

armsky™
斯凯科技
<http://www.armsky.net>

CC253x 用户指南

目 录

序言.....	1
1 简介.....	2
1.1 概览.....	3
1.1.1 CPU 和存储器.....	4
1.1.2 时钟和电源管理.....	5
1.1.3 外部设备.....	5
1.1.4 无线.....	6
1.2 应用.....	6
2 8051CPU.....	8
2.1 8051 CPU 介绍.....	8
2.2 存储器.....	8
2.2.1 存储器映射.....	9
2.2.2 CPU 存储器空间.....	10
2.2.3 物理存储器.....	11
2.2.4 XDATA 存储器存取.....	16
2.2.5 存储器仲裁.....	16
2.3 CPU 寄存器.....	17
2.3.1 数据指针.....	17
2.3.2 寄存器 R0—R7.....	18
2.3.3 程序状态字.....	18
2.3.4 累加器.....	19
2.3.5 B 寄存器.....	19
2.3.6 堆栈指针.....	19
2.4 指令集摘要.....	20
2.5 中断.....	23
2.5.1 中断屏蔽.....	24
2.5.2 中断处理.....	27
2.5.3 中断优先级.....	30
3 调试接口.....	31
3.1 调试模式.....	32
3.2 调试通信.....	32
3.3 调试命令.....	34
3.4 锁定位.....	34
3.4.1 调试配置.....	35
3.4.2 调试状态.....	35
3.4.3 硬件断点.....	37
3.4.4 Flash 编程.....	38
3.5 调试接口和功耗模式.....	38
3.6 寄存器.....	38
4 电源管理和时钟.....	39

4.1 电源管理说明.....	40
4.1.1 主动和空闲模式.....	41
4.1.2 PM1.....	41
4.1.3 PM2.....	41
4.1.4 PM3.....	41
4.2 电源管理控制.....	41
4.3 电源管理寄存器.....	42
4.4 振荡器和时钟.....	44
4.4.1 振荡器.....	45
4.4.2 系统时钟.....	45
4.4.3 32kHz 晶振.....	45
4.4.4 振荡器和时钟寄存器.....	46
4.5 定时器 Tick 产生.....	48
4.6 数据保持.....	48
5 复位.....	49
5.1 上电复位和掉电检测.....	49
5.2 时钟丢失检测.....	49
6 Flash 控制器.....	50
6.1 Flash 存储器组织.....	51
6.2 Flash 写.....	51
6.2.1 Flash 写步骤.....	51
6.2.2 对一个字进行多次写.....	52
6.2.3 DMA Flash 写.....	53
6.2.4 CPU Flash 写.....	54
6.3 Flash 页擦除.....	54
6.3.1 从 Flash 存储器执行 Flash 擦除.....	55
6.4 Flash DMA 触发.....	55
6.5 Flash 控制器寄存器.....	55
7 I/O 口.....	57
7.1 未使用的 I/O 引脚.....	57
7.2 低 I/O 供电电压.....	57
7.3 通用 I/O.....	57
7.4 通用 I/O 中断.....	58
7.5 通用 I/O DMA.....	58
7.6 外部设备 I/O.....	59
7.6.1 定时器 1.....	59
7.6.2 定时器 3.....	60
7.6.3 定时器 4.....	60
7.6.4 USART0.....	60
7.6.5 USART1.....	61
7.6.6 ADC.....	61
7.7 调试接口.....	62
7.8 32kHz XOSC 输入.....	62

7.9 无线测试输出信号.....	62
7.10 掉电信号多路器 (PMUX)	62
7.11 I/O 寄存器.....	62
8 DMA 控制器.....	71
8.1 DMA 操作.....	71
8.2 DMA 配置参数.....	73
8.2.1 源地址.....	73
8.2.2 目标地址.....	73
8.2.3 传送长度.....	73
8.2.4 可变长度 (VLEN) 设置.....	74
8.2.5 触发事件.....	74
8.2.6 源地址和目标地址增量.....	75
8.2.7 DMA 传送模式.....	75
8.2.8 DMA 优先级.....	75
8.2.9 字节或字传送.....	75
8.2.10 中断屏蔽.....	75
8.2.11 M8 设置.....	76
8.3 DMA 配置安装.....	76
8.4 停止 DMA 传送.....	76
8.5 DMA 中断.....	76
8.6 DMA 配置数据结构.....	77
8.7 DMA 存储器存取.....	77
8.8 DMA 寄存器.....	79
9 定时器 1 (16 位定时器)	81
9.1 16 位计数器.....	82
9.2 定时器 1 操作.....	82
9.3 自由运行模式.....	83
9.4 模模式.....	83
9.5 正计数/倒计数模式.....	84
9.6 通道模式控制.....	84
9.7 输入捕获模式.....	84
9.8 输出比较模式.....	84
9.9 红外信号产生和学习.....	88
9.9.1 简介.....	88
9.9.2 调制码.....	89
9.9.3 非调制编码.....	90
9.9.4 学习.....	91
9.9.4.1 载波频率检测.....	91
9.9.4.2 解调编码学习.....	91
9.9.5 其它注意事项.....	91
9.10 定时器 1 中断.....	91
9.11 定时器 1 DMA 触发.....	92
9.12 定时器 1 寄存器.....	92

9.13 以数组访问定时器寄存器.....	98
10 定时器 3 和定时器 4 (8 位定时器)	99
10.1 8 位定时器计数器.....	99
10.2 定时器 3/定时器 4 模式控制.....	99
10.2.1 自由运行模式.....	99
10.2.2 倒计数模式.....	99
10.2.3 模模式.....	100
10.2.4 正计数/倒计数模式.....	100
10.3 通道模式控制.....	100
10.4 输入捕获模式.....	100
10.5 输出比较模式.....	100
10.6 定时器 3 和定时器 4 中断.....	101
10.7 定时器 3 和定时器 4 DMA 触发.....	101
10.8 定时器 3 和定时器 4 寄存器.....	102
11 睡眠定时器.....	106
11.1 概述.....	107
11.2 定时器比较.....	107
11.3 定时器捕获.....	107
11.4 睡眠定时器寄存器.....	108
12 ADC.....	110
12.1 ADC 简介.....	111
12.2 ADC 运行.....	111
12.2.1 ADC 输入.....	111
12.2.2 ADC 转换序列.....	112
12.2.3 单个 ADC 转换.....	112
12.2.4 ADC 运行模式.....	112
12.2.5 ADC 转换结果.....	113
12.2.6 ADC 基准电压.....	113
12.2.7 ADC 转换时间.....	113
12.2.8 ADC 中断.....	114
12.2.9 ADC DMA 触发.....	114
12.2.10 ADC 寄存器.....	114
13 随机数发生器.....	117
13.1 简介.....	117
13.2 随机数发生器运行.....	117
13.2.1 半随机序列生成.....	117
13.2.2 种子数的产生.....	117
13.2.3 CRC16.....	118
13.3 随机数发生器寄存器.....	118
14 AES 协处理器.....	119
14.1 AES 操作.....	119
14.2 密钥和初始化向量.....	119
14.3 填充输入数据.....	119

14.4 CPU 接口.....	119
14.5 操作模式.....	120
14.6 CBC—MAC.....	120
14.7 CCM 模式.....	120
14.8 在各个通信层次之间共享 AES 协处理器.....	122
14.9 AES 中断.....	122
14.10 AES DMA 触发.....	123
14.11 AES 寄存器.....	123
15 看门狗定时器.....	124
15.1 看门狗模式.....	124
15.2 定时器模式.....	124
15.3 看门狗定时器寄存器.....	125
16 USART.....	125
16.1 UART 模式.....	126
16.1.1 UART 发送.....	126
16.1.2 UART 接收.....	126
16.1.3 UART 硬件流控制.....	127
16.1.4 UART 特征格式.....	127
16.2 SPI 模式.....	127
16.2.1 SPI 主模式操作.....	128
16.2.2 SPI 从模式操作.....	128
16.3 SSN 从选择引脚.....	128
16.4 波特率发生器.....	129
16.5 清除 USART.....	129
16.6 USART 中断.....	130
16.7 USART DMA 触发.....	130
16.8 USART 寄存器.....	130
17 USB 控制器.....	135
17.1 USB 简介.....	135
17.2 USB 使能.....	136
17.3 48MHz USB PLL.....	136
17.4 USB 中断.....	136
17.5 端点 0.....	136
17.6 端点 0 中断.....	137
17.6.1 错误情况.....	137
17.6.2 SETUP 传输 (IDLE 状态)	137
17.6.3 IN 传输 (TX 状态)	138
17.6.4 OUT 传输 (RX 状态)	138
17.7 端点 1-5.....	139
17.7.1 FIFO 管理.....	139
17.7.2 双缓冲.....	140
17.7.3 FIFO 存取.....	140
17.7.4 端点 1-5 中断.....	141

17.7.5 批量/中断 IN 端点.....	141
17.7.6 同步 IN 端点.....	142
17.7.7 批量/中断 OUT 端点.....	142
17.7.8 同步 OUT 端点.....	142
17.8 DMA.....	143
17.9 USB 复位.....	143
17.10 挂起和恢复.....	143
17.11 远程唤醒.....	144
17.12 USB 寄存器.....	144
18 定时器 2 (MAC 定时器)	151
18.1 定时器操作.....	152
18.1.1 概述.....	152
18.1.2 正计数.....	152
18.1.3 定时器溢出.....	153
18.1.4 定时器 delta 增量.....	153
18.1.5 定时器比较.....	153
18.1.6 溢出计数.....	153
18.1.7 溢出计数更新.....	154
18.1.8 溢出计数溢出.....	154
18.1.9 溢出计数比较.....	154
18.1.10 捕获输入.....	154
18.2 中断.....	154
18.3 事件输出 (DMA 触发和 CSP 事件)	155
18.4 定时器开始/停止同步.....	155
18.4.1 概述.....	155
18.4.2 定时器同步停止.....	155
18.4.3 定时器同步开始.....	156
18.5 定时器 2 寄存器.....	157
19 无线.....	160
19.1 RF 内核.....	161
19.1.1 中断.....	161
19.1.2 中断寄存器.....	162
19.2 FIFO 存取.....	166
19.3 DMA.....	166
19.4 存储器映射.....	166
19.4.1 RX FIFO.....	166
19.4.2 TX FIFO.....	166
19.4.3 帧过滤和源匹配存储器映射.....	167
19.5 频率和信道编程.....	168
19.6 IEEE 802.15.4-2006 调制格式.....	168
19.7 IEEE802.15.4-2006 帧格式.....	170
19.7.1 物理层.....	170
19.7.2 MAC 层.....	170

19.8 发送模式.....	171
19.8.1 TX 控制.....	171
19.8.2 TX 状态时序.....	171
19.8.3 TX FIFO 存取.....	172
19.8.4 重传.....	172
19.8.5 错误情况.....	172
19.8.6 TX 流程图.....	173
19.8.7 帧处理.....	174
19.8.8 同步头.....	175
19.8.9 帧长度域.....	175
19.8.10 帧校验序列.....	175
19.8.11 中断.....	176
19.8.12 空闲信道评估.....	176
19.8.13 输出功率编程.....	176
19.8.14 提示和技巧.....	176
19.9 接收模式.....	177
19.9.1 RX 控制.....	177
19.9.2 RX 状态时序.....	177
19.9.3 帧处理.....	177
19.9.4 同步头和帧长度域.....	178
19.9.5 帧过滤.....	178
19.9.6 源地址匹配.....	181
19.9.7 帧校验序列.....	184
19.9.8 确认传输.....	185
19.10 RX FIFO 存取.....	187
19.10.1 使用 FIFO 和 FIFOP.....	187
19.10.2 错误情况.....	188
19.10.3 接收信号强度指示器 (RSSI)	188
19.10.4 链路质量指示.....	189
19.11 无线控制状态机.....	189
19.12 随机数产生.....	191
19.13 数据包嗅探和无线测试输出信号.....	192
19.14 命令选通/CSMA-CA 选通处理器.....	193
19.14.1 指令存储器.....	193
19.14.2 数据寄存器.....	194
19.14.3 程序运行.....	194
19.14.4 中断请求.....	194
19.14.5 随机数指令.....	195
19.14.6 运行 CSP 程序.....	195
19.14.7 寄存器.....	195
19.14.8 指令集概况.....	196
19.14.9 指令集定义.....	198
19.14.9.1 DECZ.....	198

19.14.9.2 DECY.....	199
19.14.9.3 DECX.....	199
19.14.9.4 INCZ.....	199
19.14.9.5 INCY.....	199
19.14.9.6 INCX.....	199
19.14.9.7 INCMAXY.....	200
19.14.9.8 RANDXY.....	200
19.14.9.9 INT.....	200
19.14.9.10 WAITX.....	200
19.14.9.11 SETCMP1.....	201
19.14.9.12 WAIT W.....	201
19.14.9.13 WEVENT1.....	201
19.14.9.14 WEVENT2.....	202
19.14.9.15 LABEL.....	202
19.14.9.16 RPT C.....	202
19.14.9.17 SKIP S,C.....	203
19.14.9.18 STOP.....	203
19.14.9.19 SNOP.....	203
19.14.9.20 SRXON.....	204
19.14.9.21 STXON.....	204
19.14.9.22 STXONCCA.....	204
19.14.9.23 SSAMPLECCA.....	204
19.14.9.24 SRFOFF.....	205
19.14.9.25 SFLUSHRX.....	205
19.14.9.26 SFLUSHTX.....	205
19.14.9.27 SACK.....	205
19.14.9.28 SACKPEND.....	206
19.14.9.29 SNACK.....	206
19.14.9.30 SRXMASKBITSET.....	206
19.14.9.31 SRXMASKBITCLR.....	206
19.14.9.32 ISSTOP.....	207
19.14.9.33 ISSTART.....	207
19.14.9.34 ISRXON.....	207
19.14.9.35 ISRXMASKBITSET.....	207
19.14.9.36 ISRXMASKBITCLR.....	207
19.14.9.37 ISTXON.....	208
19.14.9.38 ISTXONCCA.....	208
19.14.9.39 ISSAMPLECCA.....	208
19.14.9.40 ISRFOFF.....	208
19.14.9.41 ISFLUSHRX.....	209
19.14.9.42 ISFLUSHTX.....	209
19.14.9.43 ISACK.....	209
19.14.9.44 ISACKPEND.....	209

19.14.9.45 ISNACK.....	210
19.14.9.46 ISCLEAR.....	210
19.15 RF 寄存器.....	210
19.15.1 寄存器设置更新.....	211
19.15.2 寄存器访问模式.....	211
19.15.3 寄存器描述.....	212
20. 稳压器.....	232
21 可用的软件.....	233
21.1 评估软件 SmartRFTM 软件(www.ti.com/smartrfstudio).....	233
21.2 RemoTITM 网络协议 (www.ti.com/remoti)	233
21.3 SimpliciTITM 网络协议 (www.ti.com/simpliciti)	234
21.4 TIMAC 软件 (www.ti.com/timac)	234
21.5 Z-StackTM 软件 (www.ti.com/z-stack)	235
A 缩略语.....	236
B 参考文献.....	238

序言

关于本手册

2.4GHz 的 CC253x 片上系统解决方案适合于大范围的应用。它们易于建立在基于标准协议的 IEEE 802.15.4(RemoTI™ 网络协议, TIMAC 软件, ZigBee® 兼容解决方案的 Z-Stack™ 软件) 之上, 或者专有的 SimpliciTI™ 网络协议之上。然而, 它的使用并不局限于这些协议。例如, CC253x 系列也适合于 6LoWPAN 和无线 HART 的实现。

本手册的每一章详细描述了模块或外设;但是, 并非 CC253x 系列的所有设备都具有所有的这些特性。

例如功耗和射频性能等详细技术参数, 请见该器件的具体数据手册。

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>) 见附录 C。

欲了解更多可以用于 CC253x 片上系统解决方案的相关软件 (例如, 用于射频性能和功能评估的 SmartRF™ 软件) 的信息请见第 21 章, 本章还包括了关于 RemoTI 网络协议, SimpliciTI 网络协议, TIMAC 软件和 Z-Stack 软件的详细信息。

FCC 警告

本设备仅在实验室测试环境中使用。它可以产生、使用并能辐射射频能量, 但是尚未经过测试, 其是否兼容遵循 FCC 规则 part 15 子部分 J 计算设备的限制, 其设计的目的是为了提供合理保护而免受射频干扰。在其它环境下使用本设备可能导致对射频通信的干扰, 在影响到射频通信的情况下, 要求用户采取一切措施来纠正这种干扰。

设备

CC253x 片上系统解决方案系列包括多种设备。下表提供了每个设备的不同外设、内存大小等的概况和信息。

CC253x 系列概览

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

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

每一个特殊功能寄存器 (SFR) 和 XREG 寄存器在一张单独的表格里面描述。表头以下面的格式给定:

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

每个表中的一行有下面 5 列来对寄存器进行描述:

列 1—位：指明该行描述/寻址的是寄存器的哪一位
列 2—名称：寄存器该位的具体名称
列 3—复位：寄存器该位的复位/初始值
列 4—读/写：重点说明每个单独的位的可访问性（详见表 1）
列 5—描述：对于寄存器位的详细描述，通常是对不同值的描述
在寄存器描述中，每一个寄存器位以一个符号 (R/W) 来表示这个寄存器位的存取模式。
寄存器的值总是以二进制形式给定，除非前面带有“0x”表示以十六进制形式给定。

表 1 寄存器位约定

符号	存取模式
R/W	读/写
R	只读
R0	读出为 0
R1	读出为 1
W	只写
W0	写入为 0
W1	写入为 1
H0	硬件擦除
H1	硬件置位

1 简介

正如序言中提到，CC253x 器件系列为大范围的应用提供了解决方案。为了帮助用户开发这些应用程序，本用户指南侧重于对 CC253x 器件系列不同构件的使用。关于器件的详细描述，完整的功能列表和性能参数，请参见各器件的数据手册。

1.1 概览

图 1-1 框图显示了 CC253x 器件系列不同的构造模块。并非 CC253x 系列的所有设备都具有图 1-1 所列出的所有模块或外设的特性和功能，因此，关于具体设备的框图请参见各个设备的数据手册。

可大致分为 3 类模块：CPU 和相关存储器模块，外设、时钟和电源管理模块，无线模块。

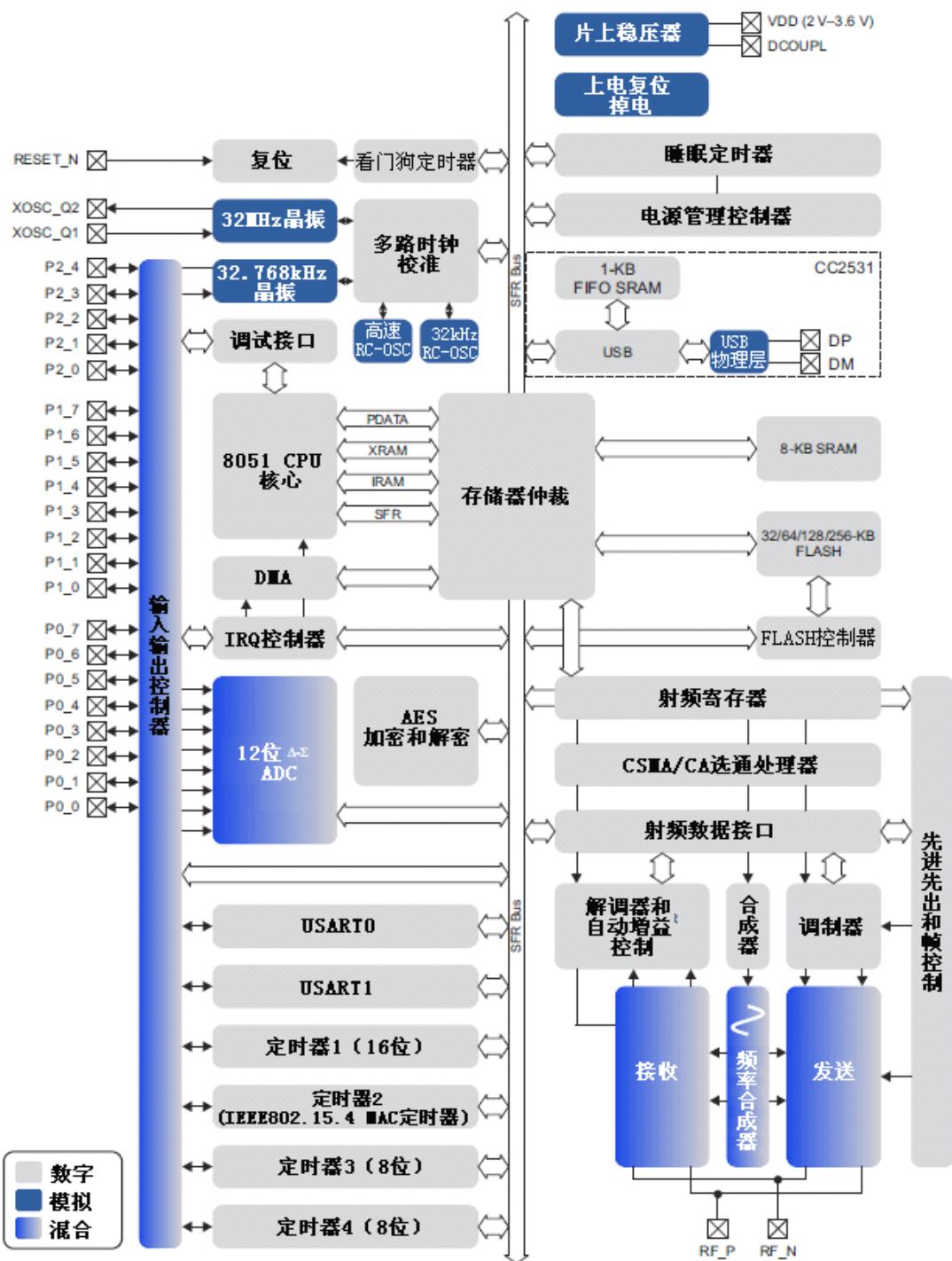


图 1-1 CC253x 框图

1.1.1 CPU 和存储器

CC253x 系列芯片中使用的 8051 CPU 核心是一个单周期的 8051 兼容核心。它有 3 个不同的存储器访问总线（特殊功能寄存器 SFR、数据 DATA 和代码/外部数据 CORE/XDATA），单周期访问 SFR,DATA 和主 SRAM，它还包含一个调试接口和扩展的 18 路输入中断单元。CPU 和存储器的详细功能请见第 2 节。

中断控制器共有 18 个中断源，分为 6 个中断组，每个中断组赋值为 4 个中断优先级之一。当该设备处于空闲模式，任何的中断可以把 CC2530 恢复到主动模式。某些中断还可以将设备从睡眠模式（功耗模式 1—3）唤醒。详细描述请见第 4 节。

存储器交叉开关/仲裁是位于系统核心，它通过 SFR 总线将 CPU 和 DMA 控制器与物理存储器和所有的外接设备连接起来。存储器仲裁有 4 个存储器访问点，访问可以被映射到 3 个物理存储器中的 1 个：1 个 8KB SRAM，Flash 存储器和 XREG/SFR 寄存器。存储器仲裁负责对访问到同一个物理存储器的同步存储器访问进行仲裁和排序。

8KB SRAM 映射到数据存储器空间和部分外部数据存储器空间。8KB SRAM 是一个超低功耗的 SRAM，甚至当数字部分掉电后（功耗模式 2 和 3）它也能保持它的数据。对于低功耗应用，这是一个很重要的特性。

32/64/128/256KB Flash 块为设备提供了在线可编程非易失性存储器，并且映射到代码和外部数据存储器空间。除了保持程序代码和常量以外，非易失性存储器允许应用程序保存必须保留的数据，以保证这些数据在设备重启后可用。使用此功能，可以实现诸如利用保存的具体网络数据，就不再需要经过完全启动、网络寻找和加入过程。

1.1.2 时钟和电源管理

数字内核和外部设备由一个 1.8V 低差稳压器供电（第 20 节）。它提供了电源管理功能（第 4 节），可以实现使用不同的功耗模式以达到低功耗运行，来延长电池寿命。复位设备有 5 种不同的复位源；详见第 5 节。

1.1.3 外部设备

CC253x 包括许多不同的外部设备，使得应用程序开发者可以进行高级应用程序开发。

调试接口（第 3 节）实现了一个专有的两线串行接口来进行在电路调试。通过调试接口可以对 Flash 存储器进行全片擦除，控制启动哪一个振荡器，停止和开始执行用户程序，在 8051 内核上执行供电指示，设置代码断点，在代码中通过指令进行单步调试。利用这些特性可以完美地表现在电路调试和外部 Flash 编程。

CC253x 包含用于存储程序代码的 Flash 存储器。通过调试接口用软件可以对 Flash 存储器进行编程。Flash 控制器（第 6 节）处理对嵌入式 Flash 存储器的写和擦除。Flash 控制器允许页擦除和 4 字节编程。

I/O 控制器（第 7 节）负责所有通用 I/O 引脚。CPU 可以配置某些引脚是由外接设备模块控制或由软件控制，如果是的话，是否每个引脚配置为输入或输出，焊盘上的上拉或下拉电阻是否是连接的。可以在每个引脚上单独使能 CPU 中断。每个连接到 I/O 引脚的外接设备可以在两种不同的 I/O 引脚位置进行选择以确保在各种应用中的灵活性。

系统内有一个通用的 5 通道 DMA 控制器（第 8 节），并且使用外部数据存储器空间来访问存储器，因此可以访问所有物理存储器。每个通道可以在存储器的任何位置用 DMA 描述来配置（触发，优先顺序，传输模式，寻址方式，源指针和目的指针，传输计数）。很多硬件外接设备（AES 核心，Flash 控制器，USART，定时器，ADC 接口）依靠 DMA 控制器在 SFR 或 XREG 地址和 Flash/SRAM 之间的数据传输来有效运行。

定时器 1（第 9 节）是一个 16 位定时器，具有定时器/计数器/脉宽调制功能。它有一个可编程分频器，1 个 16 位周期值和 5 个单独可编程计数器/捕获通道，每个通道有一个 16

位比较值。每个计数器/捕获通道可以用来当作 PWM 输出或用来捕获输入信号的边沿时间。它还可以在 IR 产生模式里进行配置，用来计算定时器 3 的周期，输出是同定时器 3 的输出相与，以产生具有最小 CPU 相互影响的已调制的用户 IR 信号（第 9.9 小节）。

定时器 2（MAC 定时器）（第 18 节）是为支持一个 IEEE 802.15.4 MAC 或其他软件中的时间跟踪协议而特别设计的。该定时器具有一个可配置时间周期和一个可以用来记录已经发生的周期数轨道的 8 位溢出计数器。它还有一个 16 位捕获寄存器，用来记录一个帧开始定界符接收/发送的精确时间或者传输完成的精确时间，以及一个可以在特定时间对无线模块产生各种命令选通信号（开始接收，开始发送等）的 16 位输出比较寄存器。

定时器 3 和定时器 4（第 10 节）是 8 位定时器，具有定时器/计数器/PWM 功能。它们有一个可编程分频器，一个 8 位周期值和一个具有 8 位比较值的可编程计数器信道。每一个计数器信道可以被用来当作 PWM 输出。

睡眠定时器（第 11 节）是一个超低功耗定时器，计数 32kHz 晶体振荡器或 32kHz RC 振荡器周期。睡眠定时器在所有运行模式下（除了功耗模式 3）都可连续运行。睡眠定时器的典型应用是被当作一个实时计数器，或者被当作一个唤醒定时器来离开功耗模式 1 或 2。

ADC（第 12 节）在理想的 32kHz~40kHz 带宽下支持 7~12 位分辨率。直流和音频转换最多可达 8 个输入通道（端口 0）。输入可以被选择为单端输入或差分输入。参考电压可以是内部 AVDD，或一个单端或差分外部信号。ADC 也有温度传感器输入通道。ADC 可以自动操作定期采样过程或通道序列转换过程。

随机数发生器（第 13 节）使用一个 16 位线性反馈移位寄存器（LFSR）来产生随机数，它可以被 CPU 读取或被命令选通处理器直接使用。随机数可以被用作安全机制所需要的随机密钥。

AES 加密/解密核心（第 14 节）允许用户用 128 位密钥的 AES 算法来加密和解密数据。该核心可以支持 IEEE802.15.4 MAC 安全、ZigBee 网络层和应用层所要求的 AES 操作。

内置看门狗定时器（第 15 节）允许 CC253x 在固件挂起时复位自己。当通过软件使能时，看门狗定时器必须被周期性擦除，否则时间一到它就会复位设备。或者它可以被配置为作为一般 32kHz 定时器使用。

USART0 和 USART1（第 16 节）均可配置为一个主/从 SPI 或一个 UART。它们提供在接收和发送时的双缓冲和硬件流控制，因而非常适合于大吞吐量全双工应用。每一个都有它自己的高精度的波特率发生器，因此可以解放普通计时器出来作其他用途。

USB2.0 全速控制器（仅 CC2531 具备）有 5 个端点，1KB FIFO RAM 的双缓冲。其功能描述请见第 17 节。

1.1.4 无线

CC253x 具有一个 IEEE 802.15.4 标准的无线收发器。RF 核心控制模拟无线模块。另外，它为 MCU 和无线之间提供了一个接口，以使得可以发送命令、读取状态、自动操作和对无线事件进行排序。无线部分还包括一个数据包过滤和地址识别模块。关于无线的详细描述请见第 19 节。

1.2 应用

如概览（1.1 节）所述，本用户指南的重点是不同模块的功能性，它们可用于建立基于



CC253x 用户指南

CC253x 系列芯片的不同类型的应用。当察看完整的应用程序开发过程时，还有其它的信息是有益的。然而，由于这些信息和帮助没有特定于某个设备（即不是 CC253x 系列芯片特有的），读者可以参考以下段落的其它信息来源。

第一步是通过购买一套开发套件（见网站上具体器件产品的相关开发套件的链接）来建立开发环境（硬件、工具等）。开发套件包含一个现成的演示和关于如何建立开发环境的信息；安装所需的驱动程序（通过安装 SmartRF 软件可以轻松完成，见第 21.1 节），设置编译工具链等等。只要安装了开发环境，就可以准备开始应用程序的开发了。

编写应用软件最简单的方法是把应用程序建立在一个可用的标准协议上（21.2 小节的 RemoTI 网络协议；21.4 小节的 TIMAC 软件；或者 21.5 小节的符合 ZigBee 解决方案的 Z-Stack 软件）；或者 TI 专有的 SimpliTi 网络协议（21.3 小节）。它们都附带有几个示例应用程序。

对于用户特定硬件的布局设计，设计人员可以在不同产品页上（B.1 小节）找到相关参考设计。设计人员通过复制这些设计，可以获得硬件的最佳性能。使用 SmartRF Studio 软件（21.1 小节）可以很容易地对开发的硬件进行测试。

如果用户开发的最终系统达不到预期的性能，建议在开发套件的硬件上运行用户的应用程序，并查看软件在开发套件的硬件上运行的结果。要检查用户特定的硬件，首先最好是使用 SmartRF Studio 软件，在相同的设置下来比较开发套件和用户特定硬件的性能。

用户还可以通过加入低功耗 RF 网络社区（B.2 小节）和订阅低功耗 RF 电子简讯（B.4 小节）来获得更多的信息和帮助。

如要联系第三方以帮助开发或者使用模块，查询德州仪器低功耗 RF 开发商网络。

2 8051CPU

该片上系统解决方案是基于增强的 8051 内核。关于此内核、存储器映射、指令集和中断的详细信息在以下小节描述。

2.1 8051 CPU 介绍

增强的 8051 内核使用标准 8051 指令集。指令执行速度比标准 8051 快，原因如下：

- 每个指令周期中的一个时钟周期与标准 8051 每个指令周期中的 12 个时钟周期相对应。
- 取消了无用的总线状态。

由于指令周期在可能的情况下包含了取指令操作所需的时间，故绝大多数单字节指令在一个时钟周期内完成。除了速度改进之外，增强的 8051 内核也包含了下列增强的架构：

- 第二数据指针
- 扩展了 18 个中断源

该 8051 内核的目标代码与工业标准 8051 微控制器目标代码兼容。也就是说，8051 内核上的目标代码编译和工业标准 8051 编译器或汇编执行是同等功率。但是，由于与标准 8051 使用不同的指令定时，现有的带有定时循环的代码可能需要修改。此外，由于外接设备单元比如定时器和串行端口不同于它们在其它的 8051 内核，包含有使用外接设备单元特殊功能寄存器 SFR 的指令代码将不能正常运行。

Flash 预取默认是不使能的，提高了 CPU 高达 33% 的性能。这是以功耗稍有增加为代价的，但是因为它更快，所以在大多数情况下提高了能源消耗。可以在 FCTL 寄存器中使能 Flash 预取。

2.2 存储器

8051 CPU 架构有 4 个不同的存储空间。8051 具有单独的用于程序存储和数据存储的存储空间。8051 存储空间如下（详细内容请见 2.2.1 节和 2.2.2 节）：

代码 (CODE)：只读存储空间，用于程序存储。存储空间地址 64 KB。

数据 (DATA)：可存取存储空间，可以直接或间接被单个周期的 CPU 指令访问。这个存储空间地址为 256 字节。数据存储空间的低 128 字节可以直接或间接访问，而高 128 字节只能够间接访问。

外部数据 (XDATA)：可存取存储空间，通常需要 4-5 个指令周期来访问。该存储空间地址为 64KB。在硬件里访问外部数据存储也比数据访问要慢，因为在 CPU 内核代码存储空间和外部数据存储空间共享一条公共总线，并且从代码存储空间进行指令预存取不能和外部数据访问并行。

特殊功能寄存器 (SFR)：可存取寄存器存储空间，可以被单个的 CPU 指令直接访问。该存储空间由 128 字节构成。对于 SFR 寄存器，它的地址可以被分成 8 等份，每个位仍然可以单独寻址。

这 4 个不同的存储空间在 8051 架构中都截然不同，但在 CC253x 中有一部分是重叠的来缓解 DMA 传输和硬件调试操作。

这四个不同的存储空间是如何映射到 3 个物理存储器（Flash 程序存储器、SRAM 和存储器映射寄存器）的，将会在 2.2.1 节和 2.2.2 节中描述。

2.2.1 存储器映射

与标准 8051 存储器映射不同之处有两个重要方面，如下所述。

首先，为了使得 DMA 控制器访问全部物理存储空间，因而允许 DMA 在不同的 8051 存储空间之间传输，部分特殊功能寄存器 SFR 和 DATA 存储空间被映射到 XDATA 存储空间。

其次，对于 CODE 存储空间映射有 2 个可选方案可以使用。第一个方案是标准 8051 映射，只有程序存储器（即 Flash 存储器）映射到 CODE 存储空间。在一个设备复位后默认使用这种映射。

第二个方案用于执行来自 SRAM 的代码。在该模式下，SRAM 被映射到 0x8000 到 (0x8000+SRAM_SIZE-1) 的区域。该映射如图 2-2 所示。执行来自 SRAM 的代码既提高了性能又降低了功耗。

XDATA 的高 32KB 是只读区，称为 XBANK。任何可用的 32KB Flash bank 都可以被映射到该区域。这使软件可以访问整个 Flash 存储器。这一区域的典型作用是用来存储另外的常量数据。

关于全部 8051 存储空间的映射的详细内容将在 2.2.2 节给出。

图 2-1 到图 2-3 的存储器映射图，显示了不同的物理存储器是如何映射到 CPU 存储空间的。可用的 Flash bank 的数量取决于 Flash 本身的小。

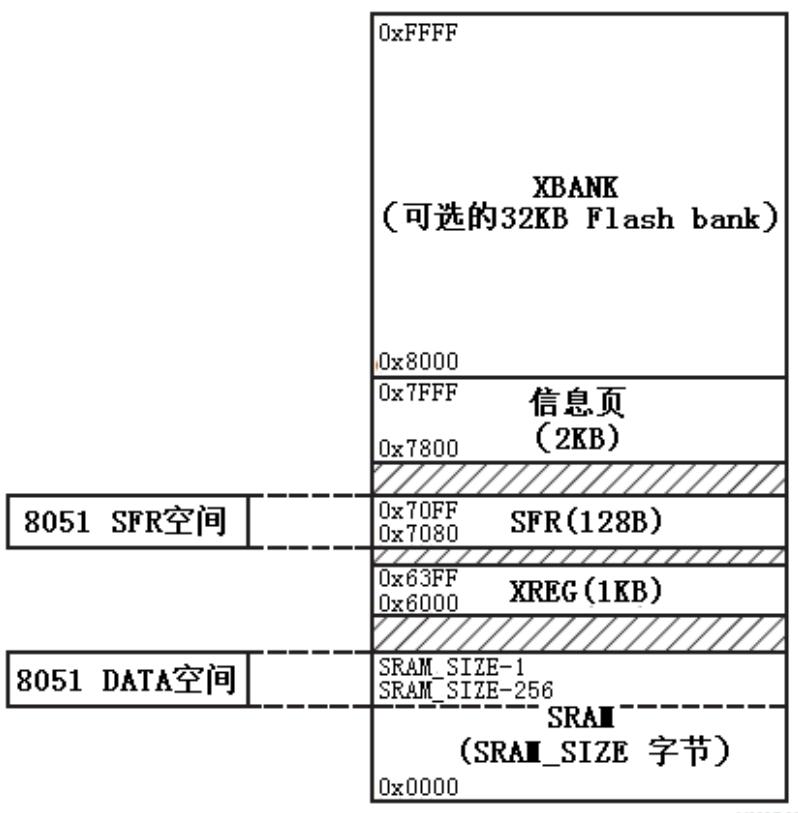


图 2-1 XDATA 存储器空间（显示 SFR 和 DATA 映射）

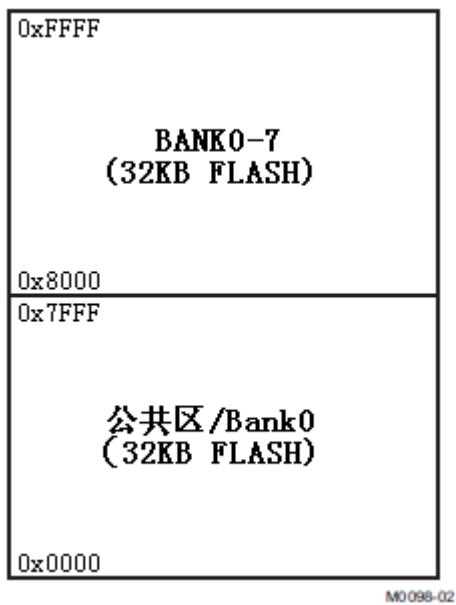


图 2-2 CODE 存储器空间

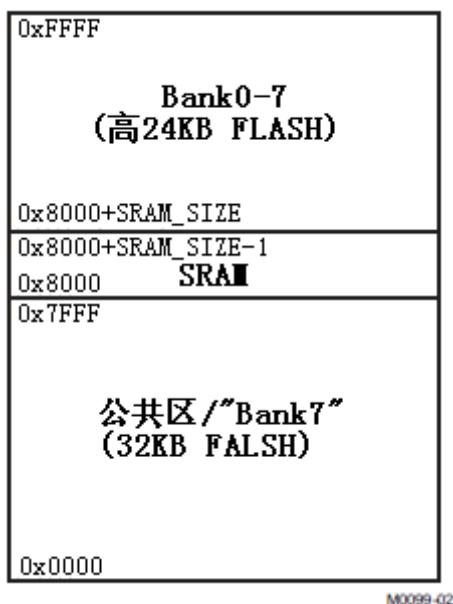


图 2-3 用于运行用 SRAM 运行代码的 CODE 存储器空间

2.2.2 CPU 存储器空间

XDATA 存储器空间。外部数据 (XDATA) 存储器的映射如图 2-1 所示。

SRAM 映射到 0x0000 到 (SRAM_SIZE-1) 的地址范围。

XREG 区域映射到 1KB 地址范围 (0x6000—0x63FF)。这些寄存器是有效地扩展 SFR 寄存器空间的额外寄存器。一些外设寄存器和大部分射频控制和数据寄存器映射到该区域。

SFR 寄存器映射到地址范围 (0x7080—0x70FF)。

Flash 信息页 (2KB) 映射到地址范围 (0x7800—0x7FFF)。这是只读区域且包含了该设备的各种信息。

XDATA 存储器空间的高 32KB (0x8000—0xFFFF) 是只读 Flash 代码 bank (XBANK)，

通过 MEMCTR.XBANK[2: 0]位被映射到任何可用的 Flash bank。

Flash 存储器、SRAM 和寄存器到 XDATA 的映射允许 DMA 控制器和 CPU 在一个统一的地址空间对所有物理存储器进行存取操作。

写入存储器映射中未执行区域（图中的阴影部分）无效。读取未执行区域返回 0x00。写只读区域即 Flash 区域将被忽略。

CODE 存储器空间。如图 2-2 所示，CODE 存储器空间为 64KB，被分为一个公共区（0x0000—0x7FFF）和一个 bank 区（0x8000—0xFFFF）。公共区通常被映射到物理 Flash 存储器的低 32KB（bank0）。Bank 区被映射到任何可用的 32KB Flash bank（从 bank0—bank7）。可用的 Flash bank 数量取决于 Flash 大小。使用 Flash bank 选择寄存器 FMAP 来选择 Flash bank。对于 CC2530F32，没有 Flash 存储器可以被映射到 bank 区。在 CC2530F32 上读取该区域返回 0x00。

要允许从 SRAM 执行程序，可以将可用的 SRAM 映射到 bank 区域的较低地址范围，从 0x8000 到（0x8000+SRAM_SIZE-1）。当前选择的 bank 的其余部分仍然映射到从（0x8000+SRAM_SIZE）到 0xFFFF 的地址范围。设置 MEMCTR.XMAP 位来使能此功能。

DATA 存储器空间。数据（DATA）存储器的 8 位地址范围，映射到 SRAM 的高端 256 字节，即从（SRAM_SIZE-256）到（SRAM_SIZE-1）的地址范围。

SFR 存储器空间。通过这个存储器空间可以对具有 128 个入口的硬件寄存器进行存取，也可以通过 XDATA 地址空间的地址范围 0x7080~0x70FF 对 SFR 寄存器进行存取。一些 CPU 特定 SFR 寄存器存在于 CPU 内核里，所以只能用 SFR 存储器空间存取，并且不能通过重复映射到 XDATA 存储器空间。这些特定 SFR 寄存器在 SFR 寄存器中列出。

2.2.3 物理存储器

RAM。所有设备带有静态 RAM。上电时 RAM 的数据是未定义的。RAM 在所有功耗模式下都可保持数据。

Flash 存储器。片上 Flash 存储器主要用来保存程序代码和常量数据。它具有以下特点：

- Flash 页大小：2KB
- Flash 页擦除时间：20ms
- Flash 全片擦除时间：20ms
- Flash 写时间（4 字节）：20us
- 数据保存（室温条件下）：100 年
- 编程/擦除次数：循环 20000 次

FLASH 存储器由一组 2KB 的页组成。在这组 2KB 的页面中有一个页面被称为信息页。它是一个 2KB 的只读区域，用于存储 CC253x 系列片上系统的多种信息，其中包括了一个来自 TI 地址范围的一个全球唯一的 IEEE 地址，它以最低位优先的形式存储在 XDATA 地址 0x780C。有关信息页的具体内容可参看 TI 后续出版的设计说明。最高有效页面（因为 CC2530 有不同 FLASH 容量的版本，所以此处我们说“最高有效页面”）中的 16 个字节（128 比特）用作“页面锁定”位和“调试锁定”位，其中 127 比特分别用于除信息页以外的其他 FLASH 页的锁定，1 比特用于调试锁定。当不处于调试模式时，页是隐式锁定的。当页的锁定位为 0 时，无法对页进行擦除/写入。当调试锁定位为 0 时，调试接口上的大多数命令被忽略。调试锁定位的主要目的是保护 FLASH 存储器中的内容不被读出。通过对 FLASH 控制器的操作来实现对 FLASH 存储器的内容的写入和擦除。

当 CPU 从 Flash 存储器读指令和常量时，它通过一个缓存取得指令。指令和常量数据

的 4 个字节被缓存在 4 字节的边界。也就是说，例如当 CPU 从地址 0x00F1 读取时，字节 0x00F0—0x00F3 被缓存。一个单独的预取单元能够预取指令的另外 4 个字节。指令缓存主要通过减少对自己 Flash 存储器访问的总时间来减少功率消耗。指令缓存可以用寄存器位 FCTL.CM[1: 0] 来禁止。但是这样将增大功率消耗，因此不推荐禁止它。当使用带预取的默认缓存模式时，从 Flash 的执行时间不能精确到周期，即不能精确确定读取一组指令所需的时钟周期数量。要获得精确到周期的执行，就要使能实时缓存模式，并且确保所有的 DMA 传输为低优先级。预取模式可提高 33% 的性能，但因为它消耗了 Flash 读，而增加了功率消耗。通常情况下，可以提高 15%—20% 性能。但是由于等待 Flash 返回指令/数据所消耗的时钟周期较少，所以总能量可能减少了（取决于应用程序）。这极度依赖于应用程序，并且需要有效的使用功耗模式。

SFR 寄存器。特殊功能寄存器 (SFR) 用于控制 8051 CPU 核心和/或外部设备的一些功能。一部分 8051 内核特殊功能寄存器与标准 8051 特殊功能寄存器相同。但是，另一部分特殊功能寄存器不同于标准 8051 特殊寄存器。它们用来与外部设备单元接口，以及控制 RF 收发器。

CC253x 的全部特殊功能寄存器地址如表 2-1 所示。其中，标准 8051 内部特殊功能寄存器用灰色背景表示，其它的为 CC253x 独有的特殊功能寄存器。

注意：所有标准 8051 内部特殊功能寄存器（表 2-1 中以灰色背景显示）只能够通过该寄存器的空间存取，因为这些寄存器没有映射到外部数据 (XDATA) 空间。只有端口寄存器 (P0, P1 和 P2) 例外，它们可以从 XDATA 读取。

表 2-1 特殊功能寄存器 (SFR) 概览

寄存器名	SFR 地址	模块	描述
ADCCON1	0xB4	ADC	模/数转换控制 1
ADCCON2	0xB5	ADC	模/数转换控制 2
ADCCON3	0xB6	ADC	模/数转换控制 3
ADCL	0xBA	ADC	模/数转换低位数据
ADCH	0xBB	ADC	模/数转换高位数据
RNDL	0xBC	ADC	随机数发生器低位数据
RNDH	0xBD	ADC	随机数发生器高位数据
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) 读取。
SP	0x81	CPU	堆栈指针
DPL0	0x82	CPU	数据指针 0 低位字节
DPH0	0x83	CPU	数据指针 0 高位字节
DPL1	0x84	CPU	数据指针 1 低位字节
DPH1	0x85	CPU	数据指针 1 高位字节
PCON	0x87	CPU	功耗模式控制
TCON	0x88	CPU	中断标志
P1	0x90	CPU	端口 1。可从 XDATA (0x7090) 读取。
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) 读取。
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
DMA1CFGH	0xD2	DMA	DMA 通道 1~4 配置低位地址
DMA1CFGL	0xD3	DMA	DMA 通道 1~4 配置高位地址
DMA0CFGH	0xD4	DMA	DMA 通道 0 配置低位地址
DMA0CFGL	0xD5	DMA	DMA 通道 0 配置高位地址
DMAARM	0xD6	DMA	DMA 通道准备工作
DMAREQ	0xD7	DMA	DMA 通道启动请求和状态
—	0xAA	—	保留
—	0x8E	—	保留
—	0x99	—	保留
—	0xB0	—	保留
—	0xB7	—	保留
—	0xC8	—	保留
P0IFG	0x89	输入/输出控制 (IOC)	端口 0 中断状态标志
P1IFG	0x8A	输入/输出控制 (IOC)	端口 1 中断状态标志
P2IFG	0x8B	输入/输出控制 (IOC)	端口 2 中断状态标志
PICTL	0x8C	输入/输出控制 (IOC)	端口引脚中断控制
P0IEN	0xAB	输入/输出控制 (IOC)	端口 0 中断使能
P1IEN	0x8D	输入/输出控制 (IOC)	端口 1 中断使能
P2IEN	0xAC	输入/输出控制 (IOC)	端口 2 中断使能
P0INP	0x8F	输入/输出控制 (IOC)	端口 0 输入模式
PERCFG	0xF1	输入/输出控制 (IOC)	外部设备 I/O 配置
APCFG	0xF2	输入/输出控制 (IOC)	模拟外部设备 I/O 配置
P0SEL	0xF3	输入/输出控制 (IOC)	端口 0 功能选择
P1SEL	0xF4	输入/输出控制 (IOC)	端口 1 功能选择
P2SEL	0xF5	输入/输出控制 (IOC)	端口 2 功能选择
P1INP	0xF6	输入/输出控制 (IOC)	端口 1 输入模式
P2INP	0xF7	输入/输出控制 (IOC)	端口 2 输入模式
P0DIR	0xFD	输入/输出控制 (IOC)	端口 0 方向
P1DIR	0xFE	输入/输出控制 (IOC)	端口 1 方向
P2DIR	0xFF	输入/输出控制 (IOC)	端口 2 方向

PMUX	0xAE	输入/输出控制 (IOC)	掉电信号多路器
MEMCTR	0xC7	存储器	存储器系统控制
FMAP	0x9F	存储器	Flash 存储器 bank 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	睡眠定时器 (ST)	睡眠定时器 0
ST1	0x96	睡眠定时器 (ST)	睡眠定时器 1
ST2	0x97	睡眠定时器 (ST)	睡眠定时器 2
STLOAD	0xAD	睡眠定时器 (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 (Timer1)	定时器 1 通道 0 捕获/比较值低位
T1CC0H	0xDB	定时器 1 (Timer1)	定时器 1 通道 0 捕获/比较值高位
T1CC1L	0xDC	定时器 1 (Timer1)	定时器 1 通道 1 捕获/比较值低位
T1CC1H	0xDD	定时器 1 (Timer1)	定时器 1 通道 1 捕获/比较值高位
T1CC2L	0xDE	定时器 1 (Timer1)	定时器 1 通道 2 捕获/比较值低位
T1CC2H	0xDF	定时器 1 (Timer1)	定时器 1 通道 2 捕获/比较值高位
T1CNTL	0xE2	定时器 1 (Timer1)	定时器 1 计数器低位
T1CNTH	0xE3	定时器 1 (Timer1)	定时器 1 计数器高位
T1CTL	0xE4	定时器 1 (Timer1)	定时器 1 控制和状态
T1CCTL0	0xE5	定时器 1 (Timer1)	定时器 1 通道 0 捕获/比较控制
T1CCTL1	0xE6	定时器 1 (Timer1)	定时器 1 通道 1 捕获/比较控制
T1CCTL2	0xE7	定时器 1 (Timer1)	定时器 1 通道 2 捕获/比较控制
T1STAT	0xAF	定时器 1 (Timer1)	定时器 1 状态
T2CTRL	0x94	定时器 2 (Timer2)	定时器 2 控制
T2EVTCFG	0x9C	定时器 2 (Timer2)	定时器 2 事件配置
T2IRQF	0xA1	定时器 2 (Timer2)	定时器 2 中断标志
T2M0	0xA2	定时器 2 (Timer2)	定时器 2 复用寄存器 0
T2M1	0xA3	定时器 2 (Timer2)	定时器 2 复用寄存器 1
T2MOVF0	0xA4	定时器 2 (Timer2)	定时器 2 复用溢出寄存器 0
T2MOVF1	0xA5	定时器 2 (Timer2)	定时器 2 复用溢出寄存器 1
T2MOVF2	0xA6	定时器 2 (Timer2)	定时器 2 复用溢出寄存器 2
T2IRQM	0xA7	定时器 2 (Timer2)	定时器 2 中断使能
T2MSEL	0xC3	定时器 2 (Timer2)	定时器 2 复用选择
T3CNT	0xCA	定时器 3 (Timer3)	定时器 3 计数器
T3CTL	0xCB	定时器 3 (Timer3)	定时器 3 控制
T3CCTL0	0xCC	定时器 3 (Timer3)	定时器 3 通道 0 比较控制

T3CC0	0xCD	定时器 3 (Timer3)	定时器 3 通道 0 比较值
T3CCTL1	0xCE	定时器 3 (Timer3)	定时器 3 通道 1 比较控制
T3CC1	0xCF	定时器 3 (Timer3)	定时器 3 通道 1 比较值
T4CNT	0xEA	定时器 4 (Timer4)	定时器 4 计数器
T4CTL	0xEB	定时器 4 (Timer4)	定时器 4 控制
T4CCTL0	0xEC	定时器 4 (Timer4)	定时器 4 通道 0 比较控制
T4CC0	0xED	定时器 4 (Timer4)	定时器 4 通道 0 比较值
T4CCTL1	0xEE	定时器 4 (Timer4)	定时器 4 通道 1 比较控制
T4CC1	0xEF	定时器 4 (Timer4)	定时器 4 通道 1 比较值
TIMIF	0xD8	TMINT	定时器 1/3/4 联合中断使能/标志
U0CSR	0x86	USART0	USART0 控制和状态
U0DBUF	0xC1	USART0	USART0 收/发数据缓存
U0BAUD	0xC2	USART0	USART0 波特率控制
U0UCR	0xC4	USART0	USART0 UART 控制
U0GCR	0xC5	USART0	USART0 通用控制
U1CSR	0xF8	USART1	USART1 控制和状态
U1DBUF	0xF9	USART1	USART1 收/发数据缓存
U1BAUD	0xFA	USART1	USART1 波特率控制
U1UCR	0xFB	USART1	USART1 UART 控制
U1GCR	0xFC	USART1	USART1 通用控制
WDCTL	0xC9	看门狗 (WDT)	看门狗定时器控制

XREG 寄存器。 XREG 寄存器是 XDATA 存储器空间里另外的寄存器。这些寄存器主要用于无线配置和控制。每一个寄存器将在 3.6 节进行完整描述。表 2-2 给出了每一个寄存器的地址空间的概述。

表 2-2 XREG 寄存器概览

XDATA 地址	寄存器名	描述
0x6000-0x61FF	—	无线寄存器
0x6200-0x622B	—	USB 寄存器
0x6249	CHVER	芯片版本号
0x624A	CHIPID	芯片标识符
0x6260	DBGDATA	调试接口写数据
0x6270	FCTL	Flash 控制
0x6271	FADDRL	Flash 地址低位
0x6272	FADDRH	Flash 地址高位
0x6273	FWDATA	Flash 写数据
0x6276	CHIPINFO0	芯片信息字节 0
0x6277	CHIPINFO1	芯片信息字节 1
0x6290	CLD	时钟丢失检测
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/比较值低位 (SFR 寄存器另外的 XREG 映射)
0x62A7	T1CC0H	定时器 1 通道 0 捕获/比较值高位 (SFR 寄存器另外的 XREG 映射)
0x62A8	T1CC1L	定时器 1 通道 1 捕获/比较值低位 (SFR 寄存器另外的 XREG 映射)
0x62A9	T1CC1H	定时器 1 通道 1 捕获/比较值高位 (SFR 寄存器另外的 XREG 映射)
0x62AA	T1CC2L	定时器 1 通道 2 捕获/比较值低位 (SFR 寄存器另外的 XREG 映射)
0x62AB	T1CC2H	定时器 1 通道 2 捕获/比较值高位 (SFR 寄存器另外的 XREG 映射)
0x62AC	T1CC3L	定时器 1 通道 3 捕获/比较值低位
0x62AD	T1CC3H	定时器 1 通道 3 捕获/比较值高位
0x62AE	T1CC4L	定时器 1 通道 4 捕获/比较值低位
0x62AF	T1CC4H	定时器 1 通道 4 捕获/比较值高位
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 存储器存取

MPAGE 寄存器在执行指令“MOVX A, @Ri”和“MOVX @Ri, A”时使用。MPAGE 给出高 8 位的地址，而寄存器 Ri 给出低 8 位的地址。

在 8051 的某些执行中，这种类型的 XDATA 存取使用 P2 执行以给定最明显的地址位。现有的软件或许因此必须适应使用 MPAGE 而不是 P2。

MPAGE (0x93) —选择存储器页

位	名称	复位	读/写	描述
7: 0	MPAGE[7: 0]	0x00	R/W	存储器页，执行 MOVX 指令时地址的高位字节

2.2.5 存储器仲裁

存储器仲裁处理 CPU 和 DMA 对所有物理存储器的存取(除了 CPU 内部寄存器)。当 CPU 和 DMA 发生存取冲突时，存储器仲裁停止其中一个总线主机来解决存取冲突。

控制寄存器 MEMCTR 和 FMAP 用来控制各种存储器子系统。下面描述这两种寄存器。

MEMCTR.XMAP 必须设置为使能从 RAM 执行程序。

Flash bank 映射寄存器 FMAP，控制物理 32KB 代码 bank 到 CODE 存储器空间里程序地址区域 0x8000—0xFFFF 的映射。

MEMCTR (0xC7) —存储器仲裁控制

位	名称	复位	读/写	描述
7: 3	—	0000	R0	保留
3	XMAP	0	R/W	XDATA 映射到代码。当该位置 1，SRAM XDATA 区域 0x0000 到 (SRAM_SIZE-1) 被映射到 0x8000 到 CODE 区域的 (0x8000+SRAM_SIZE-1)。这样就使能从 RAM 执行程序代码。

				0: 禁止 SRAM 映射到 CODE 1: 使能 SRAM 映射到 CODE
2: 0	XBANK[2: 0]	000	R/W	XDATA bank 选择。控制物理 Flash 存储器的哪一个代码 bank 映射到 XDATA 区域(0x8000—0xFFFF)。当该位为 0 时，映射到根 bank。有效设置取决于设备的 Flash 大小。一个无效的写操作被忽略，即不更新 XBANK[2: 0]。 32KB 版本：仅为 0 (即总是映射到根 bank) 64KB 版本：0—1 128KB 版本：0—3 256KB 版本：0—7

FMAP (0x9F) —Flash Bank 映射

位	名称	复位	读/写	描述
7: 3	—	0000 0	R0	未使用
2: 0	MAP[2: 0]	001	R/W	Flash bank 映射。控制哪一个 bank 映射到 CODE 存储器空间的 bank 区 (0x8000—0xFFFF)。当该位为 0 时，映射到根 bank。有效设置取决于设备的 Flash 大小。一个无效的写操作被忽略，即不更新 MAP[2: 0]。 32KB 版本：不能写值。Bank 区域只是用来从 SRAM 运行程序代码。见 MEMCTR.XMAP。 64KB 版本：0—1 128KB 版本：0—3 256KB 版本：0—7

2.3 CPU 寄存器

本节描述 CPU 内的内部寄存器。

2.3.1 数据指针

数据指针 DPTR0 和 DPTR1 以加速数据块到/从存储器的执行，主要用于 CODE 和 XDATA 空间的存取。例如：

MOVCA, @A+DPTR

MOVA, @DPTR

数据指针选择寄存器 DPS 里的数据指针选择位是位 0。通过使用数据指针即上面指令中的一个选择哪个数据指针在指令执行时有效。

两个数据指针的宽度均为两个字节，包括下面两个特殊功能寄存器：

- DPTR0—DPH0: DPL0
- DPTR1—DPH1: DPL1

DPH0 (0x83) — 数据指针 0 的高位字节

位	名称	复位	读/写	描述
7: 0	DPH0[7: 0]	0x00	R/W	数据指针 0, 高位字节

DPL0 (0x82) — 数据指针 0 的低位字节

位	名称	复位	读/写	描述
7: 0	DPL0[7: 0]	0x00	R/W	数据指针 0, 低位字节

DPH1 (0x85) — 数据指针 1 的高位字节

位	名称	复位	读/写	描述
7: 0	DPH1[7: 0]	0x00	R/W	数据指针 1, 高位字节

DPL1 (0x84) — 数据指针 1 的低位字节

位	名称	复位	读/写	描述
7: 0	DPL1[7: 0]	0x00	R/W	数据指针 1, 低位字节

DPS (0x92) — 数据指针选择

位	名称	复位	读/写	描述
7: 1	—	0000 000	R0	不使用
0	DPS	0	R/W	数据指针选择, 用来使选中的数据指针有效 0: DPTR0 1: DPTR1

2.3.2 寄存器 R0—R7

CC253x 提供了 4 个寄存器 bank (不要与只适用于 Flash 存储器组织的 CODE 存储器空间 bank 混淆), 每个具有 8 个寄存器。这些寄存器 bank 被映射到地址为 0x00—0x07, 0x08—0x0F, 0x10—0x17 和 0x18—0x1F 的 DATA 存储器空间。每个寄存器 bank 包含 8 个 8 位寄存器 R0—R7。通过程序状态字 PSW.RS[1: 0]来选择寄存器 bank 被使用。寄存器 bank0 使用内部触发器来存储值 (绕过/不使用 SRAM), bank1-3 使用 SRAM 来存储值, 以此节省功耗。通常情况下, 使用寄存器 bank0 而不用寄存器 bank1-3, 可以降低大约 200mA 的电流消耗。

2.3.3 程序状态字

程序状态字 (PSW) 用许多位来显示 CPU 的当前状态。程序状态字作为一个特殊功能寄存器是可以出入的并且它是位寻址。PSW 如下所示, 包含进位标志, 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。PSW 的未定的 2 位可被当作用户定义状态标志使用。

PSW (0xD0) — 程序状态字

位	名称	复位	读/写	描述
7	CY	0	R/W	进位标志。当最后的算术运算产生了一个进位 (加) 或借位 (减) 时置为 1, 否则对所有的算术运算清 0。
6	AC	0	R/W	BCD 操作辅助进位标志。当最后的算术运算产生了一个进位 (加) 或借位 (减) 时置为 1, 否则对所有的算术运算清 0。

5	F0	0	R/W	用户自定义，位寻址。
4: 3	RS[1: 0]	00	R/W	寄存器 bank 选择位。从 DATA 空间里的 4 个可能的寄存器选择哪一组 R7-R0 寄存器使用。 00: 寄存器 bank0, 0x00—0x07 01: 寄存器 bank1, 0x08—0x0F 10: 寄存器 bank2, 0x10—0x17 11: 寄存器 bank3, 0x18—0x1F
2	OV	0	R/W	溢出标志。由算术运算设定。当最后的算术运算产生了一个进位（加）或借位（减）时置为 1，否则对所有的算术运算清 0。
1	F1	0	R/W	用户自定义，位寻址。
0	P	0	R/W	奇偶标志。累加器的奇偶，如果它包含一个 1's 的奇数由硬件置为 1，否则清 0。

2.3.4 累加器

ACC 即累加器。这是许多算术指令、数据传输和其它指令的源和目的地。累加器（在指令中涉及累加器）的助记符指 A 而不是 ACC。

ACC (0xE0) — 累加器

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

2.3.5 B 寄存器

B 寄存器在乘法指令和除法指令执行期间被用来作为第二个 8 位参数。当不被用于这些目的时它可能用来当作高速缓存寄存器来保持临时数据。

B (0xF0) — B 寄存器

位	名称	复位	读/写	描述
7: 0	B[7: 0]	0x00	R/W	B 寄存器。用在乘法/除法指令中。

2.3.6 堆栈指针

堆栈驻留在 DATA 存储器空间里并且向上增长。PUSH 指令执行时，首先增加堆栈指针 (SP)，然后复制字节到堆栈里。复位后堆栈指针被初始化为 0x07，并且它将被增加 1 为 0x08，将从该位置启动。该位置是第二个寄存器 bank 中的第一个寄存器 (R0)。因此，为了使用多个寄存器 bank，堆栈指针应当被初始化为一个不同的位置，该位置不能被数据存储所使用。

SP (0x81) — 堆栈指针

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

2.4 指令集摘要

8051 指令集总结在表 2-3。所有的助记符版权属于因特尔公司，1980。

下面的约定使用于指令集里：

- Rn—当前选择寄存器 bank 的寄存器 R7-R0。
- Direct—8 位内部数据位置的地址。它可以为 DATA 范围 (0x00-0x7F) 或特殊功能寄存器范围 (0x80—0xFF)。
- @Ri—8 位内部数据位置，DATA 范围 (0x00—0xFF)。通过寄存器 R1 或 R0 间接寻址。
- #data—包含在指令中的 8 位常数。
- #data16—包含在指令中的 16 位常数。
- addr16—16 位目的地址。通过 LCALL 和 LJMP 使用。分支指令可以在 64KBCODE 存储器空间的任何位置。
- addr11—11 位目的地址。通过 ACALL 和 AJMP 使用。分支指令将在程序存储器的相同的 2KB 页里作为下面指令的首字节。
- rel—带符号（二进制补码）8 位偏移字节。通过 SJMP 和所有的条件跳转使用。相对于下面指令的首字节范围从 -128 到 127 字节。
- bit—DATA 范围或特殊功能寄存器的直接寻址位。

在 PSW 里影响 CPU 标志设置的指令如表 2-4 所列。注意在 PSW 寄存器上的操作或 PSW 里位的操作也会影响标志设置。还应注意，许多指令对被访问的存储器单元的周期数假定是单周期访问，即最好的情况。但并非总是如此。例如，读取 Flash 可能需要 1-3 个周期。

表 2-3 指令集摘要

助记符	描述	十六进制操作码	字节	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	添加间接 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	添加间接 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	来自带有借位 A 的减法寄存器	98-9F	1	1
SUBB A,direct	来自带有借位 A 的减法直接位	95	2	2
SUBB A,@Ri	来自带有借位 A 的减法间接 RAM	96-97	1	2
SUBB A,#data	来自带有借位 A 的直接减法数据	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	递增间接 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	递减间接 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	累加器与寄存器	58-5F	1	1
ANL A,direct	累加器与直接字节	55	2	2
ANL A,@Ri	累加器与间接 RAM	56-57	1	2
ANL A,#data	累加器与直接字节	54	2	2
ANL direct,A	直接字节与累加器	52	2	3
ANL direct,#data	直接数据与直接字节	53	3	4
ORL A,Rn	累加器或寄存器	48-4F	1	1
ORL A,direct	累加器或直接字节	45	2	2
ORL A,@Ri	累加器或间接 RAM	46-47	1	2
ORL A,#data	累加器或直接数据	44	2	2
ORL direct,A	直接字节或累加器	42	2	3
ORL direct,#data	直接字节或直接数据	43	3	4
XRL A,Rn	累加器异或寄存器	68-6F	1	1
XRL A,direct	累加器异或直接字节	65	2	2
XRL A,@Ri	累加器异或间接 RAM	66-67	1	2
XRL A,#data	累加器异或直接数据	64	2	2
XRL direct,A	直接字节异或累加器	62	2	3
XRL direct,#data	直接字节异或直接数据	63	3	4
CLR A	累加器清零	E4	1	1
CPL A	累加器取反	F4	1	1
RL A	累加器左循环移位	23	1	1
RLC A	累加器连进位标志左循环移位	33	1	1
RR A	累加器右循环移位	03	1	1
RRC A	累加器连进位标志右循环移位	13	1	1
SWAP A	累加器高低半字节互换	C4	1	1
数据传送				
MOV A,Rn	寄存器送累加器	E8-EF	1	1
MOV A,direct	直接字节送累加器	E5	2	2
MOV A,@Ri	间接 RAM 送累加器	E6-E7	1	2
MOV A,#data	直接数据送累加器	74	2	2
MOV Rn,A	累加器送寄存器	F8-FF	1	2
MOV Rn,direct	直接字节送寄存器	A8-AF	2	4
MOV Rn,#data	直接数据送寄存器	78-7F	2	2
MOV direct,A	累加器送直接字节	F5	2	3
MOV direct,Rn	寄存器送直接字节	88-8F	2	3

MOV direct1,direct2	直接字节 2 送直接字节 1	85	3	4
MOV direct,@Ri	间接 RAM 送直接字节	86-87	2	4
MOV direct,#data	直接数据送直接字节	75	3	3
MOV @Ri,A	累加器送间接 RAM	F6-F7	1	3
MOV @Ri,direct	直接字节送间接 RAM	A6-A7	2	5
MOV @Ri,#data	直接数据送间接 RAM	76-77	2	3
MOV DPTR,#data 16	16 位常数送数据指针	90	3	3
MOVC A,@A+DPTR	代码字节送累加器 (DPTR 为基址)	93	1	3
MOVC A,@A+PC	代码字节送累加器 (PC 为基址)	83	1	3
MOVX A,@Ri	外部 RAM 送累加器 (8 位地址)	E2-E3	1	3
MOVX A,@DPTR	外部 RAM 送累加器 (16 位地址)	E0	1	3
MOVX @Ri,A	累加器送外部 RAM (8 位地址)	F2-F3	1	4
MOVX @DPTR,A	累加器送外部 RAM (16 位地址)	F0	1	4
PUSH direct	直接字节压入栈	C0	2	4
POP direct	栈弹出直接字节	D0	2	3
XCH A,Rn	累加器与寄存器交换	C8-CF	1	2
XCH A,direct	累加器与直接字节交换	C5	2	3
XCH A,@Ri	累加器与间接 RAM 交换	C6-C7	1	3
XCHD A,@Ri	累加器与间接 RAM 低 4 位交换	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	子程序返回	22	1	4
RETI	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	相对短跳转	80	2	3
JMP @A+DPTR	相对长跳转	73	1	2
JZ rel	累加器为零跳转	60	2	3
JNZ rel	累加器非零跳转	70	2	3
JC rel	进位标志为 1 跳转	40	2	3
JNC	进位标志为 0 跳转	50	2	3
JB bit,rel	直接位为 1 跳转	20	3	4
JNB bit,rel	直接位为 0 跳转	30	3	4
JBC bit,direct rel	直接位为 1 跳转并清该位	10	3	4
CJNE A,direct rel	累加器与直接位不等跳转	B5	3	4
CJNE A,#data rel	累加器与直接数据不等跳转	B4	3	4
CJNE Rn,#data rel	寄存器与直接数据不等跳转	B8-BF	3	4
CJNE @Ri,#data rel	间接地址与直接数据不等跳转	B6-B7	3	4
DJNZ Rn,rel	寄存器减 1 不为零跳转	D8-DF	1	3
DJNZ direct,rel	直接字节减 1 不为零跳转	D5	3	4
NOP	空操作	00	1	1

布尔操作				
CLR C	进位标志清零	C3	1	1
CLR bit	直接位清零	C2	2	3
SETB C	进位标志置位	D3	1	1
SETB bit	直接位置位	D2	2	3
CPL C	进位标志取反	B3	1	1
CPL bit	直接位取反	B2	2	3
ANL C,bit	进位标志逻辑与直接位	82	2	2
ANL C,/bit	进位标志逻辑与直接位的反	B0	2	2
ORL C,bit	进位标志逻辑或直接位	72	2	2
ORL C,/bit	进位标志逻辑或直接位的反	A0	2	2
MOV C,bit	直接位送进位标志	A2	2	2
MOV bit,C	进位标志送直接位	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” =置为 0, “1” =置为 1, “x” =置为 0 或 1, “-” =无影响。

2.5 中断

CPU 有 18 个中断源。每个中断源有它自己的、位于一系列特殊功能寄存器中的中断请求标志。每个中断通过相应的标志请求可以单独使能或禁止。中断源的定义和中断向量见表 2-5。

中断分别组合为不同的、可以选择的优先级别。

中断使能寄存器的描述见 2.5.1 节，中断优先级设置的描述见 2.5.3 节。

2.5.1 中断屏蔽

每个中断可以通过中断使能特殊功能寄存器中的中断使能位 IEN0、IEN1 或 IEN2，使能或禁止。下面将描述 CPU 中断使能特殊功能寄存器，也可参见表 2-5。

某些外部设备会因为若干事件产生中断请求。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 RF 上。对于每个内部中断源对应的特殊功能寄存器，这些外部设备都有中断屏蔽位。

为了使能中断功能，应当执行下列步骤：

- 1 清除中断标志。
- 2.如果有，设置外部设备特殊功能寄存器中对应的各中断使能位。
- 3.设置寄存器 IEN0、IEN1 或 IEN2 中对应的各中断使能位为 1。
- 4.设置 IEN0 中的 EA 位为 1 来使能全局中断。
- 5.在该中断对应的向量地址上，运行该中断的服务程序。地址请见表 2-5。

图 2-4 给出了所有中断源和相关控制以及状态寄存器的完整概观。当调用中断服务程序时阴影框中的中断标志将被硬件自动清除。□ 表示触发，可能是因为电平源也可能是因为边缘形成。中断失去了它，它们将被当作一个电平触发（适用于端口 0，端口 1 和端口 2）。

转换器显示为默认状态，□ 和 □ 表示上升或下降沿检测，即在什么时候中断产生。作为脉冲或边缘型中断的一般规则，应当在清除源标志位之前清除 CPU 中断标志寄存器。如果可用，标志是不会自动清除的。对于电平源必须在清除 CPU 标志前清除源。

表 2-5 中断

中断号	描述	中断名称	中断向量	中断屏蔽，CPU	中断标志，CPU
0	RF 发送先进先出队列空，或 RF 接收先进先出队列满	RFERR	03h	IEN0. RFERRIE	TCON. RFERRIF ⁽¹⁾
1	ADC 转换结束	ADC	0Bh	IEN0. ADCIE	TCON. ADCIF ⁽¹⁾
2	USART0 接收完成	URX0	13h	IEN0. URX0IE	TCON. URX0IF ⁽¹⁾
3	USART1 接收完成	URX1	1Bh	IEN0. URX1IE	TCON. URX1IF ⁽¹⁾
4	AES 加密/解密完成	ENC	23h	IEN0. ENCIE	S0CON. ENCIF
5	睡眠定时器比较	ST	2Bh	IEN0. STIE	IRCON. STIF
6	端口 2 输入/USB	P2INT	33h	IEN2. P2IE	IRCON2. P2IF ⁽²⁾
7	USART0 发送完成	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) 比较/溢出	T3	5Bh	IEN1. T3IE	IRCON. T3IF ⁽¹⁾⁽²⁾
12	定时器 4 (8 位) 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	USART1 发送完成	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	看门狗定时溢出	WDT	8Bh	IEN2. WDTIE	IRCON2. WDTIF

- (1) 当调用中断服务程序时硬件清除。
(2) 存在的额外的中断屏蔽和中断标志。

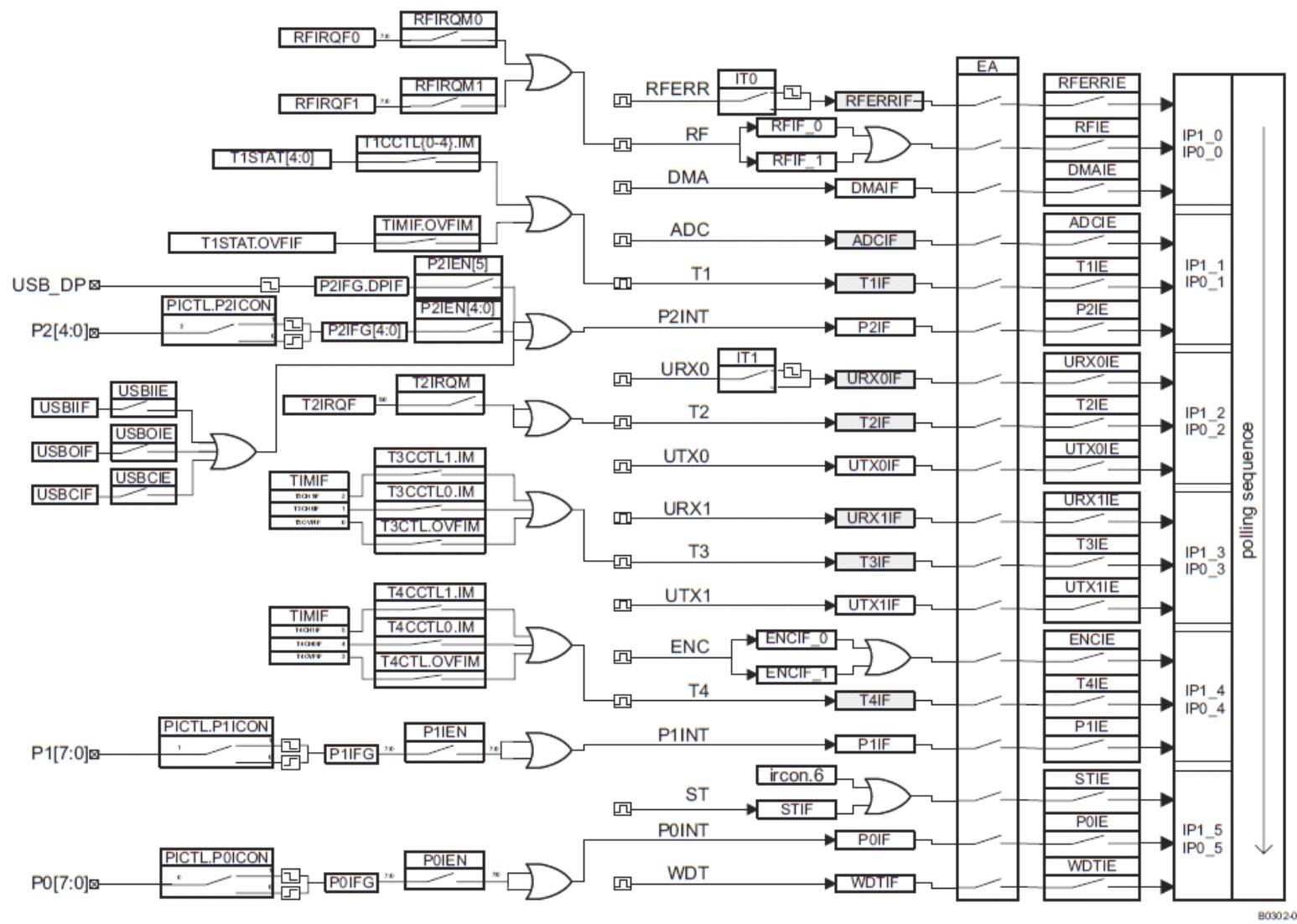


图 2-4 中断概览

IEN0 (0xA8) —— 中断使能 0

位	名称	复位	读/写	描述
7	EA	0	R/W	禁止所有中断 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或禁止
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	USART1 接收中断使能 0: 中断禁止 1: 中断使能
2	URX0IE	0	R/W	USART0 接收中断使能 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	RFERIE	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	读/写	描述
7: 6	—	00	R0	不使用, 读出来是 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	读/写	描述
7: 6	—	00	R0	不使用, 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	USART1 发送中断使能 0: 中断禁止 1: 中断使能
2	UTX0IE	0	R/W	USART0 发送中断使能 0: 中断禁止 1: 中断使能
1	P2IE	0	R/W	端口 2 和 USB 中断使能 0: 中断禁止 1: 中断使能
0	RFIE	0	R/W	RF 通用中断使能 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。当中断使能时, 首先设置中断标志, 然后在下一个指令周期, 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。

新中断的响应, 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。中断响应的时间取决于当前的指令, 最快的为 7 个机器指令周期。其中, 一个机器指令周期用于检测中断, 其余 6 个用来执行 LCALL。

注意: 如果一个中断被禁止且中断标志被轮询, 8051 汇编指令 JBC 不得用于轮询中断标志, 并且当它置位时要清除它。如果使用 JBC 来轮询中断标志, 中断标志可能立即重新生效。

TCON (0x88) —— 中断标志

位	名称	复位	读/写	描述
7	URX1IF	0	R/W H0	USART1 接收中断标志。当 USART1 接收中断产生时置为 1，当 CPU 指向中断服务程序时清除。 0: 中断未挂起 1: 中断挂起
6	—	0	R/W	不使用
5	ADCIF	0	R/W H0	ADC 中断标志。当 ADC 中断产生时置为 1，当 CPU 指向中断服务程序时清除。 0: 中断未挂起 1: 中断挂起
4	—	0	R/W	不使用
3	URX0IF	0	R/W H0	USART0 接收中断标志。当 USART0 接收中断产生时置为 1，当 CPU 指向中断服务程序时清除。 0: 中断未挂起 1: 中断挂起
2	IT1	1	R/W	保留。必须始终置为 1。置为 0 将使能低电平中断检测，这将始终保持（当中断请求被发起时触发）。
1	RFERRIF	0	R/W H0	RF TX/RX FIFO 中断标志。当 RFERR 中断产生时置为 1，当 CPU 指向中断服务程序时清除。 0: 中断未挂起 1: 中断挂起
0	IT0	1	R/W	保留。必须始终置为 1，置为 0 将使能低电平中断检测，这将始终保持（当中断请求被发起时触发）。

S0CON (0x98) —— 中断标志 2

位	名称	复位	读/写	描述
7: 2	—	0000 00	R/W	不使用
1	ENCIF_1	0	R/W	AES 中断。ENC 有两个中断标志，ENCIF_1 和 INCIF_0。设置其中一个将请求中断服务。当 AES 预处理请求中断时两个标志都应设置。 0: 中断未挂起 1: 中断挂起
0	ENCIF_0	0	R/W	AES 中断。ENC 有两个中断标志，ENCIF_1 和 INCIF_0。设置其中一个将请求中断服务。当 AES 预处理请求中断时两个标志都应设置。 0: 中断未挂起 1: 中断挂起

S1CON (0x9B) —— 中断标志 3

位	名称	复位	读/写	描述
7: 2	—	0000 00	R/W	不使用
1	RFIF_1	0	R/W	RF 产生中断。RF 有两个中断标志，RFIF_1 和 RFIF_0。设置其中

				一个将请求中断服务。当射频请求中断时两个标志都应设置。 0: 中断未挂起 1: 中断挂起
0	RFIF_0	0	R/W	RF 产生中断。RF 有两个中断标志，RFIF_1 和 RFIF_0。设置其中一个将请求中断服务。当射频请求中断时两个标志都应设置。 0: 中断未挂起 1: 中断挂起

IRCON (0xC0) —— 中断标志 4

位	名称	复位	读/写	描述
7	STIF	0	R/W	睡眠定时器中断标志。 0: 中断未挂起 1: 中断挂起
6	—	0	R/W	必须写 0。写 1 将总是使能中断源。
5	P0IF	0	R/W	端口 0 中断标志。 0: 中断未挂起 1: 中断挂起
4	T4IF	0	R/W H0	定时器 4 中断标志。当定时器 4 中断产生时置为 1，当 CPU 指向中断服务程序时清除。 0: 中断未挂起 1: 中断挂起
3	T3IF	0	R/W H0	定时器 3 中断标志。当定时器 3 中断产生时置为 1，当 CPU 指向中断服务程序时清除。 0: 中断未挂起 1: 中断挂起
2	T2IF	0	R/W H0	定时器 2 中断标志。当定时器 2 中断产生时置为 1，当 CPU 指向中断服务程序时清除。 0: 中断未挂起 1: 中断挂起
1	T1IF	0	R/W H0	定时器 1 中断标志。当定时器 1 中断产生时置为 1，当 CPU 指向中断服务程序时清除。 0: 中断未挂起 1: 中断挂起
0	DMAIF	0	R/W	DMA 完成中断标志。 0: 中断未挂起 1: 中断挂起

IRCON2 (0xE8) —— 中断标志 5

位	名称	复位	读/写	描述
7: 5	—	000	R/W	不使用。
4	WDTIF	0	R/W	看门狗定时器中断标志。 0: 中断未挂起 1: 中断挂起

3	P1IF	0	R/W	端口 1 中断标志。 0: 中断未挂起 1: 中断挂起
2	UTX1IF	0	R/W	USART1 发送中断标志。 0: 中断未挂起 1: 中断挂起
1	UTX0IF	0	R/W	USART0 发送中断标志。 0: 中断未挂起 1: 中断挂起
0	P2IF	0	R/W	端口 2 中断标志。 0: 中断未挂起 1: 中断挂起

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 所列。

中断优先组及其赋值的中断源如表 2-7 所列。每组赋值为 4 个中断优先级之一。当进行中断服务请求时，不允许被同级或较低级别的中断打断。

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 所列的响应顺序来判定哪个中断优先响应。请注意，图 2-4 中的响应顺序是基于表 2-8 中的算法，而不是如图中所列的 IP 位之间的查询。

IP1 (0xB9) — 中断优先级 1

位	名称	复位	读/写	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。参看表 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。参看表 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。参看表 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。参看表 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。参看表 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。参看表 2-7：中断优先级组。

IP0 (0xA9) — 中断优先级 0

位	名称	复位	读/写	描述
7: 6	—	00	R/W	不使用
5	IP0_IPG5	0	R/W	中断第 5 组，优先级控制位 0。参看表 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。参看表 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。参看表 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。参看表 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。参看表 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。参看表 2-7：中断优先级组。

表 2-6 优先级的设置

IP1_x	IP0_x	优先级
0	0	0—最低
0	1	1
1	0	2
1	1	3—最高

表 2-7 中断优先级组

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 相同优先级中断的响应顺序

中断编号	中断名称	响应顺序
0	RFERR	
16	RF	
8	DMA	
1	ADC	
9	T1	
2	URX0	
10	T2	
3	URX1	
11	T3	
4	ENC	
12	T4	
5	ST	
13	P0INT	
6	P2INT	
7	UTX0	
14	UTX1	
15	P1INT	
17	WDT	



3 调试接口

两线调试接口允许对片上 Flash 进行编程，提供对存储器和寄存器内容的访问，还具有断点、单步和寄存器修改等调试特性。

在调试模式下，调试接口使用 I/O 引脚 P2.1 作为调试数据线，使用 P2.2 作为调试时钟线。只有当设备不处于调试模式下时这些 I/O 引脚可以被作为通用 I/O。因此调试接口不会对任何外设 I/O 引脚造成干扰。

3.1 调试模式

当 RESET_N 输入保持为低时，通过在引脚 P2.2（调试时钟）上强制产生 2 个下降沿即进入调试模式。

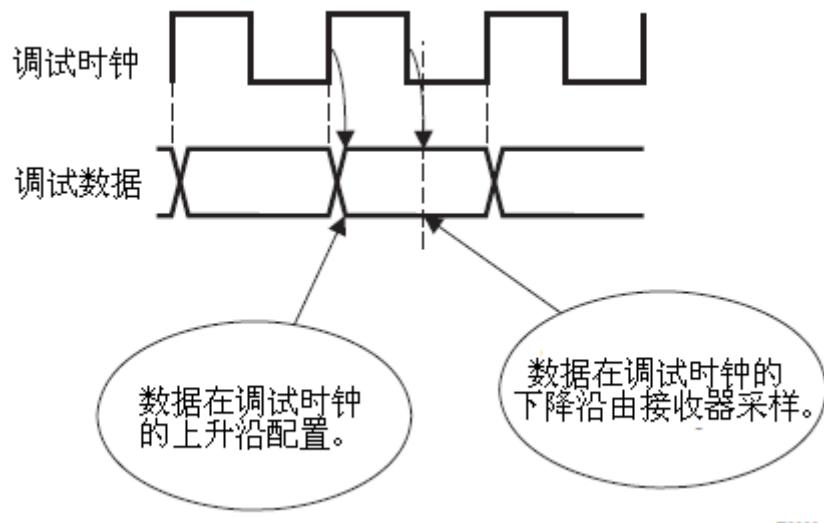
当处于调试模式时引脚 P2.1 是调试数据双向引脚，P2.2 是调试时钟输入引脚。

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，如果 CLKCONCMD.OSC=0，CLKCONCMD.CLKSPD 则应当设置为 000；如果 CLKCONCMD.OSC=1，CLKCONCMD.CLKSPD 则应当设置为 001。

3.2 调试通信

调试接口使用一个两线接口的 SPI，它由 P2.1（调试数据）和 P2.2（调试时钟）引脚组成。数据被驱动在双向调试数据引脚上，在调试时钟的正边沿，在调试时钟的负边沿采样。

调试数据引脚的方向取决于发出的命令。数据被驱动在调试时钟的正边沿，在调试时钟的负边沿采样。图 3-1 显示了数据采样。



T0302-01

图 3-1 外部调试接口时序

数据是以字节为定向的，首先以高位字节传送。一个字节的序列如图 3-2 所示。

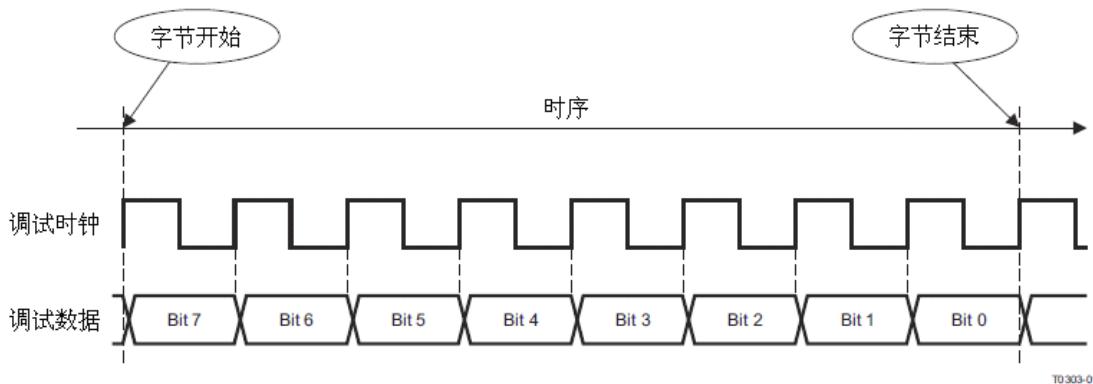


图 3-2 一个字节的传输

调试命令序列总是以主机通过串行接口发送一个命令开始。该命令对包含有紧跟的更多参数的字节数，以及是否需要一个响应进行编码。基于这个命令，调试模块控制调试数据焊盘的方向。图 3-3 显示了一个典型的命令序列。注意，为了使图清晰，简化了调试数据信号，没有显示每个位的变化。方向没有明确表示给外部，但是必须从该命令协议的主机产生。

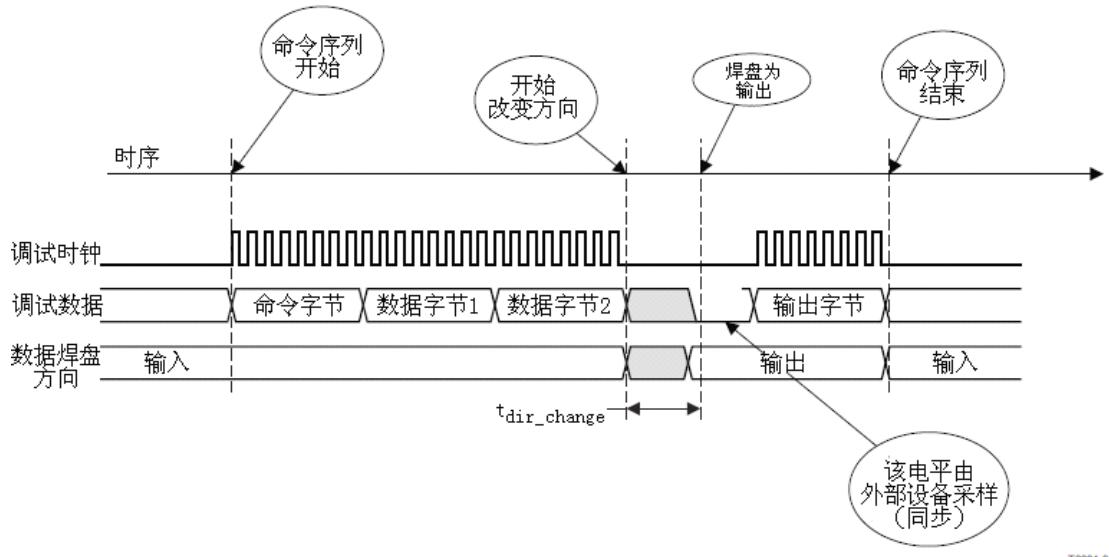


图 3-3 典型命令序列—无额外的响应等待

对于需要响应的命令，在命令和响应之间必须有一个短暂的空闲期，以允许焊盘改变方向。在最小等待时间 (t_{dir_change}) 过后，芯片通过下拉数据焊盘为低电平，来表示它是否准备好了传送响应数据。采样数据焊盘的外部调试器检测到该焊盘为低电平时，就开始时钟输出响应数据。如果数据焊盘在等待时间过后为高电平，表示芯片还未准备好传送响应数据。图 3-4 显示了等待响应的过程。

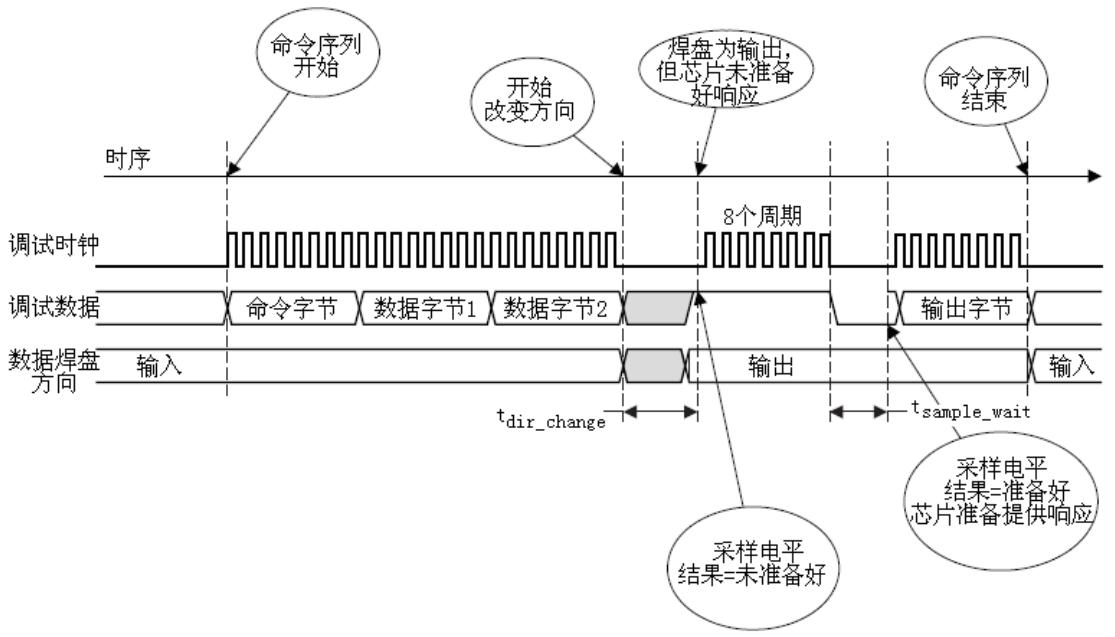


图 3-4 典型命令序列—等待响应

如果调试接口的数据线拉高，表示它还没准备好返回数据，外部设备在重新采样准备好的电平之前，必须精确地发出 8 个时钟脉冲。在变为低电平之前必须一直重复。等待周期相当于从调试接口读取一个字节，但是忽略结果。注意焊盘在调试时钟的下降沿开始改变方向。因此，焊盘驱动器在程序设计器对驱动器进行驱动，直到用户改变焊盘方向。在程序执行中应尽量使这个持续时间最短。

3.3 调试命令

调试命令如表 3-2 所示。一些调试命令将在下面的小节中进行详细描述。

最低的 3 位 (X) 可以为任意值。

3.4 锁定位

为了软件和/或存取保护，可以写入一组锁定位到较高的可用 Flash 页—锁定位页。锁定位的结构包括 FLASH_PAGE-1 锁定位和紧随其后的一个调试锁定位（见表 3-1）。该结构开始于锁定位页的地址 2032，多达 16 字节。锁定位页的其余部分（地址 0-2031）可以用来存储代码/常量，但是，如果没有进入调试调试就不能被修改。

锁定保护位 PAGELOCK[FLASH_PAGE-2: 0]用于为独立的 Flash 存储器页 (2KB) 使能擦除/写保护。每个可用页有 1 个位来控制。

当调试锁定位 DBGLOCK 设置为 0 时（见表 3-1），除了 CHIP_ERASE, READ_STATUS 和 GET_CHIP_ID，其它所有的调试命令都禁止。通过 READ_STATUS 命令来读取调试锁定位的状态（见 3.4.2 节）。

注意，由于写锁定位页或使用 CHIP_ERASE 命令，使得调试锁定位被改变后，必须复位芯片来锁定/解锁调试接口。

只能通过 CHIP_ERASE 命令来清除调试锁定位，从而解锁调试接口。

表 3-1 定义了包含 Flash 锁定保护位的 16 字节的结构。第一个字节的位 0 包含页 0 的锁定位，第一个字节的位 1 包含页 1 的锁定位，以此类推。字节 (FLASH_PAGE/8) —1 的位 7 是 DBGLOCK 位。

表 3-1 Flash 锁定保护位结构定义

位	名称	描述
FLASH_PAGES—1	DBGLOCK	调试锁定位 0: 禁止调试命令 1: 使能调试命令
FALSEH_PAGES—2: 0	PAGELOCK[FLASH_PAGES—1: 0]	页锁定位。多达 128 个页，每一个页都有一个位。对于不可用页，页锁定位不使用。 0: 页被锁定 1: 页未被锁定

3.4.1 调试配置

命令 WR_CONFIG 和 RD_CONFIG 用来访问调试配置数据字节。关于配置数据的格式和描述见表 3-3。

3.4.2 调试状态

使用 READ_STATUS 命令来读取调试状态字节。调试状态的格式和描述见表 3-4。

READ_STATUS 命令用于例如：

- 查询在 CHIP_ERASE 命令后芯片擦除的状态 (CHIP_ERASE_BUSY)
- 检查振荡器的稳定状态 (OSCILLATOR_STABLE); 所需的调试命令 HALT, RESUME, DEBUG_INSTR, STEP_REPLACE 和 STEP_INSTR.

表 3-2 调试命令

命令	指令	输入字节	输出字节	描述
CHIP_ERASE	00010XXX	0	1	执行 Flash 芯片擦除 (大量擦除) 并清除锁定位。如果除了 READ_STATUS 外的其它命令被执行，那么 CHIP_ERASE 被禁止。
WR_CONFIG	00011XXX	1	1	写配置数据，参见表 3-3。
RD_CONFIG	00100XXX	0	1	读配置数据，由 WR_CONFIG 命令设置返回值。
GET_PC	00101XXX	0	2	16 位编程计数器的返回值。不论指令代码的位 2 为何值都返回 2 个字节。
READ_STATUS	00110XXX	0	1	读状态字节，参见表 3-4。
SET_HW_BRKPNT	00111XXX	3	1	设置硬件断点。
HALT	01000XXX	0	1	停止 CPU 运行。
RESUME	01001XXX	0	1	恢复 CPU 运行。CPU 必须是处于停止运行状态，才能运行该命令。
DEBUG_INSTR	01010Xyy	1-3	1	运行调试命令。所提供的指令将由不带有程序计数器

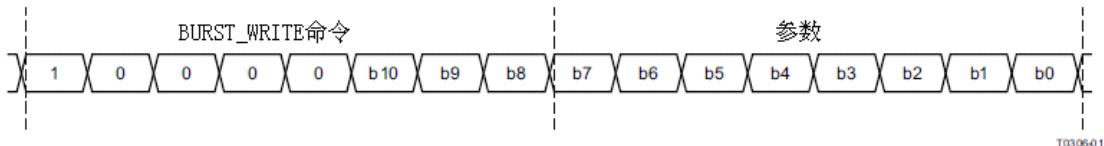
				的递增的 CPU 执行。CPU 必须处于停止运行状态才能运行该命令。注意 yy 是该命令字节后面的字节数，即有多少字节的 CPU 指令（见表 2-3）。
STEP_INSTR	01011XXX	0	1	下一步 CPU 指令。CPU 将执行接下来的来自于程序存储器的指令，并且执行后程序计数器将增加。CPU 必须处于停止运行状态才能运行该命令。
GET_BM	01100XXX	0	1	该命令与 GET_PC 命令的功能一样，但是它返回存储器 bank。它返回 1 个字节，其中最低 3 位为当前使用的存储器 bank。
GET_CHIP_ID	01101XXX	0	2	16 位芯片 ID 和版本号码的返回值。不论指令代码的位 2 为何值都返回 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。每当寄存器更新，都将产生一个 DBG_BW DMA 触发。 BURST_WRITE 命令的参数数量是可变的。突发脉冲的字节个数用命令字节的最后 3 位 (kkk) 和整个下一个字节表示。命令序列如图 3-5。突发脉冲的长度由一个 11 位值 (b10-b0) 表示。值 0 表示 2048；因此，传输的最小字节数为 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	该位置 1 时，数字稳压器在 PM2 和 PM3 期间不关闭。如果清除该位，在 PM2 和 PM3 期间调试接口被复位。
4	—	0	保留
3	TIMERS_OFF	0	禁用定时器。禁止定时器运行。忽略 TIMER_SUSPEND 位和它的功能。 0：不禁用定时器 1：禁用定时器
2	DMA_PAUSE	1	DMA 暂停。当该位为 1 时不能存取 DMA 寄存器。 0：使能 DMA 收发器 1：暂停所有的 DMA 收发器
1	TIMER_SUSPEND		暂停定时器。 当芯片停止运行时暂停定时器。在调试指令期间，定时器也将被暂停。如果程序是自由运行，但执行一个 STEP 命令时，定时器精确（或尽量接近）接收尽可能多的 tick。 0：不暂停定时器 1：暂停定时器
0	—	0	未使用。总是写 0。

表 3-4 调试状态

位	名称	描述
7	CHIP_ERASE_BUSY	Flash 芯片擦除忙。 正在进行芯片擦除时，该信号只能为高。接收到 CHIP_ERASE 命令后立即变高，Flash 完全擦除后变为低。 0: — 1: 正在进行芯片擦除
6	PCON_IDLE	PCON 空闲 0: CPU 正在运行 1: CPU 空闲（时钟门控）
5	CPU_HALTED	CPU 停止 0: CPU 运行中 1: CPU 停止
4	POWER_MODE_0	功耗模式 0 0: 选择了功耗模式 1-3 1: 选择了功耗模式 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 通过硬件断点停止
2	DEBUG_LOCKED	调试锁定。返回 DBGLOCK 位的值。 0: 调试接口未被锁定 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 在 DATA 存储器空间地址 0xFF 写数据时，该位指示这可能是一个堆栈溢出。 0: 没有堆栈溢出 1: 堆栈溢出


图 3-5 突发脉冲写命令（头 2 个字节）

3.4.3 硬件断点

调试命令 SET_HW_BRKPNT 用来设置一个硬件断点。CC253x 支持最多 4 个硬件断点。当一个硬件断点使能时它将比较 CPU 地址总线和断点。当这 2 个匹配时，CPU 停止。

当使用 SET_HW_BRKPNT 时，外部主机必须提供 3 个数据字节来定义硬件断点。硬件断点本身由 18 位组成，其中 3 位用于控制。SET_HW_BRKPNT 命令的 3 个数据字节格式

如下：

第一个数据字节包含：

- Bits 7-6：未用到
- Bits5-4：断点号码；0-3
- Bit 3：1 = 使能，0 = 禁止
- Bits2-0：存储器 bank 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。因此第二个和第三个数据字节设置 CPU CODE 地址来停止执行。

3.4.4 Flash 编程

通过调试接口来对片上 Flash 进行编程。外部主机必须首先使用 DEBUG_INSTR 调试命令发送指令来进行 Flash 控制器的 Flash 编程。

3.5 调试接口和功耗模式

当芯片处于调试模式下，功耗模式 PM2 和 PM3 可以以两种不同的方式处理。默认行为是不关闭数字稳压器。这样在保持调试模式运行的同时模拟了功耗模式。和在普通功耗模式下一样，时钟源关闭。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，即禁用调试模式。当芯片处于调试模式下，由位 5 (SOFT_POWER_MODE) 来控制这种选择。

不管处于哪种功耗模式下，调试接口仍然响应一组精简命令。通过发出一个 HALT 命令给调试接口来使芯片从功耗模式中唤醒。HALT 命令使芯片从功耗模式中唤醒，处于停止运行状态。因此，必须再发送一个 RESUME 命令来恢复软件运行。

在功耗模式下可以读取调试状态。发出一个 HALT 命令使芯片离开功耗模式时必须检查调试状态。上电所需的时间取决于芯片处于哪种功耗模式，且必须在调试状态下检查。在芯片运行之前，调试接口只接受功耗模式中可用的命令。

注意：不支持空闲模式下的调试。建议在调试时使用主动模式或其它的功耗模式。

3.6 寄存器

DBGDATA (0x6260) — 调试数据

位	名称	复位	读/写	描述
7: 0	BYTE[7: 0]	0	R	来自 BURST_WRITE 命令的调试数据。 每当使用 BURST_WRITE 命令将一个新的字节传送到调试接口时，该寄存器就被更新。该寄存器一旦被更新就产生一个 DBG_BW DMA 触发。这使得 DMA 控制器可以获取数据。

CHVER (0x6249) —芯片版本

位	名称	复位	读/写	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	读/写	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	读/写	描述
7	—	0	R0	未使用。总为 0。
6: 4	FLASHSIZE[2: 0]	取决于芯片	R	Flash 大小。 001: 32KB 010: 64KB 011: 128KB 100: 256KB
3	USB	取决于芯片	R	如果芯片有 USB 为 1, 无 USB 则为 0
2	—	1	R1	未使用。总为 1。
1: 0	—	00	R0	未使用。总为 0。

CHIPINFO1 (0x6277) —芯片信息字节 1

位	名称	复位	读/写	描述
7: 3	—	0000 0	R0	未使用。总为 0。
2: 0	SRAMSIZE[2: 0]	取决于芯片	R	SRAM 大小, 以 KB 大小减 1。例如, 一个 4KB 的芯片, 该字段设置为 011, 那么这个数加 1 就得到有多少 KB 可用。

4 电源管理与时钟

通过不同的运行模式（功耗模式）来使能低功耗运行。各种运行模式指主动模式、空闲模式和功耗模式 1、2、3（PM0-PM3）。通过关闭对模块的供电避免静态（泄漏）电源消耗来实现超低功耗运行，也可以通过使用门控时钟和关闭振荡器降低动态功耗来实现超低功耗运行。

4.1 电源管理说明

使用不同的工作模式或者说功耗模式以达到低功耗运行。超低功耗运行可以通过关闭对模块的供电来避免静态（泄漏）电源消耗，也可以通过使用门控时钟和关闭振荡器来降低动态功耗。

枚举的各种工作模式（功耗模式）被指定为主动模式，空闲模式，PM0，PM1，PM2 和 PM3。主动模式是正常运行，而 PM3 具有最低的功耗。不同功耗模式对系统运行的影响以及电源调节器和振荡器选项请见表 4-1。

表 4-1 功耗模式

功耗模式	高频振荡器	低频振荡器	电源稳压器（数字）
配置	A 32MHz 晶体振荡器 B 16MHz RC 振荡器	C 32kHz 晶体振荡器 D 32.kHz RC 振荡器	
主动/空闲模式	A 或者 B	C 或者 D	开
PM1	无	C 或者 D	开
PM2	无	C 或者 D	关
PM3	无	无	关

主动模式：全功能模式。数字核心的电源稳压器打开，16MHz RC 振荡器或者 32MHz 晶体振荡器运行，或者 2 个振荡器都运行。32kHz RC 振荡器或者 32kHz 晶体振荡器运行。

空闲模式：除了 CPU 核心停止运行（空闲），其他和主动模式相同。

PM1：数字部分的电源稳压器打开。32MHz 晶体振荡器和 16MHz RC 振荡器都不运行。32kHz RC 振荡器或者 32kHz 晶体振荡器运行。复位、一个外部中断或者睡眠定时器到期的情况下系统都将进入主动模式。

PM2：数字核心的电源稳压器关闭。32MHz 晶体振荡器和 16MHz RC 振荡器都不运行。32kHz RC 振荡器或者 32kHz 晶体振荡器运行。复位、一个外部中断或者睡眠定时器到期的情况下系统都将进入主动模式。

PM3：数字核心的电源稳压器关闭。没有振荡器运行。复位或者一个外部中断的情况下系统都将进入主动模式。

PM2/PM3 模式下，上电复位有效，掉电检测关闭，也就给定了一个有限的电压管理。在 PM2/PM3 期间，如果供电电压低于 1.4V，温度高于或等于 70°C，在重新进入主动模式之前，应当提供一个合适的运行电压，在 PM2/PM3 模式下保存的寄存器和 RAM 里的内容可能会改变。因此，应当注意系统的电源设计，以确保这种情况不会发生。通过进入主动模式可以对电压进行精确的定期监控，因为如果点与低于 1.7V 左右就会触发一个掉电检测复位。

4.1.1 主动和空闲模式

主动模式是工作的全功能模式，CPU、外围设备和RF收发器都运行。数字电源稳压器打开。

主动模式用于正常工作。在主动模式下(SLEEPCMD.MODE=0x00)，使能 PCON.IDLE 位，CPU 内核从运行变为停止并进入空闲模式。所有其他外围设备的功能正常，任何使能的中断将唤醒 CPU 内核（从空闲模式转回到主动模式）。

4.1.2 PM1

在 PM1，高频振荡器（32MHz 晶体振荡器和 16MHz RC 振荡器）断电。电源稳压器和使能的 32kHz 振荡器开启。当进入了 PM1，运行一个断电序列。

因为 PM1 使用了一个快速断电/上电序列，等待唤醒事件与期望时间相对较短（小于 3ms）时使用 PM1。

4.1.3 PM2

PM2 具有第二低的低功耗。在 PM2，上电复位、外部中断、选定的 32kHz 振荡器和睡眠定时器这些外围设备运行。在进入 PM2 之前，I/O 引脚保持 I/O 模式且置位输出值。其它的内部电路断电。电源稳压器也关闭。当进入了 PM2，运行一个断电序列。

当使用睡眠定时器作为唤醒事件，并与外部中断相结合时，通常进入 PM2。当睡眠时间超过 3ms 时，相对于 PM1，PM2 可以当作典型选择。和使用 PM1 相比，使用较短的睡眠时间不会降低系统功耗。

4.1.4 PM3

PM3 用来实现最低功耗工作模式。在 PM3，所有从电源稳压器取电的内部电路都关闭（基本上所有的数字模块；只有中断检测和 POR 电平感应除外）。内部电源稳压器和所有的振荡器也都关闭。

复位（POR 或者外部）和外部 I/O 口中断是在该模式下工作的唯一功能。在进入 PM3 之前，I/O 引脚保持 I/O 模式且置位输出值。一个复位条件或者使能的外部 I/O 中断事件将唤醒设备且让它进入主动模式（当复位返回开始执行程序时，外部中断将从它进入 PM3 的地方开始）。在该模式 RAM 的内容完全保留，寄存器的内容被部分保留（见 4.6 节）。PM3 使用和 PM2 一样的断电/上电序列。

在等待一个外部事件时 PM3 用来实现超低功耗。当期望的睡眠时间超过 3ms 时使用该模式。

4.2 电源管理控制

所需要的功耗模式通过在 SLEEPCMD 控制寄存器里的 MODE 位和 PCON.IDLE 位来选

择。设置 SFR 寄存器的 PCON.IDLE 位，进入由 SLEEP CMD.MODE 选定的模式。

一个来自端口引脚或睡眠定时器的使能中断，或者上电复位的使能中断将使设备从其它功耗模式唤醒并进入主动模式。

进入 PM1, PM2 或 PM3 后就运行一个掉电序列。当设备离开 PM1, PM2 或者 PM3 时，它从 16MHz 运行，进入功耗模式时（设置 PCON.IDLE）如果 CLKCONCMD.OSC=0，它就自动变为 32MHz。在进入功耗模式时如果设置了 PCON.IDLE 且 CLKCONCMD.OSC=1，它就继续运行于 16MHz。

为了正确运行，设置 PCON.IDLE 位的指令必须与特定的方式对齐。紧跟这条指令后面的汇编指令的第一个字节绝对不能位于 4 字节边界。此外，不能禁用缓存（见 FCTL 寄存器描述的 CM）。不遵守这个规则可能产生较高的电流消耗。满足了这个要求，在设置 PCON.IDLE 位这条指令后的第一条汇编指令，在系统唤醒之后、导致系统唤醒的中断的中断服务进程之前被执行。如果该指令是一个全局中断禁用，在唤醒后，它后面必须紧跟执行代码，但是在中断服务进程运行之前运行。

下面为在 IAR 编译器里面实现的例子。汇编代码中的一个函数中包含了设置 PCON 为 1 的命令。在一个 C 文件中调用该函数，使用这样一个声明：extern void EnterSleepModeDisableInterruptsOnWakeup(void);。RSEG NEAR_CODE:CODE:NOROOT(2) 语句确保 MOV PCON,#1 指令位于 2 字节边界。它是一个 3 字节指令，因此要求接下来的指令不能位于 4 字节边界。在下面的例子中，这个指令是 CLR EA，它禁用所有的中断。这意味着，在 IEN0.EA 位在代码中被再次置位之前，唤醒系统的中断的中断服务进程都不能执行。如果不想要此功能，那么可以用 NOP 来代替 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

本节描述电源管理寄存器。在进入 PM2 或者 PM3 时所有寄存器位保持其之前的值。

PCON (0x87) — 功耗模式控制

位	名称	复位	读/写	描述
7: 1	—	0000 000	R/W	不使用，总是写为 0000 000。
0	IDLE	0	R0/W H0	功耗模式控制。将该位写 1 强制设备进入由 SLEEP CMD.MODE 设置的功耗模式（注意 MODE=0x00 且 IDLE=1 将停止 CPU 内核）。该位总是读为 0。当运行时所有的使能中断将清除该位，而设备也将重新进入主动模式。

SLEEPCMD (0xBE) —睡眠模式控制命令

位	名称	复位	读/写	描述
7	OSC32K_CALDIS	0	R/W	禁止 32kHz RC 振荡器校准 0: 32kHz RC 振荡器校准使能 1: 32kHz RC 振荡器校准禁止 可以随时对该位进行设置，但是直到 16MHz 高频 RC 振荡器被选作为系统时钟的时钟源，该位才能生效。
6: 3	—	000 0	R0	保留
2	—	1	R/W	保留。总是写为 1
1: 0	MODE[1: 0]	00	R/W	功耗模式设置 00: 主动/空闲模式 01: 功耗模式 1 10: 功耗模式 2 11: 功耗模式 3

SLEEPSTA (0x9D) —睡眠模式控制状态

位	名称	复位	读/写	描述
7	OSC32K_CALDIS	0	R	32kHz RC 振荡器校准状态 SLEEPSTA.OSC32K_CALDIS 显示了禁用的 32kHz RC 校准的当前状态。在芯片运行于 32kHz RC 振荡器之前，该位的值不能与 SLEEPCMD.OSC32K_CALDIS 的值相同。 可以随时对该位进行设置，但是直到 16MHz 高频 RC 振荡器被选作为系统时钟的时钟源，该位才能生效。
6: 5	—	00	R	保留
4: 3	RST[1: 0]	XX	R	表示最后复位的原因的状态位。如果有多个复位，寄存器将只包含最后事件。 00: 上电复位和掉电检测 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	32kHz 时钟信号（与系统时钟同步）

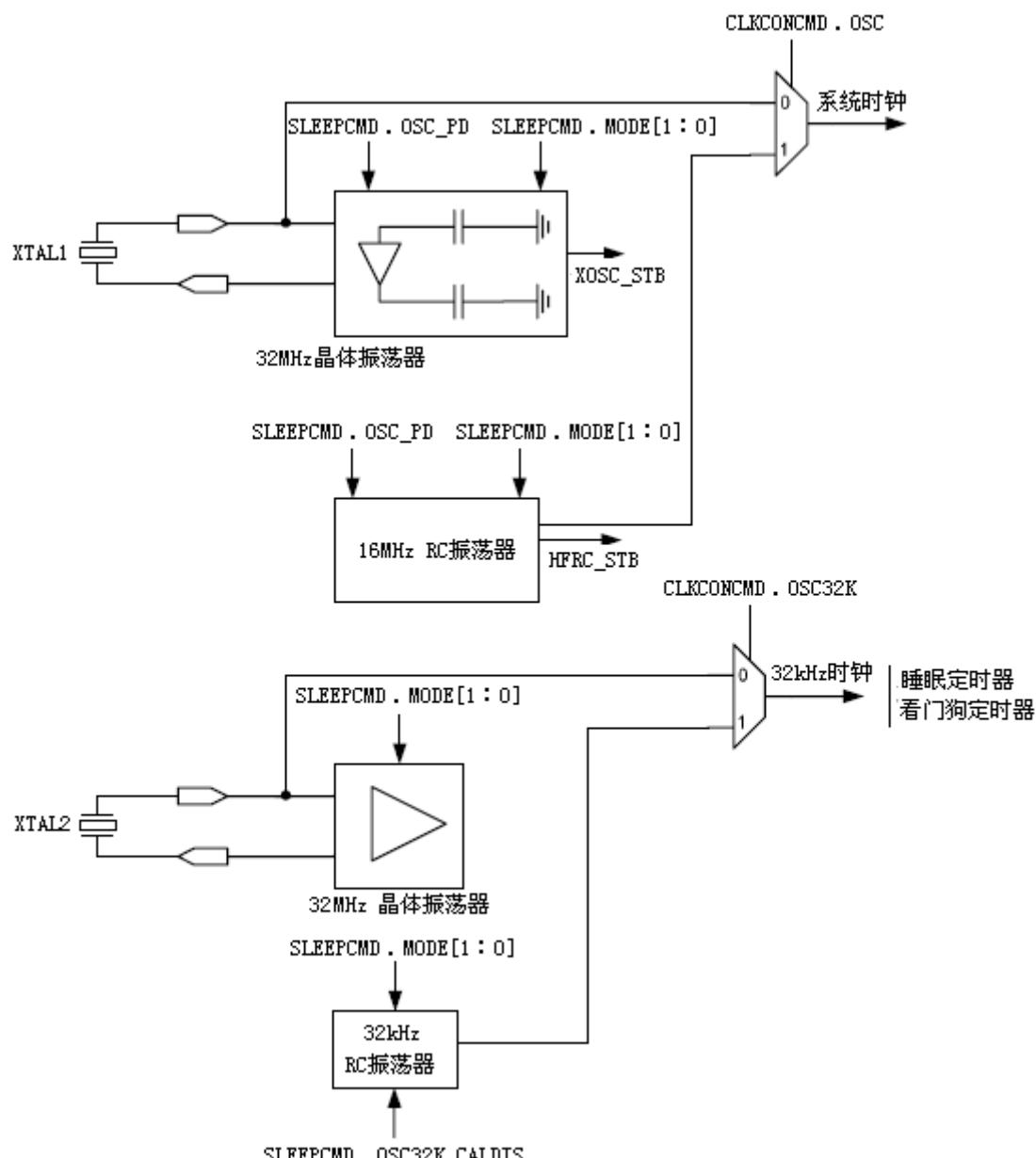


图 4-1 时钟系统概览

4.4 振荡器和时钟

CC253x 有一个内部系统时钟或主时钟。该时钟的振荡源既可以用 16MHz RC 振荡器，也可以采用 32MHz 晶体振荡器。时钟的控制可以由特殊功能寄存器 CLKCONCMD 来实现。

此外，还有一个 32kHz 时钟源也可以用 RC 振荡器或者晶体振荡器，也由 CLKCONCMD 寄存器控制。

寄存器 CLKCONSTA 是一个只读寄存器，用来获得当前时钟状态。

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 RC 振荡器。注意，运行 RF 收发器，必须使用 32MHz 晶体振荡器。

4.4.1 振荡器

图 4-1 给出了具有可用时钟源的时钟系统的概览。

设备中有 2 个高频振荡器：

- 32MHz 晶体振荡器
- 16MHz RC 振荡器

对于一些应用程序来说，32MHz 晶体振荡器的启动时间可能太长了，因此设备可以运行在 16MHz RC 振荡器，直到晶体振荡器处于稳定状态。16MHz RC 振荡器的功率比晶体振荡器要低，但因为它的精度不如晶体振荡器，所以它不能用于 RF 收发器运行。

设备中有 2 个低频振荡器：

- 32kHz 晶体振荡器
- 32kHz RC 振荡器

32kHz 晶体振荡器被设计为工作在 32.768kHz，并为要求精确时间的系统提供一个稳定的时钟信号。32kHz RC 振荡器工作在校准的 32.753kHz，只有当 32MHz 晶体振荡器启用时才能进行校准，校准可以通过使能 SLEEPCMD.OSC32K_CALDIS 位来禁止。对于 32kHz 晶体振荡器，32kHz RC 振荡器可以用于降低成本和功耗的解决方案。这两个 32kHz 振荡器不能同时运行。

4.4.2 系统时钟

系统时钟由选定的系统时钟源 32MHz 晶体振荡器或者 16MHz RC 振荡器而来。CLKCONCMD.OSC 位选择系统时钟源。请注意，使用 RF 收发器，必须选择 32MHz 晶体振荡器且它必须稳定。

请注意，CLKCONCMD.OSC 位的改变不会引起系统时钟瞬间发生改变。当 CLKCONSTA.OSC=CLKCONCMD.OSC 时，时钟源首先发生改变。因为对一个稳定时钟的要求优先于对时钟源的实际改变。另外请注意 CLKCONCMD.CLKSPD 位反映系统时钟的频率，因此它是 CLKCONCMD.OSC 位的镜像。

选择 32MHz 晶体振荡器为振荡源，且它稳定之后，即当 CLKCONSTA.OSC 位从 1 变为 0 时，就校准 16MHz RC 振荡器。

注意：从 16MHz 时钟源切换到 32MHz 时钟源（反之亦然），与 CLKCONCMD.TICKSPD 设置一致。当 CLKCONCMD.OSC 改变时，CLKCONCMD.TICKSPD 设置得较慢会导致实际源发生改变的时间较长。当 CLKCONCMD.TICKSPD 等于 000 时，转换时间最短。

4.4.3 32kHz 晶振

设备里有 2 个 32kHz 振荡器作为 32kHz 时钟的时钟源：

- 32kHz 晶体振荡器
- 32kHz RC 振荡器

默认情况下，复位后，32kHz RC 振荡器启用且被选为 32kHz 时钟源。RC 振荡器的功耗更低，但是不如 32kHz 晶体振荡器精确。32kHz 时钟源运行睡眠定时器，为看门狗定时器产生 tick，且当计算睡眠定时器的睡眠时间时它被作为定时器 2 的选通脉冲。通过

CLKCONCMD.OSC32K 寄存器位来选择哪一个振荡源作为 32kHz 振荡源。

可以随时设置寄存器位 CLKCONCMD.OSC32K，但是在 16MHz RC 振荡器作为系统时钟源之前都不起作用。当系统时钟从原来的 16MHz RC 振荡器变为 32MHz 晶体振荡器（CLKCONCMD.OSC 从 1 变为 0），如果选择了 32kHz RC 振荡器，就开始校准 32kHz RC 振荡器。校准期间，使用分频的 32MHz 晶体振荡器。校准的结果是 32kHz RC 振荡器运行在 32.753kHz。32kHz RC 振荡器校准最多可能需要 2ms 完成。通过设置寄存器位 SLEEPCMD.OSC32K_CALDIS 为 1 可以禁止校准。校准结束时，在 32kHz 时钟源上可能会产生一个多余的脉冲，而导致睡眠定时器增加 1。

注意：转换到 32kHz 晶体振荡器之后，当从 PM3 醒来且 32kHz 晶体振荡器使能时，振荡器需要长达 500ms 来稳定在正确的频率。在 32kHz 晶体振荡器稳定之前，睡眠定时器、看门狗定时器和时钟丢失探测器都不能使用。

4.4.4 振荡器和时钟寄存器

本节描述振荡器和时钟寄存器。除非另有说明，在进入 PM2 或者 PM3 时所有寄存器位保持其之前的值。

CLKCONCMD (0xC6) —时钟控制命令

位	名称	复位	读/写	描述
7	OSC32K	1	R/W	32kHz 时钟源选择。设置该位只是启动一个时钟源的更改。 CLKCONSTA.OSC32K 反应了当前的设置。当该位被改变了 16MHz 高频 RC 振荡器必须被选为系统时钟源。 0: 32kHz 晶体振荡器 1: 32kHz RC 振荡器
6	OSC	1	R/W	系统时钟源选择。设置该位只是启动一个时钟源的更改。 CLKCONSTA.OSC 反应了当前的设置。 0: 32MHz 晶体振荡器 1: 16MHz 高频 RC 振荡器
5: 3	TICKSPD[2: 0]	001	R/W	定时器 tick 输出设置。不能高于由 OSC 设置位设置的系统时钟设置。 000: 32MHz 001: 16MHz 010: 8MHz 011: 4MHz 100: 2MHz 101: 1MHz 110: 500kHz 111: 250kHz 注意 CLKCONCMD.TICKSPD 可以设置为任何值，但是其结果受 CLKCONCMD.OSC 设置的限制，即如果 CLKCONCMD.OSC=1，而 CLKCONCMD.TICKSPD=000，但是 CLKCONCMD.TICKSPD 读出为 001，实际 TICKSPD 是 16MHz。

2: 0	CLKSPD	001	R/W	<p>时钟速度。不能高于由 OSC 设置位设置的系统时钟设置。 指示当前系统时钟频率。</p> <p>000: 32MHz 001: 16MHz 010: 8MHz 011: 4MHz 100: 2MHz 101: 1MHz 110: 500kHz 111: 250kHz</p> <p>注意 CLKCONCMD.CLKSPD 可以设置为任何值，但是其结果受 CLKCONCMD.OSC 设置的限制，即如果 CLKCONCMD.OSC=1，而 CLKCONCMD.CLKSPD=000，但是 CLKCONCMD.CLKSPD 读出为 001，实际 CKKSPD 是 16MHz。</p> <p>还要注意调试器不能和一个划分过的系统时钟一起工作。当运行调试器时，如果 CLKCONCMD.OSC=0，那么 CLKCONCMD.CLKSPD 的值必须设为 000；如果 CLKCONCMD.OSC=1，那么 CLKCONCMD.CLKSPD 的值必须设为 001。</p>
------	--------	-----	-----	--

CLKCONSTA (0x9E) —时钟控制状态

位	名称	复位	读/写	描述
7	OSC32K	1	R	当前所选择的 32kHz 时钟源： 0: 32kHz 晶体振荡器 1: 32kHz RC 振荡器
6	OSC	1	R	当前所选择的系统时钟： 0: 32MHz 晶体振荡器 1: 16MHz 高频 RC 振荡器
5: 3	TICKSPD[2: 0]	001	R	当前定时器 tick 输出设置。 000: 32MHz 001: 16MHz 010: 8MHz 011: 4MHz 100: 2MHz 101: 1MHz 110: 500kHz 111: 250kHz
2: 0	CLKSPD	001	R	当前时钟速度。 000: 32MHz 001: 16MHz 010: 8MHz

				011: 4MHz 100: 2MHz 101: 1MHz 110: 500kHz 111: 250kHz
--	--	--	--	---

4.5 定时器 Tick 产生

CLKCONCMD.TICKSPD 寄存器的值控制对定时器 1, 定时器 3 和定时器 4 的一个全局时钟分配。预分频器的值的设置可以从 0.25MHz 到 32MHz。应当注意, 如果 CLKCONCMD.TICKSPD 设置的频率高于系统时钟的频率, CLKCONSTA.TICKSPD 中表示的实际预分频值与系统时钟相同。

4.6 数据保持

在功耗模式 PM2 和 PM3, 电源从许多内部电路中移除。然而, SRAM 将保持它的内容, 在 PM2 和 PM3 模式内部寄存器的内容也被保留。

除非指定了一个特定的寄存器位字段, CPU 寄存器、外围寄存器和 RF 寄存器保持它们自己的内容。进入低功耗模式 PM2 或 PM3 对于软件是透明的。注意在 PM3 模式, 睡眠定时器的值不保存。

5 复位

CC253x 有 5 个复位源。下面的事件产生复位：

- 强置输入引脚 RESET_N 为低电平
- 上电复位
- 掉电复位
- 看门狗定时器复位
- 时钟丢失复位

复位后的初始状况如下：

- I/O 引脚设置为输入、上拉状态（P1.0 和 P1.1 为输入，但是没有上拉/下拉）
- CPU 的程序计数器设置为 0x0000，程序从这里开始运行
- 所有外部设备的寄存器初始化到它们的复位值（参考有关寄存器的描述）
- 看门狗定时器禁止
- 时钟丢失检测禁止

复位期间，I/O 引脚设置为输入、上拉状态（P1.0 和 P1.1 为输入，但是没有上拉/下拉）。

5.1 上电复位和掉电检测

CC253x 含有一个在设备上电期间提供正确初始化的上电复位（POR）。此外还有一个只工作在调整的 1.8V 数字供电上的掉电检测（BOD）。掉电检测在供电电压变化期间可以保护存储器内容，供电电压变化会导致调整的 1.8V 电源低于数字逻辑、Flash 存储器和 SRAM 所要求的最低水平。

当上电初始化已经开始时，上电复位（POR）和掉电检测（BOD）将保持设备处于复位状态，直到供电电压高于上电复位和掉电检测电压。

最后一次复位的原因可以从寄存器位 SLEEPSTA.RST 读取。应该指出的是，BOD 复位将被读为 POR 复位。

5.2 时钟丢失检测

时钟丢失检测可用于注重安全的系统，来检测其中一个晶体振荡器时钟源（32MHz 晶体振荡器或者 32kHz 晶体振荡器）的停止。由于外部晶振损坏或者相应元件损坏通常会发生时钟丢失。启用了时钟丢失检测，这 2 个时钟彼此连续监测。如果其中一个时钟停止切换，在指定的最大超时时间内就会产生一个时钟丢失复位。超时时间取决于哪个时钟停止了。如果是 32kHz 时钟停止，超时时间为 0.5ms。如果 32MHz 时钟停止，超时时间为 0.25ms。当系统复位后重新启动，软件可以通过读取 SLEEPSTA.RST[1: 0]来检测产生复位的原因。复位后使用内部 RC 振荡器。因此，系统可以重新启动，然后才能正常关机。通过 CLD.EN 位使能/禁止时钟丢失检测。使用了时钟丢失检测时，假定系统时钟源为 32MHz 晶体振荡器，那么 32kHz 时钟可以是 32kHz RC 振荡器（为了获得精确的复位超时时间，应该对它进行校准）或者 32kHz 晶体振荡器。

在功耗模式 PM1 和 PM2，时钟丢失检测会自动停止，而当系统重新启动时它也会重新启动。



CC253x 用户指南

进入功耗模式 PM3 之前，切换到 16MHz RC 振荡器并且禁用时钟丢失检测。再重新进入主动模式时，开启时钟丢失检测，然后切换回 32MHz 晶体振荡器。

CLD (0x6290) —时钟丢失检测

位	名称	复位	读/写	描述
7: 1	—	0000 000	R0	保留
0	EN	0	R/W	时钟丢失检测使能

6 Flash 控制器

CC253x 包含用于存储程序代码的 Flash 存储器。通过调试接口用软件可以对 Flash 存储器进行编程。

Flash 控制器处理对嵌入式 Flash 存储器的写和擦除。嵌入式 Flash 存储器最大由 128 页组成，每页 2048 字节。

Flash 控制器具有如下特征：

- 32 位字可编程
- 页擦除
- 用于写保护和代码安全的锁定位
- Flash 页擦除时间 20ms
- Flash 整片擦除时间 20ms
- Flash 写入时间（4 字节）20us

6.1 Flash 存储器组织

Flash 存储器被分为 2048 字节的 Flash 页。存储器的最小擦除单元为一个 Flash 页，可以写入 Flash 的最小可写单元为一个 32 位字。

执行写操作时，将 1 个 16 位地址写给地址寄存器 FADDRH: FADDRL，Flash 存储器就可以字寻址。

执行页擦除时，被擦除的 Flash 存储器页通过寄存器位 FADDRH[7: 1]来寻址。

请注意在寻址 Flash 存储器时的不同。当由 CPU 来进行读取代码或数据的存取时，Flash 存储器为字节寻址；当由 Flash 控制器存取时，Flash 存储器为字寻址，一个字由 32 位组成。

下面的小节对 Flash 写和 Flash 页擦除的过程进行详细描述。

6.2 Flash 写

Flash 可以通过 1 个或多个 32 位字（4 字节）的序列连续编程，按序从起始地址（由 FADDRH: FADDRL 设置）开始。一般来说，在开始写 Flash 之前必须先对页进行擦除。页擦除操作将页的所有位都置为 1。芯片擦除命令（通过调试接口）擦除 Flash 的所有页。这是将 Flash 中的位设为 1 的唯一方法。写 1 个字到 Flash，0 位可以编程为 0，而 1 位被忽略（Flash 中的这个位不变）。因此，这些位被擦除为 1，被写为 0。可以多次写 1 个字。这将在 6.2.2 节中描述。

6.2.1 Flash 写步骤

Flash 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 个最高有效位）
2. 设置 FCTL.WRITE 为 1，启动写序列状态机。
3. 在 20us 内对 FWDATA 进行四次写操作（如果首先不进行反复操作，最后一次

FCTL.FULL 会变为 0)。首先写入最低有效位 (最后一个字节后 FCTL.FULL 变为高)。

4. 等待, 直到 FCTL.FULL 变为低 (在步骤 3, Flash 控制器已经开始编程 4 字节的写, 并准备好缓冲下一个 4 字节)。

5. 可选的状态检查步骤:

- 如果步骤 3 中, 写入 4 字节不够快, 操作超时, 那么这一步的 FCTL.BUSY (和 FCTL.WRITE) 为 0。
- 如果由于页被锁定而不能写入 4 字节到 Flash, FCTL.BUSY (和 FCTL.WRITE) 为 0, FCTL.ABORT 为 1。

6. 如果这是最后 4 字节就退出, 否则转到步骤 3。

写操作可以使用下面两种方法的任意一种来实现:

- 通过 DMA 传送 (首选方法)
- 通过 CPU, 从 SRAM 运行代码

CPU 不能存取 Flash, 例如在一个 Flash 写操作正在进行的过程中读取程序代码。因此必须从 RAM 执行程序代码对 Flash 的写操作。关于如果从 RAM 运行代码请见 2.2.1 节。

当从 RAM 执行 Flash 写操作时, CPU 继续执行在 Flash 写操作启动后的下一条指令代码 (FCTL.WRITE=1)。

写 Flash 的过程中不能进入功耗模式 PM1、PM2 或 PM3, 并且不能改变系统时钟源 (晶体振荡器/RC 振荡器)。注意设置 CLKCONSTA.CLKSPE 的值为高就不能满足 20us 的写时间的要求。如果 CLKCONSTA.CLKSPD=111, 时钟周期仅为 4us。因此在写 Flash 时建议保持 CLKCONSTA.CLKSPD 的值为 000 或 001。

6.2.2 对一个字进行多次写

以下规则适用于在擦除之间对 1 个 32 位字进行多次写入:

- 在一个 32 位字内对一个位写 2 次 0。这不会改变该位的状态。
- 可以对 1 个 32 位字写 8 次。
- 对某个位写 1 不会改变该位的状态。

这使得可以对 1 个 32 位字进行多达 8 次的写 4 个新的位。表 6-1 为写序列到 1 个字的例子。这里 b_n 表示每次更新到字的 4 个新的位。这一功能有益于使用于数据记录应用的 Flash 的寿命最大化。

表 6-1 写序列示例

步骤	写入值	写入后的 Flash 内容	注释
1	(页擦除)	0xFFFFFFFF	擦除将所有位置为 1。
2	0xFFFFFFFb ₀	0xFFFFFFFb ₀	只有写 0 的位被置为 1, 所有写 1 的位被忽略。
3	0xFFFFFFFb ₁ F	0xFFFFFFFb ₁ b ₀	只有写 0 的位被置为 1, 所有写 1 的位被忽略。
4	0xFFFFFB ₂ FF	0xFFFFFB ₂ b ₁ b ₀	只有写 0 的位被置为 1, 所有写 1 的位被忽略。
5	0xFFFFFB ₃ FFF	0xFFFFFB ₃ b ₂ b ₁ b ₀	只有写 0 的位被置为 1, 所有写 1 的位被忽略。
6	0xFFFFb ₄ FFFF	0xFFFFb ₄ b ₃ b ₂ b ₁ b ₀	只有写 0 的位被置为 1, 所有写 1 的位被忽略。
7	0xFFb ₅ FFFFFF	0xFFb ₅ b ₄ b ₃ b ₂ b ₁ b ₀	只有写 0 的位被置为 1, 所有写 1 的位被忽略。
8	0xFb ₆ FFFFFF	0xFb ₆ b ₅ b ₄ b ₃ b ₂ b ₁ b ₀	只有写 0 的位被置为 1, 所有写 1 的位被忽略。
9	0xb ₇ FFFFFF	0xb ₇ b ₆ b ₅ b ₄ b ₃ b ₂ b ₁ b ₀	只有写 0 的位被置为 1, 所有写 1 的位被忽略。

如果每个数据样值超过 4 位, 可以用连续字来存储样值。例如, 对于 16 位样值, 字一

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]。当软件要读取一个样值时, 从 Flash 中读取 4 个字, 将它们合并为 1 个 16 位样值。这样, 可以使 Flash 的寿命最大化。

6.2.3 DMA Flash 写

当使用 DMA 写操作, 要写入 Flash 的数据被保存在 XDATA 存储器空间 (RAM 或 FLASH)。一个 DMA 通道被配置来读取从存储器源地址写入的数据, 并且将该数据写入 Flash 写数据寄存器 (FWDATA), 固定目的地址和 DMA 触发事件 FLASH (DMA 配置的 TRIG[4: 0]=1 0010) 启用。因此, 当 Flash 写数据寄存器 FWDATA 已经准备好接收新数据时 Flash 控制器将触发一个 DMA 传送。DMA 通道应当被配置为单独执行模式, 带有源地址的字节大小传送设置为从数据块开始, 目的地址为固定的 FWDATA (请注意块大小, 配置数据中的 LEN 必须排列为 4 字节; 否则, 最后一个字不会写入 Flash)。还应当确保 DMA 通道的高优先级, 以保证在写过程中不被中断。如果中断超过 20us, 写操作可能超时, 将不会进行写 FCTL.WRITE 位, 而将被复位。

当 DMA 通道被启用, 通过设置 FCTL.WRITE 为 1 来开始一个 Flash 写, 这将触发第一个 DMA 传送 (DMA 和 Flash 控制器处理传送复位)。

图 6-1 显示了一个 DMA 通道如何配置, 一个 DMA 传送是如何开始将一个来自 XDATA 某个地方的数据块写入 Flash 存储器的例子。

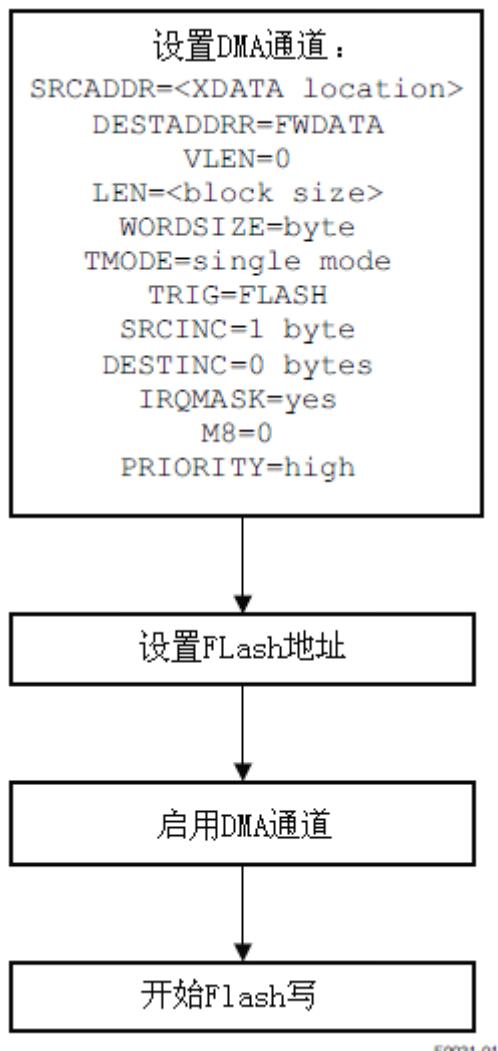


图 6-1 使用 DMA 进行 Flash 写

6.2.4 CPU Flash 写

使用 CPU 写 Flash，程序从 SRAM 执行，必须执行 6.2.1 节所列的步骤。禁止中断以保证操作不会超时。

6.3 Flash 页擦除

执行 Flash 页擦除后，被擦除页的所有字节置为 1。

将 FCTL.ERASE 置为 1 启动一个页擦除。当一个页擦除启动后，由 FADDRH[7: 1]寻址的页将被擦除。注意如果一个页擦除启动的同时有个页写入，即 FCTL.WRITE 置为 1，在进行页的写操作之前将执行页擦除。可以轮询 FCTL.BUSY 位来确定页擦除什么时候完成了。

擦除 Flash 页的过程中不能进入功耗模式 PM1、PM2 或 PM3，并且不能改变系统时钟源（晶体振荡器/RC 振荡器）。

注意：如果 Flash 页擦除操作是从 Flash 存储器内部执行的，且使能了看门狗定时器，看门狗定时器间隔必须选择为超过 20ms，即 Flash 页擦除的持续时间，以便 CPU 可以清除看门狗定时器。

6.3.1 从 Flash 存储器执行 Flash 擦除

从 Flash 存储器内部进行 Flash 页擦除所要求的步骤如图 17 所示。

注意：在从 Flash 存储器内部执行程序代码时，如果启动了一个 Flash 擦除或写操作，那么 CPU 停止，当 Flash 控制器完成操作后程序执行将从下一条指令重新开始。

如何擦除一个 Flash 页如下面的代码例子所示，使用 IAR 编译器：

```
#include <ioCC2530.h>

unsigned char erase_page_num = 3; /* page number to erase, here: flash page #3 */

/* Erase one flash page */
EA = 0; /* disable interrupts */
while (FCTL & 0x80); /* poll FCTL.BUSY and wait until flash controller is ready */
FADDRH = erase_page_num << 1; /* select the flash page via FADDRH[7:1] bits */
FCTL |= 0x01; /* set FCTL.ERASE bit to start page erase */
/* optional: wait until flash write has completed (-20 ms) */
EA = 1; /* enable interrupts */
```

6.4 Flash DMA 触发

当写入 FWDATA 寄存器的 Flash 数据已经被写入 Flash 存储器的指定位置时，Flash DMA 触发被激活，这表示 Flash 控制器准备接受写入 FWDATA 的新数据。产生 4 个触发脉冲。为了开始第一次传送，FCTL.WRITE 位必须置为 1。DMA 和 Flash 控制器接下来将自动处理已经定义的数据块（DMA 配置中的 LEN）的所有传送。更重要的是 DMA 启动优先于设置 FCTL.WRITE 位，触发源设置为 FLASH (TRIG[4: 0]=10010)，DMA 具有较高的优先级别以便传送不会中断。如果中断超过 20us，写操作超时，FCTL.WRITE 位被清除。

6.5 Flash 控制器寄存器

本节描述 Flash 控制器寄存器。

FCTL (0x6270) —Flash 控制

位	名称	复位	读/写	描述
7	BUSY	0	R	指示写或擦除正在运行。当 WRITE 或 ERASE 位置位时，该标志置位。 0：没有写或擦除操作激活 1：写或擦除操作激活
6	FULL	0	R/H0	写缓冲区满状态。在 Flash 写期间如果已经有 4 字节被写入到 FWDATA，该标志置 1。写缓冲区满，不再接受数据；即当 FULL 标志为 1 时，写入 FWDATA 将被忽略。当写缓冲区再次准备好接受 4 字节时，FULL 标志将被清除。只有当 CPU 用来写 Flash 时才需要该标志。 0：写缓冲区可以接受数据

				1: 写缓冲区已满
5	ABORT	0	R/H0	中止状态。当写操作或页擦除中止时该位置 1。当页访问被锁定时操作中止。开始写操作或页擦除时该位清 0。
4	—	0	R	保留
3: 2	CM[1: 0]	01	R/W	<p>缓存模式</p> <p>00: 缓存禁用</p> <p>01: 缓存使能</p> <p>10: 缓存使能, 预取模式</p> <p>11: 缓存使能, 实时模式</p> <p>缓存模式。禁用高速缓存会增加功耗和降低性能。对于大多数应用程序, 预取模式最高可提高 33% 的性能, 代价是增加了功耗。实时模式提供了可预见的 Flash 读取访问时间; 执行时间和缓存禁用模式的时间一样, 但是功耗较低。</p> <p>注意: 读出的值总是代表当前缓存模式。写入一个新的缓存模式就发出一个缓存模式改变请求, 这可能需要几个时钟周期来完成。如果已经有一个缓存改变请求存在, 写该寄存器被忽略。</p>
1	WRITE	0	R/W1/H0	<p>写。开始对由 FADDRH: FADDRL 给定的位置写字。在写完成之前, WRITE 位保持为 1。清除该位表示擦除已经完成, 即已经超时或中止。</p> <p>如果 ERASE 也置为 1, 在写之前对由 FADDRH[7: 0] 寻址的整页进行页擦除。当 ERASE 位为 1 时, 设置 WRITE 位为 1 不起作用。</p>
0	ERASE	0	R/W1/H0	<p>页擦除。对由 FADDRH[7: 1]给定的页进行擦除。在擦除完成之前, ERASE 位保持为 1。清除该位表示擦除已经成功完成或中止。</p> <p>当 WRITE 位为 1 时, 设置 ERASE 位为 1 不起作用。</p>

FWDATA (0x6273) —Flash 写数据

位	名称	复位	读/写	描述
7: 0	FWDATA[7: 0]	0x00	R/W	Flash 写数据。只有当 FCTL.WRITE 为 1 时才能写寄存器。

FADDRH (0x6272) —Flash 地址高字节

位	名称	复位	读/写	描述
7: 0	FADDRH[7: 0]	0x00	R/W	页地址/Flash 字地址的高字节 位[7: 1]将选择存取哪一页。

FADDRL (0x6271) —Flash 地址低字节

位	名称	复位	读/写	描述
7: 0	FADDRL[7: 0]	0x00	R/W	Flash 字地址的低字节。

7 I/O 口

CC253x有21个数字输入/输出引脚，可以配置为通用数字I/O，也可以作为外部I/O信号连接到ADC、定时器或者USART等外部设备。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口具备如下重要特性：

- 21个数字输入/输出引脚
- 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引脚都可以用于外部中断源输入口，因此如果需要，外部设备可以通过这些I/O产生中断。外部中断功能也可以唤醒睡眠模式。

7.1 未使用的 I/O 引脚

未使用的引脚应当定义电平，而不能浮空。一种方法是：该引脚不连接任何元器件，将其配置为具有内部上拉的通用输入口。这也是所有的引脚在复位期间和复位后的状态（只有P1.0和P1.1没有上拉/下拉能力）。这些引脚也可以配置为通用输出口。为了避免额外的功耗，无论引脚配置为输入口还是输出口，都不应该直接与VDD或者GND连接。

7.2 低 I/O 供电电压

在应用中数字I/O电源电压引脚的DVDD1和DVDD2低于2.6V时，寄存器位PICTL.PADSC应当置为1，以获得DC特性表中规定的输出直流特性。

7.3 通用 I/O

当用作通用I/O时，引脚可以组成3个8位口，端口0~2，定义为P0、P1和P2。其中，P0和P1是完全的8位口，而P2仅有5位可用。所有的口均可以位寻址，或通过特殊功能寄存器由P0、P1和P2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。

除了两个高输出口P1.0和P1.1之外，所有的口用于输出，均具备4mA的驱动能力；而P1.0和P1.1具备20mA的驱动能力。

寄存器PxSEL（其中x为口的标志，其值为0~2），用来设置每个端口引脚为通用I/O引脚或者是外部设备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，要改变一个引脚口的方向，使用寄存器PxDIR即可。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。

当读端口寄存器P0、P1和P2时，不管怎样，输入引脚上的逻辑值将返回引脚的配置值。这并不适用于在执行读—修改—写指令的过程中。读—修改—写指令为：ANL、ORL、XRL、JBC、CPL、INC、DEC、DJNZ、MOV、CLR和SETB。端口寄存器上下面的操作为

真：当目的地为端口寄存器 P0、P1 或 P2 的寄存器值里的一个独立位时，而不是引脚上的值，被读、修改和写返回给端口寄存器。

用作输入时，每个通用 I/O 口的引脚可以设置为上拉、下拉或三态模式。作为缺省的情况，复位之后所有的输入口均设置为上拉输入。要将输入口的某一位取消上拉或下拉，就要将 PxINP 中的对应位设置为 1。I/O 口引脚 P1.0 和 P1.1 不具备上拉/下拉能力。注意，即使外设功能为输入，引脚配置为外部设备 I/O 信号也不具备上拉/下拉能力，

在功耗模式 PM1、PM2 和 PM3，I/O 引脚保持在进入 PM1/PM2/PM3 时设置的 I/O 模式和输出值（如果可用）。

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。每个 P0、P1 和 P2 口的各位都可以中断使能，整个口中所有的位也可以中断使能。P0、P1、P2 口对应的寄存器为 IEN1 和 IEN2：

- IEN1. P0IE: P0 中断使能
- IEN2. P1IE: P1 中断使能
- IEN2. P2IE: P2 中断使能

除了这些公共中断使能之外，每个口的各位都可以通过位于 I/O 口的特殊功能寄存器 P0IEN、P1IEN 和 P2IEN 实现中断使能。当它们被使能时，即使这些 I/O 引脚被配置为外设 I/O 或者通用输出，也能产生中断。

当一个中断条件发生在任意一个 I/O 引脚上，P0—P2 中断标志寄存器里相应的中断状态标志 P0IFG、P1IFG 或 P2IFG 将被置为 1。中断状态标志的设置不考虑是否该引脚有它自己的中断使能设置。如果一个中断服务的中断状态标志通过写 0 到该标志而被清除，那么该标志必须在清除 CPU 端口中断标志 (PxIF) 之前被清除。

用于中断的 I/O 特殊功能寄存器在后面的小节中描述。寄存器概括如下：

- P0IEN: P0 中断使能
- P1IEN: P1 中断使能
- P2IEN: P2 中断使能
- PICTL: P0、P1 和 P2 中断触发沿配置
- P0IFG: 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 P1 口都关联一个 DMA 触发。对于 P0 口 DMA 的触发为 IOC_0，对于 P1 口 DMA 的触发为 IOC_1，如表 8-1 所示。

当 P0 口某个引脚上出现中断时，IOC_0 触发被激活；当 P1 口某个引脚上出现中断时，IOC_1 触发被激活。

7.6 外部设备 I/O

本节描述数字 I/O 引脚是如何配置为外部设备 I/O 引脚的。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 将在后面的小节中进行描述。

对于 USART 和定时器 I/O，选择数字 I/O 引脚上的外部设备 I/O 功能，需要将对应的寄存器位 PxSEL 置 1。

注意，外部设备单元具有两个可以选择的位置对应它们的 I/O 引脚。见表 7-1。如果存在关于 I/O 映射的冲突设置，那么可以在它们之间设置优先级（使用 P2SEL.PRIxP1 位和 P2DIR.PRIP0 位）。所有不引起冲突的结合都可以使用。

注意，即使没有使用外部设备，它通常也会在选定的位置，而使用到外部设备的引脚，就必须被赋予高优先级。有下面几种例外：USART 在 UART 模式下具有流控制的 RTS 和 CTS 引脚禁用，USART 在 SPI 主模式下配置的 SSN 引脚。

还应注意，无论 PxINP 的设置如何，有输入引脚的外部设备单元从引脚接收输入，而这可能会影响外部设备单元的状态。例如，如果 RX 引脚在作为一个 UART 引脚之前已经被激活，那么 UART 在使用之前就必须被清除。

表 7-1 外部设备 I/O 引脚映射

外部设备/ 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 Alt. 2			C	SS	MO	MI																				
												MO	MI	C	SS											
USART0 UART Alt. 2			RT	CT	TX	RX																				
												TX	RX	RT	CT											
USART1 SPI Alt. 2			MI	MO	C	SS						MI	MO	C	SS											
USART1 UART Alt. 2			RX	TX	RT	CT																				
												RX	TX	RT	CT											
TIMER1 Alt. 2		4	3	2	1	0												0	1	2						
	3	4																1	0							
TIMER 3 Alt. 2												1	0													
																		1	0							
TIMER 4 Alt. 2																				1			0			
																				Q1	Q2					
32kHz XOSC																							DC	DD		
DEBUG																										

7.6.1 定时器 1

PERCFG.T1CFG 选择是否使用可选择的位置 1 或者可选择的位置 2。

在表 7-1 中，定时器 1 的信号如下：

- 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

当指派若干外部设备到端口 0 时, 由 P2DIR. PRIPO 选择其优先顺序。当设置为 10 时, 定时器 1 通道 0—1 优先, 设置为 11 时, 定时器 1 通道 2—3 优先。要定时器 1 所有通道在可选择的位置 1 上可见, 移动 USART0 和 USART1 到可选择的位置 2。

当指派若干外部设备到端口 1 时, 由 P2SEL. PRI1P1 和 P2SEL. PRI0P1 选择其优先顺序。当 P2SEL. PRI1P1 设置为 0 而 P2SEL. PRI0P1 设置为 1 时, 定时器 1 通道优先。

7.6.2 定时器 3

PERCFG. T3CFG 选择是否使用可选择的位置 1 或者可选择的位置 2。

在表 7-1 中, 定时器 3 的信号如下:

- 0: 通道 0 比较引脚
- 1: 通道 1 比较引脚

当指派若干外部设备到端口 1 时, 由 P2SEL. PRI2P1 和 P2SEL. PRI3P1 选择其优先顺序。当这 2 位都设置为 1 时, 定时器 3 通道优先。如果 P2SEL. PRI2P1 设置为 1, P2SEL. PRI3P1 设置为 0, 定时器 3 通道的优先级高于 USART1, 但是 USART0 的优先级高于定时器 3, 自然也高于 USART1。

7.6.3 定时器 4

PERCFG. T4CFG 选择是否使用可选择的位置 1 或者可选择的位置 2。

在表 7-1 中, 定时器 4 的信号如下:

- 0: 通道 0 比较引脚
- 1: 通道 1 比较引脚

当指派若干外部设备到端口 1 时, 由 P2SEL. PRI1P1 选择其优先顺序。当设置为 1 时, 定时器 4 通道优先。

7.6.4 USART0

SFR 寄存器位 PERCFG. U0CFG 选择是否使用可选择的位置 1 或者可选择的位置 2。

在表 7-1 中, USART0 信号如下:

UART:

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

SPI:

- MI: MISO
- MO: MOSI

- C: SCK
- SS: SSN

当指派若干外部设备到端口 0 时，由 P2DIR. PRIPO 选择其优先顺序。当设置为 00 时，USART0 优先。注意，如果选择了 UART 模式而且此时硬件流控制禁止，则 USART1 或者定时器 1 就会优先使用端口 P0.4 和 P0.5。

当指派若干外部设备到端口 1 时，由 P2SEL. PRI3P1 和 P2SEL. PRIOP1 选择其优先顺序。当 P2SEL. PRI3P1 和 P2SEL. PRIOP1 均设置为 0 时，USART0 优先。注意，如果选择了 UART 模式而且此时硬件流控制禁止，则定时器 1 或者定时器 3 就会优先使用端口 P1.2 和 P1.3。

7.6.5 USART1

SFR 寄存器位 PERCFG. U1CFG 选择是否使用可选择的位置 1 或者可选择的位置 2。

在表 7-1 中，USART1 信号如下：

UART:

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

SPI:

- MI: MISO
- MO: MOSI
- C: SCK
- SS: SSN

当指派若干外部设备到端口 0 时，由 P2DIR. PRIPO 选择其优先顺序。当设置为 01 时，USART1 优先。注意，如果选择了 UART 模式而且此时硬件流控制禁止，则 USART0 或者定时器 1 就会优先使用端口 P0.2 和 P0.3。

当指派若干外部设备到端口 1 时，由 P2SEL. PRI3P1 和 P2SEL. PRI2P1 选择其优先顺序。当 P2SEL. PRI3P1 设置为 1 而 P2SEL. PRI2P1 设置为 0 时，USART1 优先。注意，如果选择了 UART 模式而且此时硬件流控制禁止，则 USART0 或者定时器 3 就会优先使用端口 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。ADC 输入最多可以使用 8 个。为了配置端口 0 的引脚为 ADC 输入，寄存器 APCCFG 的对应位必须设置为 1。该寄存器的默认值为选择端口 0 的引脚为非 ADC 输入，即数字输入/输出。

寄存器 APCCFG 中的设置将覆盖 POSEL 中的设置。

ADC 可以配置为使用通用 I/O 引脚 P2.0 作为一个外部触发来开始转换。当 P2.0 用于 ADC 外部触发时，它必须配置为在输入模式下的通用 I/O。

7.7 调试接口

端口 P2.1 和 P2.2 分别用作调试数据和时钟信号。该功能在表 7-1 中用 DD（调试数据）和 DC（调试时钟）表示。在调试模式下，调试接口控制这些引脚的方向，且这些引脚上的上拉/下拉禁用。

7.8 32kHz XOSC 输入

端口 P2.3 和 P2.4 用来连接一个外部 32kHz 晶振。不管寄存器设置，当 CLKCONCMD.OSC32K 为低时，这些端口引脚将由 32kHz 晶体振荡器使用，且这 2 个端口引脚将设置为模拟模式。

7.9 无线测试输出信号

通过使用寄存器 OBSSELx (OBSSEL0—OBSSEL5)，用户可以从 RF 内核输出不同的信号到 GPIO 引脚。这些信号可以用于低级别协议的调试，或者控制外部 PA、LNA，或者切换。控制寄存器 OBSSEL0—OBSSEL5 可以用来覆盖标准 GPIO 动作，并在引脚 P1[0: 5] 上输出 RF 内核信号 (rfc_obs_sig0, rfc_obs_sig1 和 rfc_obs_sig2)。可用信号的详细信息请见 19 节。

7.10 掉电信号多路器 (PMUX)

寄存器 PMUX 可以用来输出 32kHz 时钟和/或数字稳压器的状态。

可以使用一个 P0 引脚来输出选定的 32kHz 时钟源。用使能位 CKOEN 来使能通过 P0 输出，通过 CKOPIN (详见 PMUX 寄存器描述) 来选定具体的 P0 引脚。当 CKOEN 设置为 1 时，所选引脚的其它所有配置都被覆盖。不管处于何种功耗模式，都可以输出时钟；但是，在 PM3 模式下，时钟停止 (见第 4 节的 PM3)。

此外，可以在任意一个 P1 引脚上输出数字稳压器的状态。当 DREGSTA 位为 1 时，输出数字稳压器的状态。DREGSTAPIN 选定具体的 P1 引脚 (详见 PMUX 寄存器描述)。当 DREGSTA 设置为 1 时，所选引脚的其它所有配置都被覆盖。当 1.8V 片上数字稳压器上电 (电压经过芯片调整) 时，选定的引脚输出 1。当 1.8V 片上数字稳压器掉电，即在 PM2 和 PM3 模式下，选定的引脚输出 0。

7.11 I/O 寄存器

本节描述 I/O 端口寄存器。它们是：

- P0: 端口 0
- P1: 端口 1
- P2: 端口 2
- PERCFG: 外部设备控制寄存器

- APCFG: 模拟外部设备 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: 中断边沿寄存器
- P0IEN: 端口 0 中断使能寄存器
- P1IEN: 端口 1 中断使能寄存器
- P2IEN: 端口 2 中断使能寄存器
- PMUX: 掉电信号多路器寄存器
- 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	读/写	描述
7: 0	P0[7: 0]	0xFF	R/W	端口 0。通用 I/O 口，可以从 SFR 位寻址。该 CPU 内部寄存器可以从 XDATA (0x7080) 读取，但不能写。

P1 (0x90) —端口 1

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

P2 (0xA0) —端口 2

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

PERCFG (0xF1) —外部设备控制

位	名称	复位	读/写	描述
7	—	0	R0	未使用

6	T1CFG	0	R/W	定时器 1 的 I/O 位置 0: 选择到位置 1 1: 选择到位置 2
5	T3CFG	0	R/W	定时器 3 的 I/O 位置 0: 选择到位置 1 1: 选择到位置 2
4	T4CFG	0	R/W	定时器 4 的 I/O 位置 0: 选择到位置 1 1: 选择到位置 2
3: 2	—	00	R0	未使用
1	U1CFG	0	R/W	USART 1 的 I/O 位置 0: 选择到位置 1 1: 选择到位置 2
0	U0CFG	0	R/W	USART 0 的 I/O 位置 0: 选择到位置 1 1: 选择到位置 2

APCFG (0xF2) —模拟外部设备 I/O 配置

位	名称	复位	读/写	描述
7: 0	APCFG[7: 0]	0x00	R/W	模拟外部设备 I/O 配置。APCFG[7: 0]选择 P0.7~P0.0 作为模拟 I/O 0: 模拟 I/O 禁止 1: 模拟 I/O 使能

P0SEL (0xF3) —端口 0 功能选择

位	名称	复位	读/写	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: 外部设备功能

P1SEL (0xF4) —端口 1 功能选择

位	名称	复位	读/写	描述
7: 0	SELP1_[7: 0]	0x00	R/W	P1.7 到 P1.0 功能选择 0: 通用 I/O 1: 外部设备功能

P2SEL (0xF5) —端口 2 功能选择和端口 1 外部设备优先级控制

位	名称	复位	读/写	描述
7	—	0	R0	未使用
6	PRI3P1	0	R/W	端口 1 外部设备优先级控制。其值在 PERCFG 分配 USART0 和 USART1 到同一个引脚时，判定两者优先级的顺序 0: USART0 优先 1: USART1 优先

5	PRI2P1	0	R/W	端口 1 外部设备优先级控制。其值在 PERCFG 分配 USART1 和定时器 3 到同一个引脚时，判定两者优先级的顺序 0: USART1 优先 1: 定时器 3 优先
4	PRI1P1	0	R/W	端口 1 外部设备优先级控制。其值在 PERCFG 分配定时器 1 和定时器 4 到同一个引脚时，判定两者优先级的顺序 0: 定时器 1 优先 1: 定时器 4 优先
3	PRI0P1	0	R/W	端口 1 外部设备优先级控制。其值在 PERCFG 分配 USART0 和定时器 1 到同一个引脚时，判定两者优先级的顺序 0: USART0 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: 外部设备功能
1	SELP2_3	0	R/W	P2.3 功能选择 0: 通用 I/O 1: 外部设备功能
0	SELP2_0	0	R/W	P2.0 功能选择 0: 通用 I/O 1: 外部设备功能

P0DIR (0xFD) —端口 0 方向

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

P1DIR (0xFE) —端口 1 方向

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

P2DIR (0xFF) —端口2方向和端口0外部设备优先级控制

位	名称	复位	读/写	描述
7: 6	PRIP0[1: 0]	00	R/W	端口 0 外部设备优先级控制。这两位决定当 PERCFG 指派若干外部设备到同一个引脚时的优先顺序。 详细优先级列表： 00: 第 1 优先级: USART0 第 2 优先级: USART1 第 3 优先级: 定时器 1

				01: 第 1 优先级: USART1 第 2 优先级: USART0 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: USART1 第 3 优先级: USART0 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: USART0 第 3 优先级: USART1 第 4 优先级: 定时器 1 通道 0-1
5	—	0	R0	未使用
4: 0	DIRP2_[4: 0]	0 000	R/W	P2.4 到 P2.0 I/O 方向 0: 输入 1: 输出

P0INP (0x8F) —端口0输入模式

位	名称	复位	读/写	描述
7: 0	MDP0_[7: 0]	0x00	R/W	P0.7 到 P0.0 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	读/写	描述
7: 2	MDP1_[7: 2]	0000 00	R/W	P1.7 到 P1.2 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	未使用

P2INP (0xF7) —端口 2 输入模式

位	名称	复位	读/写	描述
7	PDUP2	0	R/W	端口 2 上拉/下拉选择。选择所有端口 2 引脚功能设置为上拉/下拉输入 0: 上拉 1: 下拉
6	PDUP1	0	R/W	端口 1 上拉/下拉选择。选择所有端口 1 引脚功能设置为上拉/下拉输入 0: 上拉 1: 下拉
5	PDUP0	0	R/W	端口 0 上拉/下拉选择。选择所有端口 0 引脚功能设置为上拉/下拉输入

				下拉输入 0: 上拉 1: 下拉
4: 0	MDP2_[4: 0]	00000	R/W	P2.4 到 P2.0 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	读/写	描述
7: 0	P0IF[7: 0]	0x00	R/W0	端口 0, 位 7~位 0 输入中断状态标志。当输入口的一个引脚上有中断请求未决信号, 其对应的标志位将置 1。

P1IFG (0x8A) —端口 1 中断状态标志

位	名称	复位	读/写	描述
7: 0	P1IF[7: 0]	0x00	R/W0	端口 1, 位 7~位 0 输入中断状态标志。当输入口的一个引脚上有中断请求未决信号, 其对应的标志位将置 1。

P2IFG (0x8B) —端口 2 中断状态标志

位	名称	复位	读/写	描述
7: 6	—	00	R0	未使用。
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信号时, 该标志位置 1, 用于检测 USB 挂起状态下的 USB 恢复事件。当 USB 控制器没有挂起时该标志位为 0。
4 : 0	P2IF[4: 0]	00000	R/W0	端口 2, 位 4~位 0 输入中断状态标志。当输入口的一个引脚上有中断请求未决信号, 其对应的标志位将置 1。

PICTL (0x8C) —中断触发沿控制

位	名称	复位	读/写	描述
7	PADSC	0	R/W	控制 I/O 引脚在输出模式的驱动能力。为 DVDD 引脚上电压占低选择输出驱动能力增强(以此确保高/低电压情况下的驱动能力一样)。 0: 最小驱动能力增强。DVDD 1/2 等于或大于 2.6V 1: 最大驱动能力增强。DVDD 1/2 小于 2.6V
6: 4	—	000	R0	未使用
3	P2ICON	0	R/W	端口 2, P2.4~P2.0 输入模式下的中断配置。该位为端口 2.4~2.0 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
2	P1ICONH	0	R/W	端口 1, P1.7~P1.4 输入模式下的中断配置。该位为端口 1.7~1.4 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, P1.3~P1.0 输入模式下的中断配置。该位为端口 1.3~1.0

				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
0	P0ICON	0	R/W	端口 0, P0.7~P0.0 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 中断使能

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

P1IEN (0x8D) —端口 1 中断使能

位	名称	复位	读/写	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 中断使能

位	名称	复位	读/写	描述
7: 6	—	00	R0	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) —掉电信号多路器

位	名称	复位	读/写	描述
7	CKOEN	0	R/W	时钟输出使能。当该位为 1 时，选定的 32kHz 时钟在一个 P0 引脚上输出。CKOPIN 选择具体的输出引脚。这将覆盖选定引脚的其它所有配置。时钟在所有功耗模式下都输出；但是，在 PM3 模式下，时钟是停止的（见第 4 节 PM3）
6: 4	CKOPIN[2: 0]	000	R/W	时钟输出引脚。选择具体的 P0 引脚来输出选定的 32kHz 时钟。
3	DREGSTA	0	R/W	数字稳压器状态。当该位为 1 时，数字稳压器的状态在一个 P1 引脚上输出。DREGSTAPIN 选择具体的引脚。当 DREGSTA 设置为 1 时，所选引脚的其它所有配置都被覆盖。当 1.8V 片上数字稳压器上电（电压经过芯片调整）时，选定的引脚输出 1。当 1.8V 片上数字稳压器掉电时，选定的引脚输出 0。
2: 0	DREGSTAPIN[2: 0]	000	R/W	数字稳压器状态引脚。选择哪一个 P1 引脚用来输出 DREGST

				A 信号。
--	--	--	--	-------

OBSSEL0 (0x6243) — 观察输出控制寄存器 0

位	名称	复位	读/写	描述
7	EN	0	R/W	P1[0]上观察输出 0 的位控制。 0—观察输入禁止 1—观察输出使能 注意：如果该位使能，会覆盖 P1.0 的标准 GPIO 动作。
6: 0	SEL[6: 0]	000 0000	R/W	选择观察输出 0 上的输出信号。 1111011 (123): rfc_obs_sig0 1111100 (124): rfc_obs_sig1 1111101 (125): rfc_obs_sig2 其它：保留

OBSSEL1 (0x6244) — 观察输出控制寄存器 1

位	名称	复位	读/写	描述
7	EN	0	R/W	P1[1]上观察输出 1 的位控制。 0—观察输入禁止 1—观察输出使能 注意：如果该位使能，会覆盖 P1.1 的标准 GPIO 动作。
6: 0	SEL[6: 0]	000 0000	R/W	选择观察输出 1 上的输出信号。 1111011 (123): rfc_obs_sig0 1111100 (124): rfc_obs_sig1 1111101 (125): rfc_obs_sig2 其它：保留

OBSSEL2 (0x6245) — 观察输出控制寄存器 2

位	名称	复位	读/写	描述
7	EN	0	R/W	P1[2]上观察输出 2 的位控制。 0—观察输入禁止 1—观察输出使能 注意：如果该位使能，会覆盖 P1.2 的标准 GPIO 动作。
6: 0	SEL[6: 0]	000 0000	R/W	选择观察输出 2 上的输出信号。 1111011 (123): rfc_obs_sig0 1111100 (124): rfc_obs_sig1 1111101 (125): rfc_obs_sig2 其它：保留

OBSSEL3 (0x6246) — 观察输出控制寄存器 3

位	名称	复位	读/写	描述
7	EN	0	R/W	P1[3]上观察输出 3 的位控制。 0—观察输入禁止 1—观察输出使能

				注意：如果该位使能，会覆盖 P1.3 的标准 GPIO 动作。
6: 0	SEL[6: 0]	000 0000	R/W	选择观察输出 3 上的输出信号。 1111011 (123): rfc_obs_sig0 1111100 (124): rfc_obs_sig1 1111101 (125): rfc_obs_sig2 其它：保留

OBSSEL4 (0x6247) — 观察输出控制寄存器 4

位	名称	复位	读/写	描述
7	EN	0	R/W	P1[4]上观察输出 4 的位控制。 0—观察输入禁止 1—观察输出使能 注意：如果该位使能，会覆盖 P1.4 的标准 GPIO 动作。
6: 0	SEL[6: 0]	000 0000	R/W	选择观察输出 4 上的输出信号。 1111011 (123): rfc_obs_sig0 1111100 (124): rfc_obs_sig1 1111101 (125): rfc_obs_sig2 其它：保留

OBSSEL5 (0x6248) — 观察输出控制寄存器 5

位	名称	复位	读/写	描述
7	EN	0	R/W	P1[5]上观察输出 5 的位控制。 0—观察输入禁止 1—观察输出使能 注意：如果该位使能，会覆盖 P1.5 的标准 GPIO 动作。
6: 0	SEL[6: 0]	000 0000	R/W	选择观察输出 5 上的输出信号。 1111011 (123): rfc_obs_sig0 1111100 (124): rfc_obs_sig1 1111101 (125): rfc_obs_sig2 其它：保留

8 DMA 控制器

CC253x 内置一个直接存储器存取（DMA）控制器，该控制器可以用来减轻 8051 CPU 内核传送数据时的负担，实现 CC253x 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 ADC 或 RF 收发器等外部设备单元传送到存储器。

DMA 控制器协调所有的 DMA 传送，确保 DMA 请求和 CPU 存取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设置的 DMA 通道，用来实现存储器—存储器的数据传送。

DMA 控制器控制超过整个 XDATA 存储器空间里全部地址范围的数据传输。由于多数 SFR 寄存器映射到 DMA 存储器空间，这些灵活的 DMA 通道可以以一种创新的方式减轻 CPU 的负担。例如，从存储器传送数据到 USART，按照定下来的周期在 ADC 和存储器之间传送数据等。使用 DMA 可以保持 CPU 在不需要唤醒的低功耗模式下，进行与外部设备单元之间的数据传送，这就降低了整个系统的功耗（请见 4.1.1 小节 CPU 低功耗模式）。注意 2.2.3 小节里说明了哪一个 SFR 寄存器不映射到 XDATA 存储器空间。

DMA 控制器的主要性能如下：

- 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
- 3 种传送模式：单独传送、数据块传送和重复传送
- 支持传送数据的长度域，设置可变传送长度
- 既可以工作在字（word—size）模式，又可以工作在字节（byte—size）模式

8.1 DMA 操作

DMA 控制器有 5 个 DMA 通道可用，即 DMA 通道 0~4。每个 DMA 通道能够从 DMA 存储器空间传送数据到另一个存储空间，比如 XDATA 空间之间。

要使用 DMA 通道，必须首先按照 8.2 和 8.3 小节所讲的进行配置。DMA 状态如图 8-1 所示。

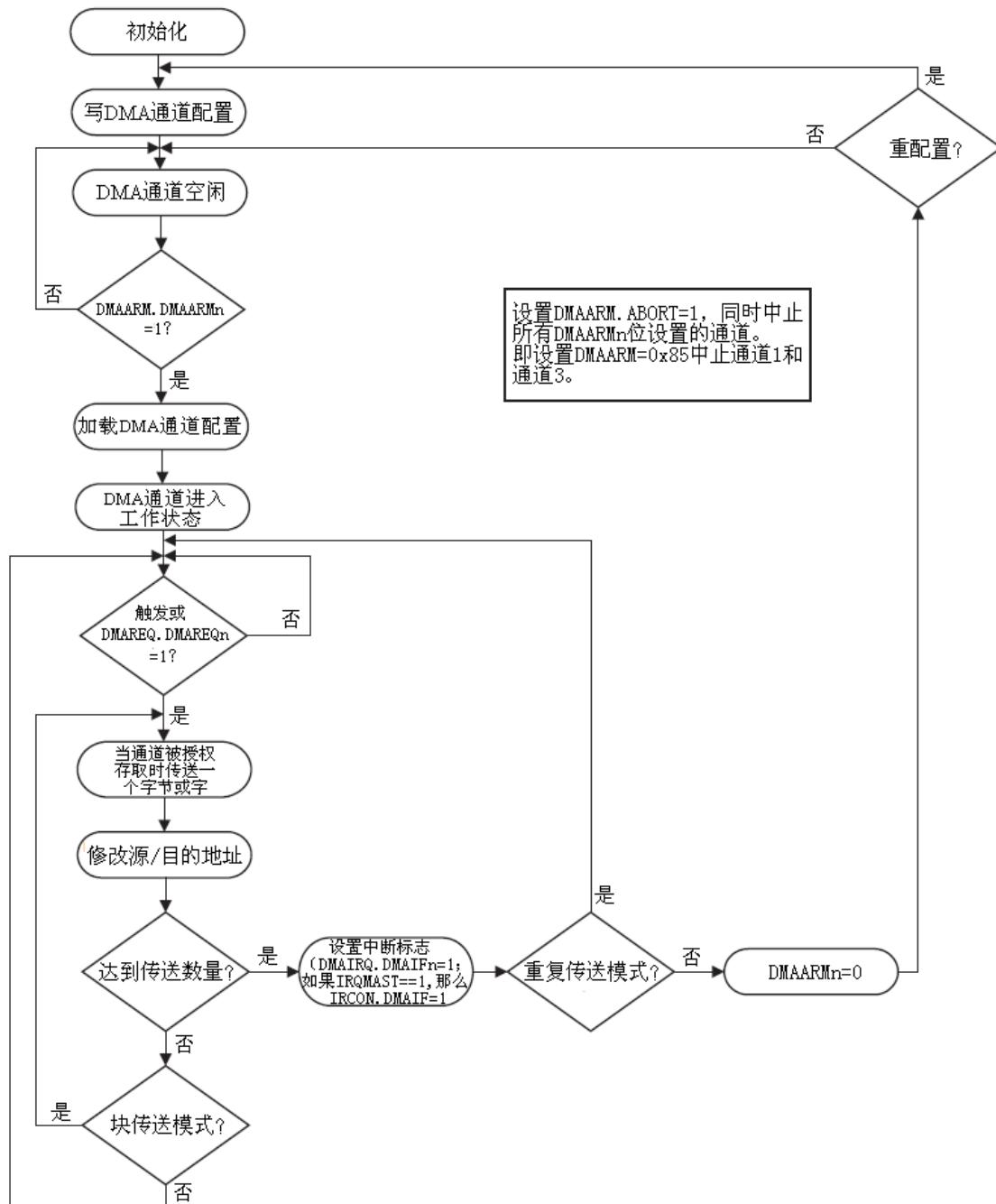
当 DMA 通道配置完毕，在允许任何传送开始之前，必须进入工作状态。DMA 通道通过将 DMA 通道工作状态寄存器中指定位 DMAARM 置 1，就可以进入工作状态。

一旦 DMA 信道进入工作状态，当设定的 DMA 触发事件发生时，传送就开始了。注意需要 9 个系统时钟的时间来装备一个通道（即获取配置数据），因此，如果 DMAARM 位置位，在开始配置通道的这个时间内出现了一个触发，那么这个触发将丢失。如果有多个 DMA 通道同时装备，那么对所有通道进行配置的时间将更长（从存储器中顺序读取）。如果所有的 5 个通道被装备，将需要 45 个系统时钟的时间，通道 1 首先准备好，然后是通道 2，通道 0 最后装备好（全部在最后的 8 个系统时钟以内）。可能的 DMA 触发事件有 32 个（见表 8-1），例如 UART 传送、定时器溢出等。DMA 通道要使用的触发事件在配置 DMA 信道时设置，所以在读取配置之前这些可用事件都是不知道的。所有的这些触发事件如表 8-1 所列。

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，强制一个 DMA 传送开始。

应当注意，如果正在配置 DMA 的时候，之前配置的触发源产生了触发事件，这些事件都算作是错过的事件，一旦 DMA 通道准备好，就立即开始传送这些错过的事件。即使所配置的新的触发源和之前配置的触发源不同。在某些情况下，这会引起传送错误。为了解决这一问题，触发源 0 必须是重配置之间的触发源。这可以通过设置虚拟的源地址和目的地址、使用一个字节的固定长度、数据块传输和触发源 0 实现。使能软件触发（DMAREQ）清除错过的触发数，当从存储器取一个新的配置的时候（除非软件对该通道写 DMAREQ）不产生新的触发。

只有当相应的 DMA 传送发生时才能清除 DMAREQ 位。当通道解除工作状态的时候不清除 DMAREQ 位。



F0033-01

图 8-1 DMA 操作

8.2 DMA 配置参数

DMA 的安装和控制由用户软件完成。本节描述在 DMA 通道能够使用之前，必须配置的参数。8.3 小节描述如何在软件中配置这些参数并传送到 DMA 控制器。

5 个 DMA 通道的行为都与下列参数有关：

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，以及预警 CPU 即将有中断请求到来之前，要传送的长度。可以在配置 DMA 参数时设置该长度，或者将 DMA 读出的第一个字节/字用作该长度。

可变长度 (VLEN) 设置： DMA 信道可以利用源数据中的第一个字节或字作为传送长度来进行可变长度传送。使用可变长度传送时，关于如何计算字节数的可变选项对于传送是可用的。

优先级别： 对于 DMA 通道，DMA 传送的优先级别与 CPU、其它 DMA 通道和存取口有关。

触发事件： 所有的 DMA 传输通过所谓的 DMA 触发事件发起。这个触发可以启动一个 DMA 块传输或单个 DMA 传输。除了配置的触发，一个 DMA 通道总是可以通过设置它的指定的 DMAREQ.DMAREQ_x 标志来触发。DMA 触发源见表 8-1。

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量或者不变。

DMA 传送模式： 传送模式决定传送是否为单个传送或者块传送，或者它们的重复传送。

字节传送或字传送： 判定每个 DMA 传送究竟是 8 位（字节）还是 16 位（字）。

中断屏蔽： 在完成 DMA 传送的基础上，产生一个中断请求。中断屏蔽位控制中断产生是使能还是禁止。

M8： 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。

8.2.1 到 8.2.11 小节将详细讲述这些配置参数。

8.2.1 源地址

DMA 通道要开始读取数据的 XDATA 存储器的地址。这可以是任何 XDATA 地址—在 RAM 中，在映射的 Flash 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 中，XREG 或 XDATA 寻址的 SFR。

8.2.3 传送长度

需要由 DMA 传送器完成传送的字节数或字数。当到达传送长度时，DMA 控制器重新

使 DMA 通道进入工作状态或解除 DMA 通道的工作状态，并且以一个中断请求预警 CPU。传送长度可以在配置里面定义或者定义为一个下面 8.2.4 小节描述的可变长度。

8.2.4 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（对于字，使用[12 : 0]位）作为传送长度。也就允许可变长度传送。使用可变长度传送时，要给出不同的传送字节长度。在任何情况下都是设置传送长度 (LEN) 为传送的最大长度。如果由首字节或首字指定的传送长度远大于 LEN，那么将传送 LEN 字节/字。如果使用可变长度传送，LEN 应当被设置为最大来允许传送长度加一。

注意，仅在选择字节长度传送数据时，才可以使用 M8 位模式 (8.2.11)。

可以同 VLEN 一起设置的选项如下：

1. 要传送的字节/字的长度，设置在第一字节/字+1 处（先传送长度字节/字，然后传送尽可能多的由长度字节/字指定的字节/字）。
2. 要传送的字节/字的长度，设置在第一字节/字处。
3. 要传送的字节/字的长度，设置在第一字节/字+2 处（先传送长度字节/字，然后传送尽可能多的由长度字节/字+1 指定的字节/字）。
4. 要传送的字节/字的长度，设置在第一字节/字+3 处（先传送长度字节/字，然后传送尽可能多的由长度字节/字+2 指定的字节/字）。

VLEN 选项如图 8-2 所示。

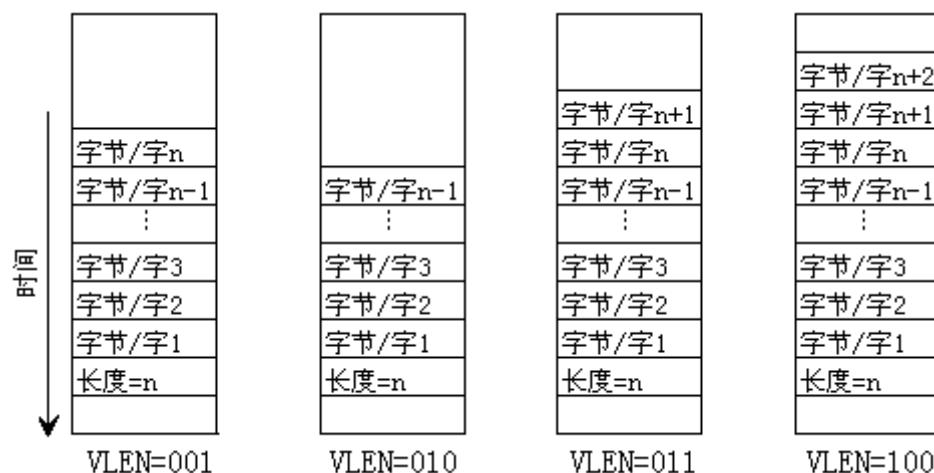


图 8-2 可变长度 (VLEN) 传送选项

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。该字段判定 DMA 信道会接受哪一个事件的触发。

8.2.6 源地址和目标地址增量

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。
- 增量为 2：每次传送之后地址指针将加上 2。
- 减量为 1：每次传送之后地址指针将减去 1。

在字节模式，1 个计数等于 1 字节；在字模式，1 个计数等于 2 字节。

8.2.7 DMA 传送模式

传送模式确定当开始传送数据时 DMA 通道如何工作。共有 4 种 DMA 传送模式：

单一模式：每当触发时，发生单一 DMA 传送；此后，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。

块传送模式：每当触发时，若干 DMA 传送按照指定的传送长度尽快传送；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，发生单一 DMA 传送；此后，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 DMA 通道重新进入工作状态。

重复的块传输模式：每当触发时，若干 DMA 传送按照指定的传送长度尽快传送；此后，通报 CPU DMA 通道重新进入工作状态。

8.2.8 DMA 优先级

可以对每个 DMA 通道配置 DMA 优先级别。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别最高，以及 DMA 存储器存取的优先级别是否超过同时发生 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（round-robin）方案应对所有的请求，确认存取对象。DMA 优先级别有 3 级：

高级：最高内部优先级别。DMA 存取总是优于 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 字节或字传送

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 CPU。该位可以屏蔽中断。

8.2.11 M8 设置

这个域的值，决定是否采用 7 位或 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

上面小节所描述的 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启动。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 配置数据结构由 8 字节构成，将在 8.6 小节进行描述。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 DMAxCFGH: DMAxCFGL 送到 DMA 控制器。一旦 DMA 通道进入工作状态，DMA 控制器就会读取由 DMAxCFGH: DMAxCFGL 地址里给定的该通道的配置数据结构。

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里的一个连续区域内，这个区域开始于保存在 DMA1CFGH: CMA1CFGL 里的地址，由 32 个字节组成。

8.4 停止 DMA 传送

正在进行的 DMA 传送或已经进入工作状态的 DMA 通道可以使用 DMAARM 寄存器来中止或解除 DMA 通道。

将 1 写入寄存器位 DMAARM. ABORT 可以中止一个或多个 DMA 通道，同时将相应的 DMAARM. DMAARMx 位写 1 选择中止某一个具体的 MDA 通道。当 DMAARM. ABORT 置为 1，未中止通道的 DMAARM. DMAARMx 位必须写为 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，特殊功能寄存器 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时，软件总是能够检测（以及清除）这个寄存器，即使 IRQMASK 已经置 1。不这样做将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 配置数据结构由 8 个字节组成，见表 8-2。

8.7 DMA 存储器存取

DMA 数据传送受字节存储次序协议的影响。注意 DMA 描述符使用大字节存储次序，而其它寄存器使用小字节存储次序。这一点必须在编译器里说明。

表 8-1 DMA 触发源

DMA 触发		功能单位	描述
序号	名称		
0	NONE	DMA	无触发，设置 DMAREQ. DMAREQx 位，开始传送
1	PREV	DMA	DMA 通道因前一个通道完成而触发
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	I/O 控制器	端口 0 的 I/O 引脚输入转换 ⁽¹⁾
13	IOC_1	I/O 控制器	端口 1 的 I/O 引脚输入转换 ⁽¹⁾
14	URX0	USART0	USART0 RX 完成
15	UTX0	USART0	USART0 TX 完成
16	URX1	USART1	USART1 RX 完成
17	UTX1	USART1	USART1 TX 完成
18	FLASH	Flash 控制器	完成写 Flash 数据
19	RADIO	无线	RF 数据包字节接收完毕（见 19.3 小节）
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，采样已经准备好
22	ADC_CH21	ADC	ADC 结束通道 1 的一次转换，采样已经准备好
23	ADC_CH32	ADC	ADC 结束通道 2 的一次转换，采样已经准备好
24	ADC_CH42	ADC	ADC 结束通道 3 的一次转换，采样已经准备好
25	ADC_CH53	ADC	ADC 结束通道 4 的一次转换，采样已经准备好
26	ADC_CH63	ADC	ADC 结束通道 5 的一次转换，采样已经准备好
27	ADC_CH74	ADC	ADC 结束通道 6 的一次转换，采样已经准备好
28	ADC_CH84	ADC	ADC 结束通道 7 的一次转换，采样已经准备好
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

⁽¹⁾ 使用该触发源必须结合端口中断使能位。注意所有中断使能的端口引脚将产生一个触发。

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 通道目标地址, 高位。注意, Flash 存储器不可直接写。
3	7: 0	DESTADDR[7: 0]	DMA 通道目标地址, 低位。注意, Flash 存储器不可直接写。
4	7: 0	VLEN[2: 0]	可变长度传送模式。在字模式, 第一个字的 12: 0 位被认为是传送长度 000: 采用 LEN 作为传送长度。 001: 传送由第一个字节/字+1 (最大到由 LEN 指定的最大值) 指定的字节/字长度。因此传送长度不包括该字节/字之长。 010: 传送由第一个字节/字 (最大到由 LEN 指定的最大值) 指定的字节/字长度。因此传送长度包括该字节/字之长。 011: 传送由第一个字节/字+2 (最大到由 LEN 指定的最大值) 指定的字节/字长度。 100: 传送由第一个字节/字+3 (最大到由 LEN 指定的最大值) 指定的字节/字长度。 101: 保留。 110: 保留。 111: 采用 LEN 作为传送长度的备用。
4	4: 0	LEN[12: 8]	DMA 通道传送长度。 当 VLEN=000/111, 采用最大允许长度。当处于 WORDSIZE 模式时, DMA 通道长度以字计量, 否则以字节计量。
5	7: 0	LEN[7: 0]	DMA 通道传送长度。 当 VLEN=000/111, 采用最大允许长度。当处于 WORDSIZE 模式时, DMA 通道长度以字计量, 否则以字节计量。
6	7	WORDSIZE	选择每个 DMA 传送采用 8 位 (0), 还是 16 位 (1)。
6	6: 0	TMODE[1: 0]	DMA 通道传送模式 00: 单一模式 01: 块传送模式 10: 重复单一模式 11: 重复块传送模式
6	4: 0	TRIG[4: 0]	选择要使用的 DMA 触发。 00000: 无触发 (写到 DMAREQ 的仅仅是触发) 00001: 前一个 DMA 通道完成。 00010~11111: 选择表 8-1 中的一个触发, 按照序号选择。
7	7: 0	SRCINC[1: 0]	源地址增量模式 (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]	目标地址增量模式 (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: 禁止中断产生 1: 使能 DMA 通道完成时的中断产生
7	2	M8	采用 8th 位模式作为 VLEN 传送长度, 仅应用在 WORDSIZE=0, 且 VLEN=000/111 时。 0: 使用全部 8 位作为传送长度 1: 使用低 7 位 (LSB) 作为传送长度
7	1: 0	PRIORITY[1: 0]	DMA 通道优先级别: 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

本节讲述与 DMA 控制器有关的 SFR 寄存器。

DMAARM (0xD6) —DMA 通道进入工作状态

位	名称	复位	读/写	描述
7	ABORT	0	R0/W	DMA 中止。该位用于停止运行中的 DMA 传送。将该位写 1 将中止所有通过设置对应的 DMAARM 位为 1 而选择的通道。 0: 正常操作 1: 中止所有选择的通道
6: 5	—	00	R/W	未使用。
4	DMAARM4	0	R/W1	DMA 通道 4 进入工作状态。 为了 DMA 通道 4 能够传送, 该位必须置 1。对于非重复传送模式, 一旦完成传送, 该位自动清 0。
3	DMAARM3	0	R/W1	DMA 通道 3 进入工作状态。 为了 DMA 通道 3 能够传送, 该位必须置 1。对于非重复传送模式, 一旦完成传送, 该位自动清 0。
2	DMAARM2	0	R/W1	DMA 通道 2 进入工作状态。 为了 DMA 通道 2 能够传送, 该位必须置 1。对于非重复传送模式, 一旦完成传送, 该位自动清 0。
1	DMAARM1	0	R/W1	DMA 通道 1 进入工作状态。

				为了 DMA 通道 1 能够传送，该位必须置 1。对于非重复传送模式，一旦完成传送，该位自动清 0。
0	DMAARM0	0	R/W1	DMA 通道 0 进入工作状态。 为了 DMA 通道 0 能够传送，该位必须置 1。对于非重复传送模式，一旦完成传送，该位自动清 0。

DMAREQ (0xD7) —DMA 通道开始请求及其状态

位	名称	复位	读/写	描述
7: 5	—	000	R0	未使用。
4	DMAREQ4	0	R/W1 H0	DMA 传送请求，通道 4。 置 1 该位，DMA 通道 4 有效（与单一触发事件有同样效果）。当 DMA 传送开始该位清 0。
3	DMAREQ3	0	R/W1 H0	DMA 传送请求，通道 3。 置 1 该位，DMA 通道 3 有效（与单一触发事件有同样效果）。当 DMA 传送开始该位清 0。
2	DMAREQ2	0	R/W1 H0	DMA 传送请求，通道 2。 置 1 该位，DMA 通道 2 有效（与单一触发事件有同样效果）。当 DMA 传送开始该位清 0。
1	DMAREQ1	0	R/W1 H0	DMA 传送请求，通道 1。 置 1 该位，DMA 通道 1 有效（与单一触发事件有同样效果）。当 DMA 传送开始该位清 0。
0	DMAREQ0	0	R/W1 H0	DMA 传送请求，通道 0。 置 1 该位，DMA 通道 0 有效（与单一触发事件有同样效果）。当 DMA 传送开始该位清 0。

DMA0CFGH (0xD5) —DMA 通道 0 配置地址高位字节

位	名称	复位	读/写	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 配置地址低位字节

位	名称	复位	读/写	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 通道 1~4 配置地址高位字节

位	名称	复位	读/写	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 通道 1~4 配置地址低位字节

位	名称	复位	读/写	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	读/写	描述
7: 5	—	000	R/W0	未使用。
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 通道传送未完成 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 通道传送未完成 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 通道传送未完成 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 通道传送未完成 1: DMA 通道传送完成/中断未决

9 定时器 1 (16 位定时器)

定时器 1 是一个支持典型定时器/计数器功能（比如输入捕获、输出比较和 PWM 功能）的独立 16 位定时器。它有 5 个独立的捕获/比较通道。每个通道使用一个 I/O 引脚。该定时器用于范围广泛的控制和测量应用，5 个通道具备正计数/倒计数模式，将允许例如电机控制应用的实现。

定时器 1 的特征如下：

- 5 个捕获/比较通道
- 上升沿、下降沿或任何边沿输入捕获
- 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
- 1、8、32 或 128 时钟分频
- 在每个捕获/比较和最终计数上产生中断请求
- DMA 触发功能

9.1 16 位计数器

定时器 1 包含一个 16 位计数器，该计数器在每个有效时钟边沿递增或递减。有效时钟边沿周期由寄存器位 CLKCONCMD.TICKSPD 定义，它设置全局系统时钟划分，提供了一个从 0.25MHz 到 32MHz 的变量时钟 tick 频率（使用 32MHz 晶体振荡器作为时钟源）。这在定时器 1 中由 T1CTL.DIV 给定的分频值进一步划分。这个分频值可以为 1、8、32 或 128。因此当使用 32MHz 晶体振荡器作为系统时钟源时，定时器 1 使用的最低时钟频率为 1953.125Hz，最高时钟频率为 32MHz。当使用 16MHz RC 振荡器作为系统时钟源时，定时器 1 使用的最高时钟频率为 16MHz。

计数器可以运行为自由运行计数器、模计数器或者在中心对齐 PWM 里使用的正计数器/倒计数器。

可以通过 2 个 8 位 SFR 寄存器 T1CNTH 和 T1CNTL（分别包含高位字节和低位字节）读取 16 位计数器的值。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 读出。因此在读 T1CNTH 之前总是首先读取 T1CNTL。

所有对 T1CNTL 寄存器的写访问将复位 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。用 T1CTL 控制寄存器设置可以启动和停止该计数器。当 T1CTL.MODE 写入一个不是 00 的值时，计数器开始运行。如果 T1CTL.MODE 写为 00，计数器在写入该值的时候停止。

9.2 定时器 1 操作

控制寄存器 T1CTL 通常用来控制定时器操作。状态寄存器 T1STAT 保持中断标志。下面将描述不同的操作模式。

9.3 自由运行模式

在自由运行操作模式，计数器从 0x0000 开始，并且在每一个有效时钟边沿增加 1。如图 9-1 所示，当计数器到达 0xFFFF（溢出）时，计数器载入 0x0000 继续进行递增。当达到最终计数值 0xFFFF，IRCON.T1IF 和 T1STAT.OVFIF 标志都置位。如果 IEN1.T1EN 和相应的中断屏蔽位 TIMIF.OVFIM 被置位，将产生一个中断请求。自由运行模式用于产生独立的时间间隔和输出信号频率。

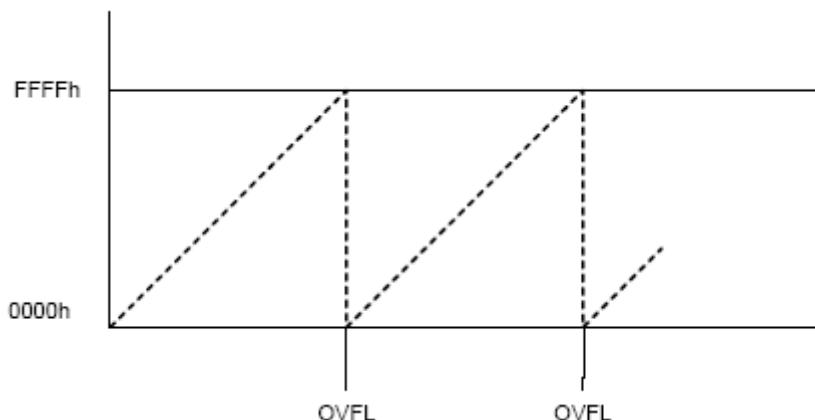


图 9-1 自由运行模式

9.4 模模式

当定时器运行于模模式，16 位计数器从 0x0000 开始，并且在每一个有效时钟边沿增加 1。当计数器达到保存在寄存器 T1CC0H: T1CC0L 里的最终计数值 T1CC0（溢出）时，计数器复位到 0x0000 继续进行递增。如果定时器从一个大于 T1CC0 的值开始，当到达最终计数值（0xFFFF）时，IRCON.T1IF 标志和 T1STAT.OVFIF 标志置位。如果 IEN1.T1EN 和相应的中断屏蔽位 TIMIF.OVFIM 被置位，将产生一个中断请求。模模式可以用在周期不必是 0xFFFF 的应用中。计数器操作如图 9-2 所示。

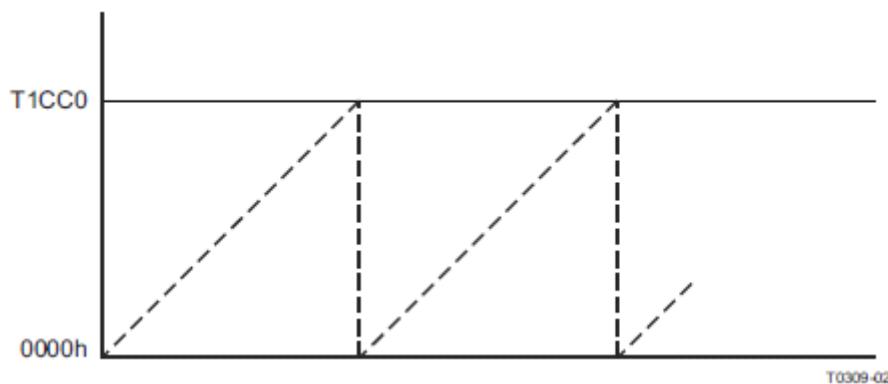


图 9-2 模模式

9.5 正计数/倒计数模式

在正计数/倒计数定时器模式，如图 9-3 所示，计数器反复从 0x0000 开始相加直达到保存在 T1CC0H: T1CC0L 里的值，然后计数器倒计数直到 0x0000。当要求带有周期的对称输出脉冲，而不是 0xFFFF 时，使用这种定时器模式，因而允许中心对齐 PWM 输出应用的实现。在这种模式下，当计数器值达到 0x0000 时，IRCON.T1IF 标志和 T1STAT.OVFIF 标志置位。如果 IEN1.T1EN 和相应的中断屏蔽位 TIMIF.OVFIM 被置位，将产生一个中断请求。

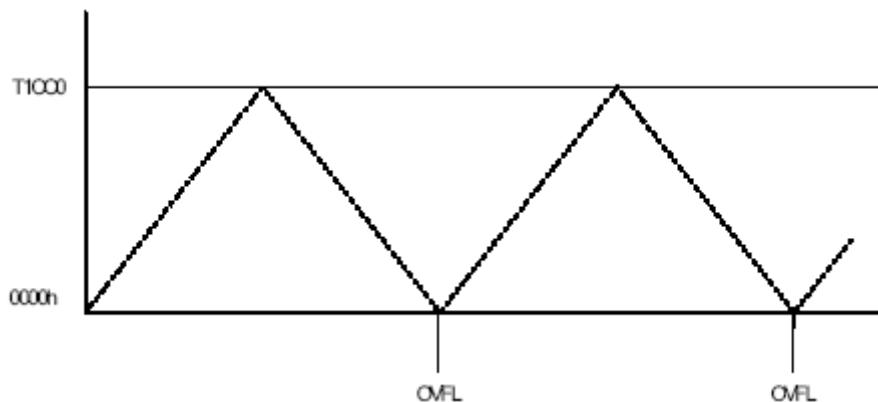


图 9-3 正计数/倒计数模式

9.6 通道模式控制

通道模式在每个通道对应的控制和状态寄存器 T1CCTL_n 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

当一个通道被配置为输入捕获通道，和该通道相连的 I/O 引脚配置为输入。定时器启动后，输入引脚上的上升沿、下降沿或任何边沿都将触发一个捕获，即 16 位计数器的内容捕获到相关的捕获寄存器中。因此，该定时器能捕获一个外部事件发生的时间。

注意：在定时器使用 I/O 引脚前，要求 I/O 引脚必须配置为定时器的外部设备引脚。

通道输入引脚和内部系统时钟同步。因此输入引脚上脉冲的最小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 读出。

当捕获发生时，IRCON.T1IF 标志和通道的中断标志 T1STAT.CHnIF (n 为通道号码) 一起设置。如果在设置 IEN1.T1EN 的时候，相应的中断屏蔽位 T1CCTL_n.IM 置 1，将产生一个中断请求。

9.8 输出比较模式

在输出比较模式，和该通道相连的 I/O 引脚配置为输出。定时器启动后，对计数器里的

内容和通道比较寄存器 T1CCnH: T1CCnL 的内容进行比较。如果比较寄存器和计数器的内容一样，输出引脚置 1，根据由 T1CCTLn.CMP 设置的比较输出模式进行复位或者切换。注意：当输出引脚运行在一个给定的输出比较模式下，它上面的所有边沿都是自由电子脉冲。写入比较寄存器 T1CCnL 是被缓冲的，这样在相应的高位寄存器 T1CCnH 写入之前，写给 T1CCnL 的值将不起作用。在计数器值为 0x00 之前，写入比较寄存器 T1CCnH: T1CCnL 对于输出比较值都不起作用。

注意：因为在模式 6 和模式 7 里 T1CC0H: T1CC0L 有一个特殊功能，也就是说对于通道 0 来说这 2 个模式是不可用的，所以通道 0 的输出比较模式较少。

当发生一个比较时，IRCON.T1IF 和通道的中断标志 T1STAT.CHnIF (n 为通道号码) 置 1。如果在设置 IEN1.T1EN 的时候，相应的中断屏蔽位 T1CCTLn.IM 置 1，将产生一个中断请求。

下面的图给出了不同定时器模式下输出比较的例子。

边沿对齐：如图 9-4 所示，使用定时器模模式且通道 1 和 2 在输出比较模式 6 或 7 (由 T1CCTLn.CMP 位定义，n 为 1 或者 2) 下，将产生一个 PWM 输出信号。T1CC0 的设置决定 PWM 信号周期，T1CCn (n 代表 PWM 通道 1 或 2) 的设置决定占空比。

也可以使用定时器自由运行模式。当使用自由运行模式，CLKCONCMD.TICKSPD 和在 T1CTL.DIV 位的预分频划分值一起设置 PWM 信号周期。PWM 信号的极性由使用的输出比较模式是 6 或 7 来决定。

如图 9-4 所示，使用输出比较模式 4 和 5 也可以产生 PWM 输出信号，如图 9-5 所示，使用模模式也可以产生 PWM 输出信号。对于简单的 PWM，最好使用输出比较模式 4 和 5。

中心对齐：选择定时器正计数/倒计数模式可以产生 PWM 输出。通道输出比较模式选择 4 或 5 (由 T1CCTLn.CMP 位定义，n 为 1 或 2) 取决于 PWM 信号的极性要求。T1CC0 决定 PWM 信号周期，T1CCn (n 代表 PWM 通道 1 或 2) 决定通道输出的占空比。

某些类型的电机驱动应用要求使用中心对齐 PWM 模式，因为 I/O 引脚传输不是排在同一个时钟边沿上，所以通常中心对齐 PWM 模式比边沿对齐 PWM 模式产生的噪声要少。

在某些应用类型中，需要在输出间定义一个延迟或停滞时间。通常这用于输出驱动一个 H 桥配置的时候，以此避免在 H 桥的一边交叉传导失控。使用 T1CCn 可以获得在 PWM 输出中的延迟或停滞时间，如下所示：

假设通道 1 和通道 2 使用定时器正计数/倒计数模式，用于驱动输入，且这 2 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：

$$t_p = T1CC0 \times 2$$

那么停滞时间即 2 个输出都为低的时间 (定时器 1 的时钟周期) 是：

$$t_d = T1CC1 - T1CC2$$

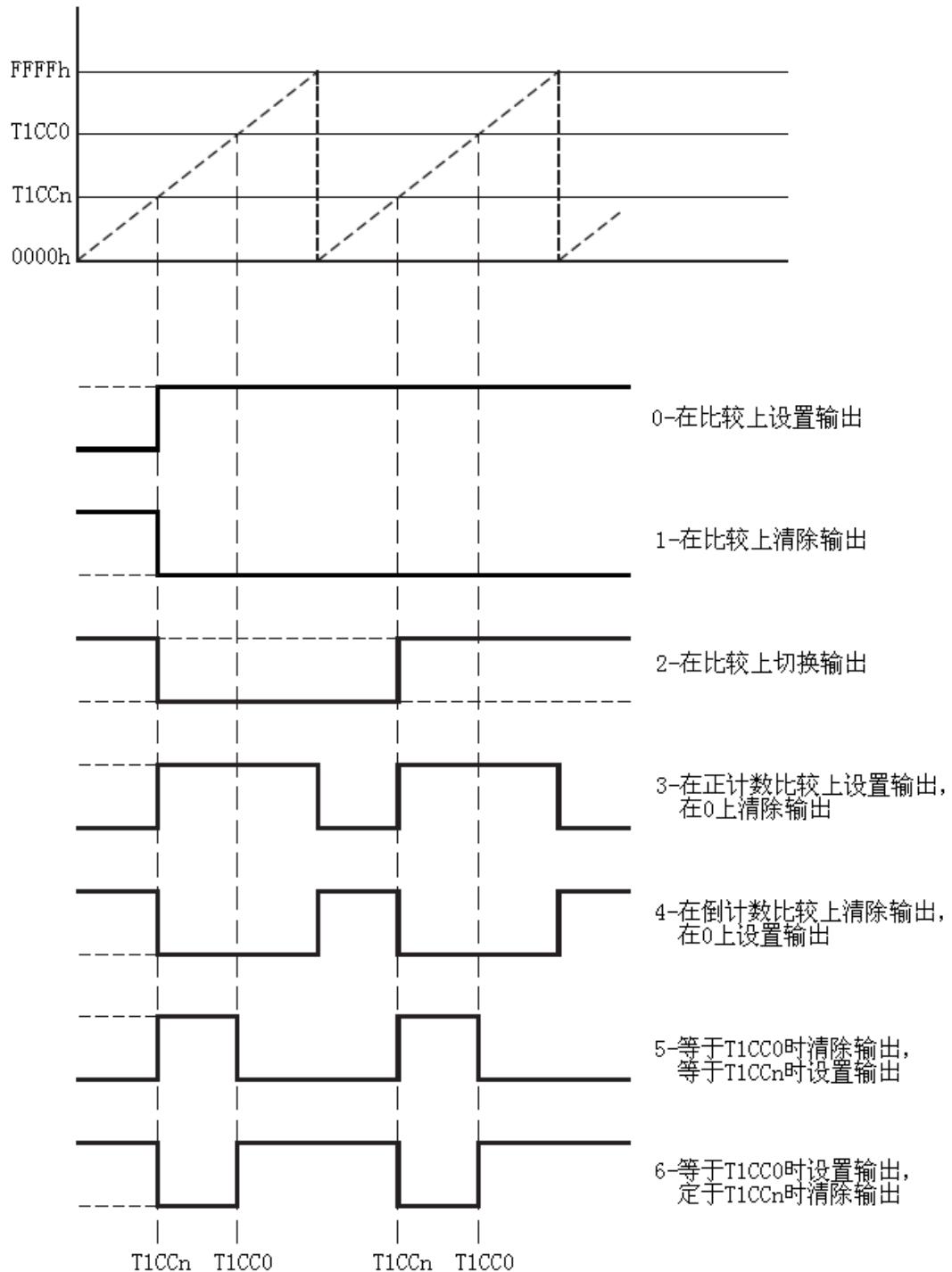
当发生下列情况时，比较输出引脚初始化为表 9-1 中列出的值：

- 一个值被写入 T1CNTL (所有的定时器 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
在比较上设置输出 (000)	0
在比较上清除输出 (001)	1
在比较上切换输出 (010)	0

在正计数/倒计数模式下，在正计数比较上设置输出，在倒计数比较上清除输出（011）	0
不是正计数/倒计数模式下，在比较上设置输出，在0上清除输出（011）	0
在正计数/倒计数模式下，在正计数比较上清除输出，在倒计数比较上设置输出（100）	1
不是正计数/倒计数模式下，在比较上清除输出，在0上设置输出（100）	1
等于T1CC0时清除输出，等于T1CCn时设置输出（101）	0
等于T1CC0时设置输出，等于T1CCn时清除输出（110）	1



T0311-01

图 9-4 输出比较模式，定时器自由运行模式

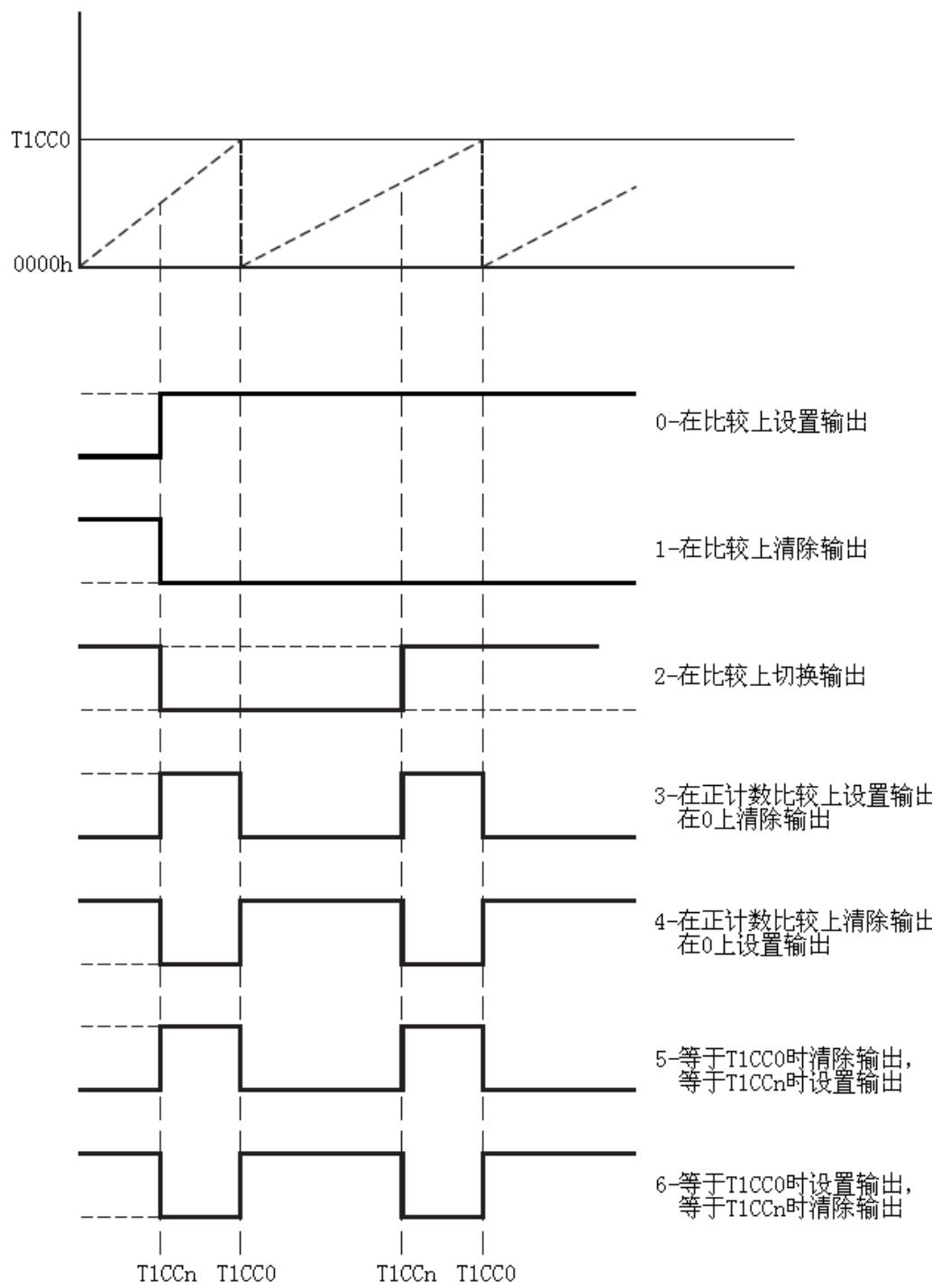


图 9-5 输出比较模式, 定时器模模式

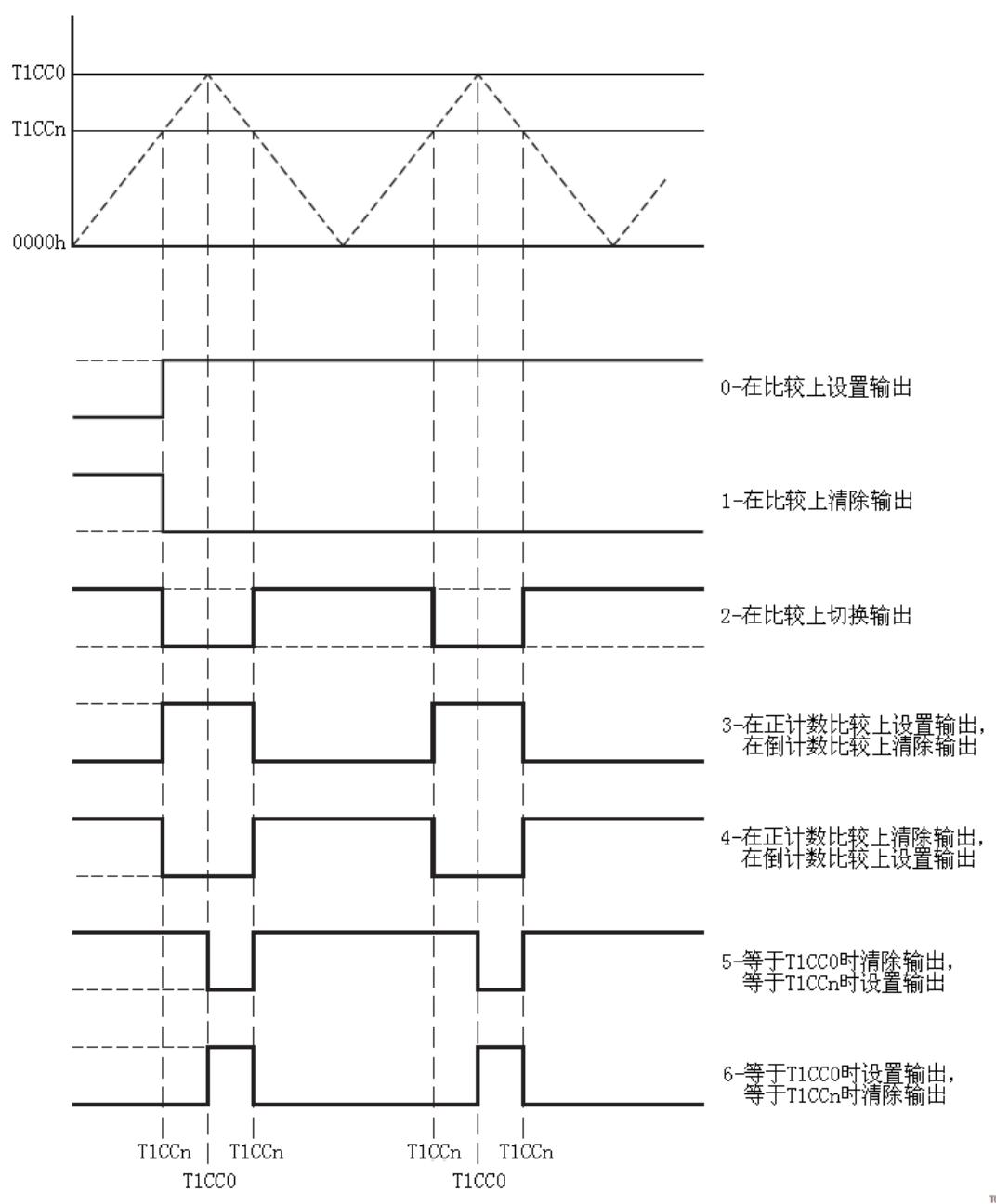


图 9-6 输出比较模式，定时器正计数/倒计数模式

9.9 红外信号产生和学习

本节描述 CC253x 设备只需最少的软件参与就可产生红外的功能。

9.9.1 简介

产生远程控制红外信号通常是通过下面两种方式之一：

- 调制码

- 非调制编码（C 码，Flash 代码）

CC253x 包含灵活的定时器功能，以最少的 CPU 参与来实现这两种类型红外信号的产生和学习。每条指令只需要一个 CPU 的干预就可实现多数红外协议。

9.9.2 调制码

使用定时器 1（16 位）和定时器 3（8 位）可以产生调制码。在模模式，定时器 3 用于产生载波。定时器 3 有一个用于输入的独立预分频器。由 TCCC0 来设置它的周期。定时器 3 通道 1 用于 PWM 输出。由 T3CC1 来设置载波的占空比。通道 1 使用比较模式：“在比较上设置输出，在 0xFF 上清除输出”（T3CCTL1.CMP=101）。表 9-2 显示了使用定时器 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	32,000kHz
红外载波频率	38kHz
系统时钟周期	0.00003125ms
红外载波周期	0.026315789ms
定时器预分频	4
定时器周期	0.000125ms
理想定时器值	210.5263158ms
实际定时器值	211
实际定时器周期	0.026375ms
实际定时器频率	37.91469194kHz
周期误差	59.21052632ns
频率误差	85.30805687Hz
频率误差%	0.2245%

寄存器位 IRCTL.IRGEN 使能定时器 1 里的红外产生模式。如果 IRGEN 位为 1，定时器 1 使用定时器 3 通道 1 的输出比较信号作为 tick，而不使用系统 tick。使用 T1CC0 来设置定时器 1 周期，定时器 1 处于模模式（T1CTL.MODE=10）和通道 0 处于比较模式（T1CCTL0.MODE=1）。通道 1 比较模式“在比较上设置输出，在 0x0000 上清除输出”（T1CCTL1.CMP=011）用于门控信号的输出。

使用 T1CC1.T1CC1 设置标志载波周期的个数，在每个定时器 1 周期需要通过 DMA 或者 CPU 来对 T1CC1.T1CC1 进行更新。注意对 T1CC1 的更新是被缓存的，在定时器 1 达到 0x0000 之前是不起作用的。

空间载波周期的个数通过 T1CC0 来设置。其值必须设置为标记载波周期和空间载波周期期望的总数。在定时器达到 0x0000 之前比较值一直是缓存的。

如图 9-7 所示，定时器 1 通道 1 的输出与定时器 3 通道 1 的输出进行与运算，作为红外输出。

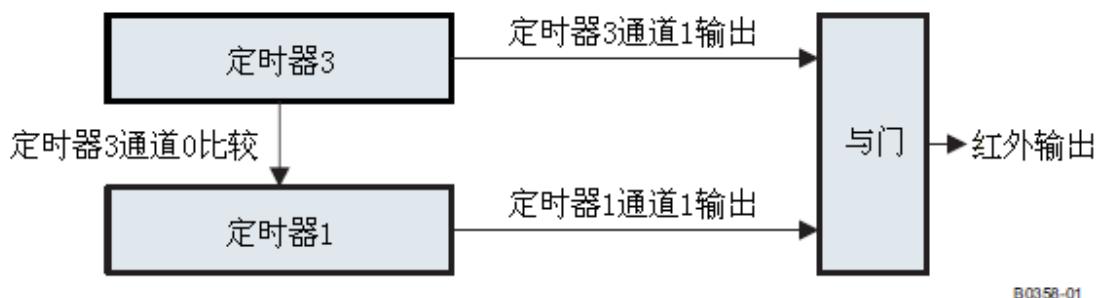


图 9-7 定时器在红外产生模式的框图

定时器3通道1输出信号和定时器1通道1输出信号的时序是同步的，这样在红外输出信号上就不会产生电子脉冲。

当IRGEN位为1时，红外输出信号被发送到引脚，而不是发送到正常的定时器1通道1输出（可参见7.6.1小节）。

图9-8为定时器3被初始化为33%占空比($T3CC0=3 \times T3CC1$)的例子。定时器1已经被初始化为3。

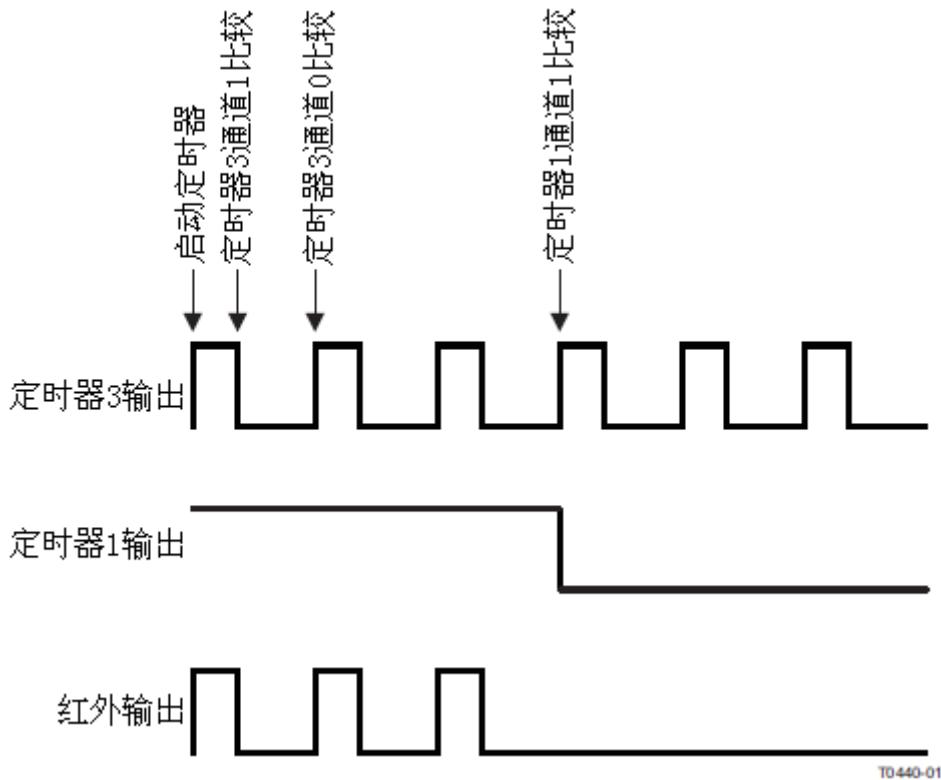


图 9-8 调制波形示例

要仅仅实现一个空间周期，T1CC1应当设置为0x00。

9.9.3 非调制编码

要产生非调制红外编码，定时器1要处于模模式。由T1CC0给定信号周期，由T1CC1给定脉冲宽度。T1CC1设置标记周期的长度，T1CC0设置标记周期和空间周期的总数。在定时器达到0x0000之前比较值一直是缓存的。如果比较值不能保持不变，那么在每个周期都必须由DMA或者CPU进行更新。

9.9.4 学习

使用定时器 1（16 位）和定时器 3（8 位）的捕获功能来完成学习。定时器 3 可以处理载波频率检测，定时器 1 可以处理调制信号的编码学习。电路如图 9-9 所示。

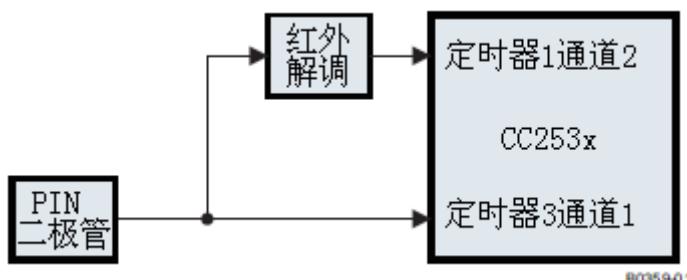


图 9-9 红外学习框图

9.9.4.1 载波频率检测

定时器 3 用于捕获和检测直接从红外 PIN 二极管的输入的载波频率。定时器应当对载波进行一定次数的采样。如果检测到载波，应当对被检测的频率进行计算得出一个平均值，这个平均值可以存储在数据库中。

9.9.4.2 解调编码学习

由一个合适的电路对红外 PIN 二极管的输出进行解调。该电路的输出当作处于捕获模式的定时器 1 某个通道的输入。

9.9.5 其它注意事项

红外输出引脚在复位期间应当处于三态或下拉状态，以避免因点亮红外 LED 而产生的不必要的功耗。注意，只有定时器 1 通道 1 的 P1.1 输出在复位期间和复位后是三态的，而没有上拉。

9.10 定时器 1 中断

为定时器 1 分配了一个中断向量。当下面任何一个定时器事件发生时，将产生一个中断请求：

- 计数器达到最终计数值（溢出或者在 0 附近）
- 输入捕获事件
- 输出比较事件

寄存器状态寄存器 T1STAT 包含最终计数值事件中断标志，以及 5 个通道的比较/捕获

事件的中断标志。只有在设置 IEN1.T1EN 时相应的中断屏蔽位置位时才能产生一个中断请求。通道 n 的中断屏蔽位为 T1CCTLn.IM，溢出事件的中断屏蔽位为 TIMIF.OVFIM。如果有其它未决中断，在产生一个新的中断请求之前，必须通过软件清除相应的中断标志。此外，如果相应的中断标志置位，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

与定时器 1 关联的 DMA 触发有 3 个。3 个 DMA 触发为 T1_CH0, T1_CH1 和 T1_CH2, 它们产生于以下的定时器比较事件：

- 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/比较值高位
- T1CCnL—定时器 1 通道 n 捕获/比较值低位

寄存器位 TIMIF.OVFIM 驻留在 TIMIF 寄存器中，将在后面和定时器 3、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	读/写	描述
7: 0	CNT[15: 8]	0x00	R	定时器计数高位字节。包括在读 T1CNTL 时缓冲的 16 位定时器计数器高位字节。

T1CNTL (0xE2) —定时器 1 计数器低位

位	名称	复位	读/写	描述
7: 0	CNT[7: 0]	0x00	R/W	定时器计数低位字节。包括 16 位定时器计数器的低位字节。往寄存器中写任何值都导致计数器被清除为 0x0000，并且初始化所有相关通道的输出引脚。

T1CTL (0xE4) —定时器 1 控制

位	名称	复位	读/写	描述
7: 4	—	0000 0	R0	保留
3: 2	DIV[1: 0]	00	R/W	预分频划分值。产生有效时钟边沿来更新计数器如下： 00: Tick 频率/1

				01: Tick 频率/8 10: Tick 频率/32 11: Tick 频率/128
1: 0	MODE[1: 0]	00	R/W	定时器 1 模式选择。定时器操作模式的选择如下： 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, 再从 T1CC0 倒计数到 0x0000, 反复计数

T1STAT (0xAF) —定时器 1 状态

位	名称	复位	读/写	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模式或模模式下达到最终计数值时置位，在正计数/倒计数模式，当倒计数达到 0 时置位。写 1 无效。
4	CH4IF	0	R/W0	定时器 1 通道 4 中断标志。当通道 4 中断条件发生时置位。写 1 无效。
3	CH3IF	0	R/W0	定时器 1 通道 3 中断标志。当通道 3 中断条件发生时置位。写 1 无效。
2	CH2IF	0	R/W0	定时器 1 通道 2 中断标志。当通道 2 中断条件发生时置位。写 1 无效。
1	CH1IF	0	R/W0	定时器 1 通道 1 中断标志。当通道 1 中断条件发生时置位。写 1 无效。
0	CH0IF	0	R/W0	定时器 1 通道 0 中断标志。当通道 0 中断条件发生时置位。写 1 无效。

T1CCTL0 (0xE5) —定时器1通道0捕获/比较控制

位	名称	复位	读/写	描述
7	RFIRQ	0	R/W	当该位为 1 时, 使用 RF 中断来捕获, 而不是常规捕获输入。
6	IM	1	R/W	通道 0 中断使能。置位时使能中断请求。
5: 3	CMP[2: 0]	000	R/W	通道 0 比较模式选择。当定时器值等于 T1CC0 中的比较值, 选择输出上的动作。 000: 在比较上设置输出 001: 在比较上清除输出 010: 在比较上切换输出 011: 在正计数比较上设置输出, 在 0 上清除输出 100: 在正计数比较上清除输出, 在 0 上设置输出 101: 未使用 110: 未使用 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: 不捕获 01: 在上升沿捕获 10: 在下降沿捕获 11: 在所有边沿上捕获

T1CC0H (0xDB) —定时器 1 通道 0 捕获/比较值高位

位	名称	复位	读/写	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) 时写该寄存器，导致在 T1CNT=0x0000 之前，T1CC0[15: 0]更新为写入值一直被延迟。

T1CC0L (0xDA) —定时器 1 通道 0 捕获/比较值低位

位	名称	复位	读/写	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。写入该寄存器的数据存储在缓冲区，但不写入 T1CC0[7: 0]，直到或者在稍后写入 T1CC0H 时才生效。

T1CCTL1 (0xE6) —定时器 1 通道 1 捕获/比较控制

位	名称	复位	读/写	描述
7	RFIRQ	0	R/W	当该位为 1 时，使用 RF 中断来捕获，而不是常规捕获输入。
6	IM	1	R/W	通道 1 中断屏蔽。置位时使能中断请求。
5: 3	CMP[2: 0]	000	R/W	通道 1 比较模式选择。当定时器值等于 T1CC1 中的比较值，选择输出上的动作。 000: 在比较上设置输出 001: 在比较上清除输出 010: 在比较上切换输出 011: 在正计数/倒计数模式，在正计数比较上设置输出，在倒计数比较上清除输出。否则在比较上设置输出，在 0 上清除输出。 100: 在正计数/倒计数模式，在正计数比较上清除输出，在倒计数比较上设置输出。否则在比较上清除输出，在 0 上设置输出。 101: 等于 T1CC0 时清除输出，等于 T1CC1 时设置输出 110: 等于 T1CC0 时设置输出，等于 T1CC1 时清除输出 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 1 通道 1 捕获或比较模式 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: 不捕获 01: 在上升沿捕获 10: 在下降沿捕获 11: 在所有边沿上捕获

T1CC1H (0xDD) —定时器 1 通道 1 捕获/比较值高位

位	名称	复位	读/写	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，导致在 T1CNT=0x0000 之前，T1CC1[15: 0]更新为写入值一直被延迟。

T1CC1L (0xDC) —定时器 1 通道 1 捕获/比较值低位

位	名称	复位	读/写	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在缓冲区，但不写入 T1CC1[7: 0]，直到或者在稍后写入 T1CC1H 时才生效。

T1CCTL2 (0xE7) —定时器 1 通道 2 捕获/比较控制

位	名称	复位	读/写	描述
7	RFIRQ	0	R/W	当该位为 1 时，使用 RF 中断来捕获，而不是常规捕获输入。
6	IM	1	R/W	通道 2 中断屏蔽。置位时使能中断请求。
5: 3	CMP[2: 0]	000	R/W	通道 2 比较模式选择。当定时器值等于 T1CC2 中的比较值，选择输出上的动作。 000: 在比较上设置输出 001: 在比较上清除输出 010: 在比较上切换输出 011: 在正计数/倒计数模式，在正计数比较上设置输出，在倒计数比较上清除输出。否则在比较上设置输出，在 0 上清除输出。 100: 在正计数/倒计数模式，在正计数比较上清除输出，在倒计数比较上设置输出。否则在比较上清除输出，在 0 上设置输出。 101: 等于 T1CC0 时清除输出，等于 T1CC2 时设置输出 110: 等于 T1CC0 时设置输出，等于 T1CC2 时清除输出 111: 初始化输出引脚。CMP[2: 0]不变。
2	MODE	0	R/W0	模式。选择定时器 1 通道 2 捕获或比较模式 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: 不捕获 01: 在上升沿捕获 10: 在下降沿捕获 11: 在所有边沿上捕获

T1CC2H (0xDF) —定时器 1 通道 2 捕获/比较值高位

位	名称	复位	读/写	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，导致在 T1CNT=0x0000 之前，T1CC2[

				15: 0]更新为写入值一直被延迟。
--	--	--	--	--------------------

T1CC2L (0xDE) —定时器 1 通道 2 捕获/比较值低位

位	名称	复位	读/写	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在缓冲区，但不写入 T1CC2[7: 0]，直到或者在稍后写入 T1CC2H 时才生效。

T1CCTL3 (0x62A3) —定时器 1 通道 3 捕获/比较控制

位	名称	复位	读/写	描述
7	RFIRQ	0	R/W	当该位为 1 时，使用 RF 中断来捕获，而不是常规捕获输入。
6	IM	1	R/W	通道 3 中断屏蔽。置位时使能中断请求。
5: 3	CMP[2: 0]	000	R/W	通道 3 比较模式选择。当定时器值等于 T1CC3 中的比较值，选择输出上的动作。 000: 在比较上设置输出 001: 在比较上清除输出 010: 在比较上切换输出 011: 在正计数/倒计数模式，在正计数比较上设置输出，在倒计数比较上清除输出。否则在比较上设置输出，在 0 上清除输出。 100: 在正计数/倒计数模式，在正计数比较上清除输出，在倒计数比较上设置输出。否则在比较上清除输出，在 0 上设置输出。 101: 等于 T1CC0 时清除输出，等于 T1CC3 时设置输出 110: 等于 T1CC0 时设置输出，等于 T1CC3 时清除输出 111: 初始化输出引脚。CMP[2: 0]不变。
2	MODE	0	R/W0	模式。选择定时器 1 通道 3 捕获或比较模式 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: 不捕获 01: 在上升沿捕获 10: 在下降沿捕获 11: 在所有边沿上捕获

T1CC3H (0x62AD) —定时器 1 通道 3 捕获/比较值高位

位	名称	复位	读/写	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，导致在 T1CNT=0x0000 之前，T1CC3[15: 0]更新为写入值一直被延迟。

T1CC3L (0x62AC) —定时器 1 通道 3 捕获/比较值低位

位	名称	复位	读/写	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在缓冲区，但不写入 T1CC3[7: 0]，直到或者在稍后写入 T1CC

				3H 时才生效。
--	--	--	--	----------

T1CCTL4 (0x62A4) —定时器 1 通道 4 捕获/比较控制

位	名称	复位	读/写	描述
7	RFIRQ	0	R/W	当该位为 1 时，使用 RF 中断来捕获，而不是常规捕获输入。
6	IM	1	R/W	通道 4 中断屏蔽。置位时使能中断请求。
5: 3	CMP[2: 0]	000	R/W	通道 4 比较模式选择。当定时器值等于 T1CC2 中的比较值，选择输出上的动作。 000: 在比较上设置输出 001: 在比较上清除输出 010: 在比较上切换输出 011: 在正计数/倒计数模式，在正计数比较上设置输出，在倒计数比较上清除输出。否则在比较上设置输出，在 0 上清除输出。 100: 在正计数/倒计数模式，在正计数比较上清除输出，在倒计数比较上设置输出。否则在比较上清除输出，在 0 上设置输出。 101: 等于 T1CC0 时清除输出，等于 T1CC4 时设置输出 110: 等于 T1CC0 时设置输出，等于 T1CC4 时清除输出 111: 初始化输出引脚。CMP[2: 0]不变。
2	MODE	0	R/W0	模式。选择定时器 1 通道 4 捕获或比较模式 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: 不捕获 01: 在上升沿捕获 10: 在下降沿捕获 11: 在所有边沿上捕获

T1CC4H (0x62AF) —定时器 1 通道 4 捕获/比较值高位

位	名称	复位	读/写	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，导致在 T1CNT=0x0000 之前，T1CC4[15: 0]更新为写入值一直被延迟。

T1CC4L (0x62AE) —定时器 1 通道 4 捕获/比较值低位

位	名称	复位	读/写	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在缓冲区，但不写入 T1CC4[7: 0]，直到或者在稍后写入 T1CC4H 时才生效。

IRCTL (0x6281) —定时器 1 红外产生控制

位	名称	复位	读/写	描述
7: 1	—	0000 000	R/W	保留
0	IRGEN	0	R/W	该位置位时，定时器 3 通道 1 和定时器 1 tick 输入之间产生一个

				连接，这样定时器可以用来产生调制红外编码（见 9.9 小节）。
--	--	--	--	---------------------------------

9.13 以数组访问定时器寄存器

对定时器 1 捕获/比较通道寄存器的访问可以当作对 XDATA 存储器空间一个连续区域的访问。这样可以作为一个简单的索引结构来对寄存器进行访问。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 未使用。

10 定时器 3 和定时器 4 (8 位定时器)

定时器 3 和 4 是两个 8 位定时器。这 2 个定时器有 2 个独立的捕获/比较通道，每个通道使用一个 I/O 引脚。

定时器 3/4 的特征如下：

- 2 个捕获/比较通道
- 设置、清除或切换输出比较
- 进行 1、2、4、8、16、32、64、128 时钟分频
- 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

10.1 8 位定时器计数器

定时器 3 和定时器 4 的所有的定时器功能都是基于主要的 8 位计数器建立的。计数器在每个有效时钟边沿递增或递减。有效时钟边沿周期由寄存器位 CLKCONCMD.TICKSPD[2:0] 定义，它由 TxCTL.DIV[2:0] (x 指的是定时器号码 3 或 4) 设定的分频值进一步划分。计数器可以作为自由运行计数器、倒计数器、模计数器或正计数/倒计数器运行。

可以通过 SFR 寄存器 TxCNT (x 指的是定时器号码 3 或 4) 读取 8 位计数器值。

用 TxCTL 控制寄存器设置可以清除和停止该计数器。当 TxCTL.START 写入 1 时，计数器开始运行。如果 TxCTL.START 写入 0，计数器在写入该值的时候停止。

10.2 定时器 3/定时器 4 模式控制

控制寄存器 TxCTL 通常用来控制定时器操作。

10.2.1 自由运行模式

在自由运行操作模式，计数器从 0x00 开始，并且在每一个有效时钟边沿增加 1。当计数器到达 0xFF 时，计数器载入 0x00 继续进行递增。当达到最终计数值 0xFF (即发生溢出)，中断标志 TIMIF.TxOVFIF 置位。如果相应的中断屏蔽位 TxCTL.OVFIM 被置位，将产生一个中断请求。自由运行模式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，在定时器启动后，计数器加载 TxC0 里的内容。然后计数器倒计数到 0x00。当达到 0x00 时，标志 TIMIF.TxOVFIF 置位。如果相应的中断屏蔽位 TxCTL.OVFIM 被置位，将产生一个中断请求。定时器倒计数模式通常用于需要一个事件超时间隔的应用中。

10.2.3 模模式

当定时器运行于模模式，8位计数器从0x00开始，并且在每一个有效时钟边沿增加1。当计数器达到保存在寄存器TxCC0里的最终计数值时，计数器复位到0x00继续进行递增。当到达最终计数时，TIMIF.TxOVFIF标志置位。如果相应的中断屏蔽位TxCTL.OVFIM被置位，将产生一个中断请求。模模式可以用在周期不必是0xFF的应用中。

10.2.4 正计数/倒计数模式

在正计数/倒计数定时器模式，计数器反复从0x00开始相加直到达到保存在TxCC0里的值，然后计数器倒计数直到0x00。当要求带有周期的对称输出脉冲，而不是0xFF时，使用这种定时器模式，因而允许中心对齐PWM输出应用的实现。

通过写入TxCTL.CLR来清除计数器，这也将复位计数方向到从0x00开始正计数的模式。

10.3 通道模式控制

每个通道（0和1）的通道模式由控制和状态寄存器TxCCTLn设置（n指的是通道号码0或1）。设置包括捕获和比较模式。

10.4 输入捕获模式

当一个通道被配置为输入捕获通道，和该通道相连的I/O引脚配置为输入。定时器启动后，输入引脚上的上升沿、下降沿或任何边沿都将触发一个捕获，即8位计数器的内容捕获到相关的捕获寄存器中。因此，定时器3和定时器4能捕获一个外部事件发生的时间。

注意：在定时器使用I/O引脚前，要求I/O引脚必须配置为定时器3/定时器4的外部设备引脚。

通道输入引脚和内部系统时钟同步。因此输入引脚上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读出。

当捕获发生时，对应实际通道的中断标志TIMIF.TxCHnIF置位。如果相应的中断屏蔽位TxCCTLn.IM置1，将产生一个中断请求。

10.5 输出比较模式

在输出比较模式，和该通道相连的I/O引脚必须配置为输出。定时器启动后，对计数器里的内容和通道比较寄存器TxCC0n的内容进行比较。如果比较寄存器和计数器的内容一样，输出引脚置1，根据由TxCCTL.CMP1:0设置的比较输出模式进行复位或者切换。注意：当输出引脚运行在一个给定的比较输出模式下，它上面的所有边沿都是自由电子脉冲。

对于简单的PWM使用，输出比较模式4和5是首选。

在计数器值为 0x00 之前，写比较寄存器 TxC0 或 TxC1 对于输出比较值不起作用。

当发生一个比较时，对应实际通道的中断标志 TIMIF.TxCHnIF 置位。如果相应的中断屏蔽位 TxCCTLn.IM 置 1，将产生一个中断请求。

当发生下列情况时，比较输出引脚初始化为表 10-1 中列出的值：

- “1”被写入 TxCNTR.CLR（所有的定时器 x 通道）
- 0x7 被写入 TxCCTLn.CMP（定时器 x，通道 n）

表 9-1 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
在比较上设置输出（000）	0
在比较上清除输出（001）	1
在比较上切换输出（010）	0
在正计数/倒计数模式下，在正计数比较上设置输出，在倒计数比较上清除输出（011）	0
不是正计数/倒计数模式下，在比较上设置输出，在 0 上清除输出（011）	0
在正计数/倒计数模式下，在正计数比较上清除输出，在倒计数比较上设置输出（100）	1
不是正计数/倒计数模式下，在比较上清除输出，在 0 上设置输出（100）	1
在比较上设置输出，在 0xFF 上清除输出（101）	0
在比较上清除输出，在 0x00 上设置输出（110）	1

10.6 定时器 3 和定时器 4 中断

为定时器 3 和定时器 4 各分配了一个中断向量 T3 和 T4。当下面任何一个定时器事件发生时，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 寄存器位 TIMIF 包含了定时器 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 个最终计数值事件的中断标志，以及 4 个通道比较事件。只有相应的中断屏蔽位置位时才能产生一个中断请求。如果有其它未决中断，在产生一个新的中断请求之前，必须通过 CPU 清除相应的中断标志。此外，如果相应的中断标志置位，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

分别有 2 个 DMA 触发与定时器 3 和定时器 4 关联。这 4 个 DMA 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_CH1：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	读/写	描述
7: 0	CNT[7: 0]	0x00	R	定时器计数字节。包括 8 位计数器的当前值。

T3CTL (0xCB) —定时器 3 控制

位	名称	复位	读/写	描述
7: 5	DIV[2: 0]	000	R/W	预分频划分值。产生有效时钟边沿用于来自 CLKCONCMD.TICKSPD 的定时器时钟如下： 000: Tick 频率/1 001: Tick 频率/2 010: Tick 频率/4 011: Tick 频率/8 100: Tick 频率/16 101: Tick 频率/32 110: Tick 频率/64 111: Tick 频率/128
4	START	0	R/W	启动定时器。置位时正常运行，清除时暂停。
3	OVFIM	1	R/W0	溢出中断屏蔽 0: 禁止中断 1: 使能中断
2	CLR	0	R0/W1	清除计数器。写 1 复位计数器为 0x00，并且初始化所有相关通道的输出引脚。总是读为 0。
1: 0	MODE[1: 0]	00	R/W	定时器 3 模式。模式的选择如下： 00: 自由运行，从 0x00 到 0xFF 反复计数 01: 倒计数，计数从 T3CC0 到 0x00 10: 模，从 0x00 到 T3CC0 反复计数 11: 正计数/倒计数，从 0x00 到 T3CC0 反复计数，从 T3CC0 倒计数到 0x00

T3CCTL0 (0xCC) —定时器 3 通道 0 捕获/比较控制

位	名称	复位	读/写	描述
7	—	0	R0	未使用。
6	IM	1	R/W	通道 0 中断屏蔽。 0: 禁止中断 1: 使能中断
5: 3	CMP[2: 0]	000	R/W	通道 0 比较输出模式选择。当定时器值等于 T3CC0 中的比较值，指定输出上的动作。 000: 在比较上设置输出 001: 在比较上清除输出 010: 在比较上切换输出

				011: 在正计数比较上设置输出，在 0 上清除输出 100: 在正计数比较上清除输出，在 0 上设置输出 101: 在比较上设置输出，在 0xFF 上清除输出 110: 在比较上清除输出，在 0x00 上设置输出 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 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: 不捕获 01: 在上升沿捕获 10: 在下降沿捕获 11: 在两种边沿上都捕获

T3CC0 (0xCD) — 定时器 3 通道 0 捕获/比较值

位	名称	复位	读/写	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，导致在 T3CNT.CNT[7: 0]=0x00 之前，T3CC0.VAL[7: 0]更新为写入值一直被延迟。

T3CCTL1 (0xCE) — 定时器 3 通道 1 捕获/比较控制

位	名称	复位	读/写	描述
7	—	0	R0	未使用。
6	IM	1	R/W	通道 1 中断屏蔽。 0: 禁止中断 1: 使能中断
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 T3CC1 中的比较值，指定输出上的动作。 000: 在比较上设置输出 001: 在比较上清除输出 010: 在比较上切换输出 011: 在正计数/倒计数模式，在正计数比较上设置输出，在倒计数比较上清除输出。否则在比较上设置输出，在 0 上清除输出 100: 在正计数/倒计数模式，在正计数比较上清除输出，在倒计数比较上设置输出。否则在比较上清除输出，在 0 上设置输出 101: 在比较上设置输出，在 0xFF 上清除输出 110: 在比较上清除输出，在 0x00 上设置输出 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 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: 不捕获 01: 在上升沿捕获 10: 在下降沿捕获 11: 在两种边沿上都捕获
------	-----------	----	-----	--

T3CC1 (0xCF) —定时器 3 通道 1 捕获/比较值

位	名称	复位	读/写	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, 导致在 T3CNT.CNT[7: 0]=0x00 之前, T3CC1.VAL[7: 0]更新为写入值一直被延迟。

T4CNT (0xEA) —定时器 4 计数器

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

T4CTL (0xEB) —定时器 4 控制

位	名称	复位	读/写	描述
7: 5	DIV[2: 0]	000	R/W	预分频划分值。产生有效时钟边沿用于来自 CLKCONCMD.TICKSPD 的定时器时钟如下: 000: Tick 频率/1 001: Tick 频率/2 010: Tick 频率/4 011: Tick 频率/8 100: Tick 频率/16 101: Tick 频率/32 110: Tick 频率/64 111: Tick 频率/128
4	START	0	R/W	启动定时器。置位时正常运行, 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 复位计数器为 0x00, 并且初始化所有相关通道的输出引脚。总是读为 0。
1: 0	MODE[1: 0]	0	R/W	定时器 4 模式。模式的选择如下: 00: 自由运行, 从 0x00 到 0xFF 反复计数 01: 倒计数。计数从 T4CC0 到 0x00 10: 模, 从 0x00 到 T4CC0 反复计数 11: 正计数/倒计数, 从 0x00 到 T4CC0 反复计数, 从 T4CC0 倒计数到 0x00

T4CCTL0 (0xEC) —定时器 4 通道 0 捕获/比较控制

位	名称	复位	读/写	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。当定时器值等于 T4CC0 中的比较值，指定输出上的动作。 000: 在比较上设置输出 001: 在比较上清除输出 010: 在比较上切换输出 011: 在正计数比较上设置输出，在 0 上清除输出 100: 在正计数比较上清除输出，在 0 上设置输出 101: 在比较上设置输出，在 0xFF 上清除输出 110: 在比较上清除输出，在 0x00 上设置输出 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 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: 不捕获 01: 在上升沿捕获 10: 在下降沿捕获 11: 在两种边沿上都捕获

T4CC0 (0xED) — 定时器 4 通道 0 捕获/比较值

位	名称	复位	读/写	描述
7: 0	VAL[7: 0]	0x00	R/W	定时器捕获/比较值通道 0。当 T4CCTL0.MODE=1 (比较模式) 时写该寄存器，导致在 T4CNT.CNT[7: 0]=0x00 之前，T4CC0.VAL[7: 0]更新为写入值一直被延迟。

T4CCTL1 (0xEE) — 定时器 4 通道 1 捕获/比较控制

位	名称	复位	读/写	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 T4CC1 中的比较值，指定输出上的动作。 000: 在比较上设置输出 001: 在比较上清除输出 010: 在比较上切换输出 011: 在正计数/倒计数模式，在正计数比较上设置输出，在倒计数比较上清除输出。否则在比较上设置输出，在 0 上清除输出 100: 在正计数/倒计数模式，在正计数比较上清除输出，在倒计数比较上设置输出。否则在比较上清除输出，在 0 上设置输出 101: 在比较上设置输出，在 0xFF 上清除输出 110: 在比较上清除输出，在 0x00 上设置输出 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 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: 不捕获 01: 在上升沿捕获 10: 在下降沿捕获 11: 在两种边沿上都捕获

T4CC1 (0xEF) —定时器 4 通道 1 捕获/比较值

位	名称	复位	读/写	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，导致在 T4CNT.CNT[7: 0]=0x00 之前，T4CC1.VAL[7: 0]更新为写入值一直被延迟。

TIMIF (0xD8) —定时器 1/3/4 中断屏蔽/标志

位	名称	复位	读/写	描述
7	—	0	R0	未使用。
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

11 睡眠定时器

睡眠定时器用来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，保持定时器 2 的定时。

睡眠定时器的主要特征如下：

- 24 位定时器正计数器，运行于 32kHz 时钟
- 24 位具有中断和 DMA 触发的比较
- 24 位捕获

11.1 概述

睡眠定时器是一个运行于 32kHz 时钟（RC 或晶体振荡器）的 24 位定时器。定时器在复位后立即启动并连续运行不间断。定时器的当前值可以从 SFR 寄存器 ST2: ST1: ST0 读取。

11.2 定时器比较

当定时器值等于 24 位比较值时发生一次定时器比较。通过写寄存器 ST2: ST1: ST0 来设置比较值。当发生一次定时器比较时，中断标志 STIF 起作用。当 STLOAD.LDRDY 为 1 时写 ST0 会启动装载新的比较值，即将最新的比较值写入 ST2、ST1 和 ST0 寄存器。装载新的比较值的过程中，STLOAD.LDRDY 为 0，在 STLOAD.LDRDY 返回 1 之前通过软件不能启动新的加载。读 ST0 就是捕获 24 位计数器的当前值。因此，在读取 ST1 和 ST2 之前必须先读取 ST0 寄存器，以捕获正确的睡眠定时器计数值。当发生定时器比较时，中断标志 STIF 被置位。每次系统时钟检测到 32kHz 的一个正边沿时，就对当前定时器值进行更新。因此，如果未检测到 32kHz 时钟上的正边沿，当系统从 PM1/2/3（系统时钟关闭）返回时，ST2: ST1: ST0 中的睡眠定时器值就不进行更新。为了确保读出的值为更新值，在读取睡眠定时器值之前，通过轮询 SLEEPSTA.CLK32K 位来等待 32kHz 时钟的正边沿。

睡眠定时器中断的中断使能位是 IEN0.STIE，中断标志是 IRCON.STIF。

当系统运行在除了 PM3 之外的所有功耗模式下，睡眠定时器都将运行。因此，在 PM3 模式下，睡眠定时器的值不保存。在 PM1 和 PM2 模式下，睡眠定时器比较事件用于唤醒设备并返回到主动运行模式的主动运行。复位之后比较值的默认值为 0xFFFFFFF。

睡眠定时器比较还可以用来作为一个 DMA 触发（表 8-1 里的 DMA 触发 11）。

注意，如果在进入 PM2 时，电源电压下降到低于 2V，睡眠定时器间隔可能会受到影响。

11.3 定时器捕获

当选定的 I/O 引脚的中断标志已经置位，并且 32kHz 时钟已经检测到这个事件时，发生定时器捕获。通过设置将要被用于触发捕获的 I/O 引脚的 STCC.PORT[1: 0] 和 STCC.PIN[2: 0] 来使能睡眠定时器捕获。当 STCS.VALID 为 1 时，可以读取 STCV2: STCV1: STCV0 里的捕获值。捕获到的数值比发生在 I/O 引脚上的事件的瞬间值要大。因此如果要求绝对时序，

那么软件应当将捕获的值减 1。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 为低。
3. 等待，直到 SLEEPSTA.CLK32K 为高。
4. 清除 P0IFG/P1IFG/P2IFG 寄存器里的引脚中断标志。

如图 11-1 所示，这个顺序以使用 P0.0 的上升沿为例。

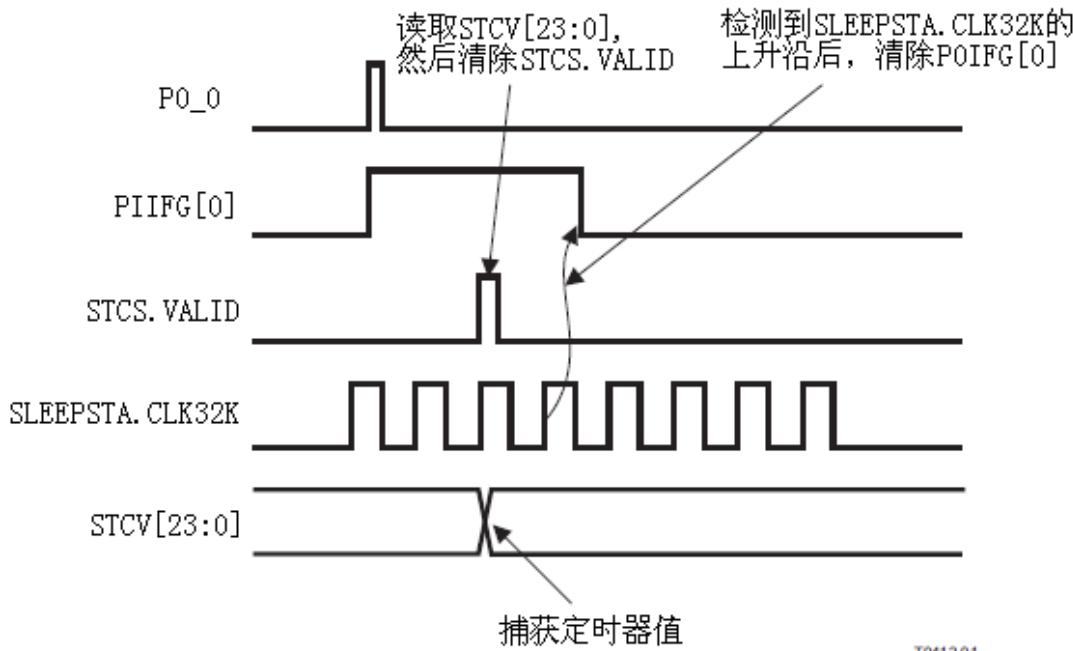


图 11-1 睡眠定时器捕获（以使用 **P0_0 的上升沿为例）**

捕获使能时，不能切换输入捕获引脚。在选择一个新的输入捕获引脚前，必须禁止捕获。要禁止捕获，遵循以下步骤（如果禁止了中断，最高使用 32kHz 周期的一半即可（约 15.25us））：

1. 禁止中断
2. 等待，直到 SLEEPSTA.CLK32K 为高。
3. 设置 STCC.PORT[1: 0] 为 3。这会禁止捕获。

11.4 睡眠定时器寄存器

睡眠定时器使用的寄存器有：

- ST2—睡眠定时器 2
- ST1—睡眠定时器 1
- ST0—睡眠定时器 0
- STLOAD—睡眠定时器装载状态
- 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) —睡眠定时器 2

位	名称	复位	读/写	描述
7: 0	ST2[7: 0]	0x00	R/W	睡眠定时器计数/比较值。读取时，寄存器返回睡眠定时器计数的高位[23: 16]。写该寄存器时设置比较值的高位[23: 16]。当读寄存器 ST0 时，读取值是锁定的。当写 ST0 时，写入值是锁定的。

ST1 (0x96) —睡眠定时器 1

位	名称	复位	读/写	描述
7: 0	ST1[7: 0]	0x00	R/W	睡眠定时器计数/比较值。读取时，寄存器返回睡眠定时器计数的中位[15: 8]。写该寄存器时设置比较值的中位[15: 8]。当读寄存器 ST0 时，读取值是锁定的。当写 ST0 时，写入值是锁定的。

ST0 (0x95) —睡眠定时器 0

位	名称	复位	读/写	描述
7: 0	ST0[7: 0]	0x00	R/W	睡眠定时器计数/比较值。读取时，寄存器返回睡眠定时器计数的低位[7: 0]。写该寄存器时设置比较值的低位[7: 0]。除非 STLOAD.LDRDY 为 1，否则忽略对该寄存器的写。

STLOAD (0xAD) —睡眠定时器装载状态

位	名称	复位	读/写	描述
7: 1	—	0000 000	R0	保留
0	LDRDY	1	R	装载就绪。当睡眠定时器装载 24 位比较值时，该位为 0；当睡眠定时器准备开始装载一个新的比较值时，该位为 1。

STCC (0x62B0) —睡眠定时器捕获控制

位	名称	复位	读/写	描述
7: 5	—	000	R0	保留
4: 3	PORT[1: 0]	11	R	端口选择。0-2 为有效设置。当该位为 3 时，捕获禁止，即选择了无效设置。
2: 0	PIN[2: 0]	111		引脚选择。当 PORT[1: 0] 为 0 或 1 时，0-7 为有效设置；当 PORT[1: 0] 为 2 时，0-5 为有效设置。如果选择了无效设置，捕获禁止。

STCS (0x62B1) —睡眠定时器捕获状态

位	名称	复位	读/写	描述
7: 1	—	0000 000	R0	保留
0	VALID	0	R/W0	捕获有效标志。当 STCV 里的捕获值被更新时该位为 1。完全清除以允许新的捕获。



CC253x 用户指南

STCV0 (0x62B2) —睡眠定时器捕获值字节 0

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

STCV0 (0x62B3) —睡眠定时器捕获值字节 1

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

12 ADC

ADC 支持多达 14 位模数转换，有效位数（ENOB）多达 12 位。ADC 包含一个具有多达 8 个独立配置通道的模拟多路转换器，一个参考电压发生器，并且通过 DMA 将转换结果写入存储器。具有多种运行模式。

12.1 ADC 简介

ADC 支持多达 14 位模数转换，有效位数（ENOB）多达 12 位。ADC 包含一个具有多达 8 个独立配置通道的模拟多路转换器，参考电压发生器，并且通过 DMA 将转换结果写入存储器。具有多种运行模式。

ADC 的主要特征如下：

- 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，单端或差分
- 参考电压可选为内部、外部单端、外部差分或 AVDD5
- 中断请求产生
- 转换结束时 DMA 触发
- 温度传感器输入
- 电池测量能力

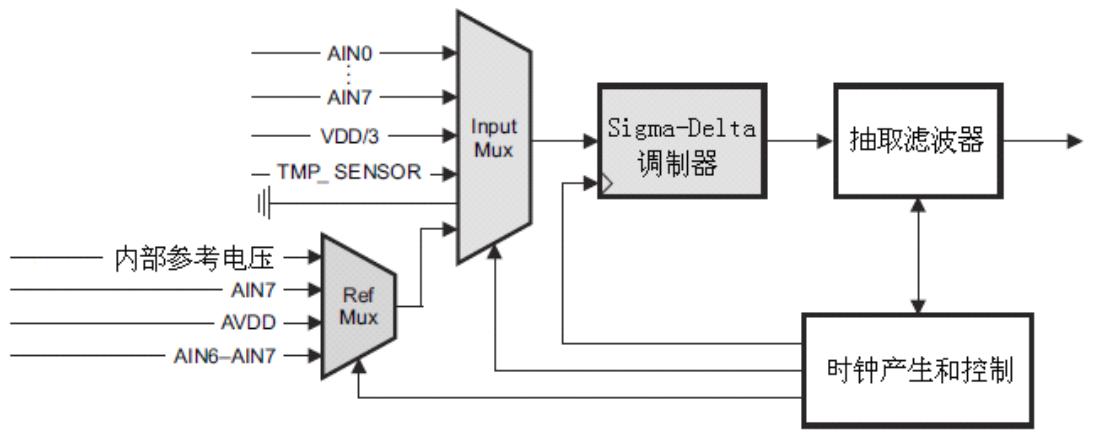


图 12-1ADC 框图

B0304-01

12.2 ADC 运行

本节描述 ADC 的一般设置和运行，并描述了由 CPU 存取的 ADC 控制和状态寄存器的使用。

12.2.1 ADC 输入

P0 端口引脚上的信号可以用作 ADC 输入。在后面的描述中这些端口引脚将被称为 AIN0

—AIN7 引脚。输入引脚 AIN0—AIN7 连接到 ADC。

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。请注意，这些引脚不能使用负电源，或者大于 VDD（未校准电源）的电源。它与在差分模式下的被转换输入对不同。

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用于温度测量的 ADC 输入。为了实现作为温度测量的 ADC 输入，寄存器 TR0.ADCTM 和 ATTEST.ATESTCTRL 必须分别按照 12.2.10 节和 19.15.3 节所描述的进行设置。

还可以选择一个对应 AVDD5/3 的电压作为 ADC 输入。这个输入允许实现例如要求电池监测功能的应用。注意，这种情况下的参考电压不能由电池电压决定，例如，AVDD5 电压不能作为参考电压。

单端输入 AIN0 到 AIN7 以通道号码 0 到 7 表示。通道号码 8-11 表示由 AIN0—AIN1, AIN2—AIN3, AIN4—AIN5 和 AIN6—AIN7 组成的差分输入。通道号码 12-15 分别表示 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 可以执行序列转换，并且将结果移动到存储器（通过 DMA），而不需要任何 CPU 干预。

APCFG 寄存器可以影响转换序列（请见 7.6.6 小节）。ADC 的 8 个模拟输入来自 I/O 引脚，不需要经过编程转变为模拟输入。虽然一个通道通常为一个序列的一部分，但是在 APCFG 里禁止了相应的模拟输入，那么该通道将被忽略。当使用差分输入时，差分输入对的 2 个输入引脚都必须在 APCFG 寄存器里设置为模拟输入引脚。

ADCCON2.SCH 寄存器位用于定义来自 ADC 输入的 ADC 转换序列。当 ADCCON2.SCH 的值设置为小于 8 时，转换序列将包含一个来自每个从 0 开始递增的通道的转换，还包含在 ADCCON2.SCH 编程的通道号码。当 ADCCON2.SCH 的值设置为 8 到 12 之间时，序列包含差分输入，将从通道 8 开始，结束于已编程的通道。如果 ADCCON2.SCH 大于或等于 12，序列仅包含选择的通道。

12.2.3 单个 ADC 转换

除了这种转换序列，ADC 可以通过编程从任何通道执行单个转换。通过写寄存器 ADCCON3 来触发一个单个转换。除非一个转换序列正在进行中，否则立即开始转换，在这种情况下，正在进行的序列转换一完成就开始执行单个转换。

12.2.4 ADC 运行模式

本节描述运行模式和转换初始化。

ADC 具有三个控制寄存器：ADCCON1、ADCCON2 和 ADCCON3。这些寄存器用于配置 ADC 和报告状态。

ADCCON1.EOC 位是一个状态位，当一个转换结束时该位置 1，当读取 ADCH 时，清

除该位。

ADCCON1.ST 位用于启动一个转换序列。当该位置 1，ADCCON1.STSEL 位为 11，且当前没有正在进行的转换时，将启动一个序列。当这个序列转换完成，该位就自动清除。

ADCCON1.STSEL 位选择哪个事件将启动一个新的转换序列。可以被选择的事件选项有：外部引脚 P2.0 上的上升沿，前一个序列的结束，定时器 1 通道 0 比较事件或 ADCCON1.ST 置 1。

ADCCON2 寄存器控制如何执行转换序列。

ADCCON2.SREF 用于选择基准电压。只有在没有转换进行的时候才能改变基准电压。

ADCCON2.SDIV 位选择抽取率，因此也设置了分辨率、完成一个转换所需的时间和采样率。只有在没有转换进行的时候才能改变抽取率。

一个序列的最后一个通道由 ADCCON2.SCH 位选择。

ADCCON3 寄存器控制单个转换的通道号码、基准电压和抽取率。在 ADCCON3 寄存器更新后，立即进行单个转换；或者如果有一个转换序列正在进行，那么在这个转换序列完成后立即进行单个转换。该寄存器位的编码与 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。这是因为这个结果是 GND 和输入信号的差值，这个输入信号总是为有符号的正 ($V_{conv}=V_{inp}-V_{inn}$ ，其中 $V_{inn}=0V$)。当输入信号等于选择的电压基准 VREF 时，达到最大值。对于差分配置，2 个引脚对之间的差值被转换，并且这个差值可以为有符号的负。对于抽取率是 512，分辨率为 12 位，当模拟输入 V_{conv} 等于 VREF 时，数字转换结果是 2047；当模拟输入等于 $-VREF$ 时，转换结果是 -2048。

当 ADCCON1.EOC 置 1 时，ADCH 和 ADCL 里的数字转换结果可用。注意，转换结果总是驻留在 ADCH 和 ADCL 寄存器结合的最高有效位部分。

当读取 ADCCON2.SCH 位时，它们将指示正在进行的转换是在哪个通道上进行的。ADCL 和 ADCH 里的转换结果通常适用于先前的转换。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 的通道号码为 12 或更大，读回值和写入值相同。

12.2.6 ADC 基准电压

模数转换的正基准电压是可选的，可以是一个内部产生的电压、AVDD5 引脚上的电压、应用在 AIN7 输入引脚的外部电压，或应用在 AIN6—AIN7 输入上的差分电压。

转换结果的准确性取决于基准电压的稳定性和噪声特性。期望电压的偏差会导致 ADC 增益误差，这与期望电压和实际电压的比例成正比。基准电压的噪声必须低于 ADC 的量化噪声，以保证达到规定的信噪比。

12.2.7 ADC 转换时间

ADC 只能运行在 32MHz 晶体振荡器，用户不能使用划分的系统时钟。4MHz 的实际 ADC

采样频率是通过固定的内部划分器产生的。执行一个转换所需的时间取决于选择的抽取率。在一般情况下，转换时间由下式给定：

$$T_{conv} = (\text{抽取率} + 16) \times 0.25\mu s$$

12.2.8 ADC 中断

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

12.2.9 ADC DMA 触发

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

对于 ADCCON2.SCH 中头 8 位可能的设置所定义的 8 个通道 AIN0—AIN7，每一个通道都有一个 DMA 触发。当通道转换里一个新的采样准备好时，DMA 触发有效。DMA 触发命名为表 8-1 中 ADC_CHsd，这里 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 转换序列的任何一个通道的新数据准备好时，ADC_CHALL 有效。

12.2.10 ADC 寄存器

本节描述 ADC 寄存器。

ADCL (0xBA) —ADC 数据低位

位	名称	复位	读/写	描述
7: 2	ADC[5: 0]	0000 00	R	ADC 转换结果的最低有效部分。
1: 0	—	00	R0	未使用。总是读为 0。

ADCH (0xBB) —ADC 数据高位

位	名称	复位	读/写	描述
7: 0	ADC[13: 6]	0x00	R	ADC 转换结果的最高有效部分。

ADCCON1 (0xB4) —ADC 控制 1

位	名称	复位	读/写	描述
7	EOC	0	R/H0	转换结束。当 ADCH 被读取时清除。如果在前一个数据被读取之前，已经完成了一个新的转换，该位保持为高。 0: 转换未完成 1: 转换完成
6	ST	0		开始转换。在转换完成之前都读为 1。 0: 没有进行中的转换 1: 如果 ADCCON1.STSEL=11 且没有序列正在进行转换

				, 就启动一个转换序列
5: 4	STSEL[1: 0]	11	R/W1	启动选择。选择哪个事件将启动一个新的转换序列。 00: P2.0 引脚上的外部触发 01: 全速。不等待触发 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 (13 节)。如果写为 01, 当操作完成后改设置将自动返回到 0x00。 00: 正常运行。(13x 展开) 01: 同步 LFSR 一次 (13x 展开) 10: 保留 11: 停止。随机数发生器关闭。
1: 0	—	11	R/W	未使用。总是置为 11。

ADCCON2 (0xB5) —ADC 控制 2

位	名称	复位	读/写	描述
7: 6	SREF[1: 0]	00	R/W	选择用于转换序列的基准电压 00: 内部基准 01: AIN7 引脚上的外部基准 10: AVDD5 引脚 11: AIN6—AIN7 差分输入上的外部基准
5: 4	SDIV[1: 0]	01	R/W	为包含在转换序列里的通道选择抽取率。抽取率也决定了分辨率和完成一个转换所需的时间。 00: 64 抽取率 (7 位分辨率) 01: 128 抽取率 (9 位分辨率) 10: 256 抽取率 (10 位分辨率) 11: 512 抽取率 (12 位分辨率)
3: 0	SCH[3: 0]	0000	R/W	序列通道选择。选择结束序列。一个序列可以从 AIN0 到 AIN7 (SCH<=7), 或从差分输入 AIN0—AIN1 到 AIN6—AIN7 (8<=SCH<=11)。对于其它设置, 只进行单一转换。读取时, 这些位就指示正在进行的转换是在哪个通道上进行的。 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: 保留 1110: 温度传感器 1111: VDD/3
--	--	--	--	--

ADCCON3 (0xB6) —ADC 控制 3

位	名称	复位	读/写	描述
7: 6	EREF[1: 0]	00	R/W	选择用于单个转换的基准电压 00: 内部基准 01: AIN7 引脚上的外部基准 10: AVDD5 引脚 11: AIN6—AIN7 差分输入上的外部基准
5: 4	EDIV[1: 0]	00	R/W	为单个转换选择抽取率。抽取率也决定了分辨率和完成一个转换所需的时间。 00: 64 抽取率 (7 位分辨率) 01: 128 抽取率 (9 位分辨率) 10: 256 抽取率 (10 位分辨率) 11: 512 抽取率 (12 位分辨率)
3: 0	ECH[3: 0]	0000	R/W	单个通道选择。选择通过写 ADCCON3 而触发的单个转换的通道号码。当单个转换完成时这些位会自动清除。 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: 保留 1110: 温度传感器 1111: VDD/3

TR0 (0x624B) —测试寄存器 0

位	名称	复位	读/写	描述
7: 1	—	0000 000	R0	保留。写为 0。
0	ACTM	0	R/W	设置为 1, 连接温度传感器和 SOC_ADC。见 19.15.3 小节所描述的 ATESST 寄存器来使能温度传感器。

13 随机数发生器

本章详细描述随机数发生器及其用途。

13.1 简介

随机数发生器特征如下：

- 产生伪随机字节，可以被 CPU 读取，或者直接被命令选通处理器使用（请见 19.14 节）。
- 计算写入到 RNDH 的字节的 CRC16。
- 由写入到 RNDL 的值产生。

随机数发生器是一个 16 位线性反馈移位寄存器 (LFSR)，带有生成多项式 $X^{16} + X^{15} + X^2 + 1$ (即 CRC16)。它根据执行的操作来使用不同等级的展开。基本形式 (不展开) 如图 13-1 所示。

当 ADCCON1.RCTRL=11 时，随机数发生器关闭。

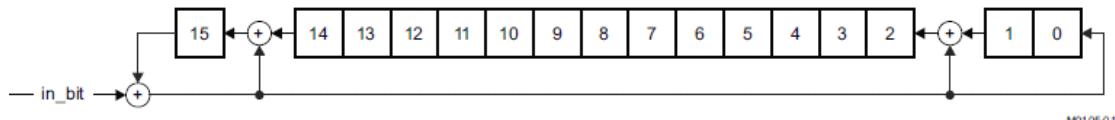


图 13-1 随机数发生器的基本结构

13.2 随机数发生器运行

随机数发生器的运行是由 ADCCON1.RCTRL 位控制的 (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 半随机序列生成

默认操作 (ADCCON1.RCTRL=00) 是在每次命令选通处理器读取随机值，就同步 LFSR 一次 (13x 展开，同步 13x 展开意味着执行 13 次反馈移位的等效操作) (见 19.14 小节)。这保证一个新的来自 LFSR 末尾的 LSB 的伪随机字节的有效性。

另一种更新 LFSR 的方法是设置 ADCCON1.RCTRL 为 01。这将同步 LFSR 一次 (13x 展开)，并且当操作完成时，ADCCON1.RCTRL 位将自动清除。

13.2.2 种子数的产生

可通过对 RNDL 寄存器进行 2 次写入来产生 LFSR。每次写入 RNDL 寄存器，LFSR 的 8 个 LSB 被复制到 8 个 MSB，8 个 LSB 被替换为写入 RNDL 的新的数据字节。

如果需要一个真正的随机值, LFSR 可以通过在 RF 接收路径给 RNDL 写入来自 IF_ADC 的随机值来产生。要使用这种产生方法, 射频必须首先上电。射频应当处于无限 RX 状态, 以避免在 RX 状态下可能的同步检测。从 IF_ADC 得到的随机值可以从 RF 寄存器 RFRND 中读取。如上所述, 读取的值作为种子值被写入 RNDL 寄存器。注意, 当射频用于正常任务时不能使用这种方法。

请注意, 种子值 0x0000 或 0x8003 必然导致同步后 LFSR 里的值不变, 由于没有值通过 in_bit (见图 13-1) 推入, 因此, 它不能用于随机数的产生。

13.2.3 CRC16

LFSR 也可以用于计算一系列字节的 CRC 值。写 RNDH 寄存器将触发一次 CRC 计算。新的字节从 MSB 末端处理, 使用一个 8x 展开, 以便一个新的字节可以在每个时钟周期被写入到 RNDH。

注意, 在开始 CRC 计算前, LFSR 必须通过写入 RNDL 来被正确产生。通常用于 CRC 计算的种子值应当为 0x0000 或 0xFFFF。

13.3 随机数发生器寄存器

本节描述随机数发生器寄存器。

RNDL (0xBC) —随机数发生器数据低位字节

位	名称	复位	读/写	描述
7: 0	RNDL[7: 0]	0xFF	R/W	随机值/种子或 CRC 结果, 低位字节 当用于随机数产生, 对该寄存器进行 2 次写入将产生随机数发生器。写该寄存器复制 LFSR 的 8 个 LSB 到 8 个 MSB, 8 个 LSB 被替换为写入的数据值。 当读取该寄存器时返回值为 LSFR 的 8 个 LSB。 当用于随机数产生, 读该寄存器返回随机数的 8 个 LSB。 当用于 CRC 计算, 读该寄存器返回 CRC 结果的 8 个 LSB。

RNDH (0xBD) —随机数发生器数据高位字节

位	名称	复位	读/写	描述
7: 0	RNDH[7: 0]	0xFF	R/W	随机值或 CRC 结果/输入数据, 高位字节 写入时, 将触发一次 CRC16 计算, 写入的数据值从 MSB 位开始处理。 当读取该寄存器时返回值为 LSFR 的 8 个 MSB。 当用于随机数产生, 读该寄存器返回随机数的 8 个 MSB。 当用于 CRC 计算, 读该寄存器返回 CRC 结果的 8 个 MSB。

14 AES 协处理器

CC253x 数据加密是由支持高级加密标准的专用协处理器 AES 完成的。正是由于有了 AES 协处理器的加密/解密操作，极大地减轻了 CC253x 内置 CPU 的负担。

AES 协处理器具有下列特性：

- 支持 IEEE 802.15.4 的全部安全机制
- ECB（电子编码加密）、CBC（密码防护链）、CFB（密码反馈）、OFB（输出反馈加密）、CTR（计数模式加密）和 CBC-MAC（密码防护链消息验证代码）模式
- 硬件支持 CCM（CTR+CBC-MAC）模式
- 128 位密钥和初始化向量（IV）/当前时间（Nonce）
- DMA 传送触发能力

14.1 AES 操作

加密一条消息必须遵循下面的步骤（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 协处理器运行 128 位的数据块。数据块一旦装入 AES 协处理器就可以加密，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 密钥和初始化向量

密钥或初始化向量（IV）/当前时间装入之前，应当发送装入密钥或 IV/当前时间的命令给协处理器。装入 IV 时，设置正确的模式是非常重要的。

装入密钥或初始化向量，将取消任何协处理器正在运行的程序。密钥一旦装入，除非重新装入，否则一直有效。

在每条消息（而不是消息块）开始之前，必须下载初始化向量。

通过 CC253x 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 CPU 接口

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直接存储器存取 (DMA) 进行存取。

当使用 DMA 存取 AED 协处理器时, 必须使用两个 DMA 通道, 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 操作模式

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 DMA 是首选。

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。下面的小节简要说明了要完成的必要步骤。

14.6 CBC—MAC

当运行 CBC-MAC 加密时, 除了最后一个数据块改为运行于 CBC 模式之外, 其余都是由协处理器按照 CBC-MAC 模式, 每次下载一个数据块。当最后一个数据块下载完毕后, 上传的数据块就是 MAC 消息 (message) 了。

CBC-MAC 解密与加密类似。上传的 MAC 消息必须通过与 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 (密码已经装入):

消息验证阶段

这个阶段发生在下面所示的步骤 1—6 之间。

1. 软件将 0 装入初始化向量 (IV)。
2. 软件创建数据块 B0。数据块 B0 的结构如图 14-1 所示。

	名称 B0				指定 在 CCM 模式中第一个验证的数据块											
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 消息验证阶段块 0

NONCE（当前时间）值没有限制。L_M是以字节为单位的消息长度。

对于 802.15.4，NONCE 有 13 个字节，而 L_M 有 2 个字节。

验证标志字节的内容如图 14-2 所示。

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		指定 CCM 模式的验证标志域					
位	7	6	5	4	3	2	1	0
名称	保留	A_Data	(M_2) / 2			L - 1		
值	0	×	×	×	×	1	0	1

图 14-2 验证标志字节

3. 如果需要某些附加验证数据(下面用 a 表示)(即 A_Data=1),则软件就会创建 A_Data 的长度域, 称为 L(a):

- (a)如果 l(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。
- (3b)如果 $0 < l(a) < 2^{16}-2^8$, 则 L(a)是 2 个 l(a)编码的 8 位字节。

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

AUTH-DATA=L(a)+ 验证数据 + (0 填充)

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

5. 软件将 B0 数据块、附加验证数据块(如果有)和消息连接起来。

输入消息=B0 + AUTH-DATA + 消息 + (消息的 0 填充)

6. 一旦由 CBC-MAC 输入消息验证结束, 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。

消息加密

7. 软件创建密钥流块 A0。注意, 在当前有 CTR 产生的例子中, L=6。内容如图 14-3 所示。

注意, 当在 OFB 模式加密验证数据 T 到产生 U 时, CTR 值必须为 0。当使用 CTR 模式加密消息块时, CTR 值必须为一个非 0 值。

加密标志字节的内容如图 14-4 所示。

	名称 A0		指定 CCM 模式的第一个 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 消息加密阶段块

	名称 FLAG/A0		指定 CCM 模式的加密标志域					
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 Load IV/Nonce 命令装载 A0。只有在选择 Load 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 值为一个非零值时必须重装入 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

消息解密

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} 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）。

消息分解阶段

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 位组，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. 软件创建密钥流块 A0。所用的方法和 CCM 加密一样。

5. 软件通过选择 Load 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 加密已经加密的验证数据 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。

基准验证标签生成

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。

消息验证校核阶段

该阶段中，利用软件来比较 T 和 MACTag。

14.8 在各个通信层次之间共享 AES 协处理器

AES 协处理器是各个层次共享的通用源。AES 协处理器每次只能用来处理一个实例。因此需要执行某些软件信号来分配和释放资源。

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 通道从 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

本节讲述 AES 协处理器寄存器。

ENCCS (0xB3) — 加密控制和状态

位	名称	复位	读/写	描述
7	—	0	R0	未使用，总是读为 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: 未使用 111: 未使用
3	RDY	1	R	加密/解密就绪状态 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 1 写入 ST 位，命令即将执行 00: 加密数据块 01: 解密数据块 10: 装入密钥 11: 装入 IV/Nonce
0	ST	0	R/W1 H0	开始执行由 CMD 设置的命令。对每个命令或 128 位数据块必须分别下达。由硬件清除该位

ENCDI (0xB1) — 加密输入数据

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

15 看门狗定时器

在 CPU 可能受到软件颠覆的情况下，看门狗定时器（WDT）可被用来作为一种恢复的手段。当软件在指定的时间间隔里不能清除 WDT 时，WDT 就会复位系统。看门狗可以用于那些会受到电气噪声、电源故障、静电放电影响的应用，或者需要高可靠性的应用。如果一个应用不需要看门狗功能，可以将看门狗定时器配置为一个间隔定时器，用于在指定的时间间隔产生中断。

看门狗定时器的主要特征如下：

- 4 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配置为通用定时器。WDCTL 寄存器控制 WDT 模块的操作。看门狗定时器包含一个由 32kHz 时钟源同步的 15 位计数器。请注意，用户并不能获得 15 位计数器的内容。15 位计数器的内容在所有功耗模式下都能保持，而当再次进入主动模式时，看门狗定时器继续计数。

15.1 看门狗模式

系统复位后看门狗定时器被禁用。要在看门狗模式下启动 WDT，WDCTL.MODE[1: 0] 位必须置为 10。看门狗定时器计数器从 0 开始递增。在看门狗模式下，如果已经使能了定时器，就不能再禁止定时器。因此，当 WDT 已经运行于看门狗模式时，往 WDCTL.MODE[1: 0] 位写 00 或 01 是不起作用的。

WDT 运行在 32.768kHz 的看门狗定时器时钟频率上（使用 32kHz 晶体振荡器）。当计数值设置为 64, 512, 8192 和 32768，时钟频率对应的超时时间为 1.9ms, 15.625ms, 0.25s 和 1s。

如果计数器达到了选定的定时器间隔值，看门狗定时器就产生一个复位信号给系统。如果在计数器达到选定的定时器间隔值之前，执行了一个看门狗清除序列，计数器就复位为 0 并继续递增。看门狗清除序列包括在一个看门狗时钟周期内，写 0xA 到 WDCTL.CLR[3: 0]，接着写 0x5 到同一个寄存器位。如果在看门狗周期结束之前，这个序列没有被完全执行，看门狗定时器就产生一个复位信号给系统。

在看门狗模式下，如果 WDT 已经使能，就不能通过写 WDCTL.MODE[1: 0] 位来改变这个模式，定时器间隔值也不能改变。

在看门狗模式，WDT 不会产生中断请求。

15.2 定时器模式

要在正常定时器模式下启动 WDT，WDCTL.MODE[1: 0] 位必须设置为 11。定时器开始工作，计数器从 0 开始递增。当计数器达到了选定的间隔值，定时器将产生一个中断请求 (IRCON2.WDTIF/IEN2.WDTIE)。

在定时器模式，可以通过写 1 到 WDCTL.CLR[0] 来清除定时器内容。当定时器被清除，

计数器内容就被置为 0。写 00 或 01 到 WDCTL.MODE[1: 0]将停止定时器并清除为 0。

通过 WDCTL.INT[1: 0]位来设置定时器间隔。在定时器运行期间，不能改变定时器间隔，当定时器启动时设置定时器间隔。在定时器模式，到达定时器间隔不会产生复位。

注意，如果选择了看门狗模式，在芯片复位前不能选择定时器模式。

15.3 看门狗定时器寄存器

本节描述看门狗定时器寄存器 WDCTL。

WDCTL (0xC0) —看门狗定时器控制

位	名称	复位	读/写	描述
7: 4	CLR[3: 0]	0000	R/W	清除定时器。在看门狗模式，当 0xA 和 0x5 相继被写到这些位，定时器被清除(定时器加载 0)。注意，只有在 0xA 被写入后，0x5 在一个看门狗时钟周期内被写入，定时器才会被清除。当看门狗定时器 IDLE 时，写这些位无效。在定时器模式，写 1 到 CLR[0]（其它 3 位不重要）将清除定时器到 0x0000。
3: 2	MODE[1: 0]	00	R/W	模式选择。该位用于选择 WDT 为看门狗模式或定时器模式。在定时器模式，设置这些位为 IDLE 将停止定时器。注意：运行于定时器模式时，要切换到看门狗模式，首先要停止 WDT，然后启动 WDT 于看门狗模式。在看门狗模式，写这些位无效。 00: IDLE 01: IDLE (位使用，相当于设置为 00) 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，它定义为 32kHz 振荡器周期的一个给定数。注意：只有在 WDT 处于 IDLE 时才能改变定时器间隔，因此定时器间隔必须在定时器启动的同时设置。 00: 运行 32kHz 晶体振荡器时，时钟周期×32,768 (~1s) 01: 时钟周期×8192 (~0.25s) 10: 时钟周期×512 (~15.625ms) 11: 时钟周期×64 (~1.9ms)

16 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。请见 12.1 小节 I/O 配置。

16.1 USART 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 RXD、TXD、可选的 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 模式提供全双工异步传送，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 USART 发送

当 USART 收/发数据缓冲器 UxDBUF 写入数据时，UART 发送启动。该字节发送到输出引脚 TXD_x。寄存器 UxDBUF 是双缓冲器。

当字节传送开始时，UxCSR.ACTIVE 位设置为 1，而当字节传送结束时，UxCSR.ACTIVE 位清 0。当传送结束时，UxCSR.TX_BYTE 位设置为 1。当 UxDBUF 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，新的数据字节能够装入数据缓冲器。

16.1.2 USART 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 RXD_x 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 位为 0。

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

注意：很重要的一点是，当应用程序已经读取 UxDBUF，不会清除 UxCSR.RX_BYT。清除了 UxCSR.RX_BYT 也就暗示 UART 确认 UART RX 移位寄存器为空，即使它可能保存有未决数据（通常是由于端到端传输引起的）。所以 UART 启动 RT/RTS 线（TTL 为低电平），它允许数据流进入 UART，导致潜在的溢出。因此 UxCSR.TX_BYT 标志紧密结合了 RT/RTS 功能，因此只能被片上系统 UART 自己控制。否则应用程序可能通常会经历这样一个事件：即使一个端到端传输清楚地表明了它应当间歇性地停止数据流，但是 RT/RTS 线仍然保持启动（TTL 为低电平）。

16.1.3 UART 硬件流控制

当 UxUCR.FLOW 设置为 1，硬件流控制使能。然后，当接收寄存器空而且接收使能时，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 PARITY 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 1。当 UxCSR 读取时，UxCSR.ERR 位清 0。

寄存器位 UxUCR.SPB 决定要传送的停止位为 1 位或 2 位。接收器总是要核对一个停止位。如果在接收期间接收到的第一个停止位不是期望的停止位电平，设置寄存器位 UxCSR.FE 为 1，发出帧出错信号。当读取 UxCSR 时，UxCSR.FE 清 0。当 UxUCR.SPB 设置为 1 时，接收器将核对两个停止位。注意当第一个停止位核对通过时，RX 中断将被置位。如果第二个停止位核对未通过，当帧错误位 UxCSR.FE 置为 1 时，将会有个延迟。这种延迟是可靠的波特率（位持续时间）。

16.2 SPI 模式

本节描述同步通信的 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。请参见 12.1 小节中有关如何将 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）和 4 线 SPI 接口
- 主/从模式
- 可配置的 SCK 极性和相位
- 可配置的 LSB 或 MSB 首先传送

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 UxDBUF 写入字节后，SPI 主模式字节传送就开始了。USART 使用波特率发生器生成 SCK 串行时钟（请见 16.4 小节），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.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 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 UxDBUF 提供，供读取。当收到的新数据在 USART 收/发数据寄存器 UxDBUF 中就绪时，接收中断产生。

当 SPI 就绪接收另一个字节用来发送时，发送中断产生。由于 UxDBUF 是双缓冲，这个操作刚好在发送开始时就发生了。请注意，直到 UxCSR.TX_BYTE 设置为 1 才能将数据写入 UxDBUF。对于 DMA 传送，这是自动处理的。对于端到端传送，要使用 DMA 传送，UxGDR.CPHA 位必须设置为 0，如果不设置为 0，传输字节会被损坏。由于系统要求，有必要进行 UxGDR.CPHA 设置和 UxCSR.TX_BYTE 轮询。

此外请注意发送中断和接收中断的区别，因为发送中断大约先于接收中断 8 位周期到达。

如上所述，SPI 主模式操作是一个 3 线接口。不需要选择输入来使能主。如果外部从要求一个从选择信号，可以通过软件使用一个通用 I/O 引脚来实现。

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 MOSI 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 MISO。

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 UxGCR.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 UxDBUF 提供，供读取。

当 SPI 从模式操作开始时，发送中断产生。

16.3 SSN 从选择引脚

在 SPI 操作模式，USART 配置为 SPI 从，使用 4 线接口，含有作为对 SPI 的输入的从选择（SSN）引脚。在 SSN 的下降沿，SPI 从有效，输入引脚 MOSI 接收数据，输出引脚 MISO 输出数据。在 SSN 的上升沿，SPI 从无效且不能接收数据。注意在 SSN 的上升沿之后，MISO 输出为三态。还应注意，SSN（上升沿）的释放必须被对齐到接收或发送字节的末尾。如果在一个字节里释放了，下一个接收到的字节将不能被正常接收，因为前一个字节的信息存在于 SPI 系统中。可以使用 USART 清除来删除该信息。

在 SPI 主模式，不使用 SSN 引脚。当 USART 作为 SPI 主操作，外部 SPI 从设备需要一个从选择信号，然后在软件中需要使用通用 I/O 引脚来执行从选择信号功能。

16.4 波特率发生器

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：

$$\text{波特率} = \frac{(256 + \text{BAUD_M}) \times 2^{\text{BAUD_E}}}{2^{28}} \times f \quad (16-1)$$

式中：f 是系统时钟频率，等于 16MHz 校准的 RC 振荡器或者 32MHz 晶体振荡器。

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 32MHz 系统时钟。该表也给出了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 (f 是系统时钟频率)。

SPI 模式下的最大波特率请见设备数据手册。

注意，在发生任何其它的 UART 或 SPI 操作之前，必须通过 UxBAUD 和 UxGCR 寄存器来设置波特率。这意味着在它完成它的启动条件之前，定时器使用的信息都是未更新的，因此需要时间来改变波特率。

表 16-1 32MHz 系统时钟的常用波特率设置

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。要注意，在一个 TX/RX 位的中间设置清除位，在该位结束之前都不会发生清除（缓冲区将立即被清除，但保持位持续时间的消息的定时器不会被清除）。因此，使用清除位应当与 USART 中断对齐，或者使用一个等待时间，这个等待时间是在当前波特率下 USART 接收到更新的数据或配置之前的 1 位持续时间。

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）。传输开始时触发 TX 中断，并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。2.5 小节中有关于这些寄存器的详细描述。中断使能和标志概括如下：

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 RX 或者 TX 完成事件激活，也就是说，该事件作为 USART 中断请求。可以配置 DMA 通道使用 USART 收/发缓冲器（即 UxDBUF）作为它的源地址或者目标地址。

DMA 触发的概括参见表 8-1。

16.8 USART 寄存器

本节描述 USART 寄存器。对于每个 USART，有 5 个寄存器（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
- UxDBUF: USARTx 收/发数据缓冲器
- UxBAUD: USARTx 波特率控制

U0CSR (0x86) —USART0 控制和状态

位	名称	复位	读/写	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 完全配置好之前不使能接收。 0: 接收器禁止 1: 接收器使能

5	SLAVE	0	R/W	SPI 主模式或从模式选择 0: SPI 主模式 1: SPI 从模式
4	FE	0	R/W0	UART 帧错误状态 0: 没有检测出帧错误 1: 收到的字节停止位电平出错
3	ERR	0	R/W0	UART 奇偶校验错误状态 0: 没有检测出奇偶校验错误 1: 收到的字节奇偶校验出错
2	RX_BYTE	0	R/W0	接收字节状态。UART 模式和 SPI 从模式。读取 U0DBUF 时自动清除该位，通过写 0 清除，有效地丢弃 U0DBUF 中的数据。 0: 没有收到字节 1: 收到的字节就绪
1	TX_BYTE	0	R/W0	发送字节状态。UART 模式和 SPI 主模式 0: 没有发送字节 1: 写到数据缓冲器寄存器的最后字节已发送
0	ACTIVE	0	R	USART 收/发激活状态。在 SPI 从模式，该位等于从选择。 0: USART 空闲 1: 在发送或者接收模式中，USART 忙

U0UCR (0xC4) —USART0 UART 控制

位	名称	复位	读/写	描述
7	FLUSH	0	R0/W1	清除单元。当设置为 1 时，该事件立即停止当前操作，返回空闲状态单元。
6	FLOW	0	R/W	UART 硬件流使能。选择使用硬件流来控制引脚 RTS 和 CTS 0: 流控制禁止 1: 流控制使能
5	D9	0	R/W	UART 奇偶校验位。如果使能了奇偶校验，写入 D9 的值决定发送的第 9 位的值，如果接收到的第 9 位于接收字节的奇偶校验不匹配，接收时报告 ERR。 如果奇偶校验使能，那就用该位设置奇偶校验电平： 0: 奇校验 1: 偶校验
4	BIT9	0	R/W	UART 9 位数据使能。当 BIT9 为 1 时，使能奇偶校验位传送（即第 9 位）。如果 PARITY 位使能了奇偶校验，第 9 位的内容由 D9 给出。 0: 8 位传送 1: 9 位传送
3	PARITY	0	R/W	UART 奇偶校验使能。除了计算奇偶校验要设置该位，还必须使能 9 位模式。 0: 奇偶校验禁止 1: 奇偶校验使能
2	SPB	0	R/W	UART 停止位数量。选择要传送的停止位的数量 0: 1 个停止位

				1: 2 个停止位
1	STOP	1	R/W	UART 停止位电平必须与起始位电平不同 0: 停止位电平低 1: 停止位电平高
0	START	0	R/W	UART 起始位电平。空闲线的极性假定与选择的起始位电平相反 0: 起始位电平低 1: 起始位电平高

U0GCR (0xC5) —USART0 通用控制

位	名称	复位	读/写	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: 当来自 CPOL 的 SCK 反相之后又返回 CPOL 时, 数据输出到 MOSI; 当来自 CPOL 的 SCK 返回 CPOL 反相时, 输入数据采样到 MISO 1: 当来自 CPOL 的 SCK 返回 CPOL 反相时, 数据输出到 MOSI; 当来自 CPOL 的 SCK 反相之后又返回 CPOL 时, 输入数据采样到 MISO
5	ORDER	0	R/W	用于传送的位顺序 0: LSB 先传送 1: MSB 先传送
4: 0	BAUD_E[4: 0]	00000	R/W	波特率指数值。BAUD_E 连同 BAUD_M 一起决定了 UART 波特率和 SPI 主 SCK 时钟频率。

U0DBUF (0xC1) —USART0 收/发数据缓冲器

位	名称	复位	读/写	描述
7: 0	DATA[7: 0]	0x00	R/W	USART 接收和发送数据。数据写入该寄存器就是将数据写入内部数据传送寄存器; 读取该寄存器, 就是将来自内部数据读取寄存器中的数据读出。

U0BAUD (0xC2) —USART0 波特率控制

位	名称	复位	读/写	描述
7: 0	BAUD_M[7: 0]	0x00	R/W	波特率尾数值。BAUD_E 连同 BAUD_M 一起决定了 UART 波特率和 SPI 主 SCK 时钟频率

U1CSR (0xF8) —USART1 控制和状态

位	名称	复位	读/写	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 完全配置好之前不使能接收。

				0: 接收器禁止 1: 接收器使能
5	SLAVE	0	R/W	SPI 主模式或从模式选择 0: SPI 主模式 1: SPI 从模式
4	FE	0	R/W0	UART 帧错误状态 0: 没有检测出帧错误 1: 收到的字节停止位电平出错
3	ERR	0	R/W0	UART 奇偶校验错误状态 0: 没有检测出奇偶校验错误 1: 收到的字节奇偶校验出错
2	RX_BYTE	0	R/W0	接收字节状态。UART 模式和 SPI 从模式。读取 U1DBUF 时自动清除该位，通过写 0 清除，有效地丢弃 U1DBUF 中的数据。 0: 没有收到字节 1: 收到的字节就绪
1	TX_BYTE	0	R/W0	发送字节状态。UART 模式和 SPI 主模式 0: 没有发送字节 1: 写到数据缓冲器寄存器的最后字节已发送
0	ACTIVE	0	R	USART 收/发激活状态。在 SPI 从模式，该位等于从选择。 0: USART 空闲 1: 在发送或者接收模式中，USART 忙

U1UCR (0xCB) —USART1 UART 控制

位	名称	复位	读/写	描述
7	FLUSH	0	R0/W1	清除单元。当设置为 1 时，该事件立即停止当前操作，返回空闲状态单元。
6	FLOW	0	R/W	UART 硬件流使能。选择使用硬件流来控制引脚 RTS 和 CTS 0: 流控制禁止 1: 流控制使能
5	D9	0	R/W	UART 数据位 9 的内容。如果使能了奇偶校验，写入 D9 的值决定发送的第 9 位的值，如果接收到的第 9 位与接收字节的奇偶校验不匹配，接收时报告 ERR。 如果奇偶校验使能，那就用该位设置奇偶校验电平： 0: 奇校验 1: 偶校验
4	BIT9	0	R/W	UART 9 位数据使能。当 BIT9 为 1 时，使能奇偶校验位传送（即第 9 位）。如果 PARITY 位使能了奇偶校验，第 9 位的内容由 D9 给出。 0: 8 位传送 1: 9 位传送
3	PARITY	0	R/W	UART 奇偶校验使能。除了计算奇偶校验要设置该位，还必须使能 9 位模式。 0: 奇偶校验禁止 1: 奇偶校验使能

2	SPB	0	R/W	UART 停止位数量。选择要传送的停止位的数量 0: 1 个停止位 1: 2 个停止位
1	STOP	1	R/W	UART 停止位电平必须与起始位电平不同 0: 停止位电平低 1: 停止位电平高
0	START	0	R/W	UART 起始位电平。空闲线的极性假定与选择的起始位电平相反 0: 起始位电平低 1: 起始位电平高

U1GCR (0xFC) —USART1 通用控制

位	名称	复位	读/写	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: 当来自 CPOL 的 SCK 反相之后又返回 CPOL 时, 数据输出到 MOSI; 当来自 CPOL 的 SCK 返回 CPOL 反相时, 输入数据采样到 MISO。 1: 当来自 CPOL 的 SCK 返回 CPOL 反相时, 数据输出到 MOSI; 当来自 CPOL 的 SCK 反相之后又返回 CPOL 时, 输入数据采样到 MISO。
5	ORDER	0	R/W	用于传送的位顺序 0: LSB 先传送 1: MSB 先传送
4: 0	BAUD_E[4: 0]	00000	R/W	波特率指数值。BAUD_E 连同 BAUD_M 一起决定了 UART 波特率和 SPI 主 SCK 时钟频率。

U1DBUF (0xF9) —USART1 收/发数据缓冲器

位	名称	复位	读/写	描述
7: 0	DATA[7: 0]	0x00	R/W	USART 接收和发送数据。数据写入该寄存器就是将数据写入内部数据传送寄存器; 读取该寄存器, 就是将来自内部数据读取寄存器中的数据读出。

U1BAUD (0xFA) —USART1 波特率控制

位	名称	复位	读/写	描述
7: 0	BAUD_M[7: 0]	0x00	R/W	波特率尾数值。BAUD_E 连同 BAUD_M 一起决定了 UART 波特率和 SPI 主 SCK 时钟频率。

17 USB 控制器

本节重点介绍了 USB 控制器的功能，并假定用户对 USB 已经很理解，并且对使用的术语和概念也比较熟悉。详见通用串行总线规范[3]。

标准 USB 术语的用途与 IN 和 OUT 有关，即 IN 总是输入主机（PC），OUT 总是输出主机。

17.1 USB 简介

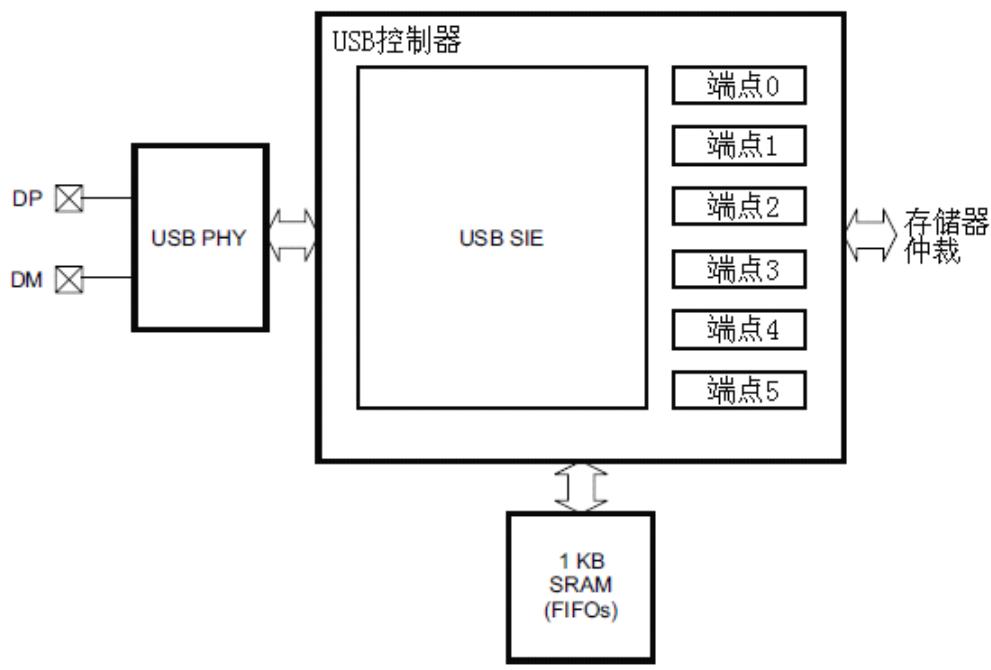
USB 控制器监控 USB 的相关活动，并处理数据包传输。

固件负责适当响应 USB 中断、上传数据包到端点 FIFO、从端点 FIFO 里下载数据。固件必须能够正确回复所有来自 USB 主机的标准请求，根据 PC 上驱动程序里执行的协议工作。

USB 控制器有如下功能：

- 全速运行（最高可达 12Mbps）
- 5 个端点（除了端点 0），可以用来作为 IN、OUT 或者 IN/OUT，可以被配置为批量/中断或同步
- 使用 1KB SRAM FIFO 来存储 USB 数据包
- 端点支持的数据包大小从 8 到 512 字节
- 支持 USB 数据包双缓冲

图 17-1 为 USB 控制器框图。USB PHY 是带有输入和输出驱动的物理接口。USB SIE 是串行接口引擎，控制数据包传输到端点或从端点传输过来。USB 控制器通过存储器仲裁连接到系统的其余部分。



17.2 USB 使能

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

17.3 48MHz USB PLL

为了 USB 控制器的正确运行，48MHz 内部 USB 锁相环必须上电。很重要的一点是，在 USB PLL 使能之前，必须选择晶体振荡器作为振荡源并且稳定运行。通过设置 USBCTRL.PLL_EN 位来使能 USB PLL，并等待 USBCTRL.PLL_LOCKED 状态标志为高。当 PLL 被锁定时，就可以安全使用 USB 控制器了。

注意：PLL 在退出工作模式之前必须被禁止，进入工作模式的时候再重新使能。

17.4 USB 中断

有 3 个中断使能屏蔽寄存器，与之相关的中断标志寄存器有 3 个。

表 17-1 USB 中断标志和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 中断的中断标志	USBCIE
USBIHF	包括端点 0 和所有 IN 端点的中断标志	USBIIE
USBOIF	标扩所有 OUT 端点的中断标志	USBOIE

注意：除了 SOF 和挂起，所有中断复位后开始使能。

USB 控制器使用中断#6 为 USB 中断。该中断号码与端口 2 输入共享；因此，如果它们被使能，中断进程还必须处理端口 2 中断。要产生一个中断请求，IEN2.P2IE 必须设为 1，并且 USBCIE、USBIIE 和 USBOIE 寄存器中期望的中断使能位也必须设为 1。当一个中断请求产生后，如果没有更高优先级的中断未决，CPU 开始执行 ISR。中断进程应该读取所有的中断标志寄存器，并且根据这些标志的状态进行动作。中断标志寄存器被读后会清除，因此各个中断标志的状态应该保存在存储器中（通常在栈的一个局部变量中），以允许对它们进行多次访问。

ISR 执行结束，读取中断标志后，应当清除中断标志，以允许可以检测新的 USB 中断和 P2 中断。清除 IRCON2.P2IF 之前不许先清除 P2IFG 寄存器里的端口 2 中断状态标志。

当从挂起状态（通常在功耗模式 PM1 下）被唤醒时，USB D+中断标志和 P2IFG.DPIF 置 1。D+ 中断标志标明，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 端点 0

端点 0 (EP0) 是一个双向控制端点，在枚举阶段期间，所有的通信都是通过端点 0 执行的。USBADDR 寄存器被设置为一个非 0 值之前，USB 控制器只能通过端点 0 通信。设置 USBADDR 寄存器为一个 1-127 的值，使得 USB 功能离开枚举阶段的默认状态而进入地址状态。所有配置的端点对于应用程序都是可用的。

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

为 32 字节。

设置 USBINDEX 寄存器为 0 选择端点 0，然后通过 USBCS0 寄存器来控制端点 0。USBCNT0 寄存器包含接收到的字节数。

17.6 端点 0 中断

下列事件可以产生一个 EP0 中断请求：

- 接收到一个数据包 (USBCS0.OUTPKT_RDY=1)
- 加载到 EP0 FIFO 的一个数据包已经被发送到 USB 主机。(当准备好传输一个新的数据包时, USBCS0.INPKT_RDY 应当置为 1。当数据包被发送后, 硬件清除该位。)
- 完成一个 IN 转换 (在转换状态阶段期间产生一个中断)。
- 发送了一个 STALL (USBCS0.SENT_STALL=1)
- 一个控制传输提前结束 (USBCS0.SETUP_END=1)

不管 EP0 中断使能位 USBIIE.EP0IE 的状态是使能还是禁止, 以上任何一个事件都会导致声明 USBIIF.EP0IF。如果 EP0 中断使能位设为 1, 那么 CPU 中断标志 IRCON2.P2IF 也会被声明。只有 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

当发生了一个协议错误时, USB 控制器发送一个 STALL 握手。如果端点 0 中断被正确使能了, 会声明 USBCS0.SENT_STALL 位并产生一个中断请求。协议错误可以是以下任何一种情况:

- 在 USBCS0.DATA_END 被设置为完成 OUT 数据阶段后, 接收到一个 OUT 令牌 (主机尝试发送比期望的数据更多的数据)
- 在 USBCS0.DATA_END 被设置为完成 IN 数据阶段后, 接收到一个 IN 令牌 (主机尝试发送比期望的数据更多的数据)
- 在 OUT 数据阶段期间, USB 主机尝试发送一个超过最大数据包大小的数据包
- 在状态阶段期间, 接收的 DATA1 数据包大小不是 0

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

如果 EP0 中断是因为声明了 USBCS0.SENT_STALL 位而产生的, 该位应当被解除声明, 固件认为传输中止 (空闲的存储器缓冲区等)。

如果 EP0 在数据阶段接收到一个意外的令牌, 就声明 USBCS0.SETUP_END 位并产生一个 EP0 中断 (如果正确使能了)。然后 EP0 切换到 IDLE 状态。然后固件应该设置 USBCS0.CLR_SETUP_END 位为 1, 并终止当前的传输。如果声明了 USBCS0.OUTPKT_RDY 位, 表示接收到另一个固件应当处理的配置数据包。

17.6.2 SETUP 传输 (IDLE 状态)

控制传输由两个或三个传输阶段构成 (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(有效载荷)固定是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 控制器拒绝该数据包。此外, USB 控制器检查配置包的内容, 以确定在控制传输中是否有数据阶段。如果有数据阶段, 当 USBCS0.CLR_OUTPKT_RDY=1 时 (USBCS0.DATA_END=0), EP0 从 IDLE 状态切换到 TX 状态 (IN 传输) 或 RX 状态 (OUT 传输)。

当接收到一个包, 声明 USBCS0.OUTPKT_RDY 位, 如果已经使能了中断还会产生一个中断请求 (EP0 中断)。当接收到一个配置包, 固件应当执行以下操作:

1. 从 EP0 FIFO 中卸载配置包
2. 检查配置包的内容并执行适当的操作
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, USBCS0.DATA_END 位必须被置位。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, 并且只能接受 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⁾)。如果要发送的字节超过 32 字节 (最大数据包大小), 数据必须被分为几个 32 字节的包发送, 随后是一个剩余包。如果要发送的字节数是 32 的倍数, 剩余包就是一个零长度数据包, 因为包大小小于 32 字节表示传输结束。

⁽¹⁾ 对于同步传输不能有来自主机的握手包

固件应当加载 EP0 FIFO 的第一个数据包, 并且只要 USBCS0.CLR_OUTPKT_RDY 位置 1, 就将 USBCS0.INPKT_RDY 位置 1。当数据包被发送后, 清除 USBCS0.INTPKT_RDY 位并产生一个 EP0 中断。然后固件可以加载更多需要的数据包。每发送一个包就产生一个 EP0 中断。当最后一个数据包被加载完毕, 固件必须设置 USBCS0.DATA_END 位和 USBCS0.INPKT_RDY 位。这开始控制传输的状态阶段。

当状态阶段完成, EP0 切换到 IDLE 状态。如果 USBCS0.SEND_STALL 位设置为 1, 状态阶段可能会失败。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 控制器就回复一个 NAK, 表示端点正在工作, 但是暂时没有数据要发送。

17.6.4 OUT 传输 (RX 状态)

如果控制传输要求接收来自主机的数据, 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, 并且只能接受 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⁾)。如果要接收的字节超过 32 字节 (最大数据包大小), 数据必须被分为几个 32 字节的包, 随后是一个剩余包。如果要接收的字节数是 32 的倍数, 剩余包就是一个零长度数据包, 因为有效载荷小于 32 字节的数据包表示传输结束。

当接收到一个数据包就设置 USBCS0.OUTPKT_RDY 位, 并产生一个 EP0 中断。当数据包已经从 EP0 FIFO 卸载, 固件应当设置 USBCS0.CLR_OUTPKT_RDY 位。当最后一个数

据包（包大小小于 32 字节）已经被接收完毕，固件还应当设置 USBCS0.CATA_END 位。这开始控制传输的状态阶段。数据包的大小保存在寄存器 USBCNT0 中。注意，只有当 USBCS0.OUTPKT_RDY=1 时，该值才有效。

当状态阶段完成，EP0 切换到 IDLE 状态。如果接收到的 DATA1 数据包不是一个零长度数据包，或者如果 USBCS0.SEND_STALL 位设置为 1，状态阶段可能会失败。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包

17.7 端点 1-5

每个端点可以只用作 IN，只用作 OUT，或者用作 IN/OUT。对于一个 IN/OUT 端点，基本上有两个端点，与端点号关联的一个 IN 端点和一个 OUT 端点。IN 端点的配置和控制通过 USBCSIL 和 USBCSIH 寄存器来实现。USBCSOL 和 USBCSOH 寄存器用来配置和控制 OUT 端点。每个 IN 和 OUT 端点都可以被配置为一个同步端点（USBCSIH.ISO=1 和/或 USBCSOH.ISO=1）或批量/中断端点（USBCSIH.ISO=0 和/或 USBCIOH.ISO=0）。USB 控制器对批量端点和中断端点的处理相同，但是这两种端点在固件方面有不同的特性。

在访问编入索引的端点寄存器之前，USBINDEX 寄存器必须有端点号的值。

17.7.1 FIFO 管理

每个端点都有一个特定大小的 FIFO 存储器字节，这个 FIFO 存储器字节可以被输入和输出数据包使用。表 17-2 给出了用于端点 1-5 的 FIFO 大小。固件负责为每个端点正确设置 USBMAXI 和 USBMAXO 寄存器，防止数据被覆盖。

如果一个端点号的 IN 和 OUT 端点都不使用双缓冲，USBMAXI 和 USBMAXO 的总和不得超过该端点的 FIFO 大小。图 17-2a) 显示了一个端点的 IN 和 OUT FIFO 存储器是如何被组织为单缓冲的。IN FIFO 从端点存储器区域的顶部减少，而 OUT FIFO 从端点存储器区域的底部增长。

如果一个端点号的 IN 或 OUT 端点使用了双缓冲，USBMAXI 和 USBMAXO 的总和不得超过该端点 FIFO 大小的一半。图 17-2b) 显示了一个使用双缓冲的端点的 IN 和 OUT FIFO 存储器。注意，第二个 OUT 缓冲区从存储器区域的中间开始，向上增长。第二个 IN 缓冲区也从存储器区域的中间开始，但是向下增长。

要配置一个端点只为 IN，设置 USBMAXO 为 0，要配置一个端点只为 OUT，设置 USBMAXI 为 0。

对于未使用的端点，USBMAXO 和 USBMAXI 都应当被设置为 0。

表 17-2 端点 1-5 的 FIFO 大小

端点号	FIFO 大小（以字节为单位）
1	32
2	64
3	128
4	256
5	512

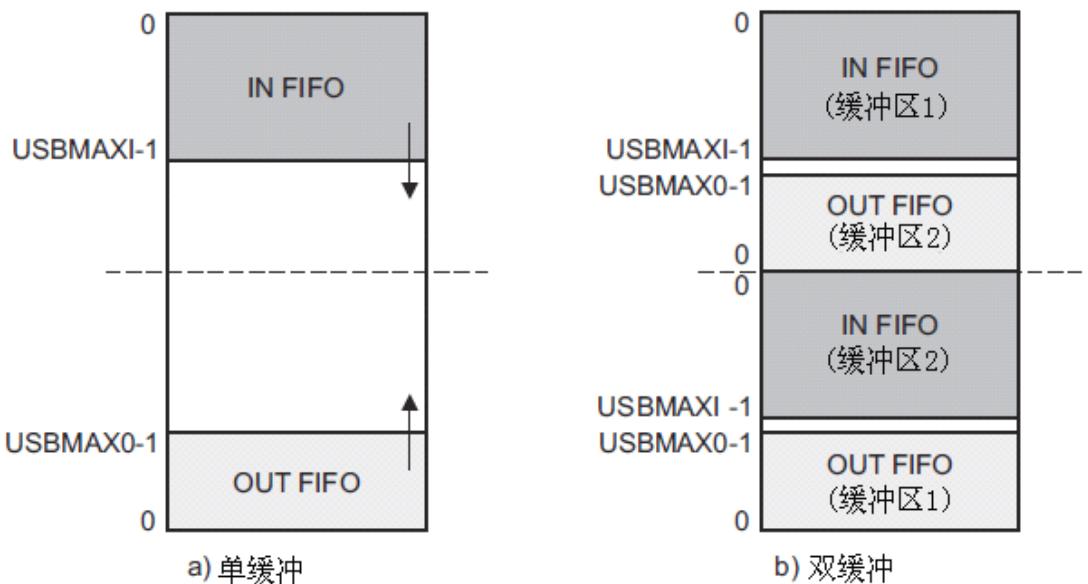


图 17-2 IN/OUT FIFO

M0106-01

17.7.2 双缓冲

要使传输更快，并减少重传的需要，可以使用双缓冲。这允许每个方向的 2 个包被缓冲到 FIFO。对于同步端点，强烈建议使用这种方法，这可以实现每个 USB 帧传输一个数据包，而不带任何重传。对于一个同步端点，每个 USB 帧发送/接收一个数据包。但是，在 USB 帧周期期间，数据包可能在任何时候被发送/接收，并且两个数据包有可能在几微秒的间隔内被发送/接收。对于同步端点，如果没有缓冲可用，一个输入包就会丢失，并且当 USB 主机请求数据时，如果没有数据包准备好传输，就会发送一个零长度数据包。对于批量和中断端点，双缓冲就不像对于同步端点那么重要，因为包不会丢失。但是，对于批量端点，双缓冲可以提高有效数据率。

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

17.7.3 FIFO 存取

通过写和读寄存器 USBF0—USBF5 来存取端点 FIFO。写一个寄存器会导致写入的字节被插入到 IN FIFO。读一个寄存器会导致 OUT FIFO 的下一个字节被提取，并且返回该字节的值。

如果一个数据包被写入一个 IN FIFO，USBCSIL.INPKT_RDY 位必须被设置为 1。如果使能了双缓冲，USBCSIL.INPKT_RDY 位在数据包被写入 IN FIFO 后立即清除，另一个数据包就可以被加载了。这不会产生一个 IN 端点中断，因为只有当一个包已经被发送了才会产生一个中断。如果使能了双缓冲，在写 IN FIFO 之前，固件应该检查 USBCSIL.PKT_PRESENT 位的状态。如果该位是 0，可以写 2 个数据包。第一个加载 IN FIFO 时，双缓冲同步端点只能加载两个包。之后，每个 USB 帧加载一个包。要发送一个零长度数据包，USBCSIL.INPKT_RDY 位应当被设置为 1，而不需加载一个数据包到 IN FIFO。

当 USBCSOL.OUTPKT_RDY 位是 1 时，可以从 OUT FIFO 读出一个数据包。如果使能了，当这种情况发生时就产生一个中断。数据包的大小保存在 USBCNTH: USBCNTL 寄存器中。注意只有当 USBCSOL.OUTPKT_RDY=1 时该值才有效。当数据包已经被从 OUT FIFO 中读出时，USBCSOL.OUTPKT_RDY 位必须被清零。如果使能了双缓冲，FIFO 中可能有两个数据包。当 USBCSOL.OUTPKT_RDY 位已经被清除，如果另一个数据包准备好，就立即声明 USBCSOL.OUTPKT_RDY 位，并产生一个中断（如果使能了），表示已经接收到一个新数据包。当 OUT FIFO 里有两个数据包时，USBCSOL.FIFO_FULL 位被置位。

OUT 端点支持自动清除功能。如果使能了自动清除功能，当已经从 OUT FIFO 读取了 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 来使能。自动清除功能可以被用来减少数据包占用 OUT FIFO 缓冲区的时间，通常用于批量端点。

IN 端点支持一个互补的自动设置功能。如果使能了自动设置功能，当已经写入 USBMAXI 字节到 IN FIFO，USBCSIL.INPKT_RDY 位被自动设置。自动设置功能通过设置 USBCSIH.AUTOSET=1 来使能。自动设置功能可以被用来减少发送一个数据包所需要的总时间，通常用于批量端点。

17.7.4 端点 1-5 中断

下列事件可以产生一个 IN EPx 中断请求（x 表示端点号）：

- 一个被加载到 IN FIFO 的数据包已经被发送到 USB 主机。（当一个新的数据包已经准备好被传送时，USBCSIL.INPKT_RDY 应当被设置为 1。当数据包已经被发送时该位由硬件清除）
- 已经发送了一个 STALL (USBCSIL.SENT_STALL=1)。只有批量/中断端点可以被停止。
- 由于 USBCSIH.FLUSH_PACKET 位被设置为 1，IN FIFO 已经被清除。

不管 IN EPx 中断使能位 USBIIE.INEPxIE 的状态是使能还是禁止，以上任何一个事件都会导致声明 USBIIF.INEPxIF。如果 IN EPx 中断使能位设置为 1，那么 CPU 中断标志 IRCON2.P2IF 也会被声明。只有 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 至的是端点号 1-5。

下列事件可以产生一个 OUT EPx 中断请求：

- 已经接收到一个数据包 (USBCSOL.OUTPKT_RDY=1)
- 已经发送了一个 STALL (USBCSIL.SENT_STALL=1)。只有批量/中断端点可以被停止。

不管 OUT EPx 中断使能位 USBOIE.OUTEPxIE 的状态是使能还是禁止，以上任何一个事件都会导致声明 USBOIF.OUTEPxIF。如果 OUT EPx 中断使能位设置为 1，那么 CPU 中断标志 IRCON2.P2IF 也会被声明。只有 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 端点

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、中断的可用带宽传输或者控制传输。

中断 IN 端点可以设置 USBCSIH.FORCE_DATA_TOG 位。当该位为 1 时，不管是否接收到一个 ACK，数据切换位不断切换。中断 IN 端点通常使用该功能来为同步端点做通信速率反馈。

通过设置 USBCISL.SEND_STALL 位为 1，可以停止一个批量/中断 IN 端点。当端点被停止，USB 控制器响应一个 STALL 握手到 IN 令牌。然后 USBCSIL.SENT_STALL 位被置位，如果使能了中断，还会产生一个中断。

大于最大包大小的批量传输，被分为几个数据包进行传输，这些数据包的大小为最大包大小，随后是一个较小的数据包，它包含了剩余字节。如果传输长度是最大包大小的倍数，最后发送的数据包时一个零长度数据包。这意味着一个大小小于最大包大小的包表示传输结束。在这种情况下，自动设置功能是很有用的，因为许多数据包都是最大包大小。

17.7.6 同步 IN 端点

同步 IN 端点用来将来自 USB 控制器的周期数据传输到主机(每个 USB 帧一个数据包)。

当 USB 主机请求数据时，如果 IN FIFO 里没有加载数据包，USB 控制器就发送一个零长度数据包，并且声明 USBCSIL.UNDERUN 位。

双缓冲要求一个数据包被加载到 IN FIFO 的帧中间，位置在将要发送的帧的前面。如果在接收到 IN 令牌之前加载了第一个数据包，数据包就像它是被加载在帧一样在同一个帧期间被发送，也就违反了双缓冲策略。因此，如果使用了双缓冲，USBPOW.ISO_WAIT_SOF 位应当被设置为 1，以避免这种情况发生。设置该位为 1，可以保证直到收到下一个 SOF 令牌，才会发送被加载的数据包。

自动设置功能通常不用于同步端点，因为根据帧的不同，包的大小也会不同（增加或减少）。

17.7.7 批量/中断 OUT 端点

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、中断的可用带宽传输或者控制传输。

通过设置 USBCSOL.SEND_STALL 位为 1 可以停止批量/中断 OUT 端点。如果端点被停止了，当主机完成了数据包发送，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，如果使能了 OUT 端点中断，还会产生一个中断请求。

正如自动设置功能对批量 IN 端点有用一样，自动清除功能对 OUT 端点也有用，因为许多包的大小都是最大包大小。

17.7.8 同步 OUT 端点

同步 OUT 端点用来将来自主机的周期数据传输到 USB 控制器（每个 USB 帧一个数据包）。

当接收到一个数据包时，如果没有缓冲区可用，就声明 USBCSOL.OVERRUN 位，并且该数据包丢失。固件通过使用双缓冲并使用 DMA 来进行有效卸载数据包，可以减少这种

情况的发生。,

OUT FIFO 中的同步数据包可能有位错误。硬件检测到这种情况，就置位 USBCSOL.DATA_ERROR。因此，固件就应当在卸载一个数据包时总是检查该位。

自动清除功能通常不用于同步端点，因为根据帧的不同，包的大小也会不同（增加或减少）。

17.8 DMA

可以使用 DMA 来填充 IN 端点 FIFO 和清空 OUT 端点 FIFO。与使用 8051 CPU 相比，使用 DMA 能明显提高读/写性能。因此，强烈建议使用 DMA，除非时序不重要，或要传输的字节很少。

USB 控制器没有 DMA 触发，意味着 DMA 传输必须由固件触发。

应当使用以字节为单位的传输。

17.9 USB 复位

当总线上检测到复位信号时，就声明 USBCIF.RSTIF 标志。如果使能了 USBCIE.RSTIE，还会声明 IRCON2.P2IF，如果 IEN2.P2IE=1，还会产生一个中断请求。当一个 USB 复位发生时，固件应当采取适当的操作。USB 复位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，通常会发生一个或多个复位，在 USB 电缆连接上后就立即发生。

当一个 USB 复位发生时，USB 控制器执行以下操作：

- USBADDR 设置为 0
- USBINDEX 设置为 0
- 所有的端点 FIFO 被清除
- USBCS0、USBCSIL、USBCSIH、USBCSOL、USBCSOH 被清零
- 除了 SOF 和挂起，所有中断被使能
- 产生一个中断请求（如果 IEN2.POIE=1 且 USBCIE.RSTIE=1）

当检测到 USB 复位时，固件应该关闭所有管道，等待一个新的枚举阶段。

17.10 挂起和恢复

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

当处于挂起模式时，来源于 USB 的电流很有限。关于这一点的详细信息请见 USB2.0 规范[3]。为了能够满足挂起电流的要求，当检测到挂起时设备应当进入 PM1。设备不能进入 PM2 或 PM3，因为这会复位 USB 控制器。进入 PM1 之前，48MHz USB PLL 必须关闭。通过设置 USBCTRL.PLL_EN 为 0 来关闭 48MHz USB PLL，并等待 USBCTRL.PLL_LOCKED 被清零。

USB 上的任何有效非空闲信号都会导致 USBCIF.RESUMEIF 被声明，并产生一个中断请求，如果使能了 USB 恢复中断就唤醒系统。

当系统从挂起状态唤醒时（进入主动模式），在 48MHz USB PLL 被激活之前，除了能访问 USBCTRL 寄存器，其它的 USB 寄存器都不能访问。通过设置 USBCTRL.PLL_EN 为 1 来激活 48MHz USB PLL，并等待，直到 USBCTRL.PLL_LOCKED 被置位。

一个 USB 复位也将系统从挂起状态唤醒。如果使能了中断会产生一个 USB 恢复中断请求，但是会设置 USBCIF.RSTIF 中断标志，而不是设置 USBCIF.RESUMEIF 中断标志。

17.11 远程唤醒

通过发送恢复信号到 USB 集线器，USB 控制器可以从挂起状态恢复。通过设置 USBPOW.RESUME 为 1 大约 10ms 后执行恢复。根据 USB2.0 规范[3]，恢复信号必须存在至少 1ms，不超过 15ms。然而，建议保持恢复信号大约 10ms。注意，必须在 USB 描述符中声明支持远程唤醒，这样 USB 主机不许授权设备执行远程唤醒（通过一个 SET_FEATURE 请求）。

17.12 USB 寄存器

本节描述用于 USB 控制和状态的所有 USB 寄存器。USB 寄存器驻留在 XDATA 存储器空间，在区域 0x6200-0x622B。这些寄存器可以分为三组：普通 USB 寄存器、索引端点寄存器和端点 FIFO 寄存器。索引端点寄存器表示当前选择的端点。USBINDEX 寄存器用来选择端点。

USBADDR (0x6200) — 功能地址

位	名称	复位	读/写	描述
7	UPDATE	0	R	写 USBADDR 寄存器时置位该位，地址有效时清除该位。
6: 0	USBADDR[6: 0]	000 0000	R/W	设备地址

USBPOW (0x6201) — 电源/控制寄存器

位	名称	复位	读/写	描述
7	ISO_WAIT_SOF	0	R/W	如果该位设置为 1，从声明 INPKT_RDY 开始，USB 控制器发送零长度数据包，直到接收到第一个 SOF 令牌。这仅适用于同步端点。
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	用于远程唤醒的设备恢复信号。根据 USB 规范，驱动恢复的周期必须为至少 1ms，不超过 15ms。建议保持该位的设置大约 10ms。
1	SUSPEND	0	R	进入挂起模式。只有当 SUSPEND_EN=1 时才使用该位。读 USBCIF 寄存器或声明 RESUME 会清除该位。
0	SUSPEND_EN	0	R/W	挂起使能。如果该位设置为 1，当 USB 保持空闲持续了 3ms 就进入挂起模式。

USBIIF (0x6202) —IN 端点和 EP0 中断标志

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

USBOIF (0x6204) —OUT 端点中断标志

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

USBCIF (0x6206) —普通 USB 中断标志

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

USBIE (0x6207) —IN 端点和 EP0 中断使能

位	名称	复位	读/写	描述
7: 6		00	R/W	保留。总是写 00
5	INEP5IE	1	R/W	IN 端点 5 中断使能 0: 中断禁止 1: 中断使能
4	INEP4IE	1	R/W	IN 端点 4 中断使能 0: 中断禁止 1: 中断使能
3	INEP3IE	1	R/W	IN 端点 3 中断使能 0: 中断禁止 1: 中断使能
2	INEP2IE	1	R/W	IN 端点 2 中断使能 0: 中断禁止

				1: 中断使能
1	INEP1IE	1	R/W	IN 端点 1 中断使能 0: 中断禁止 1: 中断使能
0	EP0IE	1	R/W	端点 0 中断使能 0: 中断禁止 1: 中断使能

USBOIE (0x6209) —OUT 端点中断使能

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

USBCIE (0x620B) —普通 USB 中断使能

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

USBFRML (0x620C) —当前帧号码(低字节)

位	名称	复位	读/写	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	读/写	描述
7: 3	—	—	R0	未使用
2: 0	FRAME[10: 8]	000	R	11位帧号码的最高有效3位

USBINDEX (0x620E) —当前端点索引寄存器

位	名称	复位	读/写	描述
7: 4	—	—	R0	未使用
3: 0	USBINDEX[3: 0]	0000	R/W	端点选择。必须设置为0-5之间的值

USBCTRL (0x620F) —USB 控制寄存器

位	名称	复位	读/写	描述
7	PLL_LOCKED	0	R	PLL 锁定状态
6: 3	—	—	R0	未使用
2	PLL_LOCK	0	R/W	保留。总是写0
1	PLL_EN	0	R/W	48MHz USB PLL 使能。该位为1时，48MHz PLL 启动。 但是，在PLL被锁定之前，USB不能被访问，即 PLL_LOCKED 为1。只有当USB_EN=1时才能设置该位。 注意：在退出主动模式之前，PLL 必须被禁止，进入主动 模式时再重新使能 PLL。
0	USB_EN	0	R/W	USB 使能。只有当CHIPID=0xB5时，该位才能被设置。当 写0到该位时，USB控制器复位。

USBMAXI (0x6210) —IN 端点{1-5}的最大包大小

位	名称	复位	读/写	描述
7: 0	USBMAXI[7: 0]	0x00	R/W	由USBINDEX 寄存器选择的 IN 端点的最大包大小（以8 字节为单位）。该寄存器的值必须对应端点的标准端点描述 符里的 wMaxPacketSize 域。该寄存器的值必须设定为大于 该端点可用的 FIFO 存储器的值。

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

位	名称	复位	读/写	描述
7	CLR_SETUP_END	0	R/W H0	设置该位为1，解除对该寄存器 SETUP_END 位的声明。 该位被自动清除。
6	CLR_OUTPKT_RDY	0	R/W H0	设置该位为1，解除对该寄存器 OUTPKT_RDY 位的声明。 该位被自动清除。
5	SEND_STALL	0	R/W H0	设置该位为1，中止当前传输。USB 控制器发送 STALL 握手，而且该位被解除声明。

4	SETUP_END	0	R	如果控制传输是由于提前结束控制传输而结束的，该位置位。FIFO 被清除，如果使能的中断，产生一个中断请求 (EP0)。设置 CLR_SETUP_END=1 来解除对该位的声明。
3	DATA_END	0	R/W H0	该位用来表示一个数据传输的结束，在下面 3 种情况下，该位必须被声明： 1: 最后一个数据包已经被加载，并且 USBCS0.INPkt_RDY 设置为 1 2: 最后一个数据包已经被卸载，并且 USBCS0.CLR_OUTPkt_RDY 设置为 1 3: 不加载 FIFO 就声明了 USBCS0.INPkt_RDY (发送一个零长度数据包) USB 控制器自动清除该位。
2	SEND_STALL	0	R/W H1	当发送了一个 STALL 握手，该位被置位。如果使能了中断，会产生一个中断请求 (EP0)。该位必须从固件里清除。
1	INPkt_RDY	0	R/W H0	当一个数据包已经被加载到 EP0 FIFO，设置该位，用来通知 USB 控制器，一个新的数据包已经准备好被传输。当数据包被发送后，该位被清除，如果使能了中断，会产生一个中断请求 (EP0)。
0	OUTPkt_RDY	0	R	接收到数据包。当一个接收的数据包已经被放入 OUT FIFO，该位置位。如果使能了中断，会产生一个中断请求 (EP0)。设置 CLR_OUTPkt_RDY=1 来解除对该位的声明。

USBCSIL (0x6211) —IN 端点{1-5}控制和状态，低位

位	名称	复位	读/写	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为 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	当发送了一个 STALL 时该位置位。FIFO 被清除，并且该寄存器的 INPkt_RDY 位被解除声明。如果使能了中断，会产生一个中断请求 (IN EP{1-5})。该位必须由固件清除。
4	SEND_STALL	0	R/W	当接收到 IN 令牌时，设置该位为 1，使 USB 控制器回复一个 STALL 握手。固件必须清除该位来结束 STALL 条件。不可能停止一个同步端点；因此，只有当 IN 端点被配置为批量/中断端点时，该位才有效。
3	FLUSH_PACKET	0	R/W H0	该位设置为 1 来清除准备从 IN FIFO 传输的下一个包。该寄存器的 INPkt_RDY 位被清除。如果由于双缓冲，IN FIFO 里有 2 个包，该位必须被置位 2 次以完全清除 IN FIFO。该位自动清除。
2	UNDERRUN	0	R/W	在同步模式，当 INPkt_RDY=0 时，如果接收到一个 IN 令牌，该位置位，并发送一个零长度数据包来响应 IN 令牌。在批量/中断模式，当返回一个 NAK 来响应一个 IN

				令牌的时候，该位置位。固件清除该位。
1	PKT_PRESENT	0	R	当 IN FIFO 里至少有 1 个包时该位为 1。
0	INPKT_RDY	0	R/W H0	当一个数据包已经被加载到 IN FIFO 时设置该位，通知 USB 控制器有一个新的数据包准备好被传输。如果数据包已经被发送，该位清除，如果使能了中断，产生一个中断请求 (IN EP{1-5})。

USBCSIH (0x6212) —IN 端点{1-5}控制和状态，高位

位	名称	复位	读/写	描述
7	AUTOSET	0	R/W	如果该位为 1，当一个为最大包大小(由 USBMAXI 指定)的数据包被加载到 IN FIFO 时，USBCSIL.INPKT_RDY 位自动声明。
6	ISO	0	R/W	选择 IN 端点类型 0: 批量/中断 1: 同步
5: 4		10	R/W	保留。总是写为 10
3	FORCE_DATA_TOG	0	R/W	设置该位强制 IN 端点数据切换，并且数据包从 IN FIFO 里清除，即使收到一个 ACK。该功能可用于同步端点报告率反馈。
2: 1		—	R0	未使用
0	IN_DBL_BUF	0	R/W	双缓冲使能 (IN FIFO) 0: 双缓冲禁止 1: 双缓冲使能

USBMAXO (0x6213) —OUT 端点{1-5}的最大包大小

位	名称	复位	读/写	描述
7: 0	USBMAXO[7: 0]	0x00	R/W	由 USBINDEX 寄存器选择的 OUT 端点的最大包大小(以8字节为单位)。该寄存器的值必须对应端点的标准端点描述符里的 wMaxPacketSize 域。该寄存器的值必须设定为大于该端点可用的 FIFO 存储器的值。

USBCSOL (0x6214) —OUT 端点{1-5}控制和状态，低位

位	名称	复位	读/写	描述
7	CLR_DATA_TOG	0	R/W H0	设置该位来复位数据切换为 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	当发送了一个 STALL 握手时该位置位。如果使能了中断，会产生一个中断请求 (OUT EP{1-5})。该位必须由固件清除。
5	SEND_STALL	0	R/W	当接收到 OUT 令牌时，设置该位为 1，使 USB 控制器回复一个 STALL 握手。固件必须清除该位来结束 STALL 条件。不可能停止一个同步端点；因此，只有当 IN 端点被配置为批量/中断端点时，该位才有效。
4	FLUSH_PACKET	0	R/W	该位设置为 1 来清除从 OUT FIFO 读取的下一个包。该寄

		H0	存器的 OUTPKT_RDY 位被清除。如果由于双缓冲，OUT FIFO 里有 2 个包，该位必须被置位 2 次以完全清除 OUT FIFO。该位自动清除。	
3	DATA_ERROR	0	R	如果接收的包里有一个 CRC 或位填充错误，该位置位。 当 OUTPKT_RDY 被清除时清除该位。只有当 OUT 端点为同步端点是该位才有效。
2	OVERRUN	0	R/W	当一个 OUT 包不能被加载到 OUT FIFO 时该位置位。固件清除该位。只有在同步模式下该位才有效。
1	FIFO_FULL	0	R	当 OUT FIFO 已经装满，不能加载更多的包时声明该位。
0	OUTPKT_RDY	0	R/W	当一个数据包已经被接收到并且准备好从 OUT FIFO 里被读取时设置该位。如果使能了中断，产生一个中断请求(OUT EP{1-5})。当包已经从 FIFO 里卸载时该位应当被清除。

USBCSOH (0x6215) —OUT 端点{1-5}控制和状态，高位

位	名称	复位	读/写	描述
7	AUTOCLEAR	0	R/W	如果该位为 1，当一个为最大包大小（由 USBMAXO 指定）的数据包从 OUT FIFO 里被卸载时，USBCSOL.OUTPKT_RDY 位自动清除。
6	ISO	0	R/W	选择 OUT 端点类型 0：批量/中断 1：同步
5: 4		00	R/W	保留。总是写为 00
3: 1		—	R0	未使用
0	OUT_DBL_BUF	0	R/W	双缓冲使能 (OUT FIFO) 0：双缓冲禁止 1：双缓冲使能

USBCNT0 (0x6216) —EP0 FIFO 里接收到的字节数 (USBINDEX=0)

位	名称	复位	读/写	描述
7: 6	—	—	R0	未使用
5: 0	USBCNT0[5: 0]	00 0000	R	EP0 FIFO 里接收到的字节数。只有当 OUTPKT_RDY 被声明时有效

USBCNTL (0x6216) —EP{1-5} OUTFIFO 里的字节数，低位

位	名称	复位	读/写	描述
7: 0	USBCNT[7: 0]	0x00	R	由 USBINDEX 寄存器选择的端点 OUT FIFO 里接收到的字节数的最低 8 位。只有当 USBCSOL.OUTPKT_RDY 被声明时有效

USBCNTH (0x6217) —EP{1-5} OUTFIFO 里的字节数，高位

位	名称	复位	读/写	描述
7: 3	—	—	R0	未使用

2: 0	USBCNT[10: 8]	000	R	由 USBINDEX 寄存器选择的端点 OUT FIFO 里接收到的字节数的最高 3 位。只有当 USBCSOL.OUTPKT_RDY 被置位时有效
------	---------------	-----	---	--

USBF0 (0x6220) —端点 0 FIFO

位	名称	复位	读/写	描述
7: 0	USBF0[7: 0]	0x00	R/W	端点 0 FIFO。读该寄存器时从 EP0 FIFO 里卸载一个字节。 写该寄存器是往 EP0 FIFO 里加载一个字节。 注意：EP0 的 FIFO 存储器用于输入和输出数据包。

USBF1 (0x6222) —端点 1 FIFO

位	名称	复位	读/写	描述
7: 0	USBF1[7: 0]	0x00	R/W	端点 1 FIFO。读该寄存器时从 EP1 FIFO 里卸载一个字节。 写该寄存器是往 EP1 FIFO 里加载一个字节。

USBF2 (0x6224) —端点 2 FIFO

位	名称	复位	读/写	描述
7: 0	USBF2[7: 0]	0x00	R/W	端点 2 FIFO。读该寄存器时从 EP2 FIFO 里卸载一个字节。 写该寄存器是往 EP2 FIFO 里加载一个字节。

USBF3 (0x6226) —端点 3 FIFO

位	名称	复位	读/写	描述
7: 0	USBF3[7: 0]	0x00	R/W	端点 3 FIFO。读该寄存器时从 EP3 FIFO 里卸载一个字节。 写该寄存器是往 EP3 FIFO 里加载一个字节。

USBF4 (0x6228) —端点 4 FIFO

位	名称	复位	读/写	描述
7: 0	USBF4[7: 0]	0x00	R/W	端点 4 FIFO。读该寄存器时从 EP4 FIFO 里卸载一个字节。 写该寄存器是往 EP4 FIFO 里加载一个字节。

USBF5 (0x622A) —端点 5 FIFO

位	名称	复位	读/写	描述
7: 0	USBF5[7: 0]	0x00	R/W	端点 5 FIFO。读该寄存器时从 EP5 FIFO 里卸载一个字节。 写该寄存器是往 EP5 FIFO 里加载一个字节。

18 定时器 2 (MAC 定时器)

定时器 2 主要用来提供用于 802.15.4 CSMA-CA 的算法定时和 802.15.4 MAC 层上的一般计时。当定时器 2 和睡眠定时器一起使用时，即使系统进入低功耗模式，仍然提供定时功能。定时器的运行速度取决于 CLKCONSTA.CLKSPD。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 32MHz，为了获得精确结果，还应当使用一个外部 32kHz 晶体振荡器。

定时器 2 的主要特征如下：

- 16 位定时器正计数，提供符号 (symbol) 周期 16us，帧 (frame) 周期 320us
- 周期可调，精度为 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
- 帧开始定界符的捕获功能
- 定时器的开始/停止与外部 32KHz 时钟同步，由睡眠定时器保持定时
- 比较和溢出产生中断
- DMA 触发能力
- 通过引入延迟计数可以调整定时器值

18.1 定时器操作

本节描述定时器 2 操作。

18.1.1 概述

复位后，定时器 2 处于空闲 (IDLE) 模式，此时已经停止运行。当 T2CTRL.RUN 设置为 1 时，定时器 2 就开始进入运行 (RUN) 模式了。该操作可以立即进行，也可以与 32kHz 时钟同步进行。请见 18.4 小节关于同步开始和停止模式的描述。

进入运行模式的定时器 2，当 T2CTRL.RUN 设置为 0 时，就会停止，从而进入空闲模式。该操作可以立即进行，也可以与 32kHz 时钟同步进行。

18.1.2 正计数

定时器 2 包含一个 16 位的定时器，其计数随每个时钟周期递增。寄存器 T2MSEL.T2MSEL 设置为 000 时，可以从寄存器 T2M1: T2M0 读取计数器值。注意读取 T2M0 时，T2M1 里的内容是锁定的，也就是说必须总是首先读 T2M0。

定时器空闲时，寄存器 T2MSEL.T2MSEL 设置为 000，通过写寄存器 T2M1: T2M0 可以修改计数器。必须首先写 T2M0。

18.1.3 定时器溢出

当定时器所计数值等于设置的定时器周期值时，发生溢出。当发生定时器溢出时，定时器的值设置为 0x0000。如果溢出中断屏蔽位 T2IRQM.TIMER2_PERM 为 1，则产生中断请求。不管中断屏蔽位是什么值，此时中断标志位 T2IRQF.TIMER2_PERF 置 1。

18.1.4 定时器 delta 增量

定时器周期可以在一个定时器周期里面通过写定时器的 delta 值予以调整。当定时器正在运行时，将定时器 delta 值写入复用寄存器 T2M1: T2M0，并且 T2MSEL.T2MSEL 设置为 000，16 位定时器在它当前计数值处停止计数，而 delta 计数器开始计数。写寄存器 T2M1 之前必须先写寄存器 T2M0。delta 计数器从写入 delta 值起，开始倒计数，直到 0 为止，然后 16 位定时器重新开始计数。

delta 倒计数的速率与定时器等同。当 delta 计数器倒计数到达 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，从而调整定时器溢出事件的溢出值。

18.1.5 定时器比较

当定时器将要计数的值接近预置的 16 位比较值之一时，就发生了定时器比较。发生定时器比较时，根据计数值达到了哪个比较值，就将相应的中断标志位 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，则产生中断请求。

18.1.6 溢出计数

每当定时器溢出时，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，同时寄存器 T2MSEL.T2MOVEFSEL 设置为 000。下面描述锁定的寄存器。

如果想要一个唯一的时间戳，即定时器和溢出计数器同时锁定，操作如下：读 T2M0，并将 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就准备好被读取了。

如果想要不首先读取定时器而读溢出计数器，那就读 T2MOVF0，并将 T2MSEL.T2MOVEFSEL 设置为 000，T2CTRL.LATCH_MODE 设置为 0。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，这样溢出计数器的值就准备好被读取了。

18.1.7 溢出计数更新

通过写寄存器 T2MOVF2: T2MOVF1: T2MOVF0 并将 T2MSEL.T2MOVFSEL 设置为 000，可以更新溢出计数值。总是首先写最低位字节，并且要写 3 个字节。一旦写入高字节，对溢出计数器的更新就开始生效。

18.1.8 溢出计数溢出

当溢出计数器所计数值等于设置的溢出周期时，发生溢出周期事件。当发生溢出周期事件时，溢出计数器设置为 0x00 0000。如果溢出中断屏蔽位 T2IRQM.TIMER2_OVF_PERM 为 1，则产生中断请求。不管中断屏蔽位是什么值，此时中断标志位 T2IRQF.TIMER2_OVF_PERF 置 1。

18.1.9 溢出计数比较

可以为溢出计数器设置 2 个比较值。通过写入寄存器 T2MOVF2: T2MOVF1: T2MOVF0，并将寄存器 T2MSEL.T2MOVFSEL 设置为 011 或 100，可以设置溢出计数器的比较值。当溢出计数器的计数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_COMPARE2M 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，此时中断标志位 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）的状态变高时动作。

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，可以从寄存器 T2M1: T2M0 中读取捕获值。溢出计数值也可以在捕获事件发生时捕获，如果寄存器 T2MSEL.T2MOVFSEL 设置为 001，可以从寄存器 T2MOVF2: T2MOVF1: T2MOVF0 中读取捕获值。

18.2 中断

定时器有 6 个可以个别屏蔽的中断源。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

中断标志在寄存器 T2IRQF 中给定。中断标志位只能通过硬件设置，且只能通过写 SFR 寄存器加以清除。

每个中断源可以通过寄存器 T2IRQM 的屏蔽位加以屏蔽。当相应的屏蔽位被置位会产生中断，否则将不产生中断。然而不管中断屏蔽位的状态是什么，中断标志位都将置位。

18.3 事件输出（DMA 触发和 CSP 事件）

定时器 2 有 2 个事件输出：T2_EVENT1 和 T2_EVENT2。这 2 个事件输出可以作为 DMA 触发或作为 CSP 条件指令里的条件。事件输出可以分别配置为下列事件中的任何事件：

- 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

使用 T2EVTCFG.TIMER2_EVENT1_CFG 和 T2EVTCFG.TIMER2_EVENT2_CFG 来配置 DMA 触发。

18.4 定时器开始/停止同步

本节描述定时器开始和停止同步。

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 32MHz 的系统时钟同步，因此，一个周期近似等于 32kHz 时钟周期。除非 32kHz 时钟和 32MHz 晶体振荡器都稳定运行，否则不要企图同步开始和停止。

开始同步时，定时器要重新装入新计算出来的定时器和溢出计数值，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

在定时器开始运行后，即进入了定时器 RUN 模式，如果 T2CTRL.SYNC 是 1，可以通过将 0 写入 T2CTRL.RUN 停止同步。在 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（触发）为止。此时，定时器停止运行，并且存储当前睡眠定时器值，而 T2CTRL.STATE 从 1 变为 0。

18.4.3 定时器同步开始

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，定时器将保持这种空闲模式，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，该计算基于当前和存储的睡眠定时器值，还有当前的 16 位定时器值。新的定时器 2 值和溢出计数值装入定时器后，定时器就进入 RUN 模式。T2CTRL.STATE=1 表示该模块正在运行。从 32kHz 时钟上升沿被采样为高起，同步开始过程经历了 86 个时钟周期。同步开始和停止功能，需要选择系统时钟频率为 32MHz。如果系统时钟频率选择为 16MHz，则需要添加一个补偿给新的计算值。

如果前一个同步还没停止就开始一个同步，就会加载不可预测的值到定时器。为了避免这种情况，先启动定时器同步，然后为随后的停止和开始启动同步模式使能。

下面给出新定时器 2 值和溢出计数值的计算方法。事实上，由于定时器 2 时钟和睡眠定时器时钟异步的，而且它们之间的比例不是整数，这样，不考虑时钟误差，计算出来的定时器值与理想值之间的误差最大达到±1。

计算新的定时器值和溢出计数值：

$$N_c = \text{CurrentSleepTimerValue}$$

$$N_{ST} = \text{StoredSleepTimerValue}$$

$$K_{ck} = \text{ClockRatio} = 976.5625^{(1)}$$

$$stw = \text{SleepTimerWidth} = 24$$

$$P_T = \text{Timer2Period}$$

$$P_{OVF} = \text{OverflowPeriod}$$

$$O_{ST} = \text{StoredOverflowCountValue}$$

$$O_{TICK} = \text{OverflowTicsWhileSleeping}$$

$$T_{ST} = \text{StoredTimerValue}$$

$$T_{OH} = \text{Overhead} = 86$$

$$N_d = N_c - N_{ST}$$

$$N_t \leq 0 \rightarrow N_d = 2^{stw} + N_t; N_t > 0 \rightarrow N_d = N_t$$

$$C = N_d \bullet K_{ck} + T_{ST} + T_{OH} \text{ (规整到最接近的整数值)}$$

$$T = C \bmod P_T$$

$$\text{Timer2Value} = T$$

$$O_{TICK} = \frac{(C - T)}{P_T}$$

$$O = (O_{TICK} + O_{ST}) \bmod P_{OVF}$$

$$\text{Timer2Overflowcount} = O$$

⁽¹⁾ 定时器 2 时钟频率 (32MHz) 和睡眠定时器时钟频率 (32kHz) 的时钟比例。

对于一个给定的定时器 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，用于在同步开始后定时器值的正确更新。这个最大值是以睡眠定时器时钟周期数来给定的，即 32kHz 时钟周期 $T_{ST(max)}$ ：

$$T_{ST(MAX)} \leq \frac{(2^{24}-1) \times P + T_{OH}}{K_{ck}}$$

18.5 定时器 2 寄存器

本节列出了与定时器 2 有关的 SFR 寄存器，列举如下：

- T2MSEL— 定时器 2 复用寄存器控制
- T2M1—定时器 2 复用计数高位
- T2M0—定时器 2 复用计数低位
- 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 有几个复用寄存器。这是为了所有的寄存器能够处于有限的 SFR 地址空间。

表 18-1 中列出的内部寄存器可以通过 T2M0、T2M1、T2MOVF0、T2MOVF1 和 T2MOVF2 间接访问。

表 18-1 内部寄存器

寄存器名称	复位	读/写	描述
t2tim[15: 0]	0x0000	R/W	保存 16 位正计数器
t2_cap[15: 0]	0x0000	R	保存正计数器的最后一个捕获值
t2_per[15: 0]	0x0000	R/W	保存正计数器周期
t2_cmp1[15: 0]	0x0000	R/W	为正计数器保存比较值 1
t2_cmp2[15: 0]	0x0000	R/W	为正计数器保存比较值 2
t2ovf[23: 0]	0x000000	R/W	保存 24 位溢出计数器
t2ovf_cap[23: 0]	0x000000	R	保存溢出计数器的最后一个捕获值
t2ovf_per[23: 0]	0x000000	R/W	保存溢出计数器周期
t2ovf_cmp1[23: 0]	0x000000	R/W	为溢出计数器保存比较值 1
t2ovf_cmp2[23: 0]	0x000000	R/W	为溢出计数器保存比较值 2

本节其余所列出的寄存器可以在 SFR 地址空间直接访问。

T2MSEL (0xC3) —定时器 2 复用选择

位	名称	复位	读/写	描述
7: 0	—	0	R0	保留。读为 0
6: 4	T2MOVFSEL	0	R/W	该寄存器的值选择当访问 T2MOVF0、T2MOVF1 和 T2MOVF2 时要修改或读取的内部寄存器。 000: t2ovf (溢出计数器) 001: t2ovf_cap (溢出捕获)

				010: t2ovf_per (溢出周期) 011: t2ovf_cmp1 (溢出比较 1) 100: t2ovf_cmp2 (溢出比较 2) 101—111: 保留
3	—	0	R0	保留。读为 0
2: 0	T2MSEL	0	R/W	该寄存器的值选择当访问 T2M0 和 T2M1 时要修改或读取的内部寄存器。 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—111: 保留

T2M0 (0xA2) —定时器 2 复用寄存器 0

位	名称	复位	读/写	描述
7: 0	T2M0	0	R/W	T2MSEL.T2MSEL 的值决定一个内部寄存器的位[7: 0]的间接返回/修改。 T2MSEL.T2MSEL 设置为 000, T2CTRL.LATCH_MODE 设置为 0, 读 T2M0 寄存器时, 定时器值 (t2tim) 被锁定。 T2MSEL.T2MSEL 设置为 000, T2CTRL.LATCH_MODE 设置为 1, 读 T2M0 寄存器时, 定时器值 (t2tim) 和溢出计数器值 (t2ovf) 被锁定。

T2M1 (0xA3) —定时器 2 复用寄存器 1

位	名称	复位	读/写	描述
7: 0	T2M1	0	R/W	T2MSEL.T2MSEL 的值决定一个内部寄存器的位[15: 8]的间接返回/修改。 T2MSEL.T2MSEL 设置为 000, 读 T2M0 寄存器时, 定时器值 (t2tim) 被锁定。 T2MSEL.T2MSEL 设置为 000, 读 T2M1 寄存器时, 返回 t2tim[15: 8]的锁定值。

T2MOVF0 (0xA4) —定时器 2 复用溢出寄存器 0

位	名称	复位	读/写	描述
7: 0	T2MOVF0	0	R/W	T2MSEL.T2MOVFSEL 的值决定一个内部寄存器的位[7: 0]的间接返回/修改。 T2MSEL.T2MOVFSEL 设置为 000, T2CTRL.LATCH_MODE 设置为 0, 读 T2MOVF0 寄存器时, 溢出计数器值 (t2ovf) 被锁定。 T2MSEL.T2MOVFSEL 设置为 000, T2CTRL.LATCH_MODE 设置为 1, 读 T2M0 寄存器时, 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	名称	复位	读/写	描述
7: 0	T2MOVF1	0	R/W	T2MSEL.T2MOVFSEL 的值决定一个内部寄存器的位[15: 8]的间接返回/修改。 T2MSEL.T2MOVFSEL 设置为 000, 读 T2MOVF1 寄存器时, 返回 t2vof[15: 8]的锁定值。

T2MOVF2 (0xA6) —定时器 2 复用溢出寄存器 2

位	名称	复位	读/写	描述
7: 0	T2MOVF2	0	R/W	T2MSEL.T2MOVFSEL 的值决定一个内部寄存器的位[23: 16]的间接返回/修改。 T2MSEL.T2MOVFSEL 设置为 000, 读 T2MOVF2 寄存器时, 返回 t2vof[23: 16]的锁定值。

T2IRQF (0xA1) —定时器 2 中断标志

位	名称	复位	读/写	描述
7: 6	—	0	R0	保留。读为 0
5	TIMER2_OVF_COMPARE2F	0	R/W	当定时器 2 溢出计数器计数值达到 t2ovf_cmp2 设置的值时, 该位置 1。
4	TIMER2_OVF_COMPARE1F	0	R/W	当定时器 2 溢出计数器计数值达到定时器 2 的 t2ovf_cmp1 设置的值时, 该位置 1。
3	TIMER2_OVF_PERF	0	R/W	当定时器 2 溢出计数器计数值等于 t2ovf_per 时, 该位置 1。
2	TIMER2_COMPARE2F	0	R/W	当定时器 2 计数器计数值达到 t2_cmp2 设置的值时, 该位置 1。
1	TIMER2_COMPARE1F	0	R/W	当定时器 2 计数器计数值达到 t2_cmp1 设置的值时, 该位置 1。
0	TIMER2_PERF	0	R/W	当定时器 2 计数器计数值等于 t2_per 时, 该位置 1。

T2IRQM (0xA7) —定时器 2 中断屏蔽

位	名称	复位	读/写	描述
7: 6	—	0	R0	保留。读为 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	名称	复位	读/写	描述
7: 4	—	0	R0	保留。读为 0

3	LATCH_MODE	0	R/W	0: T2MSEL.T2MSEL=000, 读T2M0时锁定定时器的高字节,使其准备好从T2M1中被读出。T2MOVF0.T2MOVFSEL=000,读T2MOVF0时锁定溢出计数器的两个最高字节,以使可以从T2MOF1和T2MOV2中读取它们。 1: T2MSEL.T2MSEL=000, 读T2M0时立即锁定定时器的高字节和整个溢出计数器,使得可以从T2M1、T2MOVF0、T2MOVF1和T2MOVF2中读取。
2	STATE	0	R	定时器2状态 0: 定时器空闲 1: 定时器正在运行
1	SYNC	1	R/W	0: 立即运行定时器的开始和停止,即和clk_rf_32m同步。 1: 在32kHz时钟的第一个正边沿启动定时器的开始和停止
0	RUN	0	R/W	写1到该位将启动定时器,写0到该位停止定时器。读该位返回最后写入值。

T2EVTCFG (0x9C) —定时器2CSP 接口配置

位	名称	复位	读/写	描述
7	—	0	R0	保留。读为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: 未使用
3	—	0	R0	保留。读为0
2: 0	TEMER2_EVENT2_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: 未使用

19 无线

RF 内核控制模拟无线模块。另外，它为 MCU 和无线之间提供一个接口，可以通过这个接口发出命令、读取状态和自动对无线事件进行排序。

19.1 RF 内核

RF 内核控制模拟无线模块。另外，它为 MCU 和无线之间提供一个接口，可以通过这个接口发出命令、读取状态和自动对无线事件进行排序。

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 用来提供事件的正确顺序（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步处理：读取帧长度，计算接收到的字节数，检查 FCS，最后，在成功接收帧后，可以选择性地处理自动传输 ACK 帧。它在 TX 执行类似任务，包括传输之前执行一个可选的 CCA，在传输完成后自动进入 RX 以接收一个 ACK 帧。最后，FSM 还控制在调制器/解调器和 RAM 里 TXFIFO/RXFIFO 之间的数据传输。

调制器将原始数据转换为 I/Q 信号送到发射机 DAC。这一行为遵守 IEEE 802.15.4 标准。
解调器负责从接收到的信号中检索无线数据。

自动增益控制 (AGC) 使用解调器的波幅信息。AGC 对模拟 LNA 的增益进行调整，使接收器内的信号电平基本不变。

帧过滤和源匹配支持 RF 内核中的 FSM，为了做到帧过滤和源地址匹配，要执行所有操作，如 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。它还有一个很短的 24 字节的程序存储器，使得它可以自动执行 CSMA-CA 算法。

无线 RAM 有一个用于发送数据的 FIFO (TXFIFO) 和一个用于接收数据的 FIFO (RXFIFO)。两个 FIFO 都是 128 字节长。此外，RAM 还保留了 128 字节来保存帧过滤和源匹配的参数。

定时器 2 (MAC 定时器) 用来计时无线事件，捕获输出数据包的时间戳。即使在睡眠模式定时器 2 也能保持计数。

19.1.1 中断

无线模块具有两个 CPU 中断向量：RFERR 中断（中断 0）和 RF 中断（中断 12）。其功能如下：

- RFERR：使用该中断表示无线错误情况。
- RF：使用该中断表示来自正常运行的中断。

RF 中断向量结合了 RFIF 中断。注意：这些 RF 中断均由上升沿触发。因此，例如 SFD 状态标志从 0 变为 1 时会产生一个中断。RFIF 中断标志在 19.1.2 小节进行描述。

19.1.2 中断寄存器

两个主要的中断控制 SFR 寄存器用于使能 RF 和 RFERR 中断:

- RFERR: IEN0.RFERRIE
- RF: IEN2.RFIE

两个主要的中断标志 SFR 寄存器用来保持 RF 和 RFERR 中断标志:

- RFERR: TCON.RFERRIF
- RF: S1CON.RFIF

RF 内核产生的 2 个中断是 RF 内核中若干独立源的组合。每个独立源在 RF 内核中有自己的使能和中断标志。中断标志在寄存器 RFIRQF0、RFIRQF1 和 RFIERRF 中。中断使能在寄存器 RFIRQM0、RFIRQM1 和 RFERRM 中。

使能寄存器中的中断使能位为两个 RF 中断使能独立的中断源。注意，使能一个中断源不会影响标志寄存器中状态的更新。

由于 RF 内核中使用独立的中断使能，所以来自 RF 内核的中断有 2 层使能，那么在处理这些中断时必须小心。过程描述如下。

为了清除来自 RF 内核的中断，必须清除两个标志，即 RF 内核里设置的中断标志和 S1CON 或 TCON（取决于触发了哪个中断）里设置的中断标志。如果 RF 内核里的中断标志已经被清除，但还有其它标志未清除，就会产生另一个中断。

RFIRQF0 (0xE9) —RF 中断标志

位	名称	复位	读/写	描述
7	RXMASKZERO	0	R/W0	RXENABLE 寄存器从一个非 0 状态变为全 0 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: 无中断未决 1: 中断未决
5	FRAME_ACCEPTED	0	R/W0	帧已经通过了帧过滤。 0: 无中断未决 1: 中断未决
4	SRC_MATCH_FOUND	0	R/W0	发现源匹配。 0: 无中断未决 1: 中断未决
3	SRC_MATCH_DONE	0	R/W0	源匹配完成。 0: 无中断未决 1: 中断未决
2	FIFOP	0	R/W0	RX FIFO 中的字节数已超出设置的阈值。当收到一个完整的帧时仍然增加。 0: 无中断未决 1: 中断未决
1	SFD	0	R/W0	收到或发送 SFD。 0: 无中断未决 1: 中断未决

0	ACT_UNUSED	0	R/W0	保留 0: 无中断未决 1: 中断未决
---	------------	---	------	---------------------------

RFIRQF1 (0x91) —RF 中断标志

位	名称	复位	读/写	描述
7: 6	—	0	R0	保留。读为 0
5	CSP_WAIT	0	R/W0	CSP 的一条等待指令后继续执行。 0: 无中断未决 1: 中断未决
4	CSP_STOP	0	R/W0	CSP 程序运行停止 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	无线状态机进入空闲状态 0: 无中断未决 1: 中断未决
1	TXDONE	0	R/W0	发送了一个完整的帧 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	读/写	描述
7	—	0	R0	保留。读为 0
6	STROBEERR	0	R/W0	无法处理时发送一个命令选通。在已经禁止了无线而试图再次禁止就触发，不在有效 RX 状态下而试图执行 SACK、SACKPEN 或 SNACK 命令也会触发。 0: 无中断未决 1: 中断未决
5	TXUNDERF	0	R/W0	TXFIFO 下溢出 0: 无中断未决 1: 中断未决
4	TXOVERF	0	R/W0	TXFIFO 溢出 0: 无中断未决 1: 中断未决
3	RXUNDERF	0	R/W0	RXFIFO 上溢出 0: 无中断未决 1: 中断未决

2	RXOVERF	0	R/W0	RXFIFO 溢出 0: 无中断未决 1: 中断未决
1	RXABO	0	R/W0	中止一个帧接收 0: 无中断未决 1: 中断未决
0	NLOCK	0	R/W0	频率合成器在超时后不能完成锁定，或在接收期间锁定丢失。 0: 无中断未决 1: 中断未决

RFIRQM0 (0x61A3) —RF 中断使能

位	名称	复位	读/写	描述
7	RXMASKZERO	0	R/W	RXENABLE 寄存器从一个非 0 状态变为全 0 状态。 0: 中断禁止 1: 中断使能
6	RXPKTDONE	0	R/W	接收到一个完整的帧。 0: 中断禁止 1: 中断使能
5	FRAME_ACCEPTED	0	R/W	帧已经通过了帧过滤。 0: 中断禁止 1: 中断使能
4	SRC_MATCH_FOUND	0	R/W	发现源匹配。 0: 中断禁止 1: 中断使能
3	SRC_MATCH_DONE	0	R/W	源匹配完成。 0: 中断禁止 1: 中断使能
2	FIFOP	0	R/W	RX FIFO 中的字节数已超出设置的阈值。当收到一个完整的帧时仍然增加。 0: 中断禁止 1: 中断使能
1	SFD	0	R/W	收到或发送 SFD。 0: 中断禁止 1: 中断使能
0	ACT_UNUSED	0	R/W	保留 0: 中断禁止 1: 中断使能

RFIRQM1 (0x61A4) —RF 中断使能

位	名称	复位	读/写	描述
7: 6	—	0	R0	保留。读为 0
5	CSP_WAIT	0	R/W	CSP 的一条等待指令后继续执行。

				0: 中断禁止 1: 中断使能
4	CSP_STOP	0	R/W	CSP 程序运行停止 0: 中断禁止 1: 中断使能
3	CSP_MANINT	0	R/W	来自 CSP 的手动中断产生 0: 中断禁止 1: 中断使能
2	RFIDLE	0	R/W	无线状态机进入空闲状态 0: 中断禁止 1: 中断使能
1	TXDONE	0	R/W	发送了一个完整的帧 0: 中断禁止 1: 中断使能
0	TXACKDONE	0	R/W	完整发送了一个确认帧 0: 中断禁止 1: 中断使能

RFERRM (0x61A5) —RF 错误中断使能

位	名称	复位	读/写	描述
7	—	0	R0	保留。读为 0
6	STROBEERR	0	R/W	无法处理时发送一个命令选通。在已经禁止了无线而试图再次禁止就触发，不在有效 RX 状态下而试图执行 SACK、SACKPEN 或 SNACK 命令也会触发。 0: 中断禁止 1: 中断使能
5	TXUNDERF	0	R/W	TXFIFO 下溢出 0: 中断禁止 1: 中断使能
4	TXOVERF	0	R/W	TXFIFO 溢出 0: 中断禁止 1: 中断使能
3	RXUNDERF	0	R/W	RXFIFO 上溢出 0: 中断禁止 1: 中断使能
2	RXOVERF	0	R/W	RXFIFO 溢出 0: 中断禁止 1: 中断使能
1	RXABO	0	R/W0	中止一个帧接收 0: 中断禁止 1: 中断使能
0	NLOCK	0	R/W0	频率合成器在超时后不能完成锁定，或在接收期间锁定丢失。

				0: 中断禁止 1: 中断使能
--	--	--	--	--------------------

19.2 FIFO 存取

TX FIFO 和 RX FIFO 可以通过 SFR 寄存器 RFD (0xD9) 进行存取。数据写寄存器 RFD 就是写 TX FIFO，读寄存器 RFD 就是从 RX FIFO 读取数据。

XREG 寄存器 RXFIFOCNT 和 TXFIFOCNT 提供了 FIFO 里的数据量的信息。可以通过下达 SFLUSHRX 和 SFLUSHTX 命令清除 FIFO 内容。

RFD (0xD9) —RF 数据

位	名称	复位	读/写	描述
7: 0	RFD[7: 0]	0x00	R/W	数据写入寄存器就是写入 TX FIFO。读该寄存器，就是从 RX FIFO 中读取数据。

19.3 DMA

使用直接存储器存取 (DMA) 在存储器和无线模块之间传送数据。第 8 章中已介绍过如何安装和使用 DMA 传送。

RADIO DMA 触发 (DMA 触发 19) 与无线模块有关，该触发支持 DMA 控制器。两个事件使该触发有效。引起一个 RADIO DMA 触发的第一个事件是当第一个数据存入 RX FIFO，即当 RX FIFO 从空状态变成非空状态时。引起一个 RADIO DMA 触发的第二个事件是当数据通过 SFR 寄存器 RFD，从 RX FIFO 中读出时，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，因此是 128 字节。虽然这个存储器区域用于 RX FIFO，而不以任何方式加以保护，所以在 XREG 存储器空间它仍然是可以被访问的。一般来说，只有指定的指令用于操作 RX FIFO 里的内容。RX FIFO 一次包含多个帧。

19.4.2 TX FIFO

TX FIFO 存储器区域位于地址 0x6080 到 0x60FF，因此是 128 字节。虽然这个存储器区域用于 TX FIFO，而不以任何方式加以保护，所以在 XREG 存储器空间它仍然是可以被访问的。一般来说，只有指定的指令用于操作 TX FIFO 里的内容。RX FIFO 一次只能包含一个帧。

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匹配的配置和结果；它位于区域 0x6100 到 0x617F。表 19-1 描述了此存储器空间。没有填满整个字节/字的值位于字节/字的最低有效部分。注意，这些寄存器中的值在复位后是未知的。但是，在功耗模式期间这些值是被保留的。

表 19-1 帧过滤和源匹配存储器映射

地址	寄存器/变量	字节存储次序	描述
保留			
0x6176-6x617F	临时存储		用于临时存储变量的存储器空间
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 位掩码的 8 个最高位，为每个 24 位短地址使能/禁止自动未决
0x6168	SRCSHORTPENDEN1		
0x6167	SRCSHORTPENDEN0		24 位掩码的 8 个最低位，为每个 24 位短地址使能/禁止自动未决
0x6166	SRCEXTPENDEN2		24 位掩码的 8 个最高位，为每个 12 位扩展地址使能/禁止自动未决。入口 n 映射到 SRCEXTPENDEN[2n]。所有的 SRCEXTPENDEN[2n+1] 位都不重要。
0x6165	SRCEXTPENDEN1		
0x6164	SRCEXTPENDEN0		24 位掩码的 8 个最低位，为每个 12 位扩展地址使能/禁止自动未决。条目 n 映射到 SRCEXTPENDEN[2n]。所有的 SRCEXTPENDEN[2n+1] 位都不重要。
短地址匹配结果			
0x6163	SRCRESINDEX		SRCRESMASK 里最低位 1 的位索引，或当没有源匹配时是 0x3F。如果是短地址匹配，位 5 为 0，如果是扩展地址匹配，位 5 为 1。匹配时，如果确认的自动未决位的条件符合（见 SRCMATCH.AUTOOPEN 的描述），位 6 为 1。该位并不指示实际上是否发送了确认，并且不考虑寄存器位 PENDING_OR 和选通命令 SACK/SACKPEND/SNACK。
0x6162	SRCRESMASK2		24 位掩码，表示源地址表中每个独立条目的源地址匹配。
0x6161	SRCRESMASK1		短地址匹配。当条目 panid_n+short_n 上有一个匹配时，设置 SRCRESMASK 的位 n。

0x6160	SRCRESMASK0		扩展地址匹配。当条目 ext_n 上有一个匹配时，设置 SRCRESMASK 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 频率和信道编程

通过对位于 FREQCTRL.FREQ[6: 0]的 7 位频率字编程设置载波频率。支持从 2394MHz 到 2507MHz 的载波频率。以 MHz 为单位的载波频率 f_c 由下式表示：

$$f_c = (2394 + FREQCTRL.FREQ[6:0]) \text{MHz}$$

以 1MHz 为步长进行编程。

IEEE 802.15.4-2006 指定 16 个信道，它们位于 2.4GHz 频段之内，步长为 5MHz，编号为 11~26。信道 k 的 RF 频率由 IEEE 802.15.4[1]指定如下：

$$f_c = 2405 + 5(k - 11) [\text{MHz}] \quad k \in [11, 26] \quad (19-1)$$

运行在信道 k，寄存器 FREQCTRL.FREQ 应当设置为：

$$FREQCTRL.FREQ = 11 + 5(k - 11)$$

19.6 IEEE 802.15.4-2006 调制格式

本节的目的在于介绍 IEEE 802.15.4-2006 定义的 2.4GHz 直接序列扩频（DSSS）RF 调制格式。完整描述请参阅[1]。

图 19-1 在模块层次上对调制和扩展功能进行了描述。每个字节分为两组符号，4 位一组，低位符号首先传送。对于多字节域，则是低位字节首先传送，除了与安全相关的域是高位字节首先传送。

每个符号映射到一个超过 16 位的伪随机序列，即 32 位码片序列。符号到码片的映射请见表 19-2。码片序列以 2Mchip/s 的速率传送，对于每个符号，首先传送低位码片 (C_0)。传

送的比特流和码片序列可以在 GPIO 引脚上观察到，关于如何配置 GPIO 来完成这一功能详见第 7 章。

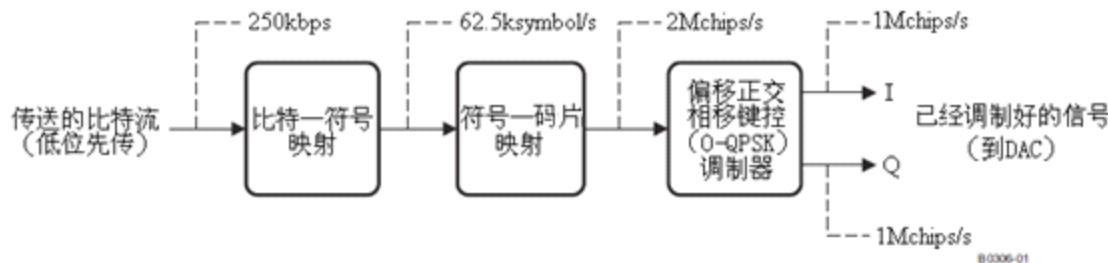
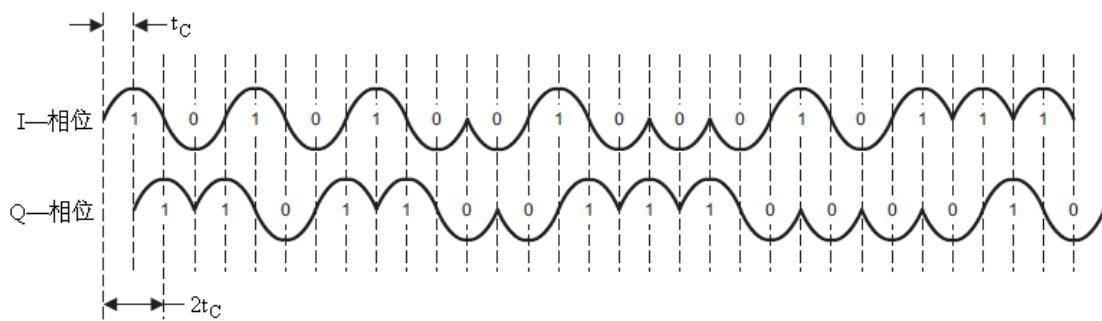


图 19-1 调制

表 19-1 IEEE 802.15.4-2006 符号—码片序列映射

符号	码片序列 ($C_0, C_1, C_2, \dots, C_{31}$)
0	11011001110000110101001000101110
1	11101101100111000011010100100010
2	00101110110110011100001101010010
3	0010001011101101100111000011010101
4	010100100010111011011001110000110101
5	0011010100100010111011011001110000
6	11000011010100100010111011011001
7	10011100001101010010001011101101
8	1000110010010110000011101111011
9	1011100011001001011000001110111
10	01111011100011001001011000000111
11	0111011110111000110010010110000
12	0000011011110111000110010010110
13	0110000001101111011100011001001
14	1001011000000110111101110001100
15	11001001011000000111011110111000

调制格式为偏移正交相移键控（O-QPSK），具有半正弦片的形状，相当于最小相位频移键控（MSK）。每片的形状如同半个正弦波，交替在同相（I）信道和正交相位（Q）信道传送。每个信道占用一个半码片偏移周期，参见图 19-2 的 0 符号。

图 19-2 传送符号 0 码片序列时的 I/Q 相位 ($t_c=0.5\mu s$)

19.7 IEEE802.15.4-2006 帧格式

本小节简要介绍 IEEE 802.15.4 帧格式[1]。内置的无线模块支持部分帧的处理。下面个小节将对此进行描述。

图 19-3 为 IEEE 802.15.4 帧格式的示意图。[1]中包括了具体帧格式（数据帧、信标帧、确认帧和 MAC 命令帧）的类似示意图。

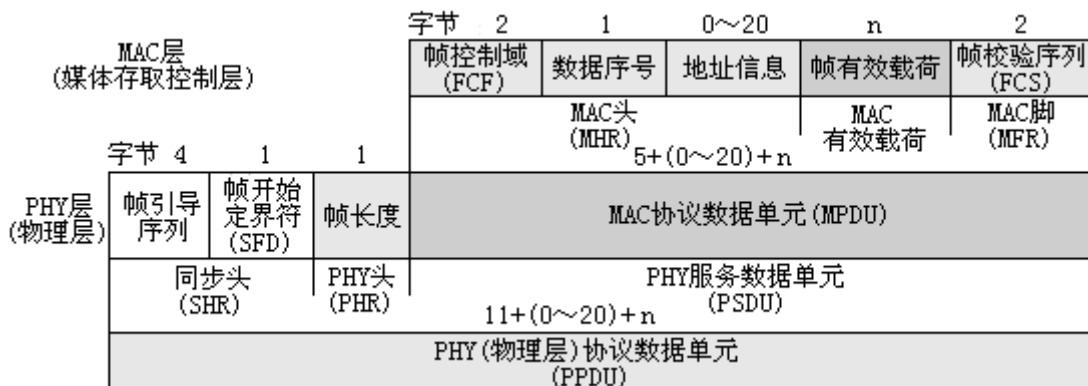


图 19-3 IEEE 802.15.4 帧格式示意图[1]

19.7.1 物理层

同步头

同步头 (SHR) 包含帧引导序列和帧开始定界符 (SFD)。IEEE 802.15.4 协议[1]中，定义帧引导序列为 4 个字节 0x00。SFD 是 1 个字节，设置为 0xA7。

PHY 头

PHY 头只包含了帧长度域。帧长度域定义了 MAC 协议数据单元 (MPDU) 中的字节数。注意，该字节数不包含长度域本身的字节，但却包含了帧校验序列 (FCS) 的字节，即使 FCS 是由 CC253x 自动插入的。

帧长度域有 7 位，最大值是 127。长度域的最高位保留，而且总是设置为 0。

PHY 服务数据单元

PHY 服务数据单元包含 MAC 协议数据单元 (MPDU)。MAC 层负责生成/解释 MPDU，内置的无线模块支持部分 MPDU 子域的处理。

19.7.2 MAC 层

如图 19-3 所示，帧控制域 (FCF)、数据序号和地址信息紧随长度域值后。加上 MAC 数据有效载荷和帧校验序列，组成了 MAC 协议数据单元 (MPDU)。FCF 的格式图 19-4 所示。详细细节请参见[1]IEEE 802.15.4 协议。

位: 0~2	3	4	5	6	7~9	10~11	12~13	14~15
帧类型	安全使能	帧未决	确认请求	内部 PAN	保留	目标地址模式	保留	源地址模式

图 19-4 帧控制域 (FCF) 格式

帧校验序列

如图 19-3 所示，帧校验序列（FCS）占两个字节，紧随最后一个 MAC 有效载荷字节之后。FCS 是通过 MAC 协议数据单元（MPDU）计算出来的，也就是说，计算 FCS 不包括长度域。

IEEE 802.15.4 协议[1]定义的 FCS 的表达式是：

$$G(s) = x^{16} + x^{12} + x^5 + 1$$

无线模块支持自动计算/校验 FCS。详细内容请参见 19.8.10 节。

19.8 发送模式

本节描述如何控制发射机，完整的帧处理和如何使用 TX FIFO。

19.8.1 TX 控制

无线模块有许多内置功能，用于帧处理的状态报告。注意，无线模块提供的功能，使得很容易实现对输出帧的时序进行精确控制。这在一个 IEEE 802.15.4/ZigBee® 系统中是非常重要的，因为这类系统对时序要求非常严格。

通过以下操作开始帧传送：

- STXON 命令选通
 - SAMPLED_CCA 信号不更新。
- STXONCCA 命令选通，只要 CCA 信号为高。
 - 中止正在进行的传送/接收，并且强制一个 TX 校准后进行传送。
 - SAMPLED_CCA 信号更新。

空闲信道评估在 19.8.12 小节进行详细描述。

通过以下命令操作中止帧传送：

- SRXON 命令选通
 - 中止正在进行的传送，并且强制一个 RX 校准。
- SRFOFF 命令选通
 - 中止正在进行的传送/接收，并且强制 FSM 到 IDLE 状态。
- STXON 命令选通
 - 中止正在进行的传送，并且强制一个 RX 校准。

STXON 传送后要使能接收器，还要将 FRMCTRL1.SET_RXENMASK_ON_TX 位置位。当执行 STXON 时，设置 RXENABLE 的位 6。当 STXONCCA 发送时，接收器在传送之前开启，然后返回（除非寄存器 RXENABLE 在此期间已经被清除）。

19.8.2 TX 状态时序

帧引导序列在 STXON 或 STXONCCA 命令选通后 192us 开始传输。这在[1]中被称为 TX 转向时间。返回到接收模式延迟时间也为 192us。

当返回到空闲或接收模式，在调制器下降信号到 DAC 时有一个 2us 的延迟。在完整的

MPDU (由长度字节定义) 被发送后或发生 TX 下溢出时自动进行下降动作。这会影响到:

- SFD 信号延长了 2us
- 无线 FSM 转换到 IDLE 状态, 延迟了 2us

19.8.3 TX FIFO 存取

TX FIFO 可以保存 128 字节, 一次只能有一个帧。只要不产生 TX 下溢出 (见 19.8.5 小节所列的错误情况), 可以在 TX 命令选通执行之前或执行之后对帧进行缓冲。

图 19-5 说明了必须写入 TX FIFO (以蓝色表示) 的字节。除非发生 TX 溢出 (见 19.8.5 小节所列的错误情况), 否则其它的字节被忽略。



M0 109-01

图 19-5 写入 TX FIFO 的帧数据

有两种方式写数据到 TX FIFO:

- 写入 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, 可以直接写入无线存储器的 RAM 区域, 该区域保存 TXFIFO。建议使用 RFD 来写数据到 TXFIFO。

TXFIFO 中的字节数存储器 TXFIFOCNT 寄存器中。

使用 SFLUSHTX 命令选通可以手动清空 TX FIFO。如果在传送期间清空 FIFO 会发生一个 TX 下溢出。

19.8.4 重传

为了支持简单的帧重传, 无线模块在 TX FIFO 传送期间不会删除 TX FIFO 的内容。成功传送一个帧后, 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, 如果数据包被完全传送才可以重传这个数据包; 也就是说, 如果一个数据包传送被中止了, 就不能再重传了。

如果要传送一个不同的帧, 就将新的帧写入 TX FIFO。在这种情况下, 在实际的写发生之前, TXFIFO 会自动清除。

19.8.5 错误情况

与 TX FIFO 相关的错误情况有 2 个:

- TX FIFO 已满而尝试写入另一个字节, 此时会发生溢出
 - TX FIFO 为空而无线模块试图取另一个字节用于传送, 此时会发生下溢出
- TX_OVERFLOW 中断标志置位表示 TX 溢出。发生这个错误时, 写入被中止, 即导致

溢出的数据字节丢失。必须使用 SFLUSHTX 命令选通来清除该错误情况。

TX_UNDERFLOW 中断标志置位表示 TX 下溢出。发生这个错误时，正在进行的传送被中止。不许使用 SFLUSHTX 命令选通来清除该错误情况。

通过设置 FRMSTRL1.IGNORE_TX_UNDERF 位来禁用 TX_UNDERFOW 异常。在这种情况下，无线模块继续传送 TX FIFO 存储器中的字节，直到由第一个字节给定的字节数（即长度字节）传送完成。

19.8.6 TX 流程图

图 19-6 将前面的章节总结为下面的流程图。

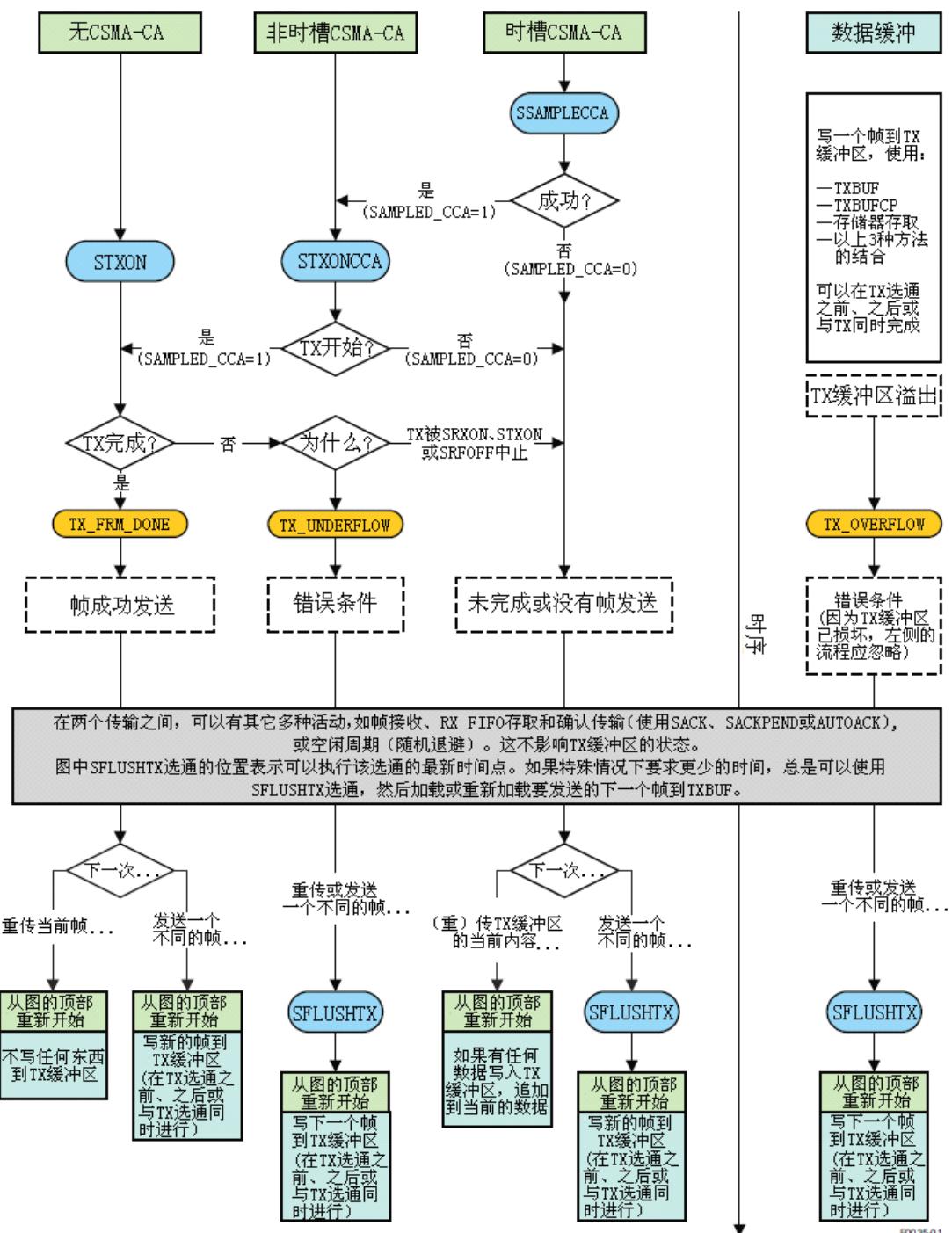


图 19.6 TX 流程图

19.8.7 帧处理

无线模块为 TX 帧执行下面的帧产生任务：



- (1) 物理层同步头的产生和自动传送，由帧引导序列和 SFD 组成。
- (2) 由帧长度域制定的字节数的传送。
- (3) FCS（可以被禁止）的计算和自动传送。

建议先写长度域，然后写 MAC 头和 MAC 有效载荷到 TX FIFO，其余部分由无线模块处理。注意长度域必须包括 2 个 FCS 字节，即使无线模块已经自动添加了这些字节。

19.8.8 同步头

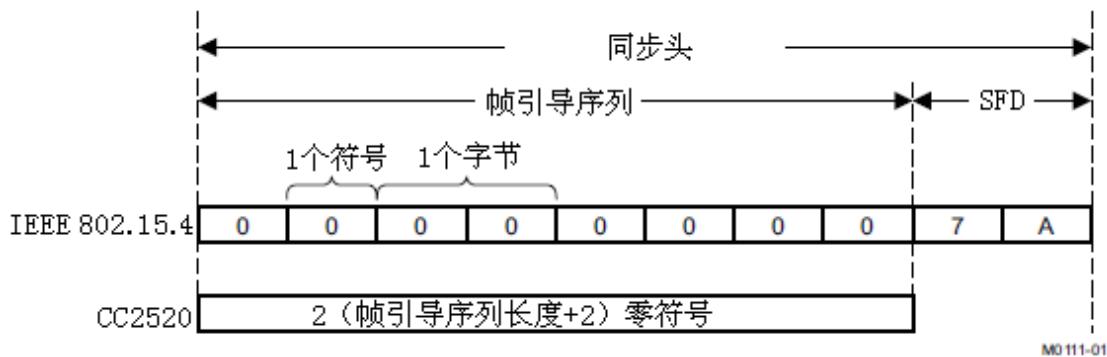


图 19-7 发送的同步头

无线模块的帧引导序列的长度是可以配置的。缺省值适应 IEEE 802.15.4 协议，不可随意改变，改变这些值将使系统不适应 IEEE 802.15.4。

帧引导序列的长度可以由寄存器位 MDMCTRL0.PREAMBLE_LENGTH 设置。图 19-7 描述了同步头与 IEEE 802.15.4 的关系。

当所需的帧引导序列字节数已经发送后，无线模块会自动发送 1 字节长的 SFD。SFD 是固定的，不能使用软件改变它。

19.8.9 帧长度域

发送了 SFD，调制器开始从 TX FIFO 读取数据。首先是帧长度域，然后是 MAC 头和 MAC 有效载荷。帧长度域决定要传送的字节数。

注意，当 AUTOCRC=1 时，最小帧长度为 3；当 AUTOCRC=0 时，最小帧长度为 1。

19.8.10 帧校验序列

当 FRMCTRL0.AUTOCRC 控制位为 1 时，FCS 域自动生成并填充到由长度域定义的传送帧的位置。FCS 不写入 TXFIFO，而是存储在一个单独的 16 位寄存器中。因此，除了在调试的情况下，建议始终使能 AUTOCRC。如果 FRMCTRL0.AUTOCRC=0，调制器则希望

在 TX FIFO 中找到到 FCS，所以软件必须生成 FCS，并且将它连同 MPDU 的其余部分写入 TX FIFO。

硬件执行 FCS 计算如图 19-8 所示。详细信息请见 IEEE 802.15.4。

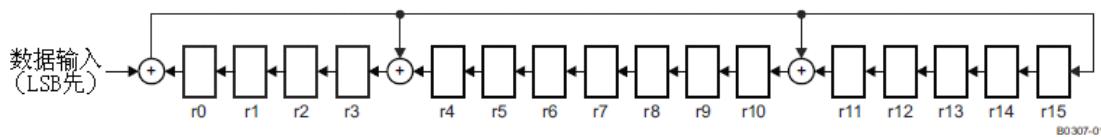


图 19-8 硬件执行帧校验序列 (FCS)

19.8.11 中断

一旦传送了帧的 SFD 域就产生 SFD 中断。在帧的末尾，如果成功传送了一个完整的帧，就产生 TX_FRM_DONE 中断。

注意，在 GPIO 上还有第二个 SFD 信号可用（通过无线观察多路器），这个 SFD 信号不能和 SFD 中断相混淆。

19.8.12 空闲信道评估

空闲信道评估 (CCA) 状态信号指示信道是否可用于传输。空闲信道评估功能用来实现 IEEE 802.15.4[1]指定的 CSMA-CA 功能。至少在 8 个符号周期后，接收器使能，CCA 有效。可以用 RSSI_VALID 状态信号来进行验证。

CCA 是基于 RSSI 值的测量和可编程设置的阈值。可以在寄存器 CCACTRL0 和 CCACTRL1 中配置其确切的行为。

有 2 种 CCA 信号变化，一种是在每个新的 RSSI 采样时进行更新，一种是只在 SSAMPLECCA/ISAMPLECCA 和 STXONCCA/ISTXONCCA 命令选通时更新。在 FSMSTAT1 寄存中，这 2 种都是可用的。

注意，在 RSSI_VALID 信号置位后 4 个时钟周期（系统时钟）进行 CCA 信号更新。

19.8.13 输出功率编程

设备的 RF 输出功率由寄存器 TXPOWER 的 7 位值控制。CC2530 数据手册给出了当中心频率设置为 2.440GHz 时，推荐设置的典型输出功率和电流消耗。注意，推荐设置只是所有可能的寄存器设置的一个小的子集。

19.8.14 提示和技巧

- 注意，在开始传送之前，TX FIFO 里不需要有完整的帧。字节可以在传输期间添加到 TX FIFO。
- 设置 MDMTSET1.MODULATION_MODE=1，就可以传送非 IEEE 802.15.4 标准的帧。

19.9 接收模式

本节描述如何控制接收机，完整的 RX 帧处理和如何使用 RX FIFO。

19.9.1 RX 控制

通过 SRXON 和 SRFOFF 命令选通或者 RXENABLE 寄存器来开启和关闭接收机。命令选通提供了一个硬开启/关闭机制，而 RXENABLE 操作提供了一个软开启/关闭机制。

通过以下操作开启接收机：

- SRXON 选通：
 - 设置 RXENABLE[7]
 - 强制转换到 RX 校准，中止正在进行的发送/接收。
- 当 FRMCTRL1.SET_RXENMASK_ON_TX 使能时，STXON 选通：
 - 设置 RXENABLE[6]
 - 发送完成后接收机使能
- 通过写 RXENMASKOR 来设置 RXENABLE!=0x00：
 - 不中止正在进行的发送/接收。

通过以下操作关闭接收机：

- SRFOFF 选通：
 - 清除 RXENABLE[7: 0]
 - 强制转换到 IDLE 模式，中止正在进行的发送/接收。
- 通过写 RXENMASKAND 来设置 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，无线模块返回到 IDLE 状态。

有两种方法来操作 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

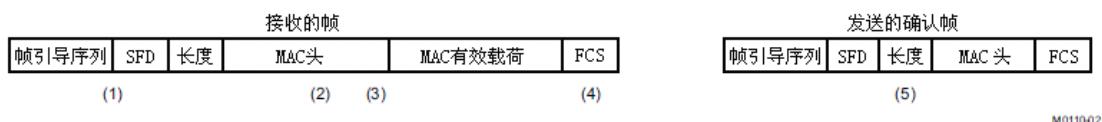
接收机在采用 19.9.1 小节中的任意一种方法来使能 RX 后 192us 准备好。这在[1]中被称为 RX 转向时间。

在帧接收后返回到接收模式，到 SFD 检测禁止，有一个 192us 的默认间隔。通过清除 FSMCTRL.RX2RX_TIME_OFF 来禁用这个间隔。

19.9.3 帧处理

无线模块集成了在 IEEE 802.15.4-2003 和-2006 中硬件要求的关于 RX 的关键部分。这减少了 CPU 的干预率，简化了软件对帧接收的处理，并且以最小的延迟给出结果。

接收单个帧期间，依次进行下面的帧处理：



- (1) 检测和移除接收到的物理层同步头（帧引导序列和 SFD），接收由帧长度域指定的字节数。
- (2) 进行帧过滤，由 IEEE 802.15.4 里 7.5.6.2 小节第三过滤级别所规定。
- (3) 匹配源地址和地址表，该地址表包含多大 24 个短地址，或 12 个扩展 IEEE 地址。元地址表存储在无线 RAM 中。
- (4) 自动 FCS 检查，并把检查结果和其它状态值（RSSI、LQI 和源匹配结果）一起添加到接收到的帧中。
- (5) 自动发送确认帧，该帧具有正确时序，帧未决位设置正确，基于来自源地址匹配的结果和 FCS 校验。

19.9.4 同步头和帧长度域

帧接收开始于帧开始定界符（SFD）检测，然后是决定接收何时完成的长度字节。SFD 信号可以在 GPIO 上输出，可以用于捕获接收帧的开始：

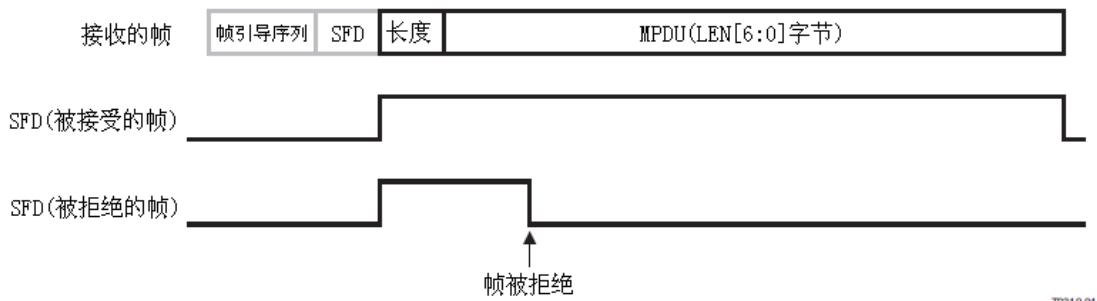


图 19-9 SFD 信号时序

帧引导序列和 SFD 不写入 RX FIFO。

无线模块使用一个相关器来检测 SFD。MDMCTRL1.CORR_THR 里的相关阈值决定接收到的 SFD 和理想的 SFD 的匹配程度如何。阈值必须小心调整：

- 如果设置得太高，无线模块会错过许多实际的 SFD，大大降低了接收机的灵敏度。
- 如果设置得太低，无线模块会检测到许多错误 SFD。虽然这不会降低接收机的灵敏度，但是效果是类似的，因为错误的帧可能会重叠实际帧的 SFD。它还会增加接收到具有正确 FCS 的错误帧的风险。

除了 SFD 检测，在 SFD 检测之前可能还需要若干有效的帧引导序列符号（也高于相关阈值）。可用选项和建议设置请见 MDMCTRL0 和 MDMCTRL1 的寄存器描述。

19.9.5 帧过滤

按照 IEEE 802.15.4,7.5.6.2 小节，第三过滤级别所规定的，帧过滤功能拒绝目标不明确的帧。此外，它对下面的内容提供过滤：

- 8 种不同的帧类型（见 FRMFILT1 寄存器）
- 帧控制域（FCF）的保留位

帧过滤功能通过下面的寄存器进行控制：

- FRMFILT1 和 FRMFILT1 寄存器
- RAM 里 LOCAL_PAN_ID、LOCAL_SHORT_ADDR 和 LOCAL_EXT_ADDR 的值

过滤算法

FRMFILT0.FRAME_FILTER_EN 位控制帧过滤是否适用。当该位禁止时，无线模块接受所有接收的帧。当该位使能时（默认设置），无线模块只接受满足以下所有要求的帧：

- 长度字节必须等于或大于最小帧长度，它从源地址模式和目的地址模式，以及 FCF 的 PAN ID 压缩子域获得。
- 保留的 FCF 位[9: 7]与 FRMFILT0.FCF_RESERVED_BITMASK 相“与”，结果必须等于 000b。
- FCF 帧版本子域的值不能高于 FRMFILT0.MAX_FRAME_VERSION。
- 源地址模式和目的地址模式不能是保留值（1）。
- 目标地址：
 - 如果目标地址是包含在帧中，那么它必须和 LOCAL_PANID 匹配或者必须为广播 PAN 标识符（0xFFFF）。
 - 如果短目标地址包含在帧中，那么它必须和 LOCAL_SHORT_ADDR 或广播地址（0xFFFF）匹配。
 - 如果扩展目标地址包含在帧中，那么它必须和 LOCAL_EXT_ADDR 匹配。
- 帧类型：
 - 只接受信标帧（0），当：
 - FRMFILT1.ACCEPT_FT0_BEACON=1
 - 长度字节 ≥ 9
 - 目标地址模式为 0（无目标地址）
 - 源地址模式为 2 或 3（即包括一个源地址）
 - 源 PAN ID 与 LOCAL_PANID 匹配，或 LOCAL_PANID 等于 0xFFFF。
 - 只接受数据（1）帧，当：
 - FRMFILT1.ACCEPT_FT1_DATA=1
 - 长度字节 ≥ 9
 - 帧中包括一个目标地址和/或源地址。如果帧中不包括目标地址，FRMFILT0.PAN_COORDINATOR 位必须置 1，而源 PAN ID 必须等于 LOCAL_PANID。
 - 只接受确认（2）帧，当：
 - FRMFILT1.ACCEPT_FT2_ACK=1
 - 长度字节=5
 - 只接受 MAC 命令（3）帧，当：
 - FRMFILT1.ACCEPT_FT3_MAC_CMD=1
 - 长度字节 ≥ 9
 - 帧中包括一个目标地址和/或源地址。如果帧中不包括目标地址，为了该帧能被接受，FRMFILT0.PAN_COORDINATOR 位必须置 1，而源 PAN ID 必须等于 LOCAL_PANID。
 - 只接受保留的帧类型（4、5、6 和 7），当：

- FRMFILT1.ACCEPT_FT4TO7_RESERVED=1 (默认为 0)
- 长度字节 \geqslant 9

在帧过滤开始前，必须执行下面的操作，它不会影响存储在 RX FIFO 里的帧数据：

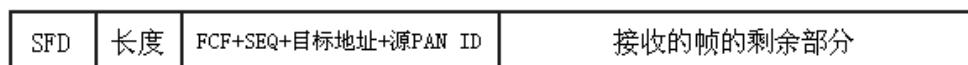
- 长度字节的位 7 被屏蔽（没关系）
- 如果 FRMFILT1.MODIFY_FT_FILTER 不是 0，FCF 帧类型子域的最高有效位应该反相或强制变为 0 或者 1。

如果一个帧被拒绝了，在这个被拒绝的帧被完全接收后（由长度域定义），无线模块仅搜寻一个新的帧，以此避免在这个被拒绝的帧里检测到错误的 SFD。注意，在帧被拒绝之前就发生了 RX 溢出，那么在这个帧被拒绝之后仍然会产生 RX 溢出。

中断

如果使能了帧过滤，并且过滤算法接受了一个接收的帧，就产生一个 RX_FRM_ACCEPTED 中断。如果禁用了帧过滤，或者在知道过滤结果之前，产生了 RX_OVERFLOW 或 RX_FRM_ABORTED，那么就不会产生 RX_FRM_ACCEPTED 中断。

图 19-10 说明了三种不同的情况（不包括移除和中止错误情况）。



帧过滤使能，帧被拒绝



帧过滤使能，帧被接受

在此期间发生FIFOP中断
(取决于FIFOPCTRL的值)



帧过滤禁止

在此期间发生FIFOP中断
(取决于FIFOPCTRL的值)



M0112-01

图 19-10 过滤情况（接收期间产生异常）

一个帧开始定界符被完全接收后，寄存器位 FSMSTAT1.SFD 变为 1，并且保持为 1 直到接收到 MPDU 的最后一个字节，或者接收的帧没有通过地址识别并且被拒绝。

提示和技巧

必须正确配置下面的寄存器设置：

- 如果设备是一个 PAN 协调器，FRMFILT0.PAN_COORDINATOR 必须置为 1，如果设备不是协调器，则置为 0。
- 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。这就禁止了符号搜索，从而防止 SFD 检测。

要重新开始正常 RX 模式，设置 FRMCTRL0.RX_MODE=00b，然后（重新）启动 RX。

在运行于繁忙的 IEEE 802.15.4 环境中期间，无线模块接收大量的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请求的发送之后，FRMFILT1.ACCEPT_FT2_ACK 置位，接收到确认帧后或等待确认帧超时后，清除 FRMFILT1.ACCEPT_FT2_ACK。
- 否则，FRMFILT1.ACCEPT_FT2_ACK 保持为 0。

改变寄存器 FRMFIT0/1 的值和存储在 RAM 的本地地址信息的时候，不必关闭接收机。但是，如果在接收 SFD 字节和源 PAN ID 之间（即在 SFD 和 RX_FRM_ACCEPTED 异常之间）发生改变，修改后的值必须被视为不关心特定帧（无线模块使用旧值或新值）。

注意，设置 MDMTEST1.MODULATION_MODE=1 可以让无线模块忽略所有的 IEEE 802.15.4 输入帧。

19.9.6 源地址匹配

无线模块支持将接收到的帧里的源地址和存储在片上存储器里的地址表进行匹配。这个地址表长 96 字节，因此它可以包含多达：

- 24 个短地址（每个 2+2 字节）
- 12 个 IEEE 扩展地址（每个 8 字节）

只有使能了帧过滤并且接收到的帧已经被接受时才进行源地址匹配。该功能由它们进行控制：

- 寄存器 SRCMATCH、SRCSHORTEN0、SRCSHORTEN1、SRCSHORTEN2、SRCEXTEN0、SRCEXTEN1 和 SRCEXTEN2
- RAM 里的源地址表

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。为了表明协调器实际上是否为设备存储了一个帧，协调器必须在返回给设备的确认帧里设置或者清除帧未决位。但是在大多数 8 位和 16 位 MCU 上，没有足够的时间来确定这一点，因此不管是否有给设备的未决帧，协调器最终都会设置未决位（如 IEEE 802.15.4 所要求）。这在功耗方面是一种浪费，因为即使没有给它的帧，轮询设备也必须在相当长的一段时间里保持它的接收机使能。通过将间接帧队列里的目标地址加载到源地址表中，并且使能 AUTOPEND 功能，无线模块就会自动设置输出确认帧里的未决位。这样，操作不再以时序为关键，因为微控制器所做的工作就是何时添加或移除间接帧队列的帧，并更新相应的源地址表。

安全材料查找：为了减少处理安全帧所需的时间，可以设置源地址表的条目以匹配 CPU

上的安全密钥表。条目表上第二级别的屏蔽允许该应用与确认帧里未决位的自动设置相结合。

其它应用：前面两个应用是源地址匹配功能的主要目标。但是，对于只用到基本 IEEE 802.15.4 帧格式的专有协议，还有其它一些有用的应用。例如，在只需要确认一组制定的节点的应用中，可以创建防火墙功能。

源地址表

源地址表开始于 RAM 的地址 0x6100。该空间由短地址和扩展地址共享，寄存器 SRCSHORTEN0/1/2 和 SRCEXTEN0/1/2 用来控制使能哪些条目。表中的所有值都是以小端（如在接收帧）标志。

- 一个短地址开始于 16 位 PAN ID，后面是 16 位短地址。这些条目存储在地址 0x6100+
(4×n)，n 为一个 0 到 23 之间的数字。
- 一个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)，n 为一个 0 到 11 之间的数字。

地址使能寄存器

软件负责分配表条目，并确保有效的短地址和扩展地址条目不重叠。短地址和扩展地址有单独的使能位：

- 短地址条目在寄存器 SRCSHORTEN0、SRCSHORTEN1 和 SRCSHORTEN2 中使能。寄存器位 n 对应短地址条目 n。
- 扩展地址条目在寄存器 SRCEXTEN0、SRCEXTEN1 和 SRCEXTEN2 中使能。在这种情况下，寄存器位 2n 对应扩展地址条目 n。当创建一个结合位向量（短地址和扩展地址使能位）时，这个映射可以很方便地找到未使用的条目。此外，读取时，寄存器位 2n+1 的值总是和寄存器位 2n 的值相同，因为一个扩展地址所占据的存储器和两个短地址条目所占据的存储器一样。

匹配算法

寄存器位 SRCMATCH.SRC_MATCH_EN 控制源地址匹配是否使能。当源地址匹配使能时（默认设置），并且一个帧通过了帧过滤算法，那么无线模块适用图 19-13 所列出的哪一种算法，取决于当前的源地址是哪种类型。

以 2 种不同的格式来报告结果：

- 一个 24 位的向量 SRCRESMASK，包含用于每个使能的并且匹配的短条目的一个 1，或者用于每个使能的并且匹配的扩展条目的两个 1（位映射和读取访问的地址使能寄存器相同）。
- 一个 7 位值 SRCRESINDEX：
 - 如果接收的帧里面没有源地址，或者接收到的源地址上没有匹配：
 - 位 6: 0: 0x3F
 - 如果接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: 如果匹配在一个短地址上，该位为 0；如果匹配在一个扩展地址上，该位为 1。
 - 位 6: AUTOPEN 功能的结果。

短源地址（模式 2） 接收的源 PAN ID 称为 srcPanid，接收的短地址称为 srcShort。	扩展源地址（模式 3） 接收的扩展地址称为 srcExt。
--	----------------------------------

<pre> 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; } } } } </pre>	<pre> SRCRESMASK=0x000000; SRCRESINDEX=0x3F; for (n = 0; n < 12; n++) { bitVector = 0x000003 << (2* n); if (SRCSHORTEN & bitVector) { if (SRCEXTEN & bitVector) { if (ext [n] == srcExt) { SRCRESMASK = bitVector; if (SRCRESINDEX == 0x3F) { SRCRESINDEX =n 0x20; } } } } } </pre>
--	---

图 19-11 短地址和扩展地址的匹配算法

一旦结果可用，就将 SRCRESMASK 和 SRCRESINDEX 写入 RF 内核存储器。

如果寄存器位 FRMCTRL0.AUTO_CRC 和 FRMCTRL0.APPEND_DATA_MODE 已经置位，SRCRESINDEX 还会被追加到接收的帧。然后该值替换 16 位状态字的 7 位 LQI 值。

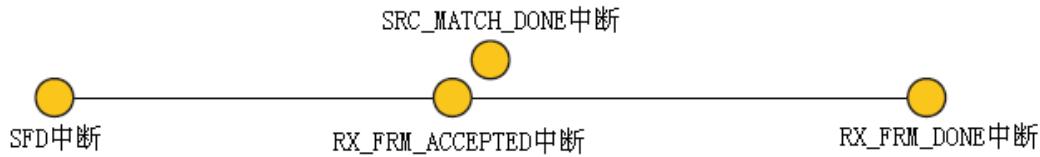
中断

如果已经使能了源地址匹配，并且匹配算法已经完成，不管结果如何，都将置位 SRC_MATCH_DONE 中断标志。如果找到一个匹配，在 SRC_MATCH_DONE 置位之前，还会立即置位 SRC_MATCH_FOUND 标志。

图 19-12 说明了设置标志的时序：

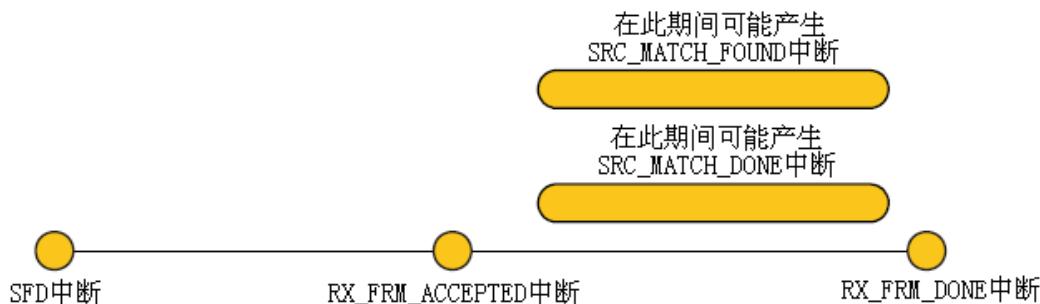
没有源地址:

SFD	长度	FCF+SEQ+目标地址	最后字节
-----	----	--------------	-------	------



有一个源地址:

SFD	长度	FCF+SEQ+目标地址+源PAN ID	源地址	最后字节
-----	----	----------------------	-----	-------	------



M0113-01

图 19-12 源地址匹配产生的中断

提示和技巧

- 在帧接收期间可以安全地修改源地址表。如果在接收机使能的时候，一个地址替换了另一个地址，那么在修改期间，应当关闭相应的使能位。这样就可以避免 RF 内核使用旧值和新值的结合值，因为它只考虑在整个源地址匹配过程中使能的条目。为了避免下一个接收的帧覆盖源地址匹配的结果，可以采取下列措施：
 - 使用追加的 SRCRESINDEX 结果，而不是用写入 RAM 的值（推荐使用这种方法）。
 - 在下一个接收的帧发生 RX_FRM_ACCEPTED 中断之前，从 RAM 读取结果。对于最短的帧类型，RX_FRM_ACCEPTED 中断发生在序列号之后，因此总的可用时间（安全系统最小的绝对最坏情况）为：
 $16\mu s$ (需要的帧引导序列) + $32\mu s$ (SFD) + $128\mu s$ (4 字节) = $176\mu s$
- 清除 FSMCTRL.RX2RX_TIME_OFF 位可以增加可用时间。这样就另外增加了 $192\mu s$ ，总共为 $368\mu s$ 。这还降低了 RX 溢出的风险。

19.9.7 帧校验序列

在接收模式下，如果使能了 FRMCTRL0.AUTOCRC，FCS 由硬件校核。用户通常只对 FCS 的正确性感兴趣，而不计较 FCS 本身的序列。因此，在接收期间，FCS 本身的序列不写入 RX FIFO 之中。反而，当 FRMCTRL0.AUTOCRC 置为 1 时，两个 FCS 字节被其它更有用的值所取代。可以在寄存器 FRMCTRL0 中设置取代 FCS 序列的值。

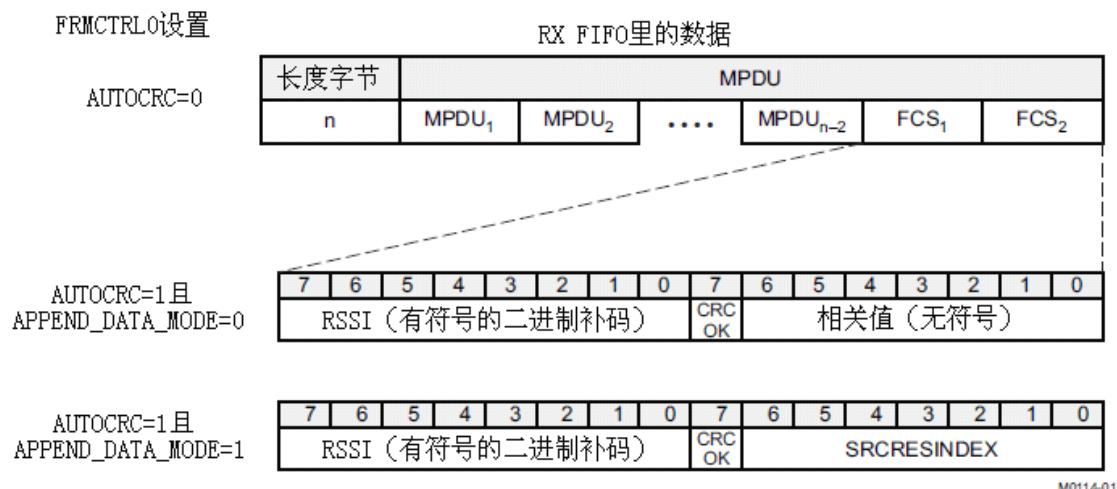


图 19-13 不同设置下 RX FIFO 里的数据

域描述：

- RSSI 值在 SFD 后的第一组 8 个符号进行测量。
- CRC_OK 位表示 FCS 是否正确（为 1 表示正确，为 0 表示不正确）。如果不正确，软件将丢弃该帧。
- 相关值是 SFD 后的第一组 8 个符号的平均相关值。
- SRCRESINDEX 和完成源地址匹配后写入 RAM 的值相同。

用于 IEEE 802.15.4 的 LQI 值的计算见 19.10.4 小节。

19.9.8 确认传输

无线模块的硬件支持在成功接收帧后（即接收帧的 FCS 必须正确），进行确认传输。图 19-14 为确认帧的格式。

字节:	4	1	1	2	1	2
帧引导序列	帧开始定界符 (SFD)	帧长度	帧控制域 (FCF)	数据序号	帧校验序列 (FCS)	
同步头 (SHR)	PHY头 (PHR)		MAC头 (MHR)		MAC脚 (MFR)	

图 19-14 确认帧格式

生成的确认帧中有 3 个可变域：

- 未决位，可以由命令选通和 AUTOPEND 功能控制
- 数据序号 (DSN)，从最后接收的帧中自动获取
- FCS，以隐含方式给出

用于设置一个 ACK 帧的未决位的源有 3 个，即 SACKPEND 选通、寄存器位 PENDING_OR 和 AUTOPEND 功能。如果 3 个源中的一个或多个被置位，就设置了未决位。

传输时序

只有在帧接收后才能立即进行确认帧的传输。传输时序由 FSMCTRL.SLOTTED_ACK 位控制。

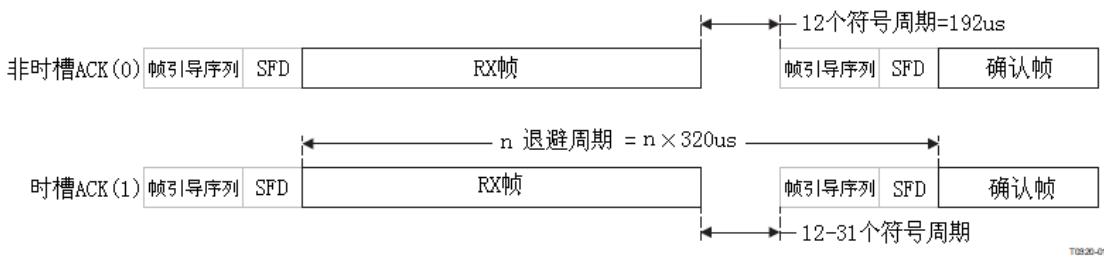


图 19-15 确认时序

对于使用非信标的 PAN, 802.15.4 要求使用非时槽模式, 对于使用信标的 PAN, 要求使用时槽模式。

手动控制

只有在帧接收期间才能发送 SACK、SACKPEND 和 SNACK 命令选通。如果在其它时间发送了这些命令选通, 它们不起作用, 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, 可以多次发送命令选通; 但是, 只有最后一次选通才有效:

- 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: 帧未决位置位的确认传输

自动控制 (AUTOACK)

当 FRMFILT0.FRAME_FILTER_EN 和 FRMCTRL0.AUTOACK 都使能时, 无线模块自动决定是否传输确认帧:

- RX 帧经过帧过滤后必须被接受 (由 RX_FRAME_ACCEPTED 异常表示)
- RX 帧里的确认请求位必须置位
- RX 帧决不能是一个信标帧或一个确认帧
- RX 帧的 FCS 必须正确

自动确认可以被 SACK、SACKPEND 和 SNACK 命令选通覆盖。例如, 如果微控制器的存储器资源很低, 而不能存储一个接收的帧, 可以在接收期间发送 SNACK 选通, 避免确认丢弃的帧。

默认情况下, AUTOACK 功能从不设置确认帧里的帧未决位。除了使用命令选通手动覆盖, 还有 2 种选择:

- 使用 AUTOPEND 功能自动控制
- 使用 FRMCTRL1.PENDING_OR 位手动控制

自动设置帧未决域 (AUTOPEND)

如果设置了 SRCMATCH.AUTOPEND 位, 源地址匹配的结果决定帧未决域的值。接收一个帧时, (可能) 返回的确认帧里的帧未决域置位, 会对下列内容进行设置:

- 设置了 FRMFILT0.FRAME_FILTER_EN。
- 设置了 SRCMATCH.SRC_MATCH_EN。

- 设置了 SRCMATCH_AUTOOPENED。
- 接收的帧与当前 SRCMATCH.PEND_DATAREQ_ONLY 设置匹配。
- 接收的源地址至少与源匹配表里的一个条目匹配，在 SRCSHORTEN 和 SRCSHORPENDEN 里都使能了，或者在 SRCEXTEN 和 SRCEXTPENDEN 里都使能了。

如果源匹配表满了，可以使用 FRMCTRL1.PENDING_OR 位来覆盖 AUTOOPENED 功能，并且临时确认所有带有帧未决域设置的帧。

19.10 RX FIFO 存取

RX FIFO 可以保存一个或多个接收的帧，只要总字节数为 128 或少于 128。有 2 种方式决定 RX FIFO 中的字节数：

- 读寄存器 RXFIFOCNT
- 结合 FIFOPCTRL.FIFOPTHR 设置，使用 FIFOP 和 FIFO 信号

通过 RFD 寄存器存取 RX FIFO。

也可以通过直接访问无线 RAM 来访问 RX FIFO 里的数据。在 RXFIRST_PTR、RXLAST_PTR 和 RXP1_PTR 中，FIFO 指针是可读的。如果想快速访问一个帧里的某个字节，而又不想首先读出整个帧，FIFO 指针很有用的。注意，当使用这种直接存取方法的时候，FIFO 指针不被更新。

ISFLUSHRX 命令选通会复位 RX FIFO，重新设置所有的 FIFO 指针，并清除所有的计数器、状态信号和粘着的错误情况。

SFLUSHRX 命令选通会复位 RX FIFO，移除所有接收的帧，并清除所有计数器、状态信号和粘着的错误情况。

19.10.1 使用 FIFO 和 FIFOP

在接收到帧的时候，如果要读出接收的帧的一小部分，FIFO 和 FIFOP 信号是很有用的：

- 如果 RX FIFO 中有一个或多个字节，FSMSTAT1.FIFO 变为高，如果发生了 RX 溢出，FSMSTAT1.FIFO 变为低。
- FSMSTAT1.FIFOP 信号变为高，当：
 - RX FIFO 里的有效字节数超过了设置在 FIFOPCTRL 里的 FIFOP 阈值。如果使能了帧过滤，帧头的字节不被视为有效，直到这个帧被接受。
 - 即使超过了 FIFOP 阈值，一个新的帧的最后一个字节也被接收。如果是这样，在下一个 FIFO 读取访问时，FIFOP 变回低。

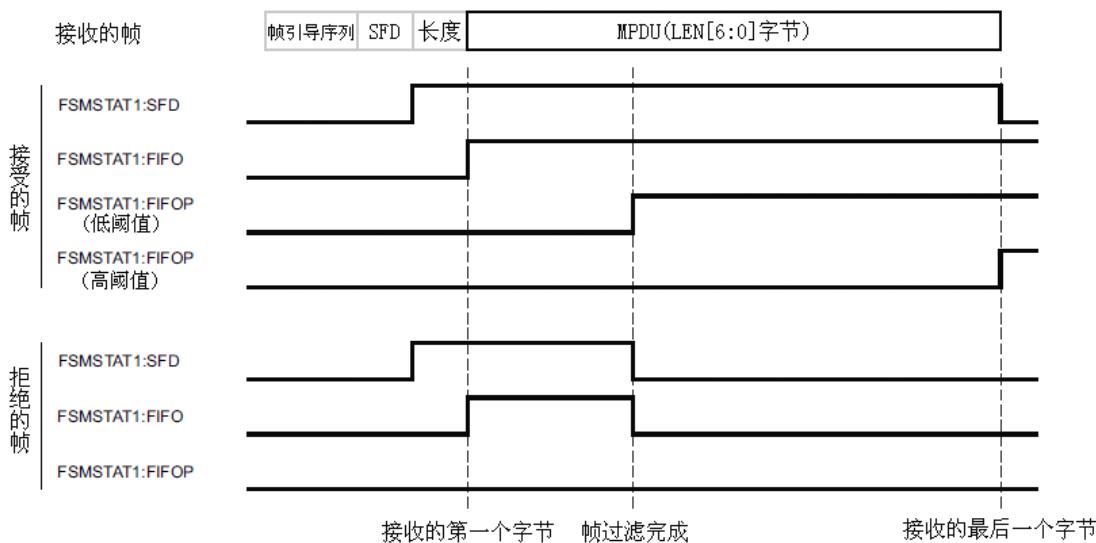


图 19-17 FIFO 和 FIFOP 信号的活动

当使用 FIFOP 作为微控制器的一个中断源时，应当使用中断服务程序来调整 FIFOP 阀值，为下一个中断做准备。在为一个帧的最后一个中断做准备的时候，阀值应该与剩余的字节数相匹配。

19.10.2 错误情况

与 RX FIFO 相关的错误情况有 2 个：

- 溢出，也就是当另一个字节被接收时 RX FIFO 已经满了的情况
- 下溢出，也就是软件试图从一个空的 RX FIFO 中读取一个字节的情况

如果 RFERRRF.RXOVERF 标志已经被置位，而且信号值 FSMSTAT1.FIFO=0 且 FSMSTAT1.FIFOP=1，则表示 RX 溢出。如果发生了错误，帧接收停止。在使用 ISFLUSHRX 选通清除错误情况之前，当前存储在 RX FIFO 里的帧可以被读出。注意，如果在帧被拒绝之前就发生的错误情况，被拒绝的帧可以产生 RX 溢出。

通过设置 RFERRF.RXUNDERF 标志来表示 RX 下溢出。RX 下溢出是一个严重的错误情况，不能在无差错的软件中发生，RXUNDERF 事件只能用于调试或用在看门狗功能里。注意，当读取操作和接收一个新字节同时发生时，不会产生 RXUNDERF 错误。

19.10.3 接收信号强度指示器 (RSSI)

无线模块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，或自动追加到接收的帧里。RSSI 值是通过 8 个符号周期内 (128us) 取平均值得到的，与 IEEE 802.15.4[1]一致。

RSSI 值是一个有符号的二进制补码，以 1dB 的步长为对数等级。

读 RSSI 值寄存器前应当先检查状态位 RSSI_VALID。RSSI_VALID 表示寄存器中的 RSSI 值真实有效，这意味着接收机已经使能了至少 8 个符号周期。

为了得出 RF 引脚上精度合理的实际信号功率 P，必须在 RSSI 值上添加一个偏移量：

$$P = RSSI - OFFSET [dBm]$$

例如，从 RSSI 寄存器中读到的值是-10，偏移量为 73db，那么 RF 的输入功率大约是-83dB。要使用的正确的偏移量值请见数据手册[2]。

在 RSSI 值第一次变为有效之后，无线模块如何更新 RSSI 寄存器是可以配置的。如果 FRMCTRL0.ENERGY_SCAN=0（默认），RSSI 寄存器包含最新的可用值，但是，如果 FRMCTRL0.ENERGY_SCAN=1，就执行一个峰值搜索，RSSI 寄存器包含从能量扫描使能以来的最大值。

19.10.4 链路质量指示

如同 IEEE 802.15.4[1]中的定义，链路质量指示 (LQI) 计量的就是所收到的包的强度和/或质量。IEEE 802.15.4 标准[1]所需的 LQI 值限制在 0~255，至少需要 8 个唯一的值。无线模块不直接提供 LQI 值，但是报告一些测量结果，微控制器可以使用这些测量结果来计算一个 LQI 值。

RSSI 的值可以用于 MAC 软件产生 LQI 值。直接使用 RSSI 值计算 LQI 值有若干缺点，例如：信道带宽内的窄带干扰会增加 RSSI 值，也就增加了 LQI 值，事实上也会降低链路质量。因此，对于每个输入的帧，无线模块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。虽然无线模块不做片码判定，但是这个无符号的 7 位数值可以看作是“片码错误率”。

正如 19.9.7 小节所述，当 MDMCTRL0AUTOCRC 已经置为 1 时，前 8 个符号的平均相关值与 RSSI、CRC OK/not OK 一起，附加在每个接收帧上。由无线模块检测出来，约为 110 的平均相关值表示最好质量的帧，而约为 50 的平均相关值表示典型的质量最差的帧。

软件必须将平均相关值转换为由 IEEE 802.15.4[1]定义的，范围为 0~255 的数值：

$$LQI = (CORR - a)b$$

式中：a 和 b 限制为 0~255，是基于包差错率 (PER) 测量的经验值，来作为一个相关值功能。

RSSI 和相关值结合起来，还可用于产生 LQI 值。

19.11 无线控制状态机

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 内核里的数据流，产生自动确认帧，以及控制所有的模拟 RF 校准。

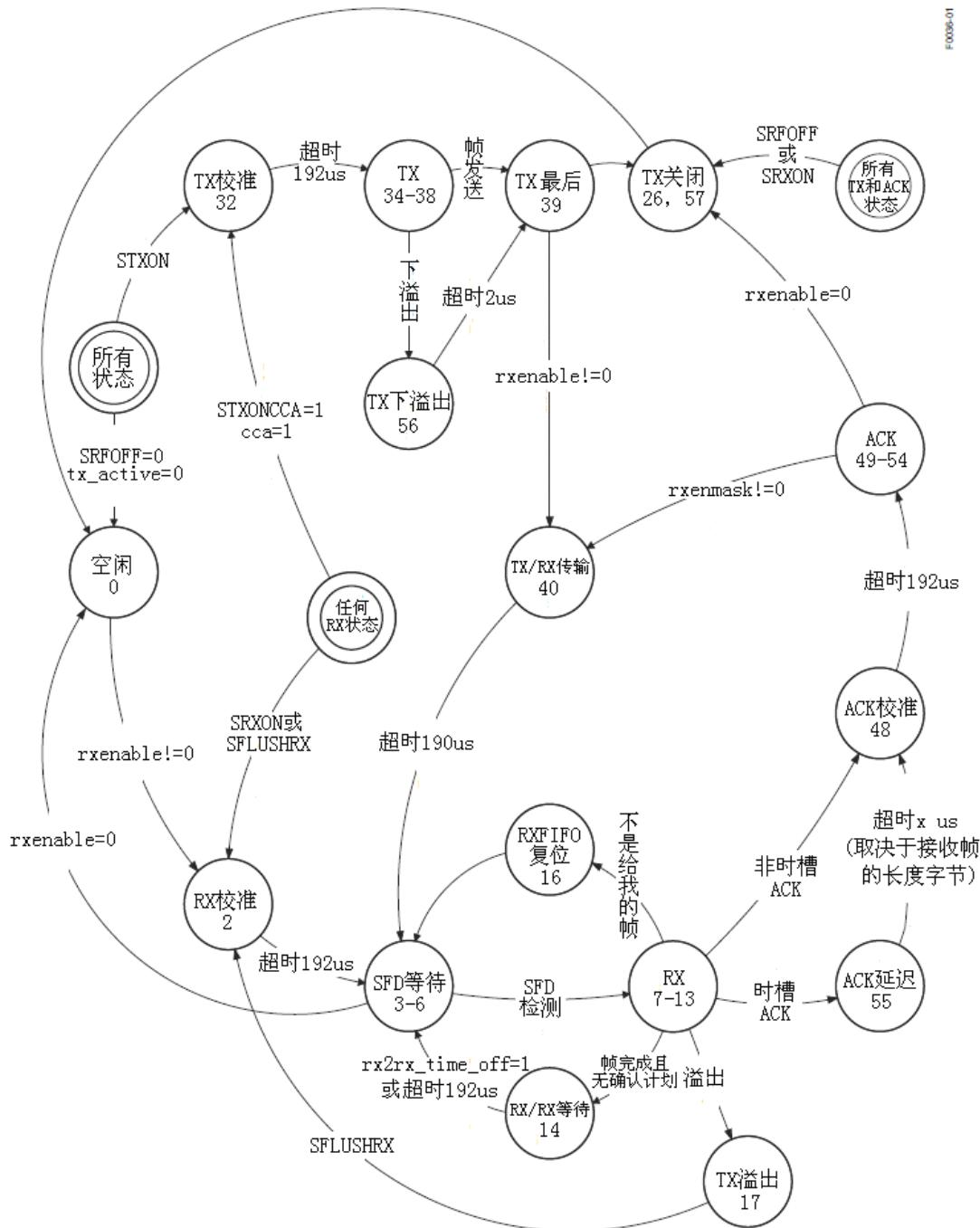


图 19-18 主要 FSM

表 19-3 列出了 FSM 状态到号码的映射，号码可以用寄存器 FSMSTAT0 中读取。注意，虽然 FSM 的状态可读，但是这一信息不能用来控制应用软件中程序流。这些状态的改变非常快（每 32MHz 时钟周期），所以一个 8MHz 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 传输	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 下溢出	56	0x38	1	0
TX 关闭	26, 57	0x1A, 0x39	1	0

19.12 随机数产生

RF 内核可以产生随机比特。当要求产生随机比特的时候，芯片必须处于 RX 模式。还必须确保芯片处于 RX 模式的时间足够长，用于瞬态消失。完成这一操作的简便方法就是等待 RSSI 有效信号变为高。

从寄存器 RFRND 中读取来自 I 或 Q 通道的单个随机比特。

随机试验表示 CC253x 的随机数产生效果良好。但是，存在轻微的直流分量。在这样一个简单的测试中，即多次读取 RFRND.IRND 寄存器，数据按字节分组。大约读出 2000 万个字节。当解释为 0-255 之间的无符号整数时，平均值为 127.6518，这就表示有一个直流分量。

图 19-19 为 2^{14} 首字节的快速傅立叶变换 (FFT)。注意，直流分量清晰可见。如 19-20 为 2000 万个值的直方图 (32 位进制)。

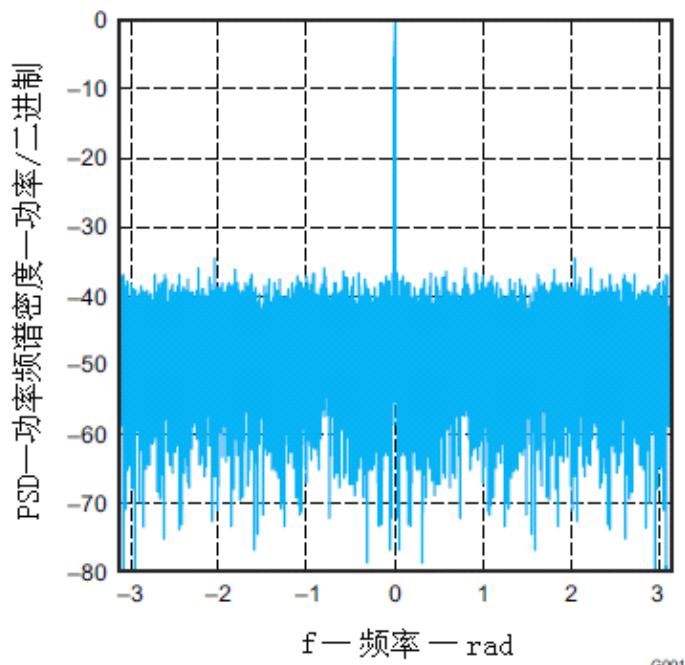


图 19-19 随机字节的 FFT

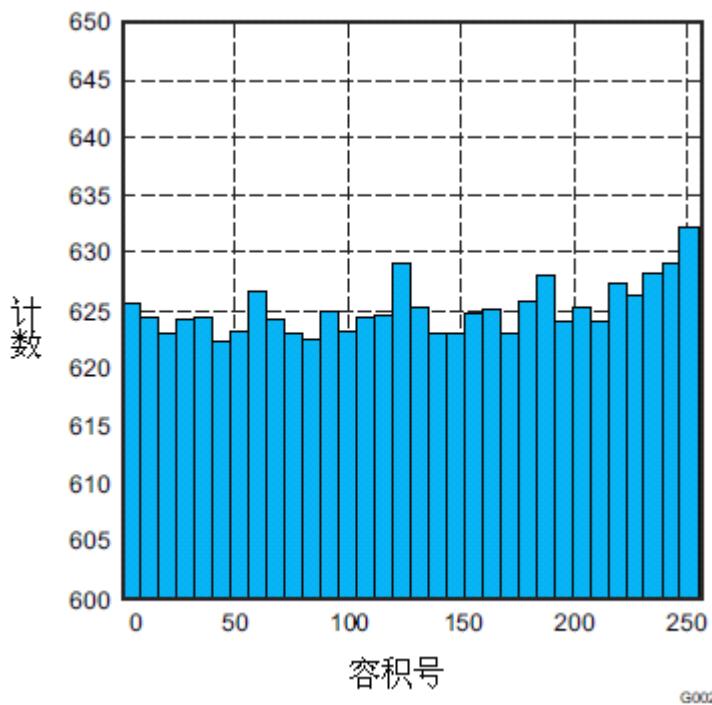


图 19-20 使用 RANDOM 指令产生的 2000 万字节的直方图

对于首个 2000 万字节的独立位，每一个的概率是

$$P(1) = 0.500602, P(0) = 1 - P(1) = 0.499398.$$

注意，要完全限定随机数发生器为真正的随机，需要做更细致的测试。互联网上有相关可用软件包，在[8][9]方面可能有用。

19.13 数据包嗅探和无线测试输出信号

数据包嗅探是一种无干扰观察发送数据或接收数据的方法。数据包分析仪输出一个时钟和一个数据信号，它们必须在时钟的上升沿被采样。这两个数据包分析仪信号作为 GPIO 输出是可见的。为了得到精确的时间戳，还应当输出 SFD 信号。

因为无线模块的数据率是 250kbps，数据包分析仪的时钟频率是 250kHz。数据为串行输出的，每个字节的最高有效位首先输出，这和实际的 RF 传输正好相反，但是处理数据时更方便。可以使用 SPI 从模式来接收数据流。

当在 TX 模式下嗅探帧时，调制器从 TX FIFO 读出的数据，和数据包分析仪输出的数据一样。但是，如果使能了自动 CRC 生成，数据包分析仪不能输出这 2 个字节。相反，它以 0x8080 替代 CRC 字节。该值永远不能作为一个接收帧的最后 2 个字节（如果使能了自动 CRC 校验），因此，它为嗅探数据接收机提供了一种方法，来区分嗅探数据是发送的帧还是接收的帧。

当在 RX 模式下嗅探帧时，解调器写入 RXFIFO 的数据，和数据分析仪输出的数据一样。换句话说，根据所设置的配置，最后 2 个字节可以是接收到的 CRC 值，或是可以自动替代 CRC 值的 CRC OK/RSSI/相关值/SRCRESINDEX 值。

要设置数据包分析仪的信号或其它一些 RF 内核观察输出（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：

第 1 步：确定哪个信号（rfc_obs_sig）要在哪个 GPIO 引脚（P1[0: 5]）上输出。使用

控制寄存器 OBSSEL x (OBSSEL0-OBSSEL5) 来进行设置, 控制观察结果输出到引脚 P1[0:5] (覆盖这些引脚的标准 GPIO 行为)。

第 2 步: 设置控制寄存器 RFC_OBS_CTRL (RFC_OBS_CTRL0 到 RFC_OBS_CTRL2) 来选择正确的信号; 例如, 对于数据包分析, 需要数据包分析仪数据信号 rfc_sniff_data, 需要相应的时钟信号 rfc_sniff_clk。

第 3 步: 为了数据包分析, 必须在 MDMTEST1 寄存器里使能数据包分析仪模块。

19.14 命令选通/CSMA-CA 选通处理器

命令选通/CSMA-CA 选通处理器 (CSP) 提供 CPU 和无线模块之间的控制接口。

CSP 通过 SFR 寄存器 RFST 以及 XREG 寄存器 CSPX、CSPY、CSPZ、CSPT、CSPSTAT、CSPCTRL 和 CSPPROG $<n>$ (n 为 0 到 23) 与 CPU 接口。CSP 向 CPU 发出中断请求。除此之外, CSP 通过观察 MAC 定时器事件与 MAC 定时器接口,。

CSP 允许 CPU 对无线模块发出命令选通, 从而控制无线模块的运行。

CSP 有两种操作模式:

- 立即命令选通执行
- 程序执行

立即命令选通作为立即命令选通指令写给 CSP, CSP 立即下达给无线模块。该模式中的立即命令选通指令仅用于控制 CSP。立即命令选通指令请见 19.14.8 的描述。

程序执行模式意味着 CSP 执行一序列用户定义的短程序, 该短程序存储在程序存储器或指令存储器之中。可用的指令在一个 20 条指令集中。19.14.8 小节定义了指令集。该短程序首先由 CPU 装入 CSP, 然后 CPU 指示 CSP 开始执行程序。

程序执行模式与 MAC 定时器一起, 允许 CSP 自动进行 CSMA-CA 运算, 这样, CSP 就成为 CPU 的一个协处理器。

下面的小节介绍 CSP 操作的详细情况。命令选通和 CSP 支持的其它指令在 19.14.9 小节中描述。

RFST (0xE1) —RF CSMA-CA/选通处理器

位	名称	复位	读/写	描述
7: 0	INSTR[7: 0]	0xD0	R/W	写入这个寄存器的数据会被写入 CSP 指令存储器。读这个寄存器会返回当前执行的 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, 来控制写入 RFST 的下一条指令存储在指令寄存器中的地址。处于调试目的, 当前加载到 CSP 的程序可以从 XREG 寄存器 CSPPROG $<n>$ 中读出。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。因此, 一个完整的 CSP 程序可能包含最大 24 个字节, 通过按照期望的顺序把每条指令写入 RFST 寄存器, 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令 SSTOP 复位到 0。

复位之后，指令存储器中充满 SNOP(无操作)指令(操作码值 0xC0)。立即选通 ISCLEAR 清除指令存储器，使用 SNOP 指令来填充指令存储器。

当 CSP 运行程序时，不可以使用 RFST 将指令写入指令存储器。否则会导致程序出错，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 (请见 19.14.3 小节)。

19.14.2 数据寄存器

CSP 有 4 个数据寄存器 CSPT、CSPX、CSPY 和 CSPZ，它们像 XREG 寄存器一样，可以被 CPU 读/写。这些寄存器也可以被某些指令读取或修改，这样，CPU 就可以设置 CSP 程序能够使用的参数，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 定时器溢出比较值。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 定时器的溢出而递减 1。当 CSPT 递减到 0 时，程序挂起，中断 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后，当立即命令选通指令 ISSRART 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 运行于设置的系统时钟频率上，为了无线模块能正确运行，系统时钟必须设置为 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 将返回当前正在执行的指令。只有一个例外，就是正在执行立即命令选通，届时，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：

- IRQ_CSP_STOP：当 CSP 执行完毕存储器中最后一条指令，且 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 CSP 在指令“WAIT W”或“WAIT X”之后，继续执行下一条指令时，该中断标志有效。
- IRQ_CSP_INT：当 CSP 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。因此如果指令 RANDXY 在上一个指令 RANDXY 之后立即发送随机数，则两次发送的随机数数值相同。

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。当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。使用 ISCLEAR 指令来清除 CSP 里的程序内容。

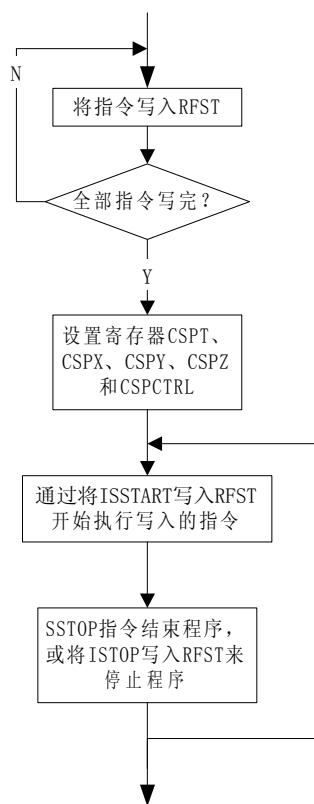


图 19-21 运行 CSP 程序

19.14.7 寄存器

CSPPROG<N> (N 的值从 0-23) (0x61C0+N) —CSP 程序

位	名称	复位	读/写	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	读/写	描述
7: 1	—	0000 000	R0	保留。读为 0
0	MCU_CTRL	0	R/W	CSP MCU 控制输入

CSPSTAT (0x61E1) —CSP 状态寄存器

位	名称	复位	读/写	描述
7: 6	—	00	R0	保留。读为 0
5	CSP_RUNNING	0	R	1: CSP 正在运行 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	读/写	功能
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	读/写	功能
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	读/写	功能
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	读/写	功能
7: 0	CSPT	0xFF	R/W	CSP T 数据寄存器。CSP 程序运行时，每次 MAC 定时器溢出，寄存器的内容就递减。当 CSPT 递减到 0 时，CSP 程序停止。设置 CSPT=0xFF 可防止寄存器递减。

19.14.8 指令集概况

本节给出指令集的概况。目的是总结和定义指令操作码。关于每个指令的描述请见 19.14.9 小节。每条指令包含一个字节用来写入寄存器 RFST，最终存储到指令存储器。

立即选通指令 (ISxxx) 不在程序中使用。当 ISxxx 指令写入寄存器 RFST 时，就立即执行。此时，如果 CSP 正在运行程序，则当前的指令延迟执行，直到立即选通指令 ISxxx 执行完毕，才重新执行 CSP 所运行的程序。

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 指令集概况

助记符	7	6	5	4	3	2	1	0	描述
SKIP<C>,<S>	0	S2	S1	S0	N	C2	C1	C0	<p>条件 C 上跳过 S 指令。当条件 (C XOR N) 为真时，跳过下一跳 S 指令，否则执行下一条指令。</p> <p>如果 S=0，重新执行条件跳转（即一直循环，直到条件为假）。跳过命令缓冲区的最后一条指令引起一个隐含 STOP 命令。条件是：</p> <p>C=0 CCA 为真</p> <p>C=1 接收到同步字，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）</p> <p>C=2 MCU 控制位为 1</p> <p>C=3 命令缓冲区为空</p> <p>C=4 寄存器 X=0</p> <p>C=5 寄存器 Y=0</p> <p>C=6 寄存器 Z=0</p> <p>C=7 RSSI_VALID=1</p>
WAIT <W>	1	0	0	W4	W3	W2	W1	W0	等待 W 次 MAC 定时器溢出。等待，直到 MAC 定时器溢出了 W 次 (W=0, 等待 32 次)，然后继续执行。继续执行时产生一个 IRQ_CSP_WAIT 中断请求。
RPT<C>	1	0	1	0	N	C2	C1	C0	条件 C 时重复循环。如果条件 C 为真，去往最后一个 LABEL 指令后的指令（地址在循环开始寄存器）；如果条件为假或没有 LABEL 指令被执行，去往下一条指令。
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。
LABEL	1	0	1	1	1	0	1	1	设置下一条指令为一个重复循环的开始。登记下一条指令的地址到循环开始寄存器。
WAIT X	1	0	1	1	1	1	0	0	等待 MAC 定时器溢出[X]次，[X]是寄存器 X 的值。每次检测到一个 MAC 定时器溢出，X 就递减。只要 X=0，就继续执行（如果指令运行时 X=0，不执行等待而是直接继续执行）。继续执行时产生一个 IRQ_CSP_WAIT 中断请求。
RANDXY	1	0	1	1	1	1	0	1	将随机值装入寄存器 X 的[Y]的低位 (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	递增寄存器 X
INCY	1	1	0	0	0	0	0	1	递增寄存器 Y
INCZ	1	1	0	0	0	0	1	0	递增寄存器 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	寄存器 Y≤min (Y+1, M)。递增 Y，但不大于 M
Sxxx	1	1	0	1	S3	S2	S1	S0	执行命令选通 S。发送命令选通 S 到 FFCTRL。 最多支持 32 个命令选通。除了一般的命令选通，还支持另外 2 个命令选通，这 2 个命令选通只适用于命令选通处理器： SNOP：无操作 SSTOP：停止命令选通处理器执行，并且使任何设置的标签无效。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，绕过命令缓冲区总的指令。如果当前的缓冲区指令是一个选通命令，它被延迟。除了一般的命令选通，还支持另外 2 个命令选通，这 2 个命令选通只适用于命令选通处理器： ISSTART：命令选通处理器从命令缓冲器的第一条指令开始执行。 ISSTOP：停止命令选通处理器执行，并且使任何设置的标签无效。发出一个 IRQ_CSP_STOP 中断请求。
ISCLEAR	1	1	1	1	1	1	1	1	立即清除 CSP 程序。复位 CSP 程序计数器。

19.14.9 指令集定义

指令的基本类型有 20 类。此外，每条选通命令和立即选通命令可以分为 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一个指令。

注意：下面定义的符号仅在本小节使用。

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：寄存器 Z 递减 1。原始值 0x00 下溢出到 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: 寄存器 Y 递减 1。原始值 0x00 下溢出到 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: 寄存器 X 递减 1。原始值 0x00 下溢出到 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: 递增 Z

描述: 寄存器 Z 递增 1。原始值 0xFF 溢出到 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: 递增 Y

描述: 寄存器 Y 递增 1。原始值 0xFF 溢出到 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: 递增 X

描述: 寄存器 X 递增 1。原始值 0xFF 溢出到 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: 递增 Y, 结果不大于 M

描述: 如果结果小于 M, 寄存器递增 1; 否则, 寄存器 Y 装入值 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: 装入随机值到 X

描述: 寄存器 X 的[Y]的低位 (LSB) 装入一个随机值。注意, 如果两个 RANDXY 指令接连下达, 则会产生两个相同的随机值。如果 Y 等于 0 或 Y 大于 7, 那么一个 8 位随机值被装入 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: 执行这条指令时声明 IRQ_CSP_INT 中断。

操作: IRQ_CAP_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 X 次 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。每次检测到一个 MAC 定时器溢出, X 就递减。只要 X=0, 程序就继续执行 (如果指令运行时 X=0, 不执行

等待而是直接继续执行)。继续执行时产生一个 IRQ_CSP_WAIT 中断请求。注意：与 WAIT W 相比，它们的不同处在于 W 是一个固定值，而 X 是一个寄存器值(X 有可能被改变，这样当 WARTX 指令运行时，实际的溢出次数与 X 的值不符号)。

操作：如果 MAC 定时器溢出=true X=X-1

当 X>0 时， PC=PC

如果 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。

程序继续运行下一条指令，当等待条件为真时，声明中断标志 IRQ_CSP_WT。

操作：当 MAC 定时器溢出次数 = true < W 时，PC=PC

如果 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。当等待条件为真时，程序继续执行下一条指令。

操作：当 MAC 定时器比较 = false 时，PC=PC

如果 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。当等待条件为真时, 程序继续执行下一条指令。

操作: 当 MAC 定时器比较 = false 时, PC=PC

如果 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

19.14.9.15 LABEL

功能: 设置循环标签

描述: 设置下一条指令为循环的起始点。如果当前指令是指令存储器中最后一条指令, 那么当前的 PC 就设置为循环的起始点。如果执行了多个标签指令, 最后执行的那个标签为有效标签。之前的标签被移除, 意味着仅支持一层循环。

操作: LABLE:=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, 那么跳转到上一个 LABEL 指令定义的指令, 即循环的起始点。如果条件 C 为假, 或者没有执行 LABEL 指令, 则程序将从下一条指令继续执行。可以通过设置 N=1 来取消条件 C, 下表就是对它的详细描述。

条件代码 C	描述	功能
000	CCA 为真	CCA = 1
001	接收到同步字, 仍然接收数据包或发送同步字, 仍然发送数据包	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 有效

RSSI_VALID=1

操作: 如果 (C xor N) = true PC = LABEL

如果 (C xor N) = false 或 LABEL = 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: 条件跳转指令**描述:** 条件 C 上跳过 S 指令 (条件 C 可能被取消; N = 1)。当条件 (C xor N) 为真时, 跳过下一跳 S 指令, 否则执行下一条指令。如果 S=0, 重新执行条件跳转 (即一直循环, 直到条件为假)。跳过命令缓冲区的最后一条指令引起一个隐含 STOP 命令。

条件代码 C	描述	功能
000	CCA 为真	CCA = 1
001	接收到同步字, 仍然接收数据包或发送同步字, 仍然发送数据包	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 有效	RSSI_VALID=1

操作: 如果 (C xor N) = true PC = PC + S + 1

如果 (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 使能和校准频率合成器

描述: 指令 SRXON 声明输出 FFCTL_SRXON_STRB, 为 RX 使能和校准频率合成器。

该指令在执行下一条指令之前, 等待无线模块应答这条命令。

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: 校准后使能 TX

描述: 指令 STXON 在频率合成器校准之后使能 TX。该指令在执行下一条指令之前, 等待无线模块应答这条命令。如果 SET_RXENMASK_ON_TX 置位, 就设置 RXENABLE 里的一个位。

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 指示一个空闲信道, 则使能校准和 TX

描述: 如果 CCA 指示一个空闲信道, 则频率合成器校准后, 指令 STXONCCA 使能 TX.

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: 采样当前的 CCA 值到 SAMPLED_CCA

描述: 当前 CCA 值被写入 XREG 的 SAMPLED_CCA。

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: 禁止 RX/TX 和频率合成器

描述: 指令 SRFOFF 声明禁止 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 缓冲器并复位解调器

描述: 指令 SFLUSHRX 清空 RX FIFO 缓冲器并复位解调器。该指令在执行下一条指令之前，等待无线模块应答该命令。

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 缓冲器

描述: 指令 SFLUSHTX 清空 RX FIFO 缓冲器。该指令在执行下一条指令之前，等待无线模块应答该命令。

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: 发送具有已经清 0 的未决域的确认帧

描述: 指令 SACK 发送一个确认帧。该指令在执行下一条指令之前，等待无线模块应答该命令。

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: 发送已经设置未决域的确认帧

描述: 指令 SACKPEN 发送一个已经设置未决域确认帧。该指令在执行下一条指令之前，等待无线模块应答该命令。

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: 中止确认帧的发送

描述: 指令 SNACK 中止发送确认到已经接收的当前帧。

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 的位

描述: 指令 SRXMASKBITSET 设置 RXENABLE 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 的位

描述: 指令 SRXMASKBITCLR 清除 RXENABLE 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，并且声明 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: 启动程序执行

描述: ISSTART 指令启动 CSP 程序从写入指令存储器的第一条指令执行

操作: PC := 0，启动执行

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 使能和校准频率合成器

描述: 指令 ISRXON 为 RX 立即使能和校准频率合成器。

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 的位

描述: 指令 ISRXMASKBITSET 立即设置 RXENABLE 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 的位

描述: 指令 ISRXMASKBITCLR 立即清除 RXENABLE 的位 5。

操作: SRXMASKBITCLR

操作码: 0xE5

7	6	5	4	3	2	1	0
1	1	1	0	0	1	0	1

19.14.9.37 ISTXON

功能: 校准后使能 TX

描述: 指令 ISTXON 在频率合成器校准之后立即使能 TX。该指令在执行下一条指令之前，等待无线模块应答这条命令。

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 指示一个空闲信道，则使能校准和 TX

描述: 如果 CCA 指示一个空闲信道，则频率合成器校准后，指令 ISTXONCCA 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: 采样当前的 CCA 值到 SAMPLED_CCA

描述: 当前 CCA 值被立即写入 XREG 的 SAMPLED_CCA。

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: 禁止 RX/TX 和频率合成器

描述: 指令 ISRFOFF 立即禁止 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 缓冲器并复位解调器

描述: 指令 ISFLUSHRX 立即清空 RX FIFO 缓冲器并复位解调器。

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 缓冲器

描述: 指令 ISFLUSHTX 立即清空 RX FIFO 缓冲器。

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: 发送具有已经清 0 的未决域的确认帧

描述: 指令 ISACK 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: 发送已经设置未决域的确认帧

描述: 指令 ISACKPEND 立即发送一个已经设置未决域确认帧。该指令在执行下一条指令之前，等待无线模块接收和解释该命令。

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: 中止确认帧的发送

描述: 指令 ISNACK 立即中止发送确认到已经接收的当前帧。

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: 指令 ISCLEAR 立即清除程序存储器, 复位程序计数器, 并终止任何正在运行的程序。不产生停止中断。LABEL 指针并清除。

操作: PC := 0, 清除程序存储器

操作码: 0xFF

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1

19.15 RF 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	SRCSHOTEN2	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	RXFISRT	RXFIFO_CNT
0x619C	TXFIFO_CNT	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为能获得最佳性能的值。

RX 和 TX 都应当设置下面的这些设置。虽然不是所有的设置对于 RX 和 TX 都是必需的，但出于简化的考虑，建议进行设置（允许一组设置被写到初始化代码中）

表 19-6 要求对默认值进行更新的寄存器

寄存器名称	新值(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带宽。
FSCAL1	0x00	和默认设置相比，减少了大约 3dB 的 VCO 泄漏。要获得最佳 EVM，建议使用默认设置。

19.15.2 寄存器访问模式

表 19-7 中的“模式”这一列指明了每一位允许哪种访问模式。“描述”这一列给出了不同访问模式的意思。

表 19-7 寄存器位访问模式

模式	描述
R	读
W	写
R0	读常数 0
R1	读常数 1
W1	只能写 1
W0	只能写 0
R*	读取的值不是实际寄存器的值，而是由模块得到的值。这通常用于可以自动产生的配置值（通过校准、动态控制等），或可以手动覆盖的寄存器值。图 19-22 为 AGCCTRL2 寄存器的结构示例。

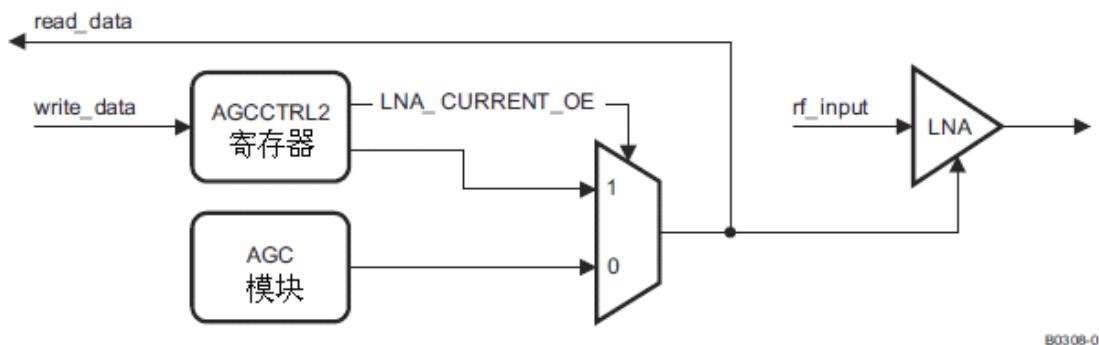


图 19-22 R*寄存器访问模式的硬件结构示例

19.15.3 寄存器描述

FRMFILT0 (0x6180) —帧过滤

位	名称	复位	读/写	功能
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:]与 FCF[9: 7]相“与”。如果结果非零，则帧过滤使能，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	0	R/W	当设备为一个 PAN 协调器时，该位应当设为 1，以接受没有目标地址的帧 (如 802.15.4(b)里 7.5.6.2 小节所规定) 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。 如果该位为 1，无线模块执行 802.15.4(b)中 7.5.6.2 小节所指定的帧过滤，即第三过滤级别。 FRMFILT0[6: 1] 和 FRMFILT1[7: 1]，与本地地址信息一起，定义过滤算法的行为。 0: 帧过滤关闭 (不关心 FRMFILT0[6: 1], FRMFILT1[7: 1] 和 SRCMATCH[2: 0]) 1: 帧过滤开启

FRMFILT1 (0x6181) —帧过滤

位	名称	复位	读/写	功能
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	这些位用于在执行帧类型过滤前修改接收的帧类型域。这个修改不影响已经写入 RX FIFO 的帧。 00: 不变 01: 反转最高位 10: 设置最高位为 0 11: 设置最高位为 1
0	—	0	R/W	保留。总是写 0

SRCMATCH (0x6182) —源地址匹配和未决位

位	名称	复位	读/写	功能
7: 3	—	0000 0	R/W	保留。总是写 0
2	PEND_DATAREQ_ONLY	1	R/W	如果该位为 1, AUTOPEND 功能也要求接收的帧是一个 DATA REQUEST 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) 返回的确认帧的未决位自动设置, 给出: — 设置了 FRMFILT0.FRAME_FILTER_EN — 设置了 SRCMATCH.SRC_MATCH_EN — 设置了 SRCMATCH.AUTOPEND — 接收的帧匹配当前的 SRCMATCH.PEND_DATAREQ_ONLY 设置 — 接收的源地址至少与源匹配表里的一个条目相匹配, 这在 SHOTR_ADDR_EN 和 SHORT_PEND_EN, 或在 EXT_ADDR_EN 和 EXT_PEND_EN 里都使能了。 注意: SHORT_PEND_EN 和 EXT_PEND_EN 的详细信息在存储器映射里进行了描述 (19.4 节)

0	SRC_MATCH_EN	1	R/W	源地址匹配使能（如果 FRMFILT0.FRAME_FILTER_EN=0，该位为 1 或 0 不重要）
---	--------------	---	-----	--

SRCSHORTEN0 (0x6183) — 短地址匹配

位	名称	复位	读/写	功能
7: 0	SHORT_ADDR_EN[7: 0]	0x00	R/W	24 位字 SHORT_ADDR_EN 的位 7: 0 部分为 24 个短地址表的位 7: 0 部分的每个条目使能/禁止 源地址匹配。 可选的安全功能：当条目被更新时，要保证源 匹配表的一个条目不被使用，就在更新时设置 相应的 SHORT_ADDR_EN 位为 0。

SRCSHORTEN1 (0x6184) — 短地址匹配

位	名称	复位	读/写	功能
7: 0	SHORT_ADDR_EN[15: 8]	0x00	R/W	24 位字 SHORT_ADDR_EN 的位 15: 8 部分为 24 个短地址表的位 15: 8 的每个条目使能/禁止源 地址匹配。见 SRCSHORTEN0 的描述。

SRCSHORTEN2 (0x6185) — 短地址匹配

位	名称	复位	读/写	功能
7: 0	SHORT_ADDR_EN[23: 16]	0x00	R/W	24 位字 SHORT_ADDR_EN 的位 23: 16 部分为 24 个短地址表的位 23: 16 的每个条目使能/禁 止源地址匹配。见 SRCSHORTEN0 的描述。

SRCEXTEN0 (0x6186) — 扩展地址匹配

位	名称	复位	读/写	功能
7: 0	EXT_ADDR_EN[7: 0]	0x00	R/W	24 位字 EXT_ADDR_EN 的位 7: 0 部分为 12 个扩展地址表的位 7: 0 的每个条目使能/禁止源 地址匹配。 写访问：扩展地址表的条目 n[0 到 7]的扩展地 址使能映射到 EXT_ADDR_EN[2n]。所有的 EXT_ADDR_EN[2n+1]位都为只读，并且写入无 效。 读访问：扩展地址表的条目 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：当条目被更新时，要保证源 匹配表的一个条目不被使用，就在更新时设置 相应的 EXT_ADDR_EN 位为 0。

SRCEXTEN1 (0x6187) —扩展地址匹配

位	名称	复位	读/写	功能
7: 0	EXT_ADDR_EN[15: 8]	0x00	R/W	24 位字 EXT_ADDR_EN 的位 15: 8 部分为 12 个扩展地址表的位 15: 8 的每个条目使能/禁止源地址匹配。 见 SRCEXTEN0 的描述。

SRCEXTEN2 (0x6188) —扩展地址匹配

位	名称	复位	读/写	功能
7: 0	EXT_ADDR_EN[23: 16]	0x00	R/W	24 位字 EXT_ADDR_EN 的位 23: 16 部分为 12 个扩展地址表的位 23: 16 的每个条目使能/禁止源地址匹配。 见 SRCEXTEN0 的描述。

FRMCTRL0 (0x6189) —帧处理

位	名称	复位	读/写	功能
7	APPEND_DATA_MODE	0	R/W	如果 AUTOCRC=0: 不关心 如果 AUTOCRC=1: 0: RSSI+crc_ok 位和 7 位相关值附加到每个接收帧的末尾。 1: RSSI+crc_ok 位和 7 位 SRCRESINDEX 附加到每个接收帧的末尾。详见表 19-1。
6	AUTOCRC	1	R/W	在 TX 模式 1: 硬件产生一个 CRC-16(ITU-T)并附加到发送的帧里。不需要将最后 2 字节写入 TXBUF。 0: 无 CRC-16 附加到帧。必须手动产生帧的最后 2 字节并写入 TXBUF(如果不这样做，发生 TX_UNDERFLOW) 在 RX 模式 1: 硬件检查 CRC-16, 用一个 16 位状态字代替 RX FIFO, 这个 16 位状态字包含一个 CRC OK 位。可以通过 APPEND_DATA_MODE 控制状态字。 0: 帧的最后 2 字节(crc-16 域)存储在 RXFIFO. CRC 校验(如果有)必须手动完成。 注意该设置不影响确认传输(包括 AUTOACK).
5	AUTO_ACK	0	R/W	定义无线模块是否自动发送确认帧。当自动确认使能时, 所有通过地址过滤被接受的帧, 都设置确认请求标志, 被接收后的 12 个符号周期, 自动确认一个有效 CRC。 0: 自动确认禁止 1: 自动确认使能
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

				的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: 正常操作, 使用 RXFIFO 01: 保留 10: RXFIFO 循环忽略 RXFIFO 溢出, 无限接收 11: 除了禁止了符号搜索, 其余和正常操作一样。如果不需要寻找符号, 可以用于 RSSI 或 CCA 测量。
1: 0	TX_MODE[1: 0]	00	R/W	设置 TX 的测试模式 00: 正常操作, 发送 TXFIFO 01: 保留。不能使用。 10: TXFIFO 循环忽略 TXFIFO 下溢出并且循环读, 无限发送 11: 发送来自 CRC 的伪随机数据, 无限发送

FRMCTRL1 (0x618A) —帧处理

位	名称	复位	读/写	功能
7: 3	—	00000	R0	读为 0
2	PENDING_OR	0	R/W	定义将要输出的确认帧的未决数据位总是设为 1, 还是由主 FSM 和地址过滤控制。 0: 未决数据位由主 FSM 和地址过滤控制 1: 未决数据位总是为 1
1	IGNORE_TX_UNDERF	0	R/W	定义是否忽略 TX 下溢出 0: 正常 TX 操作。检测 TX 下溢出, 如果发生了下溢出就中止 TX 1: 忽略 TX 下溢出。发送由长度域给出的字节数
0	SET_RXENMASK_ON_TX	1	R/W	定义 STXON 设置 RXENABLE 寄存器的位 14, 还是不改变它。 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	名称	复位	读/写	功能
7: 0	RXENMASK[7: 0]	0x00	R	RXENABLE 使能接收机。该寄存器里的一个非零值会导致主 FSM 在接收机在以下情况情况下使能它：处于空闲模式，发送完成后，确认发送完成后下面的选通可以修改 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

				<p>SRXMASKBITSET: 设置 RXENMASK 的位 5 SRXMASKBITCLR: 清除 RXENMASK 的位 5 CPU 通过访问寄存器 RXMASKSET 和 RXMASKCLR 来直接修改 RXENABLE。 如果 CSP 和 CPU 同时想要修改 RXENMASK，可能会产生冲突。要处理同时对 RXENMASK 的访问，必须遵循下面的规则： — 如果 2 个源不冲突（它们修改寄存器的不同部分），2 个修改 RXENMASK 的请求都被处理 — 如果 2 个想同时修改使能，对 RXMASKSET 和 RXMASKCLR 的总线写操作的优先级高于 CSP。强烈建议避免这种情况。</p>
--	--	--	--	--

RXMASKSET (0x618C) —RX 使能

位	名称	复位	读/写	功能
7: 0	RXENMASKSET[7: 0]	0x00	W/R0	写入时，写入的数据与 RXENMASK 相“或”，存储在 RXENMASK。

RXMASKCLR (0x618D) —RX 禁止

位	名称	复位	读/写	功能
7: 0	RXENMASKCLR[7: 0]	0x00	W/R0	写入时，写入的数据反相并与 RXENMASK 相“与”，存储在 RXENMASK。 例如，如果写 1 到该寄存器的 1 个位或多个位，RXENMASK 里的对应位将被清除。

RFIRQM0 (0x61A3) —RF 中断使能

位	名称	复位	读/写	功能
7: 0	RFIRQM[7: 0]	0x00	R/W	位使能的使能输出中断源 位 7-位 0: 7: RXMASKZERO 6: RXPKTDONE 5: FRAME_ACCEPTED 4: SRC_MATCH_FOUN 3: SRC_MATCH_DONE 2: FIFOP 1: SFD 0: ACT_UNUSED

RFIRQM1 (0x61A4) —RF 中断使能

位	名称	复位	读/写	功能
7: 0	RFIRQM[14: 8]	0x00	R/W	位使能的使能输出中断源 位 7-位 0: 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 错误中断使能

位	名称	复位	读/写	功能
7: 0	RFERRM[7: 0]	0x00	R/W	位使能的使能输出中断源 位 7-位 0: 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	名称	复位	读/写	功能
7	—	0	R0	读为 0
6: 0	FREQ[6: 0]	0x0B (2405MHz)	R/W	频率控制字。 $f_{RF} = f_{LO} = (2394 + FREQ[6:0]) \text{MHz}$ FREQ[6: 0]中的频率字是一个从 2394 的偏移值。CC253x 支持的频率范围为 2394MHz 到 2507MHz。FREQ[6: 0]可用的设置从 0 到 113。超过此范围的设置（114-127）给定的频率为 2507MHz。 IEEE 802.15.4 指定的频率范围从 2405MHz 到 2480MHz，一共 16 个信道，每个信道的步长为 5MHz。信道编号从 11 到 26。因此，对于 IEEE 802.15.4-2006 标准的系统，唯一有效的设置为 FREQ[6: 0]=11+5（信道号码—11）。

FREQTUNE (0x618E) —晶体振荡器频率调整

位	名称	复位	读/写	功能
7: 4	—	0x0	R0	读为 0
3: 0	XOSC32M_TUNE[3: 0]	0xF	R/W	调整晶体振荡器 默认设置为 1111，不调整晶体振荡器。改变默认设置，从另外的电容转换到振荡器，能有效降低晶体振荡器的频率。因此，设置越高，给出的频率也就越高。

TXPOWER (0x6190) — 控制输出功率

位	名称	复位	读/写	功能
7: 0	PA_POWER[7: 0]	0xF5	R/W	功率放大器 (PA) 功率控制。 注意：在进入 TX 之前，应当更新该值。建议只请参考 CC2530 数据手册，或见 19.8.13 小节。

TXCTRL (0x6191) — 控制 TX 设置

位	名称	复位	读/写	功能
7	—	0	R0	保留
6: 4	DAC_CURR[2: 0]	10	R/W	改变 DAC 电流
3: 2	DAC_DC[1: 0]	01	R/W	调整到 TX 混频器的直流电平
1: 0	TXMIX_CURRENT[1: 0]	0xF	R/W	发送混频器内核电流：该值越高，电流越大

FSMSTAT0 (0x6192) — 无线状态寄存器

位	名称	复位	读/写	功能
7	—	0	R	保留
6	CAL_RUNNING	0	R	频率合成器校准状态 0：校准已完成或未启动 1：校准正在进行
5: 0	FSM_FFCTRL_STATE[5: 0]	—	R	给出 FIFO 和帧控制 (FFCTRL) 有限状态机的当前状态

FSMSTAT1 (0x6193) — 无线状态寄存器

位	名称	复位	读/写	功能
7	FIFO	0	R	只要 RXFIFO 中有数据，该位就为 1。RXFIFO 溢出期间该位为 0。
6	FIFOP	0	R	如果 RXFIFO 中通过帧过滤的数据字节多于 FIFOP_THR 个字节，该位为 1。 如果 RXFIFO 中至少有一个完整的帧，该位为 1。 如果从 RXFIFO 中读出一个字节，该位再次设为 0，这使得 FIFO 中有 FIFOP_THR 个字节。 RXFIFO 溢出期间该位为 1。
5	SFD	0	R	在 TX 模式 0：接收了一个带有 SFD 的完整帧或没有发送 SFD 1：已发送 SFD 在 RX 模式 0：接收到一个完整帧或没有接收到 SFD 1：接收到 SFD
4	CCA	0	R	空闲信道评估。取决于 CCA_MODE 设置。详见 CCACTRL1。
3	SAMPLED_CCA	0	R	包括 CCA 的采样值。只要发送了一个 SSAMPLECCA 或 STXONCCA 选通，该值就被更新。

2	LOCK_STATUS	0	R	PLL 处于锁定状态时为 1，否则为 0。
1	TX_ACTIVE	0	R	状态信号。当 FFCTRL 为发送状态之一时有效
0	RX_ACTIVE	0	R	状态信号。当 FFCTRL 为接收状态之一时有效

FIFOPCTRL (0x6194) —FIFOP 阈值

位	名称	复位	读/写	功能
7	—	0	R0	读为 0
6: 0	FIFOP_THR[6: 0]	0x40	R/W	产生 FIFOP 信号时使用的阈值

FSMCTRL (0x6195) —FSM 选项

位	名称	复位	读/写	功能
7: 2	—	0000 00	R0	读为 0
1	SLOTTED_ACK	0	R/W	控制发送确认帧的时间。 0: 确认帧在收到请求确认的输入帧 12 个符号周期之后发送 1: 确认帧在收到请求确认的输入帧多于 12 个符号周期之后，在一个退避时槽边界发送。
0	RX2RX_TIME_OFF	1	R/W	定义在帧接收结束后，是否使用一个 12 个符号的超时 0: 无超时 1: 12 个符号周期超时

CCACTRL0 (0x6196) —CCA 阈值

位	名称	复位	读/写	功能
7: 0	CCA_THR[7: 0]	0xE0	R/W	空闲信道评估阈值，有符号的二进制补码，用来与 RSSI 比较。 计量单位是 1dB，偏移量大约为 76dBm。当接收信号低于这个值时，CCA 信号变高。CCA 信号可从 CCA 引脚获得，存储在 FSMSTAT1 寄存器里。 注意，为了避免 CCA 信号的错误行为，该值决不能低于 CCA_HYST—128。 注意：转化为输入电平的复位值约为 -32—76=-108dBm，这远低于灵敏度的限制。这意味着 CCA 信号永远不会指示一个空闲信道。 该寄存器必须更新为 0xF8，转化为一个大约 -8—76=-84dBm 的输入电平。

CCACTRL1 (0x6197) —其它 CCA 选项

位	名称	复位	读/写	功能
7: 5	—	000	R0	读为 0
4: 3	CCA_MODE[1:]	11	R/W	00: CCA 总是设为 1 01: 当 RSSI<CCA_THR-CCA_HYST 时，CCA=1； 当 RSSI≥CCA_THR 时，CCA=0 10: 当没有收到帧时，CCA=1，否则 CCA=0

				11: 当 RSSI<CCA_THR-CCA_HYST 且没有收到帧时, CCA=1; 当 RSSI≥CCA_THR 或收到帧时, CCA=0
2: 0	CCA_HYST[2: 0]	010	R/W	设置 CCA 滞后电平。无符号值, 单位为 dB。

RSSI (0x6198) —RSSI 状态寄存器

位	名称	复位	读/写	功能
7: 0	RSSI_VAL[7: 0]	0x80	R	RSSI 以对数为尺度的估计值, 有符号的二进制补码。 计量单位是 1dB, 为了将寄存器值转化为实际的 RSSI 值, 使用的偏移量请见数据手册。RSSI 值是一个超过 8 个符 号周期的平均值。在第一次读 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	名称	复位	读/写	功能
7: 1	—	0000 000	R0	读为 0
0	RSSI_VALID	0	R	RSSI 值有效。进入 RX 后发生 8 个符号周期。

RXFIRST (0x619A) —RXFIFO 里的第一个字节

位	名称	复位	读/写	功能
7: 0	DATA[7: 0]	0x00	R	RXFIFO 的第一个字节。 注意: 读该寄存器不会修改 FIFO 的内容

RXFIFOCNT (0x619B) —RXFIFO 里的字节数

位	名称	复位	读/写	功能
7: 0	RXFIFOCNT[7: 0]	0x00	R	RXFIFO 里的字节数。无符号整数。

TXFIFOCNT (0x619C) —TXFIFO 里的字节数

位	名称	复位	读/写	功能
7: 0	TXFIFOCNT[7: 0]	0x00	R	TXFIFO 里的字节数。无符号整数。

RXFIRST_PTR (0x619D) —RXFIFO 指针

位	名称	复位	读/写	功能
7	—	0	R	保留
6: 0	RXFIRST_PTR[6: 0]	000 0000	R	RXFIFO 里第一个字节的 RAM 地址偏移量。

RXLAST_PTR (0x619E) —RXFIFO 指针

位	名称	复位	读/写	功能
7	—	0	R	保留
6: 0	RXLAST_PTR[6: 0]	000 0000	R	RXFIFO 里最后一个字节+1 的 RAM 地址偏移量。

RXP1_PTR (0x619F) —RXFIFO 指针

位	名称	复位	读/写	功能
7: 0	RXP1_PTR[7: 0]	0x00	R	RXFIFO 的第一个帧的第一个字节的 RAM 地址偏移量。

TXFIRST_PTR (0x61A1) —TXFIFO 指针

位	名称	复位	读/写	功能
7: 0	TXFIRST_PTR[7: 0]	0x00	R	RXFIFO 里第一个字节的 RAM 地址偏移量。

TXLAST_PTR (0x61A2) —TXFIFO 指针

位	名称	复位	读/写	功能
7: 0	TXLAST_PTR[6: 0]	0x00	R	RXFIFO 里最后一个字节+1 的 RAM 地址偏移量。

MDMCTRL0 (0x61A8) —控制调制解调器

位	名称	复位	读/写	功能
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多少个零符号。注意只有一个要求，就是相关值要大于 MDMCTRL1 寄存器里设置的相关阈值。 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	定义行为或频率偏移平均过滤器。 0: 帧引导序列匹配之后，锁定平均电平。搜索下一个帧时重启频率偏移校准。 1: 连续更新平均电平。
4: 1	PREAMBLE_LENGTH[3: 0]	0010	R/W	在 TX 模式下，SFD 之前，要发送的帧引导序列字节数（2 个零符号），编码步长为 2。复位值为 2，适应 IEEE 802.15.4。 0000: 2 个前导 0 字节 0001: 3 个前导 0 字节 0010: 4 个前导 0 字节 ... 1111: 17 个前导 0 字节
0	TX_FILTER	1	R/W	定义使用的 TX 过滤器类型。IEEE 802.15.4 标准定义为普通 TX 过滤器。可以使用其它的过滤器以降低带外辐射。 0: 普通 TX 过滤器 1: 使能其它的过滤器

MDMCTRL1 (0x61A9) —控制调制解调器

位	名称	复位	读/写	功能
7: 6	—	00	R0	读为 0
5	CORR_THR_SFD	0	R/W	定义 SFD 检测的要求：

				0: 帧引导序列的某个零符号的相关值必须大于相关阈值 1: 帧引导序列的某个零符号的相关值以及 SFD 里的符号都必须大于相关阈值。
4: 0	CORR_THR[4: 0]	0x14	R/W	在 SFD 搜寻之前要求的解调器的相关阈值。 阈值调整接收机如何同步来自无线模块的数据。如果阈值设置得太低，同步会很容易发现噪音。如果设置得太高，灵敏度会下降，但是同步不太可能发现噪音。 结合 DEM_NUM_ZEROS，可以对系统进行调节，使灵敏度变高而同步噪音较少。

FREQEST (0x61AA) —RF 频率偏移估计值

位	名称	复位	读/写	功能
7: 6	FREQEST[7: 0]	0x00	R	有符号的二进制补码。包括载波和接收机 LO 之前的频率偏移估计值。偏移频率为 FREQEST × 7800Hz。DEM_AVG_MODE 控制何时更新该估计值。如果 DEM_AVG_MODE=0，直到找到同步才更新该估计值。然后频率偏移估计值被冻结，直到接收帧结束。如果 DEM_AVG_MODE=1，只要一使能解调器就更新该估计值。要计算正确的值，必须使用一个偏移量 (FREQEST_offset)，可以在数据手册里找到该偏移量。实际 FREQEST 值=FREQEST—FREQEST_offset。

RXCTRL (0x61AB) —调节接收部分

位	名称	复位	读/写	功能
7: 6	—	00	R0	保留。读为 0
5: 4	GBIAS_LNA2_REF[1: 0]	11	R/W	调整前段 LNA2/混频器 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	调整前段 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	控制接收机混频器输出电流。设置的值越大，电流越大。

FSCTRL (0x61AC) —调整频率合成器

位	名称	复位	读/写	功能
7: 6	PRE_CURRENT[1: 0]	01	R/W	预分频器电流设置
5: 4	LODIV_BUF_CURRENT_TX[1: 0]	01	R/W	调整混频器和 PA 缓冲器的电流。当 TX_ACTIVE=1 时使用。
3: 2	LODIV_VUF_CURRENT_RX[1: 0]	10	R/W	调整混频器和 PA 缓冲器的电流。当 TX_ACTIVE=0 时使用。
1: 0	LODIV_CURRENT[1: 0]	10	R/W	调节分频器电流，除了混频器和 PA 缓冲器。

FSCAL0 (0x61AD) —调节频率校准

位	名称	复位	读/写	功能
7	VCO_CURR_COMP_EN_OV	0	R/W	强制 VCO 里的电流比较器。该信号与来自校准模

				块的信号相“或”。
6	CHP_DISABLE	0	R/W	通过屏蔽来自相位检测器的向上和向下脉冲，设置为手动禁止电荷泵。
5: 2	CHP_CURRENT[3: 0]	1001	R*/W	数位向量定义呈指数级的电荷泵输出电流 如果 ffc_bw_boost=0，读取值为存储在 CHP_CURRENT 里的值。如果 ffc_bw_boost=1，读 取值为 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，定义合成器升压模式 00: 无 BW_boosting 01: 校准期间 BW_boost 为高并且约 30us 进入结算 10: BW_boost 总是打开/为高 11: 保留

FSCAL1 (0x61AE) — 调节频率校准

位	名称	复位	读/写	功能
7: 2	—	0001010	R/W0	保留
1: 0	VCO_CURR[1: 0]	01	R/W	定义 VCO 内核的电流。设置校准的电流和 VCO 电 流之前的乘法器。要获得使用的最佳值，请见 19.15.1 小节的表 19-6。

FSCAL2 (0x61AF) — 调节频率校准

位	名称	复位	读/写	功能
7	—	0	R0	保留。读为 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 电容数据设置。在校准期间编程。如果 VCO_CAPARR_OE=1，覆盖校准结果。

FSCAL3 (0x61B0) — 调节频率校准

位	名称	复位	读/写	功能
7	—	0	R0	保留。读为 0
6	VCO_DAC_EN_OV	0	R/W	为 1 时使能 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	用于校准的 cap_array 校准部分的校准精度设置 00: 80 个晶体振荡器周期 01: 100 个晶体振荡器周期 10: 125 个晶体振荡器周期 11: 250 个晶体振荡器周期

AGCCTRL0 (0x61B1) — AGC 动态范围控制

位	名称	复位	读/写	功能
7	—	0	R0	保留。读为 0
6	AGC_DR_XTND_EN	1	R/W	0: AGC 不执行 AAF 衰减调整

				1: AGC 调整 AAF 的增益, 为接收机获得额外的动态范围
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阈值, 在前端使用另外的衰减。该阈值应设置为大于 0x0C。 该功能由 ADC_DR_XTND_EN 使能。

AGCCTRL1 (0x61B2) —AGC 参考电平

位	名称	复位	读/写	功能
7: 6	—	00	R0	保留。读为 0
5: 0	AGC_REF[5: 0]	01 0001	R/W	用于 AGC 控制循环的目标值, 步长为 1dB。使用的最佳值请见 19.15.1 小节的表 19-6。

AGCCTRL2 (0x61B3) —AGC 增益覆盖

位	名称	复位	读/写	功能
7: 6	LNA_CURRENT[1: 0]	00	R*/W	给 LNA1 的覆盖值。只有在 LNA_CURRENT_OE=1 时使用。读取时, 该寄存器返回当前使用的增益设置。 00: 0dB 增益(参考电平) 01: 3dB 增益 10: 保留 11: 6dB 增益
5: 3	LNA2_CURRENT[2: 0]	000	R*/W	给 LNA2 的覆盖值。只有在 LNA_CURRENT_OE=1 时使用。读取时, 该寄存器返回当前使用的增益设置。 000: 0dB 增益(参考电平) 001: 3dB 增益 010: 6dB 增益 011: 9dB 增益 100: 12dB 增益 101: 15dB 增益 110: 18dB 增益 111: 21dB 增益
2: 1	LNA3_CURRENT[1: 0]	00	R*/W	给 LNA3 的覆盖值。只有在 LNA_CURRENT_OE=1 时使用。读取时, 该寄存器返回当前使用的增益设置。 00: 0dB 增益(参考电平) 01: 3dB 增益 10: 6dB 增益 11: 9dB 增益
0	LNA_CURRENT_OE	0	R/W	以存储在 RFR 里的值覆盖 AGC LNA 的当前设置。

AGCCTRL3 (0x61B4) —AGC 控制

位	名称	复位	读/写	功能
7	—	0	R0	保留。读为 0

6: 5	AGC_SETTLE_WAIT[2: 0]	01	R/W	增益变化后，AGC 等待模拟增益调整的时间。在此期间，AGC 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	AFF_RP[1: 0]	11	R*/W	当 AAF_RP_OE=1 时，覆盖 AGC 控制信号到 AAF。 读取时，返回使用的信号到 AAF。 00: AAF 里 9dB 的衰减 01: AAF 里 6dB 的衰减 10: AAF 里 3dB 的衰减 11: AAF 里 0dB 的衰减（参考电平）
0	AAF_RP_OE	0	R/W	以存储在 AAF_RP 里的值覆盖 AGC AAF 控制信号。

ADCTEST0 (0x61B5) —ADC 调节

位	名称	复位	读/写	功能
7: 6	ADC_VREF_ADJ[1: 0]	00	R/W	用于测试/调试的数字转换器阈值控制
5: 4	ADC_QUANT_ADJ[1: 0]	01	R/W	用于测试/调试的数字转换器阈值控制
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 调节

位	名称	复位	读/写	功能
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 电容值
1: 0	ADC_C3_ADJ[1: 0]	10	R/W	用于调整 ADC 电容值

ADCTEST2 (0x61B7) —ADC 调节

位	名称	复位	读/写	功能
7	—	0	R0	保留。读为 0
6: 5	ADC_TEST_MODE			测试模式，使能从解调器输出 ADC 数据。如果它使能，原始 ADC 数据在 GPIO 引脚上时钟输出 00: 测试模式禁用 01: 来自 I 和 Q ADC 的数据都输出，数据率 76MHz 10: 来自 I ADC 的数据输出。2 个 ADC 样本一组，数据率 38MHz 11: 来自 Q ADC 的数据输出。2 个 ADC 样本一组，数据率 38MHz

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	前馈调节
0	ADC_DAC_ROT	1	R/W	DAC DWA 方案控制 0: DWA (加扰) 禁止 1: DWA 使能

MDMTEST0 (0x61B8) — 调制解调器的测试寄存器

位	名称	复位	读/写	功能
7: 4	TX_TONE[3: 0]	0111	R/W	<p>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。步长大小由 TX_TONE 控制。如果 MDMTEST1.MODE_IF 是 0，基带音和调制数据重叠，有效给出中频调制。如果 MDMTEST1.MOD_IF 是 1，只发送基带音。</p> <p>0000: -6MHz 0001: -4MHz 0010: -3MHz 0011: -2MHz 0100: -1MHz 0101: -500kHz 0110: -4kHz 0111: 0 1000: 4kHz 1001: 500kHz 1010: 1MHz 1011: 2MHz 1100: 3MHz 1101: 4MHz 1110: 6MHz 其它: 保留</p>
3: 2	DC_WIN_SIZE[1: 0]	01	R/W	<p>控制直流移动中使用的积累转储过滤器每次转储之间被累加的样本数量</p> <p>00: 32 个样本 01: 64 个样本 10: 128 个样本 11: 256 个样本</p>
1: 0	DC_BLOCK_MODE[1: 0]	01	R/W	<p>选择操作模式:</p> <p>00: 到直流拦截器的输入信号不经过任何移除直流的尝试就传递到输出</p> <p>01: 使能直流取消。一般操作。</p> <p>10: 发现同步时冻结直流评估。搜索下一个帧时再次启动评估直流</p> <p>11: 保留</p>

MDMTEST1 (0x61B9) — 调制解调器的测试寄存器

位	名称	复位	读/写	功能
7: 5	—	000	R0	保留。读为 0
4	MOD_IF	0	R/W	0: 在一个由 MDMTEST0.TX_TONE 设置的中频进行调制 1: 发送一个带有 MDMTEST0.TX_TONE 设置的频率的基带音
3	RAMP_AMP	1	R/W	0: 禁止 DAC 输出幅度的修整斜坡 1: 在启动和完成期间使能 DAC 输出幅度的修正斜坡
2	RFC_SNIFF_EN	0	R/W	0: 数据包分析仪模块禁止 1: 数据包分析仪模块使能。接收和发送的数据可以在 GPIO 引脚上观察到
1	MODULATION_MODE	0	R/W	为 RX/TX 设置一种 RF 调制模式 0: IEEE 802.15.4 标准模式 1: 反相, 不符合 IEEE 标准
0	RESERVED	0	R/W	保留。不写

DACTEST0 (0x61BA) — DAC 覆盖值

位	名称	复位	读/写	功能
7	—	0	R0	保留。读为 0
6: 0	DAC_Q_O[6: 0]	000 0000	R/W	当 DAC_SRC=001 时, 正交相位信号分支 (Q-branch) DAC 覆盖值。 如果 DAC_SRC 设置为 ADC 数据、CORDIC 值、信道过滤数据, 那么 DAC_Q_O 控制实际复用到 DAC 的有问题的部分, 如下表所述。 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值, 那么 DAC 输出只能是 0 (最小值)。

DACTEST1 (0x61BB) — DAC 覆盖值

位	名称	复位	读/写	功能
7	—	0	R0	保留。读为 0
6: 0	DAC_I_O[6: 0]	000 0000	R/W	当 DAC_SRC=001 时, 同相信号分支 (I-branch) DAC 覆盖值。 如果 DAC_SRC 设置为 ADC 数据、CORDIC 值、信道过滤数据, 那么 DAC_I_O 控制实际复用到 DAC 的有问题的部分, 如下表所述。 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值，那么 DAC 输出只能是 0（最小值）。
--	--	--	--	---------------------------------

DACTEST2 (0x61BC) —DAC 测试设置

位	名称	复位	读/写	功能
7: 3	—	0010 1	R0	保留
2: 0	DAC_SRC[2: 0]	000	R/W	DAC_SRC 根据下面的设置来选择 TX 的 DAC 数据源： 000: 正常操作（来自调制器） 001: DAC_I_O 和 DAC_Q_O 覆盖值 010: 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，信道和直流过滤，大小由 DAC_I_O 和 DAC_Q_O 控制 100: 坐标旋转数字计算（Cordic）巨量输出和前端增益输出， 大小由 DAC_I_O 和 DAC_Q_O 控制 101: 同相 DAC 上的 RSSI 同相输出 111: 保留

ATEST (0x61BD) —模拟测试控制

位	名称	复位	读/写	功能
7: 6	—	00	R0	保留。读为 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 小节的 TR0 寄存器描述） 其它值：保留

RFRND (0x61A7) —随机数据

位	名称	复位	读/写	功能
7: 2	—	0000 00	R0	保留。读为 0
1	QRND	0	R/W	来自接收机正交相位通道的随机位
0	IRND	0	R/W	来自接收机同相通道的随机位

PTEST0 (0x61BE) —覆盖掉电寄存器

位	名称	复位	读/写	功能
7	PRE_PD	0	R/W	预分频器掉电信号。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	模数转换器掉电信号。当 PD_OVERRIDE=1
4	DAC_PD	0	R/W	数模转换器掉电信号。当 PD_OVERRIDE=1
3: 2	LNA_PD[1: 0]	0	R/W	低噪声放大器掉电信号。定义 LNA/混频器掉电模式。 00: 上电 01: LNA 关闭，混频器/稳压器开启 10: LNA/混频器关闭，稳压器开启 11: 掉电

				当 PD_OVERRIDE=1
1	TXMIX_PD	0	R/W	发送混频器掉电信号。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	名称	复位	读/写	功能
7: 4	—	0000	R0	保留。读为 0
3	PD_OVERRIDE	0	R/W	各种模块的覆盖使能/禁止。仅用于调试和测试。不可能覆盖硬编码的 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	电压控制振荡器掉电信号。当 PD_OVERRIDE=1
0	LODIV_PD	0	R/W	LO 掉电信号。当 PD_OVERRIDE=1

RFC_OBS_CTRL0 (0x61EB) —RF 观察复用器控制

位	名称	复位	读/写	功能
7	—	0	R0	保留。读为 0
6	RFC_OBS_POL0	0	R/W	RFC_OBS_MUX0 选择的信号与该位“异或”。
5: 0	RFC_OBS_MUX0	00 0000	R/W	<p>控制来自 RF 内核的那个观察信号被多路复用输出到 rfc_obs_sigs(0)。</p> <p>00 0000: 0—常量值</p> <p>00 0001: 1—常量值</p> <p>00 1000: rfc_sniff_data—来自数据包分析仪的数据。在 sniff_clk 的上升沿采样数据</p> <p>00 1001: rfc_sniff_clk—数据包分析仪数据的 250kHz 时钟</p> <p>00 1100: rss_i_valid—如果 RSSI 值自 RX 启动以来至少被更新了一次，引脚为高。离开 RX 时清除。</p> <p>00 1101: demod_cc—空闲信道评估。关于如何配置该信号的动作详见 FSMSTAT1 寄存器。</p> <p>00 1110: sampled_cc—来自解调器的 CCA 位的采样版本。只要发出了一个 SSAMPLECCA 或 STXONCCA 选通，该值就被更新。</p> <p>00 1111: sfd_sync—如果已经接收或发送了一个 SFD，引脚为高。分别在离开 RX/TX 时清除。不要与 SFD 异常相混淆。</p> <p>01 0000: tx_active—指示 FFCTRL 处于 TX 状态之一。高有效。注意：该信号可能有瑕疵，因为它没有输出触发器，是基于 FFCTRL FSM 的当前状态寄存器。</p> <p>01 0001: rx_active—指示 FFCTRL 处于 RX 状态之一。高有效。注意：该信号可能有瑕疵，因为它没有输出触发器，是基于 FFCTRL FSM 的当前状态寄存器。</p>

				<p>01 0010: ffctrl_fifo—如果 RXFIFO 里有一个或多个字节，引脚为高。RXFIFO 溢出期间为低。</p> <p>01 0011: ffctrl_fifop—如果 RXFIFO 里的字节数超过了编程阈值，或者 RXFIFO 里至少有一个完整帧，引脚为高。RXFIFO 溢出期间也为高。不要与 FIFOP 异常相混淆。</p> <p>01 0100: packet_done—接收了一个完整的帧。即接收了由长度域设置的字节数。</p> <p>01 0110: rfc_xor_rand_i_q—I 和 Q 随机输出之间“异或”。在 8MHz 更新。</p> <p>01 0111: rfc_rand_q—随机数据从接收机正交相位通道输出。在 8MHz 更新。</p> <p>01 1000: rfc_rand_i—随机数据从接收机同相通道输出。在 8MHz 更新。</p> <p>01 1001: lock_status—当 PLL 处于锁定状态为 1，否则为 0</p> <p>10 1000: pa_pd—功率放大器掉电信号</p> <p>10 1010: lna_pd—LNA 掉电信号</p> <p>其它：保留</p>
--	--	--	--	--

RFC_OBS_CTRL1 (0x61EC) —RF 观察复用器控制

位	名称	复位	读/写	功能
7	—	0	R0	保留。读为 0
6	RFC_OBS_POL1	0	R/W	RFC_OBS_MUX1 选择的信号与该位“异或”。
5: 0	RFC_OBS_MUX1	00 0000	R/W	控制来自 RF 内核的那个观察信号被多路复用输出到 rfc_obs_sigs(1)。详见 RFC_OBS_CTRL0 的描述。

RFC_OBS_CTRL2 (0x61ED) —RF 观察复用器控制

位	名称	复位	读/写	功能
7	—	0	R0	保留。读为 0
6	RFC_OBS_POL2	0	R/W	RFC_OBS_MUX2 选择的信号与该位“异或”。
5: 0	RFC_OBS_MUX2	00 0000	R/W	控制来自 RF 内核的那个观察信号被多路复用输出到 rfc_obs_sigs(2)。详见 RFC_OBS_CTRL0 的描述。

TXFILTCFG (0x61FA) —TX 过滤器配置

位	名称	复位	读/写	功能
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。减少杂散发射接近信号。使能的最佳值见 19.15.1 节的表 19-6。

20. 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）。

稳压器在功耗模式 PM2 和 PM3（见第 4 章）是禁用的。如果禁用了稳压器，当供电电源为未调整的 2V 到 3.6V 时，寄存器和 RAM 的内容保留。

注意：稳压器不能为外部电路提供电源。

21 可用的软件

本章介绍与 CC253x 系列相关的各种可用软件解决方案。它们可以从 TI 网站 www.ti.com/lprf 上免费下载，用于 TI 的 LPRF 设备。

21.1 评估软件 SmartRFTM 软件 (www.ti.com/smartrfstudio)

德州仪器的 SmartRF Studio 可用于评估无线模块的性能和功能，对探索和获得 RF-IC 产品的有关知识很有帮助。该软件帮助无线系统的设计人员在设计过程的早期阶段轻松地评估 RF-IC。它对于配置数据的产生和找到最佳外部元件值特别有用。

SmartRF Studio 软件运行于 Microsoft™ Windows™95/98 和 Microsoft Windows NT/2000/XP。可以从德州仪器网页 www.ti.com/smartrfstudio(<http://www.ti.com/litv/zip/swrc046m>)免费下载该软件。

功能

- 链路测试。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
- 一台计算机支持多达 8 个 USB 设备
- 首选寄存器设置的普通查看
- 可以对各个寄存器进行读写。每个寄存器给出详细信息
- 从文件中保存/打开配置数据
- 从文件中保存/打开寄存器设置
- 从文本文件中导出/导入寄存器值
- 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电子设备进行命令通信。但是，基于双向 RF 通信的射频（RF）远程控制可以进行非视距操作，并提供了更先进的功能。

ZigBee 联盟和 RF4CE 协会 (<http://www.zigbee.org/rf4ce>) 最近签署了用于消费电子的 ZigBee 射频（RF4CE），RF4CE 专门用于范围广泛的远程控制音频/视频消费类电子产品，比如电视和机顶盒。ZigBee RF4CE 承诺：

- 更丰富的通信和更高的可靠性
- 增强功能和灵活性
- 互操作性
- 无视线障碍

德州仪器为 ZigBee RF4CE 标准专门推出了 RemoTI 网络协议。它是一个完整的解决方案，为 TI 低功耗 RF 产品系列提供硬件和软件支持。有了 RemoTI 网络协议，我们可以提供：

- 业界领先的 RF4CE 兼容栈，具有如下特性：可互操作的 CERC 规范支持、简单 API、

易于理解的示例应用程序代码、完整的开发工具包和参考设计等等。

- 运行于行业内最佳的 IEEE 802.15.4 标准片上系统 CC2530，具有出色的 RF 共存和射频性能。4 种灵活的功耗模式包括最低电流消耗掉电模式，用于要求长电池寿命、低占空比的应用。
- 广泛的全球支持和工具，确保基于 ZigBee RF4CE 的产品开发简单、快速，并且以最低的成本完成。
- RemoTI 网络协议是一个联盟最高业内水平平台，也就是说，它作为一个测试标准，用于测试其它 ZigBee RF4CE 标准的实现。

有关 TI 的 RemoTI 网络协议的更多信息，请见德州仪器 RemoTI 网络协议网页 www.ti.com/remoti（查看信息，下载等。）

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 1GHz, 2.4GHz 和 IEEE 802.15.4 射频芯片），针对于简单、小型的 RF 网络。这一开放源码的软件是构建一个运行电池设备的网络（使用 TI 低功耗 RF 片上系统（SoC）的良好开端。SimpliciTI 网络协议的目的是在一些开箱即用的 TI RF 平台上便于实施和部署。它提供了一些示例应用程序。

主要应用

- 报警和安全：感应传感器、光传感器、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、电表
- 有源 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：
 - 设备到设备的直接通信
 - 简单星型结构，接入点能存储和转发到终端设备
 - 范围扩展，增加到 4 跳
- 简单：使用 5 个命令的 API
- 低数据率和低占空比
- 易于使用

有关 SimpliciTI 网络协议的更多信息，请见德州仪器 SimpliciTI 网络协议网页 www.ti.com/simpliciti（查看信息，下载等。）

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。

以下情况可以使用 TIMAC：

- 需要一个无线点到点或点到多点的解决方案；例如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
- 用电池供电和/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 100kbps 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系统
- 多重平台
- 便于应用开发

TIMAC 软件栈经过认证，符合 IEEE 802.15.4 标准。TIMAC 软件免费分配目标代码。使用 TIMAC 软件不需要版税。

有关 TIMAC 软件的更多信息，请见德州仪器 TIMAC 网页 www.ti.com/timac（查看信息，下载等。）

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 IEEE 802.15.4 产品和平台越来越多的产品系列。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 2 个 ZigBee 应用规范的执行—智能能源和家居自动化。用户可以轻易实现其它应用规范。

Z-Stack 软件主要特征包括：

- 完全兼容 ZigBee 和 ZigBee PRO 功能集
- 丰富的示例应用程序，包括对 ZigBee 智能能源和 ZigBee 家居自动化规范的支持
- 支持空中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，提高了接收灵敏度

Z-Stack 软件被 ZigBee 联盟评为联盟最高业内水平，适用于 ZigBee 和 ZigBee PRO 栈规范，被全世界的 ZigBee 开发人员使用。

Z-Stack 软件特别适合于：

- 智能能源（AMI）
- 家居自动化
- 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
- 报警与安全
- 资产追踪
- 需要互操作性的应用

有关 Z-Stack 软件的更多信息，请见德州仪器 Z-Stack 软件网页 www.ti.com/z-stack（查看信息，下载等。）

A 缩略语

AAF	Anti-Aliasing Filter	抗混叠过滤器
ADC	Analog to Digital Converter	模/数转换器
AES	Advanced Encryption Standard	高级加密标准
AGC	Automatical Gain Control	自动增益控制
ARIB	Association of Radio Industries and Businesses	无线电工商协会
BCD	Binary Coded Decimal	二进制编码的十进制
BER	Bit Error Rate	比特差错率（误码率）
BOD	Brown Out Detector	掉电检测
BOM	Bill of Materials	材料清单
CBC	Cipher Block Chaining	密码防护链
CBC-MAC	Cipher Block Chaining Message Authentication Code	密码防护链消息验证代码
CCA	Clear Channel Assessment	空闲信道评估
CCM	Counter Mode+CBC-MAC	计数器模式和密码防护链消息验证代码
CFB	Cipher FeedBack	密码反馈
CFR	Code of Federal Regulations	(美国) 联邦法规代码
CMRR	Common Mode Ratio Rejection	共模抑制比
CPU	Central Processing Unit	中央处理单元
CRC	Cyclic Redundancy Check	循环冗余校验
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance	载波监听多点接入-冲突避免
CSP	CSMA-CA Strobe Processor	CSMA-CA 选通处理器
CTR	Counter mode (encryption)	计数器模式（加密）
CW	Contention Window	竞争窗口（长度）
DAC	Digital to Analog Converter	数/模转换器
DC	Direct Current	直流
DMA	Direct Memory Access	直接存储器存取
DSM	Delta Sigma Modulator	Delta Sigma 调制器
DSSS	Direct Sequence Spread Spectrum	直接序列扩频
ECB	Electronic Code Book(Encryption)	电子编码加密
EM	Evaluation Module	评估模块
ENOB	Effective Number of bits	有效位数
ETSI	European Telecommunications Standards Institute	欧洲电信标准学会
EVM	Error Vector Magnitude	误差向量振幅
FCC	Federal Communications Commission	(美国) 联邦通信委员会
FCF	Frame Control Field	帧控制域
FCS	Frame Check Sequence	帧校验序列
FFCTRL	FIFO and Frame Control	先进先出和帧控制
FIFO	First In First Out	先进先出

GPIO	General purpose input output	通用输入输出
HF	High Frequency	高频
HSSD	High Speed Serial Data	高速串行数据
I/O	Input/Output	输入/输出
I/Q	In-phase/Quadrature-phase	同相/正交相位
IEEE	Institute of electrical and electronics Engineers	电气和电子工程师学会
IF	Intermediate Frequency	中频
IOC	I/O Controller	输入/输出控制
IRQ	Interrupt Request	中断请求
ISM	Industrial,Scientific and Medical	工业、科学和医学
ITU-T	International Telecommunication Union-Telecommunication Standardization Sector	国际电线联盟-电线标准化部
IV	Initialization Vector	初始化向量
KB	1024 bytes	1024 字节
kbps	kilo bits per second	千位/秒
LFSR	Linear Feedback Shift Register	线性反馈移位寄存器
LNA	Low-Noise Amplifier	低噪声放大器
LO	Local Oscillator	本地振荡器
LQI	Link Quality Indication	链路质量指示
LSB	Least Significant Bit/Byte	最低有效位/字节
MAC	Medium Access Control	媒体存取控制
MAC	Message Authentication Code	消息验证代码
MCU	Microcontroller Uint	微控制器单元
MFR	MAC Footer	MAC 帧尾
MHR	MAC Header	MAC 帧头
MIC	Message Integrity Code	消息完整性检测码
MISO	Master In Slave Out	主入从出
MOSI	Master Out Slave In	主出从入
MPDU	MAC Protocol Data Uint	MAC 协议数据单元
MSB	Most Significant Bit/Byte	最高有效位/字节
MSDU	MAC Service Data Unit	MAC 服务数据单元
MUX	Multiplexer	多路器
NA	Not Available	未用
NC	Not Connected	未连接
OFB	Output Feedback(Encryption)	输出反馈（加密）
O-QPSK	Offset-Quadrature Phase Shift Keying	偏移正交相移键控
PA	Power Amplifier	功率放大器
PC	Program Counter	程序计数器
PCB	Printed Circuit Board	印刷电路板
PER	Packet Error Rate	包差错率/误帧率
PHR	PHY Header	物理层首部
PHY	Physical Layer	物理层
PLL	Phase Locked Loop	锁相环

PM1,PM2,P	Power mode1,2 and 3	功耗模式 1、2 和 3
M3		
PMC	Power Management Controller	电源管理控制器
POR	Power On Reset	上电复位
PSDU	PHY Service Data Unit	PHY 服务数据单元
PWM	Pulse Width Modulation	脉宽调制
RAM	Random Access Memory	随机存取存储器
RBW	Resolution BandWidth	解析度带宽
RC	Resistor-Capacitor	电阻-电容
RCOSC	RC Oscillator	RC 振荡器
RF	Radio Frequency	无线射频
RSSI	Received Signal Strength Indication	接收信号强度指示
RTC	Real-Time Clock	实时时钟
RX	Receive	接收
SCK	Serial Clock	串行时钟
SFD	Start of Frame Delimiter	帧开始定界符
SFR	Special Function Register	特殊功能寄存器
SHR	Synchronization Header	同步头
SINAD	Signal-to-noise and distortion ratio	信号与噪声+失真比
SPI	Serial Peripheral Interface	串行外部设备接口
SRAM	Static Random Access Memory	静态随机存取存储器
ST	Sleep Timer	睡眠计时器
T/R	Tape and reel	卷带和轮盘
T/R	Transmit/Receive	发送/接收
THD	Total Harmonic Distortion	总谐波失真
TI	Texas Instruments	德州仪器
TX	Transmit	发送
UART	Universal Asynchronous Receiver/Transmitter	通用异步收发器
USART	Universal Synchronous/Asynchronous Receiver/Transmitter	通用同步/异步收发器
VCO	Voltage Controlled Oscillator	电压控制振荡器
VGA	Variable Gain Amplifier	可变增益放大器
WDT	WatchDog Timer	看门狗计时器
XOSC	Crystal Oscillator	晶体振荡器

B 参考文献

参考书目和其他有用的资料：

1. IEEE std. 802.15.4 - 2006: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)
<http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf>
2. CC2530 数据手册 ([SWRS081](#))