2.4GHz IEEE 802.15.4 和 ZigBee® 应用的 CC253X 片上系统解决方案

用户指南

文件编号：SWRU191
2009 年 4 月

目录

图清单

表清单

致读者

关于本手册

2.4GHz 的 CC253x 片上系统解决方案适合于广泛的应用。它们可以很容易建立在基于 IEEE 802.15.4 标准协议（RemoTI™网络协议、TIMAC 软件和用于 ZigBee®兼容解决方案的 Z-Stack™软件）上面，或是专门的 SimpliciTI™网络协议上面。但是它们的使用不仅限于这些协议。例如 CC253x 系列还适合于 6LoWPAN 和无线 HART 的实现。

本手册的每一章描述了一个模块或外设的详细信息，但是并没有完全列出 CC253x 系列所有设备的全部功能。

关于详细的技术参数，比如功率消耗和 RF 性能，见设备具体的数据手册。

德州仪器的相关文档和软件

相关文档（如 CC2530 数据手册 http://www-s.ti.com/sc/techlit/swrs081）可在附录 C 中找到。

关于可以用于 CC253x 片上系统解决方案（例如用于无线电性能和功能评估的 SmartRF™软件）的软件的更多信息见第 21 章，这一章还包括关于 RemoTI 网络协议、SimpliciTI 网络协议、TIMAC 软件和 Z-Stack 软件的更多信息。

FCC 警告

这一设备仅用于实验室测试环境。它可以产生、使用且可以辐射射频能量，但是尚未测试是否兼容根据 FCC 规则 15 部分计算设备的限制，它的设计是为了提供合理的保护，免受射频干扰。这一设备在其他环境下的运行可能导致对射频通信的干扰，在这种情况下自费的用户需要采取可能要求的纠正这一干扰的一切措施。

如果你需要帮助

所有技术支持渠道均通过 TI 产品信息中心（PIC）——www.ti.com/support 提供。

要发送电子邮件请求，请在以下链接输入您的联系信息以及您的请求——PIC 请求表。

还可以访问 TI E2E 社区的低功耗 RF 和 ZigBee 部分（www.ti.com/lprf-forum），在这里您可以很容易与其他 CC253x 用户沟通，并找到 FAQ、设计说明、应用说明、视频等等。

您还可以参见 TI 模拟&混合信号知识库。

词汇表

本用户指南使用的缩略语可在附录 A 中找到。

设备

CC253x 片上系统解决方案系列包括若干设备。下表提供了每个设备不同外设、内存大小等的概述和信息。

CC253x 系列概览

特征	CC2530F32/F64/F128/F256	CC2531F256
FLASH_SIZE	32 KB/64 KB/128 KB/256 KB	256KB
SRAM_SIZE	8KB	8KB
USB	不包括	包括

图例：

FLASH_SIZE –闪存的大小，以字节为单位

SRAM_SIZE –SRAM 的大小，以字节为单位

寄存器约定

每个 SFR 和 XREG 寄存器在一个单独的表中描述，每个表标题按照以下格式：

SFR 寄存器：寄存器名称（SFR 地址）——寄存器描述

XREG 寄存器：寄存器名称（XDATA 地址）——寄存器描述

每个表的每一行中有五列描述寄存器部分，描述如下：

列 1——位：说明指定行描述的是寄存器的哪一位

列 2——名称：寄存器部分的具体名称

列 3——复位：寄存器部分的复位/初始值

列 4——R/W：表示各个位是否可访问的密钥（更多详细信息见表 1）

列 5——描述：寄存器部分的更多信息，通常是不同值的描述

在寄存器描述中，每个寄存器位用一个符号（R/W）表示寄存器位的访问模式。寄存器值总是以二进制表示法给定，除非前缀是 0x 表示十六进制表示法。

表 1　寄存器位约定

符号	访问模式
R/W	读/写
R	只读
R0	读作 0
R1	读作 1
W	只写
W0	写作 0
W1	写作 1
H0	硬件清除
H1	硬件设置

简介

如前言所述，CC253x 设备系列为广泛的应用提供了解决方案。为了帮助用户开发这些应用，这一用户指南的重点是 CC253x 设备系列不同的构造模块的用法。关于详细的设备描述、完整的功能列表和性能参数，读者可以参见各个设备的数据手册。

为了方便获取相关信息，以下小节引导读者到本指南不同的章节。

标题 页

1.1 概述

图 1-1 的方框图显示了 CC253x 设备系列不同的构造模块。并没有列出 CC253x 所有设备的所有模块和外设的全部功能和函数，因此关于具体设备的方框图参见各个设备的数据手册。

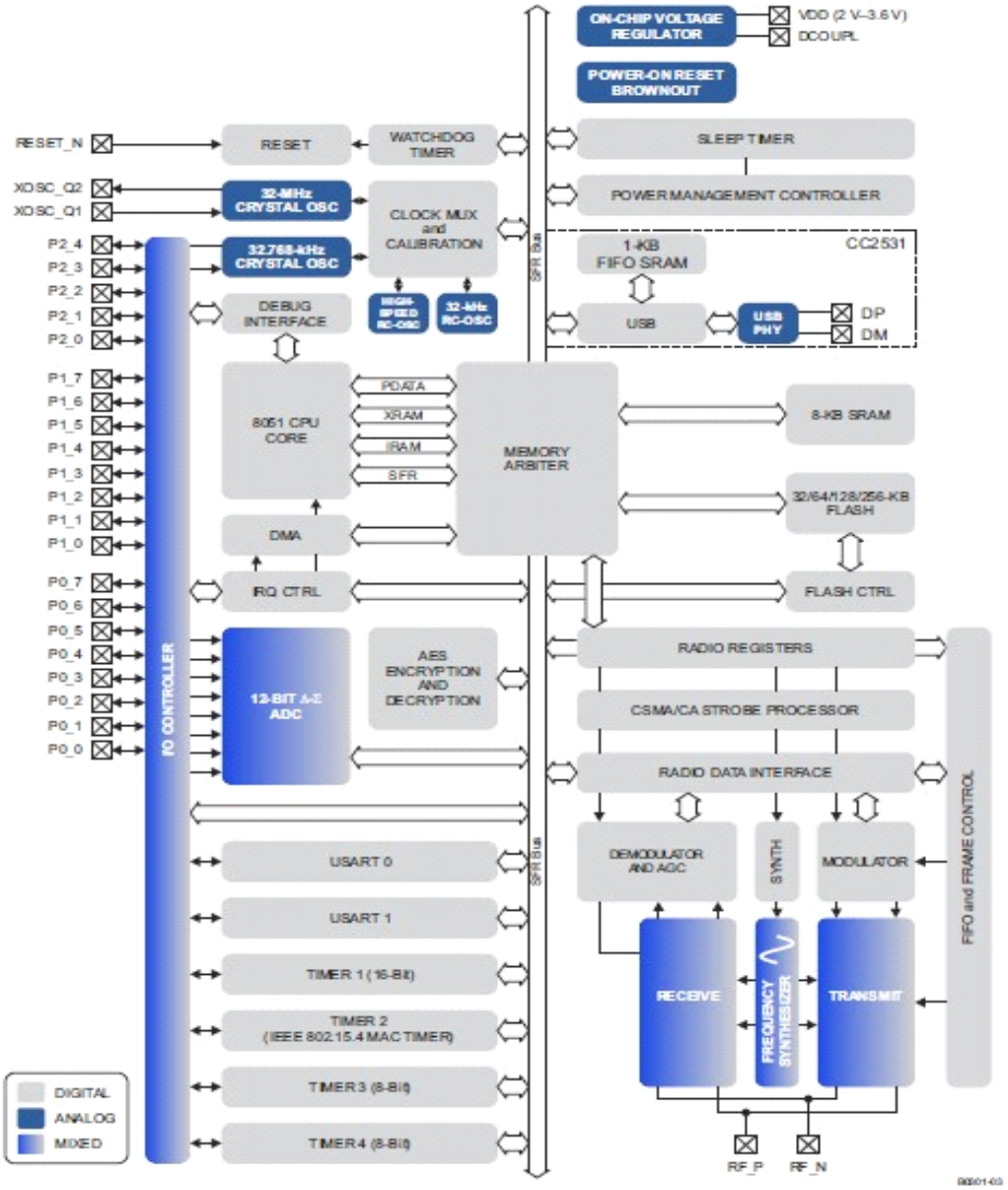


图 1-1　CC253x 方框图

模块大致可以分为三种类型：CPU 和内存相关的模块；外设、时钟和电源管理相关的模块；无线电相关的模块。

1.1.1 CPU 和内存

CC253x 设备系列使用的 **8051 CPU 内核**是一个单周期的 8051 兼容内核。它有三个不同的存储器访问总线（SFR、DATA 和 CODE/XDATA），以单周期访问 SFR、DATA 和主 SRAM。它还包括一个调试接口和一个 18 输入的扩展中断单元。CPU 和内存的详细功能见第 2 章。

中断控制器提供了 18 个中断源，分为六个中断组，每组与四个中断优先级相关。当设备从空闲模式回到活动模式，也会发出一个中断服务请求。一些中断还可以从睡眠模式唤醒设备（供电模式 1-3），详细信息参见第 4 章。

内存仲裁器位于系统中心，因为它通过 SFR 总线，把 CPU 和 DMA 控制器和物理存储器和所有外设连接在一起。内存仲裁器有四个存取访问点，访问每一个可以映射到三个物理存储器之一：一个 8-KB SRAM、一个闪存存储器和一个 XREG/SFR 寄存器。它负责执行仲裁，并确定同时到同一个物理存储器的内存访问的顺序。

8-KB SRAM 映射到 DATA 存储空间和 XDATA 存储空间的一部分。8-KB SRAM 是一个超低功耗的 SRAM，当数字部分掉电时（供电模式 2 和 3）能够保留自己的内容。这对于低功耗应用是一个很重要的功能。

32/64/128/256 KB 闪存块为设备提供了内电路可编程的非易失性程序存储器，映射到 CODE 和 XDATA 存储空间。除了保存程序代码和常量，非易失性程序存储器允许应用程序保存必须保留的数据，这样在设备重新启动之后可以使用这些数据。使用这个功能，例如可以利用已经保存的网络具体数据，就不需要经过完整的启动、网络寻找和加入过程。

1.1.2 时钟和电源管理

数字内核和外设由一个 1.8-V 低差**稳压器**供电（第 20 章）。另外 CC253x 包括一个电源管理功能（第 4 章），可以实现使用不同供电模式的长电池寿命的低功耗应用运行。有五种不同的**复位**源来复位设备，详细信息见第 5 章。

1.1.3 外设

CC2530 包括许多不同的外设，允许应用程序设计者开发先进的应用。

调试接口（第 3 章）执行一个专有的两线串行接口，用于内电路调试。通过这个调试接口，可以执行整个闪存存储器的擦除、控制使能哪个振荡器、停止和开始执行用户程序、执行 8051 内核提供的指令、设置代码断点，以及内核中全部指令的单步调试。使用这些技术，可以很好地执行内电路的调试和外部闪存的编程。

设备含有闪存存储器以存储程序代码。闪存存储器可通过用户软件和调试接口编程（如上所述）。**闪存控制器**（第 6 章）处理写入和擦除嵌入式闪存存储器。闪存控制器允许页面擦除和 4 字节编程。

I/O 控制器（第 7 章）负责所有通用 I/O 引脚。CPU 可以配置外设模块是否控制某个引脚或它们是否受软件控制，如果是的话，每个引脚配置为一个输入还是输出，是否连接衬垫里的一个上拉或下拉电阻。CPU 中断可以分别在每个引脚上使能。每个连接到 I/O 引脚的外设可以在两个不同的 I/O 引脚位置之间选择，以确保在不同应用程序中的灵活性。

系统可以使用一个多功能的五通道 **DMA 控制器**（第 8 章），使用 XDATA 存储空间访问存储器，因此能够访问所有物理存储器。每个通道（触发器、优先级、传输模式、寻址模式、源和目标指针和传输计数）用 DMA 描述符在存储器任何地方配置。许多硬件外设（AES 内核、闪存控制器、USART、定时器、ADC 接口）通过使用 DMA 控制器在 SFR 或 XREG 地址和闪存/SRAM 之间进行数据传输，获得高效率操作。

定时器 1（第 9 章）是一个 16 位定时器，具有定时器/计数器/PWM 功能。它有一个可编程的分频器，一个 16 位周期值，和五个各自可编程的计数器/捕获通道，每个都有一个 16 位比较值。每个计数器/捕获通道可以用

作一个 PWM 输出或捕获输入信号边沿的时序。它还可以配置在 IR 产生模式，计算定时器 3 周期，输出和定时器 3 的输出相与，用最小的 CPU 互动产生调制的消费型 IR 信号（见 9.9 节）。

定时器 2（MAC 定时器）（第 18 章）是专门为支持 IEEE 802.15.4 MAC 或软件中其他时槽的协议设计。定时器有一个可配置的定时器周期和一个 8 位溢出计数器，可以用于保持跟踪已经经过的周期数。一个 16 位捕获寄存器也用于记录收到/发送一个帧开始界定符的精确时间，或传输结束的精确时间，还有一个 16 位输出比较寄存器可以在具体时间产生不同的选通命令（开始 RX，开始 TX，等等）到无线模块。

定时器 3 和定时器 4（第 10 章）是 8 位定时器，具有定时器/计数器/PWM 功能。它们有一个可编程的分频器，一个 8 位的周期值，一个可编程的计数器通道，具有一个 8 位的比较值。每个计数器通道可以用作一个 PWM 输出。

睡眠定时器（第 11 章）是一个超低功耗的定时器，计算 32-kHz 晶振或 32-kHz RC 振荡器的周期。睡眠定时器在除了供电模式 3 的所有工作模式下不断运行。这一定时器的典型应用是作为实时计数器，或作为一个唤醒定时器跳出供电模式 1 或 2。

ADC（第 12 章）支持 7 到 12 位的分辨率，分别在 30 kHz 或 4 kHz 的带宽。DC 和音频转换可以使用高达八个输入通道（端口 0）。输入可以选择作为单端或差分。参考电压可以是内部电压、AVDD 或是一个单端或差分外部信号。ADC 还有一个温度传感输入通道。ADC 可以自动执行定期抽样或转换通道序列的程序。

随机数发生器（第 13 章）使用一个 16 位 LFSR 来产生伪随机数，这可以被 CPU 读取或由选通命令处理器直接使用。例如随机数可以用作产生随机密钥，用于安全。

AES 协处理器（第 14 章）允许用户使用带有 128 位密钥的 AES 算法加密和解密数据。这一内核能够支持 IEEE 802.15.4 MAC 安全、ZigBee 网络层和应用层要求的 AES 操作。

一个内置的**看门狗定时器**（第 15 章）允许设备在固件挂起的情况下复位自身。当看门狗定时器由软件使能，它必须定期清除；否则，当它超时它就复位设备。或者它可以配置用作一个通用 32-kHz 定时器。

USART 0 和 **USART 1**（第 16 章）每个被配置为一个 SPI 主/从或一个 UART。它们为 RX 和 TX 提供了双缓冲，以及硬件流控制，因此非常适合于高吞吐量的全双工应用。每个都有自己的高精度波特率发生器，因此可以使普通定时器空闲出来用作其他用途。

USB 2.0 全速控制器（仅 CC2531 可用）有 5 个端点，1KB FIFO RAM 的双缓冲。它的功能描述在第 17 章 。

1.1.4 无线电

CC253x 设备系列提供了一个 **IEEE 802.15.4 兼容无线收发器**。RF 内核控制模拟无线模块。另外，它提供了 MCU 和无线设备之间的一个接口，这使得可以发出命令、读取状态、自动操作和确定无线设备事件的顺序。无线设备还包括一个数据包过滤和地址识别模块。关于无线电的更多详细信息可以在第 19 章找到。

1.2 应用

如概述（1.1 节）所示，这一用户指南的重点是使用可用的不同模块，来建立基于 CC253x 设备系列的不同类型的应用程序。如果查看完整的应用程序开发过程，另外的一些信息很有用。但是，因为这样的信息和帮助不是设备特定的（即不是 CC253x 设备系列唯一有的），读者可以参考以下段落的另外的信息源。

第一步是通过购买一个**开发套件**（见设备具体的产品网站，找到相关开发套件的链接），设置开发环境（HW 工具等）。开发套件带有一个现成的演示和关于如何设置开发环境的信息；安装所需的驱动（通过安装**SmartRF 软件**可以轻松完成，21.1 节），设置编译器工具链，等等。只要已经安装了开发环境，就可以准备开始应用程序的开发了。

写应用程序软件的最简单的方法是把应用程序建立在可用的标准协议之一（**RemoTI** 网络协议，21.2 节；**TIMAC** 软件，21.4 节；或用于 ZigBee 兼容解决方案的 **Z-Stack 软件**，21.5 节）上面；或者专门的 **SimpliciTI** 网络协议（21.3 节）上面。它们都带有一些实例应用程序。

对于用户具体 HW 的硬件布局设计，设计人员可以在不同的产品页面（B.1 节）上找到参考设计。通过复制这些设计，设计人员能够获得最佳性能。已开发的 HW 可以使用 SmartRF Studio 软件（21.1 节）很容易地进行测试。

如果最终的系统没有达到预期的性能，建议在开发套件的硬件上尝试开发的软件，看看在这里工作地如何。要检查用户具体的 HW，使用 SmartRF Studio 软件，在相同的设置下来比较开发套件性能和用户具体的 HW，这是一个良好的开端。

用户还可以通过加入**低功耗 RF 网络社区**（B.2 节）和订阅**低功耗 RF 电子简讯**（B.4 节），找到另外的信息和帮助。

要联系第三方以帮助开发或者要使用模块，查询德州仪器**低功耗 RF 开发商网络**（B.3 节）。

8051 CPU

片上系统解决方案是基于一个增强的 8051 内核。关于此内核、存储映射、指令集和中断的更多详细信息在以下章节描述。

标题 页

2.1 8051 CPU 简介

增强型 8051 内核使用标准的 8051 指令集。因为以下原因指令执行比标准的 8051 更快：

- 每个指令周期是一个时钟，而标准的 8051 每个指令周期是 12 个时钟。
- 消除了总线状态的浪费。

因为一个指令周期与可能的内存存取是一致的，大多数单字节指令在一个时钟周期内执行。除了速度提高之外，增强型 8051 内核还包括结构上的改善：

- 第二个数据指针
- 一个扩展的 18 源中断单元

8051 内核的对象代码兼容业界标准的 8051 微控制器。即对象代码使用 8051 内核上执行的业界标准的 8051 编译器或汇编器编译，在功能上是等同的。但是，因为 8051 内核使用了不同于许多其他 8051 类型的一个指令时序，带有时序循环的已有代码可能需要修改。而且，因为诸如定时器和串行端口的外设单元不同于其他 8051 内核，包含使用外设单元 SFR 的指令的代码不能正确运行。

闪存预取默认不是使能的，但是提高了 CPU 高达 33%的性能。这一设置的代价是功率消耗略有增加，但是因为这样更快，大多数情况下提高了能源消耗。闪存预取可以在 FCTL 寄存器中使能。

2.2 存储器

8051 CPU 结构有四个不同的存储空间。8051 有单独的存储空间用于程序存储和数据存储。8051 存储空间如下（详细信息见 2.2.1 节和 2.2.2 节）：

CODE：一个只读的存储空间，用于程序存储。这一存储空间地址是 64KB。

DATA：一个读/写的数据存储空间，可以直接或间接被一个单周期 CPU 指令访问。这一存储空间地址是 256 字节。DATA 存储空间较低的 128 字节可以直接或间接寻址，较高的 128 字节只能间接寻址。

XDATA：一个读/写的数据存储空间，通常需要 4-5 个 CPU 指令周期来访问。这一存储空间地址是 64KB。而且访问 XDATA 存储器慢于访问 DATA，因为 CODE 和 XDATA 存储空间共享 CPU 内核上的一个通用总线，因此来自 CODE 的指令预取可以不必和 XDATA 访问并行执行。

SFR：一个读/写的寄存器存储空间，可以直接被一个 CPU 指令访问。这一存储空间含有 128 字节。对于地址是被 8 整除的 SFR 寄存器，每一位还可以单独寻址。

这四个存储空间在 8051 结构中是分开的，但是在设备中有部分是重叠的，以减轻 DMA 传输和硬件调试操作的负担。

不同的存储空间如何映射到三个物理存储器（闪存程序存储器、SRAM 和存储映射存储器）在 2.2.1 节和 2.2.2 节描述。

2.2.1 存储器映射

存储器映射在两个重要方面不同于标准的 8051 内存映射，如以下段落所述。

首先，为了使得 DMA 控制器访问全部物理存储空间，并由此使得 DMA 在不同 8051 存储空间之间进行传输，CODE 和 SFR 部分存储空间映射到 XDATA 存储空间。

第二，CODE 存储器空间映射可以使用两个备用机制。第一个机制是标准的 8051 映射，只有程序存储器（即闪存存储器）映射到 CODE 存储空间。这一映射设备复位后默认使用的。

第二个机制用于执行来自 SRAM 的代码。在这种模式下，SRAM 映射到 0x8000 到(0x8000 + SRAM_SIZE – 1)的区域。这一映射如图 2-2 所示。执行来自 SRAM 的代码提高了性能，并减少了功率消耗。

XDATA 较高的 32KB 是一个只读的区域，叫做 XBANK。任何可用的 32KB 闪存区可以在这里被映射出来。这使得软件可以访问整个闪存存储器。这一区域典型用作存储另外的常量数据。

所有 8051 存储空间的映射的详细信息见 2.2.2 节。

显示不同的物理存储器如何映射到 CPU 存储空间的存储映射见图 2-1 到图 2-3。可用的闪存区的编号取决于闪存大小的选项。

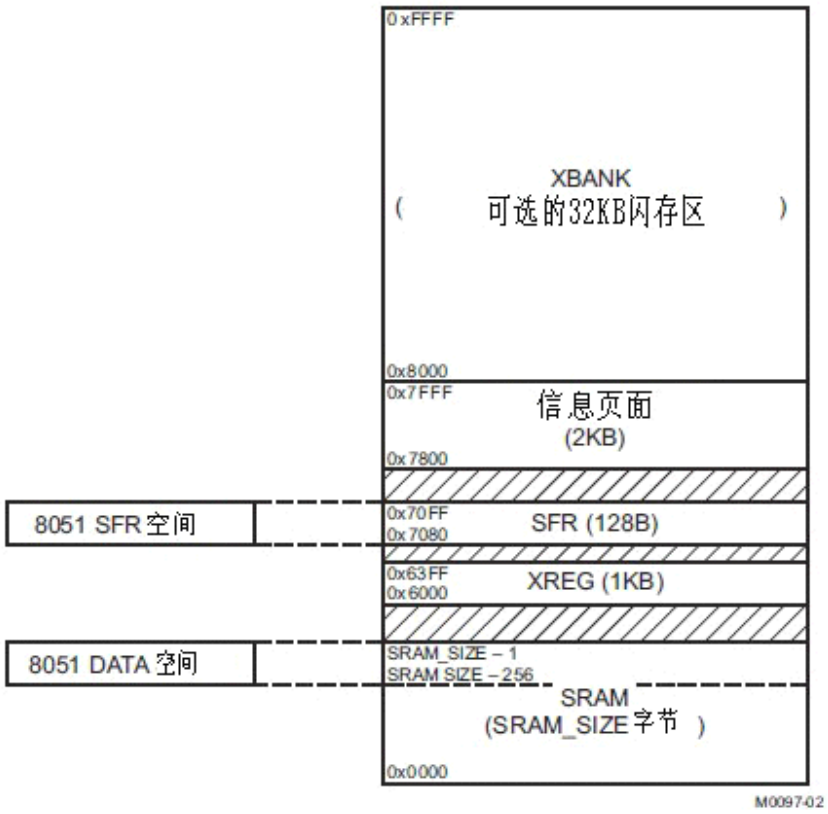


图 2-1 XDATA 存储空间（显示 SFR 和 DATA 映射）

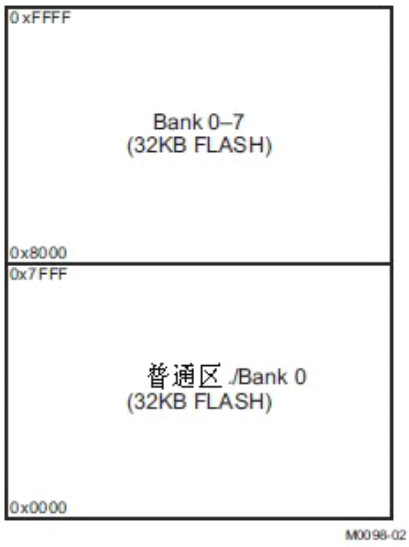


图 2-2 CODE 存储空间

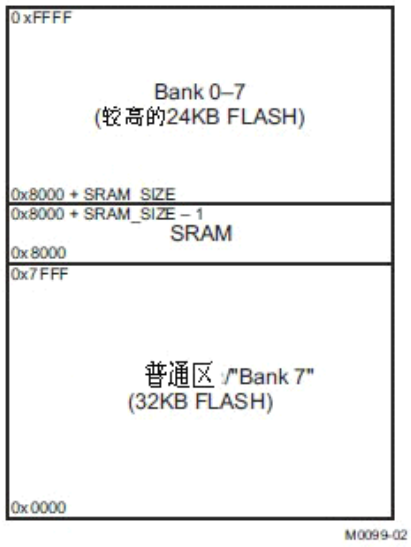


图 2-3 用于运行来自 SRAM 的代码的 CODE 存储空间

2.2.2 CPU 存储空间

XDATA 存储空间：XDATA 存储映射见图 2-1。

SRAM 映射到的地址范围是 0x0000 到(SRAM_SIZE – 1)。

XREG 区域映射到 1-KB 地址区域(0x6000–0x63FF)。这些寄存器是另外的寄存器，有效地扩展 SFR 寄存器空间。一些外设寄存器和大多数无线电控制和数据寄存器映射到这里。

SFR 寄存器映射到地址区域(0x7080–0x70FF)。

闪存信息页面（2 KB）映射到地址区域(0x7800–0x7FFF)。这是一个只读区域，包含有关设备的各种信息。

XDATA 存储空间(0x8000–0xFFFF)的较高 32KB 是一个只读的闪存代码区（XBANK），可以使用 MEMCTR.XBANK[2:0]位映射到任何一个可用的闪存区。

闪存存储器 SRAM 和寄存器到 XDATA 的映射允许 DMA 控制器和 CPU 访问在一个统一的地址空间内的所有物理存储器。

写入存储映射中未执行的区域（图中的阴影部分）没有影响。从为执行的区域读出返回 0x00。写只读区域比如闪存区域将被忽略。

CODE 存储空间：CODE 存储空间是 64KB，分为一个普通区域(0x0000–0x7FFF)和一个区域(0x8000–0xFFFF)，如图 2-2。普通区域总是映射到物理闪存存储器较低的 32KB（区 0）。另一区域可以映射到任一可用的 32-KB 闪存区（从 0 到 7）。可用的闪存区的编号取决于闪存大小的选项。使用闪存区选择寄存器 FMAP 来选择闪存区。在 32 KB 设备上，闪存存储器不能映射到上面的区域。从这些设备的这一区域读返回 0x00。

要允许从 SRAM 执行程序，可以映射可用的 SRAM 到区域（范围从 0x8000 到(0x8000+SRAM_SIZE–1)）较低的区域。当前选择的区的其余部分仍映射到地址区域(0x8000 + SRAM_SIZE)到 0xFFFF。设置 MEMCTR.XMAP 位来使能这一功能。

DATA 存储空间: DATA 存储器 8 位的地址区域映射到 SRAM 较高的 256 字节，即地址范围从(SRAM_SIZE-256)到(SRAM_SIZE-1)。

SFR 存储空间: 128 个条目的硬件寄存器区域是通过这一存储空间访问的。SFR 寄存器还可以通过 XDATA 地址空间（地址范围是(0x7080 –0x70FF)）访问。一些 CPU 具体的 SFR 寄存器驻留在 CPU 内核，只能使用 SFR 存储空间访问，不能通过复制映射到 XDATA 存储空间访问。这些具体的 SFR 寄存器列在 SFR 寄存器表中。

2.2.3 物理存储器

RAM：所有设备都包括静态 RAM。上电时，RAM 的内容是未定义的。RAM 内容在所有供电模式下都保留。

闪存存储器：片上闪存存储器主要是为了保存程序代码和常量数据。闪存存储器有以下功能：

- 页面大小：2 KB
- 闪存页面擦除时间：20ms
- 闪存芯片（批量）擦除时间：20ms
- 闪存写时间（4 字节）：20µs
- 数据保留（室温下）：100 年
- 编程/擦除次数：20,000 次

闪存存储器由一组 2 KB 的页面组成。较高的可用页面的 16 字节包括页面锁位和调试锁位。每一个页面都有一个锁位，除了不是处于调试模式就暗示锁定的锁位页面。当页面的锁位是 0，可以擦除/写该页。当调试锁位是 0，调试接口上的大多数命令被忽略。调试锁位的主要目的是保护闪存的内容不被读出。闪存控制器用作写和擦除闪存存储器的内容。

当 CPU 从闪存存储器中读指令和常量，它通过一个缓存取出指令。指令的四个字节和常量数据的四个字节被缓存，在 4 字节的边界。即例如当 CPU 从地址 0x00F1 读，字节 0x00F0–0x00F3 被缓存。一个单独的预取单元可以用来预取指令的另外 4 个字节。这一缓存主要是为了通过减少访问闪存存储器的总时间来减少功率消耗。缓存可以通过 FCTL.CM[1:0]寄存器位禁用。这样做增加了功率消耗，是不推荐的。当使用默认缓存模式和带预取的缓存模式，即不能准确确定读取一组指令花费的时钟周期数量，从闪存读的执行时间不是精确到周期的。要获得精确到周期的执行，就要使能实时缓存模式，确保所有 DMA 传输有低优先级。预取模式提高性能高达 33%，但因为浪费了闪存读，代价是增加了功率消耗。一般地，性能可以提高 15%-20%。但是由于等待闪存返回指令/数据浪费的时钟周期较少，总的能量是减少的（取决于应用程序）。这是非常依赖于应用程序的，需要有效的使用供电模式。

信息页面是一个 2 KB 的只读区域，存储设备信息。其他信息中它包括来自 TI 地址范围的一个唯一的 IEEE 地址。它以最低位优先的形式存储在 XDATA 地址 0x780C。将出版一个单独的设计说明，详细介绍信息页面的内容。

SFR 寄存器：特殊功能寄存器（SFR）控制 8051CPU 内核和/或外设的一些功能。许多 8051CPU 内核的 SFR 和标准的 8051SFR 相同。但是有一些控制功能的另外的 SFR，是标准 8051 中所没有的。另外的 SFR 用于和外设单元以及 RF 收发器接口。

表 2-1 显示了设备中所有 SFR 的地址。8051 内部 SFR 以灰色背景显示，而其他 SFR 是具体于设备的 SFR。

注意：所有内部 SFR（在表 2-1 中以灰色背景显示）只能通过 SFR 空间访问，因为这些寄存器没有映射到 XDATA 空间。但例外是端口寄存器（P0、P1 和 P2）可以从 XDATA 中读取。

表 2-1 SFR 概览

寄存器名称	SFR 地址	模块	描述
ADCCON1	0xB4	ADC	ADC 控制 1
ADCCON2	0xB5	ADC	ADC 控制 2
ADCCON3	0xB6	ADC	ADC 控制 3
ADCL	0xBA	ADC	ADC 数据低字节
ADCH	0xBB	ADC	ADC 数据高字节
RNDL	0xBC	ADC	随机数发生器数据低字节
RNDH	0xBD	ADC	随机数发生器数据高字节
ENCDI	0xB1	AES	加密/解密输入数据
ENCDO	0xB2	AES	加密/解密输出数据
ENCCS	0xB3	AES	加密/解密控制和状态
P0	0x80	CPU	端口 0。可从 XDATA（0x7080）中读出。
SP	0x81	CPU	栈指针
DPL0	0x82	CPU	数据指针 0 低字节
DPH0	0x83	CPU	数据指针 0 高字节
DPL1	0x84	CPU	数据指针 1 低字节
DPH1	0x85	CPU	数据指针 1 高字节
PCON	0x87	CPU	供电模式控制
TCON	0x88	CPU	中断标志
P1	0x90	CPU	端口 1。可从 XDATA（0x7090）读出。
DPS	0x92	CPU	数据指针选择
S0CON	0x98	CPU	中断标志 2
IEN2	0x9A	CPU	中断使能 2
S1CON	0x9B	CPU	中断标志 3
P2	0xA0	CPU	端口 2。可从 XDATA（0x70A0）读出。
IEN0	0xA8	CPU	中断使能 0
IP0	0xA9	CPU	中断优先级 0
IEN1	0xB8	CPU	中断使能 1
IP1	0xB9	CPU	中断优先级 1
IRCON	0xC0	CPU	中断标志 4
PSW	0xD0	CPU	程序状态字
ACC	0xE0	CPU	累加器
IRCON2	0xE8	CPU	中断标志 5
B	0xF0	CPU	B 寄存器
DMAIRQ	0xD1	DMA	DMA 中断标志
DMA1CFGL	0xD2	DMA	DMA 通道 1-4 配置地址低字节
DMA1CFGH	0xD3	DMA	DMA 通道 1-4 配置地址高字节
DMA0CFGL	0xD4	DMA	DMA 通道 0 配置地址低字节
DMA0CFGH	0xD5	DMA	DMA 通道 0 配置地址高字节
DMAARM	0xD6	DMA	DMA 通道准备工作
DMAREQ	0xD7	DMA	DMA 通道开始请求和状态
—	0xAA	—	保留
—	0x8E	—	保留
—	0x99	—	保留
—	0xB0	—	保留
—	0xB7	—	保留
—	0xC8	—	保留

表 2-1 SFR 概览（续表）

寄存器名称	SFR 地址	模块	描述
P0IFG	0x89	IOC	端口 0 中断状态标志
P1IFG	0x8A	IOC	端口 1 中断状态标志
P2IFG	0x8B	IOC	端口 2 中断状态标志
PICTL	0x8C	IOC	端口引脚中断屏蔽和边沿
P0IEN	0xAB	IOC	端口 0 中断屏蔽
P1IEN	0x8D	IOC	端口 1 中断屏蔽
P2IEN	0xAC	IOC	端口 2 中断屏蔽
P0INP	0x8F	IOC	端口 0 输入模式
PERCFG	0xF1	IOC	外设 I/O 控制
APCFG	0xF2	IOC	模拟外设 I/O 配置
P0SEL	0xF3	IOC	端口 0 功能选择
P1SEL	0xF4	IOC	端口 1 功能选择
P2SEL	0xF5	IOC	端口 2 功能选择
P1INP	0xF6	IOC	端口 1 输入模式
P2INP	0xF7	IOC	端口 2 输入模式
P0DIR	0xFD	IOC	端口 0 方向
P1DIR	0xFE	IOC	端口 1 方向
P2DIR	0xFF	IOC	端口 2 方向
PMUX	0xAE	IOC	掉电信号 Mux
MEMCTR	0xC7	MEMORY	内存系统控制
FMAP	0x9F	MEMORY	闪存存储器区映射
RFIRQF1	0x91	RF	RF 中断标志 MSB
RFD	0xD9	RF	RF 数据
RFST	0xE1	RF	RF 命令选通
RFIRQF0	0xE9	RF	RF 中断标志 LSB
RFERRF	0xBF	RF	RF 错误中断标志
ST0	0x95	ST	睡眠定时器 0
ST1	0x96	ST	睡眠定时器 1
ST2	0x97	ST	睡眠定时器 2
STLOAD	0xAD	ST	睡眠定时器负载状态
SLEEPCMD	0xBE	PMC	睡眠模式控制命令
SLEEPSTA	0x9D	PMC	睡眠模式控制状态
CLKCONCMD	0xC6	PMC	时钟控制命令
CLKCONSTA	0x9E	PMC	时钟控制状态
T1CC0L	0xDA	定时器 1	定时器 1 通道 0 捕获/比较值低字节
T1CC0H	0xDB	定时器 1	定时器 1 通道 0 捕获/比较值高字节
T1CC1L	0xDC	定时器 1	定时器 1 通道 1 捕获/比较值低字节
T1CC1H	0xDD	定时器 1	定时器 1 通道 1 捕获/比较值高字节
T1CC2L	0xDE	定时器 1	定时器 1 通道 2 捕获/比较值低字节
T1CC2H	0xDF	定时器 1	定时器 1 通道 2 捕获/比较值高字节
T1CNTL	0xE2	定时器 1	定时器 1 计数器低字节
T1CNTH	0xE3	定时器 1	定时器 1 计数器高字节
T1CTL	0xE4	定时器 1	定时器 1 控制和状态
T1CCTL0	0xE5	定时器 1	定时器 1 通道 0 捕获/比较控制
T1CCTL1	0xE6	定时器 1	定时器 1 通道 1 捕获/比较控制

表 2-1 SFR 概览（续表）

寄存器名称	SFR 地址	模块	描述
T1CCTL2	0xE7	定时器 1	定时器 1 通道 2 捕获/比较控制
T1STAT	0xAF	定时器 1	定时器 1 状态
T2CTRL	0x94	定时器 2	定时器 2 控制
T2EVTCFG	0x9C	定时器 2	定时器 2 事件配置
T2IRQF	0xA1	定时器 2	定时器 2 中断标志
T2M0	0xA2	定时器 2	定时器 2 复用寄存器 0
T2M1	0xA3	定时器 2	定时器 2 复用寄存器 1
T2MOVF0	0xA4	定时器 2	定时器 2 复用溢出寄存器 0
T2MOVF1	0xA5	定时器 2	定时器 2 复用溢出寄存器 1
T2MOVF2	0xA6	定时器 2	定时器 2 复用溢出寄存器 2
T2IRQM	0xA7	定时器 2	定时器 2 中断屏蔽
T2MSEL	0xC3	定时器 2	定时器 2 复用选择
T3CNT	0xCA	定时器 3	定时器 3 计数器
T3CTL	0xCB	定时器 3	定时器 3 控制
T3CCTL0	0xCC	定时器 3	定时器 3 通道 0 比较控制
T3CC0	0xCD	定时器 3	定时器 3 通道 0 比较值
T3CCTL1	0xCE	定时器 3	定时器 3 通道 1 比较控制
T3CCTL1	0xCF	定时器 3	定时器 3 通道 1 比较值
T4CNT	0xEA	定时器 4	定时器 4 计数器
T4CTL	0xEB	定时器 4	定时器 4 控制
T4CCTL0	0xEC	定时器 4	定时器 4 通道 0 比较控制
T4CC0	0xED	定时器 4	定时器 4 通道 0 比较值
T4CCTL1	0xEE	定时器 4	定时器 4 通道 1 比较控制
T4CC1	0xEF	定时器 4	定时器 4 通道 1 比较值
TIMIF	0xD8	TMINT	定时器 1/3/4 联合中断屏蔽/标志
U0CSR	0x86	USART 0	USART 0 控制和状态
U0DBUF	0xC1	USART 0	USART 0 接收/发送数据缓存
U0BAUD	0xC2	USART 0	USART 0 波特率控制
U0UCR	0xC4	USART 0	USART 0 UART 控制
U0GCR	0xC5	USART 0	USART 0 通用控制
U1CSR	0xF8	USART 1	USART 1 控制和状态
U1DBUF	0xF9	USART 1	USART 1 接收/发送数据缓存
U1BAUD	0xFA	USART 1	USART 1 波特率控制
U1UCR	0xFB	USART 1	USART 1 UART 控制
U1GCR	0xFC	USART 1	USART 1 通用控制
WDCTL	0xC9	WDT	看门狗定时器控制

XREG 寄存器: XREG 寄存器是 XDATA 存储空间中的另外的寄存器。这些寄存器主要用于无线电配置和控制。每个寄存器的完整描述在 3.6 节。表 2-2 给出了寄存器地址空间的概述。

表 2-2 XREG 寄存器概览

XDATA 地址	寄存器名称	描述
0x6000–0x61FF	—	无线电寄存器
0x6200–0x622B	—	USB 寄存器
0x6249	CHVER	芯片版本
0x624A	CHIPID	芯片识别标志
0x6260	DBGDATA	调试接口写数据
0x6270	FCTL	闪存控制
0x6271	FADDRL	闪存地址低字节
0x6272	FADDRH	闪存地址高字节
0x6273	FWDATA	闪存写数据
0x6276	CHIPINFO0	芯片信息字节 0
0x6277	CHIPINFO1	芯片信息字节 1
0x6290	CLD	时钟丢失探测
0x62A0	T1CCTL0	定时器 1 通道 0 捕获/比较控制（SFR 寄存器另外的 XREG 映射）
0x62A1	T1CCTL1	定时器 1 通道 1 捕获/比较控制（SFR 寄存器另外的 XREG 映射）
0x62A2	T1CCTL2	定时器 1 通道 2 捕获/比较控制（SFR 寄存器另外的 XREG 映射）
0x62A3	T1CCTL3	定时器 1 通道 3 捕获/比较控制
0x62A4	T1CCTL4	定时器 1 通道 4 捕获/比较控制
0x62A6	T1CC0L	定时器 1 通道 0 捕获/比较值低字节（SFR 寄存器另外的 XREG 映射）
0x62A7	T1CC0H	定时器 1 通道 0 捕获/比较值高字节（SFR 寄存器另外的 XREG 映射）
0x62A8	T1CC1L	定时器 1 通道 1 捕获/比较值低字节（SFR 寄存器另外的 XREG 映射）
0x62A9	T1CC1H	定时器 1 通道 1 捕获/比较值高字节（SFR 寄存器另外的 XREG 映射）
0x62AA	T1CC2L	定时器 1 通道 2 捕获/比较值低字节（SFR 寄存器另外的 XREG 映射）
0x62AB	T1CC2H	定时器 1 通道 2 捕获/比较值高字节（SFR 寄存器另外的 XREG 映射）
0x62AC	T1CC3L	定时器 1 通道 3 捕获/比较值低字节
0x62AD	T1CC3H	定时器 1 通道 3 捕获/比较值高字节
0x62AE	T1CC4L	定时器 1 通道 4 捕获/比较值低字节
0x62AF	T1CC4H	定时器 1 通道 4 捕获/比较值高字节
0x62B0	STCC	睡眠定时器捕获控制
0x62B1	STCS	睡眠定时器捕获状态
0x62B2	STCV0	睡眠定时器捕获值字节 0
0x62B3	STCV1	睡眠定时器捕获值字节 1
0x62B4	STCV2	睡眠定时器捕获值字节 2

2.2.4 XDATA 存储空间

MPAGE 寄存器在指令 MOVX A,@Ri 和 MOVX @Ri, A 期间使用。MPAGE 给出最高 8 位地址，而寄存器 Ri 给出最低 8 位地址。

在一些 8051 的实现中，这种类型的 XDATA 访问是使用 P2 给出最高位地址执行的。因此现有的软件必须适应使用 MPAGE 而不是 P2。

MPAGE (0x93) – 内存页面选择

位	名称	复位	R/W	描述
7:0	MPAGE[7:0]	0x00	R/W	内存页面，MOVX 指令中地址的高位

2.2.5 存储器仲裁

存储器仲裁处理 CPU 和 DMA 访问所有物理存储器（除了 CPU 内部寄存器）。当 CPU 和 DMA 之间发生访问冲突时，存储器仲裁停止总线主机之一，这样冲突就被解决。

控制寄存器 MEMCTR 和 FMAP 用于控制存储器子系统的各个方面。MEMCTR 和 FMAP 寄存器描述如下。

MEMCTR.XMAP 必须设置以使得程序从 RAM 执行。

闪存区映射寄存器 FMAP 控制物理 32-KB 代码区映射到 CODE 存储空间的程序地址区域 0x8000–0xFFFF。

MEMCTR (0xC7) – 存储器仲裁控制

位	名称	复位	R/W	描述
7:4	—	0000	R0	保留
3	XMAP	0	R/W	XDATA 映射到代码。当设置了这一位，SRAM XDATA 区域从 0x0000 到(SRAM_SIZE-1)映射到 CODE 区域的 0x8000 到(0x8000 + SRAM_SIZE – 1)。这使得程序代码从 RAM 执行。 0：SRAM 映射到 CODE 功能禁用 1：SRAM 映射到 CODE 功能使能
2:0	XBANK[2:0]	000	R/W	XDATA 区选择。控制物理闪存存储器的哪个代码区映射到 XDATA 区域(0x8000–0xFFFF)。当设置为 0，映射到根部区。 有效设置取决于设备的闪存大小。写一个无效设置被忽略，即不会更新 XBANK[2:0]。 32-KB 版本：只能是 0（即总是映射到根部区） 64-KB 版本：0-1 128-KB 版本：0-3 256-KB 版本：0-7

FMAP (0x9F) – 闪存区映射

位	名称	复位	R/W	描述
7:3	—	0000 0	R0	未使用
2:0	MAP[2:0]	001	R/W	闪存区映射。控制物理闪存存储器的哪个代码区映射到 XDATA 区域(0x8000–0xFFFF)。当设置为 0，映射到根部区。有效设置取决于设备的闪存大小。写一个无效设置被忽略，即不会更新 MAP[2:0]。 32-KB 版本：不能写值。上面的区域只能用于从 SRAM 运行程度代码。见 MEMCTR.XMAP。 64-KB 版本：0-1 128-KB 版本：0-3 256-KB 版本：0-7

2.3 CPU 寄存器

本节描述了 CPU 的内部寄存器。

2.3.1 数据指针

有两个数据指针 DPTR0 和 DPTR1 来加快数据块到存储器/从存储器取出的移动。数据指针一般用于访问 CODE 或 XDATA 空间。例如：

MOVC A,@A+DPTR

MOV A,@DPTR

数据指针选择位（数据指针选择寄存器 DPS 中的位 0）选择执行一条使用数据指针的指令（即前面所述的指令之一）期间哪个数据指针是活动的。

数据指针宽度为两字节，包括以下 SFR：

- DPTR0–DPH0:DPL0
- DPTR1–DPH1:DPL1

DPH0 (0x83) – 数据指针-0 高字节

位	名称	复位	R/W	描述
7:0	DPH0[7:0]	0x00	R/W	数据指针-0，高字节

DPL0 (0x82) – 数据指针-0 低字节

位	名称	复位	R/W	描述
7:0	DPL0[7:0]	0x00	R/W	数据指针-0，低字节

DPH1 (0x85) – 数据指针-1 高字节

位	名称	复位	R/W	描述
7:0	DPH1[7:0]	0x00	R/W	数据指针-1，高字节

DPL1 (0x84) – 数据指针-1 低字节

位	名称	复位	R/W	描述
7:0	DPL1[7:0]	0x00	R/W	数据指针-1，低字节

DPS (0x92) – 数据指针选择

位	名称	复位	R/W	描述
7:1	–	0000 000	R0	未使用
0	DPS	0	R/W	数据指针选择。选择活动的数据指针。 0：DPTR0 1：DPTR1

2.3.2 寄存器 R0-R7

有 4 组寄存器（不要与 CODE 存储空间区混淆，它只适用于闪存存储器组织），每组包括 8 个寄存器。这四组寄存器分别映射到 DATA 存储空间地址的 0x00-0x07，0x08-0x0F，0x10-0x17，0x18-0x1F。每个寄存器组包括 8 个 8 位寄存器 R0-R7。寄存器组可以通过程序状态字 PSW.RS[1:0]来选择使用。寄存器组 0 使用内部触发器来存储值（SRAM 被绕过/未使用），而组 1-3 使用 SRAM 来存储。这样做是为了节省电力。一般地，通过使用寄存器组 0，而不是寄存器组 1—3，电流消耗大约可以下降 200uA。

2.3.3 程序状态字

程序状态字（PSW）按位显示 CPU 的当前状态，可以理解为一个可位寻址的特殊功能寄存器。PSW 如下所示，包括进位标志、BCD 操作的辅助进位标志、寄存器选择位、溢出标志和奇偶标志等。PSW 的其余二位没有规定，可用于用户自定义的状态标志。

PSW（0xD0）－程序状态字

位	名称	复位	R/W	描述
7	CY	0	R/W	进位标志。当最后的算术运算结果导致进位（加法期间）或者借用（减法期间）时，设置为1，否则通过所有算术运算清零。
6	AC	0	R/W	BCD运算的辅助进位标志。当最后的算术运算导致一个进位到（加法期间）或者借用（减法期间）时，设置为1，否则通过算术运算清零。
5	F0	0	R/W	用户自定义，位寻址
4:3	RS[1:0]	00	R/W	寄存器组选择位。在DATA空间，一组R7－R0寄存器使用了选择四个可能的寄存器组。 00：寄存器组0, 0x00 – 0x07 01：寄存器组1, 0x08 – 0x0F 10：寄存器组2, 0x10 – 0x17 11：寄存器组3, 0x18 – 0x1F
2	OV	0	R/W	溢出标志，通过算术运算设置。当最后的算述运算导致一个进位（加法），借位（减法），或者溢出（乘或除）时，设置为1，否则，该位通过所有运算清0。
1	F1	0	R/W	用户自定义，位寻址
0	P	0	R/W	奇偶校验标志,假如它包含一个奇数1的数，相同的累加器通过硬件设置为1。否则清0。

2.3.4 累加器

ACC 是一个累加器，它是大多数算法指令、数据传输和其它指令的源和目标地址。指令中累加器的助记符（涉及累加器的指令中）是 A 而不是 ACC。

ACC（0xE0）－累加器

位	名称	复位	R/W	描述
7:0	ACC[7:0]	0x00	R/W	累加器

2.3.5 B 寄存器

B 寄存器在执行乘法和除法指令期间作为第二个 8 位参数运用，若不进行乘/除法运算，B 寄存器也可当成一般寄存器使用，来存储临时数据。

B（0xF0）－ B 寄存器

位	名称	复位	R/W	描述
7:0	B[7:0]	0x00	R/W	B寄存器，用于 MUL/DIV指令

2.3.6 堆栈指针

堆栈驻留在 DATA 存储空间并向上增长。PUSH 指令执行时，首先把堆栈指针（SP）加 1，然后把字节复制到堆栈中。 SP 初始地址为 0x07，复位后就增加 1，变为 0x08，这是第二个寄存器组第一个寄存器（R0）的地址。因此为了使用多个寄存器组，SP 可以初始化到一块没有用于数据存储的位置。

SP（0x81）－堆栈指针

位	名称	复位	R/W	描述
7:0	SP[7:0]	0x07	R/W	堆栈指针

2.4 指令集总结

8051 指令集总结在表 2-3 中。所有助记符的版权归© Intel 公司，1980 年所有。

以下约定在指令集总结中使用：

- Rn – 当前选择寄存器组中的寄存器 R7-R0。
- direct – 8 位内部数据位置的地址。这可以是 DATA 区域（0x00 – 0x7F）或 SFR 区域（0x80 – 0xFF）。
- @Ri-8 位内部的位置，DATA 区域（0x00 – 0xFF），通过寄存器 R1 或 R0 间接寻址。
- #data – 指令所含的 8 位常量。
- #data16 – 指令所含的 16 位常量。
- addr16 – 16 位目标地址。用于 LCALL 和 LJMP。64-KB CODE 存储空间中任何地方都可以作为一行使用。
- addr11 – 11 位目标地址。用于 ACALL 和 AJMP。这一行将在程序存储器的同一个 2KB 的页面，作为下面指令的第一个字节。
- rel-（2 的补数）8 位偏移字节。用于 SJMP 和所有另外的跳转。范围是-128 到 127 字节，和下面指令的第一个字节有关。
- bit – DATA 区域或 SFR 区域的直接寻址位。

影响 CPU 标志设置的指令列在表 2-4 中，这些标志位于 PSW 中。注意对 PSW 寄存器或 PSW 位的操作也将影响标志的设置。还要注意许多指令的周期数假定是单周期访问存储器被访问的元素，即最好的情况。这并非总是如此。例如从闪存读可能需要 1-3 个周期。

表 2-3 指令集综述

助记符	描述	十六进制操作码	字节数	周期
算术运算				
ADD A,Rn	添加寄存器到累加器	28-2F	1	1
ADD A,direct	添加直接字节到累加器	25	2	2
ADD A,@Ri	添加直接RAM到累加器	26-27	1	2
ADD A,#data	添加直接数据到累加器	24	2	2
ADDC A,Rn	添加寄存器到带有进位标志的累加器	38-3F	1	1
ADDC A,direct	添加直接字节到带有进位标志的A	35	2	2
ADDC A,@Ri	添加直接RAM到带有进位标志的A	36-37	1	2
ADDC A,#data	添加直接数据到带有进位标志的A	34	2	2
SUBB A,Rn	从带有借位的A减去寄存器	98-9F	1	1
SUBB A,direct	从带有借位的A减法直接字节	95	2	2
SUBB A,@Ri	从带有借位的A减去间接RAM	96-97	1	2
SUBB A,#data	从带有借位的A减去直接数据	94	2	2
INC A	递增累加器	04	1	1
INC Rn	递增寄存器	08-0F	1	2
INC direct	递增直接字节	05	2	3
INC @Ri	递增直接RAM	06-07	1	3
INC DPTR	递增数据指针	A3	1	1
DEC A	递减累加器	14	1	1
DEC Rn	递减寄存器	18-1F	1	2
DEC direct	递减直接字节	15	2	3
DEC @Ri	递减直接RAM	16-17	1	3
MUL AB	A和B相乘	A4	1	5
DIV AB	A除以B	84	1	5
DA A	十进制校准累加器	D4	1	1
逻辑运算				
ANL A,Rn	AND寄存器到累加器	58-5F	1	1
ANL A,direct	AND直接字节到累加器	55	2	2
ANL A,@Ri	AND间接RAM到累加器	56-57	1	2

表 2-3 指令集综述（续表）

助记符	描述	十六进制操作码	字节数	周期
ANL A,#data	AND直接数据到累加器	54	2	2
ANL direct,A	AND累加器到直接字节	52	2	3
ANL direct,#data	AND直接数据到直接字节	53	3	4
ORL A,Rn	OR寄存器到累加器	48-4F	1	1
ORL A,direct	OR直接字节到累加器	45	2	2
ORL A,@Ri	OR间接RAM到累加器	46-47	1	2
ORL A,#data	OR直接数据到累加器	44	2	2
ORL direct,A	OR累加器到直接字节	42	2	3
ORL direct,#data	OR直接数据到直接字节	43	3	4
XRL A,Rn	专用OR寄存器到累加器	68-6F	1	1
XRL A,direct	专用OR直接字节到累加器	65	2	2
XRL A,@Ri	专用OR间接RAM到累加器	66-67	1	2
XRL A,#data	专用OR直接数据到累加器	64	2	2
XRL direct,A	专用OR累加器到直接字节	62	2	3
XRL direct,#data	专用OR直接数据到直接字节	63	3	4
CLR A	清除累加器	E4	1	1
CPL A	补充累加器	F4	1	1
RL A	累加器左循环	23	1	1
RLC A	累加器通过进位左循环	33	1	1
RR A	累加器右循环	03	1	1
RRC A	累加器通过进位右循环	13	1	1
SWAP A	在累加器里交换半字节	C4	1	1
数据传输				
MOV A,Rn	移动寄存器到累加器	E8-EF	1	1
MOV A,direct	移动直接字节到累加器	E5	2	2
MOV A,@Ri	移动间接RAM到累加器	E6-E7	1	2
MOV A,#data	移动直接数据到累加器	74	2	2
MOV Rn,A	移动累加器到寄存器	F8-FF	1	2
MOV Rn,direct	移动直接字节到寄存器	A8-AF	2	4
MOV Rn,#data	移动直接数据到寄存器	78-7F	2	2
MOV direct,A	移动累加器到直接字节	F5	2	3
MOV direct,Rn	移动寄存器到直接字节	88-8F	2	3
MOV direct1,direct2	移动直接字节到直接字节	85	3	4
MOV direct,@Ri	移动间接RAM到直接字节	86-87	2	4
MOV direct,#data	移动直接数据到直接字节	75	3	3
MOV @Ri,A	移动累加器到直接RAM	F6-F7	1	3
MOV @Ri,direct	移动直接字节到间接RAM	A6-A7	2	5
MOV @Ri,#data	移动直接数据到间接RAM	76-77	2	3
MOV DPTR,#data16	加载带有16位常数的数据指针	90	3	3
MOVC A,@A+DPTR	移动关于DPTR的代码字节到累加器	93	1	3
MOVC A,@A+PC	移动关于PC到的代码字节到累加器	83	1	3
MOVX A,@Ri	移动外部的RAM（8位地址）到A	E2-E3	1	3
MOVX A,@DPTR	移动外部的RAM (16位地址)到A	E0	1	3
MOVX @Ri,A	移动A到外部RAM (8位地址)	F2-F3	1	4
MOVX @DPTR,A	移动A到外部RAM (16位地址)	F0	1	4
PUSH direct	压直接字节到栈	C0	2	4

表 2-3 指令集综述（续表）

助记符	描述	十六进制操	字节数	周期
POP direct	从栈中弹出直接字节	D0	2	3
XCH A,Rn	交换寄存器和累加器	C8-CF	1	2
XCH A,direct	交换直接字节和累加器	C5	2	3
XCH A,@Ri	交换直接RAM和累加器	C6-C7	1	3
XCHD A,@Ri	交换间接低位半字节RAM和A	D6-D7	1	3
程序分支				
ACALL addr11	绝对子程序调用	xxx11	2	6
LCALL addr16	长期子程序调用	12	3	6
RET	从子程序返回	22	1	4
RETI	从中断返回	32	1	4
AJMP addr11	绝对跳转	xxx01	2	3
LJMP addr16	长期跳转	02	3	4
SJMP rel	短期跳转 (相对地址)	80	2	3
JMP @A+DPTR	相对 DPTR的间接跳转	73	1	2
JZ rel	如果累加器是0，跳转	60	2	3
JNZ rel	如果累加器不是0，跳转	70	2	3
JC rel	如果进位标志设置，跳转	40	2	3
JNC	如果进位标志未设置，跳转	50	2	3
JB bit,rel	如果直接位设置，跳转	20	3	4
JNB bit,rel	如果直接位未设置，跳转	30	3	4
JBC bit,direct rel	如果直接位设置并且清除位，跳转	10	3	4
CJNE A,direct rel	比较直接位和A，如果不等，跳转	B5	3	4
CJNE A,#data rel	直接字节和A相比较，如果不等，跳转	B4	3	4
CJNE Rn,#data rel	直接和Reg.相比较. 如果不等，跳转	B8-BF	3	4
CJNE @Ri,#data rel	直接和间接相比较，如果不等，跳转	B6-B7	3	4
DJNZ Rn,rel	递减寄存器，如果不为零，跳转	D8-DF	2	3
DJNZ direct,rel	递减直接字节，如果不为零，跳转	D5	3	4
NOP	无操作	00	1	1
布尔变量操作				
CLR C	清空进位标志	C3	1	1
CLR bit	清空直接位	C2	2	3
SETB C	设置进位标志	D3	1	1
SETB bit	设置直接位	D2	2	3
CPL C	补充进位标志	B3	1	1
CPL bit	补充直接位	B2	2	3
ANL C,bit	AND直接位到进位标志	82	2	2
ANL C,/bit	直接位到进位的AND补充	B0	2	2
ORL C,bit	OR直接位到进位标志	72	2	2
ORL C,/bit	直接位到进位的OR补充	A0	2	2
MOV C,bit	移动直接位到进位标志	A2	2	2
MOV bit,C	移动进位标志到直接位	92	2	3

表 2-4 影响标志设置的指令(1)

指令	CY	OV	AC
ADD	x	x	x
ADDC	x	x	x
SUBB	x	x	x
MUL	0	x	-
DIV	0	x	-
DA	x	-	-
RRC	x	-	-
RLC	x	-	-
SETB C	1	-	-
CLR C	x	-	-
CPL C	x	-	-
ANL C,bit	x	-	-
ANL C,/bit	x	-	-
ORL C,bit	x	-	-
ORL C,/bit	x	-	-
MOV C,bit	x	-	-
CJNE	x	-	-

（1）0=设置为 0，1=设置为 1，x=设置为 0/1，-=不影响

2.5 中断

CPU 有 18 个中断源。每个中断源都有它自己的位于一系列 SFR 寄存器中的中断请求标志。相应标志位请求的每个中断可以分别使能或禁用。中断源的定义和中断向量如表 2-5 所示。

中断分别组合为不同的、可以选择的优先级别。

中断使能寄存器在 2.5.1 节中描述。中断优先级设置在 2.5.3 节中描述。

2.5.1 中断屏蔽

每个中断请求可以通过设置中断使能 SFR 寄存器的中断使能位 IEN0，IEN1 或者 IEN2 使能或禁止。CPU 的中断使能 SFR 如下面描述并总结在表 2-5 中。

注意某些外部设备有若干事件，可以产生与外设相关的中断请求。这些中断请求可以作用在端口 0、端口 1、端口 2、定时器 1、定时器 2、定时器 3、定时器 4 和无线电上。对于每个内部中断源对应的 SFR 寄存器，这些外部设备都有中断屏蔽位。

为了使能任一中断功能，应当采取下列步骤：

1、清除中断标志。

2、如果有，则设置 SFR 寄存器中对应的各中断使能位为 1。

3、设置寄存器 IEN0、IEN1 和 IEN2 中对应的中断使能位为 1。

4、设置 IEN0 中的 EA 位为 1 使能全局中断。

5、在该中断对应的向量地址上，运行该中断的服务程序。关于地址见表 2-5。

图 2-4 给出了所有中断源以及相关控制和状态寄存器的完整概述。阴影框是调用中断服务例程时由硬件自动清除的中断。 表示一次执行，由于电平源或由于边沿形成。对于错过这个的中断作为电平触发对待（适

用于端口 P0、P1 和 P2）。转换框用默认状态显示，⎍或⎎表示上升或下降沿探测，即在什么时候中断产生。一般来说，如果可用的话，对于不是自动清除的标志，对于脉冲或边沿形成的中断源，应在清除源标志位之前清除 CPU 中断标志寄存器。对于电平源，必须在清除 CPU 标志之前清除源。

表 2-5　中断概览

中断号码	描述	中断名称	中断向量	中断屏蔽，CPU	中断标志，CPU
0	RF TX FIFO下溢或RX FIFO溢出	RFERR	03h	IEN0.RFERRIE	TCON.RFERRIF (1)
1	ADC转换结束	ADC	0Bh	IEN0.ADCIE	TCON.ADCIF (1)
2	USART0 RX完成	URX0	13h	IEN0.URX0IE	TCON.URX0IF (1)
3	USART1 RX完成	URX1	1Bh	IEN0.URX1IE	TCON.URX1IF (1)
4	AES加密/解密完成	ENC	23h	IEN0.ENCIE	S0CON.ENCIF
5	睡眠计时器比较	ST	2Bh	IEN0.STIE	IRCON.STIF
6	端口2输入/USB	P2INT	33h	IEN2.P2IE	IRCON2.P2IF (2)
7	USART0 TX完成	UTX0	3Bh	IEN2.UTX0IE	IRCON2.UTX0IF
8	DMA传送完成	DMA	43h	IEN1.DMAIE	IRCON.DMAIF
9	定时器1 (16位)捕获/比较/溢出	T1	4Bh	IEN1.T1IE	IRCON.T1IF (1) (2)
10	定时器2	T2	53h	IEN1.T2IE	IRCON.T2IF (1) (2)
11	定时器3 (8位) 捕获/比较/溢出	T3	5Bh	IEN1.T3IE	IRCON.T3IF (1) (2)
12	定时器 4 (8位)捕获/比较/溢出	T4	63h	IEN1.T4IE	IRCON.T4IF (1) (2)
13	端口0输入	P0INT	6Bh	IEN1.P0IE	IRCON.P0IF (2)
14	USART 1 TX完成	UTX1	73h	IEN2.UTX1IE	IRCON2.UTX1IF
15	端口1输入	P1INT	7Bh	IEN2.P1IE	IRCON2.P1IF (2)
16	RF通用中断	RF	83h	IEN2.RFIE	S1CON.RFIF (2)
17	看门狗计时溢出	WDT	8Bh	IEN2.WDTIE	IRCON2.WDTIF

（1）当调用中断服务例程时清除硬件。

（2）另外的 IRQ 掩码和 IRQ 标志位存在。

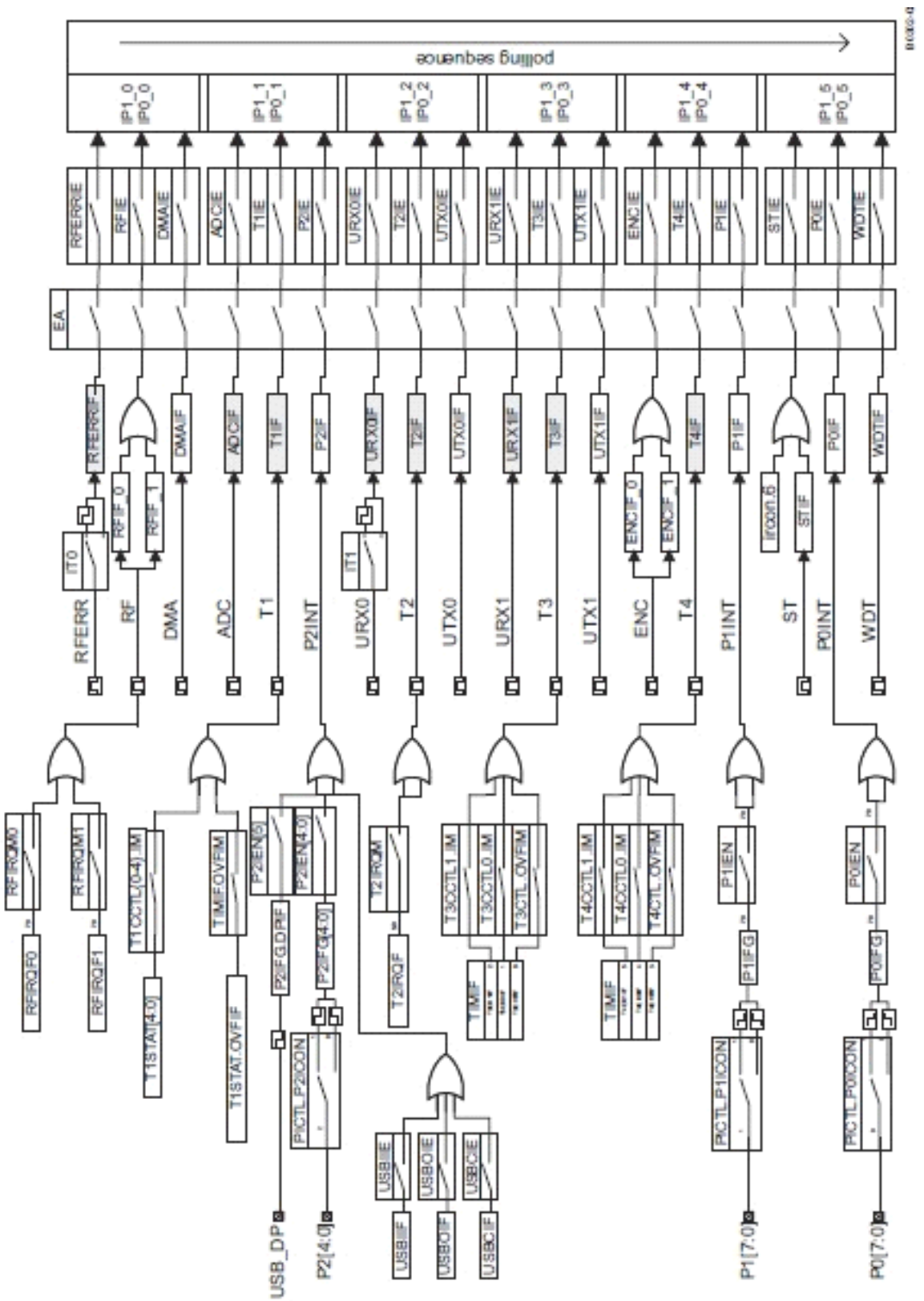

polling sequence
IP1_0 IP0_0
IP1_1 IP0_1
IP1_2 IP0_2
IP1_3 IP0_3
IP1_4 IP0_4
IP1_5 IP0_5
RFERRIE
RFIE
DMAIE
ADCIE
T1IE
P2IE
URX0IE
T2IE
UTX0IE
URX1IE
T3IE
UTX1IE
ENCIE
T4IE
P1IE
STIE
P0IE
WDTIE
EA
RFERRIF
RFIF_0
RFIF_1
DMAIF
ADCIF
T1IF
P2IF
URX0IF
T2IF
UTX0IF
URX1IF
T3IF
UTX1IF
ENCIF_0
ENCIF_1
T4IF
P1IF
irqon.6
STIF
P0IF
WDTIF
IT0
IT1
RFERR
RF
DMA
ADC
T1
P2INT
URX0
T2
UTX0
URX1
T3
UTX1
ENC
T4
P1INT
ST
P0INT
WDT
RFRQM0
RFRQM1
RFRQF0
RFRQF1
T1CCTL[0-4].IM
T1STAT[4:0]
TIMEOVFIM
T1STAT.OVFIF
P2IEN[5]
P2IEN[4:0]
P2IFGDPRF
P2IFG[4:0]
P2CTL.P2ICON
USB_DP
P2[4:0]
T2RQM
T2RQF
USBIE
USBOIE
USBCIE
USBIF
USBOIF
USBCIF
T3CCTL1.IM
T3CCTL0.IM
T3CTL.OVFIM
TIMIF
T4CCTL1.IM
T4CCTL0.IM
T4CTL.OVFIM
TIMIF
P1IEN
P1IFG
P1CTL.P1ICON
P1[7:0]
P0IEN
P0IFG
P0CTL.P0ICON
P0[7:0]

图 2-4 中断概览

IEN0 (0xA8) – 中断使能 0

位	名称	复位	R/W	描述
7	EA	0	R/W	禁用所有中断。 0: 无中断被确认 1: 通过设置对应的使能位将每个中断源分别使能和禁止
6	-	0	R0	不使用，读出来是 0
5	STIE	0	R/W	睡眠定时器中断使能 0: 中断禁止 1: 中断使能
4	ENCIE	0	R/W	AES 加密/解密中断使能 0: 中断禁止 1: 中断使能
3	URX1IE	0	R/W	USART 1 RX 中断使能 0: 中断禁止 1: 中断使能
2	URX0IE	0	R/W	USART0 RX 中断使能 0: 中断禁止 1: 中断使能
1	ADCIE	0	R/W	ADC 中断使能 0: 中断禁止 1: 中断使能
0	RFERRIE	0	R/W	RF TX/RX FIFO 中断使能 0: 中断禁止 1: 中断使能

IEN1 (0xB8) – 中断使能 1

位	名称	复位	R/W	描述
7:6	-	00	R0	不使用，读出来为0
5	P0IE	0	R/W	端口0中断使能 0: 中断禁止 1: 中断使能
4	T4IE	0	R/W	定时器4中断使能 0: 中断禁止 1: 中断使能
3	T3IE	0	R/W	定时器3中断使能 0: 中断禁止 1: 中断使能
2	T2IE	0	R/W	定时器2中断使能 0: 中断禁止 1: 中断使能
1	T1IE	0	R/W	定时器1中断使能 0: 中断禁止 1: 中断使能
0	DMAIE	0	R/W	DMA传输中断使能 0: 中断禁止 1: 中断使能

IEN2 (0x9A) – 中断使能 2

位	名称	复位	R/W	描述
7:6	-	00	R0	没有使用，读出来是0
5	WDTIE	0	R/W	看门狗定时器中断使能 0: 中断禁止 1: 中断使能
4	P1IE	0	R/W	端口1中断使能 0: 中断禁止 1: 中断使能
3	UTX1IE	0	R/W	USART 1 TX中断使能 0: 中断禁止 1: 中断使能
2	UTX0IE	0	R/W	USART 0 TX中断使能 0: 中断禁止 1: 中断使能
1	P2IE	0	R/W	端口2中断使能 0: 中断禁止 1: 中断使能
0	RFIE	0	R/W	RF一般中断使能 0: 中断禁止 1: 中断使能

2.5.2 中断处理

当中断发生时，CPU 就指向表 2-5 所描述的中断向量地址。一旦中断服务开始，就只能够被更高优先级的中断打断。中断服务程序由中断指令 RETI（从中断指令返回）终止，当 RETI 执行时，CPU 将返回到中断发生时的下一条指令。

当中断发生时，不管该中断使能或禁止，CPU 都会在中断标志寄存器中设置中断标志位。如果当设置中断标志时中断使能，那么在下一个指令周期，由硬件强行产生一个 LCALL 到对应的向量地址，运行中断服务程序。

中断的响应需要不同的时间，取决于该中断发生时 CPU 的状态。当 CPU 正在运行的中断服务程序，其优先级大于或等于新的中断时，新的中断暂不运行，直至新的中断的优先级高于正在运行的中断服务程序。在其他情况下，中断响应的时间取决于当前的指令，最快响应一个中断的时间是 7 个机器指令周期，其中 1 个机器指令周期用于探测中断，其余 6 个用来执行 LCALL。

注意： 如果一个中断被禁用且中断标志被轮询， 8051 汇编指令 JBC 不能用于轮询中断标志，当设置时要把它清除。如果使用了，中断标志可能立即重新生效。

TCON (0x88) – 中断标志

位	名称	复位	R/W	描述
7	URX1IF	0	R/WH0	USART 1 RX中断标志。当USART 1 RX中断发生时设为1且当CPU指向中断向量服务例程时清除。 0: 无中断未决 1: 中断未决
6	-	0	R/W	没有使用
5	ADCIF	0	R/WH0	ADC中断标志。ADC中断发生时设为1且CPU指向中断向量例程时清除。 0: 无中断未决 1: 中断未决
4	-	0	R/W	没有使用
3	URX0IF	0	R/WH0	USART 0 RX中断标志。当USART0中断发生时设为1且CPU指向中断向量例程时清除。 0: 无中断未决 1: 中断未决
2	IT1	1	R/W	保留。必须一直设为1。设置为零将使能低级别中断探测，几乎总是如此（启动中断请求时执行一次）。
1	RFERRIF	0	R/WH0	RF TX/RX FIFO 中断标志。当RFERR中断发生时设为1且CPU指向中断向量例程时清除。 0: 无中断未决 1: 中断未决
0	IT0	1	R/W	保留。必须一直设为1。设置为零将使能低级别中断探测，几乎总是如此（启动中断请求时执行一次）。

SOCON (0x98) – 中断标志 2

位	名称	复位	R/W	描述
7:2	-	0000 00	R/W	没有使用
1	ENCIF_1	0	R/W	AES中断。ENC有两个中断标志, ENCIF_1和ENCIF_0, 设置其中一个标志就会请求中断服务。当AES协处理器请求中断时两个标志都要设置。 0: 无中断未决 1: 中断未决
0	ENCIF_0	0	R/W	AES中断。ENC有两个中断标志, ENCIF_1和 ENCIF_0,, 设置其中一个标志就会请求中断服务。当AES协处理器请求中断时两个标志都要设置。 0: 无中断未决 1: 中断未决

S1CON (0x9B) – 中断标志 3

位	名称	复位	R/W	描述
7:2	-	0000 00	R/W	没有使用
1	RFIF_1	0	R/W	RF一般中断。RF有两个中断标志，RFIF_1和RFIF_0，设置其中一个标志就会请求中断服务。当无线设备请求中断时两个标志都要设置。 0: 无中断未决 1: 中断未决
0	RFIF_0	0	R/W	RF一般中断。RF有两个中断标志，RFIF_1和RFIF_0。设置其中一个标志就会请求中断服务。当无线电请求中断时两个标志都要设置。 0: 无中断未决 1: 中断未决

IRCON (0xC0) – 中断标志 4

位	名称	复位	R/W	描述
7	STIF	0	R/W	睡眠定时器中断标志 0: 无中断未决 1: 中断未决
6	-	0	R/W	必须写为0。写入1总是使能中断源。
5	P0IF	0	R/W	端口0中断标志 0: 无中断未决 1: 中断未决
4	T4IF	0	R/W H0	定时器4中断标志。当定时器4中断发生时设为1并且当CPU指向中断向量服务例程时清除。 0: 无中断未决 1: 中断未决
3	T3IF	0	R/W H0	定时器3中断标志。当定时器3中断发生时设为1并且当CPU指向中断向量服务例程时清除。 0: 无中断未决 1: 中断未决
2	T2IF	0	R/W H0	定时器2中断标志。当定时器2中断发生时设为1并且当CPU向量指向中断服务例程时清除。 0: 无中断未决 1: 中断未决
1	T1IF	0	R/W H0	定时器1中断标志。当定时器1中断发生时设为1并且当CPU向量指向中断服务例程时清除。 0: 无中断未决 1: 中断未决
0	DMAIF	0	R/W	DMA完成中断标志 0: 无中断未决 1: 中断未决

IRCON2 (0xE8) – 中断标志 5

位	名称	复位	R/W	描述
7:5	-	000	R/W	没有使用
4	WDTIF	0	R/W	看门狗定时器中断标志 0: 无中断未决 1: 中断未决
3	P1IF	0	R/W	端口1中断标志 0: 无中断未决 1: 中断未决
2	UTX1IF	0	R/W	USART 1 TX中断标志 0: 无中断未决 1: 中断未决
1	UTX0IF	0	R/W	USART 0 TX中断标志 0: 无中断未决 1: 中断未决
0	P2IF	0	R/W	端口2中断标志 0: 无中断未决 1: 中断未决

2.5.3 中断优先级

中断组合成为 6 个中断优先组，每组的优先级通过设置寄存器 IP0 和 IP1 实现。为了给中断（也就是它所在的中断优先组）赋值优先级，需要设置 IP0 和 IP1 的对应位，如表 2-6 所示。

中断优先级及其赋值的中断源显示在表 2-7 中。每组赋值为 4 个中断优先级之一。当进行中断服务请求时，不允许被较低级别或同级的中断打断。

当同时收到几个相同优先级的中断请求时，采取如同表 2-8 所列的轮流探测顺序来判定哪个中断优先响应。注意图 2-4 的轮询顺序是表 2-8 选择的算法，不是图中列在 IP 位之间的轮询。

IP1 (0xB9) – 中断优先级 1

位	名称	复位	R/W	描述
7:6	-	00	R/W	不使用
5	IP1_IPG5	0	R/W	中断第5组，优先级控制位1，参考表2-7：中断优先级组
4	IP1_IPG4	0	R/W	中断第4组，优先级控制位1，参考表2-7：中断优先级组
3	IP1_IPG3	0	R/W	中断第3组，优先级控制位1，参考表2-7：中断优先级组
2	IP1_IPG2	0	R/W	中断第2组，优先级控制位1，参考表2-7：中断优先级组
1	IP1_IPG1	0	R/W	中断第1组，优先级控制位1，参考表2-7：中断优先级组
0	IP1_IPG0	0	R/W	中断第0组，优先级控制位1，参考表2-7：中断优先级组

IP0 (0xA9) – 中断优先级 0

位	名称	复位	R/W	描述
7:6	-	00	R/W	没有使用
5	IP0_IPG5	0	R/W	中断第5组，优先级控制位0，参考表2-7：中断优先级组
4	IP0_IPG4	0	R/W	中断第4组，优先级控制位0，参考表2-7：中断优先级组
3	IP0_IPG3	0	R/W	中断第3组，优先级控制位0，参考表2-7：中断优先级组
2	IP0_IPG2	0	R/W	中断第2组，优先级控制位0，参考表2-7：中断优先级组
1	IP0_IPG1	0	R/W	中断第1组，优先级控制位0，参考表2-7：中断优先级组
0	IP0_IPG0	0	R/W	中断第0组，优先级控制位0，参考表2-7：中断优先级组

表 2-6　优先级设置

IP1_x	IP0_x	优先级
0	0	0 – 最低级别
0	1	1
1	0	2
1	1	3 – 最高级别

表 2-7　中断优先组

组	中断		
IPG0	RFERR	RF	DMA
IPG1	ADC	T1	P2INT
IPG2	URX0	T2	UTX0
IPG3	URX1	T3	UTX1
IPG4	ENC	T4	P1INT
IPG5	ST	P0INT	WDT

表 2-8　中断轮流探测顺序

中断向量编号	中断名称	
0	RFERR	轮流探测顺序 ↓
16	RF	
8	DMA	
1	ADC	
9	T1	
2	URX0	
10	T2	
3	URX1	
11	T3	
4	ENC	
12	T4	
11	ST	
5	P0INT	
6	P2INT	
7	UTX0	
14	UTX1	
15	P1INT	
17	WDT	

第 3 章

SWRU191–April 2009

调试接口

两线调试接口允许对片上闪存进行编程，可以访问存储器和寄存器内容，以及调试功能，比如断点、单步和寄存器修改。

调试接口使用 I/O 引脚 P2.1 和 P2.2 分别作为调试模式中的调试数据和调试时钟。当设备不在调试模式下，这些 I/O 引脚只能用于通用 I/O。因此调试接口不干预任何外设 I/O 引脚。

标题 页

3.1 调试模式

强制在引脚 P2.2（调试时钟）上进行两个上升沿传输，同时 RESET_N 输入保持低电平时，则进入调试模式。

当处于调试模式，P2.1 是调试数据的双向引脚，P2.2 是调试时钟输入引脚。

注意：注意调试器不能和一个划分过的系统时钟一起使用。当运行调试器时，当 CLKCONCMD.OSC=0 时 CLKCONCMD.CLKSPD 的值必须设置为 000，或者当 CLKCONCMD.OSC= 1 时设置为 001。

3.2 调试传输

调试接口使用一个类似于 SPI 的双线接口，包括 P2.1（调试数据）和 P2.2（调试时钟）引脚。数据在调试时钟的上升沿，双向调试数据引脚上传输，在这个时钟的下降沿进行采样。

调试数据引脚的方向取决于发出的命令。数据在调试时钟的上升沿驱动，在下降沿采样。图 3-1 显示了数据是如何采样的。

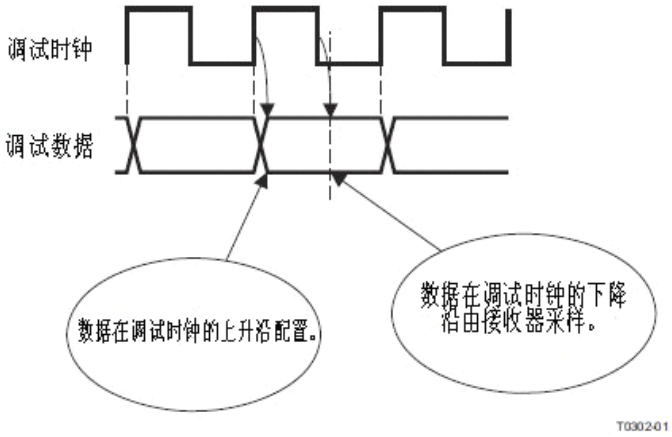


图 3-1 外部调试接口时序

数据是以字节为导向的，MSB 首先传输。一个字节的顺序如图 3-2。

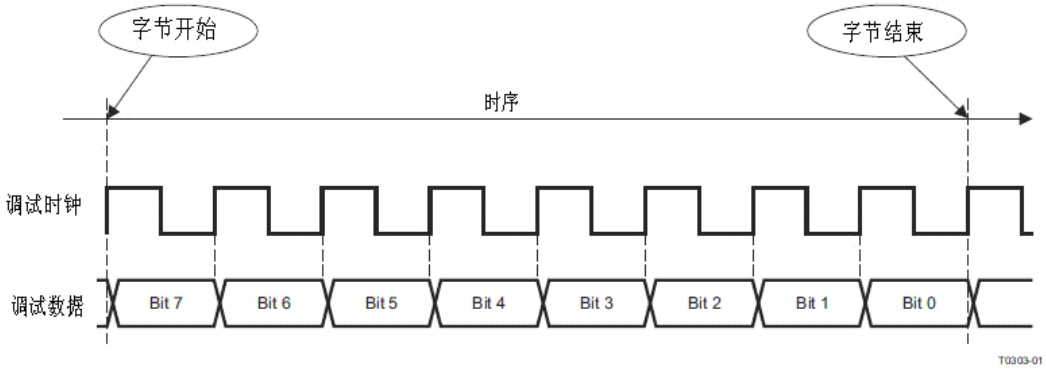


图 3-2 一个字节的传输

一个调试命令序列总是通过主机通过串行接口发送一个命令开始的。这一命令编码决定进一步参数应该含有的字节数，以及是否需要一个响应。基于这个命令，调试模块控制调试数据衬垫的方向。一个典型的命令序列如图 3-3。注意为了使图清楚，简化了调试数据信号，没有显示每个位的变化。方向不是对外部明确指明的，但是必须由命令协议的主机产生。

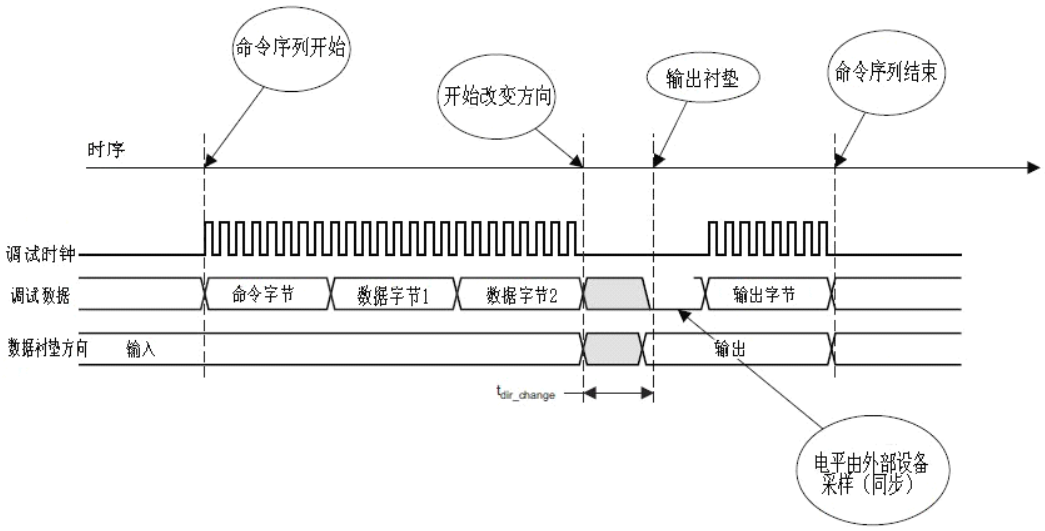


图 3-3 典型的命令序列——没有额外等待响应

对于需要响应的命令，必须在命令和响应之间有一个短暂的空闲期，以允许衬垫改变方向。最小等待时间（t_{dir_change}）过后，芯片通过下拉数据衬垫为低电平，表示是否准备好传输响应数据。对数据衬垫采样的外部调试器，探测到这一指示，开始记录响应数据。如果数据衬垫在等待时间过后是高电平，这向调试器表示芯片还没有准备好。图 3-4 显示等待是如何工作的。

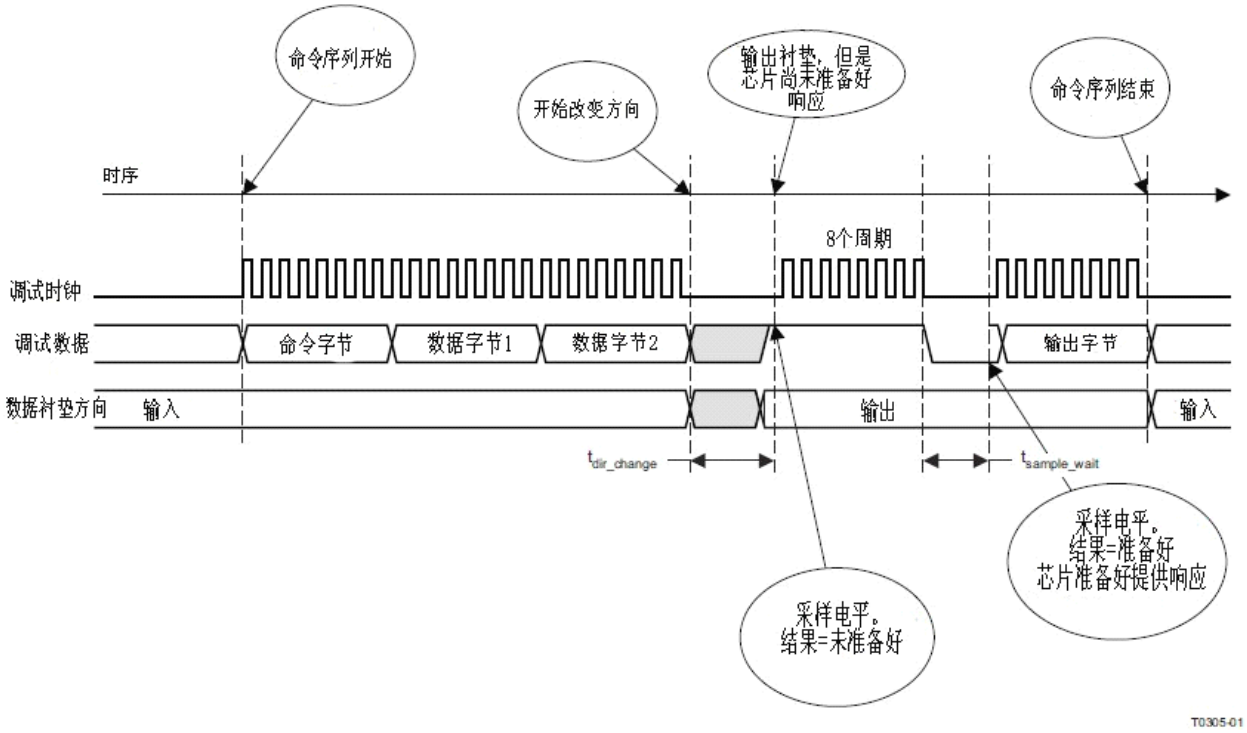


图 3-4 典型的命令序列。等待响应

如果调试接口通过上拉数据线为高电平表示它没有准备好返回数据，外部设备在重新采样准备好的电平之前，必须发出正好 8 个时钟脉冲。这必须重复直到电平变为低。等待周期相当于从调试接口读取一个字节，但是忽略结果。注意衬垫在调试时钟的下降沿开始改变方向。因此，衬垫驱动器不能驱动程序员的驱动器，直到程序员改变衬垫的方向。这个持续时间在一个程序实现中必须尽量使它最小。

3.3 调试命令

调试命令显示在表 3-2 中。以下章节详细描述了一些调试命令。

最低三位（即 X）无论什么值都可以。

3.4 锁位

为了软件和/或访问保护，可以写一组锁位到较高的可用闪存页面——锁页位面。锁位的结构包括 FLASH_PAGES-1 锁位，然后是一个调试锁位（见表 3-1）。该结构开始于锁位页面的地址 2032，占据多达 16 字节。锁位页面（地址 0-2031）的其余部分可以用于存储代码/常量，但是不进入调试模式就不能修改。

PAGELOCK[FLASH_PAGES-2:0]锁保护位用于为单独的闪存存储器页面（2 KB）使能擦除/写保护。每个可用的页面都有一个位。

当调试锁位 DBGLOCK,设置为 0（见表 3-1），除了 CHIP_ERASE、READ_STATUS 和 GET_CHIP_ID，所有调试命令都被禁用。调试锁位的状态可以使用 READ_STATUS 命令（见 3.4.2 节）读取。

注意由于一次写入锁位页面或一个 CHIP_ERASE 命令导致调试锁位被修改之后，设备必须复位来锁定/解锁调试接口。

发出一个 CHIP_ERASE 命令式清除调试锁位，从而解锁调试接口的唯一方法。

表 3-1 定义了包括闪存锁保护位的 16 字节的结构。第一个字节的位 0 包括页面 0 的锁位，第一个字节的位 1 包括页面 1 的锁位，等等。字节(FLASH_PAGES/8) – 1 的位 7 是 DBGLOCK 位。

表 3-1 闪存锁保护位结构定义

位	名称	定义
FLASH_PAGES-1	DBGLOCK	调试锁位 0：禁用调试命令 1：使能调试命令
FLASH_PAGES-2:0	PAGELOCK[FLASH_PAGES-1:0]	页面锁位。多达 128 个页面每一个都有一个位。 0：页面被锁定 1：页面没有被锁定

3.4.1 调试配置

命令 WR_CONFIG 和 RD_CONFIG 用于访问调试配置数据字节。这个配置数据的格式和描述显示在表 3-3 中。

3.4.2 调试状态

使用 READ_STATUS 命令读取调试状态字节。这个调试状态的格式和描述显示在表 3-4 中。

例如 READ_STATUS 命令用于以下：

- 在一个 CHIP_ERASE 命令后，轮询芯片擦除（CHIP_ERASE_BUSY）的状态。
- 检查振荡器是否稳定（OSCILLATOR_STABLE）；调试命令 HALT、RESUME、DEBUG_INSTR、STEP_REPLACE 和 STEP_INSTR 需要。

表 3-2 调试命令

命令	指令代码	输入字节	输出字节	描述
CHIP_ERASE	00010XXX	0	1	执行闪存芯片擦除(大量清除)和清除锁位。如果除了 READ_STATUS命令的其他命令发送，那么CHIP_ERASE的使用是禁止的。
WR_CONFIG	00011XXX	1.	1	写配置数据。详细信息参考表3-3。
RD_CONFIG	00100XXX	0	1	读配置数据。返回通过WR_CONFIG 命令设置的值。
GET_PC	00101XXX	0	2	返回16位程序计数器的值。无论在指令代码中位2的值都返回2字节。
READ_STATUS	00110XXX	0	1	读取状态字节，参考表3-4。
SET_HW_BRKPNT	00111XXX	3	1	设置硬件断点。
HALT	01000XXX	0	1	停止CPU的运行。
RESUME	01001XXX	0	1	恢复CPU的运行。该命令运行时CPU必须处于停止状态
DEBUG_INSTR	01010Xyy	1-3	1	运行调试指令。提供的指令执行时，CPU不增加程序计数器。该命令运行时CPU必须处于停止状态。注意yy是命令字节后面跟的字节数，即CPU指令有多少个字节（见表2-3）。
STEP_INSTR	01011XXX	0	1	步CPU的指令。CPU将执行来自程序存储器的下一个指令并且在执行后增加程序计数器。该命令运行时CPU必须处于停止状态。

表 3-2 调试命令（续表）

命令	指令	输入字节	输出字节	描述
GET_BM	01100XXX	0	1	该命令和GET_PC执行同样的动作，除了它返回存储器区。它返回一个字节，其中最低3位是当前使用的存储器区。
GET_CHIP_ID	01101XXX	0	2	返回16位芯片ID和版本号的值。无论指令代码位2的值都返回2个字节。
BURST_WRITE	10000kkk	2-2049	1	该命令写一个1-2048字节的序列到DBGDATA寄存器。每次寄存器被更新，就产生一个DBG_BW DMA触发。 BURST_WRITE命令的参数个数是可变的。突发脉冲的字节数使用命令（kkk）字节和整个下一字节的最后3位表明。命令序列如图3-5。突发脉冲的长度由一个11位值（b10-b0）表明。值0意思是2048；因此传输的最小字节数是1。

表 3-3 调试配置

位	名称	复位	描述
7:6	-	00	保留
5	SOFT_POWER_MODE	1	设置时，数字稳压器在PM2和PM3期间不关闭。如果这一位被清除，调试接口在PM2和PM3期间复位。
4	-	0	保留
3	TIMERS_OFF	0	禁用定时器。禁用计时器操作。这优先于TIMER_SUSPEND位及其函数。 0： 不禁用计时器 1： 禁用计时器
2	DMA_PAUSE	1	DMA暂停。当设置这个位，DMA寄存器不能被访问。 0： 使能DMA传输 1： 暂停所有DMA传输
1	TIMER_SUSPEND	1	挂起定时器。 当芯片停止挂起定时器。调试指令期间定时器操作也挂起。当执行一个STEP，如果程序是自由运行，定时器精确（或尽可能精确）接收尽可能多的标记。 0 不挂起定时器 1 挂起定时器
0	-	0	未使用。总是写入0。

表 3-4 调试状态

位	名称	描述符
7	CHIP_ERASE_BUSY	闪存芯片擦除繁忙 该信号仅在一个芯片擦除正在进行时为高电平。收到一个CHIP_ERASE命令后，它立即变为高，且闪存完全被擦除时返回低。 0： - 1： 芯片擦除正在进行
6	PCON_IDLE	PCON空闲 0： CPU正在运行 1： CPU空闲（选通时钟）
5	CPU_HALTED	CPU 停止 0： CPU 运行中 1： CPU 已停止

表 3-4 调试状态（续表）

位	状态	描述
4	POWER_MODE_0	供电模式0 0： 选择了供电模式1-3 1： 选择了供电模式0
3	HALT_STATUS	停止状态。返回CPU最后停止的原因。 0： CPU通过HALT调试命令停止 1： CPU通过软件或硬件断点停止
2	DEBUG_LOCKED	调试锁定。返回DBGLOCK位的值。 0： 调试接口没有锁定 1： 调试接口已被锁定
1	OSCILLATOR_STABLE	系统时钟振荡器稳定。 0： 振荡器不稳定 1： 振荡器稳定
0	STACK_OVERFLOW	堆栈溢出。当CPU写DATA存储空间的地址0xFF，该位代表可能发生堆栈溢出。 0 无堆栈溢出 1 堆栈溢出

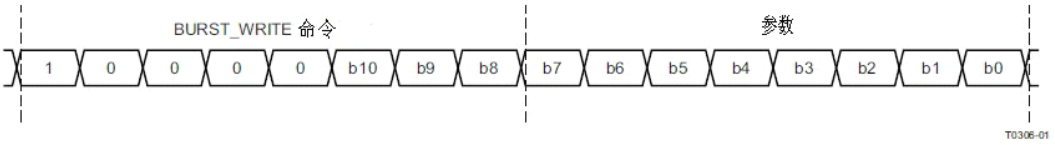


图 3-5 突发脉冲写命令（前 2 个字节）

3.4.3 硬件断点

调试命令 SET_HW_BRKPNT 用于设置四个可用的硬件断点之一。当使能一个硬件断点时，它将比较 CPU 地址总线和断点。当一个匹配发生时，CPU 停止。

当发出 SET_HW_BRKPNT 时，外部主机必须提供三个数据字节，来定义硬件断点。硬件断点本身包含 18 位，其中三位用于控制。SET_HW_BRKPNT 命令的三个数据字节如下。

第一个数据字节包含如下：

- 位 7-6：未使用
- 位 5-4：断点号码，0-3
- 位 3：1=使能，0=禁用
- 位 2-0：存储器区位。硬件断点的位 18-16 。

第二个数据字节包含硬件断点的位 15-8 。

第三个数据字节包含硬件断点的位 7-0 。因此第二和第三个字节设置执行停止在哪个 CPU CODE 地址。

3.4.4 闪存编程

片上闪存编程是通过调试接口执行的。外部主机必须首先使用 DEBUG_INSTR 调试命令使用闪存控制器发送指令，以执行闪存编程。

3.5 调试接口和供电模式

当芯片处于调试模式下，供电模式 PM2 和 PM3 可以以两种不同的方式处理。默认行为是从不关闭数字稳压器。这模拟了供电模式同时维护了调试模式操作。时钟源就像在普通供电模式下关闭。另一个选择是关闭 1.8V 内部数字电源。这导致数字部分完全关闭，即禁用了调试模式。当芯片处于调试模式，这两个选择由配置位 5（SOFT_POWER_MOD）控制。

当处于供电模式之一时，调试接口仍然响应一组精简的命令。芯片可以通过发出一个 HALT 命令给调试接口，从供电模式醒来。HALT 命令使芯片从供电模式醒来，处于停止状态。必须发出 RESUM 命令来恢复软件执行。

调试状态可以在处于供电模式的时候读取。当通过发出一个 HALT 命令离开一个供电模式时，必须检查状态。上电所需的时间取决于芯片处于哪种供电模式，且必须在调试状态中检查。在芯片可操作之前，调试接口只能接受供电模式中可用的命令。

注意：不支持空闲模式下的调试。调试时建议使用主动模式或其他供电模式。

3.6 寄存器

DBGDATA (0x6260) – 调试数据

位	名称	复位	R/W	描述
7:0	BYTE[7:0]	0	R	来自 BURST_WRITE 命令的调试数据。 该寄存器每使用 BURST_WRITE 命令传输一个新的字节到调试接口就更新。当这个字节被更新就产生一个 DBG_BW DMA 触发。这使 DMA 控制器可以取数据。

CHVER (0x6249) – 芯片版本

位	名称	复位	R/W	描述
7:0	VERSION[7:0]	取决于芯片	R	芯片版本号

CHIPID (0x624A) – 芯片 ID

位	名称	复位	R/W	描述
7:0	CHIPID[7:0]	取决于芯片	R	芯片标识符号 CC2530: 0xA5 CC2531: 0xB5

CHIPINFO0 (0x6276) – 芯片信息字节 0

位	名称	复位	R/W	描述
7	-	0	R0	未使用。总是 0。
6:4	FLASHSIZE[2:0]	取决于芯片	R	闪存大小。001 – 32 KB, 010 – 64 KB, 011 – 128 KB, 100 – 256 KB
3	USB	取决于芯片	R	如果芯片有 USB 是 1，否则是 0
2	-	1	R1	未使用。总是 0。
1:0	-	00	R0	未使用。总是 0。

CHIPINFO1 (0x6277) – 芯片信息字节 1

位	名称	复位	R/W	描述
7:3	-	0000 0	R0	未使用。总是 0。
2:0	SRAMSIZE[2:0]	取决于芯片	R	SRAM 大小，以 KB 大小减 1 为单位。例如一个 4-KB 设备的这个域设置为 011。这个数加 1 得到可用 KB 的数量。

第 4 章

SWRU191–April 2009

电源管理和时钟

低功耗运行是通过不同的运行模式（供电模式）使能的。各种运行模式指的是主动模式、空闲模式和供电模式 1、2 和 3（PM1-PM3）。超低功耗运行的实现通过关闭电源模块以避免静态（泄露）功耗，还通过使用门控时钟和关闭振荡器来降低动态功耗。

标题 页

4.1 电源管理简介

不同的运行模式或供电模式用于低功耗运行。超低功耗运行的实现通过关闭电源模块以避免静态（泄露）功耗，还通过使用门控时钟和关闭振荡器来降低动态功耗。

有五种不同的运行模式（供电模式），叫做主动模式、空闲模式、PM1、PM2 和 PM3。主动模式是一般模式，而 PM3 具有最低的功耗。不同的供电模式对系统运行的影响见图 4-1，并给出了稳压器和振荡器选择。

表 4-1 供电模式

供电模式	高频振荡器	低频振荡器	稳压器（数字）
配置	A 32 MHz XOSC B 16 MHz RCOSC	C 32 kHz XOSC D 32 kHz RCOSC	
主动/空闲模式	A 或 B	C 或 D	ON
PM1	无	C 或 D	ON
PM2	无	C 或 D	OFF
PM3	无	无	OFF

主动模式：完全功能模式。稳压器的数字内核开启，16 MHz RC 振荡器或 32 MHz 晶体振荡器运行，或者两者都运行。32 kHz RCOSC 振荡器或 32kHz XOSC 运行。

空闲模式：除了 CPU 内核停止运行（即空闲），其他和主动模式一样。

PM1：稳压器的数字部分开启。32 MHz XOSC 和 16 MHz RCOSC 都不运行。32 kHz RCOSC 或 32 kHz XOSC 运行。复位、外部中断或睡眠定时器过期时系统将转到主动模式。

PM2：稳压器的数字内核关闭。32 MHz XOSC 和 16 MHz RCOSC 都不运行。32kHz RCOSC 或 32 kHz XOSC 运行。复位、外部中断或睡眠定时器过期时系统将转到主动模式。

PM3：稳压器的数字内核关闭。所有的振荡器都不运行。复位或外部中断时系统将转到主动模式。

PM2/PM3 下 POR 是活跃的，但是 BOD 是掉电的，这给出了一个限制电压管理。如果 PM2/PM3 期间电压降至低于 1.4V，温度是 70°C 或更高，并且然后重新进入主动模式之前，回到合适的运行电压，寄存器和 RAM 在 PM2/PM3 下保存的内容可能会改变。因此，在设计系统电压时要小心，以确保这种情况不会发生。电压可以通过进入主动模式进行精确的定期监控，因为如果电压低于大约 1.7V 就触发一个 BOD 复位。

4.1.1 主动和空闲模式

主动模式是完全功能的运行模式，CPU、外设和 RF 收发器都是活动的。数字稳压器是开启的。

主动模式用于一般操作。在主动模式下（SLEEPCMD.MODE = 0x00）通过使能 PCON.IDLE 位，CPU 内核就停止运行，进入空闲模式。所有其他外设将正常工作，且 CPU 内核将被任何使能的中断唤醒（从空闲模式转换到主动模式）。

4.1.2 PM1

在 PM1 模式下，高频振荡器(32MHz XOSC 和 16MHz RCOSC)是掉电的。稳压器和使能的 32 kHz 振荡器是开启的。当进入 PM1 模式，就运行一个掉电序列。

由于 PM1 使用的上电/掉电序列较快，等待唤醒事件的预期时间相对较短（小于 3ms），就使用 PM1。

4.1.3 PM2

PM2 具有较低的功耗。在 PM2 下的上电复位时刻，外部中断、所选的 32 kHz 振荡器和睡眠定时器外设是活动的。I/O 引脚保留在进入 PM2 之前设置的 I/O 模式和输出值。所有其它内部电路是掉电的。稳压器也是关闭的。当进入 PM2 模式，就运行一个掉电序列。

当使用睡眠定时器作为唤醒事件，并结合外部中断时，一般就会进入 PM2 模式。相比较 PM1，当睡眠时间超过 3ms 时，一般选择 PM2。比起使用 PM1，使用较少的睡眠时间不会降低系统功耗。

4.1.4 PM3

PM3 用于获得最低功耗的运行模式。在 PM3 模式下，稳压器供电的所有内部电路都关闭（基本上是所有的数字模块，除了中断探测和 POR 电平传感）。内部稳压器和所有振荡器也都关闭。

复位（POR 或外部）和外部 I/O 端口中断是该模式下仅有的运行的功能。I/O 引脚保留进入 PM3 之前设置的 I/O 模式和输出值。复位条件或使能的外部 IO 中断事件将唤醒设备，使它进入主动模式（外部中断从它进入 PM3 的地方开始，而复位返回到程序执行的开始）。RAM 和寄存器的内容在这个模式下可以部分保留（见 4.6 节）。PM3 使用和 PM2 相同的上电/掉电序列。

当等待外部事件时，使用 PM3 获得超低功耗。当睡眠时间超过 3ms 时应该使用该模式。

4.2 电源管理控制

所需的供电模式通过使用 SLEEPCMD 控制寄存器的 MODE 位和 PCON.IDLE 位来选择。设置 SFR 寄存器的 PCON.IDLE 位，进入 SLEEPCMD.MODE 所选的模式。

来自端口引脚或睡眠定时器的使能的中断，或上电复位将从其他供电模式唤醒设备，使它回到主动模式。

当进入 PM1、PM2 或 PM3，就运行一个掉电序列。当设备从 PM1、PM2 或 PM3 中出来，它在 16 MHz 开始，如果当进入供电模式（设置 PCON.IDLE）且 CLKCONCMD.OSC = 0 时，自动变为 32 MHz。如果当进入供电模式设置了 PCON.IDLE 且 CLKCONCMD.OSC = 1，它继续运行在 16 MHz。

为了正确运行，设置 PCON.IDLE 位的指令必须遵循某种方式。这一指令后面跟的第一条汇编指令的第一个字节不能放在 4 字节边界。而且，缓存不能禁用（见 FCTL 寄存器描述的 CM）。不遵守这一要求可能导致较高的电流消耗。只要遵守了这一要求，设置了 PCON.IDLE 位的指令后面的第一条汇编指令在导致系统醒来的中断的 ISR 之前、但是系统醒来之后执行。如果这个指令是一个全局中断禁用，后面可以跟醒来之后、但是在 ISR 运行之前执行的代码。

IAR 编译器如何实现这一点的例子如下。设置 PCON 为 1 的命令位于写在汇编代码中的一个函数。调用此函数的一个 C 文件中，使用了这样的声明 extern void EnterSleepModeDisableInterruptsOnWakeup(void);。RSEG NEAR_CODE:CODE:NOROOT(2)语句确保 MOV PCON,#1 指令被放在一个 2 字节的边界。它是一个 3 字节的指令，这样下面的指令就不是放在 4 字节的边界，符合要求。在以下例子中，这一指令是 CLR EA，它禁用所有中断。这意味着，直到 IEN0.EA 位在代码中重新设置之后，才执行唤醒系统的中断的 ISR。如果不想要这个功能，可以使用 NOP 代替 CLR EA 指令。

```
PUBLIC    EnterSleepModeDisableInterruptsOnWakeup
    FUNCTION    EnterSleepModeDisableInterruptsOnWakeup,0201H
    RSEG    NEAR_CODE:CODE:NOROOT(2)
EnterSleepModeDisableInterruptsOnWakeup:
    MOV    PCON,#1
    CLR    EA
    RET
```

4.3 电源管理寄存器

本节描述了电源管理寄存器。在进入 PM2 或 PM3 时，所有寄存器位保留它们之前的值。

PCON (0x87) – 供电模式控制

位	名称	复位	R/W	描述
7:1	-	0000 000	R/W	未使用。总是写作 0000 000。
0	IDLE	0	R0/W H0	供电模式控制。写 1 到该位强制设备进入 SLEEP.MODE（注意 MODE=0x00 且 IDLE = 1 将停止 CPU 内核活动）设置的供电模式，这位读出来一直是 0。 当活动时，所有的使能中断将清除这个位，设备将重新进入主动模式。

SLEEPCMD (0xBE) – 睡眠模式控制

位	名称	复位	R/W	描述
7	OSC32K_CALDIS	0	R/W	禁用32 kHz RC振荡器校准 0：使能32 kHz RC振荡器校准 1：禁用32 kHz RC振荡器校准 这个设置可以在任何时间写入，但是在芯片运行在16MHz高频RC振荡器之前不起作用。
6:3	-	000 0	R0	保留
2	-	1	R/W	保留。总是写作1
1:0	MODE[1:0]	00	R/W	供电模式设置 00 ： 主动/空闲模式 01 ： 供电模式 1 10 ： 供电模式 2 11 ： 供电模式 3

SLEEPSTA (0x9D) – 睡眠模式控制状态

位	名称	复位	R/W	描述
7	OSC32K_CALDIS	0	R	32 kHz RC 振荡器校准状态 SLEEPSTA.OSC32K_CALDIS 显示禁用 32 kHz RC 校准的当前状态。在芯片运行在 32 kHz RC 振荡器之前，该位设置的值不等于 SLEEPCMD.OSC32K_CALDIS。这一设置可以在任何时间写入，但是在芯片运行在 16MHz 高频 RC 振荡器之前不起作用。
6:5	-	00	R	保留
4:3	RST[1:0]	XX	R	状态位，表示上一次复位的原因。如果有多个复位，寄存器只包括最新的事件。 00：上电复位和掉电探测 01：外部复位 10：看门狗定时器复位 11：时钟丢失复位
2:1	-	00	R	保留
0	CLK32K	0	R	32 kHz 时钟信号（与系统时钟同步）

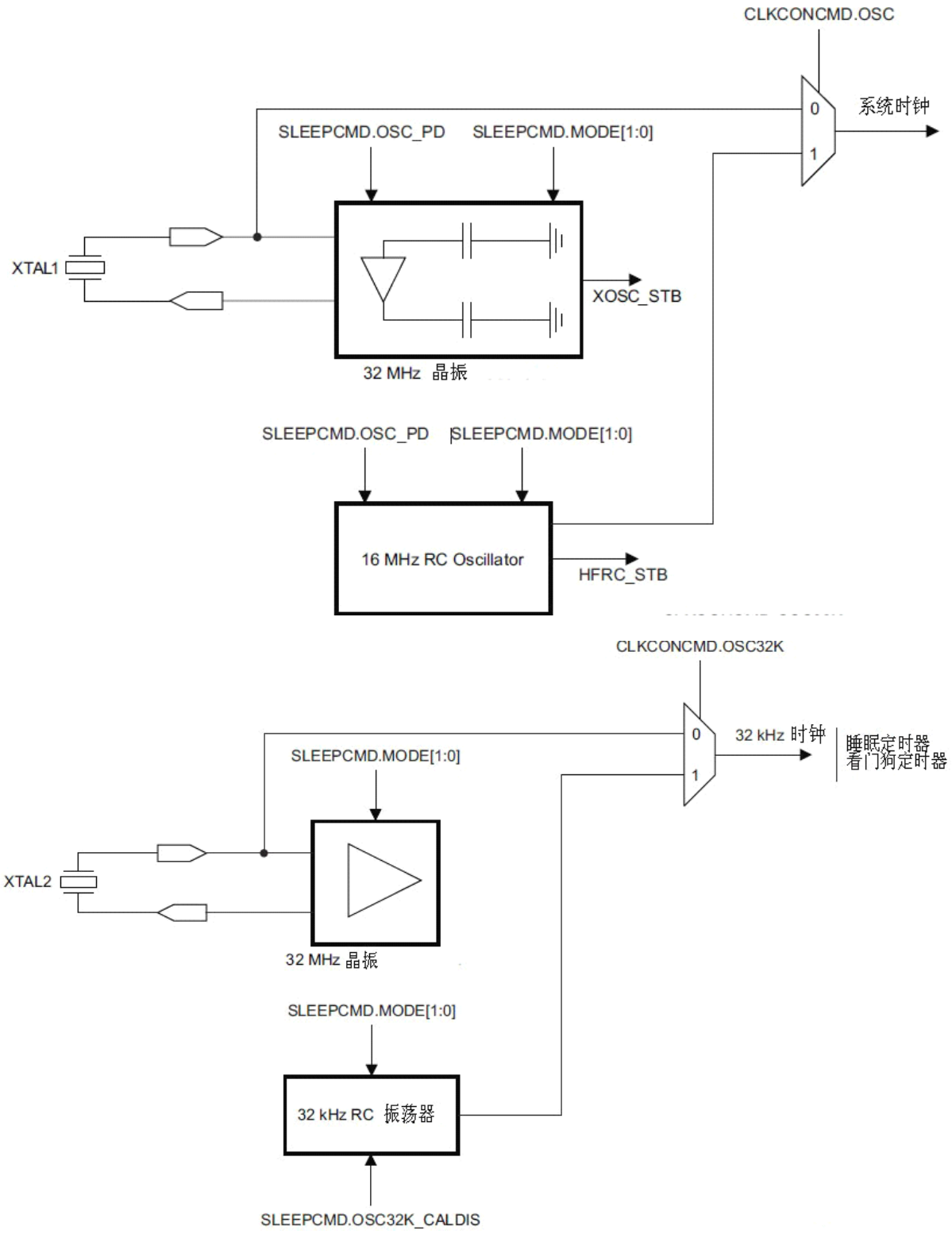

CLKCONCMD.OSC
0
1
系统时钟
SLEEPCMD.OSC_PD
SLEEPCMD.MODE[1:0]
XTAL1
XOSC_STB
32 MHz 晶振
SLEEPCMD.OSC_PD
SLEEPCMD.MODE[1:0]
16 MHz RC Oscillator
HFRC_STB
CLKCONCMD.OSC32K
0
1
32 kHz 时钟
睡眠定时器
看门狗定时器
SLEEPCMD.MODE[1:0]
XTAL2
32 MHz 晶振
SLEEPCMD.MODE[1:0]
32 kHz RC 振荡器
SLEEPCMD.OSC32K_CALDIS

图 4-1 系统时钟概览

4.4 振荡器和时钟

设备有一个内部系统时钟或主时钟。该系统时钟的源既可以用 16 MHzRC 振荡器，也可以采用 32 MHz 晶体振荡器。时钟的控制可以使用 CLKCONCMD SFR 寄存器执行。

还有一个 32 MHz 时钟源，可以是 RC 振荡器或晶振，也由 CLKCONCMD 寄存器控制。

CLKCONSTA 寄存器是一个只读的寄存器，用于获得当前时钟状态。

振荡器可以选择高精度的晶体振荡器，也可以选择低功耗的高频 RC 振荡器。注意，运行 RF 收发器，必须使用 32 MHz 晶体振荡器。

4.4.1 振荡器

图 4-1 概述了带有可用时钟源的时钟系统。

设备有两个高频振荡器：

- 32 MHz 晶振
- 16 MHz RC 振荡器

32 MHz 晶振启动时间对一些应用程序来说可能比较长，因此设备可以运行在 16 MHz RC 振荡器，直到晶振稳定。16 MHz RC 振荡器功耗少于晶振，但是由于不像晶振那么精确，不能用于 RF 收发器操作。

设备的两个低频振荡器：

- 32 kHz 晶振
- 32 kHz RC 振荡器

32 kHz XOSC 用于运行在 32.768 kHz，为系统需要的时间精度提供一个稳定的时钟信号。校准时 32 kHz RCOSC 运行在 32.753 kHz。校准只能发生在 32 kHz XOSC 使能的时候，这个校准可以通过使能 SLEEPCMD.OSC32K_CALDIS 位禁用。比起 32 kHz XOSC 解决方案，32 kHz RCOSC 振荡器应用于降低成本和电源消耗。这两个 32 kHz 振荡器不能同时运行。

4.4.2 系统时钟

系统时钟是从所选的主系统时钟源获得的，主系统时钟源可以是 32 MHz XOSC 或 16 MHz RCOSC。CLKCONCMD.OSC 位选择主系统时钟的源。注意要使用 RF 收发器，必须选择高速且稳定的 32 MHz 晶振。

注意改变 CLKCONCMD.OSC 位不会立即改变系统时钟。时钟源的改变首先在 CLKCONSTA.OSC = CLKCONCMD.OSC 的时候生效。这是因为在实际改变时钟源之前需要有稳定的时钟。还要注意 CLKCONCMD.CLKSPD 位反映系统时钟的频率，因此是 CLKCONCMD.OSC 位的映像。

选择了 32 MHz XOSC 且稳定之后，即当 CLKCONSTA.OSC 位从 1 变为 0，16 MHz RC 振荡器就被校准。

注意： 从 16 MHz 时钟变到 32 MHz 时钟源（反之亦然）与 CLKCONCMD.TICKSPD 的设置一致。当 CLKCONCMD.OSC 改变时，较慢的 CLKCONCMD.TICKSPD 设置导致实际源改变生效的时间较长。最快的转换是当 CLKCONCMD.TICKSPD 等于 000 时获得。

4.4.3 32 kHz 振荡器

设备的两个 32 kHz 振荡器作为 32 kHz 时钟的时钟源：

- 32 kHz XOSC
- 32 kHz RC RCOSC

默认复位后 32 kHz RCOSC 使能，被选为 32 kHz 时钟源。RCOSC 功耗较少，但是不如 32 kHz XOSC 精确。所选的 32 kHz 时钟源驱动睡眠定时器，为看门狗定时器产生标记，当计算睡眠定时器睡眠时间的时候用作定时器 2 的一个选通命令。选择哪个振荡器用作 32 kHz 时钟源是通过 CLKCONCMD.OSC32K 寄存器位执行的。

CLKCONCMD.OSC32K 寄存器位可以在任何时间写入，但是在 16 MHz RCOSC 成为活跃的系统时钟源之前不起作用。当系统时钟从 16 MHz RCOSC 转到 32 MHz XOSC（CLKCONCMD.OSC 从 1 到 0），32 kHz RCOSC 的校准开始，如果选择的是 32 kHz RCOSC 就开始执行。校准的结果是 32 kHz RCOSC 运行在 32.753 kHz。32 kHz RCOSC 可能需要 2ms 来完成。校准可以通过设置 SLEEPCMD.OSC32K_CALDIS 为 1 禁用。校准结束时，可能在 32 kHz 时钟源产生一个额外的脉冲，导致睡眠定时器增加 1。

注意转换到 32 MHz XOSC 后，当从 PM3 醒来且 32 MHz XOSC 使能，振荡器需要多达 500ms 来稳定在正确的频率。在 32 MHz XOSC 稳定之前，睡眠定时器、看门狗定时器和时钟丢失探测器不能使用。

4.4.4 振荡器和时钟寄存器

本节描述了振荡器和时钟寄存器。除非另有说明，进入 PM2 或 PM3 时所有寄存器位保留它们之前的值。

CLKCONCMD (0xC6) – 时钟控制命令

位	名称	复位	R/W	描述
7	OSC32K	1	R/W	32 kHz 时钟振荡器选择。设置该位只能发起一个时钟源改变。CLKCONSTA.OSC32K 反映当前的设置。当要改变该位必须选择 16 MHz RCOSC 作为系统时钟。 0 ： 32 kHz XOSC 1 ： 32 kHz RCOSC
6	OSC	1	R/W	系统时钟源选择。设置该位只能发起一个时钟源改变。CLKCONSTA.OSC 反映当前的设置。 0 ： 32 MHz XOSC 1 ： 16 MHz RCOSC
5:3	TICKSPD[2:0]	001	R/W	定时器标记输出设置。不能高于通过 OSC 位设置的系统时钟设置。 000 ： 32 MHz 001 ： 16 MHz 010 ： 8 MHz 011 ： 4 MHz 100 ： 2 MHz 101 ： 1 MHz 110 ： 500 kHz 111 ： 250 kHz 注意 CLKCONCMD.TICKSPD 可以设置为任意值，但是结果受 CLKCONCMD.OSC 设置的限制，即如果 CLKCONCMD.OSC=1 且 CLKCONCMD.TICKSPD=000，CLKCONCMD.TICKSPD 读出 001 且实际 TICKSPD 是 16 MHz。
2:0	CLKSPD	001	R/W	时钟速度。不能高于通过 OSC 位设置的系统时钟设置。表示当前系统时钟频率。 000： 32 MHz 001： 16 MHz 010： 8 MHz 011： 4 MHz 100： 2 MHz 101： 1 MHz 110： 500 kHz 111： 250 kHz 注意CLKCONCMD.CLKSPD可以设置为任意值，但是结果受CLKCONCMD.OSC设置的限制，即如果CLKCONCMD.OSC=1且CLKCONCMD.CLKSPD=000，CLKCONCMD.CLKSPD读出001且实际CLKSPD是16 MHz。 还要注意调试器不能和一个划分过的系统时钟一起工作。当运行调试器，当CLKCONCMD.OSC=0，CLKCONCMD.CLKSPD的值必须设置为000，或当CLKCONCMD.OSC=1设置为001。

CLKCONSTA (0x9E) – 时钟控制状态

位	名称	复位	R/W	描述
7	OSC32K	1	R	当前选择的 32 kHz 时钟源。 0 ： 32 kHz XOSC 1 ： 32 kHz RCOSC
6	OSC	1	R	当前选择的系统时钟。 0 ： 32 MHz XOSC 1 ： 16 MHz RCOSC
5:3	TICKSPD[2:0]	001	R	当前设置的定时器标记输出。 000 ： 32 MHz 001 ： 16 MHz 010 ： 8 MHz 011 ： 4 MHz 100 ： 2 MHz 101 ： 1 MHz 110 ： 500 kHz 111 ： 250 kHz
2:0	CLKSPD	001	R	当前时钟速度。 000： 32 MHz 001： 16 MHz 010： 8 MHz 011： 4 MHz 100： 2 MHz 101： 1 MHz 110： 500 kHz 111： 250 kHz

4.5 定时器标记产生

CLKCONCMD.TICKSPD 寄存器的值控制定时器 1、定时器 3 和定时器 4 的全局时钟划分。分频器值的设置可以从 0.25 MHz 到 32 MHz。注意如果 CLKCONCMD.TICKSPD 表示频率高于系统时钟，CLKCONSTA.TICKSPD 中指明的实际分频器值和系统时钟相同。

4.6 数据保留

在供电模式 PM2 和 PM3 下，从大部分内部电路中去除了电源。但是 SRAM 将保留它的部分内容，PM2 和 PM3 下内部寄存器的内容也保留。

除非另有指定一个给定的寄存器位域，保留其内容的寄存器是 CPU 寄存器、外设寄存器和 RF 寄存器。转换到 PM2 或 PM3 低功耗模式对软件是透明的。注意睡眠定时器的值不在 PM3 下保存。

复位

CC2430 有五个复位源。以下事件产生复位：

- 强制 RESET_N 输入引脚为低
- 上电复位条件
- 布朗输出复位条件
- 看门狗定时器复位条件
- 时钟丢失复位条件

复位之后初始条件如下：

- I/O 引脚配置为带上拉的输入（P1.0 和 P1.1 是输入，但是没有上拉/下拉）
- CPU 程序计数器装在 0x0000，程序执行从这个地址开始
- 所有外设寄存器初始化为各自复位值（参见寄存器描述）
- 看门狗定时器禁用
- 时钟丢失探测器禁用

复位期间，I/O 引脚配置为带上拉的输入（P1.0 和 P1.1 是输入，但是没有上拉/下拉）。

标题 页

5.1 上电复位和布朗输出探测器

设备包括一个上电复位(POR),在设备上电期间提供正确的初始化值。它还有一个布朗输出探测器(BOD),只能运行在规定的 1.8V 数字电压。BOD 在电压变化期间保护存储器内容，这会导致规定的 1.8V 电压下降，降低到数字逻辑、闪存存储器和 SRAM 所需的最低电平。

当使用最初电源，POR 和 BOD 将保持设备在复位状态，直到电压源上升超过上电复位和布朗输出探测电压。

最后一次复位的原因可以从寄存器位 SLEEPSTA.RST 中读出。应注意 BOD 复位将被读做 POR 复位。

5.2 时钟丢失探测器

时钟丢失探测器可以用于安全关键的系统来检测 XOSC 时钟源（32 MHz XOSC 或 32 kHz XOSC）之一的停止。这一般由于外部晶振或支持组件的损坏而发生。当时钟丢失探测器使能，这两个时钟监控器都不断运行。如果其中一个时钟停止切换，就在某个最大超时期间内产生一个时钟丢失复位。超时时间取决于哪个时钟停止。如果 32 kHz 时钟停止，超时时间是 0.5ms。如果 32 MHz 时钟停止，超时时间是 0.25ms。当系统从复位中重新恢复，软件可以通过读 SLEEPSTA.RST[1:0]探测到复位的原因。复位后，使用内部 RC 振荡器。因此，系统可以重新启动，然后可以从容掉电。时钟丢失探测器使用 CLD.EN 位使能/禁用。假定当使用时钟丢失探测器时，32 MHz XOSC 被选作系统时钟源。32 kHz 时钟可以使 32 kHz RCOSC（为了精确的复位超时时间应该校准）或 32 kHz XOSC。

在供电模式 1 和 2 下，时钟丢失探测器自动停止，当时钟重新启动时也重新启动。

在进入供电模式 3 之前，要转换到 16 MHz RCOSC 且禁用时钟丢失探测器。当重新进入主动模式，开启时钟丢失探测器，然后转换回 32 MHz XOSC。

CLD (0x6290) – 时钟丢失探测器

位	名称	复位	R/W	描述
7:1	-	0000 000	R0	保留
0	EN	0	R/W	时钟丢失探测器使能

闪存控制器

设备包含闪存存储器以存储程序代码。闪存存储器可以通过用户软件和调试接口进行编程。
闪存控制器处理写入和擦除嵌入式闪存存储器。嵌入式闪存存储器包括多达 128 页面，每页有 2048 个字节 。
闪存控制器有如下特性：

- 32 位字可编程
- 页面擦除
- 锁位，用于写入保护和代码安全
- 闪存页面擦除时间 20 ms
- 闪存芯片擦除时间 200 ms
- 闪存写入时间（4 字节）20 us

标题 页

6.1 闪存存储器组织

闪存存储器分为 2048 字节的闪存页面，闪存页面是存储器内可擦除的最小单元，而 32 位字是可以写入闪存的最小可写单元。

当执行写操作时，闪存存储器是字可寻址的，使用写入到地址寄存器 FADDRH:FADDRL 的一个 16 位地址。

执行页面擦除操作时，要被擦除的闪存页面通过寄存器位 FADDRH[7:1]寻址。

注意闪存存储器寻址的不同；当被 CPU 访问读取代码或数据时，闪存存储器是字节可寻址。当被闪存控制器访问时，闪存存储器是字可寻址，其中一个字由 32 位组成。

下节详细描述了闪存写和闪存页面擦除的步骤。

6.2 闪存写

闪存可以通过一个或多个 32 位字（4 字节）的序列连续编程，开始于起始地址（由 FADDRH:FADDRL 设置）。一般的，页面必须在写入开始之前擦除。页面擦除操作要设置页面中所有位为 1。芯片擦除命令（通过调试接口）擦除闪存中所有页面。这是设置闪存中的位为 1 的唯一方法。当写一个字到闪存，0 位可以编程为 0，1 位被忽略（闪存中这个位不改变）。因此，位被擦除为 1，可以被写为 0。可以写多次到一个字。这在 6.2.2 节描述。

6.2.1 闪存写步骤

闪存写的序列算法如下：

1. 设置 FADDRH:FADDRL 为起始地址（这是 18 位字节地址的 16 MSB）。
2. 设置 FCTL.WRITE 为 1。这启动写序列状态机制。
3. 在 20us 内写四次到 FWDATA（因为如果不首先反复，最后一次 FCTL.FULL 变为 0）。LSB 首先写入（最后一个字节后 FCTL.FULL 变为高）。
4. 等待直到 FCTL.FULL 变为低（闪存控制器已经开始编程 4 步骤 3 写入的字节，并准备好缓冲下 4 个字节）。
5. 可选的状态检查步骤：
   - 如果步骤 3 中的 4 字节写入不够快，操作超时，这一步的 FCTL.BUSY（和 FCTL.WRITE）是 0。
   - 如果由于页面被锁不能写入 4 字节到闪存，FCTL.BUSY（和 FCTL.WRITE）是 0，FCTL.ABORT 是 1。
6. 如果这是最后 4 字节，就退出，否则返回步骤 3。

写入操作可以使用以下两种方式之一执行：

- 使用 DMA 传输（首选方法）
- 使用 CPU，运行来自 SRAM 的代码

闪存写操作正在进行的时候，CPU 不能访问闪存，即读取程序代码。因此执行闪存写的程序代码必须从 RAM 执行。关于如何从 RAM 运行代码的描述见 2.2.1 节。

当从 RAM 执行一个闪存写操作，开始闪存写操作（FCTL.WRITE = 1）之后，CPU 继续从下一条指令执行代码。

当写闪存时不能进入供电模式 1、2 和 3。而且，写的时候系统时钟源（XOSC/RCOSC）不能改变。注意设置 CLKCONSTA.CLKSPD 为高值之后，就不可能符合写的时间的 20us 的时间要求。如果 CLKCONSTA.CLKSPD=111，时钟周期仅为 4us。因此写闪存时建议使 CLKCONSTA.CLKSPD 保持在 000 或 001。

6.2.2 写多次到一个字

以下规则适用于擦除之间写多次到一个 32 位字：

- 可以两次写 0 到一个 32 位字内的一个位。这不会改变该位的状态。
- 可以写 8 次到一个 32 位字
- 写 1 到一个位不会改变该位的状态

这使得可以 8 次写新的 4 位到一个 32 位字。表 6-1 展示了写序列到一个字的例子。因此 b_n 表示每次更新的写到字的新的 4 位。这一技术有益于使数据记录应用的闪存的寿命最大化。

表 6-1 写序列的例子

步骤	写入值	写后闪存内容	注释
1	（页面擦除）	0xFFFFFFFF	擦除设置所有位为 1。
2	0xFFFFFFFb_0	0xFFFFFFFb_0	只有写入 0 的位设置为 0，而所有写入 1 的位被忽略。
3	0xFFFFFFb_1F	0xFFFFFFb_1b_0	只有写入 0 的位设置为 0，而所有写入 1 的位被忽略。
4	0xFFFFFb_2FF	0xFFFFF$b_2b_1b_0$	只有写入 0 的位设置为 0，而所有写入 1 的位被忽略。
5	0xFFFFb_3FFF	0xFFFF$b_3b_2b_1b_0$	只有写入 0 的位设置为 0，而所有写入 1 的位被忽略。
6	0xFFFb_4FFFF	0xFFF$b_4b_3b_2b_1b_0$	只有写入 0 的位设置为 0，而所有写入 1 的位被忽略。
7	0xFFb_5FFFFF	0xFF$b_5b_4b_3b_2b_1b_0$	只有写入 0 的位设置为 0，而所有写入 1 的位被忽略。
8	0xFb_6FFFFFF	0xF$b_6b_5b_4b_3b_2b_1b_0$	只有写入 0 的位设置为 0，而所有写入 1 的位被忽略。
9	0xb_7FFFFFFF	0x$b_7b_6b_5b_4b_3b_2b_1b_0$	只有写入 0 的位设置为 0，而所有写入 1 的位被忽略。

如果每个数据样本多于 4 位，可以用连续的字来存储样本。例如，对于 16 位样本，第一个字存储位[3:0]，字 1 存储位[7:4]，字 2 存储位[11:8]，字 3 存储位[15:12]。当软件要读一个样本，可以从闪存中读四个字，合并为一个 16 位样本。这样，闪存的寿命可以最大化。

6.2.3 DMA 闪存写

当使用 DMA 写入操作时，要写入闪存的数据存储在 XDATA 存储空间（RAM 或 FLASH）中。一个 DMA 通道配置为从该存储源地址中读取要写入的数据，并把这个数据写入闪存写数据寄存器（FWDATA）固定的目标地址，DMA 触发事件 FLASH（DMA 配置中 TRIG[4:0]=10010）使能。因此当闪存写入数据寄存器 FWDATA 准备接收新数据时，闪存控制器将触发一个 DMA 传输。该 DMA 通道必须配置为执行单一模式，字节大小的传输，源地址设置为数据块的开始，目标地址设置位固定的 FWDATA（注意配置数据中的块大小 LEN 必须可以被 4 整除，否则，最后一个字不写入闪存）。还要保证 DMA 通道的高优先级，这样它不会在写的进程中中断。如果中断长于 20μs，写操作可能超时，写位 FCTL.WRITE 被设置为 0。

当 DMA 通道进入工作状态，通过设置 FCTL.WRITE 为 1 将触发第一个 DMA 传输（DMA 和闪存控制器处理传输的复位），开始一个闪存写。

图 6-1 的例子展示了一个 DMA 通道是如何配置的，以及如何发起一个 DMA 传输来写入 XDATA 中的一块数据到闪存存储器。

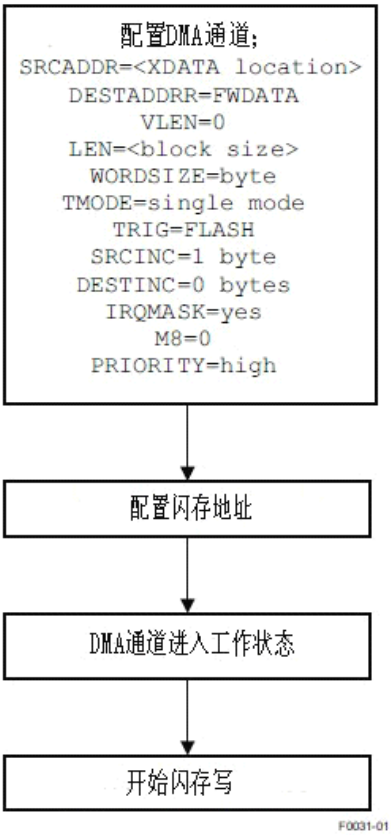


图 6-1 使用 DMA 的闪存写

6.2.4 CPU 闪存写

要使用 CPU 写闪存，必须从 SRAM 执行一个程序，且必须实现 6.2.1 节所列的步骤。禁用中断以确保操作不会超时。

6.3 闪存页面擦除

闪存页面擦除设置页面内的所有字节为 1。

通过设置 FCTL.ERASE 为 1 发起一次页面擦除。当发起一次页面擦除时，通过 FADDRH[7:1]寻址的页面将被擦除。注意如果页面擦除与页面写入同时发生，即 FCTL.WRITE 置 1，页面擦除将在页面写入操作之前执行。可以轮询 FCTL.BUSY 位，来看页面擦除是否已经完成。

当擦除一个页面时不能进入供电模式 1、2 和 3。而且，擦除的时候系统时钟源（XOSC/RCOSC）不能改变。

注意： 如果闪存页面擦除操作从闪存存储器内执行，且看门狗定时器使能，则必须选择长于 20ms（即闪存页面擦除操作的持续时间）的看门狗定时器间隔，这样 CPU 可以清除看门狗定时器。

6.3.1 从闪存存储器执行闪存擦除

注意，从闪存存储器中执行程序代码时，当发起一次闪存擦除或写操作，CPU 中止，当闪存控制器完成操作后，程序将从下一条指令继续执行。

以下代码例子是如何使用 IAR 编译器擦除一个闪存页面：

```
#include <ioCC2530.h>
unsigned char erase_page_num = 3;      /* 要擦除的页面编号，这里是闪存页面#3 */
/* 擦除一个闪存页面 */
EA = 0;                                /* 禁用中断 */
while (FCTL & 0x80);                   /* 轮询 FCTL.BUSY 并等待直到闪存控制器准备好 */
FADDRH = erase_page_num << 1;          /* 通过 FADDRH[7:1]位选择闪存页面*/
FCTL |= 0x01;                          /* 设置 FCTL.ERASE 位启动页面擦除 */
while (FCTL & 0x80);                   /*可选的：等待直到闪存写完成(~20 ms) */
EA = 1;                                /* 使能中断 */
```

6.4 闪存 DMA 触发

当要写入 FWDATA 寄存器的闪存数据已经写入到闪存存储器的指定位置，表示闪存控制器准备接受要写入 FWDATA 的新数据，激活闪存 DMA 触发。为了开始第一个传输，必须设置 FCTL.WRITE 位为 1。然后 DMA 和闪存控制器将为定义的数据块（DMA 配置中的 LEN）自动处理所有传输。更重要的是 DMA 在设置 FCTL.WRITE 位之前准备好进入工作状态，触发源设置为 FLASH（TRIG[4:0] = 10010），且 DMA 具有高优先级，这样传输不会被中断。如果中断长于 20 μ s，写操作将超时，且 FCTL.WRITE 位被清除。

6.5 闪存控制器的寄存器

本节描述了闪存控制器的寄存器。

FCTL (0x6270) – 闪存控制

位	名称	复位	R/W	描述
7	BUSY	0	R	代表写入或者擦除操作。当设置 WRITE 或 ERASE 位时设置该标志。 0： 没有活跃的写入或擦除操作 1： 有活跃的写入或擦除操作
6	FULL		R/H0	写缓存满状态。闪存写期间当 4 个字节已经被写入 FWDATA，设置该标志。写缓存满了不接受更多数据，即当设置 FULL 标志时写入 FWDATA 被忽略。当写缓存重新准备好接收 4 个更多字节，清除 FULL 标志。该标志仅在 CPU 用于写闪存时需要。 0：写缓存可以接受更多数据 1：写缓存满了
5	ABORT	0	R/H0	中止状态。当一个写操作或页面擦除中止时设置该位。当访问页面被锁时操作中止。当一个写或页面擦除开始清除中止位。
4	-	0	R	保留
3:2	CM[1:0]	01	R/W	缓存模式。 00：缓存禁用 01：缓存使能 10：缓存使能，预取模式 11：缓存使能，实时模式 缓存模式。禁用缓存会增加功耗，降低性能。预取对大多数应用程序提高了性能高达 33%，代价是可能增加了功耗。实时模式提供可预见的闪存读访问时间;执行时间等于缓存禁用模式下的时间，但是功耗较低。 *注意：读出的值总是代表当前缓存模式。写一个新的缓存模式启动一个缓存模式改变请求，可能需要一些时钟周期才能完成。如果有一个当前缓存改变请求正在进行，写这个寄存器被忽略。*
1	WRITE	0	R/W1/H0	写。开始在 FADDRH:FADDRL 给定的位置写字。WRITE 位保持 1 直到写完成。清除该位表示擦除已经完成，即已经超时或中止。 如果 ERASE 也设置为 1，在写之前执行 FADDRH[7:1]寻址的整个页面的一个页面擦除。当 ERASE 是 1，设置 WRITE 为 1 不起作用。
0	ERASE	0	R/W1/H0	页面擦除。擦除通过 FADDRH[7:1]给出的页。ERASE 位保持 1 直到写完成。清除该位表示擦除已经完成，即已经超时或中止。 当 WRITE 是 1，设置 ERASE 为 1 不起作用。

FWDATA (0x6273) – 闪存写数据

位	名称	复位	R/W	描述
7:0	FWDATA[7:0]	0x00	R0/W	闪存写数据。当FCTL.WRITE为1时才能写该寄存器。

FADDRH (0x6272) – 闪存地址高字节

位	名称	复位	R/W	描述
7:0	FADDRH[7:0]	0x00	R/W	页面地址/闪存字地址的高位字节 位[7:1]将选择要访问的页面

FADDRL (0x6271) – 闪存地址高字节

位	名称	复位	R/W	描述
7:0	FADDRL[7:0]	0x00	R/W	页面地址/闪存字地址的低位字节

I/O 端口

有 21 个数字输入/输出引脚，可以配置为通用数字 I/O 或外设 I/O 信号，配置为连接到 ADC、定时器或 USART 外设。这些 I/O 口的用途可以通过一系列寄存器配置，由用户软件加以实现。

I/O 端口具备如下重要特性：

- 21 个数字 I/O 引脚
- 可以配置为通用 I/O 或外部设备 I/O
- 输入口具备上拉或下拉能力
- 具有外部中断能力。

21 个 I/O 引脚都可以用作于外部中断源输入口。因此如果需要外部设备可以产生中断。外部中断功能也可以从睡眠模式唤醒设备。

标题 页

7.1 未使用的 I/O 引脚

未使用的 I/O 引脚电平是确定的，不能悬空。一个方法是使引脚不连接，配置引脚为具有上拉电阻的通用I/O输入。这也是所有引脚复位后的状态（除了 P1.0 和 P1.1 没有上拉/下拉功能）。或者引脚可以配置为通用 I/O 输出。这两种情况下引脚都不能直接连接到 VDD 或 GND，以避免过多的功耗。

7.2 低 I/O 电压

在数字 I/O 电压引脚 DVDD1 和 DVDD2 低于 2.6V 的应用中，寄存器位 PICTL.PADSC 应设置为 1，以获得 *DC 特性*表中所述的输出 DC 特性。

7.3 通用 I/O

用作通用 I/O 时，引脚可以组成 3 个 8 位端口，端口 0、端口 1 和端口 2，表示为 P0、P1 和 P2。其中，P0 和 P1 是完全的 8 位端口，而 P2 仅有 5 位可用。所有的端口均可以通过 SFR 寄存器 P0、P1 和 P2 位寻址和字节寻址。每个端口引脚都可以单独设置为通用 I/O 或外部设备 I/O。

除了两个高驱动输出口 P1.0 和 P1.1 各具备 20 mA 的输出驱动能力之外，所有的输出均具备 4 mA 的驱动能力。

寄存器 PxSEL，其中 x 为端口的标号 0~2，用来设置端口的每个引脚为通用 I/O 或者是外部设备 I/O 信号。作为缺省的情况，每当复位之后，所有的数字输入/输出引脚都设置为通用输入引脚。

在任何时候，要改变一个端口引脚的方向，就使用寄存器 PxDIR 来设置每个端口引脚为输入或输出。因此只要设置 PxDIR 中的指定位为 1，其对应的引脚口就被设置为输出了。

当读取端口寄存器 P0、P1 和 P2 的值，不管引脚配置如何，输入引脚上的逻辑值都被返回。这在执行读-修改-写指令期间不适用。读-修改-写指令是：ANL，ORL，XRL，JBC，CPL，INC，DEC，DJNZ，MOV，CLR 和 SETB。在一个端口寄存器上操作，以下是正确的：当目标是端口寄存器 P0、P1 或 P2 中一个独立的位，寄存器的值，而不是引脚上的值，被读取、修改并写回端口寄存器。

用作输入时，通用 I/O 端口引脚可以设置为上拉、下拉或三态操作模式。作为缺省的情况，复位之后，所有的端口均设置为带上拉的输入。要取消输入的上拉或下拉功能，就要将 PxINP 中的对应位设置为 1。I/O 端口引脚 P1.0 和 P1.1 没有上拉/下拉功能。注意配置为外设 I/O 信号的引脚没有上拉/下拉功能，即使外设功能是一个输入。

在电源模式 PM1、PM2 和 PM3 下 I/O 引脚保留当进入 PM1/PM2/PM3 时设置的 I/O 模式和输出值（如果可用的话）。

7.4 通用 I/O 中断

通用 I/O 引脚设置为输入后，可以用于产生中断。中断可以设置在外部信号的上升或下降沿触发。P0、P1 或 P2 端口都有中断使能位，对位于 IENl-2 寄存器内的端口所有的位都是公共的，如下：

- IENI.P0 IE：P0 中断使能
- IEN2.PI IE：P1 中断使能
- IEN2.P2IE：P2 中断使能

除了这些公共中断使能之外，每个端口的位都有位于 SFR 寄存器 P0IEN、P1IEN 和 P2IEN 的单独的中断使能。即使配置为外设 I/O 或通用输出的 I/O 引脚使能时都有中断产生。

当中断条件发生在 I/O 引脚之一上面，P0-P2 中断标志寄存器 P0IFG、P1IFG 或 P2IFG 中相应的中断状态标志将设置为 1。不管引脚是否设置了它的中断使能位，中断状态标志都被设置。当中断已经执行，中断状态标志被清除，该标志写入 0。这个标志必须在清除 CPU 端口中断标志（PxIF）之前被清除。

用于中断的 SFR 寄存器描述在下一节。寄存器总结如下：

- P0IEN: P0 中断使能
- P1IEN: P1 中断使能
- P2IEN: P2 中断使能
- PICTL: P0、P1 和 P2 触发沿设置
- P0FG: P0 中断标志
- P1IFG: P1 中断标志
- P2IFG: P2 中断标志

7.5 通用 I/O DMA

当用作通用 I/O 引脚时，每个 P0 和 P2 端口都关联一个 DMA 触发。这些 DMA 触发对于 P0 为 IOC_0，对于 P1 为 IOC_1，如表 8-1 所示。

当一个中断发生在 P0 引脚时 IOC_0 触发是被激活的。当一个中断发生在 P1 引脚时 IOC_1 触发是被激活的。

7.6 外设 I/O

本节描述了数字 I/O 引脚是如何配置为外设 I/O 的。对于可以通过数字输入/输出引脚和外部系统接口的每个外设单元，如何配置外设 I/O 的描述给定在以下小节中。

对于 USART 和定时器 I/O，在一个数字 I/O 引脚上选择外设 I/O 功能，需要设置对应的 PxSEL 位为 1。

注意，该外部单元具有两个可以选择的位置对应它们的 I/O 引脚，参见表 7-1。如果有关于 I/O 映射的冲突设置，可以在这些之间设置优先级（使用 P2SEL.PRIxP1 和 P2DIR.PRIP0 位）。所有不会导致冲突的组合都可以使用。

注意即使没有使用，外设一般也会出现在选定的位置，使用引脚的其他外设必须给予较高的优先级。例外情况是流量控制禁用时 UART 模式下 USART 的 RTS 和 CTS 引脚， 以及 SPI 主模式下 USART 配置的 SSN 引脚。

还要注意不管 PxINP 的设置，有输入引脚的外设单元是从引脚接收输入，这可能会影响外设单元的状态。例如如果 RX 引脚在用作一个 UART 引脚之前，可能已经有活动， UART 在使用之前必须被清除。

表 7-1 外设 I/O 引脚映射

外设/功能	P0								P1								P2				
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	4	3	2	1	0
ADC	A7	A6	A5	A4	A3	A2	A1	A0													T
USART0 SPI			C	SS	M0	MI															
Alt. 2											M0	MI	C	SS							
USART0 UART			RT	CT	TX	RX															
Alt. 2											TX	RX	RT	CT							

表 7-1 外设 I/O 引脚映射（续表）

外设/功能	P0								P1								P2				
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	4	3	2	1	0
USART 1 SPI			M1	M0	C	SS															
Alt. 2									M1	M0	C	SS									
USART 1 UART			RX	TX	RT	CT															
Alt. 2									RX	TX	RT	CT									
TIMER1		4	3	2	1	0															
Alt. 2	3	4												0	1	2					
TIMER3												1	0								
Alt. 2									1	0											
TIMER4															1	0					
Alt. 2																		1			0
32 kHz XOSC																	Q1	Q2			
DEBUG																			DC	DD	

7.6.1 定时器 1

PERCFG.T1CFG 选择是否使用备用位置 1 或备用位置 2。

在表 7-1 中，定时器 1 的信号显示如下：

- 0：通道 0 捕获/比较引脚
- 1：通道 1 捕获/比较引脚
- 2：通道 2 捕获/比较引脚
- 3：通道 3 捕获/比较引脚
- 4：通道 4 捕获/比较引脚

P2DIR.PRIP0 选择为端口 0 指派一些外设的优先顺序。当设置为 10，定时器通道 0-1 优先，当设置为 11，定时器通道 2-3 优先。要所有定时器 1 通道在备用位置 1 上可见，移动 USART 0 和 USART 1 到备用位置 2。

P2SEL.PRI1P1 和 P2SEL.PRI0P1 选择为端口 1 指派一些外设的优先顺序。当前者设置为低电平而后者设置为高电平时，定时器 1 通道优先。

7.6.2 定时器 3

PERCFG.T3CFG 选择是否使用备用位置 1 或备用位置 2。

在表 7-1 中，定时器 3 的信号显示如下：

- 0：通道 0 比较引脚
- 1：通道 1 比较引脚

P2SEL.PRI2P1 和 P2SEL.PRI3P1 选择为端口 1 指派一些外设的优先顺序。当这两个位都设置为高电平时，定时器 3 通道优先。如果 P2SEL.PRI2P1 设置为高电平且 P2SEL.PRI3P1 设置为低电平时，定时器 3 通道优先于 USART 1，但是 USART 0 优先于定时器 3 通道以及 USART 1。

7.6.3 定时器 4

PERCFG.T4CFG 选择是否使用备用位置 1 或备用位置 2。

在表 7-1 中，定时器 4 的信号显示如下：

- 0：通道 0 比较引脚
- 1：通道 1 比较引脚

P2SEL.PRI1P1 选择为端口 1 指派一些外设的优先顺序。当这个位设置时，定时器 4 通道优先。

7.6.4 USART 0

SFR 寄存器位 PERCFG.U0CFG 选择是否使用备用位置 1 或备用位置 2。

在表 7-1 中，USART 0 信号显示如下：

UART：

- RX：RXDATA
- TX：TXDATA
- RT：RTS
- CT：CTS

SPI：

- MI：MISO
- MO：MOSI
- C：SCK
- SS：SSN

P2DIR.PRIP0 选择为端口 0 指派一些外设的优先顺序。当设置为 00 时，USART 0 优先。注意如果选择了 UART 模式，且硬件流量控制禁用，UART 1 或定时器 1 将优先使用端口 P0.4 和 P0.5。

P2SEL.PRI3P1 和 P2SEL.PRI0P1 选择为端口 1 指派一些外设的优先顺序。当它们两个都设置为 0 时，USART 0 优先。注意如果选择了 UART 模式，且硬件流量控制禁用，定时器 1 或定时器 3 将优先使用端口 P1.2 和 P1.3。

7.6.5 USART 1

SFR 寄存器位 PERCFG.U1CFG 选择是否使用备用位置 1 或备用位置 2。

在表 7-1 中，USART 1 信号显示如下：

UART：

- RX：RXDATA
- TX：TXDATA
- RT：RTS
- CT：CTS

SPI:

- MI：MISO
- MO：MOSI
- C：SCK
- SS: SSN

P2DIR.PRIP0 选择为端口 0 指派一些外设时的优先顺序。当设置为 01，USART 1 优先。注意如果选择了 UART 模式，且硬件流量控制禁用，USART 0 或定时器 1 将优先使用 P0.2 和 P0.3。

P2SEL.PRI3P1 和 P2SEL.PRI2P1 选择为端口 1 指派一些外设的优先顺序。当前者设置为 1 而后者设置为 0 时，USART 1 优先。注意如果选择了 UART 模式，且硬件流量控制禁用，USART 0 或定时器 3 将优先使用 P1.4

和 P1.5。

7.6.6 ADC

当使用 ADC 时，端口 0 引脚必须配置为 ADC 输入。可以使用多达八个 ADC 输入引脚。要配置一个端口0引脚为一个 ADC 输入，APCFG 寄存器中相应的位必须设置为 1。这个寄存器的默认值选择端口 0 引脚为非 ADC 输入，即数字输入/输出。

APCFG 寄存器的设置将覆盖 P0SEL 的设置。

ADC 可以配置为使用通用 I/O 引脚 P2.0 作为内部触发器来启动转换。当用作 ADC 内部触发器时，P2.0 必须在输入模式下配置为通用 I/O。

7.7 调试接口

端口 P2.1 和 P2.2 分别用于调试数据和时钟信号。这些显示为表 7-1 中的 DD（调试数据）和 DC（调试时钟）。当处于调试模式，调试接口控制这些引脚的方向。当处于调试模式在这些引脚上禁用上拉/下拉。

7.8 32 kHz XOSC 输入

端口 P2.3 和 P2.4 用于连接一个外部 32 kHz 晶振。当 CLKCONCMD.OSC32K 是低电平时，不管寄存器设置如何，这些端口引脚由 32 kHz XOSC 使用。当 CLKCONCMD.OSC32K 是低电平，这些端口引脚将设置在模拟模式。

7.9 无线测试输出信号

通过使用 OBSSELx 寄存器（OBSSEL0-OBSSEL5），用户可以从 RF 内核输出不同的信号到 GPIO 引脚。这些信号可以用于调试低级别的协议或控制外部 PA、LNA 或交换机。控制寄存器 OBSSEL0-OBSSEL5 可以用于覆盖标准的 GPIO 行为，以及在引脚 P1[0:5]上输出 RF 内核信号（rfc_obs_sig0、rfc_obs_sig1 和 rfc_obs_sig2）。可用信号的列表见第 19 章。

7.10 掉电信号 MUX (PMUX)

PMUX 寄存器可以用于输出 32 kHz 时钟和/或数字稳压器的状态。

所选的 32 kHz 时钟源可以输出在 P0 其中的一个引脚上。使能位 CKOEN 使得输出在 P0 上，使用 CKOPIN（详细信息见 PMUX 寄存器描述）选择 P0 的引脚。当 CKOEN 被设置，所选引脚的所有其他配置都被覆盖。时钟在所有供电模式下都输出；但是，在 PM3 下时钟停止（见第 4 章的 PM3）。

而且，数字稳压器的状态可以输出在 P1 其中的一个引脚上。当 DREGSTA 位被设置，数字稳压器的状态就被输出。DREGSTAPIN 选择 P1 引脚（详细信息见 PMUX 寄存器描述）。当 DREGSTA 被设置，所选引脚的所有其他配置都被覆盖。当 1.8V 片上数字稳压器上电（芯片有调整过的电压），所选的引脚输出 1。当 1.8V 片上数字稳压器掉电，即在 PM2 和 PM3 下，所选的引脚输出 0。

7.11 I/O 引脚

I/O 端口的寄存器描述在本节中。寄存器如下：

- P0 ： 端口 0

- P1 ： 端口 1
- P2 ： 端口 2
- PERCFG ： 外设控制寄存器
- APCFG ： 模拟外设 I/O 配置
- P0SEL ： 端口 0 功能选择寄存器
- P1SEL ： 端口 1 功能选择寄存器
- P2SEL ： 端口 2 功能选择寄存器
- P0DIR ： 端口 0 方向寄存器
- P1DIR ： 端口 1 方向寄存器
- P2DIR ： 端口 2 方向寄存器
- P0INP ： 端口 0 输入模式寄存器
- P1INP ： 端口 1 输入模式寄存器
- P2INP ： 端口 2 输入模式寄存器
- P0IFG ： 端口 0 中断状态标志寄存器
- P1IFG ： 端口 1 中断状态标志寄存器
- P2IFG ： 端口 2 中断状态标志寄存器
- PICTL ： 中断边缘寄存器
- P0IEN ： 端口 0 中断掩码寄存器
- P1IEN ： 端口 1 中断掩码寄存器
- P2IEN ： 端口 2 中断掩码寄存器
- PMUX ： 掉电信号 Mux 寄存器
- OBSSEL0 ： 观察输出控制寄存器 0
- OBSSEL1 ： 观察输出控制寄存器 1
- OBSSEL2 ： 观察输出控制寄存器 2
- OBSSEL3 ： 观察输出控制寄存器 3
- OBSSEL4 ： 观察输出控制寄存器 4
- OBSSEL5 ： 观察输出控制寄存器 5

P0 (0x80) – 端口 0

位	名称	复位	R/W	描述
7:0	P0[7:0]	0xFF	R/W	端口0。通用I/O端口。可以从SFR位寻址。该CPU内部寄存器可以从XDATA (0x7080)读，但是不能写。

P1 (0x90) – 端口 1

位	名称	复位	R/W	描述
7:0	P1[7:0]	0xFF	R/W	端口1。通用I / O端口。可以从SFR位寻址。该CPU内部寄存器可以从XDATA (0x7090)读，但是不能写。

P2 (0xA0) – 端口 2

位	名称	复位	R/W	描述
7:5	-	000	R0	未使用
4:0	P2[4:0]	0x1F	R/W	端口2。通用I / O端口。可以从SFR位寻址。该CPU内部寄存器可以从XDATA (0x70A0)读，但是不能写。

PERCFG (0xF1) – 外设控制

位	名称	复位	R/W	描述
7	-	0	R0	没有使用
6	T1CFG	0	R/W	定时器 1 的 I / O 位置 0: 备用位置 1 1: 备用位置 2
5	T3CFG	0	R/W	定时器 3 的 I / O 位置 0: 备用位置 1 1: 备用位置 2
4	T4CFG	0	R/W	定时器 4 的 I / O 位置 0: 备用位置 1 1: 备用位置 2
3:2	-	00	R0	没有使用
1	U1CFG	0	R/W	USART 1的I/O位置 0 备用位置1 1 备用位置2
0	U0CFG	0	R/W	USART 0的I/O位置 0: 备用位置1 1: 备用位置2

APCFG (0xF2) – 模拟外设 I/O 配置

位	名称	复位	R/W	描述
7:0	APCFG[7:0]	0x00	R/W	模拟外设I/O配置。APCFG[7:0]选择P0.7 - P0.0作为模拟I/O 0: 模拟I/O禁用 1: 模拟I/O使能

P0SEL (0xF3) – 端口 0 功能选择

位	名称	复位	R/W	描述
7: 0	SELP0_[7:0]	0x00	R/W	P0.7到 P0.0功能选择 0: 通用I / O 1: 外设功能

P1SEL (0xF4) – 端口 1 功能选择

位	名称	复位	R/W	描述
7:0	SELP1_[7:0]	0x00	R/W	P1.7 到P0.0功能选择 0: 通用I / O 1: 外设功能

P2SEL (0xF5) – 端口 2 功能选择和端口 1 外设优先级控制

位	名称	复位	R/W	描述
7	-	0	R0	没有使用
6	PRI3P1	0	R/W	端口1外设优先级控制。当模块被指派到相同的引脚的时候，这些位确定哪个模块优先。 0: USART 0 优先 1: USART 1 优先
5	PRI2P1	0	R/W	端口1外设优先级控制。当PERCFG分配USART1和定时器3到相同的引脚的时候，这些位确定优先次序。 0: USART 1 优先 1: 定时器3优先
4	PRI1P1	0	R/W	端口1外设优先级控制。当PERCFG分配定时器1和定时器4到相同的引脚的时候，这些位确定优先次序。 0: 定时器1优先 1: 定时器4优先
3	PRI0P1	0	R/W	端口1外设优先级控制。当PERCFG分配USART 0和定时器1到相同的引脚的时候，这些位确定优先次序。 0: USART 0 优先 1: 定时器1 优先
2	SELP2_4	0	R/W	P2.4 功能选择 0: 通用 I/O 1: 外设功能
1	SELP2_3	0	R/W	P2.3功能选择 0: 通用I/O 1: 外设功能
0	SELP2_0	0	R/W	P2.0功能选择 0: 通用I/O 1: 外设功能

P0DIR (0xFD) – 端口 0 方向

位	名称	复位	R/W	描述
7:0	DIRP0_[7:0]	0x00	R/W	P0.7到P0.0的I/O方向 0: 输入 1: 输出

P1DIR (0xFE) – 端口 1 方向

位	名称	复位	R/W	描述
7:0	DIRP1_[7:0]	0x00	R/W	P1.7到P1.0的I/O方向 0: 输入 1: 输出

P2DIR (0xFF) – 端口 2 方向和端口 0 外设优先级控制

位	名称	复位	R/W	描述
7:6	PRIP0[1:0]	00	R/W	端口0外设优先级控制。当PERCFG分配给一些外设到相同引脚的时候，这些位将确定优先级。 详细优先级列表： 00： 第1优先级：USART 0 第2优先级：USART 1 第3优先级：定时器1 01： 第1优先级：USART 1 第2优先级：USART 0 第3优先级：定时器1 10： 第1优先级：定时器1通道0-1 第2优先级：USART 1 第3优先级：USART 0 第4优先级：定时器1通道2 –3 11： 第1优先级：定时器1通道2-3 第2优先级：USART 0 第3优先级：USART 1 第4优先级：定时器1通道0 –1
5	-	0	R0	不使用
4:0	DIRP2_[4:0]	0 0000	R/W	P2.4到P2.0的I/O方向 0： 输入 1： 输出

P0INP (0x8F) – 端口 0 输入模式

位	名称	复位	R/W	描述
7:0	MDP0_[7:0]	0x00	R/W	P0.7到P0.0的I/O输入模式 0： 上拉/下拉(见P2INP (0xF7)–端口2输入模式) 1： 三态

P1INP (0xF6) – 端口 1 输入模式

位	名称	复位	R/W	描述
7:2	MDP1_[7:2]	0000 00	R/W	P1.7到P1.2的I/O输入模式 0： 上拉/下拉(见P2INP (0xF7)–端口2输入模式) 1： 三态
1:0	-	00	R0	不使用

P2INP (0xF7) – 端口 2 输入模式

位	名称	复位	R/W	描述
7	PDUP2	0	R/W	端口2上拉/下拉选择。对所有端口2引脚设置为上拉/下拉输入。 0: 上拉 1: 下拉
6	PDUP1	0	R/W	端口1上拉/下拉选择。对所有端口1引脚设置为上拉/下拉输入。 0: 上拉 1: 下拉
5	PDUP0	0	R/W	端口0上拉/下拉选择。对所有端口0引脚设置为上拉/下拉输入。 0: 上拉 1: 下拉
4:0	MDP2_[4:0]	0 0000	R/W	P2.4到P2.0的I/O输入模式 0: 上拉/下拉 1: 三态

P0IFG (0x89) – 端口 0 中断状态标志

位	名称	复位	R/W	描述
7:0	P0IF[7:0]	0x00	R/W0	端口0，位7到位0输入中断状态标志。当输入端口中断请求未决信号时，其相应的标志位将置1。

P1IFG (0x8A) – 端口 1 中断状态标志

位	名称	复位	R/W	描述
7:0	P1IF[7:0]	0x00	R/W0	端口1，位7到位0输入中断状态标志。当输入端口引脚中断请求未决信号时，其相应的标志位将置1。

P2IFG (0x8B) – 端口 2 中断状态标志

位	名称	复位	R/W	描述
7:6	-	00	R0	不使用
5	DPIF	0	R/W0	USB D+中断状态标志。当D+线有一个中断请求未决时设置该标志，用于检测USB挂起状态下的USB恢复事件。当USB控制器没有挂起时不设置该标志。
4:0	P2IF[4:0]	0 0000	R/W0	端口2，位4到位0输入中断状态标志。当输入端口引脚有中断请求未决信号时，其相应的标志位将置1。

PICTL (0x8C) – 端口中断控制

位	名称	复位	R/W	描述
7	PADSC	0	R/W	控制I/O引脚在输出模式下的驱动能力。选择输出驱动能力增强来补偿引脚DVDD的低I/O电压（这为了确保在较低的电压下的驱动能力和较高电压下相同）。 0：最小驱动能力增强。DVDD1/2等于或大于2.6V 1：最大驱动能力增强。DVDD1/2小于2.6V
6:4	-	000	R0	未使用
3	P2ICON	0	R/W	端口2，4到0输入模式下的中断配置。该位为所有端口2的输入4到0选择中断请求条件。 0: 输入的上升沿引起中断 1: 输入下降沿引起中断
2	P1ICONH	0	R/W	端口1，7到4输入模式下的中断配置。该位为所有端口1的输入选择中断请求条件 0: 输入的上升沿引起中断 1: 输入的下降沿引起中断
1	P1ICONL	0	R/W	端口1，3到0输入模式下的中断配置。该位为所有端口1的输入选择中断请求条件 0: 输入的上升沿引起中断 1: 输入的下降沿引起中断
0	P0ICON	0	R/W	端口0， 7到0输入模式下的中断配置。该位为所有端口0的输入选择中断请求条件。 0: 输入的上升沿引起中断 1: 输入的下降沿引起中断

P0IEN (0xAB) – 端口 0 中断屏蔽

位	名称	复位	R/W	描述
7:0	P0_[7:0]IEN	0x00	R/W	端口P0.7到P0.0中断使能 0：中断禁用 1：中断使能

P1IEN (0x8D) – 端口 1 中断屏蔽

位	名称	复位	R/W	描述
7:0	P1_[7:0]IEN	0x00	R/W	端口P1.7到P1.0中断使能 0： 中断禁止 1： 中断使能

P2IEN (0xAC) – 端口 2 中断屏蔽

位	名称	复位	R/W	描述
7:6	-	00	R/W	未使用
5	DPIEN	0	R/W	USB D+中断使能
4:0	P2_[4:0]IEN	0 0000	R/W	端口P2.4到P2.0中断使能 0： 中断禁止 1： 中断使能

PMUX (0xAE) – 掉电信号 Mux

位	名称	复位	R/W	描述
7	CKOEN	0	R/W	时钟输出使能。当该位设置，所选的32 kHz时钟在P0引脚之一上输出。CKOPIN选择使用的引脚。这覆盖选定引脚的所有其他配置。时钟在所有供电模式下都输出，但是在PM3下时钟停止（见第4章的PM3）。
6:4	CKOPIN[2:0]	000	R/W	时钟输出引脚。选择哪个P0引脚用作输出所选的32 kHz时钟。
3	DREGSTA		R/W	数字稳压器状态。当该位设置，数字稳压器的状态在P1引脚之一上输出。DREGSTAPIN选择使用的引脚。当设置DREGSTA，它覆盖选定引脚的所有其他配置。当1.8V片上数字稳压器上电，所选引脚输出1。当1.8V片上数字稳压器掉电，所选引脚输出0。

OBSSEL0 (0x6243) – 观察输出控制寄存器 0

位	名称	复位	R/W	描述
7	EN	0	R/W	P1[0]上观察输出0的位控制。 0- 观察输出禁用 1- 观察输出使能 注意：如果使能，它重写P1.0标准的GPIO行为。
6:0	SEL[6:0]	000 0000	R/W	观察输出0上的选择输出信号 1111011 (123): rfc_obs_sig0 1111100 (124): rfc_obs_sig1 1111101 (125): rfc_obs_sig2 其他：保留

OBSSEL1 (0x6244) – 观察输出控制寄存器 1

位	名称	复位	R/W	描述
7	EN	0	R/W	P1[1]上观察输出1的位控制。 0- 观察输出禁用 1- 观察输出使能 注意：如果使能，它重写P1.1标准的GPIO行为。
6:0	SEL[6:0]	000 0000	R/W	观察输出1上的选择输出信号 1111011 (123): rfc_obs_sig0 1111100 (124): rfc_obs_sig1 1111101 (125): rfc_obs_sig2 其他：保留

OBSSEL2 (0x6245) – 观察输出控制寄存器 2

位	名称	复位	R/W	描述
7	EN	0	R/W	P1[2]上观察输出2的位控制。 0- 观察输出禁用 1- 观察输出使能 注意：如果使能，它重写P1.2标准的GPIO行为。
6:0	SEL[6:0]	000 0000	R/W	观察输出2上的选择输出信号 1111011 (123): rfc_obs_sig0 1111100 (124): rfc_obs_sig1 1111101 (125): rfc_obs_sig2 其他：保留

OBSSEL3 (0x6246) – 观察输出控制寄存器 3

位	名称	复位	R/W	描述
7	EN	0	R/W	P1[3]上观察输出3的位控制。 0- 观察输出禁用 1- 观察输出使能 注意：如果使能，它重写P1.3标准的GPIO行为。
6:0	SEL[6:0]	000 0000	R/W	观察输出3上的选择输出信号 1111011 (123): rfc_obs_sig0 1111100 (124): rfc_obs_sig1 1111101 (125): rfc_obs_sig2 其他：保留

OBSSEL4 (0x6247) – 观察输出控制寄存器 4

位	名称	复位	R/W	描述
7	EN	0	R/W	P1[4]上观察输出4的位控制。 0- 观察输出禁用 1- 观察输出使能 注意：如果使能，它重写P1.4标准的GPIO行为。
6:0	SEL[6:0]	000 0000	R/W	观察输出4上的选择输出信号 1111011 (123): rfc_obs_sig0 1111100 (124): rfc_obs_sig1 1111101 (125): rfc_obs_sig2 其他：保留

OBSSEL5 (0x6248) – 观察输出控制寄存器 5

位	名称	复位	R/W	描述
7	EN	0	R/W	P1[5]上观察输出5的位控制。 0- 观察输出禁用 1- 观察输出使能 注意：如果使能，它重写P1.5标准的GPIO行为。
6:0	SEL[6:0]	000 0000	R/W	观察输出5上的选择输出信号 1111011 (123): rfc_obs_sig0 1111100 (124): rfc_obs_sig1 1111101 (125): rfc_obs_sig2 其他：保留

DMA 控制器

直接存取访问（DMA）控制器可以用来减轻 8051CPU 内核传送数据操作的负担，从而实现在高效利用电源的条件下的高性能。只需要 CPU 极少的干预，DMA 控制器就可以将数据从诸如 ADC 或 RF 收发器的外设单元传送到存储器。

DMA 控制器协调所有的 DMA 传送，确保 DMA 请求和 CPU 存储器访问之间按照优先等级协调、合理地进行。DMA 控制器含有若干可编程的 DMA 通道，用来实现存储器-存储器的数据传送。

DMA 控制器控制整个 XDATA 存储空间的数据传送。由于大多数 SFR 寄存器映射到 DMA 存储器空间，这些灵活的 DMA 通道的操作能够以创新的方式减轻 CPU 的负担，例如，从存储器传送数据到 USART，或定期在 ADC 和存储器之间传送数据样本，等等。使用 DMA 还可以保持 CPU 在低功耗模式下与外设单元之间传送数据（见 4.1.1 节 CPU 低功耗模式），不需要唤醒，这就降低了整个系统的功耗。注意 2.2.3 节描述了哪个 SFR 寄存器映射到 XDATA 存储空间。

DMA 控制器的主要功能如下：

- 5 个独立的 DMA 通道
- 3 个可以配置的 DMA 通道优先级
- 32 个可以配置的传送触发事件
- 源地址和目标地址的独立控制
- 单独传送、数据块传送和重复传送模式
- 支持传输数据的长域域，设置可变传输长度
- 既可以工作在字模式，又可以工作在字节模式。

标题 **页**

8.1 DMA 操作

DMA 控制器有 5 个通道，即 DMA 通道 0 到通道 4。每个 DMA 通道能够从 DMA 存储器空间的一个位置传送数据到另一个位置，比如 XDATA 位置之间。

为了使用 DMA 通道，必须首先按照 8.2 节和 8.3 节所述的进行配置。图 8-1 显示了 DMA 状态图。

当 DMA 通道配置完毕后，在允许任何传输发起之前，必须进入工作状态。DMA 通道通过将 DMA 通道工作状态寄存器 DMAARM 中指定位置 1，就可以进入工作状态。

一旦 DMA 通道进入工作状态，当配置的 DMA 触发事件发生时，传送就开始了。注意一个通道准备工作状态（即获得配置数据）的时间需要 9 个系统时钟，因此如果相应的 DMAARM 位设置，触发在需要配置通道的时间内出现，期望的触发将丢失。如果多于一个 DMA 通道同时进入工作状态，所有通道配置的的时间将长一些（按顺序读取内存）。如果所有 5 个通道都进入工作状态，需要 45 个系统时钟，通道 1 首先准备好，然后是通道 2，最后是通道 0（所有都在最后 8 个系统时钟内）。可能的 DMA 触发事件有 32 个（见表 8-1），例如 UART 传输、定时器溢出等。DMA 通道要使用的触发事件由 DMA 通道配置设置，因此直到配置被读取之后，才能知道。DMA 的触发事件列在表 8-1 中。

补充一点，为了通过 DMA 触发事件开始 DMA 传送，用户软件可以设置对应的 DMAREQ 位，强制使一个 DMA 传送开始。

应该注意如果之前配置的触发器源在 DMA 正在配置的时候产生了触发事件，就被当做错过的事件，一旦 DMA 通道准备好，传输就立即开始。即使新的触发和之前的触发不同也是这样。在一些情况下，这会导致传输错误。为了纠正这一点，触发源 0 必须是重新配置之间的触发源。这通过设置虚拟源和目标地址、使用一个字节的固定长度、块传输和触发源 0 实现。使能软件触发器（DMAREQ）清除错过的触发数，从存储器中取出一个新的配置之前（除非软件为该通道写 DMAREQ），不产生新的触发。

DMAREQ 位只能在相应的 DMA 传输发生时清除。当通道解除准备工作状态时，DMAREQ 位不被清除。

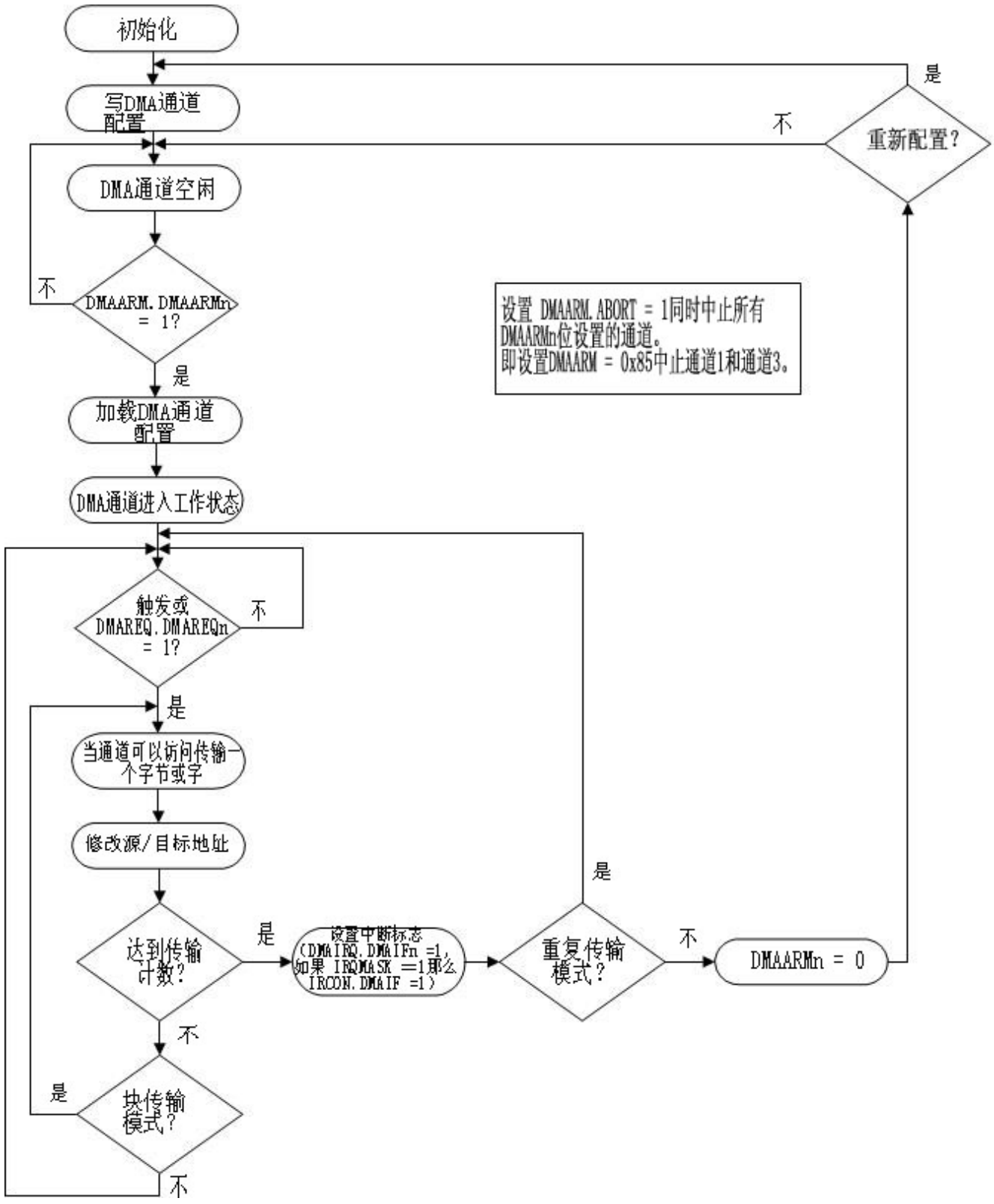

初始化
写DMA通道配置
DMA通道空闲
DMAARM.DMAARMn = 1?
不
是
加载DMA通道配置
DMA通道进入工作状态
触发或DMAREQ.DMAREQn = 1?
不
是
当通道可以访问传输一个字节或字
修改源/目标地址
达到传输计数?
是
设置中断标志(DMAIRQ.DMAIFn =1,如果 IRQMASK =1那么IRCON.DMAIF =1)
重复传输模式?
是
不
DMAARMn = 0
不
是
块传输模式?
是
不
重新配置?
是
不
设置 DMAARM.ABORT = 1同时中止所有DMAARMn位设置的通道。
即设置DMAARM = 0x85中止通道1和通道3。

图 8-1 DMA 操作

8.2 DMA 配置参数

DMA 运行的安装和控制由用户软件完成。本节描述了 DMA 通道能够使用之前，必须配置的参数。8.3 节描述了参数在软件中如何配置，并传递到 DMA 控制器。

这五个 DMA 通道每一个的行为通过下列参数配置：

源地址：DMA 通道要读的数据的首地址。

目标地址：DMA 通道从源地址读出的要写数据的首地址。用户必须确认该目标地址可写。

传送长度：在 DMA 通道重新进入工作状态或者解除工作状态之前，以及警告 CPU 即将有中断请求到来之前，要传送的长度。长度可以在配置中定义，或可以如下所述定义为 VLEN 设置。

可变长度（VLEN）设置：DMA 通道可以利用源数据中的第一个字节或字作为传送长度进行可变长度传输。使用可变长度传输时，要给出关于如何计算要传输的字节数的各种选项。

优先级别：DMA 通道的 DMA 传送的优先级别与 CPU、其它 DMA 通道和访问端口相关。

触发事件：所有 DMA 传输通过所谓的 DMA 触发事件来发起。这个触发可以启动一个 DMA 块传输或单个 DMA 传输。除了已经配置的触发，DMA 通道总是可以通过设置它的指定 DMAREQ.DMAREQx 标志来触发。DMA 触发的源描述在表 8-1 中。

源地址和目标地址增量：源地址和目标地址可以控制为增量或减少，或不改变。

传送模式：传送模式确定传送是否是单个传输，或块传输，或是它们的重复传输。

字节传送或字传送：确定每个 DMA 传输应该是 8 位（字节）或是 16 位（字）。

中断屏蔽：在完成 DMA 通道传送时，产生一个中断请求。这个中断屏蔽位控制中断产生是使能还是禁用。

M8：这个域的值，决定是否采用 7 位还是 8 位长的字节来传送数据。此模式仅仅适用于字节传送。

所有配置参数的详细描述给定在 8.2.1 节到 8.2.11 节，

8.2.1 源地址

DMA 通道开始读数据的地址，在 XDATA 存储器中。这可以是任何 XDATA 地址——在 RAM 中，在映射的闪存区（cf MEMCTR.XBANK）中，XREG 或 XDATA 寻址的 SFR。

8.2.2 目标地址

DMA 通道从源地址读出的要写数据的首地址。用户必须确认该目标地址可写。这可以是任何 XDATA 地址——在 RAM、XREG 或 XDATA 寻址的 SFR 中。

8.2.3 传送数量

DMA 传输完成之前必须传送的字节/字的个数。当达到传送数量，DMA 通道重新进入工作状态或者解除工作状态，并警告 CPU 即将有中断请求到来。传送数量可以在配置中定义，或可以如下节所述定义为可变长度设置，见 8.2.4 节。

8.2.4 VLEN 设置

DMA 通道可以利用源数据中的第一个字节或字（对于字，使用位 12：0）作为传送长度。这允许可变长度的传输。当使用可变长度传送时，要给出关于如何计算要传输的字节数的各种选项。在任何情况下，都是设置传送长度(LEN)为传送的最大长度。如果首字节或字指明的传输长度大于 LEN，那么 LEN 个字节/字将被传输。当使用可变长度传输，那么 LEN 应设置位允许传输的最大长度加一。

注意，仅在选择字节长度传送数据时才可以使用 M8 位(见 8.2.11 节)。

可以同 VLEN 一起设置的选项如下：

1、 传输首字节/字规定的个数+1 字节/字（先传输字节/字的长度，然后按照字节/字长度指定的传输尽可能多的字节/字）

2、 传输首字节/字规定的字节/字

3、 传输首字节/字规定的个数+2 字节/字（先传输字节/字的长度，然后按照字节/字长度指定+1 传输尽可能多的字节/字）

4、 传输首字节/字规定的个数+3 字节/字（先传输字节/字的长度，然后按照字节/字长度指定+2 传输尽可能多的字节/字）

图 8-2 显示了 VLEN 选项。

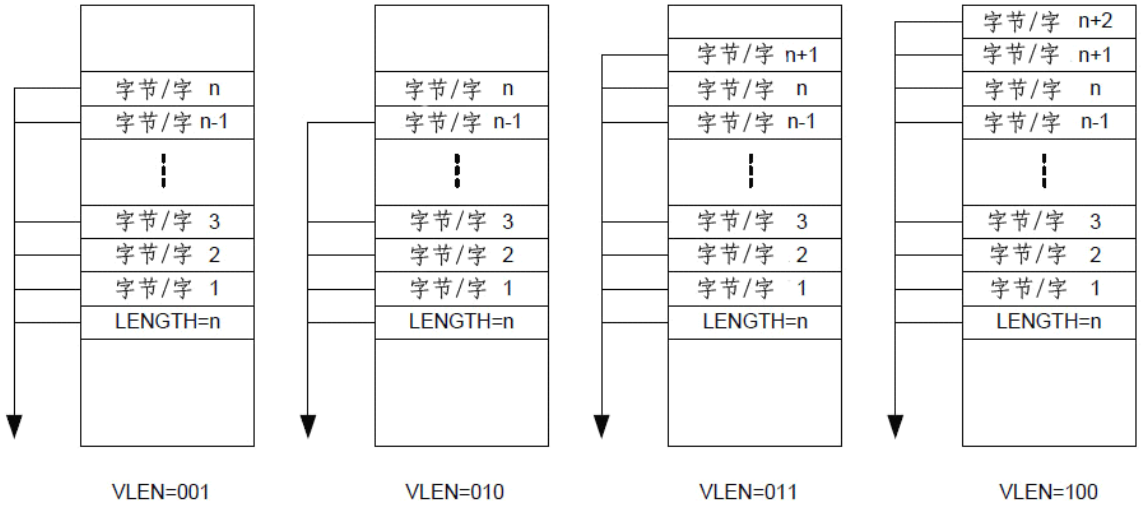


图 8-2 可变长度（VLEN）传输选项

8.2.5 触发事件

可以设置每个 DMA 通道接受单个事件的触发。这样一来，就可以判定 DMA 通道会接受哪一个事件的触发。

8.2.6 源和目标增量

当 DMA 通道进入工作状态或者重新进入工作状态时，源地址和目标地址传送到内部地址指针。其地址增量可能有下列 4 种：

- *增量为 0*。每次传送之后，地址指针将保持不变。
- *增量为 1*。每次传送之后，地址指针将加上 1 个数。
- *增量为 2*。每次传送之后，地址指针将加上 2 个数。

- *减量为 1*。每次传送之后，地址指针将减去 1 个数。

其中一个数在字节模式下等于 1 个字节，在字模式下等于 2 个字节。

8.2.7 DMA 传输模式

传输模式确定当 DMA 通道开始传输数据时是如何工作的。以下描述了四种传输模式：

单一模式：每当触发时，发生一个 DMA 传送，DMA 通道等待下一个触发。完成指定的传送长度后，传送结束，通报 CPU，解除 DMA 通道的工作状态。

块模式：每当触发时，按照传送长度指定的若干 DMA 传送被尽快传送，此后，通报 CPU，解除 DMA 通道的工作状态。

重复的单一模式：每当触发时，发生一个 DMA 传送，DMA 通道等待下一个触发。完成指定的传送长度后，传送结束，通报 CPU，且 DMA 通道重新进入工作状态。

重复的块模式：每当触发时，按照传送长度指定的若干 DMA 传送被尽快传送，此后通报 CPU，DMA 通道重新进入工作状态。

8.2.8 DMA 优先级

DMA 优先级别对每个 DMA 通道是可以配置的。DMA 优先级别用于判定同时发生的多个内部存储器请求中的哪一个优先级最高，以及 DMA 存储器存取的优先级别是否超过同时发生的 CPU 存储器存取的优先级别。在同属内部关系的情况下，采用轮转调度方案应对，确保所有的存取请求。有三种级别的 DMA 优先级：

高级：最高内部优先级别。DMA 存取总是优先于 CPU 存取。

一般级：中等内部优先级别。保证 DMA 存取至少在每秒一次的尝试中优先于 CPU 存取。

低级：最低内部优先级别。DMA 存取总是劣于 CPU 存取。

8.2.9 字节或字传输

判定已经完成的传送究竟是 8 位（字节）还是 16 位（字）。

8.2.10 中断屏蔽

在完成 DMA 传送的基础上，该 DMA 通道能够产生一个中断到处理器。这个位可以屏蔽该中断。

8.2.11 模式 8 设置

这个域的值，决定是采用 7 位还是 8 位长的字节来传送数据。此模式仅仅适用于字节传送。

8.3 DMA 配置安装

以上描述的 DMA 通道参数（诸如地址模式、传送模式和优先级别等）必须在 DMA 通道进入工作状态之前配置并激活。参数不直接通过 SFR 寄存器配置，而是通过写入存储器中特殊的 DMA 配置数据结构中配置。对于使用的每个 DMA 通道，需要有它自己的 DMA 配置数据结构。DMA 配置数据结构包含 8 字节，如 8.6 节所述。DMA 配置数据结构可以存放在由用户软件设定的任何位置，而地址通过一组 SFR，DMAxCFGH :DMAxCFGL 送到 DMA 控制器。一旦 DMA 通道进入工作状态，DMA 控制器就会读取该通道的配置数据结构，由 DMAxCFGH :DMAxCFGL 地址给出。

需要注意的是，指定 DMA 配置数据结构开始地址的方法十分重要。这些地址对于 DMA 通道 0 和 DMA 通道 1～4 是不同的：

DMA0CFGH：DMA0CFGL 给出 DMA 通道 0 配置数据结构的开始地址。

DMA1CFGH：DMA1CFGL 给出 DMA 通道 1 配置数据结构的开始地址，其后跟着通道 2-4 的配置数据结

构。

因此 DMA 控制器希望 DMA 通道 1-4 的 DMA 配置数据结构存在于存储器连续的区域内，以 DMA1CFGH:DMA1CFGL 所保存的地址开始，包含 32 个字节。

8.4 停止 DMA 传输

使用 DMAARM 寄存器来解除 DMA 通道工作状态，停止正在运行的 DMA 传送或进入工作状态的 DMA。

将 1 写入 DMAARM.ABORT 寄存器位，就会停止一个或多个进入工作状态的 DMA 通道，同时通过设置相应的 DMAARM.DMAARMx 为 1 选择停止哪个 DMA 通道。当设置 DMAARM.ABORT 为 1，非停止通道的 DMAARM.DMAARMx 位必须写入 0。

8.5 DMA 中断

每个 DMA 通道可以配置为一旦完成 DMA 传送，就产生中断到 CPU。该功能由 IRQMASK 位在通道配置时实现。当中断产生时，SFR 寄存器 DMAIRQ 中所对应的中断标志位置 1。

一旦 DMA 通道完成传送，不管在通道配置中 IRQMASK 位是何值，中断标志都会置 1。这样，当通道重新进入工作状态且 IRQMASK 的设置改变时，软件必须总是检测（并清除）这个寄存器。如果这样做失败，将会根据存储的中断标志产生一个中断。

8.6 DMA 配置数据结构

对于每个 DMA 通道，DMA 数据结构由 8 个字节组成。配置数据结构描述在表 8-2 中。

8.7 DMA 存储访问

DMA 数据传输被端约定影响。注意 DMA 描述符遵循大端约定，而其他描述符遵循小端约定。这必须在编译器中说明。

表 8-1 DMA 触发源

DMA触发器		功能单元	描述
号码	名称		
0	NONE	DMA	没有触发器，设置DMAREQ.DMAREQx位开始传送
1	PREV	DMA	DMA通道是通过完成前一个通道来触发的。
2	T1_CH0	定时器1	定时器1，比较，通道0
3	T1_CH1	定时器1	定时器1，比较，通道1
4	T1_CH2	定时器1	定时器1，比较，通道2
5	T2_EVENT1	定时器2	定时器2，事件脉冲1
6	T2_ EVENT2	定时器2	定时器2，事件脉冲2
7	T3_CH0	定时器3	定时器3，比较，通道0
8	T3_CH1	定时器3	定时器3，比较，通道1
9	T4_CH0	定时器4	定时器4，比较，通道0
10	T4_CH1	定时器4	定时器4，比较，通道1
11	ST	睡眠定时器	睡眠定时器比较
12	IOC_0	IO控制器	端口0 I/O引脚输入转换 (1)
13	IOC_1	IO控制器	端口1 I/O引脚输入转换 (1)
14	URX0	USART 0	USART 0接收完成
15	UTX0	USART 0	USART 0发送完成

（1）使用这个触发器源必须符合端口中断使能位 PICTL.P0IENL/H 和 P1IEN。注意所有中断使能的端口引脚都将产生一个触发。

表 8-1 DMA 触发源（续表）

DMA触发器		功能单元	描述
号码	名称		
16	URX1	USART 1	USART 1接收完成
17	UTX1	USART 1	USART 1发送完成
18	FLASH	闪存控制器	写闪存数据完成
19	RADIO	无线模块	接收RF字节包（详见19.3节）
20	ADC_CHALL	ADC	ADC结束一次转换，采样已经准备好
21	ADC_CH11	ADC	ADC结束通道0的一次转换，采样已准备好
22	ADC_CH21	ADC	ADC结束通道1的一次转换，采样已准备好
23	ADC_CH32	ADC	ADC结束通道2的一次转换，采样已准备好
24	ADC_CH42	ADC	ADC 结束通道3的一次转换，采样已准备好
25	ADC_CH53	ADC	ADC结束通道4的一次转换，采样已准备好
26	ADC_CH63	ADC	ADC结束通道5的一次转换，采样已准备好
27	ADC_CH74	ADC	ADC结束通道6的一次转换，采样已准备好
28	ADC_CH84	ADC	ADC结束通道7的一次转换，采样已准备好
29	ENC_DW	AES	AES加密处理器请求下载输入数据
30	ENC_UP	AES	AES加密处理器请求上传输出数据
31	DBG_BW	调试接口	调试接口突发写

表 8-2 DMA 配置数据结构

字节偏移量	位	名称	描述
0	7:0	SRCADDR[15:8]	DMA通道源地址，高位
1	7:0	SRCADDR[7:0]	DMA通道源地址，低位
2	7:0	DESTADDR[15:8]	DMA通道目的地址，高位。请注意，闪存存储器不能直接写入。
3	7:0	DESTADDR[7:0]	DMA通道目的地址，高位。请注意，闪存存储器不能直接写入。
4	7:5	VLEN[2:0]	可变长度传输模式。在字模式中，第一个字的12:0位被认为是传送长度的。 000： 采用LEN作为传送长度 001： 传送由第一个字节/字+1指定的字节/字的长度（上限到由LEN指定的最大值）。因此，传输长度不包括字节/字的长度 010: 传送通过第一个字节/字指定的字节/字的长度（上限到由LEN指定的最大值）。因此，传输长度包括字节/字的长度 011： 传送通过第一个字节/字+2指定的字节/字的长度（上限到由LEN指定的最大值）。因此，传输长度不包括字节/字的长度 100： 传送通过第一个字节/字+3指定的字节/字的长度（上限到由LEN指定的最大值）。因此，传输长度不包括字节/字的长度 101： 保留 110： 保留 111： 使用LEN作为传输长度的备用
4	4:0	LEN[12:8]	DMA的通道传送长度 当VLEN从000到111时采用最大允许长度。当处于WORDSIZE模式时，DMA通道数以字为单位，否则以字节为单位
5	7:0	LEN[7:0]	DMA的通道传送长度 当VLEN从000到111时采用最大允许长度。当处于WORDSIZE 模式时，DMA通道数以字为单位，否则以字节为单位
6	7	WORDSIZE	选择每个DMA传送是采用8位(0) 还是16位(1)。

表 8-2 DMA 配置数据结构（续表）

字节偏移量	位	名称	描述
6	6:5	TMODE[1:0]	DMA通道传送模式： 00：单个 01：块 10：重复单一 11：重复块
6	4:0	TRIG[4:0]	选择要使用的DMA触发 0 0000：无触发(写到DMAREQ仅仅是触发) 0 0001：前一个DMA通道完成 0 0010 – 1 1110：选择表8-1中展示的一个触发。触发的选择按照表中序号。
7	7:6	SRCINC[1:0]	源地址递增模式 (每次传送之后): 00：0字节/字 01：1字节/字 10：2字节/字 11：-1字节/字
7	5:4	DESTINC[1:0]	目的地址递增模式 (每次传送之后): 00：0字节/字 01：1字节/字 10：2字节/字 11：-1字节/字
7	3	IRQMASK	该通道的中断屏蔽 0：禁止中断发生 1： DMA通道完成时使能中断发生
7	2	M8	采用VLEN的第8位模式作为传送单位长度；仅应用在WORDSIZE=0且VLEN从000到111时。 0：采用所有8位作为传送长度 1：采用字节的低7位作为传送长度
7	1:0	PRIORITY[1:0]	DMA通道的优先级别： 00：低级，CPU优先 01：保证级，DMA至少在每秒一次的尝试中优先 10：高级，DMA优先 11：保留

8.8 DMA 寄存器

本节描述了与 DMA 控制器相关的 SFR 寄存器。

DMAARM（0xD6）– DMA 通道进入工作状态

位	名称	保留	R/W	描述
7	ABORT	0	R0/W	DMA停止。此位是用来停止正在进行的DMA传输。通过设置相应DMAARM位为1，写1到该位停止所有选择的通道。 0：正常运行 1：停止所有选择的通道
6:5	-	00	R/W	不使用
4	DMAARM4	0	R/W1	DMA进入工作状态通道4 为了任何DMA传输能够在该通道上发生，该位必须置1。对于非重复传输模式，一旦完成传送，该位自动清0。
3	DMAARM3	0	R/W1	DMA进入工作状态通道3 为了任何DMA传输能够在该通道上发生，该位必须置1。对于非重复传输模式，一旦完成传送，该位自动清0。
2	DMAARM2	0	R/W1	DMA 进入工作状态通道 2 为了任何 DMA 传输能够在该通道上发生，该位必须置 1。对于非重复传输模式，一旦完成传送，该位自动清 0。
1	DMAARM1	0	R/W1	DMA 进入工作状态通道 1 为了任何 DMA 传输能够在该通道上发生，该位必须置 1。对于非重复传输模式，一旦完成传送，该位自动清 0。
0	DMAARM0	0	R/W1	DMA 进入工作状态通道 0 为了任何 DMA 传输能够在该通道上发生，该位必须置 1。对于非重复传输模式，一旦完成传送，该位自动清 0。

DMAREQ（0xD7）– DMA 通道开始请求和状态

位	名称	复位	R/W	描述
7:5	-	000	R0	不使用
4	DMAREQ4	0	R/W1 H0	DMA传送请求，通道4 当设置为1时，激活DMA通道(与一个触发事件具有相同的效果)。当DMA传输开始清除该位。
3	DMAREQ3	0	R/W1 H0	DMA传送请求，通道3 当设置为1，激活DMA通道(与一个触发事件具有相同的效果)。当DMA传输开始清除该位。
2	DMAREQ2	0	R/W1 H0	DMA传送请求，通道2 当设置为1，激活DMA通道(与一个触发事件具有相同的效果)。当DMA传输开始清除该位。
1	DMAREQ1	0	R/W1 H0	DMA传送请求，通道1 当设置为1，激活DMA通道(与一个触发事件具有相同的效果)。当DMA传输开始清除该位。
0	DMAREQ0	0	R/W1 H0	DMA传送请求，通道0 当设置为1，激活DMA通道(与一个触发事件具有相同的效果)。当DMA传输开始清除该位。

DMA0CFGH（0xD5）– DMA 通道 0 配置地址高字节

位	名称	复位	R/W	描述
7:0	DMA0CFG[15:8]	0x00	R/W	DMA通道0配置地址，高位字节

DMA0CFGL（0xD4）– DMA 通道 0 配置地址低字节

位	名称	复位	R/W	描述
7:0	DMA0CFG[7:0]	0x00	R/W	DMA通道0配置地址，低位字节

DMA1CFGH（0xD3）– DMA 配置通道 1-4 的高字节地址

位	名称	复位	R/W	描述
7:0	DMA1CFG[15:8]	0x00	R/W	DMA通道1-4配置地址，高位字节

DMA1CFGL（0xD2）– DMA 配置通道 1-4 的低字节地址

位	名称	复位	R/W	描述
7:0	DMA1CFG[7:0]	0x00	R/W	DMA通道1-4配置地址，低位字节

DMAIRQ（0xD1）– DMA 中断标志

位	名称	复位	R/W	描述
7:5	-	000	R/W0	不使用
4	DMAIF4	0	R/W0	DMA通道4中断标志 0：DMA通道传送未完成 1：DMA通道传送完成/中断未决
3	DMAIF3	0	R/W0	DMA通道3中断标志 0：DMA通道传送没有完成 1：DMA通道传送完成/中断未决
2	DMAIF2	0	R/W0	DMA通道2中断标志 0：DMA通道传送没有完成 1：DMA通道传送完成/中断未决
1	DMAIF1	0	R/W0	DMA通道1中断标志 0：DMA通道传送没有完成 1：DMA通道传送完成/中断未决
0	DMAIF0	0	R/W0	DMA通道0中断标志 0：DMA通道传送没有完成 1：DMA通道传送完成/中断未决

定时器 1（16 位定时器）

定时器 1 是一个独立的 16 位定时器，支持典型的定时/计数功能，比如输入捕获，输出比较和 PWM 功能。定时器有五个独立的捕获/比较通道。每个通道定时器使用一个 I/O 引脚。定时器用于范围广泛的控制和测量应用，可用的五个通道的正计数/倒计数模式将允许诸如电机控制应用的实现。

定时器 1 的功能如下：

- 五个捕获/比较通道
- 上升沿、下降沿或任何边沿的输入捕获
- 设置、清除或切换输出比较
- 自由运行、模或正计数/倒计数操作
- 可被 1，8，32 或 128 整除的时钟分频器
- 在每个捕获/比较和最终计数上生成中断请求
- DMA 触发功能

标题 页

9.1 16 位计数器

定时器包括一个 16 位计数器，在每个活动时钟边沿递增或递减。活动时钟边沿周期由寄存器位 CLKCON.TICKSPD 定义，它设置全球系统时钟的划分，提供了从 0.25MHz 到 32MHz 的不同的时钟标签频率（可以使用 32 MHz XOSC 作为时钟源）。这在定时器 1 中由 T1CTL.DIV 设置的分频器值进一步划分。这个分频器值可以从 1、8、32 或 128。因此当 32 MHz 晶振用作系统时钟源时，定时器 1 可以使用的最低时钟频率是 1953.125Hz，最高是 32 MHz。当 16MHz RC 振荡器用作系统时钟源时，定时器 1 可以使用的最高时钟频率是 16MHz。

计数器可以作为一个自由运行计数器，一个模计数器或一个正计数/倒计数器运行，用于中心对齐的 PWM。

可以通过两个 8 位的 SFR 读取 16 位的计数器值：T1CNTH 和 T1CNTL，分别包含在高位字节和低位字节中。当读取 T1CNTL 时，计数器的高位字节在那时被缓冲到 T1CNTH，以便高位字节可以从 T1CNTH 中读出。因此 T1CNTL 必须总是在读取 T1CNTH 之前首先读取。

对 T1CNTL 寄存器的所有写入访问将复位 16 位计数器。

当达到最终计数值（溢出）时，计数器产生一个中断请求。可以用 T1CTL 控制寄存器设置启动并停止该计数器。当一个不是 00 值的写入到 T1CTL.MODE 时，计数器开始运行。如果 00 写入到 T1CTL.MODE，计数器停止在它现在的值上。

9.2 定时器 1 操作

一般来说控制寄存器 T1CTL 用于控制定时器操作。状态寄存器 T1STAT 保存中断标志。各种操作模式如下所述。

9.3 自由运行模式

在自由运行操作模式下，计数器从 0x0000 开始，每个活动时钟边沿增加 1。当计数器达到 0xFFFF（溢出），计数器载入 0x0000，继续递增它的值，如图 9-1 所示。当达到最终计数值 0xFFFF，设置标志 IRCON.T1IF 和 T1STAT.OVFIF。如果设置了相应的中断屏蔽位 TIMIF.OVFIM 以及 IEN1.T1EN，将产生一个中断请求。自由运行模式可以用于产生独立的时间间隔，输出信号频率

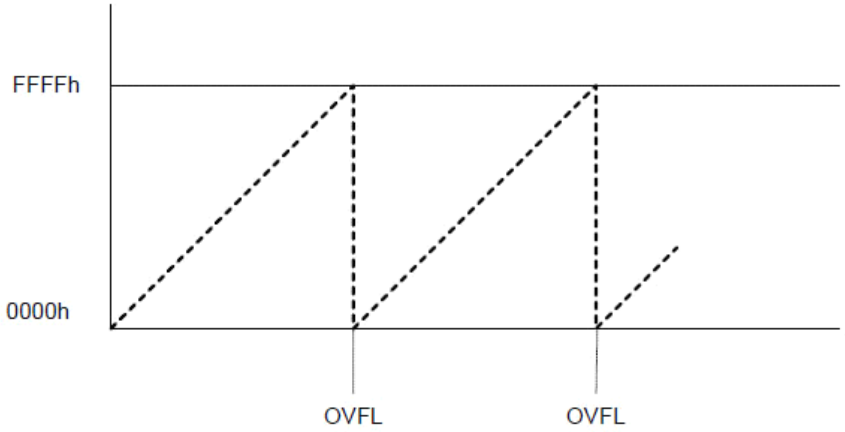


图 9-1　自由运行模式

9.4 模模式

当定时器运行在模模式，16 位计数器从 0x0000 开始，每个活动时钟边沿增加 1。当计数器达到 T1CC0（溢出），寄存器 T1CC0H:T1CC0L 保存的最终计数值，计数器将复位到 0x0000，并继续递增。如果定时器开始于 T1CC0 以上的一个值，当达到最终计数值（0xFFFF）时，设置标志 IRCON.T1IF 和 T1CTL.OVFIF。如果设置了相应的中断屏蔽位 TIMIF.OVFIM 以及 IEN1.T1EN，将产生一个中断请求。模模式可以用于周期不是 0xFFFF 的应用程序。计数器的操作展示在图 9-2 中。

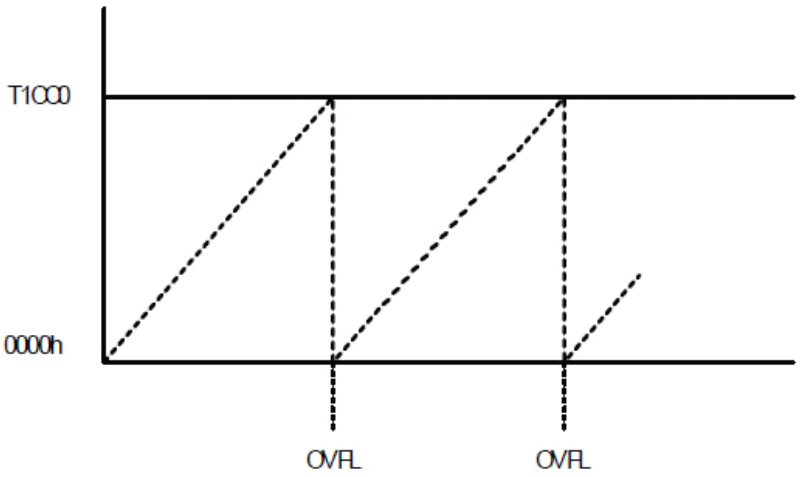


图 9-2 模模式

9.5 正计数/倒计数模式

在正计数/倒计数模式，计数器反复从 0x0000 开始，正计数直到达到 T1CC0H:T1CC0L 保存的值。然后计数器将倒计数直到 0x0000，如图 9-3 所示。这个定时器用于周期必须是对称输出脉冲而不是 0xFFFF 的应用程序，因此允许中心对齐的 PWM 输出应用的实现。在正计数/倒计数模式，当达到最终计数值时，设置标志 IRCON.T1IF 和 T1CTL.OVFIF。如果设置了相应的中断屏蔽位 TIMIF.OVFIM 以及 IEN1.T1EN，将产生一个中断请求。

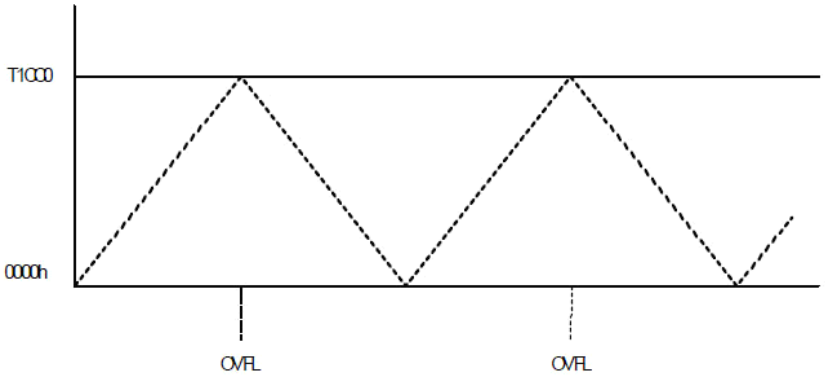


图 9-3 正计数/倒计数模式

9.6 通道模式控制

通道模式随着每个通道的控制和状态寄存器 T1CCTLn 设置。设置包括输入捕获和输出比较模式。

9.7 输入捕获模式

当一个通道配置为输入捕获通道，和该通道相关的 I/O 引脚必须被配置为输入。在启动定时器之后，输入引脚的一个上升沿、下降沿或任何边沿都将触发一个捕获，即把 16 位计数器内容捕获到相关的捕获寄存器中。因此定时器可以捕获一个外部事件发生的时间。

注意：在定时器可以使用一个输入/输出引脚之前，所需的 I/O 引脚必须配置为定时器 1 的外设引脚。

通道输入引脚和内部系统时钟是同步的。因此输入引脚上的脉冲的最低持续时间必须大于系统时钟周期。

16 位捕获寄存器的内容从寄存器 T1CCnH:T1CCnL 中读出。

当捕获发生时，要设置 IRCON.T1IF 标志和该通道的中断标志 T1STAT.CHnIF（n 是通道号码）。如果分别设置了相应的中断屏蔽位 T1CCTLn.IM，以及 IEN1.T1EN，将产生一个中断请求。

9.8 输出比较模式

在输出比较模式，与通道相关的 I/O 引脚设置为输出。在定时器启动之后，将比较计数器和通道比较寄存器 T1CCnH:T1CCnL 的内容。如果比较寄存器等于计数器的内容，输出引脚根据比较输出模式 T1CCTLn.CMP 的设置进行设置、复位或切换。注意输出引脚运行在一个给定输出比较模式下时，它上面的所有边沿都是无故障运行的。写入比较寄存器 T1CCnL 将被缓冲，这样写入到 T1CCnL 的值不起作用，直到相应的高位寄存器 T1CCnH 被写入。写入比较寄存器 T1CCnH:T1CCnL 对于输出比较值不起作用，直到定时器达到 0x00。

注意通道 0 的输出比较模式较少，因为 T1CC0H:T1CC0L 在模式 6 和 7 有一个特殊功能，这意味着这些模式对于通道 0 是不能使用的。

当发生一个比较时，设置 IRCON.T1IF 标志和该通道的中断标志 T1STAT.CHnIF（n 是通道号码）。如果分别设置了相应的中断屏蔽位 T1CCTLn.IM，以及 IEN1.T1EN，将产生一个中断请求。

不同定时器模式下输出比较模式的例子给定在以下图中。

边沿对齐：PWM 输出信号可以使用定时器在自由运行模式下，通道 1 和 2 在输出比较模式 6 或 7 下生成（由 T1CCTLn.CMP 位定义，其中 n 是 1 或 2），如图 9-4 所示。PWM 信号的周期通过设置 T1CC0 确定，通道输出的占空比由 T1CCn 确定，其中 n 是 PWM 通道 1 或 2。

也可以使用定时器自由运行模式。在这种情况下，T1CTL.DIV 位中的 CLKCON.TICKSPD 和分频器值设置 PWM 信号的周期。PWM 信号的极性由使用的是输出比较模式 6 还是 7 确定。

PWM 输出信号还可以使用图 9-4 所示的输出比较模式 4 和 5，或通过使用图 9-5 所示的模模式生成。对于简单的 PWM，最好使用使用输出比较模式 4 和 5 来生成。

中心对齐：PWM 输出可以通过选择定时器正计数/倒计数模式生成。根据 PWM 信号所需的极性选择通道输出比较模式 4 或 5（由 T1CCTLn.CMP 位定义，其中 n 是 1 或 2）。PWM 信号的周期由 T1CC0 确定，通道输出的占空比由 T1CCn 确定，其中 n 是 PWM 通道 1 或 2。

某些类型的电机驱动应用程序会需要中心对齐的 PWM 模式，一般地这比边沿对齐的 PWM 模式产生的噪音更少，因为 I/O 引脚传输不集中在同一个时钟边沿上。

在一些类型的应用程序中，需要在输出之间定义一个延迟或死亡的时间。典型地，这用于输出驱动一个 H 桥配置，以避免 H 桥的一边交叉传导失控。延迟或死亡时间可以通过使用 T1CCn 在 PWM 输出中获得，如下所示：

假定通道 1 和通道 2 使用定时器正计数/倒计数模式，用于驱动输出，且这两个通道分别使用输出比较模式 4 和 5，那么定时器周期（定时器 1 的时钟周期）是：

t_P = T1CC0 x 2

死亡时间，即两个输出都为低电平的时间，（定时器 1 的时钟周期）是：

t_D = T1CC1 – T1CC2

当下列情况发生，比较输出引脚初始化为表 9-1 所列的值：

- 一个值被写入 T1CNTL（所有定时器 1 通道）
- 0x7 被写到 T1CCTLn.CMP（通道 n）

表 9-1　初始的比较输出值（比较模式）

比较模式（T1CCTLn. CMP）	初始的比较输出
在比较设置输出（000）	0
在比较清除输出（001）	1
在比较切换输出（010）	0
在正计数/倒计数模式下，在比较正计数设置输出，在比较倒计数清除（011）	0
不是正计数/倒计数模式下，在比较设置输出，在 0 清除（011）	0
在正计数/倒计数模式下，在比较正计数清除输出，在比较倒计数设置（100）	1
不是正计数/倒计数模式下，在比较清除输出，在 0 设置（100）	1
等于 T1CC0 时清除，等于 T1CCn 时设置（101）	0
等于 T1CC0 时设置，等于 T1CCn 时清除（101）	1

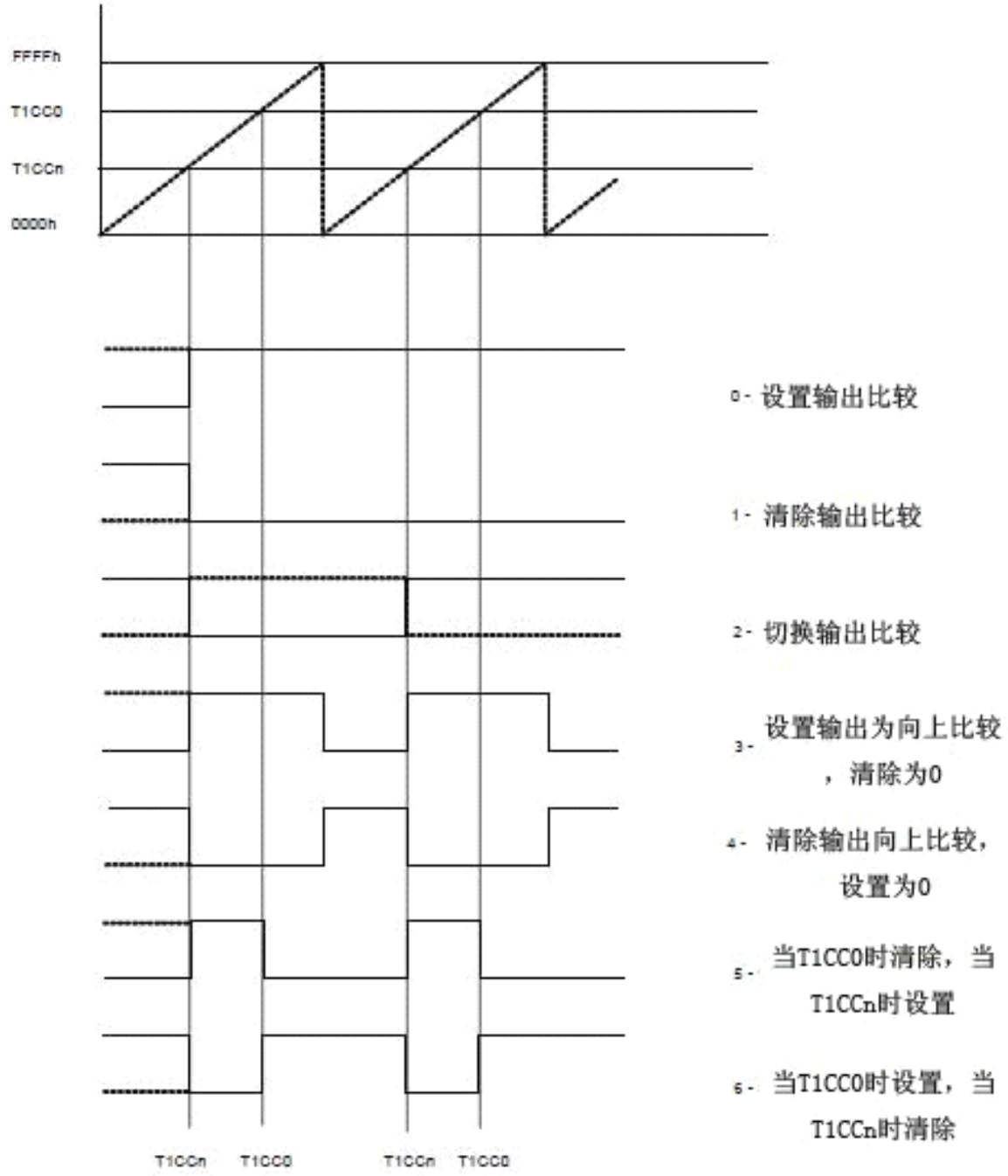


图 9-4 输出比较模式，定时器自由运行模式

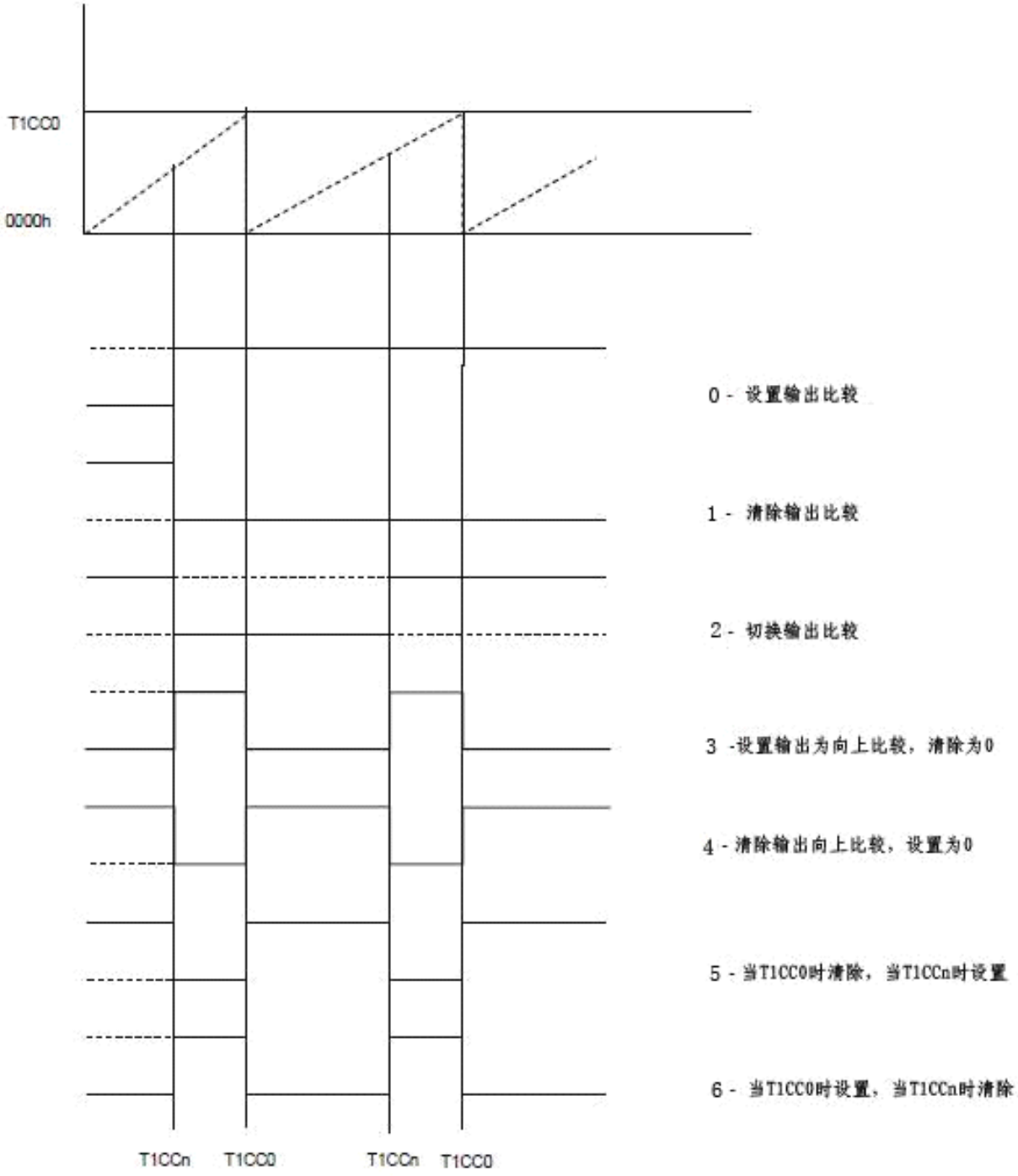


图 9-5 输出比较模式，定时器模模式

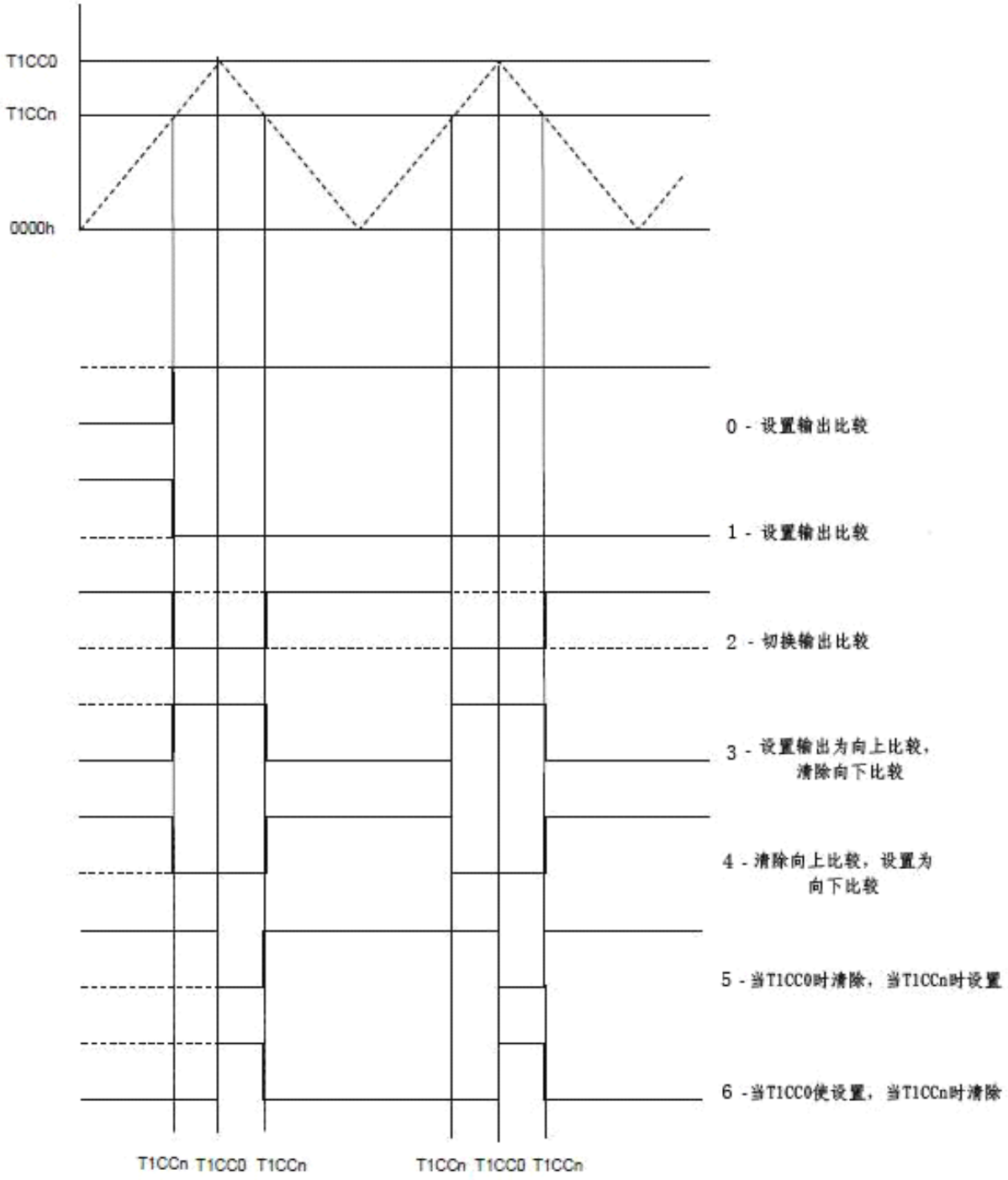


图 9-6 输出比较模式，定时器正计数/倒计数模式

9.9 IR 信号产生和线性化

本节描述了 CC253x 设备只需最少的 SW 参与即可产生 IR 的功能。

9.9.1 简介

为远程控制产生 IR 信号一般以下面两种方式之一完成：

- 调制码
- 非调制码（C 码，闪存代码）

CC253x 包括灵活的定时器功能，以最少的 CPU 参与，来执行这两种类型的 IR 信号的产生和线性化。大多数 IR 协议每个命令只需一个 CPU 干预即可实现。

9.9.2 调制码

调制码可以使用定时器 1（16 位）和定时器 3（8 位）生成。处于调制模式的定时器 3 用于产生载波。定时器 3 有一个单独的分频器，用于它的输入。它的周期使用 T3CC0 设置。定时器 3 通道 1 用于 PWM 输出。载波的占空比使用 T3CC1 设置。通道 1 使用比较模式："在比较设置输出，在 0xFF 清除"（T3CCTL1.CMP = 101）。表 9-2 显示了定时器 3 的 38kHz 载波的频率误差计算。

表 9-2 38kHz 载波的频率误差计算

描述	值
系统时钟频率	32,000 kHz
IR 载波频率	38 kHz
系统时钟周期	0.00003125 ms
IR 载波周期	0.026315789 ms
定时器分频器	4
定时器周期	0.000125 ms
理想的定时器值	210.5263158
实际的定时器值	211
实际的定时器周期	0.026375 ms
实际的定时器频率	37.91469194 kHz
周期误差	59.21052632 ns
频率误差	85.30805687 Hz
频率误差%	0.2245%

IRCTL.IRGEN 寄存器位使得 IR 产生模式处于定时器 1。当设置了 IRGEN 位，定时器 1 采用定时器 3 通道 1 的输出比较信号作为标记，而不是采用系统标记。定时器 1 周期是使用 T1CC0 设置的，定时器 1 处于调制模式（T1CTL.MODE = 10），通道 0 处于比较模式（T1CCTL0.MODE = 1）。通道 1 比较模式"在比较设置输出，在 0x0000 清除"（T1CCTL1.CMP = 011）用于输出门控信号。

*标记*载波的个数由 T1CC1. T1CC1 设置，需要每个定时器 1 周期由 DMA 或 CPU 更新一次。注意 T1CC1 的一个更新被缓冲，在定时器 1 达到 0x0000 之前不起作用。

*空间*载波的个数由 T1CC0 设置。其值必须设置为标记和*空间*载波周期希望的总数。比较值被缓冲直到定时器达到 0x0000。

定时器 1 通道 1 的输出进行定时器 3 通道 1 的输出和运算，作为 IR 的输出，如图 9-7。

图 9-7 定时器在 IR 产生模式的方框图

定时器 3 通道 1 输出和定时器 1 通道 1 输出信号的时序是同步的，这样在 IR 输出信号上就没有故障。

当设置了 IRGEN 位，IR 输出信号被送到引脚，而不是送到一般的定时器 1 通道 1 输出（也可参见 7.6.1 节）。

图 9-8 显示了定时器 3 被初始化为 33%的占空比（T3CC0 = 3× T3CC1）的例子。定时器 1 已经被初始化为 3。

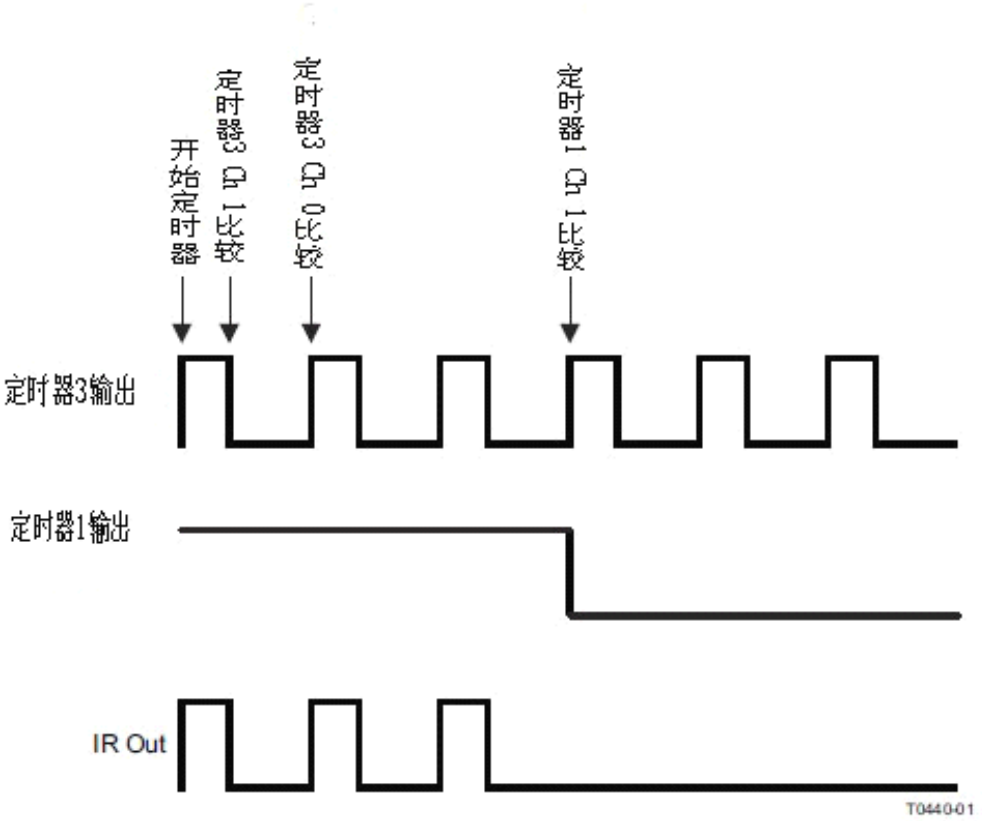


图 9-8 调制的波形示例

如果仅仅要实现一个*空间*周期，T1CC1 必须设置为 0x00。

9.9.3 非调制码

要产生非调制 IR 码，定时器 1 要处于模模式。信号的周期由 T1CC0 给定，脉冲宽度由 T1CC1 给定。T1CC1 给出标记周期的长度，T1CC0 给出*标记*和*空间*周期的总数。比较值被缓冲，直到定时器达到 0x0000。如果比较值不能保持不变，必须在每个周期由 DMA 或 CPU 更新。

9.9.4 学习

学习通过使用定时器 1（16 位）和定时器 3（8 位）的捕获功能完成。定时器 3 可以处理载波频率检测，定时器 1 可以处理调制信代码的学习。电路应该按照图 9-9 所述安装。

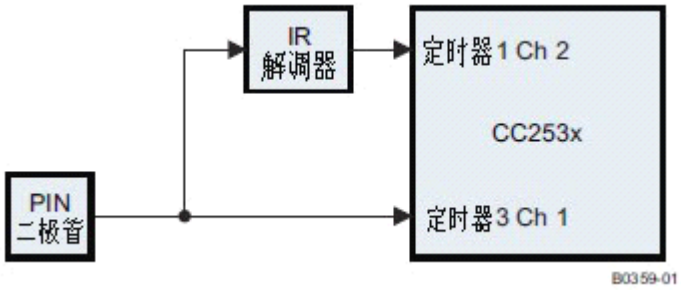


图 9-9 IR 学习的方框图

9.9.4.1 载波频率检测

定时器 3 用于捕获和检测直接从 IR 引脚二极管输入的载波频率。定时器必须对载波进行一定次数的采样。如果载波被检测，被检测出的频率必须提供平均数，这会存储在数据库中。

9.9.4.2 解调码学习

IR 引脚二极管的输出由一个合适的电路解调。这一电路的输出用作处于捕获模式的定时器 1 其中一个通道的输入。

9.9.5 其他注意事项

IR 输出引脚在复位期间必须处于三态或下拉状态，以避免点亮 IR LED 这一不必要的功耗。注意只有定时器 1 通道 1 的输出 P1.1 是三态的，在复位期间和复位后没有上拉。

9.10 定时器 1 中断

为定时器分配了一个中断向量。当下列定时器事件之一发生时，将产生一个中断请求：

- 计数器达到最终计数值（溢出或回到零）
- 输入捕获事件
- 输出比较事件

寄存器状态寄存器 T1STAT 包括最终计数值事件和五个通道比较/捕获事件的中断标志。仅当设置了相应的中断屏蔽位和 IEN1.T1EN 时，才能产生一个中断请求。中断屏蔽位是 n 个通道的 T1CCTLn.IM 和溢出事件 TIMIF.OVFIM。如果有其它未决中断，必须在一个新的中断请求产生之前，通过软件清除相应的中断标志。而且，如果设置了相应的中断标志，使能一个中断屏蔽位将产生一个新的中断请求。

9.11 定时器 1 DMA 触发

有三种 DMA 触发与定时器 1 有关。这些是 DMA 触发 T1_CH0,T1_CH1 和 T1_CH2，分别在以下定时器比较事件上产生：

- T1_CH0 – 通道 0 比较
- T1_CH1 – 通道 1 比较
- T1_CH2 – 通道 2 比较

通道 3 和 4 没有相关的触发。

9.12 定时器 1 寄存器

本节描述了定时器 1 的寄存器，由以下寄存器组成：

- T1CNTH - 定时器 1 计数高位
- T1CNTL - 定时器 1 计数低位
- T1CTL - 定时器 1 控制
- T1STAT - 定时器 1 状态
- T1CCTLn - 定时器 1 通道 n 捕获/比较控制
- T1CCnH - 定时器 1 通道 n 捕获/比较高位值
- T1CCnL - 定时器 1 通道 n 捕获/比较低位值

TIMIF.OVFIM 寄存器位驻留在 TIMIF 寄存器，和定时器 3 和定时器 4 寄存器一起描述。

T1CNTH (0xE3) – 定时器 1 计数器高位

位	名称	复位	R/W	描述
7:0	CNT[15:8]	0x00	R	定时器计数器高字节。包含在读取T1CNTL的时候定时计数器缓存的高16位字节。

T1CNTL (0xE2) – 定时器 1 计数器低位

位	名称	复位	R/W	描述
7:0	CNT[7:0]	0x00	R/W	定时器计数器低字节。包括16位定时计数器低字节。往该寄存器中写任何值，导致计数器被清除为 0x0000，初始化所有相通道的输出引脚。

T1CTL (0xE4) – 定时器 1 的控制和状态

位	名称	复位	R/W	描述
7:4	-	0000 0	R0	保留
3:2	DIV[1:0]	00	R/W	分频器划分值。产生主动的时钟边缘用来更新计数器，如下： 00： 标记频率/1 01： 标记频率/8 10： 标记频率/32 11： 标记频率/128
1:0	MODE[1:0]	00	R/W	选择定时器1模式。定时器操作模式通过下列方式选择： 00： 暂停运行。 01： 自由运行，从0x0000到0xFFFF反复计数。 10： 模，从 0x0000到T1CC0反复计数。 11：正计数/倒计数，从 0x0000到T1CC0反复计数并且从T1CC0倒计数到0x0000。

T1STAT (0xAF) – 定时器 1 状态

位	名称	复位	R/W	描述
7:6	-	0	R0	保留
5	OVFIF	0	R/W0	定时器 1 计数器溢出中断标志。当计数器在自由运行或模模式下达到最终计数值时设置，当在正/倒计数模式下达到零时倒计数。写 1 没有影响。
4	CH4IF	0	R/W0	定时器 1 通道 4 中断标志。当通道 4 中断条件发生时设置。写 1 没有影响。
3	CH3IF	0	R/W0	定时器 1 通道 3 中断标志。当通道 3 中断条件发生时设置。写 1 没有影响。
2	CH2IF	0	R/W0	定时器 1 通道 2 中断标志。当通道 2 中断条件发生时设置。写 1 没有影响。
1	CH1IF	0	R/W0	定时器 1 通道 1 中断标志。当通道 1 中断条件发生时设置。写 1 没有影响。
0	CH0IF	0	R/W0	定时器 0 通道 4 中断标志。当通道 0 中断条件发生时设置。写 1 没有影响。

1 0

T1CCTL0 (0xE5) – 定时器 1 通 0 捕获/比较控制

位	名称	复位	R/W	描述
7	RFIRQ	0	R/W	当设置时，使用RF中断捕获，而不是常规捕获输入。
6	IM	1	R/W	通道0中断屏蔽。设置时使能中断请求。
5:3	CMP[2:0]	000	R/W	通道0比较模式选择。当定时器的值等于在T1CC0中的比较值，选择操作输出 000： 在比较设置输出 001： 在比较清除输出 010： 在比较切换输出 011： 在向上比较设置输出，在0清除 100： 在向上比较清除输出，在0设置 101： 没有使用 110： 没有使用 111： 初始化输出引脚。CMP[2:0]不变。
2	MODE	0	R/W	模式。选择定时器1通道0捕获或者比较模式 0： 捕获模式 1： 比较模式
1:0	CAP[1:0]	00	R/W	通道0捕获模式选择 00： 未捕获 01： 上升沿捕获 10： 下降沿捕获 11： 所有沿捕获

T1CC0H (0xDB) – 定时器 1 通道 0 捕获/比较值高位

位	名称	复位	R/W	描述
7:0	T1CC0[15:8]	0x00	R/W	定时器1通道0捕获/比较值，高位字节。当T1CCTL0.MODE = 1（比较模式）时写0到该寄存器导致T1CC0[15:0]更新写入值延迟到T1CNT = 0x0000。

T1CC0L (0xDA) – 定时器 1 通道 0 捕获/比较值低位

位	名称	复位	R/W	描述
7:0	T1CC0[7:0]	0x00	R/W	定时器1通道0捕获/比较值，低位字节。写到该寄存器的数据被存储到一个缓存中，但是不写入T1CC0[7:0]，直到并同时后一次写T1CC0H生效。

T1CCTL1 (0xE6) – 定时器 1 通道 1 捕获/比较控制

位	名称	复位	R/W	描述
7	RFIRQ	0	R/W	设置时使用RF中断捕获，而不是常规的捕获输入。
6	IM	1	R/W	通道1中断屏蔽。中断请求时，可以设置。
5:3	CMP[2:0]	000	R/W	通道1比较模式选择。当定时器值等于在T1CC1的比较值时选择操作输出。 000： 在比较设置输出 001： 在比较清除输出 010： 在比较切换输出 011： 在向上比较设置输出，在0清除。否则在比较设置输出，在0清除。 100： 在向上比较清除输出，在0设置。否则在比较清除输出，在0设置。 101： 当等于T1CC0时清除，当等于T1CC1时设置 110： 当等于T1CC0时设置，当等于T1CC1时清除 111： 初始化输出引脚。CMP[2:0]不变。
2	MODE	0	R/W	模式。选择定时器1通道1捕获或者比较模式 0： 捕获模式 1： 比较模式
1:0	CAP[1:0]	00	R/W	通道1捕获模式选择 00： 未捕获 01： 上升沿捕获 10： 下降沿捕获 11： 所有沿捕获

T1CC1H (0xDD) – 定时器 1 通道 1 捕获/比较值高位

位	名称	复位	R/W	描述
7:0	T1CC1[15:8]	0x00	R/W	定时器1通道1捕获/比较值，高位字节。当T1CCTL1.MODE = 1（比较模式）时写该寄存器导致T1CC1[15:0]更新写入值延迟到T1CNT = 0x0000。

T1CC1L (0xDC) – 定时器 1 通道 1 捕获/比较值低位

位	名称	复位	R/W	描述
7:0	T1CC1[7:0]	0x00	R/W	定时器1通道1捕获/比较值，低位字节。写入该寄存器的数据存储到一个缓存中，但是不写入T1CC1[7:0]，直到并同时后一次写入T1CC1H生效。

T1CCTL2 (0xE7) – 定时器 1 通道 2 捕获/比较控制

位	名称	复位	R/W	描述
7	RFIRQ	0	R/W	设置时使用RF捕获而不是常规捕获输入。
6	IM	1	R/W	通道2中断屏蔽。设置时使能中断请求。
5:3	CMP[2:0]	000	R/W	通道2比较模式选择。当定时器的值等于在T1CC2中的比较值时选择操作输出。 000： 在比较设置输出 001： 在比较清除输出 010： 在比较切换输出 011： 在向上比较设置输出，在0清除。否则在比较设置输出，在0清除。 100： 在向上比较清除输出，在0设置。否则在比较清除输出，在0设置。 101： 当等于T1CC0时清除，当等于T1CC2时设置 110： 当等于T1CC0时设置，当等于T1CC2时清除 111： 初始化输出引脚。CMP[2:0]不变。
2	MODE	0	R/W	模式。选择定时器1通道2捕获或者比较模式 0： 捕获模式 1： 比较模式
1:0	CAP[1:0]	00	R/W	通道2捕获模式选择 00： 未捕获 01： 上升沿捕获 10： 下降沿捕获 11： 所有沿捕获

T1CC2H (0xDF) – 定时器 1 通道 2 捕获/比较值高位

位	名称	复位	R/W	描述
7:0	T1CC2[15:8]	0x00	R/W	定时器1通道2捕获/比较值，高位字节。当T1CCTL2.MODE = 1（比较模式）时写该寄存器导致T1CC2[15:0]更新写入值延迟到T1CNT = 0x0000。

T1CC2L (0xDE) – 定时器 1 通道 2 捕获/比较值低位

位	名称	复位	R/W	描述
7:0	T1CC2[7:0]	0x00	R/W	定时器1通道2捕获/比较值，低位字节。写入该寄存器的数据存储到一个缓存中，但是不写入T1CC2[7:0]，直到并同时后一次写入T1CC2H生效。

T1CCTL3 (0x62A3) – 定时器 1 通道 3 捕获/比较控制

位	名称	复位	R/W	描述
7	RFIRQ	0	R/W	设置时使用RF捕获而不是常规捕获输入。
6	IM	1	R/W	通道3中断屏蔽。设置时使能中断请求。
5:3	CMP[2:0]	000	R/W	通道3比较模式选择。当定时器的值等于在T1CC3中的比较值时选择操作输出。 000： 在比较设置输出 001： 在比较清除输出 010： 在比较切换输出 011： 在向上比较设置输出，在0清除。否则在比较设置输出，在0清除。 100： 在向上比较清除输出，在0设置。否则在比较清除输出，在0设置。 101： 当等于T1CC0时清除，当等于T1CC3时设置 110： 当等于T1CC0时设置，当等于T1CC3时清除 111： 初始化输出引脚。CMP[2:0]不变。
2	MODE	0	R/W	模式。选择定时器1通道3捕获或者比较模式 0： 捕获模式 1： 比较模式
1:0	CAP[1:0]	00	R/W	通道3捕获模式选择 00： 未捕获 01： 上升沿捕获 10： 下降沿捕获 11： 所有沿捕获

T1CC3H (0x62AD) – 定时器 1 通道 3 捕获/比较值高位

位	名称	复位	R/W	描述
7:0	T1CC3[15:8]	0x00	R/W	定时器1通道3捕获/比较值，高位字节。当T1CCTL3.MODE = 1（比较模式）时写该寄存器导致T1CC3[15:0]更新写入值延迟到T1CNT = 0x0000。

T1CC3L (0x62AC) – 定时器 1 通道 3 捕获/比较值低位

位	名称	复位	R/W	描述
7:0	T1CC3[7:0]	0x00	R/W	定时器1通道3捕获/比较值，低位字节。写入该寄存器的数据存储到一个缓存中，但是不写入T1CC3[7:0]，直到并同时后一次写入T1CC3H生效。

T1CCTL4 (0x62A4) – 定时器 1 通道 4 捕获/比较控制

位	名称	复位	R/W	描述
7	RFIRQ	0	R/W	设置时使用RF捕获而不是常规捕获输入。
6	IM	1	R/W	通道4中断屏蔽。设置时使能中断请求。
5:3	CMP[2:0]	000	R/W	通道4比较模式选择。当定时器的值等于在T1CC4中的比较值时选择操作输出。 000： 在比较设置输出 001： 在比较清除输出 010： 在比较切换输出 011： 在向上比较设置输出，在0清除。否则在比较设置输出，在0清除。 100： 在向上比较清除输出，在0设置。否则在比较清除输出，在0设置。 101： 当等于T1CC0时清除，当等于T1CC4时设置 110： 当等于T1CC0时设置，当等于T1CC4时清除 111： 初始化输出引脚。CMP[2:0]不变。
2	MODE	0	R/W	模式。选择定时器1通道4捕获或者比较模式 0： 捕获模式 1： 比较模式
1:0	CAP[1:0]	00	R/W	通道4捕获模式选择 00： 未捕获 01： 上升沿捕获 10： 下降沿捕获 11： 所有沿捕获

T1CC4H (0x62AF) – 定时器 1 通道 4 捕获/比较值高位

位	名称	复位	R/W	描述
7:0	T1CC4[15:8]	0x00	R/W	定时器1通道4捕获/比较值，高位字节。当T1CCTL4.MODE = 1（比较模式）时写该寄存器导致T1CC4[15:0]更新写入值延迟到T1CNT = 0x0000。

T1CC4L (0x62AE) – 定时器 1 通道 4 捕获/比较值低位

位	名称	复位	R/W	描述
7:0	T1CC4[7:0]	0x00	R/W	定时器1通道4捕获/比较值，低位字节。写入该寄存器的数据存储到一个缓存中，但是不写入T1CC4[7:0]，直到并同时后一次写入T1CC4H生效。

IRCTL (0x6281) – 定时器 1 IR 产生控制

位	名称	复位	R/W	描述
7:1	-	0000 000	R/W	保留
0	IRGEN	0	R/W	当该位设置，定时器3通道1和定时器1标记输入之间就产生了一个连接，这样定时器可以用于产生调制IR码（也可参见9.9节）。

9.13 作为数组访问定时器 1 寄存器

定时器 1 捕获/比较通道寄存器可以在 XDATA 存储空间中作为一个连续的区域被访问。这使得可以作为一个简单的索引结构方便地访问寄存器。5 个捕获/比较控制寄存器映射到 0x62A0 - 0x62A4。16 位捕获/比较值映射到 0x62A6 - 0x62AF。0x62A5 不使用。

定时器 3 和定时器 4（8 位定时器）

定时器 3 和 4 是两个 8 位的定时器。每个定时器有两个独立的比较通道，每个通道上使用一个 I/O 引脚。

定时器 3/4 的特性如下：

- 两个捕获/比较通道
- 设置、清除或切换输出比较
- 时钟分频器，可以被 1，2，4，8，16，32，64，128 整除
- 在每次捕获/比较和最终计数事件发生时产生中断请求
- DMA 触发功能

标题 页

10.1 8 位定时器计数器

定时器 3 和定时器 4 的所有定时器功能都是基于主要的 8 位计数器建立的。计数器在每个时钟边沿递增或递减。活动时钟边沿的周期由寄存器位 CLKCONCMD.TICKSPD[2:0]定义，由 TxCTL.DIV[2:0]（其中 x 指的是定时器号码，3 或 4）设置的分频器值进一步划分。计数器可以作为一个自由运行计数器，倒计数器，模计数器或正/倒计数器运行。

可以通过 SFR 寄存器 TxCNT 读取 8 位计数器的值，其中 x 指的是定时器号码，3 或 4。

清除和停止计数器是通过设置 TxCTL 控制寄存器的值实现的。当 TxCTL.START 写入 1 时，计数器开始。当 TxCTL.START 写入 0 时，计数器停留在它的当前值。

10.2 定时器 3/定时器 4 模式控制

在一般的控制寄存器中，TxCTL 用于控制定时器的运行。

10.2.1 自由运行模式

在自由运行模式操作下，计数器从 0x00 开始，每个活动时钟边沿递增。当计数器达到 0xFF，计数器载入 0x00，并继续递增。当达到最终计数值 0xFF（比如，发生了一个溢出），就设置中断标志 TIMIF.TxOVFIF。如果设置了相应的中断屏蔽位 TxCTL.OVFIM，就产生一个中断请求。自由运行模式可以用于产生独立的时间间隔和输出信号频率。

10.2.2 倒计数模式

在倒计数模式，定时器启动之后，计数器载入 TxCC0 的内容。然后计数器倒计时，直到 0x00。当达到 0x00 时，设置标志 TIMIF.TxOVFIF。如果设置了相应的中断屏蔽位 TxCTL.OVFIM，就产生一个中断请求。定时器倒计数模式一般用于需要事件超时间隔的应用程序。

10.2.3 模模式

当定时器运行在模模式，8 位计数器在 0x00 启动，每个活动时钟边沿递增。当计数器达到寄存器 TxCC0 所含的最终计数值时，计数器复位到 0x00，并继续递增。当发生这个事件时，设置标志 TIMIF.TxOVFIF 。如果设置了相应的中断屏蔽位 TxCTL.OVFIM，就产生一个中断请求。模模式可以用于周期不是 0xFF 的应用程序。

10.2.4 正/倒计数模式

在正/倒计数定时器模式下，计数器反复从 0x00 开始正计数，直到达到 TxCC0 所含的值，然后计数器倒计数，直到达到 0x00。这个定时器模式用于需要对称输出脉冲，且周期不是 0xFF 的应用程序。因此它允许中心对齐的 PWM 输出应用程序的实现。

通过写入 TxCTL.CLR 清除计数器也会复位计数方向，即从 0x00 模式正计数。

10.3 通道模式控制

对于通道 0 和 1，每个通道的模式是由控制和状态寄存器 TxCCTLn 设置的，其中 n 是通道号码 0 或 1。设置包括捕获和比较模式。

10.4 输入捕获模式

当一个通道配置为一个输入捕获通道，通道相关的 I/O 引脚配置为一个输入。定时器启动之后，输入引脚上的一个上升沿、下降沿或任何边沿都会触发一个捕获，即捕获 8 位计数器内容到相关的捕获寄存器中。因此，定时器能够捕获一个外部事件发生的时间。

注意：在定时器使用一个 I/O 引脚之前，所需的 I/O 引脚必须配置为一个定时器 3/定时器 4 外设引脚。

通道输入引脚与内部系统时钟是同步的。因此，输入引脚上的脉冲的最小持续时间必须大于系统时钟周期。

通道 n 的 8 位捕获寄存器的内容从寄存器 T3CCn/T4CCn 中读出。

当发生一个捕获，对应实际通道的中断标志就被设置。这是 TIMIF.TxCHnIF。如果相应的中断屏蔽位 TxCCTLn.IM 被设置，就产生一个中断请求。

10.5 输出比较模式

在输出比较模式下，与该通道相关的 I/O 引脚必须设置为输出。定时器启动之后，将比较计数器的内容和通道比较寄存器 TxCC0n 的内容。如果比较寄存器等于计数器的内容，根据比较输出模式 TxCCTL.CMP1:0 的设置，输出引脚被设置、复位或切换。注意当运行在一个给定的比较输出模式下，输出引脚上的所有边沿都是无故障运行的。

对于使用简单 PWM，最好使用输出比较模式 4 和 5。

写入比较寄存器 TxCC0 或 TxCC1 的值对输出比较的值不起作用，直到计数器值达到 0x00。

当发生一个比较时，将设置相应的实际通道的中断标志。这是 TIMIF.TxCHnIF。如果设置了相应的中断屏蔽位 TxCCTLn.IM，将产生一个中断请求。

当以下情况发生，比较输出引脚被初始化为表 9-1 所列的值：

- ‘1’写到 TxCNTR.CLR（所有定时器 x 通道）
- 0x7 写到 TxCCTLn.CMP（定时器 x，通道 n）

表 10-1 初始的比较输出值（比较模式）

比较模式（TxCCTLn. CMP）	初始的比较输出
在比较设置输出（000）	0
在比较清除输出（001）	1
在比较切换输出（010）	0
在正计数/倒计数模式下，在比较正计数设置输出，在比较倒计数清除（011）	0
不是正计数/倒计数模式下，在比较设置输出，在 0 清除（011）	0
在正计数/倒计数模式下，在比较正计数清除输出，在比较倒计数设置（100）	1
不是正计数/倒计数模式下，在比较清除输出，在 0 设置（100）	1
在比较设置输出，在 0xFF 清除（101）	0
在比较清除输出，在 0x00 设置（101）	1

10.6 定时器 3 和定时器 4 中断

为这两个定时器各分配了一个中断向量。这些是 T3 和 T4。当以下定时器事件之一发生时，将产生一个中

断请求：

- 计数器达到最终计数值
- 比较事件
- 捕获事件

SFR 寄存器 TIMIF 包含定时器 3 和定时器 4 的所有中断标志。寄存器位 TIMIF.TxOVFIF 和 TIMIF.TxCHnIF 分别包含 2 个最终计数值事件，以及四个通道捕获/比较事件的中断标志。仅当设置了相应的中断屏蔽位时，才会产生一个中断请求。如果有其它未决的中断，必须通过 CPU，在一个新的中断请求产生之前，清除相应的中断标志。而且，如果设置了相应的中断标志，使能一个中断屏蔽位将产生一个新的中断请求。

10.7 定时器 3 和定时器 4 DMA 触发

有两个与定时器 3 相关的 DMA 触发，同样有两个与定时器 4 相关的 DMA 触发。这些触发如下：

- T3_CH0：定时器 3 通道 0 捕获/比较
- T3_CH1：定时器 3 通道 1 捕获/比较
- T4_CH0：定时器 4 通道 0 捕获/比较
- T4_CH0：定时器 4 通道 1 捕获/比较

10.8 定时器 3 和定时器 4 寄存器

T3CNT (0xCA) – 定时器 3 计数器

位	名称	复位	R/W	描述
7:0	CNT[7:0]	0x00	R	定时器计数字节。包含8位计数器当前值

T3CTL (0xCB) – 定时器 3 控制

位	名称	复位	R/W	描述
7:5	DIV[2:0]	000	R/W	分频器划分值。产生有效时钟沿用于来自 CLKCON.TICKSPD 的定时器时钟。如下： 000： 标记频率 /1 001： 标记频率/2 010： 标记频率/4 011： 标记频率/8 100： 标记频率/16 101： 标记频率/32 110： 标记频率/64 111： 标记频率/128
4	START	0	R/W	启动定时器。正常运行时设置，暂停时清除
3	OVFIM	1	R/W0	溢出中断屏蔽 0：中断禁止 1：中断使能
2	CLR	0	R0/W1	清除计数器。写1到CLR复位计数器到0x00，并初始化相关通道所有的输出引脚。总是读作0。
1:0	MODE[1:0]	00	R/W	定时器3模式。选择以下模式： 00： 自由运行，从0x00到0xFF反复计数 01： 倒计数，从T3CC0到0x00计数 10： 模，从0x00到T3CC0重复计数 11： 正/倒计数，从0x00到T3CC0重复计数，降到0x00

T3CCTL0 (0xCC) – 定时器 3 通道 0 捕获/比较控制

位	名称	复位	R/W	描述
7	-	0	R0	未使用
6	IM	1	R/W	通道 0 中断屏蔽 0：中断禁止 1：中断使能
5:3	CMP[2:0]	000	R/W	通道0比较输出模式选择。当时钟值与在T3CC0中的比较值相等时输出特定的操作。 000： 在比较设置输出 001： 在比较清除输出 010： 在比较切换输出 011： 在比较正计数时设置输出，在0清除 100： 在比较正计数时清除输出，在0设置 101： 在比较设置输出，在0xFF清除 110： 在0x00设置，在比较清除输出 111： 初始化输出引脚。CMP[2:0]不变
2	MODE	0	R/W	模式。选择定时器3通道0捕获或者比较模式 0： 捕获模式 1： 比较模式
1:0	CAP[1:0]	00	R/W	捕获模式选择 00：无捕获 01：在上升沿捕获 10：在下降沿捕获 11：在两个边沿都捕获

T3CC0 (0xCD) – 定时器 3 通道 0 捕获/比较值

位	名称	复位	R/W	描述
7:0	VAL[7:0]	0x00	R/W	定时器捕获/比较值通道0。当T3CCTL0.MODE=1（比较模式）时写该寄存器会导致T3CC0.VAL[7:0]更新到写入值延迟到T3CNT.CNT[7:0]=0x00。

T3CCTL1 (0xCE) – 定时 3 通道 1 捕获/比较控制

位	名称	复位	R/W	描述
7	-	0	R0	未使用
6	IM	1	R/W	通道1中断屏蔽 0：中断禁止 1：中断使能
5:3	CMP[2:0]	000	R/W	通道1比较输出模式选择。 当定时器值等于在 T3CC1中的比较值时指定输出。 000： 在比较设置输出 001： 在比较清除输出 010： 在比较切换输出 011： 在比较正计数设置输出，在0清除。否则在比较设置输出，在0清除。 100： 在比较正计数清除输出，在0设置。否则在比较清除输出，在0设置。 101： 在比较设置输出，在0xFF清除 110： 在比较清除输出，在0x00设置 111： 初始化输出引脚。CMP[2:0]不变
2	MODE	0	R/W	模式。选择定时器3通道1模式 0： 捕获模式 1： 比较模式
1:0	CAP[1:0]	00	R/W	捕获模式选择。 00：无捕获 01：在上升沿捕获 10：在下降沿捕获 11：在两个沿都捕获

T3CC1 (0xCF) – 定时器 3 通道 1 捕获/比较值

位	名称	复位	R/W	描述
7:0	VAL[7:0]	0x00	R/W	定时器捕获/比较值通道1。当T3CCTL1.MODE=1（比较模式）时写该寄存器会导致T3CC1.VAL[7:0]更新写入值延迟到T3CNT.CNT[7:0]=0x00。

T4CNT (0xEA) – 定时器 4 计数器

位	名称	复位	R/W	描述
7:0	CNT[7:0]	0x00	R	定时器计数字节。包括8位计数器当前值。

T4CTL (0xEB) – 定时器 4 控制

位	名称	复位	R/W	描述
7:5	DIV[2:0]	000	R/W	分频器划分值。产生有效时钟沿，用于CLKCON.TICKSPD定时器的时钟，如下： 000： 标记频率 /1 001： 标记频率/2 010： 标记频率/4 011： 标记频率/8 100： 标记频率/16 101： 标记频率/32 110： 标记频率/64 111： 标记频率/128
4	START	0	R/W	启动定时器。设置时正常运行，清除时暂停
3	OVFIM	1	R/W0	溢出中断屏蔽
2	CLR	0	R0/W1	清除计数器。写1到CLR复位计数器到0x00，并初始化相关通道所有的输出引脚。总是读作0。
1:0	MODE[1:0]	00	R/W	定时器4模式。选择如下模式： 00： 自由运行，从0x00到0xFF反复计数 01： 倒计数，从 T4CC0到0x00计数 10： 模，从0x00到T4CC0重复计数 11： 正/倒计数，从0x00到T4CC0重复计数，并且下降到 0x00

T4CCTL0 (0xEC) – 定时器 4 通道 0 捕获/比较控制

位	名称	复位	R/W	描述
7	-	0	R0	未使用
6	IM	1	R/W	通道0中断屏蔽
5:3	CMP[2:0]	000	R/W	通道0比较输出模式选择。当定时器的值等于T4CC0中的比较值时指定输出。 000： 在比较设置输出 001： 在比较清除输出 010： 在比较切换输出 011： 正计数比较时设置输出， 在0清除 100： 在正计数比较清除输出，在0设置 101： 在比较设置输出，在0xFF清除 110： 在比较清除输出，在0x00设置 111： 初始化输出引脚。CMP[2:0]不变
2	MODE	0	R/W	模式。选择定时器4通道0模式 0： 捕获模式 1： 比较模式
1:0	CAP[1:0]	00	R/W	捕获模式选择。00-无捕获，01-在上升沿捕获，10-在下降沿捕获，11-在两个沿都捕获

T4CC0 (0xED) – 定时器 4 通道 0 捕获/比较值

位	名称	复位	R/W	描述
7:0	VAL[7:0]	0x00	R/W	定时器捕获/比较值通道0。当T4CCTL1.MODE=1（比较模式）时写该寄存器会导致T4CC1.VAL[7:0]更新写入值延迟到T4CNT.CNT[7:0]=0x00。

T4CCTL1 (0xEE) – 定时 4 通道 1 捕获/比较控制

位	名称	复位	R/W	描述
7	-	0	R0	未使用
6	IM	1	R/W	通道1中断屏蔽
5:3	CMP[2:0]	000	R/W	通道1比较输出模式选择。 当定时器值等于在T4CC1中的比较值时指定输出。 000： 在比较设置输出 001： 在比较清除输出 010： 在比较切换输出 011： 在比较正计数时设置输出，在0清除。否则在比较设置输出，在0清除。 100： 在比较正计数时清除输出，在0设置。否则在比较清除输出，在0设置。 101： 在比较设置输出，在0xFF清除 110： 在比较清除输出，在0x00设置 111： 初始化输出引脚。CMP[2:0]不变
2	MODE	0	R/W	模式。选择定时器4通道1模式 0： 捕获模式 1： 比较模式
1:0	CAP[1:0]	00	R/W	捕获模式选择。00-无捕获，01-在上升沿捕获，10-在下降沿捕获，11-在两个沿都捕获

T4CC1 (0xEF) –定时器 4 通道 1 捕获/比较值

位	名称	复位	R/W	描述
7:0	VAL[7:0]	0x00	R/W	定时器捕获/比较值通道1。当T4CCTL1.MODE=1（比较模式）时写该寄存器会导致T4CC1.VAL[7:0]更新写入值延迟到T4CNT.CNT[7:0]=0x00。

TIMIF (0xD8) – 定时器 1/3/4 中断屏蔽/标志

位	名称	复位	R/W	描述
7	-	0	R0	没有使用
6	OVFIM	1	R/W	定时器1溢出中断屏蔽
5	T4CH1IF	0	R/W0	定时器4通道1中断标志 0：无中断未决 1：中断未决
4	T4CH0IF	0	R/W0	定时器4通道0中断标志 0：无中断未决 1：中断未决
3	T4OVFIF	0	R/W0	定时器4溢出中断标志 0：无中断未决 1：中断未决
2	T3CH1IF	0	R/W0	定时器3通道1中断标志 0：无中断未决 1：中断未决
1	T3CH0IF	0	R/W0	定时器3通道0中断标志 0：无中断未决 1：中断未决
0	T3OVFIF	0	R/W0	定时器3溢出中断标志 0：无中断未决 1：中断未决

睡眠定时器

睡眠定时器用于设置系统进入和退出低功耗睡眠模式之间的周期。睡眠定时器还用于当进入低功耗睡眠模式时，维持定时器 2 的定时。

睡眠定时器的主要功能如下：

- 24 位的定时器正计数器，运行在 32kHz 的时钟频率
- 24 位的比较器，具有中断和 DMA 触发功能
- 24 位捕获

标题 页

11.1 概述

睡眠定时器是一个 24 位的定时器，运行在一个 32kHz 的时钟频率（可以是 RCOSC 或 XOSC）上。定时器在复位之后立即启动，如果没有中断就继续运行。定时器的当前值可以从 SFR 寄存器 ST2:ST1:ST0 中读取。

11.2 定时器比较

当定时器的值等于 24 位比较器的值，就发生一次定时器比较。通过写入寄存器 ST2:ST1:ST0 来设置比较值。当 STLOAD.LDRDY 是 1 写入 ST0 发起加载新的比较值，即写入 ST2、ST1 和 ST0 寄存器的最新的值。

加载期间 STLOAD.LDRDY 是 0，软件不能开始一个新的加载，直到 STLOAD.LDRDY 回到 1。读 ST0 将捕获 24 位计数器的当前值。因此，ST0 寄存器必须在 ST1 和 ST2 之前读，以捕获一个正确的睡眠定时器计数值。当发生一个定时器比较，中断标志 STIF 被设置。每次系统时钟，当前定时器值就被更新。因此，当从 PM1/2/3（这期间系统时钟关闭）返回，如果尚未在 32kHz 时钟上检测到一个正时钟边沿，ST2:ST1:ST0 中的睡眠定时器值不更新，要保证读出一个最新的值，必须在读睡眠定时器值之前，在 32kHz 时钟上通过轮询 SLEEPSTA.CLK32K 位，等待一个正的变换。

ST 中断的中断使能位是 IEN0.STIE，中断标志是 IRCON.STIF。

当运行在所有供电模式，除了 PM3 时，睡眠定时器将开始运行。因此，睡眠定时器的值在 PM3 下不保存。在 P1 和 PM2 下睡眠定时器比较事件用于唤醒设备，返回主动模式的主动操作。复位之后的比较值的默认值是 0xFFFFFF。

睡眠定时器比较还可以用作一个 DMA 触发（表 8-1 中的 DMA 触发 11）。

注意如果电压降到 2V 以下同时处于 PM2，睡眠间隔将会受到影响。

11.3 定时器捕获

当设置了已选 I/O 引脚的中断标志，且 32 kHz 时钟检测到这一事件时，发生定时器捕获。睡眠定时器通过设置将要用作触发捕获的 I/O 引脚的 STCC.PORT[1:0]和 STCC.PIN[2:0]使能。当 STCS.VALID 变为高电平，即可读 STCV2:STCV1:STCV0 的捕获值。捕获值多于在 I/O 引脚上的事件瞬间的值，因此如果时序需要，软件必须从捕获的值中间抽取一个。要使能一个新的捕获，遵循以下步骤：

1. 清除 STCS.VALID。
2. 等待直到 SLEEPSTA.CLK32K 变为低电平。
3. 等待直到 SLEEPSTA.CLK32K 变为高电平。
4. 清除 P0IFG/P1IFG/P2IFG 寄存器中的引脚中断标志。

这一顺序使用 P0.0 上的上升沿为例，见图 11-1。

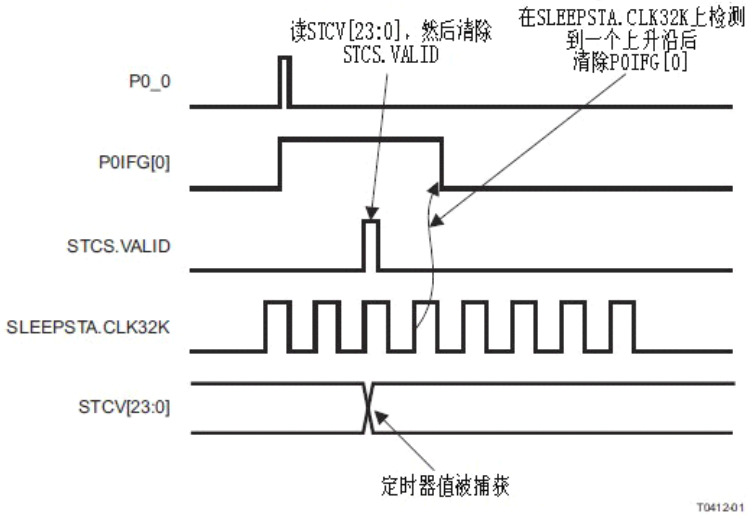


图 11-1　睡眠定时器捕获（使用 P0_0 的上升沿为例）

当捕获使能，不能切换输入捕获引脚。在选择一个新的输入捕获引脚之前，捕获必须禁用。要禁用捕获，遵循以下步骤（如果禁用了中断，使用高达一个 32 kHz 周期（~15.26 us）的一半即可）：

1. 禁用中断
2. 等待直到 SLEEPSTA.CLK32K 变为高电平。
3. 设置 STCC.PORT[1:0]为 3。这将禁用捕获。

11.4　睡眠定时器寄存器

睡眠定时器使用的寄存器是：

- ST2 – 睡眠定时器 2
- ST1 – 睡眠定时器 1
- ST0 – 睡眠定时器 0
- STLOAD – 睡眠定时器加载状态
- STCC – 睡眠定时器捕获控制
- STCS – 睡眠定时器捕获状态
- STCV0 – 睡眠定时器捕获值字节 0
- STCV1 – 睡眠定时器捕获值字节 1
- STCV2 – 睡眠定时器捕获值字节 2

ST2 (0x97) – 休眠定时器 2

位	名称	复位	R/W	描述
7:0	ST2[7:0]	0x00	R/W	休眠定时器计数/比较值。当读取时,该寄存器返回休眠定时器的高位[23:16]。当写该寄存器的值设置比较值的高位[23:16]。在读寄存器ST0的时候值的读取是锁定的。当写ST0的时候写该值是锁定的。

ST1 (0x96) – 休眠定时器 1

位	名称	复位	R/W	描述
7:0	ST1[7:0]	0x00	R/W	休眠定时器计数/比较值。当读取的时候，该寄存器返回休眠定时计数的中间位[15:8]。当写该寄存器的时候设置比较值的中间位[15:8]。在读取寄存器ST0的时候读取该值是锁定的。当写ST0的时候写该值是锁定的。

ST0 (0x95) – 休眠定时器 0

位	名称	复位	R/W	描述
7:0	ST0[7:0]	0x00	R/W	休眠定时器计数/比较值。当读取的时候，该寄存器返回休眠定时计数的低位[7:0]。当写该寄存器的时候设置比较值的低位[7:0]。写该寄存器被忽略，除非STLOAD.LDRDY是1。

STLOAD (0xAD) – 睡眠定时器加载状态

位	名称	复位	R/W	描述
7:1	-	0000 000	R0	保留
0	LDRDY	1	R	加载准备好。当睡眠定时器加载24位比较值，该位是0。当睡眠定时器准备好开始加载一个新的比较值，该位是1。

STCC (0x62B0) – 睡眠定时器捕获控制

位	名称	复位	R/W	描述
7:5	-	000	R0	保留
4:3	PORT[1:0]	11	R	端口选择。有效设置是0-2。当设置为3捕获禁用，即选择了一个无效设置。
2:0	PIN[2:0]	111		引脚选择。当PORT[1:0]是0或1有效设置是0-7，当PORT[1:0]是2有效设置是0-5。当选择了一个无效设置捕获禁用。

STCS (0x62B1) – 睡眠定时器捕获状态

位	名称	复位	R/W	描述
7:1	-	0000 000	R0	保留
0	VALID	0	R/W0	捕获有效标志。当STCV中的捕获值已被更新时设置为1。清除表示允许一个新的捕获。

STCV0 (0x62B2) – 睡眠定时器捕获值字节 0

位	名称	复位	R/W	描述
0	STCV[7:0]	0x00	R	睡眠定时器捕获值的位[7:0]。

STCV1 (0x62B3) –睡眠定时器捕获值字节 1

位	名称	复位	R/W	描述
0	STCV[15:8]	0x00	R	睡眠定时器捕获值的位[15:8]。

STCV2 (0x62B4) –睡眠定时器捕获值字节 2

位	名称	复位	R/W	描述
0	STCV[23:16]	0x00	R	睡眠定时器捕获值的位[23:16]。

ADC

ADC 支持 14 位的模拟数字转换，具有多达 12 位的 ENOB。它包括一个模拟多路转换器，具有多达 8 个各自可配置的通道，以及一个参考电压发生器。转换结果通过 DMA 写入存储器。还具有若干运行模式。

标题 页

12.1 ADC 简介

ADC 支持多达 14 位的模拟数字转换，具有多达 12 位的 ENOB（有效数字位）。它包括一个模拟多路转换器，具有多达 8 个各自可配置的通道；以及一个参考电压发生器。转换结果通过 DMA 写入存储器。还具有若干运行模式。

ADC 的主要特性如下：

- 可选的抽取率，这也设置了分辨率（7 到 12 位）
- 8 个独立的输入通道，可接受单端或差分信号
- 参考电压可选为内部单端、外部单端、外部差分或 AVDD5
- 产生中断请求
- 转换结束时的 DMA 触发
- 温度传感器输入
- 电池测量功能

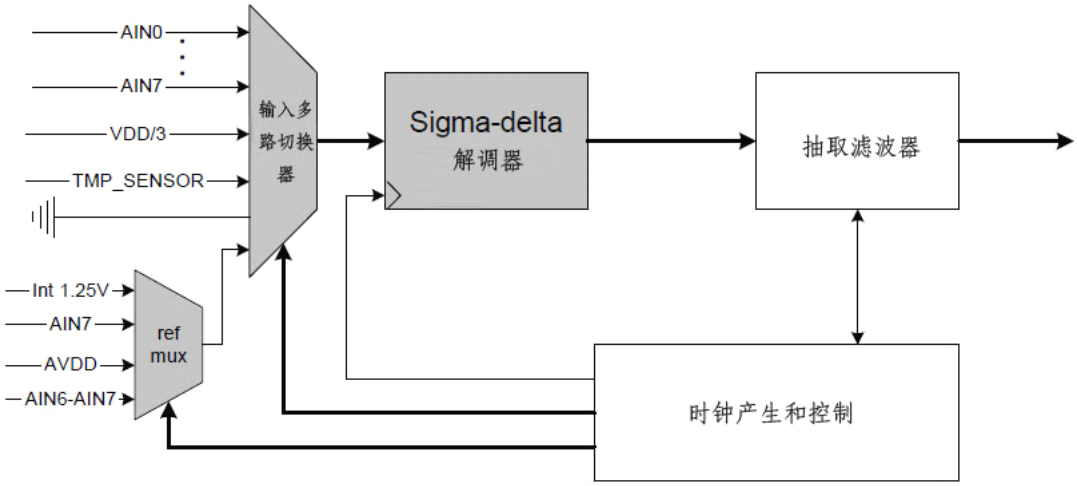


图 12-1 ADC 方框图

12.2 ADC 操作

本节描述了 ADC 的一般安装和操作，并描述了 CPU 存取的 ADC 控制和状态寄存器的使用。

12.2.1 ADC 输入

端口 0 引脚的信号可以用作 ADC 输入。在下面的描述中，这些端口引脚指的是 AIN0-AIN7 引脚。输入引脚 AIN0-AIN7 是连接到 ADC 的。

可以把输入配置为单端或差分输入。在选择差分输入的情况下，差分输入包括输入对 AIN0-1、AIN2-3、AIN4-5 和 AIN6-7。注意负电压不适用于这些引脚，大于 VDD（未调节电压）的电压也不能。它们之间的差别是在差分模式下转换。它是在差分模式下转换的输入对之间的差。

除了输入引脚 AIN0-AIN7，片上温度传感器的输出也可以选择作为 ADC 的输入，用于温度测量。为此寄存器 TR0.ADCTM 和 ATEST.ATESTCTRL 必须分别按 12.2.10 节和 19.15.3 节所述设置。

还可以输入一个对应 AVDD5/3 的电压作为一个 ADC 输入。这个输入允许诸如需要在应用中实现一个电池监测器的功能。注意在这种情况下参考电压不能取决于电源电压，比如 AVDD5 电压不能用作一个参考电压。

单端电压输入 AIN0 到 AIN7 以通道号码 0 到 7 表示。通道号码 8 到 11 表示差分输入，由 AIN0–AIN1、AIN2–AIN3、AIN4–AIN5 和 AIN6–AIN7 组成。通道号码 12 到 15 表示 GND（12）温度传感器（14），和 AVDD5/3（15）。这些值在 ADCCON2.SCH 和 ADCCON3.SCH 域中使用。

12.2.2 ADC 转换序列

ADC 将执行一系列的转换，并把结果移动到存储器（通过 DMA），不需要任何 CPU 干预。

转换序列可以被 APCFG 寄存器（7.6.6 节）影响，八位模拟输入来自 I/O 引脚，不必经过编程变为模拟输入。如果一个通道正常情况下应是序列的一部分，但是相应的模拟输入在 APCFG 中禁用，那么通道将被跳过。当使用差分输入，处于差分对的两个引脚都必须在 APCFG 寄存器中设置为模拟输入引脚。

ADCCON2.SCH 寄存器位用于定义一个 ADC 转换序列，它来自 ADC 输入。如果 ADCCON2.SCH 设置为一个小于 8 的值，转换序列包括一个转换，来自每个通道，从 0 往上，包括 ADCCON2.SCH 编程的通道号码。当 ADCCON2.SCH 设置为一个在 8 和 12 之间的值，序列包括差分输入，从通道 8 开始，在已编程的通道结束。对于 ADCCON2.SCH 大于或等于 12，序列仅包括所选的通道。

12.2.3 单个 ADC 转换

除了这一转换序列，ADC 可以编程为从任何通道执行一个转换。这样一个转换通过写 ADCCON3 寄存器触发。除非一个转换序列已经正在进行，转换立即开始，在这种情况下序列一完成单个转换就被执行。

12.2.4 ADC 运行模式

本节描述了运行模式和初始化转换。

ADC 有三种控制寄存器：ADCCON1, ADCCON2 和 ADCCON3。这些寄存器用于配置 ADC，并报告结果。

ADCCON1.EOC 位是一个状态位，当一个转换结束时，设置为高电平；当读取 ADCH 时，它就被清除。

ADCCON1.ST 位用于启动一个转换序列。当这个位设置为高电平，ADCCON1.STSEL 是 11，且当前没有转换正在运行时，就启动一个序列。当这个序列转换完成，这个位就被自动清除。

ADCCON1.STSEL 位选择哪个事件将启动一个新的转换序列。该选项可以选择为外部引脚 P2.0 上升沿或外部引脚事件，之前序列的结束事件，定时器 1 的通道 0 比较事件或 ADCCON1.ST 是 1。

ADCCON2 寄存器控制转换序列是如何执行的。

ADCCON2.SREF 用于选择参考电压。参考电压只能在没有转换运行的时候修改。

ADCCON2.SDIV 位选择抽取率（并因此也设置了分辨率和完成一个转换所需的时间，或样本率）。抽取率只能在没有转换运行的时候修改。

转换序列的最后一个通道由 ADCCON2.SCH 位选择，如上所述。

ADCCON3 寄存器控制单个转换的通道号码、参考电压和抽取率。单个转换在寄存器 ADCCON3 写入后将立即发生，或如果一个转换序列正在进行，该序列结束之后立即发生。该寄存器位的编码和 ADCCON2 是完全一样的。

12.2.5 ADC 转换结果

数字转换结果以 2 的补码形式表示。对于单端配置，结果总是为正。这是因为结果是输入信号和地面之间的差值，它总是一个正符号数（Vconv=Vinp-Vinn，其中 Vinn=0V）。当输入幅度等于所选的电压参考 VREF 时，达到最大值。对于差分配置，两个引脚对之间的差分被转换，这个差分可以是负符号数。对于抽取率是 512 的一个数字转换结果的 12 位 MSB，当模拟输入 Vconv 等于 VREF 时，数字转换结果是 2047。当模拟输入等于 -VREF 时，数字转换结果是-2048。

当 ADCCON1.EOC 设置为 1 时，数字转换结果是可以获得的，且结果放在 ADCH 和 ADCL 中。注意转换结果总是驻留在 ADCH 和 ADCL 寄存器组合的 MSB 段中。

当读取 ADCCON2.SCH 位时，它们将指示转换在哪个通道上进行。ADCL 和 ADCH 中的结果一般适用于之前的转换。如果转换序列已经结束，ADCCON2.SCH 的值大于最后一个通道号码，但是如果最后写入 ADCCON2.SCH 的通道号码是 12 或更大，将读回同一个值。

12.2.6 ADC 参考电压

模拟数字转换的正参考电压可选择为一个内部生成的电压，AVDD5 引脚，适用于 AIN7 输入引脚的外部电压，或适用于 AIN6-AIN7 输入引脚的差分电压。

转换结果的准确性取决于参考电压的稳定性和噪音属性。希望的电压有偏差会导致 ADC 增益误差，与希望电压和实际电压的比例成正比。参考电压的噪音必须低于 ADC 的量化噪音，以确保达到规定的 SNR。

12.2.7 ADC 转换时间

ADC 只能运行在 32 MHz XOSC 上，用户不能整除系统时钟。实际 ADC 采样的 4 MHz 的频率由固定的内部划分器产生。执行一个转换所需的时间取决于所选的抽取率。总的来说，转换时间由以下公式给定：

Tconv = (抽取率 + 16) x 0.25 μs。

12.2.8 ADC 中断

当通过写 ADCCON3 触发的一个单个转换完成时，ADC 将产生一个中断。当完成一个序列转换时，不产生一个中断。

12.2.9 ADC DMA 触发

每完成一个序列转换，ADC 将产生一个 DMA 触发。当完成一个单个转换，不产生 DMA 触发。

对于 ADCCON2.SCH 中头 8 位可能的设置所定义的八个通道，每一个都有一个 DMA 触发。当通道中一个新的样本准备转换，DMA 触发是活动的。DMA 触发命名为表 8-1 中的 ADC_CHsd，其中 s 是单端通道，d 是差分通道。

另外，还有一个 DMA 触发 ADC_CHALL，当 ADC 转换序列的任何通道中有新的数据准备好时，它是活动的。

12.2.10 ADC 寄存器

本节描述了 ADC 寄存器。

ADCL (0xBA) – ADC 数据低位

位	名称	复位	R/W	描述
7:2	ADC[5:0]	0000 00	R	ADC转换结果的低位部分
1:0	-	00	R0	没有使用。读出来一直是 0

ADCH (0xBB) – ADC 数据高位

位	名称	复位	R/W	描述
7:0	ADC[13:6]	0x00	R	ADC转换结果的高位部分。

ADCCON1 (0xB4) – ADC 控制 1

位	名称	复位	R/W	描述
7	EOC	0	R/ H0	转换结束。当 ADCH 被读取的时候清除。如果已读取前一数据之前，完成一个新的转换，EOC 位仍然为高。 0： 转换没有完成 1： 转换完成
6	ST	0		开始转换。读为1，直到转换完成 0： 没有转换正在进行 1: 如果 ADCCON1.STSEL = 11并且没有序列正在运行就启动一个转换序列。
5:4	STSEL[1:0]	11	R/W1	启动选择。选择该事件，将启动一个新的转换序列。 00： P2.0引脚的外部触发。 01： 全速。不等待触发器 10： 定时器1通道0比较事件 11： ADCCON1.ST = 1
3:2	RCTRL[1:0]	00	R/W	控制 16 位随机数发生器（第 13 章）。当写 01 时，当操作完成时设置将自动返回到 00。 00： 正常运行。(13X 型展开) 01： LFSR 的时钟一次(没有展开). 10： 保留 11： 停止。关闭随机数发生器
1:0	-	11	R/W	保留。一直设为 11 。

ADCCON2 (0xB5) – ADC 控制 2

位	名称	复位	R/W	描述
7:6	SREF[1:0]	00	R/W	选择参考电压用于序列转换 00: 内部参考电压 01: AIN7 引脚上的外部参考电压 10: AVDD5 引脚 11: AIN6 - AIN7 差分输入外部参考电压
5:4	SDIV[1:0]	01	R/W	为包含在转换序列内的通道设置抽取率。抽取率也决定完成转换需要的时间和分辨率。 00: 64 抽取率(7 位 ENOB) 01: 128 抽取率(9 位 ENOB) 10: 256 抽取率(10 位 ENOB) 11: 512 抽取率(12 位 ENOB)
3:0	SCH[3:0]	0000	R/W	序列通道选择。选择序列结束。一个序列可以是从 AIN0 到 AIN7 (SCH<=7) 也可以从差分输入 AIN0-AIN1 到 AIN6-AIN7 (8<=SCH<=11)。对于其他的设置，只能执行单个转换。 当读取的时候，这些位将代表有转换进行的通道号码。 0000: AIN0 0001: AIN1 0010: AIN2 0011: AIN3 0100: AIN4 0101: AIN5 0110: AIN6 0111: AIN7 1000: AIN0-AIN1 1001: AIN2-AIN3 1010: AIN4-AIN5 1011: AIN6-AIN7 1100: GND 1101: 正电压参考 1110: 温度传感器 1111: VDD/3

ADCCON3 (0xB6) – ADC 控制 3

位	名称	复位	R/W	描述
7:6	EREF[1:0]	00	R/W	选择用于额外转换的参考电压 00： 内部参考电压 01： AIN7 引脚上的外部参考电压 10： AVDD5 引脚 11： 在 AIN6-AIN7 差分输入的外部参考电压
5:4	EDIV[1:0]	00	R/W	设置用于额外转换的抽取率。抽取率也决定了完成转换需要的时间和分辨率。 00： 64 抽取率(7 位 ENOB) 01： 128 抽取率(9 位 ENOB) 10： 256 抽取率(10 位 ENOB) 11： 512 抽取率(12 位 ENOB)
3:0	ECH[3:0]	0000	R/W	单个通道选择。选择写 ADCCON3 触发的单个转换所在的通道号码。当单个转换完成，该位自动清除。 0000： AIN0 0001： AIN1 0010： AIN2 0011： AIN3 0100： AIN4 0101： AIN5 0110： AIN6 0111： AIN7 1000： AIN0-AIN1 1001： AIN2-AIN3 1010： AIN4-AIN5 1011： AIN6-AIN7 1100： GND 1101： 正电压参考 1110： 温度传感器 1111： VDD/3

TR0 (0x624B) – 测试寄存器 0

位	名称	复位	R/W	描述
7:1	-	0000 000	R0	保留。写作0。
0	ACTM	0	R/W	设置为1来连接温度传感器到SOC_ADC。也可参见ATEST寄存器描述来使能19.15.3节的温度传感器

随机数发生器

本章详细介绍了随机数发生器及其用途。

标题 页

13.1 简介

随机数发生器有如下功能。

- 产生伪随机字节，可以被 CPU 读取，或由命令选通处理器直接使用（见 19.14 节）。
- 计算写入到 RNDH 的 CRC16 字节。
- 由写入到 RNDL 的值播种。

随机数发生器是一个 16 位的线性反馈移位寄存器 LFSR，带有多项式 $X^{16} + X^{15} + X^2 + 1$（即 CRC16）。根据执行的操作，它使用不同级别的展开值。基本的形式（不展开）如图 13-1 所示。

当 ADCCON1.RCTRL=11 时，随机数发生器就关闭。

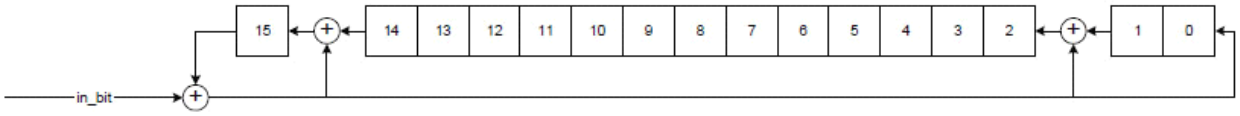


图 13-1 随机数发生器的基本结构

13.2 随机数发生器的运行

随机数发生器的运行是由 ADCCON1.RCTRL 位控制的（也可参见 12.2.10 节）。LFSR 的 16 位移位寄存器的当前值可以从 RNDH 和 RNDL 寄存器中读取。

13.2.1 伪随机数序列的生成

默认操作（ADCCON1.RCTRL 是 00）是命令选通处理器每次读取随机值，就通知 LFSR 一次（不展开的 13x，其中通知不展开的 13x 意味着执行 13 次反馈移位的一个操作等式）。这保证来自 LFSR 末端的 LSB 一个新的伪随机字节的有效性。

更新 LFSR 的另一种方式是设置 ADCCON1.RCTRL 为 01。这将每次通知 LFSR（不展开的 13x），且当操作完成时，ADCCON1.RCTRL 位将自动清除。

13.2.2 种子数的产生

LFSR 可以通过写入 RNDL 寄存器两次产生种子数。每次写入 RNDL 寄存器，LFSR 的 8 位 LSB 复制到 8 位 MSB，8 位 LSB 被替换为写入 RNDL 的新的数据字节。

当需要一个真正的随机值，LFSR 应通过写入 RNDL 产生种子，随机值来自在 RF 接收路径的 IF_ADC。要使用这种产生种子的方法，无线电必须首先上电。无线电应处于无限 TX 状态，以避免 RX 状态可能的同步检测。来自 IF_ADC 的随机值从 RF 寄存器 RFRND 中读出。读出的值作为种子值写入 RNDL 寄存器，如上所述。注意这可以在为正常任务使用无线电时完成。

请注意种子值 0x0000 或 0x8003 将总会导致 LFSR 中的值通知之后不改变，因为没有值通过 in_bit（图 13-1）推入，因此，不能用于随机数的产生。

13.2.3 CRC16

LFSR 也可以用于计算一个字节序列的 CRC 值。写入 RNDH 寄存器的操作将触发一个 CRC 计算。新的字节从 MSB 末端处理，使用一个 8x 的未展开式，这样一个新的字节可以在每个时钟周期写入到 RNDH。

注意在开始 CRC 计算之前，LFSR 必须通过写 RNDL 正确产生随机数。通常产生的用于 CRC 计算的种子数值应该是 0x0000 或 0xFFFF。

13.3 随机数发生器的寄存器

本节描述了随机数发生器的寄存器。

RNDL (0xBC) – 随机数发生器数据低字节

位	名称	复位	R/W	描述
[7:0]	RNDL[7:0]	0xFF	R/W	随机值/种子或CRC结果，低字节 当用作随机数产生，写入这个寄存器两次将把随机数播种到发生器。写该寄存器复制LFSR的8 LSB到8 MSB，并且用写入的数据值取代8 LSB。 读时从该寄存器返回LFSR的8 LSB。 当用作随机数产生，读这个寄存器返回随机数的8 LSB。当用作CRC计算，读这个寄存器返回CRC结果。

RNDH (0xBD) – 随机寄存器 RNDH

位	名称	复位	R/W	描述
[7:0]	RNDH[7:0]	0xFF	R/W	随机值或CRC结果/输入数据，高字节 当写的时候, 将触发一个CRC 16计算，写入数据值的处理开始于MSB位。 读时该寄存器返回LFSR的8 MSB。 当用作随机数产生，读这个寄存器返回随机数的8 LSB。当用作CRC计算，读这个寄存器返回CRC结果的8 LSB。

AES 协处理器

高级加密标准（AES）外部协处理器允许以最少的 CPU 参与执行加密/解密。

协处理器具有下列特性：

- 支持 IEEE802.15.4 的全部安全机制
- ECB、CBC、CBF、OFB、CTR 和 CBC～MAC 模式
- 硬件支持 CCM 模式
- 128 位密钥和 IV/Nonce
- DMA 传送触发能力

标题 页

14.1 AES 操作

加密一条消息的步骤如下(ECB, CBC)：

- 装入密码
- 装入初始化向量（IV)
- 为加密/解密而下载/上传数据。

AES 协处理器中，运行 128 位的数据块。数据块一旦装入 AES 协处理器，就开始加密。在处理下一个数据块之前，必须将加密好的数据块读出。每个数据块装入之前，必须将专用的开始命令送入协处理器。

14.2 密钥和 IV

密钥或初始化向量(IV)/当前时间装入之前，应当发送一个合适的装入密钥或 IV/当前时间的命令给协处理器。当装入 IV，设置合适的模式也是非常重要的。

装入密钥或装入 IV 操作，将取消任何正在运行的程序。密钥、当前时间一旦装入，除非重新装入，否则一直有效。

在开始每条消息（而不是消息块）之前，必须下载初始化向量。

通过设备复位，可以清除密钥和初始化向量值。

14.3 填充输入数据

AES 协处理器运行于 128 位数据块。最后一个数据块少于 128 位，因此必须在写入协处理器时，填充 0 到该数据块中。

14.4 和 CPU 通信

CPU 与协处理器之间，利用以下 3 个 SFR 寄存器进行通信：

- ENCCS，加密控制和状态寄存器
- ENCDI，加密输入寄存器
- ENCDO，加密输出寄存器

状态寄存器通过 CPU 直接读/写，而输入/输出寄存器则必须使用存储器直接存取（DMA）。

当使用 DMA 和 AES 协处理器时，有两个 DMA 通道必须使用，其中一个用于数据输入，另一个用于数据输出。在开始命令写入寄存器 ENCCS 之前，DMA 通道必须初始化。写入一条开始命令会产生一个 DMA 触发信号，传送开始。当每个数据块处理完毕时，产生一个中断。该中断用于发送一个新的开始命令到寄存器 ENCCS。

14.5 运行模式

当使用 CFB、OFB 和 CTR 模式时，128 位数据块分为 4 个 32 位的数据块。每 32 位装入 AES 协处理器，加密后再读出，直到 128 位加密完毕。注意，数据是直接通过 CPU 装入和读出的。当使用 DMA 时，就由 AES 协处理器产生的 DMA 触发自动进行，因此首选 DMA 方式。

实现加密和解密的操作类似。

CBC-MAC 模式与 CBC 模式不同。运行 CBC-MAC 模式时，除了最后一个数据块，每次以 128 位的数据块下载到协处理器。最后一个数据块装入之前，运行的模式必须改变为 CBC。当最后一个数据块下载完毕后，

上传的数据块就是 MAC 值了。

CCM 是 CBC-MAC 和 CTR 的结合模式。因此有部分 CCM 必须由软件完成。下面的章节给出了要完成的必要步骤的简短说明。

14.6 CBC-MAC

当运行 CBC-MAC 加密时，除了最后一个数据块，其余都是由协处理器按照 CBC-MAC 模式，每次下载一个数据块。在下载最后一个数据块之前，模式改为运行于 CBC 模式。当最后一个数据块下载完毕后，上传的数据块就是 MAC 消息了。

CBC-MAC 解密与加密类似。上传的 MAC 信息必须通过与 MAC 比较加以验证。

14.7 CCM 模式

CCM 模式下的消息加密，应该按照下列顺序运行（密钥已经装入）：

数据验证阶段

这个阶段发生在下面所示的步骤 1-6 之间。

1.软件将 0 装入 IV。

2.软件创建数据块 B0。数据块 B0 是 CCM 模式中第一个验证的数据块，其结构如图 14-1。

	名称 B0	在CCM模式指定身份验证的第一块														
字节	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
名称	标志	NONCE									L_M					

图 14-1 信息认证计划块 0

其中，NONCE 值没有限制。L-M 是以字节为单位的消息长度。

对于 IEEE802.15.4，NONCE 有 13 个字节，而 L_M 有 2 个字节。

验证标记字节的内容描述在图 14-2 中。

在这个实例中，L 设置为 6。因此，L_1 为 5。M 和 A_Data 可以设置为任意值。

	名称 FLAG/B0		为CCM模式指定认证的标志域					
位	7	6	5	4	3	2	1	0
名称	保留	A_Data	(M-2)/2			L-1		
值	0	x	x	x	x	1	0	1

图 14-2 认证标志字节

3.如果需要（即 A_Data=1）某些添加的验证数据（下文指明，用 a 表示），则软件就会创建 A_Data 的长度域，称为 L(a)：

- (a)如果 1(a)=0（即 A_Data=0），那么 L(a)是一个空字符串。注意 l(a)是用字节表示的。
- (b)如果 $0<l(a)<2^{16}-2^{8}$，则 L(a)是 2 个 l(a)编码的 8 位字节。

添加的验证数据附加到 A_Data 长度域 L(a)。附加的验证数据块用 0 来填充，直到最后一个附加的验证数据块填满。该字符串的长度没有限制。

AUTH_DATA=L(a)+验证数据+(0 填充)

4.最后一个信息数据块用 0 填满（当该信息的长度不是 128 的整数倍时）。

5.软件将 B0 数据块、附加的验证数据块（如果有）和信息连接起来。

输入信息=B0+AUTH-DATA+信息+(信息的 0 填充)

6.一旦由 CBC-MAC 输入信息验证结束，软件将脱离上传的缓冲器。该缓冲器的内容保持不变（M=16），或者保持缓冲器的高位 M 字节不变。与此同时，设置低位为 0（M≠16）。

结果称为 T。

信息加密

7.软件创建密钥数据块 A0。注意，在当前有 CTR 产生的例子中，L=6。其结构如下图 14-3 所示。

注意：在 OFB 模式下，当加密验证数据 T 以产生 U 时，CTR 值必须是 0。当使用 CTR 模式加密信息块时，除了 0 之外，所有的数值都可以用于 CTR 值。

加密标志字节的内容如图 14-4 所示。

	名称 A0					描述 CCM 模式的首个CTR值									
字节	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
名称	标志	NONCE											CTR			

图 14-3 信息加密程序块

	名称 FLAG/A0		描述 为CCM模式指定加密标志域					
位	7	6	5	4	3	2	1	0
名称	保留		-			L-1		
值	0	0	0	0	0	1	0	1

图 14-4 加密标志字节

8.软件通过选择 IV/Nonce 命令装载 A0。只有在选择装入 IV/Nonce 命令时，设置模式为 CFB 或 OFB 才能完成这个操作。

9.软件在验证数据 T 中，调用 CFB 或 OFB 加密。上传缓冲内容保持不变（M=16），至少 M 的首字节不变，其余字节设置为 0（M-16）。这时的结果为 U，后面将会用到。

10.软件立刻调用 CTR 模式，为刚填充完毕的消息块加密。当 CTR 值不是零必须重新装入 IV。

11.加密验证数据 U 附加到加密消息之中。这样给出最后结果 c。

结果 c=加密消息(m)+U

信息解密

CCM 模式解密

在协处理器中，CTR 的自动生成需要 32 位空间。因此最大的消息长度为 128 * 2^{32} 次方位，即 2^{36} 字节。其幂指数可以写入一个 6 位的字中，因而数值 L 设置为 6。要解密一个 CCM 模式已处理好的消息，必须按照下列顺序进行（密码已经装入）：

信息解析阶段

1.软件通过分开 M 的最右面的 8 位组（命名为 U，剩余的其他 8 位字节，称为字符串 C）来分解消息。
2.C 用 0 来填充，直到能够充满一个整数数值的 128 位数据块。
3.U 用 0 来填充，直到能够充满一个 128 位的数据块。
4.软件创建密钥数据块 A0。所用的方法和 CCM 加密一样。
5.软件通过选择 IV/Nonce 命令装入 A0，只有在选择装入 IV/Nonce 命令时，设置模式为 CFB 或 OFB 才能完成这个操作。
6.软件调用 CFB 或 OFB 加密验证数据 U。上传的缓冲器的内容保持不变（M=16），至少这些内容的前 M 个字节保持不变。其余的内容设置为 0（M≠16），此时的结果为 T。
7.软件立刻调用 CTR 模式解密已经加密的消息数据块 C，而不必重新装入 IV/CTR。

参考验证标签生成阶级

这个阶段与 CCM 加密的验证阶段相同。唯一不同的是，此时的结果名称是 MACTag，而不是 T。

信息验证校核阶段

该阶段中，利用软件来比较 T 和 MACTag。

14.8 在层之间共享 AES 协处理器

AES 协处理器是各个层次共享的公共资源。AES 协处理器每次只能用来处理一个实例。因此需要在软件中设置某些标签来安排这个通用源。

14.9 AES 中断

当一个数据块的加密或解密完成时，就产生AES中断（ENC）。该中断的使能位是IEN0.ENCIE，中断标志位是S0CON.ENCIF。

14.10 AES DMA 触发

与 AES 协处理器有关的 DMA 触发有两个，分别是 ENC_DW 和 ENC_UP。当输入数据需要下载到寄存器 ENCDI 时，ENC_DW 有效；当输出数据需要从寄存器 ENCDO 上传时，ENC_UP 有效。

要使 DMA 通道传送数据到 AES 协处理器，寄存器 ENCDI 就需要设置为目的寄存器；而要使 DMA 通道 AES 协处理器接收数据，寄存器 ENCDO 就需要设置为源寄存器。

14.11 AES 寄存器

AES 协处理器寄存器的布局展示在本节中。

ENCCS (0xB3) – 加密控制和状态

位	名称	复位	R/W	描述
7	-	0	R0	没有使用，读出来一直是 0
6:4	MODE[2:0]	000	R/W	加密/解密模式 000： CBC 001： CFB 010： OFB 011： CTR 100： ECB 101： CBC MAC 110： 没有使用 111： 没有使用
3	RDY	1	R	加密/解密准备状态 0： 加密/解密正在进行 1： 加密/解密完成
2:1	CMD[1:0]	0	R/W	当写 1 到 ST 时执行命令 00： 加密块 01： 解密块 10： 加载密钥 11： 加载 IV/临时
0	ST	0	R/W1 H0	通过 CMD 设置启动处理命令。必须被每个命令或者 128 位数据块发出。通过硬件清除。

ENCDI (0xB1) – 加密输入数据

位	名称	复位	R/W	描述
7:0	DIN[7:0]	0x00	R/W	加密输入数据

ENCDO (0xB2) – 加密输出数据

位	名称	复位	R/W	描述
7:0	DOUT[7:0]	0x00	R/W	加密输出数据

看门狗定时器

在 CPU 可能受到一个软件颠覆的情况下，看门狗定时器（WDT）用作一个恢复的方法。当软件在选定时间间隔内不能清除 WDT 时，WDT 必须就复位系统。看门狗可用于受到电气噪音、电源故障、静电放电等影响的应用，或需要高可靠性的环境。如果一个应用不需要看门狗功能，可以配置看门狗定时器为一个间隔定时器，这样可以用于在选定的时间间隔产生中断。

看门狗定时器的特性如下：

- 四个可选的定时器间隔
- 看门狗模式
- 定时器模式
- 在定时器模式下产生中断请求

WDT 可以配置为一个看门狗定时器或一个通用的定时器。WDT 模块的运行由 WDCTL 寄存器控制。看门狗定时器包括一个 15 位计数器，它的频率由 32kHz 时钟源规定。注意用户不能获得 15 位计数器的内容。在所有供电模式下，15 位计数器的内容保留，且当重新进入主动模式，看门狗定时器继续计数。

标题 页

15.1 看门狗模式

在系统复位之后，看门狗定时器就被禁用。要设置 WDT 在看门狗模式，必须设置 WDCTL.MODE[1:0]位为 10 。然后看门狗定时器的计数器从 0 开始递增。在看门狗模式下，一旦定时器使能，就不可以禁用定时器，因此，如果 WDT 位已经运行在看门狗模式下 ，再往 WDCTL.MODE[1:0]写入 00 或 10 就不起作用了。

WDT 运行在一个频率为 32.768 kHz（当使用 32 kHz XOSC）的看门狗定时器时钟上。这个时钟频率的超时期限等于 1.9ms，15.625 ms，0.25 s 和 1s，分别对应 64，512，8192 和 32768 的计数值设置。

如果计数器达到选定定时器的间隔值，看门狗定时器就为系统产生一个复位信号。如果在计数器达到选定定时器的间隔值之前，执行了一个看门狗清除序列，计数器就复位到 0，并继续递增。看门狗清除的序列包括在一个看门狗时钟周期内，写入 0xA 到 WDCTL.CLR[3:0]，然后写入 0x5 到同一个寄存器位。如果这个序列没有在看门狗周期结束之前执行完毕，看门狗定时器就为系统产生一个复位信号。

当看门狗模式下，WDT 使能，就不能通过写入 WDCTL.MODE[1:0]位改变这个模式，且定时器间隔值也不能改变。

在看门狗模式下，WDT 不会产生一个中断请求。

15.2 定时器模式

要在一般定时器模式下设置 WDT，必须把 WDCTL.MODE[1:0]位设置为 11。定时器就开始，且计数器从 0 开始递增。当计数器达到选定间隔值，定时器将产生一个中断请求（IRCON2.WDTIF/IEN2.WDTIE）。

在定时器模式下，可以通过写入 1 到 WDCTL.CLR[0]来清除定时器内容。当定时器被清除，计数器的内容就置为 0。写入 00 或 01 到 WDCTL.MODE[1:0]来停止定时器，并清除它为 0。

定时器间隔由 WDCTL.INT[1:0]位设置。在定时器操作期间，定时器间隔不能改变，且当定时器开始时必须设置。在定时器模式下，当达到定时器间隔时，不会产生复位。

注意如果选择了看门狗模式，定时器模式不能在芯片复位之前选择。

15.3 看门狗定时器寄存器

本节描述了看门狗定时器的寄存器 WDCTL。

WDCTL (0xC9) – 看门狗定时器控制

位	名称	复位	R/W	描述
7:4	CLR[3:0]	0000	R0/W	清除定时器。当0xA跟随0x5写到这些位，定时器被清除（即加载0）。注意定时器仅写入0xA后，在1个看门狗时钟周期内写入0x5时被清除。当看门狗定时器是IDLE为时写这些位没有影响。当运行在定时器模式，定时器可以通过写1到CLR[0]（不管其他3位）被清除为0x0000（但是不停止）。
3:2	MODE[1:0]	00	R/W	模式选择。该位用于启动WDT处于看门狗模式还是定时器模式。当处于定时器模式，设置这些位为IDLE将停止定时器。注意：当运行在定时器模式时要转换到看门狗模式，首先停止WDT，然后启动WDT处于看门狗模式。当运行在看门狗模式，写这些位没有影响。 00：IDLE 01：IDLE（未使用，等于00设置） 10：看门狗模式 11： 定时器模式
1:0	INT[1:0]	00	R/W	定时器间隔选择。这些位选择定时器间隔定义为32 kHz振荡器周期的规定数。注意间隔只能在WDT处于IDLE时改变，这样间隔必须在定时器启动的同时设置。 00： 定时周期×32,768 (~1 s)当运行在32 kHz XOSC 01： 定时周期×8192 (~0.25 s) 10： 定时周期×512 (~15.625 ms) 11： 定时周期×64 (~1.9 ms)

USART

USART0和USART1是串行通信接口，它们能够分别运行于异步UART模式或者同步SPI模式。两个USART具有同样的功能，可以设置在单独的I/O引脚。关于I/O配置参见12.1节。

标题 页

16.1 UART 模式

UART 模式提供异步串行接口。在 UART 模式中，接口使用 2 线或者含有引脚 RXD、TXD、可选 RTS 和 CTS 的 4 线。UART 模式的操作具有下列特点：

- 8 位或者 9 位负载数据
- 奇校验、偶校验或者无奇偶校验
- 配置起始位和停止位电平
- 配置 LSB 或者 MSB 首先传送
- 独立收发中断
- 独立收发 DMA 触发
- 奇偶校验和帧校验出错状态

UART 模式提供全双工传送，接收器中的位同步不影响发送功能。传送一个 UART 字节包含 1 个起始位、8 个数据位、1 个作为可选项的第 9 位数据或者奇偶校验位再加上 1 个或 2 个停止位。注意，虽然真实的数据包含 8 位或者 9 位，但是，数据传送只涉及一个字节。

UART 操作由 USART 控制和状态寄存器 UxCSR 以及 UART 控制寄存器 UxUCR 来控制。这里的 x 是 USART 的编号，其数值为 0 或者 1。

当 UxCSR.MODE 设置为 1 时，就选择了 UART 模式。

16.1.1 UART 发送

当 USART 收/发数据缓冲器、寄存器 UxBUF 写入数据时，该字节发送到输出引脚 TXDx。UxBUF 寄存器是双缓冲的。

当字节传送开始时，UxCSR.ACTIVE 位变为高电平，而当字节传送结束时为低。当传送结束时，UxCSR.TX_BYTE 位设置为 1。当 USART 收/发数据缓冲寄存器就绪，准备接收新的发送数据时，就产生了一个中断请求。该中断在传送开始之后立刻发生，因此，当字节正在发送时，新的字节能够装入数据缓冲器。

16.1.2 UART 接收

当 1 写入 UxCSR.RE 位时，在 UART 上数据接收就开始了。然后 UART 会在输入引脚 RXDx 中寻找有效起始位，并且设置 UxCSR.ACTIVE 位为 1。当检测出有效起始位时，收到的字节就传入到接收寄存器，UxCSR.RX_BYTE 位设置为 1。该操作完成时，产生接收中断。同时 UxCSR.ACTIVE 变为低电平。

通过寄存器 UxBUF 提供收到的数据字节。当 UxBUF 读出时，UxCSR.RX_BYTE 位由硬件清 0。

注意：当应用程序读 UxDBUF，很重要的一点是不清除 UxCSR.RX_BYTE。清除 UxCSR.RX_BYTE 暗示 UART，使得它以为 UART RX 移位寄存器为空，即使它可能保存有未决数据（一般是由于背对背传输）。所以 UART 声明（TTL 为低电平）RT/RTS 线，这会允许数据流进入 UART，导致潜在的溢出。因此 UxCSR.RX_BYTE 标志紧密结合了自动 RT/RTS 功能，因此只能被 SoC UART 本身控制。否则应用程序一般可以经历以下事件：RT/RTS 线保持声明（TTL 为低电平）的状态，即使一个背对背传输清楚地表明应该间歇性地停止数据流。

16.1.3 UART 硬件流控制

当 UxUCR.FLOW 位设置为 1，硬件流控制使能。然后，当接收寄存器为空而且接收使能时，RTS 输出变低。在 CTS 输入变低之前，不会发生字节传送。

16.1.4 UART 特征格式

如果寄存器 UxUCR 中的 BIT9 和奇偶校验位设置为 1，那么奇偶校验产生而且检测使能。奇偶校验计算出来，作为第 9 位来传送。在接收期间，奇偶校验位计算出来而且与收到的第 9 位进行比较。如果奇偶校验出错，则 UxCSR.ERR 位设置为高电平。当读取 UxCSR 时，UxC-SR.ERR 位清除。

要传送的停止位的数量设置为 1 或者 2，这取决于寄存器位 UxUCR. SPB。接收器总是要核对一个停止位。如果在接收期间收到的第一个停止位不是期望的停止位电平，就通过设置寄存器位 UxCSR.FE 为高电平，发出帧出错信号。当读取 UxCSR 时，UxCSR.FE 位清除，当 UxCSR.SPB 设置为 1 时，接收器将核对两个停止位。注意当检测到第一个停止位正确时，将设置 RX 中断。如果第二个停止位不正确，设置帧误码时将有一个延迟。这个延迟与波特率有关（位持续时间）。

16.2 SPI 模式

本节描述了同步通信的 SPI 模式。在 SPI 模式中，USART 通过 3 线接口或者 4 线接口与外部系统通信。接口包含引脚 MOSI、MISO、SCK 和 SS_N。参见 12.1 节中有关如何将 USART 引脚指派到 I/O 引脚的描述。

SPI 模式包含下列特征：

- 3 线（主要）或者 4 线 SPI 接口
- 主和从模式
- 可配置的 SCK 极性和相位
- 可配置的 LSB 或 MSB 传送

当 UxCSR.MODE 设置为 0 时，选中 SPI 模式。

在 SPI 模式中，USART 可以通过写 UxCSR.SLAVE 位来配置 SPI 为主模式或者从模式。

16.2.1 SPI 主模式操作

当寄存器 UxBUF 写入字节后，SPI 主模式字节传送就开始了。USART 使用波特率发生器（见 16.4 节）生成 SCK 串行时钟，而且传送发送寄存器提供的字节到输出引脚 MOSI。与此同时，接收寄存器从输入引脚 MISO 获取收到的字节。

当传送开始 UxCSR.ACTIVE 位变高，而当传送结束后，UxCSR.ACTIVE 位变低。当传送结束时，UxCSR.TX_BYTE 位设置为 1。

串行时钟 SCK 的极性由 UxGCR.CPOL 位选择，其相位由 UxCSR.CPHA 位选择。字节传送的顺序由 UxCSR.ORDER 位选择。

传送结束时，收到的数据字节由 UxBUF 提供读取。当这个新的数据在 UxDBUF USART 接收/发送数据寄存器中准备好，就产生一个接收中断。

当单元就绪接收另一个字节用来发送时，发送中断产生。由于 UxBUF 是双缓冲，这个操作刚好在发送开始时就发生了。注意数据不应写入 UxDBUF，直到 UxCSR.TX_BYTE 是 1。对于 DMA 传输这是自动处理的。对于使用 DMA 的背对背传输，如果传输字节不能被损坏，UxGDR.CPHA 位必须设置为 0。对于需要设置 UxGDR.CPHA 的系统，需要轮询 UxCSR.TX_BYTE。

还要注意发送中断和接收中断的区别，因为前者比后者大约提前 8 位周期到达。

如上所述的 SPI 主模式操作是一个 3 线接口。不选择输入用于使能主模式。如果外部从模式需要一个从模式选择信号，这可以使用一个通用 I/O 引脚通过软件实现。

16.2.2 SPI 从模式操作

SPI 从模式字节传送由外部系统控制。输入引脚 MISO 上的数据传送到接收寄存器，该寄存器由串行时钟 SCK 控制。SCK 为从模式输入。与此同时，发送寄存器中的字节传送到输出引脚 MOSI。

当传送开始时 UxCSR.ACTIVE 位变高，而当传送结束后，UxCSR.ACTIVE 位变低。当传送结束时，UxCSR.RX_BYTE 位设置为 1，接收中断产生。

串行时钟 SCK 的极性由 UxCSR.CPOL 位选择，其相位由 UxGCR.CPHA 位选择。字节传送的顺序由 UxGCR.ORDER 位选择。

传送结束时，收到的数据字节由 UxBUF 提供读取。

当 SPI 从模式操作开始时，发送中断。

16.3 SSN 从模式选择引脚

当 USART 运行在 SPI 模式，配置为 SPI 从模式，从模式选择（SSN）引脚使用一个 4 线接口来作为 SPI 的输入（边沿控制）。SSN 的下降沿，SPI 从模式活跃，在 MOSI 输入上接收数据，在 MOSI 输出上输出数据。SSN 的上升沿，SPI 从模式不活跃，不接收数据。还要注意 SSN 上升沿之后 MISO 输出不是三态。还要注意释放 SSN（SSN 变为高电平）必须在字节接收或发送结束。如果在字节中间释放，下一个要接收的字节将不能正确接收，因为关于之前字节的信息在 SPI 系统中。USART 清除能用于删除这个信息。

在 SPI 主模式中，不使用 SSN 引脚。当 USART 运行在 SPI 主模式，外部 SPI 从模式设备需要提供一个从模式选择信号，然后一个通用 I/O 引脚应在软件方面作为从模式选择信号功能。

16.4 波特率的产生

当运行在 UART 模式时，内部的波特率发生器设置 UART 波特率。当运行在 SPI 模式时，内部的波特率发生器设置 SPI 主时钟频率。

由寄存器 UxBAUD.BAUD_M[7：0]和 UxGCR.BAUD_E[4：0]定义波特率。该波特率用于 UART 传送，也用于 SPI 传送的串行时钟速率。波特率由下式给出：

$$\text{波特率} = \frac{(256 + BAUD\_M) * 2^{BAUD\_E}}{2^{28}} * F \qquad (16\text{-}1)$$

式中：f 是系统时钟频率，等于 16 MHz RCOSC 或者 32 MHz XOSC。

标准波特率所需的寄存器值如表 16-1 所列。该表适用于典型的 32 MHz 系统时钟。真实波特率与标准波特率之间的误差，用百分数表示。

当 BAUD_E 等于 16 且 BAUD_M 等于 0 时，UART 模式的最大波特率是 f/16 且 f 是系统时钟频率。

SPI 模式下的最大波特率见设备数据手册。

注意波特率必须通过 UxBAUD 和寄存器 UxGCR 在任何其他 UART 和 SPI 操作发生之前设置。这意味着使用这个信息的定时器不会更新，直到它完成它的起始条件，因此改变波特率是需要时间的。

表 16-1　32 MHz 系统时钟常用的波特率设置

波特率 (bps)	UxBAUD.BAUD_M	UxGCR.BAUD_E	误差(%)
2400	59	6	0.14
4800	59	7	0.14
9600	59	8	0.14
14400	216	8	0.03
19200	59	9	0.14
28800	216	9	0.03
38400	59	10	0.14
57600	216	10	0.03
76800	59	11	0.14
115200	216	11	0.03
230400	216	12	0.03

16.5　清除 USART

通过设置寄存器位 UxUCR.FLUSH 可以取消当前的操作。这一事件会立即停止当前操作并且清除全部数据缓冲器。应注意在 TX/RX 位中间设置清除位，清除将不会发生，直到这个位结束（缓冲将被立即清除但是知道位持续时间的定时器不会被清除）。因此使用清除位应符合 USART 中断，或在 USART 可以接收更新的数据或配置之前，使用当前波特率的等待时间位。

16.6　USART 中断

每个 USART 都有两个中断：RX 完成中断(URXx)和 TX 完成中断(UTXx)。当传输开始触发 TX 中断，且数据缓冲区被卸载。

USART 的中断使能位在寄存器 IEN0 和寄存器 IEN2 中，中断标志位在寄存器 TCON 和寄存器 IRCON2 中。关于这些寄存器的详细信息见 2.5 节。中断使能和标志总结如下。

中断使能：

- USART0 RX：IEN0.URX0IE
- USART1 RX：IEN0.URX1IE
- USART0 TX：IEN2.UTX0IE
- USART1 TX：IEN2.UTX1IE

中断标志：

- USART0 RX：TCON.URX0IF
- USART1 RX：TCON.URX1IF
- USART0 TX：IRCON2.UTX0IF
- USART1 TX：IRCON2.UTX1IF

16.7　USART DMA 触发

有两个 DMA 触发与每个 USART 相关。DMA 触发由事件 RX 或者 TX 完成激活，也就是说，该事件作为 DMA 中断请求。可以配置 DMA 通道使用 USART 收/发缓冲器（即 UxBUF）作为它的源地址或者目标地址。

DMA 触发的概况参见表 8-1。

16.8 USART 寄存器

本节描述了 USART 的寄存器。对于每个 USART，有 5 个如下的寄存器（x 是 USART 的编号，为 0 或者 1）：

- UxCSR：USARTx 控制和状态；
- UxUCR：USARTx UART 控制；
- UxGCR：USARTx 通用控制
- UxBUF：USART x 接收/发送数据缓冲
- UxBAUD：USART x 波特率控制

U0CSR (0x86) –USART 0 控制和状态

位	名称	复位	R/W	描述
7	MODE	0	R/W	USART模式选择 0: SPI模式 1: UART模式
6	RE	0	R/W	UART接收器使能。注意在UART完全配置之前不使能接收。 0: 禁用接收器 1: 接收器使能
5	SLAVE	0	R/W	SPI主或者从模式选择 0: SPI主模式 1: SPI从模式
4	FE	0	R/W0	UART帧错误状态 0: 无帧错误检测 1: 字节收到不正确停止位级别
3	ERR	0	R/W0	UART奇偶错误状态 0: 无奇偶错误检测 1: 字节收到奇偶错误
2	RX_BYTE	0	R/W0	接收字节状态。URAT模式和SPI从模式。当读U0DBUF该位自动清除，通过写0清除它，这样有效丢弃U0DBUF中的数据。 0: 没有收到字节 1: 准备好接收字节
1	TX_BYTE	0	R/W0	传送字节状态。URAT模式和SPI主模式 0 字节没有被传送 1 写到数据缓存寄存器的最后字节被传送
0	ACTIVE	0	R	USART传送/接收主动状态、在SPI从模式下该位等于从模式选择。 0: USART空闲 1: 在传送或者接收模式USART忙碌

U0UCR (0xC4) – USART 0 UART 控制

位	名称	复位	R/W	描述
7	FLUSH	0	R0/W1	清除单元。当设置时，该事件将会立即停止当前操作并且返回单元的空闲状态。
6	FLOW	0	R/W	UART硬件流使能。用RTS和CTS引脚选择硬件流控制的使用。 0: 流控制禁止 1: 流控制使能
5	D9	0	R/W	UART奇偶校验位。当使能奇偶校验，写入D9的值决定发送的第9位的值，如果收到的第9位不匹配收到字节的奇偶校验，接收时报告ERR。 如果奇偶校验使能， 那么该位设置以下奇偶校验级别。 0: 奇校验 1: 偶校验
4	BIT9	0	R/W	UART 9位数据使能。当该位是1时，使能奇偶校验位传输（即第9位）。如果通过PARITY使能奇偶校验，第9位的内容是通过D9给出的。 0: 8位传送 1: 9位传送
3	PARITY	0	R/W	UART奇偶校验使能。除了为奇偶校验设置该位用于计算，必须使能9位模式。 0: 禁用奇偶校验 1: 奇偶校验使能
2	SPB	0	R/W	UART停止位的位数。选择要传送的停止位的位数 0: 1位停止位 1: 2位停止位
1	STOP	1	R/W	UART停止位的电平必须不同于开始位的电平 0: 停止位低电平 1: 停止位高电平
0	START	0	R/W	UART起始位电平。闲置线的极性采用选择的起始位级别的电平的相反的电平。 0: 起始位低电平 1: 起始位高电平

U0GCR (0xC5) – USART 0 通用控制

位	名称	复位	R/W	描述
7	CPOL	0	R/W	SPI 的时钟极性 0: 负时钟极性 1: 正时钟极性
6	CPHA	0	R/W	SPI 时钟相位 0: 当 *SCK* 从 CPOL 倒置到 CPOL 时数据输出到 *MOSI*, 并且当 SCK 从 CPOL 倒置到 CPOL 时数据输入抽样到 *MISO*。 1: 当 *SCK* 从 CPOL 倒置到 CPOL 时数据输出到 *MOSI*, 并且当 *SCK* 从 CPOL 倒置到 CPOL 时数据输入抽样到 *MISO*。
5	ORDER	0	R/W	传送位顺序 0: LSB 先传送 1: MSB 先传送
4:0	BAUD_E[4:0]	0 0000	R/W	波特率指数值。BAUD_E 和 BAUD_M 决定了 UART 波特率 和 SPI 的主 SCK 时钟频率。

U0BUF (0xC1) – USART 0 接收/传送数据缓存

位	名称	复位	R/W	描述
7:0	DATA[7:0]	0x00	R/W	USART接收和传送数据。当写这个寄存器的时候数据被写到内部，传送数据寄存器。当读取该寄存器的时候，数据来自内部读取的数据寄存器。

U0BAUD (0xC2) – USART 0 波特率控制

位	名称	复位	R/W	描述
7:0	BAUD_M[7:0]	0x00	R/W	波特率小数部分的值。BAUD_E和 BAUD_M决定了UART的波特率和SPI的主SCK时钟频率。

U1CSR (0xF8) – USART 1 控制和状态

位	名称	复位	R/W	描述
7	MODE	0	R/W	USART模式选择 0: SPI模式 1: UART模式
6	RE	0	R/W	启动UART接收器。注意UART完全配置之前不使能接收。 0: 禁用接收器 1: 使能接收器
5	SLAVE	0	R/W	SPI主或者从模式选择 0: SPI主模式 1: SPI从模式
4	FE	0	R/W0	UART帧错误状态 0: 无帧错误检测 1: 字节收到不正确停止位级别
3	ERR	0	R/W0	UART奇偶校验错误状态 0: 无奇偶错误检测 1: 字节收到奇偶错误
2	RX_BYTE	0	R/W0	接收字节状态。UART模式和SPI从模式。当读U0DBUF该位自动清除，通过写0清除它，这样有效丢弃U0DBUF中的数据 0: 没有收到字节 1: 收到字节就绪
1	TX_BYTE	0	R/W0	传送字节状态。UART模式和SPI从模式 0: 字节没有传送 1: 写到数据缓存寄存器的最后字节已经传送
0	ACTIVE	0	R	USART传送/接收主动状态 0: USART空闲 1: USART在传送或者接受模式忙碌

U1UCR (0xFB) – USART 1 UART 控制

位	名称	复位	R/W	描述
7	FLUSH	0	R0/W1	清除单元。当设置的时候，该事件会立即停止当前操作且返回单元的空闲状态。
6	FLOW	0	R/W	启动UART硬件流。选择用RTS和CTS引脚控制的硬件流。 0: 禁用硬件流 1: 使能硬件流
5	D9	0	R/W	UART奇偶校验位。当使能奇偶校验，写入D9的值决定发送的第9位的值，如果收到的第9位不匹配收到字节的奇偶校验，接收时报告ERR。 如果奇偶校验使能， 那么该位设置以下奇偶校验级别。 0: 奇校验 1: 偶校验
4	BIT9	0	R/W	使能UART 9位数据。当该位是1时，使能奇偶校验位传输（即第9位）。如果通过PARITY使能奇偶校验，第9位的内容是通过D9给出的。 0: 8位传送 1: 9位传送
3	PARITY	0	R/W	启动UART奇偶校验。除了为奇偶校验设置该位用于计算，必须使能9位模式。 0: 禁用奇偶校验 1: 使能奇偶校验
2	SPB	0	R/W	UART的停止位的个数。选择传送的停止位个数 0: 停止位1个 1: 停止位2个
1	STOP	1	R/W	UART停止位电平必须不同于起始位电平。 0: 低停止位 1: 高停止位
0	START	0	R/W	UART起始位的电平。闲置线的极性采用选择的起始位级别的电平相反的电平。 0: 起始位低电平 1: 起始位高电平

U1GCR (0xFC) – USART 1 通用控制

位	名称	复位	R/W	描述
7	CPOL	0	R/W	SPI 时钟极性 0: 负时钟极性 1: 正时钟极性
6	CPHA	0	R/W	SPI 时钟相位 0: 当 *SCK* 从 CPOL 倒置到 CPOL 时数据输出到 *MOSI*，当 *SCK* 从 CPOL 倒置到 CPOL 时数据输入抽样到 MISO 1: 当 *SCK* 从 CPOL 倒置到 CPOL 时数据输出到 *MOSI*,并且当 SCK 从 CPOL 倒置到 CPOL 时数据输入抽取到 *MISO* 。
5	ORDER	0	R/W	传送位顺序 0: LSB 先传送 1: MSB 先传送
4:0	BAUD_E[4:0]	0 0000	R/W	波特率指数值。BAUD_E 和 BAUD_M 决定了 UART 波特率和 SPI 的主 SCK 时钟频率。

U1BUF (0xF9) – USART 1 接收/传送数据缓存

位	名称	复位	R/W	描述
7:0	DATA[7:0]	0x00	R/W	USART接收和传送数据。当写该寄存器的时候，数据写到内部，传送数据寄存器。当读该寄存器的时候，数据来自内部读取数据寄存器。

U1BAUD (0xFA) – USART 1 波特率控制

位	名称	复位	R/W	描述
7:0	BAUD_M[7:0]	0x00	R/W	波特率小数部分的值。BAUD_E和BAUD_M决定了UART的波特率和SPI的主SCK时钟频率。

USB 控制器

本节重点描述了 USB 控制器的功能，假定读者可以很好理解 USB 且熟悉使用的术语和概念。详细见通用串行总线规范[3]。

标准的 USB 术语的用途和 IN 和 OUT 有关，即 IN 总是进入主机（PC），OUT 总是从主机中退出。

标题 页

17.1 USB 简介

USB 控制器监控 USB 的相关活动，并处理数据包的传输。

适当响应 USB 中断、上传/下载数据包到/从端口 FIFO 是固件的责任。固件必须能够回复所有来自 USB 主机的标准的请求，并根据 PC 上驱动器中执行的协议工作。

USB 控制器有如下功能：

- 全速操作（高达 12 Mbps）
- 五个端口（除了端口 0）可以用作 IN、OUT 或 IN/OUT，可以配置为批量/中断或同步。
- 可以使用 1 KB SRAM 或 FIFO 存储 USB 数据包
- 端口支持的数据包大小从 8-512 字节
- 支持 USB 数据包的双缓冲

图 17-1 显示了 USB 控制器的方框图。USB PHY 是带有输入和输出驱动的物理层。USB SIE 是串行接口引擎，控制数据包传输到/从端口。USB 控制器通过存储仲裁器连接到系统的其余部分。

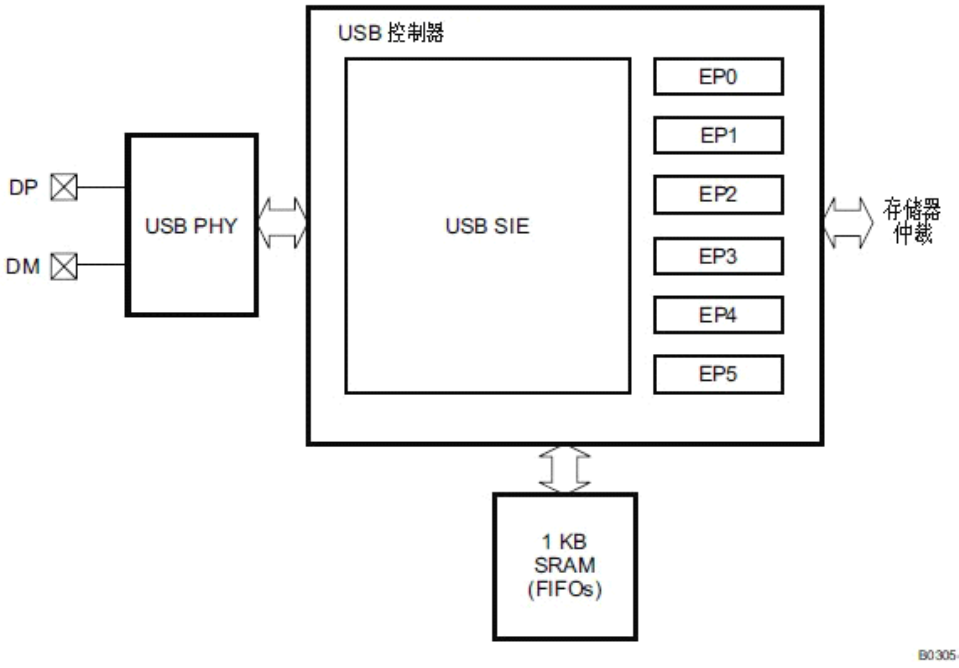


图 17-1 USB 控制器方框图

17.2 USB 使能

USB 通过设置 USBCTRL.USB_EN 为 1 使能。设置 USBCTRL.USB_EN 为 0 复位 USB 控制器。

17.3 48 MHz USB PLL

为了 USB 控制器正确运行，48 MHz 内部 USB PLL 必须上电且稳定。很重要的一点是在 USB PLL 使能之前，晶振必须被选作源且稳定。USB PLL 通过设置 USBCTRL.PLL_EN 位使能，并等待 USBCTRL.PLL_LOCKED 状态标志变为高电平。当 PLL 被锁定，使用 USB 控制器就安全了。

注意：USB 必须在退出活动模式之前禁用，在进入活动模式时重新使能。

17.4 USB 中断

有三种中断标志寄存器及其相关的中断使能屏蔽寄存器。

表 17-1 USB 中断标志中断使能屏蔽寄存器

中断标志	描述	相关的中断使能屏蔽寄存器
USBCIF	包括普通 USB 中断的标志	USBCIE
USBIIF	包括端口 0 和所有 IN 端口的中断标志	USBCIE
USBOIF	包括所有 OUT 端口的中断标志	USBOIE
注意：除了 SOF 和挂起，所有中断复位之后最初使能。		

USB 控制器使用中断#6 用于 USB 中断。这一中断号码与端口 2 输入共享；因此如果使能，中断例程也处理端口 2 中断。要产生一个中断请求，IEN2.P2IE 连同 USBCIE、USBIIE 和 USBOIE 寄存器中期望的中断使能位都必须设置为 1。当产生一个中断请求后，如果没有更高级别的中断未处理，CPU 开始执行 ISR。中断例程必须读所有中断标志寄存器，根据标志的状态采取措施。中断标志寄存器当读后被清除，因此各个中断标志的状态保存在存储器中（一般在栈的一个局部变量中），以允许多次访问它们。

ISR 执行结束，读取中断标志之后，中断标志必须被清除以允许可以检测新的 USB 和 P2 中断。在清除 IRCON2.P2IF 之前必须清除 P2IFG 寄存器中的端口 2 中断状态标志。

当从挂起状态(一般是在 PM1 下)中唤醒，USB D+中断标志 P2IFG.DPIF 被设置。D+中断标志指示在 D+ USB 数据引脚上已经有一个下降沿。这是一个恢复事件。

17.5 端口 0

端口 0（EP0）是一个双向控制端口，枚举阶段期间的所有通信都是通过这个端口执行的。USBADDR 寄存器设置为一个不是 0 的值之前，USB 控制器只能够通过端口 0 通信。设置 USBADDR 寄存器为一个 1 到 127 之间的值，USB 功能超出枚举阶段的默认状态，进入地址状态。所有配置的端口对于应用程序都是可用的。

EP0 FIFO 只能用作 IN 或 OUT，端口 0 不提供双缓冲。端口 0 的最大数据包大小固定是 32 字节。

通过设置 USBINDEX 寄存器为 0，端口 0 通过 USBCS0 寄存器控制。USBCNT0 寄存器包含收到的字节个数。

17.6 端口-0 中断

以下事件可以产生一个 EP0 中断请求：

- 收到了一个数据包（USBCS0.OUTPKT_RDY = 1）
- 加载到 EP0 FIFO 的一个数据包已经被送到 USB 主机。（当准备好传输一个新的数据包，USBCS0.INPKT_RDY 必须设置为 1）。当数据包已经被发送，该位由硬件清除。
- 完成一个 IN 转换（转换状态阶段期间产生一个中断）。
- 发送了一个 STALL（USBCS0.SENT_STALL = 1）
- 一个控制传输过早结束（USBCS0.SETUP_END = 1）

不管 EP0 中断屏蔽位的状态如何，任何一个这些事件都会导致声明 USBIIF.EP0IF。如果 EP0 中断屏蔽位设置为 1，CPU 中断标志 IRCON2.P2IF 也被声明。如果 IEN2.P2IE 和 USBIIE.EP0IE 都设置为 1，才会产生一个中断请求。

17.6.1 错误情况

当发生一个协议错误，USB 控制器发送一个 STALL 握手。如果端口 0 中断正确使能，USBCS0.SENT_STALL 位被声明，且产生一个中断请求。协议错误可以是以下一种情况：

- 设置了 USBCS0.DATA_END 完成 OUT 数据阶段之后，收到一个 OUT 命令（主机尝试发送多于预期的数据）。
- 设置了 USBCS0.DATA_END 完成 IN 数据阶段之后，收到一个 IN 命令（主机尝试接收多于预期的数据）。
- 在 IN 数据阶段期间，USB 主机尝试发送超出最大数据包大小的一个数据包。
- 在状态阶段期间，DATA1 数据包的大小不是 0。

固件也可以通过设置 USBCS0.SEND_STALL 位为 1 终止当前传输。然后 USB 控制器发送一个 STALL 握手，以响应来自 USB 主机的下一个请求。

如果通过声明 USBCS0.SENT_STALL 位导致一个 EP0 中断，该位必须被重新声明，固件必须考虑传输失败（空闲的存储缓存区等等）。

如果数据阶段 EP0 收到一个意外的命令，就声明 USBCS0.SETUP_END 位，且产生一个 EP0 中断（如果正确使能）。然后 EP0 转到 IDLE 状态。然后固件必须设置 USBCS0.CLR_SETUP_END 位为 1，并终止当前的传输。如果声明了 USBCS0.OUTPKT_RDY，这表示收到了另一个固件必须处理的配置数据包。

17.6.2 配置传输（IDLE 状态）

控制传输包括两个或三个传输阶段（配置-数据-状态或配置-状态）。第一个传输是配置传输。一个成功的配置传输包括三个连续的数据包（一个令牌包、一个数据包和一个握手包），其中数据包的数据域（负载）正好是 8 字节长，被称为配置包。在一个控制传输的配置阶段，EP0 处于 IDLE 状态。如果配置包不是 8 字节，USB 控制器拒绝数据包。而且，USB 控制器检查配置包的内容，确定在控制传输中是否有一个数据阶段。如果有一个数据阶段，当 USBCS0.CLR_OUTPKT_RDY 位设置为 1（如果 USBCS0.DATA_END = 0），EP0 转换到 TX 状态（IN 传输）或 RX 状态（OUT 传输）。

当收到一个包，声明 USBCS0.OUTPKT_RDY 位，且如果中断已经被使能就产生一个中断请求（EP0 中断）。当收到一个配置包固件必须执行以下操作：

1. 从 EP0 FIFO 中卸载配置包
2. 检查内容并执行合适的操作
3. 设置 USBCS0.CLR_OUTPKT_RDY 位为 1。这表示配置阶段结束。如果控制传输没有数据阶段，也必须设置 USBCS0.DATA_END。如果没有数据阶段，USB 控制器停留在 IDLE 状态。

17.6.3 IN 传输（TX 状态）

如果控制传输要求数据被发送到主机，在数据阶段中配置阶段后面跟一个或多个 IN 传输。在这种情况下，USB 控制器处于 TX 状态，且只能接受 IN 令牌。一个成功的 IN 传输包括两个或三个连续的包（一个令牌包、一个数据包和一个握手包(1)）。如果要发送多于 32 字节（最大数据包大小），数据包必须分为几个 32 字节的包，随后是剩余的包。如果要发送的字节数是 32 的倍数，剩余的包是零长度的数据包，因为大小小于 32 字节的数据包表示传输结束。

（1）对于同步传输不能有来自主机的握手包

固件必须下载 EP0 FIFO 得第一个数据包，且只要一设置 USBCS0.CLR_OUTPKT_RDY 位就设置 USBCS0.INPKT_RDY 位。当数据包已经被发送就清除 USBCS0.INPKT_RDY，且产生一个 EP0 中断。然后固件可以加载更多需要的数据。每发送一个数据包，就产生一个 EP0 中断。当最后一个数据包加载完毕，除了 USBCS0.INPKT_RDY，固件还必须设置 USBCS0.DATA_END。这开始控制传输的状态阶段。

当状态阶段完成，EP0 转换到 IDLE 状态。如果 USBCS0.SEND_STALL 位设置为 1 状态阶段可能失败。然后声明 USBCS0.SENT_STALL 位，并产生一个 EP0 中断。

如果当接收一个 IN 令牌时 USBCS0.INPKT_RDY 没有设置，USB 控制器回复一个 NAK，表示端口正在工作，但是暂时没有数据要发送。

17.6.4 OUT 传输（RX 状态）

如果控制传输要求从主机接收数据，在数据阶段中配置阶段后面跟一个或多个 OUT 传输。在这种情况下，USB 控制器处于 RX 状态，只能接受 OUT 令牌。一个成功的 OUT 传输包括两个或三个连续的包（一个令牌包、一个数据包和一个握手包(2)）。如果要接收多于 32 字节（最大数据包大小），数据必须分为几个 32 字节的包，随后是一个剩余的包。如果要接收的字节数是 32 的倍数，剩余的包是一个零长度的数据包，因为负载小于 32 字节的一个数据包表示传输结束。

当收到一个数据包，就设置 USBCS0.OUTPKT_RDY 位，并产生一个 EP0 中断。当数据包从 EP0 FIFO 卸载，固件必须设置 USBCS0.CLR_OUTPKT_RDY。当数据包接收完毕（数据包大小小于 32 字节），固件还必须设置 USBCS0.DATA_END 位。这开始控制传输的状态阶段。数据包的大小保存在 USBCNT0 寄存器中。注意该值仅当 USBCS0.OUTPKT_RDY = 1 时有效。

当状态阶段完成，EP0 转换到 IDLE 状态。如果收到的 DATA1 数据包不是一个零长度的数据包，或如果 USBCS0.SEND_STALL 位设置为 1，状态阶段可能失败。然后声明 USBCS0.SENT_STALL 位，并产生一个 EP0 中断。

（2）对于同步传输，没有来自设备的握手数据包。

17.7 端口 1-5

每个端口可以只用作一个 IN，只用作一个 OUT，或 IN/OUT。对于一个 IN/OUT 端口，基本上有两个端口，即一个 IN 端口和一个 OUT 端口和端口号码相关。IN 端口的配置和控制是通过 USBCSIL 和 USBCSIH 寄存器执行的。USBCSIL 和 USBCSIH 寄存器用于配置和控制 OUT 端口。每个 IN 和 OUT 端口可以配置为一个同步（USBCSIH.ISO = 1 和/或 USBCSOH.ISO = 1）或批量/中断（USBCSIH.ISO = 0 和/或 USBCSOH.ISO = 0）端口。USB 控制器处理批量和中断端口是一样的，但是在固件方面有不同的属性。

在访问索引的端口寄存器之前，USBINDEX 寄存器必须有端口号码的值。

17.7.1 FIFO 管理

每个端口都有一定的 FIFO 存储字节，输入和输出的数据包可以使用。表 17-2 显示了端口 1-5 的 FIFO 的大小。固件负责为每个端口正确设置 USBMAXI 和 USBMAXO 寄存器，以防止数据被重写。

当一个端口号码的 IN 和 OUT 端口都不使用双缓冲时，USBMAXI 和 USBMAXO 的总数不能超出端口的 FIFO 的大小。图 17-2a）显示了一个端口的 IN 和 OUT FIFO 存储器是如何组织单缓冲的。IN FIFO 减少从端口存储区域的顶部，而 OUT FIFO 从端口存储区域的底部增长。

当一个端口号码的 IN 或 OUT 端口使用了双缓冲，USBMAXI 和 USBMAXO 的总数不能超出端口的 FIFO 大小的一半。图 17-2b）显示了使用双缓冲的一个端口的 IN 和 OUT FIFO 存储器。注意第二个 OUT 缓冲从存储区域中间开始，向上增长。第二个 IN 缓冲也从存储区域的中间开始，但是向下增长。

要配置一个端口只为 IN，设置 USBMAXO 为 0，要配置一个端口只为 OUT，设置 USBMAXI 为 0。

对于未使用的端口，USBMAXO 和 USBMAXI 都必须设置为 0。

表 17-2　EP1-5 的 FIFO 大小

EP 号码	FIFO 大小（字节）
1	32
2	64
3	128
4	256
5	512

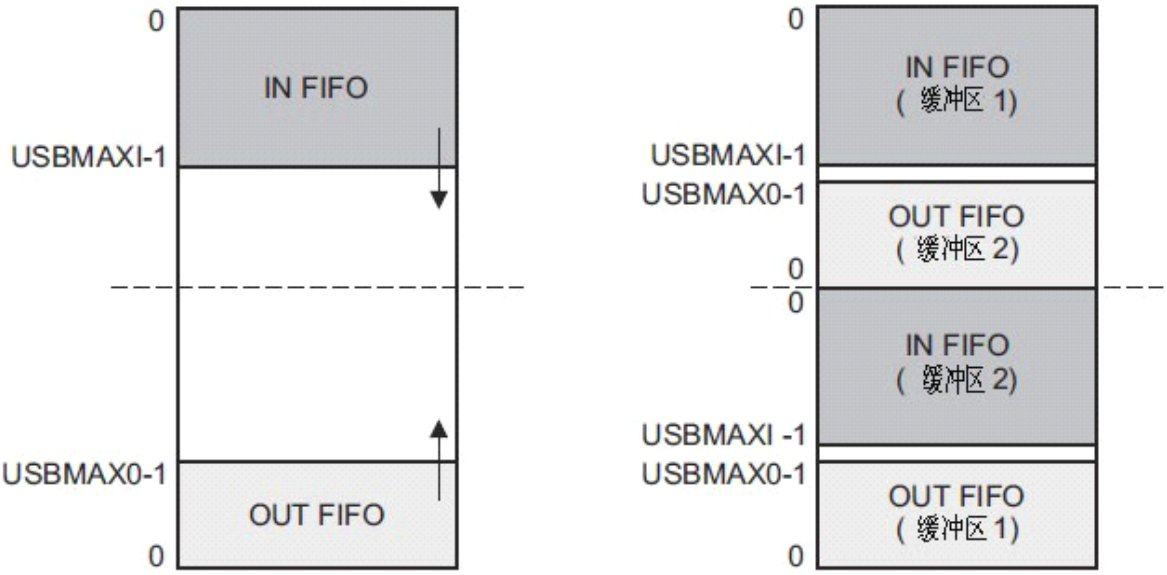


图 17-2　IN/OUT FIFO

17.7.2　双缓冲

要使得传输更快，并降低重新传输的需要，可以使用双缓冲。这允许每个方向上有两个数据包在 FIFO 中缓冲。对于同步端口强烈推荐这一方式，因为它期望每个 USB 帧传输一个数据包，且不带任何重传。对于一个同步的端口，每个 USB 帧发送/接收一个数据包。但是，数据包可能在 USB 帧周期的任何时间发送/接收，因此两个数据包有可能在几微秒的间隔内发送/接收。对于同步端口，如果没有缓冲可用，一个输入数据包就会丢失，且当 USB 主机请求数据，如果没有数据包准备好传输，就发送一个零长度的数据包。对于批量和中断端口，双缓冲不像对于同步端口一样那么关键，因为数据包不会丢失。但是双缓冲可以提高批量端口的有效数据率。

要为一个 IN 端口使能双缓冲，USBCSIH.IN_DBL_BUF 必须设置为 1。要为一个 OUT 端口使能双缓冲，设置 USBCSOH.OUT_DBL_BUF 为 1。

17.7.3 FIFO 访问

端口 FIFO 通过读和写寄存器 USBF0–USBF6 访问。写一个寄存器会导致写入的字节插入到 IN FIFO。读一个寄存器会导致抽取出 OUT FIFO 的下一个字节，并返回这一字节的值。

当一个数据包被写到 IN FIFO，USBCSIL.INPKT_RDY 位必须设置为 1。如果双缓冲使能，USBCSIL.INPKT_RDY 位在写入之后立即被清除，可以加载另一个数据包。这不会产生一个 IN 端口中断，因为只有当已经发送一个数据包才会产生一个中断。当使用双缓冲，固件必须在写 IN FIFO 之前检查 USBCSIL.PKT_PRESENT 位的状态。如果该位是 0，可以写入两个数据包。第一次加载 IN FIFO 时，同步于端口的双缓冲只能加载两个数据包。之后，每个 USB 帧就加载一个数据包。要发送一个零长度的数据包，USBCSIL.INPKT_RDY 必须设置为 1，且不往 IN FIFO 中加载一个数据包。

当 USBCSOL.OUTPKT_RDY 位是 1，可以从 OUT FIFO 读取一个数据包。如果使能，当这种情况发生就产生一个中断。数据包的大小保存在 USBCNTH:USBCNTL 寄存器中。注意仅当 USBCSOL.OUTPKT_RDY = 1 时这个值才有效。当从 OUT FIFO 读取数据包，USBCSOL.OUTPKT_RDY 位必须被清除。如果双缓冲使能，FIFO 中可以有两个数据包。如果当清除 USBCSOL.OUTPKT_RDY 位时另一个数据包准备好，就立即声明 USBCSOL.OUTPKT_RDY 位，且产生一个中断（如果使能），表示已经收到一个新的数据包。当 OUT FIFO 中有两个数据包就设置 USBCSOL.FIFO_FULL 位。

OUT 端口支持自动清除功能。当使能时，当已经从 OUT FIFO 读取 USBMAXO 字节，USBCSOL.OUTPKT_RDY 位被自动清除。自动清除功能通过设置 USBCSOH.AUTOCLEAR = 1 使能。自动清除功能可以用于减少数据包占据 OUT FIFO 缓冲区的时间，一般用于批量端口。

IN 端口支持一个补充的自动设置功能。当使能时，当已经从 IN FIFO 读取 USBMAXI 字节，USBCSIL.INPKT_RDY 位被自动设置。自动设置功能通过设置 USBCSIH.AUTOSET= 1 使能。自动设置功能可以用于减少发送一个数据包需要的总时间，一般用于批量端口。

17.7.4 端口 1-5 中断

以下事件可以产生一个 IN EPx 中断请求（x 表示端口号码）：

- 加载到 IN FIFO 的一个数据包已经被发送到 USB 主机。（当一个新的数据包准备好传输，USBCSIL.INPKT_RDY 必须设置为 1。当数据包已经被发送，该位由硬件清除。）
- 一个 STALL 被发送（USBCSIL.SENT_STALL = 1）。只有批量/中断端口可以被停止。
- 由于 USBCSIH.FLUSH_PACKET 位设置为 1，IN FIFO 被清除。

不管 IN EPx 中断屏蔽位 USBIIE.INEPxIE 的状态，以上任何一个事件都会导致声明 USBIIF.INEPxIF。如果 IN EPx 中断屏蔽位设置为 1，CPU 中断标志 IRCON2.P2IF 也被声明。仅当 IEN2.P2IE 和 USBIIE.INEPxIE 都设置为 1，才会产生一个中断请求。寄存器名称中的 x 指的是端口号码 1-5。

以下事件可以产生一个 OUT EPx 中断请求：

- 收到一个数据包（USBCSOL.OUTPKT_RDY = 1）。
- 发送一个 STALL（USBCSIL.SENT_STALL = 1）。只有批量/中断端口可以被停止。

不管 OUT EPx 中断屏蔽位 USBOIE.OUTEPxIE 的状态，以上任何一个事件都会导致声明 USBOIE.OUTEPxIE。如果 OUT EPx 中断屏蔽位设置为 1，CPU 中断标志 IRCON2.P2IF 也被声明。仅当 IEN2.P2IE 和 USBOIE.OUTEPxIE 都设置为 1，才会产生一个中断请求。

17.7.5 批量/中断 IN 端口

中断 IN 传输以一定间隔发生，而批量 IN 传输使用没有分配给同步、中断或控制传输的可用带宽。

中断 IN 端口可以设置 USBCSIH.FORCE_DATA_TOG 位。当该位设置，不管是否收到一个 ACK，数据切换位不断切换。该功能一般由用于同步端口的通信速度反馈的中断 IN 端口使用。

批量/中断 IN 端口可以通过设置 USBCSIL.SEND_STALL 位为 1 停止。当端口被停止，USB 控制器响应一个 STALL 握手到 IN 令牌。然后设置 USBCSIL.SENT_STALL 位，且如果使能，产生一个中断。

长于最大数据包大小的批量传输通过把传输分为几个数据包执行，每个数据包是最大大小，后面是一个较小的数据包，包括剩余的字节。如果传输长度是最大数据包大小的倍数，最后发送的是一个零长度的数据包。这意味着大小小于最大数据包大小的一个数据包表示传输结束。自动设置功能在这种情况下是有用的，因为许多数据包都是最大大小。

17.7.6 同步 IN 端口

同步 IN 端口用于从 USB 控制器到主机传输定期的数据（每个 USB 帧一个数据包）。

如果当 USB 主机请求数据时在 IN FIFO 中没有数据包加载，USB 控制器发送一个零长度的数据包，并且声明 USBCSIL.UNDERRUN 位。

双缓冲请求一个数据包加载到 IN FIFO，位置在帧期间，即帧要被发送到的地方前面。如果在收到 IN 令牌之前加载了第一个数据包，数据包像它被加载一样在同一个帧期间被发送，因此违反了双缓冲策略。因此，当使用双缓冲，USBPOW.ISO_WAIT_SOF 位必须设置为 1，以避免这种情况。设置该位保证直到收到下一个 SOF 令牌才会发送被加载的数据包。

自动设置功能一般不用于同步端口，因为根据帧的不同数据包的大小会增加或减少。

17.7.7 批量/中断 OUT 端口

中断 OUT 传输以一定间隔发生，而批量 OUT 传输使用没有分配给同步、中断或控制传输的可用带宽。

批量/中断 OUT 端口可以通过设置 USBCSOL.SEND_STALL 位为 1 停止。当端口被停止，当主机完成发送数据包，USB 控制器响应一个 STALL 握手。数据包被丢弃，不放在 OUT FIFO 中。当发送了 STALL 握手，USB 控制器声明 USBCSOL.SENT_STALL 位，并且如果 OUT 端口中断使能，产生一个中断请求。

正如自动设置功能对批量 IN 端口有用一样，自动设置功能对 OUT 端口也有用，因为许多数据包都是最大大小。

17.7.8 同步 OUT 端口

同步 OUT 端口用于从主机到 USB 控制器传输定期的数据（每个 USB 帧一个数据包）。

如果当收到一个数据包时没有可用的缓冲区，就声明 USBCSOL.OVERRUN 位，数据包会丢失。固件可以通过使用双缓冲并使用 DMA 有效卸载数据包减少这种情况的发生。

OUT FIFO 中的同步数据包可能有位错误。硬件检测到这一情况，就设置 USBCSOL.DATA_ERROR。因此当卸载一个数据包时固件必须总是检查该位。

自动设置功能一般不用于同步端口，因为根据帧的不同数据包的大小会增加或减少。

17.8 DMA

DMA 可以用于填充 IN 端口 FIFO 和清空 OUT 端口 FIFO。相比较使用 8051 CPU，使用 DMA 能明显提高读/写性能。因此强烈推荐使用 DMA，除非时序不是很关键，或只有很少字节要传输。

USB 控制器没有 DMA 触发，这意味着 DMA 传输必须由固件触发。

必须使用以字节为单位的传输。

17.9 USB 复位

当在总线上检测到复位信号，就声明 USBCIF.RSTIF 标志。如果 USBCIE.RSTIE 使能，也声明 IRCON2.P2IF，且如果 IEN2.P2IE = 1 就产生一个中断请求。当一个 USB 复位发生，固件必须采取合适的操作。USB 复位必须使设备处于默认状态，即它只响应地址 0（默认地址）。在枚举阶段期间一般发生一个或多个复位，在 USB 线连接上之后立即发生。

当一个 USB 复位发生， USB 控制器执行以下操作：

- USBADDR 设置为 0。
- USBINDEX 设置为 0。
- 所有 FIFO 被清除。
- USBCS0、USBCSIL、USBCSIH、USBCSOL、USBCSOH 被清除。
- 所有中断（除了 SOF 和挂起）使能。
- 产生一个中断请求（如果 IEN2.P2IE = 1 且 USBCIE.RSTIE = 1）。

当检测到 USB 复位，固件必须关闭所有管道，等待一个新的枚举阶段。

17.10 挂起和恢复

只要 USBPOW.SUSPEND_EN = 1，当 USB 已经连续空闲 3ms，USB 控制器声明 USBCIF.SUSPENDIF 并进入挂起模式。如果 USBCIE.SUSPENDIE 使能，就声明 IRCON2.P2IF，且如果 IEN2.P2IE = 1 就产生一个中断请求。

当处于挂起模式，只有有限的电流来源于 USB。关于这一点的详细信息见 USB 2.0 规范[3]。要符合挂起电流的要求，当检测到挂起设备必须回到 PM1。设备不能进入 PM2 或 PM3，因为这会复位 USB。在进入 PM1 之前，48 MHz USB PLL 必须关闭。这通过设置 USBCTRL.PLL_EN 为 0 完成，并等待 USBCTRL.PLL_LOCKED 被清除。

USB 上任何有效的非空闲信号都会导致声明 USBCIF.RESUMEIF，并产生一个中断请求，如果 USB 恢复中断使能，就唤醒系统。

当系统从挂起醒来（进入主动模式），在 48 MHz USB PLL 激活之前，除了 USBCTRL，不能访问任何 USB 寄存器。这通过设置 USBCTRL.PLL_EN 为 1 完成，并等待直到设置 USBCTRL.PLL_LOCKED。

一个 USB 复位也会从挂起唤醒设备。如果中断使能，会产生一个 USB 恢复中断请求，但是设置 USBCIF.RSTIF 中断标志，而不是 USBCIF.RESUMEIF 中断标志。

17.11 远程唤醒

USB 控制器可以通过发送恢复信号给 USB 总线，从挂起恢复。恢复是通过设置 USBPOW.RESUME 为 1 大约 10ms 执行的。根据 USB 2.0 规范[3]，恢复信号必须出现至少 1ms，不超过 15ms。但是建议保持恢复信号

大约 10ms 的时间。注意支持远程唤醒必须在 USB 描述符中宣布，这样 USB 主机必须授予设备执行远程唤醒的权力（通过一个 SET_FEATURE 请求）。

17.12 USB 寄存器

本节描述了用于控制和状态 USB 的所有 USB 寄存器。USB 寄存器驻留在区域 0x6200–0x622B 的 XDATA 存储空间中。这些寄存器可以分为三组：普通 USB 寄存器、索引端口寄存器和端口 FIFO 寄存器。索引端口寄存器表示当前选择的端口。USBINDEX 寄存器用于选择端口。

USBADDR（0x6200）– 功能地址

位	名称	复位	R/W	描述
7	UPDATE	0	R	当写 USBADDR 寄存器，且地址有效时清除，就设置该位。
6:0	USBADDR[6:0]	000 0000	R/W	设备地址

USBPOW（0x6201）– 功率/控制寄存器

位	名称	复位	R/W	描述
7	ISO_WAIT_SOF	0	R/W	当该位设置为 1，一声明 INPKT_RDY，USB 控制器就开始发送零长度的数据包，直到收到第一个 SOF 令牌。这只适用于同步端口。
6:4	-	000	R0	未使用
3	RST	0	R	复位信号期间，该位设置为 1。
2	RESUME	0	R/W	远程唤醒的设备恢复信号。根据 USB 规范，驱动恢复的持续时间必须至少是 1ms，不超过 15ms。建议该位设置保持在 10ms 左右。
1	SUSPEND	0	R	挂起模式进入。仅当时 SUSPEND_EN = 1 才使用该位。读 USBCIF 寄存器或声明 RESUME 清除该位。
0	SUSPEND_EN	0	R/W	挂起使能。当该位设置为 1，当 USB 空闲 3ms 就进入挂起模式。

USBIIF（0x6202）– IN 端口和 EP0 中断标志

位	名称	复位	R/W	描述
7:6	-	0	R0	未使用
5	INEP5IF	0	R，H0	IN 端口 5 的中断标志。读取时由硬件清除
4	INEP4IF	0	R，H0	IN 端口 4 的中断标志。读取时由硬件清除
3	INEP3IF	0	R，H0	IN 端口 3 的中断标志。读取时由硬件清除
2	INEP2IF	0	R，H0	IN 端口 2 的中断标志。读取时由硬件清除
1	INEP1IF	0	R，H0	IN 端口 1 的中断标志。读取时由硬件清除
0	EP0IF	0	R，H0	端口 0 的中断标志。读取时由硬件清除

USBOIF（0x6204）– OUT-端口中断标志

位	名称	复位	R/W	描述
7:6	-	0	R0	未使用
5	OUTEP5IF	0	R，H0	OUT 端口 5 的中断标志。读取时由硬件清除
4	OUTEP4IF	0	R，H0	OUT 端口 4 的中断标志。读取时由硬件清除
3	OUTEP3IF	0	R，H0	OUT 端口 3 的中断标志。读取时由硬件清除
2	OUTEP2IF	0	R，H0	OUT 端口 2 的中断标志。读取时由硬件清除
1	OUTEP1IF	0	R，H0	OUT 端口 1 的中断标志。读取时由硬件清除
0	-	0	R0	未使用

USBCIF（0x6206）– 普通 USB 中断标志

位	名称	复位	R/W	描述
7:4	-	0	R0	未使用
3	SOFIF	0	R，H0	帧开始中断标志。当读时由硬件清除
2	RSTIF	0	R，H0	复位中断标志。当读时由硬件清除
1	RESUMEIF	0	R，H0	恢复中断标志。当读时由硬件清除
0	SUSPENDIF	0	R，H0	挂起中断标志。当读时由硬件清除

USBIIE (0x6207) – IN 端口和 EP0 中断使能屏蔽

位	名称	复位	R/W	描述
7:6		00	R/W	保留。总是写 00
5	INEP5IE	1	R/W	IN 端口-5 中断使能 0：中断禁用 1：中断使能
4	INEP4IE	1	R/W	IN 端口-4 中断使能 0：中断禁用 1：中断使能
3	INEP3IE	1	R/W	IN 端口-3 中断使能 0：中断禁用 1：中断使能
2	INEP2IE	1	R/W	IN 端口-2 中断使能 0：中断禁用 1：中断使能
1	INEP1IE	1	R/W	IN 端口-1 中断使能 0：中断禁用 1：中断使能
0	EP0IE	1	R/W	IN 端口-0 中断使能 0：中断禁用 1：中断使能

USBOIE (0x6209) – OUT 端口中断使能屏蔽

位	名称	复位	R/W	描述
7:6		00	R/W	保留。总是写 00
5	OUTEP5IE	1	R/W	OUT 端口-5 中断使能 0：中断禁用 1：中断使能
4	OUTEP4IE	1	R/W	OUT 端口-4 中断使能 0：中断禁用 1：中断使能
3	OUTEP3IE	1	R/W	OUT 端口-3 中断使能 0：中断禁用 1：中断使能
2	OUTEP2IE	1	R/W	OUT 端口-2 中断使能 0：中断禁用 1：中断使能
1	OUTEP1IE	1	R/W	OUT 端口-1 中断使能 0：中断禁用 1：中断使能
0	-	1	R0	未使用

USBCIE (0x620B) – 普通 USB 中断-使能屏蔽

位	名称	复位	R/W	描述
7:4		-00	R0	未使用
3	SOFIE	0	R/W	帧开始中断使能 0：中断禁用 1：中断使能
2	RSTIE	1	R/W	复位中断使能 0：中断禁用 1：中断使能
1	RESUMEIE	1	R/W	恢复中断使能 0：中断禁用 1：中断使能
0	SUSPENDIE	0	R/W	挂起中断使能 0：中断禁用 1：中断使能

USBFRML (0x620C) – 当前帧号码（低字节）

位	名称	复位	R/W	描述
7:0	FRAME[7:0]	0x00	R	11 位帧号码的低字节

USBFRMH (0x620D) – 当前帧号码（高字节）

位	名称	复位	R/W	描述
7:3	-	-	R0	未使用
2:0	FRAME[10:8]	000	R	11 位帧号码的 3 MSB

USBINDEX (0x620E) – 当前-端口索引寄存器

位	名称	复位	R/W	描述
7:4	-	-	R0	未使用
3:0	USBINDEX[3:0]	0000	R/W	选择的端口。必须设置为 1-5 之间的一个值

USBCTRL (0x620F) – USB 控制器寄存器

位	名称	复位	R/W	描述
7	PLL_LOCKED	0	R	PLL 锁状态
6:3		-	R0	未使用
2	PLL_LOCK	0	R/W	保留。总是写 0
1	PLL_EN	0	R/W	48 MHz USB PLL 使能。当设置该位，48 MHz USB PLL 开始。但是，在 PLL 锁定之前不能访问 USB，即 PLL_LOCKED 是 1。该位只能在 USB_EN = 1 时设置。 注意：PLL 必须在退出主动模式之前禁用，当进入主动模式重新使能。
0	USB_EN	0	R/W	USB 使能。该位仅当 CHIPID = 0xB5 时设置。当写 0 到该位 USB 控制器复位 。

USBMAXI (0x6210) – IN 端口{1-5}的最大数据包大小

位	名称	复位	R/W	描述
7:0	USBMAXI[7:0]	0x00	R/W	IN 端口的最大数据包大小，单位是 8 字节，由 USBINDEX 寄存器选择。该寄存器的值必须对应端口的标准端口描述符的 wMaxPacketSize 域。该寄存器必须设置为大于端口的可用 FIFO 存储器的值。

USBCS0 (0x6211) – EP0 控制和状态（USBINDEX = 0）

位	名称	复位	R/W	描述
7	CLR_SETUP_END	0	R/W H0	设置该位为 1 来声明寄存器的 SETUP_END 位。该位被自动清除。
6	CLR_OUTPKT_RDY	0	R/W H0	设置该位为 1 来声明寄存器的 OUTPKT_RDY 位。该位被自动清除。
5	SEND_STALL	0	R/W H0	设置该位为 1 来确定当前传输。USB 控制器发送 STALL 握手，该位被声明。
4	SETUP_END	0	R	如果过早结束控制传输，设置该位。FIFO 被清除，且如果中断使能产生一个中断请求（EP0）。设置 CLR_SETUP_END = 1 声明该位。
3	DATA_END	0	R/W H0	该位用于通知一个数据传输结束，必须在以下三种情况下声明： 1：当加载了最后的数据包且 USBCS0.INPKT_RDY 设置为 1 2：当卸载了最后的数据包且 USBCS0.CLR_OUTPKT_RDY 设置为 1 3：当声明了 USBCS0.INPKT_RDY 但没有加载 FIFO（发送一个零长度的数据包） USB 控制器自动清除该位。
2	SENT_STALL	0	R/W H1	当发送了一个 STALL 握手，设置该位。如果中断使能产生一个中断请求（EP0）。该位由固件清除。
1	INPKT_RDY	0	R/W H0	当一个数据包加载到 EP0 FIFO 时设置该位，通知 USB 控制器一个新数据包准备好传输。当数据包已被发送，清除该位，且如果中断使能产生一个中断请求（EP0）。
0	OUTPKT_RDY	0	R	接收数据包。当一个输入的数据包被放在 OUT FIFO 时设置该位。如果中断使能产生一个中断请求（EP0）。设置 CLR_OUTPKT_RDY = 1 声明该位。

USBCSIL (0x6211) – IN EP{1-5}控制和状态，低字节

位	名称	复位	R/W	描述
7	-	-	R0	未使用
6	CLR_DATA_TOG	0	R/W H0	设置该位来复位数据切换到 0。因此设置该位强制下一个数据包成为一个 DATA0 包。该位被自动清除。
5	SENT_STALL	0	R/W	当发送一个 STALL 握手设置该位。FIFO 被清除，声明寄存器的 INPKT_RDY 位。如果中断使能产生一个中断请求（IN EP{1-5}）。该位由硬件清除。
4	SEND_STALL	0	R/W	设置该位使 USB 控制器在收到 IN 令牌时回复一个 STALL 握手。固件必须清除该位来结束 STALL 条件。不能停止一个同步端口，因此，该位仅在 IN 端口配置为批量/中断时生效。
3	FLUSH_PACKET	0	R/W H0	设置为 1 清除准备从 IN FIFO 传输的下一个数据包。该寄存器的 INPKT_RDY 位被清除。如果由于双缓冲 IN FIFO 有两个数据包，该位必须设置两次，以完全清除 IN FIFO。该位自动清除。
2	UNDERRUN	0	R/W	在同步模式，如果当 INPKT_RDY = 0 时收到一个 IN 令牌设置该位，并传输一个零长度数据包响应 IN 令牌。在批量/中断模式，当返回一个 NAK 时设置该位，以响应一个 IN 令牌。该位由固件清除。
1	PKT_PRESENT	0	R	当 IN FIFO 中至少有一个数据包时该位为 1。
0	INPKT_RDY	0	R/W H0	当一个数据包加载到 IN FIFO，设置该位以通知 USB 控制器一个新数据包准备好传输。当数据包已被发送，该位被清除，且如果中断使能产生一个中断请求（IN EP{1-5}）。

USBCSIH (0x6212) – IN EP{1-5}控制和状态，高字节

位	名称	复位	R/W	描述
7	AUTOSET	0	R/W	当该位是 1，当一个最大大小的数据包加载到 IN FIFO，USBCSIL.INPKT_RDY 位自动声明。
6	ISO	0	R/W	选择 IN 端口类型 0：批量/中断 1：同步
5:4		10	R/W	保留。总是写 10
3	FORCE_DATA_TOG	0	R/W	设置该位强制 IN 端口数据切换，来自 IN FIFO 的数据包被清除，即使收到一个 ACK。该功能可用于报告同步端口的速度反馈。
2:1		-	R0	未使用
0	IN_DBL_BUF	0	R/W	双缓冲使能（IN FIFO） 0：双缓冲禁用 1：双缓冲使能

USBMAXO (0x6213) – OUT 端口{1-5}的最大数据包大小

位	名称	复位	R/W	描述
7:0	USBMAXI[7:0]	0x00	R/W	OUT 端口的最大数据包大小，单位是 8 字节，由 USBINDEX 寄存器选择。该寄存器的值必须对应端口的标准端口描述符的 wMaxPacketSize 域。该寄存器必须设置为大于端口的可用 FIFO 存储器的值。

USBCSOL (0x6214) – OUT EP{1 – 5}控制和状态低字节

位	名称	复位	R/W	描述
7	CLR_DATA_TOG	0	R/W H0	设置该位来复位数据切换到 0。因此设置该位强制下一个数据包成为一个 DATA0 包。该位被自动清除。
6	SENT_STALL	0	R/W	当发送一个 STALL 握手设置该位。如果中断使能产生一个中断请求（OUTEP{1-5}）。该位由固件清除。
5	SEND_STALL	0	R/W	设置该位使 USB 控制器在收到 OUT 令牌时回复一个 STALL 握手。固件必须清除该位来结束 STALL 条件。不能停止一个同步端口，因此，该位仅在 IN 端口配置为批量/中断时生效。
4	FLUSH_PACKET	0	R/W H0	设置为 1 清除准备从 OUT FIFO 传输的下一个数据包。该寄存器的 OUTPKT_RDY 位被清除。如果由于双缓冲 OUT FIFO 有两个数据包，该位必须设置两次，以完全清除 OUT FIFO。该位自动清除。
3	DATA_ERROR	0	R	如果收到的数据包中有一个 CRC 或位填充错误，设置该位。当清除 OUTPKT_RDY 时就被清除。仅当 OUT 端口同步时该位有效。
2	UNDERRUN	0	R/W	当一个 OUT 包不能加载到 OUT FIFO，设置该位。固件清除该位。该位仅在同步模式有效。。
1	FIFO_FULL	0	R	当 OUT FIFO 不能加载更多数据包时声明该位。
0	OUTPKT_RDY	0	R/W	当已收到一个包且准备从 OUT FIFO 读时设置该位。如果中断使能产生一个中断请（OUT EP{1-5}）。当包已从 FIFO 卸载就清除该位。

USBCSOH (0x6215) – OUT EP{1 – 5}控制和状态高字节

位	名称	复位	R/W	描述
7	AUTOSET	0	R/W	当该位是 1，当一个最大大小的数据包从 OUT FIFO 卸载，USBCSOL.OUTPKT_RDY 位自动清除。
6	ISO	0	R/W	选择 OUT 端口类型 0：批量/中断 1：同步
5:4		00	R/W	保留。总是写 00
3:1		-	R0	未使用
0	OUT_DBL_BUF	0	R/W	双缓冲使能（OUT FIFO） 0：双缓冲禁用 1：双缓冲使能

USBCNT0 (0x6216) – EP0 FIFO（USBINDEX = 0）收到的字节数

位	名称	复位	R/W	描述
7:6	-	-	R0	未使用
5:0	USBCNT0[5:0]	00 0000	R	EP0 FIFO 收到的字节数。仅当声明 OUTPKT_RDY 时有效

USBCNTL (0x6216) – EP{1 – 5} OUT FIFO 的字节数，低字节

位	名称	复位	R/W	描述
7:0	USBCNT[7:0]	0x00	R	OUT FIFO 收到的字节数的 8LSB，由 USBINDEX 寄存器选择。仅当声明 USBCSOL.OUTPKT_RDY 时有效。

USBCNTH (0x6217) – EP{1 – 5} OUT FIFO 的字节数，高字节

位	名称	复位	R/W	描述
7:3	-	-	R0	未使用
2:0	USBCNT[10:8]	000	R	OUT FIFO 收到的字节数的 38LSB，由 USBINDEX 寄存器选择。仅当声明 USBCSOL.OUTPKT_RDY 时有效。

USBF0 (0x6220) – 端口 0 FIFO

位	名称	复位	R/W	描述
7:0	USBF0[7:0]	0x00	R/W	端口 0 FIFO。读该寄存器从 EP0 FIFO 卸载一个字节。写该寄存器加载一个字节到 EP0 FIFO。 注意：EP0 的 FIFO 存储器用于输入和输出数据包。

USBF1 (0x6222) – 端口 1 FIFO

位	名称	复位	R/W	描述
7:0	USBF1[7:0]	0x00	R/W	端口 1 FIFO 寄存器。读该寄存器从 EP1 OUT FIFO 卸载一个字节。写该寄存器加载一个字节到 EP1 IN FIFO。

USBF2 (0x6224) – 端口 2 FIFO

位	名称	复位	R/W	描述
7:0	USBF2[7:0]	0x00	R/W	端口 2 FIFO 寄存器。读该寄存器从 EP2 OUT FIFO 卸载一个字节。写该寄存器加载一个字节到 EP2 IN FIFO。

USBF3 (0x6226) – 端口 3 FIFO

位	名称	复位	R/W	描述
7:0	USBF3[7:0]	0x00	R/W	端口 3 FIFO 寄存器。读该寄存器从 EP3 OUT FIFO 卸载一个字节。写该寄存器加载一个字节到 EP3 IN FIFO。

USBF4 (0x6228) – 端口 4 FIFO

位	名称	复位	R/W	描述
7:0	USBF4[7:0]	0x00	R/W	端口 4 FIFO 寄存器。读该寄存器从 EP4 OUT FIFO 卸载一个字节。写该寄存器加载一个字节到 EP4 IN FIFO。

USBF5 (0x622A) – 端口 5 FIFO

位	名称	复位	R/W	描述
7:0	USBF5[7:0]	0x00	R/W	端口 5 FIFO 寄存器。读该寄存器从 EP5 OUT FIFO 卸载一个字节。写该寄存器加载一个字节到 EP5 IN FIFO。

定时器 2（MAC 定时器）

定时器 2 主要用于为 802.15.4 CSMA-CA 算法提供定时，以及为 802.15.4 MAC 层提供一般的计时功能。当定时器 2 和休眠定时器一起使用时，即使系统进入低功耗模式也会提供定时功能。定时器运行在 CLKCONSTA.CLKSPD 指明的速度上。如果定时器 2 和睡眠定时器一起使用，时钟速度必须设置为 32 MHz，且必须使用一个外部 32 kHz XOSC 获得精确结果。

定时器 2 的主要特性如下：

- 16 位定时器正计数提供的符号/帧周期，例如：16μs/320μs
- 可变周期可精确到 31.25ns
- 2×16 位定时器比较功能
- 24 位溢出计数
- 2×24 位溢出计数比较功能
- 帧首定界符捕捉功能
- 定时器启动/停止同步于外部 32kHz 时钟以及由睡眼定时器提供定时
- 比较和溢出产生中断
- 具有 DMA 触发功能
- 通过引入延迟可调整定时器值

标题 页

18.1 定时器操作

本节描述了定时器的操作。

18.1.1 概述

当定时器停止时，复位后它将进入定时器 IDLE 模式。当 T2CNF.RUN 设置为 1 时，定时器将启动。然后定时器将进入定时器 RUN 模式。此时定时器要么立即工作要么同步于 32kHz 时钟。关于同步启动和停止的描述见 18.4 节。

一旦定时器运行在 RUN 模式，可通过向 T2CNF.RUN 写入 0 来停止正在运行的定时器。然后定时器将进入 IDLE 模式。停止的定时器要么立即停止工作要么同步于 32kHz 时钟。

18.1.2 正计数

定时器 2 包括一个 16 位定时器，在每个时钟周期递增。计数器值可从寄存器 T2M1:T2M0 中读，寄存器 T2MSEL.T2MSEL 设置为 000。注意当读时 T2M0 寄存器 T2M1 的内容是锁定的，这意味着 T2M0 必须总是首先读。

当定时器空闲，计数器可以通过写寄存器 T2M1:T2M0 修改，寄存器 T2MSEL.T2MSEL 设置为 000。T2M0 必须首先写。

18.1.3 定时器溢出

当定时器所计数值等于所设置的定时器周期值时，发生定时器溢出。当发生定时器溢出时，定时器的值设置为 0x000。如果溢出中断屏蔽位 T2IRQM.TIMER2_PERM 是 1，将产生一个中断请求。不管中断屏蔽位是什么值，此时中断标志位 T2IRQF.TIMER2_PERF 都将设置为 1。

18.1.4 定时器 Delta 递增

定时器周期可以在一个定时周期里面通过写定时器的 delta 值进行调整。当一个定时器正在运行，一个定时器 delta 值写入复用寄存器 T2M1:T2M0，且 T2MSEL.T2MSEL 设置为 000，这时 16 位定时器在它当前计数值处停止计数，一个 delta 计数器开始计数。T2M0 寄存器必须在 T2M1 之前写。delta 计数器从写入的 delta 值起，开始倒计数，直到 0 为止。一旦 delta 计数器达到 0，16 位定时器重新开始计数。

Delta 计数器倒计数的速率与定时器等同。当 delta 倒计数变为 0 时，就不再倒计数了。除非 delta 值再一次写入。用这种方法，可以通过 delta 的值增加定时器周期，从而调整定时器的溢出值。

18.1.5 定时器比较

当定时器的计数值等于设置的 16 位比较值之一时，就发生了定时器比较。当发生定时器比较时，根据达到哪个比较值，中断标志 T2IRQF.TIMER2_COMPARE1F 或 T2IRQF.TIMER2_COMPARE2F 置 1。如果此时相应的中断屏蔽位 T2IRQM.TIMER2_COMPARE1M 或 T2IRQM.TIMER2_COMPARE2M 置 1，还产生一个中断请求。

18.1.6 溢出计数

每当计数器溢出时，24 位的溢出计数器加 1。溢出计数器的值可以从寄存器 T2MOVF2:T2MOVF1:T2MOVF0 中读出，寄存器 T2MSEL.T2MOVEFSEL 设置为 000。该寄存器的锁定如下。

如果想要一个唯一的时间戳，即定时器和溢出计数器都在同一时间锁定，操作如下：读 T2M0，T2MSEL.T2MSEL 设置为 000，T2CTRL.LATCH_MODE 设置为 1。这返回定时器值的低字节，并锁定定时器的高字节和整个溢出计数器，这样时间戳的其余部分准备好被读取。

如果只想要读溢出计数器，不首先读定时器，读 T2MOVF0，T2MSEL.T2MOVFSEL 设置为 000，T2CTRL.LATCH_MODE 设置为 1。这返回溢出计数器的低字节，并锁定溢出计数器的两个最高位字节，这样值准备好被读取。

18.1.7 溢出计数更新

溢出计数器的值可以通过写入寄存器 T2MOVF2:T2MOVF1:T2MOVF0 得到更新，T2MSEL.T2MOVFSEL 设置为 000。总是先写最低位字节，然后写其他三个字节。一旦写高字节，写入就生效。

18.1.8 溢出计数器溢出

当溢出计数器的值等于设置的溢出周期，就发生一个溢出周期事件。当发生该周期事件，溢出计数器设置为 0x00 0000。如果溢出中断屏蔽位 T2IRQM.TIMER2_OVF_PERM 是 1，产生一个中断请求。中断标志位 T2IRQF.TIMER2_OVF_PERF 设置为 1，不管中断屏蔽的值。

18.1.9 溢出计数器比较

可以为溢出计数器设置两个比较值。通过写入寄存器 T2MOVF2:T2MOVF1:T2MOVF0，寄存器 T2MSEL.T2MOVFSEL 设置为 011 或 100，可以设置比较值。当溢出计数器的值等于溢出计数器的比较值之一时，发生溢出计数比较事件。如果此时相应的溢出比较中断屏蔽位 T2IRQM.TIMER2_OVF_COMPARE1M 或 T2IRQM.TIMER2_OVF_COMPARE2M 是 1，就立刻产生一个中断请求。而不管中断屏蔽值是什么，中断标志位 T2IRQF.TIMER2_OVF_COMPARE1F 和 T2IRQF.TIMER2_OVF_COMPARE2F 置 1 。

18.1.10 捕获输入

定时器 2 具有定时器捕获功能，它在无线模块的帧开始定界符(SFD)的状态变高时捕获。

当捕获事件发生时，当前定时器内的数值就送到捕获寄存器中。如果寄存器 T2MSEL.T2MSEL 设置为 001，捕获值可以从寄存器 T2M1:T2M0 中读出。溢出计数值也可以在捕获事件发生时捕获，如果 T2MSEL.T2MOVFSEL 设置为 001，可以从寄存器 T2MOVF2:T2MOVF1:T2MOVF0 中读出。

18.2 中断

定时器有 6 个可以分别屏蔽的中断源。它们是：

- 定时器溢出
- 定时器比较 1
- 定时器比较 2
- 溢出计数溢出
- 溢出计数比较 1
- 溢出计数比较 2

中断标志给定在中断标志 T2 IRQF 寄存器中。中断标志位只能通过硬件设置，且只能通过写 SFR 寄存器清除。

每个中断源可以通过寄存器 T2 IRQM 的屏蔽位加以屏蔽。当设置了相应的屏蔽位时，将产生一个中断，否则，不产生中断。但是不管中断屏蔽位的状态是什么，都要设置中断标志位。

18.3 事件输出（DMA 触发和 CSP 事件）

定时器 2 有两个事件输出 T2_EVENT1 和 T2_EVENT2。它们可以用作 DMA 触发，或用作 CSP 条件指令的条件。事件输出可以分别配置为以下一种事件：

- 定时器溢出
- 定时器比较 1
- 定时器比较 2
- 溢出计数溢出
- 溢出计数比较 1
- 溢出计数比较 2

DMA 触发使用 T2EVTCFG.TIMER2_EVENT1_CFG 和 T2EVTCFG.TIMER2_EVENT2_CFG 配置。

18.4 定时器启动/停止同步

本节描述了定时器启动和停止的同步。

18.4.1 概述

定时器可以通过 32kHz 时钟的上升沿实现开始和停止的同步。注意，这个事件来自一个 32kHz 时钟信号，而该时钟与 32 MHz 的系统时钟同步。因此，有一个周期近似等于 32kHz 时钟周期。除非 32kHz 时钟和 32 MHz XOSC 都运行且稳定，否则不能尝试同步启动和停止。

在同步启动时，定时器要重新装入新计算出来的计数值和溢出值，这样在定时器还未停止的时候，它就能出现。

18.4.2 定时器同步停止

定时器开始运行之后，即进入定时器的 RUN 模式。当 T2CTRL.SYNC 是 1，可以通过将 0 写入 T2CTRL.RUN，来停止同步。当 T2CTRL.RUN 已经调整到 0 之后，定时器继续运行，直到 32kHz 时钟的上升沿采样为 1 为止。此时，定时器停止运行，并且存储当前睡眠定时器值，且 T2CTRL.STATE 从 1 到 0。

18.4.3 定时器同步启动

当定时器处于 IDLE 模式，且 T2CTRL.SYNC 是 1 时，通过把 1 写入 T2CTRL.RUN 开始同步。当 T2CTRL.RUN 已经置 1 后，定时器将保持 IDLE 模式，直到 32kHz 时钟的上升沿被检测出来。当这些发生时，定时器将首先计算新的值，用于 16 位定时器值和 24 位定时器溢出值，这个计算基于当前存储的睡眠定时器值和当前 16 位定时器值。新的定时器 2 值和溢出计数器值装入定时器后，定时器就进入 RUN 模式。T2CTRL.STATE = 1 表示模块正在运行。从 32kHz 时钟上升沿被取样起，同步启动过程经历了 86 个时钟周期。同步的启动和停止功能，需要选择系统时钟频率为 32 MHz。如果系统时钟频率选择为 16 MHz，则需要给新的计算值添加一个偏移。

如果之前没有一个同步停止就完成了同步启动，定时器就加载一个不可预知的值。为了避免这种情况，首先启动定时器同步，然后为随后的停止和启动使能同步模式。

下面给出新的定时器 2 值和溢出计数值的计算方法。事实上，由于定时器 2 的时钟频率和睡眠定时器时钟频率是异步的，且它们之间的比率不是整数。这样，不考虑时钟误差，计算出来的计数器值与理想值之间的误差最大达到±1。

计算新的计数器值和溢出计数值

N_c = CurrentSleepTimerValue

N_{ST} = StoredSleepTimerValue

K_{ck} = ClockRatio = 976.5625(1)

st_w = SleepTimerWidth = 24

P_T = Timer2Period

P_{OVF} = OverflowPeriod

O_{ST} = StoredOverflowCountValue

O_{TICK} = OverflowTicsWhileSleeping

T_{ST} = StoredTimerValue

T_{OH} = Overhead = 86

$N_t = N_c - N_{ST}$

$N_t <= 0$ -> $Nd = 2^{stw} + N_t$; $N_t > 0$ -> $N_d = N_t$

$C = N_d \times K_{ck} + T_{ST} + T_{OH}$（四舍五入至最接近的整数值）

$T = C \bmod P_T$

Timer2Value = T

$$O_{TICK} = \frac{(C - T)}{P_T}$$

$O = (O_{TICK} + O_{ST}) \bmod P_{OVF}$

Timer2OverflowCount = O

（1）定时器 2 时钟频率（32 MHz）和睡眠定时器时钟频率（32 kHz）的时钟比例

对于给定的定时器 2 周期值 P，在定时器 2 同步停止和开始之间有一个最大持续时间，以便启动之后定时器值正确更新。最大值给定考虑睡眠定时器时钟周期，即 32kHz 时钟周期 $T_{ST(max)}$：

$$T_{ST(max)} \le \frac{(2^{24} - 1) \times P + T_{OH}}{K_{ck}}$$

18.5 定时器 2 寄存器

本节列出了 SFR 寄存器与定时器 2 有关的部分。这些寄存器如下：

- T2MSEL--定时器 2 复用寄存器控制
- T2M1--定时器 2 复用计数高字节
- T2M0--定时器 2 复用计数低字节
- T2MOVF2--定时器 2 复用溢出计数 2
- T2MOVF1--定时器 2 复用溢出计数 1
- T2MOVF0--定时器 2 复用溢出计数 0
- T2IRQF--定时器 2 中断标志
- T2IRQM--定时器 2 中断屏蔽
- T2CSPCNF--定时器 2 事件输出配置
- T2CTRL--定时器 2 配置

定时器 2 有一些复用寄存器。这使得所有寄存器适应有限的 SFR 地址空间。内部寄存器列在表 18-1 中，可以通过 T2M0、T2M1、T2MOVF0、T2MOVF1 和 T2MOVF2 直接访问。

表 18-1　内部寄存器

寄存器名称	复位	R/W	功能
t2tim[15:0]	0x0000	R/W	保存 16 位正计数器
t2_cap[15:0]	0x0000	R	保存正计数器最后捕获的值
t2_per[15:0]	0x0000	R/W	保存正计数器的周期
t2_cmp1[15:0]	0x0000	R/W	保存正计数器的比较值 1
t2_cmp2[15:0]	0x0000	R/W	保存正计数器的比较值 2
t2ovf[23:0]	0x0000000	R/W	保存 24 位溢出计数器
t2ovf_cap[23:0]	0x0000000	R	保存溢出计数器最后捕获的值
t2ovf_per[23:0]	0x0000000	R/W	保存溢出计数器的周期
t2ovf_cmp1[23:0]	0x0000000	R/W	保存溢出计数器的比较值 1
t2ovf_cmp2[23:0]	0x0000000	R/W	保存溢出计数器的比较值 2

本节其余部分所列的寄存器可在 SFR 地址空间中直接访问。

T2MSEL (0xC3) – 定时器 2 复用选择

位	名称	复位	R/W	描述
7:0	-	0	R0	保留。读作 0
6:4	T2MOVFSEL	0	R/W	该寄存器的值选择当访问 T2MOVF0、T2MOVF1 和 T2MOVF2 时修改或读的内部寄存器。 000：t2ovf（溢出计数器） 001：t2ovf_cap（溢出捕获） 010：t2ovf_per（溢出周期） 011：t2ovf_cmp1（溢出捕获 1） 100：t2ovf_cmp2（溢出捕获 2） 101 to 111：保留
3	-	0	R0	保留。读作 0
2:0	T2MSEL	0	R/W	该寄存器的值选择当访问 T2M0 和 T2M1 时修改或读的内部寄存器。 000：t2tim（定时器计数值） 001：t2_cap（定时器捕获） 010：t2_per（定时器周期） 011：t2_cmp1（定时器比较 1） 100：t2_cmp2（定时器比较 2） 101 to 111：保留

T2M0 (0xA2) – 定时器 2 复用寄存器 0

位号码	名称	复位	R/W	功能
7:0	T2M0	0	R/W	根据 T2MSEL.T2MSEL 的值，直接返回/修改一个内部寄存器位[7:0]。 当读 T2M0 寄存器，T2MSEL.T2MSEL 设置为 000，且 T2CTRL.LATCH_MODE 设置为 0，定时器（t2tim）值被锁定。 当读 T2M0 寄存器，T2MSEL.T2MSEL 设置为 000，且 T2CTRL.LATCH_MODE 设置为 1，定时器（t2tim）和溢出计数器（t2ovf）值被锁定。

T2M1 (0xA3) – 定时器 2 复用寄存器 1

位号码	名称	复位	R/W	功能
7:0	T2M1	0	R/W	根据 T2MSEL.T2MSEL 的值，直接返回/修改一个内部寄存器位[15:8]。 当读 T2M0 寄存器，T2MSEL.T2MSEL 设置为 000，定时器（t2tim）值被锁定。 当读该寄存器，T2MSEL.T2MSEL 设置为 000，返回 t2tim[15:8]锁定的值。

T2MOVF0 (0xA4) – 定时器 2 复用溢出寄存器 0

位号码	名称	复位	R/W	功能
7:0	T2MOVF0	0	R/W	根据 T2MSEL.T2MOVFSEL 的值，直接返回/修改一个内部寄存器位[7:0]。 当读 T2MOVF0 寄存器，T2MSEL.T2MOVFSEL 设置为 000 且 T2CTRL.LATCH_MODE 设置为 0，溢出计数器值（t2ovf）被锁定。 当读 T2M0 寄存器，T2MSEL.T2MOVFSEL 设置为 000 且 T2CTRL.LATCH_MODE 设置为 0，溢出计数器值（t2ovf）被锁定。

T2MOVF1 (0xA5) – 定时器 2 复用溢出寄存器 1

位号码	名称	复位	R/W	功能
7:0	T2MOVF1	0	R/W	根据 T2MSEL.T2MOVFSEL 的值，直接返回/修改一个内部寄存器位[15:8]。 当读该寄存器，T2MSEL.T2MOVFSEL 设置为 000，返回 t2ovf[15:8]被锁定的值。

T2MOVF2 (0xA6) – 定时器 2 复用溢出寄存器 2

位号码	名称	复位	R/W	功能
7:0	T2MOVF2	0	R/W	根据 T2MSEL.T2MOVFSEL 的值，直接返回/修改一个内部寄存器位[23:16]。 当读该寄存器，T2MSEL.T2MOVFSEL 设置为 000，返回 t2ovf[23:16]被锁定的值。

T2IRQF (0xA1) – 定时器 2 中断标志

位号码	名称	复位	R/W	功能
7:6	-	0	R0	保留。读作 0
5	TIMER2_OVF_COMPARE2F	0	R/W	当定时器 2 溢出计数器到达 t2ovf_cmp2 设置的值，就设置
4	TIMER2_OVF_COMPARE1F	0	R/W	当定时器 2 溢出计数器到达定时器 2 t2ovf_cmp1 设置的值，就设置
3	TIMER2_OVF_PERF	0	R/W	当定时器 2 溢出计数器到达等于 t2ovf_per 的值，就设置
2	TIMER2_COMPARE2F	0	R/W	当定时器 2 计数器到达 t2_cmp2 设置的值，就设置
1	TIMER2_COMPARE1F	0	R/W	当定时器 2 计数器到达 t2_cmp1 设置的值，就设置
0	TIMER2_PERF	0	R/W	当定时器 2 计数器到达等于 t2_per 的值，就设置

T2IRQM (0xA7) – 定时器 2 中断屏蔽

位号码	名称	复位	R/W	功能
7:6	-	0	R0	保留。读作 0
5	TIMER2_OVF_COMPARE2M	0	R/W	使能 TIMER2_OVF_COMPARE2 中断
4	TIMER2_OVF_COMPARE1M	0	R/W	使能 TIMER2_OVF_COMPARE1 中断
3	TIMER2_OVF_PERM	0	R/W	使能 TIMER2_OVF_PER 中断
2	TIMER2_COMPARE2M	0	R/W	使能 TIMER2_COMPARE2 中断
1	TIMER2_COMPARE1M	0	R/W	使能 TIMER2_COMPARE1 中断
0	TIMER2_PERM	0	R/W	使能 TIMER2_PER 中断

T2CTRL (0x94) – 定时器 2 控制寄存器

位号码	名称	复位	R/W	功能
7:4	-	0	R0	保留。读作 0
3	LATCH_MODE	0	R/W	0：读 T2M0，T2MSEL.T2MSEL = 000 锁定定时器的高字节，使它准备好从 T2M1 读。读 T2MOVF0，T2MSEL.T2MOVFSEL= 000 锁定溢出计数器的两个最高字节，使可以从 T2MOVF1 和 T2MOVF2 读它们。 1：读 T2M0，T2MSEL.T2MSEL = 000 一次锁定定时器和整个溢出计数器，使可以从读 T2M1、T2MOVF0、T2MOVF1 和 T2MOVF2 值。
2	STATE	0	R	定时器 2 的状态 0：定时器空闲 1：定时器运行
1	SYNC	1	R/W	0：启动和停止定时器是立即的，即和 clk_rf_32m 同步。 1：启动和停止定时器在第一个正 32 kHz 时钟边沿发生。
0	RUN	0	R/W	写 1 启动定时器，写 0 停止定时器。读时，返回最后写入的值。

T2EVTCFG (0x9C) – 定时器 2 CSP 接口配置

位号码	名称	复位	R/W	功能
7	-	0	R0	保留。读作 0
6:4	TIMER2_EVENT2_CFG	0	R/W	选择触发一个 T2_EVENT2 脉冲的事件 000：t2_per_event 001：t2_cmp1_event 010：t2_cmp2_event 011：t2ovf_per_event 100：t2ovf_cmp1_event 101：t2ovf_cmp2_event 110：保留 111：无事件
3	-	0	R0	保留。读作 0
2:0	TIMER2_EVENT1_CFG	0	R/W	选择触发一个 T2_EVENT1 脉冲的事件 000：t2_per_event 001：t2_cmp1_event 010：t2_cmp2_event 011：t2ovf_per_event 100：t2ovf_cmp1_event 101：t2ovf_cmp2_event 110：保留 111：无事件

无线电

RF 内核控制模拟无线电模块。另外，它在 MCU 和无线电之间提供一个接口，这可以发出命令、读取状态和自动对无线电事件排序。

标题 页

19.1 RF 内核

RF 内核控制模拟无线电模块。另外，它在 MCU 和无线电之间提供一个接口，这可以发出命令、读取状态和自动对无线电事件排序。

FSM 子模块控制 RF 收发器的状态、发送和接收 FIFO，以及大部分动态受控的模拟信号，比如模拟模块的上电/掉电。FSM 用于为事件提供正确的顺序（比如在使能接收器之前执行一个 FS 校准）。而且，它为来自解调器的输入帧提供分布的处理：读帧长度，计算收到的字节数，检查 FCS，最后成功接收帧后，可选的处理自动传输 ACK 帧。它在 TX 执行类似的任务，包括在传输前执行一个可选的 CCA，并在接收一个 ACK 帧的传输结束后自动回到 RX。最后，FSM 控制在调制器/解调器和 RAM 的 TXFIFO/RXFIFO 之间传输数据。

调制器把原始数据转换为 I/Q 信号发送到发送器 DAC。这一行为遵守 IEEE 802.15.4 标准。

解调器负责从收到的信号中检索无线数据。

解调器的振幅信息由**自动增益控制**（AGC）使用。AGC 调整模拟 LAN 的增益，这样接收器内的信号水平大约是个常量。

帧过滤和源匹配通过执行所有操作支持 RF 内核中的 FSM，为了执行帧过滤和源地址匹配，见 IEEE 802.15.4 所定义。

频率合成器（FS）为 RF 信号产生载波。

命令选通处理器（CSP）处理 CPU 发出的所有命令。它还有一个 24 字节的很短的程序存储器，使得它可以自动执行 CSMA-CA 机制。

无线电 **RAM** 为发送数据有一个 FIFO（TXFIFO），为接收数据有一个 FIFO（RXFIFO）。这两个 FIFO 都是 128 字节长。另外，RAM 为帧过滤和源匹配存储参数，为此保留 128 字节。

定时器 2（MAC 定时器）用于为无线电事件计时，以捕获输入数据包的时间戳。这一定时器甚至在睡眠模式下也保持计数。

19.1.1 中断

无线电与 CPU 的两个中断向量有关。它们是 RFERR 中断（中断 0）和 RF 中断（中断 12），分别具有以下功能：

- RFERR：无线电的错误情况，使用这一中断表示。
- RF：使用这一中断表示的来自普通操作的中断。

RF 中断向量结合了 RFIF 的中断。注意这些 RF 中断是上升沿触发的。因此中断在比如 SFD 状态标志从 0 变为 1 时产生。RFIF 中断标志描述在 19.1.2 节。

19.1.2 中断寄存器

有两个主要的中断控制 SFR 寄存器用于使能 RF 和 RFERR 中断。它们是：

- RFERR : IEN0.RFERRIE
- RF : IEN2.RFIE

两个主要的中断标志 SFR 寄存器保存了 RF 和 RFERR 的中断标志。它们是：

- RFERR : TCON.RFERR
- RF : S1CON.RFIF

RF 内核产生的两个中断是 RF 内核中若干源的组合。每个单独的源在 RF 内核中有自己的使能和中断标志。标志可在 RFIRQF0、RFIRQF1 和 RFIERRF 中找到。中断屏蔽可在 RFIRQM0、RFIRQM1 和 RFERRM 中找到。

屏蔽寄存器中的中断使能位用于为两个 RF 中断使能单独的中断源。注意屏蔽一个中断源不会影响标志寄存器中状态的更新。

由于使用 RF 内核中的单独的中断屏蔽， 来自 RF 内核的中断有两层屏蔽，处理这些中断时必须小心。步骤描述如下。

要清除来自 RF 内核的中断，必须清除两个标志，即一个设置在 RF 内核，一个设置在 S1CON 或 TCON（取决于触发哪个中断）。如果 RF 内核中的一个标志被清除，还有其他未屏蔽的标志存在，产生另一个中断。

RFIRQF0 (0xE9) – RF 中断标志

位	名称	复位	R/W	描述
7	RXMASKZERO	0	R/W0	RXENABLE寄存器从一个非零状态到全零状态。 0: 无中断未决 1: 中断未决
6	RXPKTDONE	0	R/W0	接收到一个完整的帧。 0: 没有中断未决 1: 中断未决
5	FRAME_ACCEPTED	0	R/W0	帧经过了帧过滤。 0: 没有中断未决 1: 中断未决
4	SRC_MATCH_FOUND	0	R/W0	源匹配发现。 0: 没有中断未决 1: 中断未决
3	SRC_MATCH_DONE	0	R/W0	源匹配完成。 0: 没有中断未决 1: 中断未决
2	FIFOP	0	R/W0	RXFIFO中的字节数超过设置的阀值。当收到一个完整的帧也会激发。 0: 没有中断未决 1: 中断未决
1	SFD	0	R/W0	收到或发出SFD 0: 没有中断未决 1: 中断未决
0	ACT_UNUSED	0	R/W0	保留 0: 没有中断未决 1: 中断未决

RFIRQF1 (0x91)–RF 中断标志

位	名称	复位	R/W	描述
7:6	-	0	R0	保留。读作 0
5	CSP_WAIT	0	R/W0	CSP 的一条等待指令之后继续执行。 0: 无中断未决 1: 中断未决
4	CSP_STOP	0	R/W0	CSP 停止程序执行。 0: 无中断未决 1: 中断未决
3	CSP_MANINT	0	R/W0	来自 CSP 的手动中断产生。 0: 无中断未决 1: 中断未决
2	RFIDLE	0	R/W0	无线电状态机制进入空闲状态。 0: 无中断未决 1: 中断未决
1	TXDONE	0	R/W0	收到一个完整的帧。 0: 无中断未决 1: 中断未决
0	TXACKDONE	0	R/W0	完整发送了一个确认帧。 0: 无中断未决 1: 中断未决

RFERRF (0xBF) – RF 错误中断标志

位	名称	复位	R/W	描述
7	-	0	R0	保留。读作 0
6	STROBEERR	0	R/W0	命令选通在它无法被处理的时间发出。如果当已经禁用无线电时尝试禁用，且当不在活跃 RX 下尝试执行 SACK、SACKPEND 或 SNACK 命令，就触发。 0: 无中断未决 1: 中断未决
5	TXUNDERF	0	R/W0	TXFIFO 下溢 0: 无中断未决 1: 中断未决
4	TXOVERF	0	R/W0	TXFIFO 上溢 0: 无中断未决 1: 中断未决
3	RXUNDERF	0	R/W0	RXFIFO 下溢 0: 无中断未决 1: 中断未决
2	RXOVERF	0	R/W0	RXFIFO 上溢 0: 无中断未决 1: 中断未决
1	RXABO	0	R/W0	接收一个帧被停止 0: 无中断未决 1: 中断未决
0	NLOCK	0	R/W0	频率合成器在接收期间超时或锁丢失后，完成锁失败。 0: 无中断未决 1: 中断未决

RFIRQM0 (0x61A3) – RF 中断屏蔽

位	名称	复位	R/W	描述
7	RXMASKZERO	0	R/W	RXENABLE 寄存器从一个非零状态到全零状态。 0: 中断禁用 1: 中断使能
6	RXPKTDONE	0	R/W	收到一个完整的帧。 0: 中断禁用 1: 中断使能
5	FRAME_ACCEPTED	0	R/W	帧经过了帧过滤。 0: 中断禁用 1: 中断使能
4	SRC_MATCH_FOUND	0	R/W	源匹配被发现 0: 中断禁用 1: 中断使能
3	SRC_MATCH_DONE	0	R/W	源匹配完成 0: 无中断未决 1: 中断未决
2	FIFOP	0	R/W	RXFIFO 中的字节数超过设置的阀值。当收到一个完整的帧也激发。 0: 中断禁用 1: 中断使能
1	SFD	0	R/W	收到或发送 SFD 0: 中断禁用 1: 中断使能
0	ACT_UNUSED	0	R/W	保留 0: 中断禁用 1: 中断使能

RFIRQM1 (0x61A4) – RF 中断屏蔽

位	名称	复位	R/W	描述
7:6	-	0	R0	保留。读作 0
5	CSP_WAIT	0	R/W	CSP 的一条等待指令之后继续执行。 0: 中断禁用 1: 中断使能
4	CSP_STOP	0	R/W	CSP 停止程序执行。 0: 中断禁用 1: 中断使能
3	CSP_MANINT	0	R/W	来自 CSP 的手动中断产生。 0: 中断禁用 1: 中断使能
2	RFIDLE	0	R/W	无线电状态机制进入空闲状态。 0: 中断禁用 1: 中断使能
1	TXDONE	0	R/W	收到一个完整的帧。 0: 中断禁用 1: 中断使能
0	TXACKDONE	0	R/W	完整发送了一个确认帧。 0: 中断禁用 1: 中断使能

RFERRM (0x61A5) – RF 错误中断屏蔽

位	名称	复位	R/W	描述
7	-	0	R0	保留。读作 0
7:6	STROBEERR	0	R/W	命令选通在它无法被处理的时间发出。如果当已经禁用无线电时尝试禁用，且当不在活跃 RX 下尝试执行 SACK、SACKPEND 或 SNACK 命令，就触发。 0: 中断禁用 1: 中断使能
5	TXUNDERF	0	R/W	TXFIFO 下溢 0: 中断禁用 1: 中断使能
4	TXOVERF	0	R/W	TXFIFO 上溢 0: 中断禁用 1: 中断使能
3	RXUNDERF	0	R/W	RXFIFO 下溢 0: 中断禁用 1: 中断使能
2	RXOVERF	0	R/W	RXFIFO 上溢 0: 中断禁用 1: 中断使能
1	RXABO	0	R/W	接收一个帧被停止 0: 中断禁用 1: 中断使能
0	NLOCK	0	R/W	频率合成器在接收期间超时或锁丢失后，完成锁失败。 0: 中断禁用 1: 中断使能

19.2 FIFO 访问

可以通过 SFR 寄存器 RFD（0xD9）访问 TXFIFO 和 RXFIFO。当写入 RFD 寄存器时，数据被写入到 TXFIFO。当读取 RFD 寄存器时，数据从 RXFIFO 中读出。

XREG 寄存器 RXFIFOCNT 和 TXFIFOCNT 提供 FIFO 中的数据数量的信息。FIFO 的内容可以通过发出 SFLUSHRX 和 SFLUSHTX 清除。

RFD (0xD9) – RF 数据

位	名称	复位	R/W	描述
7:0	RFD[7:0]	0x00	R/W	数据写入寄存器，就是写入TXFIFO，当读取该寄存器的时候，就是从RXFIFO中读取数据。

19.3 DMA

可以并推荐使用直接存储访问（DMA）在存储器和无线电之间移动数据。DMA 控制器描述在第 8 章。关于如何配置和使用 DMA 传输的详细信息参见这一节。

有一个和无线电相关的 DMA 触发，支持 DMA 控制器。这就是 RADIO DMA 触发（DMA 触发 19）。RADIO DMA 触发由两个事件激活。第一个引起 RADIO DMA 触发的事件是第一个数据出现在 RXFIFO 中，即当 RXFIFO 从空状态变为非空状态。第二个引起 RADIO DMA 触发的事件是从 RXFIFO 中读取数据（通过 RFD），且 RXFIFO 中有更多的数据可用。

19.4 存储器映射

RF 内核包括 384 字节的物理 RAM，位于地址 0x6000 到 0x0617F。RF 内核的配置和状态寄存器位于地址

0x6180 到 0x61EF。

19.4.1 RX FIFO

RX FIFO 存储器区域位于地址 0x6000 到 0x607F，所以是 128 字节。尽管这一存储器区域用于 RX FIFO，但是它不以任何方式保护，因此它在 XREG 存储区域中仍然是可以访问的。一般地，只有指定的指令能用于操作 RX FIFO 的内容。RX FIFO 一次可以包括多个帧。

19.4.2 TX FIFO

TX FIFO 存储器区域位于地址 0x6080 到 0x60FF，所以是 128 字节。尽管这一存储器区域用于 TX FIFO，但是它不以任何方式保护，因此它在 XREG 存储区域中仍然是可以访问的。一般地，只有指定的指令能用于操作 TX FIFO 的内容。TX FIFO 一次只能包括一个帧。

19.4.3 帧过滤和源匹配存储器映射

帧过滤和源地址匹配功能使用 RF 内核 RAM 的一个 128 字节块来存储本地地址信息，和源地址匹配配置和结果；这位于区域 0x6100 到 0x617F。这一存取空间描述在表 19-1。没有填充整个字节/字的值位于字节/字的最低位部分。注意这些寄存器中的值复位之后是未知的。但是，这些值在供电模式期间保留。

表 19-1 帧过滤和源匹配存储器映射

地址	寄存器/变量	端模式	描述
保留			
0x6176–0x617F	暂时存储		存储空间用于暂时存储变量。
本地地址信息			
0x6174–0x6175	SHORT_ADDR	LE	目标地址过滤期间使用的短地址
0x6172–0x6173	PAN_ID	LE	目标地址过滤期间使用的 PAN ID
0x616A–0x71	EXT_ADD	LE	目标地址过滤期间使用的 IEEE 扩展地址
源地址匹配控制			
0x6169	SRCSHORTPENDEN2		24 位屏蔽的 8MSB，为每个 24 位短地址使能/禁用自动未决
0x6168	SRCSHORTPENDEN1		
0x6167	SRCSHORTPENDEN0		24 位屏蔽的 8LSB，为每个 24 位短地址使能/禁用自动未决
0x6166	SRCEXTPENDEN2		24 位屏蔽的 8MSB，为每个 12 位扩展地址使能/禁用自动未决。条目 n 映射到到 SRCEXTPENDEN[2n]。所有 SRCEXTPENDEN[2n + 1]位的值不重要。
0x6165	SRCEXTPENDEN1		
0x6164	SRCEXTPENDEN0		24 位屏蔽的 8LSB，为每个 12 位扩展地址使能/禁用自动未决。条目 n 映射到到 SRCEXTPENDEN[2n]。所有 SRCEXTPENDEN[2n + 1]位的值不重要。
源地址匹配结果 T			
0x6163	SRCRESINDEX		SRCRESMASK 的最低位 1 的位索引，或当没有源匹配时是 0x3F。匹配时，当短地址匹配位 5 是 0，当扩展地址匹配，是 1。匹配时，当确认的自动未决的条件符合（见 SRCMATCH.AUTOPEND 的描述），位 6 是 1。该位没有表明实际上是否发送了确认，不考虑 PENDING_OR 寄存器位和 SACK/SACKPEND/SNACK 选通命令。
0x6162	SRCRESMASK2		24 位屏蔽，表示源地址表中的每个单独的条目的源地址匹配

表 19-1 帧过滤和源匹配存储器映射（续表）

地址	寄存器/变量		端模式		描述
0x6161	SRCRESMASK1				短地址匹配。当条目上有一个匹配 panid_n + short_n，设置 SRCRESMASK 中的位 n。
0x6160	SRCRESMASK0				扩展地址匹配。当条目上有一个匹配 ext_n，设置 SRCRESMASK 中的位 2n 和 2n+1。
源地址表					
0x615E–0x615F	short_23	ext_11	LE	LE	两个单独的短地址条目（16 位 PAN ID 和 16 位短地址的组合）或 1 个扩展地址条目
0x615C–0x615D	panid_23		LE		
0x615A–0x615B	short_22		LE		
0x6158–0x6159	panid_22		LE		
...	...	...	...	...	...
0x610E–0x610F	short_03	ext_01	LE	LE	两个单独的短地址条目（16 位 PAN ID 和 16 位短地址的组合）或 1 个扩展地址条目
0x610C–0x610D	panid_03		LE		
0x610A–0x610B	short_02		LE		
0x6108–0x6109	panid_02		LE		
0x6106–0x6107	short_01	ext_00	LE	LE	两个单独的短地址条目（16 位 PAN ID 和 16 位短地址的组合）或 1 个扩展地址条目
0x6104–0x6105	panid_01		LE		
0x6102–0x6103	short_00		LE		
0x6100–0x6101	panid_00		LE		

19.5 频率和通道编程

频率载波可以通过编程位于 FREQCTRL.FREQ[6:0]的 7 位频率字设置。支持载波频率范围是 2394 MHz 到 2507 MHz。以 MHz 为单位的操作频率 f_c 由下式表示：$f_C = 2394 + FREQCTRL.FREQ[6:0])$ MHz，以 1 MHz 为步长，是可编程的。

IEEE802.15.4-2006 指定 16 个通道，它们位于 2.4GHz 频段之内。步长为 5 MHz，编号为 11～26。通道 k 的 RF 频率由[1]指定。

$$f_c = 2405 + 5(k - 11) \text{ [MHz]} \quad k \in [11, 26] \qquad (19\text{-}1)$$

对于操作在通道 k，FREQCTRL.FREQ 寄存器因此设置为 FREQCTRL.FREQ = 11 + 5 (k – 11)。

19.6 IEEE 802.15.4-2006 调制格式

本节目的是介绍 IEEE802.15.4 定义的 2.4GHz 直接序列扩频频谱（DSSS）的 RF 调制格式。完整描述请参阅标准文件[1] 。

调制和扩频功能如图 19-1 的方块图所示。每个字节分为两个符号，每个符号 4 位。低位符号首先传输。对于多字节域，低字节首先传输，除了与安全相关的域是高字节首先传输。

每个符号映射到 16 个伪随机序列之一，每个序列有 32 个芯片。符号到芯片的映射展示在表 19-2 中。然后芯片序列以 2 MChips/s 的速度传输，每个符号的低位芯片（C_0）首先传输。传输的比特流和芯片序列可以在 GPIO 引脚上观察到，如何配置 GPIO 完成这一功能详见第 7 章。

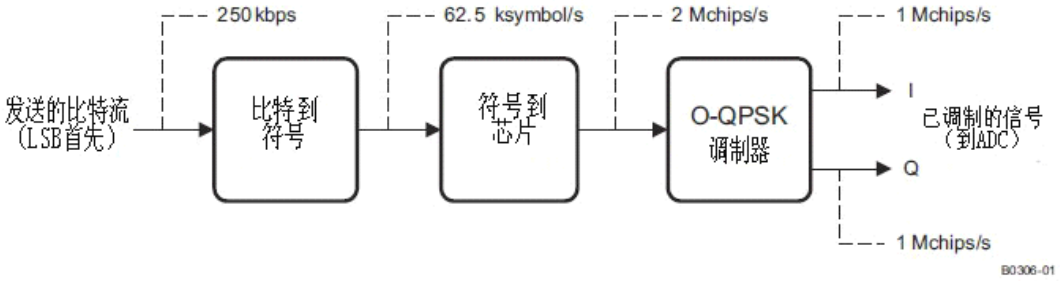


图 19-1　调制

表 19-2　IEEE 802.15.4-2006 符号到芯片的映射

符号	芯片序列(C0, C1, C2, … , C31)
0	1 1 0 1 1 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0
1	1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0
2	0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0
3	0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1
4	0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1
5	0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0
6	1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1
7	1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1
8	1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1
9	1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1
10	0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1
11	0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0
12	0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0
13	0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1
14	1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0
15	1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0

调制格式是偏移-正交相移键控（O-QPSK）和半正弦芯片成形。这相当于 MSK 调制。每个芯片形成半正弦波，轮流在一个半芯片周期偏移的 I 和 Q 通道传输。图 19-2 说明了零符号芯片序列的传输。

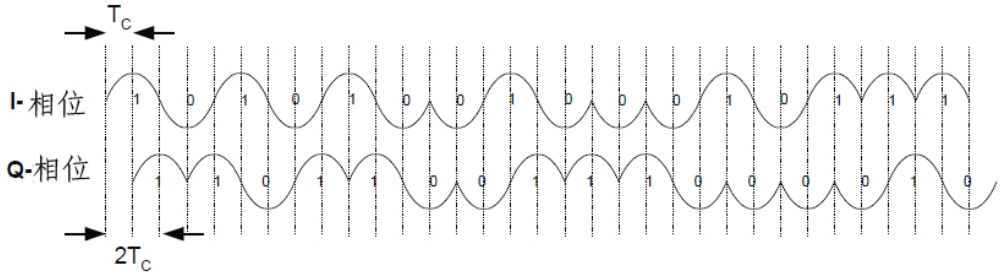


图 19-2　I / Q 当传送一个 0 符号芯片序列时的相位, TC = 0.5 μs

19.7 IEEE 802.15.4-2006 帧格式

本节给出了 IEEE 802.15.4 帧格式[1]的简短介绍。无线电有内置的支持可以处理帧的一部分。

如 19-3 展示了 IEEE 802.15.4 帧格式的示意图。类似的描述具体帧格式（数据帧、信标帧、确认帧和 MAC 命令帧）的图在标准的文件[1]中。

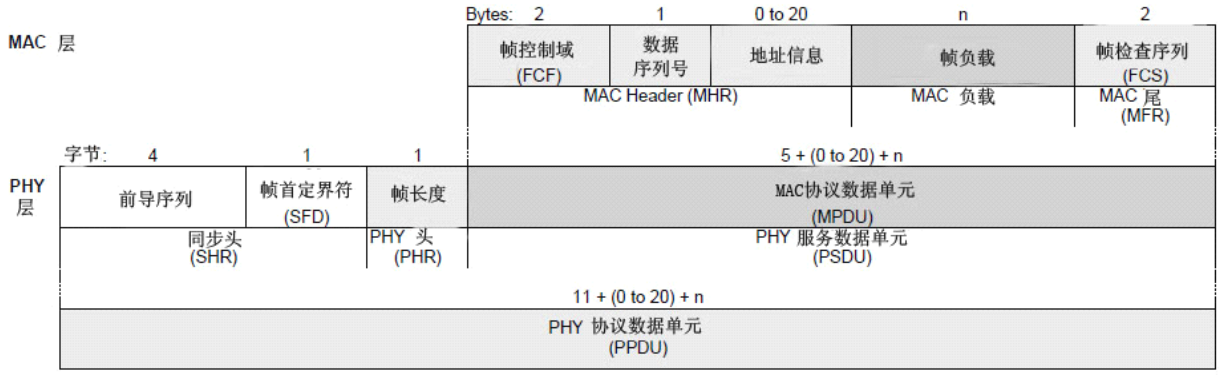


图 19-3 IEEE 802.15.4 帧格式 [1]的示意图展示

19.7.1 PHY 层

同步头

同步头（SHR）包括帧引导序列，接下来是帧开始界定符（SFD）。在 IEEE 802.15.4 规范[1]中，帧引导序列定义为四个字节的 0x00。SFD 是一个字节，设置为 0xA7。

PHY 层

PHY 头只包括帧长度域。帧长度域定义了 MPDU 中的字节数。注意长度域的值不包括长度域本身。但是它包括帧检查序列（FCS），即使这是由硬件自动插入的。

帧长度域是 7 位长，最大值是 127。长度域的最高位保留，总是设置为 0。

PHY 服务数据单元

PHY 服务数据单元包括 MAC 协议数据单元（MPDU）。产生/解释 MPDU 是 MAC 层的责任，无线电有内置的支持可以处理一些 MPDU 子域。

19.7.2 MAC 层

长度域后面的 FCF、数据序列号码和地址信息如图 19-3 所示。连同 MAC 数据负载和帧校验序列，形成了 MPDU。FCF 的格式见图 19-4。详细信息见 IEEE 802.15.4 规范[1]。

位：0-2	3	4	5	6	7-9	10-11	12-13	14-15
帧类型	使能安全	帧待定	确认请求	PAN 内部	保留	目的地址模式	保留	源地址模式

图 19-4 帧控制域的格式（FCF）

帧校验序列

如图 19-3 所示，最后一个 MAC 负载字节后面是一个 2 字节的帧校验序列（FCF）。FCF 是通过 MPDU 计算出来的，即长度域不是 FCS 的一部分。

[1]定义的 FCS 的表达式是：

$G(s) = x^{16} + x1^2 + x^5 + 1$

无线电支持自动计算/验证 FCS。详见 19.8.10 节。

19.8 发送模式

本节描述了如何控制发送器、组装帧的处理、以及如何使用 TX FIFO。

19.8.1 TX 控制

无线电有许多内置的功能，用于帧处理和报告状态。注意无线电提供的功能使得很容易地精确控制输出帧的时序。这在 IEEE 802.15.4/ZigBee®系统中是非常重要的，因为这类系统有严格的时序要求。

帧传输通过以下操作开始：

- STXON 命令选通

-没有更新 SAMPLED_CCA 信号。

- STXONCCA 命令选通，只要 CCA 信号为高。

-中止正在进行的发送/接收，强制一个 TX 校准，然后再传输。

-更新了 SAMPLED_CCA 信号。

空闲通道评估详细描述在 19.8.12 节。

帧传输通过以下命令操作中止：

- SRXON 命令选通

-中止正在进行的传输，强制一个 RX 校准

- SRFOFF 命令选通

-中止正在进行的发送/接收，强制 FSM 到 IDLE 状态。

- STXON 命令选通

-中止正在进行的传输，强制一个 RX 校准

STXON 发送之后要使能接收器，必须设置 FRMCTRL1.SET_RXENMASK_ON_TX 位。当执行 STXON 这设置 RXENABLE 的位 6。当 STXONCCA 发送，接收器在传输之前开启，然后返回（除非寄存器已经在此期间被清除）。

19.8.2 TX 状态时序

STXON 或 STXONCCA 命令选通 192us 之后开始传输帧引导序列。这在[1]中被叫做 *TX 轮转时序*。返回到接收模式也有同样的延迟。

当返回到空闲或接收模式，当调制器把信号送往 DAC 时有 2us 的延迟。这一输送在已经发送一个完整的 MPDU（由长度字节定义）或如果发生 TX 溢出之后自动发生。这会影响：

- SFD 信号，延长了 2us。
- 无线电 FSM 转换到 IDLE 状态，延迟了 2us。

19.8.3 TX FIFO 访问

TX FIFO 可以保存 128 字节，一次只能有一个帧。帧可以在执行 TX 命令选通之前或之后缓冲，只要不产生 TX 下溢（见 19.8.5 节所列的错误情况）。

图 19-5 解释了必须写到 TX FIFO 的字节（以蓝色表示）。另外的字节被忽略，除非发生 TX 溢出（见 19.8.5 节所列的错误情况）。

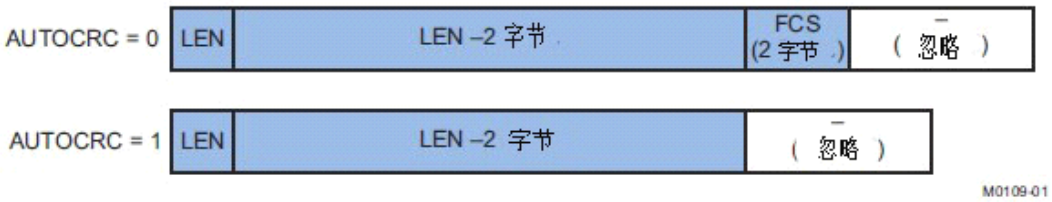


图 19-5　写入 TX FIFO 的帧数据

有两种方式写 TX FIFO。

- 写到 RFD 寄存器。
- 帧缓冲总是开始于 TX FIFO 存储器的起始地址。通过使能 FRMCTRL1.IGNORE_TX_UNDERF 位，可以直接写到无线电存储器的 RAM 区域，它保存 TX FIFO。注意建议使用 RFD 写数据到 TXFIFO。

TX FIFO 中的字节数存储在 TXFIFOCNT 寄存器中。

TX FIFO 可以使用 SFLUSHTX 命令选通手动清空。如果 FIFO 在传输期间被清空就发生 TX 下溢。

19.8.4　重传

为了支持简单的帧重传，无线电不会删除 TX FIFO 的内容，因为他们正在传输。成功发送一个帧后，FIFO 的内容保持不变。要重传同一个帧，只需通过发出一个 STXON 或 STXONCCA 命令选通重新启动 TX。注意如果数据包已经被完全发送才可以重传一个数据包，即数据包不能中止，然后再重传。

如果传输一个不同的帧，就写新的帧到 TX FIFO。在这种情况下，TX FIFO 在实际写发生之前自动清除。

19.8.5　错误情况

有两种错误情况与 TX FIFO 相关：

- 当 TX FIFO 满了且尝试写另一个字节，发生上溢。
- 当 TX FIFO 为空，且无线电尝试取另一个字节传输，发生下溢。

TX 上溢通过设置 TX_OVERFLOW 中断标志位表示。当发生这个错误，写被中止，即导致溢出的数据字节丢失。错误情况必须使用 SFLUSHTX 选通命令清除。

TX 下溢通过设置 TX_UNDERFLOW 中断标志位表示。当发生这个错误，正在进行的传输被中止。错误情况必须使用 SFLUSHTX 选通命令清除。

TX_UNDERFLOW 异常可以通过设置 FRMCTRL1.IGNORE_TX_UNDERF 位禁用。在这种情况下，无线电继续传输 TX FIFO 存储器中的字节，直到传输了第一个字节（即长度字节）给定的字节数。

19.8.6 TX 溢出图

图 19-6 的溢出图总结了以上章节：

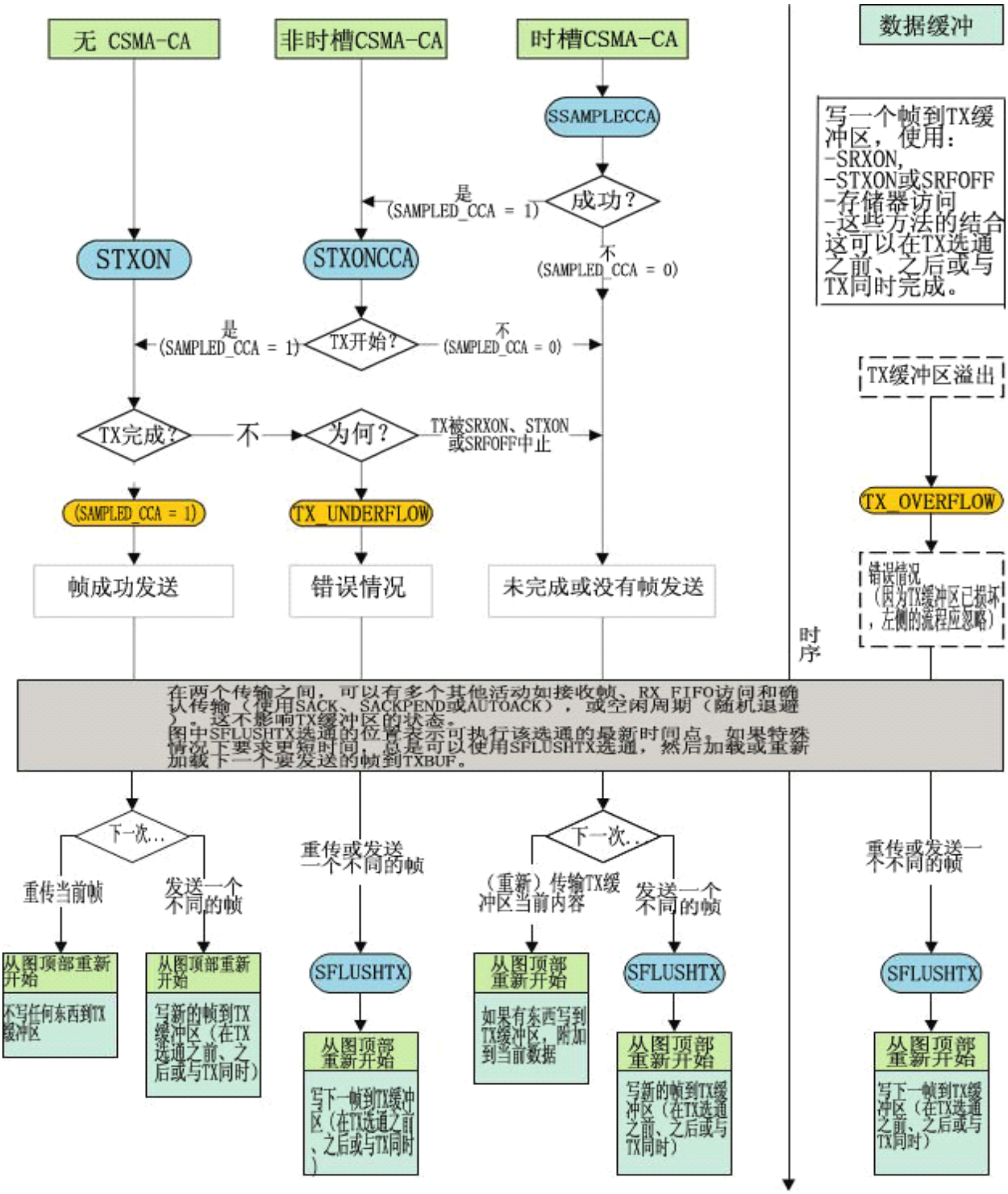


图 19-6 TX 溢出

19.8.7 帧处理

无线电为 TX 帧执行以下帧产生任务：

收到的帧

帧引导序列	SFD	LEN	MHR	MAC 负载	FCS
	（1）	（2）			（3）

（1） 产生并自动传输 PNY 层同步头，它包括帧引导序列和 SFD

（2） 传输帧长度域指定的字节数

（3） 计算并自动传输 FCS（可以禁用）

推荐用法是写长度域，然后写 MAC 头和 MAC 负载到 TX FIFO，让无线电处理其余部分。注意长度域必须包括两个 FCS 字节，即使无线电自动处理这些字节。

19.8.8 同步头

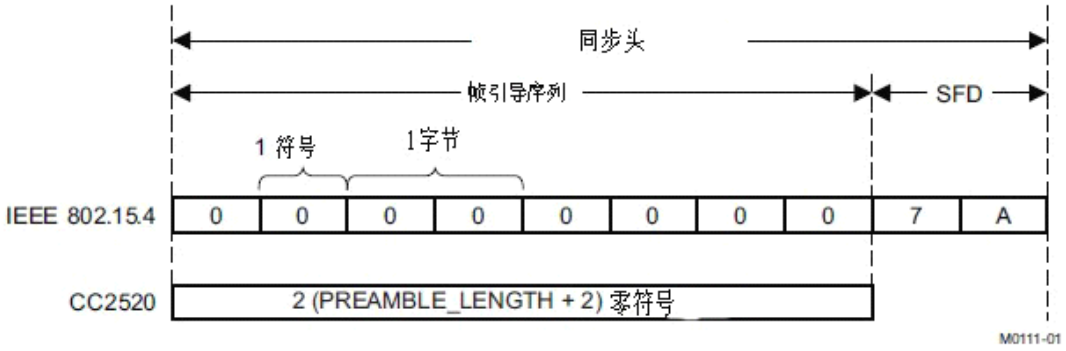


图 19-7 发送的同步头

无线电有可编程的帧引导序列长度。默认值遵守[1]，改变该值会使系统不兼容 IEEE 802.15.4。

帧引导序列长度由 MDMCTRL0.PREAMBLE_LENGTH 设置。图 19-7 显示了同步头是如何和 IEEE 802.15.4 规范相联系的。

当已经发送了所需的帧引导序列字节数，无线电自动发送 1 字节长的 SFD。SFD 是固定的，软件不能改变这个值。

19.8.9 帧长度域

当发送了 SFD，调制器开始从 TX FIFO 读数据。它期望找到帧长度域，然后是 MAC 头和 MAC 负载。帧长度域用于确定要发送多少个字节。

注意当 AUTOCRC = 1，最小帧长度是 3，当 AUTOCRC = 0，是 1。

19.8.10 帧校验序列

当设置了 FRMCTRL0.AUTOCRC 控制位，FCS 域自动产生并填充到发送帧的长度域定义的位置。FCS 不写到 TXFIFO 中，但是存储在一个单独的 16 位寄存器中。建议总是使能 AUTOCRC，除了可能用于调试目的。如果 FRMCTRL0.AUTOCRC = 0，那么调制器期望在 TX FIFO 中找到 FCS，所以软件必须产生 FCS，连同 MPDU 的其余部分写到 TX FIFO。

硬件实现 FCS 计算如图 19-8。详细信息见[1]。

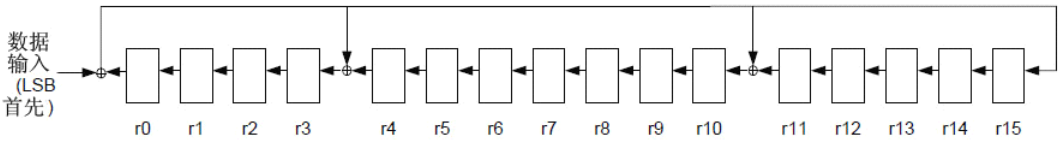


图 19-8　FCS 硬件实现

19.8.11　中断

当帧的 SFD 域已被发送就产生 SFD 中断。帧结束后，当成功发送一个完整的帧，产生 TX_FRM_DONE 中断。

注意在 GPIO 中有第二个 SFD 信号可用（通过无线电观测复用器）不能和 SFD 中断混淆。

19.8.12　空闲通道评估

空闲通道评估（CCA）状态信号表示通道是否可用于传输。CCA 功能用于实现 IEEE 802.15.4 规范[1]指定的 CSMA-CA 功能。为最后八个符号周期当接收器使能，CCA 信号有效。RSSI_VALID 状态信号可以用于验证这一点。

CCA 基于 RSSI 值和一个可编程的阀值。精确的行为可在 CCACTRL0 和 CCACTRL1 中配置。

CCA 信号有两个版本，一个在每个新的 RSSI 样本更新一次，一个只在 SSAMPLECCA/ISAMPLECCA 和 STXONCCA/ISTXONCCA 命令选通更新。它们在 FSMSTAT1 寄存器中都是可用的。

注意 CCA 信号在设置 RSSI_VALID 信号之后更新四个时钟周期（系统时钟）。

19.8.13 输出功率编程

RF 输出功率由 TXPOWER 寄存器的 7 位值控制。CC2530 数据手册显示了当中心频率设置为 2.440GHz，推荐设置的一般输出功率和电流消耗。注意推荐的设置只是所有可能的寄存器设置的一个很小的子集。

19.8.14　提示和技巧

- 注意在开始传输之前，TXFIFO 不需要有完整的帧。字节可以再传输期间增加到 TX FIFO。
- 通过设置 MDMTEST1.MODULATION_MODE = 1，可以发送不兼容 IEEE 802.15.4 的帧。

19.9　接收模式

本节描述了如何控制接收器、组装的 RX 帧处理以及如何使用 RX FIFO。

19.9.1　RX 控制

接收器分别根据 SRXON 和 SRFOFF 命令选通开启和关闭，或使用 RXENABLE 寄存器。命令选通提供一个硬开启/关闭机制，而 RXENABLE 操作提供一个软开启/关闭机制。

接收器通过以下操作开启：

- SRXON 选通：

-设置 RXENABLE[7]

-通过强制转换到 RX 校准，中止正在进行的发送/接收。

- STXON 选通，当 FRMCTRL1.SET_RXENMASK_ON_TX 使能：

-设置 RXENABLE[6]

-发送完毕后接收器使能

- 通过写 RXENMASKOR 设置 RXENABLE != 0x00：

-不中止正在进行的发送/接收。

接收器通过以下操作关闭：

- SRFOFF 选通：

-清除 RXENABLE[7:0]

-通过强制转换到 IDLE 模式，中止正在进行的发送/接收。

- 通过写 RXENMASKAND 设置 RXENABLE = 0x00

-不中止正在进行的发送/接收。一旦正在进行的发送/接收完成，无线电返回 IDLE 状态。

有若干方式操作 RXENABLE 寄存器：

- SRXMASKBITSET 和 SRXMASKBITCLR 选通（影响 RXENABLE[5]）
- SRXON、SRFOFF 和 STXON 选通，包括 FRMCTRL1.SET_RXMASK_ON_TX 设置

19.9.2 RX 状态时序

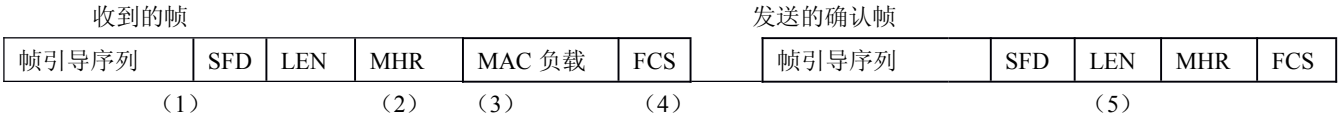
接收器通过 19.9.1 节所述的方式之一，在 RX 使能 192us 之后准备好。这在[1]中被叫做 *RX 轮转时序*。

当接收帧后返回到接收模式，有一个 192us 的默认间隔，SFD 检测禁用。这一间隔可以通过清除 FSMCTRL.RX2RX_TIME_OFF 禁用。

19.9.3 帧处理

无线电集合了 IEEE 802.15.4-2003 和-2006 中 RX 硬件方面要求的关键部分。这降低了 CPU 干预率，简化了处理帧接收的软件，且以最小的延迟给出结果。

接收一个帧期间，执行以下帧处理步骤：



（1） 检测和移除收到的 PHY 同步头（帧引导序列和 SFD），并接收帧长度域规定的字节数。

（2） 如[1]7.5.6.2 节，第三过滤级别规定的帧过滤。

（3） 匹配源地址和包括多达 24 个短地址的表，或 12 个扩展 IEEE 地址。源地址表存储在无线电 RAM 中。

（4） 自动 FCS 检查，并把该结果和其他状态值（RSSI、LQI 和源匹配结果）填入接收到的帧中。

（5） 具有正确时序的自动确认传输，且正确设置帧未决位，基于源地址匹配和 FCS 校验的的结果。

19.9.4 同步头和帧长度域

帧同步开始于检测一个帧开始界定符（SFD），然后是长度字节，它确定何时接收完成。SFD 信号可以在 GPIO 上输出，可以用于捕获收到帧的开始：

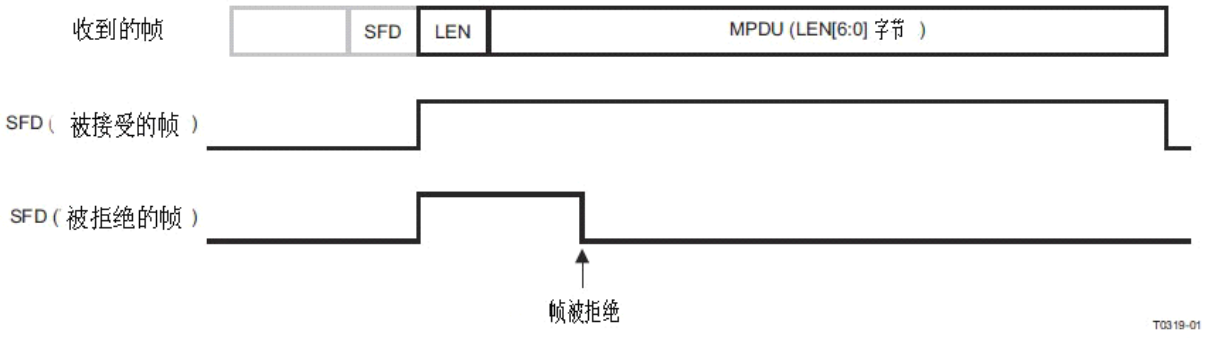


图 19-9　SFD 信号时序

帧引导序列和 SFD 不写到 RX FIFO。

无线电使用一个相关器来检测 SFD。MDMCTRL1.CORR_THR 中的相关器阀值确定收到的 SFD 必须如何密切匹配一个理想的 SFD。阀值的调整必须注意以下：

- 如果设置的太高，无线电会错过许多实际的 SFD，大大降低接收器的灵敏度。
- 如果设置的太低，无线电会检测到许多错误的 SFD。虽然这不会降低接收器的灵敏度，但是影响是类似的，因为错误的帧可能会重叠实际帧的 SFD。它还会增加接收具有正确 FCS 的错误帧的风险。

除了 SFD 检测，在 SFD 检测之前还可以请求若干有效地真引导序列符号（也在相关器阀值之上）。可用选项和推荐的设置见 MDMCTRL0 和 MDMCTRL1 的寄存器描述。

19.9.5　帧过滤

按照[1]7.5.6.2 节的第三过滤级别规定，帧过滤功能拒绝目标不明确的帧，它对以下情况提供过滤：

- 八种不同帧类型（见 FRMFILT1 寄存器）
- 帧控制域中的保留位（FCF）

该功能由以下控制：

- FRMFILT0 和 FRMFILT1 寄存器
- RAM 中的 LOCAL_PAN_ID、LOCAL_SHORT_ADDR 和 LOCAL_EXT_ADDR 值

过滤算法

FRMFILT0.FRM_FILTER_EN 位控制是否应用帧过滤。当禁用，无线电接受所有收到的帧。当使能（这是默认设置），无线电只接受符合以下全部要求的帧：

- 长度域必须等于或大于最小帧长度，它从 FCF 的源和目标地址模式，以及 PAN ID 压缩子域获得。
- 保留的 FCF 位[9:7] ANDed 以及 FRMFILT0.FCF_RESERVED_BITMASK 必须等于 000b。
- FCF 的帧版本子域的值不能高于 FRMFILT0.MAX_FRAME_VERSION。
- 源和目标地址模式不能是保留值（1）。
- 目标地址：

-如果一个目标 PAN ID 包含在帧中，它必须匹配 LOCAL_PANID 或广播 PAN 标识符（0xFFFF）。
-如果一个短地址包含在帧中，必须匹配 LOCAL_SHORT_ADDR 或广播地址（0xFFFF）。
-如果一个扩展地址包含在帧中，必须匹配 LOCAL_EXT_ADDR。

- 帧类型：

-只接受信标帧（0），当：

- FRMFILT1.ACCEPT_FT0_BEACON = 1
- 长度字节>= 9
- 目标地址模式是 0（没有目标地址）。
- 源地址模式是 2 或 3（即包含一个源地址）。
- 源 PAN ID 匹配 LOCAL_PANID，或 LOCAL_PANID 等于 0xFFFF。

-只接受数据（1）帧，当：

- FRMFILT1.ACCEPT_FT1_DATA = 1
- 长度字节>= 9
- 目标地址和/或源地址包含在帧中，如果没有目标地址包含在帧中，必须设置 FRMFILT0.PAN_COORDINATOR 位，且源 PAN ID 必须等于 LOCAL_PANID。

-只接受确认帧（2），当：

- FRMFILT1.ACCEPT_FT2_ACK = 1
- 长度字节=5

-只接受 MAC 命令帧（3），当：

- FRMFILT1.ACCEPT_FT3_MAC_CMD = 1
- 长度字节>=9
- 目标地址和/或源地址包含在帧中，如果没有目标地址包含在帧中，必须设置 FRMFILT0.PAN_COORDINATOR 位，且源 PAN ID 必须等于要接受的帧的 LOCAL_PANID。

-只接受保留的帧类型（4，5，6 和 7），当：

- FRMFILT1.ACCEPT_FT4TO7_RESERVED = 1（默认是 0）
- 长度字节>=9

过滤开始之前执行以下操作，不会影响存储在 RX FIFO 中的帧数据：

- 长度字节的位 7 被屏蔽（没有关系）。
- 如果 FRMFILT1.MODIFY_FT_FILTER 不是零，FCF 的帧类型子域的 MSB 或者颠倒或者强制变为 0 或 1。

如果拒绝了一个帧，无线电仅仅开始在拒绝帧已经被完全接收后（由长度域定义）寻找一个新的帧，以避免在帧内检测到错误的 SFD。注意如果在帧被拒绝之前发生 RX 溢出，被拒绝的帧可以产生 RX 溢出。

中断

当帧过滤使能，且过滤算法接受一个已接收的帧，就产生一个 RX_FRM_ACCEPTED 中断。如果帧过滤禁用或在知道过滤结果之前产生 RX_OVERFLOW 或 RX_FRM_ABORTED，不产生该中断。

图 19-10 说明了三种不同的脚本（不包括溢出和中止错误情况）。

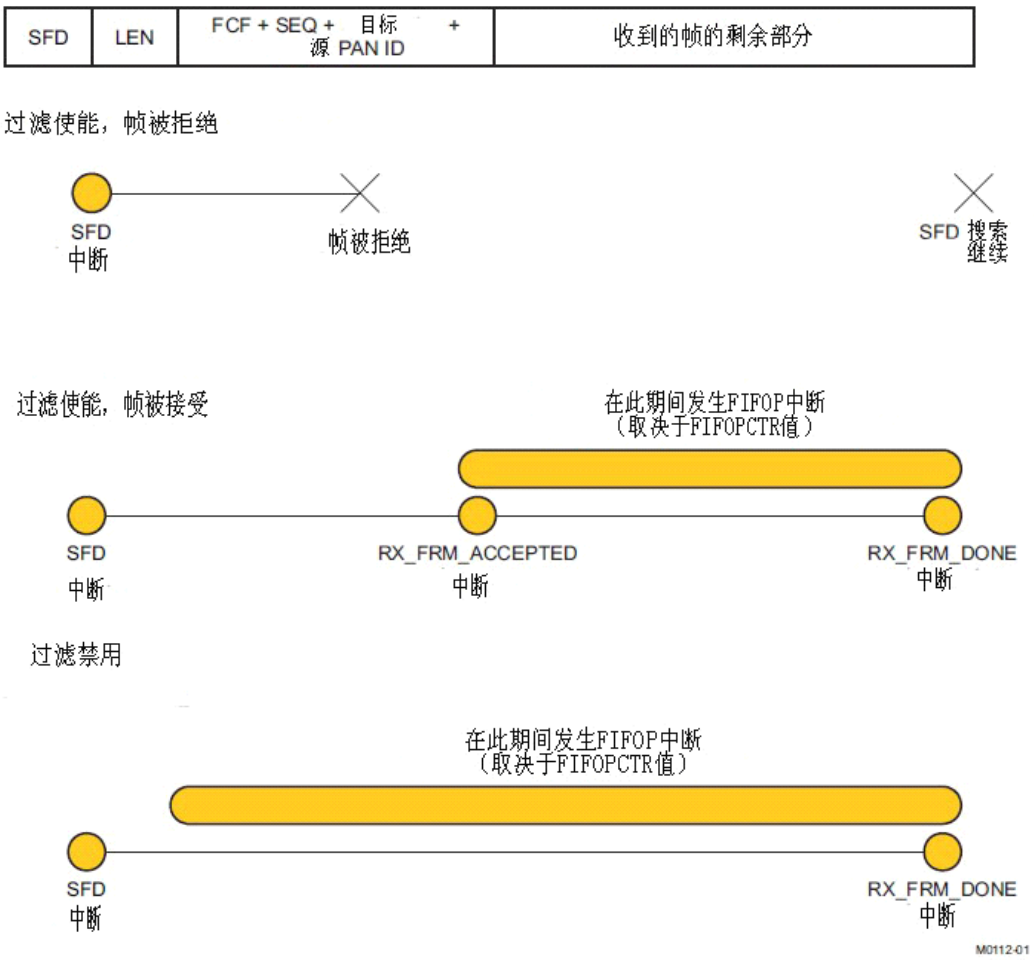


图 19-10　过滤脚本（接收期间产生异常）

当完全接收一个帧开始界定符，FSMSTAT1.SFD 寄存器位变为高，且保持高电平直到接收 MPDU 最后一个字节，或收到的帧没有通过地址识别，且已被拒绝。

提示和技巧

以下寄存器设置必须正确配置：

- 如果设备是一个 PAN 协调器，必须设置 FRMFILT0.PAN_COORDINATOR，且如果不是必须清除。
- FRMFILT0.MAX_FRAME_VERSION 必须对应 IEEE 802.15.4 标准支持的版本。
- 本地地址信息必须加载到 RAM。

要在能量检测扫描期间完全避免接收帧，设置 FRMCTRL0.RX_MODE = 11b，然后（重新）启动 RX。这禁用了符号搜索从而防止 SFD 检测。

要恢复正常的 RX 模式，设置 FRMCTRL0.RX_MODE = 00b 并（重新）启动 RX。

在一个繁忙的 IEEE 802.15.4 环境的操作中，无线电接收许多目标不明确的确认帧。要有效阻止接收这些帧，使用 FRMFILT1.ACCEPT_FT2_ACK 位来控制何时应该接收确认帧：

- 成功启动一个带有确认的发送请求之后，设置 FRMFILT1.ACCEPT_FT2_ACK，且接收确认帧或达到超时之后，再清除该位。

- 否则保持该位清除。

当改变 FRMFILT0/1 寄存器的值和存储在 RAM 中的本地地址信息时，不需要关闭接收器。但是，如果改变发生在接收 SFD 字节和源 PAN ID 之间（即在接收 *SFD* 和 *RX_FRM_ACCEPTED* 之间），修改的值必须被视作不影响特定帧（无线电使用旧值或是新值）。

注意通过设置 MDMTEST1.MODULATION_MODE = 1，可以使无线电忽略所有 IEEE 802.15.4 的输入帧。

19.9.6 源地址匹配

无线电支持收到的帧的源地址和存储在片上存储器中的一个表匹配。该表长 96 字节，因此可以包含多达:

- 24 个短地址（每个 2+2 字节）
- 12 个 IEEE 扩展地址（每个 8 字节）

仅当帧过滤也使能且收到的帧已被接受时才执行源地址匹配。该功能由以下控制：

- SRCMATCH、SRCSHORTEN0、SRCSHORTEN1、SRCSHORTEN2、SRCEXTEN0、SRCEXTEN1 和 SRCEXTEN2 寄存器
- RAM 中的源地址表

应用

帧未决位正确设置的自动确认传输：当使用间接帧传输，设备发送数据请求来轮询存储在协调器中的帧。要表示实际上是否为设备存储了一个帧，协调器必须在返回的确认帧中设置或清除帧未决位。但是在大多数 8 位或 16 位 MCU 上，没有足够的时间来确定这一点，因此协调器都设置未决位，不管设备是否有未决帧（如 IEEE 802.15.4 [1]所要求）。这在功耗方面是很浪费的，因为轮询设备必须保持它的接收器使能相当长的一段时间，即使没有帧使用。通过加载间接帧序列的目标地址到源地址表中，且使能 AUTOPEND 功能，无线电就自动设置输出确认帧的未决位。这样，操作不再是时序关键的，因为微控制器所做的努力是增加或移除间接帧序列中的帧，并更新相应的源地址表。

安全材料查询：为了减少处理安全帧所需的时间，可以配置源地址表，这样条目匹配 CPU 的安全密钥表。表条目的第二级别的屏蔽允许这一应用与自动设置确认帧的未决位相结合。

其他应用：之前的两个应用是源地址匹配功能的主要目标。但是，对于只依靠基本 IEEE 802.15.4 帧格式的专门的协议，有一些其他有用的应用。例如，可以创建防火墙功能，即只承认一组规定的节点。

源地址表

源地址表开始于 RAM 的地址 0x6100。该空间在短地址和扩展地址之间共享，SRCSHORTEN0/1/2 和 SRCEXTEN0/1/2 寄存器用于控制使能哪个条目。表中的所有值都以小端（如接收帧）表示。

- *短地址条目*开始于 16 位 PAN ID，后面是 16 位短地址。这些条目存储在地址 0x6100 + (4 × n)，其中 n 是 0 到 23 之间的数字。
- *扩展地址条目*只包括 64 位 IEEE 扩展地址。这些条目存储在地址 0x6100 + (8 × n)，其中 n 是 0 到 11 之间的数字。

地址使能寄存器

软件负责分配表条目，以确保活跃的短地址和扩展地址条目不会重叠。短地址和扩展地址有单独的使能位：

- 短地址条目在 SRCSHORTEN0、SRCSHORTEN1 和 SRCSHORTEN2 寄存器中使能。寄存器位 n 对应短地址条目 n。

- 扩展地址条目在 SRCEXTEN0、SRCEXTEN1 和 SRCEXTEN2 寄存器中使能。在这种情况下，寄存器位 2n 对应扩展地址条目 n。当创建一个（短和扩展使能位的）结合位向量时，使用这一映射可以很方便地找到未使用的条目。而且，读取时，寄存器位 2n + 1 的值总是和寄存器位 2n 的值相同，因为扩展地址占据了和两个短地址条目同样的存储器。

匹配算法

SRCMATCH.SRC_MATCH_EN 位控制源地址匹配是否使能。当使能（这是默认设置），一个帧需要经过帧过滤算法，根据给出的源地址的类型，无线电应用图 19-13 所列的算法之一。

结果以两种不同的形式报告：

- 一个 24 位的向量，叫做 SRCRESMASK，为每个具有一个匹配的使能的短条目都包括一个 1，或为每个具有一个匹配的使能的扩展条目包括两个 1（位映射和读访问的地址使能寄存器相同）。
- 一个 7 位值，叫做 SRCRESINDEX：

-当收到的帧中没有给出源地址，或在收到的源地址上没有匹配：

- 位 6:0：0x3F

-如果在收到的源地址上有一个匹配：

- 位 4:0：具有一个匹配的第一个条目的索引（即具有最小索引号的条目），0-23 用于短地址，或 0-11 用于扩展地址。
- 位 5：如果匹配是在短地址上，是 0，如果匹配是在扩展地址上，是 1。
- 位 6：AUTOPEND 功能的结果

短的源地址（模式2）
收到的源PAN ID叫做srcPanid。收到的短地址叫srcShort。

```
SRCRESMASK = 0x000000;
SRCRESINDEX = 0x3F;
for (n = 0; n < 24; n++) {
   bitVector = 0x000001 << n;
  if (SRCSHORTEN & bitVector) {
      if ((panid[n] == srcPanid) &&
          (short[n] == srcShort)) {
          SRCRESMASK |= bitVector;
          if (SRCRESINDEX == 0x3F) {
              SRCRESINDEX = n;
          }
      }
   }
}
```

扩展的源地址（模式3）
收到的扩展地址是srcExt。

```
SRCRESMASK = 0x000000;
SRCRESINDEX = 0x3F;
for (n = 0; n < 12; n++) {
   bitVector = 0x000003 << (2*n);
   if (SRCEXTEN & bitVector) {
      if (ext[n] == srxExt) {
         SRCRESMASK |= bitVector;
         if (SRCRESINDEX == 0x3F) {
            SRCRESINDEX = n | 0x20;
         }
      }
   }
}
```

图 19-11　短地址和扩展地址的匹配算法

只要结果可用，SRCRESMASK 和 SRCRESINDEX 就写到 RF 内核寄存器。

如果已经设置了 FRMCTRL0.AUTOCRC 和 FRMCTRL0.APPEND_DATA_MODE 位，SRCRESINDEX 也必须增加到接收的帧。然后该值替代 16 位状态字的 7 位 LQI 值。

中断

当源地址匹配使能，且匹配算法完成，不管结果如何，都设置 *SRC_MATCH_DONE* 中断标志。如果找到一个匹配，在 *SRC_MATCH_DONE* 之前还立即设置 SRC_MATCH_FOUND 标志。

图 19-12 说明了设置标志的时序：

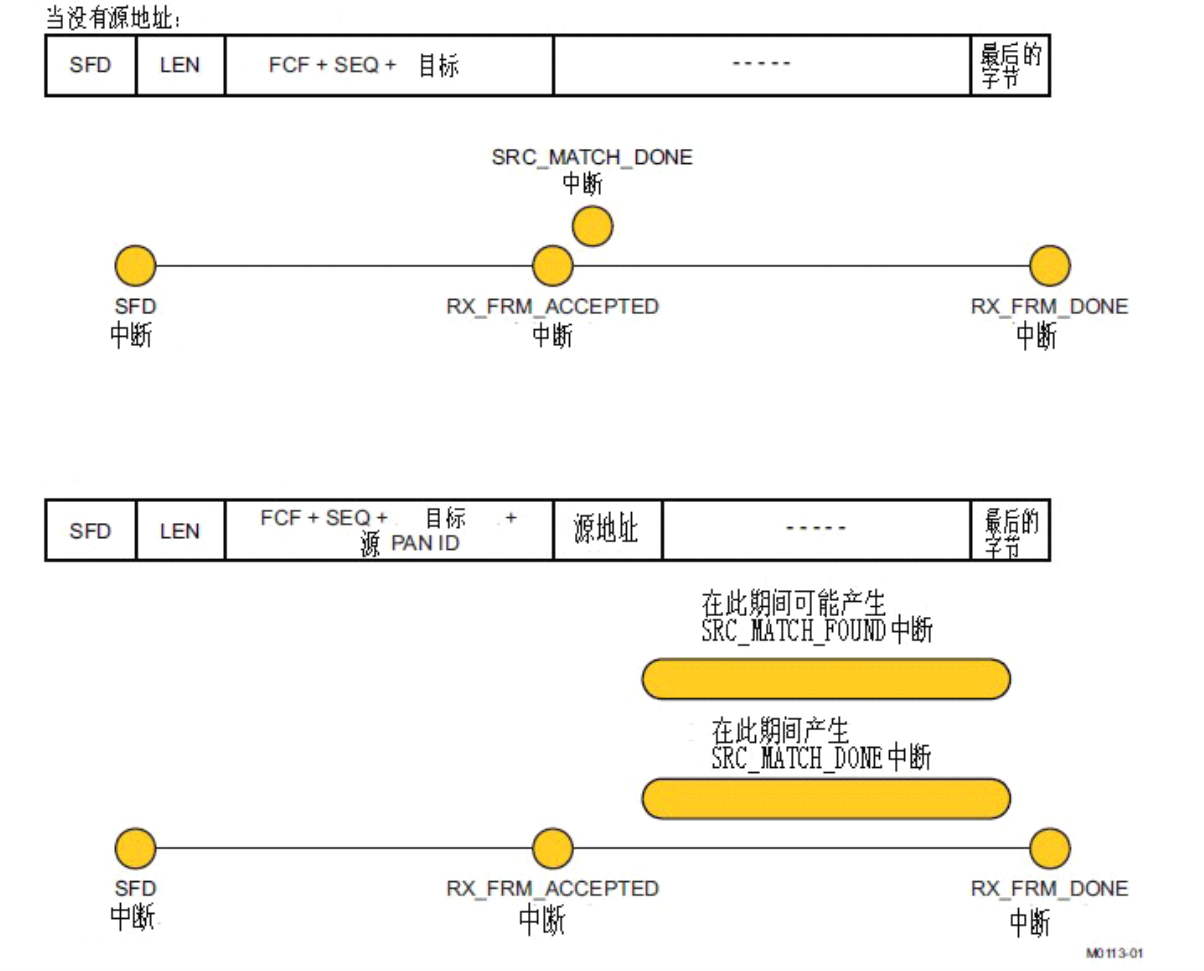


图 19-12　源地址匹配产生的中断

提示和技巧

- 源地址表可以在帧接收期间安全修改。如果在接收器活跃的时候，一个地址替代了另一个，相应的使能位必须在修改期间关闭。这将避免 RF 内核使用旧值和新值的结合，因为它只考虑在整个源匹配过程中使能的条目。

可以采取以下措施来避免下一个收到的帧重写源地址匹配的结果：

- 使用附加的 SRCRESINDEX 结果，而不是写入 RAM 的值（这是推荐的方法）。
- 在下一个收到的帧中发生 RX_FRM_ACCEPTED 之前，从 RAM 读结果。对于最短的帧类型，这发生在序列号之后，这样总的可用时间（安全系数最小的绝对最坏情况）变成：
  16 us (所需的时间) + 32 us (SFD) + 128 us (4 字节) = 176 us
- 为了增加可用时间，清除 FSMCTRL.RX2RX_TIME_OFF 位。这另外增加了 192 us，总共 368 us。这还降低了 RX 溢出的风险：

19.9.7 帧校验序列

在接收模式，如果 FRMCTRL0.AUTOCRC 使能，FCS 由硬件验证。用户一般只关心 FCS 的正确性，而不关心 FCS 序列本身。因此接收期间 FCS 序列本身不写入 RX FIFO。相反，当设置 FRMCTRL0.AUTOCRC，两个 FCS 字节被其他更有用的值替代。取代 FCS 序列的值可在寄存器 FRMCTRL0 中配置。

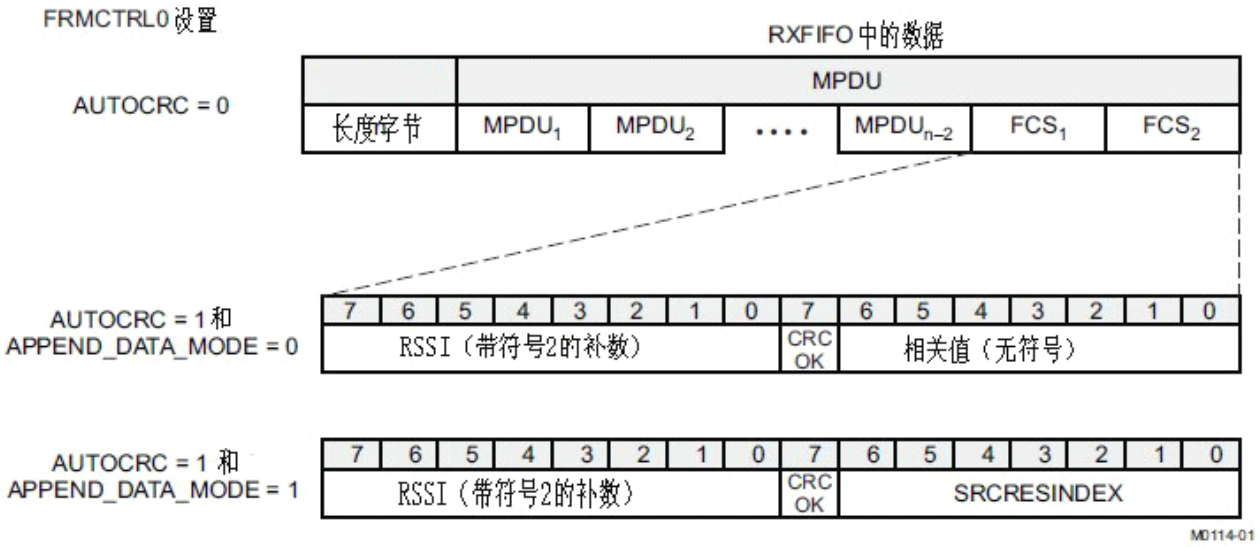


图 19-13 不同设置下 RX FIFO 的数据

域描述：

- *RSSI* 值在 SFD 后面第一组八个符号时测量。
- *CRC_OK* 位表示 DCS 是否正确（1），不正确（0）。如果不正确，软件负责丢弃该帧。
- *相关值* 是 SFD 后面第一组八个符号的平均相关值。
- *SRCRESINDEX* 和完成源地址匹配之后写入 RAM 的值相同。

计算由 IEEE 802.15.4 使用的 LQI 值见 19.10.4 节。

19.9.8 确认传输

无线电包括硬件支持成功接收帧后，进行确认传输（即收到帧的 FCS 必须是正确的）。图 19-14 显示了确认帧的格式。

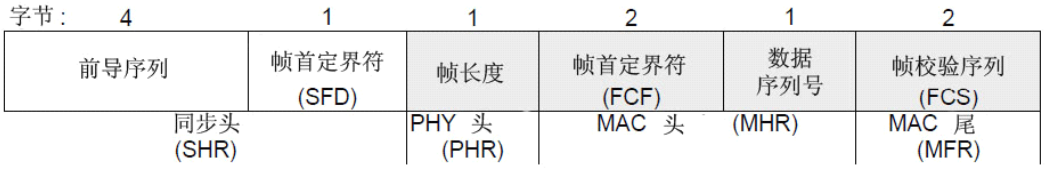


图 19-14 确认帧格式

产生的确认帧中有三个可变域：

- 未决位，可以由命令选通和 AUTOPEND 功能控制
- 数据序列号码（DSN），从最后收到的帧中自动采取
- FCS，以隐含方式给出

有三个不同的源可以设置一个 ACK 帧的未决位（即 SACKPEND 选通、PENDING_OR 寄存器位和 AUTOPEND 功能）。如果设置了一个这些源或多个这些源，就设置未决位。

传输时序

确认帧只能在接收帧后立即传输。传输时序由 FSMCTRL.SLOTTED_ACK 位控制。

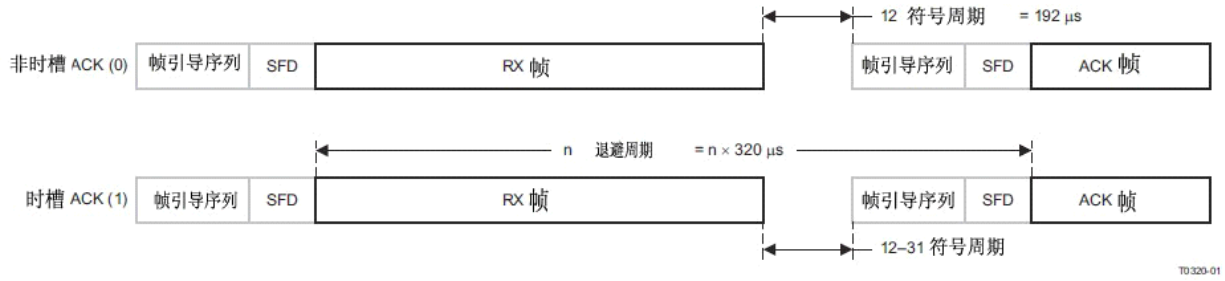


图 19-15 确认时序

802.15.4 要求非启用信标的 PAN 使用非时槽模式，启用信标的 PAN 使用时槽模式。

手动控制

SACK、SACKPEND 和 SNACK 命令选通只能在接收帧期间发出。如果选通在其他任何时间发出，它们没有影响，但是会产生一个 STROBE_ERROR 中断。

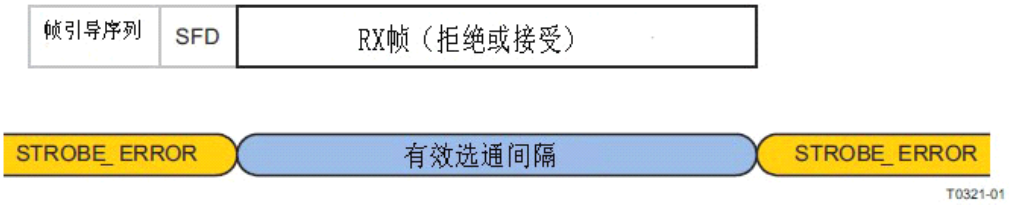


图 19-16 命令选通时序

在接收期间可以发出若干次命令选通，但是，只有最后一个选通有效：

- 无选通/SNACK/不正确的 FCS：没有确认传输
- SACK：帧未决位被清除的确认传输
- SACKPEND：帧未决位被设置的确认传输

自动控制（AUTOACK）

当 FRMFILT0.FRM_FILTER_EN 和 FRMCTRL0.AUTOACK 都使能，无线电自动确定是否传输确认帧：

- RX 帧经过帧过滤后必须被接收（由 RX_FRM_ACCEPTED 异常表示）。
- RX 帧中的确认请求位必须设置。
- RX 帧不能是一个信标或是一个确认帧。
- RX 帧的 FCS 必须正确。

自动确认可以由 SACK、SACKPEND 和 SNACK 命令选通重写。例如，如果微控制器低于存储器资源且不能存储一个收到的帧，SNACK 选通可以在接收期间发出，避免确认被丢弃的帧。

默认情况下，AUTOACK 功能不糊设置确认帧的帧未决位。除了命令选通自动重写，有两个选项：

- 自动控制，使用 AUTOPEND 功能
- 手动控制，使用 FRMCTRL1.PENDING_OR 位

自动设置帧未决域（AUTOPEND）

当设置了 SRCMATCH.AUTOPEND 位，源地址匹配的结果确定帧未决域的值。接收一个帧时，设置（可能的）返回的确认帧的未决域，只要下列条件满足：

- 设置了 FRMFILT0.FRAME_FILTER_EN。
- 设置了 SRCMATCH.SRC_MATCH_EN。
- 设置了 SRCMATCH.AUTOPEND。
- 收到的帧匹配当前设置的 SRCMATCH.PEND_DATAREQ_ONLY。
- 收到的源地址匹配至少一个源地址表条目，在 SRCSHORTEN 和 SRCSHORTPENDEN，或 SRCEXTEN 和 SRCEXTPENDEN 都使能。

如果源匹配表满了，FRMCTRL1.PENDING_OR 位可以用于覆盖 AUTOPEND 功能，暂时确认所有帧未决位设置的帧。

19.10 RX FIFO 访问

RX FIFO 可以保存一个或多个收到的帧，只要总字节数是 128 或更少。有两种方式确定 RX FIFO 中的字节数：

- 读 RXFIFOCNT 寄存器
- 使用 FIFOP 和 FIFO 信号，结合 FIFOPCTRL.FIFOPTHR 设置

RX FIFO 通过 RFD 寄存器被访问。

RX FIFO 中的数据还可以通过访问无线电 RAM 直接访问。FIFO 指针可在 RXFIRST_PTR、RXLAST_PTR 和 RXP1_PTR 中读。如果不首先读整个帧，想要快速访问帧的某个字节可以应用这一方法。注意当使用这一直接访问，FIFO 指针不被更新。

ISFLUSHRX 命令选通会复位 RX FIFO，复位所有 FIFO 指针并清除所有计数器、状态信号和标记错误条件。

SFLUSHRX 命令选通复位 RX FIFO，移除所有收到的帧并清除所有计数器、状态信号和标记错误条件。

19.10.1 使用 FIFO 和 FIFOP

当读出一小部分收到的帧时 FIFO 和 FIFOP 信号有用，在收到帧时：

- 当一个或多个字节在 RX FIFO 中，FSMSTAT1.FIFO 变为高，但是当发生 RX 溢出变为低。
- FSMSTAT1.FIFOP 信号变为高，当：

-RX FIFO 的有效字节数超过 FIFOPCTRL 编程的 FIFOP 阈值。当帧过滤使能，帧头的字节不被视为有效，直到帧被接受。

-一个新的帧的最后一个字节被接收，即使没有超过 FIFOP 阈值。如果是这样，FIFOP 在下一个 RX FIFO 读访问回到低。

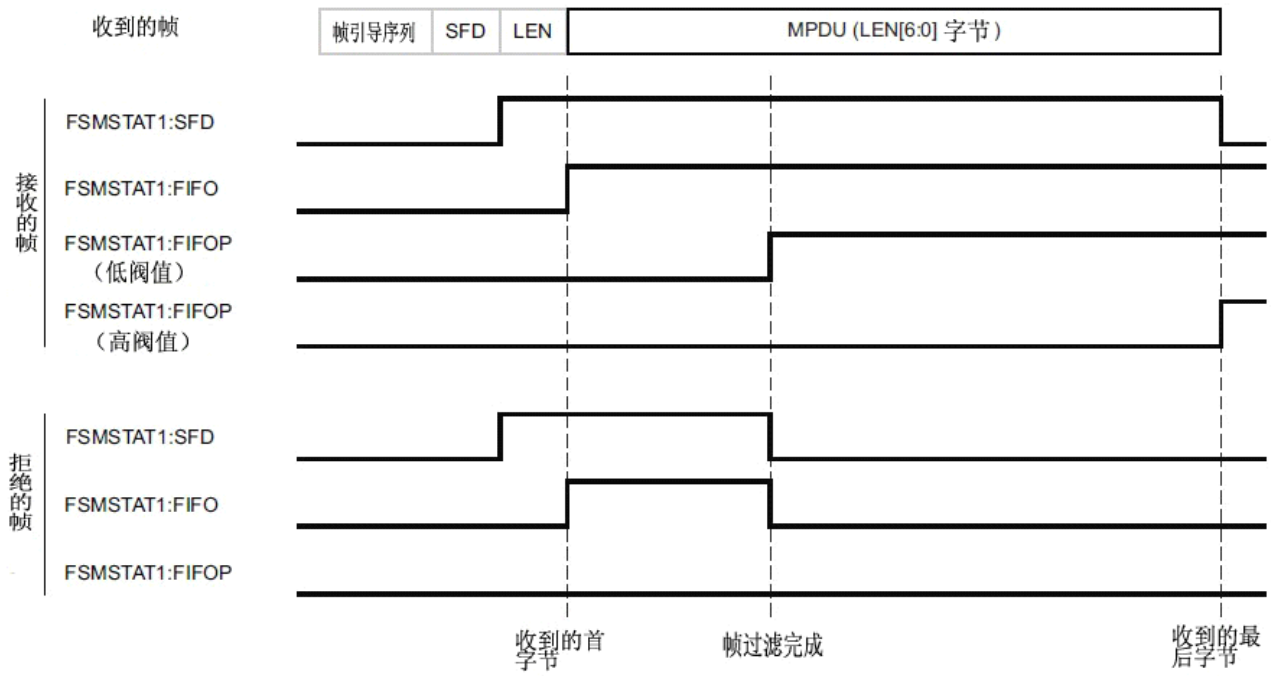


图 19-17　FIFO 和 FIFOP 信号的行为

当使用 FIFOP 作为微控制器的一个中断源，FIFOP 阀值必须由中断服务例程调整，以准备下一个中断。当为一个帧准备最后一个中断，阀值必须匹配剩余的字节数。

19.10.2　错误情况

有两种错误情况与 RX FIFO 相关：

- 上溢，在这种情况下当接收另一个字节 RX FIFO 为满
- 下溢，在这种情况下软件尝试从一个空的 RX FIFO 中读一个字节

RX 上溢通过设置 RFERRF.RXOVERF 标志，以及信号值 FSMSTAT1.FIFO = 0 和 FSMSTAT1.FIFOP = 1 表示。当发生错误，接收帧停止。当前存储在 RX FIFO 的帧可以在情况被清除之前使用 ISFLUSHRX 选通读出。注意如果在帧被拒绝之前发生情况，被拒绝的帧可以产生 RX 上溢。

RX 下溢通过设置 RFERRF.RXUNDERF 标志表示。RX 下溢是一种严重的错误情况，不能在免于错误的软件中发生，且 RXUNDERF 事件只能用于调试或一个看门狗功能。注意在接收一个新字节的同时发生读操作，不产生 RXUNDERF 错误。

19.10.3　RSSI

无线电有一个内置的接收信号强度指示器(RSSI)，计算一个 8 位有符号的数字值，可以从寄存器读出，或自动附加到收到的帧。RSSI 值总是通过 8 个符号周期内(128 μ s)取平均值得到的，与 IEEE 802.15.4[1]相符合。

RSSI 值是一个 2 的有符号补数，对数尺度是 1-dB 的步长。

在读 RSSI 值寄存器之前必须检查状态位 RSSI_VALID。RSSI_VALID 表示寄存器中的 RSSI 值事实上是否有效，这意味着接收器已经为最后八个符号周期使能。

为了以合理的精确度在 RF 引脚找到实际的带符号功率 P，必须增加一个偏移量到 RSSI 值。

P = RSSI − OFFSET [dBm]

例如，从 RSSI 寄存器读 RSSI 值-10 时偏移量是 73dB 意味着 RF 输入功率大约是-83dB。使用正确的偏移量值请参考数据手册[2]。

在第一次变为有效之后，可以配置无线电如何更新 RSSI 寄存器。如果 FRMCTRL0.ENERGY_SCAN = 0（默认），RSSI 寄存器包括最新的可用值，但是如果该位设置为 1，执行一个峰值搜索，RSSI 寄存器包括自能量扫描使能以来最大的值。

19.10.4 链路质量指示

如同 IEEE 802.15.4 标准[1]中的定义，链路质量指示(LQI)计量的就是所收到的数据包的强度和/或质量。IEEE 802.15.4 标准[1]要求的 LQI 值限制在范围 0 到 255，至少需要 8 个唯一的值。无线电不直接提供一个 LQI 值，但是报告一些测量结果，微控制器可以使用它们来计算一个 LQI 值。

MAC 软件可以使用 RSSI 值来计算 LQI 值。这一方法有若干缺点，即通道带宽内的窄带干扰会增加 RSSI，因此 LQI 值即真正的链路质量实际上降低了。因此，对于每个输入的帧，无线电提供了一个平均相关值，该值基于跟随在 SFD 后面的前 8 个符号。虽然无线电不做片码判定，但是这个无符号的 7 位数值可以看作是片码错误率的测量。

如 19.9.7 节所述，当设置 MDMCTRL0.AUTOCRC，前八个符号的平均相关值连同 RSSI 和 CRC OK/not OK，附加到每个收到的帧中。相关值~110 表示最高质量帧，而值~50 一般表示无线电检测到的最低质量帧。

软件必须将平均相关值转换为由[1]定义的，范围为 0～255 的数值，即按照下式计算：

LQI = (CORR − a) b

限制为 0~255，式中 a 和 b 是基于包差错率(PER)测量的经验值。

RSSI 和相关值结合起来，还可用于产生 LQI 值。

19.11 无线电控制状态机制

FSM 模块负责维护 TX FIFO 和 RX FIFO 指针，控制模拟动态信号，比如上电/掉电，控制 RF 内核的数据流，产生自动确认帧，以及控制所有模拟 RF 校准。

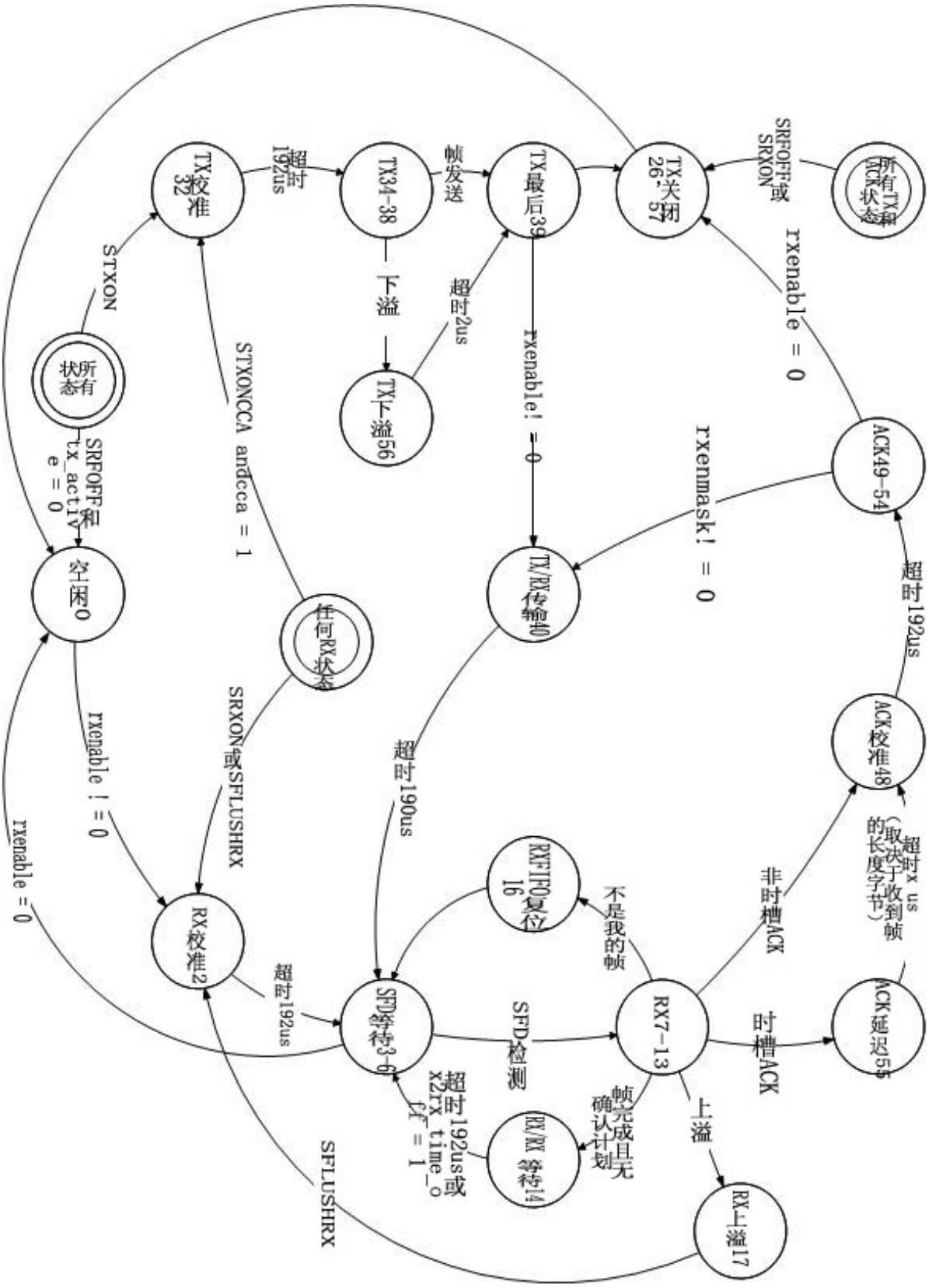


图 19-18　主要的 FSM

表 19-3 显示了 FSM 状态映射到可以从 FSMSTAT0 寄存器读出的号码。注意虽然可以读 FSM 的状态，但是这一信息不能用于控制应用程序软件的程序流。状态可以非常迅速地改变（每 32 MHz 时钟周期），且一个8 MHz SPI 不能捕获所有的活动。

表 19-3　FSM 状态映射

状态名称	状态号码，十进制	号码，十六进制	tx_active	rx_active
空闲	0	0x00	0	0
RX 校准	2	0x02	0	1
SFD 等待	3-6	0x03–0x06	0	1
RX	7-13	0x07–0x0D	0	1
RX/RX 等待	14	0x0E	0	1
RXFIFO 复位	16	0x10	0	1
RX 溢出	17	0x11	0	0
TX 校准	32	0x20	1	0
TX	34-38	0x22–0x26	1	0
TX 最后	39	0x27	1	0
TX/RX 发送	40	0x28	1	0
ACK 校准	48	0x30	1	0
ACK	49-54	0x31–0x36	1	0
ACK 延迟	55	0x37	1	0
TX 下溢	56	0x38	1	0
TX 关闭	26，57	0x1A, 0x39	1	0

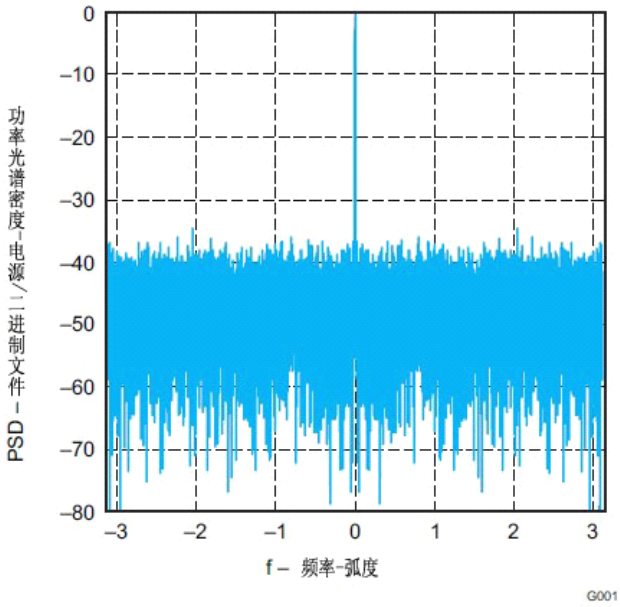
19.12　随机数的产生

RF 内核可以产生随机比特。当产生随机比特时要求芯片必须处于 RX。还必须确保芯片处于 RX 足够长的时间，用于瞬态消失。完成这一操作的一个很方便的方式是等待 RSSI 有效信号变为高。

来自 I 或 Q 通道的单个随机比特可以从寄存器 RFRND 中读。

随机测试表明，这一模块良好。但是，存在一个微小的 dc 组件。在一个简单的测试中，RFRND.IRND 寄存器被读数次，数据按字节分组。大约读出 2000 万字节。当解释为 0 到 255 之间的无符号整数，平均值是 127.6518，表示有一个 dc 组件。

2^{14} 的 FFT 首字节如图 19-19。注意 dc 组件是清晰可见的。2000 万字节的直方图（32 位二进制）如图 19-20。



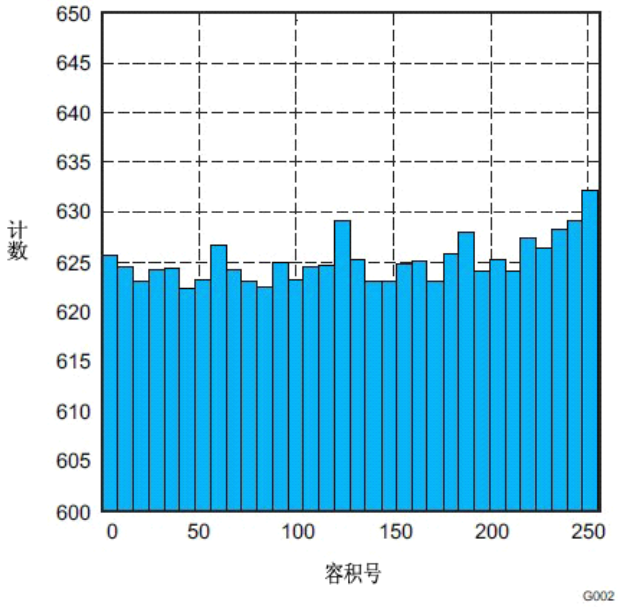


图 19-19　随机字节的 FFT　　图 19-20　2000 万字节的直方图，使用 RANDOM 指令产生

对于首个 2000 万每一位，每一个的概率是 P(1) = 0.500602 和 P(0) = 1 − P(1) =0.499398。

注意要完全限定随机数发生器为真正的随机数，需要做更细致的测试。互联网上有在这方面[8]、[9]可能有用的可用软件包。

19.13　数据包分析器和无线电测试输出信号

数据包分析器是一种无干扰观测发送或接收数据的方法。数据包分析器输出一个时钟和一个数据信号，它们必须在时钟的上升沿采样。这两个数据包分析信号在GPIO输出上观测。为了得到精确的时间戳，也要输出SFD信号。

因为无线电的数据率是 250 kbps，数据包分析器的时钟频率是 250 kHz。数据是串行输出的，每个字节的 MSB 首先输出，和实际的 RF 传输正好相反，但是当处理数据时更方便。可以使用一个 SPI 从模式来接收数据流。

当分析帧处于 TX 模式，调制器从 TX FIFO 读出的数据和数据包分析器输出的数据相同。但是，如果自动产生 CRC 使能，数据包分析器不能输出这 2 个字节。相反，它以 0x8080 替代 CRC 字节。该值不能发生在一个收到的帧的最后两个字节（当自动 CRC 校验使能），因此它为分析数据的接收器提供一种方法来区分是发送的帧还是接收的帧。

当分析帧处于 RX 模式，解调器写到 RX FIFO 的数据和数据包分析器输出的数据相同。换句话说，根据配置的设置，最后两个字节可以是收到的 CRC 值，或是可以自动替代 CRC 值的 CRC OK/RSSI/相关/SRCRESINDEX 值。

要设置数据包分析器信号或其他一些 RF 内核观测输出（总共最多 3 个：rfc_obs_sig0、rfc_obs_sig1 和 rfc_obs_sig2），用户必须遵守以下步骤：

步骤 1：确定哪个信号（rfc_obs_sig）要在哪个 GPIO 引脚（P1[0:5]）上输出。这使用 OBSSELx 控制寄存器（OBSSEL0−OBSSEL5）完成，控制观测结果输出到引脚 P1[0:5]上（覆盖那些引脚标准的 GPIO 行为）。

步骤 2：设置 RFC_OBS_CTRL 控制寄存器（RFC_OBS_CTRL0-RFC_OBS_CTRL2）来选择正确的信号（rfc_obs_sig），即对于数据包分析，需要 rfc_sniff_data 作为数据包分析器数据信号，rfc_sniff_clk 作为相应的时钟信号。

步骤 3：为了数据包分析，数据包分析器模块必须在 MDMTEST1 寄存器中使能。

19.14 命令选通/ CSMA-CA 处理器

命令选通/ CSMA-CA 处理器（CSP）提供控制 CPU 和无线电之间的通信。

CSP 通过 SFR 寄存器 RFST 以及 XREG 寄存器 CSPX、CSPY、CSPZ、CSPT、CSPSTAT、CSPCTRL 和 CSPPROG<n>（n 的范围是 0 到 23）和 CPU 通信。CSP 产生中断请求到 CPU。另外，CSP 通过观测 MAC 定时器事件和 MAC 定时器通信。

CSP 允许 CPU 发出命令选通到无线电，从而控制无线电的操作。

CSP 有两种操作模式，描述如下。

- 立即执行命令选通。
- 执行程序

立即命令选通被写作立即命令选通指令到 CSP，立即发给无线电模块。立即命令选通指令也只能用于控制 CSP。立即命令选通指令描述在 19.14.8 节。

执行程序模式意味着 CSP 从程序存储器或指令存储器执行一系列的指令，包括一个很短的用户定义的程序。可用的指令来自一个 20 条指令的集合。指令集定义在 19.14.8 节。所需的程序首先被 CPU 加载到 CSP 中，然后 CPU 指示 CSP 开始执行程序。

执行程序模式以及 MAC 定时器允许 CSP 自动执行 CSMA-CA 算法，因此充当 CPU 的协处理器。

CSP 的操作详细描述在以下章节。CSP 支持的命令选通和其他指令给定在 19.14.9 节。

RFST (0xE1) - RF CSMA-CA/选通处理器

位	名称	复位	R/W	描述
7:0	INSTR[7:0]	0xD0	R/W	写入该寄存器的数据被写到 CSP 指令存储器。读该寄存器返回当前执行的 CSP 指令。

19.14.1 指令存储器

CSP 执行从 24 字节指令存储器读出的单字节程序指令。通过 SFR 寄存器 RFST 连续写入指令存储器。指令写指针保留在 CSP 中，指明了写入 RFST 的下一条指令存储在指令存储器中的地址。为了调试目的，当前加载到 CSP 的程序可以从 XREG 寄存器 CSPPROG<n>读出。复位之后，指令写指针复位到位置 0。在每次寄存器 RFST 写入期间，指令写指针累加 1，直至到达存储器的终点，此时，指令写指针停止累加。第一个写入 RFST 的指令将存放在位置 0，也就是程序运行的起始点。至此，通过按照期望的顺序把每条指令写入 RFST 寄存器，全部的 24 条指令通过寄存器 RFST 写入指令存储器。

指令写指针可以通过下达立即命令选通指令 ISSTOP 复位到 0。除此之外，指令写指针也可以由于在程序中执行选通命 SSTOP 复位到 0。

复位之后，指令存储器中填充 SNOP（无操作）指令（操作码值 0xC0）。立即选通 ISCLEAR 清除指令存取，填充它为 SNOP 指令。

当 CSP 运行程序时，不可以使用 RFST 尝试将指令写入指令存储器。不遵守这一规则会导致程序出错，进而破坏指令存储器的内容。然而，立即命令选通指令可以写到 RFST（见 19.14.3 节）。

19.14.2 数据寄存器

CSP 有 3 个数据寄存器 CSPT、CSPX、CSPY 和 CSPZ，它们可以和 XREG 寄存器一样，被 CPU 读/写。这些寄存器可以被某些指令读取或修改，这样，CPU 就可以设置 CSP 的程序使用的参数，也可以读取 CSP 的程序状态。

任何指令都不可以修改数据寄存器 CSPT。数据寄存器 CSPT 用来设置 MAC 计数器溢出比较值。一旦运行的程序已经启动 CSP，该寄存器的内容就会因为每次 MAC 计数器的溢出而递减 1。当 CSPT 递减到 0 时，程序挂起，中断请求 IRQ_CSP_STOP 发出。如果 CPU 将 0xFF 写入数据寄存器 CSPT，则 CSPT 就不递减 1 了。

注意：如果寄存器 CSPT 不使用比较功能，那么该寄存器必须在程序运行之前设置为 0xFF。

19.14.3 程序运行

指令存储器填充完毕之后，当立即命令选通指令 ISSTART 写入寄存器 RFST 时，就开始运行程序。程序将一直运行到指令的最后位置，即运行到数据寄存器 CSPT 的内容为 0，或者运行到 SSTOP 指令已经执行或者运行到立即停止指令 ISSTOP 已经写入 RFST，或者运行到指令 SKIP 返回到超过指令存储器的最后位置。CSP 运行在系统时钟频率上，为了正确的无线电操作必须设置为 32MHz。

当程序即将运行时，可以将立即命令选通指令写入 RFST。在这种情况下，立即指令会绕过指令存储器里的指令执行，而指令存储器里的指令会在立即指令完成后执行。

程序运行期间，读 RFST 将返回当前指令即将执行的位置。只有一个例外，就是正在执行立即选通命令，这时，RFST 将返回 0xD0。

19.14.4 中断请求

CSP 有 3 个中断标志，它们可以产生 RF 中断向量，如下：

- IRQ_CSP_STOP：当 CSP 执行完毕存储器中最后一个指令，或者 CSP 由于下达指令 SSTOP 或 ISSTOP 而停止，或者寄存器 CSPT 等于 0 时，该中断标志有效
- IRQ_CSP_WT：当处理器在指令 WAIT W 或 WAITX 之后，继续执行下一条指令时，该中断标志有效
- RQ__CSP_INT：当处理器执行指令 INT 时，该中断标志有效

19.14.5 随机数指令

在更新指令 RANDXY 使用的随机数时，应当有一段时间延迟。因此如果使用这个值的指令 RANDXY 在前面的一个指令 RANDXY 之后立即发出，则两次读取的随机数数值可能相同。

19.14.6 运行 CSP 程序

装入和运行 CSP 程序的基本流程如图 19-21 所示。程序由于结束而停止运行时，当前程序遗留在程序存储器之中。这样一来，执行命令 ISSTART 就可以开始重新运行同样的程序。要清空程序的内容，使用 ISCLEAR 指令。

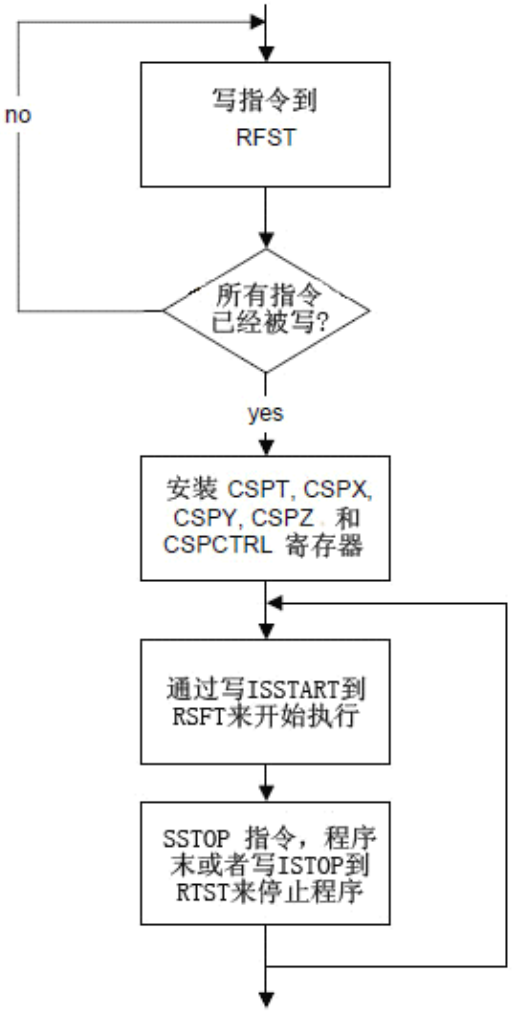


图 19-21　运行一个 CSP 程序

19.14.7　寄存器

CSPROG<N> (N 的范围从 0 到 23) (0x61C0 + N) – CSP 程序

位	名称	复位	R/W	描述
7:0	CSP_INSTR	0xd0	R	CSP 程序存储器的字节 N

CSPCTRL (0x61E0) – CSP 控制位

位	名称	复位	R/W	描述
7:1	-	0000 000	R0	保留。读作 0

CSPSTAT (0x61E1) – CSP 状态寄存器

位	名称	复位	R/W	描述
7:6	-	00	R0	保留。读作 0
5	CSP_RUNNING	0	R	1：CSP 运行 0：CSP 空闲
4:0	CSP_PC	0 0000	R	CSP 程序计数器

CSPX (0x61E2) – CSP X 寄存器

位	名称	复位	R/W	描述
7:0	CSPX	0x00	R/W	CSP X 数据寄存器。由 CSP 指令 WAITX、RANDXY、INCX、DECX 和条件指令使用

CSPY (0x61E3) – CSP Y 寄存器

位	名称	复位	R/W	描述
7:0	CSPY	0x00	R/W	CSP Y 数据寄存器。由 CSP 指令 RANDXY、INCY、DECY 和条件指令使用

CSPZ (0x61E4) – CSP Z 寄存器

位	名称	复位	R/W	描述
7:0	CSPZ	0x00	R/W	CSP Z 数据寄存器。由 CSP 指令 INCZ、DECZ 和条件指令使用

CSPT (0x61E5) – CSP T 寄存器

位	名称	复位	R/W	描述
7:0	CSPT	0xFF	R/W	CSP T 数据寄存器。当 CSP 程序运行，每次 MAC 定时器溢出内容就递减。当递减到 0，CSP 程序停止。设置 CSPT = 0xFF 防止寄存器递减。

19.14.8 指令集综述

本节给出了指令集的概述。目的是总结和定义指令操作码。每个指令的详细描述见 19.14.9 节。每个指令包括一个字节，写到 RFST 寄存器，存储在指令寄存器中。

程序中不使用立即选通指令（ISxxx）。当这些指令写到 RFST 寄存器，它们被立即执行。如果 CPU 已经执行了一个程序，当前指令被延迟，直到立即选通指令执行完毕。

对于未定义的操作码，CSP 的行为定义为无操作选通命令（SNOP）。

表 19-4 指令集综述

助记符	7	6	5	4	3	2	1	0	描述
SKIP<C>,<S>	0	S2	S1	S0	N	C2	C1	C0	条件 C 下跳过 S 指令。当条件（C XOR N）为真，跳过下一条 S 指令，或者执行下一条指令。如果 S=0，重新执行条件跳转（即忙碌循环条件为假）。跳过命令缓冲区的最后一条指令导致暗示一个 STOP 命令。 条件是： C=0 CCA 真 C=1 收到同步字，仍然接收数据包或发送同步字，仍然发送数据包（找到 SFD，帧尚未结束） C=2 MCU 控制位是 1。 C=3 命令缓冲区为空 C=4 寄存器 X=0 C=5 寄存器 Y=0 C=6 寄存器 Z=0 C=7 RSSI_VALID = 1
WAIT <W>	1	0	0	W4	W3	W2	W1	W0	等待 MAC 溢出 W 次。等待直到 MAC 定时器已经溢出了 W 次（W=0 等待 32 次），然后继续执行。当继续执行产生一个 IRQ_CSP_WAIT 中断请求。
RPT <C>	1	0	1	0	N	C2	C1	C0	当条件 C 时重复循环。如果条件 C 为真，去最后的 LABLE 指令后面的指令（地址在循环开始寄存器）；如果条件为假或没有执行 LABLE 指令，去下一条指令。 注意 SKIP 定义了条件 C，该表上面定义。
WEVENT1	1	0	1	1	1	0	0	0	等待 mact_event1 变为高，然后继续执行。
WEVENT2	1	0	1	1	1	0	0	1	等待 mact_event2 变为高，然后继续执行。
INT	1	0	1	1	1	0	1	0	产生一个 IRQ_CSP_MANINT。发出一个 IRQ_CSP_MANINT 中断请求。

表 19-4 指令集综述（续表）

助记符	7	6	5	4	3	2	1	0	描述
LABLE	1	0	1	1	1	0	1	1	设置下一条指令为重复循环的开始。将下一条指令的地址放到循环开始寄存器。
WAITX	1	0	1	1	1	1	0	0	等待 MAC 溢出[X]次，[X]是寄存器 X 的值。每次检测到一个 MAC 定时器溢出，X 递减。只要 X=0 继续执行。（如果指令运行时 X=0，不执行等待，直接继续执行。）当继续执行产生一个 IRQ_CSP_WAIT 中断请求。
RANDXY	1	0	1	1	1	1	0	1	随机值加载到寄存器 X[Y]LSB。
SETCMP1	1	0	1	1	1	1	1	0	设置输出 csp_mact_setcmp1 为高。这设置 MAC 定时器的比较值为当前定时器值。
INCX	1	1	0	0	0	0	0	0	增加寄存器 X
INCY	1	1	0	0	0	0	0	1	增加寄存器 Y
INCZ	1	1	0	0	0	0	1	0	增加寄存器 Z
DECX	1	1	0	0	0	0	1	1	递减寄存器 X
DECY	1	1	0	0	0	1	0	0	递减寄存器 Y
DECZ	1	1	0	0	0	1	0	1	递减寄存器 Z
INCMAXY<M>	1	1	0	0	1	M2	M1	M0	执行命令选通 S。发送命令选通 S 到 FFCTRL。支持多达 32 个命令选通。除了一般的命令选通，支持只能用于命令选通处理器的两个额外的命令选通： SNOP：无操作 SSTOP：停止命令选通处理器执行，使任一设置的标签无效。发出一个 IRQ_CSP_STOP 中断请求。
Isxxx	1	1	1	0	S3	S2	S1	S0	立即执行命令选通 S。立即发送命令选通 S 到 FFCTRL，绕过命令缓冲区中的指令。如果当前缓冲区指令是一个选通，它被延迟。除了一般的命令选通，支持只能用于命令选通处理器的两个额外的命令选通： ISSTART：命令选通处理器从命令缓冲区里的第一条指令开始执行。 ISSTOP：停止命令选通处理器执行，使任一设置的标签无效。发出一个 IRQ_CSP_STOP 中断请求。

19.14.9 指令集定义

指令的基本类型有 20 类。每条选通命令和立即选通指令可以分为 16 类子指令，这些子指令给出有效的 42 类不同的指令。以下小节详细描述了每个指令。

注意：本节使用以下定义

PC = CSP 程序计数器

X = RF 寄存器 CSPX

Y = RF 寄存器 CSPY

Z = RF 寄存器 CSPZ

T = RF 寄存器 CSPT

19.14.9.1 DECZ

功能：递减 Z

描述：Z 寄存器被减 1。原始值是 0x00 下溢到 0xFF。

操作：Z=Z-1

操作码：0xC5

7	6	5	4	3	2	1	0
1	1	0	0	0	1	0	1

19.14.9.2 DECY

功能：递减 Y

描述：Y 寄存器被减 1。原始值是 0x00 下溢到 0xFF。

操作：Y=Y-1

操作码：0xC4

7	6	5	4	3	2	1	0
1	1	0	0	0	1	0	0

19.14.9.3 DECX

功能：递减 X

描述：X 寄存器被减 1。原始值是 0x00 下溢到 0xFF。

操作：X=X-1

操作码：0xC3

7	6	5	4	3	2	1	0
1	1	0	0	0	0	1	1

19.14.9.4 INCZ

功能：增加 Z

描述：X 寄存器被加 1。原始值 0xFF 上溢到 0x00。

操作：Z=Z+1

操作码：0xC2

7	6	5	4	3	2	1	0
1	1	0	0	0	0	1	0

19.14.9.5 INCY

功能：增加 Y

描述：Y 寄存器被加 1。原始值 0xFF 上溢到 0x00。

操作：Y=Y+1

操作码：0xC1

7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	1

19.14.9.6 INCX

功能：增加 X

描述：X 寄存器被加 1。原始值 0xFF 上溢到 0x00。

操作：X=X+1

操作码：0xC0

7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0

19.14.9.7 INCMAXY

功能：增加 Y 不能大于 M。

描述：如果结果小于 M，Y 寄存器被加 1；否则，Y 寄存器载入值 M。

操作：Y = min(Y + 1, M)

操作码：0xC8 | M　　(M = 0 - 7)

7	6	5	4	3	2	1	0
1	1	0	0	1	M		

19.14.9.8 RANDXY

功能：加载随机值到 X。

描述：随机值加载到 X 寄存器的[Y] LSB。注意如果两个 RANDXY 指令连续发出，在两个指令中都使用同样的随机值。如果 Y 等于零或大于 7，那么加载一个 8 位的随机值到 X。

操作：X[Y − 1:0]: = RNG_DOUT[Y − 1:0], X[7:Y]: = 0

操作码：0xBD

7	6	5	4	3	2	1	0
1	0	1	1	1	1	0	1

19.14.9.9 INT

功能：中断

描述：当执行该指令时声明中断 IRQ_CSP_INT。

操作：IRQ_CSP_INT=1

操作码：0xBA

7	6	5	4	3	2	1	0
1	0	1	1	1	0	1	0

19.14.9.10 WAITX

功能： 等待 MAC 定时器溢出

描述： 等待 MAC 定时器溢出[X]次，[X]是寄存器 X 的值。每次检测到一次 MAC 定时器溢出，寄存器 X 的值就递减。只要 X=0，程序继续执行（当指令运行时如果 X=0，不执行等待，直接继续执行）。当继续执行产生一个 IRQ_CSP_WAIT 中断请求。注意：比起 WAIT W，区别是 W 是固定值，而 X 是寄存器值（有可能被改变，这样 WAITX 指令运行时溢出的次数实际上不对应 X 的值）。

操作： 当 MAC 定时器溢出=true X = X − 1

当 X > 0 PC = PC

当 X = 0 PC = PC + 1

操作码：0xBC

7	6	5	4	3	2	1	0
1	0	1	1	1	1	0	0

19.14.9.11 SETCMP1

功能： 设置 MAC 定时器的比较值为当前定时器值

描述： 设置 MAC 定时器的比较值为当前定时器值

操作： Csp_mact_setcmp1 = 1

操作码：0xBE

7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	0

19.14.9.12 WAIT W

功能： 等待 W 次 MAC 定时器溢出

描述： 等待直到 MAC 定时器溢出次数等于值 W。如果 W=0，指令等待 32 次溢出。程序继续执行下一条指令，当等待条件为真，声明中断标志 IRQ_CSP_WT。

操作： 当 MAC 定时器溢出次数=true< W PC=PC

当 MAC 定时器溢出次数=true=W PC = PC + 1

操作码：0x80 | W （W = 0 - 31）

7	6	5	4	3	2	1	0
1	0	0	W				

19.14.9.13 WEVENT1

功能： 等待直到 MAC 定时器事件 1

描述： 等待直到下一个 MAC 定时器事件。当等待条件为真程序继续执行下一条指令。

操作： 当 MAC 定时器比较=false PC=PC

当 MAC 定时器比较=true PC = PC + 1

操作码： 0xB8

7	6	5	4	3	2	1	0
1	0	1	1	1	0	0	0

19.14.9.14 WEVENT2

功能： 等待直到 MAC 定时器事件 2

描述： 等待直到下一个 MAC 定时器事件。当等待条件为真程序继续执行下一条指令。

操作： 当 MAC 定时器比较=false PC=PC

当 MAC 定时器比较=true PC = PC + 1

操作码： 0xB9

7	6	5	4	3	2	1	0
1	0	1	1	1	0	0	1

19.14.9.14 LABLE

功能： 设置循环标签

描述： 设置下一条指令为循环的开始。如果当前指令是指令存储器的最后一条指令，那么当前 PC 设置为循环的开始。如果执行若干标签指令，执行的最后一个标签是活跃的标签。前面的标签被移除，意味着只支持一个级别的循环。

操作： LABEL: = PC + 1

操作码： 0xBB

7	6	5	4	3	2	1	0
1	0	1	1	1	0	1	1

19.14.9.16 RPT C

功能： 条件重复

描述： 如果条件 C 为真，那么跳到最后一个 LABLE 指令定义的指令，即跳到循环的开始。如果条件或一个 LABLE 指令尚未执行，那么从下一条指令继续执行。条件 C 可以通过设置 N=1 否定，描述在下表。

条件代码 C	描述	功能
000	CCA 为真	CCA=1
001	接收同步字，仍然接收数据包或发送同步字，仍然发送数据包	SFD=1
010	CPU 控制为真	CSPCTRL.CPU_CTRL = 1
011	指令存储器结束	PC=23
100	寄存器 X=0	X=0
101	寄存器 Y=0	Y=0
110	寄存器 Z=0	Z=0
111	RSSI 值有效	RSSI_VALID = 1

操作： 当(C xor N) = true PC=LABLE
当(C xor N) = false 或 LABLE=未设置 PC=PC+1

操作码： 0xA0 | N | C (N = 0, 8; C = 0 - 7)

7	6	5	4	3	2	1	0
1	0	1	0	N	C		

19.14.9.17 SKIP S，C

功能： 条件跳过指令

描述： 条件 C 时跳过 S 指令（其中条件 C 可以被否定：N=1）。当条件（C xor N）为真，跳过下一条 S 指令，转而执行下一条指令。如果 S=0，重新执行条件跳转（即繁忙的循环，直到条件为假）。跳过命令缓冲区的最后一条指令导致暗示一个 STOP 命令。

条件代码 C	描述	功能
000	CCA 为真	CCA=1
001	接收同步字，仍然接收数据包或发送同步字，仍然发送数据包	SFD=1
010	CPU 控制为真	CSPCTRL.CPU_CTRL = 1
011	指令存储器结束	PC=23
100	寄存器 X=0	X=0
101	寄存器 Y=0	Y=0
110	寄存器 Z=0	Z=0
111	RSSI 值有效	RSSI_VALID = 1

操作： 当(C xor N) = true PC=PC+S+1
当(C xor N) = false PC=PC+1

操作码：

7	6	5	4	3	2	1	0
0	S			N	C		

19.14.9.18 STOP

功能： 停止程序执行

描述： SSTOP 指令停止 CSP 程序的执行。

操作： 停止执行

操作码：0xD2

7	6	5	4	3	2	1	0
1	1	0	1	0	0	1	0

19.14.9.19 SNOP

功能： 无操作

描述： 在下一条指令继续操作

操作： PC=PC+1

操作码：0xD0

7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	0

19.14.9.20 SRXON

功能： 为 RX 使能并校准频率合成器

描述： SRXON 指令声明输出 FFCTL_SRXON_STRB 来为 RX 使能并校准频率合成器。在执行下一条指令之前指令等待无线电确认命令。

操作： SRXON

操作码：0xD3

7	6	5	4	3	2	1	0
1	1	0	1	0	0	1	1

19.14.9.21 STXON

功能： 校准之后使能 TX

描述： 校准后 STXON 指令使能 TX。在执行下一条指令之前指令等待无线电确认命令。如果设置了 SET_RXENMASK_ON_TX 就设置 RXENABLE 中的一个位。

操作： STXON

操作码：0xD9

7	6	5	4	3	2	1	0
1	1	0	1	1	0	0	1

19.14.9.22 STXONCCA

功能：如果 CCA 表示清除一个通道使能校准和 TX

描述：如果 CCA 表示清除一个通道，校准后 STXONCCA 指令使能。

操作：STXONCCA

操作码：0xDA

7	6	5	4	3	2	1	0
1	1	0	1	1	0	1	0

19.14.9.23 SSAMPLECCA

功能：采样当前 CCA 值到 SAMPLED_CCA 中

描述：当前 CCA 值写到 XREG 的 SAMPLED_CCA 中。

操作：SSAMPLECCA

操作码：0xDB

7	6	5	4	3	2	1	0
1	1	0	1	1	0	1	1

19.14.9.24 SRFOFF

功能：禁用 RX/TX 和频率合成器。

描述：SRFOFF 指令声明禁用 RX/TX 和频率合成器。

操作：SRFOFF

操作码：0xDF

7	6	5	4	3	2	1	0
1	1	0	1	1	1	1	1

19.14.9.25 SFLUSHRX

功能：清除 RXFIFO 缓冲区并复位解调器

描述：SFLUSHRX 指令清除 RXFIFO 缓冲区并复位解调器。在执行下一条指令之前指令等待无线电确认命令。

操作：SFLUSHRX

操作码：0xDD

7	6	5	4	3	2	1	0
1	1	0	1	1	1	0	1

19.14.9.26 SFLUSHTX

功能：清除 TXFIFO 缓冲区

描述：SFLUSHTX 指令清除 TXFIFO 缓冲区并复位解调器。在执行下一条指令之前指令等待无线电确认命令。

操作：SFLUSHTX

操作码：0xDE

7	6	5	4	3	2	1	0
1	1	0	1	1	1	1	0

19.14.9.27 SACK

功能：发送未决域清除的确认帧

描述：SACK 指令发送一个确认帧。在执行下一条指令之前指令等待无线电确认命令。

操作：SACK

操作码：0xD6

7	6	5	4	3	2	1	0
1	1	0	1	0	1	1	0

19.14.9.28 SACKPEND

功能：发送未决域设置的确认帧

描述：SACKPEND 指令发送一个确认帧，未决域设置。在执行下一条指令之前指令等待无线电确认命令 。

操作：SACKPEND

操作码：0xD7

7	6	5	4	3	2	1	0
1	1	0	1	0	1	1	1

19.14.9.29 SNACK

功能：中止发送确认帧

描述：SACKPEND 指令中止发送确认到当前收到的帧。

操作：SNACK

操作码：0xD8

7	6	5	4	3	2	1	0
1	1	0	1	1	0	0	0

19.14.9.30 SRXMASKBITSET

功能：设置 RXENABLE 中的位

描述：SRXMASKBITSET 指令设置 RXENABLE 中的位 5。

操作：SRXMASKBITSET

操作码：0xD4

7	6	5	4	3	2	1	0
1	1	0	1	0	1	0	0

19.14.9.31 SRXMASKBITCLR

功能：清除 RXENABLE 中的位

描述：SRXMASKBITCLR 指令设置 RXENABLE 中的位 5。

操作：SRXMASKBITCLR

操作码：0xD5

7	6	5	4	3	2	1	0
1	1	0	1	0	1	0	1

19.14.9.32 ISSTOP

功能：停止程序执行

描述：ISSTOP 指令停止 CSP 程序执行，且声明 IRQ_CSP_STOP 中断标志。

操作：停止执行

操作码：0xE2

7	6	5	4	3	2	1	0
1	1	1	0	0	0	1	0

19.14.9.33 ISSTART

功能：开始程序执行

描述：ISSTART 指令从写到指令存储器的第一条指令开始执行 CSP 程序。

操作：PC := 0，开始执行

操作码：0xE1

7	6	5	4	3	2	1	0
1	1	1	0	0	0	0	1

19.14.9.34 ISRXON

功能：为 RX 使能并校准频率合成器

描述：ISRXON 指令立即为 RX 使能并校准频率合成器。

操作：SRXON

操作码：0xE3

7	6	5	4	3	2	1	0
1	1	1	0	0	0	1	1

19.14.9.35 ISRXMASKBITSET

功能：设置 RXENABLE 中的位

描述：ISRXMASKBITSET 指令立即设置 RXENABLE 中的位 5。

操作：SRXMASKBITSET

操作码：0xE4

7	6	5	4	3	2	1	0
1	1	1	0	0	1	0	0

19.14.9.36 ISRXMASKBITCLR

功能：清除 RXENABLE 中的位

描述：ISRXMASKBITCLR 指令立即清除 RXENABLE 中的位 5。

操作：SRXMASKBITCLR

操作码：0xE45

7	6	5	4	3	2	1	0
1	1	1	0	0	1	0	1

19.14.9.37 ISTXON

功能：校准之后使能 TX

描述：校准之后 ISTXON 指令立即使能 TX。在执行下一条指令之前指令等待无线电确认命令。

操作：STXON_STRB

操作码：0xE9

7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	1

19.14.9.38 ISTXONCCA

功能：如果 CCA 表示清除一个通道使能校准和 TX

描述：如果 CCA 表示清除一个通道，校准后 ISTXONCCA 指令立即使能 TX。

操作：STXONCCA

操作码：0xEA

7	6	5	4	3	2	1	0
1	1	1	0	1	0	1	0

19.14.9.39 ISSAMPLECCA

功能：采样当前 CCA 值到 SAMPLED_CCA 中

描述：当前 CCA 值立即写到 XREG 的 SAMPLED_CCA 中

操作：SSAMPLECCA

操作码：0xEB

7	6	5	4	3	2	1	0
1	1	1	0	1	0	1	1

19.14.9.40 ISRFOFF

功能：禁用 RX/TX 和频率合成器。

描述：ISRFOFF 指令立即禁用 RX/TX 和频率合成器。

操作：FFCTL_SRFOFF_STRB = 1

操作码：0xEF

7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	1

19.14.9.41 ISFLUSHRX

功能：清除 RXFIFO 缓冲区并复位解调器。

描述：ISFLUSHRX 指令立即清除 RXFIFO 缓冲区并复位解调器。

操作：SFLUSHRX

操作码：0xED

7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	1

19.14.9.42 ISFLUSHTX

功能：清除 TXFIFO 缓冲区

描述：ISFLUSHTX 指令立即清除 TXFIFO 缓冲区。

操作：SFLUSHTX

操作码：0xEE

7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	0

19.14.9.43 ISACK

功能：发送未决位清除的确认帧

描述：ISACK 指令立即发送一个确认帧。

操作：SACK

操作码：0xE6

7	6	5	4	3	2	1	0
1	1	1	0	0	1	1	0

19.14.9.44 ISACKPEND

功能：发送未决位设置的确认帧

描述：ISACKPEND 指令立即发送一个确认帧，未决域设置。在执行下一条指令之前，指令等待无线电接收并解释命令。

操作：SACKPEND

操作码：0xE7

7	6	5	4	3	2	1	0
1	1	1	0	0	1	1	1

19.14.9.45 ISNACK

功能：中止发送确认帧

描述：ISNACK 指令立即阻止向当前收到的帧发送一个确认帧。

操作：SNACK

操作码：0xE8

7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	0

19.14.9.46 ISCLEAR

功能：清除 CSP 程序存储器，复位程序计数器

描述: ISCLEAR 立即清除程序存储器，复位程序计数器，并中止任一运行的程序。不产生停止中断。LABLE 指针被清除。

操作：PC := 0，清除程序存储器

操作码：0xE8

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1

19.15 寄存器

表 19-5 寄存器概览

地址（十六进制）	+ 0x000	+ 0x001	+ 0x002	+ 0x003
0x6180	FRMFILT0	FRMFILT1	SRCMATCH	SRCSHORTEN0
0x6184	SRCSHORTEN1	SRCSHORTEN2	SRCEXTEN0	SRCEXTEN1
0x6188	SRCEXTEN2	FRMCTRL0	FRMCTRL1	RXENABLE
0x618C	RXMASKSET	RXMASKCLR	FREQTUNE	FREQCTRL
0x6190	TXPOWER	TXCTRL	FSMSTAT0	FSMSTAT1
0x6194	FIFOPCTRL	FSMCTRL	CCACTRL0	CCACTRL1
0x6198	RSSI	RSSISTAT	RXFIRST	RXFIFOCNT
0x619C	TXFIFOCNT	RXFIRST_PTR	RXLAST_PTR	RXP1_PTR
0x61A0		TXFIRST_PTR	TXLAST_PTR	RFIRQM0
0x61A4	RFIRQM1	RFERRM	RESERVED	RFRND
0x61A8	MDMCTRL0	MDMCTRL1	FREQEST	RXCTRL
0x61AC	FSCTRL	FSCAL0	FSCAL1	FSCAL2
0x61B0	FSCAL3	AGCCTRL0	AGCCTRL1	AGCCTRL2
0x61B4	AGCCTRL3	ADCTEST0	ADCTEST1	ADCTEST2
0x61B8	MDMTEST0	MDMTEST1	DACTEST0	DACTEST1
0x61BC	DACTEST2	ATEST	PTEST0	PTEST1
0x61C0	CSPPROG0	CSPPROG1	CSPPROG2	CSPPROG3
0x61C4	CSPPROG4	CSPPROG5	CSPPROG6	CSPPROG7
0x61C8	CSPPROG8	CSPPROG9	CSPPROG10	CSPPROG11
0x61CC	CSPPROG12	CSPPROG13	CSPPROG14	CSPPROG15
0x61D0	CSPPROG16	CSPPROG17	CSPPROG18	CSPPROG19
0x61D4	CSPPROG20	CSPPROG21	CSPPROG22	CSPPROG23
0x61D8				
0x61DC				
0x61E0	CSPCTRL	CSPSTAT	CSPX	CSPY
0x61E4	CSPZ	CSPT		
0x61E8				RFC_OBS_CTRL0
0x61EC	RFC_OBS_CTRL1	RFC_OBS_CTRL2		
0x61F0				
0x61F4				
0x61F8			TXFILTCFG	

19.15.1 寄存器设置更新

本节包括寄存器设置的总结，即必须从它们的默认值更新到获得最佳性能的值。

以下设置必须在 RX 和 TX 下都设置。虽然不是所有的设置对 RX 和 TX 都是必须的，但它们是为了简单推荐的（允许一组设置写到初始化代码中）。

表 19-6　需要从其默认值更新的寄存器

寄存器名称	新的值（十六进制）	描述
AGCCTRL1	0x15	调整 AGC 目标值。
TXFILTCFG	0x09	设置 TX 抗混叠过滤器以获得合适的贷款。
FSCAL1	0x00	和默认设置比较，降低 VCO 泄露大约 3dB。推荐默认设置以获得最佳 EVM。

19.15.2　寄存器访问模式

表 19-7 的模式一列显示了每一位允许哪种访问。描述一列给出了不同选项的意思。

表 19-7　寄存器位访问模式

模式	描述
R	读
W	写
R0	读常量 0
R1	读常量 1
W1	只能写 1
W0	只能写 0
R*	读的值不是实际寄存器的值，而是模块看见的值。这一般用于可以自动产生的配置值（通过校准、动态控制等等），或手动覆盖一个寄存器值。一个例子结构如图 19-22 的 AGCCTRL2 寄存器所示。

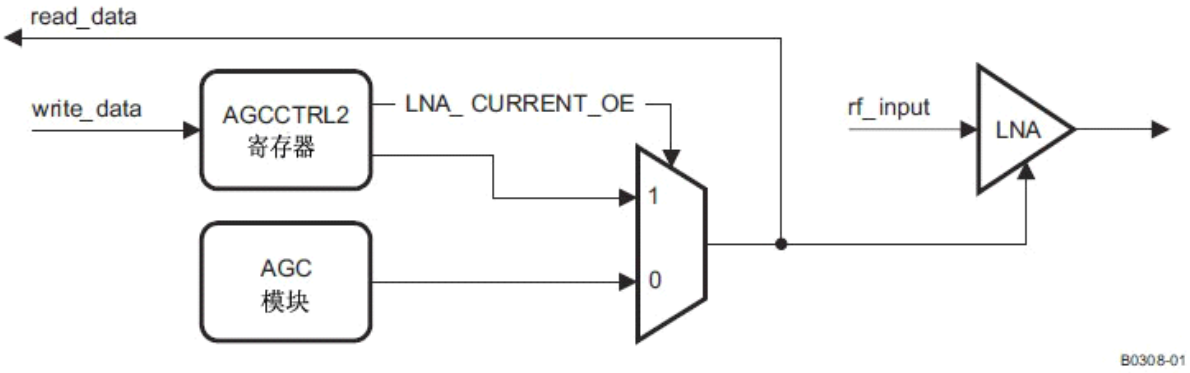


图 19-22　硬件结构 R*寄存器访问模式的例子

19.15.3　寄存器描述

FRMFILT0 (0x6180)-帧过滤

位号码	名称	复位	R/W	描述
7	-	0	R/W	保留。总是写 0。
6:4	FCF_RESERVED_MASK[2:0]	000	R/W	用于过滤帧控制域（FCF）的保留部分。FCF_RESERVED_MASK[2:0]是与 FCF[9:7]AND。如果结果非零，且帧过滤使能，该帧被拒绝。
3:2	MAX_FRAME_VERSION[1:0]	11	R/W	用于过滤帧控制域（FCF）的帧版本域。如果 FCF[13:12]（帧版本子域）高于 MAX_FRAME_VERSION[1:0]且帧过滤使能，该帧被拒绝。
1	PAN_COORDINATOR		R/W	当设备是一个 PAN 协调器，必须设置为高，以接受没有目标地址的帧（如 802.15.4(b)中 7.5.6.2 节所述）。 0：设备不是 PAN 协调器 1：设备是 PAN 协调器
0	FRAME_FILTER_EN	1	R/W	使能帧过滤 当该位设置，无线电执行 802.15.4(b)中 7.5.6.2 节所述的帧过滤，第三过滤级别。FRMFILT0[6:1]和 FRMFILT1[7:1]，以及本地地址信息定义了过滤算法的行为。

FRMFILT1 (0x6181) - 帧过滤

位号码	名称	复位	R/W	描述
7	ACCEPT_FT_4TO7_RESERVED	0	R/W	定义是否接受保留帧。保留帧的帧类型=（100，101，110，111）。 0：拒绝 1：接受
6	ACCEPT_FT_3_MAC_CMD	1	R/W	定义是否接受 MAC 命令帧。MAC 命令帧的帧类型=011。 0：拒绝 1：接受
5	ACCEPT_FT_2_ACK	1	R/W	定义是否接受确认帧。确认帧的帧类型=010。 0：拒绝 1：接受
4	ACCEPT_FT_1_DATA	1	R/W	定义是否接受数据帧。数据帧的帧类型=001。 0：拒绝 1：接受
3	ACCEPT_FT_0_BEACON	1	R/W	定义是否接受信标帧。信标帧的帧类型=000。 0：拒绝 1：接受
2:1	MODIFY_FT_FILTER[1:0]	00	R/W	在执行帧类型过滤之前，这些位用于修改一个收到帧的帧类型域。修改不影响写到 RX FIFO 中的帧。 00：不变 01：颠倒 MSB 10：设置 MSB 为 0 11：设置 MSB 为 1
0	-	0	R/W	保留。总是写 0

SRCMATCH (0x6182) – 源地址匹配和未决位

位号码	名称	复位	R/W	描述
7:3	-	0000 0	R/W	保留。总是写 0
2	PEND_DATAREQ_ONLY	1	R/W	当该位设置，AUTOPEND 功能也要求接收的帧是一个*数据请求* MAC 命令帧。
1	AUTOPEND	1	R/W	自动确认未决标志使能。 接收一个帧时，（可能）返回的确认的未决位自动设置，只要： – 设置了FRMFILT0.FRAME_FILTER_EN – 设置了SRCMATCH.SRC_MATCH_EN – 设置了SRCMATCH.AUTOPEND – 收到的帧匹配当前的SRCMATCH.PEND_DATAREQ_ONLY设置。 –收到的源地址至少匹配一个源匹配表条目，在 SHORT_ADDR_EN 和SHORT_PEND_EN，或EXT_ADDR_EN和EXT_PEND_EN中都使能。 注意：SHORT_PEND_EN 和 EXT_PEND_EN 的详细信息见存储器映射描述（19.4 节）。
0	SRC_MATCH_EN	1	R/W	源地址匹配使能（如果 FRMFILT0.FRAME_FILTER_EN = 0 该位不重要。）

SRCSHORTEN0 (0x6183) – 短地址匹配

位号码	名称	复位	R/W	描述
7:0	SHORT_ADDR_EN[7:0]	0x00	R/W	24 位字 SHORT_ADDR_EN 的 7:0 部分，为 24 个短地址表条目每一个使能/禁用源地址匹配。

SRCSHORTEN1 (0x6184) – 短地址匹配

位号码	名称	复位	R/W	描述
7:0	SHORT_ADDR_EN[15:8]	0x00	R/W	24 位字 SHORT_ADDR_EN 的 15:8 部分。见 SRCSHORTEN0 的描述。

SRCSHORTEN2 (0x6185) – 短地址匹配

位号码	名称	复位	R/W	描述
7:0	SHORT_ADDR_EN[23:16]	0x00	R/W	24 位字 SHORT_ADDR_EN 的 23:16 部分。见 SRCSHORTEN0 的描述。

SRCEXTEN0 (0x6186) – 扩展地址匹配

位号码	名称	复位	R/W	描述
7:0	EXT_ADDR_EN[7:0]	0x00	R/W RO	24 位字 EXT_ADDR_EN 的 7:0 部分。使能/禁用 12 个扩展地址表条目每一个的地址匹配。 写访问：表条目 n（0 到 7）的扩展地址使能映射到 EXT_ADDR_EN[2n]。所有 EXT_ADDR_EN[2n+1]位只读，何时写入不重要。 读访问：表条目 n（0 到 7）的扩展地址使能映射到 EXT_ADDR_EN[2n]和 EXT_ADDR_EN[2n+1]。 可选的安全功能：要确保源匹配表的一个条目在更新时未使用，更新时设置对应的 EXT_ADDR_EN 位为 0。

SRCEXTEN1 (0x6187) – 扩展地址匹配

位号码	名称	复位	R/W	描述
7:0	EXT_ADDR_EN[15:8]	0x00	R/W	24 位字 EXT_ADDR_EN 的 15:8 部分。见 SRCEXTEN0 的描述。

SRCEXTEN2 (0x6188) – 扩展地址匹配

位号码	名称	复位	R/W	描述
7:0	EXT_ADDR_EN[23:16]	0x00	R/W	24 位字 EXT_ADDR_EN 的 23:16 部分。见 SRCEXTEN0 的描述。

FRMCTRL0 (0x6189) – 帧处理

位号码	名称	复位	R/W	描述
7	APPEND_DATA_MODE	0	R/W	当 AUTOCRC = 0：不重要 当 AUTOCRC = 1： 0：RSSI + crc_ok 位和 7 位相关值附加到每个收到帧的末尾。 1：RSSI + crc_ok 位和 7 位 SRCRESINDEX 附加到每个收到帧的末尾。详见表 19-1。
6	AUTOCRC	1	R/W	在 TX 中 1：硬件产生一个 CRC-16（ITU-T）并附加到发送帧。不需要写最后 2 个字节到 TXBUF。 0：没有 CRC-16 附加到帧。帧的最后 2 个字节必须手动产生并写到 TXBUF（如果没有，发生 TX_UNDERFLOW）。
				在 RX 中 1：硬件检查一个 CRC-16，并以一个 16 位状态字代替 RX FIFO，包括一个 CRC OK 位。状态字可通过 APPEND_DATA_MODE 控制。 0：帧的最后 2 个字节（crc-16 域）存储在 RXFIFO。CRC 校验（如果有必须手动完成）。
5	AUTOACK	0	R/W	定义无线电是否自动发送确认帧。当 autoack 使能，所有经过地址过滤接受的帧都设置确认请求标志，在接收之后自动确认一个有效的 CRC12 符号周期。 0：autoack 禁用 1：autoack 使能
4	ENERGY_SCAN	0	R/W	定义 RSSI 寄存器是否包括自能量扫描使能以来最新的信号强度或峰值信号强度。 0：最新的信号强度 1：峰值信号强度
3:2	RX_MODE[1:0]	00	R/W	设置 RX 模式 00：一般模式，使用 RXFIFO 01：保留 10：RXFIFO 循环忽略 RXFIFO 的溢出，无限接收。 11：和一般模式一样，除了禁用符号搜索。当不用于找到符号可以用于测量 RSSI 或 CCA。
1:0	TX_MODE[1:0]	00	R/W	设置 TX 的测试模式 00：一般操作，发送 TXFIFO 01：保留。不能使用 10：TXFIFO 循环忽略 TXFIFO 的溢出和读循环，无限发送 11：发送来自 CRC 的伪随机数，无限发送。

FRMCTRL1 (0x618A) – 帧处理

位号码	名称	复位	R/W	描述
7:3	-	0000 0	R0	读作零。
2	PENDING_OR	0	R/W	定义输出确认帧的未决数据位总是设置为 1，还是由主要 FSM 和地址过滤控制。 0：未决数据位由主要 FSM 和地址过滤控制。 1：未决数据位总是 1。
1	IGNORE_TX_UNDERF	0	R/W	定义是否忽略 TX 溢出。 0：一般 TX 操作。检测 TX 溢出，如果发生溢出中止 TX。 1：忽略 TX 溢出。发送长度域给定的字节数。
0	SET_RXENMASK_ON_TX	1	R/W	定义 STXON 设置 RXENABLE 寄存器的位 14，还是保持不变。 0：不影响 RXENABLE。 1：设置 RXENABLE 的位 14。用于向后兼容 CC2420。

RXENABLE (0x618B) – RX 使能

位号码	名称	复位	R/W	描述
7:0	RXENMASK[7:0]	0x00	R	RXENABLE 使能寄存器。该寄存器的非零值导致主要 FSM 在传输之后且确认传输之后，空闲时使能接收器。 以下选通可以修改 RXENMASK： SRXON：设置RXENMASK的位7 STXON：如果SET_RXENMASK_ON_TX = 1，设置RXENMASK的位8 SRFOFF：清除RXENMASK的所有位 SRXMASKBITSET：设置RXENMASK的位5 SRXMASKBITCLR：清除 RXENMASK 的位 5 通过访问寄存器 RXMASKSET 和 RXMASKCLR，RXENABLE 可以直接由 CPU 控制。 如果 CSP 和 CPU 都尝试同时修改 RXENMASK，操作之间可能有冲突。要处理同时访问 RXENMASK 的情况，使用以下规则： -如果两个源不冲突（它们修改寄存器不同的部分），两个修改 RXENMASK 的请求都被处理。 -如果都尝试同时修改屏蔽，RXMASKSET 的总线写操作和 RXMASKCLR 优先于 CSP。强烈推荐避免这种情况。

RXMASKSET (0x618C) – RX 使能

位号码	名称	复位	R/W	描述
7:0	RXENMASKSET[7:0]	0x00	W/R0	写时，写入的数据和 RXENMASK 相或，并存储在 RXENMASK 中。

RXMASKCLR (0x618D) – RX 禁用

位号码	名称	复位	R/W	描述
7:0	RXENMASKCLR[7:0]	0x00	W/R0	写时，写入的数据被颠倒，然后和RXENMASK相与，并存储在RXENMASK中。 例如，如果写1到该寄存器的一个或多个位，RXENMASK中相应的位被清除。

RFIRQM0 (0x61A3) – RF 中断屏蔽

位号码	名称	复位	R/W	描述
7:0	RFIRQM[7:0]	0x00	R/W	位屏蔽的屏蔽输出中断源 位的位置 7：RXMASKZERO 6：RXPKTDONE 5：FRAME_ACCEPTED 4：SRC_MATCH_FOUND 3：SRC_MATCH_DONE 2：FIFOP 1：SFD 0：ACT_UNUSED

RFIRQM1 (0x61A4) – RF 中断屏蔽

位号码	名称	复位	R/W	描述
7:0	RFIRQM[14:8]	0x00	R/W	位屏蔽的屏蔽输出中断源 位的位置 7：保留 6：保留 5：CSP_WAIT 4：CSP_STOP 3：CSP_MANINT 2：RF_IDLE 1：TXDONE 0：TXACKDONE

RFERRM (0x61A5) – RF 错误中断屏蔽

位号码	名称	复位	R/W	描述
7:0	RFERRM[7:0]	0x00	R/W	位屏蔽的屏蔽输出中断源 位的位置 7：保留 6：STROBE_ERR 5：TXUNDERF 4：TXOVERF 3：RXUNDERF 2：RXOVERF 1：RXABO 0：NLOCK

FREQCTRL (0x618F) – 控制 RF 频率

位号码	名称	复位	R/W	描述
7	-	0	R0	读作 0
6:0	FREQ[6:0]	0x0B (2405 MHz)	R/W	频率控制字。 ¦ RF = ¦ LO = (2394 + FREQ[6:0]) MHz FREQ[6:0]中的频率字是2394的一个偏移值。设备支持的频率范围从2394 MHz到2507 MHz。FREQ[6:0]可用的设置从0到113。这一范围之外的设置(114–127)给出的频率是2507 MHz。 IEEE 802.15.4-2006指定的频率范围从2405 MHz到2480 MHz，16通道，5 MHz步长。通道编号从11到26。因此对于符合IEEE 802.15.4-2006的系统，唯一有效设置是FREQ[6:0]= 11 + 5 (通道号码 – 11)。

FREQTUNE (0x618E)晶振频率调整

位号码	名称	复位	R/W	描述
7:4	-	0x0	R0	读作 0
3:0	XOSC32M_TUNE[3:0]	0xF	R/W	调整晶振 默认设置1111不调整XOSC。修改默认设置，从另外的电容转换到振荡器能有效降低XOSC的频率。因此，较高的设置给出较高的频率。

TXPOWER (0x6190) – 控制输出功率

位号码	名称	复位	R/W	描述
7:0	PA_POWER [7:0]	0xF5	R/W	PA 功率控制。 注意：转到 TX 之前，必须更新该值。推荐值请参考 CC2530 数据手册，或见 19.8.13 节。

TXCTRL (0x6191) – 控制 TX 设置

位号码	名称	复位	R/W	描述
7	-	0	R0	保留
6:4	DAC_CURR[2:0]	10	R/W	改变 DAC 的电流。
3:2	DAC_DC[1:0]	01	R/W	根据 TX 混合器调整 dc 水平。
1:0	TXMIX_CURRENT[1:0]	0x01	R/W	发送混合器内核电流：电流随着设置的增加而增加。

FSMSTAT0 (0x6192) – 无线电状态寄存器

位号码	名称	复位	R/W	描述
7	-	0	R	保留
6	CAL_RUNNING	0	R	频率合成器校准状态 0：校准完成或未开始 1：校准正在进行
5:0	FSM_FFCTRL_STATE[5:0]	-	R	给出 FIFO 和帧控制（FFCTRL）有限状态机制的当前状态。

FSMSTAT1 (0x6193) – 无线电状态寄存器

位号码	名称	复位	R/W	描述
7	FIFO	0	R	只要 RXFIFO 中有数据，FIFO 为高。RXFIFO 溢出期间为低。
6	FIFOP	0	R	RXFIFO 中经过帧过滤的的数据多于 FIFOP_THR 字节，FIFOP 设置为高。 RXFIFO 中至少有一个完整的帧，FIFOP 设置为高。 从 RXFIFO 读一个字节，FIFOP 再次设置为低，这使得 FIFO 中有 FIFOP_THR 个字节。 RXFIFO 溢出期间 FIFOP 为高。
5	SFD	0	R	在 TX 下 0：已发送一个带有 SFD 的完整帧，或没有发送 SFD。 1：已发送 SFD。 在 RX 下 0：已接收一个完整帧，或没有接收 SFD。 1：已接收 SFD。
4	CCA	0	R	空闲通道评估。根据 CCA_MODE 的设置。详见 CCACTRL1。
3	SAMPLED_CCA	0	R	包括 CCA 的一个采样值。只要发出一个 SSAMPLECCA 或 STXONCCA 选通就更新该值。
2	LOCK_STATUS	0	R	当 PL 处于锁状态为 1，否则为 0。
1	TX_ACTIVE	0	R	状态信号，当 FFCTRL 处于发送状态之一时活跃。
0	RX_ACTIVE	0	R	状态信号，当 FFCTRL 处于接收状态之一时活跃。

FIFOPCTRL (0x6194) – FIFOP 阈值

位号码	名称	复位	R/W	描述
7	-	0	R0	读作 0
6:0	FIFOP_THR[6:0]	0x40	R/W	当产生 FIFOP 信号时使用阈值

FSMCTRL (0x6195) – FSM 选项

位号码	名称	复位	R/W	描述
7:2	-	0000 00	R0	读作 0
1	SLOTTED_ACK	0	R/W	控制传输确认帧的时序 0：请求确认的接收帧结束后 12 个符号周期之后发送确认帧。 1：请求确认的接收帧结束后，第一个退避时槽边界，多于 12 个符号周期之后发送确认帧。
0	RX2RX_TIME_OFF	1	R/W	定义接收帧结束后是否使用一个 12 符号的超时周期。

CCACTRL0 (0x6196) – CCA 阈值

位号码	名称	复位	R/W	描述
7:0	CCA_THR[7:0]	0xE0	R/W	空闲通道评估阈值，有符号的 2 的补数，是 RSSI 的补偿。 单位是 1dB，偏移大约是 76 dBm。当收到的信号低于该值 CCA 信号变为高。CCA 信号可从 CCA 引脚获得，在 FSMSTAT1 寄存器中。 注意为了避免 CCA 信号的错误行为，该值的设置必须低于 CCA_HYST – 128。 **注意**：复位值转换为大约–32 – 76 = –108 dBm 的输入水平，这远低于灵敏度限制。这意味着 CCA 信号从不指示一个空闲的通道。 该寄存器必须更新到 0xF8，转换到大约–8 – 76 = –84 dBm 的一个输出水平。

CCACTRL1 (0x6197) – 其他 CCA 选项

位号码	名称	复位	R/W	描述
7:5	-	000	R0	读作 0
4:3	CCA_MODE[1:0]	11	R/W	00：CCA 总是设置为 1 01：当 RSSI < CCA_THR – CCA_HYST，CCA=1；当 RSSI> CCA_THR，CCA=0 10：当没有接收帧，CCA=1，否则 CCA=0 11：当 RSSI < CCA_THR – CCA_HYST 且没有接收帧，CCA=1；当 RSSI> CCA_THR 或正在接收帧，CCA=0
2:0	CCA_HYST[2:0]	010	R/W	设置 CCA 滞后的级别。无符号数，以 dB 给定

RSSI (0x6198) – RSSI 状态寄存器

位号码	名称	复位	R/W	描述
7:0	RSSI_VAL[7:0]	0x80	R	RSSI 评估对数的规模，有符号数，2 的补数。 单位是 1dB。为了转换寄存器值到可在数据手册中找到的实际 RSSI 值，使用偏移。RSSI 值是 8 个符号周期的平均数。在首次读 RSSI_VAL 之前必须检查 RSSI_VALID 状态位。 复位值–128 也表示 RSSI 值无效。

RSSISTAT (0x6199) – RSSI 有效状态寄存器

位号码	名称	复位	R/W	描述
7:1	-	0000 000	R0	读作 0
0	RSSI_VALID	0	R	RSSI 值有效。进入 RX 后产生 8 个符号周期。

RXFIRST (0x619A) – RXFIFO 中的首字节

位号码	名称	复位	R/W	描述
7:0	DATA[7:0]	0x00	R	RXFIFO 中的首字节。 注意：读该寄存器不会修改 FIFO 的内容。

RXFIFOCNT (0x619B) – RXFIFO 中的字节数

位号码	名称	复位	R/W	描述
7:0	RXFIFOCNT[7:0]	0x00	R	RXFIFO 中的字节数。无符号整数。

TXFIFOCNT (0x619C) – TXFIFO 中的字节数

位号码	名称	复位	R/W	描述
7:0	TXFIFOCNT[7:0]	0x00	R	TXFIFO 中的字节数。无符号整数。

RXFIRST_PTR (0x619D) – RXFIFO 指针

位号码	名称	复位	R/W	描述
7	-	0	R	保留
6:0	RXFIRST_PTR[6:0]	000 0000	R	RXFIFO 首字节的 RAM 地址偏移

RXLAST_PTR (0x619E) – RXFIFO 指针

位号码	名称	复位	R/W	描述
7	-	0	R	保留
6:0	RXLAST_PTR[6:0]	000 0000	R	RXFIFO 最后+1 字节的 RAM 地址偏移

RXP1_PTR (0x619F) – RXFIFO 指针

位号码	名称	复位	R/W	描述
7:0	RXP1_PTR[7:0]	0x00	R	RXFIFO 首个帧的首字节的 RAM 地址偏移

TXFIRST_PTR (0x61A1) – TXFIFO 指针

位号码	名称	复位	R/W	描述
7:0	TXFIRST_PTR[7:0]	0x00	R	TXFIFO 首字节的 RAM 地址偏移

TXLAST_PTR (0x61A2) – TXFIFO 指针

位号码	名称	复位	R/W	描述
7:0	TXLAST_PTR[7:0]	0x00	R	TXFIFO 最后+1 字节的 RAM 地址偏移

MDMCTRL0 (0x61A8) – 控制调制解调器

位号码	名称	复位	R/W	描述
7:6	DEM_NUM_ZEROS[1:0]	10	R/W	当搜索同步时，设置在同步字之前必须检测到多少零符号。注意只有一个要求，相关值在 MDMCTRL1 寄存器设置的相关阀值上。 00：保留 01：1 个零符号 10：2 个零符号 11：3 个零符号
5	DEMOD_AVG_MODE	0	R/W	定义频率偏移平均过滤器的行为。 0：帧引导序列匹配后锁定平均水平。当搜索下一个帧重启频率偏移校准。 1：连续更新平均水平。
4:1	PREAMBLE_LENGTH [3:0]	0010	R/W	在SFD之前，在TX模式下要发送的帧引导序列的字节数（两个零符号），编码步长是2。复位值是2，兼容IEEE 802.15.4. 0000：2 个前导零字节 0001：3 个前导零字节 0010：4 个前导零字节 … 1111：17 个前导零字节
0	TX_FILTER	1	R/W	定义使用 TX 过滤器的类型。IEEE802.15.4 标准定义了一般的 TX 过滤器。可以使用其他的过滤器以降低带外辐射。 0：一般的 TX 过滤器 1：使能其他的过滤器

MDMCTRL1 (0x61A9) – 控制调制解调器

位号码	名称	复位	R/W	描述
7:6	-	00	R0	读作零
5	CORR_THR_SFD	0	R/W	定义 SDF 检测的要求。 0：帧引导序列的零符号之一的相关值必须在相关阀值之上。 1：帧引导序列的零符号之一的相关值以及 SFD 的符号必须都在相关阀值之上。
4:0	CORR_THR[4:0]	0x14	R/W	调制解调器相关阀值，在搜索 SFD 之前要求。 阀值调整接收器如何与来自无线电的数据同步。如果阀值设置的过低，同步会很容易地被发现噪音。如果设置的过高，灵敏度会下降，但是同步不太可能被发现噪音。 结合 DEM_NUM_ZEROS，可以调整系统，这样灵敏度高且发现的同步噪音少。

FREQEST (0x61AA) – RF 频率偏移估计

位号码	名称	复位	R/W	描述
7:6	FREQEST[7:0]	0x00	R	有符号 2 的补数。包括载波和接收器 LO 之间频率偏移的评估。偏移频率是 FREQEST ´ 7800 Hz。DEM_AVG_MODE 控制何时更新这一评估。如果 DEM_AVG_MODE = 0，直到找到同步才更新。然后频率偏移评估被冻结，直到接收帧结束。如果 DEM_AVG_MODE = 1,，只要调制解调器使能就更新。要计算正确值，必须使用一个偏移（(FREQEST_offset)，它可在数据手册中找到。实际 FREQEST 值=FREQEST - FREQEST_offset。

RXCTRL (0x61AB) – 调整接收部分

位号码	名称	复位	R/W	描述
7:6	-	00	R0	保留。读作 0
5:4	GBIAS_LNA2_REF[1:0]	11	R/W	调整前端 LNA2/混合器 PTAT 的电流输出（从 M=3 到 M=6），默认：M=5。
3:2	GBIAS_LNA_REF[1:0]	11	R/W	调整前端 LNA PTAT 的电流输出（从 M=3 到 M=6），默认：M=5。
1:0	MIX_CURRENT[1:0]	11	R/W	控制接收器混合器输出电流。电流随着设置的增加而增加。

FSCTRL (0x61AC)–调整频率合成器

位号码	名称	复位	R/W	描述
7:6	PRE_CURRENT [1:0]	01	R0	分频器电流设置
5:4	LODIV_BUF_CURRENT_TX [1:0]	01	R/W	调整混合器和 PA 缓冲器的电流。当 TX_ACTIVE = 1 使用。
3:2	LODIV_BUF_CURRENT_RX [1:0]	10	R/W	调整混合器和 PA 缓冲器的电流。当 TX_ACTIVE = 0 使用。
1:0	LODIV_CURRENT [1:0]	10	R/W	调整分频器电流，除了混合器和 PA 缓冲器。。

FSCAL0 (0x61AD) – 调整频率校准

位号码	名称	复位	R/W	描述
7	VCO_CURR_COMP_EN_OV	0	R0	设置 VCO 的电流比较器。该信号与来自校准模块的信号相或。
6	CHP_DISABLE	0	R/W	通过屏蔽来自相位探测器的向上和向下脉冲，设置它手动禁用电荷泵。
5:2	CHP_CURRENT[3:0]	1001	R*/W	数字位向量定义电荷泵在指数级上的输出电流。 如果 ffc_bw_boost = 0，读的值是存储在 CHP_CURRENT 中的值。 否则如果（ffc_bw_boost = 1），读的值是 CHP_CURRENT + 4。 如果另外的原因导致溢出，信号饱和。
1:0	BW_BOOST_MODE[1:0]	00	R/W	控制信号，定义合成器增强模式 00：无 BW_boosting 01：BW_boost 校准期间为高，设置为~30 μs。 10：BW_boost 总是开启/为高。 11：保留

FSCAL1 (0x61AE) – 调整频率校准

位号码	名称	复位	R/W	描述
7:2	-	001010	R/W0	保留
1:0	VCO_CURR[1:0]	01	R/W	定义 VCO 内核的电流。设置校准电流和 VCO 电流之间的乘数。对于使用的最佳值，见 19.15.1 节的表 19-6。

FSCAL2 (0x61AF) –调整频率校准

位号码	名称	复位	R/W	描述
7	-	0	R0	保留。读作 0
6	VCO_CAPARR_OE	0	R/W	以来自 VCO_CAPARR[5:0]的值覆盖校准结果
5:0	VCO_CAPARR[5:0]	10 0000	R*/W	VCO 电容数组设置。在校准期间编程。当 VCO_CAPARR_OE = 1 覆盖值。

FSCAL3 (0x61B0) – 调整频率校准

位号码	名称	复位	R/W	描述
7	-	0	R0	保留。读作 0
6	VCO_DAC_EN_OV	0	R/W	如果是 1 使能 VCO DAC
5:2	VCO_VC_DAC [3:0]	1010	R/W	位向量，编程 VC DAC 变容二极管的控制电压。
1:0	VCO_CAPARR_CAL_CTRL[1:0]	10	R/W	为校准的 cap_array 部分校准精确设置 00：80 XOSC 周期 01：100 XOSC 周期 10：125 XOSC 周期 11：250 XOSC 周期

AGCCTRL0 (0x61B1) – AGC 动态范围控制

位号码	名称	复位	R/W	描述
7	-	0	R0	保留。读作 0
6	AGC_DR_XTND_EN	1	R/W	0：AGC 不执行 AAF 衰减的调整。 1: AGC 调整 AAF 的 GAIN，以从接收器获得另外的动态范围 。
5:0	AGC_DR_XTND_THR[5:0]	01 1111	R/W	如果 AGC 参考值和实际值（以 dB 为单位）之间的测量误差大于该阀值，在前端使能另外的衰减。该阀值必须设置为高于 0x0C。 该功能由 AGC_DR_XTND_EN 使能。

AGCCTRL1 (0x61B2) – AGC 参考水平

位号码	名称	复位	R/W	描述
7:6	-	00	R0	保留。读作 0
5:0	AGC_REF[5:0]	01 0001	R/W	AGC 控制循环的目标值，步长是 1 –dB。使用的最佳值见 19.15.1 节的表 19-6。

AGCCTRL2 (0x61B3) – AGC 增益覆盖

位号码	名称	复位	R/W	描述
7:6	LNA1_CURRENT[1:0]	00	R*/W	覆盖 LNA 1 的值。仅当 LNA_CURRENT_OE = 1 时使用。读时该寄存器返回当前使用的增益设置。 00：0-dB 增益（参考水平） 01：3-dB 增益 10：保留 11：6-dB 增益
5:3	LNA2_CURRENT[2:0]	000	R*/W	覆盖 LNA 2 的值。仅当 LNA_CURRENT_OE = 1 时使用。读时该寄存器返回当前使用的增益设置。 000：0-dB增益（参考水平） 001：3-dB增益 010：6-dB增益 011：9-dB增益 100：12-dB 增益 101：15-dB 增益 110：18-dB 增益 111：21-dB 增益
2:1	LNA3_CURRENT[1:0]	00	R*/W	覆盖 LNA 3 的值。仅当 LNA_CURRENT_OE = 1 时使用。读时该寄存器返回当前使用的增益设置。 00：0-dB增益（参考水平） 01：3-dB增益 10：6-dB增益 11：9-dB 增益
0	LNA_CURRENT_OE	0	R/W	以存储在 RFR 中的值覆盖 AGC LNA 当前设置。

AGCCTRL3 (0x61B4) – AGC 控制

位号码	名称	复位	R/W	描述
7	-	0	R0	保留。读作0 11：6-dB 增益
6:5	AGC_SETTLE_WAIT[1:0]	01	R/W	增益变化后AGC等待模拟增益解决的时间。在此期间，AGC 的能量检测暂停。 00：15个周期 01：20个周期 10：25个周期 11：30个周期
4:3	AGC_WIN_SIZE[1:0]	01	R/W	用于 AGC 积累和转储功能的窗口大小。 00：16个样本 01：32个样本 10：64个样本 11：128 个样本
2:1	AAF_RP[1:0]	11	R*/W	当 AAF_RP_OE = 1 覆盖 AGC 控制信号。读时返回使用的信号到 AAF。 00：AAF 9-dB 的衰减 01：AAF 6-dB 的衰减 10：AAF 3-dB 的衰减 11：AAF 0-dB 的衰减（参考水平）
0	AAF_RP_OE	0	R/W	以存储在 AAF_RP 中的值覆盖 AGC AAF 控制信号。

ADCTEST0 (0x61B5) – ADC 调整

位号码	名称	复位	R/W	描述
7:6	ADC_VREF_ADJ[1:0]	00	R/W	为测试/调试量化测试阀值控制
5:4	ADC_QUANT_ADJ[1:0]	01	R/W	为测试/调试量化测试阀值控制
3:1	ADC_GM_ADJ[2:0]	000	R/W	用于测试/调试的Gm控制
0	ADC_DAC2_EN	0	R/W	为了增强ADC稳定性使能DAC2

ADCTEST1 (0x61B6) – ADC 调整

位号码	名称	复位	R/W	描述
7:4	ADC_TEST_CTRL[3:0]	0000	R/W	ADC 测试模式选择器
3:2	ADC_C2_ADJ[1:0]	11	R/W	用于调整 ADC 的电容值
1:0	ADC_C3_ADJ[1:0]	10	R/W	用于调整 ADC 的电容值

ADCTEST2 (0x61B7) – ADC 调整

位号码	名称	复位	R/W	描述
7	-	0	R0	保留。读作 0
6:5	ADC_TEST_MODE			测试模式使能 ADC 的数据从调制解调器输出。使能时，原始 ADC 数据在 GPIO 引脚上输出。 00：测试模式禁用 01：来自 I 和 Q ADC 的数据都被输出，数据率 76 MHz 10：来自 I ADC 的数据被输出。两个和两个 ADC 样本一组，数据率 38 MHz 11：来自 Q ADC 的数据被输出。两个和两个 ADC 样本一组，数据率 38 MHz
4:3	AAF_RS[1:0]	00	R/W	控制 AAF 的串联电阻
2:1	ADC_FF_ADJ[1:0]	01	R/W	调整前馈
0	ADC_DAC_ROT	1	R/W	控制 DAC DWA 机制 0：DWA（加扰）禁用 1：DWA 使能

MDMTEST0 (0x61B8) – 调制解调器的测试寄存器

位号码	名称	复位	R/W	描述
7:4	TX_TONE[3:0]	0111	R/W	通过从正弦表挑选样本，可以发送一个基带音，样本之间有一个可控制的相位步长。步长大小 TX_TONE 由控制。如果 MDMTEST1.MOD_IF 是 0，基带音和调制数据叠加，有效给出了 IF 调制。如果 MDMTEST1.MOD_IF 是 1，只发送基带音。 0000：–6 MHz 0001： –4 MHz 0010：–3 MHz 0011：–2 MHz 0100：–1 MHz 0101：–500 kHz 0110：–4 kHz 0111：0 1000：4 kHz 1001：500 kHz 1010：1 MHz 1011：2 MHz 1100：3 MHz 1101：4 MHz 1110：6 MHz 其他：保留
3:2	DC_WIN_SIZE[1:0]	01	R/W	控制 dc 删除中使用的累积转储过滤器，每次转储之间累加的采样数量。 00：32 个样本 01：64 个样本 10：128 个样本 11：256 个样本
1:0	DC_BLOCK_MODE[1:0]	01	R/W	选择操作模式： 00：到 dc 拦截器的输入信号不经过任何尝试删除 dc 传递到输出。 01：使能 dc 取消。一般操作 10：当发现同步冻结评估 dc。当搜索下一个帧再次启动评估 dc。 11：保留

MDMTEST1 (0x61B9) – 调制解调器的测试寄存器

位号码	名称	复位	R/W	描述
7:5	-	000	R0	保留。读作 0
4	MOD_IF	0	R/W	0：在任何 MDMTEST0.TX_TONE 设置的 IF 上执行调制。 1：发送一个基带音，频率由 MDMTEST0.TX_TONE 设置。
3	RAMP_AMP	1	R/W	1：启动和完成期间使能 DAC 输出幅度的慢加速。 0：禁用 DAC 输出幅度的慢加速
2	RFC_SNIFF_EN	0	R/W	0：数据包分析器模块禁用 1：数据包分析器模块使能。收到的和发送的数据可以在 GPIO 引脚上观测到。
1	MODULATION_MODE	0	R/W	为 RX/TX 设置两个 RF 调制模式之一 0：IEEE 802.15.4 兼容模式 1：返相，不兼容 IEEE
0	RESERVED	0	R/W	保留。不写。

DACTEST0 (0x61BA) – DAC 覆盖值

位号码	名称	复位	R/W	描述
7	-	0	R0	保留。读作 0
6:0	DAC_Q_O[6:0]	000 0000	R/W	当 DAC_SRC = 001 ，Q 分支 DAC 覆盖值 如果 DAC_SRC 设置为 ADC 数据、CORDIC 数据、通道已过滤的数据，那么 DAC_Q_O 控制实际复用到 DAC 的有问题的字的部分，如下所述。 00 0110 >=位6:0 00 0111 >=位7:1 00 1000 >=位 8:2 等等 如果选择了一个无效设置，那么 DAC 只输出零（最小值）。

DACTEST1 (0x61BB) – DAC 覆盖值

位号码	名称	复位	R/W	描述
7	-	0	R0	保留。读作 0
6:0	DAC_I_O[6:0]	000 0000	R/W	当 DAC_SRC = 001 ，I 分支 DAC 覆盖值 如果 DAC_SRC 设置为 ADC 数据、CORDIC 数据、通道已过滤的数据，那么 DAC_I_O 控制实际复用到 DAC 的有问题的字的部分，如下所述。 00 0110 >=位6:0 00 0111 >=位7:1 00 1000 >=位 8:2 等等 如果选择了一个无效设置，那么 DAC 只输出零（最小值）。

DACTEST2 (0x61BC) – DAC 测试设置

位号码	名称	复位	R/W	描述
7:3	-	0010 1	R0	保留。
2:0	DAC_SRC[2:0]	000	R/W	DAC_SRC 根据以下选择 TX DAC 数据源： 000：一般操作（来自调制器）。 001：抽取后的 ADC 数据，大小由 DAC_I_O 和 DAC_Q_O 控制 011：抽取后的 I/Q，通道和 dc 过滤，大小由 DAC_I_O 和 DAC_Q_O 控制 100：Cordic 值输出，前端增益输出，大小由 DAC_I_O 和 DAC_Q_O 控制 101：RSSI 在 I DAC 上输出 111：保留

ATEST (0x61BD) – 模拟测试控制

位号码	名称	复位	R/W	描述
7:6	-	00	R0	保留。读作 0
5:0	ATEST_CTRL[5:0]	00 0000	R/W	控制模拟测试模式： 00 0001：使能温度传感器（也可见 12.2.10 节 TR0 寄存器描述）。 其他值保留。

RFRND (0x61A7) – 随机数据

位号码	名称	复位	R/W	描述
7:2	-	0000 00	R0	保留。读作 0
1	QRND	0	R/W	接收器 Q 通道的随机位
0	IRND	0	R/W	接收器 I 通道的随机位

PTEST0 (0x61BE) – 覆盖掉电寄存器

位号码	名称	复位	R/W	描述
7	PRE_PD	0	R/W	分频器掉电信号。当 PD_OVERRIDE = 1
6	CHP_PD	0	R/W	电荷泵掉电信号。当 PD_OVERRIDE = 1
5	ADC_PD	0	R/W	模拟数字转换器掉电信号。当 PD_OVERRIDE = 1
4	DAC_PD	0	R/W	数字模拟转换器掉电信号。当 PD_OVERRIDE = 1
3:2	LNA_PD[1:0]	0	R/W	低噪音放大器掉电信号。定义 LNA/混合器 PD 模式。 00：上电 01：LNA 关闭，混合器/稳压器开启 10：LNA/混合器关闭，稳压器开启 11：PD 当 PD_OVERRIDE = 1
1	TXMIX_PD	0	R/W	发送混合器掉电信号。当 PD_OVERRIDE = 1
0	AAF_PD	0	R/W	抗混叠过滤器掉电信号。当 PD_OVERRIDE = 1

PTEST1 (0x61BF) – 覆盖掉电寄存器

位号码	名称	复位	R/W	描述
7:4	-	0000	R0	保留。读作 0
3	PD_OVERRIDE	0	R/W	各种模块的覆盖使能/禁用。仅用于调试和测试。不能覆盖硬编码的 BIAS_PD[1:0]。
2	PA_PD	0	R/W	功率放大器掉电信号。当 PD_OVERRIDE = 1
1	VCO_PD	0	R/W	电压控制震荡器掉电信号。当 PD_OVERRIDE = 1
0	LODIV_PD	0	R/W	LO 掉电信号。当 PD_OVERRIDE = 1

RFC_OBS_CTRL0 (0x61EB) – RF 观测复用控制

位号码	名称	复位	R/W	描述
7:4	-	0000	R0	保留。读作 0
6	RFC_OBS_POL0	0	R/W	RFC_OBS_MUX0 选择的信号与该位相异或。
5:0	RFC_OBS_MUX0	00 0000	R/W	控制来自 RF 内核的哪个可观测的信号要被输出到 rfc_obs_sigs(0)。 00 0000：0-常量值 00 0001：1-常量值 00 1000：rfc_sniff_data-来自数据包分析器的数据。sniff_clk 上升沿的样本数据。 00 1001：rfc_sniff_clk –数据包分析器数据的 250kHz 时钟。 00 1100：rssi_valid –自从 RX 启动当至少更新一次 RSSI 值，引脚为高。当离开 RX，清除。 00 1101：demod_cca –空闲通道评估。如何配置该信号的行为详见 FSMSTAT1 寄存器。 00 1110：sampled_cca –调制解调器 CCA 位的采样版本。只要发出一个 SSAMPLECCA 或 STXONCCA 选通就更新该值。 00 1111：sfd_sync –当收到货发送一个 SFD 引脚为高。分别离开 RX/TX 时清除。不要和 SFD 异常混淆。 01 0000：tx_active –表示 FFCTRL 处于 TX 状态之一。活跃为高。注意：该信号可能有故障，因为它没有输出触发器，基于寄存器 FFCTRL FSM 的当前状态。 01 0001：rx_active –表示 FFCTRL 处于 RX 状态之一。活跃为高。注意：该信号可能有故障，因为它没有输出触发器，基于寄存器 FFCTRL FSM 的当前状态。 01 0010：ffctrl_fifo –当 RXFIFO 中有一个或多个字节引脚为高。RXFIFO 溢出期间为低。 01 0011：ffctrl_fifop –当 RXFIFO 中的字节数超出编程的阀值，或 RXFIFO 至少有一个完整的帧，引脚为高。RXFIFO 溢出期间也为高。不要和 FIFO 异常混淆。 01 0100：packet_done –收到一个完整的帧。即收到长度域设置的字节数。 01 0110：rfc_xor_rand_i_q –I 和 Q 随机输出之间的异或。在 8MHz 更新。 01 0111：rfc_rand_q –随机数从接收器的 Q 通道输出。在 8MHz 更新。 01 1000：rfc_rand_i –随机数从接收器的 I 通道输出。在 8MHz 更新。 01 1001：lock_status –当 PLL 锁定是 1，否则是 0。 10 1000：pa_pd –功率放大器掉电信号 10 1010：lna_pd – LNA 掉电信号 其他：保留

RFC_OBS_CTRL1 (0x61EC) – RF 观测复用控制

位号码	名称	复位	R/W	描述
7	-	0	R0	保留。读作 0
6	RFC_OBS_POL1	0	R/W	RFC_OBS_MUX1 选择的信号与该位相异或。
5:0	RFC_OBS_MUX1	00 0000	R/W	控制来自 RF 内核的哪个可观测的信号要被输出到 rfc_obs_sigs(1)。详见 RFC_OBS_CTRL0 的描述。

RFC_OBS_CTRL2 (0x61ED) – RF 观测复用控制

位号码	名称	复位	R/W	描述
7	-	0	R0	保留。读作 0
6	RFC_OBS_POL2	0	R/W	RFC_OBS_MUX2 选择的信号与该位相异或。
5:0	RFC_OBS_MUX2	00 0000	R/W	控制来自 RF 内核的哪个可观测的信号要被输出到 rfc_obs_sigs(2)。详见 RFC_OBS_CTRL0 的描述。

TXFILTCFG (0x61FA) – TX 过滤器配置

位号码	名称	复位	R/W	描述
7:4	-	0	R0	保留
3:0	FC	0xF	R/W	设置 TX 抗混叠过滤器以获得合适的带宽。降低杂散发射接近信号使用的最佳值见 19.15.1 节的表 19-6。

稳压器

数字稳压器用于为数字内核供电。该稳压器的输出可在 DCOUPL 引脚上获得，并要求有去耦电容才能正常工作（见 CC2530 参考设计的例子）。

稳压器在供电模式 PM2 和 PM3 下禁用（见第 4 章）。当稳压器禁用，当不规范的 2V 到 3.6V 电压出现，寄存器和 RAM 的内容保留。

注意：稳压器不能用于为外部电路供电。

可用的软件

本章介绍了与 CC253x 系列相关的各种可用的软件解决方案。它们可从 TI 网站 www.ti.com/lprf 上免费下载，用于 TI LPRF 设备。

标题 页

21.1 用于评估的 SmartRF™软件 (www.ti.com/smartrfstudio)

德州仪器的 SmartRF Studio 可用于评估无线电的性能和功能，对探索和了解 RF-IC 产品很有帮助。该软件帮助无线电系统的设计人员在设计过程的早期阶段轻松地评估 RF-IC。它对于产生配置数据和找到最佳外部组件值特别有用。

SmartRF Studio 软件运行在 Microsoft™ Windows™ 95/98 和 Microsoft Windows NT/2000/XP。

SmartRF Studio 软件可以从德州仪器网页 www.ti.com/smartrfstudio (http://www.ti.com/litv/zip/swrc046m)上下载。

功能：

- 链路测试。在节点之间发送和接收数据包
- 数据包错误率（PER）测试
- 通过 USB 端口或并行端口与评估板通信
- 一个计算机支持多达 8 个 USB 设备
- 具有首选寄存器设置的普通视图
- 可以读和写各个寄存器的寄存器视图。每个寄存器给出详细信息
- 保存/打开文件的配置数据
- 保存/加载文件的寄存器设置
- 导出/导入文本文件的寄存器值
- 导出寄存器设置到一个兼容 C 的软件结构

21.2 RemoTI™网络协议(www.ti.com/remoti)

大多数现有的远程控制使用红外技术来和消费型电子设备进行命令通信。但是，射频（RF）远程控制基于双向 RF 通信，使能了非视距操作，提供了更先进的功能。

用于消费型电子的 ZigBee 射频（RF4CE）是 ZigBee 联盟和 RF4CE 协会(http://www.zigbee.org/rf4ce)最新协商的结果，专门用于部署广泛范围的远程控制音频/视频消费型电子产品，比如 TV 和机顶盒。ZigBee RF4CE 向您承诺：

- 更丰富的通信和更高的可靠性
- 增强的功能和灵活性
- 兼容性
- 无视线障碍

RemoTI 网络协议是德州仪器 ZigBee RF4CE 标准的执行。它是一个完整的解决方案，为 TI 低功耗 RF 产品系列提供了硬件和软件支持。有了 RemoTI 网络协议我们提供：

- 业界领先的 RF4CE 兼容栈集成了可互操作的 CERC 规范支持、一个简单的 API、易于理解的实例应用程序代码、完整的开发套件和参考设计，以及更多。
- 在我们同类中最佳的 IEEE 802.15.4 兼容片上系统 CC2530 上运行，具有优异的 RF 共存和 RF 性能。四种灵活的供电模式包括最低的电流消耗掉电模式，用于长电池寿命、低占空比的应用。
- 世界各地的广泛支持和工具能够确保简单、快速地开发基于 ZigBee RF4CE 的应用，而且可以以最低的成本完成。
- RemoTI 网络协议是一个黄金单元平台，即为了兼容标准，它用于测试 ZigBee RF4CE 标准的其他实现。

关于 TI 的 RemoTI 网络协议的更多信息，见德州仪器 RemoTI 网络协议网站 www.ti.com/remoti（查询信息、下载等等）。

21.3 SimpliciTI™网络协议(www.ti.com/simpliciti)

SimpliciTI 网络协议是一个低功耗 RF 协议（用于低于 1 GHz、2.4 GHz 和 IEEE 802.15.4RF IC），针对简单、小型的 RF 网络。这一开放源码的软件是构建一个网络（具有使用了 TI 低功耗 RF 片上系统（SoC）的电池供电的设备）的良好开端。SimpliciTI 网络协议专门用于在一些 TI RF 平台上简单实现和部署的开箱即用的应用。它提供了一些实例应用程序。

主要应用

- 报警和安全：占位传感器、照明传感器、一氧化碳传感器、玻璃破损探测器
- 烟雾探测器
- 自动抄表：煤气表、水表、电子表
- 动态 RFID 应用

主要功能

- 低功耗：一个 TI 专有的低功耗网络协议
- 灵活：
  -直接设备到设备通信
  -简单星型结构，具有存储和转发到终端设备的接入点
  -范围扩展，增加了四跳的范围
- 简单：使用仅有五个命令的 API
- 低数据率和低占空比
- 易于使用

关于 TI 的 SimpliciTI 网络协议的更多信息，见德州仪器 SimpliciTI 网络协议网站 www.ti.com/SimpliciTI（查询信息、下载等等）。

21.4 TIMAC 软件(www.ti.com/timac)

TIMAC 软件是一个 IEEE 802.15.4 媒体访问控制软件栈，用于 TI 的 IEEE 802.15.4 收发器和片上系统。

当以下情况可以使用 TIMAC：

- 需要一个无线点到点或点到多点解决方案，即多个传感器直接报告给一个主机
- 需要一个标准化的无线协议
- 有电池供电和/或主电源供电的节点
- 需要支持确认和重传
- 要求低数据率（大约 100-kbps 的有效数据率）

功能

- 支持 IEEE 802.15.4 标准
- 支持信标使能和非信标使能的系统
- 多个平台
- 便于部署应用程序

TIMAC 软件栈经过认证符合 IEEE 802.15.4 标准。免费为 TIMAC 软件分配目标代码。使用 TIMAC 软件不需要许可费。

关于 TIMAC 软件的更多信息，见德州仪器 TIMAC 网络协议网站 www.ti.com/timac（查询信息、下载等等）。

21.5 Z-Stack™软件(www.ti.com/z-stack)

Z-Stack 软件是 TI 的 ZigBee 兼容协议栈，用于不断增加的 IEEE 802.15.4 产品和平台系列。Z-Stack 软件栈符合 ZigBee-2006 和 ZigBee-2007 规范，支持 ZigBee 和 ZigBee PRO 功能集。Z-Stack 软件包括两个 ZigBee 应用规范的执行——智能能源和家庭自动化。其他应用规范可以容易地由用户实现。

Z-Stack 软件主要特征包括：

- 完全兼容 ZigBee 和 ZigBee PRO 功能集
- 广泛的实例应用程序，包括支持 ZigBee 智能能源和 ZigBee 家庭自动化规范
- 支持无线下载和串行引导加载
- 可以和 RF 前端（CC2590 和 CC2591）一起使用，分别支持 10dBm 和 20dBm 的输出功率，且提高了接收灵敏度

Z-Stack 软件被 ZigBee 联盟评为黄金单元，ZigBee 和 ZigBee PRO 规范，被全世界的 ZigBee 开发人员广泛使用。

Z-Stack 软件特别适合于：

- 智能能源（AMI）
- 家庭自动化
- 商业楼宇自动化
- 医疗、辅助生活或个人健康和医院护理
- 监测和控制应用
- 无线传感器网络
- 警报和安全
- 资产跟踪
- 要求有互操作性的应用程序

关于 Z-Stack 软件的更多信息，见德州仪器 Z-Stack 网络协议网站 www.ti.com/Z-Stack（查询信息、下载等等）。

缩写

本用户指南使用的缩写：

AAF	抗混叠过滤器
ADC	模拟数字转换器
AES	高级加密标准
AGC	自动增益控制
ARIB	工业和商业无线电协会
BCD	二进制编码的十进制
BER	误码率
BOD	布朗输出探测器
BOM	材料清单
CBC	密码块链接
CBC-MAC	密码块链接信息验证代码
CCA	空闲信道评估
CCM	计数器模式+ CBC-MAC
CFB	密码反馈
CFR	联邦法规
CMRR	共模抑制比
CPU	中央处理器单元
CRC	循环冗余校验
CSMA-CA	载波监听多路访问/冲突防止
CSP	CSMA/CA 选通处理器
CTR	计数器模式（加密）
CW	连续波
DAC	数字/模拟转换器
DC	直流
DMA	直接存储器存取
DSM	Delta Sigma 调制器
DSSS	直接序列扩频
ECB	电子密码本（加密）
EM	评估模块
ENOB	有效位数
ETSI	欧洲电信标准协会
EVM	误差矢量幅度
FCC	联邦通信委员会
FCF	帧控制域
FCS	帧校验序列
FFCTRL	FIFO 和帧控制
FIFO	先进先出
GPIO	通用输入输出

HF	高频
HSSD	高速串行数据
I/O	输入/输出
I/Q	同相/正交相
IEEE	电气和电子工程师协会
IF	中频
IOC	I/O 控制器
IRQ	中断请求
IR	红外
ISM	工业、科学、医疗
ITU-T	国际电信联盟-电信标准局
IV	初始化向量
KB	1024 字节
kbps	千比特每秒
LFSR	线性反馈移位寄存器
LNA	低噪声放大器
LO	本机振荡器
LQI	链路质量指示
LSB	最低有效位/字节
MAC	媒体访问控制
MAC	信息验证代码
MCU	微控制器单元
MFR	MAC 尾
MHR	MAC 头
MIC	信息完整性代码
MISO	主机输入从机输出
MOSI	主机输出从机输入
MPDU	MAC 层协议数据单元
MSB	最高有效字节
MSDU	MAC 层服务数据单元
MUX	复用器
NA	不可用
NC	未连接
OFB	输出反馈（加密）
O-QPSK	偏移-正交相移键控
PA	功率放大器
PC	程序计数器
PCB	印刷电路板
PER	封装错误率
PHR	PHY 首部
PHY	物理层
PLL	锁相环
PM1,PM2, PM3	供电模式 1、2 和 3
PMC	电源管理控制器
POR	上电复位
PSDU	PHY 服务数据单元
PWM	脉宽调制器
RAM	随机存储器

RBW	分辨率带宽
RC	电阻-电容器
RCOSC	RC 振荡器
RF	射频
RSSI	接收信号强度指示器
RTC	实时时钟
RX	接收
SCK	串行时钟
SFD	帧首定界符
SFR	特殊功能寄存器
SHR	同步首部
SINAD	信号-噪声及失真比
SPI	串行外设接口
SRAM	静态随机存储器
ST	睡眠计时器
T/R	磁带和卷轴
T/R	发送/接收
THD	总谐波失真
TI	德州仪器
TX	发送
UART	通用异步收发器
USART	通用同步/异步收发器
VCO	电压可控振荡器
VGA	可变增益放大器
WDT	看门狗
XOSC	晶振

其他信息

德州仪器为基于专有和标准的无线应用提供了经济、低功耗的 RF 解决方案的广泛选择，用于工业和消费型应用。我们的选择包括 RF 收发器、RF 发送器、RF 前端和片上系统，以及各种软件解决方案，用于低于 1 和 2.4 GHz 的频带。

另外，德州仪器提供了间接支持的广泛选择，比如开发工具、技术文档、参考设计、应用知识、客户支持、第三方和大学课程。

低功耗 RF E2E 网络社区为您提供了技术支持论坛、视频和博客，并且有机会与来自世界各地的同行工程师交流。

凭借产品解决方案广泛的选择，最终应用的可能性，以及各种技术支持，德州仪器提供了最广泛的低功耗 RF 产品系列。我们使 RF 变得容易。

以下小节介绍了在哪可以找到更多信息。

标题 **页**

B.1 德州仪器低功耗 RF 网站

德州仪器的低功耗 RF 网站有我们全部最新的产品、应用程序和设计说明、FAQ 部分、新闻和活动更新，以及更多信息。请访问 www.ti.com/lprf。

B.2 低功耗 RF 网络社区

- 论坛、视频和博客
- RF 设计帮助
- E2E 互动——发帖和阅读其他用户的问题

加入我们 www.ti.com/lprf-forum。

B.3 德州仪器低功耗 RF 开发商网络

德州仪器已经推出一个低功耗 RF 开发伙伴的广泛的网络，以帮助客户加速自己的应用程序开发。网络包括推荐的公司、RF 咨询和独立的设计机构，提供一系列的硬件模块产品和设计服务，包括：

- RF 电路、低功耗 RF 和 ZigBee 设计服务
- 低功耗 RF 和 ZigBee 模块解决方案和开发工具
- RF 认证服务和 RF 电路制造

需要模块帮助、工程服务或开发工具？

搜索低功耗 RF 开发商网络工具，找到合适的合作伙伴！www.ti.com/lprfnetwork

B.4 低功耗 RF 电子简讯

低功耗 RF 电子简讯能使您随时掌握与 TI 低功耗 RF 产品相关的最新产品、新闻发布、开发商新闻以及其他新闻和活动。低功耗 RF 电子简讯的文章包含链接以获得更多网络信息。

今天就注册 www.ti.com/lprfnewsletter。

参考书目

参考书目和其他有用的资料：

1．IEEE std. 802.15.4－2006：低速率无线个人区域网络（LR-WPAN）的无线媒体访问控制（MAC）和物理层（PHY）规范。
http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf
2. CC2530 数据手册（SWRS081）

重要声明

德州仪器公司及其子公司有权在不经过通知的情况下，随时对其产品和服务进行更正、修改、增强、改进或其它更改，并有权随时停止提供某种产品或服务。客户应在预定产品之前获得最新相关信息，并证实该信息是最新的、完整的。所有的产品的销售均遵循在订单确认时提供的 TI 销售条款和条件。

TI 保证其所售硬件产品的性能符合 TI 标准保修的适用规范。仅在保修期内，且 TI 认为必要时才会使用测试及其它质量控制技术。除非政府做了硬性规定，否则不必要测试每一种产品的各个参数。

TI 不承担应用帮助或客户产品设计的义务。客户应对其使用 TI 组件的产品和应用负责。为了使客户产品和应用的相关风险降至最低，客户应有足够的设计和操作安全措施。

TI 不对任何 TI 专利权、版权、屏蔽作品权或其它与使用了 TI 产品或服务的组合设备、机器、流程相关的 TI 知识产权中授予的直接或隐含权限做出任何保证或解释。由 TI 发布的关于第三方产品或服务的信息不能构成从 TI 获得使用这些产品或服务的许可、保证或认可。要使用这些信息可能需要获得第三方的关于专利或其它知识产权的许可，或是从 TI 获得关于 TI 专利或其它知识产权的许可。

对于 TI 的数据手册或参数表，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况下才允许复制。在复制信息的过程中对内容的篡改是非法的、欺诈性商业行为。TI 对此类篡改过的文件不承担任何责任。第三方的信息可能遵照另外的规则。

在转售 TI 的产品或服务时，如果对产品或服务参数有不同或夸大描述，则会失去相关 TI 产品或服务的明示或暗示授权，且这是违法的、欺诈性商业行为。TI 对任何此类虚假陈述不承担责任或法律义务。

TI 产品未被授权用于当 TI 产品出现问题时可能会引起严重的人身伤害或死亡的、有严格安全限制的应用（比如生活用品），除非政府执行了一项管理此种应用的协议。客户一旦购买就表示有足够的能力处理其应用的安全问题和可能产生的后果，认可并同意对关于其产品所有法定的、规范的、安全方面的要求负责，对在此类有严格安全限制的应用上使用任何 TI 产品负责，尽管关于应用的任何信息或支持都可能由 TI 提供。而且，因在这样有严格安全限制的应用上使用 TI 产品而产生的任何损失，购买者必须全价赔偿 TI 及其代理人。

除非 TI 产品由 TI 特别指定用于军事用途或“增强型塑料”，否则 TI 产品不适用于军事或航空的应用或环境。只有 TI 特别指定用于军事用途的产品才符合军事的具体标准。客户一旦购买就认可和同意了自行承担使用任何这种未被指定用于军事的 TI 产品的风险，并且对这种用法的所有合法的、规范的要求承担责任。

除非某种具体的 TI 产品由 TI 指定且符合 ISO/TS 16949 的要求，否则 TI 产品不适用于汽车的应用或环境。客户一旦购买就认可和同意如果在汽车应用中使用任何非指定产品，TI 对任何达不到此种要求的情况不负责任。

可以访问以下 URL 地址以获取有关其它 TI 产品和应用解决方案的信息：

产品		应用	
放大器	amplifier.ti.com	音频	www.ti.com/audio
数据转换器	dataconverter.ti.com	汽车	www.ti.com/automotive
DLP®产品	www.dlp.com	宽带	www.ti. com/broadband
DSP	dsp.ti.com	数字控制	www.ti.com/digitalcontrol
时钟计时	www.ti.com/clocks	医疗	www.ti.com/medical
接口	interface.ti.com	军事	www.ti.com/military
逻辑	logic.ti.com	光纤网络	www.ti.com/opticalnetwork
功率 Mgmt	power.ti.com	安全	www.ti.com/security
微控制器	microcontroller.ti.com	电话	www.ti.com/telephony
RFID	www.ti-rfid.com	视频与成像	www.ti.com/video
RF/IF 及 ZigBee 解决方案	www.ti.com/lprf	无线	www.ti.com/wireless

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas ,Texas 75265

版权声明