A/D 通道转换完成后产生一个中断；第 1 位为 1 时，A/D 通道 1 转换完成后产生一个中断，以此类推	0x00
8	ADGINTEN	该位为 1，允许在 ADDR 寄存器中的全局 DONE 标志产生一个中断。该位为 0，只有被 ADINTEN7:0 允许的 A/D 通道才会产生中断 注意： 在 BURST 模式下该位必须置为 0 (BURST = 1 在 AD0CR).	1
31:9	-	未使用，从这些位读取的数据总是 0	0

20.5.5 A/D 数据寄存器 (AD0DR0 到 AD0DR7 – 0x4001 C010 到 0x4001 C02C)

A/D 数据寄存器保存了一次 A/D 转换完成的结果，同时还包含转换完成标志和溢出标志。

Table 271. A/D 数据寄存器 (AD0DR0 到 AD0DR7 - 地址 0x4001 C010 到 0x4001 C02C) 位描述

位域	符号	描述	复位值
5:0	-	保留	0
15:6	V_VREF	当 DONE 标志为 1 时，该位域包含的二进制小数表示 ADn 引脚上的电压除以 V _{REF} 电压值的结果。该位域为 0，表明 ADn 引脚上的电压小于等于或接近 V _{SS} 引脚上的电压；该位域为 0x3FFF，表明 ADn 引脚上的电压接近等于或大于 V _{REF} 引脚上的电压。	NA
29:16	-	保留	0
30	OVERRUN	在 BURST 模式下，如果一个或多个转换结果丢失该位被置为 1，并在转换产生的结果影响 LS 位之前被覆盖 在非 FIFO 模式下，对该寄存器的读操作会将该位清 0。	0
31	DONE	当一次 A/D 转换结束该位被置为 1，对该寄存器的读操作将其清 0。	0

20.6 操作

20.6.1 硬件触发的转换

当 ADCR 中的 BURST 位为 0 且 START 字段的值包含在 010-111 之间，则当所选的管脚或定时器匹配的信号发生跳变时，A/D 转换器将启动一次转换。

20.6.2 中断

当 ADSTAT 寄存器中的 ADINT 位为 1 时，会向中断控制器发出一个中断请求。一旦已允许中断（通过 ADINTEN 寄存器）的 A/D 通道的任一 DONE 标志位为 1，ADINT 位就置 1。软件可通过中断控制器中对应 ADC 的中断允许状态位来控制是否因此产生中断。要清零相应的 DONE 标志位，必须读取产生中断的 A/D 通道的结果寄存器。

20.6.3 精度和数字接收器

无论 IOCON 块中管脚的设置如何，A/D 转换器都能够测量任何 ADC 输入管脚的电压，尽管如此，但若在 IOCON 寄存器中将管脚选为 ADC 功能，会使管脚的数字接收器禁能，从而提高转换的精度。（请见 7.3.4 节）。

21.1 如何阅读本章

各种 flash 配置和功能见表 272。

Table 272. LPC111x/LPC11Cx flash 配置

类型编号	Flash	ISP via UART	ISP via C_CAN
LPC1111	8 kB	是	否
LPC1112	16 kB	是	否
LPC1113	24 kB	是	否
LPC1114	32 kB	是	否
LPC11C12/C22	16 kB	是	是
LPC11C14/C24	32 kB	是	是

注意：除了 ISP 和 IAP 命令，寄存器可以访问 flash 控制模块来配置 flash 存储访问时间。见 21.9 节。

21.2 特性

- 在系统编程：在系统编程（ISP）是通过使用 BOOT loader 软件和 UART 串口对片内 Flash 存储器进行编程 / 再编程的方法，这种方法也可以在芯片位于终端用户板时使用。
- 在应用编程：在应用编程（IAP）是通过终端用户的应用代码对片内 Flash 存储进行擦除 / 写操作的方法。
- Flash 访问时间可以通过 flash 控制模块的寄存器来配置。
- 擦除一个扇区的时间为 100 ms ± 5%。一个 256 字节块的编程时间为 1ms ± 5%。

21.3 描述

21.3.1 引导程序

引导程序控制复位后的初始操作，并且通过 UART 或 C_CAN 完成 Flash 存储器的编程。这可能对空设备的初始编程，或对之前编程过的设备进行擦除和重新编程，或通过正在运行的系统中的应用程序对 flash 存储器编程。

每次芯片上电或复位都会执行引导程序代码。引导程序可以执行 ISP 命令处理程序或用户的应用代码。复位后若 PI00_1 引脚为低电平，会被认为是外部硬件请求通过 UART 或 C_CAN 启动 ISP 命令处理程序。

如果提供了 C_CAN 接口（LPC11Cx 部分），那么引脚 PI00_3 处于复位状态，并且引脚 PI00_1 为低电平，就决定了是 UART ISP 还是 C_CAN 的 ISP 例程被调用：

- 如果 PI00_3 是低电平，引导程序配置 C_CAN 的接口和调用 C_CAN ISP 命令处理程序。
- 如果 PI00_3 是高电平，引导程序配置 UART 串行端口，并调用 UART ISP 命令处理程序（这是默认的）。

注意：对没有 C_CAN 接口的处理器，PI00_3 引脚状态没有影响。

假设电源引脚电平正常，当 RESET 引脚上出现上升沿时，那么在最多 3ms 后，会对 PI00_1 引脚信号进行采样，并确定继续执行用户代码还是 ISP 处理程序。如果对 PI00_1 引脚采样的结果为低电平，同时看门狗溢出标志被置位，那么外部硬件启动 ISP 命令处理程序的请求将被忽略。如果并不存在执行 ISP 命令处理程序的请求（PI00_1 在复位后的采样结果为高电平），那么会搜寻有效的用户程序。如果找到有效的用户程序，那么执行控制将由此程序接管。如果并未找到有效的用户程序，那么会唤醒自动波特率例程。

注意：PI00_1 的采样可以通过编程闪存地址 0x0000 02FC 来禁止（见 21.3.7.1 节）。

21.3.2 复位后的存储器映射

引导块的大小为 16 KB，起始地址为存储器单元中的 0x1FFF 0000。引导程序被设计为从该存储器区域运行，但 ISP 和 IAP 软件都会占用部分的片内 RAM。本章后面将描述 RAM 的使用情况。复位后，驻留在片内 Flash 存储器的引导块中的中断向量也变为有效。也就是说，引导块底部的 512 个字节的内容，将出现在起始地址为 0x0000 0000 的存储区域中。

21.3.3 有效用户代码的判定标准

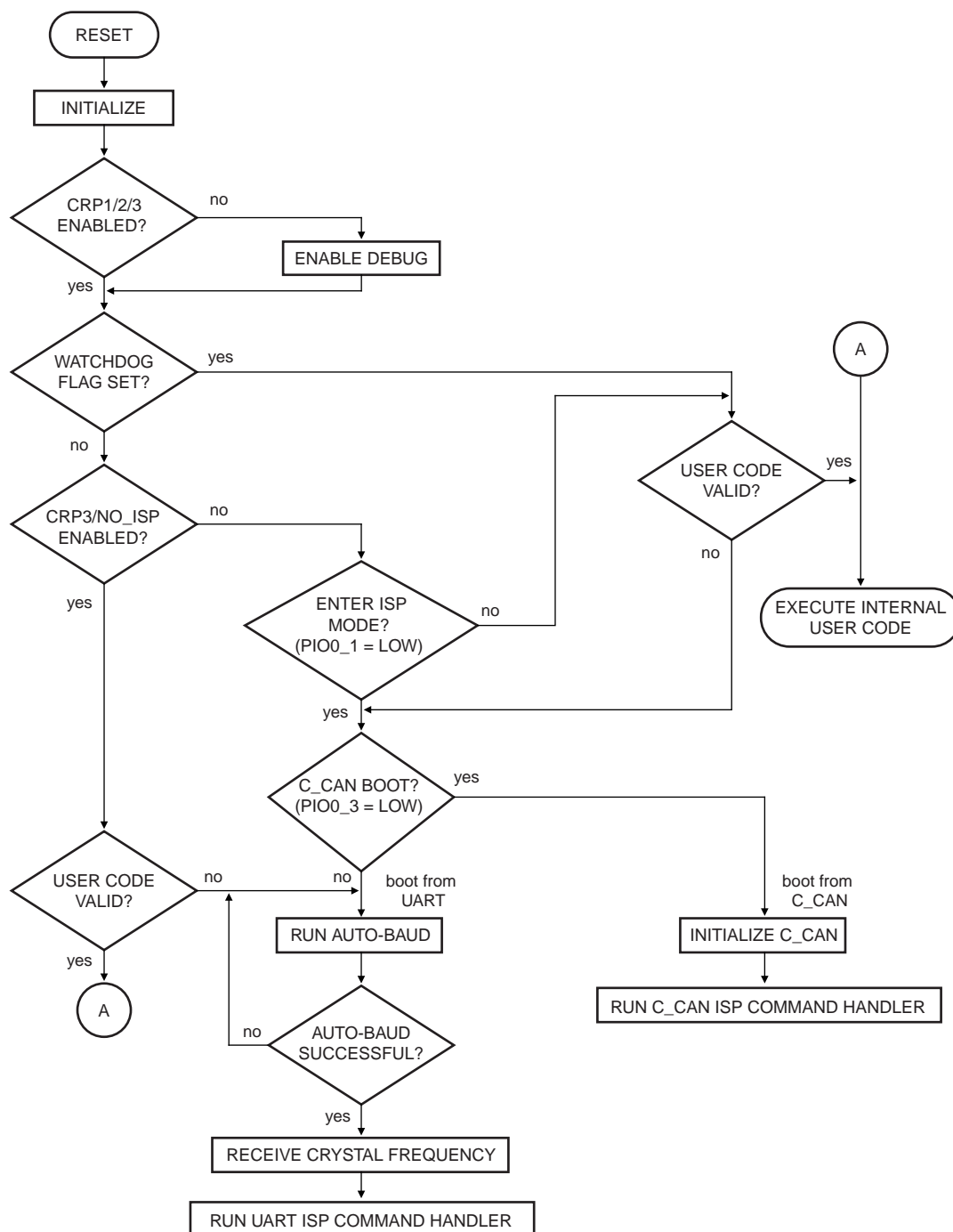
有效用户代码判定标准：保留的 ARM Cortex-M0 异常向量单元 7（在向量表中的偏移量为 0x0000 001C）应当包含表入口 0~6 的校验和的补码。这样就使前 8 个表入口的校验和为 0。引导程序代码校验 Flash 扇区 0 中的前 8 个单元，如果结果为 0，执行控制权便转移给用户代码。

如果签名无效，那么自动波特率程序通过串口 0 与主机进行同步。主机应当发送一个同步字符“?”（0x3F）并等待响应。主机的串口应设定为 8 个数据位和 1 个停止位，无校验位。自动波特率程序根据自身的频率测量接收到的同步字符的位时间，并对串口波特率发生器进行编程。它还向主机发送一个 ASCII 字符串（“Synchronized<CR><LF>”），作为响应，主机应当发送相同的字符串（“Synchronized<CR><LF>”）。自动波特率程序通过观测接收到的字符来验证是否同步。如果通过验证，则向主机发送“OK<CR><LF>”。主机应当通过发送正在运行部分的晶振频率（单位为 KHz）作为响应。例如，如果运行在 10MHz，主机的响应当为“10000<CR><LF>”。在接收到晶振频率后再向主机发送“OK<CR><LF>”。如果同步验证没有通过，那么自动波特率程序再次等待一个同步字符。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 10MHz。

在接收到晶振频率后，设备初始化并调用 ISP 命令处理程序。出于安全性的考虑，执行 Flash 擦除 / 写操作命令和 “Go” 命令之前必须执行 “Unlock（解锁）” 命令。执行其它命令时不需要执行解锁命令。每次 ISP 命令处理都要执行一次 “Unlock（解锁）” 命令。解锁命令在本章 311 页的 21.5 小节 “UART ISP 命令” 中介绍。

.

21.3.4 Boot 执行流程图



(1) 详细信息参考 [Section 21.7.8 “Reinvoke ISP \(IAP\)” on page 328](#)

Fig 68. Boot 执行流程图

21.3.5 扇区数

有些 IAP 和 ISP 命令在某些扇区工作，并说明了扇区号。下表显示 LPC111x/LPC11Cxx 设备的扇区号和内存地址之间的对应关系。

Table 273. Flash 扇区配置

扇区编号	扇区大小	扇区范围	LPC1111 8 kB flash	LPC1112/ LPC11C12/ LPC11C22 16 kB flash	LPC1113 24 kB flash	LPC1114/ LPC11C14/ LPC11C24 32 kB flash
0	4 kB	0x0000 0000 - 0x0000 0FFF	是	是	是	是
1	4 kB	0x0000 1000 - 0x0000 1FFF	是	是	是	是
2	4 kB	0x0000 2000 - 0x0000 2FFF	-	是	是	是
3	4 kB	0x0000 3000 - 0x0000 3FFF	-	是	是	是
4	4 kB	0x0000 4000 - 0x0000 4FFF	-	-	是	是
5	4 kB	0x0000 5000 - 0x0000 5FFF	-	-	是	是
6	4 kB	0x0000 6000 - 0x0000 6FFF	-	-	-	是
7	4 kB	0x0000 7000 - 0x0000 7FFF	-	-	-	是

21.3.6 Flash 内容保护机制

LPC111x/LPC11C1x 是配备了具有纠错能力的 flash 存储器。纠错模块的目的是双重的。首先，它将从存储器读出的数据字解码到输出数据字。其次，它将要写入存储器的数据字进行编码。汉明码具有对一位错误进行纠错的能力。

ECC 对正在运行的程序的操作是透明的，ECC 本身的内容存储在 flash 存储器中，用户的代码既不能读取它也不能写入。一个字节的 ECC 可以访问每连续的 128 位用户 flash 地址。因此，Flash 字节从 0x0000 0000 到 0x0000 000F 被第一个 ECC 字节保护，Flash 字节从 0x0000 0010 到 0x0000 001F 被第二个 ECC 字节保护，等等。

每当 CPU 请求从用户的 Flash 读取，包含指定的内存位置和匹配的 ECC 字节的原始数据 128 位被评估。如果 ECC 机制检测出提取的数据有一个错误，数据在提供给 CPU 之前将被纠错。当有进入用户的 Flash 的写请求时，在写用户指定的内容的同时，一并匹配计算出的 ECC 的值，该值存储在 ECC 存储器中。

当 flash 存储器的一个扇区被擦除时，相应的 ECC 字节也被擦除。一旦一个 ECC 字节被写入，只有当它先被擦除后才能更新。因此，为正确执行实施的 ECC 机制，数据必须以 16 个字节（或 16 的倍数）为一组的形式写入 flash 存储中，对齐方式如上所述。

21.3.7 代码读保护（CRP）

代码读保护是允许用户在系统中通过允许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 命令不受代码读保护的影响。

重要提示：CRP 所作出的任何改变只有在设备经过一个电源周期之后才会生效。

Table 274. 读代码保护选项

名称	在 0x0000 02FC 处编程格式	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 引脚作其他用途。
CRP1	0x12345678	<p>禁止通过 SWD 引脚访问芯片。该模式允许使用以下 ISP 命令和约束来进行局部的 Flash 更新：</p> <ul style="list-style-type: none">• 写 RAM 命令不能访问在地址低于 0x10000300 的 RAM• 将 RAM 内容复制到 Flash 命令不能写扇区 0• 只有在选择擦除所有扇区时，擦除命令才能擦除扇区 0• 比较命令被禁止• 读取存储器命令被禁止 <p>当需要 CRP，且更新 Flash 域时可使用该模式，但不能擦除所有扇区。由于在 Flash 部分更新的情况下比较命令被禁止，因此辅助装载程序应执行校验和机制来验证 Flash 的完整性。</p>
CRP2	0x87654321	<p>禁止通过 SWD 引脚访问芯片。以下 ISP 命令被禁止：</p> <ul style="list-style-type: none">• 读存储器• 写 RAM• 运行• 将 RAM 内容复制到 Flash• 比较 <p>当允许 CRP2 时，ISP 擦除命令仅允许擦除所有用户扇区的内容</p>
CRP3	0x43218765	<p>禁止通过 SWD 引脚访问芯片。如果 Flash 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。</p> <p>该模式有效的禁止了 ISP 通过 PIO0_1 引脚重载。用户的应用程序决定：是调用 IAP 来提供 Flash 更新机制，还是通过 UART 重新唤醒 ISP 命令来进行 Flash 更新。</p> <p>注意：如果选择了 CRP3，此后该设备将无法执行厂商测试。</p>

Table 275. 读代码保护与硬件 / 软件之间的交互

CRP 选项	用户代码是否有效	复位时 PIO0_1 引脚电平	SWD 是否允许	LPC111x/LPC11Cxx 是否进入 ISP 模式	ISP 模式下部分 flash 是否更新
否	否	x	是	是	是
否	是	高	是	否	NA
否	是	低	是	是	是
CRP1	是	高	否	否	NA
CRP1	是	低	否	是	是
CRP2	是	高	否	否	NA
CRP2	是	低	否	是	否
CRP3	是	x	否	否	NA
CRP1	否	x	否	是	是
CRP2	否	x	否	是	否
CRP3	否	x	否	是	否

Table 276. 不同的 CRP 级别所允许的 ISP 命令

ISP 命令	CRP1	CRP2	CRP3 (允许在非 ISP 模式下进入)
解锁	是	是	n/a
设置比特率	是	是	n/a
回应	是	是	n/a
写 RAM	是, 只在 0x1000 0300 之上	否	n/a
读存储器	否	否	n/a
准备写保护分区	是	是	n/a
复制 RAM 到 flash	是, 不到扇区 0	否	n/a
运行	否	否	n/a
擦除扇区	是, 只有当所有的扇区被擦除后扇区 0 才能被擦除	是, 所有扇区	n/a
扇区检测	否	否	n/a
读 ID 号	是	是	n/a
读 BOOT 代码版本	是	是	n/a
比较	否	否	n/a
读 UID	是	是	n/a

若 CRP 模式允许, 并且允许通过 ISP 访问芯片, 不支持或受限制的 ISP 命令将被终止执行, 同时返回代码 CODE_READ_PROTECTION_ENABLED。

21.3.7.1 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 单元编程写入 0x4E69 7370 进入 NO_ISP 模式。

21.4 UART 通信协议

所有 ISP 命令都以单个 ASCII 字符串形式发送。字符串应当以回车 (CR) 和 / 或换行 (LF) 控制字符作为结束符。多余的 <CR> 和 <LF> 将被忽略。所有 ISP 响应都是以 <CR><LF> 结束的 ASCII 字符串形式发送。数据是以 UU 编码格式发送和接收。

21.4.1 UART ISP 命令格式

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

21.4.2 UART ISP 响应格式

“返回_代码<CR><LF> 响应_0<CR><LF> 响应_1<CR><LF> ... 响应_n<CR><LF>” “数据” (数据只适用于读命令)。

21.4.3 UART 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 字节可打印的 ASCII 字符集，该编码的效率高于 Hex 格式 (Hex 格式将 1 字节的二进制数据转换成 2 字节的 ASCII Hex 数据)。发送器在发送 20 个 UU 编码行之后发送校验和。任何 UU 编码行的长度都不应超过 61 个字符 (字节)，也就是说它可以保持 45 个数据字节。接收器应当将该校验和与接收字节的校验和进行比较。如果校验和匹配，接收器响应 “OK<CR><LF>” 来继续下一次发送。如果校验和不匹配，接收器响应 “RESEND<CR><LF>”。作为响应，发送器应当重新发送那些字节。

21.4.4 UART ISP 流量控制

软件 XON/XOFF 流程控制方案可防止缓冲区溢出时的数据丢失。当数据快速到达时，发送 ASCII 控制字符 DC3 (停止) 使数据流停止。发送 ASCII 控制字符 DC1 (启动) 恢复数据流。主机也应支持相同的流控制方案。

21.4.5 UART SP 命令中止

命令可通过发送 ASCII 控制字符 “ESC” 来中止。该特性在 “ISP 命令” 一节中并没有作为一个命令。一旦接收到退出代码，ISP 命令处理器就会等待新的命令。

21.4.6 UART ISP 过程中的中断

在任何复位后，位于 Flash 引导块内的引导块中断向量都有效。

21.4.7 IAP 过程中断

在擦除 / 写操作过程中，片内 Flash 存储器不可访问。只有当用户应用程序代码启动执行时，用户 Flash 区的中断向量才有效。在调用 Flash 擦除 / 写 IAP 之前，用户应当禁止中断或确保用户中断向量在 RAM 中有效，且中断处理程序位于 RAM 中。IAP 代码不能使用

或禁止中断。

21.4.8 ISP 命令处理器使用的 RAM

ISP 命令使用片内地址 0x1000 017C 到 0x1000 025B 范围内的 RAM。用户可以使用该区域，但是在复位时内容可能会丢失。Flash 编程命令使用片内 RAM 最顶端的 32 字节。堆栈位于 RAM 顶端减去 32 字节。可使用的最大堆栈为 256 字节，堆栈是向下递增的。

21.4.9 IAP 命令处理器使用的 RAM

Flash 编程命令使用片内 RAM 最顶端的 32 字节。可使用的最大堆栈为 128 字节，堆栈是向下递增的。

21.5 UART ISP 命令

下面的命令是 ISP 命令处理器所接受的命令。每个命令都支持具体的状态代码。当接收到未定义命令时，命令处理程序会发送返回代码 INVALID_COMMAND。命令和返回代码为 ASCII 格式。

只有当接收到的 ISP 命令执行完毕时，ISP 命令处理器才会发送 CMD_SUCCESS，这时主机才能发送新的 ISP 命令。但“设置波特率”、“写 RAM”、“读存储器”和“运行”命令除外。

Table 277. UART ISP 命令总结

ISP Command	Usage	Described in
解锁	U < 解锁代码 >	见表 278
设置波特率	B < 波特率 > < 停止位 >	见表 279
回应	A <>	见表 280
写 RAM	W < 起始地址 > < 字节数 >	见表 281
读存储器	R < 地址 > < 字节数 >	见表 282
准备写操作的扇区	P <s 扇区开始数> 扇区区结束数 >	见表 283
复制 RAM 到 flash	C <Flash 地址> <RAM 地址> 字节数 >	见表 284
运行	G < 地址 > < 模式 >	见表 285
擦除扇区	E < 开始扇区号 > < 结束扇区号 >	见表 286
扇区查空	I < 扇区开始号> 结束扇区号 >	见表 287
读 ID	J	见表 288
读 BOOT 代码版本	K	见表 290
比较	M < 地址 1> < 地址 2> < 字节数 >	见表 291
读 UID	N	见表 292

21.5.1 解锁 <解锁代码> (UART ISP)

Table 278. UART ISP 解锁命令

命令	U
输入	解锁代码: 23130 ₁₀
返回代码	CMD_SUCCESS INVALID_CODE PARAM_ERROR
描述	该命令用于解锁 Flash 写、擦除和运行命令
举例	“U 23130<CR><LF>” 解锁 Flash 写 / 擦除和运行命令

21.5.2 设置波特率 <波特率> <停止位> (UART ISP)

Table 279. UART ISP 设置波特率命令

命令	B
输入	波特率 : 9600 19200 38400 57600 115200 停止位 t: 1 2
返回代码	CMD_SUCCESS INVALID_BAUD_RATE INVALID_STOP_BIT PARAM_ERROR
描述	该命令用于改变波特率。新的波特率在命令处理程序发送 CMD_SUCCESS 返回代码之后生效。
举例	“B 57600 1<CR><LF>” 设置串口波特率 57600bps 和 1 个停止位。

21.5.3 回应 <设定> (UART ISP)

Table 280. UART ISP 回应指令

命令	A
输入	设定: 开始 = 1 关 =0
返回代码	CMD_SUCCESS PARAM_ERROR
描述	回应命令的默认设定是打开。当打开时，ISP 命令处理器将接收到的串行数据发送回主机。
举例	"A 0<CR><LF>" 回应关闭。

21.5.4 写 RAM <起始地址> <字节数> (UART ISP)

主机应只在接收到 CMD_SUCCESS 返回代码后才发送数据。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 编码行后该值会复位。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。ISP 命令处理器将它与接收字节的校验和进行比较，如果校验和匹配，那么 ISP 命令处理器响应 “OK<CR><LF>” 来继续下一次发送。如果校验和不匹配，那么 ISP 命令处理器响应 “RESEND<CR><LF>”。作为响应，主机应当重新发送字节。

Table 281. UART 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。该地址应当以字为边界 字节数：写入的字节数。计数值应当为 4 的倍数
返回代码	CMD_SUCCESS ADDR_ERROR （地址不是以字为边界） ADDR_NOT_MAPPED COUNT_ERROR （字节计数值不是 4 的倍数） PARAM_ERROR CODE_READ_PROTECTION_ENABLED
描述	该命令用于将数据下载到 RAM。数据应当为 UU 编码格式。当代码读保护使能时该命令被禁止
举例	"W 268436224 4<CR><LF>" 向地址 0x1000 0300 写入四个字节数据

21.5.5 读存储器 < 地址 > < 字节数 > (UART ISP)

数据流之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 编码行后该值会复位。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。主机将它与接收字节的校验和进行比较。如果校验和匹配，那么主机响应 “OK<CR><LF>” 来继续下一次发送。如果校验和不匹配，主机响应 “RESEND<CR><LF>”。作为响应，ISP 命令处理器应当重新发送字节。

Table 282. UART ISP 读存储器命令

Command	R
输入	起始地址：被读出数据字节的地址，该地址应当以字为边界 字节数：读出的字节数。计数值应该为 4 的倍数
返回代码	CMD_SUCCESS，后面是 < 实际数据（UU 编码） > ADDR_ERROR （地址不是以字为边界） ADDR_NOT_MAPPED COUNT_ERROR （字节计数值不是 4 的倍数） PARAM_ERROR CODE_READ_PROTECTION_ENABLED
描述	该命令用于读出 RAM 或 Flash 存储器的数据。当代码读保护使能时该命令禁能
举例	"R 268435456 4<CR><LF>" 从地址 0x1000 0000 读四个字节数据

21.5.6 准备写操作的分区 <起始分区号> <结束分区号> (UART ISP)

此命令使得 flash 存储器写擦除操作分两步处理。

Table 283. UART ISP 准备写操作的扇区

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
返回代码	CMD_SUCCESS BUSY INVALID_SECTOR PARAM_ERROR
描述	该命令必须在执行 “将 RAM 内容复制到 Flash” 或 “擦除扇区” 命令之前执行。“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。该命令不能用于引导块。要准备单个扇区，可将 “起始” 和 “结束” 扇区号设置为相同值
举例	"P 0 0<CR><LF>" 准备 flash 扇区 0

21.5.7 复制 RAM 到 flash <Flash 地址> <RAM 地址> <字节数> (UART ISP)

当写入 Flash，以下限制适用：

- 1. 能够从 RAM 复制到 flash 的最小数据大小为 256 个字节（相当于一页）。
- 2. 一页包含 16 个 Flash 字，每次 flash 能够写入的最小 数是一个 flash 字，有关 ECC 的应用对 flash 写操作的限制，见 21.3.6 节。
- 3. 为了避免写干扰（内在的闪存存储器机制），在对一页进行 16 个连续的写操作后，应该执行一个擦除操作，并且注意到在执行擦除操作之后擦除整个扇区。

注意：一旦一个页面已经写了 16 次后，在未执行擦除操作的情况下仍能够写到同一个扇区的其他页中（假设这些页之前已经擦除了）。

Table 284. UART ISP 复制指令

指令	C
输入	Flash 地址 (DST)：要写入数据字节的目标 Flash 地址。目标地址的边界应当为 256 字节 RAM 地址 (SRC)：读出数据字节的源 RAM 地址 字节数：写入的字节数目。应当为 256 512 1024 4096
返回代码	CMD_SUCCESS SRC_ADDR_ERROR (地址不以字为边界) DST_ADDR_ERROR (地址边界错误) SRC_ADDR_NOT_MAPPED DST_ADDR_NOT_MAPPED COUNT_ERROR (字节计数值不是 256 512 1024 4096) SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION BUSY CMD_LOCKED PARAM_ERROR CODE_READ_PROTECTION_ENABLED
描述	该命令用于编程 Flash 存储器。“准备写操作的扇区”命令应当在该命令之前被执行。当成功执行复制命令后，受影响的扇区将自动再次受到保护。不能通过该命令来写引导块。当代码读保护允许时该命令被阻止。
举例	"C 0 268468224 512<CR><LF>"将RAM 地址0x1000 8000 开始的512 字节复制到Flash 地址 0

21.5.8 运行 <地址> <模式> (UART ISP)

Table 285. UART ISP 运行指令

指令	G
输入	地址：代码执行起始的 Flash 或 RAM 地址。该地址应当以字为边界 模式：T (执行 Thumb 模式下的程序) A (执行 ARM 模式下的程序)
返回代码	CMD_SUCCESS ADDR_ERROR ADDR_NOT_MAPPED CMD_LOCKED PARAM_ERROR CODE_READ_PROTECTION_ENABLED
描述	该命令用于执行位于 RAM 或 Flash 存储器当中的程序。一旦成功执行该命令，就有可能不再返回 ISP 命令处理程序。当代码读保护允许时该命令被阻止。
举例	"G 0 A<CR><LF>" 跳转到 ARM 模式下地址：0x0000 0000

21.5.9 擦除扇区 < 起始扇区号 > < 结束扇区号 > (UART ISP)

Table 286. UART ISP 擦除扇区指令

指令	E
输入	起始扇区号 结束扇区号：应当大于或等于起始扇区号
返回代码	CMD_SUCCESS BUSY INVALID_SECTOR SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION CMD_LOCKED PARAM_ERROR CODE_READ_PROTECTION_ENABLED
描述	该命令用于擦除片内 Flash 存储器的一个或多个扇区。引导块不能使用该命令来擦除。当代码读保护允许时，该命令只允许擦除所有用户扇区的内容
举例	"E 2 3<CR><LF>" 擦除扇区 2 和 3。

21.5.10 扇区查空 < 开始扇区号 > < 结束扇区号 > (UART ISP)

Table 287. UART ISP 扇区查空指令

指令	I
输入	起始扇区号： 结束扇区号：应该大于或等于开始扇区号
返回代码	CMD_SUCCESS SECTOR_NOT_BLANK （后跟 < 第一个非空字的偏移量 > < 非空字的内容 >） INVALID_SECTOR PARAM_ERROR
描述	该命令用于对片内 Flash 存储器的一个或多个扇区进行查空。 对扇区 0 查空总是失败，这是由于前 64 字节重新映射到 Flash 引导块。
举例	"I 2 3<CR><LF>" 对扇区 2 和 3 进行查空

21.5.11 读器件标识号 (UART ISP)

Table 288. UART ISP 读器件标识命令

命令	J
输入	无
返回代码	CMD_SUCCESS 后跟 ASCII 格式的器件标识号（见表 17.18）
描述	该命令用于读取器件的标识号

Table 289. LPC111x 和 LPC11Cxx 系列器件标识号

设备	Hex 编码
LPC111x	
LPC1111FHN33/101	0x041E 502B
LPC1111FHN33/102	0x2516 D02B
LPC1111FHN33/201	0x0416 502B
LPC1111FHN33/202	0x2516 902B

Table 289. LPC111x 和 LPC11Cxx 系列器件标识号

设备	Hex 编码
LPC1112FHN33/101	0x042D 502B
LPC1112FHN33/102	0x2524 D02B
LPC1112FHN33/201	0x0425 502B
LPC1112FHN33/202	0x2524 902B
LPC1113FHN33/201	0x0434 502B
LPC1113FHN33/202	0x2532 902B
LPC1113FHN33/301	0x0434 102B
LPC1113FHN33/302	0x2532 102B
LPC1113FBD48/301	0x0434 102B
LPC1113FBD48/302	0x2532 102B
LPC1114FHN33/201	0x0444 502B
LPC1114FHN33/202	0x2540 902B
LPC1114FHN33/301	0x0444 102B
LPC1114FHN33/302	0x2540 102B
LPC1114FBD48/301	0x0444 102B
LPC1114FBD48/302	0x2540 102B
LPC1114FA44/301	0x0444 102B
LPC1114FA44/302	0x2540 102B
LPC11Cxx	
LPC11C12FBD48/301	0x1421 102B
LPC11C14FBD48/301	0x1440 102B
LPC11C22FBD48/301	0x1431 102B
LPC11C24FBD48/301	0x1430 102B

21.5.12 读 BOOT 代码版本号 (UART ISP)

Table 290. UART ISP 读 BOOT 代码版本号命令

命令	K
输入	无
返回代码	CMD_SUCCESS 后跟 2 字节 ASCII 格式的 Boot 代码版本号 将其解释为 < 字节 1 (主) > < 字节 0 (次) >
描述	该命令用于读取 Boot 代码版本号

21.5.13 比较 < 地址 1> < 地址 2> < 字节数> (UART ISP)

Table 291. UART 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。该地址应当以字为边界。 地址 2 (SRC): 要比较的数据字节的起始 Flash 或 RAM 地址。该地址应当以字为边界。 字节数: 待比较的字节数; 计数值应当为 4 的倍数。
返回代码	CMD_SUCCESS (源和目标数据相同) COMPARE_ERROR (后跟第一个不匹配字节的地址) COUNT_ERROR (字节数不是 4 的倍数) ADDR_ERROR ADDR_NOT_MAPPED PARAM_ERROR
描述	该命令用来比较两个地址单元的存储器内容。 当源或目的地址包含从地址 0 开始的前 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。
举例	"M 8192 268468224 4<CR><LF>"将RAM 地址0x1000 8000 开始的4 个字节与Flash 地址 0x2000 开始的 4 个字节进行比较。

21.5.14 读 UID (UART ISP)

Table 292. UART ISP 读 UID 命令

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

21.5.15 UART ISP 返回代码

Table 293. UART ISP 返回代码总览

返回代码	符号	描述
0	CMD_SUCCESS	成功执行命令。只有成功执行了主机发出的命令后, 才由 ISP 处理器发送该代码。
1	INVALID_COMMAND	无效命令
2	SRC_ADDR_ERROR	源地址没有以字为边界
3	DST_ADDR_ERROR	目标地址的边界错误
4	SRC_ADDR_NOT_MAPPED	源地址的映射不在存储器映射中。无法考证运算值适用之处。
5	DST_ADDR_NOT_MAPPED	目标地址的映射不在存储器映射中。无法考证运算值适用之处。

Table 293. UART ISP 返回代码总览

返回代 码	符号	描述
6	COUNT_ERROR	字节计数值不是 4 的倍数或是一个非法值
7	INVALID_SECTOR	扇区号无效或结束扇区号大于起始扇区号
8	SECTOR_NOT_BLANK	扇区非空
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	为写操作准备扇区的命令未执行
10	COMPARE_ERROR	源和目标数据不相等
11	BUSY	Flash 编程硬件接口忙
12	PARAM_ERROR	参数不足或无效参数
13	ADDR_ERROR	地址没有以字为边界
14	ADDR_NOT_MAPPED	地址的映射不在存储器映射中。无法考证运算值的适用之处
15	CMD_LOCKED	命令被锁定
16	INVALID_CODE	解锁代码无效
17	INVALID_BAUD_RATE	无效波特率设定
18	INVALID_STOP_BIT	无效停止位设定
19	CODE_READ_PROTECTION_ENABLED	代码读保护使能

21.6 C_CAN 通讯协议

注意： C_CAN 接口仅适合于 LPC11Cxx 部分。

如果 PI00_3 在复位时为低电平，并且 ISP 入口被允许（PI00_1 低电平），那么 C_CAN 引导程序由 ROM 复位处理程序将自动激活。C_CAN 引导程序以 100 kbit / s 的 CAN 比特率初始化片上振荡器和 CAN 控制器，并设置自己的 CANopen 节点 ID 为一个固定值。该引导程序然后等待 CANopen SDO 命令和对它们的响应。这些命令允许对对象词典（OD）进行任意的读和写。对象词典包含入口，入口的地址通过一个 16 位的索引和一个 8 位的分索引编址。命令接口是对象词典的一部分。

C_CAN ISP 命令处理程序允许执行所有功能，不然这些功能就要通过 UART ISP 命令执行，见表 294。

.

SDO 指令一直处于接受，执行，和响应状态，直到命令跳转到一个特定的地址执行或者芯片被复位。

C_CAN ISP 处理程序占用固定的 CANopen 结点 ID 125(0x7D)。

Table 294. C_CAN ISP and UART ISP 命令总览

ISP 命令	C_CAN 用法	UART 用法
解锁	21. 6. 3 节	表 278
设置波特率	n/a	表 279
回应	n/a	表 280
写 RAM	21. 6. 4 节	表 281
读存储器	21. 6. 5 节	表 282
准备写扇区操作	21. 6. 6 节	表 283
复制 RAM 到 flash	21. 6. 7 节	表 284
运行	21. 6. 8 节	表 285
擦除扇区	21. 6. 9 节	表 286
扇区查空	21. 6. 10 节	表 287
读 ID 号	21. 6. 11 节	表 288
读 B00T 代码标号	21. 6. 12 节	表 290
读 UID	21. 6. 13 节	表 292
比较	21. 4. 14 节	表 293

21. 6. 1 C_CAN ISP SDO 通讯

CAN 的 ISP 节点侦听 CAN 2. 0A （11 位）消息，该消息带有标识符 0x600，外加节点 ID 0x7D，等于 0x67D。节点发送 SDO 回应并伴随 0x580 加节点号等于 0x5FD 的标识。SDO 通讯条例支持快速和分段。这意味着：通讯永远是确定的，每个请求的 CAN 消息会伴随着一个来自 ISP 结点的消息回复 。

SDO 块传输模式不支持。

有关 SDO 协议的细节，见 *CiA 301 说明*。

21. 6. 2 C_CAN ISP 对象目录

Table 295. C_CAN ISP 对象目录

索引	分类指数	数据类型	通道	描述
0x1000	00	UNSIGNED32	RO	设备类型 (ASCII “LPC1”)
0x1001	00	-	RO	错误寄存 (not used, 0x00)
0x1018	00	-		标识对象
	01	UNSIGNED32	RO	供应号 ID (not used, 0x0000 0000)
	02	UNSIGNED32	RO	部分标识号码
	03	UNSIGNED32	RO	Boot 代码版本号
0x1F50	00	-		程序数据
	01	DOMAIN	RW	程序区
0x1F51	00	-		程序控制
	01	UNSIGNED8	RW	程序控制
0x5000	00	UNSIGNED16	WO	解锁代码
0x5010	00	UNSIGNED32	RW	读存储地址

Table 295. C_CAN ISP 对象目录

索引	分类指数	数据类型	通道	描述
0x5011	00	UNSIGNED32	RW	读存储长度
0x5015	00	UNSIGNED32	RW	写 RAM 地址
0x5020	00	UNSIGNED16	WO	写扇区准备
0x5030	00	UNSIGNED16	WO	擦除扇区
0x5040	00	-		扇区擦空
	01	UNSIGNED16	WO	检查扇区
	02	UNSIGNED32	RO	第一个非空白区的偏移地址
0x5050	00	-		复制 RAM 到 Flash
	01	UNSIGNED32	RW	Flash 地址 (DST)
	02	UNSIGNED32	RW	RAM 地址 (SRC)
	03	UNSIGNED16	RW	字节数
0x5060	00	-		比较存储
	01	UNSIGNED32	RW	地址 1
	02	UNSIGNED32	RW	地址 2
	03	UNSIGNED16	RW	字节数
	04	UNSIGNED32	RO	第一个不匹配的偏移地址
0x5070	00	-		执行地址
	01	UNSIGNED32	RW	执行地址
	02	UNSIGNED8	RO	模式 ('T' 或 'A'), 只支持 'T' 支
0x5100	00	-		序列号
	01	UNSIGNED32	RO	序列号 1
	02	UNSIGNED32	RO	序列号 2
	03	UNSIGNED32	RO	序列号 3
	04	UNSIGNED32	RO	序列号 4

21.6.3 Unlock (C_CAN ISP)

写 <Unlock Code> 到 [0x5000, 0]。编写一个无效的解锁码，将返回一个专门的中止代码。

21.6.4 写 RAM (C_CAN ISP)

写地址 [0x5015, 0] 来设置 RAM 写地址，然后写二进制数据到 [0x1F50, 1]，由于这是一个 DOMAIN 入口，数据可以连续的写入。主机终止写，写地址 [0x5015, 0] 自动递增，因此对 [0x1F50, 1] 的多个连续的写周期可以完成大量数据的写入。

21.6.5 读存储器 (C_CAN ISP)

写地址 [0x5010, 0] 来设置 RAM 读地址，写 [0x5011, 0] 设置 RAM 读长度，然后从地址 [0x1F50, 1] 读二进制数据。由于这是一个 DOMAIN 入口，数据可以连续的读出。当读长度入口所写入的字节数读完时，设备停止读操作。读地址 [0x501 0, 0] 自动递增，因此对 [0x1F50, 1] 的多个连续的读周期可以完成大量数据的读入。

21.6.6 准备写操作扇区 (C_CAN ISP)

写一个 16 位的值到地址 [0x5020, 0]，低 8 位为起始扇区号，高 8 位为结束扇区号。

21.6.7 复制 RAM 到 flash (C_CAN ISP)

参数写入入口 [0x5050, 1 to 3]，写字节数到 [0x5050, 3] 启动编程。

有关写 Flash 操作的限制见 21.5.4 节。

21.6.8 运行 (C_CAN ISP)

写起始地址到 [0x5070, 0]，然后通过写值 0x1 到 [0x1F51, 1] 触发“启动应用”命令。

21.6.9 擦除扇区 (C_CAN ISP)

写一个 16 位的数到 [0x5030, 0]，起始扇区号在低八位，结束扇区号在高八位。

21.6.10 擦除扇区 (C_CAN ISP)

写一个 16 位的数到 [0x5040, 0]，起始扇区号在低八位，结束扇区号在高八位。

如果 SECTOR_NOT_BLANK 中止代码返回，入口 [0x5040, 2] 中包含的第一个非空白位置的偏移量。

21.6.11 读部分 ID (C_CAN ISP)

读 [0x1018, 2]。见表 289。

21.6.12 读 BOOT 版本代码 (C_CAN ISP)

读 [0x1018, 3]

21.6.13 读序列号 (C_CAN ISP)

读 [0x5100, 1 to 4]

21.6.14 比较 (C_CAN ISP)

写入参数到入口 [0x5060, 1 到 3]，写字节数到 [0x5060, 3] 启动比较。

如果 COMPARE_ERROR 终止代码返回，可以从入口 [0x5060, 4] 读出第一个不匹配的偏移地址。

21.6.15 C_CAN ISP SDO 终止代码

触发动作的 OD 入口在动作有误时返回了一个适当的 SDO 终止代码，终止代码是 0x0F00 0000，加上相应的 ISP 返回代码的最低字节。有关终止代码列表见表 296。

此外，用于有效访问 0D 入口的常规 CANopen SDO 终止代码也被支持。

Table 296. C_CAN ISP SDO 终止代码

UART ISP 错误代码	SDO 终止代码	值
ADDR_ERROR	SDOABORT_ADDR_ERROR	0x0F00 000D
ADDR_NOT_MAPPED	SDOABORT_ADDR_NOT_MAPPED	0x0F00 000E
CMD_LOCKED	SDOABORT_CMD_LOCKED	0x0F00 000F
CODE_READ_PROTECTION_ENABLED	SDOABORT_CODE_READ_PROTECTION_ENABLED	0x0F00 0013
COMPARE_ERROR	SDOABORT_COMPARE_ERROR	0x0F00 000A
COUNT_ERROR	SDOABORT_COUNT_ERROR	0x0F00 0006
DST_ADDR_ERROR	SDOABORT_DST_ADDR_ERROR	0x0F00 0003
DST_ADDR_NOT_MAPPED	SDOABORT_DST_ADDR_NOT_MAPPED	0x0F00 0005
INVALID_CODE	SDOABORT_INVALID_CODE	0x0F00 0010
INVALID_COMMAND	SDOABORT_INVALID_COMMAND	0x0F00 0001
INVALID_SECTOR	SDOABORT_INVALID_SECTOR	0x0F00 0007
PARAM_ERROR	SDOABORT_PARAM_ERROR	0x0F00 000C
SECTOR_NOT_BLANK	SDOABORT_SECTOR_NOT_BLANK	0x0F00 0008
SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	SDOABORT_SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	0x0F00 0009
SRC_ADDR_ERROR	SDOABORT_SRC_ADDR_ERROR	0x0F00 0002
SRC_ADDR_NOT_MAPPED	SDOABORT_SRC_ADDR_NOT_MAPPED	0x0F00 0004

21.6.16 完全兼容的 CANopen 差异

尽管引导程序使用 SDO 通讯协议和对象词典的数据组织方式，它不是一个完全的 CiA 301 标准兼容 CANopen 节点，以下特征对于标准来说是不符合或者说不符合的。

- 网络管理消息 (NMT) 处理不可用。
- 心跳消息和条目 0x1017 不可用。
- 采用专用的 SDO 终止代码显示设备错误。
- 为了加快通讯，在 SDO 分段下载 / 写到结点期间，“空”SDO 回应被缩短到一个数据字节，而不是象标准所描述的整整 8 个数据字节。
- 入口 [0x1018, 1] 供应商 ID 从 0x0000 0000 读出，而不是官方分配的唯一供应商 ID。
- 主机必须使用不同的方法来识别 CAN ISP 设备。

21.7 IAP 命令

对于在应用编程来说，应当通过寄存器 r0 中的字指针来调用 IAP 程序，该字指针指向含

有命令代码和参数的存储器（RAM）。IAP 命令的结果返回到寄存器 r1 所指向的结果表。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。参数表应当大到足够保存所有的结果以防结果的数目大于参数的数目。参数传递见图 69。参数和结果的数目根据 IAP 命令而有所不同。参数的最大数目为 5，传送到“将 RAM 内容复制到 Flash”命令。结果的最大数目为 4，由“读 UID”命令返回。命令处理器在接收到一个未定义的命令时发送状态代码 INVALID_COMMAND。IAP 程序是 Thumb 代码，驻留在地址 0x1FFF 1FF0。

可按下面方式使用 C 调用 IAP 函数。

定义 IAP 程序的入口地址。由于 IAP 地址的第 0 位是 1，因此，当程序计数器转移到该地址时会使当前指令集变为 Thumb 指令集。

```
#define IAP_LOCATION 0x1ffff1ff1
```

定义数据结构或指针，将 IAP 命令表和结果表传递给 IAP 函数：

```
unsigned long command[5];
```

```
unsigned long result[4];
```

或

```
unsigned long * command;
```

```
unsigned long * result;
```

```
command = (unsigned long *) 0x...
```

```
result = (unsigned long *) 0x...
```

定义函数类型指针，其包含 2 个参数，无返回值。需要注意的是 IAP 将函数结果和 R1 中的表格基址一同返回。

```
typedef void (*IAP) (unsigned int [ ], unsigned int [ ]);
```

```
IAP iap_entry;
```

设置函数指针：

```
iap_entry=(IAP) IAP_LOCATION;
```

可使用下面的语句来调用 IAP

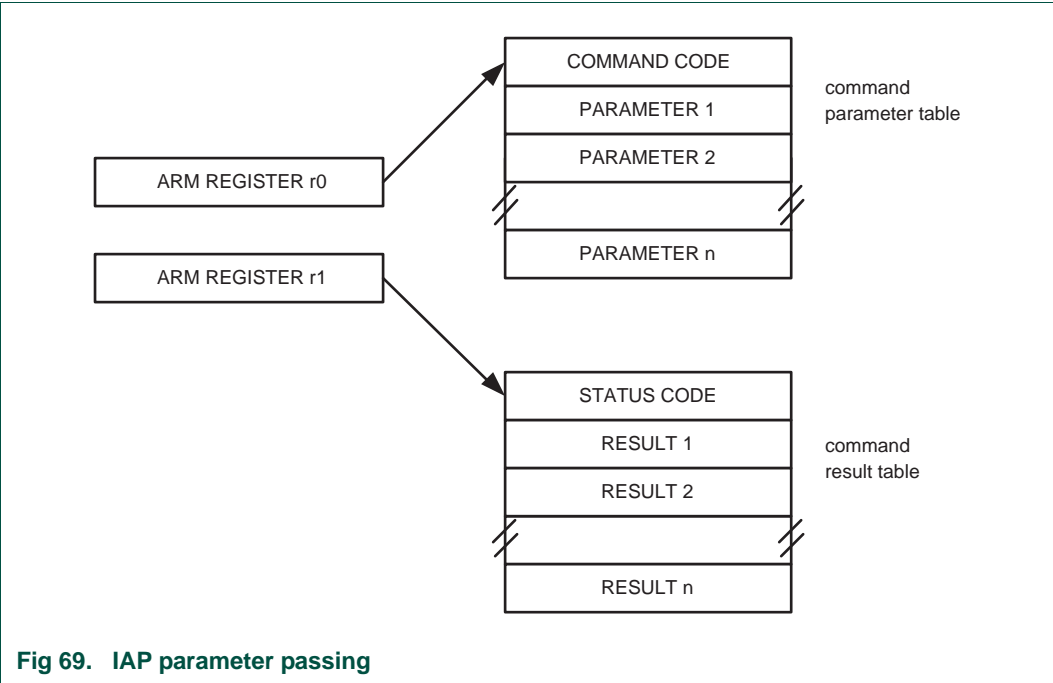
```
iap_entry (command , result);
```

根据 ARM 规范（ARM Thumb 过程调用标准 SWS ESPC 0002 A-05），可分别通过 r0、r1、r2 和 r3 寄存器来传递最多 4 个参数。另外的参数通过堆栈传递。最多 4 个参数可以分别通过 R0、R1、R2 和 R3 寄存器返回。其它的参数通过存储器间接返回。有些 IAP 调用需要多于 4 个参数。如果使用 ARM 建议的机制来传递 / 返回参数，则有可能因为不同厂商所提供的 C 编译器不同而产生问题。使用建议的参数传递机制便可降低这种风险。

在写或擦除操作过程中不能访问 Flash 存储器。那些导致 Flash 写 / 擦除操作的 IAP 命令将使用片内 RAM 顶端的 32 个字节空间来执行。如果应用程序中允许 IAP 编程，那么用户程序不应使用该空间。

Table 297. IAP 命令总结

IAP 命令	命令代码	描述见
准备写操作的扇区	50 (decimal)	表 298
复制 RAM 到 Flash	51 (decimal)	表 299
擦除扇区	52 (decimal)	表 300
扇区擦空	53 (decimal)	表 301
读部分 ID 号	54 (decimal)	表 302
读 BOOT 代码版本号	55 (decimal)	表 303
比较	56 (decimal)	表 304
重新调用 ISP	57 (decimal)	表 305
读 UID	58 (decimal)	表 306



21. 7. 1 准备写操作扇区（IAP）

该命令对 Flash 的写 / 擦除操作分两步。

Table 298. IAP 准备写操作扇区命令

命令	准备写操作扇区命令
输入	命令代码：50（十进制数） 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）

Table 298. IAP 准备写操作扇区命令

命令	准备写操作扇区命令
返回代码	CMD_SUCCESS BUSY INVALID_SECTOR
结果	无
描述	该命令必须在执行 “ 将 RAM 内容复制到 Flash ” 或 “ 擦除扇区 ” 命令之前执行。“ 将 RAM 内容复制到 Flash ” 或 “ 擦除扇区 ” 命令的成功执行会导致相关的扇区再次被保护。不能使用该命令准备启动扇区。要准备单个扇区，可将起始和结束扇区号设为相同值。

21.7.2 复制 RAM 到 flash (IAP)

写 Flash 操作命令的限制见 21.5.4。

Table 299. IAP 复制 RAM 到 flash 命令

命令	复制 RAM 到 flash
I 输入	命令代码: 51 (十进制数) 参数 0 (DST): 要写入数据字节的目标 Flash 地址。目标地址的边界应当为 256 字节 参数 1 (SRC): 读取数据字节的源 RAM 地址。该地址应当以字为边界 参数 2: 写入字节的数目。应当为 256 512 1024 4096 参数 3: 系统时钟频率 (CCLK) (单位: KHz)
返回代码	CMD_SUCCESS SRC_ADDR_ERROR (地址不以字为边界) DST_ADDR_ERROR (地址边界错误) SRC_ADDR_NOT_MAPPED DST_ADDR_NOT_MAPPED COUNT_ERROR (字节计数值不是 256 512 1024 4096) SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION BUSY
结果	无
描述	该命令用于编程 Flash 存储器。受影响的扇区应先通过调用 “ 准备写操作扇区 ” 命令准备好。当成功执行复制命令后，扇区将自动受到保护。该命令不能写引导扇区

21.7.3 擦除扇区 (IAP)

Table 300. IAP 擦除扇区命令

命令	擦除扇区
输入	命令代码: 52 (十进制数) 参数 0: 起始扇区号 参数 1: 结束扇区号 (应当大于或等于起始扇区号) 参数 2: 系统时钟频率 (CCLK) (单位: KHz)
返回代码	CMD_SUCCESS BUSY SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION INVALID_SECTOR
结果	无
描述	该命令用于擦除片内 Flash 存储器的一个或多个扇区。该命令不能擦除引导扇区。 要擦除单个扇区可将 “起始” 和 “结束” 扇区号设定为相同值。

21.7.4 扇区查空 (IAP)

Table 301. IAP 扇区查空命令

命令	扇区查空
输入	命令代码: 53 (十进制数) 参数 0: 起始扇区号 参数 1: 结束扇区号 (应当大于或等于起始扇区号)
返回代码	CMD_SUCCESS BUSY SECTOR_NOT_BLANK INVALID_SECTOR
结果	结果 0: 状态代码为 SECTOR_NOT_BLANK 时第一个非空字位置的偏移量 结果 1: 非空字位置的内容
描述	该命令用于对片内 Flash 存储器的一个或多个扇区进行查空。要查空单个扇区可将 “起始” 和 “结束” 扇区号设定为相同值

21.7.5 读器件标识号 (IAP)

Table 302. IAP 读器件标识号命令描述

命令	读器件标识号
输入	命令代码: 54 (十进制数) 参数: 无
返回代码	CMD_SUCCESS
结果	结果 0: 设备标识号
描述	该命令用于读取设备的标识号

21.7.6 读 BOOT 代码版本号 (IAP)

Table 303. IAP 读 BOOT 代码版本号命令

命令	读 BOOT 代码版本号命令
输入	命令代码: 54 (十进制数) 参数: 无
返回代码	CMD_SUCCESS
结果	结果 0: 2 字节 Boot 代码版本号 (ASCII 格式) 将其解释为 < 字节 1 (主) > < 字节 0 (次) >
描述	该命令用于读取引导代码版本号

21.7.7 比较 < 地址 1> < 地址 2> < 字节数> (IAP)

Table 304. IAP 比较命令

命令	比较
输入	命令代码: 56 (十进制数) 参数 0 (DST): 要被比较的数据字节的 Flash 或 RAM 起始地址。该地址应当以字为边界 参数 1 (SRC): 要被比较的数据字节的 Flash 或 RAM 起始地址。该地址应当以字为边界 参数 2: 待比较的字节数。计数值应当为 4 的倍数
返回代码	CMD_SUCCESS COMPARE_ERROR COUNT_ERROR (字节数不是 4 的倍数) ADDR_ERROR ADDR_NOT_MAPPED
结果	结果 0: 当状态代码为 COMPARE_ERROR 时第一个不匹配字节的偏移地址。
描述	该命令用来比较两个地址单元的存储器内容。 当源或目标地址包含从地址 0 开始的前 512 字节中的任意一个地址时, 比较的结果可能不正确。因为前 512 字节是可以重新映射到 RAM 中的。

21.7.8 重新调用 ISP (IAP)

Table 305. IAP 重新调用 ISP

命令	比较
输入	命令代码: 57 (十进制数)
返回代码	无
结果	无
描述	该命令用来唤醒 ISP 模式中的引导程序。它会映射引导向量, 设定 PCLK = CCLK, 配置 UART 引脚 RXD 和 TXD, 复位计数器 / 定时器 CT32B1 和 U0FDR (见表 135)。当内部 flash 存储器中出现有效的用户程序且 PIO0_1 引脚不可访问时, 可使用该指令强制进入 ISP 模式。

21.7.9 读 UID (IAP)

Table 306. IAP 读 UID 命令

命令	比较
输入	命令代码: 58 (十进制数)
返回代码	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 ID

21.7.10 IAP 状态代码

Table 307. IAP 状态代码总结

状态代 码	符号	描述
0	CMD_SUCCESS	成功执行命令
1	INVALID_COMMAND	无效命令
2	SRC_ADDR_ERROR	源地址不是以字为边界
3	DST_ADDR_ERROR	目标地址的边界错误
4	SRC_ADDR_NOT_MAPPED	源地址的映射不在存储器映射中。无法得到运算值适用对象的位置
5	DST_ADDR_NOT_MAPPED	目标地址的映射不在存储器映射中。无法得到运算值适用对象的位置
6	COUNT_ERROR	字节计数值不是 4 的倍数或是一个非法值
7	INVALID_SECTOR	扇区号无效
8	SECTOR_NOT_BLANK	扇区非空
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	为写操作准备扇区命令未执行
10	COMPARE_ERROR	源和目标数据不相同
11	BUSY	flash 编程硬件接口忙

21.8 调试说明

21.8.1 flash 镜像比较

根据使用的调试器和 IDE 的调试设置的不同，当调试器连接好时，可见的存储器可能是引导 ROM，内部 SRAM，或 flash。为了帮助确定哪些内存是在当前的调试环境，检查 flash 地址 0x0000 0004 的值。这个地址包含在 ARM Cortex - M0 向量表中的代码入口点，这分别是引导 ROM，内部 SRAM，或 flash 存储器的底部。

Table 308. 在调试模式下的内存映射

内存映射模式	存储地址在 0x0000 0004 可见
引导模式	0x1FFF 0000
用户 flash 模式	0x0000 0000
用户 SRAM 模式	0x1000 0000

21.8.2 串行线调试（SWD）flash 编程接口

调试工具可以写部分闪存映像到 RAM，然后按照恰当的偏移地址重复执行 IAP 调用 “将 RAM 内容复制到 Flash”。

21.9 Flash 存储器访问

根据系统时钟频率，通过写入位于 0x4003 C010 地址的 FLASHCFG 寄存器，可以对 flash 存储器访问时间作出多种不同的配置。

注意：若此寄存器设置不当，将导致 LPC111x 的 Flash 存储器的错误操作。

Table 309. Flash 配置寄存器 (FLASHCFG, address 0x4003 C010) 位描述

位	标识	值	描述	复位值
1:0	FLASHTIM		flash 存储器访问时间。FLASHTIM+1 等于 Flash 访问所占用的系统时钟数目	10
		0x0	1 个系统时钟的 Flash 访问时间（用于高于 20MHz 的系统时钟）	
		0x1	2 个系统时钟的 Flash 访问时间（用于高于 40MHz 的系统时钟）	
		0x2	3 个系统时钟的 Flash 访问时间（用于高于 50MHz 的系统时钟）	
		0x3	保留	
31:2	-	-	保留。用户软件不可更改这些位的值。位 31:2 必须以所读取的原值写回。	-

21.10 Flash 签名生成

Flash 模块包含一个内置的签名发生器。该发生器可以产生一系列 Flash 存储器的 128 位签名。一个典型的应用是，把闪存内容与计算的签名相比较（例如，在编程过程中）。

产生签名的地址范围必须对齐到 flash 字边界，即 128 位的边界。一旦被启动，签名的产生就会独立地完成。当签名处于生成过程中时，flash 存储不能因为别的原因而被访问，一个读操作会产生一个等待状态，直到签名发生器完成之后才能执行。flash 存储的外部代码（例如，内部 RAM）在签名发生过程中可以被执行。这包括中断服务，如果中断向量表被重映射到存储器中而不是 flash 存储器中。初始化签名发生的代码应该位于 flash 存储器之外。

21.10.1 签名生成的寄存器描述

Table 310. 寄存器概述：FMC (基地址 0x4003 C000)

名字	访问	偏移地址	描述	复位值	参考
FMSSTART	R/W	0x020	签名的起始地址寄存器	0	表 311
FMSSTOP	R/W	0x024	?????????	0	表 312
FMSW0	R	0x02C	Word 0 [31:0]	-	表 313
FMSW1	R	0x030	Word 1 [63:32]	-	表 314
FMSW2	R	0x034	Word 2 [95:64]	-	表 315
FMSW3	R	0x038	Word 3 [127:96]	-	表 316
FMSTAT	R	0xFE0	签名生成状态寄存器	0	21.10.1.3 节
FMSTATCLR	W	0xFE8	签名生成状态清除寄存器	-	21.10.1.4 节

21.10.1.1 签名生成地址和控制寄存器

此寄存器控制自动签名生成。签名可为 flash 存储内容的任意部分产生。用于签名发生的地址范围为：写起始地址到签名起始地址寄存器 (FMSSTART) 和写终止地址到签名终止地址寄存器 (FMSSTOP)，起始地址与结束地址必须对齐到 128 位边界，并且可以通过将字节地址除以 16 来求出。

签名发生是通过将 FMSSTOP 寄存器中的 SIG_START 位置位来启动的，置位 SIG_START 位典型地是与在一次单一写操作中的签名停止地址相结合的。

表 311 和表 312 分别列举了 FMSSTOP 寄存器的 FMSSTART 位分配。

Table 311. Flash 模块信号开始寄存器 (FMSSTART - 0x4003 C020) 位描述

位	标志	描述	复位值
16:0	START	签名生成的起始地址（对应的 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。从保留位读出的值没有被定义。	NA

Table 312. Flash 模块信号停止寄存器 (FMSSTOP - 0x4003 C024) 位描述

位	标志	值	描述	复位值
16:0	STOP		除以 16（BIST 停止地址，对应的 AHB 字节地址 [20:4]）。	0
17	SIG_START		启动控制位签名生成	0
		0	签名生成停止	
		1	初始化签名生成	
31:18	-		保留，用户软件不应向保留位写入。从保留位读出的值没有被定义。	NA

21. 10. 1. 2 签名发生结果寄存器

签名发生结果寄存器返回嵌入式签名发生器产生的 flash 签名。128 位签名通 4 个寄存器 FMSW0, FMSW1, FMSW2 和 FMSW3 反映。

所产生的 flash 签名可以用来验证 flash 存储的内容。生成的签名可以与预期的签名进行比较，这可以节省时间和空间代码。这种签名生成方法描述见 21. 10. 2 节。

表 316 分别列举了 FMSW0 和 FMSW1, FMSW2, FMSW3 寄存器的位分配方案。

Table 313. FMSW0 寄存器位描述 (FMSW0, address: 0x4003 C02C)

位	标志	描述	复位值
31:0	SW0[31:0]	128 位签名的字 0(位 31 到 0) 。	-

Table 314. FMSW1 寄存器位描述 (FMSW1, address: 0x4003 C030)

位	标志	描述	复位值
31:0	SW1[63:32]	128 位签名的字 1(位 63 到 32) 。	-

Table 315. FMSW2 寄存器位描述 (FMSW2, address: 0x4003 C034)

位	标志	描述	复位值
31:0	SW2[95:64]	128 位签名的字 2(位 95 到 64) 。	-

Table 316. FMSW3 寄存器位描述 (FMSW3, address: 0x4003 40C8)

位	标志	描述	复位值
31:0	SW3[127:96]	128 位签名的字 3(位 127 到 96) 。	-

21.10.1.3 Flash 模块状态寄存器

只读 FMSTAT 寄存器提供了一种方式来决定签名发生何时完成，签名发生的完成可以通过轮询 FMSTAT 的 SIG_DONE 位来检验。SIG_DONE 在启动一个签名发生操作前应该通过 FMSTATCLR 寄存器来清除，否则状态可能是以前操作完成的结果。

Table 317. Flash 模块状态寄存器 (FMSTAT - 0x4003 CFE0) 位描述

Bit	Symbol	Description	Reset value
1:0	-	保留，用户软件不应向保留位写入。从保留位读出的值没有被定义。	NA
2	SIG_DONE	为 1 时，先前开始的签名发生已完成。FMSTATCLR 清除此标志寄存器的描述。	0
31:3	-	保留，用户软件不应向保留位写入。从保留位读出的值没有定义。	NA

21.10.1.4 Flash 模块状态清除寄存器

FMSTATCLR 寄存器是用来清除签名生成完成标志。

Table 318. flash 模块状态清除寄存器 (FMSTATCLR - 0x0x4003 CFE8) bit description

位	标志	描述	复位位
1:0	-	保留，用户软件不应向保留位写入。从保留位读出的值没有被定义。	NA
2	SIG_DONE_CLR	此位写入 1 清除 FMSTAT 寄存器中签名发生完成标志 (SIG_DONE)。	0
31:3	-	保留，用户软件不应向保留位写入。从保留位读出的值没有被定义。	NA

21.10.2 签名生成算法及程序

签名生成

签名可以为 flash 存储内容的任何部分生成，用于签名生成的地址范围为：写起始地址到签名起始地址寄存器 (FMSSTART) 和写终止地址到签名终止地址寄存器 (FMSSTOP)。

签名发生是通过将 FMSSTOP 寄存器中的 SIG_START 位写入 1 来启动的，启动签名发生典型地与定义结束地址相结合，这个工作在相同寄存器的停止位完成。

签名生成的地址范围是与签名生成的时间范围成正比的。读 flash 存储器的签名生成使用自定时的读取机制并且不依赖于任何的为 flash 的配置时序设置。签名生成持续时间的安全估测为：

$$\text{Duration} = \text{int}((60 / \text{tcy}) + 3) \times (\text{FMSSTOP} - \text{FMSSTART} + 1)$$

当签名生成通过软件触发时，持续时间是在 AHB 时钟周期里， t_{cy} 在一个 AHB 时钟内，用纳秒表示。FMSTAT 的 SIG_DONE 位可以通过软件轮询来确定签名生成是何时完成的。

签名生成之后，一个 128 位的签名可以从 FMSW0 寄存器读取到 FMSW3 寄存器，这个 128 位签名反映了从 flash 存储读取的纠错了的数据，该签名反映了 flash 校验位和校验位的值。

内容验证

从 FMSW0 读取到 FMSW3 寄存器的签名必须等于参考签名，计算参考签名的算法见图 70。

```
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
```

Fig 70. 生成 128 位签名的算法

22.1 如何阅读本章

调试功能对所有的 LPC111x and LPC11Cxx 部分是一样的。

22.2 特点

- 支持 ARM 串行线调试模式。
- 直接调试可以访问所有的存储器，寄存器和外设。
- 调试会话不需要目标资源。
- 4 个断点。4 个指令断点可以用来重映射为代码补丁的指令地址，两个数据比较器可以用来重映射为文字值补丁的地址。
- 两个数据观测点可以用来作为触发器。

22.3 介绍

调试功能被集成到 Cortex-M0 处理器中，支持串行线调试功能，ARM Cortex-M0 配置为可支持高达 4 个断点和两个观测点。

22.4 描述

LPC111x/LPC11Cxx 调试采用串行线调试模式。

22.5 引脚描述

下表显示了与调试有关的各引脚功能，其中的一些功能和其它功能共用一个引脚，但不会在同一时间。

Table 319. 串行调试引脚描述

引脚名	类型	描述
SWCLK	输入	串行线时钟 。该引脚是在串行线调试模式下的调试逻辑时钟，该引脚内部上拉。
SWDIO	输入 / 输出	串行线调试数据输入 / 输出 。SWDIO 引脚由外部调试工具使用，用于与 LPC111x/LPC11Cxx 通信，并控制 LPC111x/LPC11Cxx，该引脚内部上拉。

22.6 调试笔记

22.6.1 调试限制

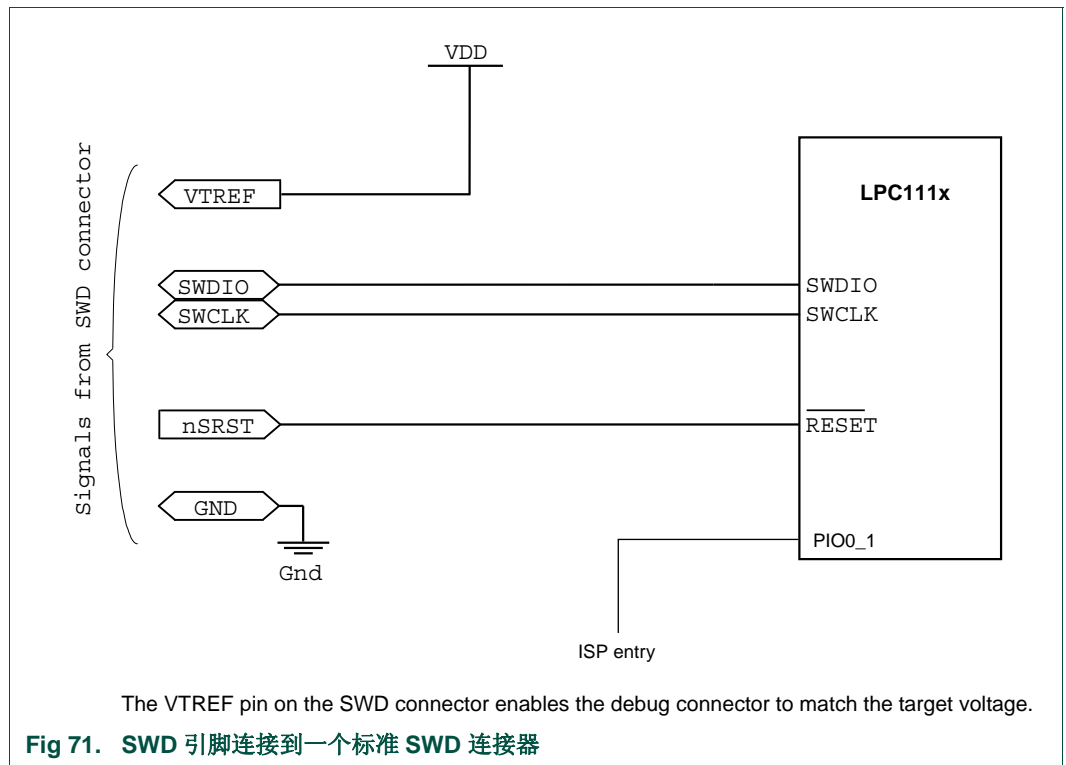
重要提示：用户应该注意调试过程中的具体限制。最重要的一点是，由于 ARM Cortex-M0 处理器集成的限制，LPC111x/LPC11Cxx 不能在通常的方式下从深度睡眠中唤醒，建议不要在调试中使用这种模式。

另外一个问题是调试模式会改变 ARM Cortex-M0 CPU 内部低功耗工作的模式，这有可能波及到整个系统。这些差异意味着调试过程中不应该进行功率测试，测试的结果可能会比通常运行情况下高。

在调试会话期间，系统滴答定时器在 CPU 任意停止的时刻下也会自动停止，而其他外设不会受影响。

22.6.2 调试连接

由于调试的原因，提供访问 ISP 入口引脚 PIO0_1 是有用的。配置系统可能由于某些原因而禁止 SWD 端口，如 PLL 配置不当，重新配置 SWD 引脚为 ADC 输入，进入深度掉电模式不能复位，等等原因，而引脚 PIO0_1 可以用于恢复功能。该引脚也可以用于其他功能，如 GPIO，但在上电或复位时不能保持为低电平。



23.1 介绍

下面的参考资料以 ARM Cortex-M0 用户指南（ARM Cortex-M0 User Guide）为蓝本。只针对 LPC111x/LPC11Cxx Cortex-M0 的具体实现做了细微的改动。ARM Cortex-M0 的文档在文献 1 和文献 2 中可以找到。

23.2 关于 Cortex-M0 处理器和核心外设

Cortex-M0 处理器是一个入门级的 32 位 ARM Cortex 处理器，可用于广泛的嵌入式应用中。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
- 向上与 Cortex-M 系列处理器兼容

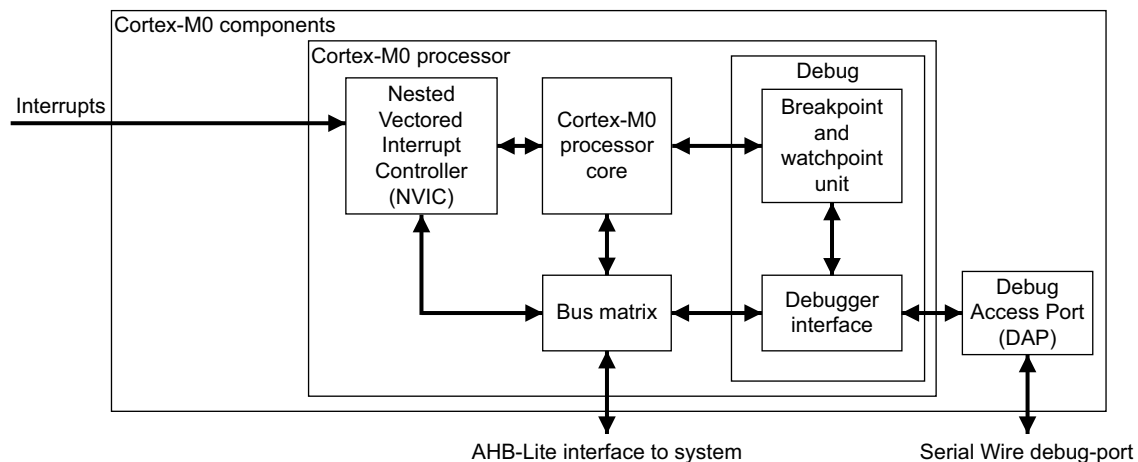


Fig 72. Cortex-M0 的具体实现

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 3 级流水线冯·诺伊曼结构。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 结构，基于 16 位 Thumb 指令集，并包含 Thumb-2 技术。

因而能提供一个现代 32 位体系结构处理器所希望的出色性能，代码密度比其他 8 位和 16 位微控制器都要高。

Cortex-M0 紧密集成了一个可配置的内嵌向量中断控制器（NVIC），提供业界领先的中断性能。NVIC 具有以下功能：

- 包含一个不可屏蔽的中断（NMI）。NMI 在 LPC111x 上不能实现
- 提供零抖动中断选项
- 提供四个中断优先级

处理器内核和 NVIC 的紧密结合使得中断服务程序（ISR）可以快速执行，极大地缩短了中断延迟。这是通过硬件寄存器堆栈、放弃与重启多加载及多存储的能力来获得的。中断程序不需要任何汇编封装代码，不用消耗任何 ISR 代码。尾链优化还极大地降低了从一个 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，提供一个深度睡眠功能，使整个设备迅速降低功耗。

23.2.1 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。

23.2.2 集成的可配置调试

Cortex-M0 处理器实现了完整的硬件调试方案，带有大量的硬件断点和观察点选项。通过一个 2 引脚串行线调试（SWD）端口，为处理器、存储器和外设调试提供了较高的系统可见性。SWD 对微控制器和别的小封装设备是很理想的。

23.2.3 Cortex-M0 处理器特性小结

- 高代码密度，具有 32 位的性能
- 工具和二进制代码向上兼容 Cortex-M 系列处理器
- 集成了极低功耗的睡眠模式
- 高效的代码执行允许更慢的处理器时钟以及更长睡眠模式的时间
- 单周期的 32 位硬件乘法器
- 零抖动的中断处理
- 广泛的调试功能

23.2.4 Cortex-M0 核心外设

Cortex-M0 核心外设：

- NVIC** — NVIC 是一个嵌入式中断控制器，支持低延迟的中断处理
- 系统时钟控制块** — 系统时钟控制块（SCB）是到处理器的编程模型接口。它提供系统执行和控制信息，包括配置、控制和系统异常的报告。
- 系统定时器** — 系统定时器（SysTick）是一个 24 位的减法定时器。可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。

23.3 处理器

23.3.1 编程模型

本节描述了 Cortex-M0 的编程模型。除了对单个内核寄存器的描述之外，本节还包含处理器模式和堆栈的相关信息。

23.3.1.1 处理器模式

处理器模式有：

- Thread 模式（线程模式）** — 用来执行应用软件。处理器在退出复位时进入 Thread 模式。
- Handler 模式（处理模式）** — 用来处理异常。处理器在完成所有的异常处理后返回到 Thread 模式。

23.3.1.2 堆栈

处理器使用满递减堆栈，这就意味着堆栈指针指向堆栈存储器中的最后一个堆栈项。当处理器将一个新的项压入堆栈时，堆栈指针递减，然后将该项写入新的存储器单元。处理器有两个堆栈，主堆栈和进程堆栈，两个堆栈有自己独立的堆栈指针副本，见 [Section 23.3.1.3.2](#)。

在线程模式下，CONTROL 寄存器控制着处理器使用主堆栈还是进程堆栈，见 [Section 23-23.3.1.3.7](#)。在处理器模式下，处理器总是使用主堆栈。处理器操作的选择如下：

Table 320. 处理器模式和堆栈使用的选择

处理器模式	用来执行	使用的堆栈
Thread	应用程序	主堆栈或进程堆栈，见 Section 23-23.3.1.3.7
Handler	异常处理程序	主堆栈

23. 3. 1. 3 内核寄存器

处理器内核寄存器有：

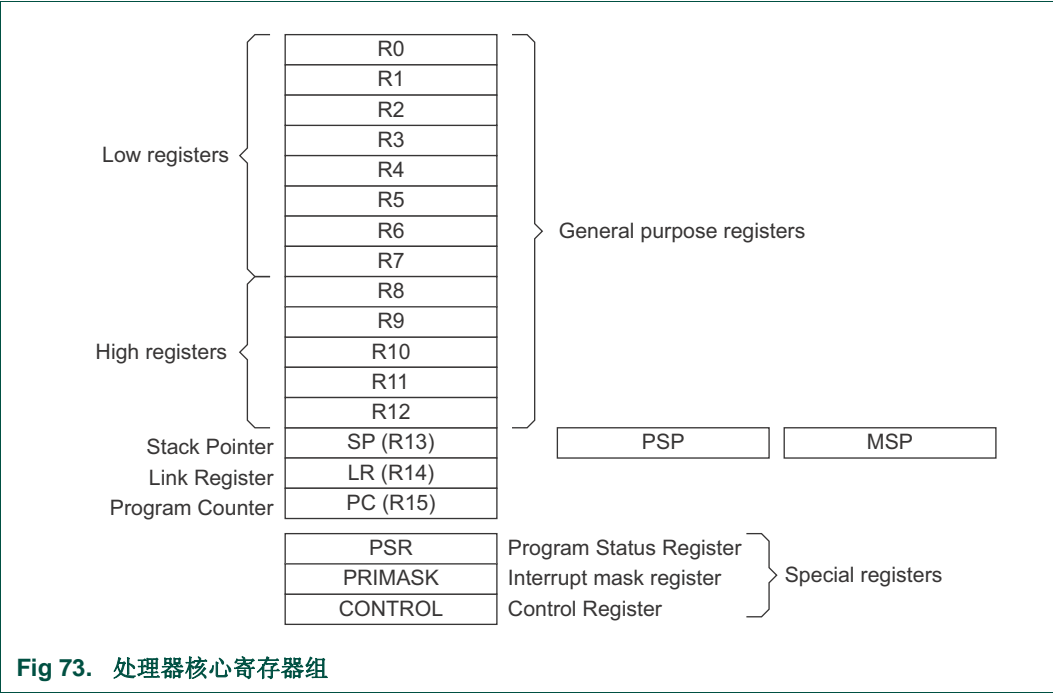


Fig 73. 处理器核心寄存器组

Table 321. 内核寄存器组小结

名称	类型 ^[1]	复位值	描述
R0-R12	RW	不可知	Section 23–23.3.1.3.1
MSP	RW	见描述	Section 23–23.3.1.3.2
PSP	RW	不可知	Section 23–23.3.1.3.2
LR	RW	不可知	Section 23–23.3.1.3.3
PC	RW	见描述	Section 23–23.3.1.3.4
PSR	RW	不可知 ^[2]	Table 23–322
APSR	RW	不可知	Table 23–323
IPSR	RO	0x00000000	Table 324
EPSR	RO	不可知 ^[2]	Table 23–325
PRIMASK	RW	0x00000000	Table 23–326
CONTROL	RW	0x00000000	Table 23–327

[1] 描述线程模式和处理模式下程序执行过程中的访问类型。调试访问可以不同。

[2] Bit[24] 是 T-bit，从复位向量的 bit[0] 加载进来。

23. 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。

23. 3. 1. 3. 2 堆栈指针

堆栈指针（SP）是寄存器 R13。在 Thread 模式中，CONTROL 寄存器的 bit[1] 指示了堆栈指针的使用情况：

- 0 = 主堆栈指针（MSP）。这是复位值。

- 1 = 进程堆栈指针 (PSP)

复位时，处理器将地址 0x00000000 的值加载到 MSP 中。

23.3.1.3.3 链接寄存器

链接寄存器（LR）是寄存器 R14。它保存子程序、函数调用和异常的返回信息。复位时，LR 的值不可知。

23.3.1.3.4 程序计数器

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

23.3.1.3.5 程序状态寄存器

程序状态寄存器（PSR）由下列三种寄存器组合而成：

- 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 的位域分配如下:

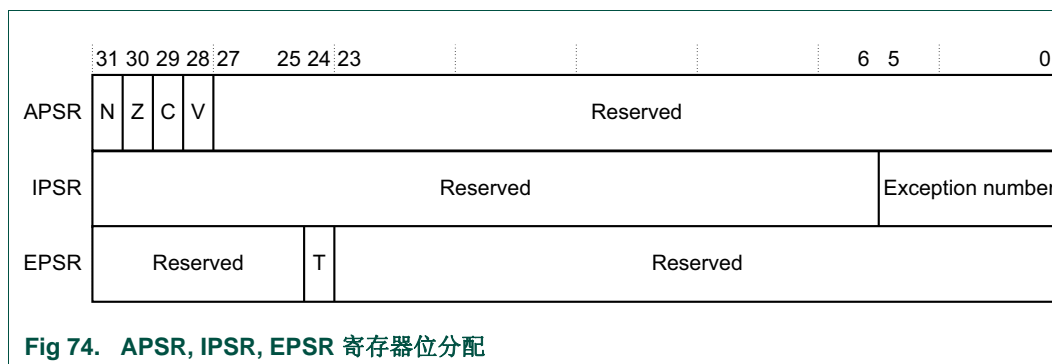


Fig 74. APSR, IPSR, EPSR 寄存器位分配

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:

Table 322. 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 EPSR 位时返回零, 处理器忽略对 EPSR 位的写操作

有关访问程序状态寄存器的更多信息请参看指令描述 [Section 23 - 23.4.7.6](#) 和 [Section 23 - 23.4.7.7](#) 。

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。有关寄存器的属性请见 [Table 23 - 321](#)。寄存器的位分配如下所示：

Table 323. APSR 位分配

位域	名称	功能
[31]	N	负值标志
[30]	Z	零标志
[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 [Section 23.4.4.1.4](#)

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。有关寄存器的属性请见 [Table 23 - 321](#)。该寄存器的位分配如下：

Table 324. IPSR 位分配

位域	名称	功能
[31:6]	-	保留
[5:0]	异常编号	这是当前异常的编号： 0 = Thread 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 Section 23-23.3.3.2

执行程序状态寄存器：EPSR 包含 Thumb 状态位。

有关 EPSR 属性请见 [Table 23 - 321](#) 的寄存器汇总。EPSR 的位分配如下：

Table 325. EPSR 位分配

位域	名称	功能
[31:25]	-	保留
[24]	T	Thumb 状态位
[23:0]	-	保留

如果应用软件使用 MRS 指令直接读取 EPSR 将始终返回零。利用 MSR 指令来写 EPSR 的操作会被忽略。故障处理程序可以检查入栈的 PSR 的 EPSR 值来确定故障的原因。请见本章“23.3.3.6”节。下面的操作可以清除 T 位的值为 0:

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

在 T 位为 0 时尝试执行指令会导致 HardFault 或锁定故障，更多信息请见 [Section 23 - 23.3.4.1](#)

可中断 – 可重启的指令：可中断 – 可重启的指令有 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，处理器就放弃指令的执行。

在处理完中断后，处理器再从头开始重新执行指令。

23.3.1.3.6 异常屏蔽寄存器

异常屏蔽寄存器禁止处理器处理异常。当异常可能影响到时间关键性任务或要求连续执行的原子代码序列时，异常就被禁止。

可以使用 MSR 和 MRS 指令、或 CPS 指令改变 PRIMASK 的值来禁止或重新允许异常。更多信息请看 [Section 23 - 23.4.7.6](#), [Section 23 - 23.4.7.7](#) 和 [Section 23 - 23.4.7.2](#)。

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。有关寄存器的属性请看 [Table 23 - 321](#)。该寄存器的位分配如下:

Table 326. PRIMASK 寄存器位分配

位域	名称	功能
[31:1]	-	保留
[0]	PRIMASK	0 = 无影响 1 = 阻止可配置优先级的所有异常被激活

23.3.1.3.7 控制寄存器

CONTROL 寄存器控制着处理器处于 Thread 模式时所使用的堆栈。该寄存器的属性请看 [Table 23 - 321](#) 的寄存器汇总。该寄存器的位分配如下:

Table 327. CONTROL 寄存器位分配

位域	名称	功能
[31:2]	-	保留
[1]	有效堆栈指针	定义当前的堆栈: 0 = MSP 是当前堆栈指针 1 = PSP 是当前堆栈指针 在 Handler 模式中, 这个位读出的 0, 写操作被忽略
[0]	-	保留

处理模式始终使用 MSP, 因此, 在处理模式下, 处理器忽略对 CONTROL 寄存器的有效堆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堆栈, 内核和异常处理器用主堆栈。

默认情况下, 线程模式使用 MSP。要将线程模式中使用的堆栈指针切换成 PSP, 只需要使用 MSR 指令将有效堆栈指针位设置为 1, 请看 [Section 23 - 23.4.7.6](#)

注意：当更改堆栈指针时, 软件必须在 MSR 指令后立刻使用一个 ISB 指令。这样来保证 ISB 之后的指令执行时使用新的堆栈指针, 请看 [Section 23 - 23.4.7.5](#)。

23.3.1.4 异常和中断

Cortex-M0 处理器支持中断和系统异常。处理器和**内嵌向量中断控制器（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。处理器使用处理模式来处理除复位之外的所有异常, 更多信息请看 [Section 23 - 23.3.3.6.1](#) 和 [Section 23 - 23.3.3.6.2](#)

NVIC 寄存器控制中断处理。更多信息请看 [Section 23 - 23.5.2](#)

23.3.1.5 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。更多信息请看 [Section 23 - 23.3.2.1](#)

23.3.1.6 Cortex 微控制器软件接口标准

ARM 为编程 Cortex-M0 微控制器提供了 **Cortex 微控制器软件接口标准（CMSIS）**。CMSIS 是设备驱动库的一个组成部分。

CMSIS 为 Cortex-M0 微控制器系统定义了：

- 一个通用的方法来：
 - 访问外设寄存器
 - 定义异常向量
- 以下名称：
 - 寄存器和核心外设的名称
 - 内核异常向量的名称
- 一个 RTOS 内核的与设备独立的接口

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

通过允许模板代码的重复使用以及将不同中间件厂商提供的与 CMSIS 兼容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访问函数。

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

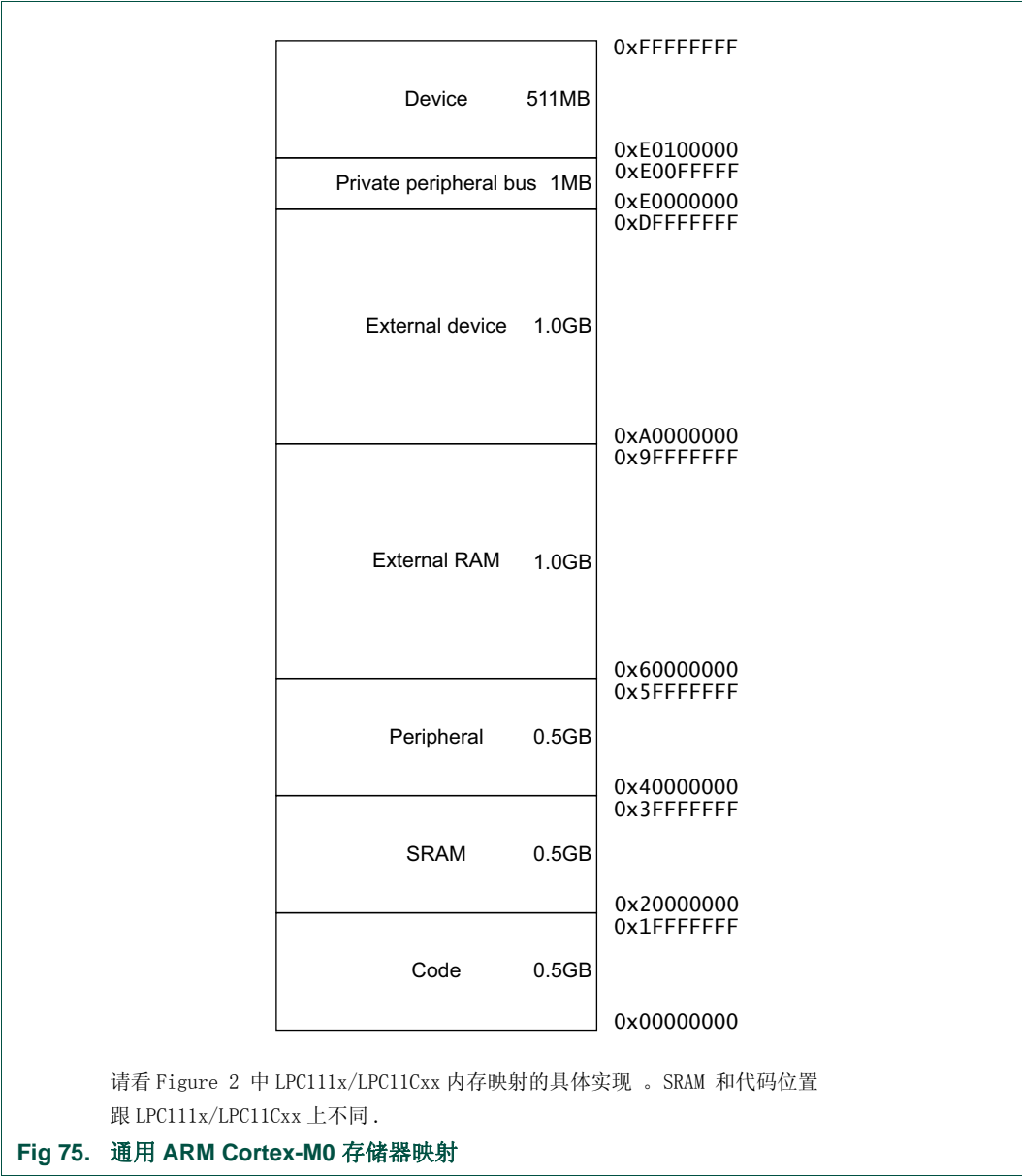
注意： 本文档使用 CMSIS 定义的寄存器缩略名称。在某些情况下，这些名称与其它文档中可能用到的结构缩略名称不同。

下面各节给出了有关 CMSIS 的更多信息：

- [Section 23.3.5.3 “Power management programming hints”](#)
- [Section 23.4.2 “Intrinsic functions”](#)
- [Section 23.5.2.1 “Accessing the Cortex-M0 NVIC registers using CMSIS”](#)
- [Section 23.5.2.8.1 “NVIC programming hints”](#).

23.3.2 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。存储器映射是：



处理器为内核外设寄存器保留了**专用外设总线（PPB）**地址范围空间，请看 [Section 23 - 23.2.](#)

23.3.2.1 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。

存储器类型是：

常规存储器 — 处理器为了提高效率，可以重新对事务进行排序，或者刻意地进行读取。

Device 存储器 — 处理器保留与 Device 存储器或强秩序存储器（Strong-ordered memory）事务相关的事务的秩序。

强秩序存储器 — 处理器保留与所有其他事务相关的事务秩序。

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，但不准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。

23.3.2.2 存储系统的访问秩序

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，只要访问秩序的重新安排不影响指令序列的行为特征就行。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，请看 [Section 23 - 23.3.2.4](#)。

但是，存储器系统不保证 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	Normal access	Device access		Strongly-ordered access
		Non-shareable	Shareable	
Normal access	-	-	-	-
Device access, non-shareable	-	<	-	<
Device access, shareable	-	-	<	<
Strongly-ordered access	-	<	<	<

Fig 76. 存储器排序限制

在表中：

- — 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

23.3.2.3 存储器访问行为

存储器映射中每个区域的访问行为如下：

Table 328. 存储器访问行为

地址范围	存储区域	存储器类型 ^[1]	XN ^[1]	描述
0x00000000-0x1FFFFFFF	Code	常规存储器	-	程序代码的可执行区域。也可以把数据保存到这里。
0x20000000-0x3FFFFFFF	SRAM	常规存储器	-	数据的可执行区域。也可以把代码保存到这里。
0x40000000-0x5FFFFFFF	外设	Device 存储器	XN	外部设备存储器。
0x60000000-0x9FFFFFFF	外部 RAM	常规存储器	-	数据的可执行区域。
0xA0000000-0xDFFFFFFF	外部设备	Device 存储器	XN	外部设备存储器。
0xE0000000-0xE0FFFFFF	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。
0xE0100000-0xFFFFFFF	Device	Device 存储器	XN	厂商提供的特定存储器。

[1] 更多信息请看 [Section 23 - 23.3.2.1](#)。

Code、SRAM 和外部 RAM 区域可以保存程序。

23.3.2.4 软件的存储器访问秩序

程序流程的指令秩序并不能保证相应的存储器事务秩序。这是因为：

- 为了提高效率，处理器可以将一些处理器访问的秩序重新安排，只要不影响指令的行为特性就行。
- 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。

[Section 23 - 23.3.2.2](#) 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。但是，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 — 数据存储屏障**（DMB）指令保证先完成重要的存储事务，再执行后面的存储事务，见 [Section 23 - 23.4.7.3](#)。
- DSB — 数据同步屏障**（DSB）指令保证先完成重要的存储器事务，再执行后面的指令，见 [Section 23 - 23.4.7.4](#)。
- ISB — 指令同步屏障**（ISB）保证所有已完成的存储事务的结果，后面的指令都能辨认出来，见 [Section 23 - 23.4.7.5](#)。

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个入口，然后又允许了相应的异常，那么就在操作之间插入一条 DMB 指令。这就能确保，如果异常在获得允许后立刻被调用，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。这就确保后面的指令执行使用的是更新后的程序。

存储映射切换 — 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。

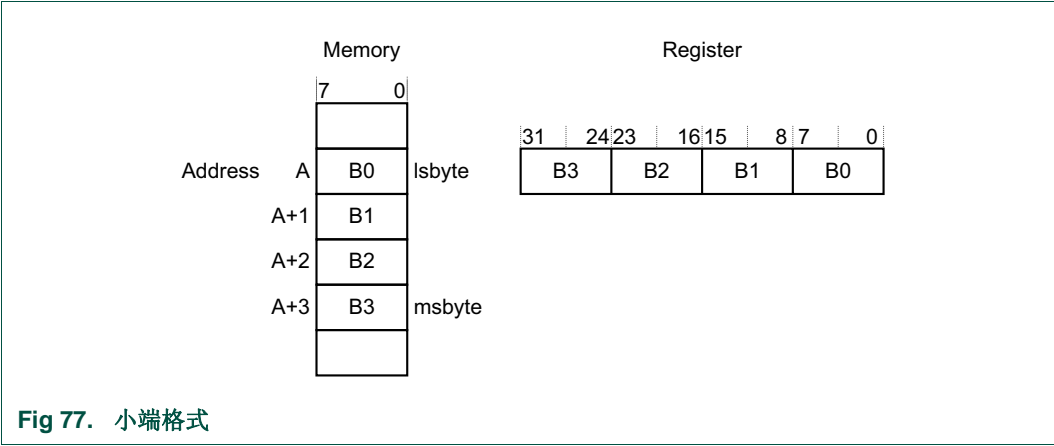
处理器保留与所有其他事务相关的交事务的秩序。

23.3.2.5 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。[Section 23-23.3.2.5.1](#) 描述了数据的字在存储器中是如何存放的

23.3.2.5.1 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：



23.3.3 异常模型

本节描述异常模型。

23.3.3.1 异常状态

每个异常都处于下面状态中的一种：

- 无效** — 异常无效，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。

有效 — 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止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。

有效且挂起 — 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。

23.3.3.2 异常类型

异常类型有：

注意： LPC111x/LPC11Cxx 没有 NMI。

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。当复位结束时，从向量表中复位入口的地址处重新启动执行。在线程模式下执行重启。

NMI — 一个**不可屏蔽的中断**（NMI）可以由外设产生，也可以由软件来触发。这是除复位之外优先级最高的异常。NMI 永远允许，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 内核函数和设备驱动程序。

PendSV — PendSV 是一个中断驱动的系统级服务请求。在 OS 环境下，当没有其它异常有效时，使用 PendSV 来进行上下文切换。

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）是外设引起的一个异常，或者是由软件请求产生的一异常。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。

Table 329. 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 and above	0 and above	中断（IRQ）	可配置 ^[3]	0x00000040 and above ^[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，请看 [Table 23-324](#)。
- [2] 更多信息请看 [Section 23.3.3.4](#)。
- [3] 请看 [Section 23-23.5.2.6](#)。
- [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时间间隔内，处理器可以执行额外的指令。

被特许的软件可以将 [Table 23-329](#) 中列出的优先级可配置的异常禁止，请看 [Section 23-23.5.2.3](#)。

有关 HardFault 的更多信息请看 [Section 23-23.3.4](#)。

23.3.3.3 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 都是由系统处理程序来处理的异常。

23.3.3.4 向量表

向量表包含堆栈指针的复位值以及所有向量处理程序的起始地址（也称为异常向量）。[Figure 23-78](#) 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。

Exception number	IRQ number	Vector	Offset
47	31	IRQ31	0xBC
.		.	.
.		.	.
.		.	.
18	2	IRQ2	0x48
17	1	IRQ1	0x44
16	0	IRQ0	0x40
15	-1	SysTick	0x3C
14	-2	PendSV	0x38
13		Reserved	
12			
11	-5	SVCall	0x2C
10			
9			
8			
7		Reserved	
6			
5			
4			
3	-13	HardFault	0x10
2	-14	NMI	0x0C
1		Reset	0x08
		Initial SP value	0x04

Fig 78. 向量表

向量表的地址固定为 0x00000000。

23. 3. 3. 5 异常优先级

如 [Table 23 - 329](#) 所示，每个异常都有对应的优先级：

- 越小的优先级值表示越高的优先级。
- 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：

- [Section 23 - 23. 5. 3. 7](#)
- [Section 23 - 23. 5. 2. 6.](#)

注意：可配置优先级的值在 0—3 之间。复位、HardFault 和 NMI 这些有固定的负优先级值的异常的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，异常编号最小的挂起异常优先处理。例如，如果 IRQ[0] 和 IRQ[1] 正在挂起，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。

23.3.3.6 异常进入和返回

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，如果另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 [Section 23 - 23.3.3.6.1](#)。

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需要处理
- 已完成的异常处理程序没有在处理一个迟来的异常

处理器从堆栈弹出数据，使处理器状态恢复到中断出现之前的状态，更多信息请看 [Section 23 - 23.3.3.6.2](#)。

尾链 — 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，就跳过堆栈弹出，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，要遵守正常的尾链规则。

23.3.3.6.1 异常进入

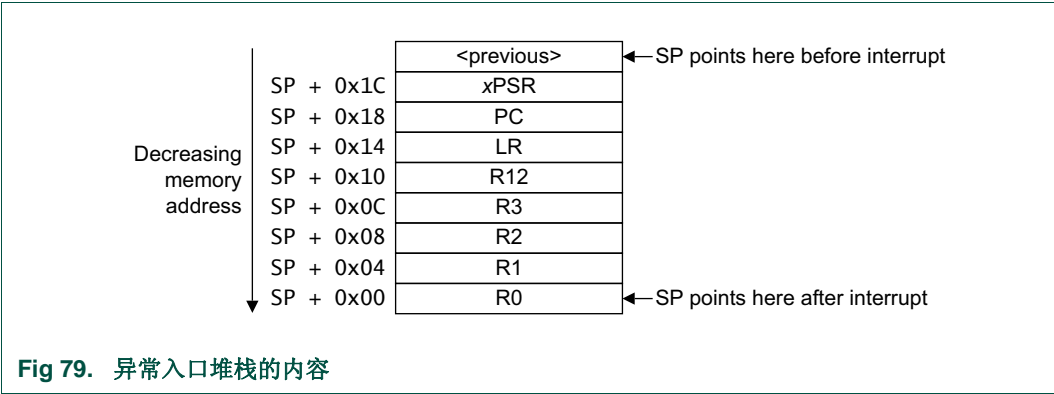
当有一个优先级足够高的挂起异常存在，并且满足下面的任何一个条件，就进入异常处理：

- 处理器处于 Thread 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

当一个异常抢占了另一个异常时，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， 请看 [Section 23 - 23.3.1.3.6](#)。优先级比这个异常低的异常要被挂起， 但不被处理器处理。

当处理器处理异常时， 除非异常是一个末链异常或迟来的异常， 否则， 处理器都把信息压入到当前的堆栈中。这个操作被称为**入栈 (stacking)**， 8 个数据字的结构被称为**栈帧 (stack frame)**。栈帧包含以下信息：



入栈后，堆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堆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。

23.3.3.6.2 异常返回

当处理器处于处理模式，并且执行下面一条指令试图将 PC 设为 EXC_RETURN 值时，出现异常返回：

- POP 指令，用来加载 PC
- BX 指令，使用任意寄存器

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 bit[31:4] 为 0xFFFFFFFF。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。EXC_RETURN 的 bit[3:0] 指出了所需的返回堆栈和处理器模式， 如 [Table 23 - 330](#) 所示：

Table 330. 异常返回行为

EXC_RETURN	描述
0xFFFFFFFF1	返回到处理模式 异常返回从主堆栈获取状态信息 返回后执行使用 MSP
0xFFFFFFFF9	返回到线程模式 异常返回从 MSP 获取状态信息 返回后执行使用 MSP
0xFFFFFFFDD	返回到线程模式 异常返回从 PSP 获取状态信息 返回后执行使用 PSP
All other values	保留

23.3.4 故障处理

故障是异常的一个子集，请看 [Section 23-23.3.3](#)。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：.

- 以等于或高于 SVCa11 的优先级执行 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

注意：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。

23.3.4.1 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，处理器进入一个锁定状态。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

注意：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。

23.3.5 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

SCR 的 SLEEPDEEP 位选择使用哪种睡眠模式， 请看 [Section 23 - 23.5.3.5](#)。

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。

23.3.5.1 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。程序中可以有空闲循环让处理器回到睡眠模式。

23.3.5.1.1 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。更多信息请看 [Section 23 - 23.4.7.12](#) 。

23.3.5.1.2 等待事件

注意：WFE 指令不能在 LPC111x/LPC11Cxx 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

更多信息请看 [Section 23 - 23.4.7.11](#) 。

如果事件寄存器为 1，表明处理器在执行 WFE 指令时不必进入睡眠模式。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 [Section 23 - 23.4.7.9](#)。软件不能直接访问这个寄存器。

23.3.5.1.3 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。

23.3.5.2 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。

23.3.5.2.1 从 WFI 或者 sleep-on-exit 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 位置位来实现这个操作。如果到来的中断被允许，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，请看 [Section 23 - 23.3.1.3.6](#)。

23.3.5.2.2 从 WFE 唤醒

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进入
- 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并唤醒处理器，即使这个中断被禁止，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 [Section 23 - 23.5.3.5](#)。

23.3.5.3 电脑管理编程提示

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
```

23.4 指令集

23.4.1 指令集汇总

处理器执行一个版本的 Thumb 指令集。[Table 331](#) 列出了所支持的指令。

注意：在 [Table 331](#) 中

- 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。

Table 331. Cortex-M0 指令

助记符	操作数	简述	标志	参考
ADCS	{Rd}, Rn, Rm	带进位加法	N,Z,C,V	Section 23–23.4.5.1
ADD{S}	{Rd}, Rn, <Rm #imm>	加法	N,Z,C,V	Section 23–23.4.5.1
ADR	Rd, label	将基于 PC 相对偏移的地址读到寄存器	-	Section 23–23.4.4.1
ANDS	{Rd}, Rn, Rm	位与操作	N,Z	Section 23–23.4.5.1
ASRS	{Rd}, Rm, <Rs #imm>	算术右移	N,Z,C	Section 23–23.4.5.3
B{cc}	label	跳转 { 有条件 }	-	Section 23–23.4.6.1
BICS	{Rd}, Rn, Rm	位清除	N,Z	Section 23–23.4.5.2
BKPT	#imm	断点	-	Section 23–23.4.7.1
BL	label	带链接的跳转	-	Section 23–23.4.6.1
BLX	Rm	带链接的间接跳转	-	Section 23–23.4.6.1
BX	Rm	间接跳转	-	Section 23–23.4.6.1
CMN	Rn, Rm	比较负值	N,Z,C,V	Section 23–23.4.5.4
CMP	Rn, <Rm #imm>	比较	N,Z,C,V	Section 23–23.4.5.4
CPSID	i	更改处理器状态，关闭中断	-	Section 23–23.4.7.2
CPSIE	i	更改处理器状态，允许中断	-	Section 23–23.4.7.2
DMB	-	数据内存屏障	-	Section 23–23.4.7.3
DSB	-	数据同步屏障	-	Section 23–23.4.7.4
EORS	{Rd}, Rn, Rm	异或	N,Z	Section 23–23.4.5.2
ISB	-	指令同步屏障	-	Section 23–23.4.7.5
LDM	Rn{!}, reglist	加载多个寄存器，访问之后会递增地址	-	Section 23–23.4.4.5
LDR	Rt, label	从基于 PC 相对偏移地址上加载寄存器	-	Section 23–23.4.4
LDR	Rt, [Rn, <Rm #imm>]	用字加载寄存器	-	Section 23–23.4.4
LDRB	Rt, [Rn, <Rm #imm>]	用字节加载寄存器	-	Section 23–23.4.4
LDRH	Rt, [Rn, <Rm #imm>]	用半字加载寄存器	-	Section 23–23.4.4
LDRSB	Rt, [Rn, <Rm #imm>]	用有符号的字节加载寄存器	-	Section 23–23.4.4
LDRSH	Rt, [Rn, <Rm #imm>]	用有符号的半字加载寄存器	-	Section 23–23.4.4
LSLS	{Rd}, Rn, <Rs #imm>	逻辑左移	N,Z,C	Section 23–23.4.5.3
U	{Rd}, Rn, <Rs #imm>	逻辑右移	N,Z,C	Section 23–23.4.5.3
MOV{S}	Rd, Rm	传输	N,Z	Section 23–23.4.5.5
MRS	Rd, spec_reg	从特殊寄存器传输到通用寄存器	-	Section 23–23.4.7.6
MSR	spec_reg, Rm	从通用寄存器传输到特殊寄存器	N,Z,C,V	Section 23–23.4.7.7
MULS	Rd, Rn, Rm	乘法，32 位结果值	N,Z	Section 23–23.4.5.6
MVNS	Rd, Rm	位非	N,Z	Section 23–23.4.5.5
NOP	-	无操作	-	Section 23–23.4.7.8
ORRS	{Rd}, Rn, Rm	逻辑或	N,Z	Section 23–23.4.5.2
POP	reglist	出栈，将堆栈的内容放入寄存器	-	Section 23–23.4.4.6
PUSH	reglist	压栈，将寄存器的内容压入堆栈	-	Section 23–23.4.4.6
REV	Rd, Rm	反转字里面的字节顺序	-	Section 23–23.4.5.7
REV16	Rd, Rm	反转每半字里面的字节顺序	-	Section 23–23.4.5.7

Table 331. Cortex-M0 指令

助记符	操作数	简述	标志	参考
REVSH	<i>Rd, Rm</i>	反转有符号半字里面的字节顺序	-	Section 23–23.4.5.7
RORS	<i>{Rd,} Rn, Rs</i>	循环右移	N,Z,C	Section 23–23.4.5.3
RSBS	<i>{Rd,} Rn, #0</i>	反向减法	N,Z,C,V	Section 23–23.4.5.1
SBCS	<i>{Rd,} Rn, Rm</i>	进位减法	N,Z,C,V	Section 23–23.4.5.1
SEV	-	发送事件	-	Section 23–23.4.7.9
STM	<i>Rn!, reglist</i>	存储多个寄存器，在访问后地址递	-	Section 23–23.4.4.5
STR	<i>Rt, [Rn, <Rm>#imm]</i>	将寄存器作为字来存储	-	Section 23–23.4.4
STRB	<i>Rt, [Rn, <Rm>#imm]</i>	将寄存器作为字节来存储	-	Section 23–23.4.4
STRH	<i>Rt, [Rn, <Rm>#imm]</i>	将寄存器作为半字来存储	-	Section 23–23.4.4
SUB{S}	<i>{Rd,} Rn, <Rm>#imm</i>	减法	N,Z,C,V	Section 23–23.4.5.1
SVC	<i>#imm</i>	超级用户调用	-	Section 23–23.4.7.10
SXTB	<i>Rd, Rm</i>	符号扩展字节	-	Section 23–23.4.5.8
SXTH	<i>Rd, Rm</i>	符号扩展半字	-	Section 23–23.4.5.8
TST	<i>Rn, Rm</i>	基于测试的逻辑与	N,Z	Section 23–23.4.5.9
UXTB	<i>Rd, Rm</i>	0 扩展字节	-	Section 23–23.4.5.8
UXTH	<i>Rd, Rm</i>	0 扩展半字	-	Section 23–23.4.5.8
WFE	-	等待事件	-	Section 23–23.4.7.11
WFI	-	等待中断	-	Section 23–23.4.7.12

23. 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 或有可能由 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 提供下列的内部函数来产生 ISO/IEC C 代码不能直接访问的指令：

Table 332. 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)
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：

Table 333. 访问特别寄存器的内部函数

特定寄存器	访问方式	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)

23. 4. 3 关于指令的描述

下列小节对如何使用指令进行了更为详细的描述：

- [Section 23.4.3.1 “Operands”](#)
- [Section 23.4.3.2 “Restrictions when using PC or SP”](#)
- [Section 23.4.3.3 “Shift Operations”](#)
- [Section 23.4.3.4 “Address alignment”](#)
- [Section 23.4.3.5 “PC-relative expressions”](#)
- [Section 23.4.3.6 “Conditional execution”](#).

23. 4. 3. 1 操作数

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

23. 4. 3. 2 使用 PC 或 SP 的限制

对于用于操作数或目的寄存的**程序计数器（PC）**或**堆栈指针（SP）**，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。

注意：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。

23. 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的寄存器中。允

许的移位长度由移位类型和指令决定，请参考各个指令的描述。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。本节中的各小节描述了各种的移位操作以及它们是如何影响进位标志的。在这些描述中，Rm 是包含着移位值的寄存器，n 是移位长度。

23. 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左边的 n 位中，请看 [Figure 23 - 80](#)。

用户可以使用 ASR 对寄存器 Rm 的带符号数进行除以 2ⁿ 的操作，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]。

备注：

- 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。

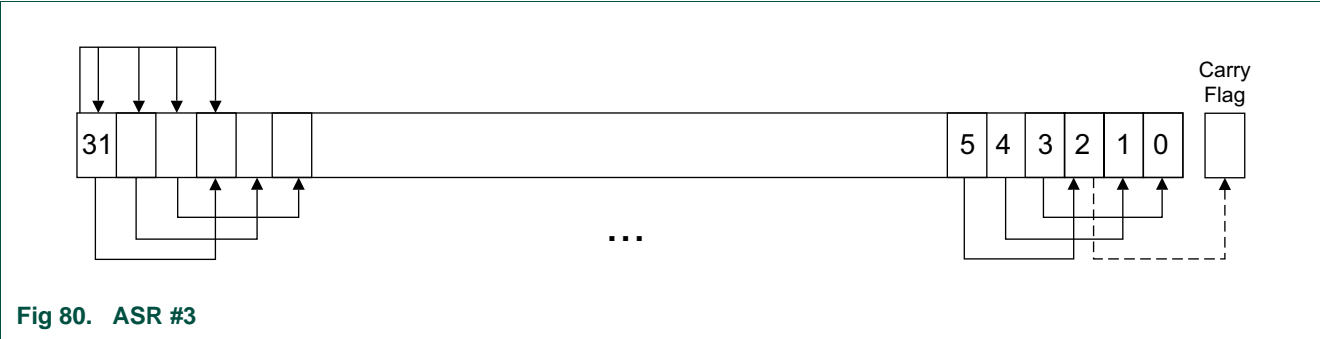


Fig 80. ASR #3

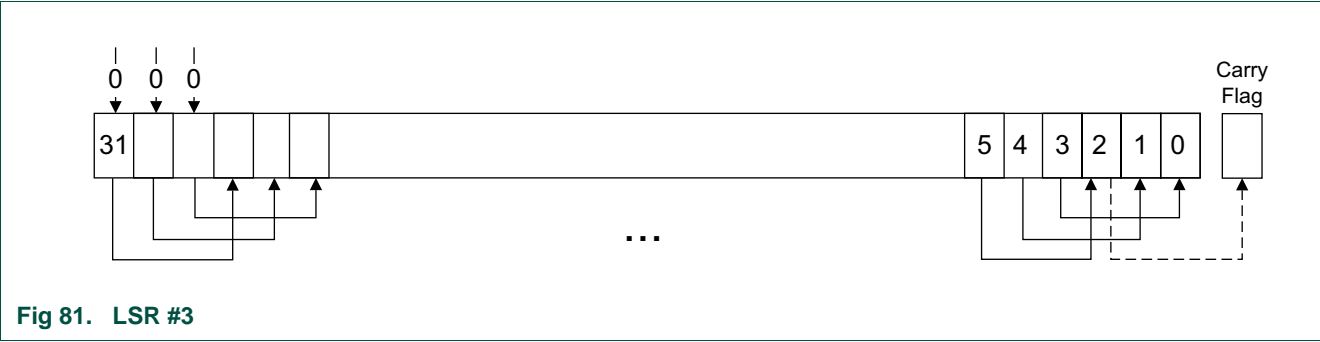
23. 4. 3. 3. 2 LSR

逻辑右移 n 位的操作是将 Rm 寄存器 Rm 左边的 32-n 个位向右移动 n 位，结果寄存器右边有 32-n 位，然后再将结果寄存器左边的 n 个位设为 0。请看 [Figure 81](#)。

如果寄存器 Rm 值为无符号的整数，用户可以使用 LSR 操作来对其值进行除以 2ⁿ 的操作。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]。

备注：

- 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。



23. 4. 3. 3. 3 LSL

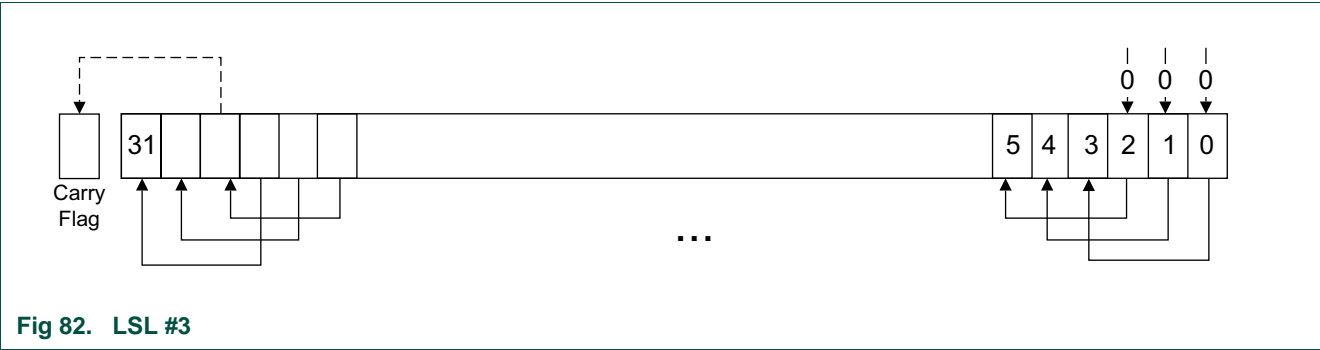
逻辑左移 n 位的操作是将 Rm 寄存器右边的 32-n 个位向左移动 n 位，结果寄存器左边有 32-n 个位，然后将结果中的寄存器右边的 n 个位设为 0。请看 [Figure 82](#)。

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

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。

备注：

- 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。



23. 4. 3. 3. 4 ROR

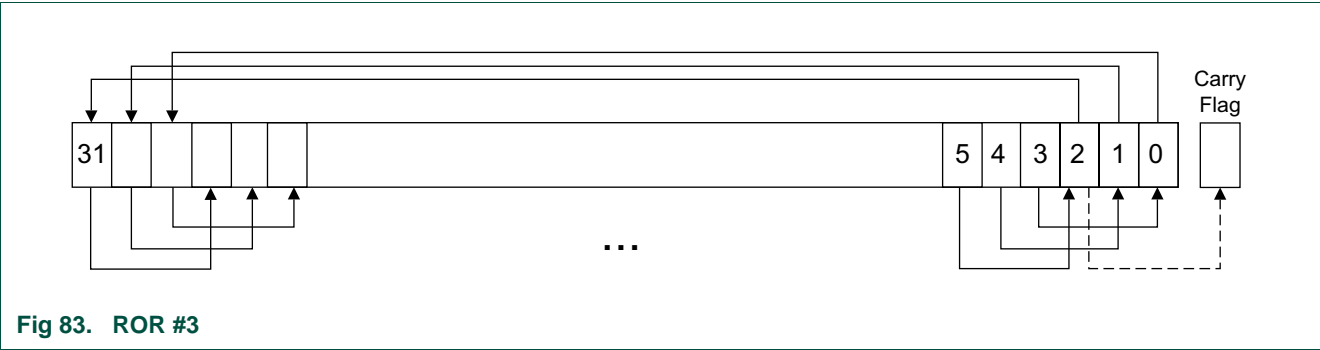
循环右移 n 位的操作是将 Rm 寄存器左边的 32-n 个位向右移动 n 位，结果是寄存器右边有 32-n 个位，然后再将寄存器右边的 n 个位移动到结果寄存器的左边的 n 个位中。请看

Figure 23 - 83。

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 [n-1] 位。

备注：

- 如果 n 为 32，那么结果与 Rm 中的值相同，且如果进位标志被更新，则会被更新为 Rm 的位 [31] 的值。
- 移位长度 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。



23. 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。

23. 4. 3. 5 PC 的相对表达式

相对 PC 表达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。汇编器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，则汇编器会产生一个错误。

备注：

- 对于大多数指令，PC 的值就是当前指令的地址加上 4 个字节。
- 汇编器可能允许用其它语法来表示 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。

23. 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。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。指令对标志的影响，请参考指令的描述。

在如下的情况下，用户可以在另一个指令中设置条件标志的基础上：

- 在指令更新标志后可立即执行条件性的跳转指令
- 在经过任意数量的没有更新标志的间隔数指令后，可以执行条件性的跳性指令

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：

- [Section 23.4.3.6.1 “The condition flags”](#)
- [Section 23.4.3.6.2 “Condition code suffixes”](#).

23.4.3.6.1 条件标志

APSR 包含了下列的条件标志：

- 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

关于 APSR 的更多信息请看 [Section 23 - 23.3.1.3.5](#)。

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
- 如果从一个负值减去一个正值得到一个正值
- 如果从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，则与加法操作相同，结果值会被丢弃除外。更多信息，详情请参考指令描述。

23.4.3.6.2 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。

[Table 334](#) 显示了使用的条件代码，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。

Table 334. 条件代码后缀

后缀	标志	意义
EQ	Z = 1	相等，最后标志设置结果为 0
NE	Z = 0	不相等，最后标志设置结果为非 0
CS or HS	C = 1	更高或相同，无符号
CC or 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 and Z = 0	更高，无符号
LS	C = 0 or Z = 1	更低或相同，无符号
GE	N = V	大于或等于，有符号
LT	N != V	少于，有符号
GT	Z = 0 and N = V	大于，有符号
LE	Z = 1 and N != V	少于或等于，有符号
AL	Can have any value	总是。当没有指定后缀时，这是默认的操作

23. 4. 4 存储器访问指令

[Table 335](#) 所示为存储器访问指令：

Table 335. 访问指令

助记符	简单描述	参考
LDR{type}	使用寄存器偏移量来加载寄存器	Section 23–23.4.4.3
LDR	基于 PC 相对地址来加载寄存器	Section 23–23.4.4.4
POP	出栈，将栈中的内容放入寄存器	Section 23–23.4.4.6
PUSH	压栈，将寄存器的内容压入堆栈	Section 23–23.4.4.6
STM	存储多个寄存器	Section 23–23.4.4.5
STR{type}	使用立即数偏移量来存储寄存器	Section 23–23.4.4.2
STR{type}	使用寄存器偏移量来存储寄存器	Section 23–23.4.4.3

23. 4. 4. 1 ADR

产生一个 PC 相对地址。

23. 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。请看 [Section 23 - 23.4.3.5](#)。

23.4.4.1.2 操作

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

ADR 指令对产生与存储位置无关的代码非常便利，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。

23.4.4.1.3 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。

23.4.4.1.4 条件标志

该指令不会改变标志。

23.4.4.1.5 示例

```
ADR    R1, TextMessage    ; 将被标签为 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
```

23.4.4.2 LDR and STR, 立即数偏移量

具有立即数偏移量的加载和存储。

23.4.4.2.1 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 是寄存器，存储器地址基于此寄存器。

Imm 是 Rn 的偏移量。如果 imm 被省略，则假设它为 0。

23.4.4.2.2 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。

23.4.4.2.3 限制

在这些指令中：

- Rt 和 Rn 必须只指定 R0-R7 的值
- 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 操作
- 计算出的地址必须能够被事务中的字节数整除，请看 [Section 23 - 23.4.3.4](#)。

23.4.4.2.4 条件标志

这些指令不改变标志。

23.4.4.2.5 Examples

```
LDR    R4, [R7]           ; 将 R7 的值作为地址，将此地址处的值载入到 R4 中
STR    R2, [R0, #const-struct] ; const-struct 是评估处于 0-1020 范围内的常量的表达式
```

23.4.4.3 LDR and STR, 寄存器偏移量

带寄存器偏移量的加载和存储。

23.4.4.3.1 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 是寄存器，存储器地址基于此寄存器
- Rm 是含有用作偏移量的值的寄存器

23.4.4.3.2 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 和 Rm 所指定的寄存器中的值之和。

23.4.4.3.3 限制

在这些指令中：

- Rt、Rn 和 Rm 必须指定 R0-R7
- 计算出的地址必须能够被加载或存储的字节数整除。 请看 [Section 23 - 23.4.3.4](#)。

23.4.4.3.4 条件标志

这些指令不改变标志。

23.4.4.3.5 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
```

23.4.4.4 LDR, PC 相对

从存储器中加载寄存器（文字数据）。

23.4.4.4.1 语法

```
LDR Rt, label
```

其中：

Rt 加载的寄存器

Label 是 PC 相对表达式，请看 [Section 23 - 23.4.3.5](#)。

23.4.4.4.2 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。

23.4.4.4.3 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。

23.4.4.4.4 条件标志

这些指令不改变标志。

23.4.4.4.5 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
```

23.4.4.5 LDM 和 STM

加载和存储多个寄存器。

23.4.4.5.1 语法

LDM $Rn\{! \}$, reglist

STM $Rn!$, reglist

其中:

Rn 是寄存器, 存储器地址基于此寄存器。

! 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。它包含着寄存器范围。若它包含着多于一个的寄存器或寄存器范围, 必须要将其用逗号隔开, 请看 [Section 23 - 23.4.4.5.5](#)。

对于 LDM, LDMIA, 它们和 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堆栈中移出。

对于 STM, STMIA, 它们和 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堆栈中。

23.4.4.5.2 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。

23.4.4.5.3 限制

在这些指令中:

- reglist 和 Rn 限制为 R0-R7
- 必须要使用写回后缀, 除非指令是 LDM 指令, 在 LDM 里, reglist 也含有 Rn , 在这种情况下, 要谨记不能用写回后缀。
- Rn 所指定的寄存器的值必须是字对齐的。更多信息请看 [Section 23 - 23.4.3.4](#)。
- 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。

23.4.4.5.4 条件标志

这些指令不改变标志。

23.4.4.5.5 示例

LDM $R0, \{R0, R3, R4\}$; LDMIA 相近于 LDM

STMIA $R1!, \{R2-R4, R6\}$

23.4.4.5.6 错误的示例

```
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
```

23.4.4.6 PUSH 和 POP

将寄存器压入满递减堆栈和将满递减堆栈中的内容移入寄存器。

23.4.4.6.1 语法

`PUSH reglist`

`POP reglist`

其中:

Reglist 是非空的寄存器列表, 用大括号括着, 它包含着寄存器范围。若它包含着多于一个的寄存器或寄存器范围, 必须要将其用逗号隔开。

23.4.4.6.2 操作

PUSH 将寄存器存放到堆栈中, 最低编号的寄存器使用低存储器地址, 最高编号的寄存器使用高存储器地址。

POP 将堆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堆栈操作。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 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。

23.4.4.6.3 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

23.4.4.6.4 条件标志

这些指令不改变标志。

23.4.4.6.5 示例

```
PUSH    {R0, R4-R7}      ; 将 R0, R4, R5, R6, R7 压入堆栈
PUSH    {R2, LR}         ; 将 R2 和链接寄存器压入堆栈
POP      {R0, R6, PC}     ; 令 R0, R6 和 PC 出栈, 然后跳转到新的 PC 值
```

23. 4. 5 通用数据处理指令

[Table 336](#) 显示了数据处理指令：

Table 336. 数据处理指令

助记符	简述	参考
ADCS	进位加法	Section 23–23.4.5.1
ADD{S}	加法	Section 23–23.4.5.1
ANDS	逻辑与	Section 23–23.4.5.2
ASRS	算术右移	Section 23–23.4.5.3
BICS	位清零	Section 23–23.4.5.2
CMN	比较负值	Section 23–23.4.5.4
CMP	比较	Section 23–23.4.5.4
EORS	异或	Section 23–23.4.5.2
LSLS	逻辑左移	Section 23–23.4.5.3
LSRS	逻辑右移	Section 23–23.4.5.3
MOV{S}	传输	Section 23–23.4.5.5
MULS	乘法	Section 23–23.4.5.6
MVNS	取反传输	Section 23–23.4.5.5
ORRS	逻辑或	Section 23–23.4.5.2
REV	反转字里面的字节顺序	Section 23–23.4.5.7
REV16	反转每半字里面的字节顺序	Section 23–23.4.5.7
REVSH	反转低半字中的字节顺序，并进行符号扩展	Section 23–23.4.5.7
RORS	循环右移	Section 23–23.4.5.3
RSBS	反向减法	Section 23–23.4.5.1
SBCS	带进位减法	Section 23–23.4.5.1
SUBS	减法	Section 23–23.4.5.1
SXTB	符号扩展字节	Section 23–23.4.5.8
SXTH	符号扩展半字	Section 23–23.4.5.8
UXTB	零扩展字节	Section 23–23.4.5.8
UXTH	零扩展半字	Section 23–23.4.5.8
TST	测试	Section 23–23.4.5.9

23. 4. 5. 1 ADC, ADD, RSB, SBC, 和 SUB

进位加法、加法、反向减法、进位减法、减法。

23. 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,
```

$\langle Rm \mid \#imm \rangle$

- 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。

23.4.5.1.2 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, 则将结果另行加 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 的值加上 Rm 的值, 或加上 imm 指定的立即数, 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 Rn 的值减去 Rm 的值, 如果进位标志置位, 则再减去一个 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。

如何使用 ADC 和 SBC 来综合处理多字算术, 请看 [Section 23.4.5.1.4](#)。

还可以参考 [Section 23 – 23.4.4.1](#)。

23.4.5.1.3 限制

[Table 337](#) 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。

Table 337. 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 Rd 和 Rm 必须不能同时指定 PC
	R0-R7	SP or 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	-	-

Table 337. ADC, ADD, RSBS, SBC 和 SUB 操作数限制

指令	Rd	Rn	Rm	imm	限制
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	-	-

23. 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    ; 加上最低位的字
ADCS    R1, R1, R3    ; 加上最高位的字，带进位
```

多字的值无需使用连续的寄存器。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    ; 减去最低位字
SBCS    R5, R5, R2    ; 减去中间的字，带进位
SBCS    R6, R6, R3    ; 减去最高位字，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。

23. 4. 5. 2 AND, ORR, EOR, 和 BIC

逻辑 AND、OR、异或和位清除 。

23. 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

23.4.5.2.2 操作

AND、EOR 和 ORR 对 R_n 和 R_m 的值按位执行 AND、异或、或操作。

BIC 指令对 R_n 上的位，与 R_m 上的相应位执行逻辑非操作后，执行 AND 操作。

条件代码标志会根据操作的结果被更新，请看 [Section 23.4.3.6.1](#)。

23.4.5.2.3 限制

在这些指令中， R_d 、 R_n 和 R_m 必须指定 R0-R7。

23.4.5.2.4 条件标志

这些指令会：

- 根据结果值来更新 N 和 Z 标志
- 不会影响 C 或 V 标志

23.4.5.2.5 示例

```
ANDS R2, R2, R1
```

```
ORRS R2, R2, R5
```

```
ANDS R5, R5, R8
```

```
EORS R7, R7, R6
```

```
BICS R0, R0, R1
```

23.4.5.3 ASR, LSL, LSR, 和 ROR

算术右移，逻辑左移，逻辑右移，循环右移。

23.4.5.3.1 语法

```
ASRS {Rd,}  $R_m$ ,  $R_s$ 
```

```
ASRS {Rd,}  $R_m$ , # $imm$ 
```

```
LSLS {Rd,}  $R_m$ ,  $R_s$ 
```

```
LSLS {Rd,}  $R_m$ , # $imm$ 
```

```
LSRS {Rd,}  $R_m$ ,  $R_s$ 
```

```
LSRS {Rd,}  $R_m$ , # $imm$ 
```

```
RORS {Rd,}  $R_m$ ,  $R_s$ 
```

其中：

R_d 是目的寄存器。如果 R_d 被省略，则假定它的值与 R_m 相同

R_m 是保存要移位的值的寄存器

R_s 是保存着移位长度（该长度要应用到 R_m 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 1 到 32

LSL — 移位长度 0 到 31

LSR — 移位长度 1 到 32

注意： MOVs Rd, Rm 是 LSLS Rd, Rm, #0 的别名。

23.4.5.3.2 操作

ASR、LSL、LSR 和 ROR 对立即数 imm 所指定的长度而锁定的 Rm 寄存器的位或者 Rs 所指定的寄存器的最低位字节值执行算术左移、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，请看 [Section 23 - 23.4.3.3](#)。

23.4.5.3.3 限制

在这些指令中，Rd、Rm 和 Rs 必须只可以指定 R0-R7。对于非立即数指令，Rd 和 Rm 必须指定相同的寄存器。

23.4.5.3.4 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。当移位长度为 0 时例外，见 [Section 23 - 23.4.3.3](#)。V 标志不变。

23.4.5.3.5 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

23.4.5.4 CMP 和 CMN

比较和比较负值。

23.4.5.4.1 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

23.4.5.4.2 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。

23.4.5.4.3 限制

对于：

- CMP 指令
指令 Rn、Rm 必须只能指定 R0-R7。
- CMN 指令：
 - Rn 和 Rm 可以指定 R0-R14
 - 立即数的范围为 0-255

23.4.5.4.4 条件标志

这些指令根据结果值来更新 N、Z、C 和 V 标志。

23.4.5.4.5 示例

CMP R2, R9

CMN R0, R2

23.4.5.5 MOV 和 MVN

传输和取反传输。

23.4.5.5.1 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， 请看 [Section 23 - 23.4.3.6](#)。

Rd 是目的寄存器

Rm 是寄存器

Imm 可以是 0-255 范围内的任何一个值

23.4.5.5.2 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。

23.4.5.5.3 限制

在这些指令中，Rd 和 Rm 必须指定 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 来所生成的地址上执行跳转操作。T- 位保持不变。

注意： 尽管可以将 MOV 用作跳转指令，但是为了软件的可移植性，AMR 强烈推荐使用 BX 或 BLX 指令来执行跳转操作。

23.4.5.5.4 条件标志

如果 S 被指定，则这些指令会：

- 根据结果值更新 N 和 Z 标志
- 不会影响 C 或 V 标志

23.4.5.5.5 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 ; 将堆栈指针的值写入 R8

MVNS R2, R0 ; 将 R0 取反写入 R2 并更新标志

23.4.5.6 MULS

使用 32 位操作数的乘法，产生 32 位的结果值。

23.4.5.6.1 语法

MULS Rd, Rn, Rm

其中：

Rd 是目的寄存器

Rn、Rm 是保存进行乘法操作值的寄存器

23.4.5.6.2 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 32 位存放在 Rd 中。条件代码标志会按照操作的结果值而被更新， 请看 [Section 23 – 23.4.3.6](#)。

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。

23.4.5.6.3 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

23.4.5.6.4 条件标志

该指令会：

- 根据结果值来更新 N 和 Z 标志

- 不会影响 C 或 V 标志

23.4.5.6.5 示例

MULS R0, R2, R0 ; 乘法操作, 标志被更新, $R0 = R0 \times R2$

23.4.5.7 REV, REV16, 和 REVSH

反转字节。

23.4.5.7.1 语法

REV Rd, *Rn*

REV16 Rd, *Rn*

REVSH Rd, *Rn*

其中:

Rd 是目的寄存器

Rn 是源寄存器

23.4.5.7.2 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

23.4.5.7.3 限制

在这些指令中, Rd 和 Rn 必须只可以指定 R0-R7。

23.4.5.7.4 条件标志

这些指令不改变标志。

23.4.5.7.5 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

23.4.5.8 SXT 和 UXT

符号扩展和 0 扩展。

23.4.5.8.1 语法

SXTB Rd, *Rm*

SXTH Rd, *Rm*

UXTB Rd, *Rm*

UXTH Rd, *Rm*

其中:

Rd 是目的寄存器

Rm 是寄存器, 其保存的值会被扩展

23.4.5.8.2 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 [15:0] 并将值进行符号扩展到 32 位
- UXTH 提取 [15:0] 并将值用 0 扩展到 32 位

23.4.5.8.3 限制

在这些指令中, *Rd* 和 *Rm* 必须只可以指定 R0-R7。

23.4.5.8.4 条件标志

这些指令不改变标志。

23.4.5.8.5 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

23.4.5.9 TST

测试位。

23.4.5.9.1 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

23.4.5.9.2 操作

该指令将一个寄存器的值与另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，其它所有位被清除为 0。

23.4.5.9.3 限制

在这些指令中，Rn 和 Rm 必须只能指定 R0-R7。

23.4.5.9.4 条件标志

这些指令：

- 会根据结果来更新 N 和 Z 标志
- 不会影响 C 或 V 标志

23.4.5.9.5 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

23.4.6 跳转和控制指令

[Table 338](#) 所示位跳转和控制指令：

Table 338. 跳转和控制指令

助记符	简述	参考
B{cc}	跳转 { 有条件 }	Section 23–23.4.6.1
BL	带链接的跳转	Section 23–23.4.6.1
BLX	带链接的间接跳转	Section 23–23.4.6.1
BX	间接跳转	Section 23–23.4.6.1

23.4.6.1 B, BL, BX, 和 BLX

跳转指令。

23.4.6.1.1 语法

B{cond} label

BL label

BX Rm

BLX Rm

其中：

cond 是可选的条件代码，请看 [Section 23 – 23.4.3.6](#).

label 是 PC 相对表达式， 请看 [Section 23 – 23.4.3.5](#).

Rm 是提供跳转地址的寄存器

23.4.6.1.2 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。

[Table 339](#) 所示为适用于各种跳转指令的跳转范围。

Table 339. 跳转范围

指令	跳转范围
B label	–2 KB 到 +2 KB
Bcond label	–256 字节 到 +254 字节
BL label	–16 MB 到 +16 MB
BX Rm	寄存器中的任何值
BLX Rm	寄存器中的任何值

23. 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

注意： Bcond 是在 Cortex-M0 处理器上唯一的条件指令。

23. 4. 6. 1. 4 条件标志

这些指令不改变标志。

23. 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，如果最后的标志被设置，指令设置 Z 标志，否则不执行跳转
```

23. 4. 7 杂项指令

[Table 340](#) 所示为余下的 Cortex-M0 指令：

Table 340. 综合指令

助记符	简述	参考
BKPT	断点	Section 23–23.4.7.1
CPSID	更改处理器状态，禁止中断	Section 23–23.4.7.2
CPSIE	更改处理器状态，允许中断	Section 23–23.4.7.2
DMB	数据内存屏障	Section 23–23.4.7.3
DSB	数据同步屏障	Section 23–23.4.7.4
ISB	指令同步屏障	Section 23–23.4.7.5
MRS	从特殊寄存器传输到寄存器	Section 23–23.4.7.6
MSR	从寄存器传输到特殊寄存器	Section 23–23.4.7.7
NOP	空操作	Section 23–23.4.7.8
SEV	发送事件	Section 23–23.4.7.9
SVC	超级用户调用	Section 23–23.4.7.10
WFE	等待事件	Section 23–23.4.7.11
WFI	等待中断	Section 23–23.4.7.12

23. 4. 7. 1 BKPT

断点。

23. 4. 7. 1. 1 语法

BKPT #*imm*

其中：

imm 是 0-255 范围内的整数。

23. 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查询系统状态。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它信息。

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。更多信息请看 [Section 23 - 23.3.4.1](#)。

23.4.7.1.3 限制

没有限制。

23.4.7.1.4 条件标志

该指令不改变标志。

23.4.7.1.5 示例

BKPT #0 ; 立即数值设为 0x0 的断点

23.4.7.2 CPS

更改处理器状态。

23.4.7.2.1 语法

CPSID i

CPSIE i

23.4.7.2.2 操作

CPS 更改 PRIMASK 特殊寄存器值。通过设置 PRIMASK，CPSID 可令中断被关闭。而通过清除 PRIMASK，CPSIE 则可允许中断。关于这些寄存器的详细描述，更多信息请看 [Section 23 - 23.3.1.3.6](#)。

23.4.7.2.3 限制

没有限制。

23.4.7.2.4 条件标志

该指令不改变标志。

23.4.7.2.5 示例

CPSID i ; 关闭所有的中断，NMI 除外（设置 PRIMASK）

CPSIE i ; 使能中断（清除 PRIMASK）

23.4.7.3 DMB

数据内存屏障。

23.4.7.3.1 语法

DMB

23.4.7.3.2 操作

DMB 用作数据内存屏障。它可确保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。它不影响其他指令（不访问内存的指令）在处理器上的执行顺序。

23.4.7.3.3 限制

没有限制。

23.4.7.3.4 条件标志

该指令不改变标志。

23.4.7.3.5 示例

DMB ; 数据内存屏障

23.4.7.4 DSB

数据同步屏障。

23.4.7.4.1 语法

DSB

23.4.7.4.2 操作

DSB 用作特殊数据同步内存屏障，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。

23.4.7.4.3 限制

没有限制。

23.4.7.4.4 条件标志

该指令不改变标志。

23.4.7.4.5 示例

DSB ; 数据同步屏障

23.4.7.5 ISB

指令同步屏障。

23.4.7.5.1 语法

ISB

23.4.7.5.2 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。

23.4.7.5.3 限制

没有限制。

23.4.7.5.4 条件标志

该指令不改变标志。

23.4.7.5.5 示例

ISB ; 指令同步屏障

23.4.7.6 MRS

将特殊寄存器的内容移动到通用寄存器中。

23.4.7.6.1 语法

MRS *Rd*, *spec_reg*

其中：

Rd 是通用目的寄存器。

spec_reg 是其中一个特殊寄存器：APSR、IPSR、EPSR、IEPSR、IAPSR、EAPSR、PSR、MSP、PSP、PRIMASK 或 CONTROL

23.4.7.6.2 操作

MRS 将特殊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。

请看 [Section 23 - 23.4.7.7](#)。

23.4.7.6.3 限制

在该指令中，*Rd* 必须不能是 SP 或 PC。

23.4.7.6.4 条件标志

该指令不改变标志。

23.4.7.6.5 示例

MRS R0, PRIMASK ; 读取 PRIMASK 值并将其写入 R0

23.4.7.7 MSR

将通用寄存器的内容传移到指定的特别寄存器中

23.4.7.7.1 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。

23.4.7.7.2 操作

MSR 使用 Rn 所指定的寄存器的值来更新其中一个特殊寄存器。

请看 [Section 23 - 23.4.7.6](#)。

23.4.7.7.3 限制

在该指令里，Rn 必须不能为 SP 和 PC。

23.4.7.7.4 条件标志

该指令明确地根据 Rn 中的值来更新标志。

23.4.7.7.5 示例

MSR CONTROL, R1 ; 读取 R1 的值，并将其写入 CONTROL 寄存器

23.4.7.8 NOP

空操作。

23.4.7.8.1 语法

NOP

23.4.7.8.2 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。

23.4.7.8.3 限制

没有限制。

23.4.7.8.4 条件标志

该指令不改变标志。

23.4.7.8.5 示例

NOP ; 空操作

23.4.7.9 SEV

发送事件。

23.4.7.9.1 语法

SEV

23.4.7.9.2 操作

SEV 将带有信号的事件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。请看 [Section 23 - 23.3.5](#)。

也可以参考 [Section 23 - 23.4.7.11](#)。

23.4.7.9.3 限制

没有限制。

23.4.7.9.4 条件标志

该指令不改变标志。

23.4.7.9.5 示例

SEV ; 发送事件

23.4.7.10 SVC

超级用户调用。

23.4.7.10.1 语法

SVC #*imm*

其中:

Imm 是 0-255 范围内的整数

23.4.7.10.2 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。

23.4.7.10.3 限制

没有限制。

23.4.7.10.4 条件标志

该指令不改变标志。

23.4.7.10.5 示例

SVC #0x32 ; 超级用户调用 (SVC 处理程序使用堆栈的 PC 来锁定立即数的位置, 然后将其提取出来)。

23.4.7.11 WFE

等待事件。

注意： 该指令不会再在 LPC111x/LPC11Cxx 上面执行。

23.4.7.11.1 语法

WFE

23.4.7.11.2 操作

如果事件寄存器为 0，则 WFE 挂起执行，直至发生以下事件之一：

- 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，如果调试允许的话
- 外设或多处理器系统里另一个处理器通过使用 SEV 指令来发出信号事件

如果事件寄存器为 1，则 WFE 将其清除为 0 并立即完成操作。

更多信息请看 [Section 23 - 23.3.5](#)。

注意：WFE 的目的只是用于省电。当写软件时，假定 WFE 作为 NOP 运行。

23.4.7.11.3 限制

没有限制。

23.4.7.11.4 条件标志

该指令不改变标志。

23.4.7.11.5 示例

WFE；等待事件

23.4.7.12 WFI

等待中断。

23.4.7.12.1 语法

WFI

23.4.7.12.2 操作

WFI

挂起执行，直至发生以下事件之一：

- 一个异常
- 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，无论调试是否被允许

注意：WFI 的目的只是用于省电。当写软件时，假定 WFI 作为 NOP 运行。

23.4.7.12.3 限制

没有限制。

23.4.7.12.4 条件标志

该指令不改变标志。

23.4.7.12.5 示例

WFI；等待中断

23.5 外设

23.5.1 关于 ARM Cortex-M0

专用外设总线（PPB）的地址映射为：

Table 341. 核心外设寄存器区

地址	核心外设	描述
0xE000E008-0xE000E00F	系统控制块	Table 23-350
0xE000E010-0xE000E01F	系统定时器	Table 23-359
0xE000E100-0xE000E4EF	内嵌向量中断控制器	Table 23-342
0xE000ED00-0xE000ED3F	系统控制块	Table 23-350
0xE000EF00-0xE000EF03	内嵌向量中断控制器	Table 23-342

在寄存器描述中，寄存器的**类型**有以下几种：

- RW — 读和写
- RO — 只读
- WO — 只写

23.5.2 内嵌向量中断控制器

本节描述**内嵌向量中断控制器**（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
 - 中断尾链
 - 一个**外部不可屏蔽中断**（NMI）。LPC111x/LPC11Cxx 没有 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：

Table 342. NVIC 寄存器小结

地址	名称	类型	复位值	描述
0xE000E100	ISER	RW	0x00000000	Section 23-23.5.2.2
0xE000E180	ICER	RW	0x00000000	Section 23-23.5.2.3

Table 342. NVIC 寄存器小结

地址	名称	类型	复位值	描述
0xE000E200	ISPR	RW	0x00000000	Section 23–23.5.2.4
0xE000E280	ICPR	RW	0x00000000	Section 23–23.5.2.5
0xE000E400-0xE000E41C	IPR0-7	RW	0x00000000	Section 23–23.5.2.6

23. 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 系列中进行软件移植。

当利用 CMSIS 来访问 NVIC 寄存器时要用到以下函数：

Table 343. CMSIS 访问 NVIC 的函数

CMSIS 函数	描述
void NVIC_EnableIRQ(IRQn_Type IRQn) ^[1]	允许中断和异常
void NVIC_DisableIRQ(IRQn_Type IRQn) ^[1]	禁止中断和异常
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 0
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 0 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，更多信息请看 [Table 329](#)

23. 5. 2. 2 中断设置允许寄存器

ISER 允许中断，并显示哪些中断被允许。有关寄存器属性请见 [Table 342](#) 。

该寄存器的位分配如下：

Table 344. ISER 位分配

位域	名称	功能
[31:0]	SETENA	中断设置 - 允许位 写： 0 = 无影响 1 = 使能中断 读： 0 = 中断被禁止 1 = 中断被允许

如果一个挂起中断被允许，NVIC 就根据它的优先级来激活该中断。如果一个中断未被允许，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。

23. 5. 2. 3 中断清除允许寄存器

ICER 禁止中断，并显示哪些中断被允许。有关寄存器属性请见 in [Table 23 - 342](#) 。

该寄存器的位分配如下：

Table 345. ICER 位分配

位域	名称	功能
[31:0]	CLRENA	中断清除 - 允许位 写： 0 = 无影响 1 = 禁止中断 读： 0 = 中断被禁止 1 = 中断被允许

23. 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。有关寄存器属性请看 [Table 23 - 342](#) 。

该寄存器的位分配如下：

Table 346. ISPR 位分配

位域	名称	功能
[31:0]	SETPEND	中断设置 - 挂起位 写： 0 = 无影响 1 = 中断状态变为挂起 读： 0 = 中断没有挂起 1 = 中断正在挂起

注意：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
- 被禁止的中断会将中断的状态设置成挂起

23. 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。有关寄存器的属性请看 [Table 23 - 342](#) 。

该寄存器的位分配如下：

Table 347. ICPR 位分配

位域	名称	功能
[31:0]	CLRPEND	中断清除 - 挂起位 写: 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

注意：向 ICPR 位写 1 不影响相应中断的有效状态。

23.5.2.6 中断优先级寄存器

IPR0–IPR7 寄存器为每个中断提供了一个两位的优先级域。这些寄存器只能字访问。有关它们的属性请看 [Table 23-342](#)。
每个寄存器有 4 个优先级域，如下所示：

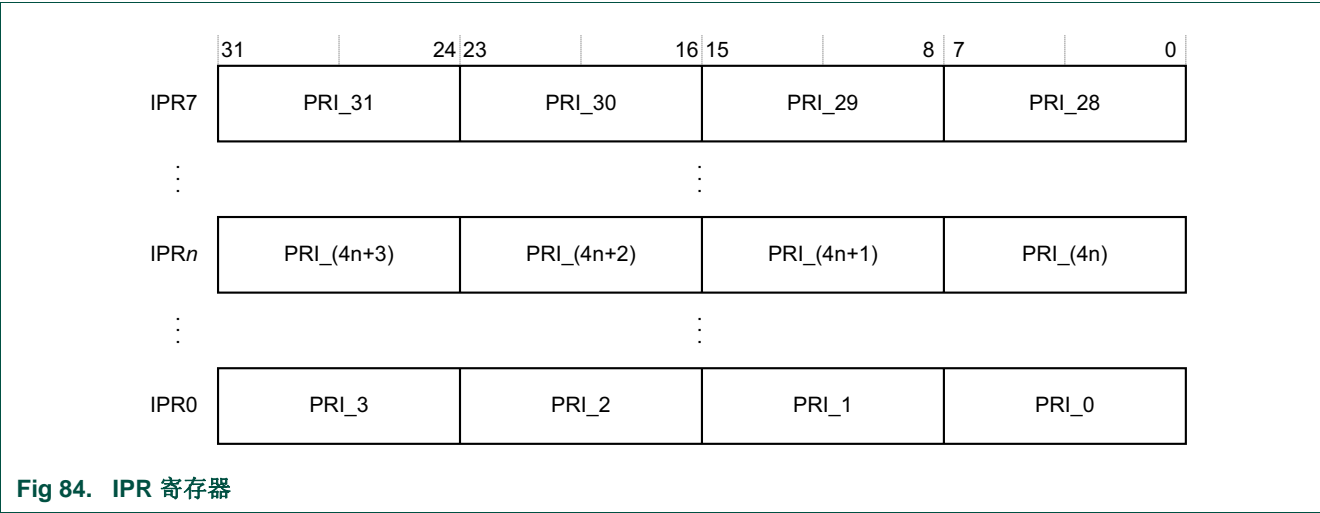


Fig 84. IPR 寄存器

Table 348. IPR 位分配

位域	名称	功能
[31:24]	Priority, byte offset 3	每个优先级域保存一个优先级值（0~3）。值越小，对应中断的优先级越高。处理器只使用每个域的 bit[7:6]，bit[5:0] 读出的为 0，写操作被忽略
[23:16]	Priority, byte offset 2	
[15:8]	Priority, byte offset 1	
[7:0]	Priority, byte offset 0	

有关中断优先级数组（提供了中断优先级的软件视角）访问的更多信息请参考 [Section 23-23.5.2.1](#)。

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 $N = M/4$ 得出
- 这个寄存器中所需优先级域的字节偏移量是 $M \text{ MOD } 4$ （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]

23.5.2.7 电平有效的中断和脉冲中断

处理器支持电平中断和脉冲中断。脉冲中断也被描述成边沿触发的中断。

电平中断一直要保持电平有效，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步采样到中断信号。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 ISP 时，它自动消除中断的挂起状态，见 [Section 23.5.2.7.1](#)。对于电平中断，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。

23.5.2.7.1 中断的硬件和软件控制

Cortex-M0 锁存所有的中断。外设中断会由于以下原因之一而变为挂起：

- 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，请见 [Section 23 - 23.5.2.4](#)。

挂起的中断一直保持挂起，直到出现以下情况之一：

- 处理器进入中断 ISR，这就使中断的状态从挂起变为有效。而且：
 - 对于电平中断，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。
- 利用软件向相应的中断清除 - 挂起寄存器位写入值。对于电平中断，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）

23.5.2.8 NVIC 使用提示和技巧

保证软件正确使用对齐的寄存器访问。处理器不支持不对齐的 NVIC 寄存器访问。

中断即使被禁止也可以进入挂起状态。禁止一个中断只阻止处理器处理中断。

23.5.2.8.1 NVIC 编程提示

软件使用 CPSIE i 和指令来允许和禁止中断。CMSIS 为这些指令提供以下内在函数：

```
void __disable_irq(void) // 禁止中断
```

void __enable_irq(void) // 允许中断

另外，CMSIS 提供了许多 NVIC 控制函数，包括：

Table 349. CMSIS 的 NVIC 控制函数

CMSIS 中断控制函数	描述
void NVIC_EnableIRQ(IRQn_t IRQn).	允许 IRQn
void NVIC_DisableIRQ(IRQn_t IRQn).	禁止 IRQn
uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn).	如果 IRQn 正在挂起，返回 True (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 19. 10。有关这些函数的更多信息，[Table 23 – 329](#) 。

23. 5. 3 系统控制块

系统控制块（SCB）提供了系统执行和控制信息，包括配置、控制和系统异常的报告。SCB 寄存器有：

Table 350. SCB 寄存器小结

地址	名称	类型	复位值	描述
0xE000ED00	CPUID	RO	0x410CC200	Section 23.5.3.2
0xE000ED04	ICSR	RW ^[1]	0x00000000	Section 23–23.5.3.3
0xE000ED0C	AIRCR	RW ^[1]	0xFA050000	Section 23–23.5.3.4
0xE000ED10	SCR	RW	0x00000000	Section 23–23.5.3.5

Table 350. SCB 寄存器小结

地址	名称	类型	复位值	描述
0xE000ED14	CCR	RO	0x00000204	Section 23–23.5.3.6
0xE000ED1C	SHPR2	RW	0x00000000	Section 23–23.5.3.7.1
0xE000ED20	SHPR3	RW	0x00000000	Section 23–23.5.3.7.2

[1] 更多信息请看寄存器描述

23.5.3.1 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，数组 SHP[1] 对应寄存器 SHPR2–SHPR3。

23.5.3.2 CPUID 寄存器

CPUID 寄存器包含处理器的型号、版本和实现信息。有关它的属性请见寄存器小结。CPUID 的位分配如下：

Table 351. CPUID 寄存器位分配

位域	名称	功能
[31:24]	Implementer	实现代码：0x41 = ARM
[23:20]	Variant	更新编号，产品版本标识符 rnpn 中 r 的值：0x0 = 版本 0
[19:16]	Constant	定义处理器结构的常量；读取的结果是：0xC = ARMv6-M 结构
[15:4]	Partno	处理器的型号：0xC20 = Cortex-M0
[3:0]	Revision	修订编号，产品版本标识符 rnpn 中 p 的值：0x0 = Patch 0

23.5.3.3 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 [Table 23 – 350](#)。ICSR 的位分配如下：

Table 352. ICSR 位分配

位域	名称	类型	功能
[31]	NMIPENDSET ^[2]	RW	NMI 设置 - 挂起位 写: 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: 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: 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: 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: 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

Table 352. ICSR 位分配

位域	名称	类型	功能
[17:12]	VECTPENDING	RO	指示优先级最高的、正在挂起的并且允许的异常的异常编号： 0 = 没有正在挂起的异常 非零 = 优先级最高的、正在挂起的并且允许的异常的异常编号
[11:6]	-	-	保留
[5:0]	VECTACTIVE ^[1]	RO	包含有效的异常编号： 0 = Thread 模式 非零 = 当前有效异常的异常编号 ^[1] 注意： 这个值减去 16 得到 CMSIS IRQ 编号，该编号标识出对应在中断清除 - 允许、设置 - 允许、清除 - 挂起、设置 - 挂起以及优先级寄存器中的位，请看 Table 23–324

[1] 这个值与 IPSR 位 [5:0] 的值相同， 请看 [Table 23 - 324](#)

[2] NMI 不能在 LPC111x/LPC11Cxx 上实现

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

23.5.3.4 应用中断和复位控制寄存器

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。有关寄存器的属性请见 [Table 23 - 350](#) 和 [Table 23 - 353](#)。

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。

AIRCR 的位分配如下：

Table 353. AIRCR 位分配

位域	名称	类型	功能
[31:16]	Read: Reserved Write: 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

23. 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。有关寄存器的属性请见 [Table 23 - 350](#) 。SCR 的位分配如下：

Table 354. SCR 位分配

位域	名称	功能
[31:5]	-	保留
[4]	SEVONPEND	挂起时发送事件位： 0 = 只有允许的中断或事件才能够唤醒处理器。不接受被禁止的中断的唤醒。 1 = 允许的事件和包括被禁止的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，影响下一个 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： 0 = 睡眠 1 = 深度睡眠
[1]	SLEEPONEXIT	指示当从处理器模式返回到线程模式时 sleep-on-exit （退出时进入睡眠）： 0 = 处理器返回到线程模式时不进入睡眠 1 = 处理器从 ISR 返回到线程模式时进入睡眠或深度睡眠 将该位设为 1 允许一个中断驱动的应用程序避免返回到一个空的主应用程序
[0]	-	保留

23.5.3.6 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 [Table 23 - 350](#)。

CCR 的位分配如下：

Table 355. CCR 位分配

位域	名称	功能
[31:10]	-	保留
[9]	STKALIGN	该位读出总是为 0，指示进入异常时堆栈按 8 字节对齐。 进入异常时，处理器使用入栈的 PSR 的 bit[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

23.5.3.7 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。有关它们的属性请见 [Table 23 - 350](#)。

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，更多信息请看 [Table 23 - 329](#)。

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：

Table 356. 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	Section 23-23.5.3.7.1
PendSV	PRI_14	Section 23-23.5.3.7.2
SysTick	PRI_15	

每个 PRI_N 域 8 位宽，但处理器只使用每个域的 bit[7:6]；bit[5:0] 读出为 0，写操作被忽略。

23.5.3.7.1 系统处理程序优先级寄存器 2

该寄存器的位分配如下：

Table 357. SHPR2 寄存器位分配

位域	名称	功能
[31:24]	PRI_11	系统处理程序 11（SVCall）的优先级
[23:0]	-	保留

23. 5. 3. 7. 2 系统处理程序优先级寄存器 3

该寄存器的位分配如下：

Table 358. SHPR3 寄存器的位分配

位域	名称	功能
[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

23. 5. 3. 8 SCB 使用提示和技巧

保证软件使用对齐的 32 位字事务来访问所有的 SCB 寄存器。

23. 5. 4 系统定时器，SysTick

当系统定时器被允许时，定时器从当前值（SYST_CVR）开始递减计数到零，下一个时钟周期的边沿处再重新装载系统定时重载寄存器（SYST_RVR）的值，然后在后面的时钟周期下继续开始递减计数。当计数器跳变到零时，COUNTFLAG 状态位被设为 1。读操作将 COUNTFLAG 位清零。

注意： SYST_CVR 的值在复位时不可知。在使用该功能之前软件应该使该寄存器清零。这确保定时器启用时从 SYST_RVR 的值开始计数，而不是从一个任意值开始计数。

注意： 如果 SYST_RVR 的值为 0，定时器在重载后将保持为当前值 0。这个机制可用于禁止定时器的某些特性，而不必通过定时器允许位来实现禁用功能。

写 SYST_CVR 会将该寄存器和 COUNTFLAG 状态位都清零。写操作导致 SYST_RVR 的值在下一个定时周期被重载到 SYST_CVR，但不触发 SysTick 异常逻辑。读操作返回的是当前被访问寄存器的值。

注意： 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：

Table 359. 系统定是寄存器小结

地址	名称	类型	复位值	描述
0xE000E010	SYST_CSR	RW	0x00000000	Section 23.5.4.1
0xE000E014	SYST_RVR	RW	不可知	Section 23–23.5.4.2
0xE000E018	SYST_CVR	RW	不可知	Section 23–23.5.4.3
0xE000E01C	SYST_CALIB	RO	0x00000004 ^[1]	Section 23–23.5.4.4

[1] SysTick 的校准值

23.5.4.1 SysTick 控制和状态寄存器

SYST_CSR 允许 SysTick 特性。有关寄存器的属性请见寄存器小结，该寄存器的位分配如下：

Table 360. SYST_CSR 位分配

位域	名称	功能
[31:17]	-	保留
[16]	COUNTFLAG	如果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	允许 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	允许计数器： 0 = 计数器被禁止 1 = 计数器被允许

23.5.4.2 SysTick 重装值寄存器

SYST_RVR 设定了加载到 SYST_CVR 的起始值。有关寄存器的属性请见 [Table 23 - 359](#)。该寄存器的位分配为：

Table 361. SYST_RVR 位分配

位域	名称	功能
[31:24]	-	保留
[23:0]	RELOAD	当计数器被允许且计数值到达 0 时加载到 SYST_CVR 的值，请见 Section 23.5.4.2.1

23.5.4.2.1 计算 RELOAD 值

RELOAD 值可以是 0x00000001-0x00FFFFFF 范围内的任何值。用户可以将 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，RELOAD 就被设为 99。

23.5.4.3 SysTick 当前值寄存器

SYST_VCR 包含 SysTick 计数器的当前值。有关寄存器的属性请见 [Table 23 - 359](#)。该寄存器的位分配如下：

Table 362. SYST_CVR 位分配

位域	名称	功能
[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

23.5.4.4 SysTick 校准值寄存器

SYST_CALIB 寄存器指明了 SysTick 的校准特性。有关寄存器的属性请见 [Table 23 - 359](#) 。该寄存器的位分配如下：

Table 363. SYST_CALIB 寄存器位分配

位域	名称	功能
[31]	NOREF	该位读出为 1。该位指明不提供独立的基准时钟
[30]	SKEW	该位读出为 1。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	该位读出为 0。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。

23.5.4.5 SysTick 使用提示和技巧

利用中断控制器时钟来更新 SysTick 计数器。如果这个时钟信号由于进入低功耗模式而终止，SysTick 计数器就停止计数。

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 计数器的重装值和当前值，正确的 SysTick 计数器初始化序列如下：

- 第 1 步：设置重装值
- 第 2 步：清除当前值
- 第 3 步：设置控制和状态寄存器

23.6 Cortex-M0 指令汇总

Table 364. Cortex M0 指令汇总

操作	描述	汇编程序	周期
Move	8 位立即数	MOV _S Rd, #<imm>	1
	(R0-R7) 到 (R0-R7)	MOV _S Rd, Rm	1
	任意寄存器到任意寄存器	MOV Rd, Rm	1
	任意寄存器到 PC	MOV PC, Rm	3
Add	3 位立即数	ADD _S Rd, Rn, #<imm>	1

Table 364. Cortex M0 指令汇总

操作	描述	汇编程序	周期
Add	R0-R7	ADDs Rd, Rn, Rm	1
	任意寄存器到任意寄存器	ADD Rd, Rd, Rm	1
	任意寄存器到 PC	ADD PC, PC, Rm	3
	8 位立即数	ADDs Rd, Rd, #<imm>	1
	带进位的	ADCS Rd, Rd, Rm	1
	立即数到 SP	ADD SP, SP, #<imm>	1
	从 SP 形成地址	ADD Rd, SP, #<imm>	1
	从 PC 形成地址	ADR Rd, <label>	1
Subtract	(R0-R7) 和 (R0-R7)	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
	带借位	SBCS Rd, Rd, Rm	1
	从 SP 减去立即数	SUB SP, SP, #<imm>	1
	相反数	RSBS Rd, Rn, #0	1
Multiply	乘法	MULS Rd, Rm, Rd	1
Compare	比较	CMP Rn, Rm	1
	负值	CMN Rn, Rm	1
	立即数	CMP Rn, #<imm>	1
Logical	与	ANDS Rd, Rd, Rm	1
	异或	EORS Rd, Rd, Rm	1
	或	ORRS Rd, Rd, Rm	1
	位清零	BICS Rd, Rd, Rm	1
	取反传送	MVNS Rd, Rm	1
	与测试	TST Rn, Rm	1
Shift	立即数逻辑左移	LSLS Rd, Rm, #<shift>	1
	寄存器逻辑左移	LSLS Rd, Rd, Rs	1
	立即数逻辑右移	LSRS Rd, Rm, #<shift>	1
	寄存器逻辑右移	LSRS Rd, Rd, Rs	1
	算术右移	ASRS Rd, Rm, #<shift>	1
	寄存器算术右移	ASRS Rd, Rd, Rs	1
Rotate	寄存器循环右移	RORS Rd, Rd, Rs	1

Table 364. Cortex M0 指令汇总

操作	描述	汇编程序	周期
Load	字, 直接偏移量	LDR Rd, [Rn, #<imm>]	2
	半字, 直接偏移量	LDRH Rd, [Rn, #<imm>]	2
	字节, 直接偏移量	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
	有符号的半字, 寄存器偏移量	LDRSH Rd, [Rn, Rm]	2
	字节, 寄存器偏移量	LDRB Rd, [Rn, Rm]	2
	有符号的字节, 寄存器偏移量	LDRSB Rd, [Rn, Rm]	2
	PC 相对值	LDR Rd, <label>	2
	SP 相对值	LDR Rd, [SP, #<imm>]	2
	乘法, 不带基地址	LDM Rn!, {<loreglist>}	1 + N ^[1]
	乘法, 带基地址	LDM Rn, {<loreglist>}	1 + N ^[1]
Store	字, 直接偏移量	STR Rd, [Rn, #<imm>]	2
Store	半字, 直接偏移量	STRH Rd, [Rn, #<imm>]	2
	字节, 直接偏移量	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
	SP 相对值	STR Rd, [SP, #<imm>]	2
	乘法	STM Rn!, {<loreglist>}	1 + N ^[1]
Push	进栈	PUSH {<loreglist>}	1 + N ^[1]
	带链接寄存器的进栈	PUSH {<loreglist>, LR}	1 + N ^[1]
Pop	出栈	POP {<loreglist>}	1 + N ^[1]
	出栈和返回	POP {<loreglist>, PC}	4 + N ^[2]
Branch	有条件的	B<cc> <label>	1 or 3 ^[3]
	无条件的	B <label>	3
	带链接的	BL <label>	4
	带交换的	BX Rm	3
	带链接和交换	BLX Rm	3
Extend	有符号的半字到字	SXTH Rd, Rm	1
	有符号的字节到字	SXTB Rd, Rm	1
	无符号半字	UXTH Rd, Rm	1
	无符号字节	UXTB Rd, Rm	1
Reverse	字中的字节	REV Rd, Rm	1
	两个半字中的字节	REV16 Rd, Rm	1
	有符号的底端半字	REVSH Rd, Rm	1
State change	超级用户调用	SVC <imm>	-[4]
	禁止中断	CPSID i	1
	允许中断	CPSIE i	1
	读特殊寄存器	MRS Rd, <specreg>	4
	写特殊寄存器	MSR <specreg>, Rn	4

Table 364. Cortex M0 指令汇总

操作	描述	汇编程序	周期
Hint	发送事件	SEV	1
	等待事件	WFE	2 ^[5]
	等待中断	WFI	2 ^[5]
	放弃	YIELD ^[6]	1
	空操作	NOP	1
Barriers	同步指令	ISB	4
	数据存储	DMB	4
	数据同步	DSB	4

[1] N 为元素的个数

[2] N 是堆栈出栈列表元素的个数，包括 PC 的数量并假设加载或存储不会产生 HardFault 异常

[3] 如果采取就为 3，不采取就为 1

[4] 周期数取决于核和调试配置

[5] 不包括等待事件或中断花费的时间。

[6] 作为 NOP 执行

24.1 缩写

Table 365. 缩写

首字母缩略词	描述
ADC	Analog-to-Digital Converter
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
BOD	BrownOut Detection
GPIO	General Purpose Input/Output
PLL	Phase-Locked Loop
SPI	Serial Peripheral Interface
SSI	Serial Synchronous Interface
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter

24.2 参考文献

- [1] ARM DUI 0497A — Cortex-M0 Devices Generic User Guide
- [2] ARM DDI 0432C — Cortex-M0 Revision r0p0 Technical Reference Manual

24.3 法律信息

24.3.1 Definitions

Draft

The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

– 24.3.2 Disclaimers

Limited warranty and liability

Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including – without limitation – lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes

NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any

time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use

NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications

Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing

all necessary testing for the customer' s applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer' s third party customer(s). NXP does not accept any liability in this respect.

Export control

This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

– 24.3.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners. I²C-bus

logo is a trademark of NXP B.V.

24.4 表格

Table 1.	LPC111x/LPC11Cxx enhancements.	3			
Table 2.	Ordering information	5	Table 28.	CLKOUT clock source select register (CLKOUTCLKSEL, address 0x4004 80E0) bit description	26
Table 3.	Ordering options	6	Table 29.	CLKOUT clock source update enable register (CLKOUTUEN, address 0x4004 80E4) bit description	26
Table 4.	LPC111x memory configuration	10	Table 30.	CLKOUT clock divider registers (CLKOUTCLKDIV, address 0x4004 80E8) bit description	27
Table 5.	LPC11Cxx memory configuration	10	Table 31.	POR captured PIO status registers 0 (PIOPORCAP0, address 0x4004 8100) bit description	27
Table 6.	Pin summary.	12	Table 32.	POR captured PIO status registers 1 (PIOPORCAP1, address 0x4004 8104) bit description	27
Table 7.	Register overview: system control block (base address 0x4004 8000)	14	Table 33.	BOD control register (BODCTRL, address 0x4004 8150) bit description.	28
Table 8.	System memory remap register (SYSTEMREMAP, address 0x4004 8000) bit description	16	Table 34.	System tick timer calibration register (SYSTCKCAL, address 0x4004 8154) bit description	28
Table 9.	Peripheral reset control register (PRESETCTRL, address 0x4004 8004) bit description.	16	Table 35.	Start logic edge control register 0 (STARTAPRP0, address 0x4004 8200) bit description	29
Table 10.	System PLL control register (SYSPLLCTRL, address 0x4004 8008) bit description	17	Table 36.	Start logic signal enable register 0 (STARTERP0, address 0x4004 8204) bit description	29
Table 11.	System PLL status register (SYSPLLSTAT, address 0x4004 800C) bit description	17	Table 37.	Start logic reset register 0 (STARTRSRP0CLR, address 0x4004 8208) bit description	30
Table 12.	System oscillator control register (SYSOSCCTRL, address 0x4004 8020) bit description.	18	Table 38.	Start logic status register 0 (STARTSRP0, address 0x4004 820C) bit description	30
Table 13.	Watchdog oscillator control register (WDTOSCCTRL, address 0x4004 8024) bit description	18	Table 39.	Allowed values for PDSLEEPCFG register . . .	30
Table 14.	Internal resonant crystal control register (IRCCTRL, address 0x4004 8028) bit description	19	Table 40.	Deep-sleep configuration register (PDSLEEPCFG, address 0x4004 8230) bit description	31
Table 15.	System reset status register (SYSRSTSTAT, address 0x4004 8030) bit description.	20	Table 41.	Wake-up configuration register (PDWAKECFG, address 0x4004 8234) bit description	32
Table 16.	System PLL clock source select register (SYSPLLCLKSEL, address 0x4004 8040) bit description	20	Table 42.	Power-down configuration register (PDRUNCFG, address 0x4004 8238) bit description	33
Table 17.	System PLL clock source update enable register (SYSPLLCLKUEN, address 0x4004 8044) bit description	21	Table 43.	Device ID register (DEVICE_ID, address 0x4004 83F4) bit description	34
Table 18.	Main clock source select register (MAINCLKSEL, address 0x4004 8070) bit description.	21	Table 44.	PLL frequency parameters.	42
Table 19.	Main clock source update enable register (MAINCLKUEN, address 0x4004 8074) bit description	22	Table 45.	PLL configuration examples.	43
Table 20.	System AHB clock divider register (SYSAHBCLKDIV, address 0x4004 8078) bit description	22	Table 46.	Flash configuration register (FLASHCFG, address 0x4003 C010) bit description.	44
Table 21.	System AHB clock control register (SYSAHBCLKCTRL, address 0x4004 8080) bit description	22	Table 47.	Register overview: PMU (base address 0x4003 8000)	45
Table 22.	SPI0 clock divider register (SSP0CLKDIV, address 0x4004 8094) bit description.	24	Table 48.	Power control register (PCON, address 0x4003 8000) bit description	45
Table 23.	UART clock divider register (UARTCLKDIV, address 0x4004 8098) bit description.	24	Table 49.	General purpose registers 0 to 3 (GPREG0 - GPREG3, address 0x4003 8004 to 0x4003 8010) bit description	46
Table 24.	SPI1 clock divider register (SSP1CLKDIV, address 0x4004 809C) bit description	25	Table 50.	General purpose register 4 (GPREG4, address 0x4003 8014) bit description	46
Table 25.	WDT clock source select register (WDTCLKSEL, address 0x4004 80D0) bit description	25	Table 51.	set_pll routine	50
Table 26.	WDT clock source update enable register (WDTCLKUEN, address 0x4004 80D4) bit description	25	Table 52.	set_power routine	54
Table 27.	WDT clock divider register (WDTCLKDIV, address 0x4004 80D8) bit description	26	Table 53.	Connection of interrupt sources to the Vectored Interrupt Controller.	56

Table 54. Register overview: I/O configuration (base address 0x4004 4000)	61	Table 82. IOCON_PIO2_11 register (IOCON_PIO2_11, address 0x4004 4070) bit description	79
Table 55. I/O configuration registers ordered by port number	63	Table 83. IOCON_R_PIO0_11 register (IOCON_R_PIO0_11, address 0x4004 4074) bit description	79
Table 56. IOCON_PIO2_6 register (IOCON_PIO2_6, address 0x4004 4000) bit description	64	Table 84. IOCON_R_PIO1_0 register (IOCON_R_PIO1_0, address 0x4004 4078) bit description	80
Table 57. IOCON_PIO2_0 register (IOCON_PIO2_0, address 0x4004 4008) bit description	65	Table 85. IOCON_R_PIO1_1 register (IOCON_R_PIO1_1, address 0x4004 407C) bit description	81
Table 58. IOCON_RESET_PIO0_0 register (IOCON_RESET_PIO0_0, address 0x4004 400C) bit description.	65	Table 86. IOCON_R_PIO1_2 register (IOCON_R_PIO1_2, address 0x4004 4080) bit description	82
Table 59. IOCON_PIO0_1 register (IOCON_PIO0_1, address 0x4004 4010) bit description	66	Table 87. IOCON_PIO3_0 register (IOCON_PIO3_0, address 0x4004 4084) bit description	82
Table 60. IOCON_PIO1_8 register (IOCON_PIO1_8, address 0x4004 4014) bit description	66	Table 88. IOCON_PIO3_1 register (IOCON_PIO3_1, address 0x4004 4088) bit description	83
Table 61. IOCON_PIO0_2 register (IOCON_PIO0_2, address 0x4004 401C) bit description	67	Table 89. IOCON_PIO2_3 register (IOCON_PIO2_3, address 0x4004 408C) bit description	83
Table 62. IOCON_PIO2_7 register (IOCON_PIO2_7, address 0x4004 4020) bit description	68	Table 90. IOCON_SWDIO_PIO1_3 register (IOCON_SWDIO_PIO1_3, address 0x4004 4090) bit description	84
Table 63. IOCON_PIO2_8 register (IOCON_PIO2_8, address 0x4004 4024) bit description	68	Table 91. IOCON_PIO1_4 register (IOCON_PIO1_4, address 0x4004 4094) bit description	85
Table 64. IOCON_PIO2_1 register (IOCON_PIO2_1, address 0x4004 4028) bit description	69	Table 92. IOCON_PIO1_11 register (IOCON_PIO1_11, address 0x4004 4098) bit description	86
Table 65. IOCON_PIO0_3 register (IOCON_PIO0_3, address 0x4004 402C) bit description	69	Table 93. IOCON_PIO3_2 register (IOCON_PIO3_2, address 0x4004 409C) bit description	86
Table 66. IOCON_PIO0_4 register (IOCON_PIO0_4, address 0x4004 4030) bit description	70	Table 94. IOCON_PIO1_5 register (IOCON_PIO1_5, address 0x4004 40A0) bit description	87
Table 67. IOCON_PIO0_5 register (IOCON_PIO0_5, address 0x4004 4034) bit description	70	Table 95. IOCON_PIO1_6 register (IOCON_PIO1_6, address 0x4004 40A4) bit description	87
Table 68. IOCON_PIO1_9 register (IOCON_PIO1_9, address 0x4004 4038) bit description	71	Table 96. IOCON_PIO1_7 register (IOCON_PIO1_7, address 0x4004 40A8) bit description	88
Table 69. IOCON_PIO3_4 register (IOCON_PIO3_4, address 0x4004 403C) bit description	71	Table 97. IOCON_PIO3_3 register (IOCON_PIO3_3, address 0x4004 40AC) bit description	89
Table 70. IOCON_PIO2_4 register (IOCON_PIO2_4, address 0x4004 4040) bit description	72	Table 98. IOCON_SCK location register (IOCON_SCK_LOC, address 0x4004 40B0) bit description	89
Table 71. IOCON_PIO2_5 register (IOCON_PIO2_5, address 0x4004 4044) bit description	72	Table 99. IOCON_DSR location register (IOCON_DSR_LOC, address 0x4004 40B4) bit description	90
Table 72. IOCON_PIO3_5 register (IOCON_PIO3_5, address 0x4004 4048) bit description	73	Table 100. IOCON_DCD location register (IOCON_DCD_LOC, address 0x4004 40B8) bit description	90
Table 73. IOCON_PIO0_6 register (IOCON_PIO0_6, address 0x4004 404C) bit description	73	Table 101. IOCON_RI location register (IOCON_RI_LOC, address 0x4004 40BC) bit description	90
Table 74. IOCON_PIO0_7 register (IOCON_PIO0_7, address 0x4004 4050) bit description.	74	Table 102. LPC111x/LPC11Cxx pin configurations.	91
Table 75. IOCON_PIO2_9 register (IOCON_PIO2_9, address 0x4004 4054) bit description	75	Table 103. LPC1113/14 and LPC11C12/C14 pin description table (LQFP48 package)	96
Table 76. IOCON_PIO2_10 register (IOCON_PIO2_10, address 0x4004 4058) bit description	75	Table 104. LPC1114 pin description table (PLCC44 package)	100
Table 77. IOCON_PIO2_2 register (IOCON_PIO2_2, address 0x4004 405C) bit description	76	Table 105. LPC1111/12/13/14 pin description table (HVQFN33 package)	104
Table 78. IOCON_PIO0_8 register (IOCON_PIO0_8, address 0x4004 4060) bit description	76	Table 106. LPC11C24/C22 pin description table (LQFP48 package)	106
Table 79. IOCON_PIO0_9 register (IOCON_PIO0_9, address 0x4004 4064) bit description	77	Table 107. GPIO configuration	110
Table 80. IOCON_SWCLK_PIO0_10 register (IOCON_SWCLK_PIO0_10, address 0x4004 4068) bit description	77	Table 108. Register overview: GPIO (base address port 0: 0x5000 0000; port 1: 0x5001 0000, port 2: 0x5002 0000; port 3: 0x5003 0000)	111
Table 81. IOCON_PIO1_10 register (IOCON_PIO1_10, address 0x4004 406C) bit description	78		

Table 109. GPIOnDATA register (GPIO0DATA, address 0x5000 0000 to 0x5000 3FFC; GPIO1DATA, address 0x5001 0000 to 0x5001 3FFC; GPIO2DATA, address 0x5002 0000 to 0x5002 3FFC; GPIO3DATA, address 0x5003 0000 to 0x5003 3FFC) bit description	111	Table 130. Modem status interrupt generation	128
Table 110. GPIOnDIR register (GPIO0DIR, address 0x5000 8000 to GPIO3DIR, address 0x5003 8000) bit description	112	Table 131. UART Line Status Register (U0LSR - address 0x4000 8014, Read Only) bit description	129
Table 111. GPIOnIS register (GPIO0IS, address 0x5000 8004 to GPIO3IS, address 0x5003 8004) bit description	112	Table 132. UART Modem Status Register (U0MSR - address 0x4000 8018) bit description	131
Table 112. GPIOnIBE register (GPIO0IBE, address 0x5000 8008 to GPIO3IBE, address 0x5003 8008) bit description	112	Table 133. UART Scratch Pad Register (U0SCR - address 0x4000 801C) bit description	131
Table 113. GPIOnIEV register (GPIO0IEV, address 0x5000 800C to GPIO3IEV, address 0x5003 800C) bit description	113	Table 134. Auto baud Control Register (U0ACR - address 0x4000 8020) bit description	132
Table 114. GPIOnIE register (GPIO0IE, address 0x5000 8010 to GPIO3IE, address 0x5003 8010) bit description	113	Table 135. UART Fractional Divider Register (U0FDR - address 0x4000 8028) bit description	135
Table 115. GPIOnRIS register (GPIO0RIS, address 0x5000 8014 to GPIO3RIS, address 0x5003 8014) bit description	113	Table 136. Fractional Divider setting look-up table	138
Table 116. GPIOnMIS register (GPIO0MIS, address 0x5000 8018 to GPIO3MIS, address 0x5003 8018) bit description	114	Table 137. UART Transmit Enable Register (U0TER - address 0x4000 8030) bit description	139
Table 117. GPIOnIC register (GPIO0IC, address 0x5000 801C to GPIO3IC, address 0x5003 801C) bit description	114	Table 138. UART RS485 Control register (U0RS485CTRL - address 0x4000 804C) bit description	139
Table 118. UART pin description	118	Table 139. UART RS485 Address Match register (U0RS485ADRMATCH - address 0x4000 8050) bit description	140
Table 119. Register overview: UART (base address: 0x4000 8000)	118	Table 140. UART RS485 Delay value register (U0RS485DLY - address 0x4000 8054) bit description	140
Table 120. UART Receiver Buffer Register (U0RBR - address 0x4000 8000 when DLAB = 0, Read Only) bit description	120	Table 141. SPI pin descriptions	145
Table 121. UART Transmitter Holding Register (U0THR - address 0x4000 8000 when DLAB = 0, Write Only) bit description	120	Table 142. Register overview: SPI0 (base address 0x4004 0000)	146
Table 122. UART Divisor Latch LSB Register (U0DLL - address 0x4000 8000 when DLAB = 1) bit description	121	Table 143. Register overview: SPI1 (base address 0x4005 8000)	146
Table 123. UART Divisor Latch MSB Register (U0DLM - address 0x4000 8004 when DLAB = 1) bit description	121	Table 144. SPI/SSP Control Register 0 (SSP0CR0 - address 0x4004 0000, SSP1CR0 - address 0x4005 8000) bit description	147
Table 124. UART Interrupt Enable Register (U0IER - address 0x4000 8004 when DLAB = 0) bit description	121	Table 145. SPI/SSP Control Register 1 (SSP0CR1 - address 0x4004 0004, SSP1CR1 - address 0x4005 8004) bit description	148
Table 125. UART Interrupt Identification Register (U0IIR - address 0x4004 8008, Read Only) bit description	122	Table 146. SPI/SSP Data Register (SSP0DR - address 0x4004 0008, SSP1DR - address 0x4005 8008) bit description	148
Table 126. UART Interrupt Handling	123	Table 147. SPI/SSP Status Register (SSP0SR - address 0x4004 000C, SSP1SR - address 0x4005 800C) bit description	149
Table 127. UART FIFO Control Register (U0FCR - address 0x4000 8008, Write Only) bit description	125	Table 148. SPI/SSP Clock Prescale Register (SSP0CPSR - address 0x4004 0010, SSP1CPSR - address 0x4005 8010) bit description	149
Table 128. UART Line Control Register (U0LCR - address 0x4000 800C) bit description	125	Table 149. SPI/SSP Interrupt Mask Set/Clear register (SSP0IMSC - address 0x4004 0014, SSP1IMSC - address 0x4005 8014) bit description	150
Table 129. UART0 Modem Control Register (U0MCR - address 0x4000 8010) bit description	126	Table 150. SPI/SSP Raw Interrupt Status register (SSP0RIS - address 0x4004 0018, SSP1RIS - address 0x4005 8018) bit description	150
		Table 151. SPI/SSP Masked Interrupt Status register (SSP0MIS - address 0x4004 001C, SSP1MIS - address 0x4005 801C) bit description	151
		Table 152. SPI/SSP interrupt Clear Register (SSP0ICR - address 0x4004 0020, SSP1ICR - address 0x4005 8020) bit description	151
		Table 153. I ² C-bus pin description	161
		Table 154. Register overview: I ² C (base address 0x4000 0000)	161
		Table 155. I ² C Control Set register (I2C0CONSET - address 0x4000 0000) bit description	162

Table 156. I ² C Status register (I2C0STAT - 0x4000 0004) bit description	164	(CANBRPE, address 0x4005 0018) bit description	212
Table 157. I ² C Data register (I2C0DAT - 0x4000 0008) bit description	164	Table 188. Message interface registers	213
Table 158. I ² C Slave Address register 0 (I2C0ADR0 - 0x4000 000C) bit description	165	Table 189. Structure of a message object in the message RAM	213
Table 159. I ² C SCL HIGH Duty Cycle register (I2C0SCLH - address 0x4000 0010) bit description	165	Table 190. CAN message interface command request registers (CANIF1_CMDREQ, address 0x4005 0020 and CANIF2_CMDREQ, address 0x4005 0080) bit description	214
Table 160. I ² C SCL Low duty cycle register (I2C0SCLL - 0x4000 0014) bit description	165	Table 191. CAN message interface command mask registers (CANIF1_CMDMSK, address 0x4005 0024 and CANIF2_CMDMSK, address 0x4005 0084) bit description - write direction	214
Table 161. SCLL + SCLH values for selected I ² C clock values	165	Table 192. CAN message interface command mask registers (CANIF1_CMDMSK, address 0x4005 0024 and CANIF2_CMDMSK, address 0x4005 0084) bit description - read direction	215
Table 162. I ² C Control Clear register (I2C0CONCLR - 0x4000 0018) bit description	166	Table 193. CAN message interface command mask 1 registers (CANIF1_MSK1, address 0x4005 0028 and CANIF2_MASK1, address 0x4005 0088) bit description	217
Table 163. I ² C Monitor mode control register (I2C0MMCTRL - 0x4000 001C) bit description	167	Table 194. CAN message interface command mask 2 registers (CANIF1_MSK2, address 0x4005 002C and CANIF2_MASK2, address 0x4005 008C) bit description	217
Table 164. I ² C Slave Address registers (I2C0ADR[1, 2, 3] - 0x4000 00[20, 24, 28]) bit description	168	Table 195. CAN message interface command arbitration 1 registers (CANIF1_ARB1, address 0x4005 0030 and CANIF2_ARB1, address 0x4005 0090) bit description	217
Table 165. I ² C Data buffer register (I2C0DATA_BUFFER - 0x4000 002C) bit description	169	Table 196. CAN message interface command arbitration 2 registers (CANIF1_ARB2, address 0x4005 0034 and CANIF2_ARB2, address 0x4005 0094) bit description	218
Table 166. I ² C Mask registers (I2C0MASK[0, 1, 2, 3] - 0x4000 00[30, 34, 38, 3C]) bit description	169	Table 197. CAN message interface message control registers (CANIF1_MCTRL, address 0x4005 0038 and CANIF2_MCTRL, address 0x4005 0098) bit description	218
Table 167. I2C0CONSET and I2C1CONSET used to configure Master mode	170	Table 198. CAN message interface data A1 registers (CANIF1_DA1, address 0x4005 003C and CANIF2_DA1, address 0x4005 009C) bit description	220
Table 168. I2C0CONSET and I2C1CONSET used to configure Slave mode	171	Table 199. CAN message interface data A2 registers (CANIF1_DA2, address 0x4005 0040 and CANIF2_DA2, address 0x4005 00A0) bit description	220
Table 169. Abbreviations used to describe an I ² C operation	177	Table 200. CAN message interface data B1 registers (CANIF1_DB1, address 0x4005 0044 and CANIF2_DB1, address 0x4005 00A4) bit description	220
Table 170. I2C0CONSET used to initialize Master Transmitter mode	177	Table 201. CAN message interface data B2 registers (CANIF1_DB2, address 0x4005 0048 and CANIF2_DB2, address 0x4005 00A8) bit description	220
Table 171. Master Transmitter mode	179	Table 202. CAN transmission request 1 register (CANTXREQ1, address 0x4005 0100) bit description	221
Table 172. Master Receiver mode	182	Table 203. CAN transmission request 2 register	
Table 173. I2C0ADR and I2C1ADR usage in Slave Receiver mode	184		
Table 174. I2C0CONSET and I2C1CONSET used to initialize Slave Receiver mode	184		
Table 175. Slave Receiver mode	185		
Table 176. Slave Transmitter mode	189		
Table 177. Miscellaneous States	191		
Table 178. CAN pin description (LPC11C12/C14)	204		
Table 179. CAN pin description (LPC11C22/C24)	204		
Table 180. Register overview: CCAN (base address 0x4005 0000)	204		
Table 181. CAN control registers (CANCNTL, address 0x4005 0000) bit description	206		
Table 182. CAN status register (CANSTAT, address 0x4005 0004) bit description	208		
Table 183. CAN error counter (CANEC, address 0x4005 0008) bit description	209		
Table 184. CAN bit timing register (CANBT, address 0x4005 000C) bit description	210		
Table 185. CAN interrupt register (CANINT, address 0x4005 0010) bit description	211		
Table 186. CAN test register (CANTEST, address 0x4005 0014) bit description	211		
Table 187. CAN baud rate prescaler extension register			

(CANTXREQ2, address 0x4005 0104) bit description	221	Table 229. PWM Control Register (TMR16B0PWWC - address 0x4000 C074 and TMR16B1PWWC - address 0x4001 0074) bit description	262
Table 204. CAN new data 1 register (CANND1, address 0x4005 0120) bit description	222	Table 230. Counter/timer pin description	267
Table 205. CAN new data 2 register (CANND2, address 0x4005 0124) bit description	222	Table 231. Register overview: 32-bit counter/timer 0 CT32B0 (base address 0x4001 4000)	267
Table 206. CAN interrupt pending 1 register (CANIR1, address 0x4005 0140) bit description	222	Table 232. Register overview: 32-bit counter/timer 1 CT32B1 (base address 0x4001 8000)	268
Table 207. CAN interrupt pending 2 register (CANIR2, addresses 0x4005 0144) bit description	223	Table 233. Interrupt Register (TMR32B0IR - address 0x4001 4000 and TMR32B1IR - address 0x4001 8000) bit description	269
Table 208. CAN message valid 1 register (CANMSGV1, addresses 0x4005 0160) bit description	223	Table 234. Timer Control Register (TMR32B0TCR - address 0x4001 4004 and TMR32B1TCR - address 0x4001 8004) bit description	270
Table 209. CAN message valid 2 register (CANMSGV2, address 0x4005 0164) bit description	223	Table 235. Timer counter registers (TMR32B0TC, address 0x4001 4008 and TMR32B1TC 0x4001 8008) bit description	270
Table 210. CAN clock divider register (CANCLKDIV, address 0x4005 0180) bit description	224	Table 236. Prescale registers (TMR32B0PR, address 0x4001 400C and TMR32B1PR 0x4001 800C) bit description	270
Table 211. Initialization of a transmit object	232	Table 237. Prescale counter registers (TMR32B0PC, address 0x4001 4010 and TMR32B1PC 0x4001 8010) bit description	271
Table 212. Initialization of a receive object	233	Table 238. Match Control Register (TMR32B0MCR - address 0x4001 4014 and TMR32B1MCR - address 0x4001 8014) bit description	271
Table 213. Parameters of the C_CAN bit time	237	Table 239. Match registers (TMR32B0MR0 to 3, addresses 0x4001 4018 to 24 and TMR32B1MR0 to 3, addresses 0x4001 8018 to 24) bit description 272	
Table 214. Counter/timer pin description	253	Table 240. Capture Control Register (TMR32B0CCR - address 0x4001 4028 and TMR32B1CCR - address 0x4001 8028) bit description	272
Table 215. Register overview: 16-bit counter/timer 0 CT16B0 (base address 0x4000 C000)	254	Table 241. Capture registers (TMR32B0CR0, addresses 0x4001 402C and TMR32B1CR0, addresses 0x4001 802C) bit description	273
Table 216. Register overview: 16-bit counter/timer 1 CT16B1 (base address 0x4001 0000)	255	Table 242. External Match Register (TMR32B0EMR - address 0x4001 403C and TMR32B1EMR - address 0x4001 803C) bit description	274
Table 217. Interrupt Register (TMR16B0IR - address 0x4000 C000 and TMR16B1IR - address 0x4001 0000) bit description	256	Table 243. External match control	275
Table 218. Timer Control Register (TMR16B0TCR - address 0x4000 C004 and TMR16B1TCR - address 0x4001 0004) bit description	256	Table 244. Count Control Register (TMR32B0CTCR - address 0x4001 4070 and TMR32B1TCR - address 0x4001 8070) bit description	276
Table 219. Timer counter registers (TMR16B0TC, address 0x4000 C008 and TMR16B1TC 0x4001 0008) bit description	256	Table 245. PWM Control Register (TMR32B0PWWC - 0x4001 4074 and TMR32B1PWWC - 0x4001 8074) bit description	276
Table 220. Prescale registers (TMR16B0PR, address 0x4000 C00C and TMR16B1PR 0x4001 000C) bit description	257	Table 246. Register overview: Watchdog timer (base address 0x4000 4000)	283
Table 221. Prescale counter registers (TMR16B0PC, address 0x4001 C010 and TMR16B1PC 0x4001 0010) bit description	257	Table 247. Watchdog Mode register (WDMOD - 0x4000 4000) bit description	283
Table 222. Match Control Register (TMR16B0MCR - address 0x4000 C014 and TMR16B1MCR - address 0x4001 0014) bit description	257	Table 248. Watchdog operating modes selection	284
Table 223. Match registers (TMR16B0MR0 to 3, addresses 0x4000 C018 to 24 and TMR16B1MR0 to 3, addresses 0x4001 0018 to 24) bit description 259		Table 249. Watchdog Timer Constant register (WDTC - 0x4000 4004) bit description	285
Table 224. Capture Control Register (TMR16B0CCR - address 0x4000 C028 and TMR16B1CCR - address 0x4001 0028) bit description	259	Table 250. Watchdog Feed register (WDFEED - 0x4000 4008) bit description	285
Table 225. Capture registers (TMR16B0CR0, address 0x4000 C02C and TMR16B1CR0, address 0x4001 002C) bit description	259	Table 251. Watchdog Timer Value register (WDTV - 0x4000 400C) bit description	285
Table 226. External Match Register (TMR16B0EMR - address 0x4000 C03C and TMR16B1EMR - address 0x4001 003C) bit description	260	Table 252. Watchdog Timer Warning Interrupt register (WDWARNINT - 0x4000 4014) bit description 286	
Table 227. External match control	261		
Table 228. Count Control Register (TMR16B0CTCR - address 0x4000 C070 and TMR16B1CTCR - address 0x4001 0070) bit description	262		

Table 253. Watchdog Timer Window register (WDWINDOW - 0x4000 4018) bit description	286	Table 289. LPC111x and LPC11Cx part identification numbers	317
Table 254. Register overview: Watchdog timer (base address 0x4000 4000)	290	Table 290. UART ISP Read Boot Code version number command	317
Table 255. Watchdog Mode register (WDMOD - address 0x4000 4000) bit description	290	Table 291. UART ISP Compare command	318
Table 256. Watchdog operating modes selection	291	Table 292. UART ISP ReadUID command	318
Table 257. Watchdog Constant register (WDTC - address 0x4000 4004) bit description	291	Table 293. UART ISP Return Codes Summary	318
Table 258. Watchdog Feed register (WDFEED - address 0x4000 4008) bit description	292	Table 294. C_CAN ISP and UART ISP command summary	320
Table 259. Watchdog Timer Value register (WDTV - address 0x4000 000C) bit description	292	Table 295. C_CAN ISP object directory	320
Table 260. Register overview: SysTick timer (base address 0xE000 E000)	294	Table 296. C_CAN ISP SDO abort codes	323
Table 261. SysTick Timer Control and status register (SYST_CSR - 0xE000 E010) bit description	295	Table 297. IAP Command Summary	325
Table 262. System Timer Reload value register (SYST_RVR - 0xE000 E014) bit description	295	Table 298. IAP Prepare sector(s) for write operation command	326
Table 263. System Timer Current value register (SYST_CVR - 0xE000 E018) bit description	295	Table 299. IAP Copy RAM to flash command	326
Table 264. System Timer Calibration value register (SYST_CALIB - 0xE000 E01C) bit description	296	Table 300. IAP Erase Sector(s) command	327
Table 265. ADC pin description	297	Table 301. IAP Blank check sector(s) command	327
Table 266. Register overview: ADC (base address 0x4001 C000)	298	Table 302. IAP Read Part Identification command	327
Table 267. A/D Control Register (AD0CR - address 0x4001 C000) bit description	299	Table 303. IAP Read Boot Code version number command	328
Table 268. A/D Global Data Register (AD0GDR - address 0x4001 C004) bit description	300	Table 304. IAP Compare command	328
Table 269. A/D Status Register (AD0STAT - address 0x4001 C030) bit description	301	Table 305. IAP Reinvoke ISP	328
Table 270. A/D Interrupt Enable Register (AD0INTEN - address 0x4001 C00C) bit description	301	Table 306. IAP ReadUID command	329
Table 271. A/D Data Registers (AD0DR0 to AD0DR7 - addresses 0x4001 C010 to 0x4001 C02C) bit description	302	Table 307. IAP Status Codes Summary	329
Table 272. LPC111x/LPC11Cx flash configurations	303	Table 308. Memory mapping in debug mode	330
Table 273. Flash sector configuration	307	Table 309. Flash configuration register (FLASHCFG, address 0x4003 C010) bit description	330
Table 274. Code Read Protection options	308	Table 310. Register overview: FMC (base address 0x4003 C000)	331
Table 275. Code Read Protection hardware/software interaction	309	Table 311. Flash Module Signature Start register (FMSSTART - 0x4003 C020) bit description	332
Table 276. ISP commands allowed for different CRP levels	309	Table 312. Flash Module Signature Stop register (FMSSTOP - 0x4003 C024) bit description	332
Table 277. UART ISP command summary	311	Table 313. FMSW0 register bit description (FMSW0, address: 0x4003 C02C)	332
Table 278. UART ISP Unlock command	312	Table 314. FMSW1 register bit description (FMSW1, address: 0x4003 C030)	332
Table 279. UART ISP Set Baud Rate command	312	Table 315. FMSW2 register bit description (FMSW2, address: 0x4003 C034)	332
Table 280. UART ISP Echo command	312	Table 316. FMSW3 register bit description (FMSW3, address: 0x4003 40C8)	332
Table 281. UART ISP Write to RAM command	313	Table 317. Flash module Status register (FMSTAT - 0x4003 CFE0) bit description	333
Table 282. UART ISP Read Memory command	313	Table 318. Flash Module Status Clear register (FMSTATCLR - 0x0x4003 CFE8) bit description	333
Table 283. UART ISP Prepare sector(s) for write operation command	314	Table 319. Serial Wire Debug pin description	335
Table 284. UART ISP Copy command	315	Table 320. Summary of processor mode and stack use options	339
Table 285. UART ISP Go command	315	Table 321. Core register set summary	340
Table 286. UART ISP Erase sector command	316	Table 322. PSR register combinations	341
Table 287. UART ISP Blank check sector command	316	Table 323. APSR bit assignments	342
Table 288. UART ISP Read Part Identification command	316	Table 324. IPSR bit assignments	342
		Table 325. EPSR bit assignments	343
		Table 326. PRIMASK register bit assignments	343
		Table 327. CONTROL register bit assignments	344
		Table 328. Memory access behavior	348
		Table 329. Properties of different exception types	350
		Table 330. Exception return behavior	355

Table 331. Cortex-M0 instructions	358	Fig 18. Auto-CTS Functional Timing	129
Table 332. CMSIS intrinsic functions to generate some Cortex-M0 instructions	359	Fig 19. Auto-baud a) mode 0 and b) mode 1 waveform	134
Table 333. insic functions to access the special registers	360	Fig 20. Algorithm for setting UART dividers	137
Table 334. Condition code suffixes	365	Fig 21. UART block diagram	143
Table 335. Access instructions	365	Fig 22. Texas Instruments Synchronous Serial Frame Format: a) Single and b) Continuous/back-to-back Two Frames Transfer	152
Table 336. Data processing instructions	371	Fig 23. SPI frame format with CPOL=0 and CPHA=0 (a) Single and b) Continuous Transfer)	153
Table 337. ADC, ADD, RSB, SBC and SUB operand restrictions	373	Fig 24. SPI frame format with CPOL=0 and CPHA=1	154
Table 338. Branch and control instructions	380	Fig 25. SPI frame format with CPOL = 1 and CPHA = 0 (a) Single and b) Continuous Transfer)	155
Table 339. Branch ranges	381	Fig 26. SPI Frame Format with CPOL = 1 and CPHA = 1	156
Table 340. Miscellaneous instructions	382	Fig 27. Microwire frame format (single transfer)	157
Table 341. Core peripheral register regions	389	Fig 28. Microwire frame format (continuous transfers)	157
Table 342. NVIC register summary	390	Fig 29. Microwire frame format setup and hold details	158
Table 343. CMISIS access NVIC functions	390	Fig 30. I ² C-bus configuration.	160
Table 344. ISER bit assignments.	391	Fig 31. Format in the Master Transmitter mode	170
Table 345. ICER bit assignments	391	Fig 32. Format of Master Receiver mode	171
Table 346. ISPR bit assignments.	391	Fig 33. A Master Receiver switches to Master Transmitter after sending Repeated START	171
Table 347. ICPR bit assignments	392	Fig 34. Format of Slave Receiver mode	172
Table 348. IPR bit assignments.	392	Fig 35. Format of Slave Transmitter mode	172
Table 349. CMSIS functions for NVIC control	394	Fig 36. I ² C serial interface block diagram	173
Table 350. Summary of the SCB registers	395	Fig 37. Arbitration procedure.	175
Table 351. CPUID register bit assignments.	395	Fig 38. Serial clock synchronization	175
Table 352. ICSR bit assignments	396	Fig 39. Format and states in the Master Transmitter mode	180
Table 353. AIRCR bit assignments	398	Fig 40. Format and states in the Master Receiver mode	183
Table 354. SCR bit assignments	398	Fig 41. Format and states in the Slave Receiver mode	187
Table 355. CCR bit assignments	399	Fig 42. Format and states in the Slave Transmitter mode	190
Table 356. System fault handler priority fields.	399	Fig 43. Simultaneous Repeated START conditions from two masters	192
Table 357. SHPR2 register bit assignments	400	Fig 44. Forced access to a busy I ² C-bus	192
Table 358. SHPR3 register bit assignments	400	Fig 45. Recovering from a bus obstruction caused by a LOW level on SDA	193
Table 359. System timer registers summary	400	Fig 46. C_CAN block diagram.	203
Table 360. SYST_CSR bit assignments	401	Fig 47. CAN core in Silent mode.	226
Table 361. SYST_RVR bit assignments	401	Fig 48. CAN core in Loop-back mode.	227
Table 362. SYST_CVR bit assignments	402	Fig 49. CAN core in Loop-back mode combined with Silent mode	227
Table 363. SYST_CALIB register bit assignments	402	Fig 50. Block diagram of a message object transfer	229
Table 364. Cortex M0- instruction summary	402	Fig 51. Reading a message from the FIFO buffer to the message buffer	235
Table 365. Abbreviations	406	Fig 52. Bit timing	238
Figures		Fig 53. CAN API pointer structure.	240
Fig 1. LPC111x/LPC11Cx block diagram	8	Fig 54. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.	264
Fig 2. LPC111x/LPC11Cx memory map.	11	Fig 55. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled	264
Fig 3. LPC111x/LPC11Cx CGU block diagram	14	Fig 56. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled	264
Fig 4. Start-up timing	35	Fig 57. 16-bit counter/timer block diagram	265
Fig 5. System PLL block diagram	41	Fig 58. Sample PWM waveforms with a PWM cycle length	
Fig 6. Power profiles pointer structure	48		
Fig 7. LPC111x/102/202/302 clock configuration for power API use	49		
Fig 8. Power profiles usage	53		
Fig 9. Standard I/O pin configuration	59		
Fig 10. Pin configuration LQFP48 package	92		
Fig 11. Pin configuration PLCC44 package	93		
Fig 12. Pin configuration HVQFN 33 package.	94		
Fig 13. Pin configuration LQFP48 package	95		
Fig 14. Pin configuration (LPC11C22/C24)	96		
Fig 15. Masked write operation to the GPIO DATA register.	115		
Fig 16. Masked read operation	116		
Fig 17. Auto-RTS Functional Timing	128		

of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.	278
Fig 59. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled	278
Fig 60. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled	278
Fig 61. 32-bit counter/timer block diagram.	279
Fig 62. Windowed Watchdog Timer (WWDT) block diagram	282
Fig 63. Early Watchdog Feed with Windowed Mode Enabled	286
Fig 64. Correct Watchdog Feed with Windowed Mode Enabled	287
Fig 65. Watchdog Warning Interrupt	287
Fig 66. Watchdog block diagram	292
Fig 67. System tick timer block diagram	293
Fig 68. Boot process flowchart	306
Fig 69. IAP parameter passing	325
Fig 70. Algorithm for generating a 128-bit signature	334
Fig 71. Connecting the SWD pins to a standard SWD connector	336
Fig 72. Cortex-M0 implementation.	337
Fig 73. Processor core register set	340
Fig 74. APSR, IPSR, EPSR register bit assignments	341
Fig 75. Generic ARM Cortex-M0 memory map	346
Fig 76. Memory ordering restrictions	347
Fig 77. Little-endian format	349
Fig 78. Vector table	352
Fig 79. Exception entry stack contents	354
Fig 80. ASR #3	361
Fig 81. LSR #3.	362
Fig 82. LSL #3	362
Fig 83. ROR #3	363
Fig 84. IPR register	392

24.5 内容

Chapter 1: LPC111x/LPC11Cxx Introductory information

1.1	Introduction	3	1.4	Block diagram	8
1.2	Features	4	1.5	ARM Cortex-M0 processor	9
1.3	Ordering information	5			

Chapter 2: LPC111x/LPC11Cxx Memory mapping

2.1	How to read this chapter	10	2.2	Memory map	10
-----	--------------------------------	----	-----	------------------	----

Chapter 3: LPC111x/LPC11Cxx System configuration (SYSCON)

3.1	How to read this chapter	12	3.5.32	Deep-sleep mode configuration register	30
	C_CAN controller	12	3.5.33	Wake-up configuration register	31
	Entering Deep power-down mode	12	3.5.34	Power-down configuration register	32
	Enabling sequence for UART clock	12	3.5.35	Device ID register	33
3.2	General description	12	3.6	Reset	34
3.3	Pin description	12	3.7	Start-up behavior	35
3.4	Clock generation	13	3.8	Brown-out detection	35
3.5	Register description	14	3.9	Power management	36
3.5.1	System memory remap register	16	3.9.1	Active mode	36
3.5.2	Peripheral reset control register	16	3.9.1.1	Power configuration in Active mode	36
3.5.3	System PLL control register	17	3.9.2	Sleep mode	36
3.5.4	System PLL status register	17	3.9.2.1	Power configuration in Sleep mode	36
3.5.5	System oscillator control register	18	3.9.2.2	Programming Sleep mode	37
3.5.6	Watchdog oscillator control register	18	3.9.2.3	Wake-up from Sleep mode	37
3.5.7	Internal resonant crystal control register	19	3.9.3	Deep-sleep mode	37
3.5.8	System reset status register	20	3.9.3.1	Power configuration in Deep-sleep mode	37
3.5.9	System PLL clock source select register	20	3.9.3.2	Programming Deep-sleep mode	37
3.5.10	System PLL clock source update enable register	21	3.9.3.3	Wake-up from Deep-sleep mode	38
3.5.11	Main clock source select register	21	3.9.4	Deep power-down mode	38
3.5.12	Main clock source update enable register	21	3.9.4.1	Power configuration in Deep power-down mode	39
3.5.13	System AHB clock divider register	22	3.9.4.2	Programming Deep power-down mode	39
3.5.14	System AHB clock control register	22	3.9.4.3	Wake-up from Deep power-down mode	39
3.5.15	SPI0 clock divider register	24	3.10	Deep-sleep mode details	40
3.5.16	UART clock divider register	24	3.10.1	IRC oscillator	40
3.5.17	SPI1 clock divider register	24	3.10.2	Start logic	40
3.5.18	WDT clock source select register	25	3.10.3	Using the general purpose counter/timers to create a self-wake-up event	40
3.5.19	WDT clock source update enable register	25	3.11	System PLL functional description	41
3.5.20	WDT clock divider register	25	3.11.1	Lock detector	41
3.5.21	CLKOUT clock source select register	26	3.11.2	Power-down control	42
3.5.22	CLKOUT clock source update enable register	26	3.11.3	Divider ratio programming	42
3.5.23	CLKOUT clock divider register	27		Post divider	42
3.5.24	POR captured PIO status register 0	27		Feedback divider	42
3.5.25	POR captured PIO status register 1	27		Changing the divider values	42
3.5.26	BOD control register	28	3.11.4	Frequency selection	42
3.5.27	System tick counter calibration register	28	3.11.4.1	Normal mode	43
3.5.28	Start logic edge control register 0	28	3.11.4.2	Power-down mode	43
3.5.29	Start logic signal enable register 0	29	3.12	Flash memory access	43
3.5.30	Start logic reset register 0	29			
3.5.31	Start logic status register 0	30			

Chapter 4: LPC111x/LPC11Cxx Power Monitor Unit (PMU)

4.1	How to read this chapter	45	4.2	Introduction	45
-----	--------------------------------	----	-----	--------------------	----

4.3	Register description	45	4.3.3	General purpose register 4	46
4.3.1	Power control register	45	4.4	Functional description	47
4.3.2	General purpose registers 0 to 3	46			

Chapter 5: LPC111x/LPC11Cxx Power profiles

5.1	How to read this chapter	48	5.5.1.4.4	System clock less than or equal to the expected value	52
5.2	Features	48	5.5.1.4.5	System clock greater than or equal to the expected value	52
5.3	Description	48	5.5.1.4.6	System clock approximately equal to the expected value	52
5.4	Definitions	49	5.6	Power routine	53
5.5	Clocking routine	49	5.6.1	set_power	53
5.5.1	set_pll	49	5.6.1.1	Param0: main clock	54
5.5.1.1	Param0: system PLL input frequency and Param1: expected system clock	50	5.6.1.2	Param1: mode	54
5.5.1.2	Param2: mode	50	5.6.1.3	Param2: system clock	54
5.5.1.3	Param3: system PLL lock time-out	51	5.6.1.4	Code examples	55
5.5.1.4	Code examples	51	5.6.1.4.1	Invalid frequency (device maximum clock rate exceeded)	55
5.5.1.4.1	Invalid frequency (device maximum clock rate exceeded)	51	5.6.1.4.2	An applicable power setup	55
5.5.1.4.2	Invalid frequency selection (system clock divider restrictions)	51			
5.5.1.4.3	Exact solution cannot be found (PLL)	52			

Chapter 6: LPC111x/LPC11Cxx Nested Vectored Interrupt Controller (NVIC)

6.1	How to read this chapter	56	6.3	Features	56
6.2	Introduction	56	6.4	Interrupt sources	56

Chapter 7: LPC111x/LPC11Cxx I/O configuration (IOCONFIG)

7.1	How to read this chapter	58	7.4.16	IOCON_PIO2_5	72
	C_CAN pins	58	7.4.17	IOCON_PIO3_5	73
	Pseudo open-drain function	58	7.4.18	IOCON_PIO0_6	73
	Pull-up level	58	7.4.19	IOCON_PIO0_7	74
7.2	Features	58	7.4.20	IOCON_PIO2_9	74
7.3	General description	59	7.4.21	IOCON_PIO2_10	75
7.3.1	Pin function	59	7.4.22	IOCON_PIO2_2	76
7.3.2	Pin mode	59	7.4.23	IOCON_PIO0_8	76
7.3.3	Hysteresis	60	7.4.24	IOCON_PIO0_9	77
7.3.4	A/D-mode	60	7.4.25	IOCON_SWCLK_PIO0_10	77
7.3.5	I ² C mode	60	7.4.26	IOCON_PIO1_10	78
7.3.6	Open-drain Mode	60	7.4.27	IOCON_PIO2_11	79
7.4	Register description	60	7.4.28	IOCON_R_PIO0_11	79
7.4.1	IOCON_PIO2_6	64	7.4.29	IOCON_R_PIO1_0	80
7.4.2	IOCON_PIO2_0	65	7.4.30	IOCON_R_PIO1_1	81
7.4.3	IOCON_PIO_RESET_PIO0_0	65	7.4.31	IOCON_R_PIO1_2	82
7.4.4	IOCON_PIO0_1	66	7.4.32	IOCON_PIO3_0	82
7.4.5	IOCON_PIO1_8	66	7.4.33	IOCON_PIO3_1	83
7.4.6	IOCON_PIO0_2	67	7.4.34	IOCON_PIO2_3	83
7.4.7	IOCON_PIO2_7	68	7.4.35	IOCON_SWDIO_PIO1_3	84
7.4.8	IOCON_PIO2_8	68	7.4.36	IOCON_PIO1_4	85
7.4.9	IOCON_PIO2_1	69	7.4.37	IOCON_PIO1_11	86
7.4.10	IOCON_PIO0_3	69	7.4.38	IOCON_PIO3_2	86
7.4.11	IOCON_PIO0_4	70	7.4.39	IOCON_PIO1_5	87
7.4.12	IOCON_PIO0_5	70	7.4.40	IOCON_PIO1_6	87
7.4.13	IOCON_PIO1_9	70	7.4.41	IOCON_PIO1_7	88
7.4.14	IOCON_PIO3_4	71	7.4.42	IOCON_PIO3_3	89
7.4.15	IOCON_PIO2_4	72	7.4.43	IOCON_SCK_LOC	89

7.4.44	IOCON_DSR_LOC	90	7.4.46	IOCON_RI_LOC	90
7.4.45	IOCON_DCD_LOC	90			

Chapter 8: LPC111x/LPC11Cxx Pin configuration

8.1	How to read this chapter	91	8.3	LPC11Cxx Pin configuration	95
8.2	LPC111x Pin configuration	92	8.4	LPC111x/LPC11Cxx Pin description	96

Chapter 9: LPC111x/LPC11Cxx General Purpose I/O (GPIO)

9.1	How to read this chapter	110	9.3.6	GPIO interrupt mask register	113
9.2	Introduction	110	9.3.7	GPIO raw interrupt status register	113
9.2.1	Features	110	9.3.8	GPIO masked interrupt status register	113
9.3	Register description	110	9.3.9	GPIO interrupt clear register	114
9.3.1	GPIO data register	111	9.4	Functional description	114
9.3.2	GPIO data direction register	112	9.4.1	Write/read data operation	114
9.3.3	GPIO interrupt sense register	112		Write operation	114
9.3.4	GPIO interrupt both edges sense register ..	112		Read operation	116
9.3.5	GPIO interrupt event register	113			

Chapter 10: LPC111x/LPC11Cxx UART

10.1	How to read this chapter	117	10.5.12	UART Auto-baud Control Register (U0ACR - 0x4000 8020)	132
10.2	Basic configuration	117	10.5.13	Auto-baud	132
10.3	Features	117	10.5.14	Auto-baud modes	133
10.4	Pin description	118	10.5.15	UART Fractional Divider Register (U0FDR - 0x4000 8028)	135
10.5	Register description	118	10.5.15.1	Baud rate calculation	136
10.5.1	UART Receiver Buffer Register (U0RBR - 0x4000 8000, when DLAB = 0, Read Only) ..	120	10.5.15.1.1	Example 1: UART_PCLK = 14.7456 MHz, BR = 9600	138
10.5.2	UART Transmitter Holding Register (U0THR - 0x4000 8000 when DLAB = 0, Write Only) ..	120	10.5.15.1.2	Example 2: UART_PCLK = 12 MHz, BR = 115200	138
10.5.3	UART Divisor Latch LSB and MSB Registers (U0DLL - 0x4000 8000 and U0DLM - 0x4000 8004, when DLAB = 1)	120	10.5.16	UART Transmit Enable Register (U0TER - 0x4000 8030)	138
10.5.4	UART Interrupt Enable Register (U0IER - 0x4000 8004, when DLAB = 0)	121	10.5.17	UART RS485 Control register (U0RS485CTRL - 0x4000 804C)	139
10.5.5	UART Interrupt Identification Register (U0IIR - 0x4004 8008, Read Only)	122	10.5.18	UART RS485 Address Match register (U0RS485ADRMATCH - 0x4000 8050)	140
10.5.6	UART FIFO Control Register (U0FCR - 0x4000 8008, Write Only)	124	10.5.19	UART1 RS485 Delay value register (U0RS485DLY - 0x4000 8054)	140
10.5.7	UART Line Control Register (U0LCR - 0x4000 800C)	125	10.5.20	RS-485/EIA-485 modes of operation	140
10.5.8	UART Modem Control Register	126		RS-485/EIA-485 Normal Multidrop Mode (NMM)	141
10.5.8.1	Auto-flow control	127		RS-485/EIA-485 Auto Address Detection (AAD) mode	141
10.5.8.1.1	Auto-RTS	127		RS-485/EIA-485 Auto Direction Control	141
10.5.8.1.2	Auto-CTS	128		RS485/EIA-485 driver delay time	142
10.5.9	UART Line Status Register (U0LSR - 0x4000 8014, Read Only)	129		RS485/EIA-485 output inversion	142
10.5.10	UART Modem Status Register	131	10.6	Architecture	142
10.5.11	UART Scratch Pad Register (U0SCR - 0x4000 801C)	131			

Chapter 11: LPC111x/LPC11Cxx SPI0/1 with SSP

11.1	How to read this chapter	144	11.5	Pin description	145
11.2	Basic configuration	144	11.6	Register description	145
11.3	Features	144	11.6.1	SPI/SSP Control Register 0	146
11.4	General description	144	11.6.2	SPI/SSP0 Control Register 1	147

11.6.3	SPI/SSP Data Register	148	11.7.2	SPI frame format	152
11.6.4	SPI/SSP Status Register	149	11.7.2.1	Clock Polarity (CPOL) and Phase (CPHA) control	152
11.6.5	SPI/SSP Clock Prescale Register	149	11.7.2.2	SPI format with CPOL=0,CPHA=0.	153
11.6.6	SPI/SSP Interrupt Mask Set/Clear Register	149	11.7.2.3	SPI format with CPOL=0,CPHA=1.	154
11.6.7	SPI/SSP Raw Interrupt Status Register	150	11.7.2.4	SPI format with CPOL = 1,CPHA = 0.	154
11.6.8	SPI/SSP Masked Interrupt Status Register .	150	11.7.2.5	SPI format with CPOL = 1,CPHA = 1.	156
11.6.9	SPI/SSP Interrupt Clear Register	151	11.7.3	Semiconductor Microwire frame format	156
11.7	Functional description	151	11.7.3.1	Setup and hold time requirements on CS with respect to SK in Microwire mode	158
11.7.1	Texas Instruments synchronous serial frame format	151			

Chapter 12: LPC111x/LPC11Cxx I2C-bus controller

12.1	How to read this chapter	159	12.9.7	Serial clock generator	175
12.2	Basic configuration	159	12.9.8	Timing and control	176
12.3	Features	159	12.9.9	Control register, CONSET and CONCLR ..	176
12.4	Applications	159	12.9.10	Status decoder and status register.	176
12.5	General description	159	12.10	Details of I2C operating modes	176
12.5.1	I2C Fast-mode Plus	160	12.10.1	Master Transmitter mode	177
12.6	Pin description	161	12.10.2	Master Receiver mode	181
12.7	Register description	161	12.10.3	Slave Receiver mode	184
12.7.1	I2C Control Set register (I2C0CONSET - 0x4000 0000)	162	12.10.4	Slave Transmitter mode	188
12.7.2	I2C Status register (I2C0STAT - 0x4000 0004) ...	164	12.10.5	Miscellaneous states	190
12.7.3	I2C Data register (I2C0DAT - 0x4000 0008) .	164	12.10.5.1	STAT = 0xF8	190
12.7.4	I2C Slave Address register 0 (I2C0ADR0- 0x4000 000C)	164	12.10.5.2	STAT = 0x00	190
12.7.5	I2C SCL HIGH and LOW duty cycle registers (I2C0SCLH - 0x4000 0010 and I2C0SCLL- 0x4000 0014)	165	12.10.6	Some special cases	191
12.7.5.1	Selecting the appropriate I2C data rate and duty cycle	165	12.10.6.1	Simultaneous Repeated START conditions from two masters	191
12.7.6	I2C Control Clear register (I2C0CONCLR - 0x4000 0018)	166	12.10.6.2	Data transfer after loss of arbitration	192
12.7.7	I2C Monitor mode control register (I2C0MMCTRL - 0x4000 001C)	166	12.10.6.3	Forced access to the I2C-bus.	192
12.7.7.1	Interrupt in Monitor mode	167	12.10.6.4	I2C-bus obstructed by a LOW level on SCL or SDA	193
12.7.7.2	Loss of arbitration in Monitor mode	168	12.10.6.5	Bus error	193
12.7.8	I2C Slave Address registers (I2C0ADR[1, 2, 3] - 0x4000 00[20, 24, 28])	168	12.10.7	I2C state service routines	193
12.7.9	I2C Data buffer register (I2C0DATA_BUFFER - 0x4000 002C)	168	12.10.8	Initialization	194
12.7.10	I2C Mask registers (I2C0MASK[0, 1, 2, 3] - 0x4000 00[30, 34, 38, 3C])	169	12.10.9	I2C interrupt service	194
12.8	I2C operating modes	169	12.10.10	The state service routines	194
12.8.1	Master Transmitter mode	169	12.10.11	Adapting state services to an application. .	194
12.8.2	Master Receiver mode	170	12.11	Software example	194
12.8.3	Slave Receiver mode	171	12.11.1	Initialization routine	194
12.8.4	Slave Transmitter mode	172	12.11.2	Start Master Transmit function	194
12.9	I2C implementation and operation	172	12.11.3	Start Master Receive function	195
12.9.1	Input filters and output stages.	173	12.11.4	I2C interrupt routine	195
12.9.2	Address Registers, ADDR0 to ADDR3.	174	12.11.5	Non mode specific states.	195
12.9.3	Address mask registers, MASK0 to MASK3.	174	12.11.5.1	State: 0x00	195
12.9.4	Comparator.	174	12.11.5.2	Master States	195
12.9.5	Shift register, DAT.	174	12.11.5.3	State: 0x08	195
12.9.6	Arbitration and synchronization logic	174	12.11.5.4	State: 0x10	196
			12.11.6	Master Transmitter states	196
			12.11.6.1	State: 0x18	196
			12.11.6.2	State: 0x20	196
			12.11.6.3	State: 0x28	196
			12.11.6.4	State: 0x30	197
			12.11.6.5	State: 0x38	197
			12.11.7	Master Receive states	197
			12.11.7.1	State: 0x40	197
			12.11.7.2	State: 0x48	197
			12.11.7.3	State: 0x50	197

12.11.7.4	State: 0x58	198	12.11.8.8	State: 0x98	200
12.11.8	Slave Receiver states	198	12.11.8.9	State: 0xA0	200
12.11.8.1	State: 0x60	198	12.11.9	Slave Transmitter states	200
12.11.8.2	State: 0x68	198	12.11.9.1	State: 0xA8	200
12.11.8.3	State: 0x70	198	12.11.9.2	State: 0xB0	200
12.11.8.4	State: 0x78	199	12.11.9.3	State: 0xB8	200
12.11.8.5	State: 0x80	199	12.11.9.4	State: 0xC0	201
12.11.8.6	State: 0x88	199	12.11.9.5	State: 0xC8	201
12.11.8.7	State: 0x90	199			

Chapter 13: LPC111x/LPC11Cx C_CAN controller

13.1	How to read this chapter	202	13.6.3.3	CAN new data 1 register	221
13.2	Basic configuration	202	13.6.3.4	CAN new data 2 register	222
13.3	Features	202	13.6.3.5	CAN interrupt pending 1 register	222
13.4	General description	203	13.6.3.6	CAN interrupt pending 2 register	223
13.5	Pin description	204	13.6.3.7	CAN message valid 1 register	223
13.6	Register description	204	13.6.3.8	CAN message valid 2 register	223
13.6.1	CAN protocol registers	206	13.6.4	CAN timing register	224
13.6.1.1	CAN control register	206	13.6.4.1	CAN clock divider register	224
13.6.1.2	CAN status register	207	13.7	Functional description	224
13.6.1.3	CAN error counter	209	13.7.1	C_CAN controller state after reset	224
13.6.1.4	CAN bit timing register	210	13.7.2	C_CAN operating modes	224
	Baud rate prescaler	210	13.7.2.1	Software initialization	224
	Time segments 1 and 2	210	13.7.2.2	CAN message transfer	225
	Synchronization jump width	210	13.7.2.3	Disabled Automatic Retransmission (DAR)	225
13.6.1.5	CAN interrupt register	211	13.7.2.4	Test modes	226
13.6.1.6	CAN test register	211	13.7.2.4.1	Silent mode	226
13.6.1.7	CAN baud rate prescaler extension register	212	13.7.2.4.2	Loop-back mode	226
13.6.2	Message interface registers	212	13.7.2.4.3	Loop-back mode combined with Silent mode	227
13.6.2.1	Message objects	213	13.7.2.4.4	Basic mode	227
13.6.2.2	CAN message interface command request registers	213	13.7.2.4.5	Software control of pin CAN_TXD	228
13.6.2.3	CAN message interface command mask registers	214	13.7.3	CAN message handler	228
13.6.2.4	IF1 and IF2 message buffer registers	216	13.7.3.1	Management of message objects	229
13.6.2.4.1	CAN message interface command mask 1 registers	217	13.7.3.2	Data Transfer between IFx Registers and the Message RAM	230
13.6.2.4.2	CAN message interface command mask 2 registers	217	13.7.3.3	Transmission of messages between the shift registers in the CAN core and the Message buffer	230
13.6.2.4.3	CAN message interface command arbitration 1 registers	217	13.7.3.4	Acceptance filtering of received messages	231
13.6.2.4.4	CAN message interface command arbitration 2 registers	218	13.7.3.4.1	Reception of a data frame	231
13.6.2.4.5	CAN message interface message control registers	218	13.7.3.4.2	Reception of a remote frame	231
13.6.2.4.6	CAN message interface data A1 registers	219	13.7.3.5	Receive/transmit priority	232
13.6.2.4.7	CAN message interface data A2 registers	220	13.7.3.6	Configuration of a transmit object	232
13.6.2.4.8	CAN message interface data B1 registers	220	13.7.3.7	Updating a transmit object	232
13.6.2.4.9	CAN message interface data B2 registers	220	13.7.3.8	Configuration of a receive object	233
13.6.3	Message handler registers	220	13.7.3.9	Handling of received messages	233
13.6.3.1	CAN transmission request 1 register	221	13.7.3.10	Configuration of a FIFO buffer	234
13.6.3.2	CAN transmission request 2 register	221	13.7.3.10.1	Reception of messages with FIFO buffers	234
			13.7.3.10.2	Reading from a FIFO buffer	234
			13.7.4	Interrupt handling	235
			13.7.5	Bit timing	236
			13.7.5.1	Bit time and bit rate	237

Chapter 14: LPC11Cx C_CAN on-chip drivers

14.1	How to read this chapter	239	14.3	General description	239
14.2	Features	239	14.3.1	Differences to fully-compliant CANopen	239

14.4	API description	240	14.4.9	CAN/CANopen callback functions	245
14.4.1	Calling the C_CAN API	240	14.4.10	CAN message received callback	245
14.4.2	CAN initialization	241	14.4.11	CAN message transmit callback	246
14.4.3	CAN interrupt handler	241	14.4.12	CAN error callback	246
14.4.4	CAN Rx message object configuration	241	14.4.13	CANopen SDO expedited read callback	247
14.4.5	CAN receive	242	14.4.14	CANopen SDO expedited write callback	247
14.4.6	CAN transmit	242	14.4.15	CANopen SDO segmented read callback	248
14.4.7	CANopen configuration	243	14.4.16	CANopen SDO segmented write callback	249
14.4.8	CANopen handler	244	14.4.17	CANopen fall-back SDO handler callback	250

Chapter 15: LPC111x/LPC11Cxx 16-bit counter/timer CT16B0/1

15.1	How to read this chapter	252	15.7.6	Match Control Register (TMR16B0MCR and TMR16B1MCR)	257
15.2	Basic configuration	252	15.7.7	Match Registers (TMR16B0MR0/1/2/3 - addresses 0x4000 C018/1C/20/24 and TMR16B1MR0/1/2/3 - addresses 0x4001 0018/1C/20/24)	258
15.3	Features	252	15.7.8	Capture Control Register (TMR16B0CCR and TMR16B1CCR)	259
15.4	Applications	252	15.7.9	Capture Register (CT16B0CR0 - address 0x4000 C02C and CT16B1CR0 - address 0x4001 002C)	259
15.5	Description	253	15.7.10	External Match Register (TMR16B0EMR and TMR16B1EMR)	260
15.6	Pin description	253	15.7.11	Count Control Register (TMR16B0CTCR and TMR16B1CTCR)	261
15.7	Register description	253	15.7.12	PWM Control register (TMR16B0PPMC and TMR16B1PPMC)	262
15.7.1	Interrupt Register (TMR16B0IR and TMR16B1IR)	255	15.7.13	Rules for single edge controlled PWM outputs	263
15.7.2	Timer Control Register (TMR16B0TCR and TMR16B1TCR)	256	15.8	Example timer operation	264
15.7.3	Timer Counter (TMR16B0TC - address 0x4000 C008 and TMR16B1TC - address 0x4001 0008)	256	15.9	Architecture	265
15.7.4	Prescale Register (TMR16B0PR - address 0x4000 C00C and TMR16B1PR - address 0x4001 000C)	256			
15.7.5	Prescale Counter register (TMR16B0PC - address 0x4000 C010 and TMR16B1PC - address 0x4001 0010)	257			

Chapter 16: LPC111x/LPC11Cxx 32-bit counter/timer CT32B0/1

16.1	How to read this chapter	266	16.7.6	Match Control Register (TMR32B0MCR and TMR32B1MCR)	271
16.2	Basic configuration	266	16.7.7	Match Registers (TMR32B0MR0/1/2/3 - addresses 0x4001 4018/1C/20/24 and TMR32B1MR0/1/2/3 addresses 0x4001 8018/1C/20/24)	272
16.3	Features	266	16.7.8	Capture Control Register (TMR32B0CCR and TMR32B1CCR)	272
16.4	Applications	266	16.7.9	Capture Register (TMR32B0CR0 - address 0x4001 402C and TMR32B1CR0 - address 0x4001 802C)	273
16.5	Description	267	16.7.10	External Match Register (TMR32B0EMR and TMR32B1EMR)	273
16.6	Pin description	267	16.7.11	Count Control Register (TMR32B0CTCR and TMR32B1CTCR)	275
16.7	Register description	267	16.7.12	PWM Control Register (TMR32B0PPMC and TMR32B1PPMC)	276
16.7.1	Interrupt Register (TMR32B0IR and TMR32B1IR)	269	16.7.13	Rules for single edge controlled PWM outputs	277
16.7.2	Timer Control Register (TMR32B0TCR and TMR32B1TCR)	269	16.8	Example timer operation	278
16.7.3	Timer Counter (TMR32B0TC - address 0x4001 4008 and TMR32B1TC - address 0x4001 8008)	270	16.9	Architecture	279
16.7.4	Prescale Register (TMR32B0PR - address 0x4001 400C and TMR32B1PR - address 0x4001 800C)	270			
16.7.5	Prescale Counter Register (TMR32B0PC - address 0x4001 4010 and TMR32B1PC - address 0x4001 8010)	270			

Chapter 17: LPC111x/LPC11Cxx Windowed WatchDog Timer (WDT)

17.1	How to read this chapter	280	17.7.1	Watchdog Mode register	283
17.2	Basic configuration	280	17.7.2	Watchdog Timer Constant register	284
17.3	Features	280	17.7.3	Watchdog Feed register	285
17.4	Applications	281	17.7.4	Watchdog Timer Value register	285
17.5	General description	281	17.7.5	Watchdog Timer Warning Interrupt register	285
17.6	Clock control	282	17.7.6	Watchdog Timer Window register	286
17.7	Register description	283	17.7.7	Watchdog timing examples	286

Chapter 18: LPC111x/LPC11Cxx WatchDog Timer (WDT)

18.1	How to read this chapter	288	18.7.1	Watchdog Mode register (WDMOD - 0x4000 0000)	290
18.2	Basic configuration	288	18.7.2	Watchdog Timer Constant register (WDTC - 0x4000 4004)	291
18.3	Features	288	18.7.3	Watchdog Feed register (WDFEED - 0x4000 4008)	291
18.4	Applications	289	18.7.4	Watchdog Timer Value register (WDTV - 0x4000 400C)	292
18.5	Description	289	18.8	Block diagram	292
18.6	WDT clocking	289			
18.7	Register description	290			

Chapter 19: LPC111x/LPC11Cxx System tick timer (SysTick)

19.1	How to read this chapter	293	19.5.3	System Timer Current value register	295
19.2	Basic configuration	293	19.5.4	System Timer Calibration value register (SYST_CALIB - 0xE000 E01C)	296
19.3	Features	293	19.6	Functional description	296
19.4	General description	293	19.7	Example timer calculations	296
19.5	Register description	294		Example (system clock = 50 MHz)	296
19.5.1	System Timer Control and status register	294			
19.5.2	System Timer Reload value register	295			

Chapter 20: LPC111x/LPC11Cxx ADC

20.1	How to read this chapter	297	20.5.3	A/D Status Register (AD0STAT - 0x4001 C030)	301
20.2	Basic configuration	297	20.5.4	A/D Interrupt Enable Register (AD0INTEN - 0x4001 C00C)	301
20.3	Features	297	20.5.5	A/D Data Registers (AD0DR0 to AD0DR7 - 0x4001 C010 to 0x4001 C02C)	301
20.4	Pin description	297	20.6	Operation	302
20.5	Register description	298	20.6.1	Hardware-triggered conversion	302
20.5.1	A/D Control Register (AD0CR - 0x4001 C000)	298	20.6.2	Interrupts	302
20.5.2	A/D Global Data Register (AD0GDR - 0x4001 C004)	300	20.6.3	Accuracy vs. digital receiver	302

Chapter 21: LPC111x/LPC11Cxx Flash programming firmware

21.1	How to read this chapter	303	21.3.7.1	ISP entry protection	310
21.2	Features	303	21.4	UART Communication protocol	310
21.3	General description	303	21.4.1	UART ISP command format	310
21.3.1	Bootloader	303	21.4.2	UART ISP response format	310
21.3.2	Memory map after any reset	304	21.4.3	UART ISP data format	310
21.3.3	Criterion for Valid User Code	304	21.4.4	UART ISP flow control	310
21.3.4	Boot process flowchart	306	21.4.5	UART SP command abort	310
21.3.5	Sector numbers	307	21.4.6	Interrupts during UART ISP	310
21.3.6	Flash content protection mechanism	307	21.4.7	Interrupts during IAP	311
21.3.7	Code Read Protection (CRP)	308	21.4.8	RAM used by ISP command handler	311

21.4.9	RAM used by IAP command handler	311	21.6.10	Blank check sectors (C_CAN ISP).	322
21.5	UART ISP commands	311	21.6.11	Read PartID (C_CAN ISP).	322
21.5.1	Unlock <Unlock code> (UART ISP)	312	21.6.12	Read boot code version (C_CAN ISP).	322
21.5.2	Set Baud Rate <Baud Rate> <stop bit> (UART ISP).	312	21.6.13	Read serial number (C_CAN ISP).	322
21.5.3	Echo <setting> (UART ISP)	312	21.6.14	Compare (C_CAN ISP)	322
21.5.4	Write to RAM <start address> <number of bytes> (UART ISP).	312	21.6.15	C_CAN ISP SDO abort codes	322
21.5.5	Read Memory <address> <no. of bytes> (UART ISP).	313	21.6.16	Differences to fully-compliant CANopen	323
21.5.6	Prepare sector(s) for write operation <start sector number> <end sector number> (UART ISP)	314	21.7	IAP commands	324
21.5.7	Copy RAM to flash <Flash address> <RAM address> <no of bytes> (UART ISP)	314	21.7.1	Prepare sector(s) for write operation (IAP)	325
21.5.8	Go <address> <mode> (UART ISP).	315	21.7.2	Copy RAM to flash (IAP)	326
21.5.9	Erase sector(s) <start sector number> <end sector number> (UART ISP).	316	21.7.3	Erase Sector(s) (IAP).	327
21.5.10	Blank check sector(s) <sector number> <end sector number> (UART ISP).	316	21.7.4	Blank check sector(s) (IAP)	327
21.5.11	Read Part Identification number (UART ISP)	316	21.7.5	Read Part Identification number (IAP)	327
21.5.12	Read Boot code version number (UART ISP)	317	21.7.6	Read Boot code version number (IAP)	328
21.5.13	Compare <address1> <address2> <no of bytes> (UART ISP).	318	21.7.7	Compare <address1> <address2> <no of bytes> (IAP).	328
21.5.14	ReadUID (UART ISP).	318	21.7.8	Reinvoke ISP (IAP)	328
21.5.15	UART ISP Return Codes	318	21.7.9	ReadUID (IAP).	329
21.6	C_CAN communication protocol	319	21.7.10	IAP Status Codes.	329
21.6.1	C_CAN ISP SDO communication.	320	21.8	Debug notes	329
21.6.2	C_CAN ISP object directory	320	21.8.1	Comparing flash images	329
21.6.3	Unlock (C_CAN ISP)	321	21.8.2	Serial Wire Debug (SWD) flash programming interface	330
21.6.4	Write to RAM (C_CAN ISP)	321	21.9	Flash memory access.	330
21.6.5	Read memory (C_CAN ISP).	321	21.10	Flash signature generation	331
21.6.6	Prepare sectors for write operation (C_CAN ISP)	322	21.10.1	Register description for signature generation	331
21.6.7	Copy RAM to flash (C_CAN ISP)	322	21.10.1.1	Signature generation address and control registers	331
21.6.8	Go (C_CAN ISP)	322	21.10.1.2	Signature generation result registers.	332
21.6.9	Erase sectors (C_CAN ISP).	322	21.10.1.3	Flash Module Status register	333
			21.10.1.4	Flash Module Status Clear register	333
			21.10.2	Algorithm and procedure for signature generation	333
				Signature generation	333
				Content verification	334

Chapter 22: LPC111x/LPC11Cxx Serial Wire Debug (SWD)

22.1	How to read this chapter	335	22.5	Pin description	335
22.2	Features	335	22.6	Debug notes	336
22.3	Introduction	335	22.6.1	Debug limitations	336
22.4	Description	335	22.6.2	Debug connections	336

Chapter 23: LPC111x/LPC11Cxx Appendix: ARM Cortex-M0 reference

23.1	Introduction	337	23.3.1.3	Core registers	339
23.2	About the Cortex-M0 processor and core peripherals	337	23.3.1.3.1	General-purpose registers.	340
23.2.1	System-level interface	338	23.3.1.3.2	Stack Pointer	340
23.2.2	Integrated configurable debug	338	23.3.1.3.3	Link Register	341
23.2.3	Cortex-M0 processor features summary	338	23.3.1.3.4	Program Counter	341
23.2.4	Cortex-M0 core peripherals	338	23.3.1.3.5	Program Status Register	341
23.3	Processor	339	23.3.1.3.6	Exception mask register	343
23.3.1	Programmers model.	339	23.3.1.3.7	CONTROL register	343
23.3.1.1	Processor modes	339	23.3.1.4	Exceptions and interrupts	344
23.3.1.2	Stacks	339	23.3.1.5	Data types	344
			23.3.1.6	The Cortex Microcontroller Software Interface Standard.	344

23.3.2	Memory model	345	23.4.4.3.1	Syntax	367
23.3.2.1	Memory regions, types and attributes	346	23.4.4.3.2	Operation	368
23.3.2.2	Memory system ordering of memory accesses	347	23.4.4.3.3	Restrictions	368
23.3.2.3	Behavior of memory accesses	347	23.4.4.3.4	Condition flags	368
23.3.2.4	Software ordering of memory accesses	348	23.4.4.3.5	Examples	368
23.3.2.5	Memory endianness	349	23.4.4.4	LDR, PC-relative	368
23.3.2.5.1	Little-endian format	349	23.4.4.4.1	Syntax	368
23.3.3	Exception model	349	23.4.4.4.2	Operation	368
23.3.3.1	Exception states	349	23.4.4.4.3	Restrictions	368
23.3.3.2	Exception types	350	23.4.4.4.4	Condition flags	368
23.3.3.3	Exception handlers	351	23.4.4.4.5	Examples	369
23.3.3.4	Vector table	351	23.4.4.5	LDM and STM	369
23.3.3.5	Exception priorities	352	23.4.4.5.1	Syntax	369
23.3.3.6	Exception entry and return	353	23.4.4.5.2	Operation	369
23.3.3.6.1	Exception entry	353	23.4.4.5.3	Restrictions	369
23.3.3.6.2	Exception return	354	23.4.4.5.4	Condition flags	370
23.3.4	Fault handling	355	23.4.4.5.5	Examples	370
23.3.4.1	Lockup	355	23.4.4.5.6	Incorrect examples	370
23.3.5	Power management	356	23.4.4.6	PUSH and POP	370
23.3.5.1	Entering sleep mode	356	23.4.4.6.1	Syntax	370
23.3.5.1.1	Wait for interrupt	356	23.4.4.6.2	Operation	370
23.3.5.1.2	Wait for event	356	23.4.4.6.3	Restrictions	370
23.3.5.1.3	Sleep-on-exit	357	23.4.4.6.4	Condition flags	371
23.3.5.2	Wake-up from sleep mode	357	23.4.4.6.5	Examples	371
23.3.5.2.1	Wake-up from WFI or sleep-on-exit	357	23.4.5	General data processing instructions	371
23.3.5.2.2	Wake-up from WFE	357	23.4.5.1	ADC, ADD, RSB, SBC, and SUB	372
23.3.5.3	Power management programming hints	357	23.4.5.1.1	Syntax	372
23.4	Instruction set	357	23.4.5.1.2	Operation	372
23.4.1	Instruction set summary	357	23.4.5.1.3	Restrictions	373
23.4.2	Intrinsic functions	359	23.4.5.1.4	Examples	373
23.4.3	About the instruction descriptions	360	23.4.5.2	AND, ORR, EOR, and BIC	373
23.4.3.1	Operands	360	23.4.5.2.1	Syntax	374
23.4.3.2	Restrictions when using PC or SP	360	23.4.5.2.2	Operation	374
23.4.3.3	Shift Operations	361	23.4.5.2.3	Restrictions	374
23.4.3.3.1	ASR	361	23.4.5.2.4	Condition flags	374
23.4.3.3.2	LSR	361	23.4.5.2.5	Examples	374
23.4.3.3.3	LSL	362	23.4.5.3	ASR, LSL, LSR, and ROR	374
23.4.3.3.4	ROR	363	23.4.5.3.1	Syntax	374
23.4.3.4	Address alignment	363	23.4.5.3.2	Operation	375
23.4.3.5	PC-relative expressions	363	23.4.5.3.3	Restrictions	375
23.4.3.6	Conditional execution	364	23.4.5.3.4	Condition flags	375
23.4.3.6.1	The condition flags	364	23.4.5.3.5	Examples	375
23.4.3.6.2	Condition code suffixes	364	23.4.5.4	CMP and CMN	375
23.4.4	Memory access instructions	365	23.4.5.4.1	Syntax	376
23.4.4.1	ADR	365	23.4.5.4.2	Operation	376
23.4.4.1.1	Syntax	365	23.4.5.4.3	Restrictions	376
23.4.4.1.2	Operation	366	23.4.5.4.4	Condition flags	376
23.4.4.1.3	Restrictions	366	23.4.5.4.5	Examples	376
23.4.4.1.4	Condition flags	366	23.4.5.5	MOV and MVN	376
23.4.4.1.5	Examples	366	23.4.5.5.1	Syntax	376
23.4.4.2	LDR and STR, immediate offset	366	23.4.5.5.2	Operation	377
23.4.4.2.1	Syntax	366	23.4.5.5.3	Restrictions	377
23.4.4.2.2	Operation	366	23.4.5.5.4	Condition flags	377
23.4.4.2.3	Restrictions	367	23.4.5.5.5	Example	377
23.4.4.2.4	Condition flags	367	23.4.5.6	MULS	377
23.4.4.2.5	Examples	367	23.4.5.6.1	Syntax	377
23.4.4.3	LDR and STR, register offset	367	23.4.5.6.2	Operation	378
			23.4.5.6.3	Restrictions	378

23.4.5.6.4	Condition flags	378	23.4.7.6	MRS	385
23.4.5.6.5	Examples	378	23.4.7.6.1	Syntax	385
23.4.5.7	REV, REV16, and REVSH	378	23.4.7.6.2	Operation	385
23.4.5.7.1	Syntax	378	23.4.7.6.3	Restrictions	385
23.4.5.7.2	Operation	378	23.4.7.6.4	Condition flags	386
23.4.5.7.3	Restrictions	379	23.4.7.6.5	Examples	386
23.4.5.7.4	Condition flags	379	23.4.7.7	MSR	386
23.4.5.7.5	Examples	379	23.4.7.7.1	Syntax	386
23.4.5.8	SXT and UXT	379	23.4.7.7.2	Operation	386
23.4.5.8.1	Syntax	379	23.4.7.7.3	Restrictions	386
23.4.5.8.2	Operation	379	23.4.7.7.4	Condition flags	386
23.4.5.8.3	Restrictions	379	23.4.7.7.5	Examples	386
23.4.5.8.4	Condition flags	379	23.4.7.8	NOP	386
23.4.5.8.5	Examples	380	23.4.7.8.1	Syntax	386
23.4.5.9	TST	380	23.4.7.8.2	Operation	386
23.4.5.9.1	Syntax	380	23.4.7.8.3	Restrictions	386
23.4.5.9.2	Operation	380	23.4.7.8.4	Condition flags	387
23.4.5.9.3	Restrictions	380	23.4.7.8.5	Examples	387
23.4.5.9.4	Condition flags	380	23.4.7.9	SEV	387
23.4.5.9.5	Examples	380	23.4.7.9.1	Syntax	387
23.4.6	Branch and control instructions	380	23.4.7.9.2	Operation	387
23.4.6.1	B, BL, BX, and BLX	381	23.4.7.9.3	Restrictions	387
23.4.6.1.1	Syntax	381	23.4.7.9.4	Condition flags	387
23.4.6.1.2	Operation	381	23.4.7.9.5	Examples	387
23.4.6.1.3	Restrictions	381	23.4.7.10	SVC	387
23.4.6.1.4	Condition flags	382	23.4.7.10.1	Syntax	387
23.4.6.1.5	Examples	382	23.4.7.10.2	Operation	387
23.4.7	Miscellaneous instructions	382	23.4.7.10.3	Restrictions	387
23.4.7.1	BKPT	383	23.4.7.10.4	Condition flags	387
23.4.7.1.1	Syntax	383	23.4.7.10.5	Examples	388
23.4.7.1.2	Operation	383	23.4.7.11	WFE	388
23.4.7.1.3	Restrictions	383	23.4.7.11.1	Syntax	388
23.4.7.1.4	Condition flags	383	23.4.7.11.2	Operation	388
23.4.7.1.5	Examples	383	23.4.7.11.3	Restrictions	388
23.4.7.2	CPS	383	23.4.7.11.4	Condition flags	388
23.4.7.2.1	Syntax	383	23.4.7.11.5	Examples	388
23.4.7.2.2	Operation	383	23.4.7.12	WFI	388
23.4.7.2.3	Restrictions	384	23.4.7.12.1	Syntax	388
23.4.7.2.4	Condition flags	384	23.4.7.12.2	Operation	389
23.4.7.2.5	Examples	384	23.4.7.12.3	Restrictions	389
23.4.7.3	DMB	384	23.4.7.12.4	Condition flags	389
23.4.7.3.1	Syntax	384	23.4.7.12.5	Examples	389
23.4.7.3.2	Operation	384	23.5	Peripherals	389
23.4.7.3.3	Restrictions	384	23.5.1	About the ARM Cortex-M0	389
23.4.7.3.4	Condition flags	384	23.5.2	Nested Vectored Interrupt Controller	389
23.4.7.3.5	Examples	384	23.5.2.1	Accessing the Cortex-M0 NVIC registers using CMSIS	390
23.4.7.4	DSB	384	23.5.2.2	Interrupt Set-enable Register	390
23.4.7.4.1	Syntax	384	23.5.2.3	Interrupt Clear-enable Register	391
23.4.7.4.2	Operation	384	23.5.2.4	Interrupt Set-pending Register	391
23.4.7.4.3	Restrictions	384	23.5.2.5	Interrupt Clear-pending Register	392
23.4.7.4.4	Condition flags	385	23.5.2.6	Interrupt Priority Registers	392
23.4.7.4.5	Examples	385	23.5.2.7	Level-sensitive and pulse interrupts	393
23.4.7.5	ISB	385	23.5.2.7.1	Hardware and software control of interrupts	393
23.4.7.5.1	Syntax	385	23.5.2.8	NVIC usage hints and tips	394
23.4.7.5.2	Operation	385	23.5.2.8.1	NVIC programming hints	394
23.4.7.5.3	Restrictions	385	23.5.3	System Control Block	394
23.4.7.5.4	Condition flags	385			
23.4.7.5.5	Examples	385			

23.5.3.1	The CMSIS mapping of the Cortex-M0 SCB registers	395	23.5.3.7.2	System Handler Priority Register 3	400
23.5.3.2	CPUID Register	395	23.5.3.8	SCB usage hints and tips	400
23.5.3.3	Interrupt Control and State Register	395	23.5.4	System timer, SysTick	400
23.5.3.4	Application Interrupt and Reset Control Register	397	23.5.4.1	SysTick Control and Status Register	401
23.5.3.5	System Control Register	398	23.5.4.2	SysTick Reload Value Register	401
23.5.3.6	Configuration and Control Register	399	23.5.4.2.1	Calculating the RELOAD value	401
23.5.3.7	System Handler Priority Registers	399	23.5.4.3	SysTick Current Value Register	401
23.5.3.7.1	System Handler Priority Register 2	399	23.5.4.4	SysTick Calibration Value Register	402
			23.5.4.5	SysTick usage hints and tips	402
			23.6	Cortex-M0 instruction summary	402

Chapter 24: Supplementary information

24.1	Abbreviations	406	24.3.3	Trademarks	407
24.2	References	406	24.4	Tables	408
24.3	Legal information	407	24.5	Figures	415
24.3.1	Definitions	407	24.6	Contents	417
24.3.2	Disclaimers	407			

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

Table 1.	LPC111x/LPC11Cxx enhancements	3	(CLKOUTCLKSEL, address 0x4004 80E0) bit description	26
Table 2.	Ordering information	5	Table 29.	CLKOUT clock source update enable register (CLKOUTUEN, address 0x4004 80E4) bit description
Table 3.	Ordering options	6	Table 30.	CLKOUT clock divider registers (CLKOUTCLKDIV, address 0x4004 80E8) bit description
Table 4.	LPC111x memory configuration	10	Table 31.	POR captured PIO status registers 0 (PIOPORCAP0, address 0x4004 8100) bit description
Table 5.	LPC11Cxx memory configuration	10	Table 32.	POR captured PIO status registers 1 (PIOPORCAP1, address 0x4004 8104) bit description
Table 6.	Pin summary	12	Table 33.	BOD control register (BODCTRL, address 0x4004 8150) bit description
Table 7.	Register overview: system control block (base address 0x4004 8000)	14	Table 34.	System tick timer calibration register (SYSTCKCAL, address 0x4004 8154) bit description
Table 8.	System memory remap register (SYSMEMREMAP, address 0x4004 8000) bit description	16	Table 35.	Start logic edge control register 0 (STARTAPRP0, address 0x4004 8200) bit description
Table 9.	Peripheral reset control register (PRESETCTRL, address 0x4004 8004) bit description	16	Table 36.	Start logic signal enable register 0 (STARTERP0, address 0x4004 8204) bit description
Table 10.	System PLL control register (SYSPLLCTRL, address 0x4004 8008) bit description	17	Table 37.	Start logic reset register 0 (STARTRSRP0CLR, address 0x4004 8208) bit description
Table 11.	System PLL status register (SYSPLLSTAT, address 0x4004 800C) bit description	17	Table 38.	Start logic status register 0 (STARTSRP0, address 0x4004 820C) bit description
Table 12.	System oscillator control register (SYSOSCCTRL, address 0x4004 8020) bit description	18	Table 39.	Allowed values for PDSLEEPCFG register
Table 13.	Watchdog oscillator control register (WDTOSCCTRL, address 0x4004 8024) bit description	18	Table 40.	Deep-sleep configuration register (PDSLEEPCFG, address 0x4004 8230) bit description
Table 14.	Internal resonant crystal control register (IRCCTRL, address 0x4004 8028) bit description	19	Table 41.	Wake-up configuration register (PDAWAKECFG, address 0x4004 8234) bit description
Table 15.	System reset status register (SYSRSTSTAT, address 0x4004 8030) bit description	20	Table 42.	Power-down configuration register (PDRUNCFG, address 0x4004 8238) bit description
Table 16.	System PLL clock source select register (SYSPLLCLKSEL, address 0x4004 8040) bit description	20	Table 43.	Device ID register (DEVICE_ID, address 0x4004 83F4) bit description
Table 17.	System PLL clock source update enable register (SYSPLLCLKUEN, address 0x4004 8044) bit description	21	Table 44.	PLL frequency parameters
Table 18.	Main clock source select register (MAINCLKSEL, address 0x4004 8070) bit description	21	Table 45.	PLL configuration examples
Table 19.	Main clock source update enable register (MAINCLKUEN, address 0x4004 8074) bit description	22	Table 46.	Flash configuration register (FLASHCFG, address 0x4003 C010) bit description
Table 20.	System AHB clock divider register (SYSAHBCLKDIV, address 0x4004 8078) bit description	22	Table 47.	Register overview: PMU (base address 0x4003 8000)
Table 21.	System AHB clock control register (SYSAHBCLKCTRL, address 0x4004 8080) bit description	22	Table 48.	Power control register (PCON, address 0x4003 8000) bit description
Table 22.	SPI0 clock divider register (SSP0CLKDIV, address 0x4004 8094) bit description	24	Table 49.	General purpose registers 0 to 3 (GPREG0 - GPREG3, address 0x4003 8004 to 0x4003 8010) bit description
Table 23.	UART clock divider register (UARTCLKDIV, address 0x4004 8098) bit description	24	Table 50.	General purpose register 4 (GPREG4, address 0x4003 8014) bit description
Table 24.	SPI1 clock divider register (SSP1CLKDIV, address 0x4004 809C) bit description	25	Table 51.	set_pll routine
Table 25.	WDT clock source select register (WDTCLKSEL, address 0x4004 80D0) bit description	25	Table 52.	set_power routine
Table 26.	WDT clock source update enable register (WDTCLKUEN, address 0x4004 80D4) bit description	25	Table 53.	Connection of interrupt sources to the Vectored Interrupt Controller
Table 27.	WDT clock divider register (WDTCLKDIV, address 0x4004 80D8) bit description	26	Table 54.	Register overview: I/O configuration (base address 0x4004 4000)
Table 28.	CLKOUT clock source select register		Table 55.	I/O configuration registers ordered by port number

Table 56. IOCON_PIO2_6 register (IOCON_PIO2_6, address 0x4004 4000) bit description	64	description	79
Table 57. IOCON_PIO2_0 register (IOCON_PIO2_0, address 0x4004 4008) bit description	65	Table 84. IOCON_R_PIO1_0 register (IOCON_R_PIO1_0, address 0x4004 4078) bit description	80
Table 58. IOCON_RESET_PIO0_0 register (IOCON_RESET_PIO0_0, address 0x4004 400C) bit description.	65	Table 85. IOCON_R_PIO1_1 register (IOCON_R_PIO1_1, address 0x4004 407C) bit description	81
Table 59. IOCON_PIO0_1 register (IOCON_PIO0_1, address 0x4004 4010) bit description	66	Table 86. IOCON_R_PIO1_2 register (IOCON_R_PIO1_2, address 0x4004 4080) bit description	82
Table 60. IOCON_PIO1_8 register (IOCON_PIO1_8, address 0x4004 4014) bit description	66	Table 87. IOCON_PIO3_0 register (IOCON_PIO3_0, address 0x4004 4084) bit description	82
Table 61. IOCON_PIO0_2 register (IOCON_PIO0_2, address 0x4004 401C) bit description	67	Table 88. IOCON_PIO3_1 register (IOCON_PIO3_1, address 0x4004 4088) bit description	83
Table 62. IOCON_PIO2_7 register (IOCON_PIO2_7, address 0x4004 4020) bit description	68	Table 89. IOCON_PIO2_3 register (IOCON_PIO2_3, address 0x4004 408C) bit description	83
Table 63. IOCON_PIO2_8 register (IOCON_PIO2_8, address 0x4004 4024) bit description	68	Table 90. IOCON_SWDIO_PIO1_3 register (IOCON_SWDIO_PIO1_3, address 0x4004 4090) bit description	84
Table 64. IOCON_PIO2_1 register (IOCON_PIO2_1, address 0x4004 4028) bit description	69	Table 91. IOCON_PIO1_4 register (IOCON_PIO1_4, address 0x4004 4094) bit description	85
Table 65. IOCON_PIO0_3 register (IOCON_PIO0_3, address 0x4004 402C) bit description	69	Table 92. IOCON_PIO1_11 register (IOCON_PIO1_11, address 0x4004 4098) bit description	86
Table 66. IOCON_PIO0_4 register (IOCON_PIO0_4, address 0x4004 4030) bit description	70	Table 93. IOCON_PIO3_2 register (IOCON_PIO3_2, address 0x4004 409C) bit description	86
Table 67. IOCON_PIO0_5 register (IOCON_PIO0_5, address 0x4004 4034) bit description	70	Table 94. IOCON_PIO1_5 register (IOCON_PIO1_5, address 0x4004 40A0) bit description	87
Table 68. IOCON_PIO1_9 register (IOCON_PIO1_9, address 0x4004 4038) bit description	71	Table 95. IOCON_PIO1_6 register (IOCON_PIO1_6, address 0x4004 40A4) bit description	87
Table 69. IOCON_PIO3_4 register (IOCON_PIO3_4, address 0x4004 403C) bit description	71	Table 96. IOCON_PIO1_7 register (IOCON_PIO1_7, address 0x4004 40A8) bit description	88
Table 70. IOCON_PIO2_4 register (IOCON_PIO2_4, address 0x4004 4040) bit description	72	Table 97. IOCON_PIO3_3 register (IOCON_PIO3_3, address 0x4004 40AC) bit description	89
Table 71. IOCON_PIO2_5 register (IOCON_PIO2_5, address 0x4004 4044) bit description	72	Table 98. IOCON_SCK location register (IOCON_SCK_LOC, address 0x4004 40B0) bit description	89
Table 72. IOCON_PIO3_5 register (IOCON_PIO3_5, address 0x4004 4048) bit description	73	Table 99. IOCON_DSR location register (IOCON_DSR_LOC, address 0x4004 40B4) bit description	90
Table 73. IOCON_PIO0_6 register (IOCON_PIO0_6, address 0x4004 404C) bit description	73	Table 100. IOCON_DCD location register (IOCON_DCD_LOC, address 0x4004 40B8) bit description	90
Table 74. IOCON_PIO0_7 register (IOCON_PIO0_7, address 0x4004 4050) bit description.	74	Table 101. IOCON_RI location register (IOCON_RI_LOC, address 0x4004 40BC) bit description	90
Table 75. IOCON_PIO2_9 register (IOCON_PIO2_9, address 0x4004 4054) bit description	75	Table 102. LPC111x/LPC11Cx pin configurations.	91
Table 76. IOCON_PIO2_10 register (IOCON_PIO2_10, address 0x4004 4058) bit description	75	Table 103. LPC1113/14 and LPC11C12/C14 pin description table (LQFP48 package)	96
Table 77. IOCON_PIO2_2 register (IOCON_PIO2_2, address 0x4004 405C) bit description	76	Table 104. LPC1114 pin description table (PLCC44 package)	100
Table 78. IOCON_PIO0_8 register (IOCON_PIO0_8, address 0x4004 4060) bit description	76	Table 105. LPC1111/12/13/14 pin description table (HVQFN33 package)	104
Table 79. IOCON_PIO0_9 register (IOCON_PIO0_9, address 0x4004 4064) bit description	77	Table 106. LPC11C24/C22 pin description table (LQFP48 package)	106
Table 80. IOCON_SWCLK_PIO0_10 register (IOCON_SWCLK_PIO0_10, address 0x4004 4068) bit description	77	Table 107. GPIO configuration	110
Table 81. IOCON_PIO1_10 register (IOCON_PIO1_10, address 0x4004 406C) bit description	78	Table 108. Register overview: GPIO (base address port 0: 0x5000 0000; port 1: 0x5001 0000, port 2: 0x5002 0000; port 3: 0x5003 0000)	111
Table 82. IOCON_PIO2_11 register (IOCON_PIO2_11, address 0x4004 4070) bit description	79	Table 109. GPIO0nDATA register (GPIO0nDATA, address 0x5000 0000 to 0x5000 3FFC; GPIO1nDATA, address 0x5001 0000 to 0x5001 3FFC; GPIO2nDATA, address 0x5002 0000 to 0x5002	
Table 83. IOCON_R_PIO0_11 register (IOCON_R_PIO0_11, address 0x4004 4074) bit			

3FFC; GPIO3DATA, address 0x5003 0000 to 0x5003 3FFC) bit description	111	0x4000 8018) bit description	131
Table 110. GPIO0DIR register (GPIO0DIR, address 0x5000 8000 to GPIO3DIR, address 0x5003 8000) bit description	112	Table 133. UART Scratch Pad Register (UOSCR - address 0x4000 801C) bit description	131
Table 111. GPIO0IS register (GPIO0IS, address 0x5000 8004 to GPIO3IS, address 0x5003 8004) bit description	112	Table 134. Auto baud Control Register (U0ACR - address 0x4000 8020) bit description	132
Table 112. GPIO0IBE register (GPIO0IBE, address 0x5000 8008 to GPIO3IBE, address 0x5003 8008) bit description	112	Table 135. UART Fractional Divider Register (U0FDR - address 0x4000 8028) bit description	135
Table 113. GPIO0IEV register (GPIO0IEV, address 0x5000 800C to GPIO3IEV, address 0x5003 800C) bit description	113	Table 136. Fractional Divider setting look-up table	138
Table 114. GPIO0IE register (GPIO0IE, address 0x5000 8010 to GPIO3IE, address 0x5003 8010) bit description	113	Table 137. UART Transmit Enable Register (U0TER - address 0x4000 8030) bit description	139
Table 115. GPIO0RIS register (GPIO0RIS, address 0x5000 8014 to GPIO3RIS, address 0x5003 8014) bit description	113	Table 138. UART RS485 Control register (U0RS485CTRL - address 0x4000 804C) bit description	139
Table 116. GPIO0MIS register (GPIO0MIS, address 0x5000 8018 to GPIO3MIS, address 0x5003 8018) bit description	114	Table 139. UART RS485 Address Match register (U0RS485ADRMATCH - address 0x4000 8050) bit description	140
Table 117. GPIO0IC register (GPIO0IC, address 0x5000 801C to GPIO3IC, address 0x5003 801C) bit description	114	Table 140. UART RS485 Delay value register (U0RS485DLY - address 0x4000 8054) bit description	140
Table 118. UART pin description	118	Table 141. SPI pin descriptions	145
Table 119. Register overview: UART (base address: 0x4000 8000)	118	Table 142. Register overview: SPI0 (base address 0x4004 0000)	146
Table 120. UART Receiver Buffer Register (U0RBR - address 0x4000 8000 when DLAB = 0, Read Only) bit description	120	Table 143. Register overview: SPI1 (base address 0x4005 8000)	146
Table 121. UART Transmitter Holding Register (U0THR - address 0x4000 8000 when DLAB = 0, Write Only) bit description	120	Table 144. SPI/SSP Control Register 0 (SSP0CR0 - address 0x4004 0000, SSP1CR0 - address 0x4005 8000) bit description	147
Table 122. UART Divisor Latch LSB Register (U0DLL - address 0x4000 8000 when DLAB = 1) bit description	121	Table 145. SPI/SSP Control Register 1 (SSP0CR1 - address 0x4004 0004, SSP1CR1 - address 0x4005 8004) bit description	148
Table 123. UART Divisor Latch MSB Register (U0DLM - address 0x4000 8004 when DLAB = 1) bit description	121	Table 146. SPI/SSP Data Register (SSP0DR - address 0x4004 0008, SSP1DR - address 0x4005 8008) bit description	148
Table 124. UART Interrupt Enable Register (U0IER - address 0x4000 8004 when DLAB = 0) bit description	121	Table 147. SPI/SSP Status Register (SSP0SR - address 0x4004 000C, SSP1SR - address 0x4005 800C) bit description	149
Table 125. UART Interrupt Identification Register (U0IIR - address 0x4004 8008, Read Only) bit description	122	Table 148. SPI/SSP Clock Prescale Register (SSP0CPSR - address 0x4004 0010, SSP1CPSR - address 0x4005 8010) bit description	149
Table 126. UART Interrupt Handling	123	Table 149. SPI/SSP Interrupt Mask Set/Clear register (SSP0IMSC - address 0x4004 0014, SSP1IMSC - address 0x4005 8014) bit description	150
Table 127. UART FIFO Control Register (U0FCR - address 0x4000 8008, Write Only) bit description	125	Table 150. SPI/SSP Raw Interrupt Status register (SSP0RIS - address 0x4004 0018, SSP1RIS - address 0x4005 8018) bit description	150
Table 128. UART Line Control Register (U0LCR - address 0x4000 800C) bit description	125	Table 151. SPI/SSP Masked Interrupt Status register (SSP0MIS - address 0x4004 001C, SSP1MIS - address 0x4005 801C) bit description	151
Table 129. UART0 Modem Control Register (U0MCR - address 0x4000 8010) bit description	126	Table 152. SPI/SSP interrupt Clear Register (SSP0ICR - address 0x4004 0020, SSP1ICR - address 0x4005 8020) bit description	151
Table 130. Modem status interrupt generation	128	Table 153. I ² C-bus pin description	161
Table 131. UART Line Status Register (U0LSR - address 0x4000 8014, Read Only) bit description	129	Table 154. Register overview: I ² C (base address 0x4000 0000)	161
Table 132. UART Modem Status Register (U0MSR - address 0x4000 8018) bit description	131	Table 155. I ² C Control Set register (I2C0CONSET - address 0x4000 0000) bit description	162
		Table 156. I ² C Status register (I2C0STAT - 0x4000 0004) bit description	164
		Table 157. I ² C Data register (I2C0DAT - 0x4000 0008) bit description	164

Table 158. I ² C Slave Address register 0 (I2C0ADR0-0x4000 000C) bit description	165	RAM	213
Table 159. I ² C SCL HIGH Duty Cycle register (I2C0SCLH - address 0x4000 0010) bit description	165	Table 190. CAN message interface command request registers (CANIF1_CMDREQ, address 0x4005 0020 and CANIF2_CMDREQ, address 0x4005 0080) bit description	214
Table 160. I ² C SCL Low duty cycle register (I2C0SCLL - 0x4000 0014) bit description	165	Table 191. CAN message interface command mask registers (CANIF1_CMDMSK, address 0x4005 0024 and CANIF2_CMDMSK, address 0x4005 0084) bit description - write direction	214
Table 161. SCLL + SCLH values for selected I ² C clock values	165	Table 192. CAN message interface command mask registers (CANIF1_CMDMSK, address 0x4005 0024 and CANIF2_CMDMSK, address 0x4005 0084) bit description - read direction	215
Table 162. I ² C Control Clear register (I2C0CONCLR - 0x4000 0018) bit description	166	Table 193. CAN message interface command mask 1 registers (CANIF1_MSK1, address 0x4005 0028 and CANIF2_MASK1, address 0x4005 0088) bit description	217
Table 163. I ² C Monitor mode control register (I2C0MMCTRL - 0x4000 001C) bit description	167	Table 194. CAN message interface command mask 2 registers (CANIF1_MSK2, address 0x4005 002C and CANIF2_MASK2, address 0x4005 008C) bit description	217
Table 164. I ² C Slave Address registers (I2C0ADR[1, 2, 3]-0x4000 00[20, 24, 28]) bit description	168	Table 195. CAN message interface command arbitration 1 registers (CANIF1_ARB1, address 0x4005 0030 and CANIF2_ARB1, address 0x4005 0090) bit description	217
Table 165. I ² C Data buffer register (I2C0DATA_BUFFER - 0x4000 002C) bit description	169	Table 196. CAN message interface command arbitration 2 registers (CANIF1_ARB2, address 0x4005 0034 and CANIF2_ARB2, address 0x4005 0094) bit description	218
Table 166. I ² C Mask registers (I2C0MASK[0, 1, 2, 3] - 0x4000 00[30, 34, 38, 3C]) bit description	169	Table 197. CAN message interface message control registers (CANIF1_MCTRL, address 0x4005 0038 and CANIF2_MCTRL, address 0x4005 0098) bit description	218
Table 167. I2C0CONSET and I2C1CONSET used to configure Master mode.	170	Table 198. CAN message interface data A1 registers (CANIF1_DA1, address 0x4005 003C and CANIF2_DA1, address 0x4005 009C) bit description	220
Table 168. I2C0CONSET and I2C1CONSET used to configure Slave mode.	171	Table 199. CAN message interface data A2 registers (CANIF1_DA2, address 0x4005 0040 and CANIF2_DA2, address 0x4005 00A0) bit description	220
Table 169. Abbreviations used to describe an I ² C operation.	177	Table 200. CAN message interface data B1 registers (CANIF1_DB1, address 0x4005 0044 and CANIF2_DB1, address 0x4005 00A4) bit description	220
Table 170. I2C0CONSET used to initialize Master Transmitter mode	177	Table 201. CAN message interface data B2 registers (CANIF1_DB2, address 0x4005 0048 and CANIF2_DB2, address 0x4005 00A8) bit description	220
Table 171. Master Transmitter mode.	179	Table 202. CAN transmission request 1 register (CANTXREQ1, address 0x4005 0100) bit description	221
Table 172. Master Receiver mode.	182	Table 203. CAN transmission request 2 register (CANTXREQ2, address 0x4005 0104) bit description	221
Table 173. I2C0ADR and I2C1ADR usage in Slave Receiver mode.	184	Table 204. CAN new data 1 register (CANND1, address 0x4005 0120) bit description	222
Table 174. I2C0CONSET and I2C1CONSET used to initialize Slave Receiver mode	184		
Table 175. Slave Receiver mode	185		
Table 176. Slave Transmitter mode.	189		
Table 177. Miscellaneous States	191		
Table 178. CAN pin description (LPC11C12/C14).	204		
Table 179. CAN pin description (LPC11C22/C24).	204		
Table 180. Register overview: CCAN (base address 0x4005 0000)	204		
Table 181. CAN control registers (CANCNTL, address 0x4005 0000) bit description	206		
Table 182. CAN status register (CANSTAT, address 0x4005 0004) bit description	208		
Table 183. CAN error counter (CANEC, address 0x4005 0008) bit description	209		
Table 184. CAN bit timing register (CANBT, address 0x4005 000C) bit description	210		
Table 185. CAN interrupt register (CANINT, address 0x4005 0010) bit description	211		
Table 186. CAN test register (CANTEST, address 0x4005 0014) bit description	211		
Table 187. CAN baud rate prescaler extension register (CANBRPE, address 0x4005 0018) bit description	212		
Table 188. Message interface registers.	213		
Table 189. Structure of a message object in the message			

Table 205. CAN new data 2 register (CANND2, address 0x4005 0124) bit description	222	Table 231. Register overview: 32-bit counter/timer 0 CT32B0 (base address 0x4001 4000)	267
Table 206. CAN interrupt pending 1 register (CANIR1, address 0x4005 0140) bit description.	222	Table 232. Register overview: 32-bit counter/timer 1 CT32B1 (base address 0x4001 8000)	268
Table 207. CAN interrupt pending 2 register (CANIR2, addresses 0x4005 0144) bit description.	223	Table 233. Interrupt Register (TMR32B0IR - address 0x4001 4000 and TMR32B1IR - address 0x4001 8000) bit description	269
Table 208. CAN message valid 1 register (CANMSGV1, addresses 0x4005 0160) bit description.	223	Table 234. Timer Control Register (TMR32B0TCR - address 0x4001 4004 and TMR32B1TCR - address 0x4001 8004) bit description	270
Table 209. CAN message valid 2 register (CANMSGV2, address 0x4005 0164) bit description.	223	Table 235. Timer counter registers (TMR32B0TC, address 0x4001 4008 and TMR32B1TC 0x4001 8008) bit description	270
Table 210. CAN clock divider register (CANCLKDIV, address 0x4005 0180) bit description	224	Table 236. Prescale registers (TMR32B0PR, address 0x4001 400C and TMR32B1PR 0x4001 800C) bit description	270
Table 211. Initialization of a transmit object.	232	Table 237. Prescale counter registers (TMR32B0PC, address 0x4001 4010 and TMR32B1PC 0x4001 8010) bit description	271
Table 212. Initialization of a receive object	233	Table 238. Match Control Register (TMR32B0MCR - address 0x4001 4014 and TMR32B1MCR - address 0x4001 8014) bit description	271
Table 213. Parameters of the C_CAN bit time.	237	Table 239. Match registers (TMR32B0MR0 to 3, addresses 0x4001 4018 to 24 and TMR32B1MR0 to 3, addresses 0x4001 8018 to 24) bit description	272
Table 214. Counter/timer pin description.	253	Table 240. Capture Control Register (TMR32B0CCR - address 0x4001 4028 and TMR32B1CCR - address 0x4001 8028) bit description	272
Table 215. Register overview: 16-bit counter/timer 0 CT16B0 (base address 0x4000 C000)	254	Table 241. Capture registers (TMR32B0CR0, addresses 0x4001 402C and TMR32B1CR0, addresses 0x4001 802C) bit description	273
Table 216. Register overview: 16-bit counter/timer 1 CT16B1 (base address 0x4001 0000)	255	Table 242. External Match Register (TMR32B0EMR - address 0x4001 403C and TMR32B1EMR - address 0x4001 803C) bit description	274
Table 217. Interrupt Register (TMR16B0IR - address 0x4000 C000 and TMR16B1IR - address 0x4001 0000) bit description	256	Table 243. External match control	275
Table 218. Timer Control Register (TMR16B0TCR - address 0x4000 C004 and TMR16B1TCR - address 0x4001 0004) bit description	256	Table 244. Count Control Register (TMR32B0CTCR - address 0x4001 4070 and TMR32B1TCR - address 0x4001 8070) bit description	276
Table 219. Timer counter registers (TMR16B0TC, address 0x4000 C008 and TMR16B1TC 0x4001 0008) bit description	256	Table 245. PWM Control Register (TMR32B0PWWC - 0x4001 4074 and TMR32B1PWWC - 0x4001 8074) bit description.	276
Table 220. Prescale registers (TMR16B0PR, address 0x4000 C00C and TMR16B1PR 0x4001 000C) bit description	257	Table 246. Register overview: Watchdog timer (base address 0x4000 4000)	283
Table 221. Prescale counter registers (TMR16B0PC, address 0x4001 C010 and TMR16B1PC 0x4000 0010) bit description	257	Table 247. Watchdog Mode register (WDMOD - 0x4000 4000) bit description	283
Table 222. Match Control Register (TMR16B0MCR - address 0x4000 C014 and TMR16B1MCR - address 0x4001 0014) bit description	257	Table 248. Watchdog operating modes selection	284
Table 223. Match registers (TMR16B0MR0 to 3, addresses 0x4000 C018 to 24 and TMR16B1MR0 to 3, addresses 0x4001 0018 to 24) bit description	259	Table 249. Watchdog Timer Constant register (WDTC - 0x4000 4004) bit description	285
Table 224. Capture Control Register (TMR16B0CCR - address 0x4000 C028 and TMR16B1CCR - address 0x4001 0028) bit description.	259	Table 250. Watchdog Feed register (WDFEED - 0x4000 4008) bit description	285
Table 225. Capture registers (TMR16B0CR0, address 0x4000 C02C and TMR16B1CR0, address 0x4001 002C) bit description	259	Table 251. Watchdog Timer Value register (WDTV - 0x4000 400C) bit description	285
Table 226. External Match Register (TMR16B0EMR - address 0x4000 C03C and TMR16B1EMR - address 0x4001 003C) bit description	260	Table 252. Watchdog Timer Warning Interrupt register (WDWARNINT - 0x4000 4014) bit description	286
Table 227. External match control.	261	Table 253. Watchdog Timer Window register (WDWINDOW - 0x4000 4018) bit description	286
Table 228. Count Control Register (TMR16B0CTCR - address 0x4000 C070 and TMR16B1CTCR - address 0x4001 0070) bit description.	262	Table 254. Register overview: Watchdog timer (base address 0x4000 4000)	290
Table 229. PWM Control Register (TMR16B0PWWC - address 0x4000 C074 and TMR16B1PWWC - address 0x4001 0074) bit description.	262		
Table 230. Counter/timer pin description.	267		

Table 255. Watchdog Mode register (WDMOD - address 0x4000 4000) bit description	290	Table 291. UART ISP Compare command	318
Table 256. Watchdog operating modes selection	291	Table 292. UART ISP ReadUID command	318
Table 257. Watchdog Constant register (WDTC - address 0x4000 4004) bit description	291	Table 293. UART ISP Return Codes Summary	318
Table 258. Watchdog Feed register (WDFEED - address 0x4000 4008) bit description	292	Table 294. C_CAN ISP and UART ISP command summary	320
Table 259. Watchdog Timer Value register (WDTV - address 0x4000 000C) bit description	292	Table 295. C_CAN ISP object directory	320
Table 260. Register overview: SysTick timer (base address 0xE000 E000).	294	Table 296. C_CAN ISP SDO abort codes.	323
Table 261. SysTick Timer Control and status register (SYST_CSR - 0xE000 E010) bit description	295	Table 297. IAP Command Summary	325
Table 262. System Timer Reload value register (SYST_RVR - 0xE000 E014) bit description	295	Table 298. IAP Prepare sector(s) for write operation command	326
Table 263. System Timer Current value register (SYST_CVR - 0xE000 E018) bit description	295	Table 299. IAP Copy RAM to flash command.	326
Table 264. System Timer Calibration value register (SYST_CALIB - 0xE000 E01C) bit description	296	Table 300. IAP Erase Sector(s) command	327
Table 265. ADC pin description	297	Table 301. IAP Blank check sector(s) command	327
Table 266. Register overview: ADC (base address 0x4001 C000)	298	Table 302. IAP Read Part Identification command	327
Table 267. A/D Control Register (AD0CR - address 0x4001 C000) bit description	299	Table 303. IAP Read Boot Code version number command	328
Table 268. A/D Global Data Register (AD0GDR - address 0x4001 C004) bit description	300	Table 304. IAP Compare command	328
Table 269. A/D Status Register (AD0STAT - address 0x4001 C030) bit description	301	Table 305. IAP Reinvoke ISP	328
Table 270. A/D Interrupt Enable Register (AD0INTEN - address 0x4001 C00C) bit description	301	Table 306. IAP ReadUID command	329
Table 271. A/D Data Registers (AD0DR0 to AD0DR7 - addresses 0x4001 C010 to 0x4001 C02C) bit description	302	Table 307. IAP Status Codes Summary	329
Table 272. LPC111x/LPC11Cx flash configurations.	303	Table 308. Memory mapping in debug mode	330
Table 273. Flash sector configuration	307	Table 309. Flash configuration register (FLASHCFG, address 0x4003 C010) bit description	330
Table 274. Code Read Protection options.	308	Table 310. Register overview: FMC (base address 0x4003 C000)	331
Table 275. Code Read Protection hardware/software interaction	309	Table 311. Flash Module Signature Start register (FMSSTART - 0x4003 C020) bit description	332
Table 276. ISP commands allowed for different CRP levels	309	Table 312. Flash Module Signature Stop register (FMSSTOP - 0x4003 C024) bit description	332
Table 277. UART ISP command summary	311	Table 313. FMSW0 register bit description (FMSW0, address: 0x4003 C02C)	332
Table 278. UART ISP Unlock command	312	Table 314. FMSW1 register bit description (FMSW1, address: 0x4003 C030)	332
Table 279. UART ISP Set Baud Rate command	312	Table 315. FMSW2 register bit description (FMSW2, address: 0x4003 C034)	332
Table 280. UART ISP Echo command	312	Table 316. FMSW3 register bit description (FMSW3, address: 0x4003 40C8)	332
Table 281. UART ISP Write to RAM command	313	Table 317. Flash module Status register (FMSTAT - 0x4003 CFE0) bit description	333
Table 282. UART ISP Read Memory command	313	Table 318. Flash Module Status Clear register (FMSTATCLR - 0x0x4003 CFE8) bit description.	333
Table 283. UART ISP Prepare sector(s) for write operation command	314	Table 319. Serial Wire Debug pin description.	335
Table 284. UART ISP Copy command	315	Table 320. Summary of processor mode and stack use options	339
Table 285. UART ISP Go command	315	Table 321. Core register set summary	340
Table 286. UART ISP Erase sector command	316	Table 322. PSR register combinations	341
Table 287. UART ISP Blank check sector command	316	Table 323. APSR bit assignments	342
Table 288. UART ISP Read Part Identification command	316	Table 324. IPSR bit assignments	342
Table 289. LPC111x and LPC11Cx part identification numbers	317	Table 325. EPSR bit assignments	343
Table 290. UART ISP Read Boot Code version number command	317	Table 326. PRIMASK register bit assignments	343
		Table 327. CONTROL register bit assignments	344
		Table 328. Memory access behavior	348
		Table 329. Properties of different exception types	350
		Table 330. Exception return behavior	355
		Table 331. Cortex-M0 instructions	358
		Table 332. CMSIS intrinsic functions to generate some Cortex-M0 instructions	359
		Table 333. insic functions to access the special	

registers	360
Table 334. Condition code suffixes	365
Table 335. Access instructions	365
Table 336. Data processing instructions	371
Table 337. ADC, ADD, RSB, SBC and SUB operand restrictions	373
Table 338. Branch and control instructions	380
Table 339. Branch ranges	381
Table 340. Miscellaneous instructions	382
Table 341. Core peripheral register regions	389
Table 342. NVIC register summary	390
Table 343. CMSIS access NVIC functions	390
Table 344. ISER bit assignments	391
Table 345. ICER bit assignments	391
Table 346. ISPR bit assignments	391
Table 347. ICPR bit assignments	392
Table 348. IPR bit assignments	392
Table 349. CMSIS functions for NVIC control	394
Table 350. Summary of the SCB registers	395
Table 351. CUID register bit assignments	395
Table 352. ICSR bit assignments	396
Table 353. AIRCR bit assignments	398
Table 354. SCR bit assignments	398
Table 355. CCR bit assignments	399
Table 356. System fault handler priority fields	399
Table 357. SHPR2 register bit assignments	400
Table 358. SHPR3 register bit assignments	400
Table 359. System timer registers summary	400
Table 360. SYST_CSR bit assignments	401
Table 361. SYST_RVR bit assignments	401
Table 362. SYST_CVR bit assignments	402
Table 363. SYST_CALIB register bit assignments	402
Table 364. Cortex M0- instruction summary	402
Table 365. Abbreviations	406

Fig 1.	LPC111x/LPC11Cxx block diagram	8	Fig 46.	C_CAN block diagram.	203
Fig 2.	LPC111x/LPC11Cxx memory map.	11	Fig 47.	CAN core in Silent mode.	226
Fig 3.	LPC111x/LPC11Cxx CGU block diagram	14	Fig 48.	CAN core in Loop-back mode.	227
Fig 4.	Start-up timing	35	Fig 49.	CAN core in Loop-back mode combined with Silent mode	227
Fig 5.	System PLL block diagram	41	Fig 50.	Block diagram of a message object transfer	229
Fig 6.	Power profiles pointer structure	48	Fig 51.	Reading a message from the FIFO buffer to the message buffer	235
Fig 7.	LPC111x/102/202/302 clock configuration for power API use	49	Fig 52.	Bit timing	238
Fig 8.	Power profiles usage	53	Fig 53.	CAN API pointer structure.	240
Fig 9.	Standard I/O pin configuration	59	Fig 54.	Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.	264
Fig 10.	Pin configuration LQFP48 package	92	Fig 55.	A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled	264
Fig 11.	Pin configuration PLCC44 package	93	Fig 56.	A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled	264
Fig 12.	Pin configuration HVQFN 33 package.	94	Fig 57.	16-bit counter/timer block diagram	265
Fig 13.	Pin configuration LQFP48 package	95	Fig 58.	Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.	278
Fig 14.	Pin configuration (LPC11C22/C24)	96	Fig 59.	A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled	278
Fig 15.	Masked write operation to the GPIODATA register.	115	Fig 60.	A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled	278
Fig 16.	Masked read operation	116	Fig 61.	32-bit counter/timer block diagram	279
Fig 17.	Auto-RTS Functional Timing	128	Fig 62.	Windowed Watchdog Timer (WWDt) block diagram	282
Fig 18.	Auto-CTS Functional Timing	129	Fig 63.	Early Watchdog Feed with Windowed Mode Enabled.	286
Fig 19.	Auto-baud a) mode 0 and b) mode 1 waveform	134	Fig 64.	Correct Watchdog Feed with Windowed Mode Enabled.	287
Fig 20.	Algorithm for setting UART dividers.	137	Fig 65.	Watchdog Warning Interrupt	287
Fig 21.	UART block diagram	143	Fig 66.	Watchdog block diagram.	292
Fig 22.	Texas Instruments Synchronous Serial Frame Format: a) Single and b) Continuous/back-to-back Two Frames Transfer.	152	Fig 67.	System tick timer block diagram	293
Fig 23.	SPI frame format with CPOL=0 and CPHA=0 (a) Single and b) Continuous Transfer).	153	Fig 68.	Boot process flowchart	306
Fig 24.	SPI frame format with CPOL=0 and CPHA=1	154	Fig 69.	IAP parameter passing	325
Fig 25.	SPI frame format with CPOL = 1 and CPHA = 0 (a) Single and b) Continuous Transfer).	155	Fig 70.	Algorithm for generating a 128-bit signature	334
Fig 26.	SPI Frame Format with CPOL = 1 and CPHA = 1.	156	Fig 71.	Connecting the SWD pins to a standard SWD connector	336
Fig 27.	Microwire frame format (single transfer)	157	Fig 72.	Cortex-M0 implementation	337
Fig 28.	Microwire frame format (continuous transfers)	157	Fig 73.	Processor core register set	340
Fig 29.	Microwire frame format setup and hold details	158	Fig 74.	APSR, IPSR, EPSR register bit assignments	341
Fig 30.	I ² C-bus configuration	160	Fig 75.	Generic ARM Cortex-M0 memory map.	346
Fig 31.	Format in the Master Transmitter mode.	170	Fig 76.	Memory ordering restrictions.	347
Fig 32.	Format of Master Receiver mode	171	Fig 77.	Little-endian format	349
Fig 33.	A Master Receiver switches to Master Transmitter after sending Repeated START.	171	Fig 78.	Vector table	352
Fig 34.	Format of Slave Receiver mode	172	Fig 79.	Exception entry stack contents	354
Fig 35.	Format of Slave Transmitter mode	172	Fig 80.	ASR #3	361
Fig 36.	I ² C serial interface block diagram	173	Fig 81.	LSR #3	362
Fig 37.	Arbitration procedure	175	Fig 82.	LSL #3.	362
Fig 38.	Serial clock synchronization.	175	Fig 83.	ROR #3	363
Fig 39.	Format and states in the Master Transmitter mode	180	Fig 84.	IPR register	392
Fig 40.	Format and states in the Master Receiver mode	183			
Fig 41.	Format and states in the Slave Receiver mode.	187			
Fig 42.	Format and states in the Slave Transmitter mode	190			
Fig 43.	Simultaneous Repeated START conditions from two masters	192			
Fig 44.	Forced access to a busy I ² C-bus	192			
Fig 45.	Recovering from a bus obstruction caused by a LOW level on SDA.	193			

Chapter 1: LPC111x/LPC11Cxx Introductory information

1.1	Introduction	3	1.4	Block diagram	8
1.2	Features	4	1.5	ARM Cortex-M0 processor	9
1.3	Ordering information	5			

Chapter 2: LPC111x/LPC11Cxx Memory mapping

2.1	How to read this chapter	10	2.2	Memory map	10
------------	---------------------------------------	-----------	------------	-------------------------	-----------

Chapter 3: LPC111x/LPC11Cxx System configuration (SYSCON)

3.1	How to read this chapter	12	3.5.32	Deep-sleep mode configuration register	30
	C_CAN controller	12	3.5.33	Wake-up configuration register	31
	Entering Deep power-down mode	12	3.5.34	Power-down configuration register	32
	Enabling sequence for UART clock	12	3.5.35	Device ID register	33
3.2	General description	12	3.6	Reset	34
3.3	Pin description	12	3.7	Start-up behavior	35
3.4	Clock generation	13	3.8	Brown-out detection	35
3.5	Register description	14	3.9	Power management	36
3.5.1	System memory remap register	16	3.9.1	Active mode	36
3.5.2	Peripheral reset control register	16	3.9.1.1	Power configuration in Active mode	36
3.5.3	System PLL control register	17	3.9.2	Sleep mode	36
3.5.4	System PLL status register	17	3.9.2.1	Power configuration in Sleep mode	36
3.5.5	System oscillator control register	18	3.9.2.2	Programming Sleep mode	37
3.5.6	Watchdog oscillator control register	18	3.9.2.3	Wake-up from Sleep mode	37
3.5.7	Internal resonant crystal control register	19	3.9.3	Deep-sleep mode	37
3.5.8	System reset status register	20	3.9.3.1	Power configuration in Deep-sleep mode	37
3.5.9	System PLL clock source select register	20	3.9.3.2	Programming Deep-sleep mode	37
3.5.10	System PLL clock source update enable register	21	3.9.3.3	Wake-up from Deep-sleep mode	38
3.5.11	Main clock source select register	21	3.9.4	Deep power-down mode	38
3.5.12	Main clock source update enable register	21	3.9.4.1	Power configuration in Deep power-down mode	39
3.5.13	System AHB clock divider register	22	3.9.4.2	Programming Deep power-down mode	39
3.5.14	System AHB clock control register	22	3.9.4.3	Wake-up from Deep power-down mode	39
3.5.15	SPI0 clock divider register	24	3.10	Deep-sleep mode details	40
3.5.16	UART clock divider register	24	3.10.1	IRC oscillator	40
3.5.17	SPI1 clock divider register	24	3.10.2	Start logic	40
3.5.18	WDT clock source select register	25	3.10.3	Using the general purpose counter/timers to create a self-wake-up event	40
3.5.19	WDT clock source update enable register	25	3.11	System PLL functional description	41
3.5.20	WDT clock divider register	25	3.11.1	Lock detector	41
3.5.21	CLKOUT clock source select register	26	3.11.2	Power-down control	42
3.5.22	CLKOUT clock source update enable register	26	3.11.3	Divider ratio programming	42
3.5.23	CLKOUT clock divider register	27		Post divider	42
3.5.24	POR captured PIO status register 0	27		Feedback divider	42
3.5.25	POR captured PIO status register 1	27		Changing the divider values	42
3.5.26	BOD control register	28	3.11.4	Frequency selection	42
3.5.27	System tick counter calibration register	28	3.11.4.1	Normal mode	43
3.5.28	Start logic edge control register 0	28	3.11.4.2	Power-down mode	43
3.5.29	Start logic signal enable register 0	29	3.12	Flash memory access	43
3.5.30	Start logic reset register 0	29			
3.5.31	Start logic status register 0	30			

Chapter 4: LPC111x/LPC11Cxx Power Monitor Unit (PMU)

4.1	How to read this chapter	45	4.3.1	Power control register	45
4.2	Introduction	45	4.3.2	General purpose registers 0 to 3	46
4.3	Register description	45	4.3.3	General purpose register 4	46

4.4 Functional description 47**Chapter 5: LPC111x/LPC11Cxx Power profiles**

5.1	How to read this chapter	48	5.5.1.4.4	System clock less than or equal to the expected value	52
5.2	Features	48	5.5.1.4.5	System clock greater than or equal to the expected value	52
5.3	Description	48	5.5.1.4.6	System clock approximately equal to the expected value	52
5.4	Definitions	49	5.6	Power routine	53
5.5	Clocking routine	49	5.6.1	set_power	53
5.5.1	set_pll	49	5.6.1.1	Param0: main clock	54
5.5.1.1	Param0: system PLL input frequency and Param1: expected system clock	50	5.6.1.2	Param1: mode	54
5.5.1.2	Param2: mode	50	5.6.1.3	Param2: system clock	54
5.5.1.3	Param3: system PLL lock time-out	51	5.6.1.4	Code examples	55
5.5.1.4	Code examples	51	5.6.1.4.1	Invalid frequency (device maximum clock rate exceeded)	55
5.5.1.4.1	Invalid frequency (device maximum clock rate exceeded)	51	5.6.1.4.2	An applicable power setup	55
5.5.1.4.2	Invalid frequency selection (system clock divider restrictions)	51			
5.5.1.4.3	Exact solution cannot be found (PLL)	52			

Chapter 6: LPC111x/LPC11Cxx Nested Vectored Interrupt Controller (NVIC)

6.1	How to read this chapter	56	6.3	Features	56
6.2	Introduction	56	6.4	Interrupt sources	56

Chapter 7: LPC111x/LPC11Cxx I/O configuration (IOCONFIG)

7.1	How to read this chapter	58	7.4.18	IOCON_PIO0_6	73
	C_CAN pins	58	7.4.19	IOCON_PIO0_7	74
	Pseudo open-drain function	58	7.4.20	IOCON_PIO2_9	74
	Pull-up level	58	7.4.21	IOCON_PIO2_10	75
7.2	Features	58	7.4.22	IOCON_PIO2_2	76
7.3	General description	59	7.4.23	IOCON_PIO0_8	76
7.3.1	Pin function	59	7.4.24	IOCON_PIO0_9	77
7.3.2	Pin mode	59	7.4.25	IOCON_SWCLK_PIO0_10	77
7.3.3	Hysteresis	60	7.4.26	IOCON_PIO1_10	78
7.3.4	A/D-mode	60	7.4.27	IOCON_PIO2_11	79
7.3.5	I ² C mode	60	7.4.28	IOCON_R_PIO0_11	79
7.3.6	Open-drain Mode	60	7.4.29	IOCON_R_PIO1_0	80
7.4	Register description	60	7.4.30	IOCON_R_PIO1_1	81
7.4.1	IOCON_PIO2_6	64	7.4.31	IOCON_R_PIO1_2	82
7.4.2	IOCON_PIO2_0	65	7.4.32	IOCON_PIO3_0	82
7.4.3	IOCON_PIO_RESET_PIO0_0	65	7.4.33	IOCON_PIO3_1	83
7.4.4	IOCON_PIO0_1	66	7.4.34	IOCON_PIO2_3	83
7.4.5	IOCON_PIO1_8	66	7.4.35	IOCON_SWDIO_PIO1_3	84
7.4.6	IOCON_PIO0_2	67	7.4.36	IOCON_PIO1_4	85
7.4.7	IOCON_PIO2_7	68	7.4.37	IOCON_PIO1_11	86
7.4.8	IOCON_PIO2_8	68	7.4.38	IOCON_PIO3_2	86
7.4.9	IOCON_PIO2_1	69	7.4.39	IOCON_PIO1_5	87
7.4.10	IOCON_PIO0_3	69	7.4.40	IOCON_PIO1_6	87
7.4.11	IOCON_PIO0_4	70	7.4.41	IOCON_PIO1_7	88
7.4.12	IOCON_PIO0_5	70	7.4.42	IOCON_PIO3_3	89
7.4.13	IOCON_PIO1_9	70	7.4.43	IOCON_SCK_LOC	89
7.4.14	IOCON_PIO3_4	71	7.4.44	IOCON_DSR_LOC	90
7.4.15	IOCON_PIO2_4	72	7.4.45	IOCON_DCD_LOC	90
7.4.16	IOCON_PIO2_5	72	7.4.46	IOCON_RI_LOC	90
7.4.17	IOCON_PIO3_5	73			

Chapter 8: LPC111x/LPC11Cxx Pin configuration

8.1	How to read this chapter	91	8.3	LPC11Cxx Pin configuration	95
8.2	LPC111x Pin configuration	92	8.4	LPC111x/LPC11Cxx Pin description	96

Chapter 9: LPC111x/LPC11Cxx General Purpose I/O (GPIO)

9.1	How to read this chapter	110	9.3.6	GPIO interrupt mask register	113
9.2	Introduction	110	9.3.7	GPIO raw interrupt status register	113
9.2.1	Features	110	9.3.8	GPIO masked interrupt status register	113
9.3	Register description	110	9.3.9	GPIO interrupt clear register	114
9.3.1	GPIO data register	111	9.4	Functional description	114
9.3.2	GPIO data direction register	112	9.4.1	Write/read data operation	114
9.3.3	GPIO interrupt sense register	112		Write operation	114
9.3.4	GPIO interrupt both edges sense register	112		Read operation	116
9.3.5	GPIO interrupt event register	113			

Chapter 10: LPC111x/LPC11Cxx UART

10.1	How to read this chapter	117	10.5.12	UART Auto-baud Control Register (U0ACR - 0x4000 8020)	132
10.2	Basic configuration	117	10.5.13	Auto-baud	132
10.3	Features	117	10.5.14	Auto-baud modes	133
10.4	Pin description	118	10.5.15	UART Fractional Divider Register (U0FDR - 0x4000 8028)	135
10.5	Register description	118	10.5.15.1	Baud rate calculation	136
10.5.1	UART Receiver Buffer Register (U0RBR - 0x4000 8000, when DLAB = 0, Read Only)	120	10.5.15.1.1	Example 1: UART_PCLK = 14.7456 MHz, BR = 9600	138
10.5.2	UART Transmitter Holding Register (U0THR - 0x4000 8000 when DLAB = 0, Write Only)	120	10.5.15.1.2	Example 2: UART_PCLK = 12 MHz, BR = 115200	138
10.5.3	UART Divisor Latch LSB and MSB Registers (U0DLL - 0x4000 8000 and U0DLM - 0x4000 8004, when DLAB = 1)	120	10.5.16	UART Transmit Enable Register (U0TER - 0x4000 8030)	138
10.5.4	UART Interrupt Enable Register (U0IER - 0x4000 8004, when DLAB = 0)	121	10.5.17	UART RS485 Control register (U0RS485CTRL - 0x4000 804C)	139
10.5.5	UART Interrupt Identification Register (U0IIR - 0x4004 8008, Read Only)	122	10.5.18	UART RS485 Address Match register (U0RS485ADRMATCH - 0x4000 8050)	140
10.5.6	UART FIFO Control Register (U0FCR - 0x4000 8008, Write Only)	124	10.5.19	UART1 RS485 Delay value register (U0RS485DLY - 0x4000 8054)	140
10.5.7	UART Line Control Register (U0LCR - 0x4000 800C)	125	10.5.20	RS-485/EIA-485 modes of operation	140
10.5.8	UART Modem Control Register	126		RS-485/EIA-485 Normal Multidrop Mode (NMM)	141
10.5.8.1	Auto-flow control	127		RS-485/EIA-485 Auto Address Detection (AAD) mode	141
10.5.8.1.1	Auto-RTS	127		RS-485/EIA-485 Auto Direction Control	141
10.5.8.1.2	Auto-CTS	128		RS485/EIA-485 driver delay time	142
10.5.9	UART Line Status Register (U0LSR - 0x4000 8014, Read Only)	129		RS485/EIA-485 output inversion	142
10.5.10	UART Modem Status Register	131	10.6	Architecture	142
10.5.11	UART Scratch Pad Register (U0SCR - 0x4000 801C)	131			

Chapter 11: LPC111x/LPC11Cxx SPI0/1 with SSP

11.1	How to read this chapter	144	11.6.1	SPI/SSP Control Register 0	146
11.2	Basic configuration	144	11.6.2	SPI/SSP0 Control Register 1	147
11.3	Features	144	11.6.3	SPI/SSP Data Register	148
11.4	General description	144	11.6.4	SPI/SSP Status Register	149
11.5	Pin description	145	11.6.5	SPI/SSP Clock Prescale Register	149
11.6	Register description	145	11.6.6	SPI/SSP Interrupt Mask Set/Clear Register	149
			11.6.7	SPI/SSP Raw Interrupt Status Register	150

11.6.8	SPI/SSP Masked Interrupt Status Register .	150	11.7.2.2	SPI format with CPOL=0,CPHA=0.	153
11.6.9	SPI/SSP Interrupt Clear Register	151	11.7.2.3	SPI format with CPOL=0,CPHA=1.	154
11.7	Functional description	151	11.7.2.4	SPI format with CPOL = 1,CPHA = 0.	154
11.7.1	Texas Instruments synchronous serial frame format	151	11.7.2.5	SPI format with CPOL = 1,CPHA = 1.	156
11.7.2	SPI frame format	152	11.7.3	Semiconductor Microwire frame format . . .	156
11.7.2.1	Clock Polarity (CPOL) and Phase (CPHA) control.	152	11.7.3.1	Setup and hold time requirements on CS with respect to SK in Microwire mode	158

Chapter 12: LPC111x/LPC11Cxx I2C-bus controller

12.1	How to read this chapter	159	12.9.9	Control register, CONSET and CONCLR . .	176
12.2	Basic configuration	159	12.9.10	Status decoder and status register.	176
12.3	Features	159	12.10	Details of I²C operating modes	176
12.4	Applications	159	12.10.1	Master Transmitter mode	177
12.5	General description	159	12.10.2	Master Receiver mode	181
12.5.1	I ² C Fast-mode Plus	160	12.10.3	Slave Receiver mode	184
12.6	Pin description	161	12.10.4	Slave Transmitter mode	188
12.7	Register description	161	12.10.5	Miscellaneous states	190
12.7.1	I ² C Control Set register (I2C0CONSET - 0x4000 0000)	162	12.10.5.1	STAT = 0xF8	190
12.7.2	I ² C Status register (I2C0STAT - 0x4000 0004)	164	12.10.5.2	STAT = 0x00	190
12.7.3	I ² C Data register (I2C0DAT - 0x4000 0008)	164	12.10.6	Some special cases	191
12.7.4	I ² C Slave Address register 0 (I2C0ADRO - 0x4000 000C)	164	12.10.6.1	Simultaneous Repeated START conditions from two masters	191
12.7.5	I ² C SCL HIGH and LOW duty cycle registers (I2C0SCLH - 0x4000 0010 and I2C0SCLL - 0x4000 0014)	165	12.10.6.2	Data transfer after loss of arbitration	192
12.7.5.1	Selecting the appropriate I ² C data rate and duty cycle	165	12.10.6.3	Forced access to the I ² C-bus.	192
12.7.6	I ² C Control Clear register (I2C0CONCLR - 0x4000 0018)	166	12.10.6.4	I ² C-bus obstructed by a LOW level on SCL or SDA	193
12.7.7	I ² C Monitor mode control register (I2C0MMCTRL - 0x4000 001C)	166	12.10.6.5	Bus error	193
12.7.7.1	Interrupt in Monitor mode	167	12.10.7	I ² C state service routines	193
12.7.7.2	Loss of arbitration in Monitor mode	168	12.10.8	Initialization	194
12.7.8	I ² C Slave Address registers (I2C0ADR[1, 2, 3] - 0x4000 00[20, 24, 28])	168	12.10.9	I ² C interrupt service	194
12.7.9	I ² C Data buffer register (I2C0DATA_BUFFER - 0x4000 002C)	168	12.10.10	The state service routines	194
12.7.10	I ² C Mask registers (I2C0MASK[0, 1, 2, 3] - 0x4000 00[30, 34, 38, 3C])	169	12.10.11	Adapting state services to an application. . .	194
12.8	I²C operating modes	169	12.11	Software example	194
12.8.1	Master Transmitter mode	169	12.11.1	Initialization routine	194
12.8.2	Master Receiver mode	170	12.11.2	Start Master Transmit function	194
12.8.3	Slave Receiver mode	171	12.11.3	Start Master Receive function	195
12.8.4	Slave Transmitter mode	172	12.11.4	I ² C interrupt routine	195
12.9	I²C implementation and operation	172	12.11.5	Non mode specific states	195
12.9.1	Input filters and output stages.	173	12.11.5.1	State: 0x00	195
12.9.2	Address Registers, ADDR0 to ADDR3.	174	12.11.5.2	Master States	195
12.9.3	Address mask registers, MASK0 to MASK3.	174	12.11.5.3	State: 0x08	195
12.9.4	Comparator.	174	12.11.5.4	State: 0x10	196
12.9.5	Shift register, DAT.	174	12.11.6	Master Transmitter states	196
12.9.6	Arbitration and synchronization logic	174	12.11.6.1	State: 0x18	196
12.9.7	Serial clock generator.	175	12.11.6.2	State: 0x20	196
12.9.8	Timing and control	176	12.11.6.3	State: 0x28	196
			12.11.6.4	State: 0x30	197
			12.11.6.5	State: 0x38	197
			12.11.7	Master Receive states	197
			12.11.7.1	State: 0x40	197
			12.11.7.2	State: 0x48	197
			12.11.7.3	State: 0x50	197
			12.11.7.4	State: 0x58	198
			12.11.8	Slave Receiver states	198
			12.11.8.1	State: 0x60	198
			12.11.8.2	State: 0x68	198

12.11.8.3	State: 0x70	198	12.11.9	Slave Transmitter states	200
12.11.8.4	State: 0x78	199	12.11.9.1	State: 0xA8	200
12.11.8.5	State: 0x80	199	12.11.9.2	State: 0xB0	200
12.11.8.6	State: 0x88	199	12.11.9.3	State: 0xB8	200
12.11.8.7	State: 0x90	199	12.11.9.4	State: 0xC0	201
12.11.8.8	State: 0x98	200	12.11.9.5	State: 0xC8	201
12.11.8.9	State: 0xA0	200			

Chapter 13: LPC111x/LPC11Cxx C_CAN controller

13.1	How to read this chapter	202	13.6.3.3	CAN new data 1 register	221
13.2	Basic configuration	202	13.6.3.4	CAN new data 2 register	222
13.3	Features	202	13.6.3.5	CAN interrupt pending 1 register	222
13.4	General description	203	13.6.3.6	CAN interrupt pending 2 register	223
13.5	Pin description	204	13.6.3.7	CAN message valid 1 register	223
13.6	Register description	204	13.6.3.8	CAN message valid 2 register	223
13.6.1	CAN protocol registers	206	13.6.4	CAN timing register	224
13.6.1.1	CAN control register	206	13.6.4.1	CAN clock divider register	224
13.6.1.2	CAN status register	207	13.7	Functional description	224
13.6.1.3	CAN error counter	209	13.7.1	C_CAN controller state after reset	224
13.6.1.4	CAN bit timing register	210	13.7.2	C_CAN operating modes	224
	Baud rate prescaler	210	13.7.2.1	Software initialization	224
	Time segments 1 and 2	210	13.7.2.2	CAN message transfer	225
	Synchronization jump width	210	13.7.2.3	Disabled Automatic Retransmission (DAR)	225
13.6.1.5	CAN interrupt register	211	13.7.2.4	Test modes	226
13.6.1.6	CAN test register	211	13.7.2.4.1	Silent mode	226
13.6.1.7	CAN baud rate prescaler extension register	212	13.7.2.4.2	Loop-back mode	226
13.6.2	Message interface registers	212	13.7.2.4.3	Loop-back mode combined with Silent mode	227
13.6.2.1	Message objects	213	13.7.2.4.4	Basic mode	227
13.6.2.2	CAN message interface command request registers	213	13.7.2.4.5	Software control of pin CAN_TXD	228
13.6.2.3	CAN message interface command mask registers	214	13.7.3	CAN message handler	228
13.6.2.4	IF1 and IF2 message buffer registers	216	13.7.3.1	Management of message objects	229
13.6.2.4.1	CAN message interface command mask 1 registers	217	13.7.3.2	Data Transfer between IFx Registers and the Message RAM	230
13.6.2.4.2	CAN message interface command mask 2 registers	217	13.7.3.3	Transmission of messages between the shift registers in the CAN core and the Message buffer	230
13.6.2.4.3	CAN message interface command arbitration 1 registers	217	13.7.3.4	Acceptance filtering of received messages	231
13.6.2.4.4	CAN message interface command arbitration 2 registers	218	13.7.3.4.1	Reception of a data frame	231
13.6.2.4.5	CAN message interface message control registers	218	13.7.3.4.2	Reception of a remote frame	231
13.6.2.4.6	CAN message interface data A1 registers	219	13.7.3.5	Receive/transmit priority	232
13.6.2.4.7	CAN message interface data A2 registers	220	13.7.3.6	Configuration of a transmit object	232
13.6.2.4.8	CAN message interface data B1 registers	220	13.7.3.7	Updating a transmit object	232
13.6.2.4.9	CAN message interface data B2 registers	220	13.7.3.8	Configuration of a receive object	233
13.6.3	Message handler registers	220	13.7.3.9	Handling of received messages	233
13.6.3.1	CAN transmission request 1 register	221	13.7.3.10	Configuration of a FIFO buffer	234
13.6.3.2	CAN transmission request 2 register	221	13.7.3.10.1	Reception of messages with FIFO buffers	234
			13.7.3.10.2	Reading from a FIFO buffer	234
			13.7.4	Interrupt handling	235
			13.7.5	Bit timing	236
			13.7.5.1	Bit time and bit rate	237

Chapter 14: LPC11Cxx C_CAN on-chip drivers

14.1	How to read this chapter	239	14.4	API description	240
14.2	Features	239	14.4.1	Calling the C_CAN API	240
14.3	General description	239	14.4.2	CAN initialization	241
14.3.1	Differences to fully-compliant CANopen	239	14.4.3	CAN interrupt handler	241

14.4.4	CAN Rx message object configuration	241	14.4.11	CAN message transmit callback	246
14.4.5	CAN receive	242	14.4.12	CAN error callback	246
14.4.6	CAN transmit	242	14.4.13	CANopen SDO expedited read callback	247
14.4.7	CANopen configuration	243	14.4.14	CANopen SDO expedited write callback	247
14.4.8	CANopen handler	244	14.4.15	CANopen SDO segmented read callback	248
14.4.9	CAN/CANopen callback functions	245	14.4.16	CANopen SDO segmented write callback	249
14.4.10	CAN message received callback	245	14.4.17	CANopen fall-back SDO handler callback	250

Chapter 15: LPC111x/LPC11Cxx 16-bit counter/timer CT16B0/1

15.1	How to read this chapter	252	15.7.6	Match Control Register (TMR16B0MCR and TMR16B1MCR)	257
15.2	Basic configuration	252	15.7.7	Match Registers (TMR16B0MR0/1/2/3 - addresses 0x4000 C018/1C/20/24 and TMR16B1MR0/1/2/3 - addresses 0x4001 0018/1C/20/24)	258
15.3	Features	252	15.7.8	Capture Control Register (TMR16B0CCR and TMR16B1CCR)	259
15.4	Applications	252	15.7.9	Capture Register (CT16B0CR0 - address 0x4000 C02C and CT16B1CR0 - address 0x4001 002C)	259
15.5	Description	253	15.7.10	External Match Register (TMR16B0EMR and TMR16B1EMR)	260
15.6	Pin description	253	15.7.11	Count Control Register (TMR16B0CTCR and TMR16B1CTCR)	261
15.7	Register description	253	15.7.12	PWM Control Register (TMR16B0PWM and TMR16B1PWM)	262
15.7.1	Interrupt Register (TMR16B0IR and TMR16B1IR)	255	15.7.13	Rules for single edge controlled PWM outputs	263
15.7.2	Timer Control Register (TMR16B0TCR and TMR16B1TCR)	256	15.8	Example timer operation	264
15.7.3	Timer Counter (TMR16B0TC - address 0x4000 C008 and TMR16B1TC - address 0x4001 0008)	256	15.9	Architecture	265
15.7.4	Prescale Register (TMR16B0PR - address 0x4000 C00C and TMR16B1PR - address 0x4001 000C)	256			
15.7.5	Prescale Counter register (TMR16B0PC - address 0x4000 C010 and TMR16B1PC - address 0x4001 0010)	257			

Chapter 16: LPC111x/LPC11Cxx 32-bit counter/timer CT32B0/1

16.1	How to read this chapter	266	16.7.6	Match Control Register (TMR32B0MCR and TMR32B1MCR)	271
16.2	Basic configuration	266	16.7.7	Match Registers (TMR32B0MR0/1/2/3 - addresses 0x4001 4018/1C/20/24 and TMR32B1MR0/1/2/3 addresses 0x4001 8018/1C/20/24)	272
16.3	Features	266	16.7.8	Capture Control Register (TMR32B0CCR and TMR32B1CCR)	272
16.4	Applications	266	16.7.9	Capture Register (TMR32B0CR0 - address 0x4001 402C and TMR32B1CR0 - address 0x4001 802C)	273
16.5	Description	267	16.7.10	External Match Register (TMR32B0EMR and TMR32B1EMR)	273
16.6	Pin description	267	16.7.11	Count Control Register (TMR32B0CTCR and TMR32B1CTCR)	275
16.7	Register description	267	16.7.12	PWM Control Register (TMR32B0PWM and TMR32B1PWM)	276
16.7.1	Interrupt Register (TMR32B0IR and TMR32B1IR)	269	16.7.13	Rules for single edge controlled PWM outputs	277
16.7.2	Timer Control Register (TMR32B0TCR and TMR32B1TCR)	269	16.8	Example timer operation	278
16.7.3	Timer Counter (TMR32B0TC - address 0x4001 4008 and TMR32B1TC - address 0x4001 8008)	270	16.9	Architecture	279
16.7.4	Prescale Register (TMR32B0PR - address 0x4001 400C and TMR32B1PR - address 0x4001 800C)	270			
16.7.5	Prescale Counter Register (TMR32B0PC - address 0x4001 4010 and TMR32B1PC - address 0x4001 8010)	270			

Chapter 17: LPC111x/LPC11Cxx Windowed WatchDog Timer (WDT)

17.1	How to read this chapter	280	17.2	Basic configuration	280
------	------------------------------------	-----	------	-------------------------------	-----

17.3	Features	280	17.7.2	Watchdog Timer Constant register	284
17.4	Applications	281	17.7.3	Watchdog Feed register	285
17.5	General description	281	17.7.4	Watchdog Timer Value register	285
17.6	Clock control	282	17.7.5	Watchdog Timer Warning Interrupt register	285
17.7	Register description	283	17.7.6	Watchdog Timer Window register	286
17.7.1	Watchdog Mode register	283	17.7.7	Watchdog timing examples	286

Chapter 18: LPC111x/LPC11Cxx WatchDog Timer (WDT)

18.1	How to read this chapter	288	18.7.1	Watchdog Mode register (WDMOD - 0x4000 0000)	290
18.2	Basic configuration	288	18.7.2	Watchdog Timer Constant register (WDTC - 0x4000 4004)	291
18.3	Features	288	18.7.3	Watchdog Feed register (WDFEED - 0x4000 4008)	291
18.4	Applications	289	18.7.4	Watchdog Timer Value register (WDTV - 0x4000 400C)	292
18.5	Description	289	18.8	Block diagram	292
18.6	WDT clocking	289			
18.7	Register description	290			

Chapter 19: LPC111x/LPC11Cxx System tick timer (SysTick)

19.1	How to read this chapter	293	19.5.3	System Timer Current value register	295
19.2	Basic configuration	293	19.5.4	System Timer Calibration value register (SYST_CALIB - 0xE000 E01C)	296
19.3	Features	293	19.6	Functional description	296
19.4	General description	293	19.7	Example timer calculations	296
19.5	Register description	294		Example (system clock = 50 MHz)	296
19.5.1	System Timer Control and status register ..	294			
19.5.2	System Timer Reload value register	295			

Chapter 20: LPC111x/LPC11Cxx ADC

20.1	How to read this chapter	297	20.5.3	A/D Status Register (AD0STAT - 0x4001 C030)	301
20.2	Basic configuration	297	20.5.4	A/D Interrupt Enable Register (AD0INTEN - 0x4001 C00C)	301
20.3	Features	297	20.5.5	A/D Data Registers (AD0DR0 to AD0DR7 - 0x4001 C010 to 0x4001 C02C)	301
20.4	Pin description	297	20.6	Operation	302
20.5	Register description	298	20.6.1	Hardware-triggered conversion	302
20.5.1	A/D Control Register (AD0CR - 0x4001 C000)	298	20.6.2	Interrupts	302
20.5.2	A/D Global Data Register (AD0GDR - 0x4001 C004)	300	20.6.3	Accuracy vs. digital receiver	302

Chapter 21: LPC111x/LPC11Cxx Flash programming firmware

21.1	How to read this chapter	303	21.4.2	UART ISP response format	310
21.2	Features	303	21.4.3	UART ISP data format	310
21.3	General description	303	21.4.4	UART ISP flow control	310
21.3.1	Bootloader	303	21.4.5	UART SP command abort	310
21.3.2	Memory map after any reset	304	21.4.6	Interrupts during UART ISP	310
21.3.3	Criterion for Valid User Code	304	21.4.7	Interrupts during IAP	311
21.3.4	Boot process flowchart	306	21.4.8	RAM used by ISP command handler	311
21.3.5	Sector numbers	307	21.4.9	RAM used by IAP command handler	311
21.3.6	Flash content protection mechanism	307	21.5	UART ISP commands	311
21.3.7	Code Read Protection (CRP)	308	21.5.1	Unlock <Unlock code> (UART ISP)	312
21.3.7.1	ISP entry protection	310	21.5.2	Set Baud Rate <Baud Rate> <stop bit> (UART ISP)	312
21.4	UART Communication protocol	310	21.5.3	Echo <setting> (UART ISP)	312
21.4.1	UART ISP command format	310			

21.5.4	Write to RAM <start address> <number of bytes> (UART ISP)	312	21.6.13	Read serial number (C_CAN ISP)	322
21.5.5	Read Memory <address> <no. of bytes> (UART ISP)	313	21.6.14	Compare (C_CAN ISP)	322
21.5.6	Prepare sector(s) for write operation <start sector number> <end sector number> (UART ISP)	314	21.6.15	C_CAN ISP SDO abort codes	322
21.5.7	Copy RAM to flash <Flash address> <RAM address> <no of bytes> (UART ISP)	314	21.6.16	Differences to fully-compliant CANopen	323
21.5.8	Go <address> <mode> (UART ISP)	315	21.7 IAP commands	324	
21.5.9	Erase sector(s) <start sector number> <end sector number> (UART ISP)	316	21.7.1	Prepare sector(s) for write operation (IAP)	325
21.5.10	Blank check sector(s) <sector number> <end sector number> (UART ISP)	316	21.7.2	Copy RAM to flash (IAP)	326
21.5.11	Read Part Identification number (UART ISP)	316	21.7.3	Erase Sector(s) (IAP)	327
21.5.12	Read Boot code version number (UART ISP)	317	21.7.4	Blank check sector(s) (IAP)	327
21.5.13	Compare <address1> <address2> <no of bytes> (UART ISP)	318	21.7.5	Read Part Identification number (IAP)	327
21.5.14	ReadUID (UART ISP)	318	21.7.6	Read Boot code version number (IAP)	328
21.5.15	UART ISP Return Codes	318	21.7.7	Compare <address1> <address2> <no of bytes> (IAP)	328
21.6 C_CAN communication protocol	319		21.7.8	Reinvoke ISP (IAP)	328
21.6.1	C_CAN ISP SDO communication	320	21.7.9	ReadUID (IAP)	329
21.6.2	C_CAN ISP object directory	320	21.7.10	IAP Status Codes	329
21.6.3	Unlock (C_CAN ISP)	321	21.8 Debug notes	329	
21.6.4	Write to RAM (C_CAN ISP)	321	21.8.1	Comparing flash images	329
21.6.5	Read memory (C_CAN ISP)	321	21.8.2	Serial Wire Debug (SWD) flash programming interface	330
21.6.6	Prepare sectors for write operation (C_CAN ISP)	322	21.9 Flash memory access	330	
21.6.7	Copy RAM to flash (C_CAN ISP)	322	21.10 Flash signature generation	331	
21.6.8	Go (C_CAN ISP)	322	21.10.1	Register description for signature generation	331
21.6.9	Erase sectors (C_CAN ISP)	322	21.10.1.1	Signature generation address and control registers	331
21.6.10	Blank check sectors (C_CAN ISP)	322	21.10.1.2	Signature generation result registers	332
21.6.11	Read PartID (C_CAN ISP)	322	21.10.1.3	Flash Module Status register	333
21.6.12	Read boot code version (C_CAN ISP)	322	21.10.1.4	Flash Module Status Clear register	333
			21.10.2	Algorithm and procedure for signature generation	333
				Signature generation	333
				Content verification	334

Chapter 22: LPC111x/LPC11Cxx Serial Wire Debug (SWD)

22.1 How to read this chapter	335	22.5 Pin description	335
22.2 Features	335	22.6 Debug notes	336
22.3 Introduction	335	22.6.1	Debug limitations
22.4 Description	335	22.6.2	Debug connections

Chapter 23: LPC111x/LPC11Cxx Appendix: ARM Cortex-M0 reference

23.1 Introduction	337	23.3.1.3.4	Program Counter	341
23.2 About the Cortex-M0 processor and core peripherals	337	23.3.1.3.5	Program Status Register	341
23.2.1	System-level interface	23.3.1.3.6	Exception mask register	343
23.2.2	Integrated configurable debug	23.3.1.3.7	CONTROL register	343
23.2.3	Cortex-M0 processor features summary	23.3.1.4	Exceptions and interrupts	344
23.2.4	Cortex-M0 core peripherals	23.3.1.5	Data types	344
23.3 Processor	339	23.3.1.6	The Cortex Microcontroller Software Interface Standard	344
23.3.1	Programmers model	23.3.2	Memory model	345
23.3.1.1	Processor modes	23.3.2.1	Memory regions, types and attributes	346
23.3.1.2	Stacks	23.3.2.2	Memory system ordering of memory accesses	347
23.3.1.3	Core registers	23.3.2.3	Behavior of memory accesses	347
23.3.1.3.1	General-purpose registers	23.3.2.4	Software ordering of memory accesses	348
23.3.1.3.2	Stack Pointer	23.3.2.5	Memory endianness	349
23.3.1.3.3	Link Register			

23.3.2.5.1 Little-endian format	349	23.4.4.4.2 Operation	368
23.3.3 Exception model	349	23.4.4.4.3 Restrictions	368
23.3.3.1 Exception states	349	23.4.4.4.4 Condition flags	368
23.3.3.2 Exception types	350	23.4.4.4.5 Examples	369
23.3.3.3 Exception handlers	351	23.4.4.5 LDM and STM	369
23.3.3.4 Vector table	351	23.4.4.5.1 Syntax	369
23.3.3.5 Exception priorities	352	23.4.4.5.2 Operation	369
23.3.3.6 Exception entry and return	353	23.4.4.5.3 Restrictions	369
23.3.3.6.1 Exception entry	353	23.4.4.5.4 Condition flags	370
23.3.3.6.2 Exception return	354	23.4.4.5.5 Examples	370
23.3.4 Fault handling	355	23.4.4.5.6 Incorrect examples	370
23.3.4.1 Lockup	355	23.4.4.6 PUSH and POP	370
23.3.5 Power management	356	23.4.4.6.1 Syntax	370
23.3.5.1 Entering sleep mode	356	23.4.4.6.2 Operation	370
23.3.5.1.1 Wait for interrupt	356	23.4.4.6.3 Restrictions	370
23.3.5.1.2 Wait for event	356	23.4.4.6.4 Condition flags	371
23.3.5.1.3 Sleep-on-exit	357	23.4.4.6.5 Examples	371
23.3.5.2 Wake-up from sleep mode	357	23.4.5 General data processing instructions	371
23.3.5.2.1 Wake-up from WFI or sleep-on-exit	357	23.4.5.1 ADC, ADD, RSB, SBC, and SUB	372
23.3.5.2.2 Wake-up from WFE	357	23.4.5.1.1 Syntax	372
23.3.5.3 Power management programming hints	357	23.4.5.1.2 Operation	372
23.4 Instruction set	357	23.4.5.1.3 Restrictions	373
23.4.1 Instruction set summary	357	23.4.5.1.4 Examples	373
23.4.2 Intrinsic functions	359	23.4.5.2 AND, ORR, EOR, and BIC	373
23.4.3 About the instruction descriptions	360	23.4.5.2.1 Syntax	374
23.4.3.1 Operands	360	23.4.5.2.2 Operation	374
23.4.3.2 Restrictions when using PC or SP	360	23.4.5.2.3 Restrictions	374
23.4.3.3 Shift Operations	361	23.4.5.2.4 Condition flags	374
23.4.3.3.1 ASR	361	23.4.5.2.5 Examples	374
23.4.3.3.2 LSR	361	23.4.5.3 ASR, LSL, LSR, and ROR	374
23.4.3.3.3 LSL	362	23.4.5.3.1 Syntax	374
23.4.3.3.4 ROR	363	23.4.5.3.2 Operation	375
23.4.3.4 Address alignment	363	23.4.5.3.3 Restrictions	375
23.4.3.5 PC-relative expressions	363	23.4.5.3.4 Condition flags	375
23.4.3.6 Conditional execution	364	23.4.5.3.5 Examples	375
23.4.3.6.1 The condition flags	364	23.4.5.4 CMP and CMN	375
23.4.3.6.2 Condition code suffixes	364	23.4.5.4.1 Syntax	376
23.4.4 Memory access instructions	365	23.4.5.4.2 Operation	376
23.4.4.1 ADR	365	23.4.5.4.3 Restrictions	376
23.4.4.1.1 Syntax	365	23.4.5.4.4 Condition flags	376
23.4.4.1.2 Operation	366	23.4.5.4.5 Examples	376
23.4.4.1.3 Restrictions	366	23.4.5.5 MOV and MVN	376
23.4.4.1.4 Condition flags	366	23.4.5.5.1 Syntax	376
23.4.4.1.5 Examples	366	23.4.5.5.2 Operation	377
23.4.4.2 LDR and STR, immediate offset	366	23.4.5.5.3 Restrictions	377
23.4.4.2.1 Syntax	366	23.4.5.5.4 Condition flags	377
23.4.4.2.2 Operation	366	23.4.5.5.5 Example	377
23.4.4.2.3 Restrictions	367	23.4.5.6 MULS	377
23.4.4.2.4 Condition flags	367	23.4.5.6.1 Syntax	377
23.4.4.2.5 Examples	367	23.4.5.6.2 Operation	378
23.4.4.3 LDR and STR, register offset	367	23.4.5.6.3 Restrictions	378
23.4.4.3.1 Syntax	367	23.4.5.6.4 Condition flags	378
23.4.4.3.2 Operation	368	23.4.5.6.5 Examples	378
23.4.4.3.3 Restrictions	368	23.4.5.7 REV, REV16, and REVSH	378
23.4.4.3.4 Condition flags	368	23.4.5.7.1 Syntax	378
23.4.4.3.5 Examples	368	23.4.5.7.2 Operation	378
23.4.4.4 LDR, PC-relative	368	23.4.5.7.3 Restrictions	379
23.4.4.4.1 Syntax	368	23.4.5.7.4 Condition flags	379

23.4.5.7.5 Examples	379	23.4.7.7.1 Syntax	386
23.4.5.8 SXT and UXT	379	23.4.7.7.2 Operation	386
23.4.5.8.1 Syntax	379	23.4.7.7.3 Restrictions	386
23.4.5.8.2 Operation	379	23.4.7.7.4 Condition flags	386
23.4.5.8.3 Restrictions	379	23.4.7.7.5 Examples	386
23.4.5.8.4 Condition flags	379	23.4.7.8 NOP	386
23.4.5.8.5 Examples	380	23.4.7.8.1 Syntax	386
23.4.5.9 TST	380	23.4.7.8.2 Operation	386
23.4.5.9.1 Syntax	380	23.4.7.8.3 Restrictions	386
23.4.5.9.2 Operation	380	23.4.7.8.4 Condition flags	387
23.4.5.9.3 Restrictions	380	23.4.7.8.5 Examples	387
23.4.5.9.4 Condition flags	380	23.4.7.9 SEV	387
23.4.5.9.5 Examples	380	23.4.7.9.1 Syntax	387
23.4.6 Branch and control instructions	380	23.4.7.9.2 Operation	387
23.4.6.1 B, BL, BX, and BLX	381	23.4.7.9.3 Restrictions	387
23.4.6.1.1 Syntax	381	23.4.7.9.4 Condition flags	387
23.4.6.1.2 Operation	381	23.4.7.9.5 Examples	387
23.4.6.1.3 Restrictions	381	23.4.7.10 SVC	387
23.4.6.1.4 Condition flags	382	23.4.7.10.1 Syntax	387
23.4.6.1.5 Examples	382	23.4.7.10.2 Operation	387
23.4.7 Miscellaneous instructions	382	23.4.7.10.3 Restrictions	387
23.4.7.1 BKPT	383	23.4.7.10.4 Condition flags	387
23.4.7.1.1 Syntax	383	23.4.7.10.5 Examples	388
23.4.7.1.2 Operation	383	23.4.7.11 WFE	388
23.4.7.1.3 Restrictions	383	23.4.7.11.1 Syntax	388
23.4.7.1.4 Condition flags	383	23.4.7.11.2 Operation	388
23.4.7.1.5 Examples	383	23.4.7.11.3 Restrictions	388
23.4.7.2 CPS	383	23.4.7.11.4 Condition flags	388
23.4.7.2.1 Syntax	383	23.4.7.11.5 Examples	388
23.4.7.2.2 Operation	383	23.4.7.12 WFI	388
23.4.7.2.3 Restrictions	384	23.4.7.12.1 Syntax	388
23.4.7.2.4 Condition flags	384	23.4.7.12.2 Operation	389
23.4.7.2.5 Examples	384	23.4.7.12.3 Restrictions	389
23.4.7.3 DMB	384	23.4.7.12.4 Condition flags	389
23.4.7.3.1 Syntax	384	23.4.7.12.5 Examples	389
23.4.7.3.2 Operation	384	23.5 Peripherals	389
23.4.7.3.3 Restrictions	384	23.5.1 About the ARM Cortex-M0	389
23.4.7.3.4 Condition flags	384	23.5.2 Nested Vectored Interrupt Controller	389
23.4.7.3.5 Examples	384	23.5.2.1 Accessing the Cortex-M0 NVIC registers using CMSIS	390
23.4.7.4 DSB	384	23.5.2.2 Interrupt Set-enable Register	390
23.4.7.4.1 Syntax	384	23.5.2.3 Interrupt Clear-enable Register	391
23.4.7.4.2 Operation	384	23.5.2.4 Interrupt Set-pending Register	391
23.4.7.4.3 Restrictions	384	23.5.2.5 Interrupt Clear-pending Register	392
23.4.7.4.4 Condition flags	385	23.5.2.6 Interrupt Priority Registers	392
23.4.7.4.5 Examples	385	23.5.2.7 Level-sensitive and pulse interrupts	393
23.4.7.5 ISB	385	23.5.2.7.1 Hardware and software control of interrupts	393
23.4.7.5.1 Syntax	385	23.5.2.8 NVIC usage hints and tips	394
23.4.7.5.2 Operation	385	23.5.2.8.1 NVIC programming hints	394
23.4.7.5.3 Restrictions	385	23.5.3 System Control Block	394
23.4.7.5.4 Condition flags	385	23.5.3.1 The CMSIS mapping of the Cortex-M0 SCB registers	395
23.4.7.5.5 Examples	385	23.5.3.2 CPUID Register	395
23.4.7.6 MRS	385	23.5.3.3 Interrupt Control and State Register	395
23.4.7.6.1 Syntax	385	23.5.3.4 Application Interrupt and Reset Control Register	397
23.4.7.6.2 Operation	385	23.5.3.5 System Control Register	398
23.4.7.6.3 Restrictions	385	23.5.3.6 Configuration and Control Register	399
23.4.7.6.4 Condition flags	386		
23.4.7.6.5 Examples	386		
23.4.7.7 MSR	386		

23.5.3.7	System Handler Priority Registers	399	23.5.4.2	SysTick Reload Value Register	401
23.5.3.7.1	System Handler Priority Register 2	399	23.5.4.2.1	Calculating the RELOAD value	401
23.5.3.7.2	System Handler Priority Register 3	400	23.5.4.3	SysTick Current Value Register	401
23.5.3.8	SCB usage hints and tips	400	23.5.4.4	SysTick Calibration Value Register	402
23.5.4	System timer, SysTick	400	23.5.4.5	SysTick usage hints and tips	402
23.5.4.1	SysTick Control and Status Register	401	23.6	Cortex-M0 instruction summary	402

Chapter 24: Supplementary information

24.1	Abbreviations	406
24.2	References	406
24.3	Legal information	407
24.3.1	Definitions	407
24.3.2	Disclaimers	407
24.3.3	Trademarks	407
24.4	Tables	408
24.5	Figures	415
24.6	Contents	417