



dsPIC30F
系列参考手册
高性能
数字信号控制器

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。

代码保护功能处于持续发展。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 (Digital Millennium Copyright Act)》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc. 及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc. 的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和 / 或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 名称和徽标组合、Microchip 徽标、Accuron、dsPIC、KEELOQ、microID、MPLAB、PIC、PICmicro、PICSTART、PRO MATE、PowerSmart、rfPIC 和 SmartShunt 均为 Microchip Technology Incorporated 在美国及其他国家或地区的注册商标。

AmpLab、FilterLab、MigratableMemory、MXDEV、MXLAB、SEEVAL、SmartSensor 和 The Embedded Control Solutions Company 均为 Microchip Technology Incorporated 在美国的注册商标。

Analog-for-the-Digital Age、Application Maestro、CodeGuard、dsPICDEM、dsPICDEM.net、dsPICworks、ECAN、ECONOMONITOR、FanSense、FlexROM、fuzzzyLAB、In-Circuit Serial Programming、ICSP、ICEPIC、Linear Active Thermistor、Mindi、MiWi、MPASM、MPLIB、MPLINK、PICKit、PICDEM、PICDEM.net、PICLAB、PICKtail、PowerCal、PowerInfo、PowerMate、PowerTool、REAL ICE、rfLAB、rfPICDEM、Select Mode、Smart Serial、SmartTel、Total Endurance、UNI/O、WiperLock 和 ZENA 均为 Microchip Technology Incorporated 在美国及其他国家或地区的商标。

SQTP 是 Microchip Technology Incorporated 在美国的服务标记。

在此提及的所有其他商标均为各持有公司所有。

© 2006, Microchip Technology Incorporated. 版权所有。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
== ISO/TS 16949:2002 ==

Microchip 位于美国亚利桑那州 Chandler 和 Tempe、位于俄勒冈州 Gresham 及位于加利福尼亚州 Mountain View 的全球总部、设计中心和晶圆生产厂均通过了 ISO/TS-16949:2002 认证。公司在 PIC® 8 位单片机、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器及模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

目录

	页码
第 1 章 简介	1-1
简介	1-2
本手册所针对的产品	1-2
器件结构	1-3
开发支持	1-4
风格和符号约定	1-4
相关文档	1-6
版本历史	1-7
第 2 章 CPU	2-1
简介	2-2
编程模型	2-4
软件堆栈指针	2-8
CPU 寄存器描述	2-11
算术逻辑单元 (ALU)	2-17
DSP 引擎	2-18
除法支持	2-27
指令流类型	2-27
循环结构	2-30
地址寄存器相依性	2-35
寄存器映射	2-38
相关应用笔记	2-40
版本历史	2-41
第 3 章 数据存储器	3-1
简介	3-2
数据空间地址发生器单元 (AGU)	3-5
模寻址	3-7
位反转寻址	3-14
控制寄存器说明	3-18
相关应用笔记	3-23
版本历史	3-24
第 4 章 程序存储器	4-1
程序存储器地址映射	4-2
程序计数器	4-4
从程序存储器存取数据	4-4
来自数据空间的程序空间可视性	4-8
写程序存储器	4-10
PSV 代码示例	4-11
相关应用笔记	4-12
版本历史	4-13

目录

	页码
第 5 章 闪存和 EEPROM 编程	5-1
简介	5-2
表指令操作	5-2
控制寄存器	5-5
运行时自编程 (RTSP)	5-10
数据 EEPROM 编程	5-15
设计技巧	5-21
相关应用笔记	5-22
版本历史	5-23
第 6 章 复位中断	6-1
简介	6-2
不可屏蔽陷阱	6-6
中断处理时序	6-11
中断控制和状态寄存器	6-14
中断设置流程	6-42
设计技巧	6-44
相关应用笔记	6-45
版本历史	6-46
第 7 章 振荡器	7-1
简介	7-2
器件时钟和 MIPS	7-5
振荡器配置	7-6
振荡器控制寄存器——OSCCON 和 OSCTUN	7-13
主振荡器	7-20
晶体振荡器 / 陶瓷谐振器	7-22
为晶振、时钟模式、C1、C2 和 Rs 确定最佳的值	7-24
外部时钟输入	7-25
外部 RC 振荡器	7-26
锁相环 (PLL)	7-30
低功耗 32 kHz 晶体振荡器	7-31
振荡器起振定时器 (OST)	7-31
内部快速 RC 振荡器 (FRC)	7-31
内部低功耗 RC (LPRC) 振荡器	7-32
故障保护时钟监视器 (FSCM)	7-32
可编程振荡器后分频器	7-33
时钟切换工作原理	7-34
设计技巧	7-38
相关应用笔记	7-39
版本历史	7-40

目录

	页码
第 8 章 复位	8-1
简介	8-2
复位时的时钟源选择	8-5
POR: 上电复位	8-5
外部复位 (EXTR)	8-7
软件复位指令 (SWR)	8-7
看门狗超时复位 (WDTR)	8-7
欠压复位 (BOR)	8-8
使用 RCON 状态位	8-10
器件复位时间	8-11
器件起振时间曲线	8-13
特殊功能寄存器复位状态	8-16
设计技巧	8-17
相关应用笔记	8-18
版本历史	8-19
第 9 章 低压检测 (LVD)	9-1
简介	9-2
LVD 工作原理	9-5
设计技巧	9-6
相关应用笔记	9-7
版本历史	9-8
第 10 章 看门狗定时器和低功耗模式	10-1
简介	10-2
低功耗模式	10-2
休眠模式	10-2
空闲模式	10-4
低功耗指令与中断同时发生	10-5
看门狗定时器	10-6
外设模块禁止 (PMD) 寄存器	10-9
设计技巧	10-10
相关应用笔记	10-11
版本历史	10-12
第 11 章 I/O 端口	11-1
简介	11-2
I/O 端口控制寄存器	11-3
外设复用	11-4
端口描述	11-6
电平变化通知 (CN) 引脚	11-7
休眠和空闲模式下的 CN 工作	11-8
相关应用笔记	11-11
版本历史	11-12

目录

	页码
第 12 章 定时器	12-1
简介	12-2
不同的定时器	12-3
控制寄存器	12-6
工作模式	12-9
定时器预分频器	12-14
定时器中断	12-14
读写 16 位定时器模块寄存器	12-15
低功耗 32 kHz 晶振输入	12-15
32 位定时器配置	12-16
32 位定时器的工作模式	12-18
读写 32 位定时器	12-21
低功耗状态下的定时器工作	12-21
使用定时器模块的外设	12-22
设计技巧	12-24
相关应用笔记	12-25
版本历史	12-26
第 13 章 输入捕捉	13-1
简介	13-2
输入捕捉寄存器	13-3
定时器选择	13-4
输入捕捉事件模式	13-4
捕获缓冲器的操作	13-8
输入捕捉中断	13-9
UART 自动波特率支持	13-9
低功耗状态下的输入捕捉工作	13-10
I/O 引脚控制	13-10
与输入捕捉模块相关的特殊功能寄存器	13-11
设计技巧	13-12
相关应用笔记	13-13
版本历史	13-14
第 14 章 输出比较	14-1
简介	14-2
输出比较寄存器	14-3
工作模式	14-4
低功耗状态下的输出比较工作	14-23
I/O 引脚控制	14-23
设计技巧	14-26
相关应用笔记	14-27
版本历史	14-28

目录

	页码
第 15 章 电机控制 PWM	15-1
简介	15-2
控制寄存器	15-4
PWM 时基	15-16
PWM 占空比比较单元	15-20
互补 PWM 输出模式	15-26
死区时间控制	15-27
独立 PWM 输出模式	15-30
PWM 输出改写	15-31
PWM 输出和极性控制	15-34
PWM 故障引脚	15-34
PWM 更新锁定	15-37
PWM 特殊事件触发器	15-38
器件低功耗模式下的工作	15-38
用于器件仿真的特殊功能	15-39
相关应用笔记	15-42
版本历史	15-43
第 16 章 正交编码器接口 (QEI)	16-1
模块简介	16-2
控制和状态寄存器	16-4
可编程数字噪声滤波器	16-9
正交解码器	16-10
16 位向上 / 向下位置计数器	16-12
QEI 用作备用 16 位定时器 / 计数器	16-16
正交编码器接口中断	16-17
I/O 引脚控制	16-18
低功耗模式下的 QEI 工作	16-19
复位的影响	16-19
设计技巧	16-21
相关应用笔记	16-22
版本历史	16-23

目录

	页码
第 17 章 10 位 A/D 转换器	17-1
简介	17-2
控制寄存器	17-4
A/D 转换结果缓冲器	17-4
A/D 转换术语和转换过程	17-11
A/D 模块配置	17-13
参考电压源的选择	17-13
A/D 转换时钟的选择	17-13
采样模拟输入的选择	17-14
模块使能	17-16
采样 / 转换过程的说明	17-16
如何开始采样	17-17
如何停止采样和开始转换	17-18
采样 / 转换工作的控制	17-29
如何将转换结果写入缓冲器的说明	17-30
转换过程示例	17-31
A/D 采样要求	17-45
读取 A/D 转换结果缓冲器	17-46
传递函数	17-47
A/D 转换的精度 / 误差	17-47
连接注意事项	17-47
初始化	17-48
A/D 转换速度	17-49
在休眠和空闲模式下工作	17-55
复位的影响	17-55
与 10 位 A/D 转换器相关的特殊功能寄存器	17-56
设计技巧	17-57
相关应用笔记	17-58
版本历史	17-59

目录

	页码
第 18 章 12 位 A/D 转换器	18-1
简介	18-2
控制寄存器	18-4
A/D 转换结果缓冲器	18-4
A/D 转换术语和转换过程	18-10
A/D 模块配置	18-11
参考电压源的选择	18-11
A/D 转换时钟的选择	18-12
采样模拟输入的选择	18-12
模块使能	18-14
如何开始采样	18-14
如何停止采样和开始转换	18-14
采样 / 转换工作的控制	18-19
如何将转换结果写入缓冲器的说明	18-19
转换过程示例	18-21
A/D 采样要求	18-26
读取 A/D 转换结果缓冲器	18-27
传递函数	18-28
A/D 转换的精度 / 误差	18-28
连接注意事项	18-28
初始化	18-29
A/D 转换速度	18-30
在休眠和空闲模式下工作	18-33
复位的影响	18-33
与 12 位 A/D 转换器相关的特殊功能寄存器	18-34
设计技巧	18-35
相关应用笔记	18-36
版本历史	18-37
第 19 章 UART	19-1
简介	19-2
控制寄存器	19-3
UART 波特率发生器 (BRG)	19-8
UART 配置	19-10
UART 发送器	19-11
UART 接收器	19-14
使用 UART 进行 9 位通信	19-18
接收中止字符	19-19
初始化	19-20
UART 的其他特性	19-21
UART 在 CPU 休眠和空闲模式下的工作	19-21
与 UART 模块相关的寄存器	19-22
设计技巧	19-23
相关应用笔记	19-24
版本历史	19-25

目录

	页码
第 20 章 串行外设接口 (SPITM)	20-1
简介	20-2
状态和控制寄存器	20-4
工作模式	20-7
SPI 主控模式时钟频率	20-19
低功耗模式下的工作	20-20
与 SPI 模块相关的特殊功能寄存器	20-22
相关应用笔记	20-23
版本历史	20-24
第 21 章 I2CTM 模块	21-1
概述	21-2
I ² C 总线特性	21-4
控制和状态寄存器	21-7
使能 I ² C 操作	21-13
作为主器件在单主机环境下通信	21-15
作为主器件在多主机环境下通信	21-29
作为从器件通信	21-32
I ² C 总线的连接注意事项	21-47
在 PWRSAV 指令执行期间的模块操作	21-49
复位的影响	21-49
设计技巧	21-50
相关应用笔记	21-51
版本历史	21-52
第 22 章 数据转换器接口 (DCI)	22-1
简介	22-2
控制寄存器说明	22-2
编解码器接口基础与术语	22-8
DCI 工作原理	22-10
使用 DCI 模块	22-17
低功耗模式下的工作	22-28
与 DCI 相关的寄存器	22-28
设计技巧	22-30
相关应用笔记	22-31
版本历史	22-32

目录

	页码
第 23 章 CAN 模块	23-1
简介	23-2
CAN 模块的控制寄存器	23-2
CAN 模块特点	23-28
CAN 模块的实现	23-29
CAN 模块工作模式	23-36
报文接收	23-39
发送	23-49
错误检测	23-58
CAN 波特率	23-60
中断	23-64
时间标记	23-65
CAN 模块 I/O	23-65
CPU 低功耗模式下的工作	23-66
CAN 协议综述	23-68
相关应用笔记	23-72
版本历史	23-73
第 24 章 器件配置	24-1
简介	24-2
器件配置寄存器	24-2
配置位说明	24-6
器件标识寄存器	24-7
相关应用笔记	24-8
版本历史	24-9
第 25 章 开发工具支持	25-1
简介	25-2
Microchip 硬件和语言工具	25-2
第三方硬件 / 软件工具和应用程序库	25-6
dsPIC30F 硬件开发板	25-11
相关应用笔记	25-15
版本历史	25-16
第 26 章 附录	26-1

目录

页码

注:

第 1 章 简介

目录

本章包括下列主题：

1.1	简介	1-2
1.2	本手册所针对的产品	1-2
1.3	器件结构	1-3
1.4	开发支持	1-4
1.5	风格和符号约定	1-4
1.6	相关文档	1-6
1.7	版本历史	1-7

1.1 简介

Microchip 是单片机和模拟半导体的领先供应商。公司致力于生产满足嵌入式控制市场需求的产品。我们是以下产品的领先供应商：

- 8 位通用单片机（PIC® MCU）
- dsPIC30F 16 位单片机
- 专用和标准的非易失性存储器件
- 安全器件（KEELOQ®）
- 专用标准产品

如欲获得您所感兴趣的产品列表，请申请一份 Microchip 产品线目录卡。该文献可以从当地的 Microchip 销售办事处获得，或者从 Microchip 的网站（www.microchip.com）下载。

1.2 本手册所针对的产品

PIC 和 dsPIC30F 器件根据其指令字长度和数据路径长度来划分。现有的器件系列包括：

1. 低档： 12 位指令字长， 8 位数据路径
2. 中档： 14 位指令字长， 8 位数据路径
3. 高档： 16 位指令字长， 8 位数据路径
4. 增强型： 16 位指令字长， 8 位数据路径
5. dsPIC30F： 24 位指令字长， 16 位数据路径

本手册描述了 dsPIC30F 16 位 MCU 系列器件。

本手册阐明了 dsPIC30F MCU 系列的架构和外设模块的工作原理，但并不涉及每个器件工作的具体情况。如果需要某个器件的详细信息，请用户参见该器件的数据手册。数据手册中包含的信息如下：

- 器件存储器映射
- 器件引脚排列和封装细节
- 器件电气规范
- 器件上包含的外设列表

本手册给出了代码示例。这些例子有时需要针对特定器件编写，而不是普遍针对系列编写的（尽管对于大多数其他器件来说，它们都是可行的）。对于寄存器文件映射有所不同的器件，可能需要对代码进行一些修改。

1.3 器件结构

dsPIC30F 器件的每个部分都划分为以下三个部分：

1. CPU 内核
2. 系统集成
3. 外设

1.3.1 CPU 内核

CPU 内核是器件运行所必需的基本部分。手册中与 CPU 内核有关的章节包括：

1. CPU
2. 数据存储器
3. 程序存储器
4. DSP 引擎
5. 中断

1.3.2 系统集成

系统集成功能有助于：

- 降低系统成本
- 提高系统可靠性
- 提高设计灵活性

手册的以下章节对 dsPIC30F 系统集成功能进行了探讨：

1. 振荡器
2. 复位
3. 低电压检测
4. 看门狗定时器和低功耗模式
5. 闪存和 EEPROM 编程
6. 器件配置

1.3.3 外设

dsPIC30F 具备许多允许器件与外界交换信息的外设。本手册中讨论的外设包括：

1. I/O 端口
2. 定时器
3. 输入捕捉模块
4. 输出比较模块
5. 正交编码器接口 (QEI)
6. 10 位 AD 转换器
7. 12 位 AD 转换器
8. UART 模块
9. SPI™ 模块
10. I²C™ 模块
11. 数据转换器接口 (DCI) 模块
12. CAN 模块

1.3.4 存储器技术

撰写本书时，所有 dsPIC30F 器件都使用闪存程序存储器技术。闪存程序存储器是电可擦除和可编程的。

1.4 开发支持

Microchip 提供了各种开发工具，用户可以高效地开发和调试应用代码。Microchip 的开发工具可以分为四类：

1. 代码生成
2. 硬件 / 软件调试
3. 器件编程器
4. 产品评估板

第 25 章 “开发工具支持” 中对每种 Microchip 开发工具进行了完整描述。随着新工具的开发，最新的产品简介和用户指南可通过 Microchip 网站 (www.microchip.com) 或从当地 Microchip 销售办事处获得。

Microchip 还提供其他辅助工具以缩短开发周期。包括：

- 应用笔记
- 参考设计
- Microchip 网站
- 当地销售办事处提供的现场应用支持
- 公司的技术支持热线

Microchip 网站还列出了其他有参考价值的网站。

1.5 风格和符号约定

本文档采用了一些特定的风格和字体格式约定。大多数格式约定对于需要强调的文字作了区分。MCU 行业有很多符号和未约定的字的定义 / 缩写。表 1-1 提供了本文档中所包含的许多约定的说明。文档末尾的术语表提供了更多手册中使用的字和缩写的定义。

1.5.1 文档约定

表 1-1 定义了本手册所使用的某些符号和术语。

表 1-1: 文档约定

符号或术语	说明
置位	强制位 / 寄存器的值为逻辑 “1”。
清零	强制位 / 寄存器的值为逻辑 “0”。
复位	1) 强制寄存器 / 位回复到默认状态。 2) 器件复位发生后器件所处的状态。某些位将被强制变为 “0”（如中断使能位），而其他位则被置 “1”（如 I/O 数据方向位）。
0xnn 或 nnh	用十六进制数字系统指定数字 “nn”。此约定用于代码示例。例如，0x13F 或 13Fh。
B “bbbbbbbbb”	用二进制数字系统指定数字 “bbbbbbbbb”。这种约定用于文本以及图表中。例如，B “10100000”。
R-M-W	读—修改—写。这表示寄存器或端口值被读取，然后该值在修改后再写回寄存器或端口。单条指令（如置位，BSET）或一个指令序列可执行读—修改—写操作。
:（半角冒号）	用来指定范围，或一串寄存器 / 位 / 引脚。 例如 TMR3:TMR2 表示用两个 16 位寄存器串连成一个 32 位定时器值。 连接顺序（从左到右）通常表示一种位置关系（MSb 到 LSb，高位到低位）。
< >	在特定寄存器中指定位的位置。 例如 PTCON<PTMOD1:PTMOD0>（或 PTMOD<1:0>），指定了寄存器及其相关的多个位或单个位的位置。
MSb, MSbit, LSb, LSbit	表示字段中的最低有效位或最高有效位。
MSByte, MSWord, LSByte, LSWord	表示许多位构成的字段中的最低 / 最高有效字节或字。
Courier New 字体	用于代码示例、二进制数和文本中的指令助记符。
Times 字体	用于公式和变量。
<i>Times, 粗体, 斜体</i>	用于图表 / 等式 / 例子中项目的说明性文字。
注	“注”表示需要强调的信息，可以帮助您避免常见错误，或提醒您注意同一系列不同器件间的操作区别。在表格中，“注”始终位于表格的下方（如本表格），除此以外“注”总是在阴影方框中出现（如下所示）。 注： 这是一个在阴影注释框中的“注”。

1.5.2 电气规范

本手册中有一些关于电气规范及其参数编号的参考信息。表 1-2 所示为 dsPIC30F 器件参数编号方式的约定。虽然器件之间的实际参数值有所不同，但参数的编号代表了惟一的一组特性和条件在不同数据手册中保持一致。本手册描述了某一系列的器件，因此不详细说明参数值。有关某器件的实际参数值，用户应参考该器件数据手册的“电气规范”部分。

表 1-2: 电气规范参数编号方式的约定

参数编号格式	注释
DXXX	DC 规范
AXXX	模拟外设的 DC 规范
XXX	时序 (AC) 规范
PDXXX	器件编程 DC 规范
PXXX	器件编程时序 (AC) 规范

图注: XXX 代表编号。

1.6 相关文档

Microchip 及其他渠道会提供更多的文档，有助于您使用 dsPIC30F MCU 进行开发。下面列出了最常用的文档，当然还有其它文档可供参考。请浏览 Microchip 网站 (www.microchip.com) 查阅最新发布的技术文档。

1.6.1 Microchip 文档

撰写本书时，Microchip 提供以下 dsPIC30F 文档。其中许多文档提供了特定应用的信息，这些信息给出了 dsPIC30F MCU 使用、编程和设计的实例。

1. *dsPIC30F Programmer's Reference Manual (DS70030)*

dsPIC30F Programmer's Reference Manual 提供有关 dsPIC30F 编程器的模型和指令集的信息。此文档还提供每条指令和指令语法示例的说明。

2. *dsPIC30F Family Overview (DS70043)*

此文档概括了现有 dsPIC30F 系列中的不同器件，包括器件引脚排列、存储器大小和可用的外设。

3. *dsPIC30F Data Sheets (DS70082 和 DS70083)*

此数据手册包含特定器件信息，如引脚排列和封装细节、电气规范和存储器映射。

1.6.2 第三方文档

有一些文档是由全球的第三方渠道提供的。Microchip 并没有查证这些文档的技术准确性。然而，这些文档有助于理解 dsPIC30F 器件的工作原理。如需获取与 dsPIC30F 相关的第三方文档的信息，请参见 Microchip 网站。

1.7 版本历史

版本 A

这是本文档的初始发行版。

版本 B

本章未包含任何技术内容或编辑排版上的修订，然而我们谨更新此节以反映本手册已更新到版本 B。

版本 C

本章未包含任何技术内容修订，然而我们谨更新此节以反映本手册已更新到版本 C。

注:

第 2 章 CPU

目录

本章包括下列内容：

2.1	简介	2-2
2.2	编程模型	2-4
2.3	软件堆栈指针	2-8
2.4	CPU 寄存器描述	2-11
2.5	算术逻辑单元 (ALU)	2-17
2.6	DSP 引擎	2-18
2.7	除法支持	2-27
2.8	指令流类型	2-27
2.9	循环结构	2-30
2.10	地址寄存器相依性	2-35
2.11	寄存器映射	2-38
2.12	相关应用笔记	2-40
2.13	版本历史	2-41

2.1 简介

dsPIC30F CPU 模块采用 16 位（数据）改良的哈佛架构，并带有增强型指令集包含对 DSP 的有力支持。CPU 拥有 24 位指令字，指令字带有长度可变的操作码字段。程序计数器（PC）为 24 位宽，可以寻址高达 $4M \times 24$ 位的用户程序存储器空间。单周期指令预取机制用来帮助维持吞吐量并提供可预测的执行。除了改变程序流的指令、双字移动（MOV.D）指令和表指令以外，所有指令都在单个周期内执行。使用 DO 和 REPEAT 指令支持无开销的程序循环结构，这两个指令在任何时候都可被中断。

dsPIC30F 器件在编程模型中有 16 个 16 位工作寄存器。每个工作寄存器都可以充当数据、地址或地址偏移寄存器。第 16 个工作寄存器（W15）作为软件堆栈的指针工作，用于中断和调用。

dsPIC30F 指令集有两类指令：MCU 类指令和 DSP 类指令。这两类指令无缝地集成到架构中并从同一个执行单元执行。指令集包括很多寻址模式，指令的设置可使 C 编译器的效率达到最优。

数据空间可以作为 32K 字或 64K 字节寻址，并被分成两块，称为 X 和 Y 数据存储器。每个存储块有各自独立的地址发生单元（Address Generation Unit, AGU）。MCU 类指令只通过 X 存储器 AGU 进行操作，可将整个存储器映射空间作为一个线性数据空间访问。某些 DSP 指令通过 X 和 Y 的 AGU 进行操作以支持双操作数读操作，这将数据地址空间分成两个部分。X 和 Y 数据空间的边界视具体器件而定。

可以选择将数据空间存储器映射空间的高 32 KB 映射到由 8 位程序空间可视性页（Program Space Visibility Page, PSVPAG）寄存器定义的任何 16K 程序字边界内的程序空间内。程序空间到数据空间的映射功能让任何指令都能象访问数据空间一样访问程序空间。另外，在带有外部总线的器件上可将 RAM 连接到程序存储器总线用来扩展内部数据 RAM。

X 和 Y 寻址空间都支持无开销循环缓冲器（模寻址）。模寻址省去了 DSP 算法的软件边界检查开销。此外，X AGU 的循环寻址可以与任何 MCU 类指令一起使用。X AGU 还支持位反转寻址，大幅简化了基 2（Radix-2）FFT 算法对输入或输出数据的重新排序。

CPU 支持固有（无操作数）、相对、立即数、存储器直接、寄存器直接和寄存器间接寻址模式。每条指令根据其功能要求，与一个预定义的寻址模式组相关。每条指令最多支持 6 种寻址模式。

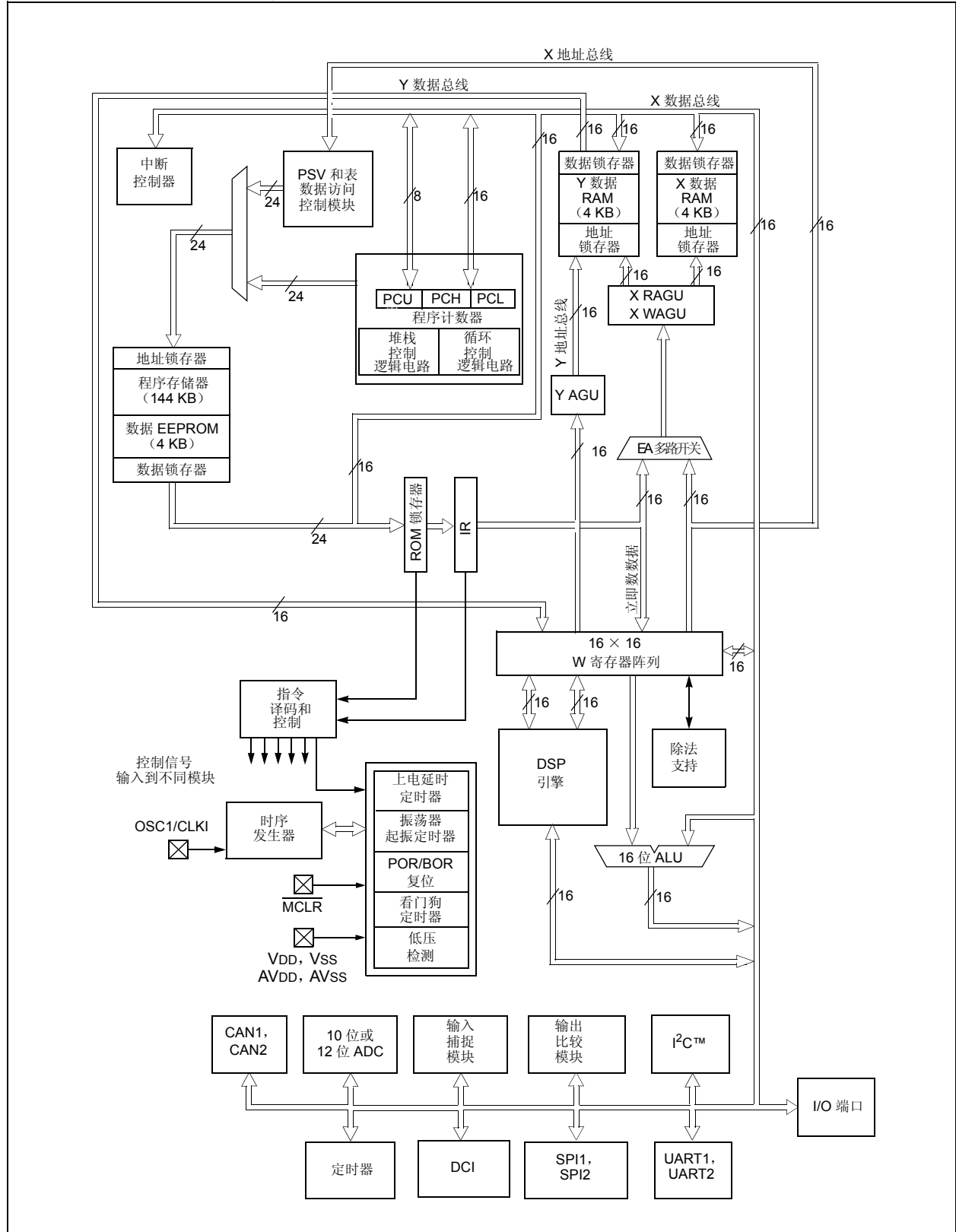
对于大多数指令，在每个指令周期 dsPIC30F 能执行一次数据（或程序数据）存储器读操作、一次工作寄存器（数据）读操作、一次数据存储器写操作和一次程序（指令）存储器读操作。因此可以支持 3 个操作数的指令，使 $A + B = C$ 操作能在单周期内执行。

DSP 引擎具备一个高速 17 位 $\times$ 17 位乘法器、一个 40 位 ALU、两个 40 位饱和累加器和一个 40 位双向桶形移位寄存器。该桶形移位寄存器在单个周期内至多可将一个 40 位的值右移 15 位或左移 16 位。DSP 指令可以无缝地与所有其他指令一起操作，其设计可实现最佳的实时性能。MAC 指令和其他相关指令可以同时从存储器中取出两个数据操作数并将两个 W 寄存器相乘。这要求数据空间对于这些指令拆分为两块，但对所有其他指令保持线性。这是通过为每个地址空间指定某些工作寄存器，以透明和灵活的方式实现的。

dsPIC30F 具有向量异常（Exception）机制，带有多达 8 个不可屏蔽陷阱源和 54 个中断源。可以为每个中断源分配 7 个优先级之一。

图 2-1 中所示为 CPU 的框图。

图 2-1: dsPIC30F CPU 内核框图



2.2 编程模型

图 2-2 中所示为 dsPIC30F 的编程模型。编程模型中的所有寄存器都是存储器映射的，并且可以由指令直接控制。表 2-1 中提供了对各个寄存器的描述。

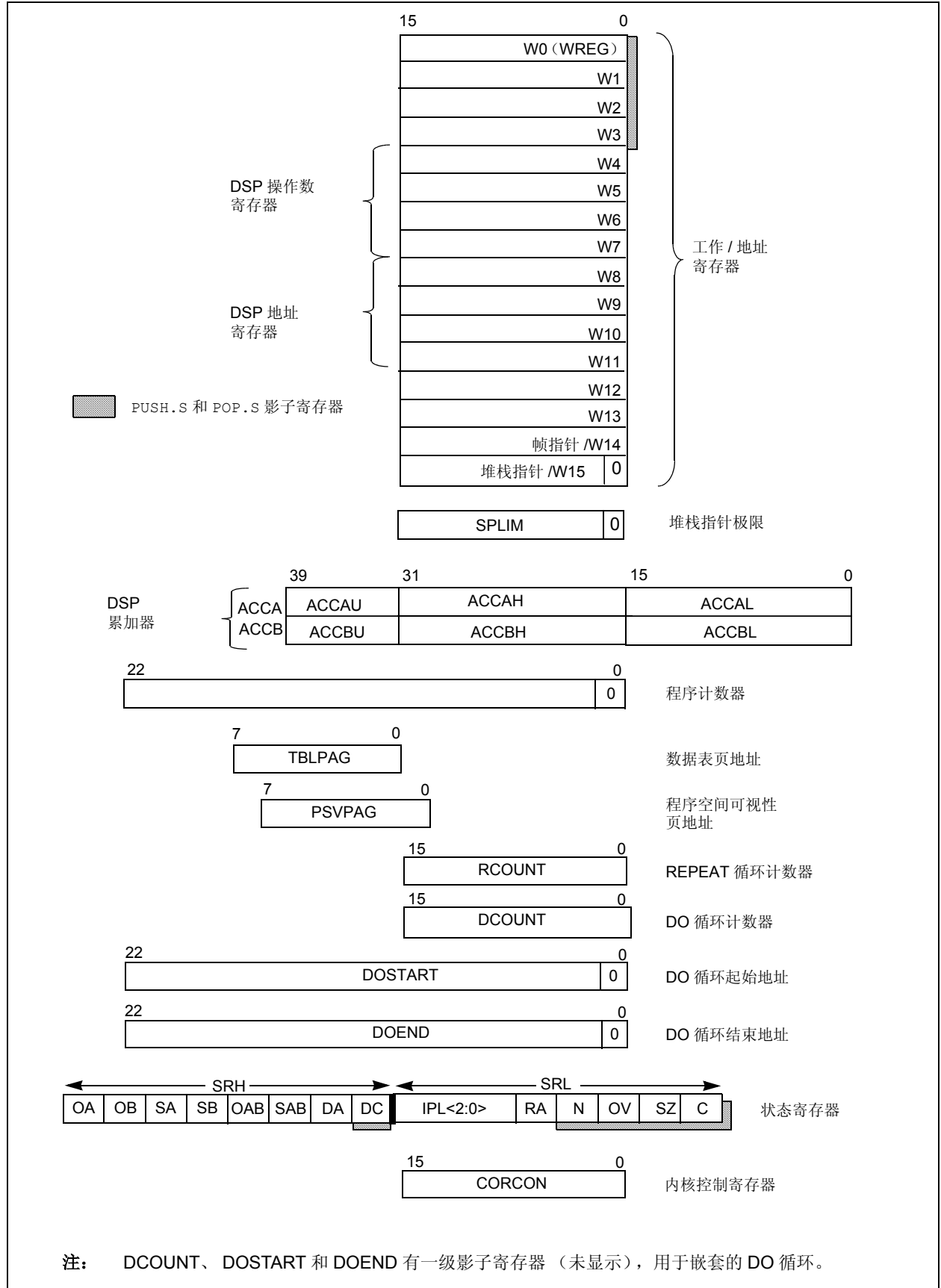
表 2-1：编程模型的寄存器描述

寄存器名称	描述
W0 到 W15	工作寄存器阵列
ACCA, ACCB	40 位 DSP 累加器
PC	23 位程序计数器
SR	ALU 和 DSP 引擎状态寄存器
SPLIM	堆栈指针极限值寄存器
TBLPAG	表存储器页地址寄存器
PSVPAG	程序空间可视性页地址寄存器
RCOUNT	REPEAT 循环计数寄存器
DCOUNT	DO 循环计数寄存器
DOSTART	DO 循环起始地址寄存器
DOEND	DO 循环结束地址寄存器
CORCON	包含 DSP 引擎和 DO 循环控制位

除了编程模型中包含的寄存器之外，dsPIC30F 还包含模寻址、位反转寻址和中断的控制寄存器。这些寄存器将在本文档随后的章节中介绍。

与编程模型有关的所有寄存器都是存储器映射的，如在第 2-38 页上的表 2-8 中所示。

图 2-2: 编程模型



2.2.1 工作寄存器阵列

16 个工作寄存器（W）可以作为数据、地址或地址偏移寄存器。W 寄存器的功能由访问它的指令的寻址模式决定。

dsPIC30F 指令集可被分成两种指令类型：寄存器和文件寄存器指令。寄存器指令可以把每个 W 寄存器用作数据值或地址偏移值。例如：

```
MOV    W0,W1      ; move contents of W0 to W1
MOV    W0,[W1]    ; move W0 to address contained in W1
ADD    W0,[W4],W5; add contents of W0 to contents pointed
                  ; to by W4. Place result in W5.
```

2.2.1.1 W0 和文件寄存器指令

W0 是一个特殊的工作寄存器，因为它是可在文件寄存器指令中使用的惟一的工作寄存器。文件寄存器指令对在指令操作码和 W0 中包含的指定的存储器地址进行操作。在文件寄存器指令中，W1 到 W15 不可被指定为目标寄存器。

文件寄存器指令对只有一个 W 寄存器的现有 PIC® 器件提供向后兼容性。在汇编器语法中使用标号“WREG”来表示数据寄存器指令中的 W0。例如：

```
MOV    WREG,0x0100 ; move contents of W0 to address 0x0100
ADD    0x0100,WREG ; add W0 to address 0x0100, store in W0
```

注： 关于寻址模式和指令语法的完整说明，请参见 dsPIC30F Programmer's Reference Manual (DS70030)。

2.2.1.2 W 寄存器存储器映射

因为 W 寄存器是存储器映射的，所以在文件寄存器指令中访问 W 寄存器是可能的，示例如下：

```
MOV    0x0004, W10 ; equivalent to MOV W2, W10
```

其中 0x0004 是 W2 存储器中的地址。

此外，执行一条试图将 W 寄存器用作地址指针和操作数目标的指令也是可能的。例如：

```
MOV    W1,[W2++]
```

其中：

```
W1 = 0x1234
W2 = 0x0004 ; [W2] addresses W2
```

在上面的示例中，W2 的内容是 0x0004。因为 W2 被用作地址指针，它指向存储器中的单元 0x0004。W2 也映射到存储器中的该地址空间。虽然这是不太可能发生的事件，但不到运行时不可能检测到。dsPIC30F 确保数据写操作将占据主导，使上面示例中 W2 的结果为 0x1234。

2.2.1.3 W 寄存器和字节模式指令

把 W 寄存器阵列当作目标寄存器的字节指令只影响目标寄存器的最低有效字节。因为工作寄存器是存储器映射的，所以可以通过对数据存储器空间进行字节宽度的访问来控制工作寄存器的最低和最高有效字节。

2.2.2 影子寄存器

如图 2-2 中所示，在编程模型中的许多寄存器都有相关的影子寄存器。影子寄存器都不能直接访问。有两种类型的影子寄存器：一类被 PUSH.S 和 POP.S 指令使用，另一类被 DO 指令使用。

2.2.2.1 PUSH.S 和 POP.S 影子寄存器

在执行函数调用或中断服务程序（Interrupt Service Routine, ISR）时，PUSH.S 和 POP.S 指令可用于快速的现场保存 / 恢复。PUSH.S 指令会将下列寄存器的值传输到它们各自的影子寄存器：

- W0……W3
- SR（仅 N、OV、Z、C 和 DC 位）

POP.S 指令会将值从影子寄存器恢复到这些寄存器单元。下面是使用 PUSH.S 和 POP.S 指令的代码示例：

MyFunction:

```
PUSH.S           ; Save W registers, MCU status
MOV    #0x03,W0  ; load a literal value into W0
ADD    RAM100    ; add W0 to contents of RAM100
BTSC   SR,#Z     ; is the result 0?
BSET   Flags,#IsZero ; Yes, set a flag
POP.S           ; Restore W regs, MCU status
RETURN
```

PUSH.S 指令会改写先前保存在影子寄存器中的内容。影子寄存器深度只有一级，所以如果多个软件任务使用影子寄存器，必须小心。

用户必须确保使任何使用影子寄存器的任务均不会被同样使用该影子寄存器且具有更高优先级的任务中断。如果允许较高优先级的任务中断较低优先级的任务，较低优先级任务保存在影子寄存器中的内容将被较高优先级任务改写。

2.2.2.2 DO 循环影子寄存器

当执行 DO 指令时，下列寄存器的内容将自动保存在影子寄存器中。

- DOSTART
- DOEND
- DCOUNT

DO 影子寄存器的深度为一级，允许两个循环自动嵌套。更多详细信息，请参见第 2.9.2.2 节“DO 循环嵌套”。

2.2.3 未初始化的 W 寄存器的复位

W 寄存器阵列（除 W15 之外）在发生所有复位时被清零，并且在被写入前视为未经初始化。试图把未初始化的寄存器用作地址指针将会复位器件。

必须执行字写操作来初始化 W 寄存器。字节写操作不会影响初始化检测逻辑。

2.3 软件堆栈指针

W15 作为专用的软件堆栈指针并被异常处理、子程序调用和返回自动修改。但是，W15 可以被任何指令以与所有其他 W 寄存器相同的方式引用。这样就简化了对堆栈指针的读、写和控制（例如，创建堆栈帧）。

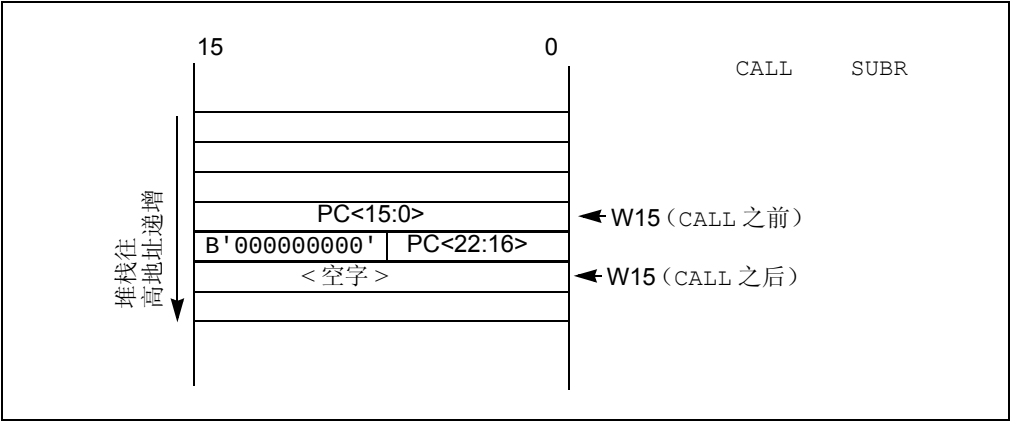
注： 为了防止偏离的堆栈访问，W15<0> 被硬件固定为 0。

所有复位均将 W15 初始化为 0x0800。此地址确保在所有 dsPIC30F 器件中，堆栈指针（SP）将指向有效的 RAM 并允许在 SP 被用户软件初始化前发生的不可屏蔽异常陷阱使用堆栈。在初始化期间，用户可以将 SP 重新编程以指向数据空间内的任何单元。

堆栈指针总是指向第一个可用的空字并从低地址到高地址填充软件堆栈。堆栈出栈（读）时，堆栈指针先减；堆栈进栈（写）时，堆栈指针后加，如图 2-3 所示。

当 PC 压入堆栈时，PC<15:0> 被压入第一个可用的堆栈字，然后 PC<22:16> 被压入第二个可用的堆栈单元。对于任何 CALL 指令执行期间的 PC 进栈，进栈前 PC 的 MSB 是以零扩展的，如图 2-3 所示。例外处理期间，PC 的 MSB 与 CPU 状态寄存器 SR 的低 8 位相连。这样就使 SRL 的内容在中断处理期间能被自动保存。

图 2-3: CALL 指令的堆栈操作



2.3.1 软件堆栈示例

使用 PUSH 和 POP 指令可控制软件堆栈。PUSH 和 POP 指令相当于将 W15 用作目标指针的 MOV 指令。例如，要把 W0 的内容压入堆栈，可通过：

PUSH W0

此语法相当于：

MOV W0, [W15++]

要把栈顶的内容返回 W0，可通过：

POP W0

此语法相当于：

MOV [--W15], W0

图 2-4 到图 2-7 给出了如何使用软件堆栈的示例。图 2-4 所示为器件初始化时的软件堆栈。W15 已经初始化为 0x0800。此外，此示例假设 0x5A5A 和 0x3636 这两个值已被分别写入 W0 和 W1。图 2-5 中堆栈第一次进栈，W0 中包含的值被复制到堆栈中。W15 自动更新以指向下一个可用的堆栈单元（0x0802）。在图 2-6 中，W1 的内容被压入堆栈。图 2-7 中，堆栈出栈并且栈顶值（先前从 W1 压入）被写入 W3。

图 2-4: 器件复位时的堆栈指针

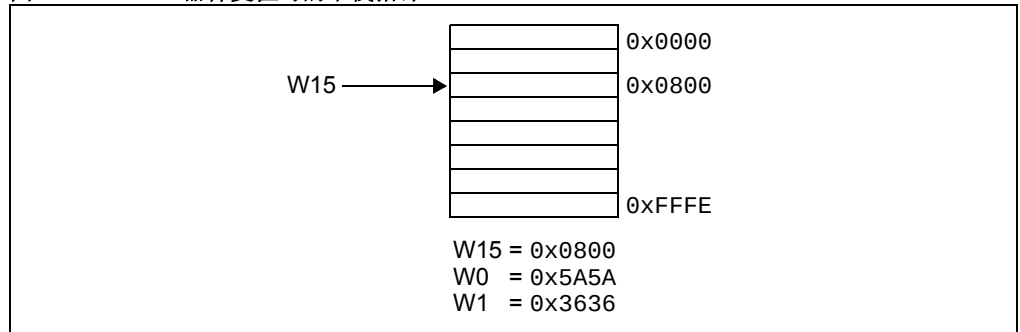


图 2-5: 第一次执行 PUSH 指令后的堆栈指针

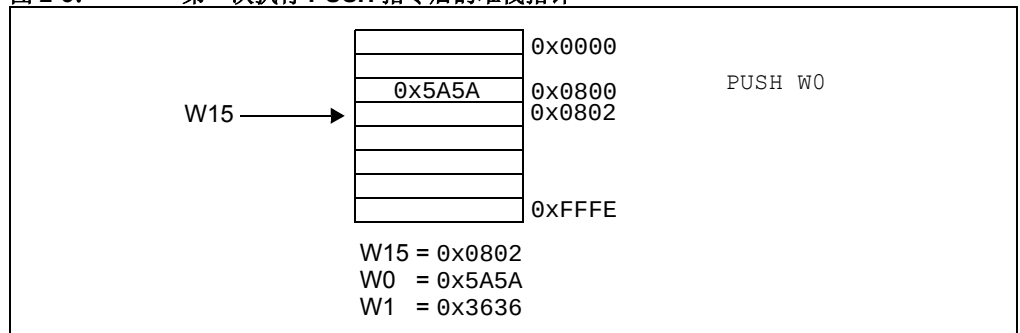


图 2-6: 第二次执行 PUSH 指令后的堆栈指针

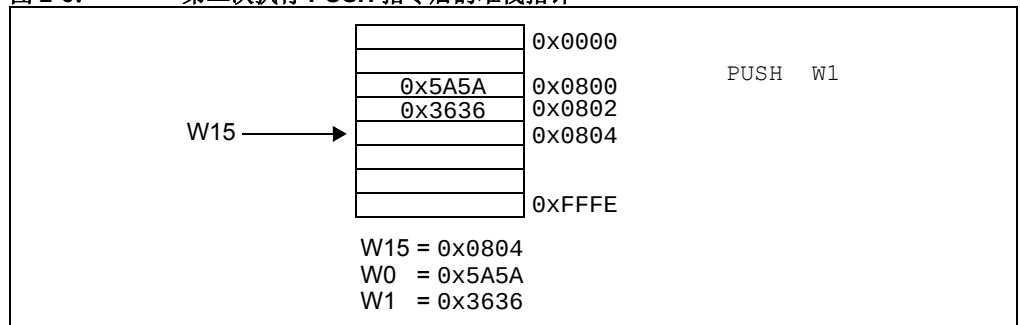
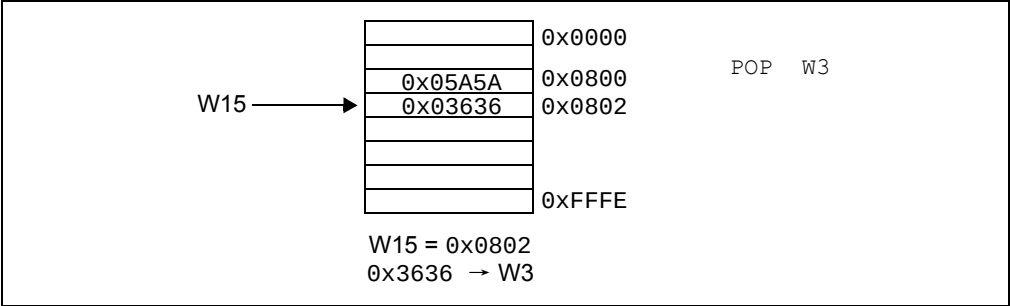


图 2-7: 执行一条 POP 指令后的堆栈指针



2.3.2 W14 软件堆栈帧指针

帧是堆栈中用户定义的存储器段，供单个子程序使用。W14 是特殊工作寄存器，因为通过使用 LNK（link，连接）和 ULNK（unlink，不连接）指令可以把它用作堆栈帧指针。当不用作帧指针时，W14 可被指令当作普通的工作寄存器使用。

若要获取将W14用作堆栈帧指针的软件示例，请参见dsPIC30F Programmer's Reference Manual (DS70030)。

2.3.3 堆栈指针上溢

有一个与堆栈指针相关的堆栈极限寄存器（SPLIM），复位时为 0x0000。SPLIM 是一个 16 位寄存器，但是 SPLIM<0> 被固定为 “0”，因为所有的堆栈操作必须字对齐。

直到一个字写入 SPLIM 后才使能堆栈上溢检查，在此之后，只能通过器件复位禁止堆栈上溢检查。所有将 W15 用作源或目标寄存器而产生的有效地址将与 SPLIM 中的值作比较。如果堆栈指针（W15）的内容比 SPLIM 寄存器的内容大 2，并且执行了进栈操作，将不会产生堆栈错误陷阱（Stack Error Trap）。堆栈错误陷阱将在随后的进栈操作时产生。所以，例如，如果想要在堆栈指针递增超出RAM中的地址0x2000时，引起堆栈错误陷阱，可将SPLIM初始化为0x1FFE。

注： 堆栈错误陷阱可以由任何使用W15寄存器的内容来产生有效地址（Effective Address, EA）的指令引起。所以，如果 W15 的内容比 SPLIM 寄存器的内容大 2，并且执行了一条 CALL 指令或发生了中断，那么将产生堆栈错误陷阱。

如果已经使能了堆栈上溢检查，则当 W15 有效地址计算越过了数据空间的末尾（0xFFFF）时，也将产生堆栈错误陷阱。

注： 对堆栈指针极限寄存器 SPLIM 的写操作后面不应紧跟一个使用 W15 的间接读操作。

更多有关堆栈错误陷阱的信息，请参见第 6 章 “复位中断”。

2.3.4 堆栈指针下溢

发生复位时，堆栈初始化为 0x0800。如果堆栈指针地址小于 0x0800，堆栈错误陷阱将会产生。

注： 一般而言，数据空间中 0x0000 和 0x07FF 之间的单元预留给内核和外设的特殊功能寄存器。

2.4 CPU 寄存器描述

2.4.1 SR: CPU 状态寄存器

dsPIC30F 的 CPU 有一个 16 位状态寄存器 (SR)，它的低字节称为低状态寄存器 (Lower Status Register, SRL)。SR 的高字节称为 SRH。寄存器 2-1 中给出了对 SR 的详细描述。

SRL 包含了所有的 MCU ALU 操作状态标志，加上 CPU 中断优先级状态位 IPL<2:0> 和 REPEAT 循环有效状态位 RA (SR<4>)。异常处理期间，SRL 与 PC 的 MSB 相连形成一个完整的字值，然后将该字值压入堆栈。

SRH 包含 DSP 加法器 / 减法器状态位、DO 循环有效位 DA (SR<9>) 和辅助进位标志位 DC (SR<8>)。

SR 位可读 / 写，但以下各位例外：

1. DA 位 (SR<8>)：DA 是只读位。
2. RA 位 (SR<4>)：RA 是只读位。
3. OA、OB (SR<15:14>) 和 OAB (SR<11>) 位：这些位是只读位而且只能被 DSP 引擎硬件修改。
4. SA、SB (SR<13:12>) 和 SAB (SR<10>) 位：这些位是只读只清零的，而且只能被 DSP 引擎硬件置 1。一旦被置 1，它们就保持置位状态直到被用户清零，与任何随后的 DSP 操作的结果无关。

注： 清零 SAB 位的同时将清零 SA 和 SB 位。

注： dsPIC30F Programmer's Reference Manual (DS70030) 中提供了对受各条指令影响的 SR 位的描述。

2.4.2 CORCON: 内核控制寄存器

CORCON 寄存器包含控制 DSP 乘法器和 DO 循环硬件操作的位。CORCON 寄存器还包含 IPL3 状态位，它与 IPL<2:0> (SR<7:5>) 相连形成 CPU 中断优先级。

dsPIC30F 系列参考手册

寄存器 2-1: SR: CPU 状态寄存器

高字节:							
R-0	R-0	R/C-0	R/C-0	R-0	R/C-0	R-0	R/W-0
OA	OB	SA	SB	OAB	SAB	DA	DC
bit 15							bit 8

低字节: (SRL)							
R/W-0 ⁽²⁾	R/W-0 ⁽²⁾	R/W-0 ⁽²⁾	R-0	R/W-0	R/W-0	R/W-0	R/W-0
IPL<2:0>			RA	N	OV	Z	C
bit 7							bit 0

- bit 15 **OA:** 累加器 A 上溢状态位
1 = 累加器 A 上溢
0 = 累加器 A 未上溢
- bit 14 **OB:** 累加器 B 上溢状态位
1 = 累加器 B 上溢
0 = 累加器 B 未上溢
- bit 13 **SA:** 累加器 A 饱和 “粘着” (Sticky) 状态位
1 = 累加器 A 饱和或在某时已经饱和
0 = 累加器 A 未饱和
注: 此位可读或被清零 (不能置位)。
- bit 12 **SB:** 累加器 B 饱和 “粘着” 状态位
1 = 累加器 B 饱和或在某时已经饱和
0 = 累加器 B 未饱和
注: 此位可读或被清零 (不能置位)。
- bit 11 **OAB:** OA || OB 组合的累加器上溢状态位
1 = 累加器 A 或 B 已经上溢
0 = 累加器 A 和 B 都未上溢
- bit 10 **SAB:** SA || SB 组合的累加器 “粘着” 状态位
1 = 累加器 A 或 B 饱和或在过去某时已经饱和
0 = 累加器 A 和 B 都未饱和
注: 此位可读或被清零 (不能置位)。清零此位的同时将清零 SA 和 SB。
- bit 9 **DA:** DO 循环有效位
1 = 正在进行 DO 循环
0 = 未进行 DO 循环
- bit 8 **DC:** MCU ALU 半进位 / 借位标志位
1 = 结果的第 4 低位 (对于字节大小的数据) 或第 8 低位 (对于字大小的数据) 发生了向高位的进位
0 = 结果的第 4 低位 (对于字节大小的数据) 或第 8 低位 (对于字大小的数据) 未发生向高位的进位

寄存器 2-1: SR: CPU 状态寄存器 (续)

bit 7-5 IPL<2:0>: CPU 中断优先级级别状态位 (1)

- 111 = CPU 中断优先级级别是 7 (15)。禁止用户中断。
- 110 = CPU 中断优先级级别是 6 (14)
- 101 = CPU 中断优先级级别是 5 (13)
- 100 = CPU 中断优先级级别是 4 (12)
- 011 = CPU 中断优先级级别是 3 (11)
- 010 = CPU 中断优先级级别是 2 (10)
- 001 = CPU 中断优先级级别是 1 (9)
- 000 = CPU 中断优先级级别是 0 (8)

注 1: IPL<2:0> 位与 IPL<3> 位 (CORCON<3>) 相连以形成 CPU 中断优先级级别。如果 IPL<3> = 1, 那么括号中的值表示 IPL。当 IPL<3> = 1 时, 禁止用户中断。
2: 当 NSTDIS = 1 (INTCON1<15>) 时, IPL<2:0> 状态位为只读。

bit 4 RA: REPEAT 循环有效位
1 = 正在进行 REPEAT 循环
0 = 未进行 REPEAT 循环

bit 3 N: MCU ALU 负标志位
1 = 结果为负
0 = 结果为非负 (零或正)

bit 2 OV: MCU ALU 溢出标志位
此位用于带符号的算术运算 (2 进制补码)。它表明数量级的溢出, 这样导致了符号位改变状态。
1 = 带符号的算术运算中发生溢出 (本次运算)
0 = 未发生溢出

bit 1 Z: MCU ALU 零标志位
1 = 影响 Z 位的运算在过去某时将该位置 1
0 = 影响 Z 位的最近一次运算已经将该位清零 (也就是说, 一个非零结果)

bit 0 C: MCU ALU 进位 / 借位标志位
1 = 结果的最高有效位发生了进位
0 = 结果的最高有效位未发生进位

图注:		
R = 可读位	W = 可写位	U = 未用位, 读作 0
C = 只清零位	S = 只置 1 位	-n = POR 时的值
1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 2-2: CORCON: 内核控制寄存器

高字节:							
U-0	U-0	U-0	R/W-0	R/W-0	R-0	R-0	R-0
—	—	—	US	EDT	DL<2:0>		
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-1	R/W-0	R/C-0	R/W-0	R/W-0	R/W-0
SATA	SATB	SATDW	ACCSAT	IPL3	PSV	RND	IF
bit 7							bit 0

bit 15-13 未用: 读作 0

bit 12 **US**: DSP 乘法无符号 / 带符号控制位

1 = DSP 引擎乘法无符号

0 = DSP 引擎乘法带符号

bit 11 **EDT**: DO 循环提前终止控制位

1 = 在当前循环的迭代结束时终止执行 DO 循环

0 = 无影响

注: 此位总是读作 0。

bit 10-8 **DL<2:0>**: DO 循环嵌套级状态位

111 = 7 个 DO 循环有效

•

•

001 = 1 个 DO 循环有效

000 = 0 个 DO 循环有效

bit 7 **SATA**: AccA 饱和和使能位

1 = 使能累加器 A 饱和

0 = 禁止累加器 A 饱和

bit 6 **SATB**: AccB 饱和和使能位

1 = 使能累加器 B 饱和

0 = 禁止累加器 B 饱和

bit 5 **SATDW**: 来自 DSP 引擎的数据空间写操作饱和和使能位

1 = 使能数据空间写操作饱和

0 = 禁止数据空间写操作饱和

bit 4 **ACCSAT**: 累加器饱和模式选择位

1 = 9.31 饱和 (超级饱和)

0 = 1.31 饱和 (正常饱和)

bit 3 **IPL3**: CPU 中断优先级状态位 3

1 = CPU 中断优先级高于 7

0 = CPU 中断优先级等于或低于 7

注: IPL3 位与 IPL<2:0> 位 (SR<7:5>) 相连形成 CPU 中断优先级。

寄存器 2-2: CORCON: 内核控制寄存器 (续)

- bit 2 **PSV:** 数据空间中的程序空间可视性使能位
1 = 程序空间在数据空间中可视
0 = 程序空间在数据空间中不可视
- bit 1 **RND:** 舍入模式选择位
1 = 使能带偏置的 (传统) 舍入
0 = 使能非偏置 (收敛) 舍入
- bit 0 **IF:** 整数或小数乘法器模式选择位
1 = 使能 DSP 乘法运算器的整数模式
0 = 使能 DSP 乘法运算器的小数模式

图注:			
R = 可读位	W = 可写位	U= 未用位, 读作 0	C = 位可被清零
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

2.4.3 其他 dsPIC30F CPU 控制寄存器

以下所列的寄存器与 dsPIC30F CPU 内核有关，本手册的其他章节会对它们进行更详细的描述。

2.4.3.1 TBLPAG: 表页寄存器

TBLPAG 寄存器用于在读表和写表操作过程中保存程序存储器地址的高 8 位。表指令用于传输程序存储空间和数据存储空间之间的数据。更多详细信息，请参见第 4 章“程序存储器”。

2.4.3.2 PSVPAG: 程序空间可视性页寄存器

程序空间可视性允许用户将程序存储空间的 32 KB 区域映射到数据地址空间的高 32 KB。此特性允许通过在数据存储器上操作的 dsPIC30F 指令对常数数据进行透明访问。PSVPAG 寄存器选择映射到数据地址空间的程序存储空间的 32 KB 区域。更多有关 PSVPAG 寄存器的信息，请参见第 4 章“程序存储器”。

2.4.3.3 MODCON: 模控制寄存器

MODCON 寄存器用于使能并配置模寻址（循环缓冲）。更多有关模寻址的详细信息，请参见第 3 章“数据存储器”。

2.4.3.4 XMODSRT, XMODEND: X 模起始和结束地址寄存器

XMODSRT 和 XMODEND 寄存器保持 X 数据存储地址空间中执行模（循环）缓冲的起始和结束地址。更多有关模寻址的详细信息，请参见第 3 章“数据存储器”。

2.4.3.5 YMODSRT, YMODEND: Y 模起始和结束地址寄存器

YMODSRT 和 YMODEND 寄存器保持 Y 数据存储地址空间中执行模（循环）缓冲的起始和结束地址。更多有关模寻址的详细信息，请参见第 3 章“数据存储器”。

2.4.3.6 XBREV: X 模位反转寄存器

XBREV 寄存器用于设置位反转寻址的缓冲区大小。更多有关位反转寻址的详细信息，请参见第 3 章“数据存储器”。

2.4.3.7 DISICNT: 禁止中断计数寄存器

DISI 指令使用 DISICNT 寄存器将优先级为 1-6 的中断在指定的几个周期内禁止。更多的信息，请参见第 6 章“复位中断”。

2.5 算术逻辑单元 (ALU)

dsPIC30F ALU 为 16 位宽，能进行加、减、单位移位和逻辑运算。除非特别指明，算术运算一般是以 2 的补码 (2's Complement) 的方式进行的。根据不同的操作，ALU 可能会影响 SR 寄存器中的进位标志位 (C)、全零标志位 (Z)、负标志位 (N)、溢出标志位 (OV) 和辅助进位标志位 (DC) 的值。在减法操作中，C 和 DC 位分别作为借位和辅助借位位。

根据所使用的指令模式，ALU 可以执行 8 位或 16 位操作。根据指令的寻址模式，ALU 操作的数据可以来自 W 寄存器阵列或数据存储器。同样，ALU 的输出数据可以被写入 W 寄存器阵列或数据存储器单元。

有关受到指令、寻址模式和 8 位 /16 位指令模式影响的 SR 位的信息可以参见 dsPIC30F Programmer's Reference Manual (DS70030)。

- 注**
- 1:** 字节操作使用 16 位 ALU，并可以产生再加 8 位的结果。但是，如果要保持和 PIC 器件的向后兼容性，所有字节操作的 ALU 结果必须被回写为一个字节（即，不修改 MSByte），且只根据结果 LSByte 的状态更新 SR 寄存器。
 - 2:** 字节模式中执行的所有寄存器指令只会影响 W 寄存器的 LSByte。可以使用访问 W 寄存器的存储器映射内容的文件寄存器指令修改任何 W 寄存器的 MSByte。

2.5.1 字节到字的转变

dsPIC30F 有两条指令有助于混合 8 位和 16 位 ALU 操作。符号扩展 (SE) 指令获取 W 寄存器或数据存储器的一个字节值并创建存储在 W 寄存器中的符号扩展字值。零扩展 (ZE) 指令清零 W 寄存器或数据存储器中字值的 8 MSb 并将结果放在目标 W 寄存器中。

2.6 DSP 引擎

DSP 引擎是一个硬件模块，它从 W 寄存器阵列装入数据，但它有自己专门的结果寄存器。DSP 引擎由与 MCU ALU 相同的指令译码器驱动。此外，W 寄存器阵列中还产生所有操作数有效地址（EA）。虽然 MCU ALU 和 DSP 引擎资源都可以通过指令集中的所有指令共享，但是 DSP 不能与 MCU 指令流同时操作。

DSP 引擎由以下组件组成：

- 高速 17 位 × 17 位乘法器
- 桶形移位寄存器
- 40 位加法器 / 减法器
- 两个目标累加寄存器
- 带可选模式的舍入逻辑
- 带可选模式的饱和逻辑

DSP 引擎的数据输入来自以下资源：

1. 直接来自双源操作数 DSP 指令的 W 阵列（寄存器 W4、W5、W6 或 W7）。通过 X 和 Y 存储器数据总线预取 W4、W5、W6 和 W7 寄存器的数据值。
2. 来自所有其他 DSP 指令的 X 存储器数据总线。

从 DSP 引擎输出的数据被写入以下目标之一：

1. 由执行的 DSP 指令定义的目标累加器。
2. 到数据存储器地址空间中任何单元的 X 存储器数据总线。

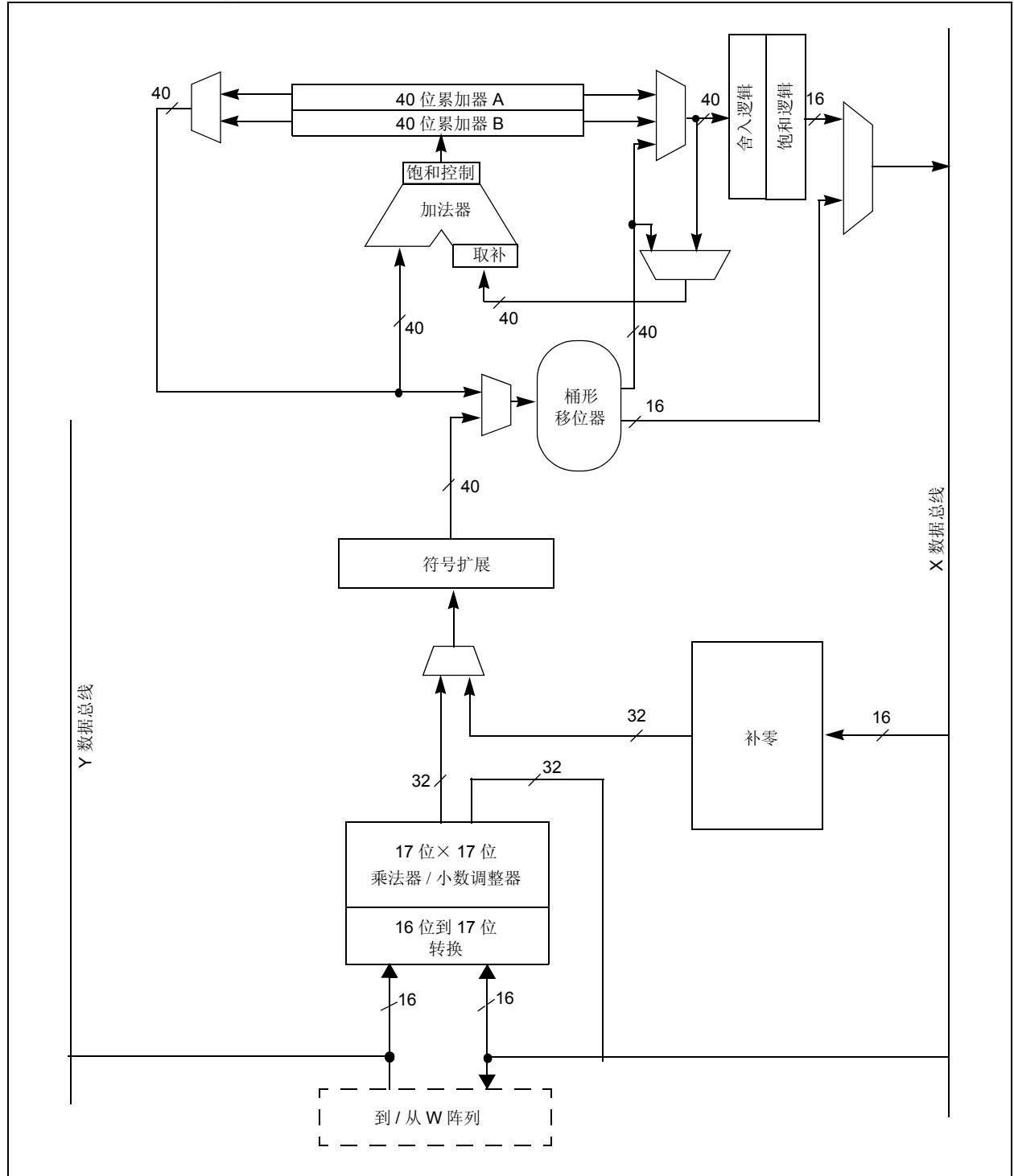
DSP 引擎能够执行固有的“累加器到累加器”的操作，而无需额外数据。

MCU 移位和乘法指令使用 DSP 引擎硬件获得结果。在这些操作中使用 X 存储器数据总线进行数据读写。

图 2-8 所示为 DSP 引擎的框图。

注： 详细代码示例和与本节相关的指令语法，请参见 dsPIC30F Programmer's Reference Manual（DS70030）。

图 2-8: DSP 引擎框图



2.6.1 数据累加器

共有两个 40 位数据累加器 ACCA 和 ACCB，它们是表 2-3 所列 DSP 指令的结果寄存器。每个累加器经存储器映射到以下三个寄存器，其中“x”表示特定的累加器：

- ACCxL: ACCx<15:0>
- ACCxH: ACCx<31:16>
- ACCxU: ACCx<39:32>

对于使用累加器的小数操作，小数点位于 bit 31 的右侧。存储在每个累加器中的小数值范围为 -256.0 到 $(256.0 - 2^{-31})$ 。对于使用累加器的整数操作，小数点位于 bit 0 的右侧。可存储在每个累加器中的整数值范围为 -549,755,813,888 到 $(549,755,813,887)$ 。

2.6.2 乘法器

dsPIC30F 具备一个由 MCU ALU 和 DSP 引擎共享的 17 位 × 17 位的乘法器。此乘法器可以进行有符号或无符号的运算，而且支持 1.31 小数 (Q.31) 或 32 位整数结果。

此乘法器取 16 位输入数据并将其转换为 17 位数据。进入乘法器的有符号操作数将进行符号扩展。无符号的输入操作数将进行零扩展。17 位转换逻辑对于用户是透明的，并允许乘法器支持有符号和无符号混合 / 无符号的乘法运算。

IF 控制位 (CORCON<0>) 确定表 2-3 所列指令的整数 / 小数操作。IF 位不会影响表 2-4 所列的 MCU 指令，因为这些 MCU 总是进行整数操作。对于小数操作，乘法器将乘积左移 1 位来进行小数调整。结果的 LSbit 总是保持清零。在器件复位时，默认乘法器为 DSP 操作的小数模式。

硬件中，每个模式的数据表示方式如下：

- 整数数据固有表示为带符号的 2 的补码值，其中 MSbit 被定义为符号位。一般来说，N 位 2 的补码整数的范围为 2^{N-1} 到 $2^{N-1} - 1$
- 小数数据表示为一个 2 的补码小数，其中 MSbit 被定义为符号位，并暗示小数点就在符号位之后 (Q.X 格式)。带这种暗示小数点的 N 位 2 的补码小数的范围为 -1.0 到 $(1 - 2^{1-N})$ 。

图 2-9 和图 2-10 说明了乘法器硬件如何处理整数和小数模式的数据。整数和小数模式的数据范围如表 2-2 所列。

图 2-9: 0x4001 的整数和小数表示



图 2-10: 0xC002 的整数和小数表示

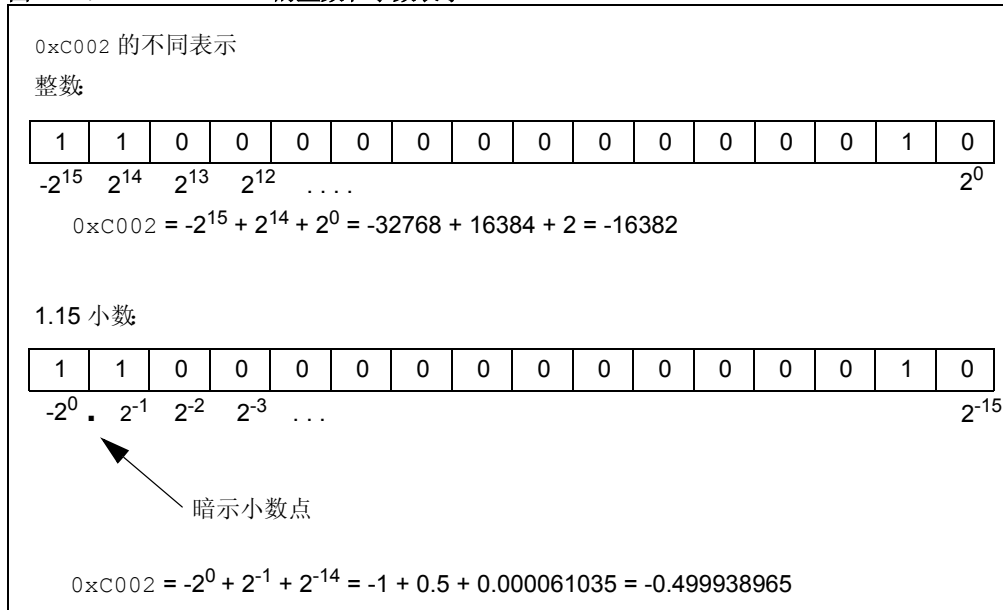


表 2-2: dsPIC30F 数据范围

寄存器大小	整数范围	小数范围	小数精度
16 位	-32768 到 32767	-1.0 到 $(1.0 \cdot 2^{-15})$ (Q.15 格式)	3.052×10^{-5}
32 位	-2,147,483,648 到 2,147,483,647	-1.0 到 $(1.0 \cdot 2^{-31})$ (Q.31 格式)	4.657×10^{-10}
40 位	-549,755,813,888 到 549,755,813,887	-256.0 到 $(256.0 \cdot 2^{-31})$ (带 8 保护位的 Q.31 格式)	4.657×10^{-10}

2.6.2.1 DSP 乘法指令

表 2-3 总结了使用乘法器的 DSP 指令。

表 2-3: 用乘法器的 DSP 指令

DSP 指令	描述	代数等式
MAC	两数相乘后与累加器相加，其值置入累加器或一数平方后与累加器相加，其值置入累加器	$a = a + b \cdot c$ $a = a + b^2$
MSC	从累加器中减去两数乘积，其值置入累加器	$a = a - b \cdot c$
MPY	乘，其值置入累加器	$a = b \cdot c$
MPY.N	乘并将结果取负后的值置入累加器	$a = -b \cdot c$
ED	偏欧氏距离 (Partial Euclidean Distance)	$a = (b - c)^2$
EDAC	将偏欧氏距离与累加器相加，其值置入累加器	$a = a + (b - c)^2$

注： 使用乘法器的 DSP 指令可以小数（1.15）或整数模式进行操作。

US 控制位（CORCON<12>）决定 DSP 乘法指令是有符号的（默认）还是无符号的。US 位不会影响到 MCU 乘法指令，MCU 有专门处理有符号或无符号运算的指令。如果 US 位被置位，表 2-3 中显示的指令输入操作数被认为是无符号的值，它总是零扩展到乘法器值的第 17 位。

2.6.2.2 MCU 乘法指令

使用同一个乘法器支持 MCU 乘法指令。MCU 乘法指令包括 16 位有符号、无符号和混合符号整数的乘法，如表 2-4 所示。所有 MUL 指令执行的乘法运算都产生整数结果。MUL 指令可被引导使用字节或字大小的操作数。字节输入操作数将产生 16 位结果而字输入操作数将产生 32 位结果到 W 阵列中的指定寄存器。

表 2-4: 使用乘法器的 MCU 指令

MCU 指令	描述
MUL/MUL.UU	将两个无符号的整数相乘
MUL.SS	两个有符号的整数相乘
MUL.SU/MUL.US	将一个有符号的整数和一个无符号的整数相乘

- 注 1: 使用乘法器的 MCU 指令只可在整数模式下操作。
- 2: MCU 乘法运算的结果为 32 位长并存储在一对 W 寄存器中。

2.6.3 数据累加器加法器 / 减法器

数据累加器有一个 40 位的加法器 / 减法器，它带有乘积结果的自动符号扩展逻辑（如果带符号）。它可以选择两个累加器（A 或 B）之一作为它的预累加源和后累加目标。对于 ADD（累加器）和 LAC 指令，将被累加或装入的数据可选择通过桶形移位器在累加之前进行调整。

40 位加法器 / 减法器可选择将操作数输入中的一个取反以改变结果的符号（不改变操作数）。负数在乘法和减法（MSC）或乘法和取反操作（MPY.N）中使用。

40 位加法器 / 减法器额外带有一个饱和控制区块，使能时，可用于控制累加器数据饱和度。

2.6.3.1 累加器状态位

提供了六个状态寄存器位以支持饱和和溢出控制。它们位于 CPU 状态寄存器 SR 中，如下表所列

表 2-5: 累加器溢出和饱和状态位

状态位	位置	描述
OA	SR<15>	累加器 A 溢出到保护位（ACCA<39:32>）
OB	SR<14>	累加器 B 溢出到保护位（ACCB<39:32>）
SA	SR<13>	ACCA 已饱和（bit 31 溢出并饱和） 或 ACCA 溢出到保护位并饱和（bit 39 溢出并饱和）
SB	SR<12>	ACCB 饱和（bit 31 溢出并饱和） 或 ACCB 溢出到保护位并饱和（bit 39 溢出并饱和）
OAB	SR<11>	OA 和 OB 逻辑或操作
SAB	SR<10>	SA 和 SB 逻辑或操作。 清零 SAB 的同时将清零 SA 和 SB。

OA 和 OB 位是只读的，且每当数据通过累加器加 / 减法逻辑时就被修改。置位时，它们表示最近的操作溢出到累加器保护位（32 到 39 位）。此类型的溢出并不是灾难性的，保护位保存累加器数据。OAB 状态位是 OA 和 OB 的逻辑或结果值。

OA 和 OB 位在置位时可以选择让其产生算法错误陷阱。通过置位相应的溢出陷阱标志使能位 OVATE:OVATE（INTCON1<10:9>）可以使能此陷阱。陷阱事件使用户可在需要时马上采取纠正行动。

每当数据通过累加器饱和和逻辑时，SA 和 SB 位被置位。一旦被置位，这些位就保持置位状态直到被用户清零。SAB 状态位表示 SA 和 SB 的逻辑或值。当 SAB 被清零时，SA 和 SB 位也将被清零。被置位时，这些位表示累加器已经溢出其最大范围（32 位饱和为 bit 31 或 40 位饱和为 bit 39）且将饱和（如果饱和逻辑被使能）。

当饱和逻辑没有使能时，SA 和 SB 位表示发生了灾难性的溢出（累加器符号已被破坏）。如果 COVTE（INTCON1<8>）位被置位，在饱和逻辑被禁止时 SA 和 SB 位将产生算法错误陷阱。

注： 更多有关算法警告陷阱的信息，请参见第 6 章“复位中断”。

注： 用户必须切记，根据累加器饱和和逻辑是否使能，SA、SB 和 SAB 状态位可能具有不同的含义。累加器饱和模式通过 CORCON 寄存器控制。

2.6.3.2 饱和及溢出模式

器件支持三种饱和及溢出模式：

1. 累加器 39 位饱和：

在此模式中，饱和逻辑将最大正 9.31 值（0x7FFFFFFF）或最大负 9.31 值（0x80000000）装入目标累加器。SA 或 SB 位被置位并保持置位直到被用户清零。此饱和度模式对于扩展累加器的动态范围很有用处。

要配置此饱和模式，必须置位 ACCSAT（CORCON<4>）位。此外，SATA 和 / 或 SATB（CORCON<7 和 / 或 6>）位必须置位以使能累加器饱和逻辑。

2. 累加器 31 位饱和：

在此模式中，饱和逻辑将最大正 1.31 值（0x007FFFFFFF）或最大负 1.31 值（0xFF800000）装入目标累加器。SA 或 SB 位被置位并保持置位直到被用户清零。当此饱和度模式有效时，除了对累加器值进行符号扩展以外，不使用保护位 32 到 39。因此，SR 中的 OA、OB 或 OAB 位不会置位。

要配置此溢出和饱和模式，必须清零 ACCSAT（CORCON<4>）位。此外，SATA 和 / 或 SATB（CORCON<7 和 / 或 6>）位必须置位以使能累加器饱和逻辑。

3. 累加器灾难性溢出：

如果 SATA 和 / 或 SATB（CORCON<7 和 / 或 6>）位没有置位，则累加器不会执行饱和操作，且允许累加器溢出到 bit 39（破坏它的符号）。如果 COVTE 位（INTCON1<8>）被置位，灾难性的溢出会导致算法错误陷阱。

注意，只有执行一条 DSP 指令，并因此通过 40 位 DSP ALU 修改了两个累加器中的一个，才可进行累加器饱和和溢出检测。当通过 MCU 类别指令将累加器作为存储器映射寄存器进行访问时，饱和和溢出检测不会发生。此外，表 2-5 中的累加器状态位将不会被修改。但是 MCU 状态位（Z、N、C、OV 和 DC）将根据访问累加器的 MCU 指令进行修改。

注： 有关算法错误陷阱的更多信息，请参见第 6 章“复位中断”。

2.6.3.3 数据空间写饱和

除了加法器 / 减法器饱和，对数据空间进行写操作也会饱和但不会影响源累加器的内容。此特性使得在中间计算阶段，在不牺牲累加器动态范围的情况下对数据进行限制。置位 SATDW 控制位（CORCON<5>）将使能数据空间写饱和逻辑。在器件复位时，数据空间写饱和逻辑是默认为使能的。

数据空间写饱和和特性与 SAC 和 SAC.R 指令共同工作。执行这些指令时从不修改累加器中保存的值。硬件通过以下步骤得到饱和的写结果：

1. 读出的数据根据指令中指定的算法移位值进行调整。
2. 经过调整的数据被舍入（仅 SAC.R）。
3. 被调整 / 舍入的值根据保护位的值饱和为 16 位结果。对于数据值大于 0x007FFF 的情况，写入存储器的数据饱和为最大正 1.15 的值，0x7FFF。对于输入数据小于 0xFF8000 的情况，写入存储器的数据饱和为最大负 1.15 的值，0x8000。

2.6.3.4 累加器“回写”

MAC 和 MSC 指令可选择将累加器的舍入值写入数据空间存储器，当然，是在该累加器不作为当前操作的目标单元的情况下。通过 X 总线对组合的 X 和 Y 地址空间执行此写操作。在某些 FFT 和 LMS 算法中，累加器的回写特性是很有好处的。

累加器回写硬件支持以下寻址模式：

1. W13，寄存器直接寻址：
非目标累加器的舍入内容以 1.15 小数结果写入 W13。
2. [W13]+=2，带后加的寄存器间接寻址：
非目标累加器的舍入内容以 1.15 小数结果写入 W13 指向的地址空间。然后 W13 加 2。

2.6.4 舍入逻辑

舍入逻辑在累加器写（存储）过程中可以执行传统的（偏置）或收敛的（无偏置）舍入功能。舍入模式由 RND（CORCON<1>）位的状态决定。它会产生一个 16 位的 1.15 数据值，该值被送到数据空间写饱和逻辑：如果此指令不指明舍入，就会存储一个截断的 1.15 数据值。

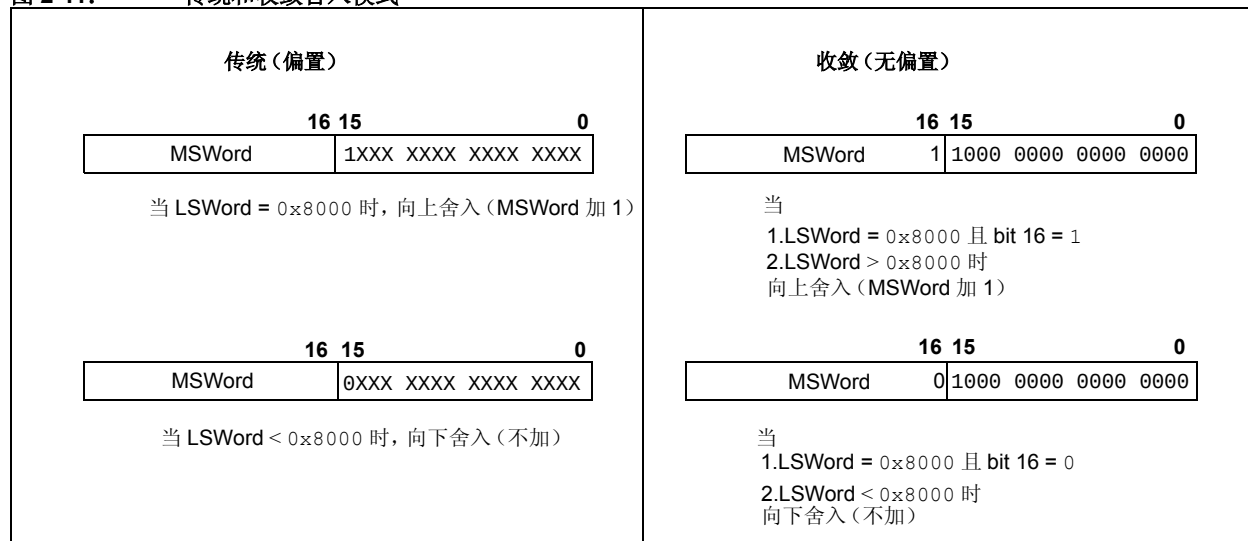
图 2-11 所示为两种舍入模式。传统舍入使用累加器的 bit 15，对它进行零扩展并将扩展值加到 MSWord（16 到 31 位）（保护和溢出位除外）。如果累加器的 LSWord 在 0x8000 和 0xFFFF 之间（包括 0x8000），MSWord 加 1。如果累加器的 LSWord 在 0x0000 和 0x7FFF 之间，MSWord 不变。此算法的结果对于一系列随机舍入操作，值将稍稍偏大。

除非 LSWord 等于 0x8000，否则收敛的（或不偏置）舍入操作方式与传统舍入操作方式相同。在这种情况下，要对 MSWord 的 LSbit（累加器的 bit 16）进行检测。如果它为“1”，MSWord 加 1。如果它为“0”，MSWord 不变。假设 bit 16 本身是随机的，此机制将消除任何可能累加的舍入偏置。

SAC 和 SAC.R 指令通过 X 总线将目标累加器中的内容以截断的（SAC）或舍入的（SAC.R）方式存入数据存储器（受饱和度影响，参见第 2.6.3.3 节“数据空间写饱和”）。

注意对于 MAC 类指令，累加器回数据写路径总是根据舍入操作而定的。

图 2-11： 传统和收敛舍入模式



2.6.5 桶形移位寄存器

桶形移位寄存器在单个周期内最多可算术右移 16 位或左移 16 位。DSP 指令或 MCU 指令可使用桶形移位器进行多位移位。

移位器需要一个带符号的二进制值确定移位操作的幅度（位数）和方向：

- 正值则将操作数右移
- 负值则将操作数左移
- 值为“0”则不改变操作数

桶形移位器为 40 位宽以适应累加器的宽度。为 DSP 移位操作提供了 40 位输出结果，而为 MCU 移位操作提供 16 位结果。

表 2-6 中提供了使用桶形移位器的所有指令。

表 2-6: 使用 DSP 引擎桶形移位器的指令

指令	描述
ASR	数据存储器单元的算术多位右移
LSR	数据存储器单元的逻辑多位右移
SL	数据存储器单元的多位左移
SAC	存储带可选移位的 DSP 累加器
SFTAC	移位 DSP 累加器

2.6.6 DSP 引擎模式选择

通过 CPU 内核配置寄存器（CORCON）可选择前面小节中讨论过的 DSP 引擎的各种可操作特性。这些特性如下所列：

- 小数或整数乘法运算。
- 传统或收敛舍入。
- ACCA 的自动饱和和开 / 关。
- ACCB 的自动饱和和开 / 关。
- 写数据存储器的自动饱和和开 / 关。
- 累加器饱和模式选择。

2.6.7 DSP 引擎陷阱事件

通过中断控制寄存器（INTCON1）可选择产生各种算术错误陷阱用于处理 DSP 引擎中的异常事件。如下所列：

- 使用 OVATE（INTCON1<10>）使能 ACCA 溢出陷阱。
- 使用 OVBTE（INTCON1<9>）使能 ACCB 溢出陷阱。
- 使用 COVTE（INTCON1<8>）使能灾难性的 ACCA 和 / 或 ACCB 溢出陷阱。

当用户使用 SFTAC 指令尝试移位一个超过最大允许范围（+/- 16 位）的值时也会产生算术错误陷阱。无法禁止此陷阱源。指令可执行完毕，但移位的结果不会被写入目标累加器。

有关 INTCON1 寄存器中的位和算术错误陷阱的更多信息，请参见第 6 章“复位中断”。

2.7 除法支持

dsPIC30F 支持以下类型的除法操作：

- DIVE: 16/16 有符号的小数除法
- DIV.SD: 32/16 有符号除法
- DIV.UD: 32/16 无符号除法
- DIV.SW: 16/16 有符号除法
- DIV.UW: 16/16 无符号除法

所有除法指令的商都被放在 W0 中，余数放在 W1 中。可以将 16 位除数放在任何一个 W 寄存器中。16 位被除数可以放在任何一个 W 寄存器中，而 32 位被除数必须放在相邻的两个 W 寄存器中。

所有的除法指令都是迭代操作且必须在一个 REPEAT 循环中执行 18 次。用户负责编程 REPEAT 指令。一个完整的除法操作需要执行 19 个指令周期。

除法流是可中断的，就象任何其他 REPEAT 循环一样。在循环的每次迭代后所有数据都存储在各自的数据寄存器中，因此由用户负责在 ISR 中保存相关的 W 寄存器。虽然 W 寄存器的中间值对于除法硬件很重要，但它们对于用户没有意义。在一个 REPEAT 循环中除法指令必须被执行 18 次才能产生一个有意义的结果。

除法指令的更多信息和编程示例，请参见 dsPIC30F Programmer's Reference Manual (DS70030)。

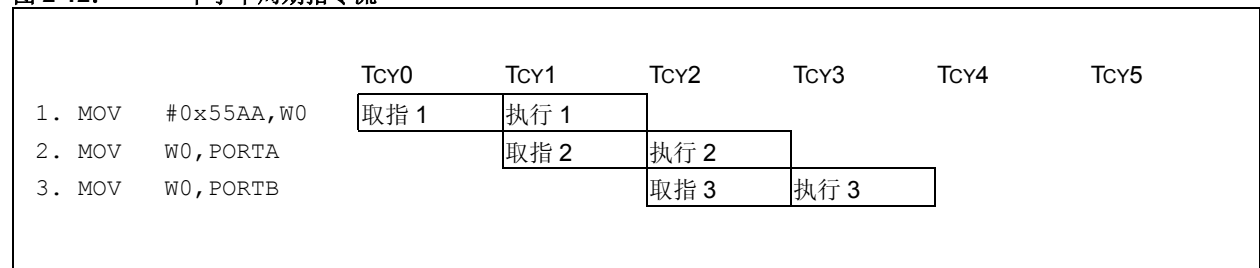
2.8 指令流类型

dsPIC30F 构架中的大部分指令占用程序存储器的一个字并在单个周期内执行。指令预取指机制方便了单周期（1 Tcy）执行。但是，某些指令需要执行 2 或 3 个指令周期。因此，dsPIC[®] 架构中有 7 种不同类型的指令流。下面对它们进行了说明：

1. 1 个指令字，1 个指令周期：

执行这些指令需要一个指令周期，如图 2-12 所示。大部分指令是单字单周期指令。

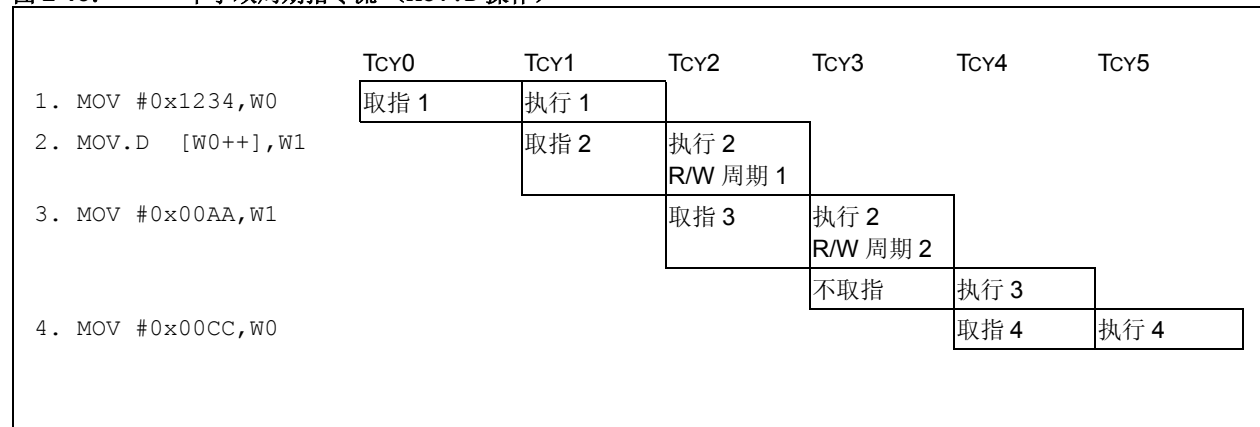
图 2-12: 单字单周期指令流



2. 1 个指令字，2 个指令周期：

在这些指令中，没有同时进行预取指。这一类型的指令只有 MOV.D 指令（装入和存储双字）。完成这些指令需要两个周期，如图 2-13 所示。

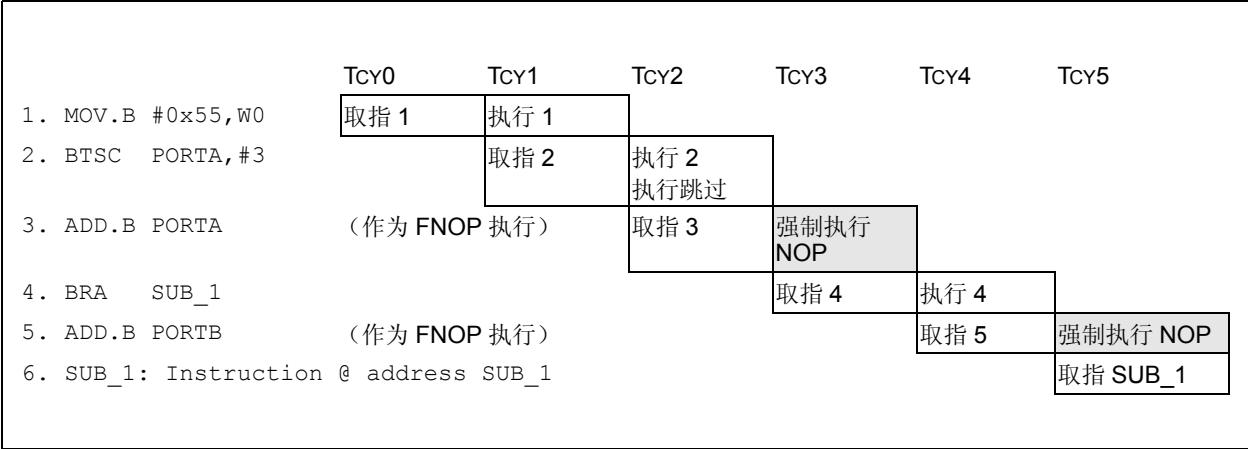
图 2-13: 单字双周期指令流（MOV.D 操作）



3. 1 个指令字，2 个或 3 个指令周期的程序流变化：

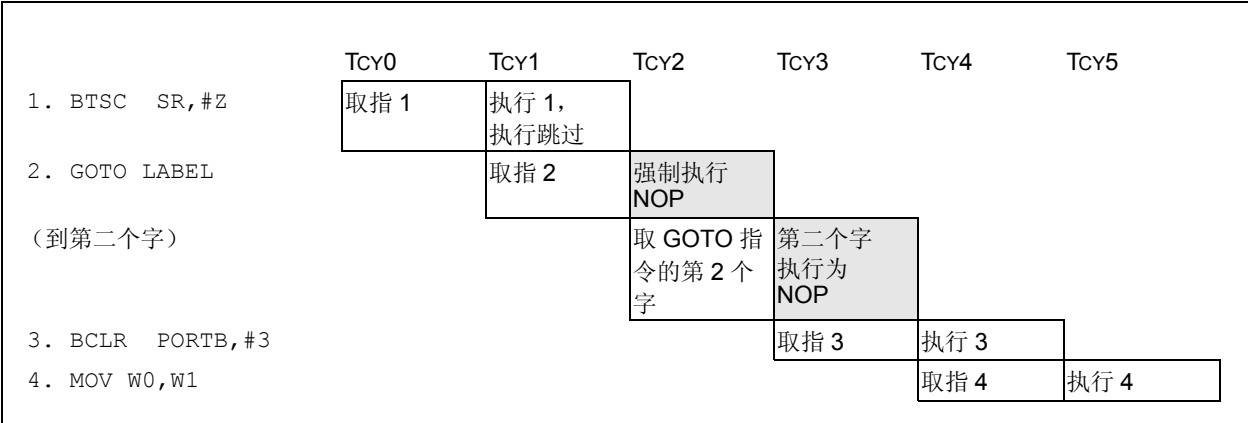
这些指令包括相对调用、转移指令以及跳过指令。当指令改变 PC 时（而非使它加计数），程序存储器预取指数数据必须被丢弃。这使指令执行需要两个有效周期，如图 2-4 所示。

图 2-14: 单字双周期指令流（程序流变化）



当双字指令被跳过时，则需三个周期。在这种情况下，程序存储器预取指数数据将被丢弃，而取出双字指令的第二个字。指令的第二个字作为 NOP 执行，如图 2-5 所示。

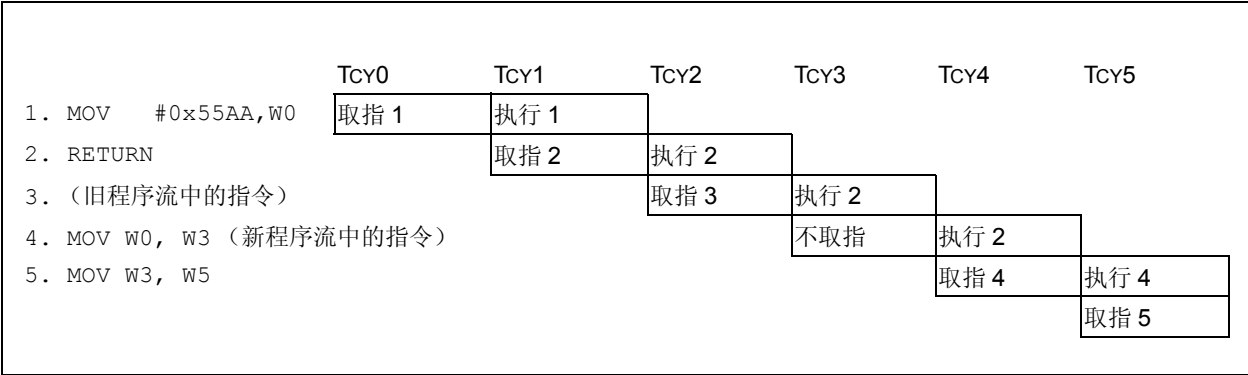
图 2-15: 单字 3 周期指令流（双字指令被跳过）



4. 1 个指令字，3 个指令周期（RETFIE、RETURN 和 RETLW）：

RETFIE、RETURN 和 RETLW 指令用于从子程序调用或中断服务程序返回，需要执行 3 个指令周期，如图 2-16 所示。

图 2-16: 单字 3 周期指令流（RETURN、RETFIE 和 RETLW）



5. 读表 / 写表指令:

这些指令会暂停取指以在程序存储器中插入一个读或写周期。如图 2-17 所示，表操作期间所取的指令将保存 1 个周期，并在紧随表操作之后的周期执行。

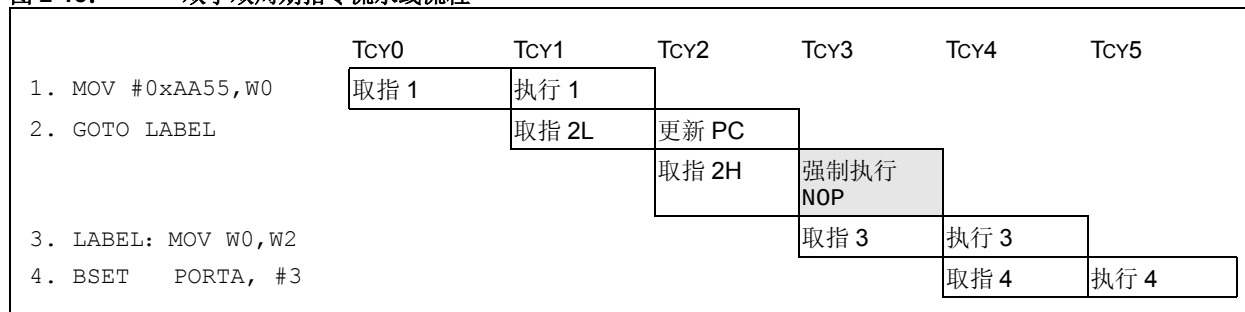
图 2-17: 指令流水线——流程表操作



6. 双指令字，双指令周期:

在这些指令中，指令后的取指包含数据。如图 2-18 所示，这会产生一个双周期指令。如果 CPU 只取了双自指令的第二和字而未取第一个字，则第二个字将被编码以作为 NOP 执行。这在双字指令被跳过指令（SKIP）跳过时尤其重要（参见图 2-15）。

图 2-18: 双字双周期指令流水线流程



7. 地址寄存器相依:

由于 X 数据空间的读写操作之间存在数据地址的相依性，这些指令会遭遇停顿（Stall）。为了解决这种资源冲突，插入了一个额外的周期（参见第 2.10 节“地址寄存器相依性”中的讨论）。

图 2-19: 单字单周期（带指令停顿）指令流水线流程



2.9 循环结构

dsPIC30F 支持 REPEAT 和 DO 指令结构，以提供无条件自动程序循环控制。REPEAT 指令用于实现单指令程序循环。DO 指令用于实现多指令程序循环。这两个指令都使用 CPU 状态寄存器 SR 中的控制位来临时修改 CPU 操作。

2.9.1 REPEAT 循环结构

REPEAT 指令会使紧随其后的一条指令重复一定次数。可以使用指令中的立即数或某个 W 寄存器中的值来指定重复的次数。W 寄存器选项可使循环计数为一软件变量。

REPEAT 循环中的指令至少执行一次。REPEAT 循环中的迭代次数是 14 位立即数值 +1 或 Wn+1。

下面列出了两种 REPEAT 指令的语法形式：

REPEAT #lit14 ; RCOUNT <-- lit14
(有效目标指令)

或

REPEAT Wn ; RCOUNT <-- Wn
(有效目标指令)

2.9.1.1 REPEAT 操作

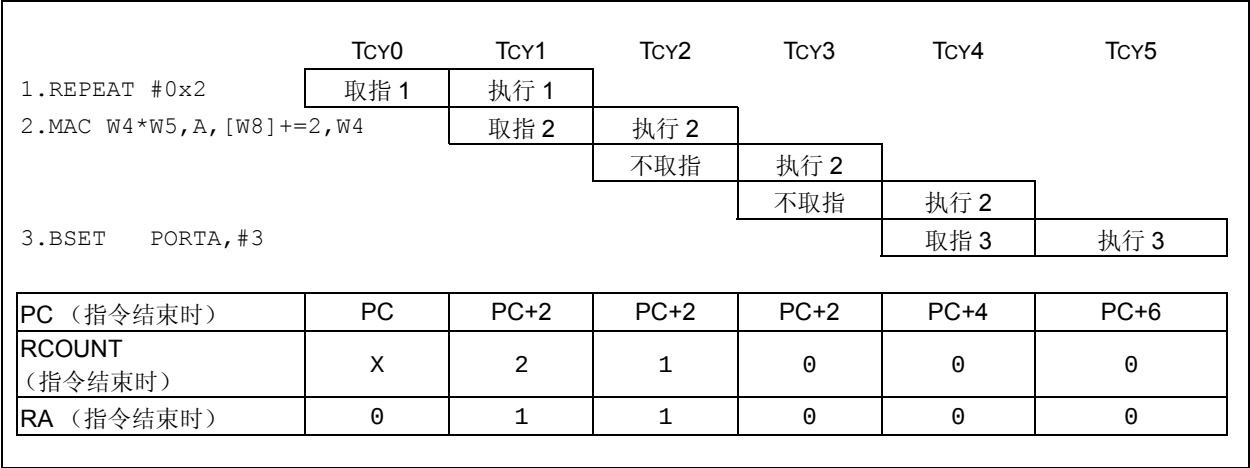
REPEAT 操作的循环计数保存在 14 位 RCOUNT 寄存器中，该寄存器是存储器映射的。RCOUNT 由 REPEAT 指令初始化。如果 RCOUNT 为非零值，REPEAT 指令将 REPEAT 有效状态位，或 RA (SR<4>) 状态位置 1。

RA 是一个只读位，不能用软件修改。对于大于 0 的 REPEAT 循环计数值，PC 不会递增。在 RCOUNT = 0 前，PC 递增被禁止。REPEAT 循环的指令流程示例请参见图 2-20。

对于等于 0 的循环计数值，REPEAT 的作用相当于 NOP，并且 RA (SR<4>) 位不置位。REPEAT 循环在开始前本质上是被禁止的，这样可以在预取后续指令时（即在正常的执行流程中）让目标指令只执行一次。

注： 紧随 REPEAT 指令之后的指令（即目标指令）总是至少执行一次。此指令的执行次数总是会比 14 位立即数或 W 寄存器操作数的指定值多一次。

图 2-20: REPEAT 指令流水线流程



2.9.1.2 中断 REPEAT 循环

REPEAT 指令循环可以在任何时候被中断。

在异常处理期间，RA 状态保留在堆栈中以便让用户在（任何数量的）嵌套中断中执行更多的 REPEAT 循环。SRL 存入堆栈后，RA 状态位被清零以便从 ISR 内部恢复正常执行流程。

注： 如果 REPEAT 循环被中断且正在处理 ISR，用户必须在 ISR 内部执行另一个 REPEAT 指令前先将 RCOUNT（REPEAT 计数寄存器）存入堆栈。

注： 如果在 ISR 内部使用 REPEAT，用户必须在执行 RETFIE 前先将 RCOUNT 的值出栈。

使用 RETFIE 从 ISR 返回 REPEAT 循环不需要任何特殊处理。中断会在 RETFIE 的第三个周期预取要重复的指令。当 SRL 寄存器被弹出堆栈时，堆栈的 RA 位将会恢复。此时如果它置位，那么中断的 REPEAT 循环将会恢复。

注： 如果重复的指令（REPEAT 循环中的目标指令）正使用 PSV 访问 PS 中的数据，从异常处理程序中返回后第一次执行该指令将需要两个指令周期。类似于循环中的第一次迭代，时序限制将不允许第一条指令在单个指令周期中访问位于 PS 中的数据。

2.9.1.2.1 REPEAT 循环的提前终止

通过用软件将 RCOUNT 寄存器清零，可以在 ISR 中比正常情况提前终止中断的 REPEAT 循环。

2.9.1.3 REPEAT 指令的限制

除了以下指令外，其他指令均可紧随 REPEAT 之后：

1. 程序流控制指令（任何转移、比较和跳过、子程序调用或返回等指令）
2. 另一个 REPEAT 或 DO 指令。
3. DISI、ULNK、LNK、PWRSV 或 RESET。
4. MOV.D 指令。

注： 某些指令和/或指令寻址模式可以在 REPEAT 循环中执行，但是其循环基本没有意义。

2.9.2 DO 循环结构

DO 指令能将一组跟在它后面的指令执行指定次数而无需软件开销。到结束地址（包括该地址）的指令集都会被重复执行。DO 指令的重复计数值可由指令中声明的一个 14 位立即数或 W 寄存器的内容指定。下面列出了两种 DO 指令的语法形式：

```
DO      #lit14, LOOP_END      ; DCOUNT <-- lit14
指令 1
指令 2
:
:
LOOP_END: 指令 n

DO      Wn, LOOP_END          ; DCOUNT <-- Wn<13:0>
指令 1
指令 2
:
:
LOOP_END: 指令 n
```

DO 循环结构提供了以下功能：

- 可以使用一个 W 寄存器来指定循环次数。这使得循环次数可在运行时指定。
- 不需要顺序执行指令（即可以使用转移或子程序调用等）。
- 循环结束地址不一定要大于起始地址。

2.9.2.1 DO 循环寄存器及其工作原理

DO 循环执行的迭代次数是（14 位立即数值 +1）或（Wn 值 +1）。如果使用 W 寄存器指定迭代的次数，则 W 寄存器的两个 MSb 不用于指定循环计数。DO 循环的操作类似于 C 编程语言中的“do-while”结构，因为循环中的指令总是至少会执行一次。

dsPIC30F 有三个和 DO 循环相关的寄存器：DOSTART、DOEND 和 DCOUNT。这些寄存器都是存储器映射的并在 DO 指令执行时由硬件自动装入。DOSTART 保存 DO 循环的起始地址，DOEND 则保存 DO 循环的结束地址。DCOUNT 寄存器保存循环要执行的迭代次数。DOSTART 和 DOEND 是保存 PC 值的 22 位寄存器。这些寄存器的 MSb 和 LSb 固定为 0。更多详细信息请参见图 2-2。因为 PC<0> 总是被强制设为 0，所以 LSb 不保存在这些寄存器中。

DA 状态位（SR<9>）表明单个 DO 循环（或嵌套 DO 循环）正在进行中。当执行了 DO 指令时，DA 位会置位使能 PC 地址和 DOEND 寄存器在随后每个指令周期的比较。当 PC 与 DOEND 中的值匹配时，DCOUNT 会递减。如果 DCOUNT 寄存器非零，PC 中会装入 DOSTART 寄存器所包含的地址，以便开始另一个 DO 循环迭代。

当 DCOUNT = 0 时，DO 循环会终止。如果没有其他嵌套的 DO 循环在进行中，则 DA 位也会被清零。

注： DO 循环结构中的指令组总是会至少执行一次。DO 循环的执行次数总是会比立即数或 W 寄存器操作数的指定值多一次。

2.9.2.2 DO 循环嵌套

DOSTART、DOEND 和 DCOUNT 寄存器都有一个与之相关的影子寄存器，这样 DO 循环硬件就能支持一层自动嵌套。用户可以访问 DOSTART、DOEND 和 DCOUNT 寄存器，而且在需要时，这些寄存器可以被手工允许更多的嵌套。

DO 级别位 DL<2:0> (CORCON<10:8>) 表示当前执行的 DO 循环的嵌套级别。当第一个 DO 指令执行后，DL<2:0> 被设为 B “001” 以表明正在进行一级 DO 循环。DA (SR<9>) 也会置位。当另一个 DO 指令在第一个 DO 循环中被执行时，在被新的循环值更新前，DOSTART、DOEND 和 DCOUNT 寄存器被传输到影子寄存器中。DL<2:0> 位被置为 B “010”，表明第二个嵌套的 DO 循环正在进行。DA (SR<9>) 也会保持置位。

如果在应用中不需要超过一级的 DO 循环嵌套，就没什么需要特别注意之处。如果用户需要一级以上的 DO 循环嵌套，可以通过在执行下一个 DO 循环前手工保存 DOSTART、DOEND 和 DCOUNT 寄存器来实现。只要 DL<2:0> 为 B “010” 或更大的值，就应该保存这些寄存器。

当 DO 循环终止且 DL<2:0> = B “010” 时，DOSTART、DOEND 和 DCOUNT 寄存器会自动从其影子寄存器中恢复。

注： DL<2:0> (CORCON<10:8>) 位被组合（经过逻辑“或”）以形成 DA (SR<9>) 位的值。如果正在执行嵌套的 DO 循环，仅当与最外层循环关联的循环计数满时，DA 位才会被清零。

2.9.2.3 中断 DO 循环

DO 循环可以在任何时候被中断。如果在 ISR 中有另一个 DO 循环要执行，用户必须检查 DL<2:0> 状态位并按需要保存 DOSTART、DOEND 和 DCOUNT 寄存器。

如果用户能确保在以下情况下只有一级 DO 循环会被执行，则无需进行特别处理：

- 后台和任何 ISR 处理程序中（如果使能了中断嵌套）或
- 后台或任何 ISR（如果禁止了中断嵌套）

或者，可有最多两个（嵌套的）DO 循环在后台执行，或在任何：

- 一个 ISR 处理程序（如果使能了中断嵌套），或
- ISR（如果禁止了中断嵌套）中

我们假设在任何陷阱处理程序中都不使用 DO 循环。

使用 RETFIE 指令从 ISR 返回 DO 循环不需要任何特殊处理。

2.9.2.4 提前终止 DO 循环

有两种方法可以在正常情况以前终止 DO 循环：

1. EDT (CORCON<11>) 位可以让用户在 DO 循环完成所有循环前终止 DO 循环。在 EDT 位写入 1 将强制循环完成正在进行的迭代，然后终止。如果 EDT 在循环的倒数第二条或最后一条指令执行期间置位，会再进行一次循环迭代。EDT 会始终读作 0，对其清零没有影响。当 EDT 位置位后，用户也可选择跳出 DO 循环。
2. 或者，代码可以在任何点跳出循环，但是最后一个不可能是流控制指令的指令除外。虽然 DA 位会使能 DO 循环硬件，但是除非在预取指时遇到了倒数第二个指令的地址，否则它不起作用。不推荐使用此方法终止 DO 循环。

注： 因为硬件会继续检查 DOEND 地址，所以不推荐不使用 EDT 来退出 DO 循环。

2.9.2.5 DO 循环限制

DO 循环有以下各项限制：

- 选择循环中的最后指令
- 循环长度（离第一个指令的偏移量）
- 读取 DOEND 寄存器

所有的 DO 循环必须包含至少 2 条指令，因为循环终止测试是在倒数第二条指令中执行的。对于单指令循环，应该使用 REPEAT。

紧接在 DO 指令或对 DOEND SFR 进行数据寄存器写操作后，用户软件无法用指令读取特殊功能寄存器 DOEND。

在 DO 循环最后一条指令前两个指令执行的指令不应该修改以下任一项：

- IPL（SR<7:5>）位管理的 CPU 优先级
- IEC0、IEC1 和 IEC2 寄存器管理的外设中断使能位
- IPC0 到 IPC11 寄存器管理的外设中断优先级位

如果不遵守上述限制，DO 循环的执行可能会不正确。

2.9.2.5.2 最末指令限制

DO 循环中执行的最后一条指令有一定的限制。DO 循环中最后一条指令不能是：

1. 流程控制指令（例如，任何转移、比较并跳过、GOTO、CALL、RCALL 或 TRAP）。
2. RETURN、RETFIE 和 RETLW 作为 DO 循环的最后一条指令可以正常工作，但必须由用户返回到循环中以完成循环。
3. 另一个 REPEAT 或 DO 指令。
4. REPEAT 循环的目标指令。由此限制推出倒数第二条指令也不能是 REPEAT。
5. 任何在程序空间占据两个字的指令。
6. DISI 指令

2.9.2.5.3 循环长度限制

循环长度指 DO 循环中最后一条指令相对于第一条指令的有符号偏移。将循环中第一条指令的地址与循环长度相加，就会得到循环中最后一条指令的地址。有些循环长度值应该被避免。

1. 循环长度 = -2

执行会从循环的第一条指令开始（即从 [PC]）并一直进行，直到预取了循环的结束地址（在此例中为 [PC-4]）。因为这是 DO 指令的第一个字，所以它会再次执行 DO 指令，重新初始化 DCOUNT 并预取 [PC]。在预取循环的结束地址 [PC-4] 之前，这个过程就会永远进行下去。这个 n 值有可能会造成无限循环（受到看门狗定时器复位的影响）。

```
end_loop: DO #33, end_loop ;DO is a two-word instruction
          NOP              ;2nd word of DO executes as a NOP
          ADD W2,W3,W4      ;First instruction in DO loop([PC])
```

2. 循环长度 = -1

执行会从循环的第一条指令开始（即从 [PC]）并一直进行，直到预取了循环的结束地址（[PC-2]）。因为循环的结束地址是 DO 指令的第二个字，它将作为 NOP 执行，但是仍然会预取 [PC]。然后此循环会再次执行。只要循环的结束地址 [PC-2] 被预取指，循环就不会终止，此过程会一直进行。如果 DCOUNT 寄存器的值变为零并且后续的减操作导致借位，循环将会终止。但是，在这种情况下，循环外的第一条指令又将是第一条循环指令。

```
DO #33, end_loop ;DO is a two-word instruction
end_loop: NOP      ;2nd word of DO executes as a NOP
          ADD W2,W3,W4 ;First instruction in DO loop([PC])
```

3. 循环长度 = 0

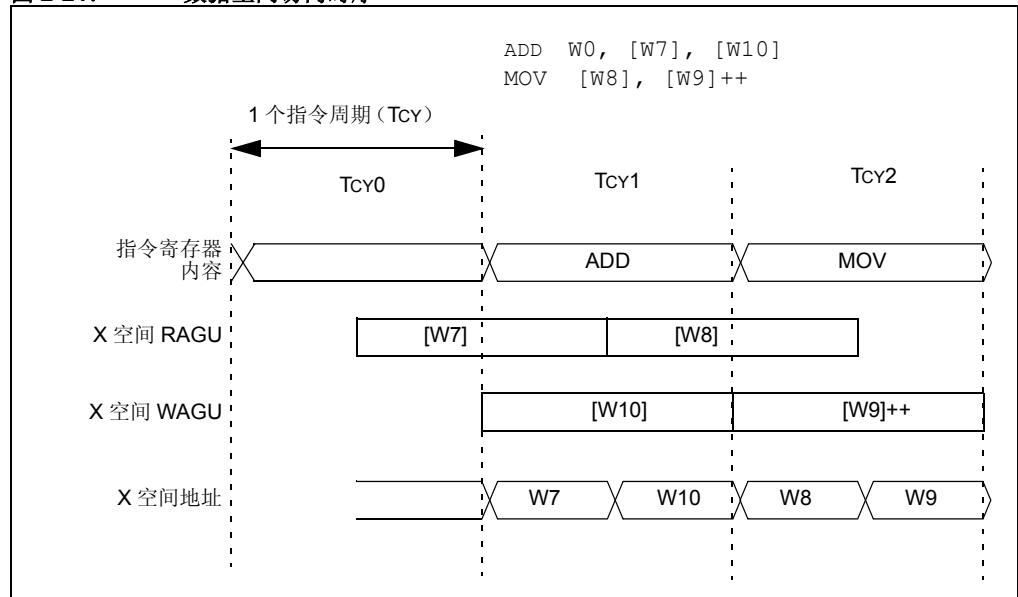
执行会从循环的第一条指令开始（即从 [PC]）并一直进行，直到预取了循环的结束地址（[PC]）。如果循环要继续，此预取指将导致 DO 循环硬件将 DOEND 地址（[PC]）装入 PC 供下一次取指（仍将是 [PC]）。在循环的第一次真正的迭代后，循环的第一条指令将会反复执行，直到循环计数下溢，循环终止。当发生此情况时，循环外的第一条指令将是 [PC] 后的那条指令。

```
DO #33, end_loop ;DO is a two-word instruction
NOP              ;2nd word of DO executes as a NOP
end_loop: ADD W2,W3,W4 ;First instruction in DO loop([PC])
```

2.10 地址寄存器相依性

dsPIC30F 架构支持大部分 MCU 类指令的数据空间读取（源）和数据空间写入（目标）操作。AGU 进行的有效地址（EA）计算和后续数据空间读或写，每个都需要 1 个指令周期来完成。如图 2-21 所示，此时序导致每个指令的数据空间读写操作部分重叠。因为此重叠，“先写后读”（Read-After-Write, RAW）数据相依性可能会在指令边界处发生。RAW 数据相依性在运行时由 dsPIC30F CPU 检测并处理。

图 2-21: 数据空间访问时序



2.10.1 先写后读相依性规则

如果 **W** 寄存器在当前指令中被用作写操作目标，并且预取的指令读取的也是同一个 **W** 寄存器，则适用下列规则：

1. 如果对目标的写入（当前指令）不修改 **Wn** 的内容，则不发生停顿。
或
2. 如果对源的读取（预取的指令）不使用 **Wn** 计算 **EA**，也不会发生停顿。

在每个指令周期中，**dsPIC30F** 硬件会自动检查以确定是否将发生 **RAW** 数据相依性。如果不足上述条件，**CPU** 会在执行预取的指令前自动增加一个指令周期的延时。指令停顿可以在下一条（预取的）指令要使用写入的数据前，使目标 **W** 寄存器有足够的时间执行写入。

表 2-7： 先写后读相依性总结

使用 Wn 的目标寻址模式	使用 Wn 的源寻址模式	状态	示例 (Wn = W2)
直接寻址	直接寻址	允许	ADD.w W0, W1, W2 MOV.w W2, W3
直接寻址	间接寻址	停顿	ADD.w W0, W1, W2 MOV.w [W2], W3
直接寻址	带有修改的间接寻址	停顿	ADD.w W0, W1, W2 MOV.w [W2++], W3
间接寻址	直接寻址	允许	ADD.w W0, W1, [W2] MOV.w W2, W3
间接寻址	间接寻址	允许	ADD.w W0, W1, [W2] MOV.w [W2], W3
间接寻址	带有修改的间接寻址	允许	ADD.w W0, W1, [W2] MOV.w [W2++], W3
带有修改的间接寻址	直接寻址	允许	ADD.w W0, W1, [W2++] MOV.w W2, W3
间接寻址	间接寻址	停顿	ADD.w W0, W1, [W2] MOV.w [W2], W3 ; W2=0x0004 (mapped W2)
间接寻址	带有修改的间接寻址	停顿	ADD.w W0, W1, [W2] MOV.w [W2++], W3 ; W2=0x0004 (mapped W2)
带有修改的间接寻址	间接寻址	停顿	ADD.w W0, W1, [W2++] MOV.w [W2], W3
带有修改的间接寻址	带有修改的间接寻址	停顿	ADD.w W0, W1, [W2++] MOV.w [W2++], W3

2.10.2 指令停顿周期

指令停顿本质上是附加在指令读阶段前的一个指令周期的等待时间，以便让前面的写操作先完成再发生下一个读操作。为了达到中断延迟的目的，应该注意停顿周期与检测到它的指令后的那个指令是相关的（即停顿周期总是在指令执行周期之前）。

如果检测到了 RAW 数据相依性，dsPIC30F 将开始指令停顿周期。在指令停顿期间，会发生下列事件：

1. 正在进行的（上一条指令的）写操作可以正常完成。
2. 在指令停顿周期结束前不会寻址数据空间。
3. 在指令停顿周期结束前禁止 PC 递增。
4. 指令停顿周期结束前禁止再次取指。

2.10.2.1 指令停顿周期和中断

当会造成指令停顿的两个相邻指令与中断事件同时发生时，可能会产生以下两个结果之一：

1. 中断可能会在第一个指令执行时发生。这种情况下，允许第一条指令完成，而第二个指令则将在 ISR 完成后执行。这样，因为异常过程为第一条指令提供了完成写阶段的时间，停顿周期将在第二条指令中被消除。
2. 中断可能会在第二条指令执行时发生。在这种情况下，允许第二条指令和附加的停顿周期在 ISR 前执行。这样，与第二条指令关联的停顿周期会正常执行。但是，停顿周期实际上会被嵌入到异常过程时序内。如果一个正常的双周期指令被中断，异常过程将会继续。

2.10.2.2 指令停顿周期和流程更改指令

CALL和RCALL指令使用W15写入堆栈，并且如果下一条指令读取的源使用W15，可能会因此在下一次指令前强制执行指令停顿。

RETFIE和RETURN指令永远不能在下一条指令前强制执行指令停顿，因为这些指令都只能执行读操作。但是，用户应该注意 RETLW 指令能强制执行停顿，因为它会在最后一个周期写入 W 寄存器。

因为 GOTO 和转移指令不执行写操作，所以永远不能强制执行指令停顿。

2.10.2.3 指令停顿以及 DO 和 REPEAT 循环

除了增加指令停顿周期外，RAW 数据相依性不会影响 DO 或 REPEAT 循环的工作。

REPEAT 循环中预取的指令在循环完成或发生异常前不会改变。虽然寄存器相关性检查会跨指令边界进行，在 REPEAT 循环中 dsPIC30F 实际上会比较同一个指令的源和目标地址。

DO 循环的最后一条指令会预取循环起始地址处的指令或下一条指令（循环外）。指令停顿的决定是由循环的最后一条指令和预取指令的内容做出的。

2.10.2.4 指令停顿和程序空间可视性（PSV）

当通过使能 PSV（CORCON<2>）位将程序空间（PS）映射到数据空间，并且 X 空间 EA 处于可见程序空间范围时，读或写周期会被重新定向到程序空间中的地址。从程序空间访问数据最多需要花费 3 个指令周期。

PSV地址空间的指令操作与任何其他指令一样，会受到RAW数据相关性和后续指令停顿的影响。

考虑以下代码段：

```
ADD    W0, [W1], [W2++]    ; PSV = 1, W1=0x8000, PSVPAG=0xAA
MOV    [W2], [W3]
```

此指令序列将需要 5 个指令周期来执行。增加的 2 个指令周期用于通过 W1 执行 PSV 访问。此外，为了解决 W2 造成的 RAW 数据相关性，插入了一个指令停顿周期。

2.11 寄存器映射

表 2-8 提供了与 dsPIC30F CPU 内核相关的寄存器的总结。

表 2-8: dsPIC30F 内核寄存器映射

名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位状态	
W0	0000	W0 (WREG)																0000 0000 0000 0000	
W1	0002	W1																0000 0000 0000 0000	
W2	0004	W2																0000 0000 0000 0000	
W3	0006	W3																0000 0000 0000 0000	
W4	0008	W4																0000 0000 0000 0000	
W5	000A	W5																0000 0000 0000 0000	
W6	000C	W6																0000 0000 0000 0000	
W7	000E	W7																0000 0000 0000 0000	
W8	0010	W8																0000 0000 0000 0000	
W9	0012	W9																0000 0000 0000 0000	
W10	0014	W10																0000 0000 0000 0000	
W11	0016	W11																0000 0000 0000 0000	
W12	0018	W12																0000 0000 0000 0000	
W13	001A	W13																0000 0000 0000 0000	
W14	001C	W14																0000 0000 0000 0000	
W15	001E	W15																0000 0000 0000 0000	
SPLIM	0020	SPLIM																0000 0000 0000 0000	
ACCAL	0022	ACCAL																0000 0000 0000 0000	
ACCAH	0024	ACCAH																0000 0000 0000 0000	
ACCAU	0026	ACCA<39> 的符号扩展									ACCAU							0000 0000 0000 0000	
ACCBL	0028	ACCBL																0000 0000 0000 0000	
ACCBH	002A	ACCBH																0000 0000 0000 0000	
ACCBU	002C	ACCB<39> 的符号扩展									ACCBU							0000 0000 0000 0000	
PCL	002E	PCL																0000 0000 0000 0000	
PCH	0030	—	—	—	—	—	—	—	—	—	PCH							0	0000 0000 0000 0000
TBLPAG	0032	—	—	—	—	—	—	—	—	TBLPAG								0000 0000 0000 0000	
PSVPAG	0034	—	—	—	—	—	—	—	—	PSVPAG								0000 0000 0000 0000	
RCOUNT	0036	RCOUNT																xxxx xxxx xxxx xxxx	
DCOUNT	0038	DCOUNT																xxxx xxxx xxxx xxxx	
DOSTARTL	003A	DOSTARTL																0	xxxx xxxx xxxx xxx0
DOSTARTH	003C	—	—	—	—	—	—	—	—	—	DOSTARTH								0000 0000 00xx xxxx
DOENDL	003E	DOENDL																0	xxxx xxxx xxxx xxx0
DOENDH	0040	—	—	—	—	—	—	—	—	—	DOENDH								0000 0000 00xx xxxx
SR	0042	OA	OB	SA	SB	OAB	SAB	DA	DC	IPL2	IPL1	IPL0	RA	N	OV	Z	C	0000 0000 0000 0000	

表 2-8: dsPIC30F 内核寄存器映射 (续)

名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位状态	
CORCON	0044	—	—	—	US	EDT	DL2	DL<1:0>		SATA	SATB	SATDW	ACCSAT	IPL3	PSV	RND	IF	0000 0000 0010 0000	
MODCON	0046	XMODEN	YMODEN	—	—	BWM<3:0>				YWM<3:0>				XWM<3:0>				0000 0000 0000 0000	
XMODSRT	0048	XMODSRT<15:0>															0	xxxx xxxx xxxx xxx0	
XMODEND	004A	XMODEND<15:0>															1	xxxx xxxx xxxx xxx1	
YMODSRT	004C	YMODSRT<15:0>															0	xxxx xxxx xxxx xxx0	
YMODEND	004E	YMODEND<15:0>															1	xxxx xxxx xxxx xxx1	
XBREV	0050	BREN	XBREV<14:0>															xxxx xxxx xxxx xxxx	
DISICNT	0052	—	—	DISICNT<13:0>															0000 0000 0000 0000
保留	0054 - 007E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000	

图注: x = 未初始化
注: 如需具体内核寄存器映射的详细信息, 请参见器件数据手册。

2.12 相关应用笔记

本节列出了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。目前与 dsPIC30F CPU 模块相关的应用笔记有：

标题

应用笔记编号

目前没有相关的应用笔记。

注： 如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

2.13 版本历史

版本 A

这是本文档的初始发行版。

版本 B

此版本添加了 dsPIC30F CPU 模块的更多技术内容。

版本 C

此版本加入了本文档更新时已知的所有勘误表。

注:

第 3 章 数据存储器

目录

本章包括下列主要内容：

3.1	简介	3-2
3.2	数据空间地址发生器单元（AGU）	3-5
3.3	模寻址	3-7
3.4	位反转寻址	3-14
3.5	控制寄存器说明	3-18
3.6	相关应用笔记	3-23
3.7	版本历史	3-24

3.1 简介

dsPIC30F 数据宽度为 16 位。所有内部寄存器和数据空间存储器都是以 16 位宽度组织的。dsPIC30F 的具有两个数据空间。可以单独访问数据空间（对于某些 DSP 指令）或将它作为一个 64 KB 线性地址范围一起访问（对于 MCU 指令）。使用两个地址发生单元（Address Generation Units, AGU）和独立的地址路径访问数据空间。

图 3-1 所示为数据空间存储器映射图示例。

数据存储器 0x0000 到 0x07FF 之间的地址空间保留用于器件的特殊功能寄存器（SFR）。SFR 包含 CPU 和器件上的外设的控制和状态位。

RAM 从地址 0x0800 开始，分成两个区块，分别为 X 和 Y 数据空间。对于数据写操作，总是将 X 和 Y 数据空间作为一个线性数据空间访问。对于数据读操作，可以分别单独访问 X 和 Y 存储器空间或将它们作为一个线性空间访问。用 MCU 类指令进行的数据读操作总是将 X 和 Y 数据空间作为一个组合的数据空间访问。具有两个源操作数的 DSP 指令（如 MAC 指令）分别单独访问 X 和 Y 数据空间以支持同时对这两个源操作数进行读操作。

MCU 指令可以使用任何 W 寄存器作为地址指针进行数据读写操作。

在数据读过程中，DSP 类指令将 Y 地址空间与整个数据空间隔开。W10 和 W11 作为从 Y 数据空间读取数据的地址指针。剩下的数据空间被称为 X 空间，其实更精确的应描述为“X 减 Y”空间。W8 和 W9 作为使用 DSP 类指令对 X 数据空间读取数据的地址指针。

图 3-2 所示为使用 MCU 类和 DSP 类指令时，数据存储器空间是如何映射的。注意 W 寄存器数和指令类型将决定在进行数据读操作时如何访问地址空间。尤其需要注意的是，MCU 指令将 X 和 Y 存储器作为一个组合的数据空间访问。MCU 指令可以使用任何 W 寄存器作为地址指针进行读写操作。DSP 指令可以同时预取指两个数据操作数，它将数据存储器分割为两个空间。在这种情况下必须使用指定的 W 寄存器作为读操作的地址指针。

某些 DSP 指令能够将不作为该指令目标的累加器存储到数据存储器。此功能叫做“累加器回写”。累加器回写操作必须使用 W13 作为地址指针，用来指向组合数据存储器空间。

对于 DSP 类指令，所有存储器读操作时，W8 和 W9 均应指向实现的 X 存储器空间。如果 W8 或 W9 指向 Y 存储器空间，将返回为零。如果 W8 或 W9 指向未用的存储器地址，会产生一个地址错误陷阱。

对于 DSP 类指令，所有存储器读操作时，W10 和 W11 均应指向实现的 Y 存储器空间。如果 W10 或 W11 指向实现的 X 存储器空间，将返回为零。如果 W10 或 W11 指向未用的存储器地址，会产生一个地址错误陷阱。有关地址错误陷阱的其他信息，请参见第 6 章“复位中断”。

注： 数据存储器映射以及 X 和 Y 数据空间的划分视具体器件而定。更多详细信息，请参见特定的 dsPIC30F 器件数据手册。

图 3-1: 数据存储器映射示例

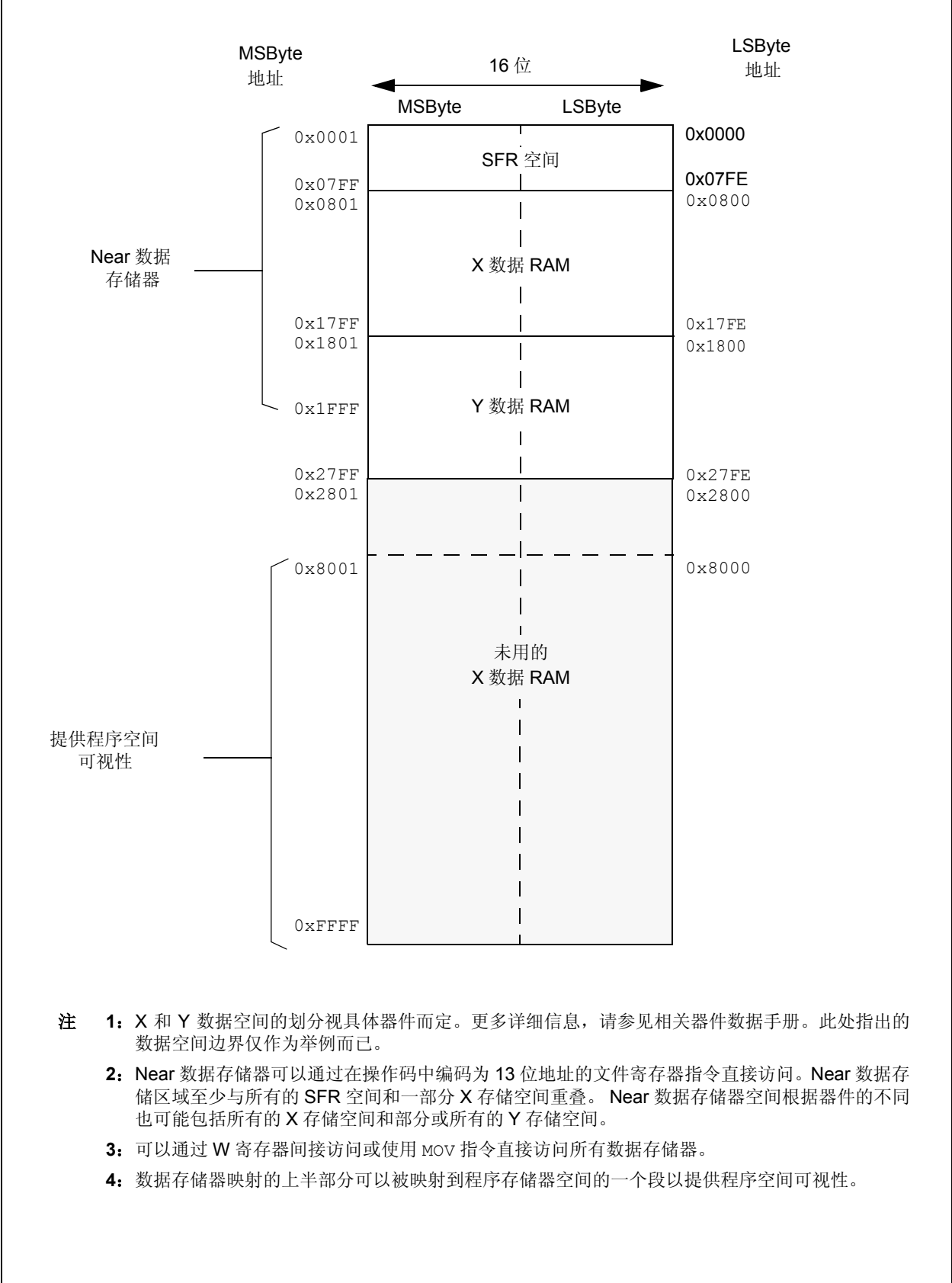
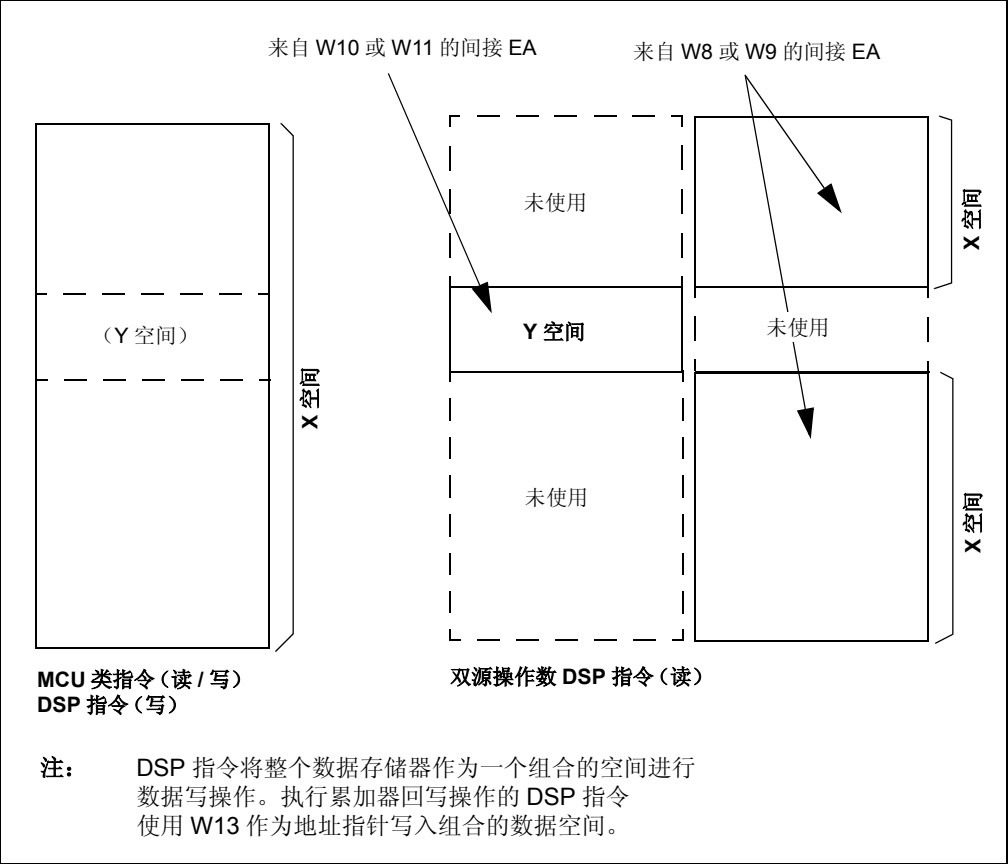


图 3-2: MCU 和 DSP 指令的数据空间



3.1.1 Near 数据存储器

在数据存储空间的 0x0000 到 0x1FFF 之间保留了一个 8 KB 的地址空间 (称为 Near 数据存储器)。可通过所有文件寄存器指令中的 13 位绝对地址字段直接对 Near 数据存储器寻址。

Near 数据区域中所包含的存储器区域由各个不同的 dsPIC30F 器件所实现的数据存储器数量决定。Near 数据存储器区域至少将包含所有 SFR 和某些 X 数据存储器。对于具备较小数据存储空间的器件, Near 数据区域可能包括所有 X 存储空间和部分或全部的 Y 存储空间。更多细节请参见图 3-1。

注: 可以使用 MOV 指令直接寻址整个 64K 数据空间。更多详细信息, 请参见 dsPIC30F Programmer's Reference Manual (DS70030)。

3.2 数据空间地址发生器单元 (AGU)

dsPIC30F 包含一个 X AGU 和一个 Y AGU 以产生数据存储器地址。X 和 Y AGU 都可以产生任何 64 KB 范围内的有效地址 (EA)。但是，对物理存储器范围以外的 EA 进行数据读操作会返回全零；对物理存储器范围以外的 EA 进行数据写操作无效。此外，还会产生地址错误陷阱。有关地址错误陷阱的更多信息，请参见第 6 章 “复位中断”。

3.2.1 X 地址发生器单元

X AGU 可以被所有指令使用并支持所有寻址模式。X AGU 由一个读 AGU (X RAGU) 和一个写 AGU (X WAGU) 组成，它们可以在指令周期的不同阶段，各自独立地在不同的读写总线上进行操作。所有将 X 和 Y 数据空间看作一个组合的数据空间的指令均将 X 读数据总线作为返回数据路径。X 数据总线也是双操作数读指令 (DSP 类指令) 的 X 地址空间数据路径。对于所有指令来说，X 写数据总线是写入组合的 X 和 Y 数据空间的唯一路径。

X RAGU 在上一个指令周期里使用刚刚预取的指令中的信息开始计算它的有效地址。在指令周期开始时，X RAGU EA 即出现在地址总线上。

X WAGU 在指令周期开始时开始计算它的有效地址。在指令的写操作阶段，EA 出现在地址总线上。

X RAGU 和 X WAGU 都支持模寻址。

只有 X WAGU 支持位反转寻址。

3.2.2 Y 地址发生器单元

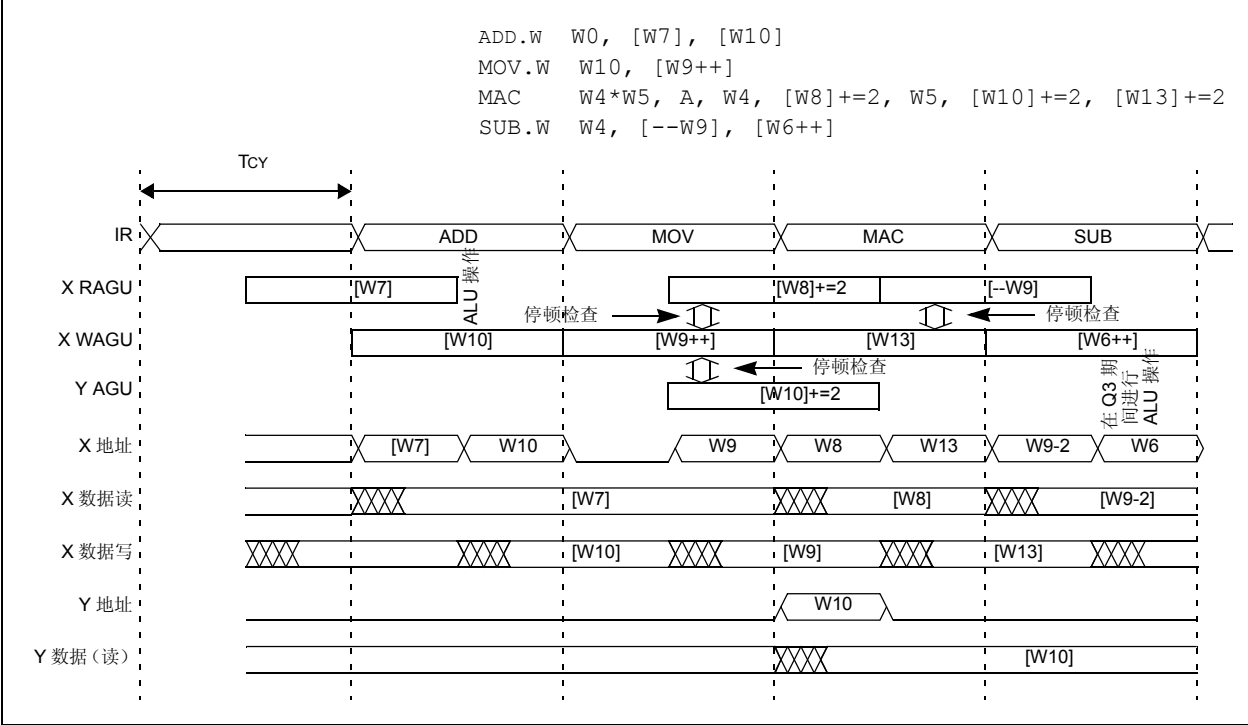
Y 数据存储空间有一个 AGU，支持从 Y 数据存储空间进行数据读操作。从不使用 Y 存储器总线进行数据写操作。Y AGU 和 Y 存储器总线的功能是支持 DSP 类指令同时进行数据读操作。

Y AGU 时序与 X RAGU 时序的相同之处为，它在上一个指令周期里使用预取的指令中的信息开始计算它的有效地址。在指令周期开始时，EA 即出现在地址总线上。

对于使用 Y AGU 的 DSP 类指令，Y AGU 支持模寻址和后修改寻址模式。

注： Y AGU 不支持数据写操作。所有数据通过 X WAGU 写入组合的 X 和 Y 数据空间。只有在双源操作数 DSP 指令的数据读过程中使用 Y AGU。

图 3-3: 数据空间访问时序



3.2.3 地址发生器单元和 DSP 类指令

DSP 类指令将 Y AGU 和 Y 存储器数据路径与 X RAGU 一起使用，从而提供两条可同时对数据进行读操作的路径。例如，MAC 指令可以同时预取指两个操作数用于下一条乘法运算。

DSP 类指令指定两个 W 寄存器指针（W8 和 W9），总是通过 X RAGU 进行操作并寻址 X 数据空间（不寻址 Y 数据空间）另外指定两个 W 寄存器指针（W10 和 W11），总是通过 Y AGU 进行操作并寻址 Y 数据空间（不寻址 X 数据空间）。任何由 DSP 类指令执行的数据写操作都发生在组合的 X 和 Y 数据空间中，并且该写操作通过 X 总线实现。因此，可以对任何地址执行写操作而不管 EA 指向的地址。

Y AGU 只支持与 DSP 类指令相关的后修改寻址模式。更多有关寻址模式的信息，请参见 dsPIC30F Programmer's Reference Manual（DS70030）。Y AGU 也支持自动循环缓冲区的模寻址。当 X AGU 被视为组合线性空间的一部分时，所有其他（MCU）类指令可以通过 X AGU 访问 Y 数据地址空间。

3.2.4 数据对齐方式

ISA 支持对通过 X 存储器 AGU 访问数据的所有 MCU 指令进行的字和字节操作。在字操作中，16 位数据地址的 LSB 被忽略。字数据是以小尾数（Little Edian）格式对齐的，在这种对齐方式下，LSByte 放在偶数地址（LSB = 0），而 MSByte 放在奇数地址（LSB = 1）。对于字节操作，用数据地址的 LSB 来选择所访问的字节。寻址的字节放在内部数据总线的低 8 位。根据所执行的是字节还是字访问，将自动调整所有有效地址计算。例如，对地址指针进行后加计数的字操作将使地址加 2。

注： 所有字访问必须与偶数地址（LSB = 0）对齐。不支持偏离的字数据取指，因此在字节和字混合操作时或对现有 PIC 代码进行转换时必须小心。试图进行偏离的字读或写操作时，会产生地址错误陷阱。此时，偏离的读操作会完成，但偏离的写操作不被执行。随后会对陷阱进行处理，使系统能在执行错误地址之前检查机器状态。

图 3-4: 数据对齐

	15	MSByte	8	7	LSByte	0	
0001	字节 1			字节 0			0000
0003	字节 3			字节 2			0002
0005	字节 5			字节 4			0004
	字 0						0006
	字 1						0008
	长字 <15:0>						000A
	长字 <31:16>						000C

3.3 模寻址

模寻址（又称循环寻址）提供了一种使用硬件自动支持循环数据缓冲区的方法。目的是在执行许多 DSP 算法中典型的紧密循环代码时，不需要用软件执行数据地址边界检查。

除 W15 以外，任何 W 寄存器都可以被选择作为指向模缓冲区的指针。模寻址的硬件在所选 W 寄存器保存的地址上执行边界检查，并在需要时自动调整指向缓冲区边界的指针值。

dsPIC30F 模寻址可以在数据或程序空间操作（因为两种空间的数据指针机制的本质是相同的）。每个 X（也提供指向程序空间的指针）和 Y 数据空间中都有一个受支持的循环缓冲区。

模数据缓冲区长度可以是最大为 32K 字的任何长度。模缓冲逻辑电路支持使用字或字节作为数据单位的缓冲区。但是，模逻辑电路只在字地址边界执行地址边界检查，所以字节模缓冲区长度必须为偶数。此外，不能使用 Y AGU 实现字节大小的模缓冲，这是因为不支持通过 Y 存储器数据总线进行字节访问。

3.3.1 模起始和结束地址选择

有四个地址寄存器可用于指定模缓冲区的起始和结束地址：

- XMODSRT: X AGU 模起始地址寄存器
- XMODEND: X AGU 模结束地址寄存器
- YMODSRT: Y AGU 模起始地址寄存器
- YMODEND: Y AGU 模结束地址寄存器

模缓冲区的起始地址必须在偶字节地址边界上。XMODSRT 和 YMODSRT 寄存器的 LSB 固定设为“0”以确保模起始地址正确。模缓冲区的结束地址必须在奇字节地址边界上。XMODSRT 和 YMODSRT 寄存器的 LSB 固定设为“1”以确保模结束地址正确。

每个模缓冲区选择的起始和结束地址都有一定限制，这取决于实现的缓冲是递增还是递减。对于递增缓冲区，W 寄存器指针在缓冲区地址范围内递增。当递增缓冲区达到结束地址时，W 寄存器指针复位并指向缓冲区起始地址。对于递减缓冲区，W 寄存器指针在缓冲区地址范围内递减。当递减缓冲区达到起始地址时，W 寄存器指针复位并指向缓冲区结束地址。

注： 用户必须决定应用所需要的是递增还是递减模缓冲区。根据实现的模缓冲区是递增还是递减，会有一定的地址限制。

3.3.1.1 模起始地址

数据缓冲区起始地址是任意的，但是对于递增模缓冲区，该起始地址必须在一个 2 的零次方边界内。递减模缓冲区的模起始地址可以是任何值，并使用所选缓冲区的结束地址和缓冲区长度来计算。

例如，如果选择递增缓冲区的缓冲区长度为 50 字（100 字节），则缓冲区起始字节地址的最低有效位必须包含 7 个零。因此，有效起始地址可能为 0xNN00 和 0xNN80，其中“N”为任何十六进制值。

3.3.1.2 模结束地址

数据缓冲区结束地址可以为任意值，但对于递减缓冲区，该结束地址必须在一个含有“1”的边界内。递增模缓冲区的模结束地址可以是任何值，并使用所选缓冲区的起始地址和缓冲区长度来计算。

例如，如果选择缓冲大小（模值）为 50 字（100 字节），则递减模缓冲区的缓冲结束字节地址的最低有效位必须包含 7 个“1”。因此，有效结束地址可能为 0xNNFF 和 0xNN7F，其中“x”为任何十六进制值。

注： 如果所需的模缓冲区长度为 2 的偶数次幂，即可以选择满足递增和递减缓冲区要求的模起始和结束地址。

3.3.1.3 模地址计算

递增模缓冲区的结束地址必须根据所选的起始地址和缓冲区长度以字节为单位计算。可以用公式 3-1 来计算结束地址。

公式 3-1: 递增缓冲区的模结束地址

结束地址 = 起始地址 + 缓冲区长度 - 1

如公式 3-2 所示，递减模缓冲区的起始地址根据所选的结束地址和缓冲长度计算。

公式 3-2: 递减缓冲区的模起始地址

起始地址 = 结束地址 - 缓冲区长度 + 1

3.3.1.4 与模寻址 SFR 相关的数据依赖关系

在对模寻址控制寄存器 MODCON 进行写操作后，不应紧跟一个使用任何 W 寄存器进行的间接读操作。例 3-1 中的代码段将导致不可预料的结果。

注 1: 使用 POP 指令将栈顶 (TOS) 单元的内容弹出到 MODCON，也可以实现对 MODCON 的写入。在对 MODCON 执行写操作后，紧跟其后的命令不得为任何执行间接读操作的命令。

2: 用户应该注意某些指令隐含了执行间接读操作。它们是：POP、RETURN、RETFIE、RETLW 和 ULNK。

例 3-1: 错误的 MODCON 初始化

MOV #0x8FF4, w0 ;Initialize MODCON
MOV w0, MODCON
MOV [w1], w2 ;Incorrect EA generated here

要避免这种初始化问题，可在紧跟 MODCON 初始化之后的指令中，使用除间接读以外的任何寻址模式。如例 3-2 所示，一个简单的解决办法就是在初始化 MODCON 之后添加一个 NOP。

例 3-2: 正确的 MODCON 初始化

MOV #0x8FF4, w0 ;Initialize MODCON
MOV w0, MODCON
NOP ;See Note below
MOV [w1], w2 ;Correct EA generated here

在写模地址 SFR 后立即执行间接读操作还存在其他情形：

- XMODSRT
- XMODEND
- YMODSRT
- YMODEND

如果已经在 MODCON 中使能了模寻址，那么在写 X（或 Y）模地址 SFR 之后不应该紧跟一个间接读操作，而该间接读使用了指定作为从 X 数据空间（或 Y 数据空间）访问模缓冲区的 W 寄存器。例 3-3 中的代码段显示初始化与 X 数据空间相关的 SFR，将如何导致无法预期的结果。在 Y 空间进行初始化情况也大致相同。

例 3-3: 错误的模寻址设置

```
MOV #0x8FF4, w0 ;Modulo addressing enabled
MOV w0, MODCON ;in X-data space using w4
                  ;for buffer access
MOV #0x1200, w4 ;XMODSRT is initialized
MOV w4, XMODSRT
MOV #0x12FF, w0 ;XMODEND is initialized
MOV w0, XMODEND
MOV [w4++], w5 ;Incorrect EA generated
```

要避免这个问题，可以在初始化模数地址 SFR 之后插入一个 NOP 或执行任何操作，就是不要执行使用 W 寄存器访问该缓冲区的间接读操作。如例 3-4 中所示。另一个方法是在初始化模起始和结束地址 SFR 之后对 MODCON 进行模寻址。

例 3-4: 正确的模寻址设置

```
MOV #0x8FF4, w0 ;Modulo addressing enabled
MOV w0, MODCON ;in X-data space using w4
                  ;for buffer access
MOV #0x1200, w4 ;XMODSRT is initialized
MOV w4, XMODSRT
MOV #0x12FF, w0 ;XMODEND is initialized
MOV w0, XMODEND
NOP ;See Note below
MOV [w4++], w5 ;Correct EA generated here
```

注： 此外，也可以执行不使用 W 寄存器访问该缓冲区执行进行间接读操作的其他指令。

3.3.2 W 地址寄存器的选择

模寻址所使用的 X 地址空间指针 W 寄存器 (XWM) 存储在 MODCON<3:0> 中 (参见寄存器 3-1)。XMODSRT、XMODEND 和 XWM 寄存器这几种选择在 X RAGU 和 X WAGU 之间共享。当 XWM 被设置为除 15 之外的任何值且 XMODEN 位置 1 时 (MODCON<15>), X 数据空间的模寻址被使能。由于 W15 是专用的软件堆栈的指针, 所以不能作为模寻址指针使用。

模寻址所使用的 Y 地址空间指针 W 寄存器 (YWM) 存储在 MODCON<7:4> 中 (参见寄存器 3-2)。当 YWM 被设置为除 15 之外的任何值且 YMODEN 位置 1 时 (MODCON<14>), Y 数据空间的模寻址被使能。

注: 在写 MODCON 寄存器后, 不应执行使用 W 寄存器进行间接读操作的指令, 否则可能产生不可预料的结果。某些指令中隐含了间接读操作。它们是: POP、RETURN、RETFIE、RETLW 和 ULNK。

3.3.3 模寻址的适用性

模寻址可以被应用于与所选 W 寄存器相关的有效地址 (EA) 计算。用户应意识到, 地址边界检测功能会为递增缓冲区寻找等于或大于地址上边界的地址, 并为递减缓冲区寻找等于或小于地址下边界的地址, 这是非常重要的。因此, 地址变化可能会因此越过边界, 但仍然可以正确调整。请记住, 通过模硬件对 W 寄存器指针进行自动调整的过程是单向的。也就是说, 当用于递增缓冲区的 W 寄存器指针进行递减运算时, 模硬件可能不会正确调整 W 寄存器指针, 反之亦然。这一法则的例外情况是缓冲区长度为 2 的偶次幂, 且可以选择起始和结束地址以同时满足递增和递减的模缓冲区边界要求。

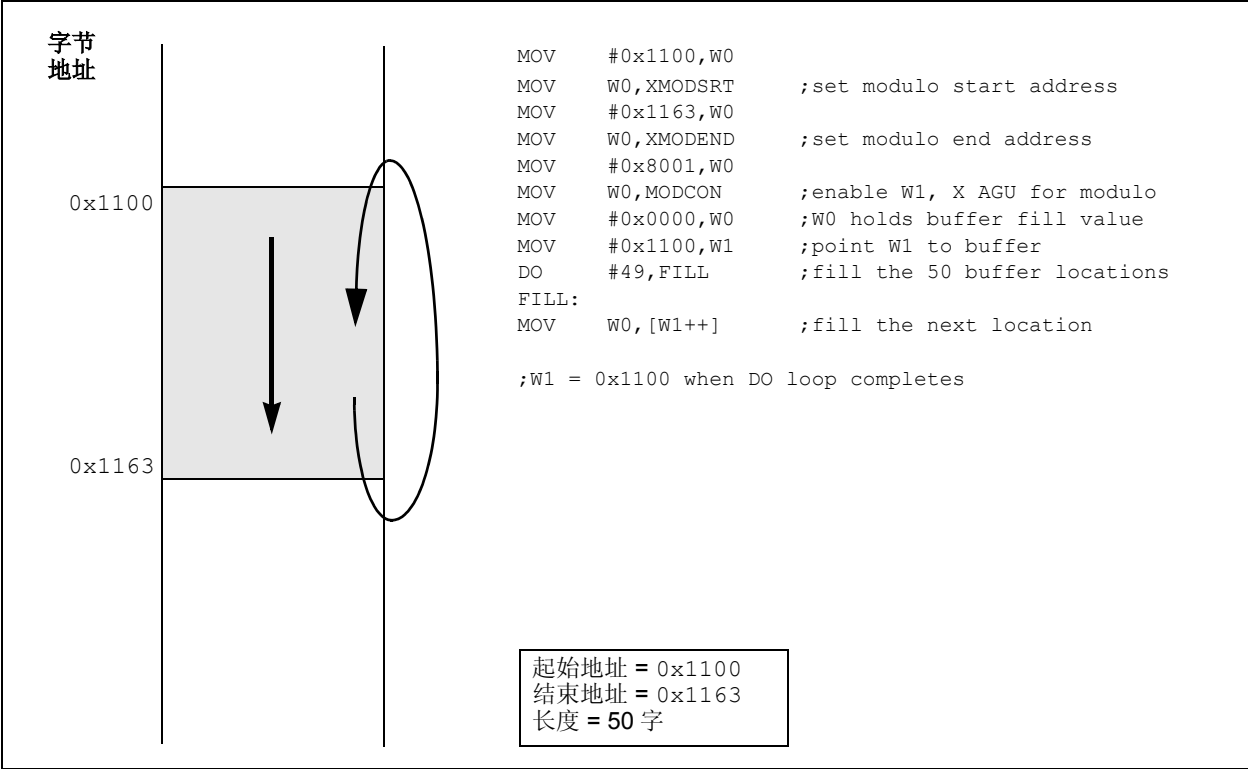
新的 EA 可能超过模缓冲区边界最多一个缓冲区长度, 但仍可以成功地纠正。记住何时使用寄存器索引 ([Wb + Wn]) 和立即数偏移 ([Wn + lit10]) 寻址模式是非常重要的。用户应切记寄存器索引和立即数偏移寻址模式不会改变保存在 W 寄存器中的值。只有带预修改和后修改的间接寻址模式 ([Wn++], [Wn--], [++Wn], [--Wn]) 才会改变 W 寄存器的地址值。

3.3.4 递增模缓存区的模寻址的初始化

以下步骤描述了递增循环缓冲区的设置步骤。不管是用 X AGU 还是 Y AGU，这些步骤都大体相同。

- 1. 确定缓冲区长度在 16 位数据字之内。将这个值乘以 2 得到以字节为单位的缓冲区长度。
- 2. 根据所需要的缓冲区长度选择位于二进制 0 边界的缓冲区起始地址。记住以字为单位的缓冲区长度必须乘以 2 才能得到字节地址范围。例如，长度为 100 字（200 字节）的缓冲区可以使用 0xXX00 作为起始地址。
- 3. 使用在步骤 1 中选择的缓冲区长度和在步骤 2 中选择的缓冲区起始地址，计算缓冲区结束地址。使用公式 3-1 计算缓冲区结束地址。
- 4. 将步骤 2 中选择的缓冲区起始地址装入 XMODSRT（YMODSRT）寄存器。
- 5. 将步骤 3 中计算的缓冲区结束地址装入 XMODEND（YMODEND）寄存器。
- 6. 写入 MODCON 寄存器中的 XWM<3:0>（YWM<3:0>）位，选择将用于访问循环缓冲区的 W 寄存器。
- 7. 将 MODCON 寄存器中的 XMODEN（YMODEN）位置 1，使能循环缓冲区。
- 8. 将指向缓冲区的地址装入所选的 W 寄存器。
- 9. 当执行带预/后递增的间接访问时，W 寄存器地址会在缓冲结束时自动调整（参见图 3-5）。

图 3-5: 递增缓冲模寻址操作示例

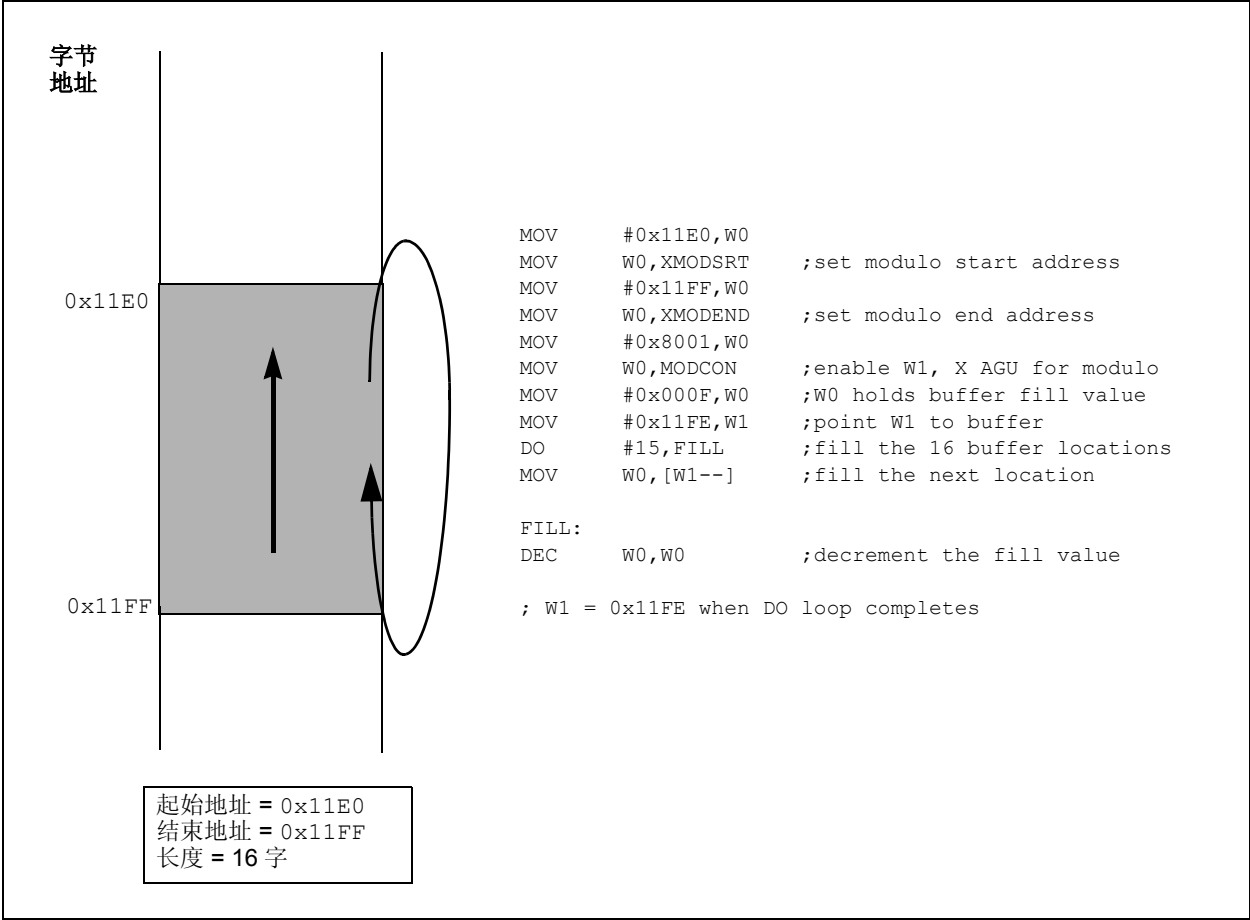


3.3.5 递减模缓冲区的模寻址的初始化

以下步骤描述了递减循环缓冲区的设置步骤。不管是用 X AGU 还是 Y AGU，这些步骤都大体相同。

- 1. 确定缓冲区长度在 16 位数据字之内。将这个值乘以 2 得到以字节为单位的缓冲区长度。
- 2. 根据所需要的缓冲区长度选择位于二进制 “1” 边界的缓冲区结束地址。记住以字为单位的缓冲区长度必须乘以 2 才能得到字节地址范围。例如，长度为 128 字（256 字节）的缓冲区可以使用 0xFFFF 作为结束地址。
- 3. 使用在步骤 1 中选择的缓冲区长度和第 2 步中选择的缓冲区结束地址计算缓冲区起始地址。使用公式 3-2 计算缓冲区起始地址。
- 4. 将步骤 3 中选择的缓冲区起始地址装入 XMODSRT（YMODSRT）寄存器。
- 5. 将步骤 2 中选择的缓冲区结束地址装入 XMODEND（YMODEND）寄存器。
- 6. 写入 MODCON 寄存器中的 XWM<3:0>（YWM<3:0>）位，选择将用于访问循环缓冲区的 W 寄存器。
- 7. 将 MODCON 寄存器中的 XMODEN（YMODEN）位置 1 使能循环缓冲区。
- 8. 将指向缓冲区的地址装入所选的 W 寄存器。
- 9. 当执行带预减/后减的间接访问时，W 寄存器地址会在缓冲结束时自动调整（参见图 3-6）。

图 3-6: 递减缓冲区模寻址操作示例



3.4 位反转寻址

3.4.1 位反转寻址简介

位反转寻址简化了基 2 FFT 算法进行的数据重新排序。只通过 X WAGU 支持位反转寻址。如图 3-7 所示，通过交换中心点周围的位单元中的二进制值可以有效地创建地址指针“镜像”，从而完成位反转寻址。表 3-1 所示为 4 位地址字段的位反转序列示例。

图 3-7: 位反转地址示例

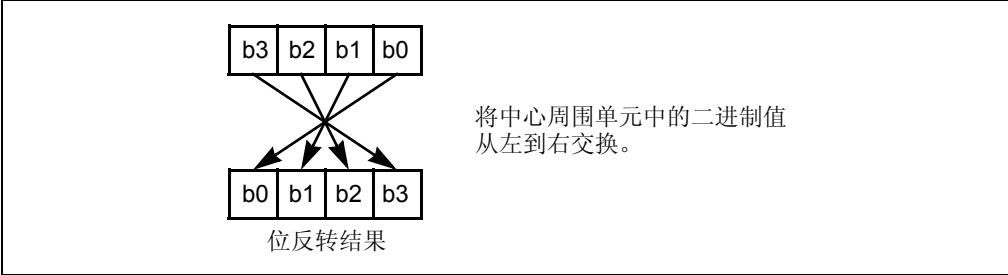


表 3-1: 位反转地址序列（16 个位）

正常地址					位反转 地址				
A3	A2	A1	A0	十进制	A3	A2	A1	A0	十进制
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	8
0	0	1	0	2	0	1	0	0	4
0	0	1	1	3	1	1	0	0	12
0	1	0	0	4	0	0	1	0	2
0	1	0	1	5	1	0	1	0	10
0	1	1	0	6	0	1	1	0	6
0	1	1	1	7	1	1	1	0	14
1	0	0	0	8	0	0	0	1	1
1	0	0	1	9	1	0	0	1	9
1	0	1	0	10	0	1	0	1	5
1	0	1	1	11	1	1	0	1	13
1	1	0	0	12	0	0	1	1	3
1	1	0	1	13	1	0	1	1	11
1	1	1	0	14	0	1	1	1	7
1	1	1	1	15	1	1	1	1	15

3.4.2 位反转寻址操作

只有 XWAGU 支持位反转寻址，该寻址由 MODCON 和 XBREV 特殊功能寄存器控制位反转寻址的使能方式如下：

- 1. 使用 BWM 控制位（MODCON<11:8>）将位反转寻址分配给一个 W 寄存器。
- 2. 通过置位 BREN 控制位（XBREV<15>）使能位反转寻址。
- 3. 通过 XB 控制位（XBREV<14:0>）置位 XAGU 位反转修改器。

当位反转寻址被使能时，只有在使用带预增或后增间接寻址模式的寄存器时（[Wn++], [++Wn]），位反转寻址硬件才产生位反转地址。此外，生成的位反转地址只供字模式指令使用。所有其他寻址模式指令或字节模式指令均不能使用位反转地址（将生成正常地址）。

注： 在写 MODCON 寄存器后不能执行使用 W 寄存器进行间接读操作的指令。这会产生不可预料的结果。某些指令中隐含了间接读操作。它们是 POP、RETURN、RETFIE、RETLW 和 ULNK。

3.4.2.1 模寻址和位反转寻址

可以使用同一个 W 寄存器同时使能模寻址和位反转寻址，但位反转寻址操作使能时，总是具有较高的数据写操作优先级。例如，以下的设置条件将同一 W 寄存器分配给模寻址和位反转寻址：

- 使能 X 模寻址（XMODEN = 1）
- 使能位反转寻址（BREN = 1）
- 将 W1 分配给模寻址（XWM<3:0> = 0001）
- 将 W1 分配给位反转寻址（XWM<3:0> = 0001）

对于用 W1 作为指针的数据读操作，将发生模地址边界检查。对于用 W1 作为目标指针的数据写操作，位反转硬件将为数据重新排序纠正 W1。

3.4.2.2 与 XBREV 相关的数据相依性

如果已经通过置位 BREN（XBREV<15>）位使能了位反转寻址，那么在写入了 XBREV 寄存器之后，不能紧跟一个使用 W 寄存器的间接读操作，而该 W 寄存器被指定为位反转地址指针。

3.4.3 位反转修改量

装入 XBREV 寄存器的值是一个常数，它间接定义了位反转数据缓冲区的大小。表 3-2 是常用位反转缓冲区使用的 XB 修改量的汇总。

表 3-2: 位反转地址修改量

缓冲区大小（字）	XB 位反转寻址修改量
32768	0x4000
16384	0x2000
8192	0x1000
4096	0x0800
2048	0x0400
1024	0x0200
512	0x0100
256	0x0080
128	0x0040
64	0x0020
32	0x0010
16	0x0008
8	0x0004
4	0x0002
2	0x0001

注： 只有上述位反转修改量会产生有效的位反转地址序列。

位反转硬件通过执行 W 内容和 XB 修改常数的“逆进位”来修改 W 寄存器地址。逆进位加法是从左到右将位相加而不是从右到左。如果在一个位的位置发生了进位，进位位被加到其右侧的下一个位。例 3-5 演示了将 0x0008 作为 XB 修改量的逆进位加法和 W 寄存器的结果值。注意 XB 修改器将一个位位置向左移产生字地址值。

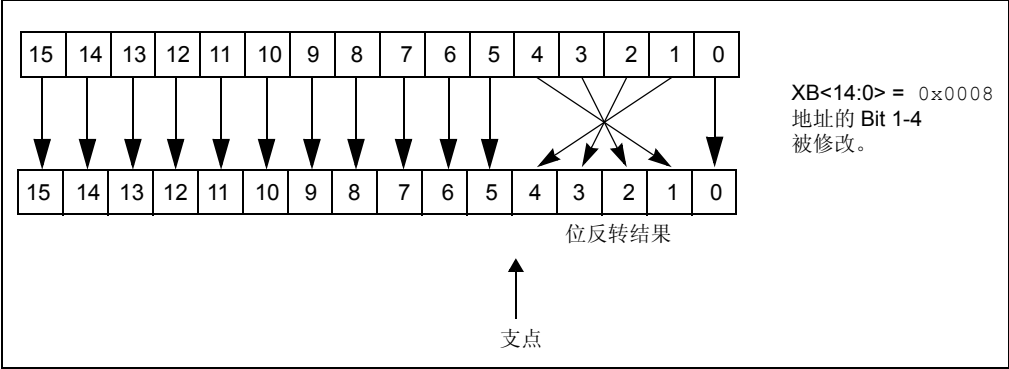
例 3-5: XB 地址计算

0000 0000 0000 0000	Wn 指向字 0
+1 0000	Wn = Wn + XB
0000 0000 0001 0000	Wn 指向字 8
+1 0000	Wn = Wn + XB
0000 0000 0000 1000	Wn 指向字 4
+1 0000	Wn = Wn + XB
0000 0000 0001 1000	Wn 指向字 12
+1 0000	Wn = Wn + XB
0000 0000 0000 0100	Wn 指向字 2
+1 0000	Wn = Wn + XB
0000 0000 0001 0100	Wn 指向字 10

当 XB<14:0> = 0x0008 时，位反转缓冲区大小为 16 字。W 寄存器的 Bit 1-4 将处于位反转地址修正状态，但 Bit 5-15（支点之外）将不会被位反转硬件修改。由于位反转硬件只在字地址上工作，所以不会更改 Bit 0。

XB 修改器控制位反转地址修改的“支点（Pivot Point）”。支点以外的位不会进行位反转地址修正。

图 3-8: 16 字缓冲区的位反转地址修改



3.4.4 位反转寻址代码示例

以下代码示例读取了 16 个数据字序列并将数据按照位反转顺序写入新单元。W0 是读地址指针，而 W1 是写地址指针，用于位反转修改。

```
; Set XB for 16-word buffer, enable bit reverse addressing
MOV    #0x8008,W0
MOV    W0,XBREV
; Setup MODCON to use W1 for bit reverse addressing
MOV    #0x01FF,W0
MOV    W0,MODCON
; W0 points to input data buffer
MOV    #Input_Buf,W0
; W1 points to bit reversed data
MOV    #Bit_Rev_Buf,W1
; Re-order the data from Input_Buf into Bit_Rev_Buf
REPEAT #15
MOV    [W0++],[W1++]
```

3.5 控制寄存器说明

以下寄存器用于控制模和位反转寻址：

- MODCON: 模寻址控制寄存器
- XMODSRT: X AGU 模起始地址寄存器
- XMODEND: X AGU 模结束地址寄存器
- YMODSRT: Y AGU 模起始地址寄存器
- YMODEND: Y AGU 模结束地址寄存器
- XBREV: X AGU 位反转寻址控制寄存器

下面会为您提供各寄存器的详细说明。

寄存器 3-1: MODCON: 模寻址和位反转寻址控制寄存器

高字节:							
R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
XMODEN	YMODEN	—	—	BWM<3:0>			
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YWM<3:0>				XWM<3:0>			
bit 7				bit 0			

- bit 15 **XMODEN:** X RAGU 和 X WAGU 模寻址使能位
1 = X AGU 模寻址使能
0 = X AGU 模寻址禁止
- bit 14 **YMODEN:** Y AGU 模寻址使能位
1 = Y AGU 模寻址使能
0 = Y AGU 模寻址禁止
- bit 13-12 未用: 读作 0。
- bit 11-8 **BWM<3:0>:** 用于位反转寻址的 X WAGU 寄存器选择位
1111 = 禁止位反转寻址
1110 = 选择 W14 用于位反转寻址
1101 = 选择 W13 用于位反转寻址
.
.
0000 = 选择 W0 用于位反转寻址
- bit 7-4 **YWM<3:0>:** 用于模寻址的 Y AGU W 寄存器选择位
1111 = 禁止模寻址
1010 = 选择 W10 用于模寻址
1011 = 选择 W11 用于模寻址

注: YWM<3:0> 控制位的其他设置均被保留, 不要使用。

- bit 3-0 **XWM<3:0>:** 用于模寻址的 X RAGU 和 X WAGU W 寄存器选择位
1111 = 禁止模寻址
1110 = 选择 W14 用于模寻址
.
.
0000 = 选择 W0 用于模寻址

注: 在写 MODCON 寄存器后不能执行使用 W 寄存器进行间接读操作的指令。否则可能产生不可预料的结果。某些指令中隐含了间接读操作。它们是: POP、RETURN、RETFIE、RETLW 和 ULNK。

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知位

寄存器 3-2: XMODSRT: X AGU 模寻址起始寄存器

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XS<15:8>							
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
XS<7:1>							0
bit 7				bit 0			

bit 15-1 XS<15:1>: X RAGU 和 X WAGU 模寻址起始地址位

bit 0 未用: 读作 0

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知位

寄存器 3-3: XMODEND: X AGU 模寻址结束寄存器

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XE<15:8>							
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1
XE<7:1>							1
bit 7				bit 0			

bit 15-1 XE<15:1>: X RAGU 和 X WAGU 模寻址结束地址位

bit 0 未用: 读作 1

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n=POR 时的值	1= 置位	0= 清零	x= 未知位

寄存器 3-4: YMODSRT: Y AGU 模寻址起始寄存器

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YS<15:8>							
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
YS<7:1>							0
bit 7				bit 0			

bit 15-1 **YS<15:1>**: Y AGU 模寻址起始地址位
bit 0 未用: 读作 0

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知位

寄存器 3-5: YMODEND: Y AGU 模寻址结束寄存器

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YE<15:8>							
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1
YE<7:1>							1
bit 7				bit 0			

bit 15-1 **YE<15:1>**: Y AGU 模寻址结束地址位
bit 0 未用: 读作 1

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知位

寄存器 3-6: XBREV: X 写 AGU 位反转寻址控制寄存器

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BREN	XB<14:8>						
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XB<7:0>							
bit 7							bit 0

bit 15 **BREN:** 位反转寻址 (X AGU) 使能位
1 = 位反转寻址使能
0 = 位反转寻址禁止

bit 14-0 **XB<14:0>:** X AGU 位反转修改量位
0x4000 = 32768 字缓冲
0x2000 = 16384 字缓冲
0x1000 = 8192 字缓冲
0x0800 = 4096 字缓冲
0x0400 = 2048 字缓冲
0x0200 = 1024 字缓冲
0x0100 = 512 字缓冲
0x0080 = 256 字缓冲
0x0040 = 128 字缓冲
0x0020 = 64 字缓冲
0x0010 = 32 字缓冲
0x0008 = 16 字缓冲
0x0004 = 8 字缓冲
0x0002 = 4 字缓冲
0x0001 = 2 字缓冲

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知位

3.6 相关应用笔记

本节列出了与手册的本章节相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。目前与数据存储器模块相关的应用笔记有：

标题	应用笔记编号
目前没有相关的应用笔记。	

注：如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

3.7 版本历史

版本 A

这是本文档的初始发行版。

版本 B

此版本添加了 dsPIC30F 数据存储器模块的更多技术内容。

版本 C

此版本加入了本文档更新时已知的所有勘误表。

第 4 章 程序存储器

目录

本章包括下列主题：

4.1	程序存储器地址映射	4-2
4.2	程序计数器	4-4
4.3	从程序存储器存取数据	4-4
4.4	来自数据空间的程序空间可视性	4-8
4.5	写程序存储器	4-10
4.6	PSV 代码示例	4-11
4.7	相关应用笔记	4-12
4.8	版本历史	4-13

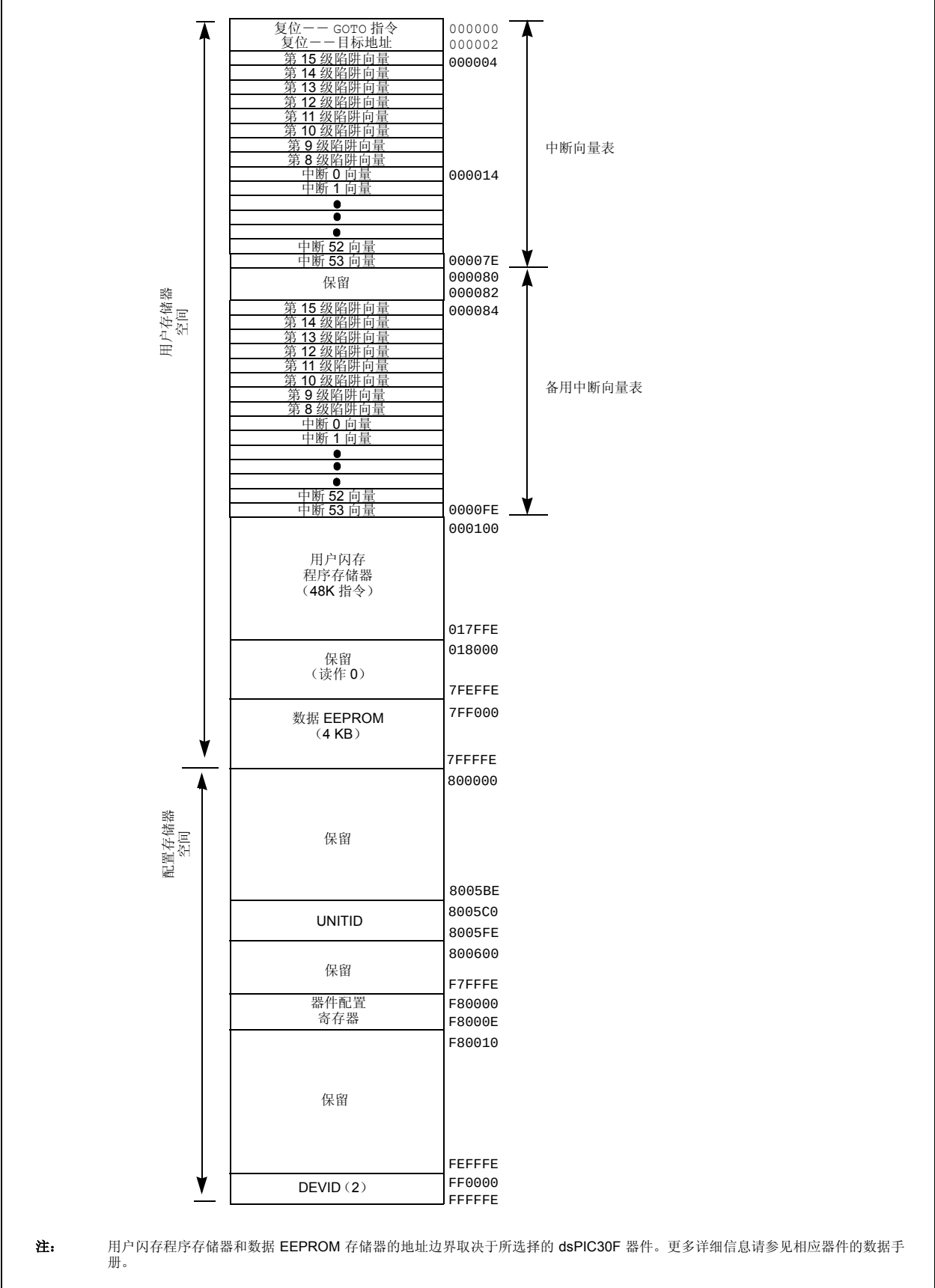
4.1 程序存储器地址映射

dsPIC30F器件有 $4\text{M} \times 24$ 位程序存储器地址空间，如图 4-1所示。访问程序空间有三种方法可用：

1. 通过 23 位 PC。
2. 通过读表（TBLRD）和写表（TBLWT）指令。
3. 通过把程序存储器的 32 KB 段映射到数据存储器地址空间。

程序存储器映射空间被划分为用户程序空间 and 用户配置空间。用户程序空间包含复位向量、中断向量表、程序存储器和数据 EEPROM 存储器。用户配置空间包含用于设置器件选项的非易失性配置位和器件 ID 单元。

图 4-1: 程序空间存储器映射示例

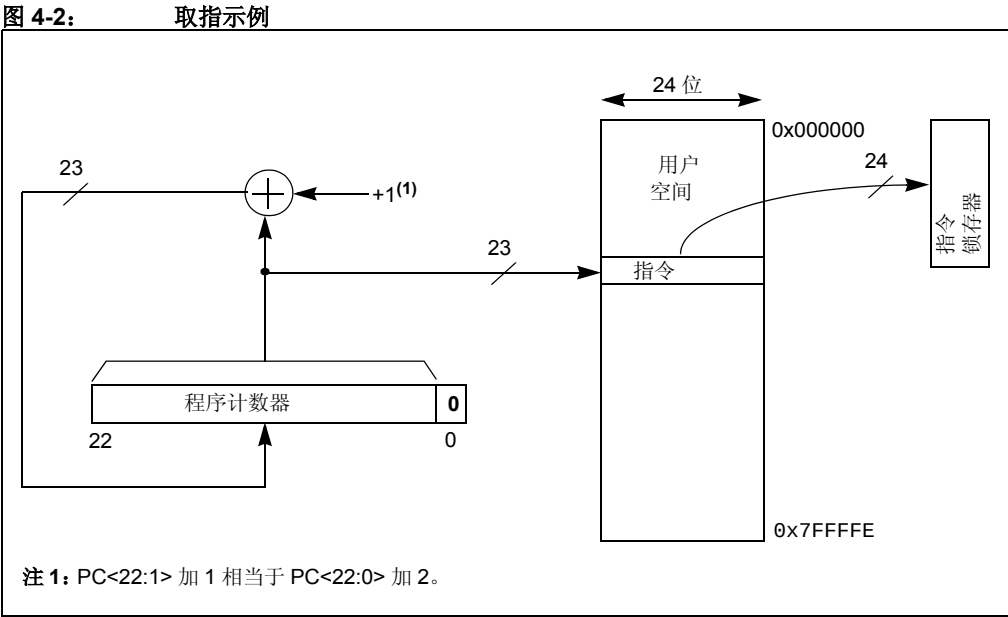


4.2 程序计数器

PC 以 2 为增量且 LSb 置为 “0” 以使之与数据空间寻址相兼容。用 PC<22:1> 在 4M 程序存储器空间中对连续指令字寻址。每个指令字为 24 位宽。

程序存储器地址的 LSb (PC<0>) 保留为字节选择位，用于从使用程序空间可视性或表指令的数据空间访问程序存储器。对于通过 PC 取指的情况，不需要该字节选择位。所以，此时 PC<0> 总是置为 “0”。

取指示例如图 4-2 所示。注意 PC<22:1> 加 1 相当于 PC<22:0> 加 2。



4.3 从程序存储器存取数据

可以使用两种方法在程序存储器和数据存储器空间之间传送数据，即：通过特殊的表指令或通过把 32 KB 程序空间页重新映射到数据空间的上半部分。TBLRDL 和 TBLWTL 指令提供了读或写程序空间内任何地址的 LSWord 的直接方法（无需通过数据空间），很适合某些应用。TBLRDH 和 TBLWTH 指令是可以把一个程序字的高 8 位作为数据存取的惟一方法。

4.3.1 表指令综述

我们提供了一组表指令用于将字节或字大小的数据在程序空间和数据空间之间传送。读表指令用于把数据从程序存储器空间读入数据存储器空间。写表指令可以把数据从数据存储器空间写入程序存储器空间。

注： 使用表指令的详细代码示例请参见第 5 章 “闪存和 EEPROM 编程”。

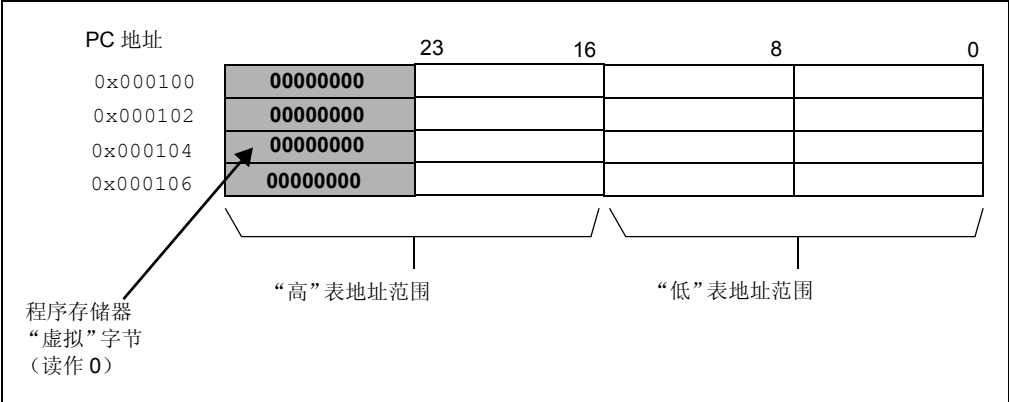
表指令共有以下四条：

- TBLRDL：读表的低位字
- TBLWTL：写表的低位字
- TBLRDH：读表的高位字
- TBLWTH：写表的高位字

对于表指令，程序存储器可以视作并排放置的两个 16 位字宽的地址空间，每个地址空间都有相同的地址范围，见图 4-3。这样使程序空间可作为由字节或对齐的字寻址的 16 位宽，64 KB 的页被访问（即，与数据空间相同）。

TBLRDL 和 TBLWTL 访问程序存储器的 LS 数据字，而 TBLRDH 和 TBLWTH 访问高位字。由于程序存储器只有 24 位宽，所以后一个字空间的高字节不存在（虽然它是可寻址的）。因此称之为“虚拟”字节。

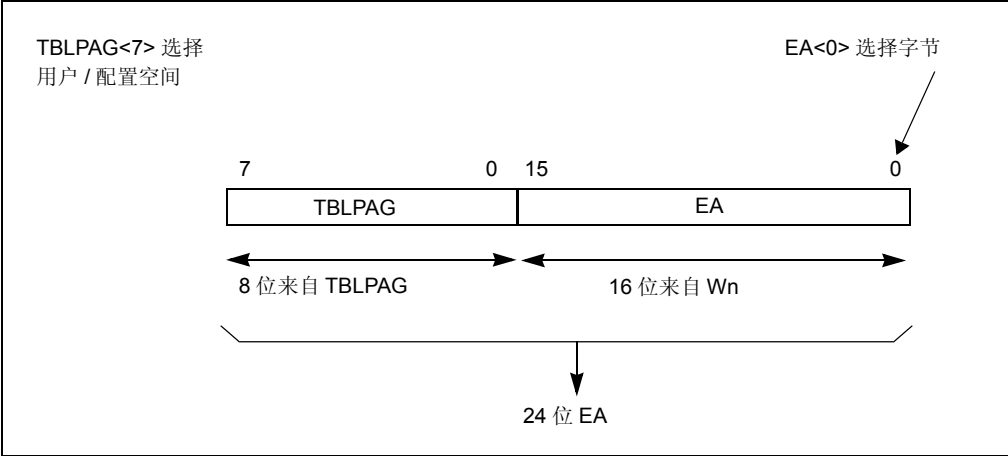
图 4-3： 表操作的高和低地址区域



4.3.2 表地址的生成

对于所有表指令，W 寄存器地址值与 8 位数据表页寄存器 TBLPAG 相连，形成一个 23 位有效的程序空间地址加上一个字节选择位，如图 4-4 所示。由于 W 寄存器提供了 15 位的程序空间地址，所以程序存储器中的数据表页的大小为 32K 字。

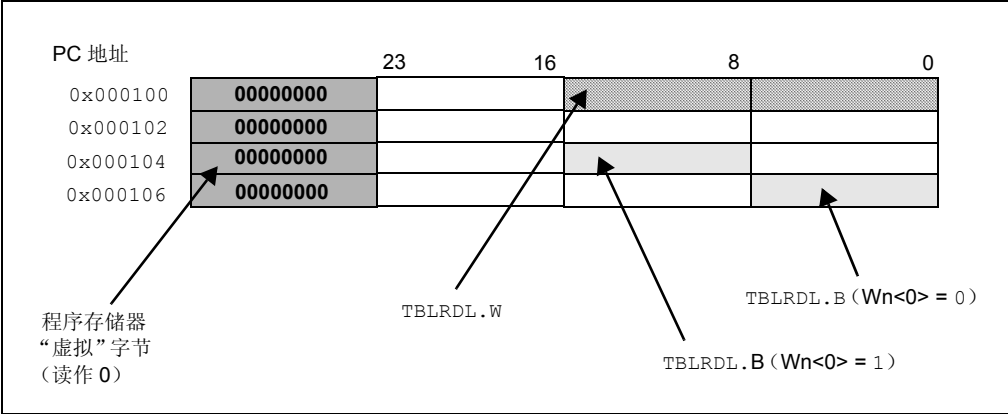
图 4-4： 表操作的地址生成



4.3.3 程序存储器低位字访问

TBLRD L 和 TBLWT L 指令用于访问程序存储器数据的低 16 位。对于以字宽为单位的表访问，W 寄存器地址的 LSb 被忽略。对于以字节宽为单位的访问，W 寄存器地址的 LSb 决定读哪一个字节。图 4-5 演示了用 TBLRD L 和 TBLWT L 指令访问的程序存储器数据区域。

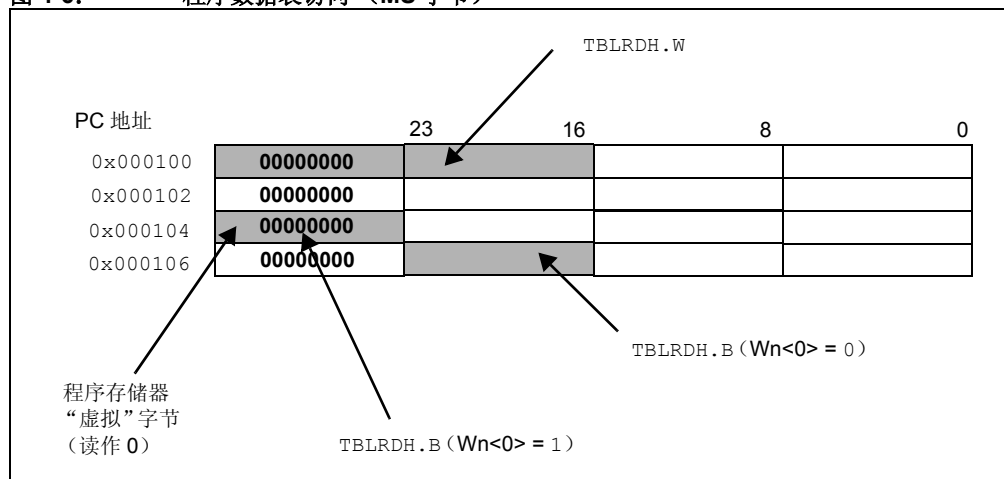
图 4-5： 程序数据表访问（LSWord）



4.3.4 程序存储器高位字访问

TBLRDH 和 TBLWTH 指令用于访问程序存储器数据的高 8 位。为了正交性，这些指令也支持字或字节访问模式，但是程序存储器数据的高字节将总是返回“0”，如图 4-6 所示。

图 4-6: 程序数据表访问 (MS 字节)



4.3.5 程序存储器中的数据存储器

假设对于大多数应用，高字节（P<23:16>）将不会用于存储数据，使程序存储器有 16 位宽供数据存储。建议将程序数据的最高字节编程为 NOP 或作为非法操作码值，以防止器件意外执行所存储的数据。TBLRDH 和 TBLWTH 指令主要用于阵列编程 / 校验和那些要求压缩数据存储的应用。

4.4 来自数据空间的程序空间可视性

可选择将 dsPIC30F 数据存储器地址空间的高 32 KB 映射到任何 16K 字程序空间页。这种操作模式被称为程序空间可视性 (PSV)，它提供对存储在 X 数据空间的常数数据的透明访问，而无需使用特殊指令 (即，TBLRD 和 TBLWT 指令)。

4.4.1 PSV 的配置

通过将 PSV 位 (CORCON<2>) 置 1 使能程序空间可视性。有关对 CORCON 寄存器的描述参见第 2 章 “CPU”。

当 PSV 使能时，在数据存储器映射空间上半部分的每个数据空间地址将直接映射到一个程序地址 (参见图 4-7)。PSV 视窗允许访问该 24 位程序字的低 16 位。程序存储器数据的高 8 位应该编程，以强制对其的访问为非法指令或 NOP，从而维持器件的鲁棒性 (Robutness)。请注意表指令是读每个程序存储器字的高 8 位的惟一方法。

图 4-8 显示了如何生成 PSV 地址。PSV 地址的 15 个 LSb 由包含有效地址的 W 寄存器提供。W 寄存器的 MSb 不用于形成该地址，而是用于指定是从程序空间执行 PSV 访问还是从数据存储器空间执行正常的访问。如果使用的 W 寄存器有效地址大于或等于 0x8000，使能 PSV 时，数据访问会从程序存储器空间进行。当 W 寄存器的有效地址小于 0x8000 时，所有访问将从数据存储器空间进行。

余下的地址位由 PSVPAG 寄存器 (PSVPAG<7:0>) 提供，如图 4-8 所示。PSVPAG 位与 W 寄存器中保存有效地址的 15 个 LSb 相连形成一个 23 位的程序存储器地址。PSV 只能用来访问程序存储器空间中的值。必须用表指令来访问用户配置空间中的值。

W 寄存器值的 LSb 用作字节选择位，该位允许使用 PSV 的指令以字节或字模式运行。

4.4.2 X 和 Y 数据空间的 PSV 映射

对于 dsPIC30F 系列中大多数器件，Y 数据空间在数据空间上半部分以外，所以 PSV 区域将映射到 X 数据空间。X 和 Y 映射将影响在算法中如何使用 PSV。

例如，可以使用 PSV 映射空间存储有限冲激响应 (Finite Impulse Response, FIR) 滤波器算法的系数数据。FIR 滤波器将包含原先滤波器输入数据的数据缓冲器中各个值与包含常数滤波器系数的数据缓冲器中的值相乘。通过在 REPEAT 循环内使用 MAC 指令执行 FIR 算法。MAC 指令的每次迭代预取一个原先输入的值和一个将在下一次迭代中相乘的系数值。预取的值中的一个必须位于 X 数据存储器空间，而另一个必须位于 Y 数据存储器空间。

为了满足 FIR 滤波器算法的 PSV 映射要求，用户必须将原先输入的数据放在 Y 存储器空间中，而将滤波器系数放在 X 存储器空间中。

图 4-7：程序空间可视性操作

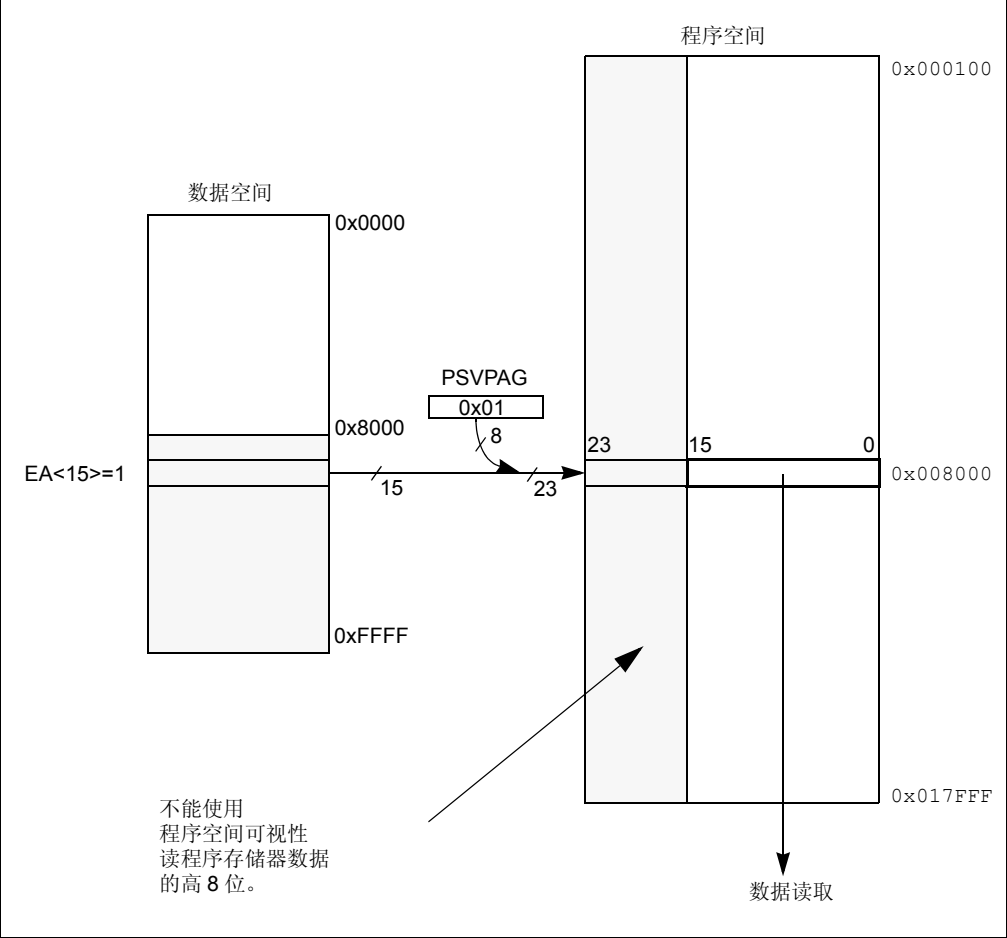
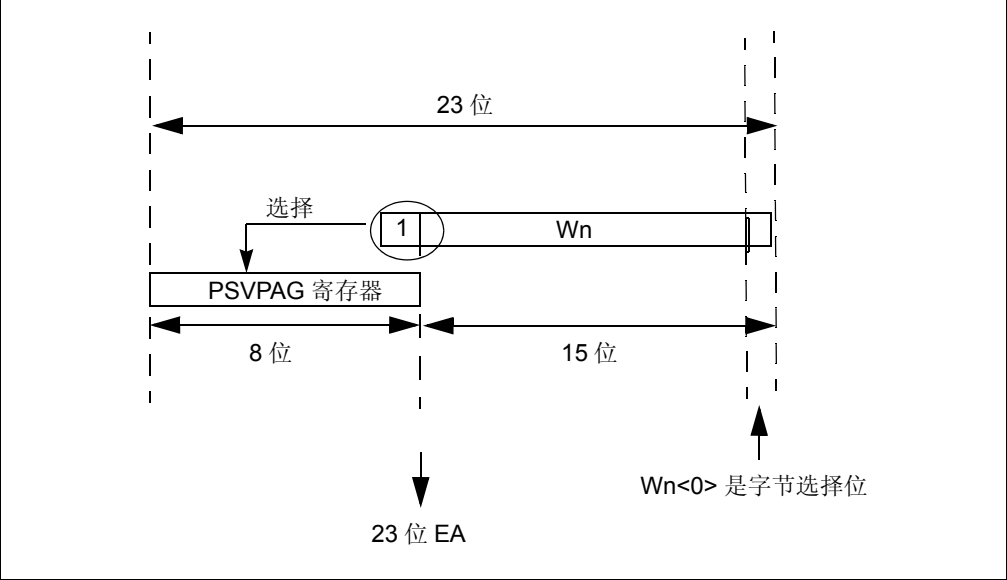


图 4-8：程序空间可视性地址的生成



4.4.3 PSV 时序

除了下列指令只需要一个额外周期即可执行完毕以外，其他使用 PSV 的指令需要两个额外的指令周期才能执行完毕：

- 带数据预取操作数的 MAC 类指令
- 所有 MOV 指令，包括 MOV.D 指令

额外的指令周期是用于取程序存储器总线上的 PSV 数据。

4.4.3.1 在 REPEAT 循环中使用 PSV

在一个 REPEAT 循环内使用 PSV 的指令消除了访问程序存储器内数据所需的额外指令周期，从而不会产生执行时间上的开销。然而，以下 REPEAT 循环迭代将产生两个指令周期的开销才能执行完毕：

- 第一次迭代
- 最后一次迭代
- 由于中断，指令在退出该循环前的指令执行
- 中断处理后，重新进入该循环的指令执行

4.4.3.2 PSV 和指令停顿

欲获取更多有关使用 PSV 的指令停顿的信息，请参见第 2 章“CPU”。

4.5 写程序存储器

dsPIC30F 系列器件包含内部程序闪存存储器用于执行用户代码。用户可用两种方法对该存储器编程：

1. 运行时自编程（Run-Time Self Programming, RTSP）
2. 在线串行编程（In-Circuit Serial Programming™, ICSP™）

通过执行 TBLWT 指令实现 RTSP。通过使用 SPI 接口和集成的引导加载程序软件实现 ICSP。更多有关 RTSP 的详细信息，请参见第 5 章“闪存和 EEPROM 编程”。ICSP 规范可从 Microchip Technology 网站下载（www.microchip.com）。

4.6 PSV 代码示例

4.6.1 C 语言 PSV 代码示例

```
// PSV code example in C
// When defined as below the const string uses the PSV feature of dsPIC

const unsigned char hello[] = {"Hello World:\r\n"};
unsigned char *TXPtr;          // Transmit pointer

int main(void)
{
    // Initialize the UART1
    U1MODE = 0x8000;
    U1STA = 0x0000;
    U1BRG = ((FCY/16)/BAUD) - 1;    // set baud rate = BAUD
    TXPtr = &hello[0];              // point to first char in string
    U1STABits.UTXEN = 1;            // Initiate transmission
    while (1)
    {
        while (*TXPtr)              // while valid char in string ...
            if (!U1STABits.UTXBF)    // and buffer not full ...
                U1TXREG = *TXPtr++;  // transmit string via UART

        DelayNmSec(500);             // delay for 500 mS
        TXPtr = &hello[0];          // re-initialize pointer to first char
    }

    // end main
}
```

4.6.2 汇编语言 PSV 代码示例

```
.equ CORCONL, CORCON
.section .const, "r"
hello:
    .ascii "Hello World:\n\r\0"

.global __reset ;Declare the label for the start of code

.text                ;Start of Code section

__reset:

    clr    U1STA
    mov    #0x8000,W0    ; enable UART module
    mov    W0,U1MODE
    mov    #BR,W0        ; set baudrate using formula value
    mov    W0, U1BRG      ; /
    bset   U1STA,#UTXEN   ; initiate transmission
Again:
    rcall   Delay500mSec  ; delay for 500 mS
    mov     #psvpage(hello),w0
    mov     w0, PSVPAG
    bset.b  CORCONL,#PSV
    mov     #psvoffset(hello),w0
TxSend:
    mov.b   [w0++], w1    ; get char in string
    cp      w1,#0         ; if Null
    bra     Z,Again       ; then re-initialize
BufferTest:
    btsc    U1STA,#UTXBF   ; see if buffer full
    bra     BufferTest      ; wait till empty
    mov     w1,U1TXREG      ; load value in TX buffer
    bra     TxSend         ; repeat for next char.
```

4.7 相关应用笔记

本节列出了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是专门为 dsPIC30F 产品系列编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。目前与程序存储器模块相关的应用笔记有：

标题

应用笔记编号

目前没有相关的应用笔记。

注： 如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

4.8 版本历史

版本 A

这是本文档的初始发行版。

版本 B

本章未包含任何技术内容或编辑排版上的修订，然而我们谨更新此节以反映本手册已更新到版本 B。

版本 C

此版本加入了本文档更新时已知的所有勘误表。

版本 D

增加了第 4.6 节 “PSV 代码示例”。

注:

第 5 章 闪存和 EEPROM 编程

目录

本章包括下列内容：

5.1	简介	5-2
5.2	表指令操作	5-2
5.3	控制寄存器	5-5
5.4	运行时自编程 (RTSP)	5-10
5.5	数据 EEPROM 编程	5-15
5.6	设计技巧	5-21
5.7	相关应用笔记	5-22
5.8	版本历史	5-23

5.1 简介

本章介绍了闪存程序存储器和数据 EEPROM 存储器的编程技术。dsPIC30F 系列器件内部包含了用于执行用户代码的程序闪存存储器。用户可以使用以下两种方法对此存储器编程：

1. 运行时自编程 (RTSP)
2. 在线串行编程 (ICSP™)

RTSP 是由用户软件执行的。ICSP 是通过与器件的串行数据连接进行的，并且编程速度比 RTSP 快得多。本章将介绍 RTSP 技术。ICSP 协议的描述请参见 dsPIC30F 编程规范文档，该文档可从 Microchip 网站下载。

数据 EEPROM 能够映射到程序存储器空间。EEPROM 由一个 16 位宽的存储器构成，存储器大小最大可达 2K 字 (4 KB)。EEPROM 的存储容量取决于具体器件。如需更多信息，请参见器件数据手册。

用于数据 EEPROM 的编程技术与用于闪存程序存储器的 RTSP 编程技术类似。闪存和数据 EEPROM 的编程操作的主要区别在于在每个编程 / 擦除周期所能够编程或擦除的数据量不同。

5.2 表指令操作

表指令操作提供了一种在 dsPIC30F 器件的程序存储器空间和数据存储器空间之间传输数据的方法。由于在闪存程序存储器和数据 EEPROM 的编程中需要使用表指令，因此在这里概述了表指令。有四种基本表指令：

- TBLRDL：读表低位
- TBLRDH：读表高位
- TBLWTL：写表低位
- TBLWTH：写表高位

TBLRDL 和 TBLWTL 指令用于读写程序存储器空间的 <15:0> 位。TBLRDL 和 TBLWTL 能以字模式或字节模式访问程序存储器。

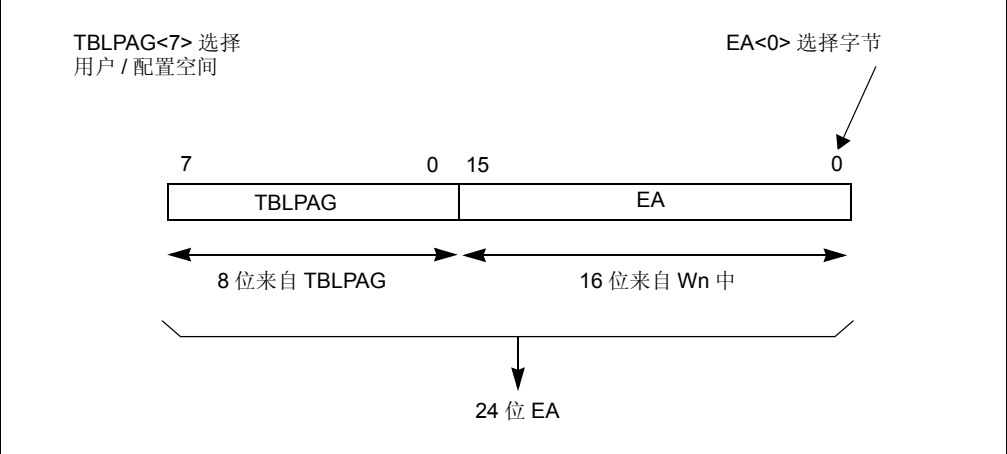
TBLRDH 和 TBLWTH 指令用于读写程序存储器空间的 <23:16> 位。TBLRDH 和 TBLWTH 能以字模式或字节模式访问程序存储器。因为程序存储器只有 24 位宽，所以 TBLRDH 和 TBLWTH 指令能寻址程序存储器中并不存在的高位字节地址。这个字节称为“虚拟字节”。读取虚拟字节总是会返回 0x00，而对其写则不起作用。

请始终记住可以将 24 位程序存储器当作两个并列的、共享同一个地址范围的 16 位空间。因此，TBLRDL 和 TBLWTL 指令可访问“低”程序存储器空间 (PM<15:0>)。TBLRDH 和 TBLWTH 指令访问“高”程序存储器空间 (PM<31:16>)。读取或写入 PM<31:24> 可以访问虚拟 (未用的) 字节。当在字节模式下使用任何的表指令时，表地址的 LSb 将被用作字节选择位。LSb 决定将访问高位或低位程序存储器空间的哪个字节。

图 5-1 显示了使用表指令寻址程序存储器的方法。24 位程序存储器地址由 TBLPAG 寄存器的 <7:0> 位以及表指令指定的 W 寄存器中的有效地址 (EA) 组成。图 5-1 中给出了 24 位程序计数器以供参考。EA 的高 23 位用于选择程序存储器单元。对于字节模式的表指令，W 寄存器 EA 的 LSb 用于选择 16 位程序存储器字中要寻址的字节。“1”选择 <15:8> 位，“0”选择 <7:0> 位。W 寄存器 EA 的 LSb 在字模式下的表指令中会被忽略。

除了指定程序存储器地址外，表指令还指定作为要写入的程序存储器数据源或要读取的程序存储器目标的 W 寄存器 (或指向存储器单元的 W 寄存器指针)。对于字节模式的写表操作，源工作寄存器的 <15:8> 位会被忽略。

图 5-1： 表指令的寻址



5.2.1 使用读表指令

读表需要两个步骤。首先，使用 TBLPAG 寄存器和一个 W 寄存器建立一个地址指针。然后就可以读取地址单元的程序存储器内容。

5.2.1.1 字模式读取

以下代码示例说明了如何使用字模式下的表指令读程序存储器中的一个字。

```
; Setup the address pointer to program space
MOV    #tblpage(PROG_ADDR),W0    ; get table page value
MOV    W0,TBLPAG                 ; load TBLPAG register
MOV    #tbloffset(PROG_ADDR),W0  ; load address LS word
; Read the program memory location
TBLRDH [W0],W3                   ; Read high byte to W3
TBLRDL [W0],W4                   ; Read low word to W4
```

5.2.1.2 字节模式读

```
; Setup the address pointer to program space
MOV    #tblpage(PROG_ADDR),W0    ; get table page value
MOV    W0,TBLPAG                 ; load TBLPAG register
MOV    #tbloffset(PROG_ADDR),W0  ; load address LS word
; Read the program memory location
TBLRDH.B [W0],W3                 ; Read high byte to W3
TBLRDL.B [W0++],W4               ; Read low byte to W4
TBLRDL.B [W0++],W5               ; Read middle byte to W5
```

在上面的代码示例中，在读取低字节时的后加操作符会导致工作寄存器中的地址加 1。这会将 EA<0> 置位为“1”，以访问第三个写指令中的中间字节。最后的后加操作将 W0 设置回一个偶数地址，指向下一个程序存储器单元。

注： Microchip 的 dsPIC30F 汇编器提供了 tblpage() 和 tbloffset() 伪指令。这些伪指令会为表指令从程序存储器地址值中选择适当的 TBLPAG 和 W 寄存器值。如需详细信息，请参见 MPLAB ASM30, MPLAB LINK30 and Utilities User's Guide (DS51317)。

5.2.2 使用写表指令

写表指令的影响取决于器件程序存储器地址空间使用的存储技术类型。程序存储器地址空间可以包含易失性或非易失性程序存储器、非易失性数据存储器以及外部总线接口（External Bus Interface, EBI）。例如，如果写表指令在 EBI 地址区域内执行，写入的数据将放在 EBI 数据线上。

5.2.2.1 写表保持锁存器

写表指令不会直接写非易失性程序和数据存储器。相反，写表指令会装入存储写入数据的保持锁存器（Holding Latch）。保持锁存器不是存储器映射的，并且只能使用写表指令访问。当所有的保持锁存器都被装入后，通过执行一个特殊的指令序列即可开始实际存储器编程操作。

保持锁存器的数量将决定能够编程的最大存储区大小，并且它会随着非易失性存储器类型和器件的不同而变化。例如，对于某个特定的器件，保持锁存器的数量可能会因为程序存储器、数据 EEPROM 存储器和器件配置寄存器的不同而不同。

一般来说，程序存储器会分段为行和板（Panel）。每个板都具有一组写表保持锁存器。这可以允许一次编程多个存储器板，从而降低器件的总编程时间。对于每个存储器板，一般都有足够的保持锁存器用于一次编程一行存储区。存储器逻辑电路会根据写表指令使用的地址值自动判断要装入哪一组写锁存器。

更多详细信息，请参考特定的器件数据手册。

5.2.2.2 字模式写入

可使用以下序列以字模式写入一个程序存储器锁存单元。

```
; Setup the address pointer to program space
MOV    #tblpage(PROG_ADDR),W0    ; get table page value
MOV    W0,TBLPAG                 ; load TBLPAG register
MOV    #tbloffset(PROG_ADDR),W0  ; load address LS word
; Load write data into W registers
MOV    #PROG_LOW_WORD,W2
MOV    #PROG_HI_BYTE,W3
; Perform the table writes to load the latch
TBLWTL W2,[W0]
TBLWTH W3,[W0++]
```

在此示例中，W3 的高字节部分无关紧要，因为此数据将会被写入到虚拟字节单元中。在第二个 TBLWTH 指令后，W0 被后加 2 以准备写入下一个程序存储器单元。

5.2.2.3 字节模式写入

要以字节模式写入一个程序存储器锁存单元，可以使用以下代码序列：

```
; Setup the address pointer to program space
MOV     #tblpage(PROG_ADDR),W0      ; get table page value
MOV     W0,TBLPAG                   ; load TBLPAG register
MOV     #tbloffset(PROG_ADDR),W0    ; load address LS word
; Load data into working registers
MOV     #LOW_BYTE,W2
MOV     #MID_BYTE,W3
MOV     #HIGH_BYTE,W4
; Write data to the latch
TBLWTH.B W4,[W0]                    ; write high byte
TBLWTL.B W2,[W0++]                  ; write low byte
TBLWTL.B W3,[W0++]                  ; write middle byte
```

在上面的代码示例中，在写入低字节时的后加操作会导致 W0 中的地址加 1。这会将 EA<0> 置为 1，以访问第三个写指令中的中间字节。最后的后加将 W0 设置回一个偶数地址，以指向下一个程序存储器单元。

5.3 控制寄存器

闪存和数据 EEPROM 编程操作是使用以下非易失性存储器（Non-Volatile Memory，NVM）控制寄存器控制的：

- NVMCON：非易失性存储器控制寄存器
- NVMKEY：非易失性存储器密钥寄存器（Key Register）
- NVMADR：非易失性存储器地址寄存器

5.3.1 NVMCON 寄存器

NVMCON 寄存器是闪存和 EEPROM 编程 / 擦除操作的主控制寄存器。此寄存器选择闪存或 EEPROM 存储器，确定执行的将是擦除还是编程操作，并用于开始编程或擦除周期。

寄存器 5-1 所示为 NVMCON 寄存器。NVMCOM 的低字节用于配置将执行的 NVM 操作类型。为了方便起见，表 5-1 概括各种编程和擦除操作的 NVMCON 设置值。

表 5-1： NVMCON 寄存器值

RTSP 编程和擦除操作的 NVMCON 寄存器值			
存储器类型	操作	数据大小	NVMCON 值
闪存 PM	擦除	1 行（32 个指令字）	0x4041
	编程	1 行（32 个指令字）	0x4001
数据 EEPROM	擦除	1 个数据字	0x4044
		16 个数据字	0x4045
		整个 EEPROM	0x4046
	编程	1 个数据字	0x4004
		16 个数据字	0x4005
配置寄存器	写 (1)	1 个配置寄存器	0x4008

注 1：可以不执行擦除周期而向器件配置寄存器（FG5 除外）写入新值。

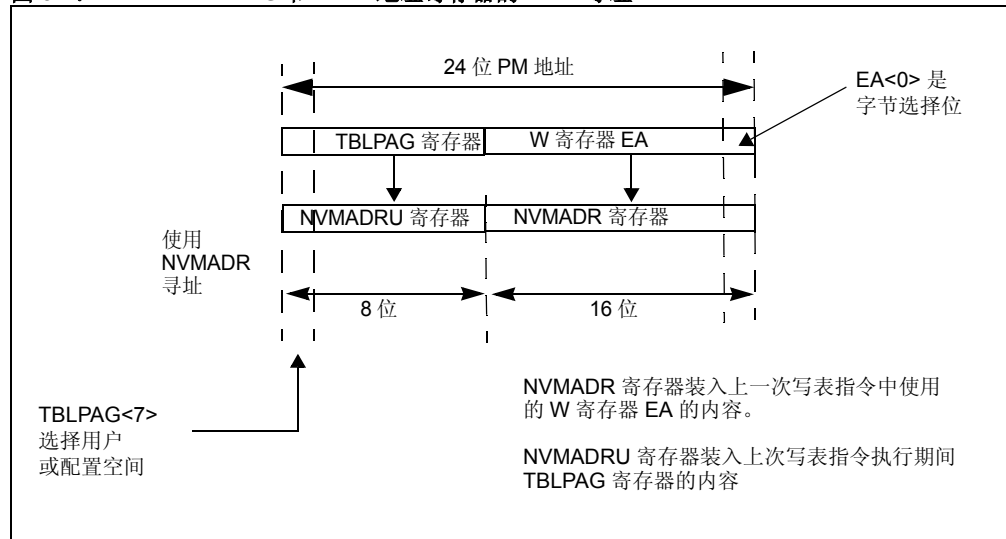
5.3.2 NVM 地址寄存器

有两个 NVM 地址寄存器 NVMADRU 和 NVMADR。将这两个寄存器连在一起可构成编程操作所选行或字的 24 位有效地址 (EA)。NVMADRU 寄存器用于保存 EA 的高 8 位，NVMADR 寄存器则用于保存 EA 的低 16 位。

一对寄存器 NVMADRU:NVMADR 会捕捉上一次执行的写表指令的 EA<23:0>，并选择闪存或 EEPROM 存储器的行进行写入 / 擦除。图 5-2 显示了用于编程和擦除操作的程序存储器 EA 组成方式。

虽然 NVMADRU 和 NVMADR 寄存器会由写表指令自动装入，用户还是可以在编程操作开始前直接修改其内容。在擦除操作前需要先写入这些寄存器，因为任何擦除操作都不需要写表指令。

图 5-2: TBLPAG 和 NVM 地址寄存器的 NVM 寻址



5.3.3 NVMKEY 寄存器

NVMKEY 是一个只写寄存器，用于防止闪存或 EEPROM 存储器的误写 / 误擦除。要开始编程或擦除序列，必须严格按照如下顺序执行下列步骤：

1. 将 0x55 写入 NVMKEY。
2. 将 0xAA 写入 NVMKEY。
3. 执行两个 NOP 指令。

在此序列后，就可以在一个指令周期中写入 NVMCON 寄存器。在多数情况下，用户只需要将 NVMCON 寄存器中的 WR 位置 1 就可以开始编程或擦除周期。在解锁序列中应该禁止中断。下面的代码示例说明了解锁序列是如何执行的：

```

PUSH      SR          ; Disable interrupts, if enabled
MOV       #0x00E0,W0
IOR       SR

MOV       #0x55,W0
MOV       #0xAA,W0
MOV       W0,NVMKEY
MOV       W0,NVMKEY    ; NOP not required
BSET     NVMCON,#WR    ; Start the program/erase cycle
NOP
NOP
POP       SR          ; Re-enable interrupts

```

如需更多编程示例，请参见第 5.4.2 节 “闪存编程操作”。

寄存器 5-1: NVMCON: 非易失性存储器控制寄存器

高字节:							
R/S-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
WR	WREN	WRERR	—	—	—	—	—
bit 15			bit 8				

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PROGOP<7:0>							
bit 7			bit 0				

- bit15

WR:

写（编程或擦除）控制位

1 = 开始数据 EEPROM 或程序闪存擦除或写周期
（只可用软件将 WR 位置 1，但不能清零。）

0 = 写周期完成
- bit14

WREN:

写（擦除或编程）使能位

1 = 使能擦除或编程操作

0 = 不允许任何操作（器件在写 / 擦除操作完成时将此位清零）
- bit13

WRERR:

闪存错误标志位

1 = 写操作提前终止（由于编程操作期间的任何 MCLR 或 WDT 复位）

0 = 写操作成功完成
- bit12-8

保留:

用户代码应该在这些单元中写入 0
- bit7-0

PROGOP<7:0>:

编程操作命令字节位

擦除操作:
0x41 = 从程序闪存中擦除 1 行（32 个指令字）
0x44 = 从数据 EEPROM 擦除 1 个数据字
0x45 = 从数据 EEPROM 擦除 1 行（16 个数据字）
0x46 = 擦除整个 EEPROM

编程操作:
0x01 = 将 1 行（32 指令字）编程入闪存程序存储器
0x04 = 将 1 个数据字编入数据 EEPROM
0x05 = 将 1 行（16 个数据字）编程入数据 EEPROM
0x08 = 将 1 个数据字编程入器件配置寄存器

图注:

R = 可读位	W = 可写位	U = 未用位，读作 0
S = 可设置位	-n = POR 时的值	1 = 置位
0 = 清零	x = 未知	

寄存器 5-2: **NVMADR: 非易失性存储器地址寄存器**

高字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
NVMADR<15:8>							
bit 15				bit 8			

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
NVMADR<7:0>							
bit 7				bit 0			

bit15-0 **NVMADR<15:0>: NV 存储器写地址位**
在程序或数据闪存存储器中选择要编程或擦除的单元。
用户可以读写此寄存器。此寄存器保持上一次执行的写表指令的 **EA<15:0>** 地址，直到用户写入。

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 5-3: NVMADRU: 非易失性存储器高地址寄存器

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15				bit 8			

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
NVMADRU<7:0>							
bit 7				bit 0			

bit15-8 未用：读作 0

bit7-0 **NVMADRU<7:0>**: NV 存储器高地址写入位
选择地址的高 8 位对程序或数据闪存进行编程或擦除。
用户可读写此寄存器。在前一个写表指令执行时，此寄存器中将包含寄存器 TBLPAG 中的值，直到用户写入新值。

图注:

R = 可读位	W = 可写位	U = 未用位，读作 0
-n = POR 的值	1 = 置位	0 = 清零
		x = 未知位

寄存器 5-4: NVMKEY: 非易失性存储器密钥寄存器

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15				bit 8			

低字节:							
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
NVMKEY<7:0>							
bit 7				bit 0			

bit15-8 未用：读作 0

bit7-0 **NVMKEY<7:0>**: 密钥寄存器（只写）位

图注:

R = 可读位	W = 可写位	U = 未用位，读作 0
-n = POR 的值	1 = 置位	0 = 清零
		x = 未知位

5.4 运行时自编程 (RTSP)

RTSP 允许用户代码修改闪存程序存储器的内容。RTSP 是使用 TBLRD (读表)、TBLWT (写表) 指令和 NVM 控制寄存器实现的。通过 RTSP, 用户可以一次在程序存储器中擦除 32 条指令 (96 字节), 也可以在程序存储器中一次写入 32 条指令 (96 字节)。

5.4.1 RTSP 工作原理

dsPIC30F 闪存程序存储器是由行和板构成的。每行由 32 条指令 (96 字节) 组成。板的大小取决于具体的 dsPIC30F 器件。更多信息请参见器件数据手册。通常, 每个板由 128 行组成 (4K × 24 条指令)。RTSP 可以让用户每次擦除一行 (32 条指令) 以及一次编程 32 条指令。

程序存储器的每个板包括能够保存 32 条编程数据指令的写锁存器。这些锁存器不是存储器映射的。用户访问写锁存器的惟一方法是使用写表指令。在实际编程操作前, 必须先用写表指令将待写数据装入板写锁存器。待编程入板的数据通常是按以下顺序装入写锁存器的: 指令 0, 指令 1, 依此类推。装入的指令字必须来自四个地址边界中的“偶数”组合 (例如不允许装入指令 3、4、5 和 6)。换一种说法, 此要求需要四个指令的起始程序存储器地址必须有 3 个 LSb 等于 0。所有的 32 位写锁存器必须在编程操作期间写入, 以确保覆盖保存在锁存器中的旧数据。

RTSP 编程的基本步骤是先建立一个表指针, 然后执行一系列 TBLWT 指令以装入写锁存器。编程是通过将 NVMCON 寄存器的特殊位置 1 进行的。需要将 32 条 TBLWTL 和 32 条 TBLWTH 指令装入四条指令。如果需要对多个不连续的程序存储器区进行编程, 应该为每个区域和下一个要写入的一组写锁存器修改表指针。

因为只能写入表锁存器, 因此所有对闪存程序存储器的写表操作都需要 2 个指令周期。实际的编程操作是使用 NVMCON 寄存器开始的。

5.4.2 闪存编程操作

在 RTSP 模式中对内部闪存程序存储器进行编程或擦除必需使用编程 / 擦除操作。编程或擦除操作由器件自动计时, 持续时间的标称值为 2 ms。将 WR 位 (NVMCON<15>) 置 1 将开始操作, 当操作完成时 WR 位会自动清零。

CPU 将停顿 (等待) 直到编程操作完成。在此期间, CPU 不会执行任何指令, 也不会响应中断。如果在编程周期中发生了任何中断, 则它们将等待到该周期结束后再进行处理。

5.4.2.1 闪存程序存储器编程算法

用户可以按行（32 个指令字）擦除和编程闪存程序存储器。一般过程如下：

1. 读一行程序闪存（32 个指令字）并以数据“镜像”方式保存到数据 RAM。必须从一个偶数 32 字程序存储器地址边界读取 RAM 镜像。
2. 用新的程序存储器数据更新 RAM 数据镜像。
3. 擦除程序闪存行。
 - 设置 NVMCON 寄存器以擦除闪存程序存储器中的一行。
 - 将要擦除的行地址写入 NVMADRU 和 NVMADR 寄存器。
 - 禁止中断。
 - 将密钥序列写入 NVMKEY 以使能擦除。
 - 将 WR 位置 1。这将开始擦除周期。
 - 在擦除周期中 CPU 会停顿。
 - 当擦除周期结束时 WR 位会清零。
 - 重新允许中断。
4. 将 RAM 中的 32 个指令字写入闪存程序存储器写锁存器。
5. 将 32 个指令字编入程序闪存。
 - 设置 NVMCON 以对一行闪存程序存储区编程。
 - 禁止中断。
 - 将密钥序列写入 NVMKEY 以使能编程周期。
 - 将 WR 位置 1。这将开始编程周期。
 - 在编程周期中 CPU 会停顿。
 - 当编程周期结束时 WR 位会被硬件清零。
 - 重新允许中断。
6. 如有必要，重复步骤 1 到 6 以对所需容量的闪存程序存储器编程。

注： 用户应切记使用 RTSP 修改的最小程序存储容量为 32 个指令字单元。因此，在开始擦除周期前，应在通用 RAM 中储存这些单元的镜像，这是很重要的。在任何编程前，必须对任何先前写过的单元执行一个擦除周期。

5.4.2.2 擦除程序存储区的一行

以下为一个可用于擦除程序存储区一行（32 指令）的代码序列。NVMCON 寄存器配置为擦除程序存储区的一行。NVMADRU 和 NVMADR 寄存器装入了待擦除行的地址。程序存储器必须在“偶数”行边界擦除。因此，当在一个行被擦除时，写入 NVMADR 寄存器的值的 6 个 LSB 不起作用。

擦除操作是通过在 WR 控制位（NVMCON<15>）置 1 前，写一个特殊解锁或密钥序列到 NVMKEY 寄存器开始的。必须严格按以下顺序执行此解锁序列，且不可中断。因此，在写此序列前，必须禁止中断。

要让 CPU 恢复工作，应在代码处插入两个 NOP 指令。最后可允许中断（如果需要）。

```
; Setup NVMCON to erase one row of Flash program memory
    MOV    #0x4041,W0
    MOV    W0,NVMCON
; Setup address pointer to row to be ERASED
    MOV    #tblpage(PROG_ADDR),W0
    MOV    W0,NVMADRU
    MOV    #tbloffset(PROG_ADDR),W0
    MOV    W0,NVMADR
; Disable interrupts, if enabled
    PUSH   SR
    MOV    #0x00E0,W0
    IOR    SR
; Write the KEY sequence
    MOV    #0x55,W0
    MOV    W0,NVMKEY
    MOV    #0xAA,W0
    MOV    W0,NVMKEY
; Start the erase operation
    BSET   NVMCON,#WR
; Insert two NOPs after the erase cycle (required)
    NOP
    NOP
; Re-enable interrupts, if needed
    POP    SR
```

注： 当擦除程序存储区的一行时，用户直接将待擦除地址的高 8 位写入 NVMADRU 和 NVMADR 寄存器。NVMADRU 和 NVMADR 寄存器的内容共同构成了待擦除程序存储区行的完整地址。

NVMADRU 和 NVMADR 寄存器指定所有闪存擦除和编程操作的地址。但是，对于闪存程序存储器操作，用户并不一定要直接写入这两个寄存器。这是因为用于写程序存储器的写表指令会自动将 TBLPAG 寄存器的内容和写表地址传输到 NVMADRU 和 NVMADR 寄存器。

可以修改上面的代码示例来执行一个“无效”写表操作，以捕捉程序存储器擦除地址。

5.4.2.3 装入写锁存器

以下为一个可用于向写锁存器装入 768 位（32 指令字）的指令序列。需要将 32 条 TBLWTL 和 32 条 TBLWTH 指令装入由表指针选定的写锁存器。

TBLPAG 寄存器中装入了程序存储器地址的 8 个 MSb。对于闪存编程操作，用户不需要写 NVMADRU:NVMADR 这对寄存器。当执行每个写表指令时，程序存储器地址的 24 位会被自动捕捉到 NVMADRU:NVMADR 这对寄存器中。程序存储器必须在一个“偶数”32 指令字地址边界编程。而 NVMADR 寄存器中捕捉到的值的 6 个 LSb 在编程操作期间并不使用。

32 指令字行不一定要顺序写入。写表地址的 6 个 LSb 决定要写入的锁存器。但是，在每个编程周期应将所有的 32 个指令字写入以覆盖旧数据。

注： 下列代码示例是后续示例中所提到的“Load_Write_Latch”代码。

```
; Set up a pointer to the first program memory location to be written.
MOV          #tblpage(PROG_ADDR),W0
MOV          W0,TBLPAG
MOV          #tbloffset(PROG_ADDR),W0

; Perform the TBLWT instructions to write the latches
; W0 is incremented in the TBLWTH instruction to point to the
; next instruction location.

MOV          #LOW_WORD_0,W2
MOV          #HIGH_BYTE_0,W3
TBLWTL      W2,[W0]
TBLWTH      W3,[W0++]          ; 1st_program_word
MOV          #LOW_WORD_1,W2
MOV          #HIGH_BYTE_1,W3
TBLWTL      W2,[W0]
TBLWTH      W3,[W0++]          ; 2nd_program_word
MOV          #LOW_WORD_2,W2
MOV          #HIGH_BYTE_2,W3
TBLWTL      W2,[W0]
TBLWTH      W3,[W0++]          ; 3rd_program_word
MOV          #LOW_WORD_3,W2
MOV          #HIGH_BYTE_3,W3
TBLWTL      W2,[W0]
TBLWTH      W3,[W0++]          ; 4th_program_word
.....
.....
MOV          #LOW_WORD_31,W2
MOV          #HIGH_BYTE_31,W3
TBLWTL      W2,[W0]
TBLWTH      W3,[W0++]          ; 32nd_program_word
```

5.4.2.4 单行编程示例

以下为单行编程示例：

```
; Setup NVMCON to write 1 row of program memory
MOV     #0x4001,W0
MOV     W0,NVMCON

; Load the 32 program memory write latches
CALL    Load_Write_Latch(1)

; Disable interrupts, if enabled
PUSH    SR
MOV     #0x00E0,W0
IOR     SR
; Write the KEY sequence
MOV     #0x55,W0
MOV     W0,NVMKEY
MOV     #0xAA,W0
MOV     W0,NVMKEY
; Start the programming sequence
BSET    NVMCON,#WR
; Insert two NOPs after programming
NOP
NOP
; Re-enable interrupts, if required
POP     SR
```

注 1：参见第 5.4.2.3 节 “装入写锁存器”

5.4.3 写入器件配置寄存器

可以使用 RTSP 写器件配置寄存器。RTSP 允许不首先执行擦除周期而分别重写每个配置寄存器（FG5 除外）。当写入配置寄存器时必须要小心，因为它们控制着关键的器件工作参数，如系统时钟源、PLL 倍频比例和 WDT 使能等。

编程一个器件配置寄存器的步骤与闪存程序存储器的编程步骤类似，不同之处在于此时只需要 TBLWTL 指令。这是因为在各器件配置寄存器中未使用高 8 位。此外，必须置位写表地址的第 23 位以访问配置寄存器。有关器件配置寄存器的完整描述，请参见第 24 章 “器件配置” 和器件数据手册。

5.4.3.1 配置寄存器的写入算法

1. 使用 TBLWTL 指令将新配置值写入写表锁存器。
2. 配置 NVMCON（NVMCON = 0x4008）以允许写入配置寄存器。
3. 如果已经允许了中断，请将其禁止。
4. 将密钥序列写入 NVMKEY。
5. 通过将 WR（NVMCON<15>）置 1 开始写序列
6. 当写入完成后将恢复 CPU 执行。
7. 如有必要，重新允许中断。

5.4.3.2 写配置寄存器的代码示例

下列代码序列可用于修改器件配置寄存器：

```
; Set up a pointer to the location to be written.
MOV      #tblpage(CONFIG_ADDR),W0
MOV      W0,TBLPAG
MOV      #tbloffset(CONFIG_ADDR),W0
; Get the new data to write to the configuration register
MOV      #ConfigValue,W1
; Perform the table write to load the write latch
TBLWTL   W1,[W0]
; Configure NVMCON for a configuration register write
MOV      #0x4008,W0
MOV      W0,NVMCON
; Disable interrupts, if enabled
PUSH     SR
MOV      #0x00E0,W0
IOR      SR
; Write the KEY sequence
MOV      #0x55,W0
MOV      W0,NVMKEY
MOV      #0xAA,W0
MOV      W0,NVMKEY
; Start the programming sequence
BSET     NVMCON,#WR
; Insert two NOPs after programming
NOP
NOP
; Re-enable interrupts, if required
POP      SR
```

5.5 数据 EEPROM 编程

与程序存储器类似，EEPROM 存储块（Block）是通过读表和写表操作访问的。因为 EEPROM 存储器只有 16 位宽，所以其操作不需要使用 TBLWTH 和 TBLRDH 指令。数据 EEPROM 的编程和擦除步骤与闪存程序存储器类似，区别在于数据 EEPROM 为快速数据存取进行了优化。在数据 EEPROM 上可以执行以下编程操作：

- 擦除一个字
- 擦除一行（16 个字）
- 擦除整个数据 EEPROM
- 编程一个字
- 编程一行（16 个字）

在正常操作中（整个 VDD 工作范围），数据 EEPROM 可读写。与闪存程序存储器不同，在 EEPROM 编程或擦除操作时，正常程序执行不会停止。

EEPROM 擦除和编程操作是通过 NVMCON 和 NVMKEY 寄存器执行的。编程软件负责等待操作完成。软件可以使用以下三种方法之一检测 EEPROM 擦除或编程操作的完成时间：

- 用软件查询 WR 位（NVMCON<15>）。当操作完成时 WR 位会被清零。
- 用软件查询 NVMIF 位（IFS0<12>）。当操作完成时 NVMIF 位会被置 1。
- 允许 NVM 中断。当操作完成时，CPU 会被中断。ISR 可以处理更多的编程操作。

注： 当编程或擦除操作执行过程中，如果用户试图读取 EEPROM，会得到不可预料的结果。

5.5.1 EEPROM 单字编程算法

1. 擦除一个 EEPROM 字。
 - 设置 NVMCON 寄存器以擦除一个 EEPROM 字。
 - 将要擦除的字的地址写入 NVMADRU 和 NVMADR 寄存器。
 - 将 NVMIF 状态位清零并允许 NVM 中断（可选）。
 - 将密钥序列写入 NVMKEY。
 - 将 WR 位置 1。这将开始擦除周期。
 - 查询 WR 位或等待 NVM 中断。
2. 将数据字写入数据 EEPROM 写锁存器。
3. 将数据字编程入 EEPROM。
 - 设置 NVMCON 寄存器以编程一个 EEPROM 字。
 - 将 NVMIF 状态位清零并允许 NVM 中断（可选）。
 - 将密钥序列写入 NVMKEY。
 - 将 WR 位置 1。这将开始编程周期。
 - 查询 WR 位或等待 NVM 中断。

5.5.2 EEPROM 行编程算法

如果需要将多个字编程入 EEPROM，每次擦除并编程 16 个字（1 行）会比较快。向 EEPROM 编程 16 个字的过程如下：

1. 读一行数据 EEPROM（16 个字）并以数据“镜像”方式保存到数据 RAM。要修改的 EEPROM 部分必须处于偶数 16 字地址边界内。
2. 使用新数据更新数据镜像。
3. 擦除 EEPROM 行。
 - 设置 NVMCON 寄存器以擦除 EEPROM 的一行。
 - 将要擦除的行的起始地址写入寄存器 NVMADRU 和 NVMADR。
 - 将 NVMIF 状态位清零并允许 NVM 中断（可选）。
 - 将密钥序列写入 NVMKEY。
 - 将 WR 位置 1。这将开始擦除周期。
 - 查询 WR 位或等待 NVM 中断。
4. 将 16 个数据字写入数据 EEPROM 写锁存器。
5. 将一行数据编程到数据 EEPROM。
 - 设置 NVMCON 寄存器以编程 EEPROM 的一行。
 - 将 NVMIF 状态位清零并允许 NVM 中断（可选）。
 - 将密钥序列写入 NVMKEY。
 - 将 WR 位置 1。这将开始编程周期。
 - 查询 WR 位或等待 NVM 中断。

5.5.3 擦除数据 EEPROM 存储器的一个字

NVMADRU 和 NVMADR 寄存器中必须装入要擦除的数据 EEPROM 的地址。因为只访问 EEPROM 的一个字，NVMADR 的所有 LSb 对擦除操作没有影响。必须配置 NVMCON 寄存器以擦除 EEPROM 存储器的一个字。

将 WR 控制位置 1 (NVMCON<15>) 开始擦除。在将 WR 控制位置 1 前应该在 NVMKEY 寄存器中写入一个特殊的解锁或密钥序列。需严格按以下顺序执行此解锁序列，且不可中断。因此，在写此序列前，必须禁止中断。

```
; Set up a pointer to the EEPROM location to be erased.
MOV      #tblpage(EE_ADDR),W0
MOV      W0,NVMADRU
MOV      #tbloffset(EE_ADDR),W0
MOV      W0,NVMADR
; Setup NVMCON to erase one word of data EEPROM
MOV      #0x4044,W0
MOV      W0,NVMCON
; Disable interrupts while the KEY sequence is written
PUSH     SR
MOV      #0x00E0,W0
IOR      SR
; Write the KEY sequence
MOV      #0x55,W0
MOV      W0,NVMKEY
MOV      #0xAA,W0
MOV      W0,NVMKEY
; Start the erase cycle
BSET     NVMCON,#WR
; Re-enable interrupts
POP      SR
```

5.5.4 写数据 EEPROM 存储器中的一个字

假设用户已经擦除了要编程的 EEPROM 单元，使用写表指令写一个写锁寄存器。TBLPAG 寄存器中装入了 EEPROM 地址的 8 个 MSb。当执行写表时，EEPROM 地址的 16 个 LSb 会被自动捕捉到 NVMADR 寄存器中。NVMADR 寄存器的所有 LSb 对编程操作没有影响。配置 NVMCON 寄存器以编程数据 EEPROM 的一个字。

将 WR 控制位置 1 (NVMCON<15>) 开始编程操作。在将 WR 控制位置 1 前应该在 NVMKEY 寄存器中写入一个特殊的解锁或密钥序列。需严格按以下顺序执行此解锁序列，且不可中断。因此，在写此序列前，必须禁止中断。

```
; Setup a pointer to data EEPROM
MOV          #tblpage(EE_ADDR),W0
MOV          W0,TBLPAG
MOV          #tbloffset(EE_ADDR),W0
; Write data value to holding latch
MOV          EE_DATA,W1
TBLWTL      W1,[ W0]
; NVMADR captures write address from the TBLWTL instruction.
; Setup NVMCON for programming one word to data EEPROM
MOV          #0x4004,W0
MOV          W0,NVMCON
; Disable interrupts while the KEY sequence is written
PUSH        SR
MOV          #0x00E0,W0
IOR          SR
; Write the key sequence
MOV          #0x55,W0
MOV          W0,NVMKEY
MOV          #0xAA,W0
MOV          W0,NVMKEY
; Start the write cycle
BSET        NVMCON,#WR
; Re-enable interrupts, if needed
POP         SR
```

5.5.5 擦除数据 EEPROM 的一行

配置 NVMCON 寄存器以擦除 EEPROM 存储器的一行。NVMADRU 和 NVMADR 寄存器必须指向要擦除的行。必须在偶数地址边界擦除数据 EEPROM。因此，NVMADR 寄存器的 5 个 LSb 对擦除的行没有影响。

将 WR 控制位置 1（NVMCON<15>）开始擦除。在将 WR 控制位置 1 前应该在 NVMKEY 寄存器中写入一个特殊的解锁或密钥序列。需严格按以下顺序执行此解锁序列，且不可中断。因此，在写此序列前，必须禁止中断。

```
; Set up a pointer to the EEPROM row to be erased.
MOV      #tblpage(EA_ADDR),W0
MOV      W0,NVMADRU
MOV      #tbloffset(EA_ADDR),W0
MOV      W0,NVMADR
; Setup NVMCON to erase one row of EEPROM
MOV      #0x4045,W0
MOV      W0,NVMCON
; Disable interrupts while the KEY sequence is written
PUSH     SR
MOV      #0x00E0,W0
IOR      SR
; Write the KEY Sequence
MOV      #0x55,W0
MOV      W0,NVMKEY
MOV      #0xAA,W0
MOV      W0,NVMKEY
; Start the erase operation
BSET     NVMCON,#WR
; Re-enable interrupts, if needed
POP      SR
```

5.5.6 写数据 EEPROM 存储器的一行

要写数据 EEPROM 的一行，必须在编程序列开始前写入所有的十六个写锁寄存器。TBLPAG 寄存器中装入了 EEPROM 地址的 8 个 MSb。当执行每次写表时，EEPROM 地址的 16 个 LSb 会被自动捕捉到 NVMADR 寄存器中。数据 EEPROM 行编程必须在偶数地址边界上发生，因此 NVMADR 的 5 个 LSb 对编程的行没有影响。

将 WR 控制位置 1 (NVMCON<15>) 开始编程操作。在将 WR 控制位置 1 前应该在 NVMKEY 寄存器中写入一个特殊的解锁或密钥序列。需严格按以下顺序执行此解锁序列，且不可中断。因此，在写此序列前，必须禁止中断。

```
; Set up a pointer to the EEPROM row to be programmed.
MOV      #tblpage(EE_ADDR),W0
MOV      W0,TBLPAG
MOV      #tbloffset(EE_ADDR),W0
; Write the data to the programming latches.
MOV      data_ptr,W1      ; Use W1 as pointer to the data.
TBLWTL [W1++],[W0++]      ; Write 1st data word
TBLWTL [W1++],[W0++]      ; Write 2nd data word
TBLWTL [W1++],[W0++]      ; Write 3rd data word
TBLWTL [W1++],[W0++]      ; Write 4th data word
TBLWTL [W1++],[W0++]      ; Write 5th data word
TBLWTL [W1++],[W0++]      ; Write 6th data word
TBLWTL [W1++],[W0++]      ; Write 7th data word
TBLWTL [W1++],[W0++]      ; Write 8th data word
TBLWTL [W1++],[W0++]      ; Write 9th data word
TBLWTL [W1++],[W0++]      ; Write 10th data word
TBLWTL [W1++],[W0++]      ; Write 11th data word
TBLWTL [W1++],[W0++]      ; Write 12th data word
TBLWTL [W1++],[W0++]      ; Write 13th data word
TBLWTL [W1++],[W0++]      ; Write 14th data word
TBLWTL [W1++],[W0++]      ; Write 15th data word
TBLWTL [W1++],[W0++]      ; Write 16th data word
; The NVMADR captures last table access address.
; Setup NVMCON to write one row of EEPROM
MOV      #0x4005,W0
MOV      W0,NVMCON
; Disable interrupts while the KEY sequence is written
PUSH     SR
MOV      #0x00E0,W0
IOR      SR
; Write the KEY sequence
MOV      #0x55,W0
MOV      W0,NVMKEY
MOV      #0xAA,W0
MOV      W0,NVMKEY
; Start the programming operation
BSET     NVMCON,#WR
; Re-enable interrupts, if needed
POP      SR
```

注： 此代码段使用了十六个写表指令以使该代码示例更清楚。此代码段也可以通过在一个 REPEAT 循环中使用一个写表指令得到简化。

5.5.7 读数据 EEPROM 存储器

TBLRD 指令会读取当前编程地址处的字。此示例使用了 W0 作为数据闪存的指针。结果存入寄存器 W4。

```
; Setup pointer to EEPROM memory
MOV    #tblpage (EE_ADDR), W0
MOV    W0, TBLPAG
MOV    #tbloffset (EE_ADDR), W0
; Read the EEPROM data
TBLRDL [W0], W4
```

注： 程序空间可视性（PSV）也可用于读取程序存储器地址空间中的单元。有关 PSV 的更多信息，可参见第 4 章“程序存储器”。

5.6 设计技巧

问 1： 我无法让器件正确编程或擦除。我的代码看来是正确的。可能是什么原因呢？

答： 当开始编程或擦除周期时应禁止中断，以确保密钥序列的执行不被打断。通过将当前的 CPU 优先级提高到 7，可以禁止中断。本章中的代码示例是通过将当前的 SR 寄存器值保存在堆栈中，然后使用 0x00E0 与 SR 进行“或”操作以强制 IPL<2:0> = 111 来禁止中断的。如果没有允许优先级为 7 的中断，则可使用 DISI 指令在密钥序列执行时暂时禁止中断。

问 2： 有什么简便的方法可以不使用读表指令读取数据 EEPROM？

答： 数据 EEPROM 映射到了程序存储器空间。使用 PSV 将 EEPROM 区域映射到数据存储器空间。有关 PSV 的更多信息，可参见第 4 章“程序存储器”。

5.7 相关应用笔记

本节列出了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是专门为 dsPIC30F 产品系列编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。目前与闪存和 EEPROM 编程模块相关的应用笔记有：

标题	应用笔记编号
dsPIC30F 在无传感器 BLDC 控制中的应用	AN901

注： 如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

5.8 版本历史

版本 A

这是本文档的初始发行版。

版本 B

此版本包括了对 dsPIC30F 闪存和 EEPROM 编程模块的技术内容修改。

版本 C

此版本加入了本文档更新时已知的所有勘误表。

版本 D

此版本收录了 dsPIC30F 闪存和 EEPROM 编程模块的技术内容的变更。

注:

第 6 章 复位中断

目录

本章包括下列主题：

6.1	简介	6-2
6.2	不可屏蔽陷阱	6-6
6.3	中断处理时序	6-11
6.4	中断控制和状态寄存器	6-14
6.5	中断设置流程	6-42
6.6	设计技巧	6-44
6.7	相关应用笔记	6-45
6.8	版本历史	6-46

6.1 简介

dsPIC30F 中断控制器模块将大量外设中断请求信号减少到一个到 dsPIC30F CPU 的中断请求信号，它具有以下特性：

- 多达 8 个处理器异常和软件陷阱
- 7 个用户可选择的优先级
- 具有多达 62 个向量的中断向量表（Interrupt Vector Table, IVT）
- 每个中断或异常源都有惟一的向量
- 指定的用户优先级中的固定优先级
- 用于支持调试的备用中断向量表（Alternate Interrupt Vector Table, AIVT）
- 固定的中断入口和返回延时

6.1.1 中断向量表

中断向量表（IVT）如图 6-1 所示。IVT 位于程序存储器中，起始单元地址是 0x000004。IVT 包含 62 个向量，这些向量由 8 个不可屏蔽陷阱向量和最多 54 个中断源组成。一般来说，每个中断源都有自己的向量。每个中断向量都包含 24 位宽的地址。编程到每个中断向量单元的值是有关中断服务程序（ISR）的起始地址。

6.1.2 备用向量表

如图 6-1 所示，备用中断向量表（AIVT）位于 IVT 之后。ALTIVT 控制位（INTCON2<15>）提供对 AIVT 的访问。如果 ALTIVT 位置 1，所有中断和异常处理将使用备用向量而不是默认的向量。备用向量与默认向量的结构相同。

AIVT 支持仿真和调试功能，它提供了一种不需要将中断向量再编程就可以在应用和支持环境之间切换的方法。此特性也支持运行时在不同应用之间切换以便评估各种软件算法。如果不需要 AIVT，应该用 IVT 中所使用的相同地址编程 AIVT。

6.1.3 复位顺序

由于复位处理中不包含中断控制器，所以器件复位并不是真的异常情况。dsPIC30F 器件清零它的寄存器，作为对迫使 PC 为零的复位的响应。然后处理器开始在地址为 0x000000 的单元处执行程序。用户在复位地址上编程 GOTO 指令会使程序执行重新定位到相应的起始程序。

注： 应该使用包含 RESET 指令的默认中断处理程序的地址编程 IVT 和 AIVT 中的所有未用的向量单元。

图 6-1: 中断向量表

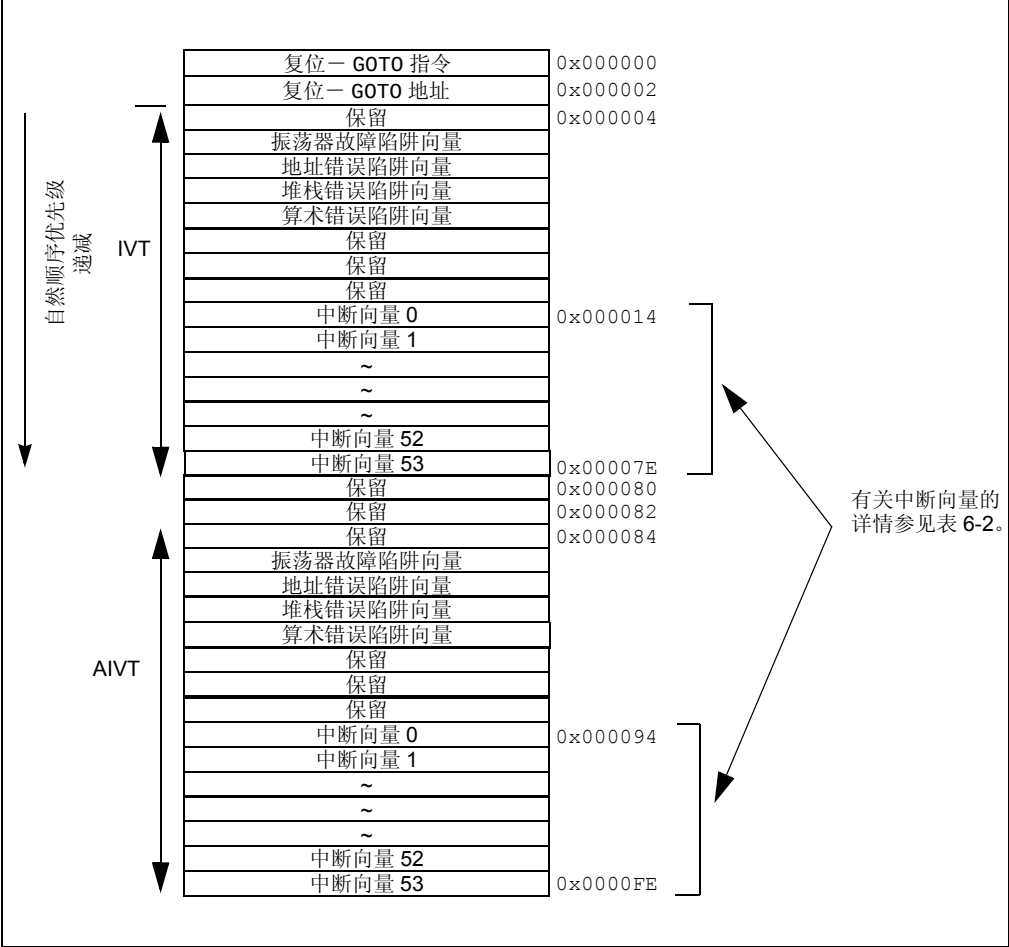


表 6-1: 陷阱向量详情

向量编号	IVT 地址	AIVT 地址	陷阱源
0	0x000004	0x000084	保留
1	0x000006	0x000086	振荡器故障
2	0x000008	0x000088	地址错误
3	0x00000A	0x00008A	堆栈错误
4	0x00000C	0x00008C	算术错误
5	0x00000E	0x00008E	保留
6	0x000010	0x000090	保留
7	0x000012	0x000092	保留

表 6-2: 中断向量详情

向量编号	IVT 地址	AIVT 地址	中断源
8	0x000014	0x000094	INT0——外部中断 0
9	0x000016	0x000096	IC1——输入比较 1
10	0x000018	0x000098	OC1——输出比较 1
11	0x00001A	0x00009A	T1——Timer 1
12	0x00001C	0x00009C	IC2——输入捕捉 2
13	0x00001E	0x00009E	OC2——输出比较 2
14	0x000020	0x0000A0	T2——Timer 2
15	0x000022	0x0000A2	T3——Timer 3
16	0x000024	0x0000A4	SPI1
17	0x000026	0x0000A6	U1RX——UART1 接收器
18	0x000028	0x0000A8	U1TX——UART1 发送器
19	0x00002A	0x0000AA	ADC——ADC 转换完成
20	0x00002C	0x0000AC	NVM——NVM 写完成
21	0x00002E	0x0000AE	I ² C 从操作——报文检测
22	0x000030	0x0000B0	I ² C 主操作——报文事件完成
23	0x000032	0x0000B2	电平变化通知中断
24	0x000034	0x0000B4	INT1——外部中断 1
25	0x000036	0x0000B6	IC7——输入捕捉 7
26	0x000038	0x0000B8	IC8——输入捕捉 8
27	0x00003A	0x0000BA	OC3——输出比较 3
28	0x00003C	0x0000BC	OC4——输出比较 4
29	0x00003E	0x0000BE	T4——Timer 4
30	0x000040	0x0000C0	T5——Timer 5
31	0x000042	0x0000C2	INT2——外部中断 2
32	0x000044	0x0000C4	U2RX——UART2 接收器
33	0x000046	0x0000C6	U2TX——UART2 发送器
34	0x000048	0x0000C8	SPI2
35	0x00004A	0x0000CA	CAN1
36	0x00004C	0x0000CC	IC3——输入捕捉 3
37	0x00004E	0x0000CE	IC4——输入捕捉 4
38	0x000050	0x0000D0	IC5——输入捕捉 5
39	0x000052	0x0000D2	IC6——输入捕捉 6
40	0x000054	0x0000D4	OC5——输出比较 5
41	0x000056	0x0000D6	OC6——输出比较 6
42	0x000058	0x0000D8	OC7——输出比较 7
43	0x00005A	0x0000DA	OC8——输出比较 8
44	0x00005C	0x0000DC	INT3——外部中断 3
45	0x00005E	0x0000DE	INT4——外部中断 4
46	0x000060	0x0000E0	CAN2
47	0x000062	0x0000E2	PWM——PWM 周期匹配
48	0x000064	0x0000E4	QE1——位置计数器比较
49	0x000066	0x0000E6	DCI——编解码器传输完成
50	0x000068	0x0000E8	LVD——低压检测
51	0x00006A	0x0000EA	FLTA——MCPWM 故障 A
52	0x00006C	0x0000EC	FLTB——MCPWM 故障 B
53-61	0x00006E-0x00007E	0x00006E-0x00007E	保留

6.1.4 CPU 优先级状态

CPU 可以工作在 16 个优先级 (0 – 15) 之中的一个。中断或陷阱源的优先级必须大于当前 CPU 的优先级以便开始异常处理。外设和外部中断源可以编程为优先级 0 – 7，而 CPU 优先级 8 – 15 是为陷阱源保留的。陷阱是不可屏蔽的中断源，用于检测硬件和软件问题（参见第 6.2 节“不可屏蔽陷阱”）。每个陷阱源的优先级是固定的并且一个优先级只可分配给一个陷阱。注意编程为优先级 0 的中断源是被有效禁止的，因为它的优先级永远不会大于 CPU 的优先级。

以下四个状态位用于显示当前的 CPU 优先级：

- SR<7:5> 中的 IPL<2:0> 状态位
- CORCON<3> 中的 IPL3 状态位

IPL<2:0> 状态位是可读写的，这样用户可以修改这些位以禁止所有优先级低于给定优先级的中断源。例如，当 IPL<2:0> = 011 时，CPU 就不会被任何编程的优先级为 0、1、2 或 3 的中断源中断。

陷阱事件的优先级比任何用户中断源都高。当 IPL3 位被置 1 时表示正在处理陷阱事件。用户可以清零 IPL3 位，但不能将其置 1。在某些应用中，人们可能会倾向于在发生陷阱时将 IPL3 位清零，并跳转到另一条，而不是原先导致陷阱发生的那条指令的下一条。

可通过设置 IPL<2:0> = 111 禁止所有用户中断源。

注： 当中断嵌套被禁止时，IPL<2:0> 位变成只读位。更多信息请参见第 6.2.4.2 节“中断嵌套”。

6.1.5 中断优先级

可以为每个外设中断源分配 7 个优先级之一。每个单独的中断的用户可分配中断优先级控制位位于 IPCx 寄存器中每个半字节的 3 位最低有效位中。每个半字节的 Bit 3 不使用并读作 0。这些位定义了分配给特定中断的优先级。可用的优先级从 1 开始为最低优先级，7 级为最高优先级。如果与中断源有关的 IPC 位被全部清零，则中断源被有效禁止。

由于特定的优先级会被分配给一个以上的中断请求源，所以在给定的用户分配级别内提供了一种解决优先级冲突的方法。根据每个中断源在 IVT 中的位置，它们都有一个自然顺序优先级。表 6-2 所示为每个中断源在 IVT 中的位置。中断向量的编号越低，自然优先级越高，而向量的编号越高，自然优先级越低。任何待处理的中断源的总优先级都首先由该中断源在 IPCx 寄存器中用户分配的优先级决定，然后由 IVT 中的自然顺序优先级决定。

自然顺序优先级只用于解决具有相同用户分配优先级的同时待处理的中断之间的冲突。一旦优先级冲突被解决，异常处理过程就开始了，CPU 只可以被具有较高用户分配优先级的中断源中断。具有相同的用户分配优先级但具有较高的自然顺序优先级的中断在异常处理过程开始后成为待处理的中断，它将保持待处理状态直到当前的异常处理过程结束。

用户能够为每个中断源分配七个优先级之一表示用户可以给低自然顺序优先级的中断分配非常高的总优先级。例如：可以给 PLVD（可编程低压检测）分配优先级 7 并给 INT0（外部中断 0）分配优先级 1，这样就给了它一个很低的有效优先级。

注： IVT 中的外设和中断源随特定 dsPIC30F 器件而有所不同。此文档中所列的中断源为 dsPIC30F 器件中找到的所有中断源的综合列表。更多详细信息请参见特定的器件数据手册。

6.2 不可屏蔽陷阱

可以将陷阱看作不可屏蔽的可嵌套中断，它遵循固定的优先级结构。陷阱旨在为用户提供一种方法改正正在调试和应用中工作时的错误操作。如果用户不想对陷阱错误条件事件采取纠正行动，那么必须在陷阱向量中装入将在器件复位的软件程序地址。否则，陷阱向量将编程为纠正陷阱条件的服务程序地址。

dsPIC30F 有四个不可屏蔽陷阱源：

- 振荡器故障陷阱
- 堆栈错误陷阱
- 地址错误陷阱
- 算术错误陷阱

注意，很多陷阱条件只有在发生的时候才能检测到。因此，引起陷阱的指令可在异常处理开始之前完成。所以，用户必须改正会引起陷阱的指令行为。

每个陷阱源具有固定的优先级，如同它在 IVT 中的位置。振荡器故障陷阱具有最高的优先级，而算术错误陷阱具有最低的优先级（参见图 6-1）。此外，陷阱源分为“硬”陷阱和“软”陷阱两类。

6.2.1 软陷阱

算术错误陷阱（优先级 11）和堆栈错误陷阱（优先级 12）都属于“软”陷阱源。软陷阱可以视为不可屏蔽中断源，它们的优先级如其在 IVT 中的位置并保持不变。软陷阱的处理过程和中断相像，在异常处理之前，它需要两个周期进行采样和响应。因此，在软陷阱被响应之前可以执行一些额外指令。

6.2.1.1 堆栈错误陷阱（软陷阱，优先级 12）

发生复位时，堆栈初始化为 0x0800。只要堆栈指针地址小于 0x0800，就会产生堆栈错误陷阱。

有一个与堆栈指针相关的堆栈极限寄存器（SPLIM），在复位时不初始化。在对 SPLIM 进行读写操作之前，堆栈溢出检测不被使能。

所有将 W15 用作源或目标指针而产生的有效地址（EA）将与 SPLIM 中的值作比较。如果 EA 大于 SPLIM 寄存器中的内容，将产生堆栈错误陷阱。此外，如果 EA 计算值超过了数据空间的结束地址（0xFFFF），也会产生堆栈错误陷阱。

可以在软件中通过查询 STKERR 状态位（INTCON1<2>）检测堆栈错误。要避免再次进入陷阱服务程序，必须在用 RETFIE 指令从陷阱返回之前用软件清零 STKERR 状态标志位。

6.2.1.2 算术错误陷阱（软陷阱，级别 11）

以下事件中的任何一件都会导致算术错误陷阱产生：

- 累加器 A 溢出
- 累加器 B 溢出
- 灾难性累加器溢出
- 除以 0
- 移位累加器（SFTAC）运算超过 +/-16 位

INTCON1 寄存器中有三个使能位，可使能三种类型的累加器溢出陷阱。OVATE 控制位（INTCON1<10>）用于使能累加器 A 溢出事件的陷阱。OVBTE 控制位（INTCON1<9>）用于使能累加器 B 溢出事件的陷阱。COVTE 控制位（INTCON1<8>）用于使能任何一个累加器灾难性溢出的陷阱。

累加器 A 或累加器 B 溢出事件定义为从 bit 31 进位。注意如果使能了累加器的 31 位饱和模式就不会发生累加器溢出。灾难性累加器溢出定义为从任何一个累加器的 bit 39 进位。如果使能了累加器饱和（31 位或 39 位）就不会发生灾难性溢出。

不能禁止被 0 除陷阱。在执行除法指令的 REPEAT 循环的第一个迭代中执行被 0 除检测。

不能禁止累加器移位陷阱。SFTAC 指令可被用于将累加器移位一个立即数的值或某个 W 寄存器中的值。如果移位值超过 +/-16 位，将产生算术陷阱。此时仍会执行 SFTAC 指令，但移位结果不会被写入目标累加器。

可以通过查询 MATHERR 状态位（INTCON1<4>）在软件中检测到算术错误陷阱。要避免反复进入陷阱服务程序，必须在用 RETFIE 指令从陷阱返回之前用软件清零 MATHERR 状态标志位。在 MATHERR 状态位被清零之前，所有会引起陷阱的条件都必须被清除。如果陷阱是由于累加器溢出而产生的，OA 和 OB 状态位（SR<15:14>）必须清零。OA 和 OB 状态位是只读的，因此用户必须在软件中在溢出的累加器上执行无效操作（比如加 0），从而使得硬件能够清零 OA 或 OB 状态位。

6.2.2 硬陷阱

硬陷阱仅包括优先级 13 到 15 的异常例程。地址错误（优先级 13）和振荡器错误（优先级 14）陷阱都属于这一类。

和软陷阱一样，硬陷阱也可以被看作不可屏蔽的中断源。硬陷阱和软陷阱之间的区别在于引起陷阱的指令执行完之后，硬陷阱会强制 CPU 停止代码执行。在陷阱被响应和处理之前，正常程序执行流程不会恢复。

6.2.2.1 陷阱优先级和硬陷阱冲突

如果在处理任何一个优先级较低的陷阱时发生优先级较高的陷阱，低优先级陷阱的处理会暂时停止而高优先级陷阱将被响应并处理。在高优先级陷阱处理完成之前，低优先级陷阱将保持在待处理状态。

每个发生的硬陷阱均必须先被响应，才可继续执行任何代码。如果在优先级较高的陷阱待处理、响应或处理过程中产生了较低优先级的陷阱，就会产生硬陷阱冲突。产生冲突的原因是在对较高优先级的陷阱处理完成之前，不能响应较低优先级的陷阱。

器件会自动在硬陷阱冲突条件下复位。在发生复位时，TRAPR 状态位（RCON<15>）被置 1，这样就可以用软件检测到冲突条件。

6.2.2.2 振荡器故障陷阱（硬陷阱，级别 14）

以下任何一个原因都将产生振荡器故障陷阱事件：

- 故障保护时钟监视器（Fail-Safe Clock Monitor，FSCM）被使能并检测到系统丢失时钟源。
- 在使用 PLL 的正常工作期间检测到 PLL 失锁。
- FSCM 被使能且 PLL 在上电复位（POR）时锁定失败。

通过查询 OSCFAIL 状态位（INTCON1<1>）或 CF 状态位（OSCCON<3>）可以用软件检测振荡器故障陷阱事件。要避免重复进入陷阱服务程序，必须在用 RETFIE 指令从陷阱返回之前用软件清零 OSCFAIL 状态标志位。

更多有关 FSCM 的信息请参见第 7 章“振荡器”和第 24 章“器件配置”。

6.2.2.3 地址错误陷阱（硬陷阱，级别 13）

以下段落说明会导致地址错误陷阱产生的工作情形：

1. 试图取不对齐的数据字。当一条指令执行了一个有效地址的 LSb 置 1 的字访问时产生此条件。dsPIC30F CPU 要求所有字访问与一个偶地址边界对齐。
2. 一个位操作指令使用有效地址的 LSb 置 1 的间接寻址模式。
3. 试图从未用的数据地址空间获取数据。
4. 执行“BRA #literal”指令或“GOTO #literal”指令，其中 literal 是未用的程序存储器地址。
5. 修改 PC 使其指向未用的程序存储器地址后，执行指令。通过将值装入堆栈和执行 RETURN 指令可修改 PC。

只要发生地址错误陷阱，数据空间写操作就会被禁止，这样数据就不会遭到破坏。

可以通过查询 ADDRERR 状态位（INTCON1<3>）用软件检测到地址错误。要避免重复进入陷阱服务程序，必须在用 RETFIE 指令从陷阱返回之前用软件清零 ADDRERR 状态标志位。

注：	在 MAC 类指令中，数据空间被分成 X 和 Y 空间。在这些指令中，未用的 X 空间包括所有 Y 空间，而未用的 Y 空间包括所有 X 空间。

6.2.3 禁止中断指令

DISI（禁止中断）指令能够将中断禁止长达 16384 个指令周期。当必须执行时间很紧的代码段时，此指令很有用处。

DISI 指令只禁止优先级为 1 — 6 的中断。即使 DISI 指令有效，优先级为 7 的中断和所有陷阱事件仍然可以中断 CPU。

DISI 指令和 DISICNT 寄存器配合工作。当 DISICNT 寄存器非零时，优先级为 1 — 6 的中断被禁止。DISICNT 寄存器在每个后续的指令周期上递减。当 DISICNT 寄存器倒数到 0 时，优先级为 1 — 6 的中断被再次允许。DISI 指令指定的值包括所有由于 PSV 访问，指令停顿等所花费的周期。

DISICNT 寄存器是可读写的。用户可通过清零 DISICNT 寄存器提前终止上一个 DISI 指令的影响。可以通过写入或增加 DISICNT 来延长中断被禁止的时间。

注意如果 DISICNT 寄存器是零，直接写一个非零值到寄存器不能禁止中断。必须首先使用 DISI 指令禁止中断。一旦 DISI 指令被执行且 DISICNT 值成为非零值即可通过修改 DISICNT 的内容来延长中断禁止时间。

注： 建议不要使用软件修改 DISICNT 寄存器。

只要由于执行 DISI 指令造成中断被禁止，DISI 状态位（INTCON2<14>）就被置 1。

注： 如果未分配给 CPU 优先级 7 的中断源，可用 DISI 指令快速禁止所有用户中断源。

6.2.4 中断操作

在每个指令周期都对所有中断事件标志进行采样。IFSx 寄存器中的标志位等于 1 表示有等待处理的中断请求（Interrupt Request, IRQ）。如果中断使能（IECx）寄存器中相应的位被置 1，IRQ 将会导致中断产生。在对 IRQ 采样后余下的指令周期，将评估所有待处理的中断请求的优先级。

当 CPU 响应 IRQ 时，指令不会被中止。当在采样 IRQ 时正在执行的指令将会继续执行完毕，此后才会执行 ISR。

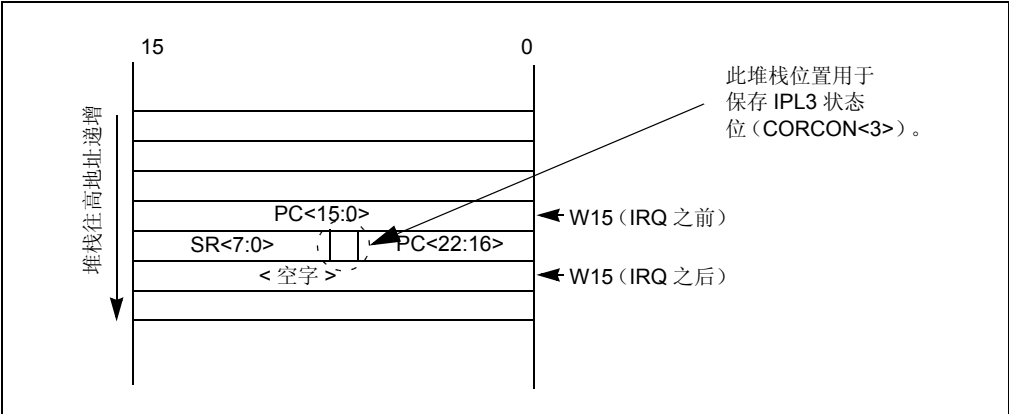
当 IPL<2:0> 状态位（SR<7:5>）表明有一个待处理的 IRQ，它的用户分配的优先级大于当前处理器的优先级，则处理器将收到中断请求。然后处理器将以下信息保存到软件堆栈中：

- 当前的 PC 值
- 处理器状态寄存器（SRL）低字节
- IPL3 状态位（CORCON<3>）

这三个保存在堆栈上的值使返回 PC 地址值、MCU 状态位和当前处理器优先级能够自动保存。

在以上信息被保存在堆栈上之后，CPU 将待处理的中断优先级写入 IPL<2:0> 位的位置。这样将禁止所有优先级小于或等于它的中断，直到使用 RETFIE 指令终止中断服务程序（ISR）。

图 6-2: 中断事件的堆栈操作



6.2.4.1 从中断返回

RETFIE（从中断返回）指令将使 PC 返回地址、IPL3 状态位和 SRL 寄存器出栈以将处理器恢复为中断序列以前的状态和优先级。

6.2.4.2 中断嵌套

在默认情况下中断是可嵌套的。任何正在被处理的 ISR 都可以被另一个具有更高用户分配优先级的中断源中断。可以选择通过置位 NSTDIS 控制位 (INTCON1<15>) 禁止中断嵌套。当 NSTDIS 控制位被置 1 时，所有处理中的中断将通过设置 IPL<2:0> = 111 强制 CPU 的优先级为 7。这将有效屏蔽其他所有中断源，直到执行 RETFIE 指令。当中断嵌套被禁止时，用户分配的中断优先级无效，除非是为了解决同时产生的待处理中断之间的冲突。

当禁止中断嵌套时，IPL<2:0> 位变成只读。这将防止用户软件将 IPL<2:0> 设置为一个较低的值，从而防止其有效地重新使能中断嵌套。

6.2.5 从休眠和空闲模式唤醒

任何使用 IECx 寄存器中相应的控制位分别允许的中断源都可以将处理器从休眠或空闲模式唤醒。当中断源的中断状态标志位被置 1，且中断源通过 IEC 控制寄存器中相应的位被使能时，唤醒信号被发送到 dsPIC30F CPU。当器件从休眠或空闲模式唤醒时，会发生以下两种行为中的一种：

1. 如果该中断源的中断优先级大于当前 CPU 的优先级，处理器将处理该中断并转移到该中断源的 ISR。
2. 如果中断源的用户分配中断优先级小于或等于当前 CPU 的优先级，处理器将继续执行，即开始执行先前将 CPU 置入休眠或空闲模式的 PWRSAV 指令后紧跟的那条指令。

注： 分配为 CPU 优先级 0 的用户中断源不能将 CPU 从休眠或空闲模式唤醒，因为中断源实际上是被禁止的。要使用中断作为唤醒源，中断的 CPU 优先级必须被分配为 CPU 优先级 1 或更高。

6.2.6 A/D 转换器外部转换请求

INT0 外部中断请求引脚与 A/D 转换器共享作为外部转换请求信号引脚。INT0 中断源有可编程的边沿极性，它也可用于 A/D 转换器外部转换请求功能。

6.2.7 外部中断支持

dsPIC30F 支持多达 5 个外部中断引脚源 (INT0—INT4)。每个外部中断引脚都有边沿检测电路以检测中断事件。INTCON2 寄存器有 5 个控制位 (INT0EP—INT4EP)，选择边沿检测电路的极性。每个外部中断引脚都可以被编程为在上升沿或下降沿事件中中断 CPU。更多的详情请参见寄存器 6-4。

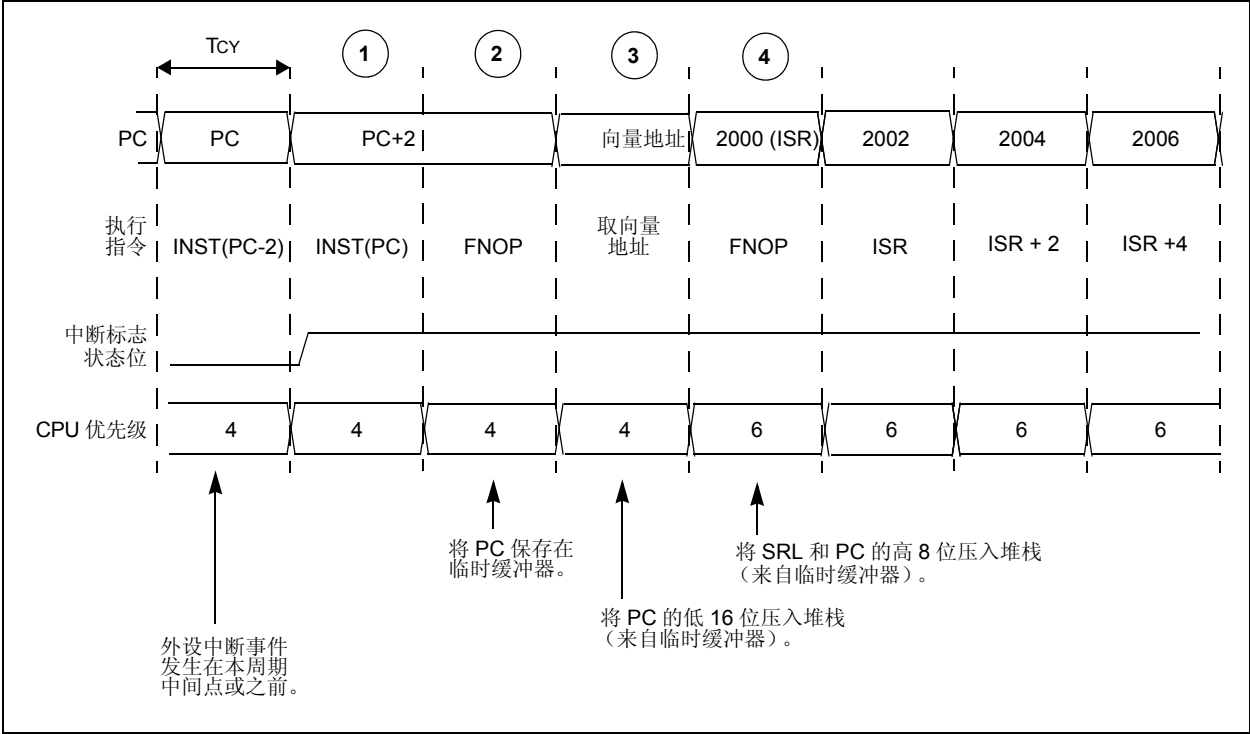
6.3 中断处理时序

6.3.1 单周期指令的中断延迟

图 6-3 所示为在单周期指令中产生外设中断时的事件序列。中断处理需要四个指令周期。每个周期都在图中编号以供参考。

在外设中断发生后的指令周期中，中断标志状态位置 1。在此指令周期中，当前指令完成。在中断事件后的第二个指令周期中，PC 和 SRL 寄存器的内容被存入临时缓冲寄存器。中断处理的第二个周期被执行为一个 NOP，以保持与双周期指令中所进行的序列的一致性（参见第 6.3.2 节“双周期指令的中断延迟”）。在第三个周期中，PC 被装入中断源的向量表地址并取指 ISR 的起始地址。在第四个周期中，PC 被装入 ISR 地址。当 ISR 中的第一个指令被取指时，第四个周期被执行为 NOP。

图 6-3: 单周期指令中的中断时序



6.3.2 双周期指令的中断延迟

双周期指令的中断延迟（Interrupt Latency）和单周期指令相同。中断处理的第一个和第二个周期允许双周期指令完成执行。图 6-4 中的时序图所示为在执行双周期指令之前，指令周期中发生外设中断事件的情况。

图 6-5 给出了外设中断和双周期指令的第一周期同时发生时的时序。在这种情况下，中断处理的完成情况与单周期指令相同（参见第 6.3.1 节“单周期指令的中断延迟”）。

图 6-4: 双周期指令的中断时序

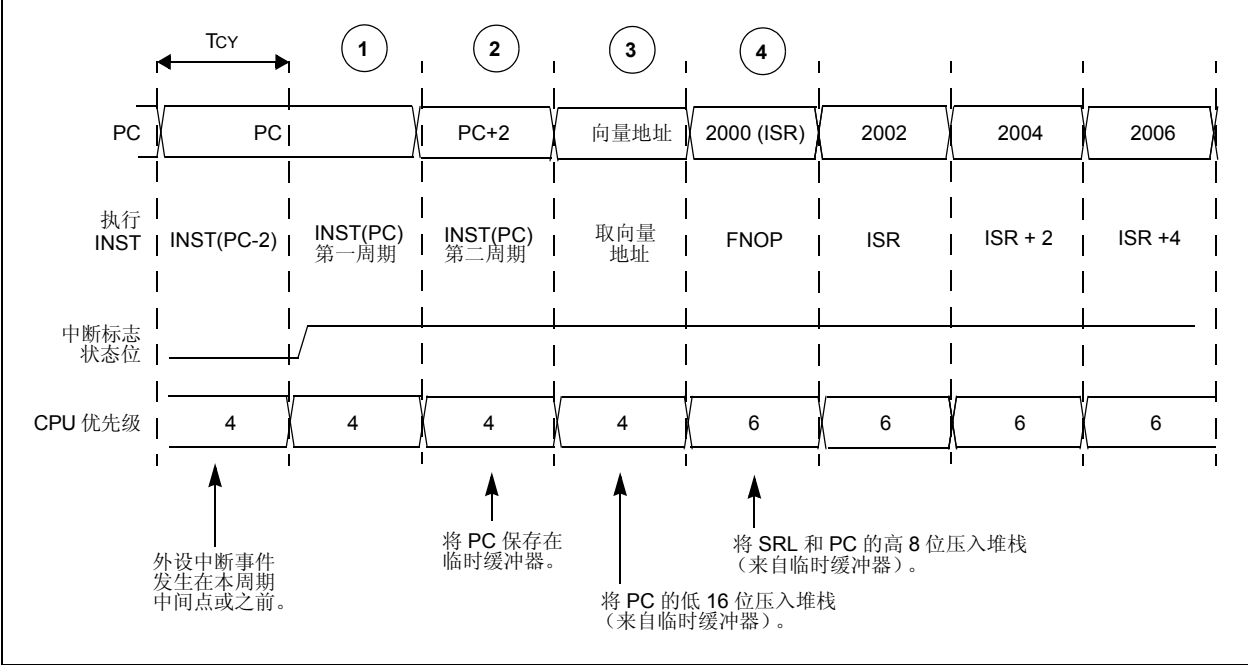
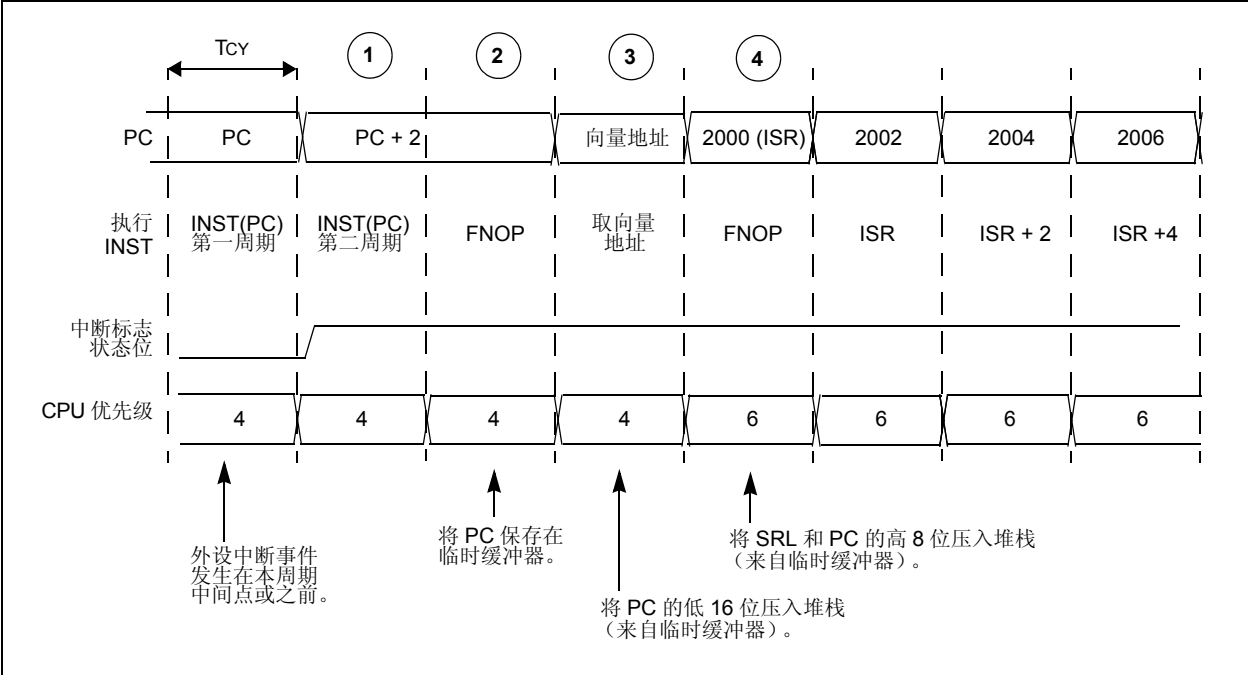


图 6-5: 中断时序，中断在双周期指令的第一个周期发生

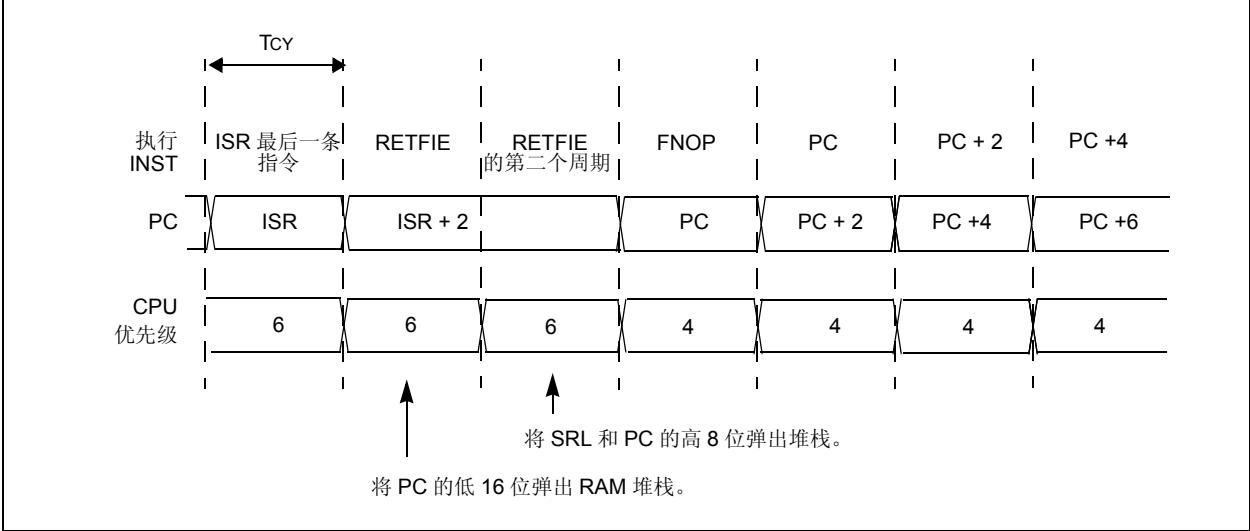


6.3.3 从中断返回

“从中断返回”指令 RETFIE，可退出一个中断或陷阱程序。

在 RETFIE 指令的第一个周期中，PC 的高位和 SRL 寄存器从堆栈弹出。在第二个周期中，入栈的 PC 值的低 16 位从堆栈弹出。第三个指令周期用于取出由更新的程序计数器寻址的指令。此周期执行为一个 NOP。

图 6-6: 从中断返回时序



6.3.4 中断延迟的特殊条件

当外设中断源待处理时，dsPIC30F 允许完成当前指令。对于单周期或双周期指令，中断延迟是一样的。但是，根据中断发生的时间，某些条件可以让中断延迟增加一周。如果固定的中断延迟对应用非常关键的话，用户应该避免出现这些条件。这些条件如下：

- 执行了使用 PSV 访问程序存储空间值的 MOV.D 指令。
- 给所有双周期指令附加一个指令停顿周期。
- 给所有执行 PSV 访问的单周期指令附加一个指令停顿周期。
- 一个位测试并跳过指令（BTSC，BTSS）使用 PSV 访问程序存储空间的值。

6.4 中断控制和状态寄存器

以下寄存器与中断控制器有关：

- **INTCON1 和 INTCON2 寄存器**

全局中断控制功能来自这两个寄存器。INTCON1 包含中断嵌套禁止（Nesting Disable, NSTDIS）位，以及用于处理陷阱源的控制和状态标志。INTCON2 寄存器控制外部中断请求信号的行为和备用向量表的使用。

- **IFSx：中断标志状态寄存器**

所有中断请求标志都保存在 IFSx 寄存器中，其中 “x” 表示寄存器编号。每个中断源都有一个状态位，它们由各自的外设和外部信号置 1 并通过软件清零。

- **IECx：中断允许控制寄存器**

所有中断允许控制位都保存在 IECx 寄存器中，其中 “x” 表示寄存器编号。这些控制位用于分别允许来自外设或外部信号的中断。

- **IPCx：中断优先级控制寄存器**

每个用户中断源都可以分配为 8 个优先级之一。IPC 寄存器用于为每个中断源设置中断优先级。

- **SR：CPU 状态寄存器**

SR 并不是中断控制器硬件中的特定部分，但它包含 IPL<2:0> 状态位（SR<7:5>），该位显示当前 CPU 的优先级。用户可以通过写 IPL 位改变当前 CPU 的优先级。

- **CORCON：内核控制寄存器**

CORCON 也不是中断控制器硬件的特定部分，但它包含 IPL3 状态位，该位显示当前 CPU 的优先级。IPL3 是只读位，因此陷阱事件就不能被用户软件屏蔽。

后面几页将对每个寄存器进行详细描述。

注：	中断源的总数和类型取决于不同的器件。更多详细信息请参见特定的器件数据手册。

6.4.1 控制寄存器的中断分配

分配给 IFSx、IECx 和 IPCx 寄存器的中断源与表 6-2 所列的顺序相同。例如，所示的 INT0（外部中断 0）带有向量编号并且其自然顺序优先级为 “0”。因此，INT0IF 状态位在 IFS0<0> 中。INT0 中断使用 IEC0 寄存器的 bit 0 作为它的允许位，同时 IPC0<2:0> 位为 INT0 中断分配中断优先级。

寄存器 6-1: SR: 状态寄存器 (CPU 中)

高字节:							
R-0	R-0	R/C-0	R/C-0	R-0	R/C-0	R-0	R-0
OA	OB	SA	SB	OAB	SAB	DA	DC
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
IPL<2:0>			RA	N	OV	Z	C
bit 7				bit 0			

bit 7-5 IPL<2:0>: CPU 中断优先级状态位

111 = CPU 中断优先级为 7 (15)。禁止用户中断。

110 = CPU 中断优先级为 6 (14)

101 = CPU 中断优先级为 5 (13)

100 = CPU 中断优先级为 4 (12)

011 = CPU 中断优先级为 3 (11)

010 = CPU 中断优先级为 2 (10)

001 = CPU 中断优先级为 1 (9)

000 = CPU 中断优先级为 0 (8)

注 1: IPL<2:0> 位与 IPL<3> 位 (CORCON<3>) 相连以形成 CPU 中断优先级。如果 IPL<3> = 1, 那么括号中的值表示 IPL。

2: 当 NSTDIS = 1 (INTCON1<15>) 时, IPL<2:0> 状态位为只读位。

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

C = 位可被清零

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

寄存器 6-2: CORCON: 内核控制寄存器

高字节:							
U-0	U-0	U-0	R/W-0	R/W-0	R-0	R-0	R-0
—	—	—	US	EDT	DL<1:0>		
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-1	R/W-0	R/C-0	R/W-0	R/W-0	R/W-0
SATA	SATB	SATDW	ACCSAT	IPL3	PSV	RND	IF
bit 7				bit 0			

bit 3 IPL3: CPU 中断优先级状态位 3

1 = CPU 中断优先级大于 7

0 = CPU 中断优先级小于或等于 7

注 1: IPL3 位与 IPL<2:0> 位 (SR<7:5>) 相连以形成 CPU 中断优先级。

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

C = 位可被清零

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

dsPIC30F 系列参考手册

寄存器 6-3: INTCON1: 中断控制寄存器 1

高字节:							
R/W-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
NSTDIS	—	—	—	—	OVATE	OVBTE	COVTE
bit 15				bit 8			

低字节:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—
bit 7				bit 0			

- bit 15 **NSTDIS:** 中断嵌套禁止位
1 = 禁止中断嵌套
0 = 使能中断嵌套
- bit 14-11 未用: 读作 0
- bit 10 **OVATE:** 累加器 A 溢出陷阱使能位
1 = 使能累加器 A 溢出陷阱
0 = 禁止陷阱
- bit 9 **OVBTE:** 累加器 B 溢出陷阱使能位
1 = 使能累加器 B 溢出陷阱
0 = 禁止陷阱
- bit 8 **COVTE:** 灾难性溢出陷阱使能位
1 = 使能累加器 A 或 B 的灾难性溢出时的陷阱
0 = 禁止陷阱
- bit 7-5 未用: 读作 0
- bit 4 **MATHERR:** 算术错误状态位
1 = 发生了溢出陷阱
0 = 未发生溢出陷阱
- bit 3 **ADDRERR:** 地址错误陷阱状态位
1 = 发生了地址错误陷阱
0 = 未发生地址错误陷阱
- bit 2 **STKERR:** 堆栈错误陷阱状态位
1 = 发生了堆栈错误陷阱
0 = 未发生堆栈错误陷阱
- bit 1 **OSCFAIL:** 振荡器故障陷阱状态位
1 = 发生了振荡器故障陷阱
0 = 未发生振荡器故障陷阱
- bit 0 未用: 读作 0

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 6-4: INTCON2: 中断控制寄存器 2

高字节:							
R/W-0	R-0	U-0	U-0	U-0	U-0	U-0	U-0
ALTIVT	DISI	—	—	—	—	—	—
bit 15		bit 8					

低字节:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
bit 7			bit 0				

- bit 15 **ALTIVT:** 使能备用中断向量表位
1 = 使用备用向量表
0 = 使用标准（默认）向量表
- bit 14 **DISI:** DISI 指令状态位
1 = DISI 指令有效
0 = DISI 指令无效
- bit 13-5 未用：读作 0
- bit 4 **INT4EP:** 外部中断 #4 边沿检测极性选择位
1 = 负边沿处中断
0 = 正边沿处中断
- bit 3 **INT3EP:** 外部中断 #3 边沿检测极性选择位
1 = 负边沿处中断
0 = 正边沿处中断
- bit 2 **INT2EP:** 外部中断 #2 边沿检测极性选择位
1 = 负边沿处中断
0 = 正边沿处中断
- bit 1 **INT1EP:** 外部中断 #1 边沿检测极性选择位
1 = 负边沿处中断
0 = 正边沿处中断
- bit 0 **INT0EP:** 外部中断 #0 边沿检测极性选择位
1 = 负边沿处中断
0 = 正边沿处中断

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 6-5: IFS0: 中断标志状态寄存器 0

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF
bit 15							
							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0IF
bit 7							
							bit 0

- bit 15 **CNIF**: 输入变化通知标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 14 **MI2CIF**: I²C 总线冲突标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 13 **SI2CIF**: I²C 传输完成中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 12 **NVMIF**: 非易失性存储器写完成中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 11 **ADIF**: A/D 转换完成中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 10 **U1TXIF**: UART1 发送器中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 9 **U1RXIF**: UART1 接收器中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 8 **SPI1IF**: SPI1 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 7 **T3IF**: Timer3 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 6 **T2IF**: Timer2 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 5 **OC2IF**: 输出比较通道 2 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 4 **IC2IF**: 输入捕捉通道 2 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 3 **T1IF**: Timer1 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 2 **OC1IF**: 输出比较通道 1 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求

寄存器 6-5: IFS0: 中断标志状态寄存器 0 (续)

- bit 1 **IC1IF:** 输入捕捉通道 1 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 0 **INT0IF:** 外部中断 0 标志状态位
1 = 发生中断请求
0 = 未发生中断请求

图注:			
R = 可读位	W = 可写位	U= 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 6-6: IFS1: 中断标志状态寄存器 1

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF
bit 7							bit 0

- bit 15 **IC6IF:** 输入捕捉通道 6 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 14 **IC5IF:** 输入捕捉通道 5 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 13 **IC4IF:** 输入捕捉通道 4 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 12 **IC3IF:** 输入捕捉通道 3 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 11 **C1IF:** CAN1 (组合的) 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 10 **SPI2IF:** SPI2 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 9 **U2TXIF:** UART2 发送器中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 8 **U2RXIF:** UART2 接收器中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 7 **INT2IF:** 外部中断 2 标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 6 **T5IF:** Timer5 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 5 **T4IF:** Timer4 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 4 **OC4IF:** 输出比较通道 4 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 3 **OC3IF:** 输出比较通道 3 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 2 **IC8IF:** 输入捕捉通道 8 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求

寄存器 6-6: IFS1: 中断标志状态寄存器 1 (续)

- bit 1 IC7IF: 输入捕捉通道 7 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 0 INT1IF: 外部中断 1 标志状态位
1 = 发生中断请求
0 = 未发生中断请求

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 6-7: IFS2: 中断标志状态寄存器 2

高字节:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FLTBF	FLTAIF	LVDIF	DCIIF	QEIIF
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF
bit 7							bit 0

bit 15-13 未用: 读作 0

bit 12 **FLTBF**: 故障 B 输入中断标志状态位

1 = 发生中断请求

0 = 未发生中断请求

bit 11 **FLTAIF**: 故障 A 输入中断标志状态位

1 = 发生中断请求

0 = 未发生中断请求

bit 10 **LVDIF**: 可编程的低电压检测中断标志状态位

1 = 发生中断请求

0 = 未发生中断请求

bit 9 **DCIIF**: 数据转换器接口中断标志状态位

1 = 发生中断请求

0 = 未发生中断请求

bit 8 **QEIIF**: 正交编码器接口中断标志状态位

1 = 发生中断请求

0 = 未发生中断请求

bit 7 **PWMIF**: 电机控制脉宽调制中断标志状态位

1 = 发生中断请求

0 = 未发生中断请求

bit 6 **C2IF**: CAN2 (组合的) 中断标志状态位

1 = 发生中断请求

0 = 未发生中断请求

bit 5 **INT4IF**: 外部中断 4 标志状态位

1 = 发生中断请求

0 = 未发生中断请求

bit 4 **INT3IF**: 外部中断 3 标志状态位

1 = 发生中断请求

0 = 未发生中断请求

bit 3 **OC8IF**: 输出比较通道 8 中断标志状态位

1 = 发生中断请求

0 = 未发生中断请求

bit 2 **OC7F**: 输出比较通道 7 中断标志状态位

1 = 发生中断请求

0 = 未发生中断请求

寄存器 6-7: IFS2: 中断标志状态寄存器 2 (续)

- bit 1 OC6IF: 输出比较通道 6 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求
- bit 0 OC5IF: 输出比较通道 5 中断标志状态位
1 = 发生中断请求
0 = 未发生中断请求

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 6-8: IEC0: 中断允许控制寄存器 0

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE
bit 7							bit 0

- bit 15 **CNIE:** 输入变化通知中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 14 **MI2CIE:** I²C 总线冲突中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 13 **SI2CIE:** I²C 传输结束中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 12 **NVMIE:** 非易失性存储器写完成中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 11 **ADIE:** A/D 转换完成中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 10 **U1TXIE:** UART1 发送器中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 9 **U1RXIE:** UART1 接收器中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 8 **SPI1IE:** SPI1 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 7 **T3IE:** Timer3 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 6 **T2IE:** Timer2 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 5 **OC2IE:** 输出比较通道 2 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 4 **IC2IE:** 输出比较通道 2 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 3 **T1IE:** Timer1 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 2 **OC1IE:** 输出比较通道 1 中断允许位
1 = 允许中断请求
0 = 不允许中断请求

寄存器 6-8: IEC0: 中断允许控制寄存器 0 (续)

- bit 1 IC1IE: 输入捕捉通道 1 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 0 INT0IE: 外部中断 0 允许位
1 = 允许中断请求
0 = 不允许中断请求

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 6-9: IEC1: 中断允许控制寄存器 1

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE
bit 7							bit 0

- bit 15 **IC6IE:** 输入捕捉通道 6 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 14 **IC5IE:** 输入捕捉通道 5 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 13 **IC4IE:** 输入捕捉通道 4 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 12 **IC3IE:** 输入捕捉通道 3 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 11 **C1IE:** CAN1 (组合的) 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 10 **SPI2IE:** SPI2 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 9 **U2TXIE:** UART2 发送器中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 8 **U2RXIE:** UART2 发送器中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 7 **INT2IE:** 外部中断 2 允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 6 **T5IE:** Timer5 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 5 **T4IE:** Timer4 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 4 **OC4IE:** 输出比较通道 4 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 3 **OC3IE:** 输出比较通道 3 中断允许位
1 = 允许中断请求
0 = 不允许中断请求
- bit 2 **IC8IE:** 输入捕捉通道 8 中断允许位
1 = 允许中断请求
0 = 不允许中断请求

寄存器 6-9: IEC1: 中断允许控制寄存器 1 (续)

- bit 1 **IC7IE:** 输入捕捉通道 7 中断允许位
 1 = 允许中断请求
 0 = 不允许中断请求
- bit 0 **INT1IE:** 外部中断 1 允许位
 1 = 允许中断请求
 0 = 不允许中断请求

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 6-10: IEC2: 中断允许控制寄存器 2

高字节:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FLTBIIE	FLTAIE	LVDIE	DCIIE	QEIIIE
bit 15			bit 8				

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWMIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE
bit 7			bit 0				

bit 15-13 未用: 读作 0

bit 12 **FLTBIIE**: 故障 B 输入中断允许位

1 = 允许中断请求
0 = 不允许中断请求

bit 11 **FLTAIE**: 故障 A 中断允许位

1 = 允许中断请求
0 = 不允许中断请求

bit 10 **LVDIE**: 可编程低压检测中断允许位

1 = 允许中断请求
0 = 不允许中断请求

bit 9 **DCIIE**: 数据转换器接口中断允许位

1 = 允许中断请求
0 = 不允许中断请求

bit 8 **QEIIIE**: 正交编码器接口中断允许位

1 = 允许中断请求
0 = 不允许中断请求

bit 7 **PWMIE**: 电机控制脉宽调制中断允许位

1 = 允许中断请求
0 = 不允许中断请求

bit 6 **C2IE**: CAN2 (组合的) 中断允许位

1 = 允许中断请求
0 = 不允许中断请求

bit 5 **INT4IE**: 外部中断 4 允许位

1 = 允许中断请求
0 = 不允许中断请求

bit 4 **INT3IE**: 外部中断 3 允许位

1 = 允许中断请求
0 = 不允许中断请求

bit 3 **OC8IE**: 输出比较通道 8 中断允许位

1 = 允许中断请求
0 = 不允许中断请求

bit 2 **OC7IE**: 输出比较通道 7 中断允许位

1 = 允许中断请求
0 = 不允许中断请求

寄存器 6-10: IEC2: 中断允许控制寄存器 2 (续)

- bit 1 **OC6IE:** 输出比较通道 6 中断允许位
 1 = 允许中断请求
 0 = 不允许中断请求
- bit 0 **OC5IE:** 输出比较通道 5 中断允许位
 1 = 允许中断请求
 0 = 不允许中断请求

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 6-11: **IPC0: 中断优先级控制寄存器 0**

高字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T1IP<2:0>			—	OC1IP<2:0>		
bit 15				bit 8			

低字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	IC1IP<2:0>			—	INT0IP<2:0>		
bit 7				bit 0			

bit 15 **未用:** 读作 0

bit 14-12 **T1IP<2:0>:** Timer1 中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

bit 11 **未用:** 读作 0

bit 10-8 **OC1IP<2:0>:** 输出比较通道 1 中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

bit 7 **未用:** 读作 0

bit 6-4 **IC1IP<2:0>:** 输入捕捉通道 1 中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

bit 3 **未用:** 读作 0

bit 2-0 **INT0IP<2:0>:** 外部中断 0 优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

图注:			
R = 可读位	W = 可写位	U= 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 6-12: IPC1: 中断优先级控制寄存器 1

高字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T3IP<2:0>			—	T2IP<2:0>		
bit 15				bit 8			

低字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	OC2IP<2:0>			—	IC2IP<2:0>		
bit 7				bit 0			

- bit 15 未用: 读作 0
- bit 14-12 **T3IP<2:0>**: Timer3 中断优先级位
111 = 中断优先级为 7 (最高优先级中断)
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 11 未用: 读作 0
- bit 10-8 **T2IP<2:0>**: Timer2 中断优先级位
111 = 中断优先级为 7 (最高优先级中断)
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 7 未用: 读作 0
- bit 6-4 **OC2IP<2:0>**: 输出比较通道 2 中断优先级位
111 = 中断优先级为 7 (最高优先级中断)
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 3 未用: 读作 0
- bit 2-0 **IC2IP<2:0>**: 输入捕捉通道 2 中断优先级位
111 = 中断优先级为 7 (最高优先级中断)
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

图注:			
R = 可读位	W = 可写位	U= 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 6-13: **IPC2: 中断优先级控制寄存器 2**

高字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	ADIP<2:0>			—	U1TXIP<2:0>		
bit 15				bit 8			

低字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	U1RXIP<2:0>			—	SPI1IP<2:0>		
bit 7				bit 0			

- bit 15 **未用:** 读作 0
- bit 14-12 **ADIP<2:0>:** A/D 转换完成中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 11 **未用:** 读作 0
- bit 10-8 **U1TXIP<0>:** UART1 发送器中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 7 **未用:** 读作 0
- bit 6-4 **U1RXIP<0>:** UART1 接收器中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 3 **未用:** 读作 0
- bit 2-0 **SPI1IP<2:0>:** SPI1 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 6-14: **IPC3: 中断优先级控制寄存器 3**

高字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	CNIP<2:0>			—	MI2CIP<2:0>		
bit 15				bit 8			

低字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	SI2CIP<2:0>			—	NVMIP<2:0>		
bit 7				bit 0			

- bit 15 **未用:** 读作 0
- bit 14-12 **CNIP<2:0>:** 输入变化通知中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 11 **未用:** 读作 0
- bit 10-8 **MI2CIP<2:0>:** I²C 总线冲突中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 7 **未用:** 读作 0
- bit 6-4 **SI2CIP<2:0>:** I²C 传输完成中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 3 **未用:** 读作 0
- bit 2-0 **NVMIP<2:0>:** 非易失性存储器写中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 6-15: **IPC4: 中断优先级控制寄存器 4**

高字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	OC3IP<2:0>			—	IC8IP<2:0>		
bit 15				bit 8			

低字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	IC7IP<2:0>			—	INT1IP<2:0>		
bit 7				bit 0			

- bit 15 **未用:** 读作 0
- bit 14-12 **OC3IP<2:0>:** 输出比较通道 3 中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 11 **未用:** 读作 0
- bit 10-8 **IC8IP<2:0>:** 输入捕捉通道 8 中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 7 **未用:** 读作 0
- bit 6-4 **IC7IP<2:0>:** 输入捕捉通道 7 中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 3 **未用:** 读作 0
- bit 2-0 **INT1IP<2:0>:** 外部中断 1 优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

图注:
R = 可读位 W = 可写位 U = 未用位, 读作 0
-n = POR 时的值 1 = 置位 0 = 清零 x = 未知

寄存器 6-16: IPC5: 中断优先级控制寄存器 5

高字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	INT2IP<2:0>			—	T5IP<2:0>		
bit 15				bit 8			

低字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T4IP<2:0>			—	OC4IP<2:0>		
bit 7				bit 0			

- bit 15 未用：读作 0
- bit 14-12 **INT2IP<2:0>**: 外部中断 2 优先级位

111 = 中断优先级为 7（最高优先级中断）

.

.

.

001 = 中断优先级为 1

000 = 中断源被禁止
- bit 11 未用：读作 0
- bit 10-8 **T5IP<2:0>**: Timer5 中断优先级位

111 = 中断优先级为 7（最高优先级中断）

.

.

.

001 = 中断优先级为 1

000 = 中断源被禁止
- bit 7 未用：读作 0
- bit 6-4 **T4IP<2:0>**: Timer4 中断优先级位

111 = 中断优先级为 7（最高优先级中断）

.

.

.

001 = 中断优先级为 1

000 = 中断源被禁止
- bit 3 未用：读作 0
- bit 2-0 **OC4IP<2:0>**: 输出比较通道 4 中断优先级位

111 = 中断优先级为 7（最高优先级中断）

.

.

.

001 = 中断优先级为 1

000 = 中断源被禁止

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 6-17: **IPC6: 中断优先级控制寄存器 6**

高字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	C1IP<2:0>			—	SPI2IP<2:0>		
bit 15				bit 8			

低字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	U2TXIP<2:0>			—	U2RXIP<2:0>		
bit 7				bit 0			

- bit 15 **未用:** 读作 0
- bit 14-12 **C1IP<2:0>:** CAN1 （组合的）中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 11 **未用:** 读作 0
- bit 10-8 **SPI2IP<2:0>:** SPI2 中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 7 **未用:** 读作 0
- bit 6-4 **U2TXIP<2:0>:** UART2 发送器中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 3 **未用:** 读作 0
- bit 2-0 **U2RXIP<2:0>:** UART2 接收器中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 6-18: **IPC7: 中断优先级控制寄存器 7**

高字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	IC6IP<2:0>			—	IC5IP<2:0>		
bit 15				bit 8			

低字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	IC4IP<2:0>			—	IC3IP<2:0>		
bit 7				bit 0			

- bit 15 **未用:** 读作 0
- bit 14-12 **IC6IP<2:0>:** 输入捕捉通道 6 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 11 **未用:** 读作 0
- bit 10-8 **IC5IP<2:0>:** 输入捕捉通道 5 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 7 **未用:** 读作 0
- bit 6-4 **IC4IP<2:0>:** 输入捕捉通道 4 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 3 **未用:** 读作 0
- bit 2-0 **IC3IP<2:0>:** 输入捕捉通道 3 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 6-19: **IPC8: 中断优先级控制寄存器 8**

高字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	OC8IP<2:0>			—	OC7IP<2:0>		
bit 15				bit 8			

低字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	OC6IP<2:0>			—	OC5IP<2:0>		
bit 7				bit 0			

bit 15 **未用:** 读作 0

bit 14-12 **OC8IP<2:0>:** 输出比较通道 8 中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

bit 11 **未用:** 读作 0

bit 10-8 **OC7IP<2:0>:** 输出比较通道 7 中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

bit 7 **未用:** 读作 0

bit 6-4 **OC6IP<2:0>:** 输出比较通道 6 中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

bit 3 **未用:** 读作 0

bit 2-0 **OC5IP<2:0>:** 输出比较通道 5 中断优先级位
111 = 中断优先级为 7 （最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 6-20: IPC9: 中断优先级控制寄存器 9

高字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	PWMIP<2:0>			—	C2IP<2:0>		
bit 15				bit 8			

低字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	INT4IP<2:0>			—	INT3IP<2:0>		
bit 7				bit 0			

- bit 15 未用：读作 0
- bit 14-12 **PWMIP<2:0>**: 电机控制脉宽调制中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 11 未用：读作 0
- bit 10-8 **C2IP<2:0>**: CAN2（组合的）中断优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 7 未用：读作 0
- bit 6-4 **INT4IP<2:0>**: 外部中断 4 优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 3 未用：读作 0
- bit 2-0 **INT3IP<2:0>**: 外部中断 3 优先级位
111 = 中断优先级为 7（最高优先级中断）
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 6-21: IPC10: 中断优先级控制寄存器 10

高字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	FLTAIP<2:0>			—	LVDIP<2:0>		
bit 15				bit 8			

低字节:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	DCIIP<2:0>			—	QEIIP<2:0>		
bit 7				bit 0			

- bit 15 未用: 读作 0
- bit 14-12 **FLTAIP<2:0>**: 故障 A 输入中断优先级位
111 = 中断优先级为 7 (最高优先级中断)
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 11 未用: 读作 0
- bit 10-8 **LVDIP<2:0>**: 可编程低压检测中断优先级位
111 = 中断优先级为 7 (最高优先级中断)
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 7 未用: 读作 0
- bit 6-4 **DCIIP<2:0>**: 数据转换器接口中断优先级位
111 = 中断优先级为 7 (最高优先级中断)
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止
- bit 3 未用: 读作 0
- bit 2-0 **QEIIP<2:0>**: 正交编码器接口中断优先级位
111 = 中断优先级为 7 (最高优先级中断)
.
.
.
001 = 中断优先级为 1
000 = 中断源被禁止

图注:
R = 可读位 W = 可写位 U = 未用位, 读作 0
-n = POR 时的值 1 = 置位 0 = 清零 x = 未知

寄存器 6-22: IPC11: 中断优先级控制寄存器 11

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15				bit 8			

低字节:							
U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0
—	—	—	—	—	FLTBP<2:0>		
bit 7					bit 0		

- bit 15-3 未用: 读作 0
- bit 2-0 **FLTBP<2:0>**: 故障 B 输入中断优先级位
- 111 = 中断优先级为 7 (最高优先级中断)
-
-
-
- 001 = 中断优先级为 1
- 000 = 中断源被禁止

图注:			
R = 可读位	W = 可写位	U= 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

6.5 中断设置流程

6.5.1 初始化

以下步骤说明了如何配置中断源：

1. 如果不需要中断嵌套，将 **NSTDIS** 控制位（**INTCON1<15>**）置 1。
2. 通过写相应的 **IPCx** 控制寄存器中的控制位选择中断源的用户分配优先级。优先级取决于特定的应用和中断源类型。如果不需要多个优先级，所有允许的中断源的 **IPCx** 寄存器控制位均可以编程为同一个非零值。

注： 在器件复位时， IPC 寄存器被初始化，所有用户中断源被分配为优先级 4。

3. 在相关的 **IFSx** 状态寄存器中清零与外设相关的中断标志状态位。
4. 通过在相应的 **IECx** 控制寄存器中置 1 与中断源相关的中断允许控制位，使能中断源。

6.5.2 中断服务程序

用来用正确向量地址声明 **ISR** 和初始化 **IVT** 的方法将由编程语言（即 **C** 或汇编）和用于开发此应用程序的语言开发工具套件决定。一般情况下，用户必须清零在 **ISR** 中处理的中断的中断源在相应 **IFSx** 寄存器中的中断标志。否则，在退出中断服务程序后会立即再次进入 **ISR**。如果 **ISR** 用汇编语言编码，必须用 **RETFIE** 指令终止它以便使保存的 **PC** 值、**SRL** 值和旧的 **CPU** 优先级出栈。

6.5.3 陷阱服务程序

除了必须清零 **INTCON1** 寄存器中的相关陷阱状态标志以避免反复进入 **TSR**，陷阱服务程序（**TSR**）的编码方式类似于 **ISR**。

6.5.4 中断禁止

可以使用以下流程禁止所有用户中断：

1. 使用 **PUSH** 指令将当前 **SR** 值压入软件堆栈。
2. 将值 **0xE0** 和 **SRL** 进行或操作强制 **CPU** 的优先级别为 7。

若要允许用户中断，可使用 **POP** 指令恢复先前的 **SR** 值。

注意只有优先级小于或等于 7 的用户中断可以被禁止。陷阱源（级别 8 — 级别 15）不能被禁止。

DISI 指令提供了一种方便的方法，可以将优先级别 1 — 6 的中断禁止一段固定的时间。**DISI** 指令不能禁止级别为 7 的中断源。

表 6-3: 与中断控制器相关的特殊功能寄存器

SFR 名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位状态
INTCON1	0080	NSTDIS	—	—	—	—	OVATE	OVATE	COVTE	—	—	—	MATHERR	ADDRERR	STKERR	OSCFail	—	0000 0000 0000 0000
INTCON2	0082	ALTIVT	—	—	—	—	—	—	—	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP	0000 0000 0000 0000
IFT0IF	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0	0000 0000 0000 0000
IFS1	0086	IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IFS2	0088	—	—	—	FLTBIIF	FLTAIF	LVDIF	DCIIF	QEIIIF	PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF	0000 0000 0000 0000
IEC0	008C	CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IEC2	0090	—	—	—	FLTBIE	FLTAIE	LVDIE	DCIIE	QEIIIE	PWMIIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			0100 0100 0100 0100
IPC1	0096	—	T31P<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>			0100 0100 0100 0100
IPC2	0098	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			0100 0100 0100 0100
IPC3	009A	—	CNIP<2:0>			—	MI2CIP<2:0>			—	SI2CIP<2:0>			—	NVMIP<2:0>			0100 0100 0100 0100
IPC4	009C	—	OC3IP<2:0>			—	IC8IP<2:0>			—	IC7IP<2:0>			—	INT1IP<2:0>			0100 0100 0100 0100
IPC5	009E	—	INT2IP<2:0>			—	T5IP<2:0>			—	T4IP<2:0>			—	OC4IP<2:0>			0100 0100 0100 0100
IPC6	00A0	—	C1IP<2:0>			—	SPI2IP<2:0>			—	U2TXIP<2:0>			—	U2RXIP<2:0>			0100 0100 0100 0100
IPC7	00A2	—	IC6IP<2:0>			—	IC5IP<2:0>			—	IC4IP<2:0>			—	IC3IP<2:0>			0100 0100 0100 0100
IPC8	00A4	—	OC8IP<2:0>			—	OC7IP<2:0>			—	OC6IP<2:0>			—	OC5IP<2:0>			0100 0100 0100 0100
IPC9	00A6	—	PWMIP<2:0>			—	C2IP<2:0>			—	INT4IP<2:0>			—	INT3IP<2:0>			0100 0100 0100 0100
IPC10	00A8	—	FLTAIP<2:0>			—	LVDIP<2:0>			—	DCIIP<2:0>			—	QEIIIP<2:0>			0100 0100 0100 0100
IPC11	00AA	—	—	—	—	—	—	—	—	—	—	—	—	—	FLTBIIP<2:0>			0000 0000 0000 0100

注：可能不是所有的中断源及其相关的控制位在某个特定器件上都具备。详细信息请参见器件数据手册。

6.6 设计技巧

问 1: *当两个中断源同时等待处理并具有相同的用户分配优先级时会发生什么？*

答: 自然顺序优先级最高的中断源具有优先权。中断源的中断向量表（IVT）地址决定自然顺序优先级。IVT 地址越小的中断源自然顺序优先级越高。

问 2: *是否可以使用 DISI 指令禁止所有中断源和陷阱源？*

答: DISI 指令不能禁止陷阱或优先级为 7 的中断源。但是，如果用户应用中没有允许优先级为 7 的中断源，可以使用 DISI 指令作为一种禁止所有中断源的简便方法。

6.7 相关应用笔记

本节列出了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是专门为 dsPIC30F 产品系列编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与中断模块相关的应用笔记有：

标题	应用笔记编号
目前没有相关的应用笔记。	

注：

如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

6.8 版本历史

版本 A

这是本文档的初始发行版。

版本 B

此版本为 dsPIC30F 中断模块添加了额外的技术内容。

版本 C

此版本加入了本文档更新时已知的所有勘误表。

第 7 章 振荡器

目录

本章包括下列主题：

7.1	简介	7-2
7.2	器件时钟和 MIPS	7-5
7.3	振荡器配置	7-6
7.4	振荡器控制寄存器——OSCCON 和 OSCTUN	7-13
7.5	主振荡器	7-20
7.6	晶体振荡器 / 陶瓷谐振器	7-22
7.7	为晶振、时钟模式、C1、C2 和 Rs 确定最佳的值	7-24
7.8	外部时钟输入	7-25
7.9	外部 RC 振荡器	7-26
7.10	锁相环（PLL）	7-30
7.11	低功耗 32 kHz 晶体振荡器	7-31
7.12	振荡器起振定时器（OST）	7-31
7.13	内部快速 RC 振荡器（FRC）	7-31
7.14	内部低功耗 RC（LPRC）振荡器	7-32
7.15	故障保护时钟监视器（FSCM）	7-32
7.16	可编程振荡器后分频器	7-33
7.17	时钟切换工作原理	7-34
7.18	设计技巧	7-38
7.19	相关应用笔记	7-39
7.20	版本历史	7-40

7.1 简介

本章描述了 dsPIC30F 器件的通用系列、传感器系列和电机控制系列的振荡器系统工作原理。该振荡器系统包含以下模块和功能：

- 可选择多种外部和内部振荡器作为时钟源
- 片上 PLL 可提高内部工作频率
- 不同时钟源之间的时钟切换
- 可节省系统功耗的可编程时钟后分频器
- 故障保护时钟监视器（FSCM）可检测时钟故障并采取故障保护措施
- 器件时钟由特殊功能寄存器和非易失性配置位控制

图 7-1 所示为振荡器系统的简化框图。

7.1.1 振荡器系统功能综述

dsPIC30F 器件的通用系列、传感器系列和电机控制系列具有以下三种版本的振荡器系统之一——版本 1、版本 2 和版本 3。表 7-1 列出了三种版本的振荡器系统的特性。

注： 欲了解您所使用的 dsPIC30F 器件的振荡器系统版本，请参见器件数据手册。

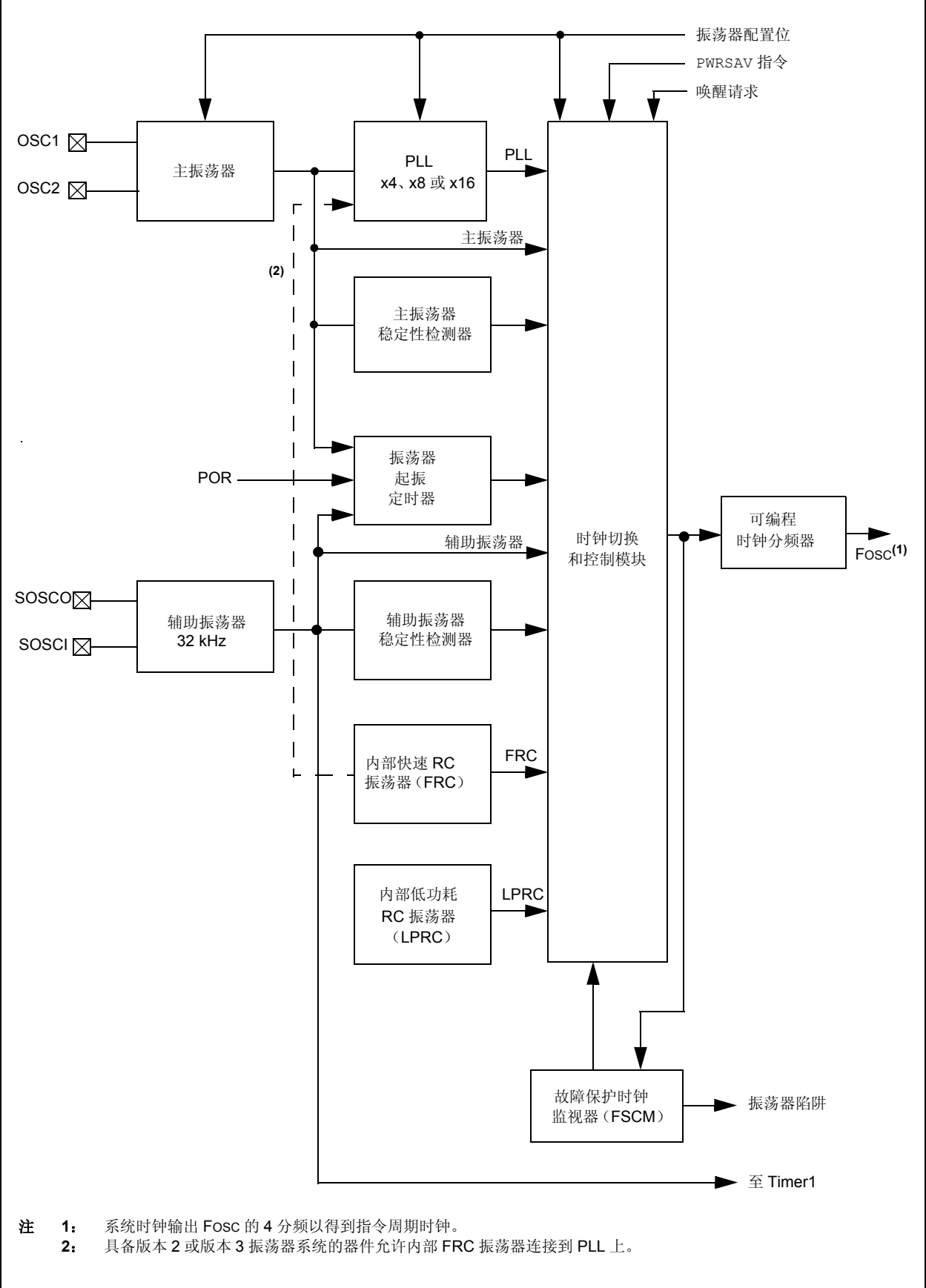
表 7-1： 不同器件振荡器系统特性汇总

振荡器系统	dsPIC30F 器件	特性汇总
版本 1	30F6010, 30F6011, 30F6012, 30F6013, 30F6014	振荡源： • 带多种时钟模式的主振荡器——XT、EC 和 HS • 辅助振荡器（低功耗 32 kHz 晶振） • FRC 振荡器：快速内部 RC（7.37 MHz） • LPRC 振荡器：低功耗内部 RC（512 kHz） PLL 时钟倍频器： • 输入频率范围为 4 MHz-10 MHz • 4 倍频增益模式（F_{OUT} = 16 MHz-40 MHz） • 8 倍频增益模式（F_{OUT} = 32 MHz-80 MHz） • 16 倍频增益模式（F_{OUT} = 64 MHz-120 MHz） • PLL VCO 锁闭指示加“失锁（out of lock）”陷阱选项 • PLL 输入源由以下提供： - XT 或 EC 主振荡器 时钟分频选项： 用于器件时钟的普通后分频器（除以 4、16 或 64） 故障保护时钟监视器（FSCM）： 检测时钟故障并切换到内部 FRC 振荡器
版本 2	30F2010, 30F4011, 30F4012, 30F5011, 30F5013	振荡器系统版本 2 在版本 1 的基础上增加了以下功能： • 内部 FRC 振荡器还可作为 PLL 的输入，加快了执行速度并不再需要外部时钟源（此特性适用于除 30F2010 外的所有器件） • 为内部 FRC 振荡器增加了用户调整功能

表 7-1: 不同器件振荡器系统特性汇总

振荡器系统	dsPIC30F 器件	特性汇总
VERSION 3	30F2011, 30F2012, 30F3010, 30F3011, 30F3012, 30F3013, 30F3014, 30F4013, 30F5015, 30F5016, 30F6010A, 30F6011A, 30F6012A, 30F6013A, 30F6014A, 30F6015	振荡器系统版本 3 在版本 2 的基础上增加了以下功能： • HS 振荡器还作为 PLL 的输入，扩大了晶体频率的选择范围

图 7-1: 振荡器系统框图



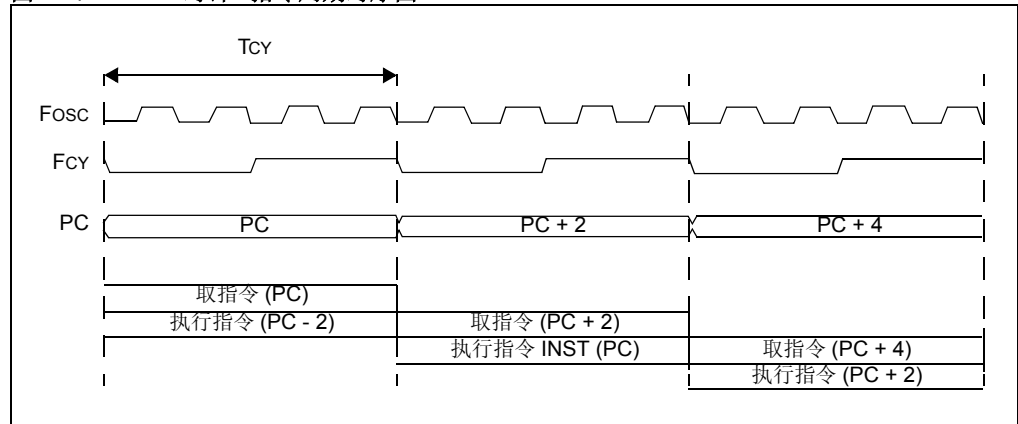
7.2 器件时钟和 MIPS

如图 7-1 所示, 可使用四个时钟源中的一个提供系统时钟。这些时钟源是主振荡器、辅助振荡器、内部快速 RC (FRC) 振荡器或低功耗 RC (LPRC) 振荡器。主振荡器时钟源可选择使用内部 PLL。所选时钟源频率可有选择地通过可编程时钟分频器降低。来自可编程后分频器的输出就是系统时钟源 Fosc。

将系统时钟源四分频可产生内部指令周期时钟 Fcy。在本文档中, 此指令周期时钟由 Fosc/4 表示。图 7-2 中的时序图给出了系统时钟源和指令执行之间的关系。

在主振荡器的某些工作模式下, OSC2 I/O 引脚上可提供内部指令周期时钟 Fcy 的输出 (见第 7.3 节 “振荡器配置”)。

图 7-2: 时钟 / 指令周期时序图



公式 7-1: MIPS 和时钟源振荡器频率的关系

$$F_{CY} = \frac{F_{OSC}}{4} = \left(\frac{\text{时钟源振荡器频率} * \text{PLL 倍频器}}{\text{编程后分频器} * 4} \right)$$

7.3 振荡器配置

在器件发生上电复位事件时使用的振荡源（和工作模式）用非易失性配置位选择。振荡器配置位位于 Fosc 配置寄存器内。

使用 Fosc 非易失性配置寄存器的 FOS 位选择在上电复位时使用的振荡源。默认（未编程的）振荡源是主振荡器。对这些位进行编程可选择使用辅助振荡器或内部振荡器中的一个。

使用 Fosc 非易失性配置寄存器的 FPR 位选择主振荡器的工作模式。

dsPIC30F 器件的通用系列、传感器系列和电机控制系列有三种版本的振荡器系统。Fosc 非易失性配置寄存器的定义根据版本而有所不同，如以下子章节所述。

7.3.1 振荡器系统版本 1 配置

具备振荡器系统版本 1 的器件，其 Fosc 非易失性配置寄存器如寄存器 7-1 所示。可通过 FPR 位选择对应的工作模式，如表 7-2 所示。

7.3.2 振荡器系统版本 2 配置

具备振荡器系统版本 2 的器件，其 Fosc 非易失性配置寄存器如寄存器 7-2 所示。可通过 FPR 位选择对应的工作模式，如表 7-3 所示。

7.3.3 振荡器系统版本 3 配置

具备振荡器系统版本 3 的器件，其 Fosc 非易失性配置寄存器如寄存器 7-3 所示。可通过 FPR 位选择对应的工作模式，如表 7-4 所示。

7.3.4 时钟切换模式配置位

FCKSM<1:0> 配置位（Fosc<15:14>）用于使能 / 禁止器件时钟切换和故障保护时钟监视器（FSCM）。当这些位未编程（默认）时，禁止时钟切换和 FSCM。在所有振荡器版本中，这些位的定义和功能都相同。

寄存器 7-1: FOSC: 振荡器系统版本 1 的振荡器配置寄存器

高字节:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23				bit 16			

中间字节:							
R/P	R/P	U	U	U	U	R/P	R/P
FCKSM<1:0>		—	—	—	—	FOS<1:0>	
bit 15				bit 8			

低字节:							
U	U	U	U	R/P	R/P	R/P	R/P
—	—	—	—	FPR<3:0>			
bit 7				bit 0			

- bit 23-16 未用: 读为 0
- bit 15-14 FCKSM<1:0>: 时钟切换模式选择熔丝位

1x = 禁止时钟切换, 禁止故障保护时钟监视器

01 = 使能时钟切换, 禁止故障保护时钟监视器

00 = 使能时钟切换, 使能故障保护时钟监视器
- bit 13-10 未用: 读为 0
- bit 9-8 FOS<1:0>: POR 时振荡源选择位

11 = 主振荡器 (主振荡器模式通过 FPR<3:0> 选择)

10 = 内部低功耗 RC 振荡器

01 = 内部快速 RC 振荡器

00 = 低功耗 32 kHz 振荡器 (Timer1 振荡器)
- bit 7-4 未用: 读为 0
- bit 3-0 FPR<3:0>: 主振荡器组的振荡器模式选择位, 请参见表 7-2

图注:		
R = 可读位	P = 可编程位	U = 未用位

表 7-2: 振荡器系统版本 1: 用于时钟选择的配置位值

振荡器模式	振荡器源	FOS<1:0>	FPR<3:0>	OSC2 引脚功能
带 16x PLL 的 EC	主振荡器	1 1	1 1 1 1	I/O (注 4)
带 8x PLL 的 EC	主振荡器	1 1	1 1 1 0	I/O
带 4x PLL 的 ECIO	主振荡器	1 1	1 1 0 0	I/O
EC	主振荡器	1 1	1 0 1 1	Fosc/4
保留	主振荡器	1 1	1 0 1 0	n/a
ERC	主振荡器	1 1	1 0 0 1	Fosc/4
ERCIO	主振荡器	1 1	1 0 0 0	I/O
带 16x PLL 的 XT	主振荡器	1 1	0 1 1 1	(注 3)
带 8x PLL 的 XT	主振荡器	1 1	0 1 1 0	(注 3)
带 4x PLL 的 XT	主振荡器	1 1	0 1 0 1	(注 3)
XT	主振荡器	1 1	0 1 0 0	(注 3)
HS	主振荡器	1 1	0 0 1 x	(注 3)
XTL	主振荡器	1 1	0 0 0 x	(注 3)
LP	辅助振荡器	0 0	— — — —	(注 1, 2)
FRC	内部振荡器	0 1	— — — —	(注 1, 2)
LPRC	内部振荡器	1 0	— — — —	(注 1, 2)

- 注 1: OSC2 引脚功能由主振荡器模式的配置决定 (FPR<3:0> 配置位)。
- 2: 注意, 即使始终选择辅助振荡器或内部时钟源, OSC1 引脚仍不能用作 I/O 引脚。
- 3: 在这些振荡器模式中, 在 OSC1 和 OSC2 引脚之间连接有一个晶振。
- 4: 对于未编程 (已擦除) 器件, 这是默认振荡器模式。未编程配置位的值为 “1”。
- 5: XTL – XTL 晶振模式 (200 kHz-4 MHz 晶体)
- 6: XT – XT 晶振模式 (4 MHz-10 MHz 晶体)
- 7: HS – HS 晶振模式 (10 MHz-25 MHz 晶体)

寄存器 7-2: FOSC: 振荡器系统版本 2 的振荡器配置寄存器

高字节:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23				bit 16			

中间字节:							
R/P	R/P	U	U	U	U	R/P	R/P
FCKSM<1:0>		—	—	—	—	FOS<1:0>	
bit 15				bit 8			

低字节:							
U	U	U	U	R/P	R/P	R/P	R/P
—	—	—	—	FPR<3:0>			
bit 7				bit 0			

- bit 23-16 未用: 读为 0
- bit 15-14 **FCKSM<1:0>**: 时钟切换模式选择熔丝位
1x = 禁止时钟切换, 禁止故障保护时钟监视器
01 = 使能时钟切换, 禁止故障保护时钟监视器
00 = 使能时钟切换, 使能故障保护时钟监视器
- bit 13-10 未用: 读为 0
- bit 9-8 **FOS<1:0>**: POR 时振荡源选择位
11 = 主振荡器 (主振荡器模式通过 FPR<3:0> 选择)
10 = 内部低功耗 RC 振荡器
01 = 内部快速 RC 振荡器
00 = 低功耗 32 kHz 振荡器 (Timer1 振荡器)
- bit 7-4 未用: 读为 0
- bit 3-0 **FPR<3:0>**: 主振荡器组的振荡器模式选择位, 请参见表 7-3

图注:		
R = 可读位	P = 可编程位	U = 未用位

表 7-3: 振荡器系统版本 2: 用于时钟选择的配置位值

振荡器模式	振荡器源	FOS<1:0>		FPR<3:0>				OSC2 引脚功能
EC	主振荡器	1	1	1	0	1	1	CLKO
ECIO	主振荡器	1	1	1	1	0	0	I/O
带 4x PLL 的 EC	主振荡器	1	1	1	1	0	1	I/O
带 8x PLL 的 EC	主振荡器	1	1	1	1	1	0	I/O
带 16x PLL 的 EC	主振荡器	1	1	1	1	1	1	I/O (注 4)
ERC	主振荡器	1	1	1	0	0	1	CLKO
ERCIO	主振荡器	1	1	1	0	0	0	I/O
XT	主振荡器	1	1	0	1	0	0	(注 3)
带 4x PLL 的 XT	主振荡器	1	1	0	1	0	1	(注 3)
带 8x PLL 的 XT	主振荡器	1	1	0	1	1	0	(注 3)
带 16x PLL 的 XT	主振荡器	1	1	0	1	1	1	(注 3)
XTL	主振荡器	1	1	0	0	0	0	(注 3)
HS	主振荡器	1	1	0	0	1	0	(注 3)
带 4x PLL 的 FRC	主振荡器	1	1	0	0	0	1	I/O
带 8x PLL 的 FRC	主振荡器	1	1	1	0	1	0	I/O
带 16x PLL 的 FRC	主振荡器	1	1	0	0	1	1	I/O
LP	辅助振荡器	0	0	—	—	—	—	(注 1, 2)
FRC	内部 FRC	0	1	—	—	—	—	(注 1, 2)
LPRC	内部 LPRC	1	0	—	—	—	—	(注 1, 2)

- 注 1: OSC2 引脚功能由主振荡器模式选择位 (FPR<3:0>) 决定。
- 2: 注意, OSC1 引脚不能用作 I/O 引脚, 即使始终选择了辅助振荡器或内部时钟源。
- 3: 在这些振荡模式下, 在 OSC1 和 OSC2 引脚之间连接有晶体。
- 4: 这是未编程 (擦除后) 器件的默认振荡模式。未编程配置位的值为 “1”。
- 5: XTL – XTL 晶振模式 (200 kHz-4 MHz 晶体)
- 6: XT – XT 晶振模式 (4 MHz-10 MHz 晶体)
- 7: HS – HS 晶振模式 (10 MHz-25 MHz 晶体)

寄存器 7-3: FOSC: 振荡器系统版本 3 的振荡器配置寄存器

高字节:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23				bit 16			

中间字节:							
R/P	R/P	U	U	U	R/P	R/P	R/P
FCKSM<1:0>		—	—	—	FOS<2:0>		
bit 15				bit 8			

低字节:							
U	U	U	R/P	R/P	R/P	R/P	R/P
—	—	—	FPR<4:0>				
bit 7			bit 0				

bit 23-16 未用: 读为 0

bit 15-14 **FCKSM<1:0>**: 时钟切换和监视器选择配置位

1x = 禁止时钟切换, 禁止故障保护时钟监视器

01 = 使能时钟切换, 禁止故障保护时钟监视器

00 = 使能时钟切换, 使能故障保护时钟监视器

bit 13-10 未用: 读为 0

bit 9-8 **FOS<2:0>**: POR 时振荡器组选择位

111 = PLL 振荡器; PLL 源通过 FPR<4:0> 位选择

011 = EXT: 外部振荡器; OSC1/OSC2 引脚; 外部振荡器配置通过 FPR<4:0> 位选择

010 = LPRC: 内部低功耗 RC

001 = FRC: 内部快速 RC

000 = LPOSC: 低功耗晶振; SOSCI/SOSCO 引脚

bit 7-4 未用: 读为 0

bit 3-0 **FPR<4:0>**: 主振荡器组的振荡器模式选择位, 请参见表 7-4。

图注:

R = 可读位

P = 可编程位

U = 未用位

表 7-4: 振荡器系统版本 3: 用于时钟选择的配置位值

振荡器模式	振荡器源	FOS<2:0>			FPR<4:0>					OSC2 引脚功能
带 4x PLL 的 ECIO	PLL	1	1	1	0	1	1	0	1	I/O
带 8x PLL 的 ECIO	PLL	1	1	1	0	1	1	1	0	I/O
带 16x PLL 的 ECIO	PLL	1	1	1	0	1	1	1	1	I/O
带 4x PLL 的 FRC	PLL	1	1	1	0	0	0	0	1	I/O
带 8x PLL 的 FRC	PLL	1	1	1	0	1	0	1	0	I/O
带 16x PLL 的 FRC	PLL	1	1	1	0	0	0	1	1	I/O
带 4x PLL 的 XT	PLL	1	1	1	0	0	1	0	1	(注 3)
带 8x PLL 的 XT	PLL	1	1	1	0	0	1	1	0	(注 3)
带 16x PLL 的 XT	PLL	1	1	1	0	0	1	1	1	(注 3)
带 4x PLL 的 HS2	PLL	1	1	1	1	0	0	0	1	(注 3)
带 8x PLL 的 HS2	PLL	1	1	1	1	0	0	1	0	(注 3)
带 16x PLL 的 HS2	PLL	1	1	1	1	0	0	1	1	(注 3)
带 4x PLL 的 HS3	PLL	1	1	1	1	0	1	0	1	(注 3)
带 8x PLL 的 HS3	PLL	1	1	1	1	0	1	1	0	(注 3)
带 16x PLL 的 HS3	PLL	1	1	1	1	0	1	1	1	(注 3)
ECIO	外部	0	1	1	0	1	1	0	0	I/O
XT	外部	0	1	1	0	0	1	0	0	(注 3)
HS	外部	0	1	1	0	0	0	1	0	(注 3)
EC	外部	0	1	1	0	1	0	1	1	CLKOUT
ERC	外部	0	1	1	0	1	0	0	1	CLKOUT
ERCIO	外部	0	1	1	0	1	0	0	0	I/O
XTL	外部	0	1	1	0	0	0	0	0	(注 3)
LP	辅助振荡器	0	0	0	x	x	x	x	x	(注 1, 2)
FRC	内部 FRC	0	0	1	x	x	x	x	x	(注 1, 2)
LPRC	内部 LPRC	0	1	0	x	x	x	x	x	(注 1, 2)

- 注 1: OSC2 引脚功能由 (FPR<4:0>) 决定。
- 2: OSC1 引脚不能用作 I/O 引脚, 即使始终选择了辅助振荡器或内部时钟源。
- 3: 在这些振荡模式下, 在 OSC1 和 OSC2 引脚之间连接有晶体。
- 4: XTL – XTL 晶振模式 (200 kHz-4 MHz 晶体)
- 5: XT – XT 晶振模式 (4 MHz-10 MHz 晶体)
- 6: HS – HS 晶振模式 (10 MHz-25 MHz 晶体)

7.4 振荡器控制寄存器——OSCCON 和 OSCTUN

振荡器系统的运行时控制和状态通过特殊功能寄存器提供给用户。表 7-5 汇总了振荡器系统版本 1、版本 2 和版本 3 所提供的运行时控制特性。欲了解您所使用的 dsPIC30F 器件的振荡器系统版本，请参见器件数据手册。

表 7-5: 振荡器控制 SFR

振荡器系统	振荡器控制 SFR 特性汇总
版本 1	通过 OSCCON SFR 进行控制。请参见寄存器 7-4。
版本 2	通过 OSCCON SFR 进行控制。用户可通过 OSCCON 的 TUN<3:0> 位调整 FRC 振荡器。请参见寄存器 7-5。
版本 3	通过 OSCCON 和 OSCTUN SFR 进行控制。用户可通过 OSCTUN 的 TUN<3:0> 位调整 FRC 振荡器。请参见寄存器 7-6 和寄存器 7-7。

OSCCON 控制寄存器提供了时钟切换控制和时钟源状态信息。

OSCCON 的 COSC 状态位均为只读位，用于指示器件赖以工作的振荡源。上电复位时，COSC 位置为 FOS 配置位的值，在时钟切换操作结束时将改变，以指示新的振荡源。

OSCCON 的 NOSC 状态位为控制位，用于选择时钟切换操作中的新时钟源。上电复位时或欠压复位时，NOSC 位置为 FOS 配置位的值，在时钟切换操作时由用户软件修改。

POST<1:0> 控制位（OSCCON<8:7>）控制系统时钟分频比。

LOCK 状态位（OSCCON<5>）为只读位，指示 PLL 电路的状态。

CF 状态位（OSCCON<3>）是可读写状态位，指示时钟故障。

LPOSCEN 控制位（OSCCON<1>）用于使能或禁止 32 kHz 低功耗晶振。

OSWEN 控制位（OSCCON<0>）用于启动时钟切换操作。时钟切换成功后 OSWEN 位自动清零。

TUN<3:0> 位供用户将内部 FRC 振荡器调整为高于或低于 7.37 MHz 的标称值。

注： OSCCON 寄存器被写保护，因为它控制器件的时钟切换机制。对 OSCCON 的写操作指令的说明请参见第 7.4.1 节“防止对 OSCCON 的误写操作”。

7.4.1 防止对 OSCCON 的误写操作

对 OSCCON 寄存器的写操作被有意复杂化了，因为它控制时钟的切换和分频。

要写入 OSCCON 的低字节，必须执行以下代码序列并不得在其中插入任何指令：

```
Byte Write 0x46 to OSCCONL
Byte Write 0x57 to OSCCONL
```

在此序列后，允许对 OSCCONL 进行一个指令周期的写入。此时可写入所需值或使用位操作指令。

要写入 OSCCON 的高字节，必须执行以下代码序列并不得在其中插入任何指令：

```
Byte Write 0x78 to OSCCONH
Byte Write 0x9A to OSCCONH
```

在此序列后，允许对 OSCCONH 进行一个指令周期的写入。此时可写入所需值或使用位操作指令。

dsPIC30F 系列参考手册

寄存器 7-4: OSCCON: 振荡器控制寄存器——振荡器系统版本 1

高字节:							
U-0	U-0	R-y	R-y	U-0	U-0	R/W-y	R/W-y
—	—	COSC<1:0>		—	—	NOSC<1:0>	
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R-0	U-0	R/W-0	U-0	R/W-0	R/W-0
POST<1:0>		LOCK	—	CF	—	LPOSCEN	OSWEN
bit 7						bit 0	

- bit 15-14 **未用:** 读为 0
- bit 13-12 **COSC<1:0>:** 电流振荡器源状态位
 11 = 主振荡器
 10 = 内部 LPRC 振荡器
 01 = 内部 FRC 振荡器
 00 = 低功耗 32 kHz 晶振 (Timer1)
- bit 11-10 **未用:** 读为 0
- bit 9-8 **NOSC<1:0>:** 新振荡器组选择位
 11 = 主振荡器
 10 = 内部 LPRC 振荡器
 01 = 内部 FRC 振荡器
 00 = 低功耗 32 kHz 晶振 (Timer1)
- bit 7-6 **POST<1:0>:** 振荡器后分频器选择位
 11 = 振荡器后分频器将时钟 64 分频
 10 = 振荡器后分频器将时钟 16 分频
 01 = 振荡器后分频器将时钟 4 分频
 00 = 振荡器后分频器不将时钟分频
- bit 5 **LOCK:** PLL 锁定状态位
 1 = 指示 PLL 锁定
 0 = 指示 PLL 失锁 (或被禁止)
 POR 或 BOR 时复位
 启动有效时钟切换时复位
 PLL 启动后达到 PLL 锁定时置 1
 失锁时复位
 PLL 不被选择为系统时钟时读为零
- bit 4 **未用:** 读为 0
- bit 3 **CF:** 时钟故障状态位
 1 = FSCM 检测到时钟故障
 0 = FSCM 未检测到时钟故障
 POR 或 BOR 时复位
 启动有效时钟切换时复位
 检测到时钟故障时置 1
- bit 2 **未用:** 读为 0
- bit 1 **LPOSCEN:** 32 kHz LP 振荡器使能位
 1 = 使能 LP 振荡器
 0 = 禁止 LP 振荡器
 POR 或 BOR 时复位
- bit 0 **OSWEN:** 振荡器切换使能位
 1 = 请求切换到 NOSC<1:0> 位指定的振荡器
 0 = 振荡器切换完成

图注:

R = 可读位	W = 可写位	U = 未用位, 读为 0
-n = POR 时的值	1 = 置 1	0 = 清零
y = POR 或 BOR 时通过配置位设置的值	x = 未知	

寄存器 7-5: OSCCON: 振荡器控制寄存器——振荡器系统版本 2

高字节:							
R/W-0	R/W-0	R-y	R-y	R/W-0	R/W-0	R/W-y	R/W-y
TUN3	TUN2	COSC<1:0>		TUN1	TUN0	NOSC<1:0>	
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R-0	U-0	R/W-0	U-0	R/W-0	R/W-0
POST<1:0>		LOCK	—	CF	—	LPOSCEN	OSWEN
bit 7							bit 0

bit 15-14 **TUN<3:2>**: TUN 位字段的高 2 位。详情请参见 TUN<1:0> (OSCCON<11:10>) 的说明。

bit 13-12 **COSC<1:0>**: 当前振荡器源状态位

11 = 主振荡器
10 = 内部 LPRC 振荡器
01 = 内部 FRC 振荡器
00 = 低功耗 32 kHz 晶振 (Timer1)

bit 11-10 **TUN<1:0>**: TUN 位字段的低 2 位

TTUN<3:0>指定的4位字段使用户可对标称频率为7.37 MHz的内部快速RC振荡器进行调整。例如, 用户可对 FRC 振荡器在 $\pm 12\%$ (960 kHz) 范围内以步长 1.5% 对出厂校准频率进行调整, 如下:

TUN<3:0> = 0111 提供最高频率

.....

TUN<3:0> = 0000 提供出厂校准频率

.....

TUN<3:0> = 1000 提供最低频率

注: 要了解您所使用器件上 FRC 振荡器的精确调整范围和调整步长, 请参见具体器件的数据手册。

bit 9-8 **NOSC<1:0>**: 新振荡器组选择位

11 = 主振荡器
10 = 内部 LPRC 振荡器
01 = 内部 FRC 振荡器
00 = 低功耗 32 kHz 晶振 (Timer1)

bit 7-6 **POST<1:0>**: 振荡器后分频器选择位

11 = 振荡器后分频器将时钟 64 分频
10 = 振荡器后分频器将时钟 16 分频
01 = 振荡器后分频器将时钟 4 分频
00 = 振荡器后分频器不将时钟分频

bit 5 **LOCK**: PLL 锁定状态位

1 = 指示 PLL 锁定
0 = 指示 PLL 失锁 (或被禁止)
POR 或 BOR 时复位
启动有效时钟切换时复位
PLL 启动后达到 PLL 锁定时置 1
失锁时复位
PLL 不被选择为系统时钟时读为零

bit 4 **未用**: 读为 0

bit 3 **CF**: 时钟故障状态位

1 = FSCM 检测到时钟故障
0 = FSCM 未检测到时钟故障
POR 或 BOR 时复位
启动有效时钟切换时复位
检测到故障时置 1

bit 2 **未用**: 读为 0

OOSCCON: 振荡器控制寄存器——振荡器系统版本 2（续）

- bit 1

LPOSCEN: 32 kHz LP 振荡器使能位
1 = 使能 LP 振荡器
0 = 禁止 LP 振荡器
POR 或 BOR 时复位
- bit 0

OSWEN: 振荡器切换使能位
1 = 请求切换到 NOSC<1:0> 位指定的振荡器
0 = 振荡器切换完成

图注:			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知
y = POR 或 BOR 时通过配置位设置的值			

寄存器 7-6: OSCCON: 振荡器控制寄存器——振荡器系统版本 3

高字节:							
U-0	R-y	R-y	R-y	U-0	R/W-y	R/W-y	R/W-y
—	COSC<2:0>			—	NOSC<2:0>		
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R-0	U-0	R/W-0	U-0	R/W-0	R/W-0
POST<1:0>		LOCK	—	CF	—	LPOSCEN	OSWEN
bit 7				bit 0			

bit 15 未用: 读为 0

bit 14-12 **COSC<2:0>**: 电流振荡器组选择位 (只读)

111 = PLL 振荡器; PLL 源通过 FPR<4:0> 位选择

011 = 外部振荡器; OSC1/OSC2 引脚; 外部振荡器配置通过 FPR<4:0> 位选择

010 = LPRC 内部低功耗 RC

001 = FRC 内部快速 RC

000 = LP 晶振; SOSC1/SOSCO 引脚

POR 或 BOR 时置为 FOS<2:0> 值

时钟切换成功时装入 NOSC<2:0>

FSCM 检测到故障并将时钟切换为 FRC 时置为 FRC 值

bit 11 未用: 读为 0

bit 10-8 **NOSC<2:0>**: 新振荡器组选择位

111 = PLL 振荡器; PLL 源通过 FPR<4:0> 位选择

011 = 外部振荡器; OSC1/OSC2 引脚; 外部振荡器配置通过 FPR<4:0> 位选择

010 = LPRC 内部低功耗 RC

001 = FRC 内部快速 RC

000 = LP 晶振; SOSC1/SOSCO 引脚

POR 或 BOR 时置为 FOS<2:0> 值

bit 7-6 **POST<1:0>**: 振荡器后分频器选择位

11 = 振荡器后分频器将时钟 64 分频

10 = 振荡器后分频器将时钟 16 分频

01 = 振荡器后分频器将时钟 4 分频

00 = 振荡器后分频器不将时钟分频

bit 5 **LOCK**: PLL 锁定状态位 (只读)

1 = 指示 PLL 锁定

0 = 指示 PLL 失锁 (被禁止)

POR 或 BOR 时复位

启动有效时钟切换时复位

PLL 启动后达到 PLL 锁定时置 1

失锁时复位

PLL 不被选择为系统时钟时读为零

bit 4 未用: 读为 0

OSCCON: 振荡器控制寄存器——振荡器系统版本 3（续）

- bit 3 **CF:** 时钟故障检测位（可由应用读取 / 清零）
 1 = FSCM 检测到时钟故障
 0 = FSCM 未检测到时钟故障
 POR 或 BOR 时复位
 启动有效时钟切换时复位
 检测到时钟故障时置 1
- bit 2 **未用:** 读为 0
- bit 1 **LPOSCEN:** 32 kHz 辅助（LP）振荡器使能位
 1 = 使能辅助振荡器
 0 = 禁止辅助振荡器
 POR 或 BOR 时复位
- bit 0 **OSWEN:** 振荡器切换使能位
 1 = 请求振荡器切换到由 NOSCG<2:0> 位指定的选择
 0 = 振荡器切换完成
 POR 或 BOR 时复位
 时钟切换成功时复位
 冗余时钟切换（即请求将时钟切换为当前振荡器）后复位
 FSCM 切换到（第 1 组）FRC 振荡器后复位

图注:			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知
y = POR 或 BOR 时通过配置位设置的值			

寄存器 7-7: OSCTUN: FRC 振荡器调整寄存器——仅限振荡器系统版本 3

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15				bit 8			

低字节:							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	TUN<3:0>			
bit 7				bit 0			

bit 15-4 未用：读为 0

bit 3-0 **TUN<3:0>**: TUN<3:0> 指定的 4 位字段使用户可对标称频率为 7.37 MHz 的内部快速 RC 振荡器进行调整。
TUN<3:0> = 0111 提供最高频率
.....
TUN<3:0> = 0000 提供出厂校准频率
.....
TUN<3:0> = 1000 提供最低频率

注 1: 要了解您所使用器件上 FRC 振荡器的精确调整范围和调整步长，请参见具体器件的数据手册。
2: 有些器件可能有超过 4 个 TUN 位。请参见具体器件的数据手册以确定可供用户调整 FRC 振荡器的 TUN 位的个数。

图注:			
R = 可读位	W = 可写位	U = 未用位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知
y = POR 或 BOR 时通过配置位设置的值			

7.5 主振荡器

主振荡器连接在 dsPIC30F 系列器件的 OSC1 和 OSC2 引脚上。表 7-6 总结了主振荡器的各种工作模式。一般来说，可以将主振荡器配置为外部时钟输入、外部 RC 网络或外部晶振模式。在以后各节中将描述主振荡器工作模式的详细信息。

Fosc 非易失性配置寄存器的 FPR 位选择主振荡器的工作模式。

表 7-6: 主振荡器工作模式

振荡器模式 ⁽⁴⁾	说明
XTL	OSC1:OSC2 上的 200 kHz-4 MHz 晶体
XT	OSC1:OSC2 上的 4 MHz-10 MHz 晶体
带 4x PLL 的 XT	OSC1:OSC2 上的 4 MHz-10 MHz 晶体，4x PLL 使能
带 8x PLL 的 XT	OSC1:OSC2 上的 4 MHz-10 MHz 晶体，8x PLL 使能
带 16x PLL 的 XT	OSC1:OSC2 上的 4 MHz-10 MHz 晶体，16x PLL 使能 ⁽¹⁾
LP	SOSCO:SOSCI 上的 32 kHz 晶体 ⁽²⁾
HS	10 MHz-25 MHz 晶体
带 4x PLL 的 HS/2	10 MHz-25 MHz 晶体，2 分频，4x PLL 使能
带 8x PLL 的 HS/2	10 MHz-25 MHz 晶体，2 分频，8x PLL 使能
带 16x PLL 的 HS/2	10 MHz-25 MHz 晶体，2 分频，16x PLL 使能 ⁽¹⁾
带 4x PLL 的 HS/3	10 MHz-25 MHz 晶体，3 分频，4x PLL 使能
带 8x PLL 的 HS/3	10 MHz-25 MHz 晶体，3 分频，8x PLL 使能
带 16x PLL 的 HS/3	10 MHz-25 MHz 晶体，3 分频，16x PLL 使能 ⁽¹⁾
EC	外部时钟输入 (0-40 MHz)
ECIO	外部时钟输入 (0-40 MHz)，OSC2 引脚为 I/O
带 4x PLL 的 EC	外部时钟输入 (4-10 MHz)，OSC2 引脚为 I/O，4x PLL 使能 ⁽¹⁾
带 8x PLL 的	外部时钟输入 (4-10 MHz)，OSC2 引脚为 I/O，8x PLL 使能 ⁽¹⁾
带 16x PLL 的 EC	外部时钟输入 (4-10 MHz)，OSC2 引脚为 I/O，16x PLL 使能 ⁽¹⁾
ERC	外部时钟输入，OSC2 引脚为 Fosc/4 输出 ⁽³⁾
ERCIO	外部时钟输入，OSC2 引脚为 I/O ⁽³⁾
FRC	7.37 MHz 内部快速 RC 振荡器
带 4x PLL 的 FRC	7.37 MHz 内部快速 RC 振荡器，4x PLL 使能
带 8x PLL 的 FRC	7.37 MHz 内部快速 RC 振荡器，8x PLL 使能
带 16x PLL 的 FRC	7.37 MHz 内部快速 RC 振荡器，16x PLL 使能
LPRC	512 kHz 内部快速 RC 振荡器

- 注**
- 1: 必须满足 dsPIC30F 的最大工作频率 120 MHz。
 - 2: LP 振荡器可方便地共用为系统时钟，以及 Timer1 的实时时钟。
 - 3: 需要外部 RC。工作频率最高为 4 MHz。
 - 4: 本表所列工作模式为振荡器系统版本 1、版本 2 和版本 3 的所有模式。

7.5.1 振荡器模式选择指导方针

XT、XTL 和 HS 模式的主要区别在于振荡器电路内部反相器的增益不同，这就让它们有了不同的工作频率范围。通常情况下，应该在符合规范的前提下选择上述增益最小的振荡器模式。因为此时振荡器电路的动态电流 (I_{DD}) 比较小。每个振荡模式的频率范围的上下限为建议的截止频率值，但只要经过了彻底的确认（电压、温度和元件差异，如电阻、电容和内部振荡电路），也可接受不同的增益模式选择。

在所有 EC 和 ECIO 模式中禁止振荡器反馈电路。OSC1 引脚为高阻抗输入，可以由 CMOS 驱动器驱动。

ERC 和 ERCIO 模式为器件振荡提供成本最低的解决方案（只需要一个外部电阻和电容）。这些模式提供的振荡频率也是差异最大的。

如果将主振荡器配置为外部时钟输入或外部 RC 网络模式，就不需要 OSC2 引脚支持振荡器功能。对于这些模式，OSC2 引脚可以被用作额外的器件 I/O 引脚或时钟输出引脚。当 OSC2 引脚用作时钟输出引脚时，输出频率为 $F_{osc}/4$ 。

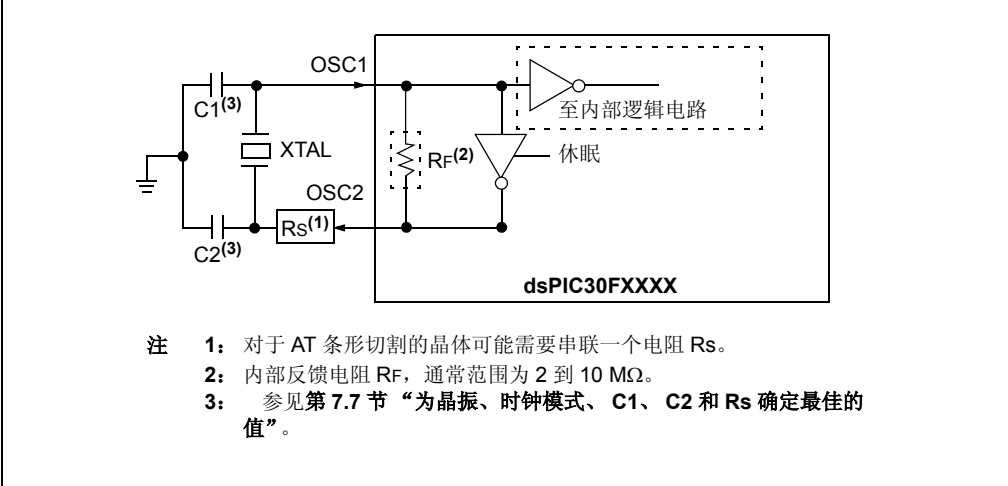
XTL 模式是低功耗 / 低频率模式。这种模式在三种晶振模式中功耗是最小的。XT 模式是中等功耗 / 中等频率模式，而 HS 模式是振荡频率最高的晶振模式。

使用 PLL 电路的 EC 和 XT 模式能提供最高的器件工作频率。由于 PLL 被使能以对振荡器频率进行倍频，因此振荡器电路消耗的电流最多。

7.6 晶体振荡器 / 陶瓷谐振器

在 XT、XTL 和 HS 模式下，OSC1 和 OSC2 引脚连接一个晶振或陶瓷谐振器以建立振荡（图 7-3）。dsPIC30F 振荡器的设计要求使用平行切割的晶体。采用顺序切割的晶体所产生的频率将超出晶体制造厂商给出的范围。

图 7-3: 晶振或陶瓷谐振器的工作原理（XT、XTL 或 HS 振荡器模式）



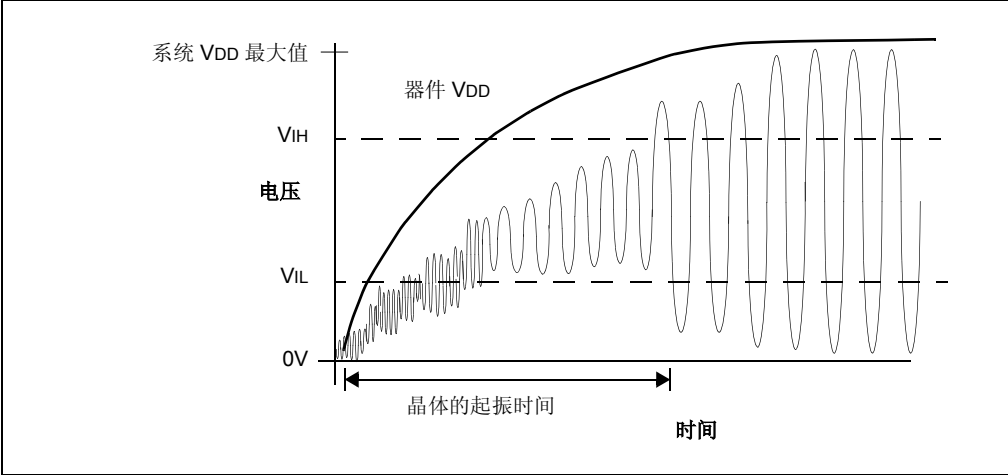
7.6.1 振荡器 / 谐振器起振

随着器件电压从 V_{SS} 上升，振荡器开始振荡。振荡器起振所需的时间由很多因素决定。包括：

- 晶振 / 谐振器频率
- 使用的电容器值（图 7-3 中的 C1 和 C2）
- 器件 V_{DD} 上升时间
- 系统温度
- 串联电阻的阻值和类型（如果使用的话）（图 7-3 中的 R_s ）
- 器件的振荡器模式选择（选择内部振荡器反相器的增益）
- 晶体质量
- 振荡器电路布局
- 系统噪声

图 7-4 给出了典型的振荡器 / 谐振器起振的特性图。

图 7-4: 振荡器 / 谐振器起振特性曲线示例



7.6.2 调整振荡器电路

由于 Microchip 器件具有宽工作范围（频率、电压和温度；取决于订购的具体器件和版本）而且外部元件（晶体和电容等）的质量和厂商不同，因此需要进行工作验证，以保证元件的选择符合应用的要求。

在选择和安装这些外部元件时，需要考虑多种因素。这些因素包括：

- 放大器增益
- 所需频率
- 晶体谐振频率
- 工作温度
- 供电电压范围
- 起振时间
- 稳定性
- 晶体寿命
- 功耗
- 简化电路
- 标准元件的使用
- 元件数量

7.6.3 振荡器从休眠模式起振

振荡器从休眠状态唤醒时是最难起振的。这是因为两个负载电容都被充电到某个静态值，而相位差在唤醒时又最小。因此需要更多的时间来获得稳定振荡。同时也不要忘记低电压、高温和低频时钟模式也限制了环路增益，这反过来又会影响到起振。下面的每一种因素都会延长起振时间：

- 低频设计（低增益时钟模式）
- 低噪声环境（如电池驱动的器件）
- 在屏蔽的盒内工作（在嘈杂的 RF 环境之外）
- 低电压
- 高温
- 从休眠模式唤醒

噪声实际上有助于缩短振荡器的起振时间，因为它为振荡器提供加速启动功能。

进入休眠模式前，可将应用切换到内部 FRC（+PLL）振荡器以缩短器件从休眠唤醒的时间。

7.7 为晶振、时钟模式、C1、C2 和 Rs 确定最佳的值

选择元件的最佳方法是利用相关的知识进行大量的试验、测量和测试。

通常晶振的选择只依据它们的并联谐振频率，但是其他的参数也可能对您的设计很重要，如温度或频率容差。应用笔记 AN588 “PIC[®] Microcontroller Oscillator Design Guide” 是了解更多晶振工作和它们的订购信息的绝佳参考。

dsPIC30F 内部振荡器电路是并联振荡器电路，需要选择并联谐振晶振。负载电容通常规定在 22 pF 到 33 pF 范围内。负载电容在这个范围内，晶体的振荡频率最接近所需的频率。正如后面将要描述的那样，为了优化其他方面的性能，有时稍微更改一下负载电容值也是很有必要的。

时钟模式主要是根据需要的晶体振荡器频率来选择的。XT、XTL 和 HS 振荡器模式的主要区别在于振荡电路内部反相器的增益不同，从而使频率范围也不同。一般来说，应该在符合规范的前提下选择上述增益最小的振荡器模式。因为此时电路的动态电流（IDD）较小。每个振荡器模式的频率范围的上下限为建议的截止频率，但只要经过了彻底的确认（电压、温度和元件差异，如电阻、电容和内部振荡电路），也可以接受不同的增益模式选择。

最开始应该根据晶振生产厂商和器件数据手册提供的列表中所建议的负载电容，C1 和 C2（参见图 7-3）也应在初期选定。由于晶振生产厂商、供电电压和上述其他因素可能导致电路与出厂前确定特性参数过程中所使用的电路不尽相同，因此数据手册中提供的值只能作为一个初始参考。

理想状况下，电容应该选择可以让振荡器在电路期望工作的最高温度和最低 VDD 条件下产生振荡。高温和低 VDD 都会对环路增益产生限制，所以如果电路能工作在这些极端情况下，设计者更能确信振荡器在其他温度和供电电压下，都能正常工作。在最高增益情况（最高 VDD 电压和最低工作温度）下，输出正弦波应不被限幅，而在最低增益情况（最低 VDD 电压和最高工作温度）下，其幅度应该足够大，以满足器件数据手册上所列的时钟信号的逻辑输入要求。

一种缩短起振时间的方法是将 C2 的值选得比 C1 大。这使得在上电时，通过晶体的振荡信号产生较大的相移，以加速振荡器起振。

这两个电容除了辅助晶振产生适当的频率响应之外，增加它们的电容值还能降低环路增益。可以选择 C2 来影响回路的总增益。晶体振荡器过驱。增大 C2 可以降低增益（也可参见有关 Rs 的讨论）。如果电容值过大，会储存和释放过量的电流通过晶体，所以 C1 和 C2 不能过大。然而，测量晶体的功耗（瓦特数）非常难，但如果您所选择的电容值并未偏离建议值太远，就不必担心这个问题。

如果其他外部元件都选好之后，发现晶振仍然过驱动，此时可在电路中接入一个串联电阻 Rs。晶振是否过驱可通过示波器观察 OSC2 引脚（驱动引脚）来判断。将示波器探头连接到 OSC1 引脚，会使引脚负载过重，对系统性能产生不利的影响。注意，示波器探头会将其自身的负载电容加到被测电路中，因此在设计时应考虑到此问题（即，如果一个电路在 C2 为 22 pF 时工作状态最佳，示波器探头负载电容为 10 pF，当探头接入引脚时，实际上连接到该引脚的电容变为了 33 pF）。振荡输出信号不应出现限幅或畸变。过驱动晶振则可能导致电路振荡频率跃变到一个高次谐波上，甚至会损坏晶体。

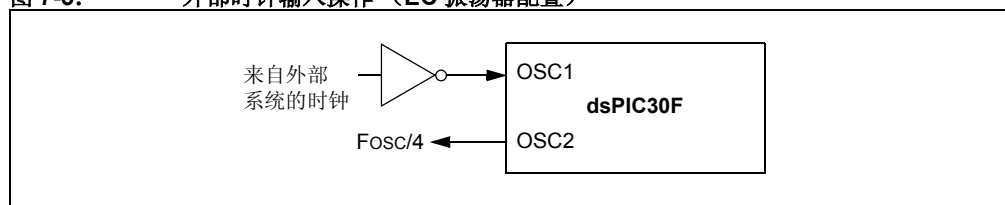
OSC2 引脚的信号应为一个平滑的正弦波，可轻松跨越时钟输入引脚的最大和最小电平值（VDD 为 5V 时，时钟输入引脚的峰值为 4V 到 5V，通常即可满足要求）。设置电路能达到上述要求的一个简单办法依然是在设计预期使用环境的最低温度和最高 VDD 条件下测试电路，然后观察输出。此时，时钟输出的振幅应该最大。如果在靠近 VDD 和 VSS 处正弦波出现限幅或失真，而增加负载电容又可能使流过晶体的电流过大或使电容值远离厂商的负载规范值。要调整晶振电流，可以在晶振反相器输出引脚和 C2 之间添加一个可调电位器，然后调节它直至正弦波平滑。晶振在低温和 VDD 上限时驱动电流最大。应该在这些极限条件下调整可调电位器的值以防止过驱。然后用一个最接近标准阻值的串联电阻 Rs 代替可调电位器。如果 Rs 太高，超过 20 k Ω ，输入与输出的隔离度将过大，使时钟更易受到噪声的影响。如果您发现只有这么高的阻值才能防止晶振过驱，尝试增加 C2 的值进行补偿或更改振荡器工作模式。尝试获取这样的组合，Rs 在 10k 左右或小于 10k 且电容离厂商规范值不远。

7.8 外部时钟输入

使用外部时钟的主振荡模式有两种，分别是 EC 和 ECIO。

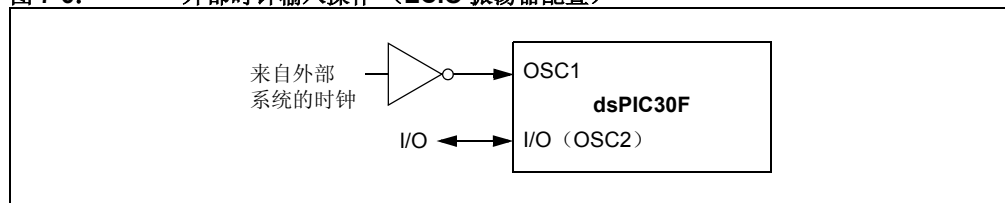
在 EC 模式下（图 7-5），OSC1 引脚可以由 CMOS 驱动器驱动。在此模式下，OSC1 引脚为高阻态且 OSC2 引脚为时钟输出（Fosc/4）。此输出时钟对于测试和同步非常有用。

图 7-5: 外部时钟输入操作（EC 振荡器配置）



在 ECIO 模式下（图 7-6）OSC1 引脚可以由 CMOS 驱动器驱动。在此模式下，OSC1 引脚为高阻态且 OSC2 引脚为通用 I/O 引脚。OSC1 和 OSC2 之间的反馈器件被关闭以节省电流。

图 7-6: 外部时钟输入操作（ECIO 振荡器配置）



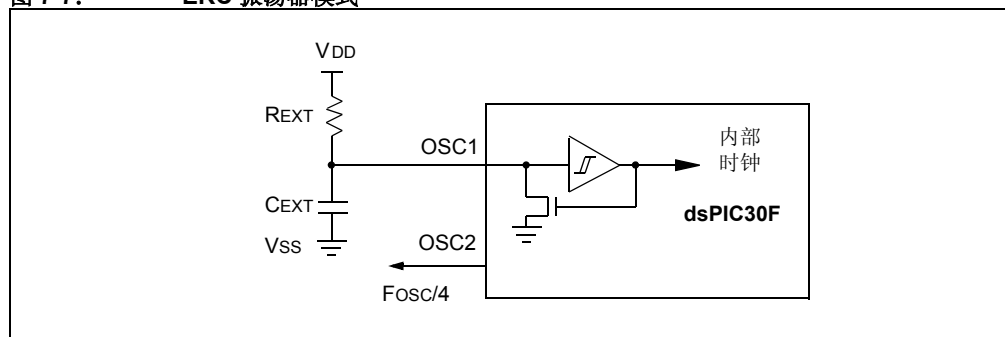
7.9 外部 RC 振荡器

对于时序要求不高的应用，主振荡器的 ERC 和 ERCIO 模式更能节约成本。RC 振荡器的频率是以下几个值的函数：

- 供电电压
- 外部电阻（R_{EXT}）值
- 外部电容（C_{EXT}）值
- 工作温度

除此之外，由于正常制造工艺参数的差异，每个器件的振荡器频率都不相同。而不同封装类型的引线电容间的差异也会影响振荡频率，特别当 C_{EXT} 值较小时更是如此。用户还需要考虑由于所使用的外部 R_{EXT} 和 C_{EXT} 元件容差造成的频率变化。图 7-7 显示了 R/C 组合的连接方式。对于 R_{EXT} 值低于 2.2 k Ω 的情况，振荡器工作可能变得不稳定或完全停止。对于 R_{EXT} 值很高的情况（如 1 M Ω ），振荡器易受噪声、湿度和漏电流的干扰。因此，建议所使用的 R_{EXT} 值在 3 k Ω 和 100 k Ω 之间。

图 7-7: ERC 振荡器模式



尽管在没有外部电容（C_{EXT} = 0 pF）的情况下振荡器仍可工作，但考虑到噪声和稳定性等因素，我们仍建议使用一个大于 20 pF 的电容。在没有外部电容或外部电容很小的情况下，由于其他外部电容的变化（如 PCB 上的布线电容和封装引线间的电容），振荡频率会发生很大的变化。

OSC2/CLKO 引脚上输出信号频率为振荡器频率的 4 分频，该信号可用作测试或同步其他逻辑电路。

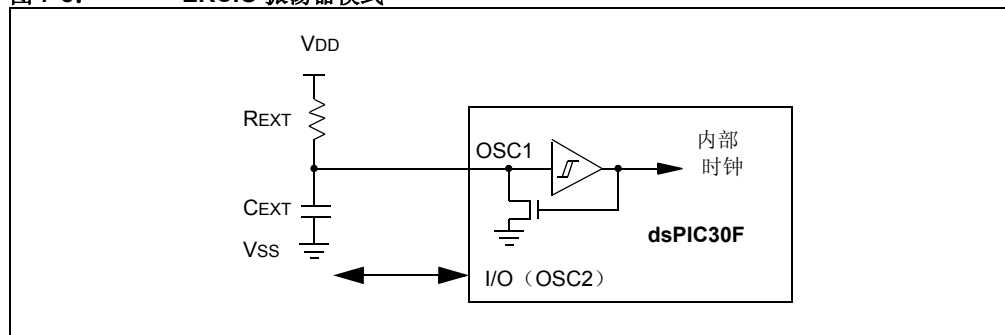
注： 当振荡器配置为 ERC 或 ERCIO 模式时，OSC1 引脚上不应该连接外部时钟。

7.9.1 I/O 使能的外部 RC 振荡器

ERCIO 振荡器模式的工作方式与 ERC 振荡器模式极其相似。惟一的区别就是 OSC2 引脚被配置为 I/O 引脚。

在 RC 模式中，用户需要考虑到由所使用的外部 REXT 和 CEXT 元件的容差、制造工艺的不同、电压和温度所引起的时钟频率差异。图 7-8 显示了带有 I/O 引脚的 RC 组合的连接方式。

图 7-8: ERCIO 振荡器模式



7.9.2 外部 RC 起振

RC 振荡器不会产生起振延时。给 VDD 上电时振荡器就起振。

注： 在器件开始执行代码之前，用户应该验证 VDD 是否在规范之内。

7.9.3 RC 工作频率

以下各图给出了在选定各 RC 组件值时，外部 RC 振荡器频率随器件电压变化的曲线。

注： 以下各图仅作为 RC 组件选择的粗略指南使用。实际频率根据系统温度和器件有所不同。更多 RC 振荡器特性数据请参见特定器件数据手册。

图 7-9: 典型外部 RC 振荡频率与 VDD 的关系曲线 (Cext = 20 pF)

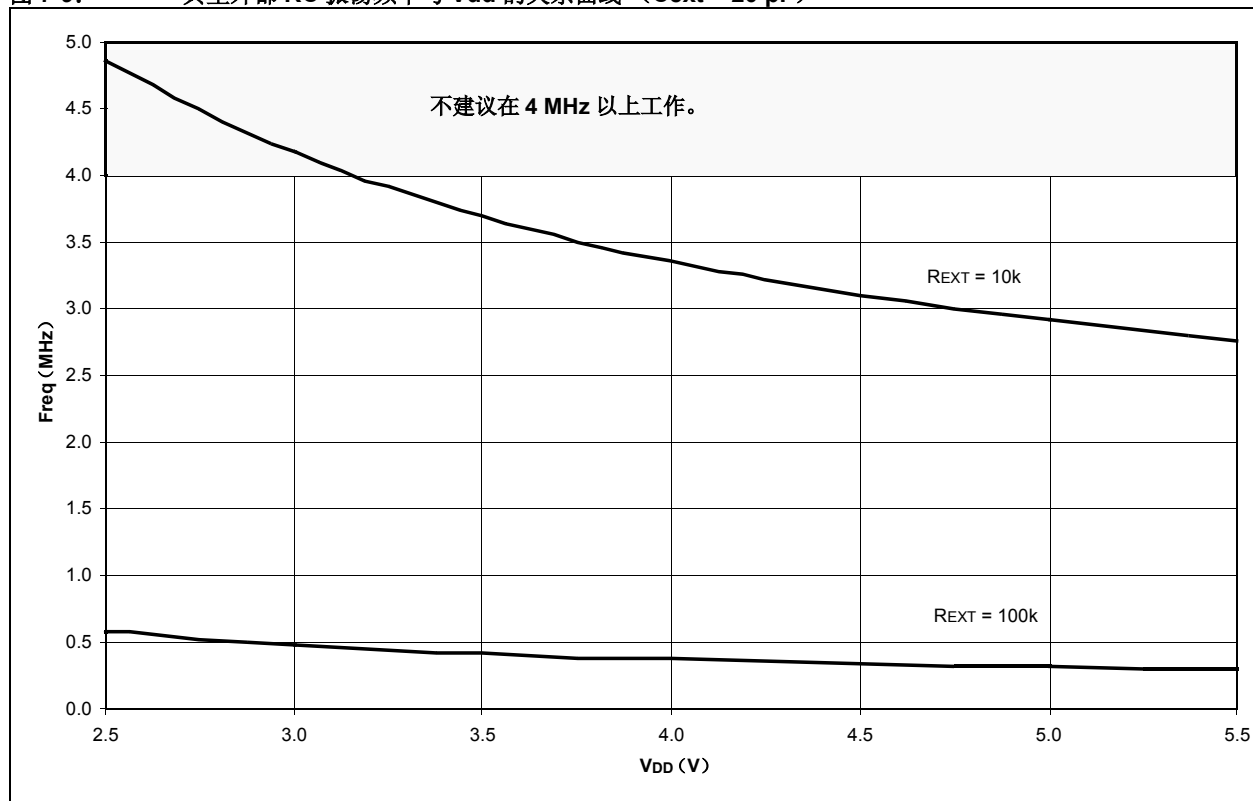


图 7-10: 典型外部 RC 振荡频率与 VDD 的关系曲线 (Cext = 100 pF)

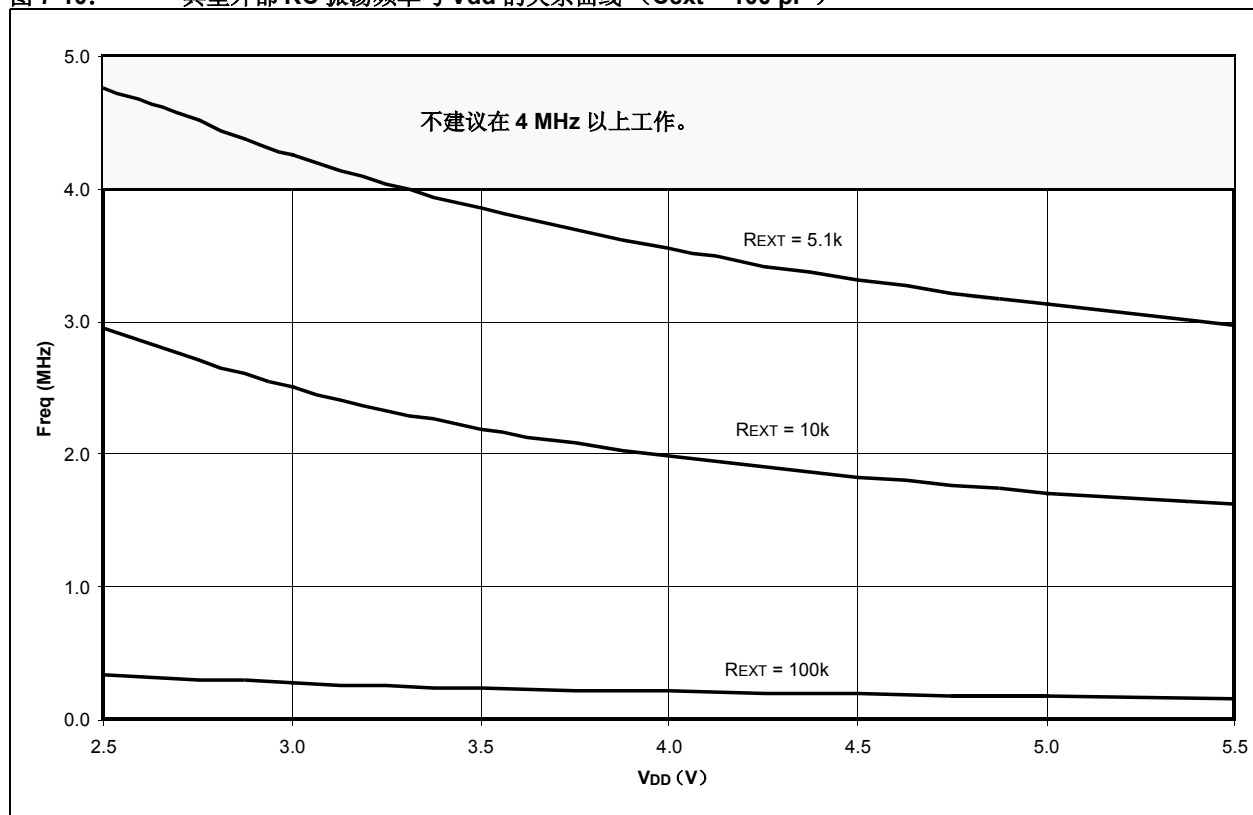
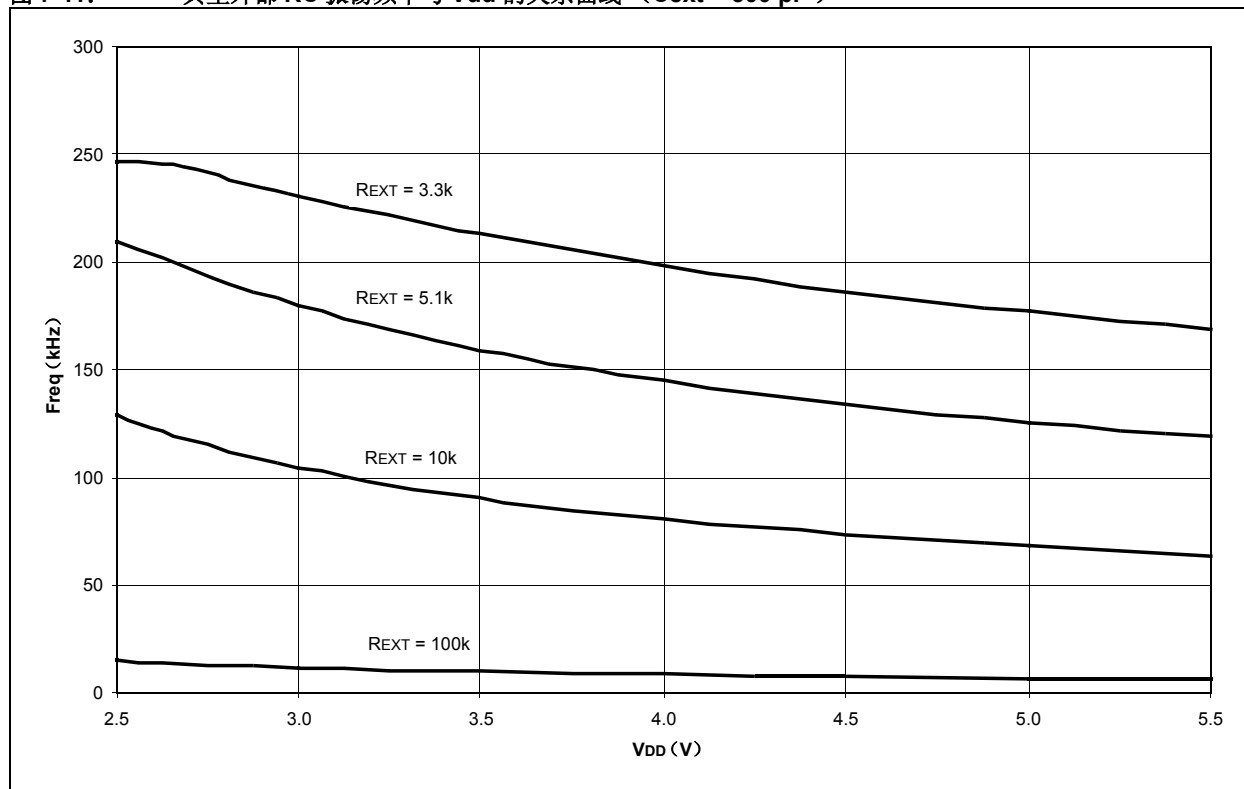


图 7-11: 典型外部 RC 振荡频率与 Vdd 的关系曲线 (Cext = 300 pF)



7.10 锁相环（PLL）

使用 FPR<3:0> 振荡器配置位可以将锁相环使能为 4 倍频、8 倍频或 16 倍频工作模式。表 7-7 总结了每个工作模式的输入和输出频率范围。

注： 有时 PLL 输出频率范围会超出 dsPIC30F 器件的最大工作频率。详细信息请参见特定器件数据手册的“电气规范”一章。

表 7-7: PLL 频率范围

F _{IN}	PLL 倍频器	F _{OUT}
4 MHz — 10 MHz	4 倍频	16 MHz — 40 MHz
4 MHz — 10 MHz	8 倍频	32 MHz — 80 MHz
4 MHz — 7.5 MHz	16 倍频	64 MHz — 120 MHz

7.10.1 PLL 锁定状态

PLL 电路可以检测到 PLL 进入锁相状态的时刻。同样，也可以检测到 PLL 失锁的时刻。PLL 达到锁定状态的延时为 T_{LOCK}。T_{LOCK} 的标称值为 20 μs。更多信息请参见特定器件数据手册的“电气规范”部分。

LOCK 位是只读状态位（OSCCON<5>），反映 PLL 的锁定状态。LOCK 位在上电复位时清零。

7.10.1.1 在时钟切换的过程中 PLL 失锁

当在时钟切换操作（包括上电复位）中选择 PLL 作为目标时钟源时，LOCK 位清零。在达到锁相状态后 LOCK 位置 1。如果 PLL 锁定失败，时钟切换电路就不会切换到 PLL 输出作为系统时钟，而是使用旧的时钟源继续运行。

7.10.1.2 在上电复位过程中失去 PLL 锁定

如果 PLL 在上电复位（POR）时锁定失败且使能了故障保护时钟监视器（FSCM），FRC 振荡器将成为器件时钟源并产生一个时钟故障陷阱。

7.10.1.3 在器件正常工作过程中失去 PLL 锁定

如果在正常工作过程中 PLL 至少在 4 个时钟周期中失去锁定，那么 LOCK 位被清零，表示 PLL 失锁。此外，还会产生一个时钟故障陷阱。在这种情况下，处理器继续使用 PLL 时钟源运行。如果需要，用户可以在陷阱服务程序中切换到另一个时钟源。

注： 更多有关振荡器故障陷阱的详细信息，请参见第 6 章“复位中断”。

注： 在器件正常工作期间，PLL 失锁会产生时钟故障陷阱，但系统时钟源不会改变。检测 PLL 是否失锁不必使能 FSCM。

7.11 低功耗 32 kHz 晶体振荡器

LP 和辅助振荡器是特别为使用 32 kHz 晶振的低功耗运行而设计的。LP 振荡器位于器件的 SOSC0 和 SOSC1 引脚上，用作低功耗运行的辅助晶振时钟源。LP 振荡器还可驱动 Timer1 作为实时时钟应用。

7.11.1 LP 振荡器使能

以下控制位影响 LP 振荡器的工作：

1. OSCCON 寄存器 (OSCCON<13:12>) 中的 COSC<1:0> 位。
2. OSCCON 寄存器 (OSCCON<1>) 中的 LPOSCEN 位。

当 LP 振荡器使能时，SOSC0 和 SOSC1 I/O 引脚由振荡器控制且不能用作其他 I/O 功能。

7.11.1.1 LP 振荡器的连续工作

如果置位 LPOSCEN 控制位 (OSCCON<1>)，将始终使能 LP 振荡器。有两种方式可保持 LP 振荡器运行。第一，保持 LP 振荡器总是开启可以快速切换到 32 kHz 系统时钟进行低功耗运行。如果主振荡器是晶振型时钟源，返回快速主振荡也需要振荡器起振时间（参见第 7.12 节“振荡器起振定时器 (OST)”）。第二，在使用 Timer1 作为实时时钟源时，振荡器应该总是保持在开启状态。

7.11.1.2 LP 振荡器间断工作

当 LPOSCEN 控制位 (OSCCON<1>) 被清零时，LP 振荡器只在被选为当前器件时钟源时才能运行 (COSC<1:0> = 00)。如果 LP 振荡器是当前器件的时钟源而该器件进入休眠模式时，LP 振荡器将会禁止。

7.11.2 将 LP 振荡器用于 Timer1 的操作

LP 振荡器可以在实时时钟应用中用作 Timer1 的时钟源。更多详细信息请参见第 12 章“定时器”。

7.12 振荡器起振定时器 (OST)

为了确保晶体振荡器（或陶瓷谐振器）已经起振并稳定，必须提供振荡器起振定时器。它是一个简单的 10 位计数器，在计数 1024 个 T_{osc} 周期后再将振荡器时钟提供给系统使用。超时溢出周期用 T_{OST} 表示。振荡器引脚上输出的振荡器信号的振幅必须达到 V_{IL} 和 V_{IH} 阈值，才能使 OST 开始对周期进行计数。

每次振荡器必须重启时（即在 POR、BOR 或从休眠模式唤醒时）都会有一个 T_{OST} 时间。振荡器起振定时器适用于 LP 振荡器和主振荡器的 XT、XTL 和 HS 模式。

7.13 内部快速 RC 振荡器 (FRC)

FRC 振荡器是快速（标称值为 7.37 MHz）内部 RC 振荡器。该振荡器用于在不使用外部晶体、陶瓷谐振器或 RC 网络的情况下，提供器件工作的频率范围。具备振荡器系统版本 2 或 3 的器件也可选择将 FRC 振荡器作为 PLL 的输入频率。

7.14 内部低功耗 RC (LPRC) 振荡器

LPRC 振荡器是看门狗定时器 (WDT) 的一个组件, 振荡频率标称值为 512 kHz。LPRC 振荡器是上电延时定时器 (PWRT) 电路、WDT 和时钟监视器电路的时钟源。它也可用于为功耗要求严格而对时序的精度要求不高的应用提供低频率时钟源选项。

注: LPRC 振荡器的振荡频率根据器件电压和工作温度的不同而有所不同。详细信息请参见特定器件数据手册的“电气规范”部分。

7.14.1 使能 LPRC 振荡器

因为 LPRC 振荡器是 PWRT 的时钟源, 所以它总是在上电复位时使能。在 PWRT 超时后, 如果以下条件之一为真, 则 LPRC 振荡器将保持为开启状态:

- 故障保护时钟监视器使能。
- WDT 使能。
- 选择 LPRC 作为系统时钟 ($COSC<1:0> = 10$)。

如果以上条件都不为真, 则 LPRC 将在 PWRT 超时后关闭。

7.15 故障保护时钟监视器 (FSCM)

故障保护时钟监视器 (FSCM) 允许器件在即便发生外部时钟故障的情况下继续运行。可以在 FOSC 器件配置寄存器中编程 FCKSM 位 (时钟切换和监视器位) 使能 FSCM 功能。更多详细信息请参见第 7.3 节“振荡器配置”。如果使能了 FSCM 功能, LPRC 内部振荡器将始终保持运行 (休眠模式除外)。

在发生振荡器故障时, FSCM 会产生时钟故障陷阱并将系统时钟切换到 FRC 振荡器。之后用户即可选择尝试重新启动振荡器或执行控制关闭操作。

在切换到 FRC 振荡器时, FSCM 模块将进行以下操作:

1. $COSC<1:0>$ 位装入“01”。
2. CF 位置 1 表示时钟发生故障。
3. OSWEN 控制位清零取消所有待处理的时钟切换。

注: 更多有关振荡器故障陷阱的信息, 请参见第 6 章“复位中断”。

7.15.1 FSCM 延时

在 POR、BOR 或从休眠模式唤醒时, 在 FSCM 开始监控系统时钟源之前会插入一个标称值为 100 μ s 的延时 (T_{FSCM})。FSCM 延时是为了在未使用上电延时定时器 (PWRT) 的情况下为振荡器和 / 或 PLL 提供时间以使其稳定。FSCM 延时在内部系统复位信号 SYSRST 发出后产生。FSCM 延时时序信息请参见第 8 章“复位”。

当使能 FSCM 并选择以下任一器件系统时钟源作为系统时钟时, 即产生 FSCM 延时 T_{FSCM} :

- EC+PLL
- XT+PLL
- XT
- HS
- HS/2 或 HS/3 + PLL
- XTL
- LP

注: T_{FSCM} 的规范值请参见器件数据手册的“电气规范”部分。

7.15.2 FSCM 和缓慢振荡器起振

如果选择的器件振荡器在退出 POR、BOR 或休眠模式后起振缓慢，那么在振荡器起振之前 FSCM 延时可能就超时了。在这种情况下，FSCM 会启动时钟故障陷阱。在它发生时，COSC<1:0> 位（OSCCON<13:12>）将装载，FRC 振荡器选择位。这将有效关闭尝试起振的原振荡器。用户可以检测到这种情况并在陷阱服务程序中将时钟切换回需要的振荡器。

7.15.3 FSCM 和 WDT

发生时钟故障时，WDT 不受影响，并将继续在 LPRC 时钟上运行。

7.16 可编程振荡器后分频器

后分频器允许用户通过降低 CPU 和外设使用的时钟频率降低功耗。任何时候都可以通过 POST<1:0> 控制位（OSCCON<7:6>）更改后分频值。

为了确保时钟的平滑过渡，在时钟改变之前会有一段延时。时钟后分频器将保持时钟选择多路开关不变直到出现 64 分频输出的下降沿。因而切换延时最高可达 64 个系统时钟周期取决于 POST<1:0> 控制位何时被写入。图 7-13 所示为改变三种后分频值情况下的后分频器的工作。

图 7-12: 可编程振荡器后分频器

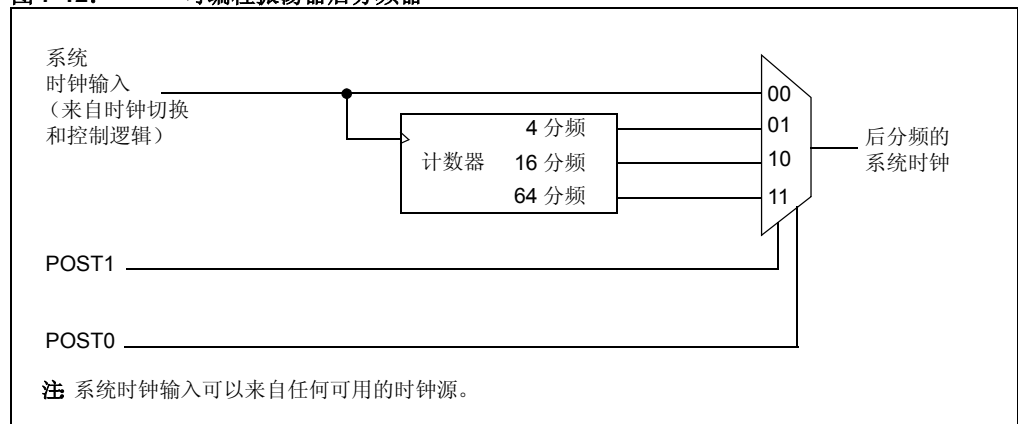
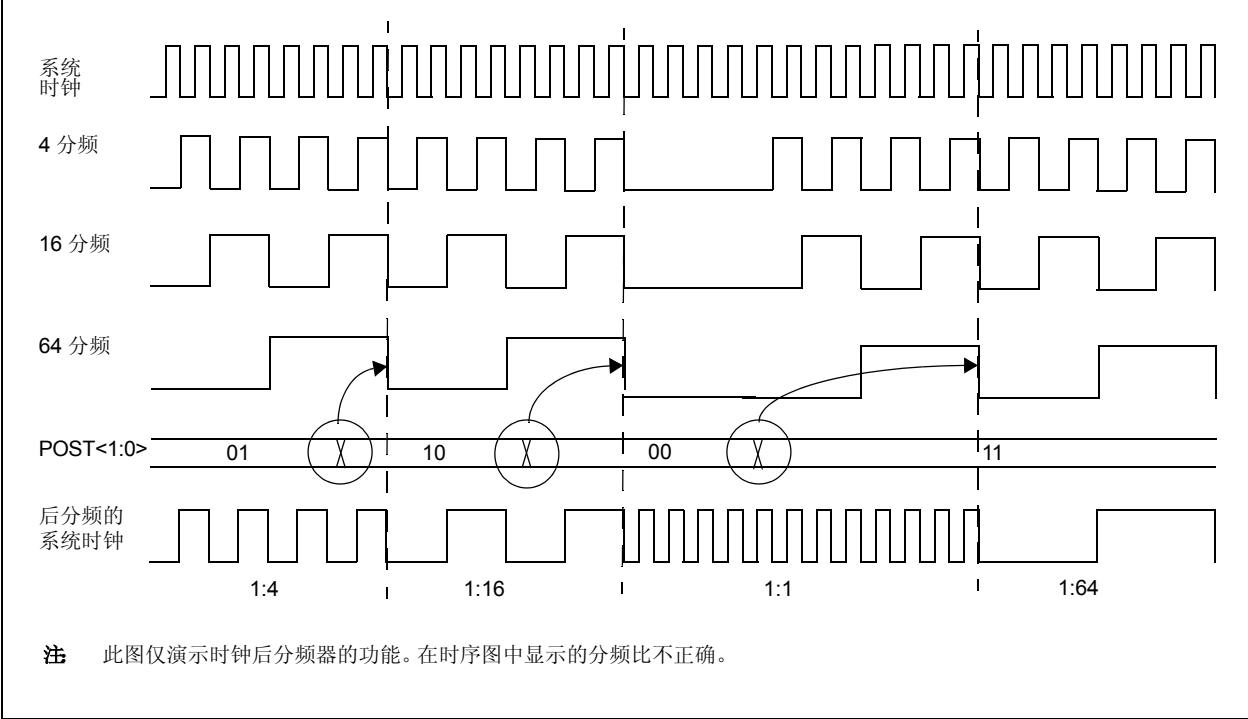


图 7-13: 后分频器更新时序



7.17 时钟切换工作原理

在器件操作过程中为时钟切换选择的可用时钟源如下：

- OSC1/OSC2 引脚上的主振荡器
- SOSCO/SOSCI 引脚上的低功耗 32 kHz 晶体（辅助）振荡器
- 内部快速 RC（FRC）振荡器
- 内部低功耗 RC（LPRC）振荡器

注： 主振荡器有多种工作模式（EC、RC、XT 和 FRC 等）。FOSC 器件配置寄存器中的 FPR 配置位决定主振荡器的工作模式（详情请参见第 7.3 节“振荡器配置”）。

7.17.1 时钟切换使能

要使能时钟切换，必须将 Fosc 配置寄存器中的 FCKSM1 配置位编程为“0”（详情请参见第 7.3 节“振荡器配置”）。

如果 FCKSM1 配置位为“1”（未编程），则禁止时钟切换功能。同时也将禁止故障保护时钟监视器功能：这是默认设置。当禁止时钟切换时，OSCCON 寄存器的 NOSC 控制位不控制时钟选择。但是，OSCCON 寄存器的 COSC 位会反映出由 Fosc 配置寄存器中的 FPR 和 FOS 配置位所选择的时钟源。当禁止时钟切换时，OSCCON 寄存器的 OSWEN 控制位不受影响。它总是保持为“0”。

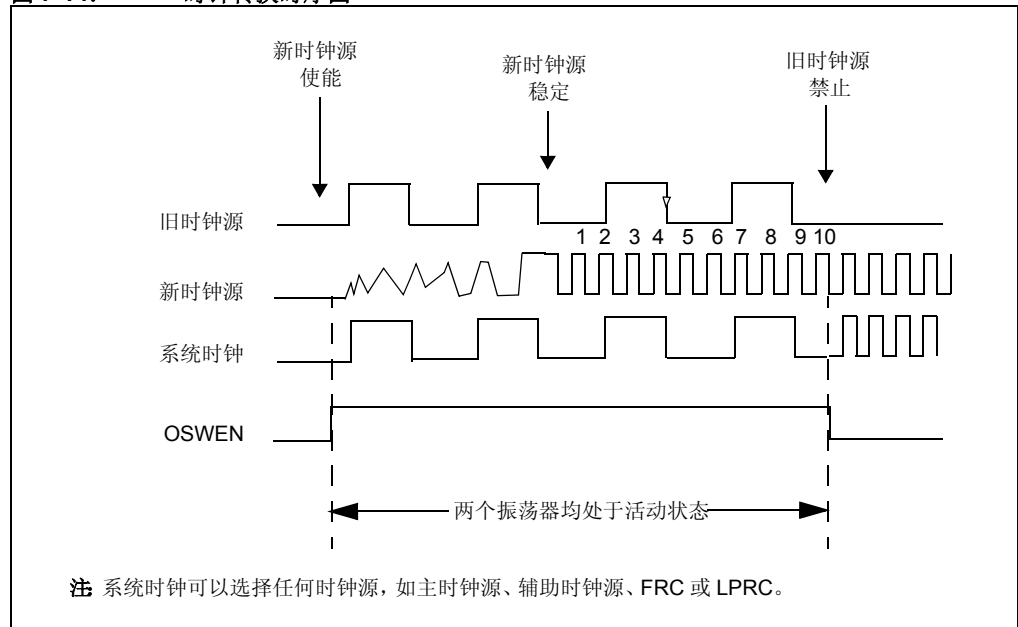
7.17.2 振荡器切换顺序

硬件和软件通过以下步骤改变器件时钟源。（以下所述步骤使用振荡器系统版本 1 的 OSCCON 寄存器定义。若需振荡器系统版本 2 和版本 3 的 OSCCO 寄存器说明，请参见第 7.4 节“振荡器控制寄存器——OSCCON 和 OSTUN”）：

1. 如果需要的话，读 COSC<1:0> 状态位（OSCCON<13:12>）确定当前振荡源。
2. 执行解锁序列以允许写入 OSCCON 寄存器的高字节。
3. 将适当的值写入 NOSC<1:0> 控制位（OSCCON<9:8>）以选择新的时钟源。
4. 执行解锁序列以允许写入 OSCCON 寄存器的低字节。
5. 将 OSWEN 位（OSCCON<0>）置 1。这将启动振荡器切换。
6. 时钟切换硬件将 NOSC<1:0> 控制位的新值和 COSC<1:0> 状态位的值进行比较。如果两者相同，时钟切换则为冗余操作。在这种情况下，OSWEN 位被自动清零且停止时钟切换。
7. 如果启动了有效时钟切换，将清零 LOCK（OSCCON<5>）和 CF（OSCCON<3>）状态位。
8. 如果新振荡器不在运行中，硬件会将它开启。如果必须开启晶振，硬件将等待直到 OST 超时。如果新的振荡源使用 PLL，硬件将等待直到检测到 PLL 锁定（LOCK = 1）。
9. 硬件会等待新时钟源起振后 10 个时钟周期，然后执行时钟切换。
10. 硬件清零 OSWEN 位表示时钟转换成功。此外，NOSC<1:0> 位的值被传送到 COSC<1:0> 状态位中。
11. 时钟切换完成。此时旧时钟源将被关闭，但以下情况例外：
 - 如果使能 WDT 或 FSCM，LPRC 振荡器保持为打开。
 - 如果 LPOSCEN = 1（OSCCON<1>），LP 振荡器保持打开。

注： 在整个时钟切换过程中，处理器将继续执行。受执行时间影响的代码不应在此时执行。

图 7-14: 时钟转换时序图



7.17.3 时钟切换技巧

- 如果目标时钟源是晶体振荡器，时钟切换时间由振荡器起振时间的决定。
- 如果新的时钟源未起振或不存在，时钟切换硬件将等待 10 个同步周期。由于 OSWEN 位（OSCCON<0>）始终保持置 1 状态，因此用户可以通过检测该位以判断发生了这种情况。
- 如果新时钟源使用 PLL，在达到锁定之前不会产生时钟切换。由于 PLL 失锁时 LOCK 位清零且 OSWEN 位置 1，因此用户可以检测到这种状态。
- 用户可能希望在执行时钟切换时考虑 POST<1:0> 控制位（OSCCON<7:6>）的设置。切换到低频时钟源（比如后分频比大于 1:1 的 LP 振荡器）时，将导致器件运行非常缓慢。

注： 当应用使能故障保护时钟监视器时，不应尝试切换到频率低于 100 kHz 的时钟。如果进行了这样的时钟切换，器件可能产生一个振荡器故障陷阱并切换到快速 RC 振荡器。

7.17.4 中止时钟切换

在时钟切换未完成时，可以通过清零 OSWEN 位复位时钟切换逻辑电路。清零 OSWEN 位（OSCCON<0>）将：

1. 放弃时钟切换
2. 如果使用了 OST，则将它停止并复位
3. 如果使用了 PLL，则将它停止

可以在任何时间中止时钟切换过程。

7.17.5 在时钟切换过程中进入休眠模式

如果器件在时钟切换操作过程中进入休眠模式，将中止时钟切换操作。此时处理器保持旧的时钟选择并清零 OSWEN 位。然后将正常执行 PWRSAV 指令。

7.17.6 进行时钟切换的建议代码序列

执行以下步骤可更改振荡源：

- 在 OSCCON 寄存器未锁定或写序列过程中禁止中断。
- 执行解锁序列以写入 OSCCON 的高字节。
- 将新的振荡源写入 NOSC 控制位。
- 执行解锁序列以写入 OSCCON 的低字节。
- 将 OSWEN 位置 1。
- 在对时钟要求不高时继续执行不受时钟影响的代码。（可选）
- 调用适当数量的软件延时（周期计数）以使振荡器和 / 或 PLL 起振。
- 检查 OSWEN 位是否为“0”。如果是，我们就已成功更改了振荡源。
- 如果 OSWEN 位仍为置 1 状态，检查 LOCK 位以确定故障原因。

7.17.7 时钟切换代码示例

7.17.7.1 启动时钟切换

以下代码序列给出了解锁 OSCCON 寄存器和开始时钟切换操作的方法：

```
;Place the new oscillator selection in W0
;OSCCONH (high byte) Unlock Sequence
MOV    #OSCCONH, w1
MOV    #0x78, w2
MOV    #0x9A, w3
MOV.B  w2, [w1]
MOV.B  w3, [w1]

;Set new oscillator selection
MOV.B  WREG, OSCCONH

;OSCCONL (low byte) unlock sequence
MOV    #OSCCONL, w1
MOV.B  #0x01, w0
MOV    #0x46, w2
MOV    #0x57, w3
MOV.B  w2, [w1]
MOV.B  w3, [w1]

;Start oscillator switch operation
MOV.b  w0, [w1]
```

7.17.7.2 中止时钟切换

以下代码序列用于中止不成功的时钟切换：

MOV	OSCCON, W0	; Read OSCCON into W0
BCLR	W0, #OSWEN	; Clear bit 0 in W0
MOV	#OSCCON, W1	; pointer to OSCCON
MOV.B	#0x46, W2	; first unlock code
MOV.B	#0x57, W3	; second unlock code
MOV.B	W2, [W1]	; write first unlock code
MOV.B	W3, [W1]	; write second unlock code
MOV.B	W0, [W1]	; ABORT the switch

7.18 设计技巧

问 1: 当在上电后使用示波器观察 OSC2 引脚，发现没有时钟信号，是什么原因？

答:

1. 器件进入休眠模式而没有唤醒源（如，WDT、MCLR 或中断）将其唤醒。验证代码确保器件不在未提供唤醒源的情况下进入休眠。如果可能的话，尝试在 MCLR 上用低脉冲唤醒器件。上电时保持 MCLR 为低电平也可以给晶振更多时间起振，但在 MCLR 引脚电平为高电平之前，程序计数器不会开始计数。
2. 为所需的频率选择了错误的时钟模式。对于空白器件，默认振荡器为 EC + 16 倍频 PLL。大多数器件在买来时钟选择为默认模式，而默认模式不会使晶振或谐振器产生振荡。验证时钟模式是否已经正确编程。
3. 没有按照正确的上电顺序上电。如果在上电之前通过 I/O 引脚为 CMOS 模块上电，可能会发生故障（锁存，或不正确起振等等）。也可能由于欠压条件、起振时供电线路噪声过大和 VDD 上升缓慢而产生问题。可以尝试断开 I/O 外部连接，并使用已知的、好的、电压快速上升的电源为器件上电。欠压和上电序列的注意事项请参见器件数据手册中的上电信息。
4. 连接到晶振的 C1 和 C2 电容器没有正确连接或取值不正确。确定所有连接正确。器件数据手册中有关这些组件的值一般能够使振荡器运行，但不一定是符合您设计的最佳值。

问 2: 器件启动了，但频率比晶振谐振频率高很多。

答: 此振荡器电路的增益太高。请参见第 7.6 节“晶体振荡器 / 陶瓷谐振器”，有助于选择 C2（可能需要更高的值）、Rs（可能需要接入）和时钟模式（可能选择了错误模式）。对于低频晶体，如常用的 32.768 kHz 低频晶振，上述情况尤其可能发生。

问 3: 设计好的电路运行正常，但频率有点不稳定。如何进行调整？

答: 改变 C1 的值会对振荡频率造成一些影响。如果使用的是串联谐振晶体，其谐振频率与具有相同标定频率的并联谐振晶体的谐振频率将有所不同。确认您使用了并联谐振晶体。

问 4: 电路板工作正常，但会突然退出或丢失时间。

答: 当然应进行软件检查，以调查时间丢失的原因。除此之外，还有可能是因为振荡器的输出幅度不够高而不能可靠地触发振荡器的输入。检查 C1 和 C2 值并确保器件配置位设置正确以选择所希望的振荡器模式。

问 5: 我把示波器探头连接到振荡器引脚时，并没有看到预期的结果。为什么？

答: 记住示波器探头自身也是有电容的。当示波器探头连接到振荡电路会改变振荡器的特性。考虑使用低电容（有效的）的探头。

7.19 相关应用笔记

本节列出了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与振荡器模块相关的应用笔记有：

标题	应用笔记编号
PIC® Microcontroller Oscillator Design Guide	AN588
Low Power Design using PIC® Microcontrollers	AN606
Crystal Oscillator Basics and Crystal Selection for rfPIC® and PIC® Devices	AN826

注： 如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

7.20 版本历史

版本 A

这是本文档的初始发行版。

版本 B

此版本为 dsPIC30F 振荡器模块添加了技术内容的修改。

版本 C

此版本加入了本文档更新时已知的所有勘误表。

版本 D

此版本加入了在 dsPIC30F 器件的通用系列、传感器系列和电机控制系列上实现的三种版本的振荡器系统（版本 1、版本 2 和版本 3）的详细信息。

第 8 章 复位

目录

本章包括下列主题：

8.1	简介	8-2
8.2	复位时的时钟源选择	8-5
8.3	POR：上电复位	8-5
8.4	外部复位（EXTR）	8-7
8.5	软件复位指令（SWR）	8-7
8.6	看门狗超时复位（WDTR）	8-7
8.7	欠压复位（BOR）	8-8
8.8	使用 RCON 状态位	8-10
8.9	器件复位时间	8-11
8.10	器件起振时间曲线	8-13
8.11	特殊功能寄存器复位状态	8-16
8.12	设计技巧	8-17
8.13	相关应用笔记	8-18
8.14	版本历史	8-19

8.1 简介

复位模块结合了所有复位源并控制器件的主复位信号 $\overline{\text{SYSRST}}$ 。以下列出了器件的复位源：

- POR: 上电复位
- EXTR: 引脚复位 ($\overline{\text{MCLR}}$)
- SWR: RESET 指令
- WDTR: 看门狗定时器复位
- BOR: 欠压复位
- TRAPR: 陷阱冲突复位
- IOPR: 非法操作码复位
- UWR: 未初始化的 W 寄存器复位

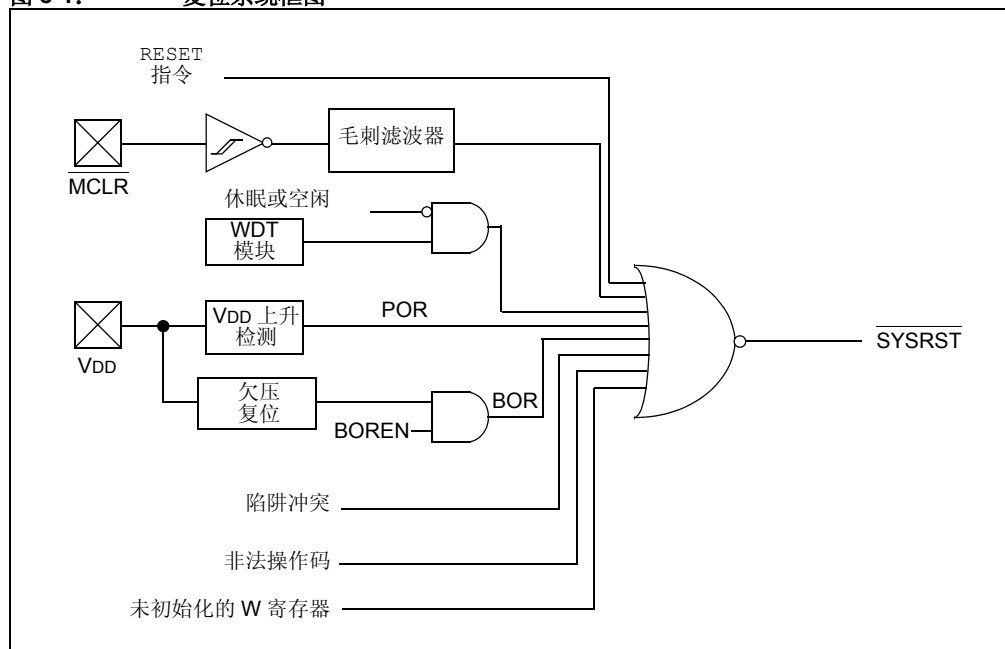
图 8-1 所示为复位模块的简化框图。任何激活的复位源都会产生 $\overline{\text{SYSRST}}$ 信号。许多与 CPU 和外设相关的寄存器均会被强制变为一个已知的“复位状态”。多数寄存器都不受复位影响；它们的状态在 POR 时未知，而在其他复位时不变。

注： 如需了解寄存器复位状态的信息，请参见本数据手册中特定外设或 CPU 章节。

任何类型的器件复位都会将 RCON 寄存器中相应的状态位置 1 以表明复位类型（参见寄存器 8-1）。POR 会清零除 POR 和 BOR 位（RCON<2:1>）以外的所有位，而 POR 和 BOR 位被置 1。用户可以在代码执行过程中的任何时间置位或清零任何位。RCON 位仅用作状态位。在软件中将特定的状态位置 1 不会导致器件复位发生。

RCON 寄存器也具有其他与低压检测模块、看门狗定时器和器件低功耗状态相关的位。本手册的其他章节中将讨论这些位的功能。

图 8-1: 复位系统框图



寄存器 8-1: RCON: 复位控制寄存器

高字节:							
R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
TRAPR	IOPUWR	BGST	LVDEN	LVDL<3:0>			
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1
EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR
bit 7				bit 0			

- bit 15 **TRAPR:** 陷阱复位标志位
1 = 发生了陷阱冲突复位
0 = 未发生陷阱冲突复位
- bit 14 **IOPUWR:** 非法操作码或未初始化的 W 寄存器访问复位标志位
1 = 检测到非法操作码、非法地址模式或未初始化的 W 寄存器用作地址指针而导致复位
0 = 非法操作码或未初始化的 W 寄存器复位都没有发生
- bit 13 **BGST:** 带隙稳态位
1 = 带隙已稳定
0 = 带隙不稳定且 LVD 中断应该被禁止
- bit 12 **LVDEN:** 低压检测电源使能位
1 = 使能 LVD, LVD 电路上电
0 = 禁止 LVD, LVD 电路掉电
- bit 11-8 **LVDL<3:0>:** 低压检测限制位
有关更详细信息, 请参见第 9 章 “低压检测 (LVD)”。
- bit 7 **EXTR:** 外部复位 (MCLR) 引脚位
1 = 发生主清零 (引脚) 复位
0 = 未发生主清零 (引脚) 复位
- bit 6 **SWR:** 软件 RESET (指令) 标志位
1 = 执行了 RESET 指令
0 = 未执行 RESET 指令
- bit 5 **SWDTEN:** WDT 位的软件使能 / 禁止
1 = WDT 启用
0 = WDT 关闭

注: 如果 FWDTEN 熔丝位为 “1” (未编程), 则 WDT 总是使能, 而不管 SWDTEN 位是否置 1。
- bit 4 **WDTO:** 看门狗定时器超时标志位
1 = WDT 发生超时
0 = WDT 未发生超时
- bit 3 **休眠:** 从休眠状态唤醒标志位
1 = 器件处于休眠模式
0 = 器件未处于休眠模式
- bit 2 **空闲:** 从空闲状态唤醒标志位
1 = 器件处于空闲模式
0 = 器件不处于空闲模式

寄存器 8-1: RCON: 复位控制寄存器（续）

- bit 1

BOR: 欠压复位标志位
1 = 发生欠压复位。注意 BOR 在上电复位后将置 1。
0 = 未发生欠压复位
- bit 0

POR: 上电复位标志位
1 = 发生上电延时复位
0 = 未发生上电延时复位

注: 所有复位状态位都可以用软件置 1 或清零。用软件将这些位中的一位置 1 不会导致器件复位。

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = 上电复位时的值	1= 置位	0 = 清零	x = 未知

8.2 复位时的时钟源选择

如果使能了时钟切换功能，器件复位时的系统时钟源选择如表 8-1 所示。如果禁止了时钟切换功能，则总是根据振荡器配置熔丝位选择系统时钟源。更多详细信息，请参见第 7 章“振荡器”。

表 8-1: 不同复位类型与振荡器选择关系（使能时钟切换功能）

复位类型	时钟选择的依据
POR	振荡器配置熔丝位
BOR	振荡器配置熔丝位
EXTR	COSC 控制位（OSCCON<13:12>）
WDTR	COSC 控制位（OSCCON<13:12>）
SWR	COSC 控制位（OSCCON<13:12>）

8.3 POR：上电复位

有两个阈值电压与上电复位（POR）相关。第一个电压是器件阈值电压 VPOR。器件的阈值电压是器件逻辑电路可正常工作的电压。第二个与 POR 事件相关的电压是 POR 电路阈值电压，它的标称值为 1.85V。

上电事件在检测到 VDD 电压上升时会产生内部上电复位脉冲。复位脉冲在 VPOR 上产生。器件供电电压的特性必须符合特定的起始电压和上升速率要求以产生 POR 脉冲。尤其重要的是，在新的 POR 开始之前，VDD 必须降到 VPOR 以下。如需了解 VPOR 和 VDD 上升速率规范的更多信息，请参见器件数据手册的“电气规范”一章。

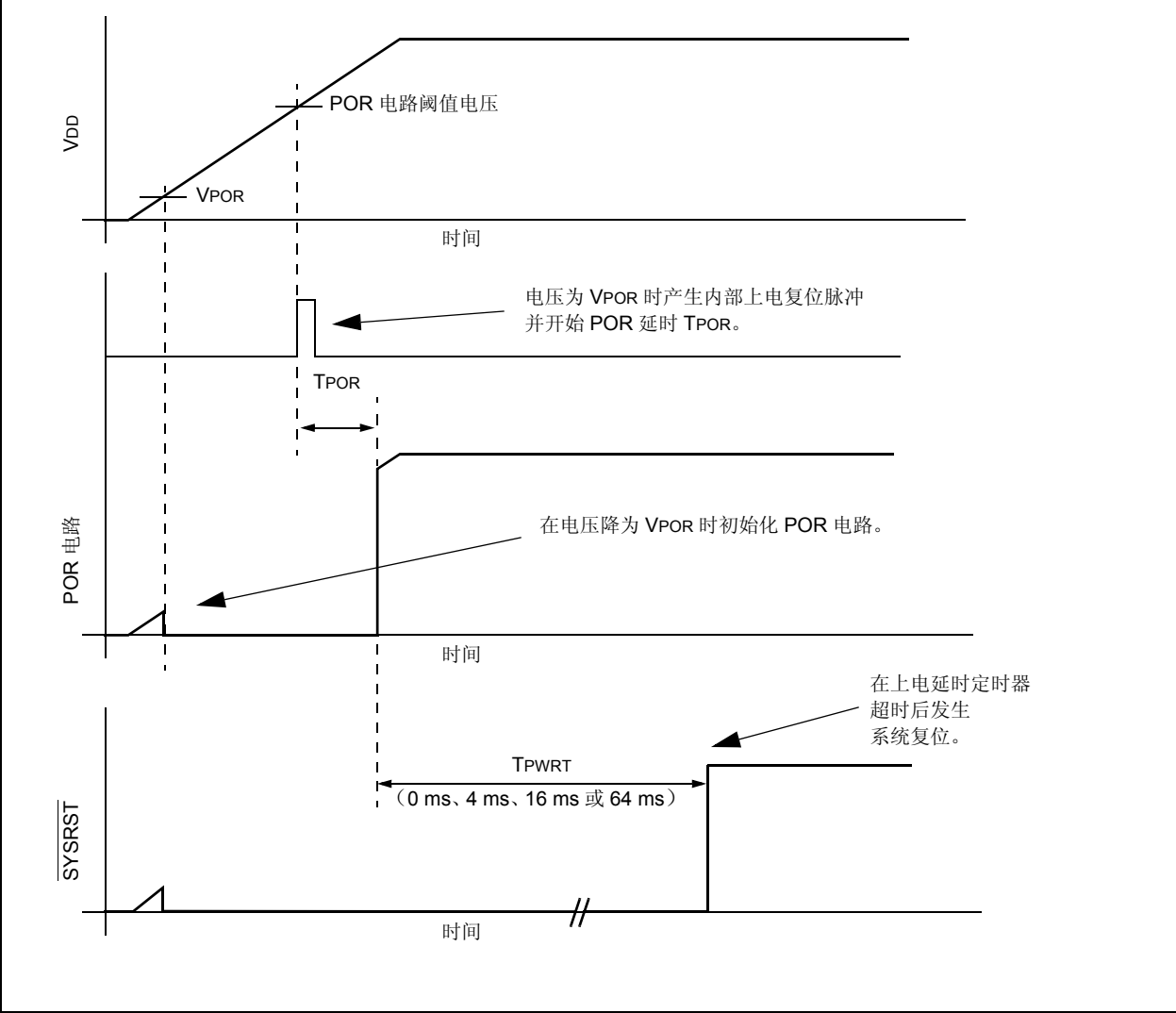
POR 脉冲将复位 POR 定时器并将器件置于复位状态。POR 也会选择振荡器配置位指定的器件时钟源。

在产生上电复位脉冲之后，POR 电路会产生一段短时间的延时 TPOR，其标称值为 10 μs，以确保内部器件偏置电路稳定。此外，还可使用户选择的上电延时超时时间（TPWRT）。TPWRT 参数的值基于器件配置位的设置，它可以为 0 ms（无延时）、4 ms、16 ms 或 64 ms。器件上电延时的总延时为 TPOR + TPWRT。当这些延时结束后，将在指令周期时钟的下一个前沿产生 SYSRST 信号，同时 PC 跳转到复位向量。

图 8-2 所示为 SYSRST 信号的时序。当 VDD 电压降到阈值电压（VT）以下时，初始化上电复位。当 VDD 超过 POR 电路阈值电压的瞬时，插入一个 POR 延时。最后，在 SYSRST 信号产生前，还将发生一段 PWRT 延时时间（TPWRT）。

上电事件会置位 POR 和 BOR 状态位（RCON<1:0>）。

图 8-2: VDD 上升过程中的 POR 模块时序图



注: 当器件退出复位状态（开始正常操作）时，其工作参数（电压、频率和温度等）都应在相应的工作范围之内，否则器件将不能正常工作。用户必须确保从第一次上电到 **SYSRST** 变为无效之间的时间足够长，以使所有工作参数都符合规范。

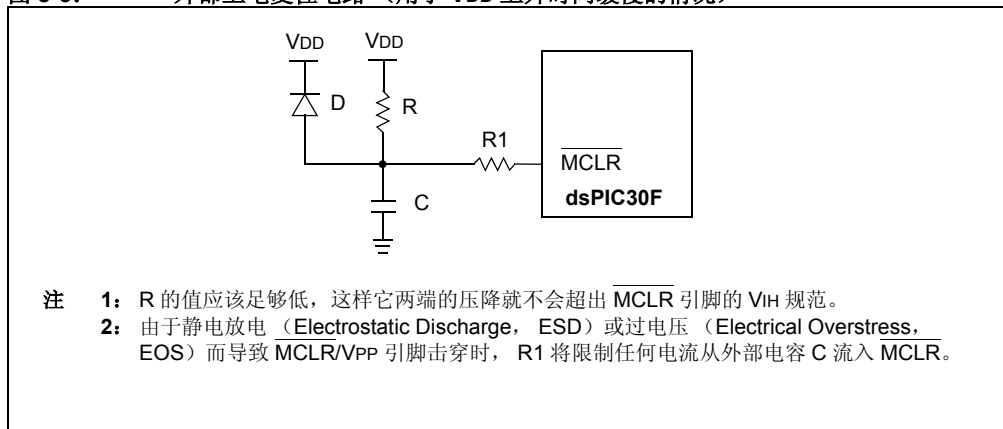
8.3.1 使用 POR 电路

要利用 POR 电路，只要直接将 $\overline{\text{MCLR}}$ 引脚连接到 V_{DD} 。这将消除通常创建上电复位延时所需要的外部 RC 组件。必须有一个 V_{DD} 最小上升时间。更详细信息，请参见特定器件数据手册的“电气规范”一章。

根据不同的应用，可能需要在 $\overline{\text{MCLR}}$ 引脚和 V_{DD} 之间连接一个电阻。此电阻可以用来减弱 $\overline{\text{MCLR}}$ 引脚上来自于供电线路的噪声。当将器件安装到应用电路时，需要将器件编程电压 V_{PP} 加到 $\overline{\text{MCLR}}$ 引脚，此时也需要使用该电阻。大多数器件的 V_{PP} 电压都是 13V。

图 8-3 所示是用于供电电压上升速率缓慢的一种 POR 电路。如果器件在 V_{DD} 处于有效工作范围之前退出复位状态，才需要使用外部上电复位电路。二极管 D 在 V_{DD} 掉电时可以帮助电容快速放电。

图 8-3: 外部上电复位电路（用于 V_{DD} 上升时间缓慢的情况）



8.3.2 上电延时定时器（PWRT）

在器件 POR 或 BOR（欠压复位）产生 $\overline{\text{SYSRST}}$ 之前，PWRT 提供可选的延时（TPWRT）。PWRT 延时是除 POR 延时（TPOR）之外额外提供的延时。PWRT 时间延时标称值可以为 0 ms、4 ms、16 ms 或 64 ms（参见图 8-2）。

可使用 FBORPOR 器件配置寄存器中的 $\text{FPWRT}<1:0>$ 配置熔丝位选择 PWRT 延时。更多详细信息，请参见第 24 章“器件配置”。

8.4 外部复位（EXTR）

只要 $\overline{\text{MCLR}}$ 引脚拉为低电平，如果在 $\overline{\text{MCLR}}$ 上的输入脉冲比规定的最小宽度更长（更详细的信息，请参见特定器件数据手册的“电气规范”），器件都会异步产生 $\overline{\text{SYSRST}}$ 信号。当 $\overline{\text{MCLR}}$ 引脚被释放后，在下一个指令时钟周期将产生 $\overline{\text{SYSRST}}$ 信号并开始复位向量取指。处理器将保持 EXTR 发生之前已经在使用的时钟源。EXTR 状态位（ $\text{RCON}<7>$ ）将被置 1 以表明 $\overline{\text{MCLR}}$ 复位。

8.5 软件复位指令（SWR）

任何时候只要执行了 RESET 指令，器件都会产生 $\overline{\text{SYSRST}}$ 信号，从而将器件置于特殊复位状态。复位状态不会重新初始化时钟。在 RESET 指令之前生效的时钟源仍将继续使用。 $\overline{\text{SYSRST}}$ 会在下一个指令周期产生，并开始复位向量取指。

8.6 看门狗超时复位（WDTR）

任何时候只要发生看门狗超时，器件都将异步产生 $\overline{\text{SYSRST}}$ 信号。时钟源仍然保持不变。注意在休眠或空闲模式发生 WDT 超时将唤醒处理器，但不会复位处理器。如需了解更多信息，请参见第 10 章“看门狗定时器和低功耗模式”。

8.7 欠压复位 (BOR)

BOR (欠压复位) 模块是基于内部参考电压电路的。BOR 模块的主要用途是在发生欠压条件时产生器件复位。欠压条件通常由 AC 电源上的干扰信号 (即由于不良的电源传输线路造成的 AC 周期波形丢失) 或接入大负载时过电流造成电压下降产生的。

BOR 模块允许选择以下电压跳变点之一:

- $V_{BOR} = 2.0V$
- $V_{BOR} = 2.7V$
- $V_{BOR} = 4.2V$
- $V_{BOR} = 4.5V$

注: 这里所述的 BOR 电压跳变点都是标称值, 仅供设计参考。BOR 电压限制规范的详细信息, 请参见特定器件数据手册的“电气规范”一章。

在欠压复位时, 器件会根据器件配置位值 ($FPR<3:0>$, $FOS<1:0>$) 选择系统时钟源。如果使能了 PWRT 延时, 那么就会在 $\overline{SYSRST}$ 信号产生之前加上一段 PWRT 延时时间 ($TPWRT$)。

如果选择了晶体振荡器时钟源, 欠压复位将调用振荡器起振定时器 (Oscillator Start-up Timer, OST)。系统时钟将保持到 OST 超时。如果系统时钟源来自 PLL, 则时钟将被保持到 LOCK 位 ($OSCCON<5>$) 置 1。

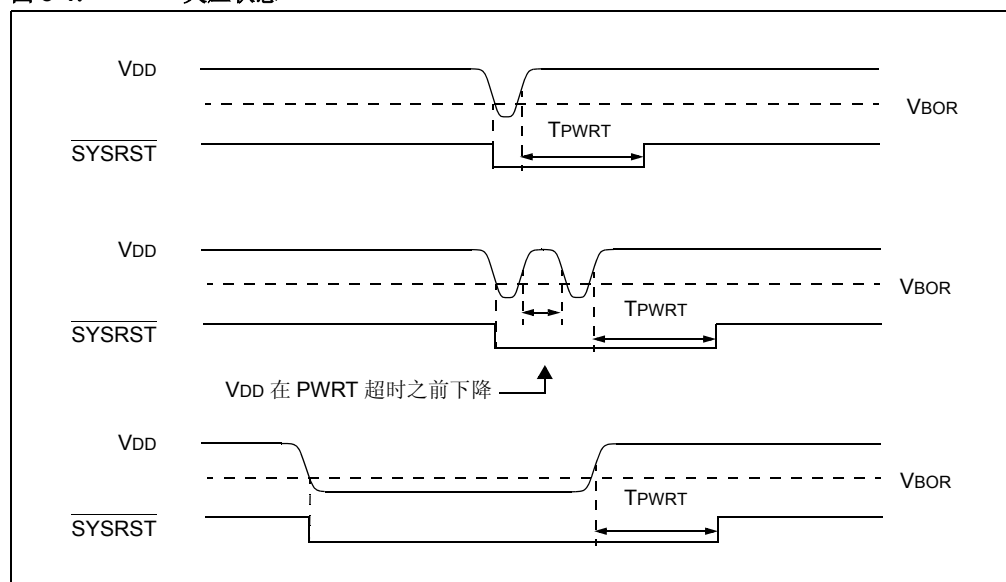
BOR 状态位 ($RCON<1>$) 将被置 1 以表明发生了 BOR。

使能时, BOR 电路将继续在休眠或空闲模式下工作, 并当 V_{DD} 下降到 BOR 阈值电压以下时, 复位器件。

有关 BOR 的电气规范, 请参考相应的器件数据手册的“电气规范”一章。

图 8-4 所示为典型的欠压情况。如图, 每当 V_{DD} 电压上升到高于 V_{BOR} 跳变点时, 将开始 PWRT 延时 (如果被使能)。

图 8-4: 欠压状态



8.7.1 BOR 配置

BOR 模块通过器件配置熔丝位被使能 / 禁止。

默认情况下 BOR 模块使能，可以将 BOREN 器件配置熔丝位编程为 0（FBORPOR<7>）来禁止该模块（降低功耗）。BOREN 配置熔丝位在 FBORPOR 器件配置寄存器中。使用 BORV<1:0> 配置熔丝位（FBOR<5:4>）选择 BOR 电压跳变点（VBOR）。更多详细信息，请参见第 24 章“器件配置”。

8.7.2 BOR 工作时的电流消耗

BOR 电路依赖于内部参考电压电路，该参考电路和其他外设器件（如低压检测模块）共享。只要使能其中一个与它相关的外设，内部参考电压就会被激活。由于这个原因，BOR 被禁止时，用户也许观察不到预期的电流消耗变化。

8.7.3 非法操作码复位

如果器件试图执行从程序存储器中取出非法操作码值，将会产生器件复位。非法操作码复位功能可以阻止器件执行用于存储常数的程序存储器部分。要利用非法操作码复位，只能使用每个程序存储器部分的低 16 位存储数据值。高 8 位应该被编程为非法操作码值 0x3F。

如果非法操作码值导致器件复位产生，IOPUWR 状态位（RCON<14>）被置 1。

8.7.4 未初始化的 W 寄存器复位

所有复位中都将清零 W 寄存器阵列（除 W15 之外），并在写入前将 W 寄存器视为未被初始化。试图将未初始化的寄存器用作地址指针会使器件复位。此外，IOPUWR 状态位（RCON<14>）会被置 1。

8.7.5 陷阱冲突复位

只要同时有多个硬陷阱中断源待处理，就会产生器件复位。TRAPR 状态位（RCON<15>）会被置 1。更多有关陷阱冲突复位的信息，请参见第 6 章“复位中断”。

8.8 使用 RCON 状态位

用户可以在任何器件复位后读取 RCON 寄存器，确定复位的原因。

注： RCON 寄存器中的状态位应该在被读取后清零，这样器件下一次复位后的 RCON 值才有意义。

表 8-2 提供了复位标志位操作的汇总。

表 8-2: 复位标志位的操作

标志位	置位原因:	清零原因:
TRAPR (RCON<15>)	陷阱冲突事件	POR
IOPWR (RCON<14>)	非法操作码或访问未初始化的 W 寄存器	POR
EXTR (RCON<7>)	MCLR 复位	POR
SWR (RCON<6>)	RESET 指令	POR
WDTO (RCON<4>)	WDT 超时	PWRSV 指令， POR
SLEEP (RCON<3>)	PWRSV #SLEEP 指令	POR
IDLE (RCON<2>)	PWRSV #IDLE 指令	POR
BOR (RCON<1>)	POR, BOR	
POR (RCON<0>)	POR	

注： 所有复位标志位由用户软件置位或清零。

8.9 器件复位时间

表 8-3 总结了各种器件复位的复位时间。注意在 POR 延时和 PWRT 延时结束后会产生系统复位信号 **SYSRST**。

器件实际开始执行代码的时间还取决于系统振荡器延时，它包括振荡器起振定时器（OST）的延时和 PLL 锁定时间。OST 和 PLL 锁定时间与相应的 **SYSRST** 延时同时发生。

FSCM 延时决定在 **SYSRST** 信号发出后 FSCM 开始监视系统时钟源的时间。

表 8-3: 各种器件复位的复位延时

复位类型	时钟源	SYSRST 延时	系统时钟延时	FSCM 延时	注
POR	EC、EXTRC、FRC、LPRC	TPOR+ TPWRT	—	—	1, 2
	EC + PLL	TPOR + TPWRT	TLOCK	TFSCM	1, 2, 4, 5
	XT、HS、XTL、LP	TPOR + TPWRT	TOST	TFSCM	1, 2, 3, 5
	XT + PLL	TPOR + TPWRT	TOST + TLOCK	TFSCM	1, 2, 3, 4, 5
BOR	EC、EXTRC、FRC、LPRC	TPWRT	—	—	2
	EC + PLL	TPWRT	TLOCK	TFSCM	1, 2, 4, 5
	XT、HS、XTL、TFSCMLP	TPWRT	TOST	TFSCM	1, 2, 3, 5
	XT + PLL	TPWRT	TOST + TLOCK	TFSCM	1, 2, 3, 4, 5
MCLR	任何时钟	—	—	—	
WDT	任何时钟	—	—	—	
软件	任何时钟	—	—	—	
非法操作码	任何时钟	—	—	—	
未初始化的 W 寄存器	任何时钟	—	—	—	
陷阱冲突	任何时钟	—	—	—	

- 注
- 1: TPOR = 上电复位延时（标称值为 10 μs）。
 - 2: TPWRT = 由 FPWRT<1:0> 配置位决定的其他上电延时。此延时标称值为 0 ms、4 ms、16 ms 或 64 ms。
 - 3: TOST = 振荡器起振定时器延时。在将振荡器时钟输出给系统前，10 位计数器会先计数 1024 个振荡周期。
 - 4: TLOCK = PLL 锁定时间（标称值为 20 μs）。
 - 5: TFSCM = 故障保护时钟监视器延时（标称值为 100 μs）。

8.9.1 POR 和长振荡器起振时间

振荡器起振电路及其相关延时定时器与上电时发生的器件复位延时没有关系。某些晶振电路（尤其是低频晶振）的起振时间会相对较长。因此，在 **SYSRST** 信号产生后，可能会发生以下一种或多种情况。

- 振荡电路未起振。
- 振荡器起振定时器尚未超时（如果使用了晶体振荡器）。
- PLL 未用锁定（如果使用了 PLL）。

在有效时钟源输出供系统使用前，器件不会开始执行代码。因此，如果必须确定复位延时，必须考虑到振荡器和 PLL 启振延时。

8.9.2 故障保护时钟监视器（FSCM）和器件复位

如果使能了 FSCM，它将在 **SYSRST** 产生时开始监视系统时钟源。如果此时没有可用的有效时钟源，器件将自动切换到 **FRC** 振荡器，并且用户可以在陷阱服务程序中切换到所需的晶体振荡器。

8.9.2.1 晶振和 PLL 时钟源的 FSCM 延时

当系统时钟源由晶体振荡器和 / 或 PLL 提供时，在 **POR** 和 **PWRT** 延时后会自动插入一小段延时（**T_{FSCM}**）。在此延时结束前，FSCM 不会开始监视系统时钟源。FSCM 延时标称值为 100 μ s，为振荡器和 / 或 PLL 稳定下来提供了额外的时间。在多数情况下，当禁止了 **PWRT** 时，FSCM 延时会在器件复位时防止振荡器故障陷阱。

8.10 器件起振时间曲线

图 8-5 到图 8-8 所示为几种工作情况下与器件复位有关的延时的图形化时间曲线。图 8-5 所示为使用了晶体振荡器和 PLL 作为系统时钟并且禁止 PWRT 时的延时曲线。在 VPOR 阈值电压处产生内部上电复位脉冲。内部上电复位脉冲后会产生一小段 POR 延时（在器件开始工作前总是会插入 POR 延时）。

如果使能了 FSCM，则它在 FSCM 延时结束后开始监视系统时钟的活动。如图 8-5 所示，振荡器和 PLL 延时在故障保护时钟监视器（FSCM）使能前结束。但是，这些延时也有可能直到 FSCM 使能前还未结束。在这种情况下，FSCM 会检测时钟故障并产生一个时钟故障陷阱。如果 FSCM 延时提供的时间不足以让振荡器和 PLL 稳定下来，可以使能 PWRT，以便在器件开始工作和 FSCM 开始监视系统时钟前提供更长的延时。

图 8-5: 器件复位延时（使用晶振和 PLL 时钟源，禁止 PWRT）

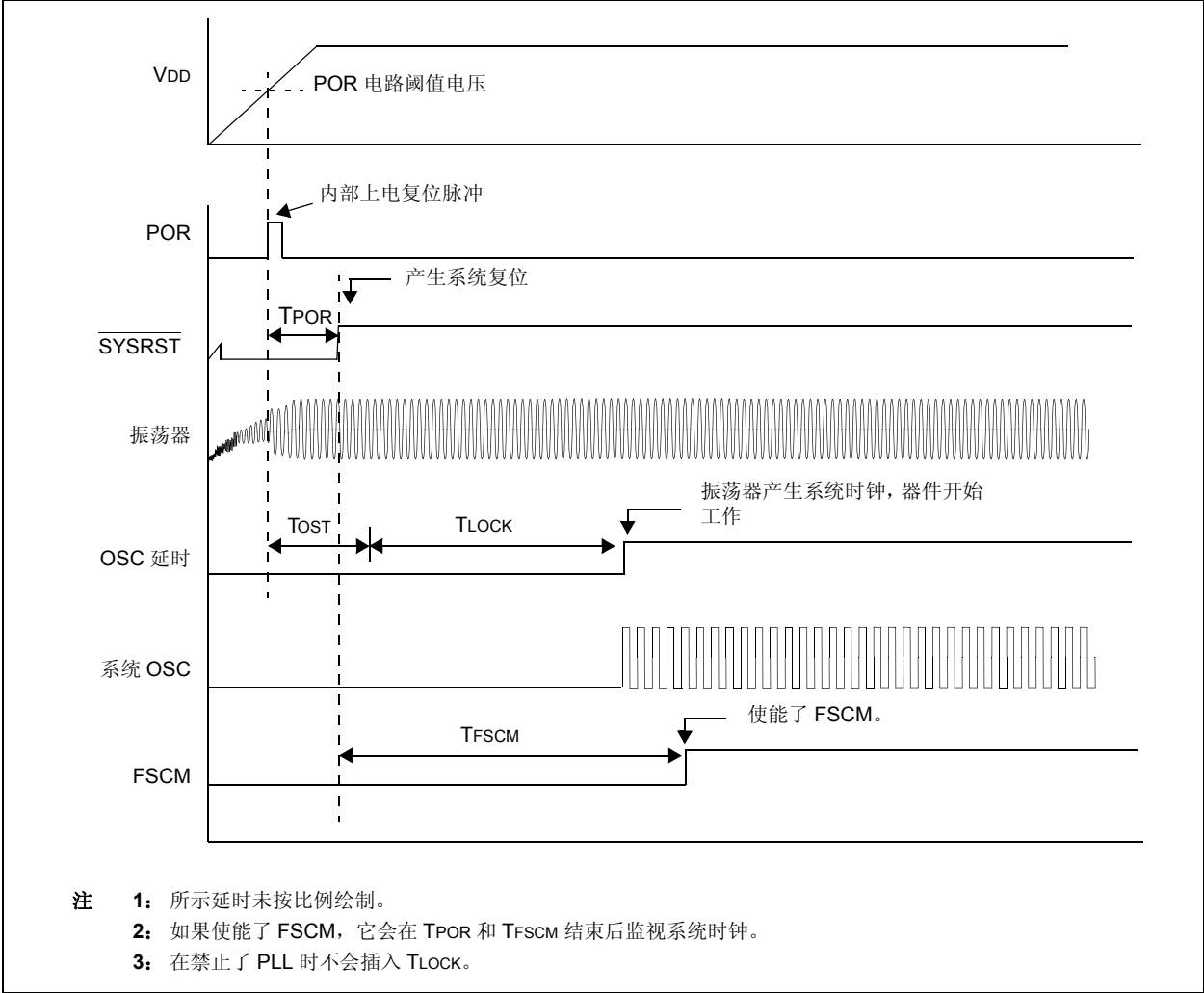


图 8-6 与图 8-5 显示的复位时间线相似，不同之处在于图 8-6 中使能了 PWRT，以便在 $\overline{\text{SYSRST}}$ 产生之前延长延时。

如果使能了 FSCM，它会在 T_{FSCM} 结束后开始监视系统时钟。注意，在多数情况下，在 T_{FSCM} 的基础上增加额外的 PWRT 延时将提供充足的时间使系统时钟源稳定下来。

图 8-6: 器件复位延时（使用晶振和 PLL 时钟源，使能 PWRT）

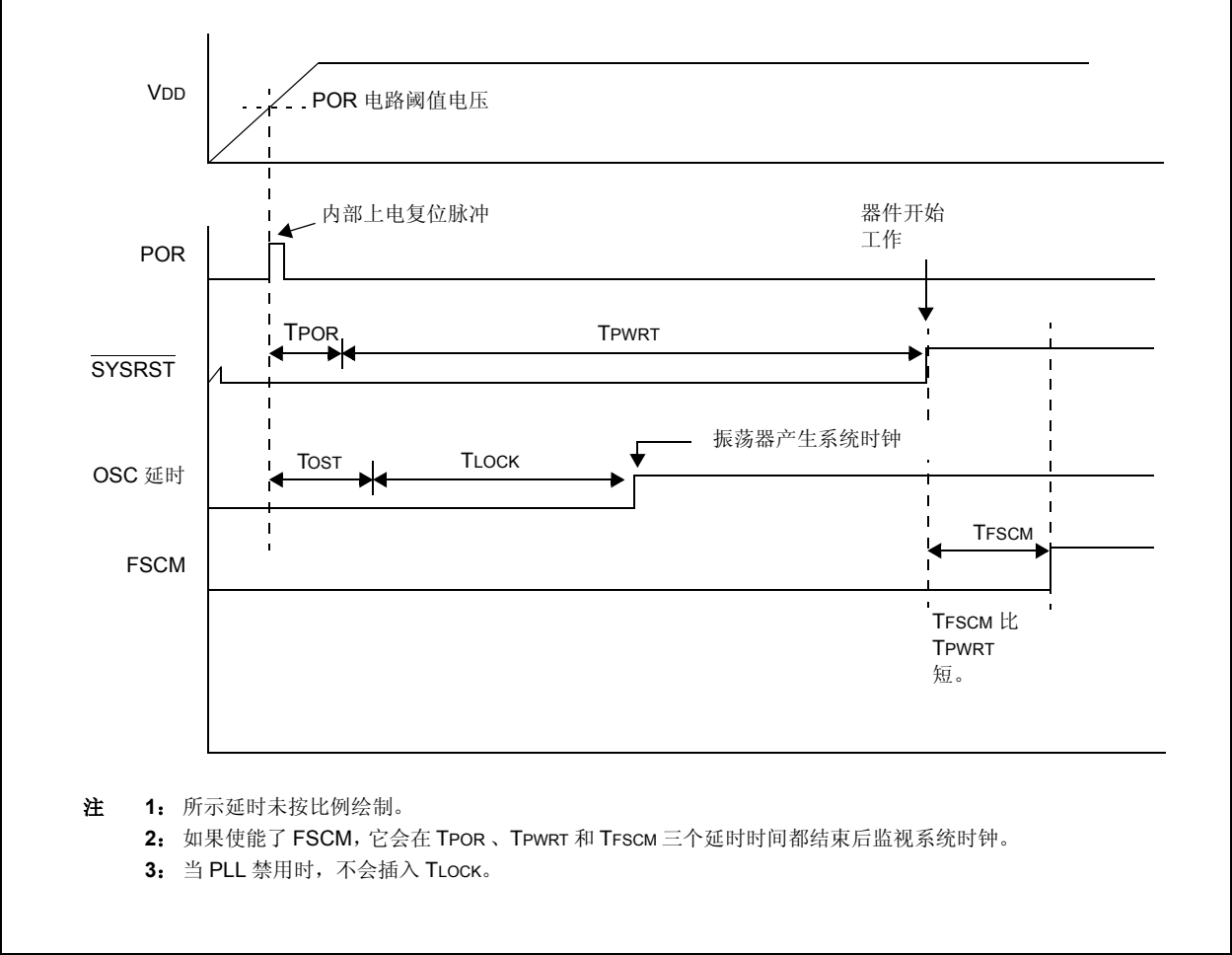


图 8-7 所示的复位时间曲线是使用 EC 和 PLL 时钟源作为系统时钟并使能了 PWRT 时的示例。此例与图 8-6 中所示的时间曲线类似，不同之处为未发生振荡器起振定时器延时 T_{OST} 。

图 8-7： 器件复位延时（使用 EC 和 PLL 时钟，使能 PWRT）

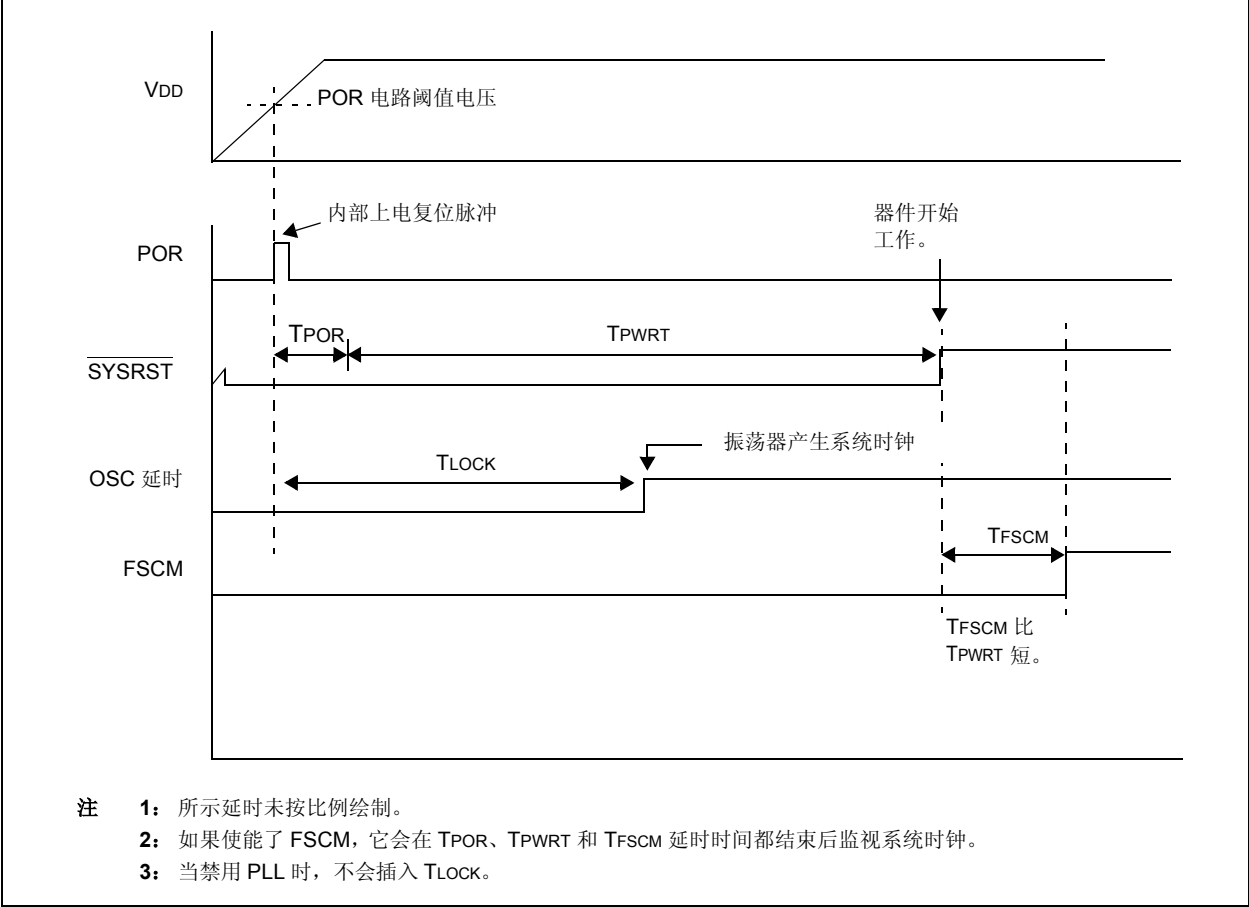
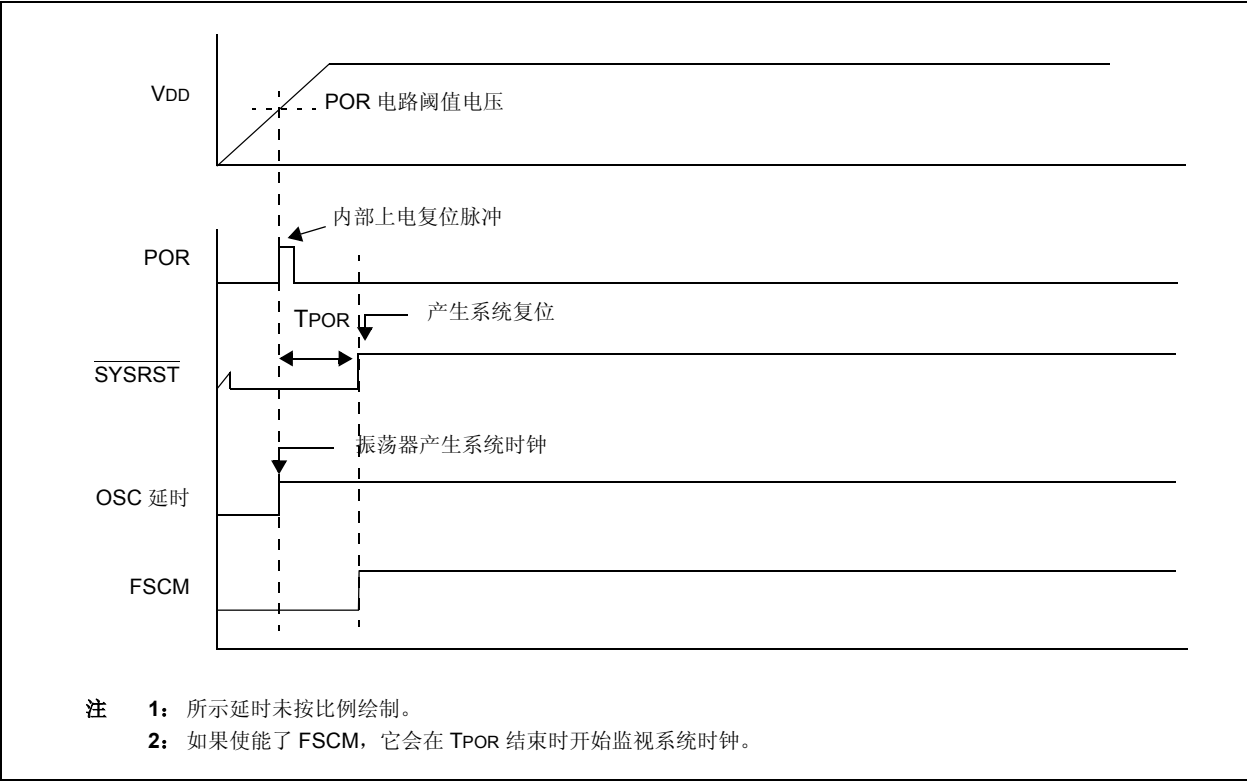


图 8-8 所示的复位时间线是在选择了不带 PLL 的 EC 或 RC 系统时钟源并禁止 PWRT 时的示例。注意此配置提供的复位延时最少。POR 延时是系统开始工作前惟一的延时。即使使能了 FSCM，也不会发生 FSCM 延时，因为系统时钟源不是晶体振荡器或 PLL 产生的。

图 8-8: 器件复位延时（使用 EC 或 RC 时钟，禁止 PWRT）



8.11 特殊功能寄存器复位状态

多数与 dsPIC30F CPU 和外设有关的特殊功能寄存器（SFR）会在器件复位时复位为某个特定值。SFR 是按其外设或 CPU 功能分组的，其复位值在本手册的各相应部分说明。

除了两个寄存器外，所有其他 SFR 的复位值都不受复位类型的影响。复位控制寄存器 RCON 的复位值取决于器件复位的类型。振荡器控制寄存器 OSCCON 的复位值取决于复位类型和在 FOSC 器件配置寄存器中的振荡器配置位的编程值（参见表 8-1）。

8.12 设计技巧

问 1: *如何使用 RCON 寄存器?*

答: 复位后的初始化代码应该检查 RCON 并确定复位源。在某些应用中, 可利用此信息采取适当措施纠正造成复位的错误。在读取了 RCON 寄存器中的所有状态位后, 应将其清零, 这样才能确保 RCON 值在器件下一次复位后能提供有意义的结果。

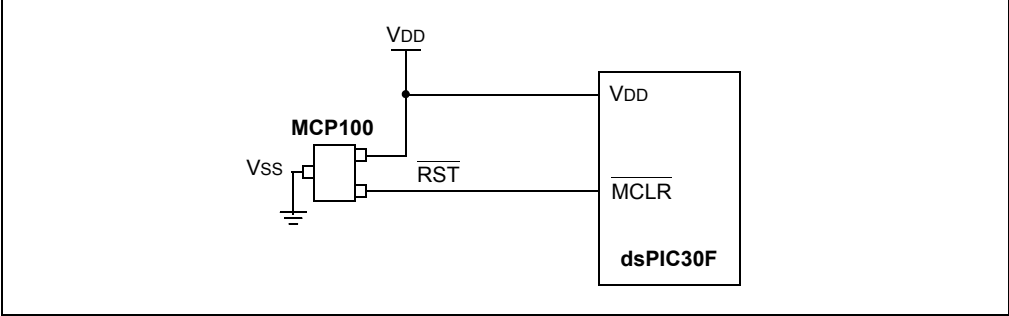
问 2: *在使用电池供电的应用中应如何使用 BOR?*

答: BOR 功能不是为检测低电池电压而设计的, 在使用电池供电的系统中应禁用此功能 (以节省电流消耗)。可以使用低压检测外设来检测电池何时没电。

问 3: *BOR 模块不具备我的应用所需的可编程跳变点。我应如何解决此问题?*

答: 在某些应用中, 器件即使有可编程 BOR 跳变点电平也可能不处于该应用所需的电平。图 8-9 所示为一个使用 MCP100 系统监视的外部欠压保护电路。

图 8-9: 使用 MCP100 的外部欠压保护电路



问 4: *我使用一个 16 位地址初始化了一个 W 寄存器, 但当我尝试将该寄存器作为地址使用时, 器件会复位。*

答: 因为所有的数据地址都是 16 位值, 如果 W 寄存器是用于存放字的, 未初始化的 W 寄存器逻辑电路仅能识别出已经正确初始化了的寄存器。试图将两个字节移入某个 W 寄存器时, 即使这两个字节是先后移入的也不能完成, 如果该 W 寄存器在操作中作为地址指针使用, 就会导致器件复位。

8.13 相关应用笔记

本节列出了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是专门为 dsPIC30F 产品系列编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与复位模式相关的应用笔记有：

标题	应用笔记编号
Power-up Trouble Shooting	AN607
Power-up Considerations	AN522

注： 如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

8.14 版本历史

版本 A

这是本文档的初始发行版。

版本 B

本章未包含任何技术内容或编辑排版上的修订，然而我们谨更新此节以反映本手册已更新到版本 B。

版本 C

本章未包含任何技术内容上的修订，然而我们谨更新此节以反映本手册已更新到版本 C。

注:

第 9 章 低压检测 (LVD)

目录

本章包括下列主题：

9.1	简介	9-2
9.2	LVD 工作原理	9-5
9.3	设计技巧	9-6
9.4	相关应用笔记	9-7
9.5	版本历史	9-8

9.1 简介

LVD 模块可应用于电池供电的应用场合。当电池消耗能量时，电池电压缓慢下降。电池的源阻抗也随着能量的损耗而不断增大。LVD 模块用于检测电池电压（即器件的 V_{DD} 电压）何时低于阈值，阈值被认为是应用中接近电池没电时的值。这使应用程序有足够的时间关闭。

LVD 模块使用内部参考电压与供电电池电压进行比较。阈值电压 V_{LVD} ，可在运行时编程。

图 9-1 所示是一种可能的使用电池供电的电压曲线。器件电压会随时间逐渐下降。当器件电压等于 V_{LVD} 电压时，LVD 逻辑产生中断。中断发生的时刻为 T_A 。这使应用软件可在器件电压不再在有效工作电压范围之前关闭系统。电压点 V_B 是最小有效工作电压规范值。该电压点所对应的时刻是 T_B 。器件关闭所需要的总时间是 $T_B - T_A$ 。

图 9-1: 典型的低压检测应用

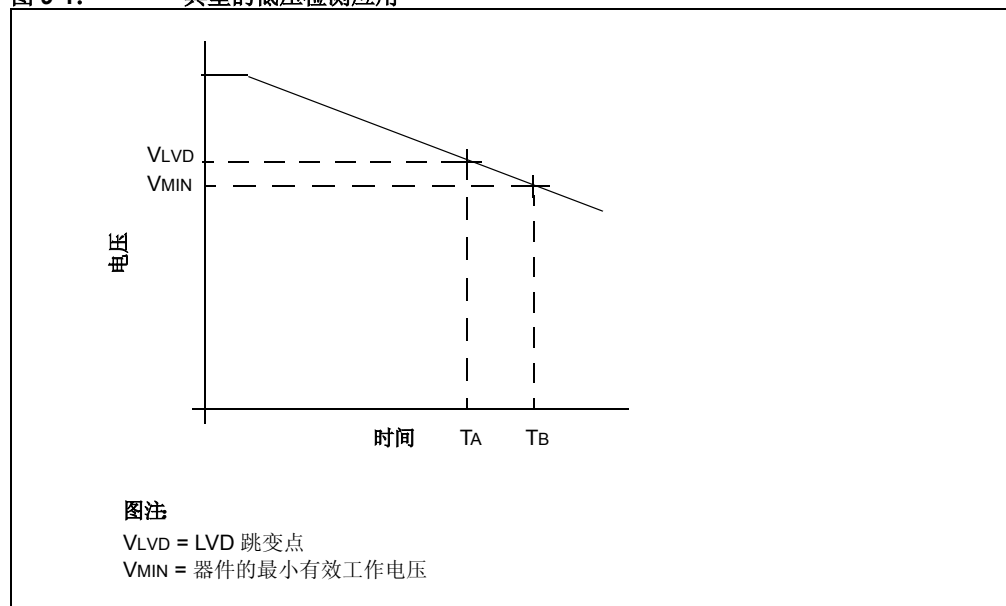
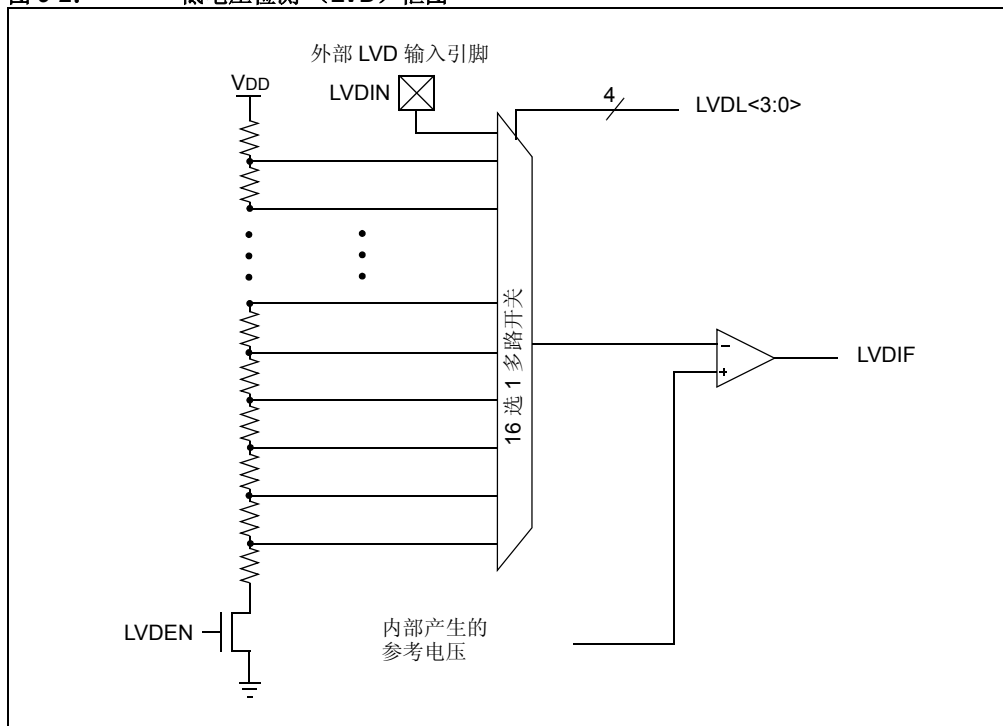


图 9-2 所示为 LVD 模块的框图。比较器使用内部产生的参考电压作为设置点。当器件所选抽头的器件的输出电压低于参考电压时，LVDIF 位（IFS2<10>）就会置 1。

电阻分压器的每个节点对应一个“跳变点”电压。可用软件编程指定该电压为 16 个值中的任意一个。

图 9-2: 低电压检测（LVD）框图



9.1.1 LVD 控制位

LVD 模块控制位位于 RCON 寄存器中。

LVDEN 位（RCON<12>）使能低压检测模块。当 LVDEN = 1 时，使能 LVD 模块。如果对功耗要求较高的话，可以清零 LVDEN 位以最大限度地节省功耗。

9.1.1.1 LVD 跳变点选择

LVDL<3:0> 位（RCON<11:8>）用于选择 LVD 跳变点。有 15 个跳变点选项，可从连接到 VDD 的内部分压器选择这些选项。如果没有适合应用的跳变点选项，可以选择在 LVDIN 引脚上施加外部 LVD 采样电压（引脚位置请参见特定器件数据手册）。外部 LVD 输入的标称跳变点电压为 1.24V。LVD 外部输入选项要求用户选择外部分压器电路的值，该电路将在预期的 VDD 电压时产生 LVD 中断。

9.1.2 内部参考电压

LVD 使用的内部带隙参考电压电路，该电路需要一段标称长度的时间才能达到稳定。详细信息请参见特定器件数据手册的“电气规范”。BGST 状态位（RCON<13>）表明何时带隙参考电压已经稳定。在使能了 LVD 模块之后，用户应该用软件查询 BGST 状态位。在稳定时间结束时，应该清零 LVDIF 位（IFS2<10>）。有关 LVD 模块的设置步骤，请参见第 9.2 节“LVD 工作原理”。器件上的其他外设也可以使用带隙参考电压电路，这样该电路可以在 LVD 模块使能之前被激活（和稳定）。

dsPIC30F 系列参考手册

寄存器 9-1: RCON: 复位控制寄存器

高字节:							
R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
TRAPR	IOPUWR	BGST	LV DEN	LV DL<3:0>			
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1
EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR
bit 7							bit 0

- bit 13 **BGST**: 带隙稳定位
1 = 带隙已稳定
0 = 带隙未稳定且应禁止 LVD 中断
- bit 12 **LV DEN**: 低电压检测电源使能位
1 = 使能 LVD, LVD 电路上电
0 = 禁止 LVD, LVD 电路掉电
- bit 11-8 **LV DL<3:0>**: 低压检测限制位
1111 = 输入到 LVD 的电压来自 LVDIN 引脚 (阈值的标称值为 1.24V)
1110 = 4.6V
1101 = 4.3V
1100 = 4.1V
1011 = 3.9V
1010 = 3.7V
1001 = 3.6V
1000 = 3.4V
0111 = 3.1V
0110 = 2.9V
0101 = 2.8V (复位时的默认值)
0100 = 2.6V
0011 = 2.5V
0010 = 2.3V
0001 = 2.1V
0000 = 1.9V

注: 这里显示的电压阈值仅用作设计参考未用。更详细信息, 请参见器件数据手册的“电气规范”。

图注:			
R = 可读位	W = 可写位	U = 未用, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知位

注: 对 RCON 寄存器中其他位的描述请参见第 8 章“复位”。

9.2 LVD 工作原理

由于 LVD 模块可以监控器件电压的状态，所以它可以为应用增加鲁棒性。当器件电压进入电压范围并靠近有效工作电压范围的下限，器件将会保存各项数值以确保“平稳”地关闭系统。

注： 系统设计应该确保在器件退出有效工作范围或被强制进入欠压复位前，应用软件有足够的时间保存数值。

供电电压下降可能相对较慢，这取决于器件的电源。这意味着 LVD 模块不需要一直工作。为了降低电流需求，只需要在检测电压时，短时间使能 LVD 电路。检测完成之后，可以禁止 LVD 模块。

9.2.1 LVD 初始化步骤

根据设置 LVD 模块需要以下步骤：

1. 如果使用了外部 LVD 输入引脚（LVDIN），确保复用到此引脚的所有其他外设都被禁止，且已通过设置 TRISx 寄存器中的相应位将引脚配置为输入。
2. 将需要的值写入 LVDL 控制位（RCON<11:8>），该控制位可以用于选择需要的 LVD 阈值电压。
3. 通过清零 LVDIE 位（IEC2<10>）确保禁止了 LVD 中断。
4. 通过置位 LVDEN 位（RCON<12>）使能 LVD 模块。
5. 如果需要的话，查询 BGST 状态位（RCON<13>），等待内部参考电压稳定（参见第 9.1.2 节“内部参考电压”）。
6. 在允许中断前确保 LVDIF 位（IFS2<10>）清零。如果 LVDIF 位置 1，则器件 VDD 可能会低于选定的 LVD 阈值电压。
7. 通过写 LVDIP<2:0> 控制位（IPC10<10:8>）将 LVD 中断设置为需要的 CPU 优先级。
8. 通过置位 LVDIE 控制位允许 LVD 中断。

一旦 VDD 降低到低于编程的 LVD 阈值以下，LVDIF 位就将保持置位。当 LVD 模块中断了 CPU 时，ISR 中会采取以下两种动作之一：

1. 清零 LVDIE 控制位以禁止更多的 LVD 模块中断并采用相应的关闭步骤。
- 或
2. 使用 LVDL 控制位降低 LVD 电压阈值并清零 LVDIF 状态位。此方法可以用于跟踪逐渐减小的电池电压。

9.2.2 LVD 运行时的电流消耗

LVD 电路依赖于内部参考电压电路，该电路和其他外设器件，如欠压复位（BOR）模块共享。只要使能其中一个与它相关的外设，内部参考电压就会被激活。由于这个原因，LVD 模块被禁止时，用户可能观察不到预期的电流消耗变化。

9.2.3 在休眠和空闲模式下的工作原理

使能后，LVD 电路在休眠或空闲模式下将继续工作。如果器件电压越过了跳变点，LVDIF 位将会被置位。

从休眠或空闲模式退出的标准如下：

- 如果置位 LVDIE 位（IEC2<10>），器件将从休眠或空闲模式唤醒。
- 如果分配的 LVD 中断优先级小于或等于当前的 CPU 优先级，器件将被唤醒，并继续执行启动休眠或空闲模式的 PWSAV 指令的后一条指令代码。
- 如果分配的 LVD 中断优先级大于当前的 CPU 优先级，器件将被唤醒并且开始 CPU 异常处理程序。CPU 将从 LVD ISR 的第一条指令代码开始执行。

9.3 设计技巧

问 1: *LVD 电路似乎会产生随机中断?*

答: 在允许 LVD 中断之前确保内部参考电压稳定。这可通过在使能 LVD 模块之后查询 BGST 状态位 (RCON<13>) 判断内部参考电压是否稳定。延时结束之后, LVDIF 位应该被清零, 然后可以置位 LVDIE 位。

问 2: *如何降低模块的电流消耗?*

答: 使用低电压检测监控器件电压。电源通常是电压下降缓慢的电池。这意味着可以在大部分时间禁止 LVD 电路, 而且只在进行器件电压检测时偶尔使能 LVD 电路。

问 3: *在电池供电的应用中是否应该使能 BOR 电路?*

答: BOR 电路旨在使器件避免由于 AC 线电压的波动而造成错误操作。通常情况下采用电池供电的应用不需要 BOR 电路, 并且为了降低电流消耗, 可以禁用 BOR。

9.4 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 产品系列编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与低压检测模块相关的应用笔记有：

标题	应用笔记编号
目前没有相关的应用笔记。	

注： 如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

9.5 版本历史

版本 A

这是本文档的初始发行版。

版本 B

本章未包含任何技术内容或编辑排版上的修改，然而我们谨更新此节以反映本手册已更新到版本 B。

版本 C

本章未包含任何技术内容上的修改，然而我们谨更新此节以反映本手册已更新到版本 C。

第 10 章 看门狗定时器和低功耗模式

目录

本章包括下列主题：

10.1 简介	10-2
10.2 低功耗模式	10-2
10.3 休眠模式	10-2
10.4 空闲模式	10-4
10.5 低功耗指令与中断同时发生	10-5
10.6 看门狗定时器	10-6
10.7 外设模块禁止 (PMD) 寄存器	10-9
10.8 设计技巧	10-10
10.9 相关应用笔记	10-11
10.10 版本历史	10-12

10.1 简介

本章介绍 dsPIC30F 器件系列的看门狗定时器（WDT）和低功耗模式。dsPIC DSC 器件有两种低功耗模式，可以通过执行 PWRSAV 指令进入：

- 休眠模式：CPU、系统时钟源和任何依靠系统时钟源工作的外设都被禁止。这是器件的最低功耗模式。
- 空闲模式：CPU 被禁止，但是系统时钟源继续工作。外设继续工作，但可以有选择地禁止。

WDT 在使能时使用内部 LPRC 时钟源工作，而且如果 WDT 没有被软件清零，它可以通过复位器件来检测系统软件的异常情况。可以使用 WDT 后分频器选择不同的 WDT 超时周期。WDT 也可用于将器件从休眠或空闲模式唤醒。

10.2 低功耗模式

dsPIC30F 器件系列有两种特殊的低功耗模式，即休眠模式和空闲模式。可以通过执行特殊的 PWRSAV 指令进入这两种模式。

以下是 PWRSAV 指令的汇编语法：

```
PWRSAV #SLEEP_MODE      ; Put the device into SLEEP mode
PWRSAV #IDLE_MODE       ; Put the device into IDLE mode
```

注： SLEEP_MODE 和 IDLE_MODE 是所选器件的汇编器包含（Include）文件中定义的常数。

在中断使能、WDT 超时或器件复位时，器件会退出低功耗模式。当器件退出这两种操作模式之一时，称做“唤醒”。以下各节中将说明低功耗模式的特性。

10.3 休眠模式

休眠模式具有如下特性：

- 系统时钟源关闭。如果使用了片上振荡器，就将其关闭。
- 在没有 I/O 引脚输出电流的前提下，器件当前功耗最小。
- 由于系统时钟源被禁止，所以故障保护时钟监视器（FSCM）在休眠模式下不工作。
- 如果 WDT 被使能，LPRC 时钟将继续在休眠模式下运行。
- 低压检测电路如果被使能，则继续在休眠模式下保持工作。
- BOR 电路如果被使能，则继续在休眠模式下工作。
- WDT 如果被使能，则在进入休眠模式之前自动清零。
- 某些外设可能会继续在休眠模式运行。这些外设包括检测输入信号电平变化的 I/O 引脚，或使用外部时钟输入的外设。任何根据系统时钟源工作的外设都会在休眠模式禁止。

发生下列事件之一时，使用器将从休眠模式退出或被唤醒：

- 任何单独允许的中断源
- 任何形式的器件复位
- WDT 超时

10.3.1 从休眠模式唤醒时的时钟选择

处理器将重新启用在进入休眠模式之前有效的时钟源。

10.3.2 从休眠模式唤醒时的延时

表 10-1 所示为与从休眠模式唤醒相关的上电延时和振荡器起振延时。在所有情况下，均将有一段 POR 延时（标称值 TPOR = 10 μs），以使内部器件电路在内部系统复位信号 SYSRST 释放之前趋于稳定。

表 10-1: 从休眠模式退出的延时

时钟源	SYSRST 延时	振荡器 延时	FSCM 延时	注
EC、EXTRC	TPOR	—	—	1
EC + PLL	TPOR	TLOCK	TFSCM	1, 3, 4
XT + PLL	TPOR	TOST + TLOCK	TFSCM	1, 2, 3, 4
XT、HS、XTL	TPOR	TOST	TFSCM	1, 2, 4
LP（在休眠时关闭）	TPOR	TOST	TFSCM	1, 2, 4
LP（在休眠时开启）	TPOR	—	—	1
FRC、LPRC	TPOR	—	—	1

- 注 1: TPOR = 上电复位延时（标称值为 10 μs）。
2: TOST = 振荡器起振定时器延时。10 位计数器计数 1024 个振荡周期后，将振荡器时钟释放给系统使用。
3: TLOCK = PLL 锁定时间（标称值为 20 μs）。
4: TFSCM = 故障保护时钟监视器延时（标称值为 100 μs）

注： 有关 TPOR、TFSCM 和 TLOCK 的规范值，请参见 dsPIC30F 器件数据手册的“电气规范”一章。

10.3.3 在使用晶体振荡器或 PLL 模式下从休眠模式唤醒

如果系统时钟源来自晶体振荡器和 / 或 PLL，则在系统时钟源供器件使用之前必须有一段振荡器起振定时器（OST）和 / 或 PLL 锁定时间。作为该规则的一个特例，如果系统时钟源为 LP 振荡器且它在休眠模式运行就不需要振荡器延时。注意，尽管采用了多种不同的延时，在 POR 延时结束时晶体振荡器（和 PLL）不一定能起振和运行。

10.3.4 FSCM 延时和休眠模式

如果下列条件为真，当从休眠模式唤醒时，标称值为 100 μs 的延时（TFSCM）将在 POR 延时超时后加入：

- 在休眠模式中振荡器关闭。
- 系统时钟来自晶体振荡源和 / 或 PLL。

在多数情况下，在器件恢复指令执行前，FSCM 延时为 OST 超时和 PLL 提供足够延时。如果 FSCM 被使能，它将在 FSCM 延时超时后开始监控系统时钟源。

10.3.5 振荡器缓慢起振

当上电延时超时后，OST 和 PLL 锁定时间可能还没有超时。

如果 FSCM 被使能，器件将检测到此条件并将其作为一个时钟故障，然后产生时钟故障陷阱。器件将切换到 FRC 振荡器，用户可以在时钟故障陷阱服务程序中重新使能晶体振荡源。

如果 FSCM 未被使能，器件在时钟稳定之前不会开始执行代码。从用户角度来看，器件将处于休眠状态直到振荡器时钟起振。

10.3.6 中断时从休眠模式唤醒

CPU 优先级分配为零的用户中断源不能将 CPU 从休眠模式唤醒，因为此中断源被有效禁止了。要使用中断作为唤醒源，此中断的 CPU 优先级必须被分配为 1 或更高。

任何使用 IECx 寄存器中相应的 IE 控制位单独允许的中断源都可以将处理器从休眠模式唤醒。当器件从休眠模式唤醒时，会产生以下两种情况之一：

- 如果中断分配的优先级小于或等于当前 CPU 的优先级，器件将被唤醒并继续执行启动休眠模式的 PWRSAV 指令之后的指令代码。
- 如果中断源所分配的优先级别大于当前 CPU 的优先级，器件将被唤醒并开始进入 CPU 异常处理。代码将从 ISR 的第一条指令处继续执行。

休眠状态位（RCON<3>）在唤醒时被置 1。

10.3.7 复位时从休眠模式唤醒

所有的器件复位都会将处理器从休眠模式唤醒。任何唤醒处理器的复位源（除 POR 以外）都会置位休眠状态位（RCON<3>）以表明器件先前处于休眠模式。

在上电复位时，休眠位被清零。

10.3.8 在看门狗定时器超时从休眠模式唤醒

如果在器件处于休眠模式时看门狗定时器（WDT）被使能并超时，处理器将被唤醒。休眠和 WDTO 状态位（RCON<3>，RCON<4>）都被置位以表明器件由于 WDT 超时而恢复工作。注意，此事件不会使器件复位。器件从启动休眠模式的 PWRSAV 指令之后的指令继续运行。

10.4 空闲模式

CPU 优先级分配为零的用户中断源不能将 CPU 从空闲模式唤醒，因为此中断源被有效禁止了。要使用中断作为唤醒源，此中断的 CPU 优先级必须被分配为 1 或更高。

当器件进入空闲模式时，发生以下事件：

- CPU 将停止执行指令。
- WDT 被自动清零。
- 系统时钟源将保持有效，而且在默认情况下外设模块将使用系统时钟源继续正常工作。可以在空闲模式中使用“stop-in-idle”（在空闲模式停止）控制位有选择地关闭外设。（更多细节请参见外设说明。）
- 如果 WDT 或 FSCM 被使能，LPRC 也将保持有效。

在发生以下事件时，处理器将从空闲模式唤醒：

- 任何单独允许的中断
- 任何器件复位源
- WDT 超时。

在从空闲模式唤醒时，时钟再次供 CPU 使用且指令立即从 PWRSAV 指令之后的一条指令或 ISR 中的第一条指令开始执行。

10.4.1 中断时从空闲模式唤醒

任何使用 IECx 寄存器中相应的 IE 控制位单独允许且优先级大于当前 CPU 的优先级的中断都可以将处理器从空闲模式唤醒。当器件从空闲模式唤醒时会产生以下两种情况之一：

- 如果中断分配的优先级小于或等于当前 CPU 的优先级，器件将被唤醒并继续执行启动空闲模式的 PWRSAV 指令之后的指令代码。
- 如果为中断源分配的优先级别大于当前 CPU 的优先级，器件将被唤醒并开始进行 CPU 异常处理。代码将从 ISR 的第一条指令处开始执行。

空闲状态位（RCON<2>）在唤醒时被置 1。

10.4.2 复位时从空闲模式唤醒

任何复位（除 POR 以外），都会将 CPU 从空闲模式唤醒。在除 POR 之外的所有器件复位时，空闲状态位被置 1（RCON<2>）以表明器件先前处于空闲模式。在上电复位时，空闲位被清零。

10.4.3 WDT 超时从空闲模式唤醒

如果 WDT 被使能，处理器将在 WDT 超时从空闲模式唤醒并继续执行启动空闲模式的 PWRSAV 指令之后的指令代码。注意在这种情况下 WDT 超时不会使器件复位。WDTO 和空闲状态位（RCON<4>，RCON<2>）都会被置 1。

10.4.4 从空闲模式唤醒的延时

与从休眠模式唤醒不同的是，不存在与从空闲模式唤醒相关的延时。系统时钟在空闲模式时仍然运行，因此在唤醒时不需要起振时间。

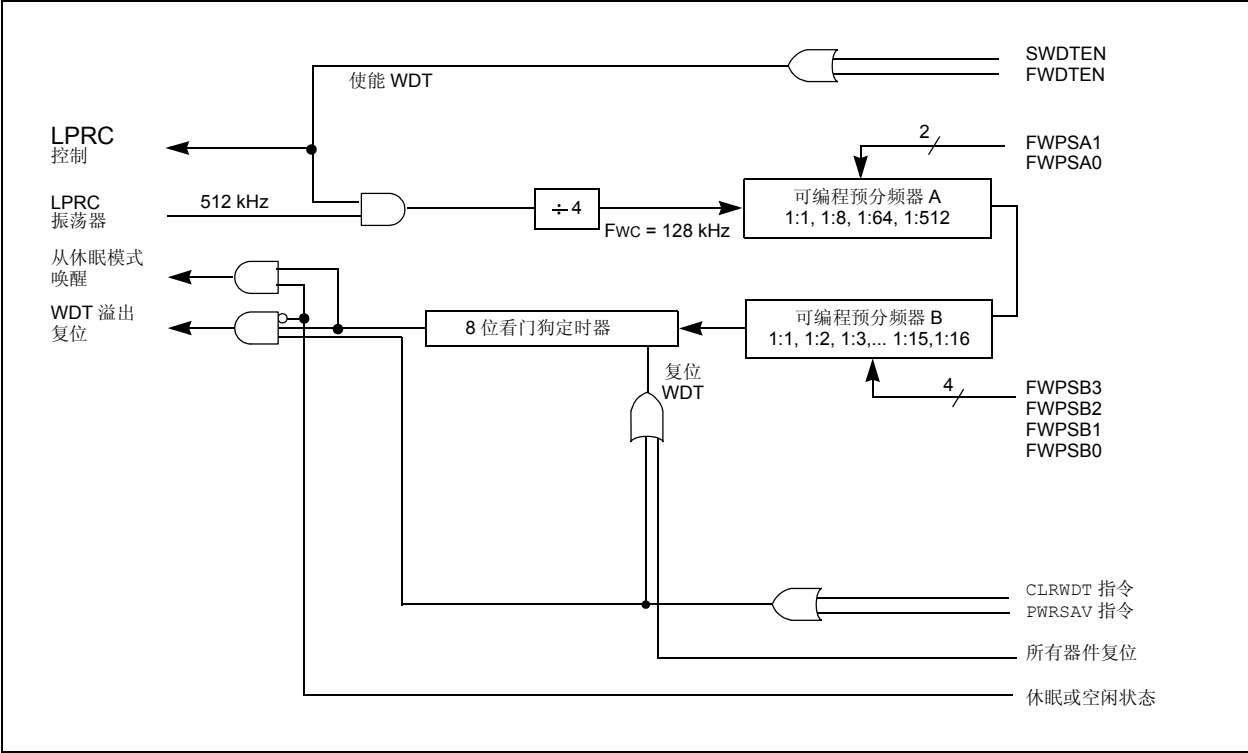
10.5 低功耗指令与中断同时发生

任何与 PWRSAV 指令执行同时产生的中断都将延时响应，直到完成进入休眠或空闲模式。然后器件将从休眠或空闲模式唤醒。

10.6 看门狗定时器

看门狗定时器（WDT）的主要功能是在出现软件异常事件时复位处理器。WDT 是自由运行的定时器，它在 LPRC 振荡器上运行而不需要外部组件。因此，即使系统时钟源（例如，晶体振荡器）出现故障，WDT 定时器仍然会继续工作。图 10-1 所示为 WDT 的框图。

图 10-1: WDT 框图



10.6.1 使能和禁止 WDT

WDT 的使能或禁止由 FWDT 器件配置寄存器中的 FWDTEN 器件配置位控制。FWDT 寄存器的值在器件编程时被写入。当 FWDTEN 配置位置 1 时，WDT 被使能。这是一个已被擦除的器件的默认值。FWDT 器件配置寄存器的更多细节请参见第 24 章“器件配置”。

10.6.1.1 软件控制的 WDT

如果 FWDTEN 器件配置位置 1 时，则 WDT 总是被使能的。但是，当 FWDTEN 配置位被编程为“0”后，可以选择用用户软件控制 WDT。

通过置位 SWDTEN 控制位（RCON<5>）用软件使能 WDT。在任何器件复位时 SWDTEN 控制位都会被清零。软件 WDT 选项允许用户在关键代码段使能 WDT 并在非关键代码段禁止 WDT，以最大限度地降低功耗。

10.6.2 WDT 工作

如果被使能，WDT 将进行加计数直到溢出或“超时”。除了在休眠或空闲模式，WDT 超时将强制器件复位。要阻止 WDT 超时复位，用户必须使用 CLRWDT 指令定期清零看门狗定时器。CLRWDT 指令也将清零 WDT 预分频器。

如果 WDT 在休眠或空闲模式超时，器件将唤醒并从 PWRSV 指令处开始继续执行代码。

在这两种情况下，WDTO 位 (RCON<4>) 将被置 1 以表明器件复位或唤醒事件是由于 WDT 超时而产生的。如果 WDT 将 CPU 从休眠或空闲模式唤醒，休眠状态位 (RCON<3>) 或空闲状态位 (RCON<2>) 也将置 1 以表明器件先前处于低功耗模式。

10.6.3 WDT 定时器周期选择

WDT 的时钟源是内部 LPRC 振荡器，它的标称频率为 512 kHz。LPRC 时钟被进一步 4 分频以便为 WDT 提供 128 kHz 时钟。WDT 的计数器为 8 位宽，因此 WDT 的标称超时周期 (TWDT) 为 2 ms。

10.6.3.1 WDT 预分频器

WDT 有两个时钟预分频器，预分频器 A 和预分频器 B，他们的超时周期范围很广。预分频器 A 可以配置为 1:1、1:8、1:64 或 1:512 的分频比。预分频器 B 可以配置为从 1:1 到 1:16 的任何分频比。使用这些预分频器可以得到范围在 2 ms 到 16 秒（标称值）之间的超时周期。

使用 FWDT 器件配置寄存器中的 FWPSA<1:0>（预分频器 A）和 FWPSB<3:0>（预分频器 B）配置位选择预分频器设置。FWPSA<1:0> 和 FWPSB<3:0> 值是在器件编程时写入的。有关 WDT 预分频器配置位的更多信息，请参见第 24 章“器件配置”。

可以用以下方法计算 WDT 超时周期：

公式 10-1: WDT 超时周期

WDT 周期 = 2 ms • 预分频比 A • 预分频比 B

注： WDT 超时周期与 LPRC 振荡器的频率直接相关。LPRC 振荡器的频率随着器件工作电压和温度的变化而变化。LPRC 时钟频率规范请参见特定的 dsPIC30F 器件数据手册。

表 10-2 所示为不同预分频器选择的超时周期：

表 10-2: 不同预分频 A 和预分频 B 设置的 WDT 超时周期

预分频器 B 值	预分频器 A 值			
	1	8	64	512
1	2	16	128	1024
2	4	32	256	2048
3	6	48	384	3072
4	8	64	512	4096
5	10	80	640	5120
6	12	96	768	6144
7	14	112	896	7168
8	16	128	1024	8192
9	18	144	1152	9216
10	20	160	1280	10240
11	22	176	1408	11264
12	24	192	1536	12288
13	26	208	1664	13312
14	28	224	1792	14336
15	30	240	1920	15360
16	32	256	2048	16384

注：所有时间值都以 ms 为单位。

10.6.4 复位看门狗定时器

WDT 和它的所有预分频器在以下事件复位：

- 在任何器件复位时
- 执行 PWRSAV 指令时（即，进入休眠或空闲模式）
- 当在正常执行过程中，使用 CLRWDT 指令

10.6.5 WDT 在休眠或空闲模式的工作

如果 WDT 使能，它将在休眠或空闲模式继续运行。当发生 WDT 超时，器件将唤醒且代码执行将从执行 PWRSAV 指令处开始继续执行。

WDT 对于低功耗系统设计非常有用，因为它可用于定期将器件从休眠模式唤醒，以便检查系统状态并采取必要的措施。注意 SWDTEN 位在这方面很有用处。如果在正常运行（FWDTEN = 0）过程中 WDT 被禁止，则可以使用 SWDTEN 位（RCON<5>）在进入休眠模式之前打开 WDT。

10.7 外设模块禁止（PMD）寄存器

外设模块禁止（PMD）寄存器提供了一种禁止外设模块的方法，它可通过停止某一模块的所有时钟源来禁止该外设模块。通过适当的 PMD 控制位禁止外设时，该外设处于最小功耗状态。与该外设相关的控制和状态寄存器也将被禁止，因此向这些寄存器执行写操作将不起作用，而读操作则无效。

使能一个外设模块的惟一方法是将 PMD 寄存器中的相关位清零，同时相应 dsPIC 器件支持该外设。如果器件中有该外设，则在缺省情况下，它在 PMD 寄存器中被使能。

注： 如果某个 PMD 位置 1，相应模块在延时一个指令周期后被禁止。同样地，如果某个 PMD 位被清零，相应模块在延时一个指令周期后被使能（假设模块控制寄存器已配置为使能模块操作）。

PMD 寄存器的具体操作详情，请查阅器件数据手册。

10.8 设计技巧

问 1: *尽管我在软件主循环中插入了一个 CLRWDT 指令，但器件仍然复位了。*

答: 确定包含 CLRWDT 指令的软件循环符合 WDT 最小值规范（不是典型值）。而且，确定考虑到了中断处理时间。

问 2: *软件在进入休眠或空闲模式之前应该做些什么？*

答: 确信将要唤醒器件的源的 IE 位置 1。此外，确保特定的中断源可以唤醒器件。当器件处于休眠模式时，某些中断源不工作。

如果器件将要进入空闲模式，确定每个器件外设的“Stop-in-Idle”（在空闲模式停止）控制位被正确置 1。这些控制位确定外设是否继续在空闲模式工作。更多细节请参见本手册各个外设的相关章节。

问 3: *如何分辨哪个外设能将器件从休眠或空闲模式唤醒？*

答: 您可以查询每个允许的中断源的 IF 位，确定唤醒源。

10.9 相关应用笔记

本节列举了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是针对 dsPIC30F 产品系列而编写的，但是概念是相关的，通过适当修改即可使用，但使用中可能会受到一定限制。当前与看门狗定时器和低功耗模式相关的应用笔记有：

标题	应用笔记编号
Low Power Design using PIC® Microcontrollers	AN606

注：如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

10.10 版本历史

版本 A

这是本文档的初始发行版。

版本 B

本章未包含任何技术内容或编辑排版上的修订，然而我们谨更新此节以反映本手册已更新到版本 B。

版本 C

此版本加入了本文档更新时已知的所有勘误表。

版本 D

增加了第 10.7 节“外设模块禁止（PMD）寄存器”。

注:

注:

第 11 章 I/O 端口

目录

本章包括下列主题：

11.1 简介	11-2
11.2 I/O 端口控制寄存器	11-3
11.3 外设复用	11-4
11.4 端口描述	11-6
11.5 电平变化通知 (CN) 引脚	11-7
11.6 休眠和空闲模式下的 CN 工作	11-8
11.7 相关应用笔记	11-11
11.8 版本历史	11-12

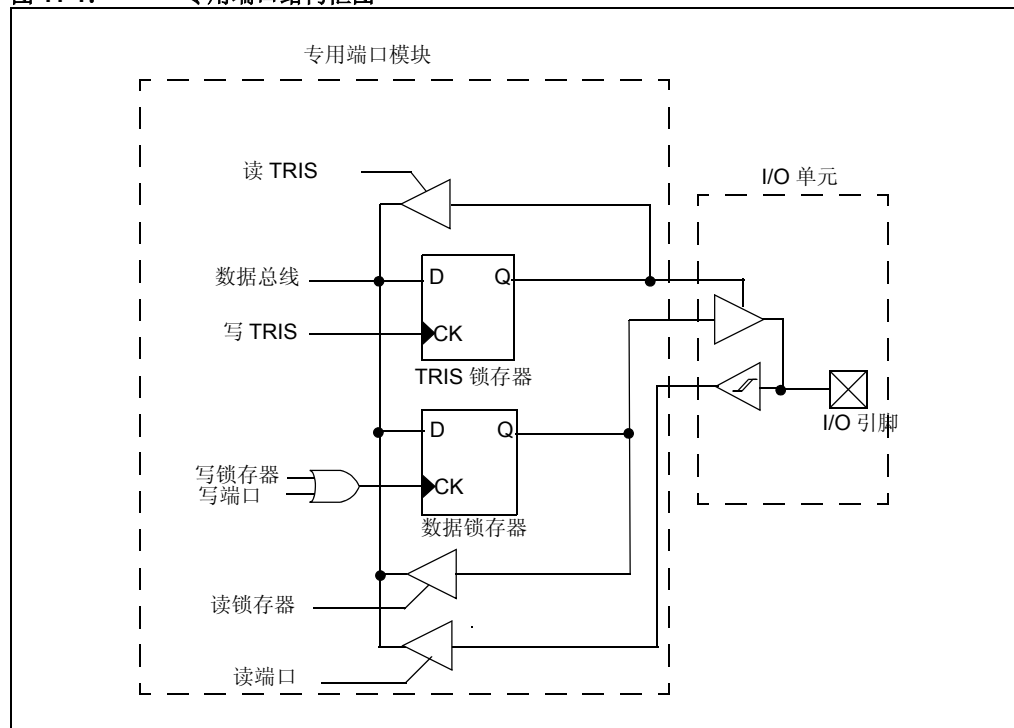
11.1 简介

本章提供 dsPIC30F 系列器件 I/O 端口的信息。器件的所有引脚（除 VDD、VSS、MCLR 和 OSC1/CLKI 以外）均由外设和通用 I/O 端口共用。

通用 I/O 端口可供 dsPIC30F 监视和控制其他器件。大多数 I/O 引脚与备用功能复用。复用将取决于不同器件上的外设功能部件。一般来说，当相应的外设使能时，其对应的引脚将不再作为通用 I/O 引脚使用。

图 11-1 所示为典型 I/O 端口的框图。该框图没有画出 I/O 引脚上可能复用的外设功能。

图 11-1: 专用端口结构框图



11.2 I/O 端口控制寄存器

所有 I/O 端口都有三个与该端口的工作直接相关的寄存器，其中字母“x”表示指定的 I/O 端口号。

- TRISx: 数据方向寄存器
- PORTx: I/O 端口寄存器
- LATx: I/O 锁存寄存器

器件上的每个 I/O 引脚在 TRIS、PORT 和 LAT 寄存器中都分别有一个相关的位。

注：端口和可用 I/O 引脚的总数将取决于不同的器件。在一个给定的器件中，可能并没有实现端口控制寄存器中的所有位。更多详细信息，请参见特定的器件数据手册。

11.2.1 TRIS 寄存器

TRISx 寄存器控制位决定与该 I/O 端口相关的各个引脚是输入引脚还是输出引脚。若某个 I/O 引脚的 TRIS 位为“1”，则该引脚是输入引脚。若某个 I/O 引脚的 TRIS 位为“0”，则该引脚被配置为输出引脚。这很好记，因为 1 很像 I (Input, 输入)，0 很像 O (Output, 输出)。复位以后，所有端口引脚被定义为输入。

11.2.2 PORT 寄存器

通过 PORTx 寄存器访问 I/O 引脚上的数据。读 PORTx 寄存器是读取 I/O 引脚上的值，而写 PORTx 寄存器是将值写入端口数据锁存器。

很多指令，如 BSET 和 BCLR 指令，都是读—修改—写操作指令。因此，写一个端口就意味着读该端口的引脚电平，修改读到的值，然后再将改好的值写入端口数据锁存器。当与端口相关的一些 I/O 引脚被配置为输入时，并且在 PORTx 寄存器上使用读—修改—写命令时应该特别小心。如果某个配置为输入的 I/O 引脚在过了一段时间后变成输出引脚，则该 I/O 引脚上将会输出一个不期望的值。产生这种情况的原因是读—修改—写指令读取了输入引脚上的瞬时值，并将该值装入端口数据锁存器。

11.2.3 LAT 寄存器

与 I/O 引脚相关的 LATx 寄存器消除了可能在执行读—修改—写指令过程中发生的问题。读 LATx 寄存器将返回保存在端口输出锁存器中的值，而不是 I/O 引脚上的值。对与某个 I/O 端口相关的 LAT 寄存器进行读—修改—写操作，避免了将输入引脚值写入端口锁存器的可能性。写 LATx 寄存器与写 PORTx 寄存器的结果相同。

PORT 和 LAT 寄存器之间的差异可以归纳如下：

- 写 PORTx 寄存器就是将数据值写入该端口锁存器。
- 写 LATx 寄存器就是将数据值写入该端口锁存器。
- 读 PORTx 寄存器就是读取 I/O 引脚上的数据值。
- 读 LATx 寄存器就是读取保存在该端口锁存器中的数据值。

对某个器件无效的任何位以及与其相关的数据和控制寄存器都将被禁止。这意味着对应的 LATx 和 TRISx 寄存器以及该端口引脚将读作 0。

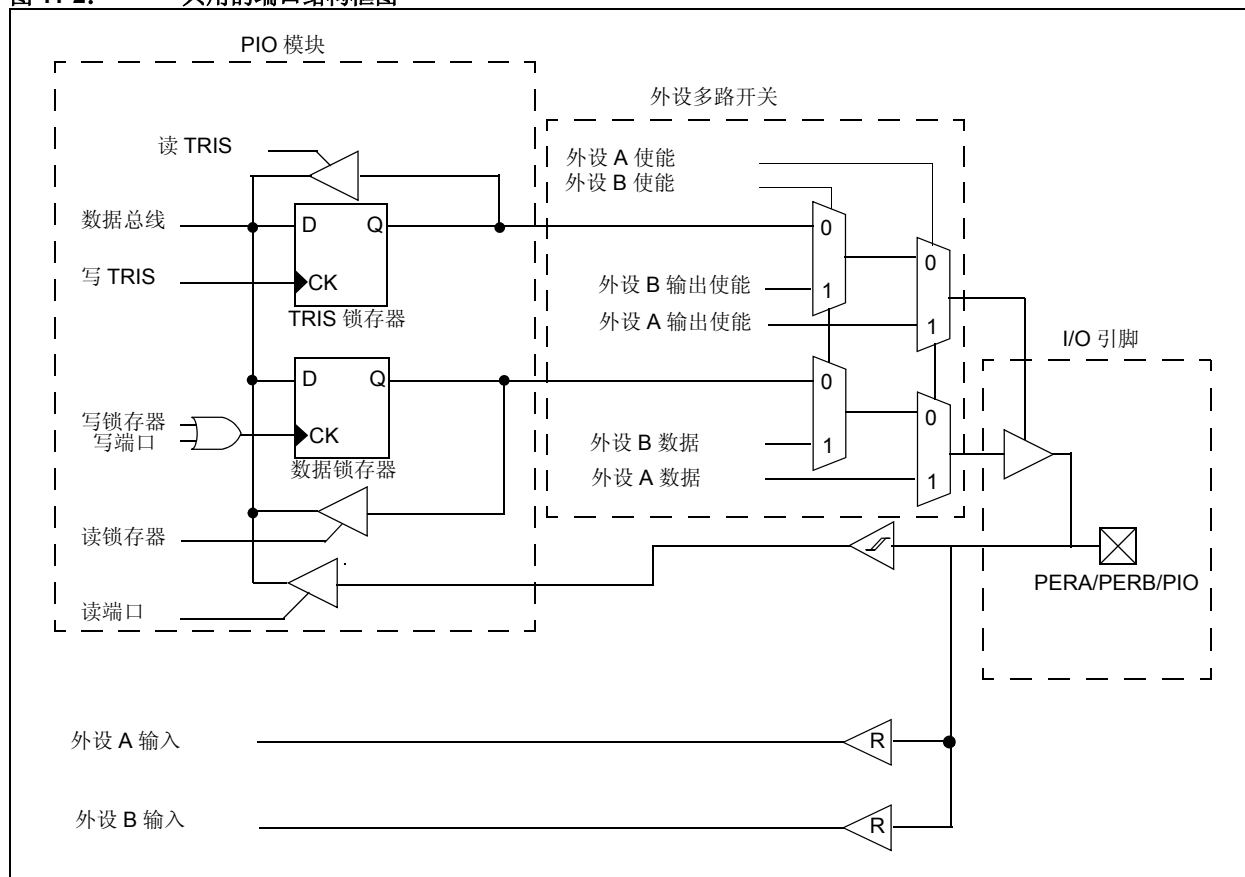
11.3 外设复用

当某个外设使能时，相关引脚的输出驱动器通常由模块控制，而有些可由用户设置。可以通过输入数据路径读该 I/O 引脚，但该 I/O 端口位的输出驱动器一般被禁止。

与另一个外设共用一个引脚的 I/O 端口总是服从于该外设。外设的输出缓冲器数据和控制信号提供一对多路开关。该多路开关选择是外设还是相关的端口拥有输出数据的所有权以及 I/O 引脚的控制信号。图 11-2 显示了端口如何与其他外设共用，以及与外设连接的相关 I/O 引脚。

注： 为了将 PORTB 引脚用作数字 I/O，ADPCFG 寄存器中的相应位必须置为“1”（即使关闭了 A/D 模块也应如此）。

图 11-2: 共用的端口结构框图



11.3.1 I/O 与多个外设复用

对于有些 dsPIC30F 器件，尤其是那些带有少量 I/O 引脚数较少的器件，其每个 I/O 引脚可能要复用多种外设功能。图 11-2 所示为两个外设与同一个 I/O 引脚复用的示例。

I/O 引脚的名称定义了与该引脚相关的各个功能的优先级。如图 11-2 所示，概念化的 I/O 引脚与两个外设复用（“外设 A”和“外设 B”），并命名为“PERA/PERB/PIO”。

已为 I/O 引脚选择了适当的名称，以使用户可以方便地识别分配给该引脚的功能的优先级。对于图 11-2 中的示例，外设 A 对引脚的控制具有最高优先权。若外设 A 和外设 B 同时使能，外设 A 将控制 I/O 引脚。

11.3.1.1 软件输入引脚控制

分配给某个 I/O 引脚的一些功能可能是那些不控制引脚输出驱动器的输入功能。这类外设的一个示例就是输入捕捉模块。如果使用相应的 TRIS 控制位将与输入捕捉相关的 I/O 引脚配置为输出引脚，用户可以通过其相应的 PORT 寄存器手动影响输入捕捉引脚的状态。这种做法在有些情形下很有用，尤其适用于当没有外部信号连接到输入引脚的情况下进行测试。

参见图 11-2，外设多路开关的结构将决定外设输入引脚是否可以通过使用 PORT 寄存器用软件控制。当图中所示的概念化的外设使能时，会断开 I/O 引脚与端口数据的连接。

一般而言，下列外设允许通过 PORT 寄存器手动控制它们的输入引脚：

- 外部中断引脚
- 定时器时钟输入引脚
- 输入捕捉引脚
- PWM 故障引脚

大多数串行通信外设使能时，将完全控制 I/O 引脚，所以不能通过相应的 PORT 寄存器影响与该外设相关的输入引脚。这些外设列举如下：

- SPI™
- I²C™
- DCI
- UART
- CAN

11.3.1.2 引脚控制概述

当某个外设使能时，相关引脚的输出驱动器通常由模块控制，而有些可由用户设置。“模块控制”（Module Control）是指相关端口的引脚输出驱动器被禁止，并且该引脚只能由外设控制或访问。“用户可设置”（User Settable）是指相关外设端口的引脚输出驱动器可由用户通过相关的 TRISx SFR 进行配置。必须正确设置寄存器 TRISx 以使外设正常工作。对于“用户可设置”的外设引脚，实际的端口引脚状态总是可通过 PORTx SFR 读取。

输入捕捉外设就是一个很好的用户可设置外设的例子。用户必须写入相关的 TRIS 寄存器，将输入捕捉引脚配置为输入。当输入捕捉使能时，由于 I/O 引脚电路依然是激活的，利用软件和一个“窍门”来手工产生一个捕捉事件。利用相关 TRIS 寄存器将输入捕捉引脚配置为输出。之后，软件即可向相应的 LAT 驱动寄存器写入值，以对输入捕捉引脚进行内部控制并强制产生捕捉事件。

另一个例子是，可将一个 INTx 引脚配置为输出，然后通过写入相关的 LATx 位即可产生 INTx 中断（使能时）。

UART 就是一个模块控制外设的例子。当 UART 使能时，寄存器 PORT 和 TRIS 不起作用，不能用于读写 RX 和 TX 引脚。dsPIC 上所具备的大多数通讯外设都是模块控制外设。

例如，SPI 模块可配置为主模式，而主模式下只需用到 SDO 引脚。在这种情况下，清零（置为逻辑 0）相关的 TRISx 位即可将 SDI 引脚配置为通用输出引脚。表 11-1 对 dsPIC 的外设以及相关引脚输出控制和端口引脚读取状态作了小结。

表 11-1: 端口引脚控制小结表

外设模块 “使能状态”	外设引脚	TRISx —— 引脚输入控制	PORTx —— 引脚读取
SPI™ (x = 1 或 2)	SDOx	<DISSDO = 1>, 用户可设置 <DISSDO = 0>, 模块控制	是
	SDIx	用户可设置	是
	SCKx	模块控制	是
	SSx	<SSEN = 0>, 用户可设置 <SSEN = 1>, 模块控制	是
UART (x = 1 或 2)	UxRX	模块控制	是
	UxTX	<UTXEN = 0>, 用户可设置 <UTXEN = 1>, 模块控制	是
I²C™	SCL	模块控制	是
	SDA	模块控制	是
输入电平变化通知	CN0 - CN23	用户可设置	是
输入捕捉	IC1 - IC8	用户可设置	是
输出比较	OC1 - OC8	模块控制	是
数据转换器接口	COFS	模块控制	是
	CCLK	模块控制	是
	CSDI	模块控制	是
	CSDO	模块控制	是
电机控制 PWM	PWMx	模块控制	是
	FLTA/B	用户可设置	是
QEI	QEA	模块控制 (QEI 模式) 用户可设置 (16 位定时器模式)	是
	QEB	模块控制 (QEI 模式) 用户可设置 (16 位定时器模式)	是
	INDX	模块控制 (QEIM<2:0>= 100 或 110) 所有其他模式均为用户可设置	是
CAN (x = 1 或 2)	CxRX	模块控制	是
	CxTX	模块控制	是
INTx	INT0 — INT5	用户可设置	是

11.4 端口描述

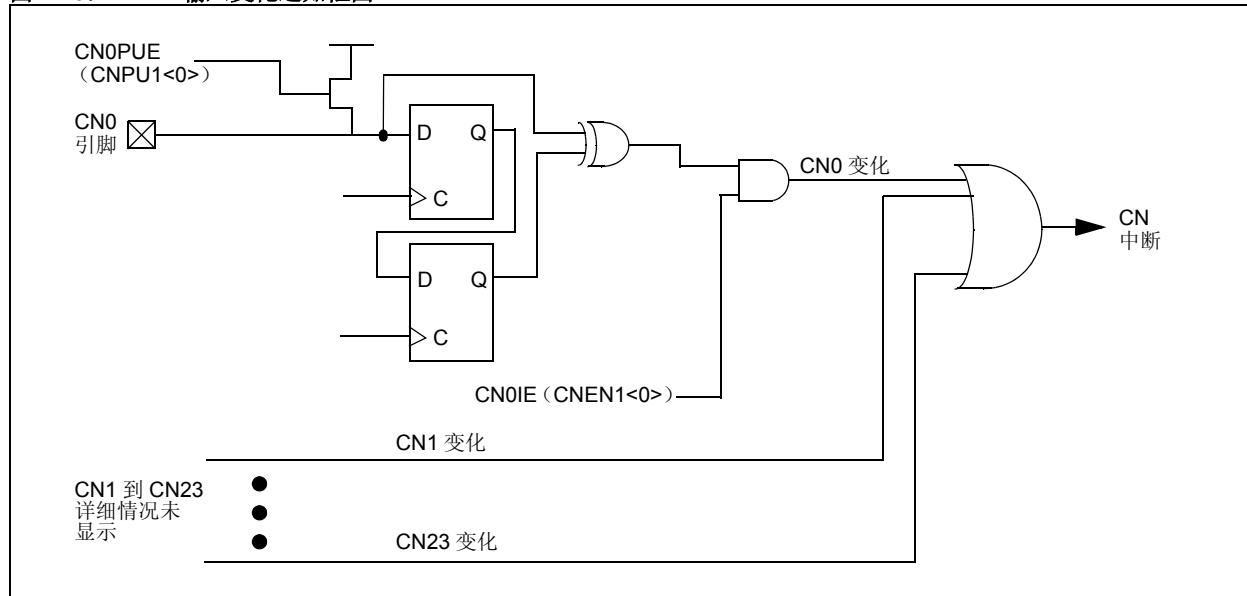
可用的 I/O 端口和外设复用详情的介绍，请参见器件数据手册。

11.5 电平变化通知 (CN) 引脚

电平变化通知 (Change Notification, CN) 引脚使 dsPIC30F 器件能够向处理器发出中断请求, 以响应所选择的输入引脚上的状态变化。可以选择 (使能) 多达 24 个输入引脚来产生 CN 中断。可用的 CN 输入引脚总数取决于所选的 dsPIC30F 器件。更多详细信息请参见器件数据手册。

图 11-3 显示了 CN 硬件的基本功能。

图 11-3: 输入变化通知框图



11.5.1 CN 控制寄存器

有四个与 CN 模块相关的控制寄存器。CNEN1 和 CNEN2 寄存器包含 CNxIE 控制位，其中“x”表示 CN 输入引脚的编号。要让某个 CN 输入引脚中断 CPU，必须将其 CNxIE 位置 1。

CNPU1 和 CNPU2 寄存器包含 CNxPUE 控制位。每个 CN 引脚连接着一个弱上拉器件，该器件可以通过 CNxPUE 控制位使能或禁止。弱上拉器件充当连接到该引脚的电流源，并且当连接了按钮或键盘设备时，有了该器件即可不再需要外部电阻。关于 CN 上拉器件的电流规范，请参见器件数据手册的“电气规范”部分。

11.5.2 CN 的配置和操作

CN 引脚配置如下：

1. 通过将 TRISx 寄存器中的相关位置 1，确保 CN 引脚配置为数字输入引脚。
2. 通过将 CNEN1 和 CNEN2 寄存器中的相应位置 1，允许所选择的 CN 引脚中断。
3. 如果想打开所选择的 CN 引脚的弱上拉器件，请将 CNPU1 和 CNPU2 寄存器中的相应位置 1。
4. 清零中断标志位 CNIF (IFS0<15>)。
5. 使用 CNIP<2:0> 控制位 (IPC3<14:12>) 为 CN 中断选择所需的中断优先级。
6. 使用 CNIE (IEC0<15>) 控制位允许 CN 中断。

当 CN 中断发生时，用户应该读与该 CN 引脚相关的 PORT 寄存器。这样做将清除引脚电平不匹配条件，并设置 CN 逻辑电路以检测下一次引脚电平变化。可以将当前的端口值与上一次 CN 中断时得到的端口读出值比较，来确定发生过变化的引脚。

CN 引脚有最小输入脉宽规范。更多详细信息，请参见器件数据手册的“电气规范”部分。

11.6 休眠和空闲模式下的 CN 工作

CN 模块在休眠或空闲模式下继续工作。如果使能的 CN 引脚中的一个改变了状态，CNIF (IFS0<15>) 状态位将被置 1。如果置位 CNIE 位 (IEC0<15>)，器件将从休眠或空闲模式唤醒并恢复工作。

如果为 CN 中断分配的优先级等于或低于当前 CPU 的优先级，器件会从紧随 SLEEP 或 IDLE 指令后的那条指令继续执行。

如果为 CN 中断分配的优先级高于当前 CPU 的优先级，器件将从 CN 中断向量地址继续执行。

寄存器 11-1: CNEN1: 输入变化通知中断使能寄存器 1

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN15IE	CN14IE	CN13IE	CN12IE	CN11IE	CN10IE	CN9IE	CN8IE
bit 15							bit 8
低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN7IE	CN6IE	CN5IE	CN4IE	CN3IE	CN2IE	CN1IE	CN0IE
bit 7							bit 0

bit 15-0 CNxIE: 输入变化通知中断使能位
1= 允许输入电平变化中断
0= 禁止输入电平变化中断

图注:			
R= 可读位	W= 可写位	U= 未用位, 读作 0	
-n= POR 时的值	1= 置位	0= 清零	x= 未知

寄存器 11-2: CNEN2: 输入变化通知中断使能寄存器 2

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	—	—	—
bit 15							bit 8
低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN23IE	CN22IE	CN21IE	CN20IE	CN19IE	CN18IE	CN17IE	CN16IE
bit 7							bit 0

bit 15-8 未用: 读作 0
bit 7-0 CNxIE: 输入变化通知中断使能位
1= 允许输入电平变化中断
0= 禁止输入电平变化中断

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = 上电复位时的值	1 = 置位	0 = 清零	x= 未知

dsPIC30F 系列参考手册

寄存器 11-3: CNPU1: 输入变化通知上拉使能寄存器 1

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN15PUE	CN14PUE	CN13PUE	CN12PUE	CN11PUE	CN10PUE	CN9PUE	CN8PUE
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN7PUE	CN6PUE	CN5PUE	CN4PUE	CN3PUE	CN2PUE	CN1PUE	CN0PUE
bit 7				bit 0			

bit 15-0 **CNxPUE**: 输入变化通知上拉使能位

1= 使能输入电平变化上拉

0= 禁止输入电平变化上拉

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

寄存器 11-4: CNPU2: 输入变化通知上拉使能寄存器 2

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN23PUE	CN22PUE	CN21PUE	CN20PUE	CN19PUE	CN18PUE	CN17PUE	CN16PUE
bit 7				bit 0			

bit 15-8 未用: 读作 0

bit 7-0 **CNxPUE**: 输入变化通知上拉使能位

1= 使能输入电平变化上拉

0= 禁止输入电平变化上拉

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

11.7 相关应用笔记

本节列出了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。目前与 I/O 端口模块相关的应用笔记有：

标题	应用笔记编号
Implementing Wake-up on Key Stroke	AN552

注：如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

11.8 版本历史

版本 A

这是本文档的初始发行版。

版本 B

此版本为 dsPIC30F I/O 端口模块添加了额外的技术内容。

版本 C

本章未包含任何技术内容上的修订，然而我们谨更新此节以反映本手册已更新到版本 C。

版本 D

此版本增加了第 11.3.1.2 节“引脚控制概述”。

第 12 章 定时器

目录

本章包括下列主题：

12.1 简介	12-2
12.2 不同的定时器	12-3
12.3 控制寄存器	12-6
12.4 工作模式	12-9
12.5 定时器预分频器	12-14
12.6 定时器中断	12-14
12.7 读写 16 位定时器模块寄存器	12-15
12.8 低功耗 32 kHz 晶振输入	12-15
12.9 32 位定时器配置	12-16
12.10 32 位定时器的工作模式	12-18
12.11 读写 32 位定时器	12-21
12.12 低功耗状态下的定时器工作	12-21
12.13 使用定时器模块的外设	12-22
12.14 设计技巧	12-24
12.15 相关应用笔记	12-25
12.16 版本历史	12-26

12.1 简介

视具体器件，dsPIC30F 器件系列提供了几个 16 位定时器。这些定时器被指定为 Timer1、Timer2、Timer3……等。

每个定时器模块均为 16 位定时器 / 计数器，由下列可读 / 写寄存器组成：

- TMRx: 16 位定时器计数寄存器
- PRx: 与该定时器相关的 16 位周期寄存器
- TxCON: 与该定时器相关的 16 位控制寄存器

每个定时器模块还有与中断控制相关的位：

- 中断使能控制位 (TxIE)
- 中断标志状态位 (TxIF)
- 中断优先级控制位 (TxIP<2:0>)

除了某些例外的功能之外，所有 16 位定时器都有相同的功能电路。16 位定时器分为三种类型以说明其功能上的差异：

- A 类型时基
- B 类型时基
- C 类型时基

有些 16 位定时器可以组合成为 32 位定时器。

本章不介绍与外设器件相关的专用定时器。例如，其中包括与电机控制 PWM 模块和正交编码器接口 (QEI) 模块相关的时基。

12.2 不同的定时器

除了某些例外之处，dsPIC30F 器件上所有可用的 16 位定时器在功能上是相同的。这些 16 位定时器分为三种功能类型：A 类型定时器、B 类型定时器和 C 类型定时器。

注: 有关可用的定时器和每个定时器的类型的信息, 请参见器件数据手册。

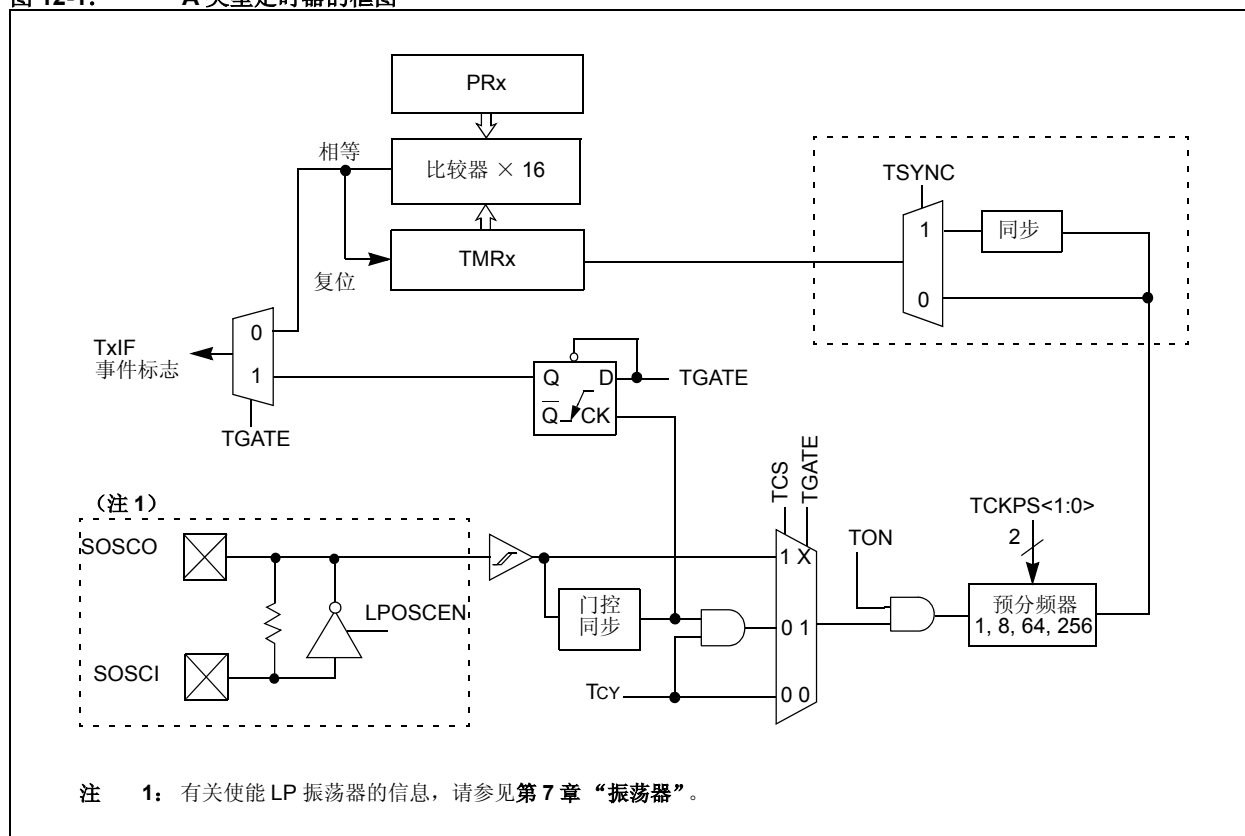
12.2.1 A 类型定时器

在大多数 dsPIC30F 器件上,至少有一个 A 类型定时器。对于大多数 dsPIC30F 器件,Timer1 是 A 类型定时器。A 类型定时器与其他类型的定时器相比,有下列独特的功能:

- 可以使用器件的低功耗 **32 kHz** 振荡器作为时钟源工作
- 可以在使用外部时钟源的异步模式下工作

特别值得一提的是，A 类型定时器独特的功能使它可以用于实时时钟（Real-Time Clock，RTC）应用。图 12-1 所示为 A 类型定时器的框图。

图 12-1: A 类型定时器的框图



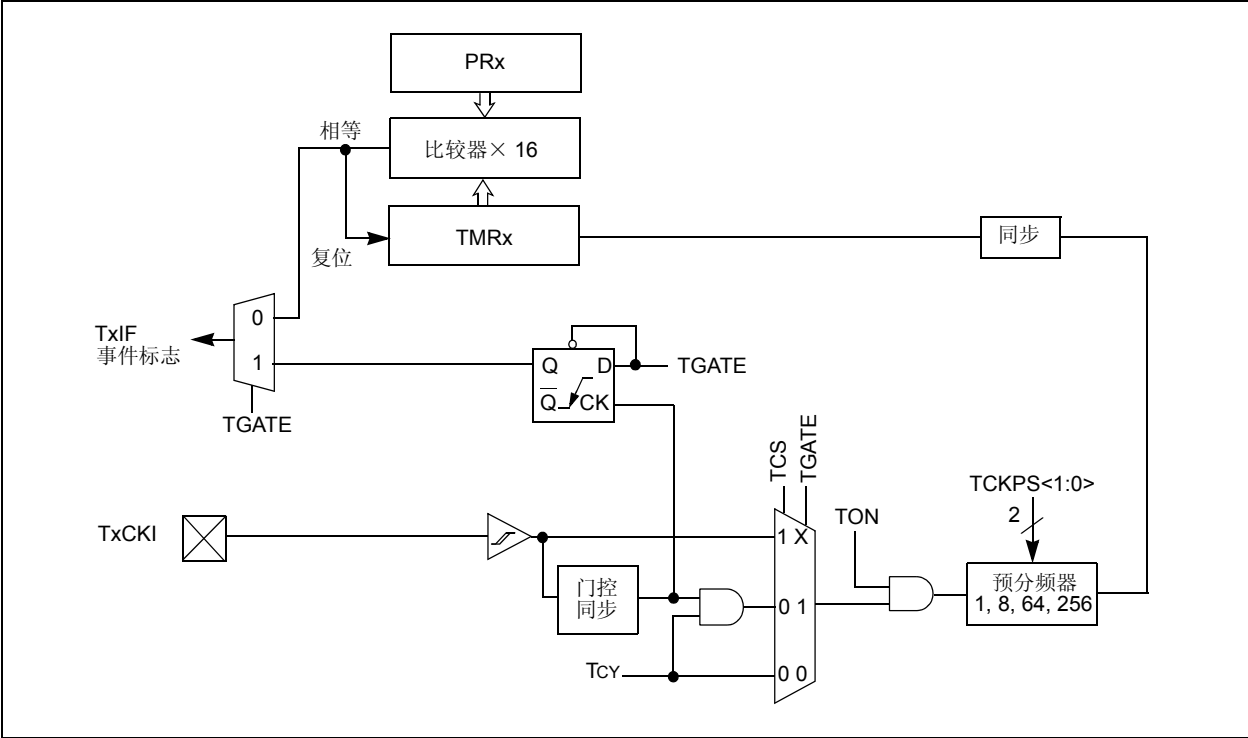
12.2.2 B 类型定时器

在大多数 dsPIC30F 器件上，如果存在 Timer2 和 Timer4，它们是 B 类型定时器。与其他类型的定时器相比，B 类型定时器有下列独特的功能：

- B 类型定时器可以和 C 类型定时器相连形成 32 位定时器。B 类型定时器的 TxCON 寄存器具备 T32 控制位，用来使能 32 位定时器功能。
- B 类型定时器的时钟同步在预分频逻辑后执行。

图 12-2 所示为 B 类型定时器的框图。

图 12-2: B 类型定时器的框图



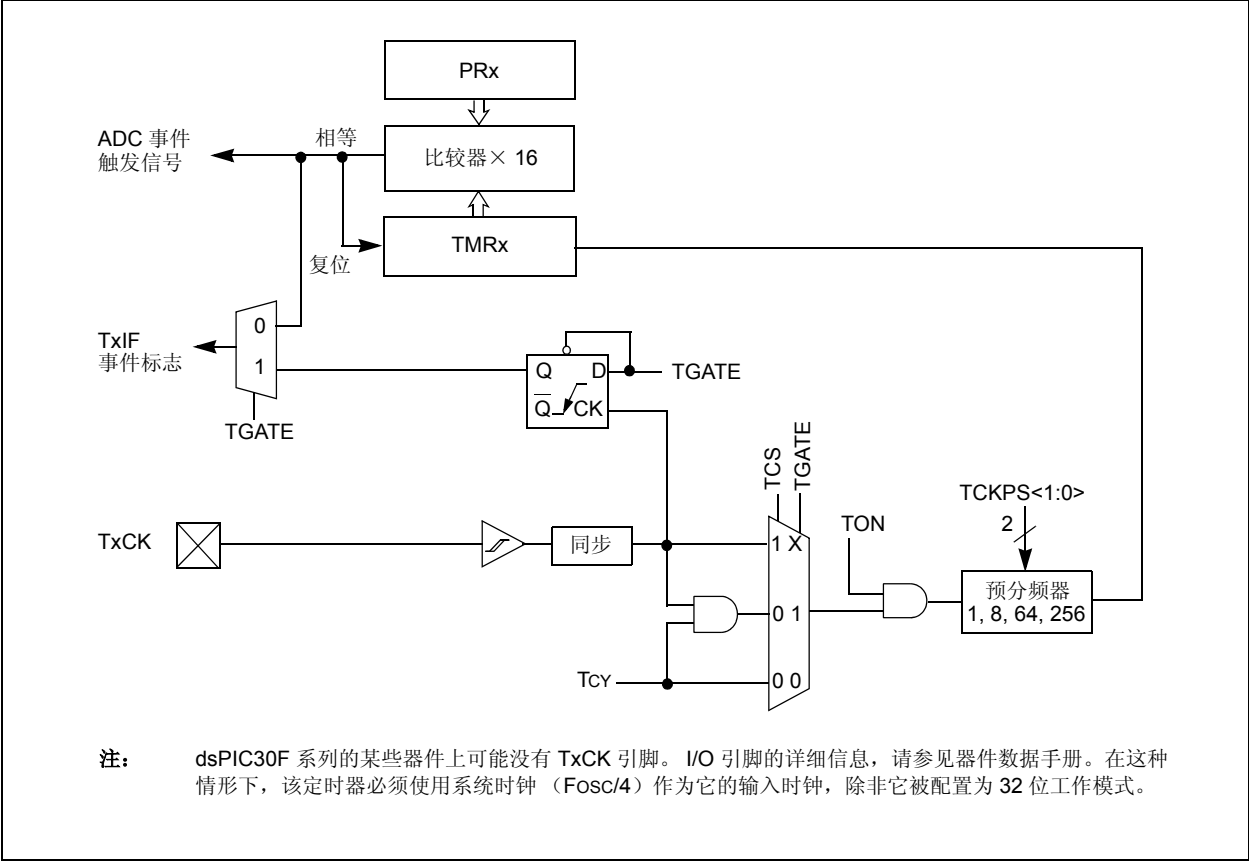
12.2.3 C 类型定时器

在大多数 dsPIC30F 器件上，Timer3 和 Timer5 是 C 类型定时器。与其他类型的定时器相比，C 类型定时器有下列独特的功能：

- C 类型定时器可以和 B 类型定时器相连形成 32 位定时器。
- 在某个给定的器件上，至少有一个 C 类型定时器能够触发 A/D 转换。

图 12-3 所示为 C 类型定时器的框图。

图 12-3: C 类型定时器的框图



12.3 控制寄存器

寄存器 12-1: TxCON: A 类型时基寄存器

高字节:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15				bit 8			

低字节:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		—	TSYNC	TCS	—
bit 7				bit 0			

- bit 15 **TON:** 定时器开控制位
1 = 启动定时器
0 = 停止定时器
- bit 14 **未用:** 读作 0
- bit 13 **TSIDL:** 空闲模式停止位
1 = 当器件进入空闲模式时, 定时器不继续工作
0 = 在空闲模式定时器继续工作
- bit 12-7 **未用:** 读作 0
- bit 6 **TGATE:** 定时器门控时间累加使能位
1 = 门控时间累加使能
0 = 门控时间累加禁止
(当 TGATE = 1 时, TCS 必须设置为 0。如果 TCS = 1, 该位读作 0)
- bit 5-4 **TCKPS<1:0>:** 定时器输入时钟预分频选择位
11 = 预分频比是 1:256
10 = 预分频比是 1:64
01 = 预分频比是 1:8
00 = 预分频比是 1:1
- bit 3 **未用:** 读作 0
- bit 2 **TSYNC:** 定时器外部时钟输入同步选择位
当 TCS = 1 时:
1 = 同步外部时钟输入
0 = 不同步外部时钟输入
当 TCS = 0 时:
此位被忽略。读作 0。当 TCS = 0 时, Timer1 使用内部时钟。
- bit 1 **TCS:** 定时器时钟源选择位
1 = 来自 TxCK 引脚的外部时钟
0 = 内部时钟 (Fosc/4)
- bit 0 **未用:** 读作 0

图注:

R = 可读位	W = 可写位	U = 未用位, 读作 0
-n = POR 时的值	1 = 置位	0 = 清零 x = 未知

寄存器 12-2: TxCON: B 类型时基寄存器

高字节:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							
bit 8							

低字节:							
U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		T32	—	TCS	—
bit 7				bit 0			

- bit 15
- TON: 定时器开控制位

当 T32 = 1 (处于 32 位定时器模式) 时:

1 = 启动 32 位成对 TMRx:TMRy 定时器

0 = 停止 32 位成对 TMRx:TMRy 定时器

当 T32 = 0 (处于 16 位定时器模式) 时:

1 = 启动 16 位定时器

0 = 停止 16 位定时器
- bit 14
- 未用: 读作 0
- bit 13
- TSIDL: 空闲模式停止位

1 = 当器件进入空闲模式时, 定时器停止工作

0 = 定时器在空闲模式继续工作
- bit 12-7
- 未用: 读作 0
- bit 6
- TGATE: 定时器门控时间累加使能位

1 = 定时器门控时间累加使能

0 = 定时器门控时间累加禁止

(当 TGATE = 1 时, TCS 必须设置为逻辑 0)
- bit 5-4
- TCKPS<1:0>: 定时器输入时钟预分频选择位

11 = 预分频比是 1:256

10 = 预分频比是 1:64

01 = 预分频比是 1:8

00 = 预分频比是 1:1
- bit 3
- T32: 32 位定时器模式选择位

1 = TMRx 和 TMRy 形成 32 位定时器

0 = TMRx 和 TMRy 为独立的 16 位定时器
- bit 2
- 未用: 读作 0
- bit 1
- TCS: 定时器时钟源选择位

1 = 来自 TxCK 引脚的外部时钟

0 = 内部时钟 (Fosc/4)
- bit 0
- 未用: 读作 0

图注:

R = 可读位	W = 可写位	U = 未用位, 读作 0
-n = POR 时的值	1 = 置位	0 = 清零 x = 未知

寄存器 12-3: TxCON: C 类型时基寄存器

高字节:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

低字节:							
U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	U-0
—	TGATE	TCKPS<1:0>		—	—	TCS	—
bit 7							bit 0

- bit 15 **TON:** 定时器开控制位
1 = 启动 16 位 TMRx
0 = 停止 16 位 TMRx
- bit 14 **未用:** 读作 0
- bit 13 **TSIDL:** 空闲模式停止位
1 = 当器件进入空闲模式时, 模块停止工作
0 = 模块在空闲模式继续工作
- bit 12-7 **未用:** 读作 0
- bit 6 **TGATE:** 定时器门控时间累加使能位
1 = 定时器门控时间累加使能
0 = 定时器门控时间累加禁止 (如果 TCS = 1, 读作 0)
(当 TGATE = 1 时, TCS 必须设置为逻辑 0)
- bit 5-4 **TCKPS<1:0>:** 定时器输入时钟预分频选择位
11 = 预分频比是 1:256
10 = 预分频比是 1:64
01 = 预分频比是 1:8
00 = 预分频比是 1:1
- bit 3-2 **未用:** 读作 0
- bit 1 **TCS:** 定时器时钟源选择位
1 = 来自 TxCK 引脚的外部时钟
0 = 内部时钟 (Fosc/4)
- bit 0 **未用:** 读作 0

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

12.4 工作模式

每个定时器模块可以工作在以下几种模式之一：

- 作为同步定时器
- 作为同步计数器
- 作为门控定时器
- 作为异步计数器（仅 A 类型时基）

定时器模式由下列位决定：

- TCS (TxCON<1>)：定时器时钟源控制位
- TSYNC (T1CON<2>)：定时器同步控制位（仅限 A 类型时基）
- TGATE (TxCON<6>)：定时器门控控制位

使用 TON 控制位 (TxCON<15>) 使能或禁止每个定时器模块。

注： 只有 A 类型时基支持外部异步时钟模式。

12.4.1 定时器模式

所有类型的定时器都可以在定时器模式下工作。在定时器模式下，定时器的输入时钟由内部系统时钟 (Fosc/4) 提供。当使能为该模式时，对于 1:1 的预分频器设置，定时器的计数值在每个指令周期都会加 1。通过清零 TCS 控制位 (TxCON<1>) 选择定时器模式。同步模式控制位 TSYNC (T1CON<2>) 在该模式下不起作用，因为使用了系统时钟源产生定时器时钟。

例 12-1: 使用系统时钟的 16 位定时器的初始化代码

```
; The following code example will enable Timer1 interrupts,
; load the Timer1 Period register and start Timer1.

; When a Timer1 period match interrupt occurs, the interrupt
; service routine must clear the Timer1 interrupt status flag
; in software.

CLR    T1CON                ; Stops the Timer1 and reset control reg.
CLR    TMR1                 ; Clear contents of the timer register
MOV    #0xFFFF, w0         ; Load the Period register
MOV    w0, PR1              ; with the value 0xFFFF

BSET   IPC0, #T1IP0         ; Setup Timer1 interrupt for
BCLR   IPC0, #T1IP1         ; desired priority level
BCLR   IPC0, #T1IP2         ; (this example assigns level 1 priority)
BCLR   IFS0, #T1IF          ; Clear the Timer1 interrupt status flag
BSET   IEC0, #T1IE          ; Enable Timer1 interrupts
BSET   T1CON, #TON           ; Start Timer1 with prescaler settings
                                ; at 1:1 and clock source set to
                                ; the internal instruction cycle

; Example code for Timer1 ISR

__T1Interrupt:
    BCLR   IFS0, #T1IF      ; Reset Timer1 interrupt flag
                                ; User code goes here.
    RETFIE                  ; Return from ISR
```

12.4.2 使用外部时钟输入的同步计数器模式

当 TCS 控制位 (TxCON<1>) 置 1 时, 定时器的时钟源由外部提供, 所选的定时器在 TxCK 引脚上的输入时钟的每个上升沿进行加 1 计数。

必须为 A 类型时基使能外部时钟同步。这通过将 TSYNC 控制位 (TxCON<2>) 置 1 实现。对于 B 类型和 C 类型时基, 外部时钟输入总是与系统指令周期时钟 Tcy 同步。

当定时器在同步计数器模式下工作时, 对外部时钟高电平和低电平有最短时间的要求。通过在一个指令周期内的两个不同时间对外部时钟信号进行采样, 可以实现外部时钟源与器件指令时钟的同步。

使用同步的外部时钟源工作的定时器在休眠模式下不工作, 因为同步电路在休眠模式下是关闭的。

注: 当在同步计数器模式下使用 Timerx 时, 外部输入时钟必须满足一定的高电平和低电平的最短时间要求。更多详细信息, 请参见器件数据手册的“电气规范”部分。

例 12-2: 使用外部时钟输入的 16 位同步计数器模式的初始化代码

```
; The following code example will enable Timer1 interrupts, load the
; Timer1 Period register and start Timer1 using an external clock
; and a 1:8 prescaler setting.

; When a Timer1 period match interrupt occurs, the interrupt service
; routine must clear the Timer1 interrupt status flag in software.

CLR    T1CON                ; Stops the Timer1 and reset control reg.
CLR    TMR1                 ; Clear contents of the timer register
MOV     #0x8CFF, w0          ; Load the Period register
MOV     w0, PR1              ; with the value 0x8CFF

BSET    IPC0, #T1IP0         ; Setup Timer1 interrupt for
BCLR    IPC0, #T1IP1         ; desired priority level
BCLR    IPC0, #T1IP2         ; (this example assigns level 1 priority)
BCLR    IFS0, #T1IF          ; Clear the Timer1 interrupt status flag
BSET    IEC0, #T1IE          ; Enable Timer1 interrupts
MOV     #0x8016, W0           ; Start Timer1 with prescaler settings at
                               ; 1:8 and clock source set to the external
MOV     w0, T1CON             ; clock in the synchronous mode

; Example code for Timer1 ISR

__T1Interrupt:
    BCLR    IFS0, #T1IF      ; Reset Timer1 interrupt flag
                               ; User code goes here.
    RETFIE                    ; Return from ISR
```

12.4.3 使用外部时钟输入的 A 类型定时器异步计数器模式

通过使用连接到 TxCK 引脚的外部时钟源，A 类型时基能够在异步计数模式下工作。当 TSYNC 控制位 (TxCON<2>) 清零时，外部时钟输入不与器件系统时钟源同步。该时基继续进行与内部器件时钟异步的递增计数。

异步工作的时基对于以下应用是有益的：

- 时基可以在休眠模式下工作，并能够在发生周期寄存器匹配时产生中断，将唤醒处理器。
- 在实时时钟应用中，可以使用低功耗 32 kHz 振荡器作为时基的时钟源。

详情请参见第 12.12.1 节“休眠模式下的定时器工作”。

注

1: 只有 A 类型时基支持异步计数器模式。

2: 当在异步计数器模式下使用 Timerx 时，外部输入时钟必须满足一定的高电平和低电平最短时间要求。更多详细信息，请参见器件数据手册的“电气规范”部分。

3: 在异步模式下读 Timer1，可能会产生意外的结果。

例 12-3: 使用外部时钟输入的 16 位异步计数器模式的初始化代码

```
; The following code example will enable Timer1 interrupts, load the
; Timer1 Period register and start Timer1 using an asynchronous
; external clock and a 1:8 prescaler setting.

; When a Timer1 period match interrupt occurs, the interrupt service
; routine must clear the Timer1 interrupt status flag in software.

CLR    T1CON                ; Stops the Timer1 and reset control reg.
CLR    TMR1                 ; Clear contents of the timer register
MOV    #0x7FFF, w0          ; Load the Period register
MOV    w0, PR1              ; with the value 0x7FFF

BSET   IPC0, #T1IP0         ; Setup Timer1 interrupt for
BCLR   IPC0, #T1IP1         ; desired priority level
BCLR   IPC0, #T1IP2         ; (this example assigns level 1 priority)
BCLR   IFS0, #T1IF          ; Clear the Timer1 interrupt status flag
BSET   IEC0, #T1IE          ; Enable Timer1 interrupts
MOV    #0x8012, w0          ; Start Timer1 with prescaler settings at
                             ; 1:8 and clock source set to the external
MOV    w0, T1CON            ; clock in the asynchronous mode

; Example code for Timer1 ISR

__T1Interrupt:
    BCLR   IFS0, #T1IF      ; Reset Timer1 interrupt flag
                             ; User code goes here.
    RETFIE                  ; Return from ISR
```

12.4.4 使用快速外部时钟源的定时器工作原理

在一些应用中，可能希望使用其中一个定时器为来自频率相对较高的外部时钟源的时钟沿计数。在这些情形中，A 类型和 B 类型时基是对外部时钟源计数最合适的选择，因为这些定时器的时钟同步逻辑位于定时器预分频器之后（参见图 12-1 和图 12-2）。这样就可能使用更高的外部时钟频率而不违反预分频器所要求的高电平和低电平最短时间。当为 A 类型或 B 类型时基选择的定时器预分频器分频比不是 1:1 时，外部时钟输入的高电平和低电平最短时间会被选择的预分频比减小。

A 类型时基是独特的，因为它可以在异步时钟模式下工作，从而消除了任何预分频器的时序要求。

注意在所有情形中，都不能超出外部时钟信号的高电平和低电平最短时间要求。这些最短时间是满足 I/O 引脚的时序要求所必需的。

想要了解与时基相关的外部时钟时序规范，请参见器件数据手册。

12.4.5 门控时间累加模式

门控时间累加模式允许内部定时器寄存器根据加在 **TxCCK** 引脚上的高电平时间进行递增计数。在门控时间累加模式下，定时器时钟源来自于内部系统时钟。当 **TxCCK** 引脚为高电平状态时，定时器寄存器将递增计数，直到发生周期匹配或 **TxCCK** 引脚变为低电平状态。引脚状态从高电平到低电平的转变会将 **TxIF** 中断标志位置 1。根据边沿发生的时间，中断标志位在 **TxCCK** 引脚上信号下降沿产生后的一或两个指令周期置 1。

必须将 **TGATE** 控制位（**TxCON<6>**）置位以使能门控时间累加模式。必须使能定时器（**TON**（**TxCON<15>**）= 1），并且把内部时钟设置为定时器时钟源（**TCS**（**TxCON<1>**）= 0）。

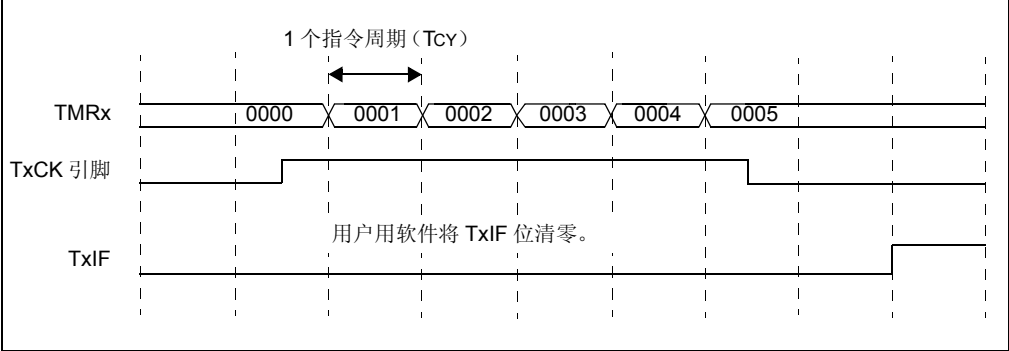
当加在 **TxCCK** 引脚上的信号出现上升沿时，门控控制电路开始工作；当加在 **TxCCK** 引脚上的信号出现下降沿时，门控控制电路终止工作。当外部门控信号为高电平时，对应的定时器将进行递增计数。

门控信号的下降沿会终止计数工作，但是不会复位定时器。如果想让定时器在门控输入信号的下一个上升沿出现时从零开始计数，用户必须复位该定时器。门控信号的下降沿产生中断。

注： 在门控时间累加模式下，当发生定时器周期匹配时，定时器不会中断 CPU。

定时器计数的精度与定时器时钟周期直接相关。对于预分频比为 1:1 的定时器预分频器，定时器时钟周期为一个指令周期。对于预分频比为 1:256 的定时器预分频器，定时器时钟周期为 256 个指令周期。可将定时器时钟精度与门控信号的脉冲宽度相关联。更多关于门控脉宽要求的详细信息，请参见器件数据手册中的“电气规范”部分。

图 12-4: 门控定时器模式工作



例 12-4: 16 位门控时间累加模式的初始化代码

```
; The following code example will enable Timer2 interrupts, load the
; Timer2 Period register and start Timer2 using an internal clock
; and an external gate signal. On the falling edge of the gate
; signal a Timer2 interrupt occurs. The interrupt service
; routine must clear the Timer2 interrupt status flag in software .

CLR      T2CON                ; Stops the Timer2 and reset control reg.
CLR      TMR2                ; Clear contents of the timer register
MOV      #0xFFFF, w0        ; Load the Period register with
MOV      w0, PR2             ; the value 0xFFFF

BSET     IPC1, #T2IP0        ; Setup Timer2 interrupt for
BCLR     IPC1, #T2IP1        ; desired priority level
BCLR     IPC1, #T2IP2        ; (this example assigns level 1 priority)
BCLR     IFS0, #T2IF         ; Clear the Timer2 interrupt status flag
BSET     IEC0, #T2IE         ; Enable Timer2 interrupts
BSET     T2CON, #TGATE        ; Set up Timer2 for operation in Gated
                                ; Time Accumulation mode
BSET     T2CON, #TON          ; Start Timer2

; Example code for Timer2 ISR
__T2Interrupt:
    BCLR     IFS0, #T2IF      ; Reset Timer2 interrupt flag
                                ; User code goes here.
    RETFIE                    ; Return from ISR
```

12.5 定时器预分频器

所有 16 位定时器的输入时钟 (Fosc/4 或外部时钟) 的预分频比选项为 1:1、1:8、1:64 和 1:256。使用 TCKPS<1:0> 控制位 (TxCON<5:4>) 选择时钟的预分频比。当发生以下情况中的任何一种时, 预分频器计数器清零:

- 写 TMRx 寄存器
- TON (TxCON<15>) 清零
- 任何器件复位

注: 写 TxCON 时, TMRx 寄存器不会清零。

12.6 定时器中断

根据工作模式的不同, 16 位定时器可以在发生周期匹配或外部门控信号的下降沿出现时产生中断。

当下列条件之一为真时, TxIF 位置 1:

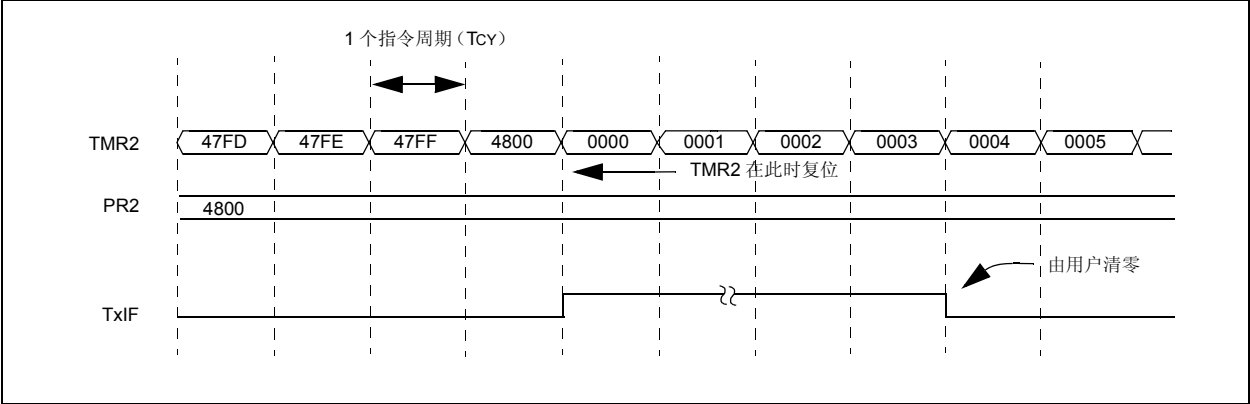
- 定时器的计数值与对应的周期寄存器匹配, 而且该定时器模块不工作在门控时间累加模式。
- 当定时器工作在门控时间累加模式下时, 检测到 “门控” 信号的下降沿。

TxIF 位必须用软件清零。

通过对应的定时器中断使能位 TxIE, 可以将定时器使能作为中断源。此外, 为了使该定时器成为中断源, 必须对中断优先级位 (TxIP<2:0>) 写入非零值。更多详细信息, 请参见第 6 章 “复位中断”。

注: 当周期寄存器装载了 0x0000 且定时器被使能时, 会发生特殊情形。在这种配置下, 将不会产生定时器中断。

图 12-5: 定时器周期匹配时的中断时序



12.7 读写 16 位定时器模块寄存器

- 可以字节（8 位）或字（16 位）为单位写定时器模块的所有 SFR。
- 定时器模块的所有 SFR 只能作为一个字（16 位）读取。

12.7.1 写 16 位定时器

当定时器模块工作时，可以对该定时器及其对应的周期寄存器进行写操作。在执行字节写操作时用户应该对以下情况有所认识：

- 如果定时器正在增计数，此时对定时器的低字节进行写操作，那么该定时器的高字节不受影响。如果将 0xFF 写入该定时器的低字节，在此次写操作之后的下一个定时器计数时钟将使低字节计满返回到 0x00，并对定时器的高字节产生一个进位。
- 如果定时器正在递增计数，此时对定时器的高字节进行写操作，那么该定时器的低字节不受影响。如果当此次写操作发生时定时器的低字节为 0xFF，那么在下一个定时器计数时钟将从定时器低字节产生进位，并且该进位会使定时器的高字节加 1。

当通过一条指令将字或字节写入 TMRx 寄存器时，TMRx 寄存器的递增计数被屏蔽且在该指令周期内都不会发生递增计数。

在实时计时应用中，应该避免对使用异步时钟源的定时器进行写操作。更多的详细信息请参见第 12.4.1 节“定时器模式”。

12.7.2 读 16 位定时器

对定时器及其相关 SFR 的所有读操作必须以字为单位读取（16 位）。字节读取不起作用（将返回“0”）。

当该模块工作时，可以对定时器及其对应的周期寄存器进行读操作。读 TMRx 寄存器不会阻止定时器在同一个指令周期进行递增计数。

12.8 低功耗 32 kHz 晶振输入

在各种不同的器件中，A 类型定时器模块均可使用低功耗 32 kHz 晶振用于实时时钟（RTC）应用。

- 当 LP 振荡器使能且该定时器被配置为使用外部时钟源时，LP 振荡器成为该定时器的时钟源。
- 通过将 OSCCON 寄存器中的 LPOSCEN 控制位置 1，可以使能 LP 振荡器。
- 32 kHz 晶振连接到 SOSCO/SOSCI 器件引脚。

更多详细信息，请参见第 7 章“振荡器”。

12.9 32 位定时器配置

B 类型和 C 类型 16 位定时器模块可以组合形成 32 位定时器模块。C 类型时基成为组成的定时器的 MSWord，而 B 类型时基是 LSWord。

当配置为 32 位工作时，B 类型时基的控制位控制 32 位定时器的工作。C 类型时基的 TxCON 寄存器中的控制位不起作用。

组合的 32 位定时器使用 C 类型时基的中断使能、中断标志和中断优先级控制位进行中断控制。在 32 位定时器工作中，不使用 B 类型时基的中断控制和状态位。

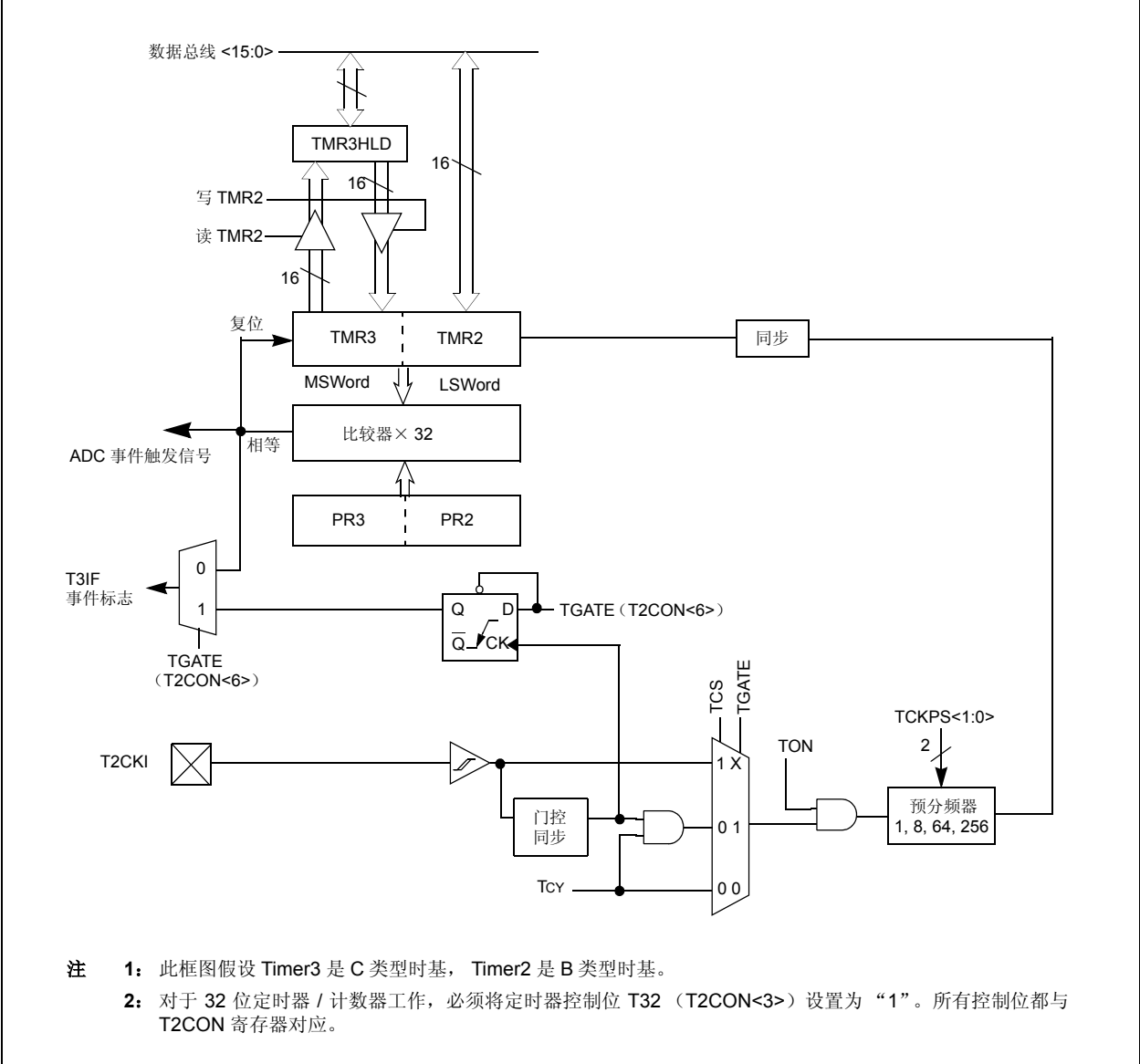
注： 关于可以组合的特定的 B 类型和 C 类型时基的信息，请参见器件数据手册。

以下的配置设置假设 Timer3 是 C 类型时基，而 Timer2 是 B 类型时基：

- TON (T2CON<15>) = 1。
- T32 (T2CON<3>) = 1。
- TCKPS<1:0> (T2CON<5:4>) 用于为 Timer2 设置预分频器模式 (B 类型时基)。
- TMR3:TMR2 寄存器对包含定时器模块的 32 位值；TMR3 (C 类型时基) 寄存器是该 32 位定时器值的最高有效字，而 TMR2 (B 类型时基) 寄存器是该 32 位定时器值的最低有效字。
- PR3:PR2 寄存器对包含 32 位周期值，该值用于与 TMR3:TMR2 定时器值作比较。
- T3IE (IEC0<7>) 用于允许该配置的 32 位定时器中断。
- T3IF (IFS0<7>) 用作该定时器中断的状态标志。
- T3IP<2:0> (IPC1<14:12>) 为该 32 位定时器设置中断优先级。
- T3CON<15:0> 是无关位。

图 12-6 中所示为使用 Timer2 和 Timer3 的 32 位定时器模块示例的框图。

图 12-6: B 类型和 C 类型定时器对的框图（32 位定时器）



12.10 32 位定时器的的工作模式

12.10.1 定时器模式

例 12-5 展示了如何在定时器模式下配置 32 位定时器。此例假设 Timer2 是 B 类型时基，而 Timer3 是 C 类型时基。对于 32 位定时器工作模式，必须将 T2CON 寄存器（B 类型时基）中的 T32 控制位置 1。当 Timer2 和 Timer3 被配置为 32 位定时器时，T3CON 控制位被忽略。只需要 T2CON 控制位用于设置和控制。32 位定时器模块使用 Timer2 时钟和门控输入，但是产生中断会将 T3IF 标志位置 1。Timer2 和 Timer3 分别是 32 位定时器的 LSWord 和 MSWord。来自 TMR2 的溢出（进位）使 TMR3 进行递增计数。32 位定时器进行递增计数，直到与由 PR2 和 PR3 组合形成的 32 位周期寄存器中预先装入的值相匹配，然后计满回零并继续计数。要使 32 位定时器能计数到最大值，把值 0xFFFFFFFF 装入 PR3:PR2。若允许中断，中断将在周期匹配时产生。

例 12-5: 使用指令周期作为输入时钟的 32 位定时器的初始化代码

```
; The following code example will enable Timer3 interrupts, load the
; Timer3:Timer2 Period Register and start the 32-bit timer module
; consisting of Timer3 and Timer2.

; When a 32-bit period match interrupt occurs, the user must clear
; the Timer3 interrupt status flag in software.

CLR    T2CON                ; Stops any 16/32-bit Timer2 operation
CLR    T3CON                ; Stops any 16-bit Timer3 operation
CLR    TMR3                 ; Clear contents of the Timer3 timer register
CLR    TMR2                 ; Clear contents of the Timer2 timer register
MOV    #0xFFFF, w0         ; Load the Period Register 3
MOV    w0, PR3              ; with the value 0xFFFF
MOV    w0, PR2              ; Load the Period Register2 with value 0xFFFF

BSET   IPC1, #T3IP0         ; Setup Timer3 interrupt for
BCLR   IPC1, #T3IP1         ; desired priority level
BCLR   IPC1, #T3IP2         ; (this example assigns level 1 priority)
BCLR   IFS0, #T3IF          ; Clear the Timer3 interrupt status flag
BSET   IEC0, #T3IE          ; Enable Timer3 interrupts
BSET   T2CON, #T32          ; Enable 32-bit Timer operation
BSET   T2CON, #TON          ; Start 32-bit timer with prescaler
                        ; settings at 1:1 and clock source set to
                        ; the internal instruction cycle

; Example code for Timer3 ISR

__T3Interrupt:
BCLR   IFS0, #T3IF          ; Reset Timer3 interrupt flag
                        ; User code goes here.
RETFIE                          ; Return from ISR
```

12.10.2 同步计数器模式

在同步计数器模式下，32 位定时器与 16 位定时器的工作方式。例 12-6 展示了如何在同步计数器模式下配置 32 位定时器。此示例假设 Timer2 是 B 类型时基，而 Timer3 是 C 类型时基。

例 12-6: 使用外部时钟输入的 32 位同步计数器模式的初始化代码

```
; The following code example will enable Timer2 interrupts, load
; the Timer3:Timer2 Period register and start the 32-bit timer
; module consisting of Timer3 and Timer2.

; When a 32-bit period match interrupt occurs, the user must clear
; the Timer3 interrupt status flag in the software.

CLR      T2CON          ; Stops any 16/32-bit Timer2 operation
CLR      T3CON          ; Stops any 16-bit Timer3 operation
CLR      TMR3           ; Clear contents of the Timer3 timer register
CLR      TMR2           ; Clear contents of the Timer2 timer register
MOV      #0xFFFF, w0    ; Load the Period Register3
MOV      w0, PR3         ; with the value 0xFFFF
MOV      w0, PR2         ; Load the Period Register2 with value 0xFFFF

BSET     IPC1, #T3IP0    ; Setup Timer3 interrupt for
BCLR     IPC1, #T3IP1    ; desired priority level
BCLR     IPC1, #T3IP2    ; (this example assigns level 1 priority)
BCLR     IFS0, #T3IF     ; Clear the Timer3 interrupt status flag
BSET     IEC0, #T3IE     ; Enable Timer3 interrupts
MOV      #0x801A, w0     ; Enable 32-bit Timer operation and start
MOV      w0, T2CON       ; 32-bit timer with prescaler settings at
                        ; 1:8 and clock source set to external clock

; Example code for Timer3 ISR

__T3Interrupt:
BCLR     IFS0, #T3IF     ; Reset Timer3 interrupt flag
                        ; User code goes here.
RETFIE                        ; Return from ISR
```

12.10.3 异步计数器模式

B 类型和 C 类型时基不支持异步外部时钟模式。所以，不支持任何 32 位异步计数器模式。

12.10.4 门控时间累加模式

在门控时间累加模式下，32 位定时器与 16 位定时器工作方式类似。例 12-7 展示了如何在门控时间累加模式下配置 32 位定时器。此例假设 Timer2 是 B 类型时基，而 Timer3 是 C 类型时基。

例 12-7: 32 位门控时间累加模式的初始化代码

```
; The following code example will enable Timer2 interrupts, load the
; Timer3:Timer2 Period register and start the 32-bit timer module
; consisting of Timer3 and Timer2. When a 32-bit period match occurs
; the timer will simply roll over and continue counting.

; However, when at the falling edge of the Gate signal on T2CK
; an interrupt is generated, if enabled. The user must clear the
; Timer3 interrupt status flag in the software.

CLR    T2CON                ; Stops any 16/32-bit Timer2 operation
CLR    T3CON                ; Stops any 16-bit Timer3 operation
CLR    TMR3                 ; Clear contents of the Timer3 register
CLR    TMR2                 ; Clear contents of the Timer2 register
MOV    #0xFFFF, w0         ; Load the Period Register3
MOV    w0, PR3              ; with the value 0xFFFF
MOV    w0, PR2              ; Load the Period Register2 with value 0xFFFF

BSET   IPC1, #T3IP0         ; Setup Timer3 interrupt for
BCLR   IPC1, #T3IP1         ; desired priority level
BCLR   IPC1, #T3IP2         ; (this example assigns level 1 priority)
BCLR   IFS0, #T3IF          ; Clear the Timer3 interrupt status flag
BSET   IEC0, #T3IE          ; Enable Timer3 interrupts
MOV    #0x804C, w0          ; Enable 32-bit Timer operation and
MOV    w0, T2CON             ; Start 32-bit timer in gated time
                                ; accumulation mode.

; Example code for Timer3 ISR

__T3Interrupt:
BCLR   IFS0, #T3IF          ; Reset Timer3 interrupt flag
                                ; User code goes here.
RETfie                          ; Return from ISR
```


12.11 读写 32 位定时器

为了使 32 位读 / 写操作在 32 位定时器的 LSWord 和 MSWord 之间同步, 使用了额外的控制逻辑电路和保持寄存器 (参见图 12-6)。每个 C 类型时基都有一个称为 TMRxHLD 的寄存器, 当读或写该定时器的一对寄存器时使用它。只有其对应的定时器被配置为 32 位工作时才会使用 TMRxHLD 寄存器。

假设 TMR3:TMR2 形成一对 32 位定时器; 用户应该首先从 TMR2 寄存器读取定时器的 LSWord。读 LSWord 将会自动把 TMR3 的内容传送给 TMR3HLD 寄存器。然后用户可以读 TMR3HLD, 以得到定时器值的 MSWord。下面的示例展示了此过程:

例 12-8: 读 32 位定时器

```
; The following code segment reads the 32-bit timer formed by the
; Timer3-Timer2 pair into the registers W1(MS Word) and W0(LS Word).

MOV TMR2, W0      ;Transfer the LSW into W1
MOV TMR3HLD, W1   ;Transfer the MSW from the holding register to W0
```

要将值写入 TMR3:TMR2 这对寄存器, 用户应该首先将 MSWord 写入 TMR3HLD 寄存器。当定时器值的 LSWord 被写入 TMR2 时, TMR3HLD 的内容将会自动传送到 TMR3 寄存器。

12.12 低功耗状态下的定时器工作

12.12.1 休眠模式下的定时器工作

当器件进入休眠模式后, 会禁止系统时钟。如果定时器模块使用内部时钟源 (Fosc/4) 运行, 该定时器也会被禁止。

A 类型定时器与其他定时器模块不同, 因为它能使用外部时钟源异步工作。由于这个差别, A 类型时基模块可以在休眠模式下继续工作。要在休眠模式下工作, A 类型时基必须作如下配置:

- Timer1 模块使能, TON = 1 (T1CON<15>) 且
- 选择外部时钟源作为 Timer1 时钟源, TCS = 1 (T1CON<1> = 1) 以及
- TSYNC 位 (T1CON<2>) 设置为逻辑 0 (异步计数器模式使能)。

注: 只有 Timer1 模块才支持异步计数器工作模式。

满足了上述所有条件后, 当器件处于休眠模式时, Timer1 将继续计数并检测周期匹配。当定时器和周期寄存器发生匹配时, TxIF 位将被置 1 并可以产生中断, 从而选择将器件从休眠模式唤醒。更多详细信息, 请参见第 10 章 “看门狗定时器和低功耗模式”。

当在异步模式下使用 32.768 kHz 的实时振荡器执行 SLEEP 指令以保持计时单位为秒的实时时, 应确保连接到输入端的晶体处于正常工作, 这是非常重要的。这可通过检查 TMR1 中是否有一个非零值并执行 SLEEP 指令做到。如果未做到这一点, 将导致执行正常休眠操作后, 由于定时器超时而无法从休眠中唤醒。

代码示例:

```
mov.b #0x46,w1      ; follow write sequence ...
mov.b #0x57,w2      ; for OSCCONL writes.
mov #OSCCONL,w3
mov.b w1,[w3]
mov.b w2,[w3]
bset OSCCONL,#LPOSCEN ;enable 32Khz external xtal
clr TMR1            ; set up TMR1 for ...
mov #0x7FFF,W0      ;interrupts every 1.0 Sec
mov W0,PR1
```

```
bclr    IFS0, #T1IF      ; clr interrupt flag
mov     #0x8002, W0      ; External clock, 1:1; start TMR1
mov     W0, T1CON
bset    IEC0, #T1IE      ; enable interrupt
TestLPOperation:
mov     TMR1, W0          ; check for non-zero value in TMR1
cp      W0, #0
bra     Z, TestLPOperation ; if zero then recheck
pwrsav  #0                ; non-zero then execute sleep
```

12.12.2 空闲模式下的定时器工作

当器件进入空闲模式时，系统时钟源保持工作，但 CPU 停止执行代码。定时器模块可以选择在空闲模式下继续工作。

TSIDL 位 (TxCON<13>) 选择在空闲模式下定时器模块是停止还是继续正常工作。如果 TSIDL = 0，在空闲模式下该模块将继续工作。如果 TSIDL = 1，在空闲模式下该模块将停止工作。

12.13 使用定时器模块的外设

12.13.1 输入捕捉 / 输出比较的时基

输入捕捉和输出比较外设可以选择两个定时器模块中的一个作为它们的时基。更多详细信息请参见第 13 章“输入捕捉”、第 14 章“输出比较”和器件数据手册。

12.13.2 A/D 特殊事件触发信号

在各个不同的器件上，在 16 位和 32 位模式下，当发生周期匹配时，C 类型时基都能够产生特殊 A/D 转换触发信号。该定时器模块为 A/D 采样逻辑电路提供了转换启动信号。

- 如果 T32 = 0，当 16 位定时器寄存器 (TMRx) 与各自相应的 16 位周期寄存器 (PRx) 之间发生匹配时，会产生 A/D 特殊事件触发信号。
- 如果 T32 = 1，当 32 位定时器寄存器 (TMRx:TMRy) 与对应的 32 位组合的周期寄存器 (PRx:PRy) 之间发生匹配时，会产生 A/D 特殊事件触发信号。

特殊事件触发信号总是由定时器产生。必须在 A/D 转换器控制寄存器中选择触发源。更多信息，请参见第 17 章“10 位 A/D 转换器”、第 18 章“12 位 A/D 转换器”和器件数据手册。

12.13.3 定时器作为外部中断引脚

每个定时器的外部时钟输入引脚都可以用作额外的中断引脚。为了提供中断，对定时器周期寄存器 PRx 写入非零值，而将 TMRx 寄存器初始化为一个比写入周期寄存器的值小 1 的值。定时器必须配置一个 1:1 的时钟预分频器。当检测到外部时钟信号的下一个上升沿时，将产生中断。

12.13.4 I/O 引脚控制

当定时器模块使能，并配置为外部时钟或门控工作时，用户必须确保 I/O 引脚方向被配置为输入。使能该定时器模块不会配置引脚方向。

表 12-1: 与定时器模块相关的特殊功能寄存器

SFR 名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的值
TMR1	0100	Timer1 寄存器																0000 0000 0000 0000
PR1	0102	Timer1 周期寄存器																1111 1111 1111 1111
T1CON	0104	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	TSYNC	TCS	—	0000 0000 0000 0000
TMR2	0106	Timer2 寄存器																0000 0000 0000 0000
TMR3HLD	0108	Timer3 保持寄存器 (仅在 32 位模式下使用)																0000 0000 0000 0000
TMR3	010A	Timer3 寄存器																0000 0000 0000 0000
PR2	010C	Timer2 周期寄存器																1111 1111 1111 1111
PR3	010E	Timer3 周期寄存器																1111 1111 1111 1111
T2CON	0110	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000 0000 0000 0000
T3CON	0112	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000 0000 0000 0000
TMR4	0114	Timer4 寄存器																0000 0000 0000 0000
TMR5HLD	0116	Timer5 保持寄存器 (仅在 32 位模式下使用)																0000 0000 0000 0000
TMR5	0118	Timer5 寄存器																0000 0000 0000 0000
PR4	011A	Timer4 周期寄存器																1111 1111 1111 1111
PR5	011C	Timer5 周期寄存器																1111 1111 1111 1111
T4CON	011E	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000 0000 0000 0000
T5CON	0120	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000 0000 0000 0000
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0IF	0000 0000 0000 0000
IFS1	0086	IC61F	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IEC0	008C	CNIE	MI2CIE	IC2IE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			INT0IP<2:0>				0100 0100 0100 0100
IPC1	0096	—	T3IP<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			IC2IP<2:0>				0100 0100 0100 0100
IPC5	009E	—	INT2IP<2:0>			—	T5IP<2:0>			—	T4IP<2:0>			OC4IP<2:0>				0100 0100 0100 0100

注: 要获取特定存储器映射的详细信息, 请参见器件数据手册。

12.14 设计技巧

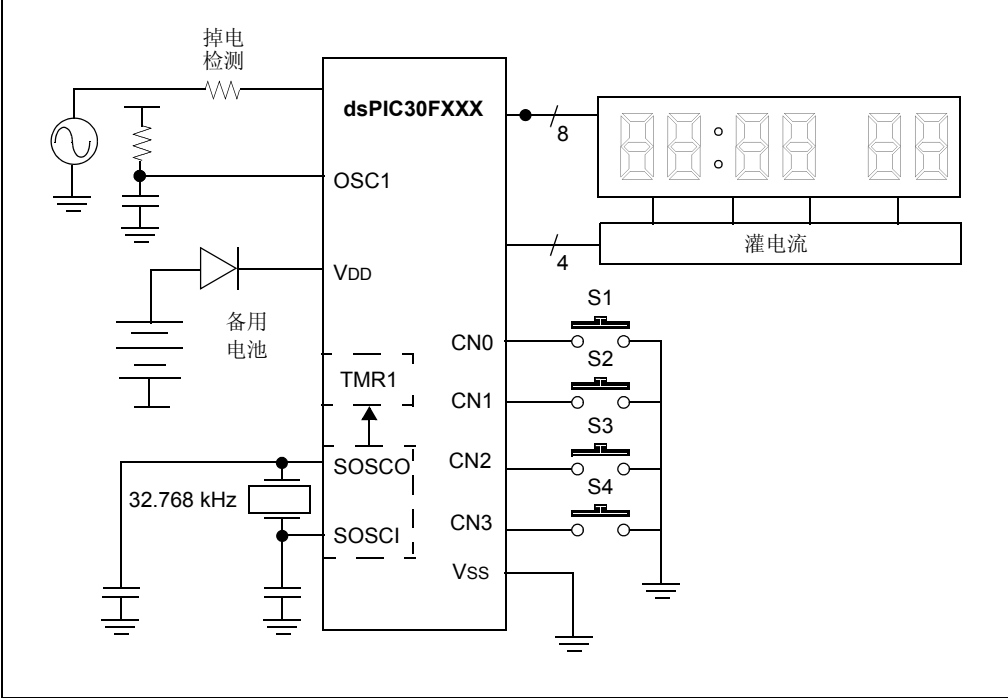
问 1: 可以使用定时器模块将器件从休眠模式唤醒吗?

答: 可以, 但是只有 Timer1 可以把器件从休眠模式唤醒。这是因为 Timer1 允许 TMR1 寄存器对外部的异步的时钟源进行递增计数。当 TMR1 寄存器与 PR1 寄存器的值相等时, 如果已经使用 T1IE 控制位允许了 Timer1 中断, 器件将从休眠模式唤醒。更多详细信息请参见第 12.12.1 节 “休眠模式下的定时器工作”。

12.14.1 应用示例

应用示例如图 12-7 所示, 其中 Timer1 (A 类型时基) 由 32.768 kHz 的外部振荡器驱动。32.768 kHz 外部振荡器通常用于需要实时时钟的应用场合, 但是在需要低功耗的应用场合也可以使用。Timer1 振荡器可将器件置于休眠状态而定时器将继续递增计数。当 Timer1 溢出时, 中断会唤醒器件, 从而更新相应的寄存器。

图 12-7: Timer1 应用



在此例中, 32.768 kHz 的晶振被用作实时时钟的时基。如果时钟需要每隔 1 秒被更新一次, 那么就必须将一个值装入周期寄存器 PR1, 使它与 Timer1 以需要的速率匹配。每隔 1 秒就发生 Timer1 匹配事件的情形中, 应该将值 0x8000 装入 PR1 寄存器。

注: 为了使实时时钟准确工作, 永远不要对 TMR1 寄存器进行写操作, 因为 Timer1 时钟源与系统时钟异步。对 TMR1 寄存器的写操作会破坏实时计数器的值, 从而导致计时不准。

12.15 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与定时器模块相关的应用笔记有：

标题	应用笔记编号
Using Timer1 in Asynchronous Clock Mode	AN580
Yet Another Clock Featuring the PIC16C924	AN649

注：如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

12.16 版本历史

版本 A

这是本文档的初始发行版。

版本 B

此版本添加了 dsPIC30F 定时器模块的更多技术内容。

版本 C

本章未包含任何技术内容上的修订，然而我们谨更新此节以反映本手册已更新到版本 C。

版本 D

在第 12.12.1 节“休眠模式下的定时器工作”中增加了代码示例。

第 13 章 输入捕捉

目录

本章包括下列主题：

13.1 简介	13-2
13.2 输入捕捉寄存器	13-3
13.3 定时器选择	13-4
13.4 输入捕捉事件模式	13-4
13.5 捕获缓冲器的操作	13-8
13.6 输入捕捉中断	13-9
13.7 UART 自动波特率支持	13-9
13.8 低功耗状态下的输入捕捉工作	13-10
13.9 I/O 引脚控制	13-10
13.10 与输入捕捉模块相关的特殊功能寄存器	13-11
13.11 设计技巧	13-12
13.12 相关应用笔记	13-13
13.13 版本历史	13-14

13.1 简介

本章描述了输入捕捉模块及其相关的工作模式。输入捕捉模块用于在输入引脚上有事件发生时，捕捉来自两个可选时基之一的定时器值。输入捕捉功能在需要进行频率（时间周期）和脉冲测量的应用中是相当有用的。图 13-1 展示了输入捕捉模块的简化框图。

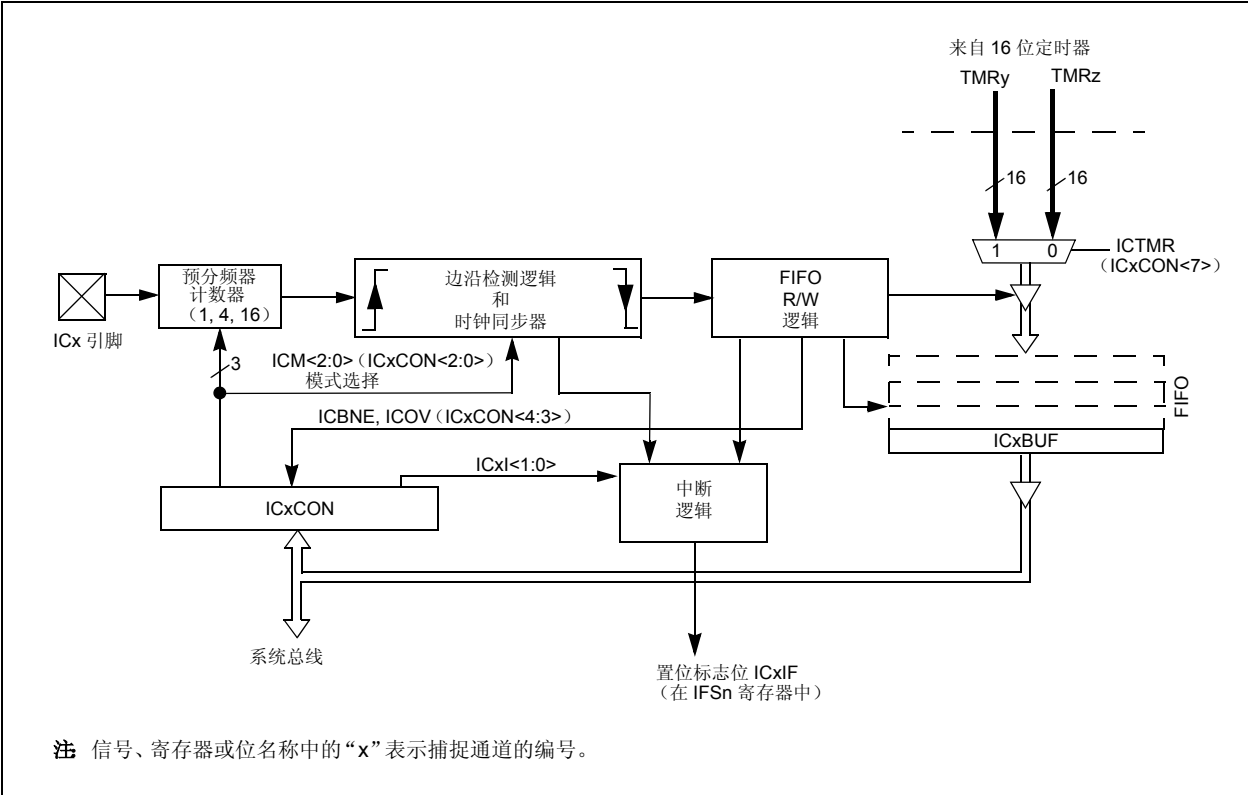
关于某特定器件可用通道数量的更多信息请参见特定器件的数据手册。所有输入捕捉通道在功能上都是相同的。在本章中，引脚名称或寄存器名称中的“x”表示特定的输入捕捉通道。

输入捕捉模块有多种工作模式，可通过 ICxCON 寄存器选择。这些工作模式包括：

- 在 ICx 引脚输入电平的下降沿捕捉定时器值
- 在 ICx 引脚输入电平的上升沿捕捉定时器值
- 在 ICx 引脚输入电平的第 4 个上升沿捕捉定时器值
- 在 ICx 引脚输入电平的第 16 个上升沿捕捉定时器值
- 在 ICx 引脚输入电平的每个上升沿和下降沿捕捉定时器值

输入捕捉模块有 4 级 FIFO 缓冲器。用户可以选择产生 CPU 中断所需要的捕捉事件数量。

图 13-1: 输入捕捉框图



注 信号、寄存器或位名称中的“x”表示捕捉通道的编号。

13.2 输入捕捉寄存器

dsPIC30F 器件中的每个捕捉通道都有以下寄存器，寄存器名称中的“x”代表捕捉通道的编号：

- ICxCON：输入捕捉控制寄存器
- ICxBUF：输入捕捉缓冲寄存器

寄存器 13-1: ICxCON：输入捕捉 x 控制寄存器

高字节:							
U-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
—	—	ICSIDL	—	—	—	—	—
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R-0, HC	R-0, HC	R/W-0	R/W-0	R/W-0
ICTMR	ICI<1:0>	ICOV	ICBNE	ICM<2:0>			
bit 7				bit 0			

bit 15-14 未用：读作 0

bit 13 **ICSIDL**：输入捕捉模块在空闲时停止控制位
 1 = 输入捕捉模块在 CPU 空闲模式将停止
 0 = 输入捕捉模块在 CPU 空闲模式将继续工作

bit 12-8 未用：读作 0

bit 7 **ICTMR**：输入捕捉定时器选择位
 1 = 捕捉事件时捕捉 TMR2 的内容
 0 = 捕捉事件时捕捉 TMR3 的内容

注： 可供选择的定时器可能会和上述不同。更多详细信息请参见器件数据手册。

bit 6-5 **ICI<1:0>**：每次中断的捕捉次数选择位
 11 = 每 4 次捕捉事件中中断一次
 10 = 每 3 次捕捉事件中中断一次
 11 = 每 2 次捕捉事件中中断一次
 10 = 每 1 次捕捉事件中中断一次

bit 4 **ICOV**：输入捕捉溢出状态标志（只读）位
 1 = 发生了输入捕捉溢出
 0 = 未发生输入捕捉溢出

bit 3 **ICBNE**：输入捕捉缓冲器空状态（只读）位
 1 = 输入捕捉缓冲器非空，至少可以再读一次捕捉值
 0 = 输入捕捉缓冲器为空

bit 2-0 **ICM<2:0>**：输入捕捉模式选择位
 111 = 当器件处于休眠或空闲模式时，输入捕捉仅用做中断引脚功能
 （只检测上升沿，所有其他控制位都不适用。）
 110 = 未使用（模块禁止）
 101 = 捕捉模式，每 16 个上升沿捕捉一次
 100 = 捕捉模式，每 4 个上升沿捕捉一次
 011 = 捕捉模式，每 1 个上升沿捕捉一次
 010 = 捕捉模式，每 1 个下降沿捕捉一次
 001 = 捕捉模式，每个边沿（上升沿和下降沿）捕捉一次
 （ICI<1:0> 不控制该模式下的中断产生。）
 000 = 输入捕捉模块关闭

图注：

HC = 由硬件清零

HS = 由硬件置 1

R = 可读位

W = 可写位

U = 未用位，读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

13.3 定时器选择

每个 dsPIC30F 器件可能有一个或多个输入捕捉通道。每个通道都可以选择两个 16 位定时器之一作为时基。关于可以被选用的具体定时器，请参见器件数据手册。

可以通过 ICTMR 控制位 (ICxCON<7>) 实现定时器源的选择。定时器可以被设置为使用内部时钟源 (Fosc/4) 或使用在 TxCK 引脚上外接的同步外部时钟源。

13.4 输入捕捉事件模式

当 ICx 引脚上有事件发生时，输入捕捉模块捕捉所选的时基寄存器的 16 位值。可以被捕捉的事件分为下列 3 类：

1. 简单捕捉事件模式
 - 在 ICx 引脚输入电平的下降沿捕捉定时器值
 - 在 ICx 引脚输入电平的上升沿捕捉定时器值
2. 每个边沿（上升和下降）都捕捉定时器值
3. 预分频捕捉事件模式
 - 在 ICx 引脚输入电平的第 4 个上升沿捕捉定时器值
 - 在 ICx 引脚输入电平的第 16 个上升沿捕捉定时器值

通过设置相应的输入捕捉模式位 ICM<2:0> (ICxCON<2:0>) 可以配置上述输入捕捉模式。

13.4.1 简单捕捉事件

捕捉模块能够根据 ICx 引脚上输入信号的边沿选择（捕捉模式定义的上升沿或下降沿）捕捉定时器计数值 (TMR2 或 TMR3)。这些模式可以通过分别设置 ICM<2:0> (ICxCON<2:0>) 位为 “010” 或 “011” 来指定。在这些模式下，不使用预分频计数器。简单捕捉事件的简化时序图，参见图 13-3 和图 13-2。

输入捕捉逻辑电路根据内部相位时钟检测和同步捕捉引脚信号的上升或下降沿。如果出现上升 / 下降沿，捕捉模块逻辑将会把当前时基值写入捕捉缓冲器并发信号给中断产生逻辑。当发生的捕捉事件的数量与 ICI<1:0> 控制位指定的数量匹配时，那么相应的捕捉通道中断标志位 ICxIF 将会在捕捉缓冲器写事件之后 2 个指令周期置 1。

如果捕捉时基在每个指令周期都加 1 的话，捕捉到的计数值将会是 ICx 引脚有事件发生后 1 或 2 个指令周期出现的值。这个延时是随 ICx 沿事件而变化的，而实际上 ICx 沿事件与指令周期时钟和输入捕捉逻辑电路延时相关。如果到捕捉时基的输入时钟被预分频，那么捕捉值的延时将被消除。详情见图 13-3 和图 13-2。

输入捕捉引脚有最小高低电平时间规范。详细信息请参见器件数据手册的“电气规范”部分。

图 13-2: 简单捕捉事件时序图，时基预分频比 = 1:1

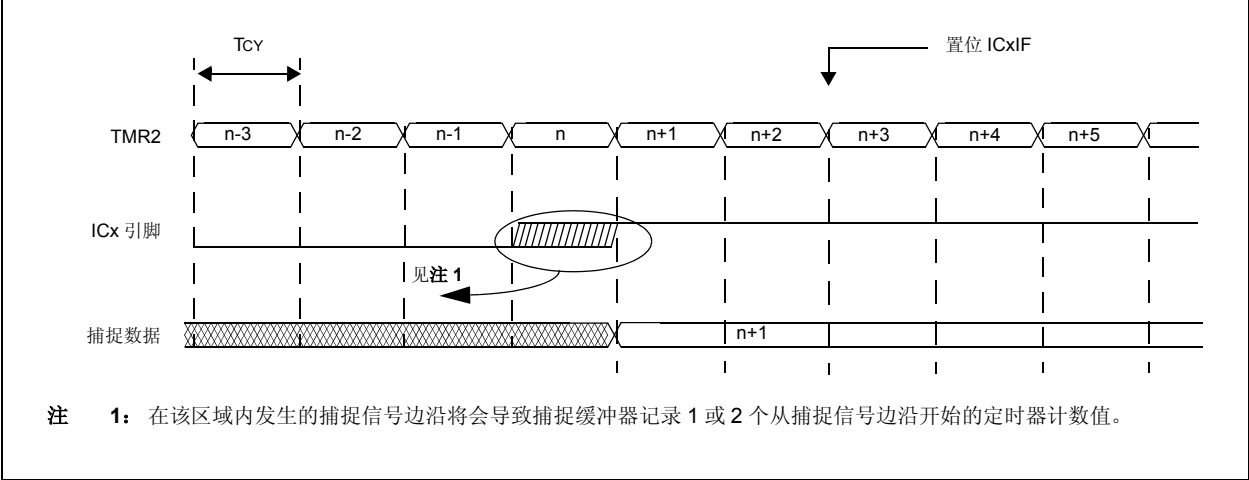
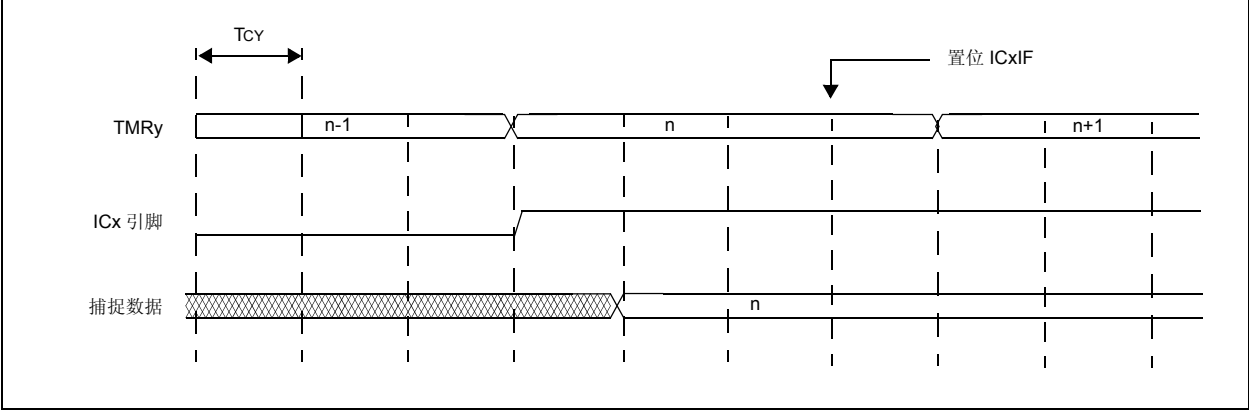


图 13-3: 简单捕捉事件时序图，时基预分频比 = 1:4



13.4.2 预分频器捕捉事件

捕捉模块有两个预分频捕捉模式。预分频模式分别由设置 ICM<2:0> (ICxCON<2:0>) 位为 “100” 或为 “101” 来选择。在这些模式下，捕捉模块每计数引脚的 4 或 16 个上升沿才发生一次捕捉事件。

捕捉预分频计数器在每个捕捉引脚的有效的上升沿上增加。引脚上的上升沿有效地作为计数器的时钟。当预分频计数器等于 4 或 16 个计数（取决于所选择的模式）时，计数器将输出一个 “有效的” 捕捉事件信号，随后将该信号与指令周期时钟同步。该同步了的捕捉事件信号将触发一个捕获缓冲写事件，同时向中断产生逻辑电路发出信号。各个捕捉通道中断状态标志 ICxIF，在捕捉缓冲写事件后的 2 个指令周期后被置 1。

如果捕捉时基在每个指令周期加 1，捕捉到的计数值将是同步捕捉事件发生后 1 或 2 个指令周期出现的值。

输入捕捉引脚有最小高低电平时间规范。更多详细信息，请参见器件数据手册的 “电气规范” 部分。

从一个预分频设置切换到另一个会产生一个中断。而且，预分频计数器将不会被清零，所以，第一个捕捉可能会从一个非零预分频比开始。例 13-1 显示了建议在不同的预分频比设置之间切换的方法。

预分频器计数器在如下情况下被清零：

- 捕捉通道被关闭（即，ICM<2:0> = 000）。
- 任何器件复位。

预分频器计数器在如下情况下不被清零：

- 用户从一个活动的捕捉模式切换到另一个模式。

例 13-1: 预分频的捕捉代码示例

```

; The following code example will set the Input Capture 1 module
; for interrupts on every second capture event, capture on every
; fourth rising edge and select Timer 2 as the time-base. This
; code example clears ICxCON to avoid unexpected interrupts.

BSET  IPC0,  #IC1IP0    ; Setup Input Capture 1 interrupt for
BCLR  IPC0,  #IC1IP1    ; desired priority level
BCLR  IPC0,  #IC1IP2    ; (this example assigns level 1 priority)
BCLR  IFS0,  #IC1IF     ; Clear the IC1 interrupt status flag
BSET  IEC0,  #IC1IE     ; Enable IC1 interrupts

CLR   IC1CON          ; Turn off Input Capture 1 Module.
MOV   #0x00A2, w0     ; Load the working register with the new
MOV   w0, IC1CON       ; prescaler mode and write to IC1CON

MOV   #IC1BUF, w0      ; Create capture data fetch pointer
MOV   #TEMP_BUFF, w1   ; Create data storage pointer
                        ; Assumes TEMP_BUFF is already defined

; The following code shows how to read the capture buffer when
; W0 contains the capture buffer address.

; Example code for Input Capture 1 ISR:
__IC1Interrupt:
BCLR  IFS0,  #IC1IF     ; Reset respective interrupt flag
MOV   [w0++], [w1++]    ; Read and save off first capture entry
MOV   [w0], [w1]        ; Read and save off second capture entry
                        ; Remaining user code here
RETFIE                  ; Return from ISR

```

注: 建议用户在切换到新的模式之前, 关闭捕捉模块 (即, 将 OCM<2:0> (ICxCON<2:0>) 清零)。如果用户切换到一个新的捕捉模式, 预分频计数器不会清零。所以, 由于预分频计数器的值为非零, 有可能 (在模式切换时) 产生第一个捕获事件及其相关的中断。

13.4.3 边沿检测模式

捕捉模块可以在 ICx 引脚上输入信号的每个上升和下降沿捕捉一个时基计数值。边沿检测模式通过设置 ICM<2:0> (ICxCON<2:0>) 位为 “001” 来选择。在该模式下，没有使用捕捉预分频计数器。简化的时序图请参考图 13-4。

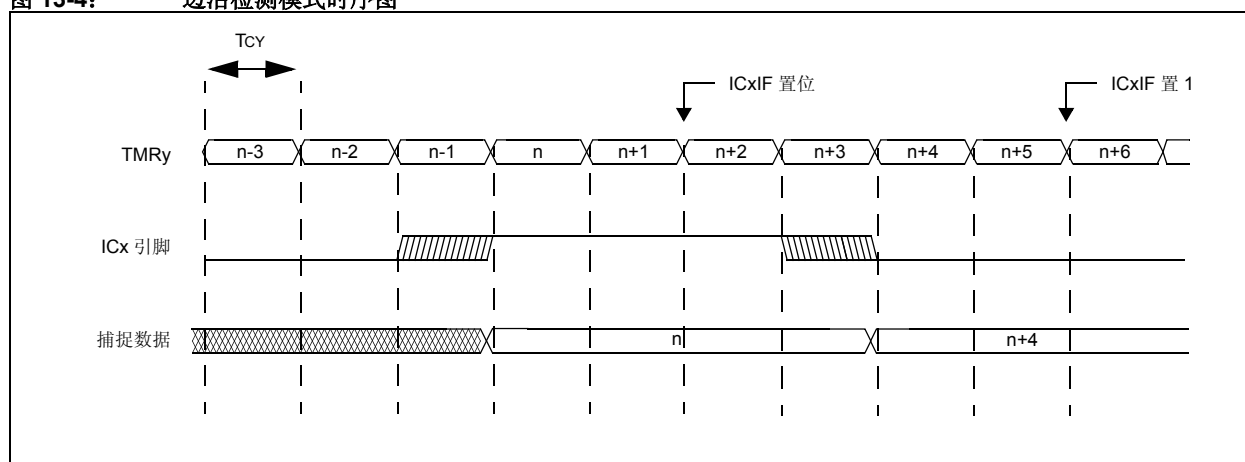
当输入捕捉模块被配置成边沿检测模式时，该模块将：

- 在每个上升沿或下降沿将输入捕捉中断标志置 1 (ICxIF)。
- 没有在这种模式中使用捕捉时中断模式位 ICI<1:0> (ICxCON<6:5>)。每个捕捉事件都将产生中断。
- 不置位捕捉溢出位 ICOV (ICxCON<4>)。

与简单捕获事件模式相同，输入捕捉逻辑电路根据内部相位时钟检测和同步捕捉引脚信号的上升和下降沿。如果发生上升或下降沿，捕捉模块逻辑电路将当前的时基值写到捕捉缓冲区，然后向中断产生逻辑电路发出信号。相应的捕捉通道中断状态标志 ICxIF，在捕捉缓冲写事件后的 2 个指令周期后被置 1。

捕捉定时器计数值为在 ICx 引脚上的边沿发生后的 1 或 2 个 Tcy (指令周期) 的值 (参见图 13-4)。

图 13-4: 边沿检测模式时序图



13.5 捕获缓冲器的操作

每个捕捉通道有一个与之相关的 4 级深的 FIFO 缓冲器。因为 ICxBUF 寄存器是存储器映射的，所以它是用户可见的缓冲寄存器。

当输入捕捉模块复位时，ICM<2:0> = 000 (ICxCON<2:0>)，输入捕捉逻辑电路将：

- 清零溢出条件标志 (即，清零 ICxOV (ICxCON<4>))。
- 复位捕捉缓冲为空状态 (即，清零 ICBNE <3>)。

在下列条件下读 FIFO 缓冲器将导致不确定的结果：

- 在输入捕捉模块先被禁用，一段时间以后重新被使能时。
- 在 FIFO 缓冲为空时对 FIFO 缓冲器执行读操作时。
- 在器件复位后。

有两个状态标志提供 FIFO 缓冲器的状态。

- ICBNE (ICxCON<3>)：输入捕捉缓冲器非空
- ICOV (ICxCON<4>)：输入捕捉溢出

13.5.1 输入捕获缓冲器非空 (ICBNE)

ICBNE 只读状态位 (ICxCON<3>) 在第一个输入捕捉事件发生时被置 1, 并且一直保持置 1 状态, 直到所有的捕捉事件都从捕捉缓冲器中读取。举例来说, 如果发生了三个捕捉事件, 那么, 必需对捕捉缓冲器进行 3 次读操作后才能将 ICBNE (ICxCON<3>) 标志清零。如果发生了四个捕捉事件, 那么必须进行四次读操作才可清零 ICBNE (ICxCON<3>) 标志。捕捉缓冲器的每次读操作都将使余下的字移动到下一个可用的栈顶单元。由于 ICBNE 反映捕捉缓冲器的状态, ICBNE 状态位在器件复位时将清零。

13.5.2 输入捕捉溢出 (ICOV)

当捕捉缓冲器溢出时, ICOV 只读状态位 (ICxCON<4>) 将被置 1。在缓冲器被四个捕捉缓冲事件充满, 而在读缓冲器之前发生第五个捕捉事件的情况下, 将产生溢出条件, ICOV (ICxCON<4>) 位将被设置成逻辑 1, 而且不产生相应的捕捉事件中断。另外, 第五个捕捉事件将不被纪录, 而且之后所有的捕捉事件将不会改变当前的缓冲器内容。

为了清除溢出条件, 捕捉缓冲器必须被读取 4 次。在第四次读取时, ICOV (ICxCON<4>) 状态标志将被清零, 捕捉通道将恢复正常的工作。

清除溢出条件可以通过下面的方法来实现:

- 设置 ICM<2:0> (ICxCON<2:0>) = 000
- 读捕捉缓冲器, 直到 ICBNE (ICxCON<3>) = 0
- 任何器件复位

13.5.2.1 ICOV 和只中断模式

输入捕捉模块还可以配置为外部中断引脚功能。要配置为这种模式, ICI<1:0> (ICxCON<6:5>) 位必须被设置为 “00”。中断的产生与缓冲器的读操作无关。

13.6 输入捕捉中断

输入捕捉模块能根据选定的捕捉事件的次数来产生中断。捕捉事件定义为将时基值写入捕捉缓冲器中。这种设置通过控制位 ICI<1:0> (ICxCON<6:5>) 来配置。

除非在 ICI<1:0> = 00 的情况下, 否则在缓冲器溢出条件清除前将不产生任何中断 (参见第 13.5.2 节 “输入捕捉溢出 (ICOV)”)。当捕捉缓冲器通过复位条件或读操作被清空时, 中断计数将会复位。这使中断计数得以与 FIFO 入口状态重新同步。

13.6.1 中断控制位

每个输入捕捉通道都有中断标志状态位 (ICxIF)、中断允许位 (ICxIE) 和中断优先级控制位 (ICxIP<2:0>)。关于外设中断更多的信息, 请参见第 6 章 “复位中断”。

13.7 UART 自动波特率支持

当 UART 配置为自动波特率工作模式, ABAUD = 1 (UxMODE<5>) 时, 输入捕捉模块可以被 UART 模块使用。当 ABAUD 控制位置 1 时, UART 的 RX 引脚在内部被连接到指定的输入捕捉模块的输入端。和捕捉模块相关的 I/O 引脚将被断开。波特率可以通过在接收到 NULL 字符时测量起始位的宽度来确定。注意, 要利用自动波特率功能, 捕捉模块必须设置为边沿检测模式 (在每个上升或下降沿捕捉)。分配给每个 UART 的输入捕捉模块取决于所选择的不同的 PIC30F 器件。更多关于自动波特率支持的详细信息请参见器件数据手册。

13.8 低功耗状态下的输入捕捉工作

13.8.1 休眠模式下的输入捕捉工作

当器件进入休眠模式后，系统时钟被禁止。在休眠模式，输入捕捉模块只能当作外部中断源。该模式通过设置控制位 $ICM<2:0> = 111$ 来使能。在该模式下，捕捉引脚的上升沿将使设备从休眠状态中唤醒。如果对应模块中断位被使能，同时模块的优先级达到要求，将产生中断。

当捕捉模块被配置为除 $ICM<2:0> = 111$ 外的模式并且 dsPIC30F 确实进入了休眠模式时，外部引脚的任何上升或下降沿都不会产生一个从休眠模式唤醒的条件。

13.8.2 空闲模式下的输入捕捉工作

当器件进入空闲模式时，系统时钟源保持工作，但 CPU 停止执行代码。OCSIDL 位 ($OCxCON<13>$) 选择在空闲模式下捕捉模块是停止还是继续工作。

如果 $ICSIDL = 0$ ($ICxCON<13>$)，该模块在空闲模式将继续工作。此时输入捕捉模块具备完整功能，包括 4:1 和 16:1 捕捉预分频比的设置，它们通过控制位 $ICM<2:0>$ ($ICxCON<2:0>$) 来定义。这些模式要求在空闲模式下所选择的定时器是使能的。

如果输入捕捉配置为 $ICM<2:0> = 111$ 的模式，输入捕捉引脚将仅作为外部中断引脚。在该模式下，捕捉引脚上的上升沿将导致器件从空闲模式唤醒。捕捉时基不一定要使能。如果相应的模块中断允许位被置位并且用户指定的优先级高于当前 CPU 的优先级，将产生中断。

如果 $ICSIDL = 1$ ($ICxCON<13>$)，模块在空闲模式将停止工作。模块在空闲模式停止工作时将执行和在休眠模式中一样的功能（参见第 13.8.1 节“休眠模式下的输入捕捉工作”）。

13.8.3 器件从休眠 / 空闲中唤醒

如果器件在空闲或休眠模式下，输入捕捉事件能使唤醒器件或产生中断（如果中断被允许）。

当捕捉事件发生时，如果下列事件为真，则输入捕捉模块将从休眠或空闲模式中唤醒，而与定时器被使能无关：

- 输入捕捉模式位， $ICM<2:0> = 111$ ($ICxCON<2:0>$)，且
- 中断允许位 ($ICxIE$) 置位。

同样的唤醒特性将中断 CPU，如果：

- 相应的中断被允许 ($ICxIE = 1$)，并且具有所需要的优先级。

该唤醒特性对于增加额外的外部引脚中断很有用。当在这个模式下使用输入捕捉模块时，下面的情况为真：

- 在该模式下，未使用捕捉预分频计数器。
- $IC1<1:0>$ ($ICxCON<6:5>$) 位不适用。

13.9 I/O 引脚控制

当使能捕捉模块时，用户必须通过将相关的 TRIS 位置 1，以保证 I/O 引脚的方向被配置为输入。当捕捉模块使能后，引脚方向不能设置。而且，其他和该输入引脚复用的外设也必须禁止。

13.10 与输入捕捉模块相关的特殊功能寄存器

表 13-1: 输入捕捉模块的存储器映射示例

SFR 名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位状态
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0F	0000 0000 0000 0000
IFS1	0086	IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IEC0	008C	CNIE	MI2CIE	SI2CIE	IR12	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	IC6IE	EI30	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			0100 0100 0100 0100
IPC1	0096	—	T31P<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>			0100 0100 0100 0100
IPC4	009C	—	OC3IP<2:0>			—	IC8IP<2:0>			—	IC7IP<2:0>			—	INT1IP<2:0>			0100 0100 0100 0100
IPC7	00A2	—	IC6IP<2:0>			—	IC5IP<2:0>			—	IC4IP<2:0>			—	IC3IP<2:0>			0100 0100 0100 0100
IC1BUF	0140	输入 1 捕捉寄存器																uuuu uuuu uuuu uuuu
IC1CON	0142	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBNE	ICM<2:0>		0000 0000 0000 0000		
IC2BUF	0144	输入 2 捕捉寄存器																uuuu uuuu uuuu uuuu
IC2CON	0146	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBNE	ICM<2:0>		0000 0000 0000 0000		
IC3BUF	0148	输入 3 捕捉寄存器																uuuu uuuu uuuu uuuu
IC3CON	014A	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBNE	ICM<2:0>		0000 0000 0000 0000		
IC4BUF	014C	输入 4 捕捉寄存器																uuuu uuuu uuuu uuuu
IC4CON	014E	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBNE	ICM<2:0>		0000 0000 0000 0000		
IC5BUF	0150	输入 5 捕捉寄存器																uuuu uuuu uuuu uuuu
IC5CON	0152	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBNE	ICM<2:0>		0000 0000 0000 0000		
IC6BUF	0154	输入 6 捕捉寄存器																uuuu uuuu uuuu uuuu
IC6CON	0156	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBNE	ICM<2:0>		0000 0000 0000 0000		
IC7BUF	0158	输入 7 捕捉寄存器																uuuu uuuu uuuu uuuu
IC7CON	015A	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBNE	ICM<2:0>		0000 0000 0000 0000		
IC8BUF	015C	输入 8 捕捉寄存器																uuuu uuuu uuuu uuuu
IC8CON	015E	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>	ICOV	ICBNE	ICM<2:0>		0000 0000 0000 0000		

图注: u = 未初始化

注: 特定存储器映射有关细节参见器件数据手册。

13.11 设计技巧

问 1: *可以使用输入捕捉模块将器件从休眠模式唤醒吗?*

答: 可以。当输入捕捉模块配置为 $ICM<2:0> = 111$ ，同时各通道的中断允许位被置 1, $ICxIE = 1$ 时，捕捉引脚上的上升沿会将器件从休眠中唤醒（参见第 13.8 节“低功耗状态下的输入捕捉工作”）。

13.12 相关应用笔记

本节列出了和手册的本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与输入捕捉模块相关的应用笔记有：

标题	应用笔记编号
Using the CCP Module(s)	AN594
Implementing Ultrasonic Ranging	AN597

注：如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

13.13 版本历史

版本 A

这是本文档的初始发行版。

版本 B

本章未包含任何技术内容或编辑排版上的修订，然而我们谨更新此节以反映本手册已更新到版本 B。

版本 C

本章未包含任何技术内容修订，然而我们谨更新此节以反映本手册已更新到版本 C。

第 14 章 输出比较

目录

本章包括下列主题：

14.1	简介	14-2
14.2	输出比较寄存器	14-3
14.3	工作模式	14-4
14.4	低功耗状态下的输出比较工作	14-23
14.5	I/O 引脚控制	14-23
14.6	设计技巧	14-26
14.7	相关应用笔记	14-27
14.8	版本历史	14-28

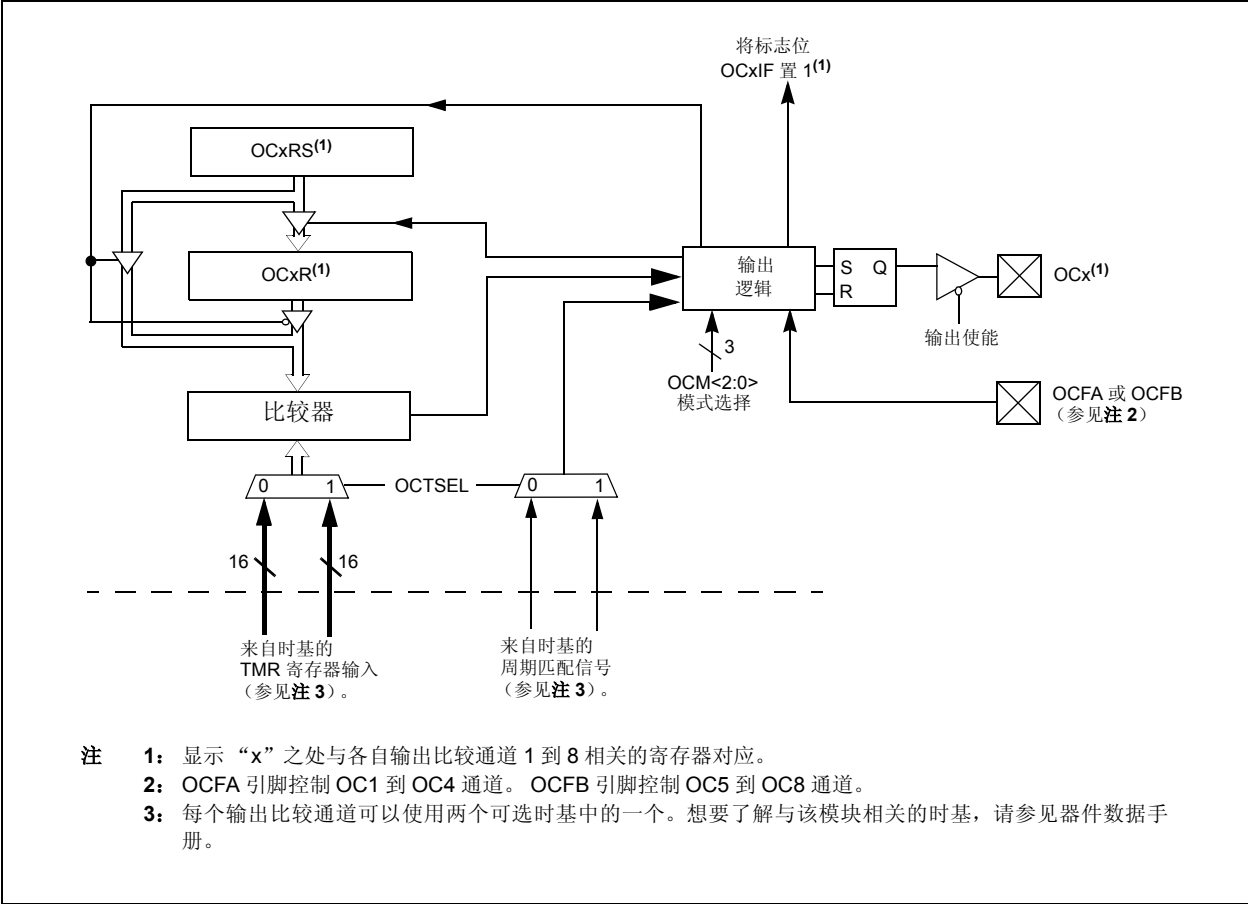
14.1 简介

输出比较模块具有将所选时基值与一个或两个比较寄存器的值（取决于所选的工作模式）作比较的功能。此外，它在比较匹配事件发生时能产生单个输出脉冲或一连串输出脉冲。如同大多数 dsPIC 外设一样，它在匹配事件发生时也能产生中断。

dsPIC30F 器件可有至多八个输出比较通道，以符号 OC1、OC2 和 OC3 等表示。关于在某个特定器件上可用的通道数，请参见特定器件数据手册。所有输出比较通道在功能上是都相同的。在本章中，引脚、寄存器或位名称中的“x”表示特定的输出比较通道。

每个输出比较通道可以使用两个可选时基中的一个。使用 OCTSEL 位（OCxCON<3>）选择时基。有关可以与各个编号的输出比较通道一同使用的特定定时器的信息，请参见器件数据手册。

图 14-1: 输出比较模块框图



- 注 1: 显示“x”之处与各自输出比较通道 1 到 8 相关的寄存器对应。
- 注 2: OCFA 引脚控制 OC1 到 OC4 通道。OCFB 引脚控制 OC5 到 OC8 通道。
- 注 3: 每个输出比较通道可以使用两个可选时基中的一个。想要了解与该模块相关的时基，请参见器件数据手册。

14.2 输出比较寄存器

每个输出比较通道均有下列寄存器：

- OCxCON：通道的控制寄存器
- OCxR：输出比较通道的数据寄存器
- OCxRS：输出比较通道的辅助数据寄存器

八个比较通道的控制寄存器被命名为 OC1CON 到 OC8CON。所有八个控制寄存器的位定义均相同，表示为以下公共寄存器定义。OCxCON 中的 “x” 表示输出比较通道的编号。

寄存器 14-1： OCxCON：输出比较 x 控制寄存器

高字节:							
U-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
—	—	OCSIDL	—	—	—	—	—
bit 15							bit 8

低字节:							
U-0	U-0	U-0	R-0, HC	R/W-0	R/W-0	R/W-0	
—	—	—	OCFLT	OCTSEL	OCM<2:0>		
bit 7							

- bit 15-14 未用：读作 0
- bit 13 **OCSIDL**：在空闲模式下停止输出比较控制位
1 = 输出比较 x 将在 CPU 空闲模式下停止
0 = 输出比较 x 将在 CPU 空闲模式下继续工作
- bit 12-5 未用：读作 0
- bit 4 **OCFLT**：PWM 错误条件状态位
1 = 产生了 PWM 错误条件（仅可在硬件中清零）
0 = 未产生 PWM 错误条件
（仅当 OCM<2:0> = 111 时，才使用该位。）
- bit 3 **OCTSEL**：输出比较定时器选择位
1 = Timer 3 是比较 x 的时钟源
0 = Timer 2 是比较 x 的时钟源
- 注： 要了解输出比较模块可用的特定时基，请参见器件数据手册。
- bit 2-0 **OCM<2:0>**：输出比较模式选择位
111 = OCx 处于 PWM 模式，错误引脚使能
110 = OCx 处于 PWM 模式，错误引脚禁止
101 = 初始化 OCx 引脚为低电平，在 OCx 引脚上产生连续的输出脉冲
100 = 初始化 OCx 引脚为低电平，在 OCx 引脚上产生单个输出脉冲
011 = 比较匹配事件使 OCx 引脚的电平交替翻转
010 = 初始化 OCx 引脚为高电平，比较匹配事件强制 OCx 引脚为低电平
001 = 初始化 OCx 引脚为低电平，比较匹配事件强制 OCx 引脚为高电平
000 = 输出比较通道禁止

图注:			
HC = 由硬件清零			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

14.3 工作模式

每个输出比较模块均有以下工作模式：

- 单比较匹配模式
- 双比较匹配模式产生
 - 单个输出脉冲
 - 连续输出脉冲
- 简单脉宽调制模式
 - 带有故障保护输入
 - 不带故障保护输入

注 1：建议用户在切换到新的模式之前，关闭输出比较模块（即，将 OCM<2:0>（OCxCON<2:0>）清零）。

2：在本章中，对与所选的定时器源相关的任何 SFR 的引用，均用“y”下标表示。例如，PRy 是所选定时器源的周期寄存器，而 TyCON 是所选定时器源的定时器控制寄存器。

14.3.1 单比较匹配模式

当控制位 OCM<2:0>（OCxCON<2:0>）设置为“001”、“010”或“011”时，所选的输出比较通道被分别配置为三种单输出比较匹配模式中的一种模式。

在单比较模式下，把一个值装入 OCxR 寄存器，并将该值与所选的递增计数器寄存器 TMRy 的值作比较。当比较匹配事件发生时，将产生下列事件之一：

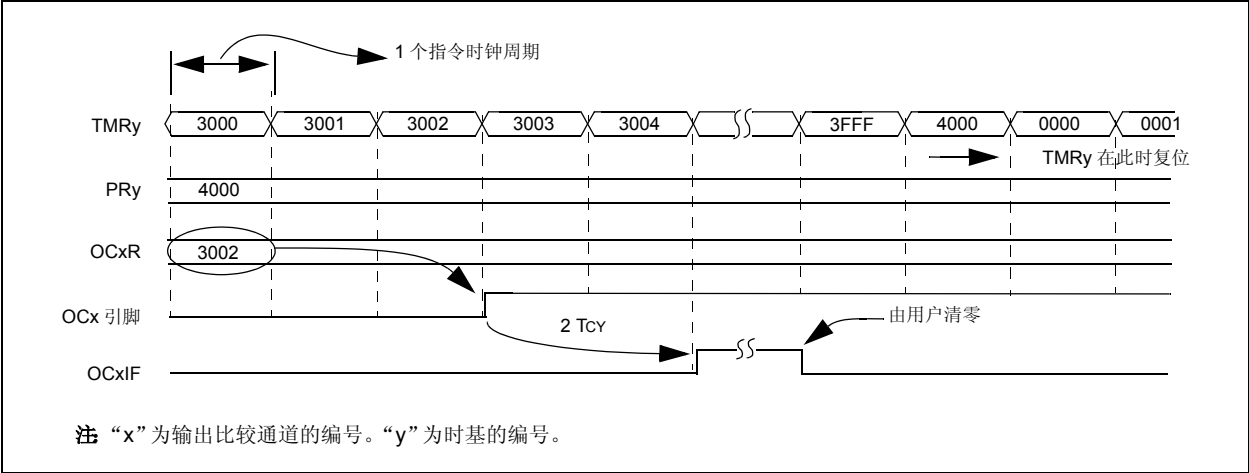
- 当 OCx 引脚的初始状态为低电平时，比较匹配事件强制该引脚为高电平。在单比较匹配事件发生时，产生中断。
- 当 OCx 引脚的初始状态为高电平时，比较匹配事件强制该引脚为低电平。在单比较匹配事件发生时，产生中断。
- 比较匹配事件使 OCx 引脚电平交替翻转。翻转事件是连续的，且每次翻转事件都会产生一次中断。

14.3.1.1 比较模式输出驱动为高电平

要将输出比较模块配置为这种模式，请设置控制位 $OCM<2:0> = 001$ 。还应该使能比较时基。一旦使能了此比较模式，输出引脚 OCx 将初始化驱动为低电平，并保持该低电平直到 $TMRy$ 和 $OCxR$ 寄存器之间发生匹配。参见图 14-2，注意以下关键时序：

- 在比较时基与 $OCxR$ 寄存器发生比较匹配后的下一个指令时钟， OCx 引脚驱动为高电平。该 OCx 引脚将保持高电平直到改变模式或该模块被禁止。
- 比较时基将计数到相关的周期寄存器中包含的值后，在下一个指令时钟复位为 $0x0000$ 。
- 在 OCx 引脚驱动为高电平后再过两个指令时钟，相应通道的中断标志位 $OCxIF$ 被置 1。

图 14-2： 单比较模式：在比较匹配事件发生时设置 OCx 为高电平

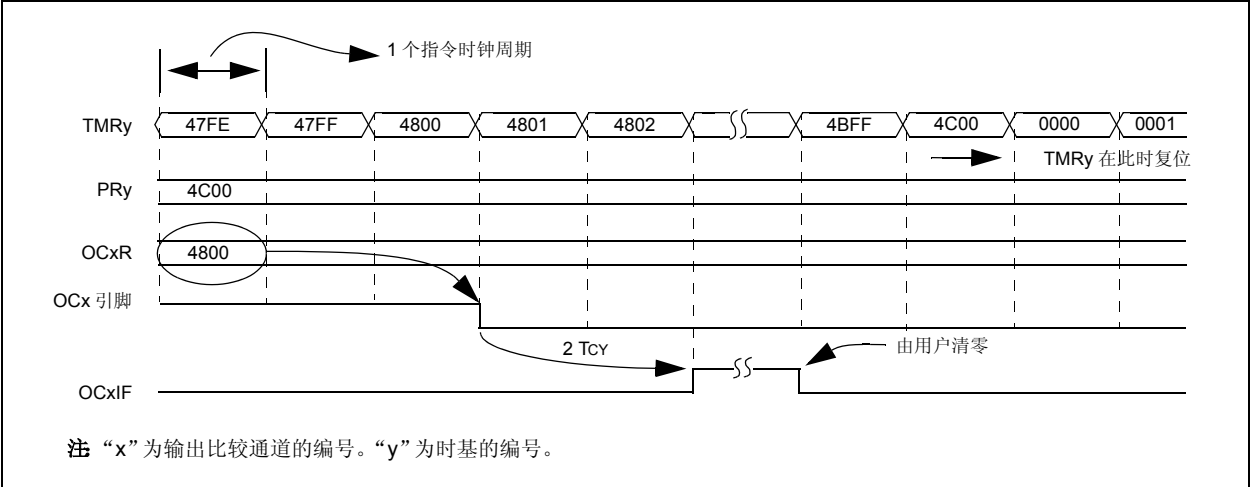


14.3.1.2 比较模式输出驱动为低电平

要将输出比较模块配置为这种模式，请设置控制位 $OCM<2:0> = 010$ 。还必须使能比较时基。一旦使能了此比较模式，输出引脚 OCx 将初始化驱动为高电平，并保持该高电平直到定时器和 $OCxR$ 寄存器之间发生匹配。参见图 14-3，注意以下关键时序事件：

- 在比较时基与 $OCxR$ 寄存器发生比较匹配后的下一个指令时钟， OCx 引脚驱动为低电平。该 OCx 引脚将保持低电平直到改变模式或该模块被禁止。
- 比较时基将计数到相关的周期寄存器中包含的值后，在下一个指令时钟复位为 $0x0000$ 。
- 在 OCx 引脚驱动为低电平后再过两个指令时钟，相应通道的中断标志位 $OCxIF$ 被置 1。

图 14-3： 单比较模式：在比较匹配事件发生时强制 OCx 为低电平



14.3.1.3 单比较模式电平交替翻转输出

要将输出比较模块配置为这种模式，请设置控制位 $OCM<2:0> = 011$ 。此外，必须选择并使能 Timer 2 或 Timer 3。一旦使能了此比较模式，输出引脚 OCx 将初始化驱动为低电平，并在随后每一次当时基和 $OCxR$ 寄存器之间发生匹配事件时，交替输出高低电平。参见图 14-4 和图 14-5，注意以下关键时序事件：

- 在比较时基与 $OCxR$ 寄存器发生比较匹配后的下一个指令时钟， OCx 引脚电平翻转。该 OCx 引脚将保持此新状态直到发生下一次翻转事件、改变模式或该模块被禁止。
- 比较时基将计数到相关周期寄存器中的值后，在下一个指令时钟复位为 $0x0000$ 。
- 在 OCx 引脚电平翻转之后再过两个指令时钟，相应通道的中断标志位 $OCxIF$ 被置 1。

注： 器件复位时，内部 OCx 引脚输出逻辑被设置为逻辑“0”。但是，在交替翻转模式下， OCx 引脚的工作状态可以通过用户软件设置。例 14-1 所示为在交替翻转工作模式下，定义所需的初始化 OCx 引脚状态的代码示例。

图 14-4： 单比较模式：在比较匹配事件发生时输出电平翻转（ $PR2 > OCxR$ ）

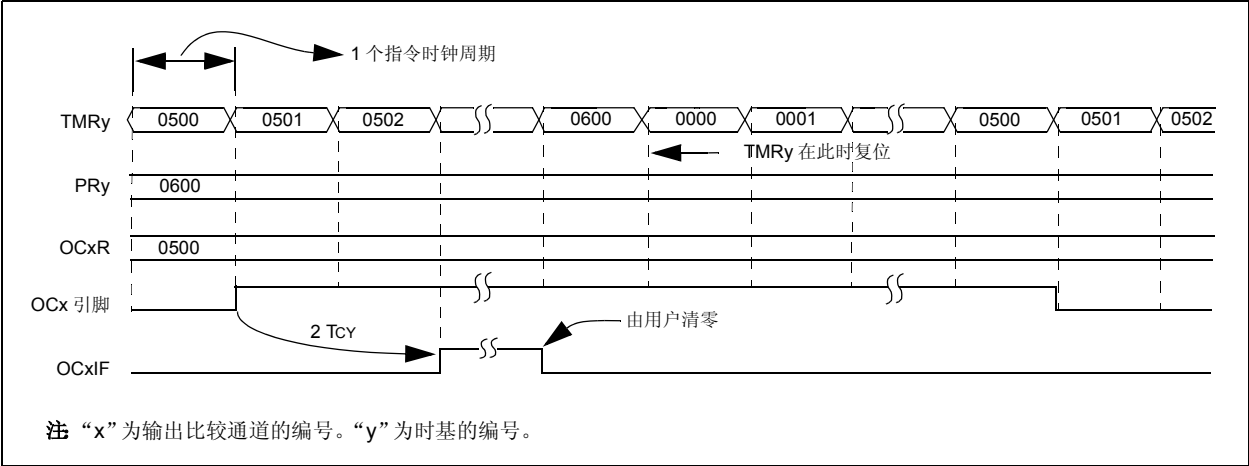
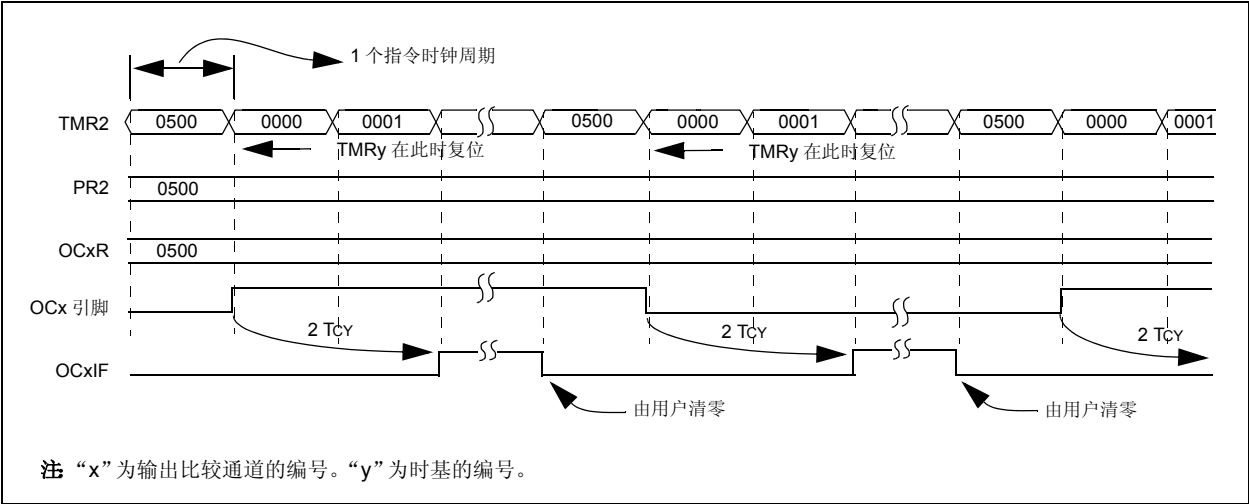


图 14-5： 单比较模式：在比较匹配事件发生时输出翻转（ $PR2=OCxR$ ）



例 14-1: 比较模式电平交替翻转模式引脚状态设置

```
; The following code example illustrates how to define the initial
; OC1 pin state for the output compare toggle mode of operation.

; Toggle mode with initial OC1 pin state set low

MOV     0x0001, w0      ; load setup value into w0
MOV     w0, OC1CON      ; enable module for OC1 pin low, toggle high
BSET    OC1CON, #1      ; set module to toggle mode with initial pin
                        ; state low

; Toggle mode with initial OC1 pin state set high

MOV     0x0002, w0      ; load setup value into w0
MOV     w0, OC1CON      ; enable module for OC1 pin high, toggle low
BSET    OC1CON, #0      ; set module to toggle mode with initial pin
                        ; state high
```

例 14-2 给出了单比较模式电平交替翻转事件的配置和中断服务程序的代码示例。

例 14-2: 比较模式电平交替翻转设置和中断服务程序

```
; The following code example will set the Output Compare 1 module
; for interrupts on the toggle event and select Timer 2 as the clock
; source for the compare time-base. It is assumed in that Timer 2
; and Period Register 2 are properly configured. Timer 2 will
; be enabled here.

CLR     OC1CON           ; Turn off Output Compare 1 Module.
MOV     #0x0003, w0      ; Load the working register with the new
MOV     w0, OC1CON       ; compare mode and write to OC1CON
MOV     #0x0500, w0      ; Initialize Compare Register 1
MOV     w0, OC1R         ; with 0x0500
BSET    IPC0, #OC1IP0    ; Setup Output Compare 1 interrupt for
BCLR    IPC0, #OC1IP1    ; desired priority level
BCLR    IPC0, #OC1IP2    ; (this example assigns level 1 priority)
BCLR    IFS0, #OC1IF     ; Clear Output Compare 1 interrupt flag
BSET    IEC0, #OC1IE     ; Enable Output Compare 1 interrupts
BSET    T2CON, #TON      ; Start Timer2 with assumed settings

; Example code for Output Compare 1 ISR:

__OC1Interrupt:
    BCLR    IFS0, #OC1IF  ; Reset respective interrupt flag
                        ; Remaining user code here
    RETFIE                ; Return from ISR
```

14.3.2 双比较匹配模式

当控制位 $OCM<2:0> = 100$ 或 101 ($OCxCON<2:0>$) 时, 所选的输出比较通道被配置为如下两种双比较匹配模式之一:

- 单输出脉冲模式
- 连续输出脉冲模式

在双比较模式下, 该模块在处理比较匹配事件时使用 $OCxR$ 和 $OCxRS$ 寄存器。将 $OCxR$ 寄存器的值与递增计数器 $TMRy$ 的计数值作比较, 并且在比较匹配事件发生时, 在 OCx 引脚上产生脉冲的前 (上升) 沿。然后 $OCxRS$ 寄存器与同一个递增计数器 $TMRy$ 的计数值作比较, 并且在比较匹配事件发生时, 在 OCx 引脚上产生脉冲的后 (下降) 沿。

14.3.2.1 双比较模式: 单输出脉冲

要将输出比较模块配置为单输出脉冲模式, 请设置控制位 $OCM<2:0> = 100$ 。另外, 必须选择并使能比较时基。一旦使能了此模式, 输出引脚 OCx 将驱动为低电平, 并保持该低电平直到时基和 $OCxR$ 寄存器之间发生匹配。参见图 14-6 和图 14-7, 注意以下一些关键时序事件:

- 在比较时基与 $OCxR$ 寄存器发生比较匹配后的下一个指令时钟, OCx 引脚驱动为高电平。 OCx 引脚将保持为高电平直到时基和 $OCxRS$ 寄存器之间发生下一次匹配事件。此时, 该引脚将驱动为低电平。该 OCx 引脚将保持低电平直到改变模式或该模块被禁止。
- 比较时基将计数到相关的周期寄存器中包含的值后, 在下一个指令时钟复位为 $0x0000$ 。
- 如果比较时基周期寄存器包含的值小于 $OCxRS$ 寄存器包含的值, 那么就不会产生脉冲的下降沿。 OCx 引脚将保持高电平直到 $OCxRS \leq PRy$ 、模式改变或复位条件产生。
- 在 OCx 引脚被驱动为低电平后 (单脉冲的下降沿) 再过 2 个指令时钟, 相应通道的中断标志位 $OCxIF$ 被置 1。

图 14-6 给出了通用双比较模式产生输出单个脉冲的过程。图 14-7 给出了另一个时序示例, 图中, $OCxRS > PRy$ 。在此示例中, 不产生脉冲下降沿, 因为比较时基在计数达到 $0x4100$ 前就复位了。

图 14-6: 双比较模式

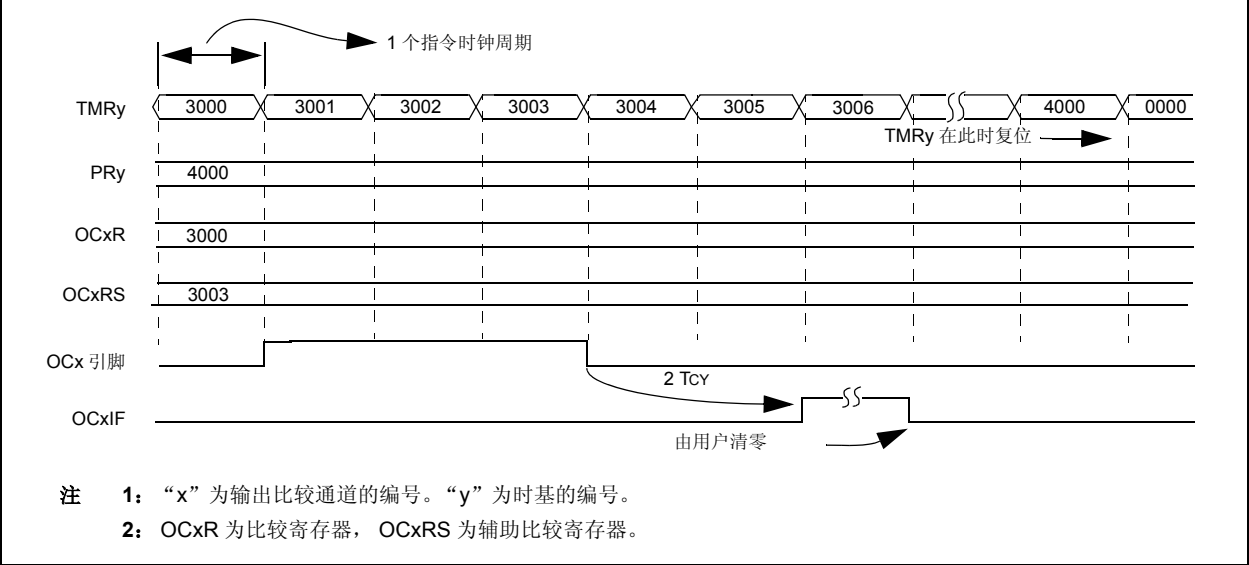
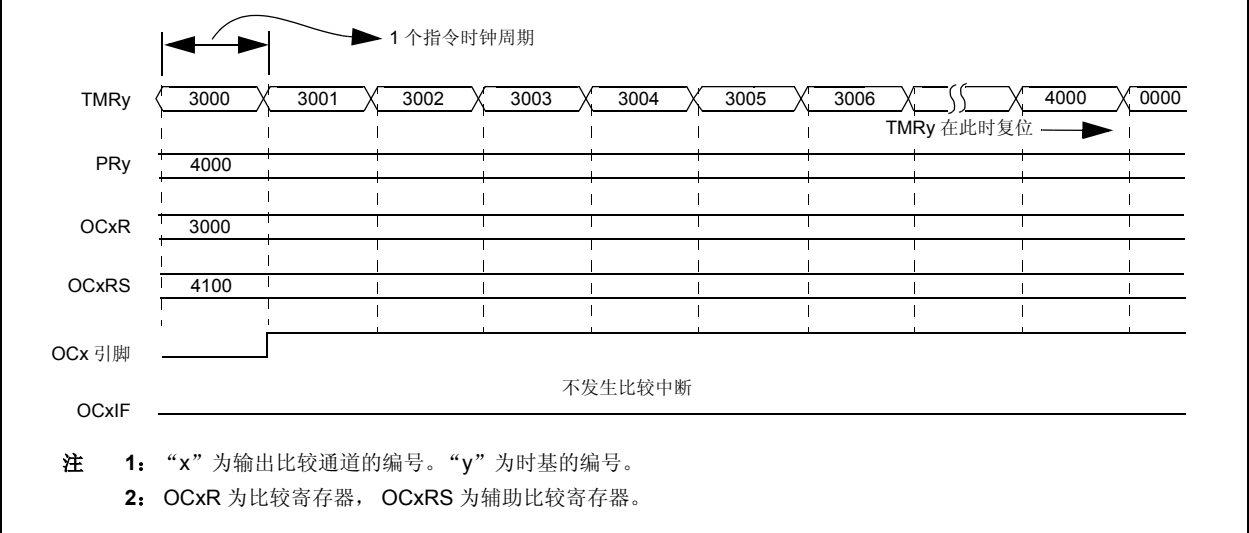


图 14-7: 双比较模式: 单输出脉冲 (OCxRS > PR2)



14.3.2.2 设置产生单脉冲输出

当控制位 $OCM<2:0>$ ($OCxCON<2:0>$) 设置为 “100” 时, 所选的输出比较通道将 OCx 引脚初始化为低电平并产生单输出脉冲。

要产生单输出脉冲, 需要遵循以下步骤 (这些步骤假设定时器源起初是关闭的, 但这并不是对模块工作的要求):

1. 确定指令时钟周期时间。将定时器源的外部时钟频率 (如果使用了) 和定时器预分频比的设置考虑进去。
2. 计算相对于 $TMRy$ 起始值 ($0x0000$) 的到达输出脉冲上升沿的时间。
3. 根据所需的脉冲宽度和脉冲上升沿时间计算到达脉冲下降沿的时间。
4. 将以上步骤2和步骤3中计算出的值分别写入比较寄存器 $OCxR$ 和辅助比较寄存器 $OCxRS$ 。
5. 将定时器周期寄存器 PRy 的值设置为大于或等于辅助比较寄存器 $OCxRS$ 中的值。
6. 设置 $OCM<2:0> = 100$, 并将 $OCTSEL$ ($OCxCON<3>$) 位设置为所需定时器源的对应值。此时 OCx 引脚状态被驱动为低电平。
7. 将 TON ($TyCON<15>$) 位置为 “1”, 它将使能比较时基并开始计数。
8. 在 $TMRy$ 和 $OCxR$ 第一次匹配时, OCx 引脚将被驱动为高电平。
9. 当递增计数器 $TMRy$ 和辅助比较寄存器 $OCxRS$ 发生匹配时, 在 OCx 引脚上驱动输出脉冲的第二个边沿 (即后沿, 从高到低)。 OCx 引脚上不会驱动输出更多脉冲, OCx 引脚将保持为低电平。发生第二次比较匹配事件后, 会导致 $OCxIF$ 中断标志位置 1, 如果已通过置位 $OCxIE$ 位允许了该中断, 那么将产生中断。更多有关外设中断的信息, 请参见第 6 章 “复位中断”。
10. 如果需要再次启动单脉冲输出的话, 要更改定时器和比较寄存器的设置, 然后进行写操作将 $OCM<2:0>$ ($OCxCON<2:0>$) 位设置为 “100”。不要求禁止或重新使能定时器并清零 $TMRy$ 寄存器, 但这样做对于从已知的事件时间边界定义脉冲很有利。

在输出脉冲的出现下降沿之后, 不必禁止输出比较模块。重写 $OCxCON$ 寄存器的值可以启动另一个脉冲。

例 14-3 给出了配置单输出脉冲事件的代码示例。

例 14-3: 单输出脉冲设置和中断服务程序

```
; The following code example will set the Output Compare 1 module
; for interrupts on the single pulse event and select Timer 2
; as the clock source for the compare time base. It is assumed
; that Timer 2 and Period Register 2 are properly initialized.
; Timer 2 will be enabled here.

CLR      OC1CON          ; Turn off Output Compare 1 Module.
MOV      #0x0004, w0     ; Load the working register with the new
MOV      w0, OC1CON      ; compare mode and write to OC1CON
MOV      #0x3000, w0     ; Initialize Compare Register 1
MOV      w0, OC1R        ; with 0x3000
MOV      #0x3003, w0     ; Initialize Secondary Compare Register 1
MOV      w0, OC1RS       ; with 0x3003
BSET     IPC0, #OC1IP0    ; Setup Output Compare 1 interrupt for
BCLR     IPC0, #OC1IP1    ; desired priority level
BCLR     IPC0, #OC1IP2    ; (this example assigns level 1 priority)
BCLR     IFS0, #OC1IF     ; Clear Output Compare 1 interrupt flag
BSET     IEC0, #OC1IE     ; Enable Output Compare 1 interrupts

BSET     T2CON, #TON      ; Start Timer2 with assumed settings

; Example code for Output Compare 1 ISR:

__OC1Interrupt:
BCLR     IFS0, #OC1IF     ; Reset respective interrupt flag
; Remaining user code here
RETFIE   ; Return from ISR
```


14.3.2.3 双比较模式产生单输出脉冲的特殊情况

应该了解，根据 OCxR、OCxRS 和 PRy 值的关系，输出比较模式还有一些对应的独特条件。表 14-1 说明了这些特殊条件和这些条件所导致的模块的工作情况。

表 14-1: 双比较模式产生单输出脉冲的特殊情况

SFR 逻辑关系	特殊条件	工作原理	OCx 引脚的输出
PRy >= OCxRS 且 OCxRS > OCxR	OCxR = 0 初始化 TMRy = 0	在 TMRy 从 0x0000 到 PRy 的第一次迭代中，OCx 引脚保持为低电平，不产生脉冲。在 TMRy 复位到零后（周期匹配时），OCx 引脚由于与 OCxR 匹配驱动为高电平。在下次 TMRy 与 OCxRS 匹配时，OCx 引脚驱动为低电平并保持该低电平。第二次比较匹配导致 OCxIF 位置 1。 还有两种初始条件可供参考。 a] 初始化 TMRy = 0 并设置 OCxR >= 1 b] 初始化 MRY = PRy (PRy > 0) 并设置 OCxR = 0	根据设置的情况，脉冲将以 PRy 寄存器中的值延时
PRy >= OCxR 且 OCxR >= OCxRS	OCxR >= 1 且 PRy >= 1	TMRy 计数到 OCxR 且比较匹配事件发生时（即 MRY = OCxR）时，OCx 引脚被驱动为高电平状态。然后 TMRy 继续计数并最后在周期匹配时（即 PRy = TMRy）复位。然后定时器从 0x0000 重新开始计数直到达到 OCxRS 中的值且比较匹配事件发生（即 MRY = OCxRS），此时 OCx 引脚被驱动为低电平状态。第二次比较匹配导致 OCxIF 位置 1。	脉冲
OCxRS > PRy 且 PRy >= OCxR	无	在 OCx 引脚只产生上升沿。OCxIF 将不会置 1。	上升沿 / 转变为高电平
OCxR = OCxRS = PRy = 0x0000	无	在定时器和周期寄存器的值匹配后延迟了两个指令时钟周期，才在 OCx 引脚产生输出脉冲。第二次比较匹配将导致 OCxIF 位置 1。	延迟的脉冲
OCxR > PRy	无	不支持此模式，定时器在匹配条件发生之前复位。	保持为低电平

- 注 1: 这里考虑到的所有情形，都假设 TMRy 寄存器初始化为 0x0000。
- 2: OCxR = 比较寄存器，OCxRS = 辅助比较寄存器，TMRy = 定时器计数器，PRy = 定时器周期寄存器。

14.3.2.4 双比较模式：连续输出脉冲

要将输出比较模块配置为这种模式，请设置控制位 $OCM<2:0> = 101$ 。此外，还应该选择和使能比较时基。一旦使能了此比较模式，输出引脚 OCx 将驱动为低电平，并保持该低电平直到比较时基和 $OCxR$ 寄存器之间发生匹配。参见图 14-8 和第 14.3.2.5 节，需要注意的一些关键时序事件：

- 在比较时基与 $OCxR$ 寄存器发生比较匹配后的下一个指令时钟， OCx 引脚驱动为高电平。 OCx 引脚将保持为高电平，直到发生下一次时基和 $OCxRS$ 寄存器匹配，此时引脚被驱动为低电平。在用户不加干涉的情况下，在 OCx 引脚上会重复产生从低到高边沿和从高到低边沿的脉冲序列。
- OCx 引脚上将产生连续脉冲，直到改变模式或模块被禁止。
- 比较时基将计数到相关周期寄存器中所包含的值，然后在下一个指令时钟复位为 $0x0000$ 。
- 如果比较时基周期寄存器值小于 $OCxRS$ 寄存器值，就不会产生下降沿。 OCx 引脚将保持为高电平，直到 $OCxRS \leq PR2$ 、发生模式改变或器件复位。
- 在 OCx 引脚被驱动为低电平后（单脉冲的下降沿）再过 2 个指令时钟，相应通道的中断标志位 $OCxIF$ 置位。

图 14-8 给出了通用双比较模式产生连续输出脉冲的过程。图 14-9 给出了另一个当 $OCxRS > PRy$ 的时序示例。在此示例中，不产生脉冲下降沿，这是由于时基在计数达到 $OCxRS$ 值前就已经复位了。

图 14-8： 双比较模式：连续输出脉冲（ $PR2 = OCxRS$ ）

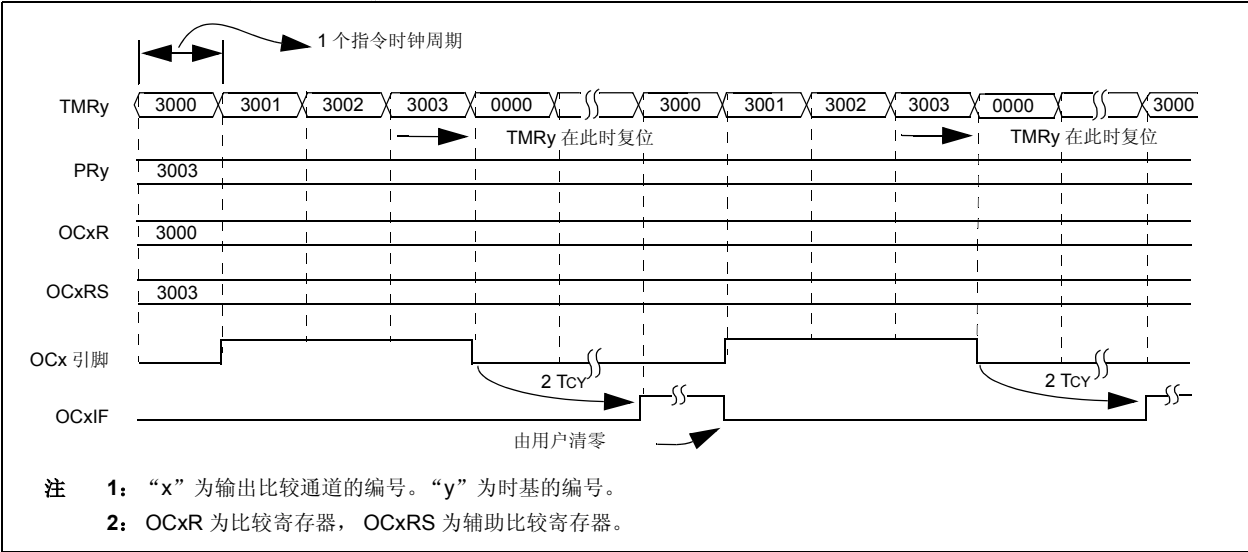
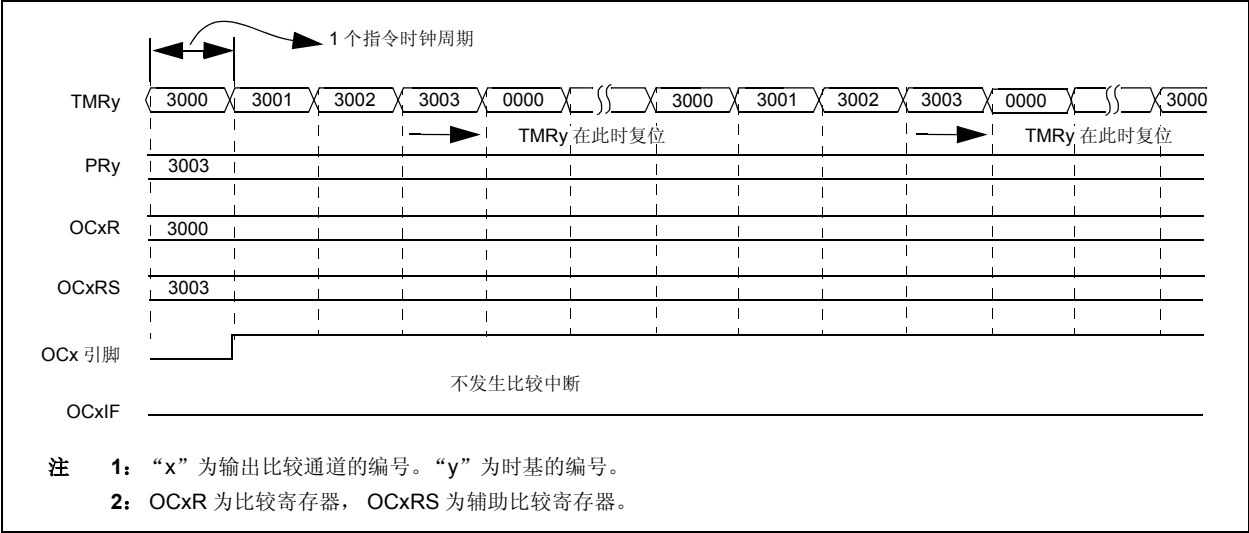


图 14-9: 双比较模式: 连续输出脉冲 (PR2 = OCxRS)



14.3.2.5 产生连续输出脉冲的设置

当控制位 OCxM<2:0> (OCxCON<2:0>) 设置为 “101” 时, 所选的输出比较通道将 OCx 引脚初始化为低电平, 并在每次比较匹配时产生脉冲输出。

用户要配置模块产生连续输出的脉冲流, 需要遵循以下步骤 (这些步骤假设定时器源在开始时是关闭的, 但是模块工作并无此要求):

1. 确定指令时钟周期时间。考虑定时器源的外部时钟频率 (如果使用了) 和定时器预分频比的设置。
2. 计算与 TMRy 起始值 (0x0000) 相对的到达输出脉冲上升沿的时间。
3. 根据所需的脉冲宽度和到达脉冲上升沿时间计算到达脉冲下降沿时间。
4. 将以上步骤2和步骤3中计算出的值分别写入比较寄存器OCxR和辅助比较寄存器OCxRS。
5. 将定时器周期寄存器 PRy 的值设置为大于或等于辅助比较寄存器 OCxRS 中的值。
6. 设置 OCM<2:0> = 101, 并将 OCTSEL (OCxCON<3>) 位设置为所需的定时器源。现在 OCx 引脚状态被驱动为低电平。
7. 将 TON (TyCON<15>) 位置为 1, 使能比较时基。
8. 在 TMRy 和 OCxR 第一次匹配时, OCx 引脚将被驱动为高电平。
9. 当比较时基 TMRy 和辅助比较寄存器 OCxRS 发生匹配时, OCx 引脚驱动输出脉冲的第二个边沿 (即后沿, 从高到低)。
10. 第二次比较匹配事件会导致 OCxIF 中断标志位置 1。
11. 当比较时基和相应的周期寄存器中的值匹配时, TMRy 寄存器复位为 0x0000 并重新开始计数。
12. 重复步骤8到步骤11, 可无限地产生连续脉冲流。OCxIF 标志位在每个 OCxRS – TMRy 比较事件发生时置 1。

例 14-4 给出了配置连续输出脉冲事件的代码示例。

例 14-4: 连续输出脉冲设置和中断服务

```
; The following code example will set the Output Compare 1 module
; for interrupts on the continuous pulse event and select Timer 2
; as the clock source for the compare time-base. It is assumed
; that Timer 2 and Period Register 2 are properly configured.
; Timer 2 will be enabled here.

CLR    OC1CON           ; Turn off Output Compare 1 Module.
MOV    #0x0005, W0      ; Load the working register with the new
MOV    W0, OC1CON       ; compare mode and write to OC1CON
MOV    #0x3000, W0      ; Initialize Compare Register 1
MOV    W0, OC1R         ; with 0x3000
MOV    #0x3003, W0      ; Initialize Secondary Compare Register 1
MOV    W0, OC1RS        ; with 0x3003
BSET   IPC0, #OC1IP0    ; Setup Output Compare 1 interrupt for
BCLR   IPC0, #OC1IP1    ; desired priority level
BCLR   IPC0, #OC1IP2    ; (this example assigns level 1 priority)
BCLR   IFS0, #OC1IF     ; Clear Output Compare 1 interrupt flag
BSET   IEC0, #OC1IE     ; Enable Output Compare 1 interrupts

BSET   T2CON, #TON      ; Start Timer2 with assumed settings

; Example code for Output Compare 1 ISR:

__OC1Interrupt:
BCLR   IFS0, #OC1IF     ; Reset respective interrupt flag
; Remaining user code here
RETFIE                      ; Return from ISR
```

14.3.2.6 双比较模式产生连续输出脉冲的特殊情况

根据 OCxR、OCxRS 和 PRy 的关系，输出比较模式可能不能提供预期的结果。表 14-2 说明了这些特殊条件和这些条件所导致的模块工作情况。

表 14-2: 双比较模式产生连续输出脉冲的特殊情况

SFR 逻辑关系	特殊条件	工作原理	OCx 引脚的输出
PRy >= OCxRS 且 OCxRS > OCxR	OCxR = 0 初始化 TMRy = 0	在 TMRy 从 0x0000 计数到 PRy 的第一次迭代中，OCx 引脚保持为低电平，不产生脉冲。在 TMRy 复位到零后（周期匹配时），OCx 引脚变为高电平。在下次 TMRy 与 OCxRS 匹配时，OCx 引脚变为低电平。如果 OCxR = 0 且 Ry = OCxRS，引脚将在一个时钟周期内保持为低电平，然后被驱动为高电平直到下次 TMRy 与 OCxRS 匹配。第二次比较匹配导致 OCxIF 位置 1。 还有两种初始条件可供参考。 a) 初始化 TMRy = 0 并设置 OCxR >= 1 b) 初始化 MRy = PRy (PRy > 0) 并设置 OCxR = 0	根据设置的情况，输出连续脉冲，而第一个脉冲将以 PRy 寄存器中的值延时。
PRy >= OCxR 且 OCxR >= OCxRS	OCxR >= 1 且 PRy >= 1	TMRy 计数到 OCxR 且比较匹配事件发生时（即 MRy = OCxR）时，OCx 引脚被驱动为高电平状态。然后 TMRy 继续计数，并最后在周期匹配时（即 PRy = TMRy）复位。然后在定时器从 0x0000 重启并计数到 OCxRS 中的值且比较匹配事件发生时（即 TMRy = OCxR），OCx 引脚被驱动为低电平状态。第二次比较匹配导致 OCxIF 位置 1。	连续脉冲
OCxRS > PRy 且 PRy >= OCxR	无	OCx 引脚上的电平转变只会发生一次，直到 OCxRS 寄存器的值变为小于或等于周期寄存器（PRy）的值。在第二次电平转变发生时，OCxIF 被置 1。	上升沿 / 转变为高电平
OCxR = OCxRS = PRy = 0x0000	无	连续输出脉冲在 OCx 引脚上产生。在定时器和周期寄存器发生匹配后延迟了两个指令时钟周期才产生第一个脉冲。第二次比较匹配导致 OCxIF 位置位	第一个脉冲延迟。产生连续脉冲。
OCxR > PRy	无	不支持此模式，定时器在匹配条件产生之前就已经复位了。	保持为低电平

注 1: 这里考虑到的所有情形，都假设 TMRy 寄存器初始化为 0x0000。
2: OCxR = 比较寄存器，OCxRS = 辅助比较寄存器，TMRy = 定时器计数器，PRy = 定时器周期寄存器。

14.3.3 脉宽调制模式

当控制位 $OCM<2:0>$ ($OCxCON<2:0>$) 设置为 “110” 或 “111” 时, 所选的输出比较通道被配置为 PWM (脉宽调制) 工作模式。

以下两种 PWM 模式可用:

- 不带故障保护输入的 PWM 模式
- 带故障保护输入的 PWM 模式

OCFA 或 OCFB 故障输入引脚用于第二种 PWM 模式。在此模式中, $OCFx$ 引脚上的异步逻辑级别 “0” 会导致所选的 PWM 通道关闭 (如第 14.3.3.1 节 “带故障保护输入引脚的 PWM” 中所述)。

在 PWM 模式中, $OCxR$ 寄存器是只读从动占空比寄存器, 而 $OCxRS$ 是可由用户写入的缓冲寄存器, 以更新 PWM 占空比。在每个定时器与周期寄存器匹配事件产生时 (PWM 周期结束时), 占空比寄存器 $OCxR$ 就被装载 $OCxRS$ 的内容。在每个 PWM 周期边界置位 $TyIF$ 中断标志位。

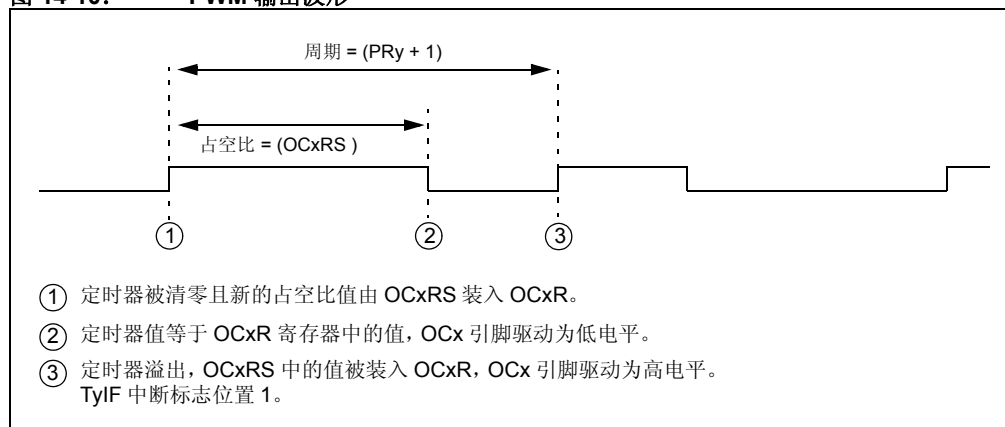
当将输出比较模块配置为 PWM 操作时, 需遵循以下步骤:

1. 通过写入所选定时器的周期寄存器 (PRy) 设置 PWM 周期。
2. 通过写入 $OCxRS$ 寄存器设置 PWM 占空比。
3. 使用初始占空比写入 $OxCR$ 寄存器。
4. 如果需要的话, 为定时器和输出比较模块允许中断。要使用 PWM 错误引脚, 需要输出比较中断。
5. 通过写输出比较模式位 $OCM<2:0>$ ($OCxCON<2:0>$) 将输出比较模块配置为两种 PWM 工作模式中的一种。
6. 设置 $TMRy$ 预分频比, 并通过设置 TON ($TxCON<15>$) = 1 使能时基。

注: $OCxR$ 寄存器应该在输出比较模块第一次使能之前被初始化。当模块在 PWM 模式工作时, $OCxR$ 寄存器成为只读占空比寄存器。 $OCxR$ 寄存器中保存的值成为第一个 PWM 周期的占空比。直到发生时基周期匹配, 才将占空比缓冲寄存器 $OCxRS$ 的内容传送到 $OCxR$ 。

图 14-10 所示为 PWM 输出波形示例。

图 14-10: PWM 输出波形



14.3.3.1 带故障保护输入引脚的 PWM

当输出比较模式位 OCM<2:0> (OCxCON<2:0>) 置为 “111” 时，所选的输出比较通道被配置为 PWM 工作模式。第 14.3.3 节 “脉宽调制模式” 描述的所有功能，以及输入故障保护功能均在此模式下适用。

故障保护通过 OCFA 和 OCFB 引脚提供。OCFA 引脚与输出比较通道 1 到 4 相关联，而 OCFB 引脚与输出比较通道 5 到 8 相关联。

如果在 OCFA/OCFB 引脚上检测到了逻辑 “0”，所选的 PWM 输出引脚将被置于高阻态。用户可以选择在 PWM 引脚上提供上拉或下拉电阻，以便在产生故障条件时提供需要的电平状态。PWM 输出立即关闭且 PWM 引脚不连接到器件时钟源。此状态将保持到：

- 外部故障条件消除，且
- 通过写入相关的模式位 OCM<2:0> (OCxCON<2:0>)，重新使能 PWM 模式。

故障条件会导致相应的中断标志 OCxIF 位置 1，且在允许中断时产生中断。当检测到故障条件时，OCFLT 位 (OCx-CON<4>) 被置为高电平（逻辑 “1”）。此位是只读位，而且只有在外部故障条件被清除，且通过写相关的模式位 OCM<2:0> (OCxCON<2:0>) 使能 PWM 模式时才能被清零。

注： 如果使能了外部故障引脚，则当器件处于休眠或空闲模式时，这些引脚将继续控制 OCx 输出引脚。

14.3.3.2 PWM 周期

PWM 周期可通过写入 PRy（定时器周期寄存器）来指定。PWM 周期可以使用以下公式计算：

公式 14-1: 计算 PWM 周期

PWM 周期 = [(PRy) + 1] • Tcy • (TMRy 预分频比)

PWM 频率 = 1/[PWM 周期]

注： 若 PRy 值为 N，会产生 N + 1 个时基计数周期的 PWM 周期。例如：将 7 写入 PRy 寄存器会产生由 8 个时基周期组成的周期。

14.3.3.3 PWM 占空比

PWM 占空比是通过写入 OCxRS 寄存器指定的。在任何时间都可以写入 OCxRS 寄存器，但是 PRy 和 TMRy 发生匹配（即周期完成）前占空比值不会被锁存到 OCxR。这可以为 PWM 占空比提供双重缓冲，这对于 PWM 的无故障操作是极其重要的。在 PWM 模式中，OCxR 是只读寄存器。

以下是 PWM 占空比的部分重要边界参数：

- 如果占空比寄存器 OCxR 装入了 0x0000，OCx 引脚将保持低电平（占空比为 0%）。
- 如果 OCxR 大于 PRy（定时器周期寄存器），则引脚将保持高电平（占空比为 100%）。
- 如果 OCxR 等于 PRy，OCx 引脚在一个时基计数周期内为低电平，而在其余所有的计数周期内均为高电平。

PWM 模式时序的详细信息参见图 14-11。表 14-3 和表 14-3 分别显示了工作在 10 和 30 MIPS 下器件的 PWM 频率和精度示例。

公式 14-2: 计算最大 PWM 精度

最大 PWM 精度 (位) =
$$\frac{\log_{10}\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log_{10}(2)} \text{ 位}$$

例 14-5: PWM 周期和占空比计算

所需 PWM 频率为 52.08 kHz,
FOSC = 10 MHz, 4 倍频 PLL（40 MHz 器件时钟速率）(Tcy = 4/FOSC)
Timer2 预分频比设置: 1:1

1/52.08 kHz

=

(PR2+1) × Tcy × (Timer2 预分频比)

19.20 ms

=

(PR2+1) × 0.1 ms × (1)

PR2

=

191

计算频率为 48 kHz、器件速率为 40 MHz 时的占空比最大精度

1/52.08 kHz

=

2^{PWM 精度} × 1/40 MHz × 1

19.20 ms

=

2^{PWM 精度} × 25 ns × 1

768

=

2^{PWM 精度}

log₁₀(768)

=

(PWM 精度) × log₁₀(2)

PWM 精度

=

9.5 bit

图 14-11: PWM 输出时序

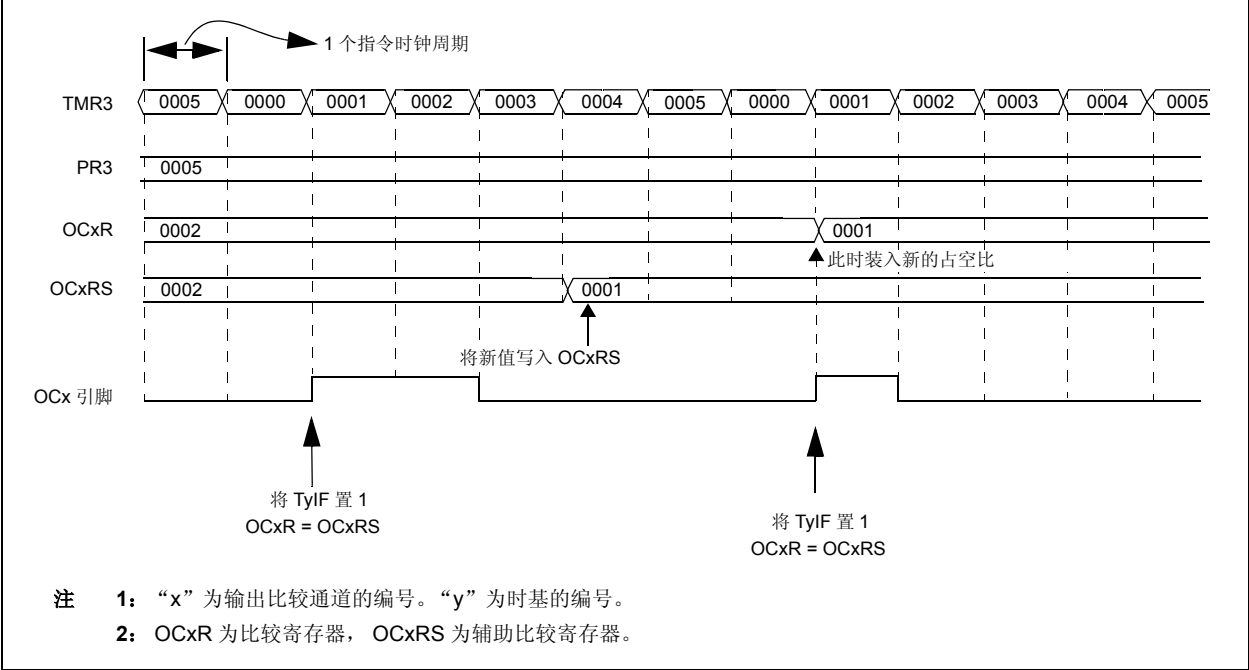


表 14-3: 10 MIP 时的 PWM 频率和精度示例 (Fosc = 40 MHz)

PWM 频率	19 Hz	153 Hz	305 Hz	2.44 kHz	9.77 kHz	78.1 kHz	313 kHz
定时器预分频比	8	1	1	1	1	1	1
周期寄存器值	0xFFFF	0xFFFF	0x7FFF	0x0FFF	0x03FF	0x007F	0x001F
精度 (位)	16	16	15	12	10	7	5

表 14-4: 30 MIP 时的 PWM 频率和精度示例 (Fosc = 120 MHz)

PWM 频率	57 Hz	458 Hz	916 Hz	7.32 kHz	29.3 kHz	234 kHz	938 kHz
定时器预分频比	8	1	1	1	1	1	1
周期寄存器值	0xFFFF	0xFFFF	0x7FFF	0x0FFF	0x03FF	0x007F	0x001F
精度 (位)	16	16	15	12	10	7	5

例 14-4 显示了 PWM 工作模式的配置和中断服务程序代码。

例 14-6: PWM 模式脉冲设置和中断服务程序

```
; The following code example will set the Output Compare 1 module
; for PWM mode w/o FAULT pin enabled, a 50% duty cycle and a
; PWM frequency of 52.08 kHz at Fosc = 40 MHz. Timer2 is selected as
; the clock for the PWM time base and Timer2 interrupts
; are enabled.

CLR      OC1CON                ; Turn off Output Compare 1 Module.

MOV      #0x0060, w0           ; Initialize Duty Cycle to 0x0060
MOV      w0, OC1RS             ; Write duty cycle buffer register
MOV      w0, OC1R              ; Write OC1R to initial duty cycle value

MOV      #0x0006, w0           ; Load the working register with the new
MOV      w0, OC1CON            ; compare mode and write to OC1CON
MOV      #0x00BF, w0           ; Initialize PR2 with 0x00BF
MOV      w0, PR2               ;

BSET     IPC0, #T2IP0          ; Setup Timer 2 interrupt for
BCLR     IPC0, #T2IP1          ; desired priority level
BCLR     IPC0, #T2IP2          ; (this example assigns level 1 priority)
BCLR     IFS0, #T21IF          ; Clear Timer 2 interrupt flag
BSET     IEC0, #T21IE          ; Enable Timer 2 interrupts
BSET     T2CON, #TON           ; Start Timer2 with assumed settings

; Example code for Timer 2 ISR:

__T2Interrupt:
BCLR     IFS0, #T21IF          ; Reset respective interrupt flag
; Remaining user code here
RETFIE                                ; Return from ISR
```

14.4 低功耗状态下的输出比较工作

14.4.1 休眠模式下的输出比较工作

当器件进入休眠模式后，系统时钟会被禁止。在休眠模式下，输出比较通道将把引脚驱动为与进入休眠模式前相同的激活状态。然后模块在这种状态下停止工作。

例如，如果引脚为高电平，那么在进入休眠状态之后引脚仍然为高电平。同样的，如果引脚为低电平，那么在进入休眠状态之后引脚仍然为低电平。在这两种情况下，当器件被唤醒时，输出比较模块都将恢复工作。

14.4.2 空闲模式下的输出比较工作

当器件进入空闲模式时，系统时钟源保持工作且 CPU 停止执行代码。OCSIDL (OCxCON<13>) 位选择在空闲模式下捕捉模块是停止还是继续工作。

- 如果 OCSIDL = 1，模块在空闲模式将停止工作。当模块在空闲模式中停止 (OCxSIDL = 1) 时，其中执行的程序与休眠模式相同。
- 如果 OCSIDL = 0，只有时基被设置在空闲模式工作时，模块才会在空闲模式下继续工作。如果 OCSIDL 位是逻辑 0，输出比较通道将在 CPU 空闲模式下工作。此外，必须将相应的 TxSIDL 位设为逻辑 0 以使能时基。

注： 如果使能了外部故障引脚，则当器件处于空闲或休眠模式时，这些引脚将继续控制相关的 OCx 输出引脚。

14.5 I/O 引脚控制

当使能了输出比较模块时，I/O 引脚方向由比较模块控制。当比较模块被禁止时，它会将 I/O 引脚控制归还给相应的引脚 LAT 和 TRIS 控制位。

当使能了具有故障保护输入模式的 PWM 时，必须通过将相应的 TRIS SFR 位置 1 以将 OCFx 故障引脚配置为输入。使能此特殊 PWM 模式并不会将 OCFx 故障引脚配置为输入。

表 14-5: 与输出比较模块 1 至 8 相关的引脚

引脚名称	引脚类型	缓冲器类型	说明
OC1	O	—	输出比较 /PWM 通道 1
OC2	O	—	输出比较 /PWM 通道 2
OC3	O	—	输出比较 /PWM 通道 3
OC4	O	—	输出比较 /PWM 通道 4
OC5	O	—	输出比较 /PWM 通道 5
OC6	O	—	输出比较 /PWM 通道 6
OC7	O	—	输出比较 /PWM 通道 7
OC8	O	—	输出比较 /PWM 通道 8
OCFA	I	ST	PWM 故障保护输入 A (通道 1 – 4)
OCFB	I	ST	PWM 故障保护输入 B (通道 5 – 8)

图注： ST = CMOS 电平施密特触发输入， I = 输入， O = 输出

表 14-6: 与输出比较模块相关的寄存器映射示例

SFR 名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位状态
TMR2	0106	Timer2 寄存器																0000 0000 0000 0000
TMR3	010A	Timer3 寄存器																0000 0000 0000 0000
PR2	010C	周期寄存器 2																1111 1111 1111 1111
PR3	010E	周期寄存器 3																1111 1111 1111 1111
T2CON	0110	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000 0000 0000 0000
T3CON	0112	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000 0000 0000 0000
OC1RS	0180	输出比较 1 辅助寄存器																uuuu uuuu uuuu uuuu
OC1R	0182	输出比较 1 寄存器																uuuu uuuu uuuu uuuu
OC1CON	0184	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC2RS	0186	输出比较 2 辅助寄存器																uuuu uuuu uuuu uuuu
OC2R	0188	输出比较 2 寄存器																uuuu uuuu uuuu uuuu
OC2CON	018A	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC3RS	018C	输出比较 3 辅助寄存器																uuuu uuuu uuuu uuuu
OC3R	018E	输出比较 3 寄存器																uuuu uuuu uuuu uuuu
OC3CON	0190	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC4RS	0192	输出比较 4 辅助寄存器																uuuu uuuu uuuu uuuu
OC4R	0194	输出比较 4 寄存器																uuuu uuuu uuuu uuuu
OC4CON	0196	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC5RS	0198	输出比较 5 辅助寄存器																uuuu uuuu uuuu uuuu
OC5R	019A	输出比较 5 寄存器																uuuu uuuu uuuu uuuu
OC5CON	019C	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC6RS	019E	输出比较 6 辅助寄存器																uuuu uuuu uuuu uuuu
OC6R	01A0	输出比较 6 寄存器																uuuu uuuu uuuu uuuu
OC6CON	01A2	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC7RS	01A4	输出比较 7 辅助寄存器																uuuu uuuu uuuu uuuu
OC7R	01A6	输出比较 7 寄存器																uuuu uuuu uuuu uuuu
OC7CON	01A8	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
OC8RS	01AA	输出比较 8 辅助寄存器																uuuu uuuu uuuu uuuu
OC8R	01AC	输出比较 8 寄存器																uuuu uuuu uuuu uuuu
OC8CON	01AE	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000 0000 0000 0000
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SP11IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0	0000 0000 0000 0000

图注: u = 未初始化

注: 寄存器映射取决于器件上输出比较模块的数量。如需更多详细信息, 参见器件数据手册。

表 14-6: 与输出比较模块相关的寄存器映射示例 (续)

SFR 名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位状态
IFS1	0086	IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IFS2	0088	—	—	—	FLTBIF	FLTAIF	LVDIF	DCIIF	QEIIIF	PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF	0000 0000 0000 0000
IEC0	008C	CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SP11IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IEC2	0090	—	—	—	FLTBIE	FLTAIE	LVDIE	DCIIE	QEIIIE	PWMIIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			0100 0100 0100 0100
IPC1	0096	—	T3IP<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>			0100 0100 0100 0100
IPC4	009C	—	OC3IP<2:0>			—	IC8IP<2:0>			—	IC7IP<2:0>			—	INT1IP<2:0>			0100 0100 0100 0100
IPC5	009E	—	INT2IP<2:0>			—	T5IP<2:0>			—	T4IP<2:0>			—	OC4IP<2:0>			0100 0100 0100 0100
IPC8	00A4	—	OC8IP<2:0>			—	OC7IP<2:0>			—	OC6IP<2:0>			—	OC5IP<2:0>			0100 0100 0100 0100

图注: u = 未初始化
注: 寄存器映射取决于器件上输出比较模块的数量。如需更多详细信息, 参见器件数据手册。

14.6 设计技巧

问 1: *即使当 OCSIDL 位没有置 1 时，输出比较引脚也是停止工作。为什么？*

答: 当相关定时器源的 TSIDL 位 (TxCON<13>) 置 1 时，最可能会发生此问题。因此，当执行 PWRSAV 指令时，实际上是定时器进入了空闲模式。

问 2: *当所选时基配置为 32 位模式时，是否能使用输出比较模块？*

答: 不能。当定时器配合输出比较模块使用时，应该将 T32 位 (TxCON<3>) 清零。

14.7 相关应用笔记

本节列出了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与输出比较模块相关的应用笔记有：

标题	应用笔记编号
目前没有相关的应用笔记。	

注：如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

14.8 版本历史

版本 A

这是本文档的初始发行版。

版本 B

本章未包含任何技术内容或编辑排版上的修订，然而我们谨更新此节以反映本手册已更新到版本 B。

版本 C

本章未包含任何技术内容修订，然而我们谨更新此节以反映本手册已更新到版本 C。

第 15 章 电机控制 PWM

目录

本章包括下列主题：

15.1 简介	15-2
15.2 控制寄存器	15-4
15.3 PWM 时基	15-16
15.4 PWM 占空比比较单元	15-20
15.5 互补 PWM 输出模式	15-26
15.6 死区时间控制	15-27
15.7 独立 PWM 输出模式	15-30
15.8 PWM 输出改写	15-31
15.9 PWM 输出和极性控制	15-34
15.10 PWM 故障引脚	15-34
15.11 PWM 更新锁定	15-37
15.12 PWM 特殊事件触发器	15-38
15.13 器件低功耗模式下的工作	15-38
15.14 用于器件仿真的特殊功能	15-39
15.15 相关应用笔记	15-42
15.16 版本历史	15-43

15.1 简介

电机控制 PWM（MCPWM）模块简化了产生多种同步脉宽调制输出的任务。特别是它还能支持以下电源和电机控制应用：

- 三相交流感应电机（AC Induction Motor, ACIM）
- 开关磁阻（Switched Reluctance, SR）电机
- 直流无刷（Brushless DC, BLDC）电机
- 不间断电源（UPS）

PWM 模块具有如下特性：

- 专用时基支持 $T_{CY}/2$ PWM 边沿精度
- 每个 PWM 发生器都有两个输出引脚
- 每个配对输出引脚均可互补或独立工作
- 用于互补模式的硬件死区时间发生器
- 可由器件配置位设置输出引脚极性
- 多种输出模式：
 - 边沿对齐模式
 - 中心对齐模式
 - 带双更新的中心对齐模式
 - 单事件模式
- 手动改写用于 PWM 输出引脚的寄存器
- 占空比更新可配置为立即更新或与 PWM 同步
- 有可编程功能的硬件故障输入引脚
- 用于同步 A/D 转换的特殊事件触发器
- 每个与 PWM 相关的输出引脚都可以被单独使能

15.1.1 不同 MCPWM 模块

根据所选的 dsPIC30F 器件的不同有两种 MCPWM 模块。一种是 8 输出模块，它通常见于 64 引脚或以上的器件上。还有一种是 6 输出的 MCPWM 模块，它通常见于引脚数小于 64 的较小器件上。给定的 dsPIC30F 器件也可能具有一个以上的 MCPWM 模块。

如需更多详细信息，参见特定器件数据手册。

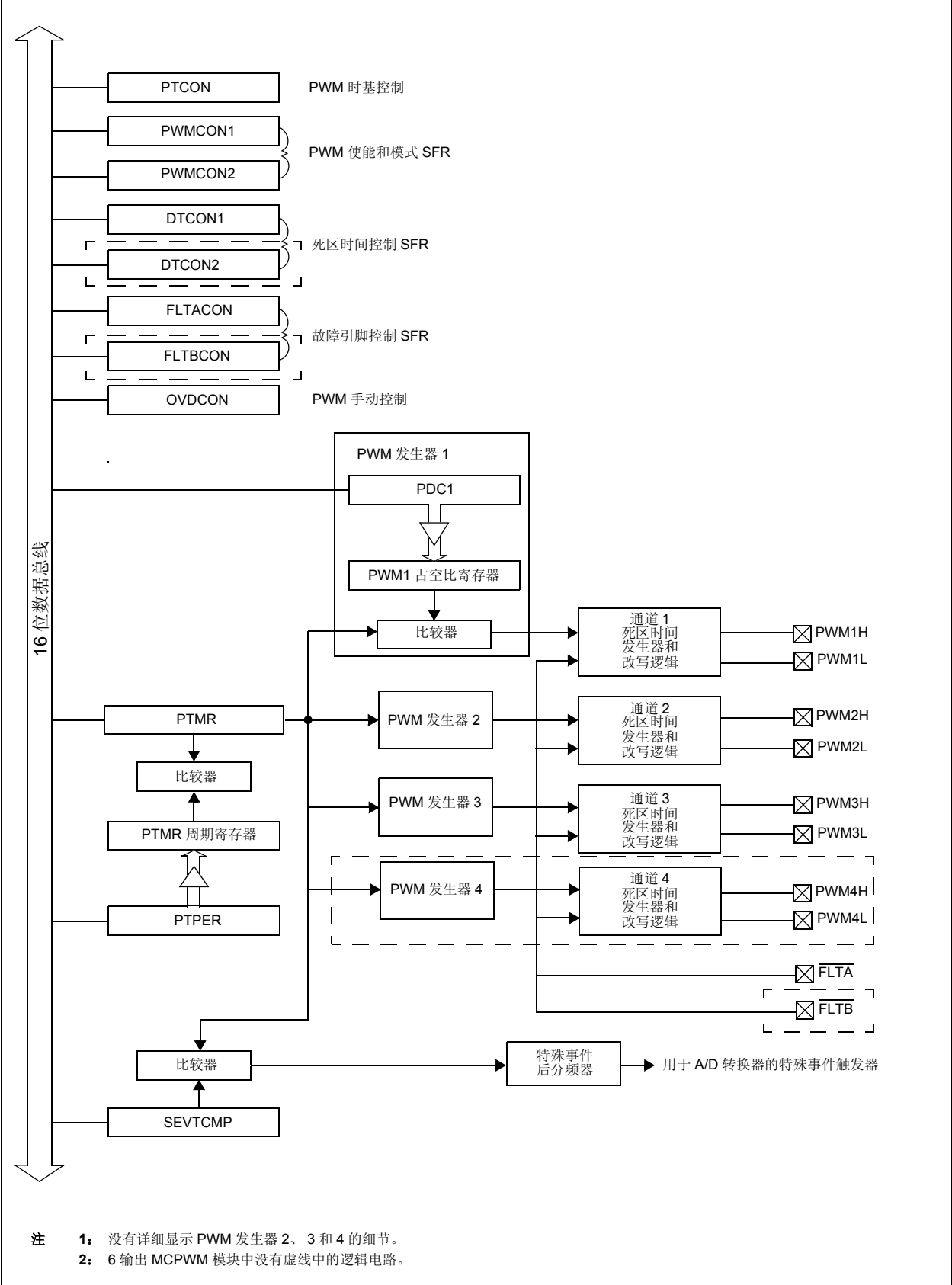
表 15-1： 功能部件总结：6 输出 MCPWM 和 8 输出 MCPWM 的对比

功能部件	6 输出 MCPWM 模块	8 输出 MCPWM 模块
I/O 引脚	6	8
PWM 发生器	3	4
故障输入引脚	1	2
死区时间发生器	1	2

6 输出的 MCPWM 模块可用于单相或 3 相电源的应用，而 8 输出的 MCPWM 能支持 4 相电机应用。表 15-1 提供了 6 输出和 8 输出的 MCPWM 模块的功能部件总结。两种模块都支持多种单相负载。8 输出 MCPWM 还为应用中提供了更高的灵活性，因为它支持两个故障引脚和两个可编程死区时间。在随后的各节中将更详细的讨论这些功能。

图 15-1 所示为 MCPWM 模块的简化框图。

图 15-1: MCPWM 框图



15.2 控制寄存器

以下寄存器控制 MCPWM 模块的工作：

- PTCN: PWM 时基控制寄存器
- PTMR: PWM 时基寄存器
- PTPER: PWM 时基周期寄存器
- SEVTCMP: PWM 特殊事件比较寄存器
- PWMCON1: PWM 控制寄存器 1
- PWMCON2: PWM 控制寄存器 2
- DTCON1: 死区时间控制寄存器 1
- DTCON2: 死区时间控制寄存器 2
- FLTACON: 故障 A 控制寄存器
- FLTBCON: 故障 B 控制寄存器
- PDC1: PWM 占空比寄存器 1
- PDC2: PWM 占空比寄存器 2
- PDC3: PWM 占空比寄存器 3
- PDC4: PWM 占空比寄存器 4

此外，还有三个与 MCPWM 模块相关的器件配置位以设置初始复位状态和 I/O 引脚的极性。这些配置位位于 FBORPOR 器件配置寄存器中。更多详情请参见第 24 章 “器件配置”。

寄存器 15-1: PTCON: PWM 时基控制寄存器

高字节:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
PTEN	—	PTSIDL	—	—	—	—	—
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>	
bit 7				bit 0			

- bit 15 **PTEN:** PWM 时基定时器使能位
1 = PWM 时基开启
0 = PWM 时基关闭
- bit 14 未用: 读作 0
- bit 13 **PTSIDL:** 空闲模式 PWM 时基停止位
1 = PWM 时基在 CPU 空闲模式停止
0 = PWM 时基在 CPU 空闲模式运行
- bit 12-8 未用: 读作 0
- bit 7-4 **PTOPS<3:0>:** PWM 时基输出后分频比选择位
1111 = 1:16 后分频

0001 = 1:2 后分频
0000 = 1:1 后分频
- bit 3-2 **PTCKPS<1:0>:** PWM 时基输入时钟预分频比选择位
11 = PWM 时基输入时钟周期为 64 Tcy (1:64 预分频)
10 = PWM 时基输入时钟周期为 16 Tcy (1:16 预分频)
01 = PWM 时基输入时钟周期为 4 Tcy (1:4 预分频)
00 = PWM 时基输入时钟周期为 1 Tcy (1:1 预分频)
- bit 1-0 **PTMOD<1:0>:** PWM 时基模式选择位
11 = PWM 时基工作在带双 PWM 更新中断的连续向上 / 向下模式
10 = PWM 时基工作在连续向上 / 向下计数模式
01 = PWM 时基工作在单事件模式
00 = PWM 时基工作在自由运行模式

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

寄存器 15-2: PTMR: PWM 时基寄存器

高字节:							
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTDIR	PTMR <14:8>						
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTMR <7:0>							
bit 7							bit 0

bit 15 **PTDIR:** PWM 时基计数方向状态位（只读）
1 = PWM 时基向下计数
0 = PWM 时基向上计数

bit 14-0 **PTMR<14:0>:** PWM 时基寄存器计数值

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

寄存器 15-3: PTPER: PWM 时基周期寄存器

高字节:							
U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	PTPER<14:8>						
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTPER<7:0>							
bit 7							bit 0

bit 15 **未用:** 读作 0

bit 14-0 **PTPER<14:0>:** PWM 时基周期值位

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

寄存器 15-4: SEVTCMP: 特殊事件比较寄存器

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SEVTDIR	SEVTCMP<14:8>						
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SEVTCMP<7:0>							
bit 7							bit 0

bit 15 **SEVTDIR:** 特殊事件触发器时基方向位 ⁽¹⁾

1 = 当 PWM 时基向下计数时触发特殊事件。

0 = 当 PWM 时基向上计数时触发特殊事件。

bit 14-0 **SEVTCMP<14:0>:** 特殊事件比较值位 ⁽²⁾

注 1: SEVTDIR 与 PTDIR (PTMR<15>) 比较以产生特殊事件触发信号。

2: SEVTCMP<14:0> 与 PTMR<14:0> 比较以产生特殊事件触发信号。

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = 上电复位时的值

1 = 置位

0 = 清零

x = 未知

寄存器 15-5: PWMCON1: PWM 控制寄存器 1

高字节:							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	PMOD4	PMOD3	PMOD2	PMOD1
bit 15							bit 8

低字节:							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PEN4H	PEN3H	PEN2H	PEN1H	PEN4L	PEN3L	PEN2L	PEN1L
bit 7							bit 0

bit 15-12 未用: 读作 0

bit 11-8 **PMOD4:PMOD1:** PWM 配对 I/O 引脚模式位

1 = PWM 配对 I/O 引脚处于独立输出模式

0 = PWM 配对 I/O 引脚处于互补输出模式

bit 7-4 **PEN4H-PEN1H:** PWMxH I/O 使能位 ⁽¹⁾

1 = PWMxH 引脚使能为 PWM 输出

0 = PWMxH 引脚禁止。I/O 引脚成为通用 I/O

bit 3-0 **PEN4L-PEN1L:** PWMxL I/O 使能位 ⁽¹⁾

1 = PWMxL 引脚使能为 PWM 输出

0 = PWMxL 引脚禁止。I/O 引脚成为通用 I/O

注 1: PENxH 和 PENxL 位的复位状态取决于在器件配置寄存器 FBORPOR 中的器件配置位 PWM/PIN 的值。

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = 上电复位时的值

1 = 置位

0 = 清零

x = 未知

寄存器 15-6: PWMCON2: PWM 控制寄存器 2

高字节:							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	SEVOPS<3:0>			
bit 15				bit 8			

低字节:							
U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	IUE	OSYNC	UDIS
bit 7				bit 0			

bit 15-12 未用: 读作 0

bit 11-8 **SEVOPS<3:0>**: PWM 特殊事件触发器输出后分频比选择位

1111 = 1:16 后分频

0001 = 1:2 后分频

0000 = 1:1 后分频

bit 7-3 未用: 读作 0

bit 2 **IUE**: 立即更新使能位 ⁽¹⁾

1 = 立即更新有效的 PDC 寄存器

0 = 对有效的 PDC 寄存器的更新与 PWM 时基同步

bit 1 **OSYNC**: 输出改写同步位

1 = 通过设置 OVDCON 寄存器, 使得输出改写与 PWM 时基同步

0 = 通过设置 OVDCON 寄存器, 使得输出改写在下一个 Tcy 边沿发生

bit 0 **UDIS**: PWM 更新禁止位

1 = 禁止从占空比和周期缓冲寄存器更新

0 = 使能从占空比和周期缓冲寄存器更新

注 1: IUE 位在 dsPIC30F6010 器件中未用。

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = 上电复位时的值

1 = 置位

0 = 清零

x = 未知

寄存器 15-7: DTCON1: 死区时间控制寄存器 1

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTBPS<1:0>		DTB<5:0>					
bit 15		bit 8					

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTAPS<1:0>		DTA<5:0>					
bit 7		bit 0					

- bit 15-14 **DTBPS<1:0>**: 死区时间单元 B 预分频比选择位
- 11 = 死区时间单元 B 的时钟周期为 8 T_{CY}
 - 10 = 死区时间单元 B 的时钟周期为 4 T_{CY}
 - 01 = 死区时间单元 B 的时钟周期为 2 T_{CY}
 - 00 = 死区时间单元 B 的时钟周期为 T_{CY}
- bit 13-8 **DTB<5:0>**: 死区时间单元 B 的无符号 6 位死区时间值位
- bit 7-6 **DTAPS<1:0>**: 死区时间单元 A 预分频比选择位
- 11 = 死区时间单元 A 的时钟周期为 8 T_{CY}
 - 10 = 死区时间单元 A 的时钟周期为 4 T_{CY}
 - 01 = 死区时间单元 A 的时钟周期为 2 T_{CY}
 - 00 = 死区时间单元 A 的时钟周期为 T_{CY}
- bit 5-0 **DTA<5:0>**: 死区时间单元 A 的无符号 6 位死区时间值位

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 15-8: DTCON2: 死区时间控制寄存器 2

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							
bit 8							

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTS4A	DTS4I	DTS3A	DTS3I	DTS2A	DTS2I	DTS1A	DTS1I
bit 7							
bit 0							

bit 15-8 未用: 读作 0

bit 7 **DTS4A:** PWM4 信号变为有效的死区时间选择位

1 = 由单元 B 提供死区时间

0 = 由单元 A 提供死区时间

bit 6 **DTS4I:** PWM4 信号变为无效的死区时间选择位

1 = 由单元 B 提供死区时间

0 = 由单元 A 提供死区时间

bit 5 **DTS3A:** PWM3 信号变为有效的死区时间选择位

1 = 由单元 B 提供死区时间

0 = 由单元 A 提供死区时间

bit 4 **DTS3I:** PWM3 信号变为无效的死区时间选择位

1 = 由单元 B 提供死区时间

0 = 由单元 A 提供死区时间

bit 3 **DTS2A:** PWM2 信号变为有效的死区时间选择位

1 = 由单元 B 提供死区时间

0 = 由单元 A 提供死区时间

bit 2 **DTS2I:** PWM2 信号变为无效的死区时间选择位

1 = 由单元 B 提供死区时间

0 = 由单元 A 提供死区时间

bit 1 **DTS1A:** PWM1 信号变为有效的死区时间选择位

1 = 由单元 B 提供死区时间

0 = 由单元 A 提供死区时间

bit 0 **DTS1I:** PWM1 信号变为无效的死区时间选择位

1 = 由单元 B 提供死区时间

0 = 由单元 A 提供死区时间

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = 上电复位时的值

1 = 置位

0 = 清零

x = 未知

寄存器 15-9: **FLTACON: 故障 A 控制寄存器**

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
FAOV4H	FAOV4L	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L
bit 15				bit 8			

低字节:							
R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
FLTAM	—	—	—	FAEN4	FAEN3	FAEN2	FAEN1
bit 7				bit 0			

- bit 15-8 **FAOV4H-FAOV1L:** 故障输入 A PWM 改写值位
1 = PWM 输出引脚在发生外部故障输入事件时驱动为有效
0 = PWM 输出引脚在发生外部故障输入事件时驱动为无效
- bit 7 **FLTAM:** 故障 A 模式位
1 = 在逐个周期模式中, 故障 A 输入引脚起作用
0 = 故障 A 输入引脚将所有控制引脚锁存在 FLTACON<15:8> 中编程的状态。
- bit 6-4 **未用:** 读作 0
- bit 3 **FAEN4:** 故障输入 A 使能位
1 = PWM4H/PWM4L 这对引脚由故障输入 A 控制
0 = PWM4H/PWM4L 这对引脚不由故障输入 A 控制
- bit 2 **FAEN3:** 故障输入 A 使能位
1 = PWM3H/PWM3L 这对引脚由故障输入 A 控制
0 = PWM3H/PWM3L 这对引脚不由故障输入 A 控制
- bit 1 **FAEN2:** 故障输入 A 使能位
1 = PWM2H/PWM2L 这对引脚由故障输入 A 控制
0 = PWM2H/PWM2L 这对引脚不由故障输入 A 控制
- bit 0 **FAEN1:** 故障输入 A 使能位
1 = PWM1H/PWM1L 这对引脚由故障输入 A 控制
0 = PWM1H/PWM1L 这对引脚不由故障输入 A 控制

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 15-10: FLTBCON: 故障 B 控制寄存器

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
FBOV4H	FBOV4L	FBOV3H	FBOV3L	FBOV2H	FBOV2L	FBOV1H	FBOV1L
bit 15							
bit 8							

低字节:							
R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
FLTBM	—	—	—	FBEN4	FBEN3	FBEN2	FBEN1
bit 7							
bit 0							

- bit 15-8 **FBOV4H:FBOV1L**: 故障输入 B PWM 改写值位
1 = PWM 输出引脚在发生外部故障输入事件时驱动为有效
0 = PWM 输出引脚在发生外部故障输入事件时驱动为无效
- bit 7 **FLTBM**: 故障 B 模式位
1 = 在逐个周期模式中, 故障 B 输入引脚起作用
0 = 故障 B 输入引脚将所有控制引脚锁存为在 FLTBCON<15:8> 中编程的状态。
- bit 6-4 未用: 读作 0
- bit 3 **FAEN4**: 故障输入 B 使能位 ⁽¹⁾
1 = PWM4H/PWM4L 这对引脚由故障输入 B 控制
0 = PWM4H/PWM4L 这对引脚不由故障输入 B 控制
- bit 2 **FAEN3**: 故障输入 B 使能位 ⁽¹⁾
1 = PWM3H/PWM3L 这对引脚由故障输入 B 控制
0 = PWM3H/PWM3L 这对引脚不由故障输入 B 控制
- bit 1 **FAEN2**: 故障输入 B 使能位 ⁽¹⁾
1 = PWM2H/PWM2L 这对引脚由故障输入 B 控制
0 = PWM2H/PWM2L 这对引脚不由故障输入 B 控制
- bit 0 **FAEN1**: 故障输入 B 使能位 ⁽¹⁾
1 = PWM1H/PWM1L 这对引脚由故障输入 B 控制
0 = PWM1H/PWM1L 这对引脚不由故障输入 B 控制
- 注 1: 如果两者同时使能的话, 故障引脚 A 的优先级高于故障引脚 B。

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = 上电复位时的值

1 = 置位

0 = 清零

x = 未知

寄存器 15-11: OVDCON: 改写控制寄存器

高字节:							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
POVD4H	POVD4L	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
POUT4H	POUT4L	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L
bit 7							bit 0

- bit 15-8 **POVD4H-POVD1L**: PWM 输出改写位
1 = PWMxx I/O 引脚上的输出由 PWM 发生器控制
0 = PWMxx I/O 引脚上的输出由相应的 POUTxx 位中的值控制
- bit 7-0 **POUT4H-POUT1L**: PWM 手动输出位
1 = 在相应的 POVDxx 位被清零时 PWMxx I/O 引脚驱动为有效
0 = 在相应的 POVDxx 位被清零时 PWMxx I/O 引脚驱动为无效

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

寄存器 15-12: PDC1: PWM 占空比寄存器 1

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM 占空比 1 的 15-8 位							
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM 占空比 1 的 7-0 位							
bit 7							bit 0

- bit 15-0 **PDC1<15:0>**: PWM 占空比 1 值位

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 15-13: PDC2: PWM 占空比寄存器 2

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM 占空比 2 的 15-8 位							
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM 占空比 2 的 7-0 位							
bit 7				bit 0			

bit 15-0 **PDC2<15:0>**: PWM 占空比寄存器 2 值位

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

寄存器 15-14: PDC3: PWM 占空比寄存器 3

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM 占空比 3 的 15-8 位							
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM 占空比 3 的 7-0 位							
bit 7				bit 0			

bit 15-0 **PDC3<15:0>**: PWM 占空比寄存器 3 值位

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

寄存器 15-15: PDC4: PWM 占空比寄存器 4

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM 占空比 4 的 15-8 位							
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWM 占空比 4 的 7-0 位							
bit 7				bit 0			

bit 15-0 PDC1<15:0>: PWM 占空比寄存器 4 值位

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

寄存器 15-16: FBORPOR: BOR 和 POR 器件配置寄存器

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23				bit 16			

中字节:							
U-0	U-0	U-0	U-0	U-0	R/P	R/P	R/P
—	—	—	—	—	PWMPIN	HPOL	LPOL
bit 15				bit 8			

低字节:							
R/P	U-0	R/P	R/P	U-0	U-0	R/P	R/P
BOREN	—	BORV<1:0>		—	—	FPWRT<1:0>	
bit 7				bit 0			

- bit 10 PWMPIN: MPWM 驱动器初始化位
1 = I/O 端口控制复位时的引脚状态 (PWMCON1<7:0> = 0x00)
0 = PWM 模块控制复位时的引脚状态 (PWMCON1<7:0> = 0xFF)
- bit 9 HPOL: MCPWM 高边驱动器 (PWMxH) 极性位
1 = PWMxH 引脚上的输出信号极性为高电平有效
0 = PWMxH 引脚上的输出信号极性为低电平有效
- bit 8 LPOL: MCPWM 低边驱动器 (PWMxL) 极性位
1 = PWMxL 引脚上的输出信号极性为高电平有效
0 = PWMxL 引脚上的输出信号极性为低电平有效

注: 有关寄存器上其他配置位的信息, 请参见第 24 章 “器件配置”。

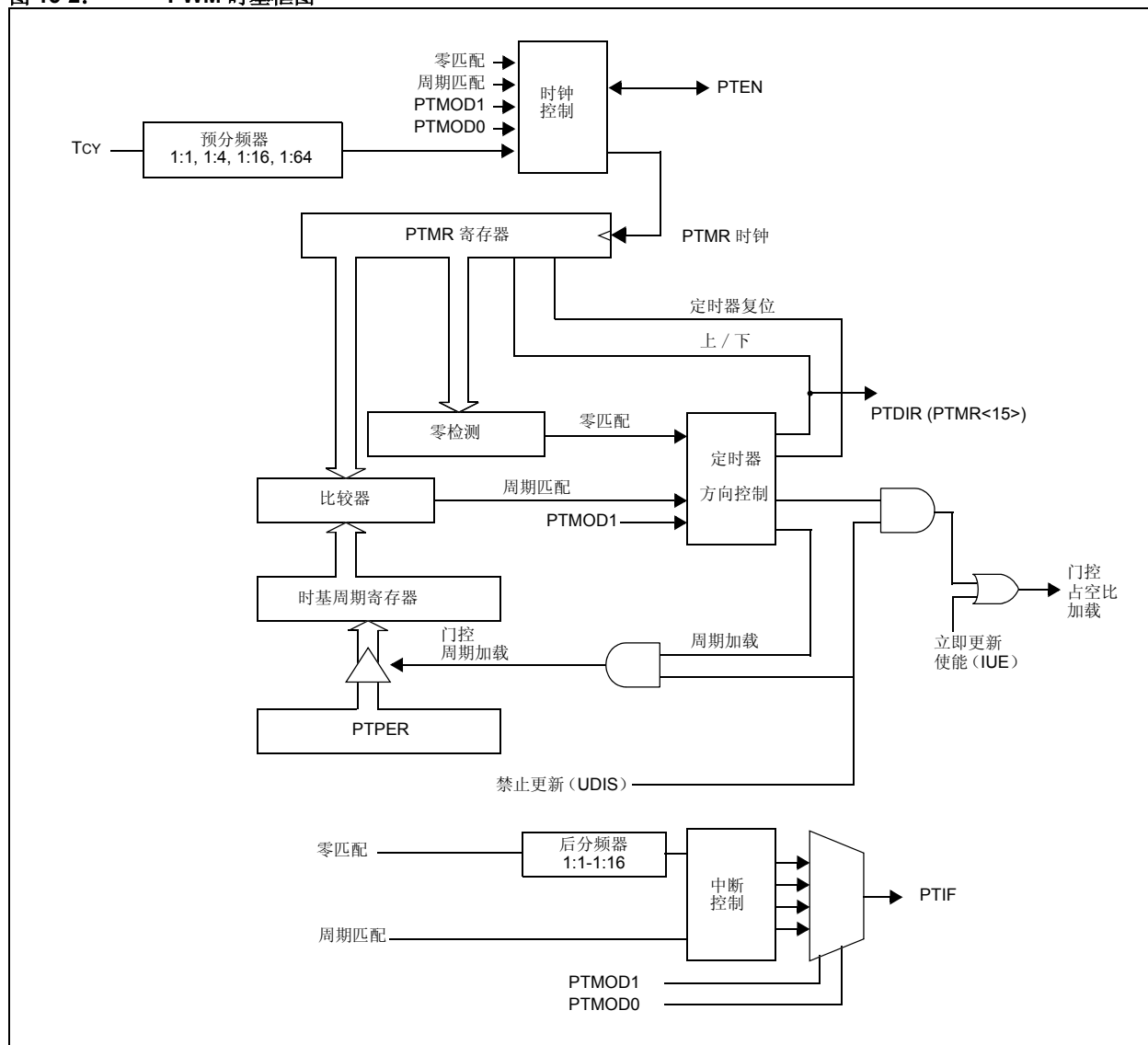
图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知
P = 可编程配置位			

15.3 PWM 时基

PWM 时基由一个带有预分频器和后分频器的 15 位定时器提供（见图 15-2）。时基的 15 位可通过 PTMR 寄存器访问。PTMR<15> 为一个只读状态位 PTDIR，显示 PWM 时基当前的计数方向。如果 PTDIR 状态位清零，则表示 PTMR 正在向上计数。如果 PTDIR 置 1，则表示 PTMR 正在向下计数。

通过置位 / 清零 PTEN 位（PTCON<15>）来使能 / 禁止时基。当 PTEN 位由软件清零时，PTMR 位不会清零。

图 15-2: PWM 时基框图



可以将 PWM 时基配置为以下四种不同的工作模式：

1. 自由运行模式
2. 单事件模式
3. 连续向上 / 向下计数模式
4. 带双更新中断的连续向上 / 向下计数模式。

这四个模式通过 PTMOD<1:0> 控制位 (PTCON<1:0>) 选择。

注： PWM 时基模式决定模块产生的 PWM 信号的类型 (更多细节, 请参见第 15.4.2 节、第 15.4.3 节和第 15.4.4 节)。

15.3.1 自由运行模式

在自由运行模式中, 时基将向上计数直到与 PTPER 寄存器中的值发生匹配。PTMR 寄存器在接下来的输入时钟边沿复位, 且只要 PTEN 位保持置 1, 时基仍将继续向上计数。

15.3.2 单事件模式

在单事件计数模式中, PWM 时基在 PTEN 位置位时将开始向上计数。当 PTMR 值与 PTPER 寄存器匹配时, PTMR 寄存器将在接下来的输入时钟边沿复位, 且由硬件清零 PTEN 位以停止时基。

15.3.3 向上 / 向下计数模式

在连续向上 / 向下计数模式中, PWM 时基将向上计数直到与 PTPER 寄存器中的值发生匹配。定时器将在接下来的输入时钟边沿开始向下计数, 并继续向下计数直到达到 0。PTDIR 位 PTMR<15> 是只读位, 显示计数方向。当定时器向下计数时 PTDIR 位将置 1。

15.3.4 PWM 时基预分频器

PTMR 的输入时钟 (Tcy) 的预分频选项有 1:1、1:4、1:16 或 1:64, 通过控制位 PTCKPS<1:0> (PTCON<3:2>) 选择。当发生以下情况中的任何一种时, 预分频器计数器清零:

- 对 PTMR 寄存器写
- 对 PTCN 寄存器写
- 任何器件复位

当写入 PTCN 时, PTMR 寄存器不会清零。

15.3.5 PWM 时基后分频器

PTMR 的匹配输出可以选择通过一个 4 位后分频器 (可进行 1:1 到 1:16 的分频, 包括 1:1 和 1:16) 选择性地进行后分频并产生中断。当 PWM 占空比不需要在每个 PWM 周期被更新时, 后分频器非常有用。

当发生以下情况中的任何一种时, 后分频器计数器将清零:

- 对 PTMR 寄存器写
- 对 PTCN 寄存器写
- 任何器件复位

当写入 PTCN 时, PTMR 寄存器不会清零。

15.3.6 PWM 时基中断

PWM 时基根据模式选择位 PTMOD<1:0> (PTCON<1:0>) 和时基后分频器位 PTOPS<3:0> (PTCON<7:4>) 产生中断信号。

- 自由运行模式

当 PWM 时基处于自由运行模式 (PTMOD<1:0> = 00) 时, 在 PTMR 寄存器与 PTPER 寄存器匹配而复位到 “0” 时, 产生中断。在此定时器模式中使用后分频比选择位以减小中断事件发生的频率。

- 单事件模式

当 PWM 时基处于单事件模式时 (PTMOD<1:0> = 01) 时, 在 PTMR 寄存器与 PTPER 寄存器匹配而复位到 “0” 时, 产生中断。此时 PTEN 位 (PTCON<15>) 也被清零以禁止 PTMR 继续加计数。后分频比选择位对此定时器模式没有影响。

- 向上 / 向下计数模式

在向上 / 下计数模式 (PTMOD<1:0> = 10) 中, 每当 PTMR 寄存器的值变为零时都会发生中断事件, 这时 PWM 时基开始向上计数。在此定时器模式中使用后分频比选择位以减小中断事件发生的频率。

- 带双更新的向上 / 向下计数模式

在双更新模式 (PTMOD<1:0> = 11) 中, 每次 PTMR 寄存器等于零和每当发生周期匹配时都产生中断。后分频器选择位对此定时器模式没有影响。

由于 PWM 占空比在每个周期可更新两次, 所以双更新模式可使控制循环带宽加倍。PWM 信号的每个上升沿和下降沿都可以用双更新模式控制。

15.3.7 PWM 周期

PTPER 寄存器为 PTMR 设置计数周期。用户必须将 15 位值写入 PTPER<14:0>。当 PTMR<14:0> 的值与 PTPER<14:0> 中的值匹配时, 时基将复位为 0, 或在下一个时钟输入边沿改变计数方向。具体执行哪一种行为取决于时基的工作模式。

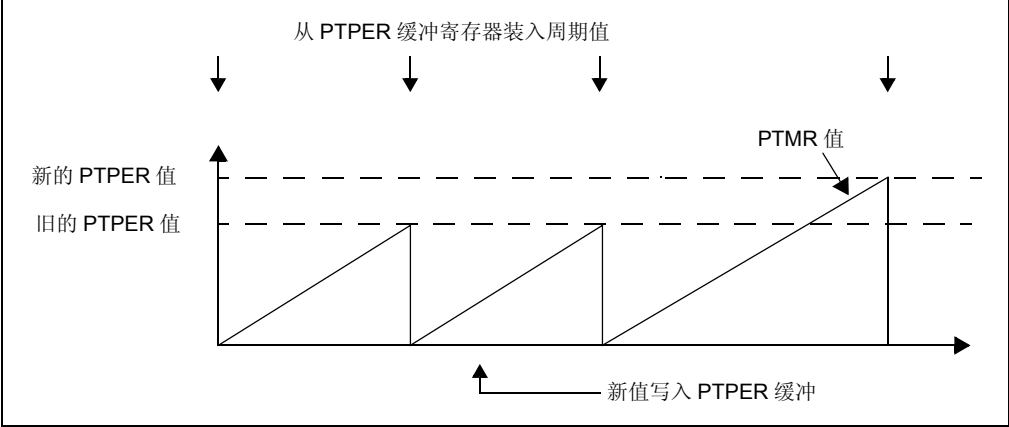
时基周期被双缓冲以使 PWM 信号可随时更改周期而不产生毛刺。PTPER 寄存器作为实际时基周期寄存器的缓冲寄存器, 用户不能对它进行访问。PTPER 寄存器的内容在以下时候装载到实际时基周期寄存器中:

- 自由运行和单事件模式: 当 PTMR 寄存器在与 PTPER 寄存器发生匹配后复位为零时。
- 向上 / 向下计数模式: 当 PTMR 寄存器为零时。

当 PWM 时基被禁止 (PTEN = 0) 时, PTPER 寄存器中保存的值被自动装入时基周期寄存器。

图 15-3 和图 15-4 指出了将 PTPER 寄存器中的值装入时基周期寄存器的时间。

图 15-3: 自由运行计数模式下 PWM 周期缓冲器的更新



PWM 周期可通过下式确定:

公式 15-1: 自由运行计数模式下 PWM 周期的计算 (PTMOD = 10 或 11)

例:

$$PTPER = \frac{FCY}{F_{PWM} \cdot (PTMR \text{ 预分频比})} - 1$$

$$FCY = 20 \text{ MHz}$$

$$F_{PWM} = 20,000 \text{ Hz}$$

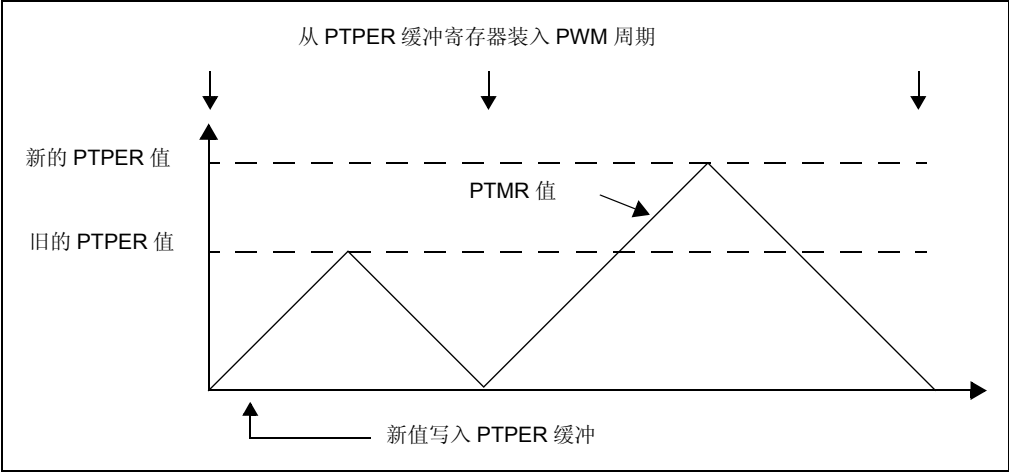
$$PTMR \text{ 预分频比} = 1:1$$

$$PTPER = \frac{20,000,000}{20,000 \cdot 1} - 1$$

$$= 1000 - 1$$

$$= 999$$

图 15-4: 向上 / 向下计数模式下 PWM 周期缓冲器的更新



公式 15-2: 向上 / 向下计数模式下 PWM 周期的计算 (PTMOD = 00 或 01)

$$PTPER = \frac{F_{CY}}{F_{PWM} \cdot (PTMR \text{ 预分频比})} - 1$$

例:

$$F_{CY} = 20 \text{ MHz}$$

$$F_{PWM} = 20,000 \text{ Hz}$$

$$PTMR \text{ 预分频比} = 1:1$$

$$PTPER = \frac{20,000,000}{20,000 \cdot 1} - 1$$

$$= 500 - 1$$

$$= 499$$

15.4 PWM 占空比较单元

MCPWM 模块有四个 PWM 发生器。有四个 16 位特殊功能寄存器用于为 PWM 发生器指定占空比值，如下：

- PDC1
- PDC2
- PDC3
- PDC4

在后面的讨论中，PDCx 指的是四个 PWM 占空比寄存器中的任何一个。

15.4.1 PWM 占空比精度

给定器件振荡器的最大精度（以位为单位）和 PWM 频率可以用以下公式确定：

公式 15-3: PWM 精度

$$Resolution = \frac{\log\left(\frac{2T_{PWM}}{T_{CY}}\right)}{\log(2)}$$

表 15-2 所示为选择不同执行速度和 PTPER 值时的 PWM 精度和频率。表 15-2 中的 PWM 频率用于边沿对齐（自由运行 PTMR）的 PWM 模式。而对于中心对齐模式（向上 / 向下 PTMR 模式），PWM 频率为表 15-3 中值的 1/2。

表 15-2: PWM 频率和分辨率示例，预分频比 1:1，边沿对齐的 PWM

T _{CY} (F _{CY})	PTPER 值	100% 的 PDCx 值	PWM 分辨率	PWM 频率
33 ns (30 MHz)	0x7FFF	0xFFFF	16 位	915 Hz
33 ns (30 MHz)	0x3FF	0x7FF	11 位	29.3 kHz
50 ns (20 MHz)	0x7FFF	0xFFFF	16 位	610 Hz
50 ns (20 MHz)	0x1FF	0x3FF	10 位	39.1 kHz
100 ns (10 MHz)	0x7FFF	0xFFFF	16 位	305 Hz
100 ns (10 MHz)	0xFF	0x1FF	9 位	39.1 kHz
200 ns (5 MHz)	0x7FFF	0xFFFF	16 位	153 Hz
200 ns (5 MHz)	0x7F	0xFF	8 位	39.1 kHz

表 15-3: PWM 频率和分辨率示例，预分频比 1:1，中心对齐的 PWM

TcY (FcY)	PTPER 值	100% 时的 PDCx 值	PWM 分辨率	PWM 频率
33 ns (30 MHz)	0x7FFF	0xFFFF	16 位	458 Hz
33 ns (30 MHz)	0x3FFF	0x7FFF	15 位	916 Hz
50 ns (20 MHz)	0x7FFF	0xFFFF	16 位	305 Hz
50 ns (20 MHz)	0x1FFF	0x3FFF	14 位	1.22 kHz
100 ns (10 MHz)	0x7FFF	0xFFFF	16 位	153 Hz
100 ns (10 MHz)	0xFFF	0x1FFF	13 位	1.22 kHz
200 ns (5 MHz)	0x7FFF	0xFFFF	16 位	76.3 Hz
200 ns (5 MHz)	0x7FF	0xFFF	12 位	1.22 kHz

MCPWM 模块能够产生精度为 $T_{cY}/2$ 的 PWM 信号沿。预分频比为 1:1 时，PTMR 在每个 T_{cY} 进行加计数。为了达到 $T_{cY}/2$ 边沿精度， $PDCx<15:1>$ 与 $PTMR<14:0>$ 进行了比较以判断占空比是否匹配。 $PDCx<0>$ 确定 PWM 信号边沿在 T_{cY} 边界发生还是在 $T_{cY}/2$ 边界发生。当 PWM 时基预分频比为 1:4、1:16 或 1:64 时， $PDCx<0>$ 与预分频器计数器时钟的 MSb 进行比较确定发生 PWM 边沿的时间。

PTMR 和 PDCx 分辨率如图 15-5 所示。图中 PTMR 分辨率为 T_{cY} 而 PDCx 分辨率为 $T_{cY}/2$ ，选择的预分频比为 1:1。

图 15-5: PTMR 和 PDCx 分辨率时序图。自由运行模式，预分频比 1:1

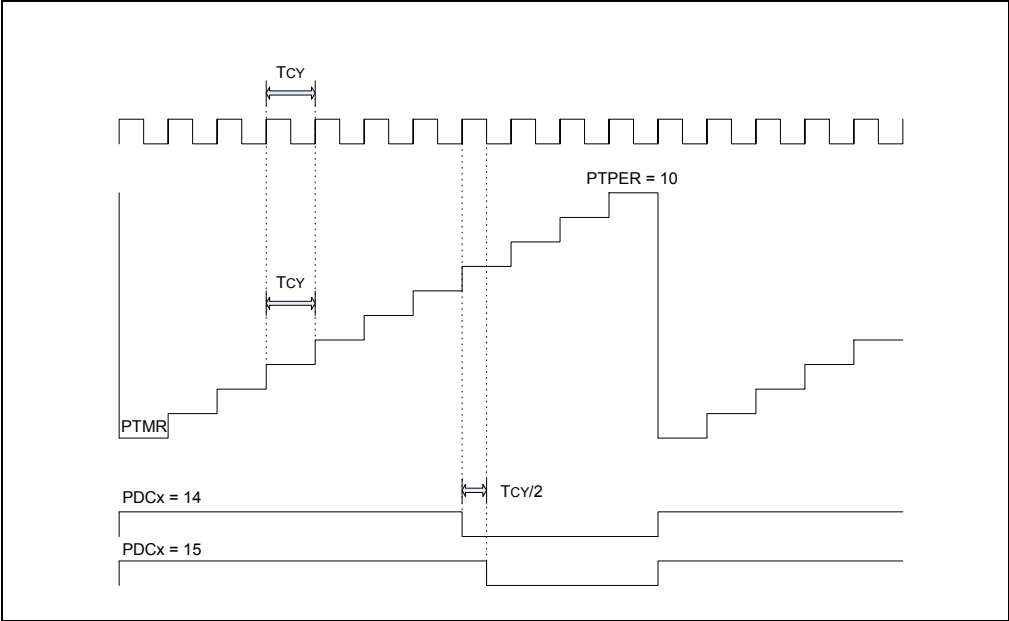
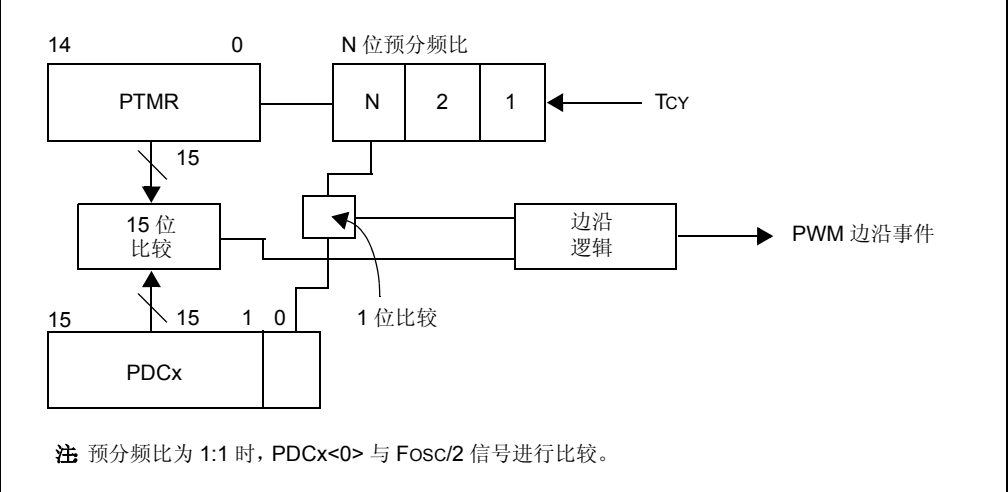


图 15-6: 占空比较逻辑



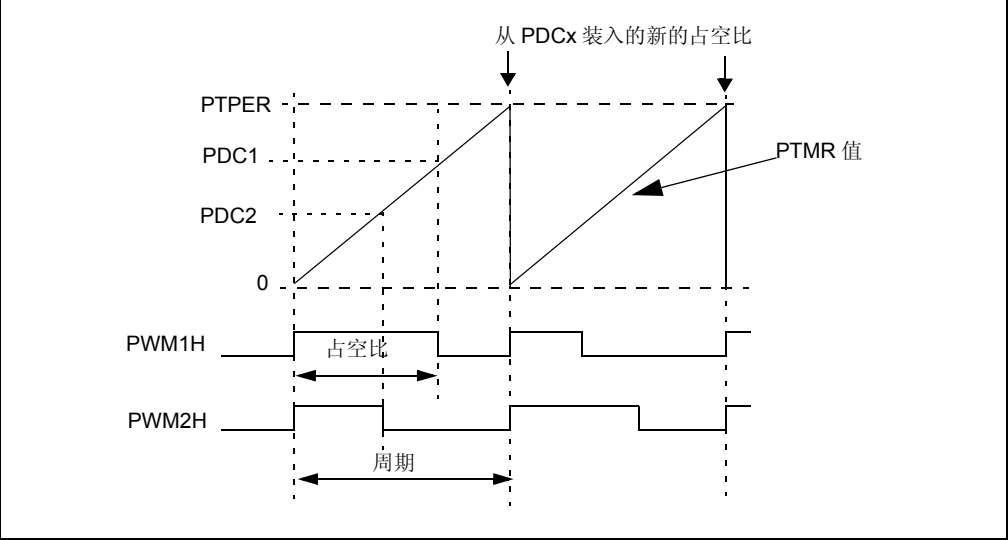
15.4.2 边沿对齐的 PWM

当 PWM 时基工作在自由运行模式时，模块产生边沿对齐的 PWM 信号。给定 PWM 通道的输出信号的周期由装入 PTPER 的值指定，其占空比由相应的 PDCx 寄存器指定（参见图 15-7）。假设占空比非零且立即更新未被使能（IUE = 0），所有使能的 PWM 发生器的输出在 PWM 周期开始（PTMR = 0）时被驱动为有效。当 PTMR 的值与 PWM 发生器的占空比值发生匹配时，各 PWM 输出都被驱动为无效。

如果 PDCx 寄存器中的值为 0，则相应的 PWM 引脚的输出在整个 PWM 周期内都将为无效。此外，如果 PDCx 寄存器中的值大于 PTPER 寄存器中保存的值，那么 PWM 引脚的输出在整个 PWM 周期内都将有效。

如果使能了立即更新（IUE = 1），则在新值写入任一有效的 PDC 寄存器时，新的占空比值即被装入。

图 15-7: 边沿对齐的 PWM

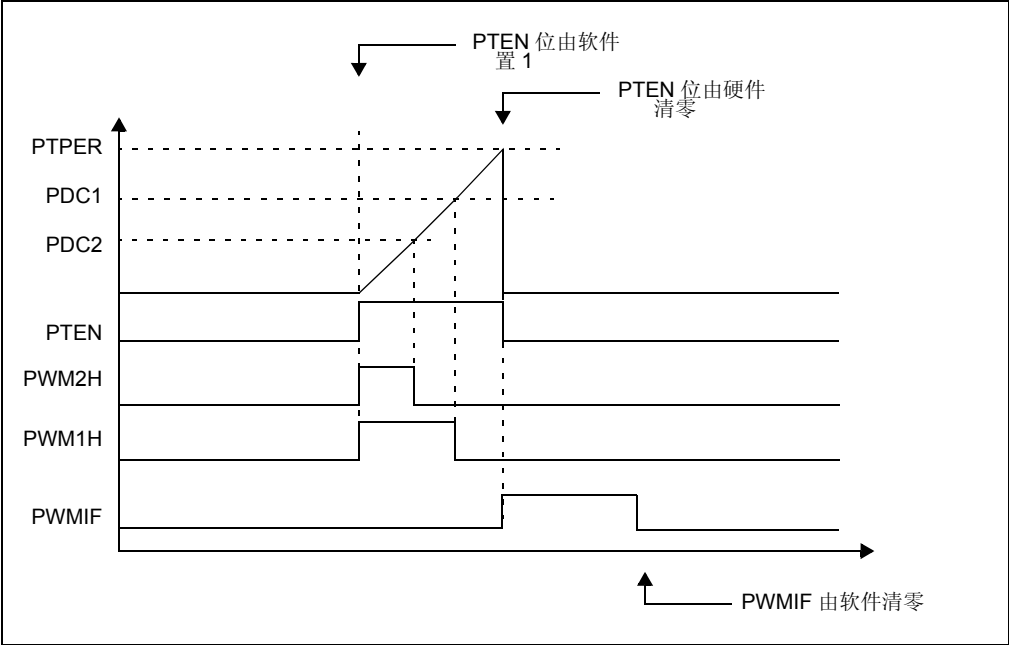


15.4.3 单事件 PWM 工作

当 PWM 时基配置为单事件模式 (PTMOD<1:0> = 01) 时, PWM 模块将产生单脉冲输出。此工作模式对于驱动某些类型的电子换相电机很有用。该模式尤其适用于高速 SR 电机的运行。在单事件模式下, 只能产生边沿对齐的输出。

在单事件模式中, 当 PTEN 位置 1 时, PWM I/O 引脚被驱动为有效状态。当 PTMR 的值与占空比寄存器的值匹配时, PWM I/O 引脚被驱动为无效状态。当与 PTPER 寄存器的值匹配时, PTMR 寄存器被清零, 所有的有效 PWM I/O 引脚都将驱动为无效状态, PTEN 位清零, 并且会产生一个中断。PWM 模块将停止工作, 直到 PTEN 在软件中被重新置 1。

图 15-8: 单事件 PWM 工作



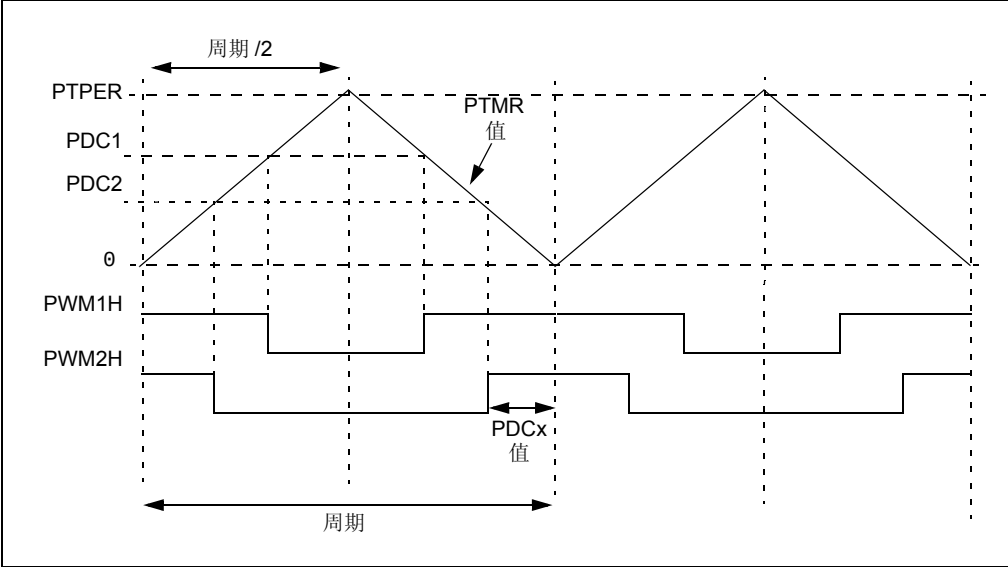
15.4.4 中心对齐的 PWM

当 PWM 时基配置为两个向上 / 向下计数模式 ($PTMOD<1:0> = 1x$) 之一时, 模块将产生中心对齐的 PWM 信号。

当占空比寄存器的值与 PTMR 的值相匹配, 并且 PWM 时基正在向下计数 ($PTDIR = 1$) 时, PWM 比较输出驱动为有效状态。当 PWM 时基正在向上计数 ($PTDIR = 0$), 且 PTMR 寄存器中的值与占空比值匹配时, PWM 比较输出将驱动为无效状态。

如果特定占空比寄存器中的值为 0, 则相应 PWM 引脚的输出在整个 PWM 周期中都将为无效。此外, 如果占空比寄存器中的值大于 PTPER 寄存器中保存的值, 则 PWM 引脚的输出在整个 PWM 周期内都将有效。

图 15-9: 中心对齐的 PWM



15.4.5 占空比寄存器缓冲

四个 PWM 占空比寄存器 PDC1 — PDC4 都采用了缓冲计数以使 PWM 输出更新时无毛刺。对于每个发生器, 都有可由用户访问的 PDCx 寄存器 (缓冲寄存器) 和保存实际比较值的非存储器映射的占空比寄存器。PWM 占空比是在 PWM 周期的特定时间使用 PDCx 寄存器中的值更新的, 以避免 PWM 输出信号产生毛刺。

当 PWM 时基工作在自由运行或单事件模式 ($PTMOD<1:0> = 0x$) 时, 只要 PTMR 与 PTPER 寄存器发生了匹配, PWM 占空比就会更新, 同时 PTMR 复位为 0。

注: 当 PWM 时基被禁止 ($PTEN = 0$) 时, 任何对 PDCx 寄存器的写入都会立即更新占空比。这可以使占空比的改变在 PWM 信号发生被使能前生效。

当 PWM 时基工作在向上 / 向下计数模式 ($PTMOD<1:0> = 10$) 时, 当 PTMR 寄存器的值为 0 且 PWM 时基开始向上计数时, 更新占空比。图 15-10 显示了此 PWM 时基模式下占空比更新发生的时间。

当 PWM 时基处于带有双重更新的向上 / 向下计数模式 ($PTMOD<1:0> = 11$) 时, 当 PTMR 寄存器的值为 0 以及 PTMR 寄存器的值与 PTPER 寄存器中的值匹配时, 都会更新占空比。图 15-11 显示了此 PWM 时基模式的占空比更新发生的时间。

图 15-10: 向上 / 向下计数模式中的占空比更新时间

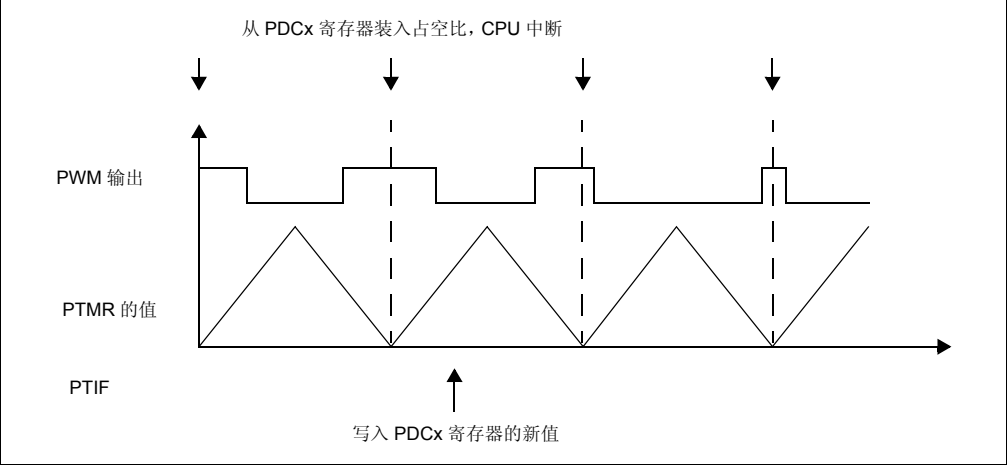
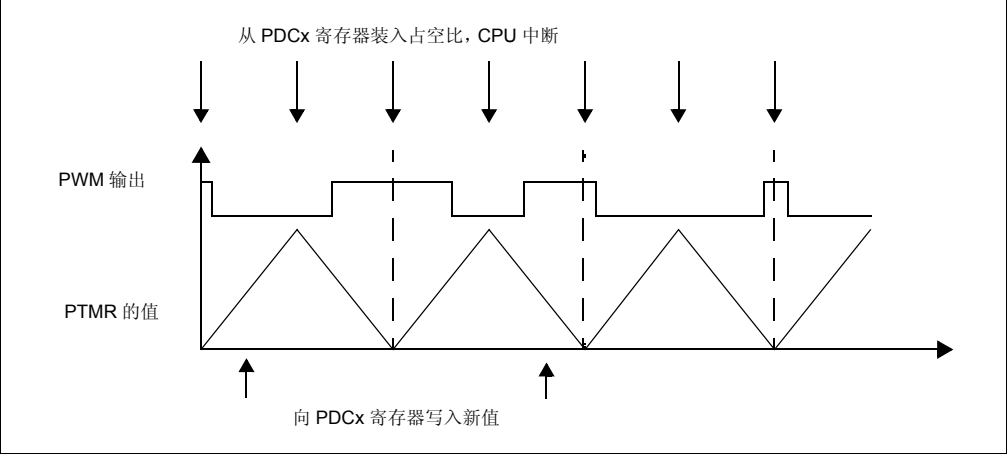


图 15-11: 双重更新向上 / 向下计数模式中的占空比更新时间



15.4.6 PWM 占空比立即更新

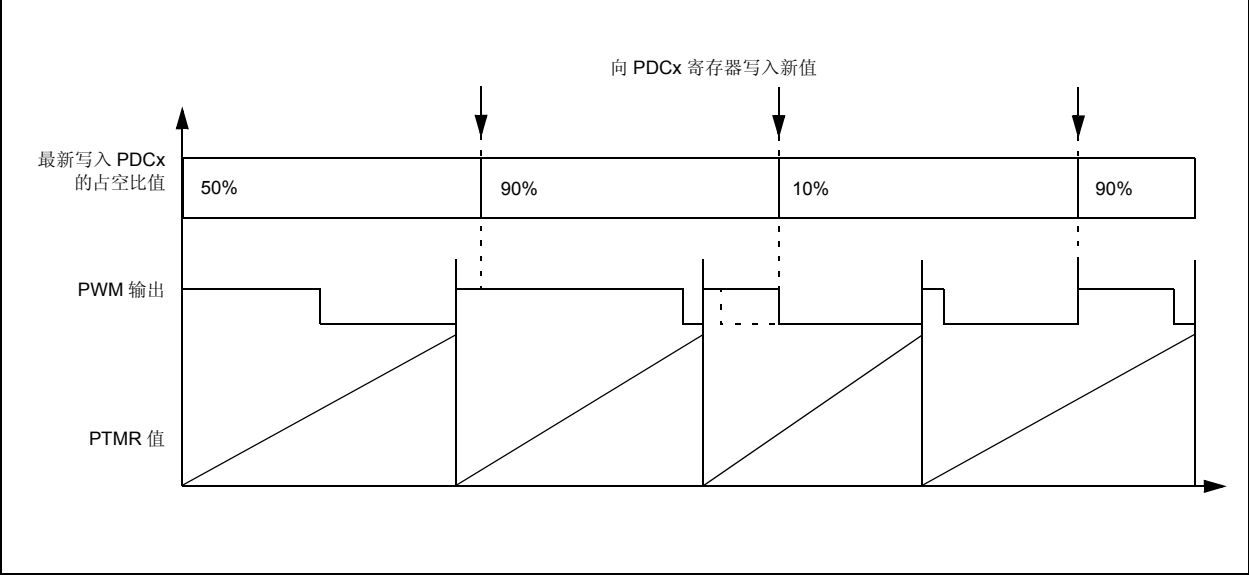
当立即更新使能位置 1 ($IUE = 1$) 时, 任何向占空比寄存器的写入操作均将立即更新为新的占空比。这种功能为用户提供了一种选择, 使其可以立即更新有效的 PWM 占空比寄存器, 而不必等到当前时基周期结束。在闭环伺服应用中, 如果使能了立即更新 ($IUE = 1$), 系统稳定性将由于系统观察到发出系统校正命令之间时间的缩短而提高。

如果新占空比写入时 PWM 输出有效, 且新占空比小于当前时基的值, 则 PWM 脉宽将变窄。如果新占空比写入时 PWM 输出有效, 且新占空比大于当前时基的值, 则 PWM 脉宽将变宽。如果新占空比写入时 PWM 输出无效, 且新占空比大于当前时基的值, PWM 输出将立即有效且对新写入的占空比值保持有效。

图 15-12 所示为立即更新使能 ($IUE = 1$) 时发生占空比更新的时间。

注: IUE 位在 dsPIC30F6010 器件中未用。

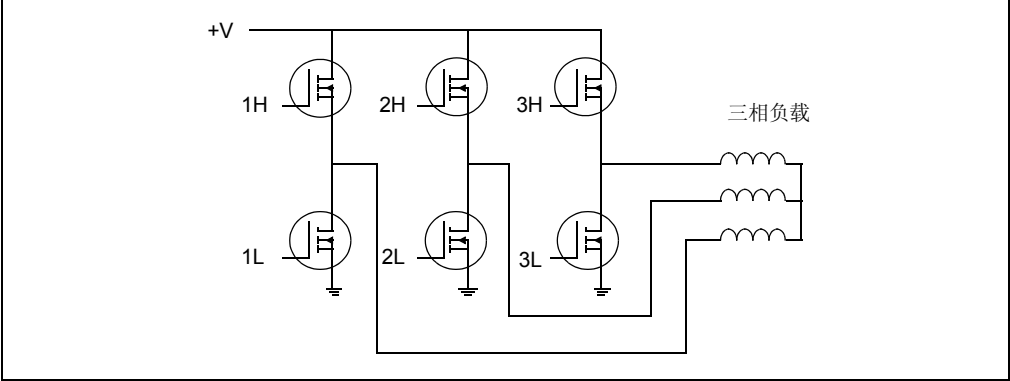
图 15-12: 立即更新使能 (IUE = 1) 时的占空比更新时间



15.5 互补 PWM 输出模式

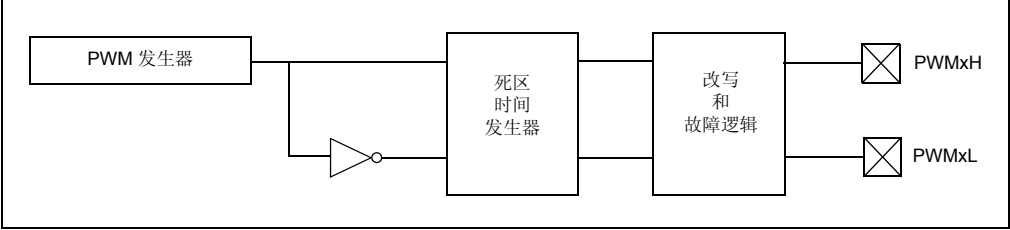
互补输出模式用于驱动与图 15-13 所示类似的逆变器负载。此逆变器拓扑常用于 ACIM 和 BLDC 应用。在互补输出模式中，一对 PWM 输出不能同时有效。每个 PWM 通道和每一对输出引脚均按图 15-14 所示进行内部配置。在器件切换的过程中，有一段两个引脚的输出均无效的短暂时期，此时可以选择插入一个死区时间（见 15.6 “死区时间控制”）。

图 15-13: 互补 PWM 输出的典型负载



通过将 PWMCON1 中相应的 PMODx 位清零，可以将对应的一对 PWM I/O 引脚选择为互补模式。在器件复位时，PWM I/O 引脚默认设置为互补模式。

图 15-14: PWM 通道框图，互补模式



15.6 死区时间控制

当任何一对 PWM I/O 引脚工作在互补输出模式时，死区时间的生成被自动使能。因为电源输出设备不可能瞬时完成切换，所以必须在关闭互补的一对 PWM 输出中的一个和开启另一个晶体管之间提供一定的时间。

6 输出的 PWM 模块有一个可编程死区时间。8 输出的 PWM 模块允许编程两个不同的死区时间。这两个死区时间可以用以下两种方法之一来提高用户灵活性：

- 可以对 PWM 输出信号进行优化使高边和低边晶体管的关断时间不同。在一对互补对中下边晶体管的关断事件和高边晶体管的导通事件之间插入第一个死区时间。在高边晶体管的关断事件和低边晶体管的导通事件之间插入第二个死区时间。
- 两个死区时间可以单独分配给一对 PWM I/O 引脚。此工作模式可以使 PWM 模块可以单独对每一对 PWM I/O 引脚驱动不同的晶体管 / 负载。

15.6.1 死区时间发生器

PWM 模块的每一对互补输出都有一个 6 位的向下计数器，用于插入死区时间。如图 15-15 所示，每个死区时间单元都有与占空比比较输出相连的上升沿和下降沿检测器。

在检测到 PWM 边沿事件时，两个可能的死区时间之一就被装入定时器。根据边沿是上升沿还是下降沿，一对互补输出中的一个将延时到定时器计数减到零才跳变。图 15-16 所示为在一对 PWM 输出中插入死区时间的时序图。为了说明得更清楚，图中上升沿和下降沿事件的两个不同死区时间被放大了。

图 15-15: 一对输出引脚的死区时间单元框图

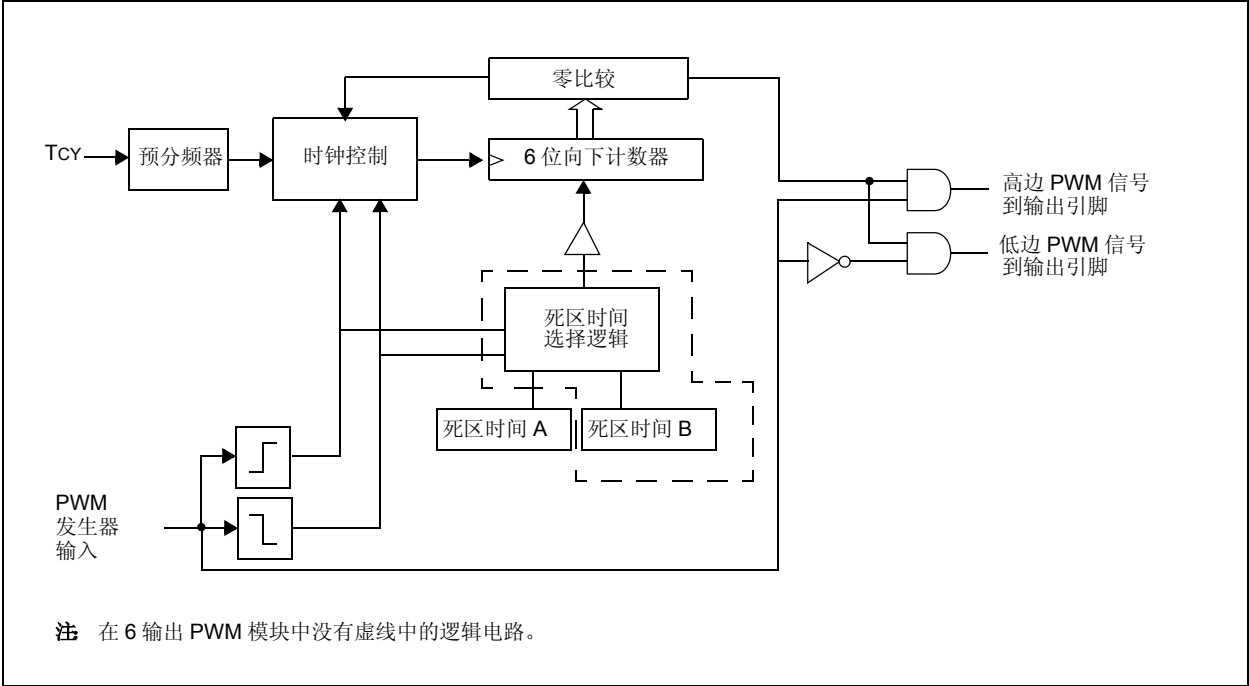
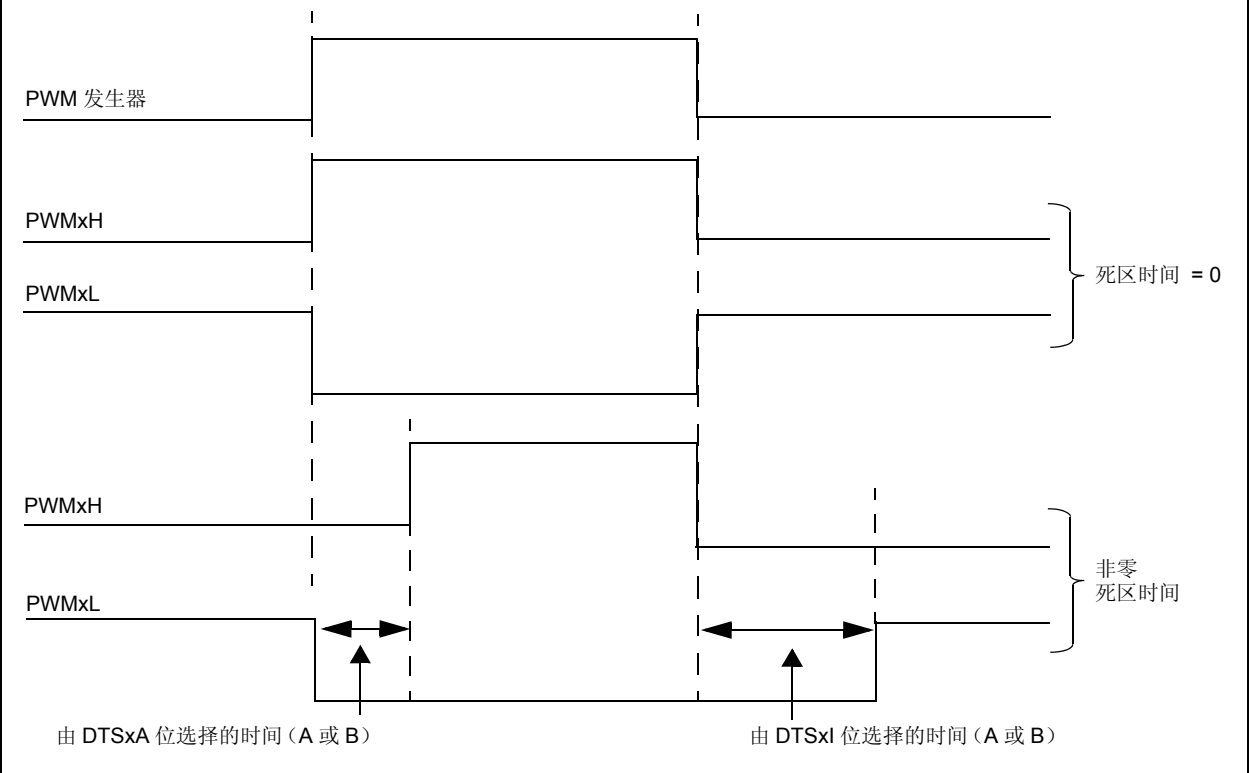


图 15-16: 死区时间插入图



15.6.2 死区时间分配

注： 死区时间分配逻辑电路只适用于包含 8 输出的 PWM 模块的不同 dsPIC 器件。6 输出的 PWM 模块只使用死区时间 A。

DTCON2 寄存器包含控制位，可以将两个可编程死区时间分配到每个互补输出。每个互补输出都有两个死区时间分配控制位。例如，用 DTS1A 和 DTS1I 控制位选择用于 PWM1H/PWM1L 这一对互补输出的死区时间。一对死区时间选择控制位分别称为“死区时间选择有效”（Dead-Time-Select-Active）和“死区时间选择无效”（Dead-Time-Select-Inactive）。这一对控制位中各位的功能如下：

- DTSxA 控制位选择在高边输出被驱动为有效前插入死区时间。
- DTSxI 控制位选择在低边 PWM 输出被驱动为有效前插入死区时间。

表 15-4 总结了每个死区时间选择控制位的功能。

表 15-4: 死区时间选择位

位	功能
DTS1A	选择 PWM1H/PWM1L 死区时间在 PWM1H 被驱动为有效前插入。
DTS1I	选择 PWM1H/PWM1L 死区时间在 PWM1L 被驱动为有效前插入。
DTS2A	选择 PWM1H/PWM1L 死区时间在 PWM2H 被驱动为有效前插入。
DTS2I	选择 PWM1H/PWM1L 死区时间在 PWM2L 被驱动为有效前插入。
DTS3A	选择 PWM1H/PWM1L 死区时间在 PWM3H 被驱动为有效前插入。
DTS3I	选择 PWM1H/PWM1L 死区时间在 PWM3L 被驱动为有效前插入。
DTS4A	选择 PWM1H/PWM1L 死区时间在 PWM4H 被驱动为有效前插入。
DTS4I	选择 PWM1H/PWM1L 死区时间在 PWM4L 被驱动为有效前插入。

15.6.3 死区时间范围

死区时间 A 和死区时间 B 是通过选择输入时钟预分频比和 6 位无符号死区时间计数值来设置的。死区时间单元提供了四种输入时钟预分频器选项，使用户根据器件的工作频率选择适当的死区时间范围。可以为两个死区时间值中的每一个独立地选择时钟预分频器选项。死区时间时钟预分频比是使用 DTCON1 SFR 中的 DTAPS<1:0> 和 DTBPS<1:0> 控制位选择的。每个死区时间均可选择以下时钟预分频器选项：

- Tcy
- 2 Tcy
- 4 Tcy
- 8 Tcy

公式 15-4: 死区时间计算

$$DT = \frac{\text{死区时间}}{\text{预分频比} \cdot Tcy}$$

注： DT（死区时间）是 DTA<5:0> 或 DTB<5:0> 寄存器中的值。

表 15-5 举例显示了死区时间范围与所选的输入时钟预分频器和器件工作频率之间的关系。

表 15-5: 死区时间范围示例

Tcy (Fcy)	预分频器选择	精度	死区时间范围
33 ns (30 MHz)	4 Tcy	130 ns	130 μs—9 μs
50 ns (20 MHz)	4 Tcy	200 ns	200 μs—12 μs
100 ns (10 MHz)	2 Tcy	200 ns	200 μs—12 μs
100 ns (10 MHz)	1 Tcy	100 ns	100 μs—6 μs

15.6.4 死区时间失真

对于 PWM 占空比较小的情况，死区时间相对于有效 PWM 时间的比例可能会较大。在极端的情况下，当占空比小于或等于编程占空比时，不会产生 PWM 脉冲。在这些情况下，插入的死区时间将会导致 PWM 模块产生的波形失真。通过保持 PWM 占空比至少比死区时间大三倍，用户即可以确保死区时间失真最小。用其他技术也可以纠正死区时间失真，例如使用闭环电流控制。

占空比接近 100% 时也会产生类似的失真。应用中最大占空比的选择应该确保 PWM 信号的最小无效时间比死区时间大三倍以上。

15.7 独立 PWM 输出模式

独立 PWM 输出模式对于驱动诸如图 15-17 所示的一类负载很有用。当 PWMCON1 寄存器中的相应 PMOD 位置 1 时，某一对 PWM 输出就处于独立输出模式。在独立模式中，死区时间发生器被禁止，并且对于给定的一对输出引脚，引脚状态没有限制。

图 15-17: 不对称逆变器

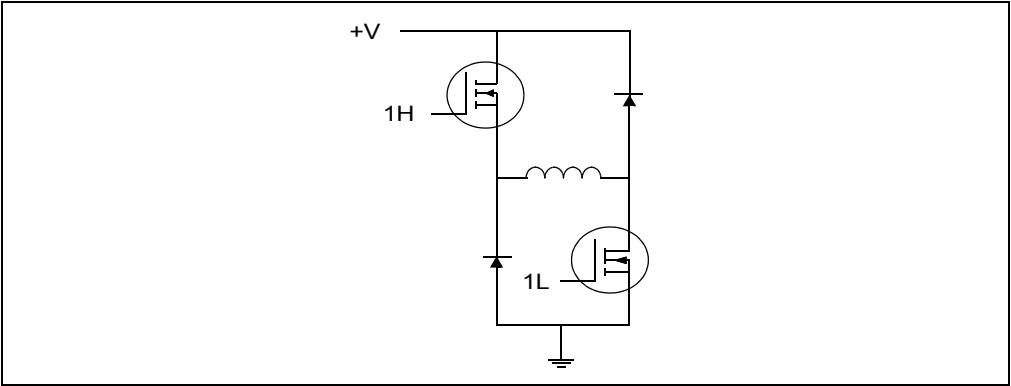
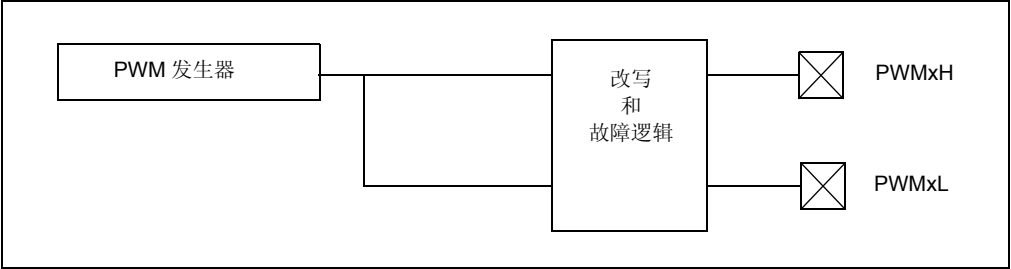


图 15-18: 独立模式、一对单输出引脚 PWM 框图



15.8 PWM 输出改写

PWM 输出改写位可以让用户手动将 PWM I/O 引脚驱动为指定逻辑状态，而不受占空比较单元的影响。在控制各种电子换相电机时，PWM 改写位很有用。

所有与 PWM 输出改写功能相关的控制位都在 OVDCON 寄存器中。OVDCON 寄存器的高半部分包含 8 个位 (POVDxx)，决定哪个 PWM I/O 引脚将要被改写。OVDCON 寄存器的低半部分也包含 8 个位 (POUTxx)，决定当通过 POVDxx 位改写时 PWM I/O 引脚的状态。

POVD 位为低有效控制位。当 POVD 位置 1 时，相应的 POUTxx 位对 PWM 输出没有影响。当一个 POVD 位清零时，相应的 PWM I/O 引脚的输出将由 POUT 位的状态决定。当 POUT 位置 1 时，PWM 引脚将被驱动为有效状态。当 POUT 引脚清零时，PWM 引脚将被驱动为无效状态。

15.8.1 互补输出模式的改写控制

当一对 PWM I/O 引脚工作在互补模式 (PMODx = 0) 时，PWM 模块将不会允许对输出进行某些改写。模块将不允许同一对输出的两个引脚同时变为有效。每对输出的高边引脚总是占有优先权。

注： 在插入死区时间期间，如果手动改写 PWM 通道，死区时间仍将持续。

15.8.2 改写同步

如果 OSYNC 位置 1 (PWMCON2<1>)，所有通过 OVDCON 寄存器执行的输出改写将与 PWM 时基同步。同步的输出改写将发生在以下时间：

- 若是边沿对齐模式，则当 PTMR 为 0 时。
- 若是中心对齐模式，则当 PTMR 为 0 时或者
- 当 PTMR 与 PTPER 的值匹配时。

当使能了改写同步功能时，该功能可用于在 PWM 输出引脚上避免不希望的窄脉冲。

15.8.3 输出改写示例

图 15-19 显示了使用 PWM 输出改写功能可能会产生的波形示例。该图显示了一个 BLDC 电机的六步换相序列。该电机通过一个如图 15-13 所示的三相逆变器驱动。当检测到适当的转子位置时，PWM 输出会切换到序列中下一个换相状态。在此例中，PWM 输出被驱动为特定的逻辑状态。表 15-6 列出了用于产生图 15-19 中信号的 OVDCON 寄存器值。

PWM 占空比寄存器可以和 OVDCON 寄存器配合使用。占空比寄存器控制流经负载的电流，OVDCON 寄存器控制换相。图 15-20 就显示了这样一个例子。表 15-7 列出了用于产生图 15-20 中信号的 OVDCON 寄存器值。

表 15-6: PWM 输出改写示例 1

状态	OVDCON<15:8>	OVDCON<7:0>
1	00000000b	00100100b
2	00000000b	00100001b
3	00000000b	00001001b
4	00000000b	00011000b
5	00000000b	00010010b
6	00000000b	00000110b

图 15-19: PWM 输出改写示例 1

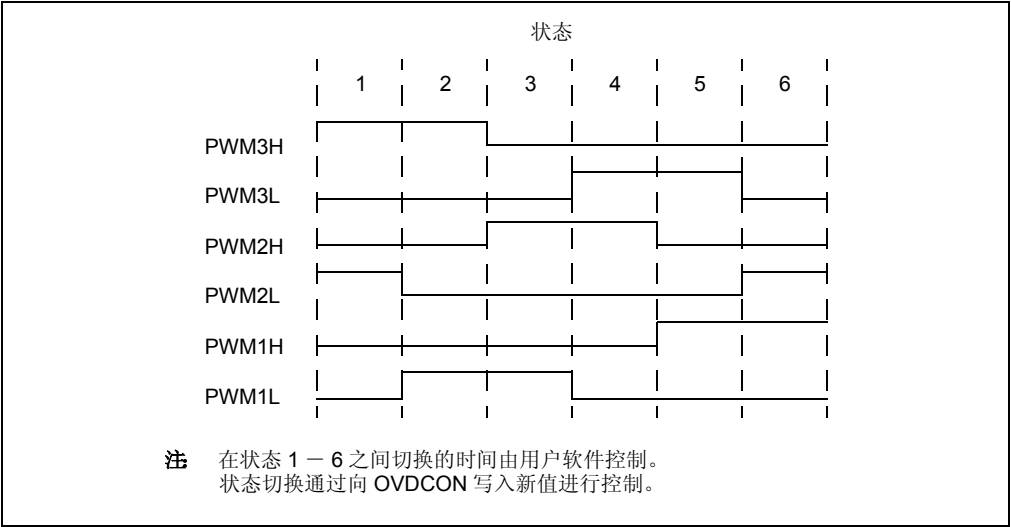
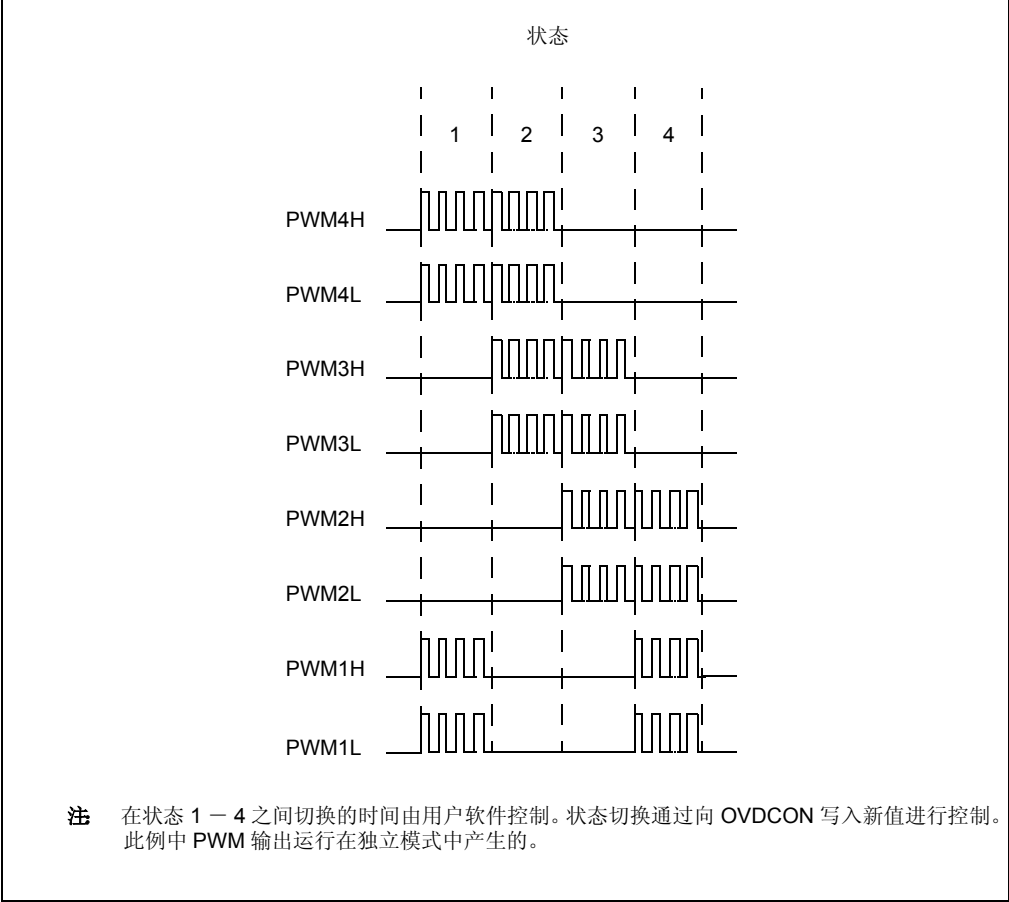


表 15-7: PWM 输出改写示例 2

状态	OVDCON<15:8>	OVDCON<7:0>
1	11000011b	00000000b
2	11110000b	00000000b
3	00111100b	00000000b
4	00001111b	00000000b

图 15-20: PWM 输出改写示例 2



15.9 PWM 输出和极性控制

PWMCON1 中的 PENxx 控制位用于使能每个 PWM 输出引脚供模块使用。当引脚使能为 PWM 输出时，控制引脚的 PORT 和 TRIS 寄存器被禁止。

除了 PENxx 控制位，在器件配置寄存器 FBORPOR 中还有三个配置位提供 PWM 输出引脚控制。

- 配置位 HPOL
- 配置位 LPOL
- 配置位 PWMPIN

这三个配置位与位于 PWMCON1 的 PWM 使能位（PENxx）配合工作。这些配置位确保在发生器件复位后，PWM 引脚处于正确的状态。

15.9.1 输出极性控制

PWM I/O 引脚的极性是在器件编程的过程中，通过器件配置寄存器中 FBORPOR 的配置位 HPOL 和 LPOL 设置的。配置位 HPOL 设置高边 PWM 输出 PWM1H — PWM4H 的输出极性。配置位 LPOL 设置低边 PWM 输出 PWM1L — PWM4L 的输出极性。

如果极性配置位编程为 1，相应的 PWM I/O 引脚的输出极性将为高电平有效。如果极性配置位编程为 0，则相应的 PWM 引脚极性为低电平有效。

15.9.2 PWM 输出引脚复位状态

配置位 PWMPIN 决定器件复位时的 PWM 输出引脚的行为，且该位还可用于消除对由 PWM 模块控制的器件所连接的外部上拉 / 下拉电阻的需要。

如果配置位 PWMPIN 编程为 1，控制位 PENxx 将在器件复位时清零。因此，所有 PWM 输出将为三态，并由相应的 PORT 和 TRIS 寄存器控制。

如果配置位 PWMPIN 编程为 0，控制位 PENxx 将在器件复位时置 1。所有的 PWM 引脚在器件复位时使能为 PWM 输出，并将处于由 HPOL 和 LPOL 配置位规定的无效状态。

15.10 PWM 故障引脚

有两个与 PWM 模块相关的故障引脚，它们是 $\overline{FLTA}$ 和 $\overline{FLT B}$ 。当使能时，可以选择用这些引脚将 PWM I/O 引脚驱动为定义的状态。此操作无需软件干预即可发生，因此可以快速处理故障事件。

根据不同的 dsPIC 器件，故障引脚也可能会有其他复用功能。当用作故障输入时，每个故障引脚都可通过其相应 PORT 寄存器读取。 $\overline{FLTA}$ 和 $\overline{FLT B}$ 引脚作为低电平有效输入引脚，因此可以轻易地将许多输入源一起进行“线或”后，通过一个外部上拉电阻输入给同一个引脚。当不与 PWM 模块一起使用时，这些引脚可以作为通用功能 I/O 或其他复用功能。每个故障引脚都有与其相关的中断向量、中断标志位、中断使能位和中断优先级位。

$\overline{FLTA}$ 引脚的功能由 FLTACON 寄存器控制， $\overline{FLT B}$ 引脚的功能由 FLTBCON 寄存器控制。

15.10.1 故障引脚使能位

寄存器 FLTACON 和 FLTBCON 各有 4 个控制位（FxEN1 — FxEN4），这些控制位决定某对 PWM I/O 引脚是否要由故障输入引脚控制。要将某一对 PWM I/O 引脚使能为故障改写，必须置位寄存器 FLTACON 或 FLTBCON 中的相应位。

如果寄存器 FLTACON 或 FLTBCON 中所有的使能位都被清零，则这对故障输入引脚 PWM 模块没有影响，并且不会产生故障中断。

15.10.2 故障状态

特殊功能寄存器 FLTACON 和 FLTBCON 各有 8 个位，这些位决定当故障输入引脚变为有效时每个 PWM I/O 引脚的状态。当这些位清零时，PWM I/O 引脚将被驱动为无效状态。当这些位置 1 时，PWM I/O 引脚将被驱动为有效状态。有效和无效状态与 PWM I/O 引脚被定义的极性（通过 HPOL 和 LPOL 器件配置位设置）相对应。

当 PWM 模块的一对 I/O 处于互补模式，并且两个引脚都编程为在产生故障状态时驱动为有效时，存在一种特殊情况。在互补模式中高边的引脚将始终优先，因此两个 I/O 引脚不能同时被驱动为有效。

15.10.3 故障输入模式

每个故障输入引脚都有两种工作模式：

- **锁存模式：**当故障引脚驱动为低电平时，PWM 输出将进入 FLT_xCON 寄存器定义的状态。PWM 输出将保持在此状态，直到故障引脚被驱动为高电平并且相应的中断标志（FLT_xIF）由软件清零。当这两种行为都发生后，PWM 输出将在下一个 PWM 周期开始时或在半周期边界返回到正常工作状态，而无论立即更新使能位（IUE）为何值。如果中断标志在故障状态结束前清零，PWM 模块将等到故障引脚不再有效时才恢复输出。
- **逐个周期模式：**当故障输入引脚驱动为低电平时，只要故障引脚保持为低电平，PWM 输出将会一直保持定义的故障状态。在故障引脚被驱动为高电平后，PWM 输出将在下一个 PWM 周期开始时（或中心对齐模式的半周期边界）返回正常工作状态，即使在立即更新被使能的情况下也是如此。

各故障输入引脚的工作模式通过控制位 FLTAM 和 FLTBM（FLTACON<7> 和 FLTBCON<7>）选择。

15.10.3.1 进入故障状态

当故障引脚被使能并驱动为低电平时，无论 PDC_x 和 OVDCON 寄存器中的值如何，PWM 引脚都会立即驱动为其编程的故障状态。故障操作的优先级高于所有其他 PWM 控制寄存器。

15.10.3.2 退出故障状态

必须通过外部电路将故障输入引脚驱动为高电平并将故障中断标志清零（仅限锁存模式），才能清除故障状态。在故障引脚条件被清除后，PWM 模块将在下一个 PWM 周期或半周期边界恢复 PWM 输出信号。对于产生边沿对齐的 PWM 信号的情况，PWM 输出将会在 PTMR = 0 时恢复。对于产生中心对齐的 PWM 信号的情况，PWM 输出将会在任意 PTMR = 0 或 PTMR = PTPER 先发生时恢复。

PWM 时基被禁止（PTEN = 0）时，将发生上述规则例外的事件。如果 PWM 时基被禁止，PWM 模块将在故障条件被清除后立即恢复 PWM 输出信号。

15.10.4 故障引脚优先级

如果两个故障输入引脚均被分配为控制某一对 PWM 引脚，为 $\overline{\text{FLTA}}$ 输入引脚编程的故障状态将优先于 $\overline{\text{FLT B}}$ 输入引脚。

当故障 A 的条件被清除时，会发生两个行为之一。如果 $\overline{\text{FLT B}}$ 输入仍然被使能，PWM 输出将在下个周期或半周期边界返回 $\overline{\text{FLT BCON}}$ 寄存器中编程的状态。如果 $\overline{\text{FLT B}}$ 输入被禁止，PWM 输出将在下个周期或半周期边界返回正常工作状态。

注：当 $\overline{\text{FLTA}}$ 引脚编程为锁存模式时，PWM 输出将不会返回故障 B 状态或正常工作状态，直到故障 A 中断标志被清零并且 $\overline{\text{FLTA}}$ 引脚禁止。

15.10.5 故障引脚软件控制

每个故障引脚都可以通过软件手动控制。因为每个故障输入都与端口 I/O 引脚共用，通过将对应的 TRIS 位清零，可以将端口（PORT）引脚配置为输出。当引脚的 PORT 位清零时，故障输入被激活。

注：当通过软件控制故障输入时，用户应该特别注意。如果故障引脚的 TRIS 位清零，故障输入就无法从外部驱动。

15.10.6 故障时序示例

图 15-21: 故障时序示例，逐个周期模式

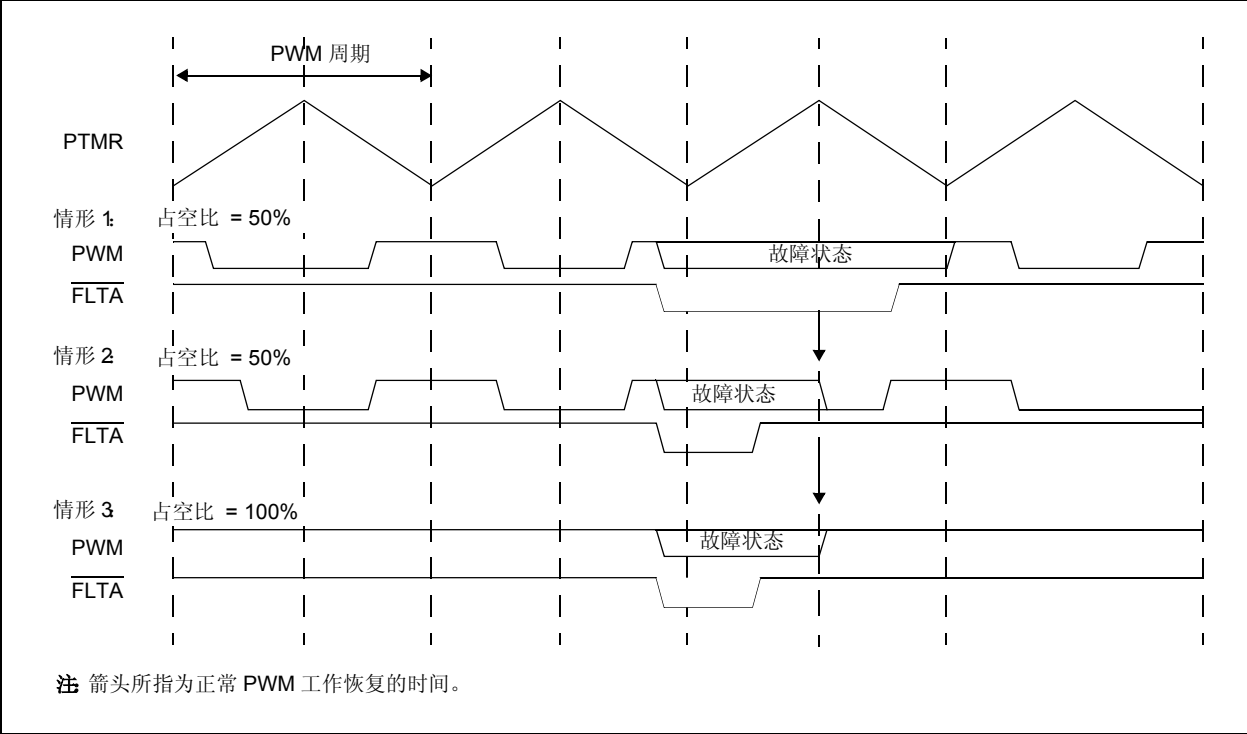


图 15-22: 故障时序示例，锁存模式

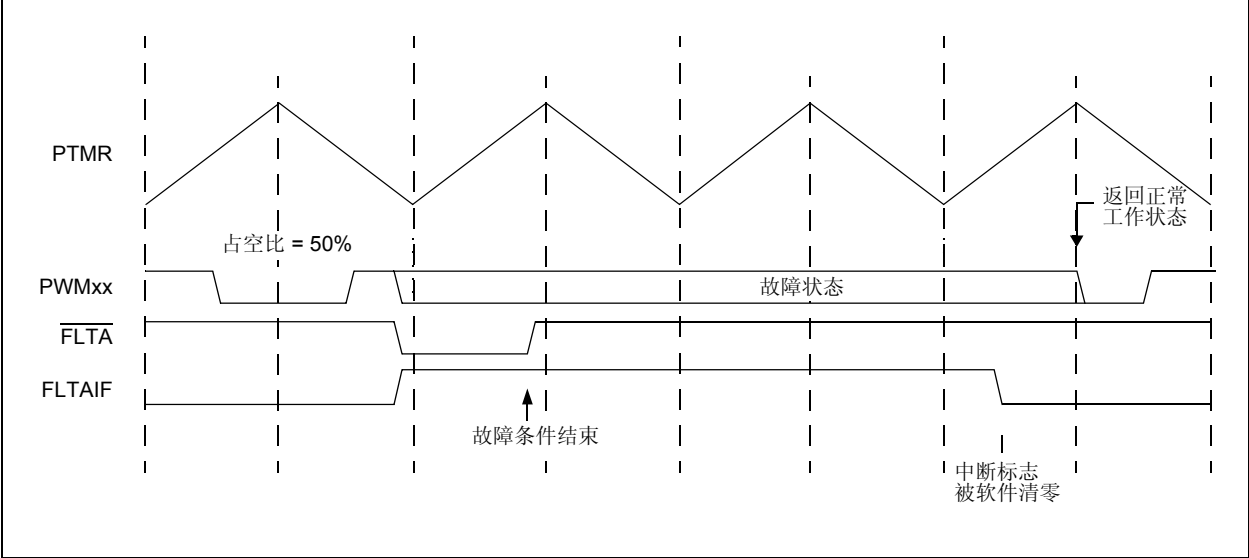
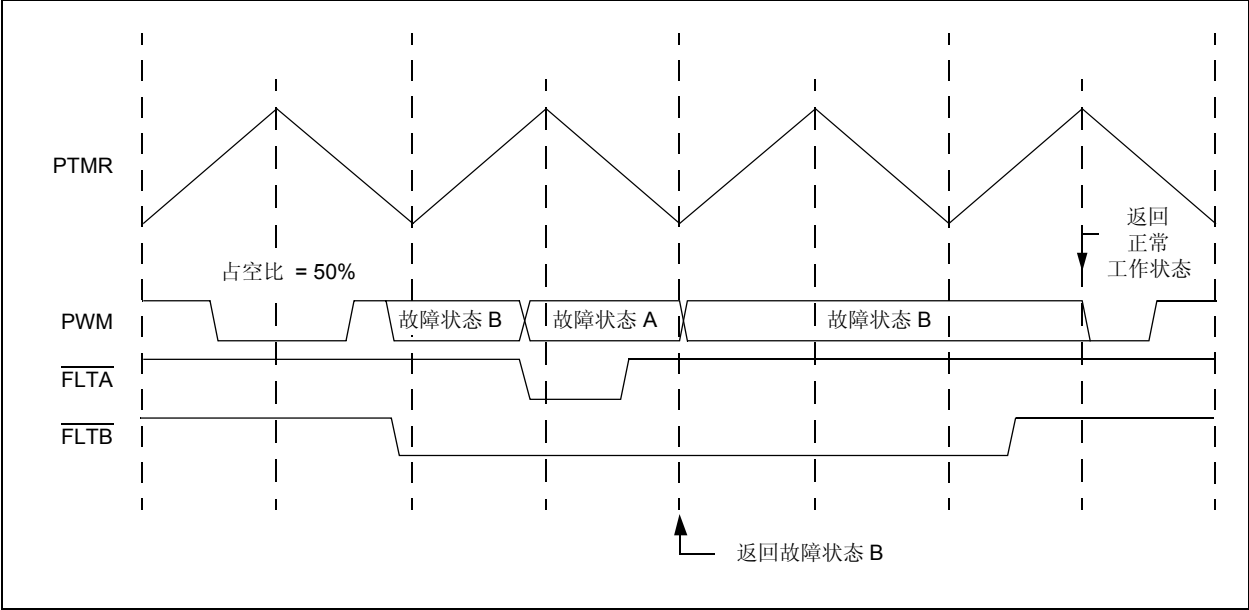


图 15-23: 故障时序示例，逐个周期模式，优先级工作



15.11 PWM 更新锁定

在某些应用中，在新值生效前写入所有的占空比和周期寄存器是很重要的。更新禁止功能允许用户指定模块可以使用新占空比和周期值的时间。通过将 **UDIS** 控制位 (**PWMCON2<0>**) 置 1 可启用 **PWM** 更新锁定功能。

UDIS 位会影响所有的占空比寄存器 (**PDC1** — **PDC4**) 和 **PWM** 时基周期缓冲器 **PTPER**。要执行更新锁定，用户应该执行以下步骤：

- 将 **UDIS** 位置 1。
- 如果适用，写所有占空比寄存器和 **PTPER**。
- 将 **UDIS** 位清零以重新启用更新。

注： 要使用 **PWM** 更新锁定功能，必须禁止立即更新 (**IUE** = 0)。

15.12 PWM 特殊事件触发器

PWM 模块有一个特殊事件触发器，可以使 A/D 转换与 PWM 时基同步。可以将 A/D 采样和转换时间编程为在 PWM 周期中的任何时间发生。特殊事件触发器可以使用户将采集 A/D 转换结果的时间与占空比值更新的时间之间的延迟减到最小。

PWM 特殊事件触发器有一个 SFR（SEVTCMP）和四个后分频器控制位（SEVOPS<3:0>）用于控制其工作方式。用于产生特殊事件触发信号的 PTMR 值装入 SEVTCMP 寄存器。

当 PWM 时基处于向上 / 向下计数模式时，还需要一个控制位指定特殊事件触发信号的计数方向。此计数方向是通过使用 SEVTCMP 的 MSb 中的 SEVTDIR 控制位选择的。如果 SEVTDIR 位清零，特殊事件触发信号将在 PWM 时基的向上计数周期产生。如果 SEVTDIR 位置 1，特殊事件触发信号将在 PWM 时基的向下计数周期产生。如果 PWM 时基不配置为向上 / 向下计数模式，SEVTDIR 控制位不起作用。

15.12.1 特殊事件触发器使能

PWM 模块总是会产生特殊事件触发信号。此信号可以由 A/D 模块选用。如需更多有关使用特殊事件触发器的信息，请参见第 17 章“10 位 A/D 转换器”。

15.12.2 特殊事件触发器后分频器

PWM 特殊事件触发器有一个允许后分频比为 1:1 到 1:16 的后分频器。当不需要在每个 PWM 周期同步 A/D 转换时，后分频器是很有用的。通过写 PWMCON2 SFR 中的 SEVOPS<3:0> 控制位可配置后分频器。

特殊事件输出后分频器在下列事件发生时清零：

- 对 SEVTCMP 寄存器的任何写入。
- 任何器件复位。

15.13 器件低功耗模式下的工作

15.13.1 休眠模式下的 PWM 工作

当器件进入休眠模式后，系统时钟被禁止。因为 PWM 时基的时钟来自系统时钟源（Tcy），所以它也会被禁止。所有使能的 PWM 输出引脚都会被冻结在进入休眠模式之前有效的输出状态。

如果 PWM 模块用于控制电源应用中的负载，在执行 PWRSAV 指令前，用户应该将 PWM 模块的输出置为一个“安全”的状态。根据不同的应用，当 PWM 输出冻结在特定输出状态下时，负载可能会开始消耗额外的电流。例如，如以下代码示例所示，OVDCON 寄存器可以用于手动关闭 PWM 输出引脚。

```
; This code example drives all PWM pins to the inactive state  
; before executing the PWRSAV instruction.
```

```
CLR      OVDCON      ; Force all PWM outputs inactive  
PWRSAV   #0          ; Put the device in SLEEP mode  
SETM.B   OVDCONH     ; Set POVD bits when device wakes.
```

如果通过 FLTxCN 寄存器使能了故障 A 和故障 B 输入引脚来控制 PWM 引脚，则这两个输入引脚将在器件处于休眠模式时继续正常工作。当器件处于休眠模式时，如果其中一个故障引脚被驱动为低电平，PWM 输出将被驱动为在 FLTxCN 寄存器中编程的故障状态。

故障输入引脚还具有将 CPU 从休眠中唤醒的功能。如果故障中断使能位置 1 (FLTxiE = 1)，则当故障引脚被驱动为低电平时，器件将从休眠中被唤醒。如果故障引脚中断的优先级高于当前 CPU 的优先级，则当器件被唤醒时，将在故障引脚中断向量处开始程序执行。否则，程序将在 PWRSAV 指令后的下一条指令开始执行。

15.13.2 空闲模式下的 PWM 工作

当器件进入空闲模式时，系统时钟源保持工作而 CPU 停止执行代码。PWM 模块可以选择继续在空闲模式工作。通过 PTSIDL 位 (PTCON<13>) 可选择 PWM 模块在空闲模式是停止工作还是继续正常工作。

如果 PTSIDL = 0，当器件进入空闲模式时，模块将继续正常工作。如果使能了 PWM 时基中断，则可以使用它将器件从空闲状态唤醒。如果 PWM 时基中断使能位置 1 (PTIE = 1)，则 PWM 时基中断产生时，器件将从空闲模式唤醒。如果 PWM 时基中断的优先级高于当前 CPU 的优先级，则当器件被唤醒时，将在 PWM 时基中断向量处开始程序执行。否则，程序将从 PWRSAV 指令后的下一个指令开始执行。

如果 PTSIDL = 1，在空闲模式下模块将停止工作。如果 PWM 模块被编程为在空闲模式停止工作，PWM 输出和故障输入引脚的工作情况将与休眠模式的工作状况相同（参见 15.13.1 “休眠模式下的 PWM 工作”中的讨论）。

15.14 用于器件仿真的特殊功能

PWM 模块的一个特殊功能可以支持调试环境。当硬件仿真器或调试器停下来以检查存储器内容时，所有使能的 PWM 引脚均可以选择显为三态。用户应该连接上拉或下拉电阻，以确保当器件停止执行时，PWM 输出被驱动为正确状态。

器件复位时，PWM 输出引脚的功能和输出引脚极性由三个器件配置位决定（参见 15.9 “PWM 输出和极性控制”）。硬件调试器或仿真器工具提供了改变这些配置位值的方法。如需更多信息，请参见相应工具的用户手册。

表 15-8: 与 8 输出 PWM 模块相关的寄存器

名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值
INTCON1	0080	NSTDIS	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
INTCON2	0082	ALTIVT	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
IFS2	0088	—	—	—	FLTBIF	FLTAIF	—	—	—	PWMIF	—	—	—	—	—	—	—	0000 0000 0000 0000
IEC2	0090	—	—	—	FLTBIE	FLTAIE	—	—	—	PWMIE	—	—	—	—	—	—	—	0000 0000 0000 0000
IPC9	00A6	—	PWMIP<2:0>			—	—	—	—	—	—	—	—	—	—	—	—	0100 0100 0100 0100
IPC10	00A8	—	FLTAIP<2:0>			—	—	—	—	—	—	—	—	—	—	—	—	0100 0100 0100 0100
IPC11	00AA	—	—	—	—	—	—	—	—	—	—	—	—	—	FLTBIP<2:>			0000 0000 0000 0000
PTCON	01C0	PTEN	—	PTSIDL	—	—	—	—	—	PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>		0000 0000 0000 0000
PTMR	01C2	PTDIR	PWM 时机寄存器															0000 0000 0000 0000
PTPER	01C4	—	PWM 时机周期寄存器															0111 1111 1111 1111
SEVTCMP	01C6	SEVTDIR	PWM 特殊事件比较寄存器															0000 0000 0000 0000
PWMCON1	01C8	—	—	—	—	PMOD4	PMOD3	PMOD2	PMOD1	PEN4H	PEN3H	PEN2H	PEN1H	PEN4L	PEN3L	PEN2L	PEN1L	0000 0000 0000 0000
PWMCON2	01CA	—	—	—	—	SEVOPS<3:0>				—	—	—	—	—	IUE	OSYNC	UDIS	0000 0000 0000 0000
DTCON1	01CC	DTBPS<1:0>		死区时间 B 值寄存器						DTAPS<1:0>		死区时间 A 值寄存器						0000 0000 0000 0000
DTCON2	01CE	—	—	—	—	—	—	—	—	DTS4A	DTS4I	DTS3A	DTS3I	DTS2A	DTS2I	DTS1A	DTS1I	0000 0000 0000 0000
FLTACON	01D0	FAOV4H	FAOV4L	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L	FLTAM	—	—	—	FAEN4	FAEN3	FAEN2	FAEN1	0000 00-0 0000 0000
FLTBCON	01D2	FBOV4H	FBOV4L	FBOV3H	FBOV3L	FBOV2H	FBOV2L	FBOV1H	FBOV1L	FLTBM	—	—	—	FBEN4	FBEN3	FBEN2	FBEN1	0000 0000 0000 0000
OVDCON	01D4	POVD4H	POVD4L	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L	POUT4H	POUT4L	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L	1111 1111 00-0 0000
PDC1	01D6	PWM 占空比 1 寄存器																0000 0000 0000 0000
PDC2	01D8	PWM 占空比 2 寄存器																0000 0000 0000 0000
PDC3	01DA	PWM 占空比 3 寄存器																0000 0000 0000 0000
PDC4	01DC	PWM 占空比 4 寄存器																0000 0000 0000 0000

- 注
- 1: 控制位 PENxx 的复位状态取决于器件配置位 PWMIPIN 的状态。
 - 2: 阴影的寄存器和位单元在 6 输出 MCPWM 模块中未使用。
 - 3: IUE 位在器件 dsPIC30F6010 中未用。

表 15-9: 与 6 输出 PWM 模块相关的寄存器

名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值
INTCON1	0080	NSTDIS	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
INTCON2	0082	ALTIVT	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
IFS2	0088	—	—	—	—	FLATIF	—	—	—	PWMIF	—	—	—	—	—	—	—	0000 0000 0000 0000
IEC2	0090	—	—	—	—	FLTAIE	—	—	—	PWMIE	—	—	—	—	—	—	—	0000 0000 0000 0000
IPC9	00A6	—	PWMIP<2:0>			—	—	—	—	—	—	—	—	—	—	—	—	0100 0100 0100 0100
IPC10	00A8	—	FLTAIP<2:0>			—	—	—	—	—	—	—	—	—	—	—	—	0100 0100 0100 0100
PTCON	01C0	PTEN	—	PTSIDL	—	—	—	—	—	PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>		0000 0000 0000 0000
PTMR	01C2	PTDIR	PWM 时基寄存器															0000 0000 0000 0000
PTPER	01C4	—	PWM 时基周期寄存器															0111 1111 1111 1111
SEVTCMP	01C6	SEVTDIR	PWM 特殊事件比较寄存器															0000 0000 0000 0000
PWMCON1	01C8	—	—	—	—	—	PMOD3	PMOD2	PMOD1	—	PEN3H	PEN2H	PEN1H	—	PEN3L	PEN2L	PEN1L	0000 0000 0000 0000
PWMCON2	01CA	—	—	—	—	—	—	—	—	—	—	—	—	—	IUE	OSYNC	UDIS	0000 0000 0000 0000
DTCON1	01CC	—	—	—	—	—	—	—	—	DTAPS<1:0>		死区时基 A 值寄存器						0000 0000 0000 0000
保留	01CE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
FLTACON	01D0	—	—	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L	FLTAM	—	—	—	FAEN4	FAEN3	FAEN2	FAEN1	0000 00-0 0000 0000
保留	01D2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
OVDCON	01D4	—	—	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L	—	—	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L	1111 1111 00-0 0000
PDC1	01D6	PWM 占空比 1 寄存器																0000 0000 0000 0000
PDC2	01D8	PWM 占空比 2 寄存器																0000 0000 0000 0000
PDC3	01DA	PWM 占空比 3 寄存器																0000 0000 0000 0000

- 注
- 1: 控制位 PENxx 的复位状态取决于器件配置位 PWMPIN 的状态。
 - 2: 阴影的寄存器和位单元在 6 输出 MCPWM 模块中未使用。
 - 3: IUE 位在器件 dsPIC30F6010 中未用。

15.15 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相关的，通过适当修改即可使用，但使用中可能会受到一定限制。当前与 MCPWM 模块相关的应用笔记有：

标题	应用笔记编号
PIC18CXXX/PIC16CXXX Servomotor	AN696
dsPIC30F 在无传感器 BLDC 控制中的应用	AN901
使用 dsPIC30F 实现交流感应电机的矢量控制	AN908
使用 dsPIC30F2010 控制带传感器的 BLDC	AN957
使用 dsPIC30F MCU 控制交流感应电机	AN984

注：如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

15.16 版本历史

版本 A

这是本文档的初始发行版。

版本 B

此版本扩充了有关 dsPIC30F MCPWM 模块的信息。

版本 C

此版本加入了本文档更新时已知的所有勘误表。

版本 D

此版本加入了立即更新使能功能（IUE）位。

注:

第 16 章 正交编码器接口 (QEI)

目录

本章包括下列主题:

16.1 模块简介	16-2
16.2 控制和状态寄存器	16-4
16.3 可编程数字噪声滤波器	16-9
16.4 正交解码器	16-10
16.5 16 位向上 / 向下位置计数器	16-12
16.6 QEI 用作备用 16 位定时器 / 计数器	16-16
16.7 正交编码器接口中断	16-17
16.8 I/O 引脚控制	16-18
16.9 低功耗模式下的 QEI 工作	16-19
16.10 复位的影响	16-19
16.11 设计技巧	16-21
16.12 相关应用笔记	16-22
16.13 版本历史	16-23

16.1 模块简介

16.1.1 功能综述

正交编码器（又名增量式编码器或光电式编码器），用于检测旋转运动系统的位置和速度。正交编码器可以对多种电机控制应用实现闭环控制，诸如开关磁阻（SR）电机和交流感应电机（ACIM）。

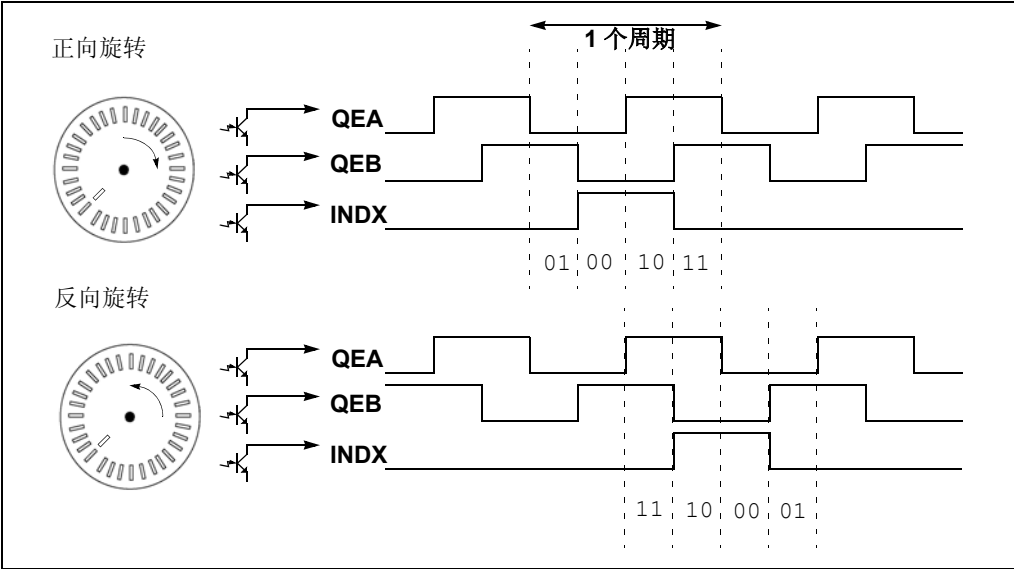
典型的增量式编码器包括一个放置在电机传动轴上的开槽的轮子和一个用于检测该轮上槽口的发射器 / 检测器模块。通常，有三个输出，分别为：A 相、B 相和索引（INDEX），所提供的信息可被解码，用以提供有关电机轴的运动信息，包括距离和方向。

A 相（QEA）和 B 相（QEB）这两个通道间的关系是惟一的。如果 A 相超前 B 相，那么电机的旋转方向被认为是正向的。如果 A 相落后 B 相，那么电机的旋转方向则被认为是反向的。第三个通道称为索引脉冲，每转一圈产生一个脉冲，作为基准用来确定绝对位置。这三个信号的相关时序图，参见图 16-1。

编码器产生的正交信号可以有四种各不相同的状态。这些状态在图 16-1 中用一个计数周期表示。请注意，当旋转的方向改变时，这些状态的顺序与此相反。

正交解码器捕捉相位信号和索引脉冲，并将信息转换为位置脉冲的数字计数值。通常，当传动轴向某一个方向旋转时，该计数值将递增计数；而当传动轴向另一个方向旋转时，则递减计数。

图 16-1: 正交编码器接口信号

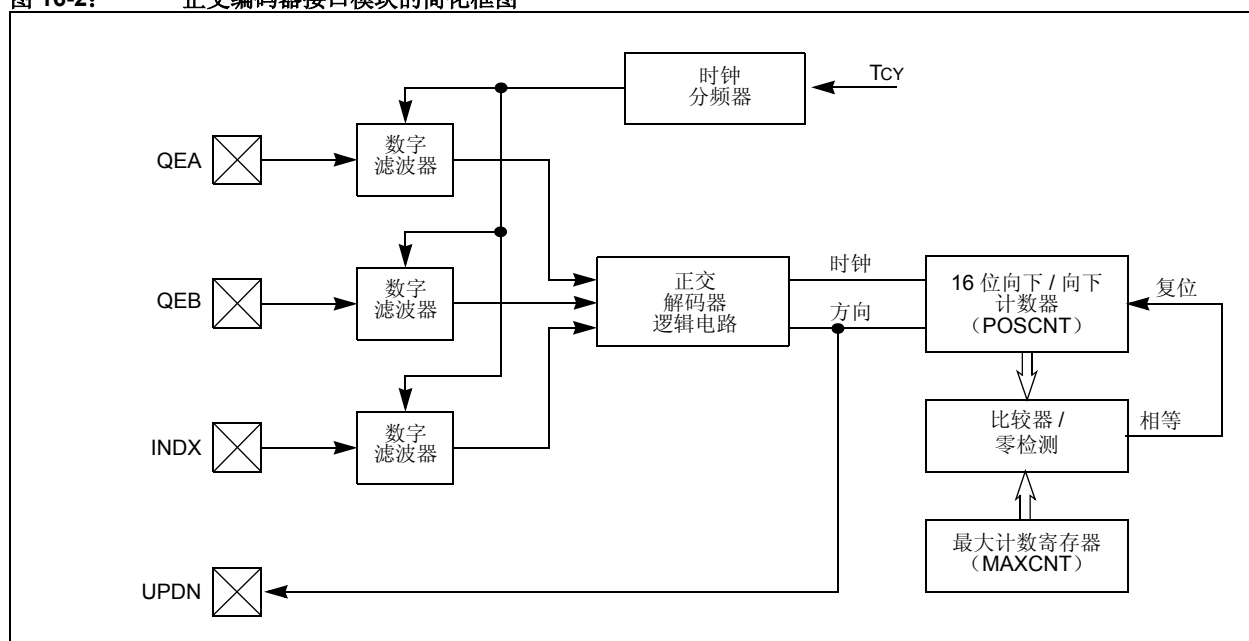


正交编码器接口 (QEI) 模块提供了与增量式编码器的接口。QEI 由对 A 相和 B 相信号进行解码的正交解码器逻辑以及用于累计计数值的向上 / 向下计数器组成。输入端上的数字毛刺滤波器对输入信号进行滤波。图 16-2 为 QEI 模块的简化框图。

QEI 模块包括：

- 三个输入引脚，即两个相位信号和一个索引脉冲
- 输入端上的可编程数字噪声滤波器
- 提供计数器脉冲和计数方向的正交解码器
- 16 位向上 / 向下位置计数器
- 计数方向状态
- X2 和 X4 计数分辨率
- 两种位置计数器复位模式
- 通用 16 位定时器 / 计数器模式
- 由 QEI 或计数器事件产生的中断

图 16-2: 正交编码器接口模块的简化框图



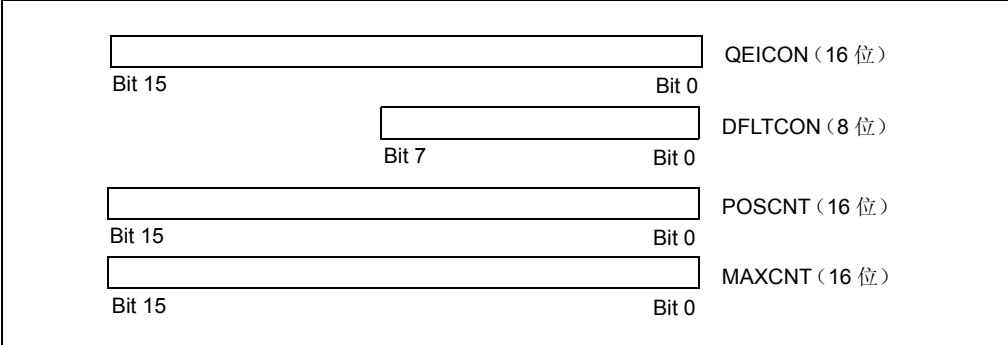
16.2 控制和状态寄存器

QEI 模块有四个用户可访问的寄存器。这些寄存器可以字节或字模式进行访问。图 16-3 中示出了这些寄存器，如下所述：

- 控制 / 状态寄存器（QEICON）：该寄存器允许对 QEI 操作和表示模块状态的状态标志进行控制。
- 数字滤波器控制寄存器（DFLTCON）：该寄存器允许对数字输入滤波器进行控制。
- 位置计数寄存器（POSCNT）：该单元允许读 / 写 16 位位置计数器。
- 最大计数寄存器（MAXCNT）：MAXCNT 寄存器用于保持某个值，在某些操作中，该值将与 POSCNT 寄存器的值进行比较。

注： POSCNT 寄存器允许以字节模式进行访问。然而，以字节模式读取时，该寄存器的值可能会在随后的读操作过程中部分更新。请使用字模式进行读 / 写操作或确保计数器在字节操作期间不计数。

图 16-3: QEI 编程模型



寄存器 16-1 和寄存器 16-4 定义了 QEI 模块控制和数字滤波器控制寄存器（QEICON 和 DFLTCON）。

寄存器 16-1: QEICON: QEI 控制寄存器

高字节:							
R/W-0	U-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
CNTERR	—	QEISIDL	INDEX	UPDN	QEIM<2:0>		
bit 15					bit 8		

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SWPAB	PCDOUT	TQGATE	TQCKPS<1:0>		POSRES	TQCS	UDSRC
bit 7				bit 0			

- bit 15 **CNTERR:** 计数错误状态标志位
1 = 发生了位置计数错误
0 = 未发生位置计数错误
(仅当 QEIM<2:0> = 110 或 100 时, CNTERR 标志位适用。)
- bit 14 **未用位:** 读作 0
- bit 13 **QEISIDL:** 空闲模式停止位
1 = 当器件进入空闲模式时, 模块不再继续工作
0 = 在空闲模式下, 模块继续工作
- bit 12 **INDEX:** 索引引脚状态位 (只读)
1 = 索引引脚为高电平
0 = 索引引脚为低电平
- bit 11 **UPDN:** 位置计数器方向状态位
1 = 位置计数器方向为正 (+)
0 = 位置计数器方向为负 (-)
(当 QEIM<2:0> = 1xx 时为只读位)
(当 QEIM<2:0> = 001 时为可读 / 写位)
- bit 10-8 **QEIM<2:0>:** 正交编码器接口模式选择位
111 = 正交编码器接口使能 (x4 模式), 通过与 (MAXCNT) 匹配将位置计数器复位
110 = 正交编码器接口使能 (x4 模式), 通过索引脉冲将位置计数器复位
101 = 正交编码器接口使能 (x2 模式), 通过与 (MAXCNT) 匹配将位置计数器复位
100 = 正交编码器接口使能 (x2 模式), 通过索引脉冲将位置计数器复位
011 = 未使用 (模块禁止)
010 = 未使用 (模块禁止)
001 = 启动 16 位定时器
000 = 正交编码器接口 / 定时器关闭
- bit 7 **SWPAB:** A 相和 B 相输入交换选择位
1 = A 相和 B 相输入已交换
0 = A 相和 B 相输入未交换
- bit 6 **PCDOUT:** 位置计数器方向状态输出使能位
1 = 位置计数器方向状态输出使能 (I/O 引脚的状态由 QEI 逻辑控制)
0 = 位置计数器方向状态输出禁止 (正常的 I/O 引脚操作)
- bit 5 **TQGATE:** 定时器门控时间累加使能位
1 = 定时器门控时间累加使能
0 = 定时器门控时间累加禁止
- bit 4-3 **TQCKPS<1:0>:** 定时器输入时钟预分频比选择位
11 = 预分频比是 1:256
10 = 预分频比是 1:64
01 = 预分频比是 1:8
00 = 预分频比是 1:1
(预分频器仅用于 16 位定时器模式)

寄存器 16-2: QEICON: QEI 控制寄存器（续）

- bit 2 **POSRES:** 位置计数器复位使能位
1 = 索引脉冲可使位置计数器复位
0 = 索引脉冲不能使位置计数器复位
（仅当 QEIM<2:0> = 100 或 110 时，该位适用。）
- bit 1 **TQCS:** 定时器时钟源选择位
1 = 来自 QEA 引脚（上升沿）的外部时钟
0 = 内部时钟（Tcy）
- bit 0 **UDSRC:** 位置计数器方向选择控制位
1 = QEB 引脚状态定义位置计数器方向
0 = 控制 / 状态位 UPDN（QEICON<11>）定义定时器计数器（POSCNT）方向
 注： 当配置为 QEI 模式时，此控制位是“无关位”。

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

寄存器 16-3: DFLTCON: 数字滤波器控制寄存器 (仅 dsPIC30F6010)

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	CEID
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
QEOUT	QECK<2:0>			INDOUT	INDCK<2:0>		
bit 7							bit 0

bit 15-9 未用: 读作 0

bit 8 **CEID:** 计数错误中断禁止位

1 = 禁止位置计数错误中断

0 = 使能位置计数错误中断

bit 7 **QEOUT:** QEA/QEB 数字滤波器输出使能位

1 = 数字滤波器输出使能

0 = 数字滤波器输出禁止 (正常的引脚操作)

bit 6-4 **QECK<2:0>:** QEA/QEB 数字滤波器时钟分频选择位

111 = 时钟分频比为 1:256

110 = 时钟分频比为 1:128

101 = 时钟分频比为 1:64

100 = 时钟分频比为 1:32

011 = 时钟分频比为 1:16

010 = 时钟分频比为 1:4

001 = 时钟分频比为 1:2

000 = 时钟分频比为 1:1

bit 3 **INDOUT:** 索引通道数字滤波器输出使能位

1 = 数字滤波器输出使能

0 = 数字滤波器输出禁止 (正常的引脚操作)

bit 2-0 **INDCK<2:0>:** 索引通道数字滤波器时钟分频选择位

111 = 时钟分频比为 1:256

110 = 时钟分频比为 1:128

101 = 时钟分频比为 1:64

100 = 时钟分频比为 1:32

011 = 时钟分频比为 1:16

010 = 时钟分频比为 1:4

001 = 时钟分频比为 1:2

000 = 时钟分频比为 1:1

注: 选用不同的 dsPIC30F 器件, DFLTCON 寄存器中可用的控制位可能会有所不同。详细信息, 请参见寄存器 16-2 和寄存器 16-3。

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = 上电复位时的值

1 = 置位

0 = 清零

x = 未知

dsPIC30F 系列参考手册

寄存器 16-4: DFLTCON: 数字滤波器控制寄存器 (除 dsPIC30F6010 之外的所有 dsPIC30F 器件)

高字节:							
U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	IMV<1:0>		CEID
bit 15					bit 8		

低字节:					
R/W-0	R/W-0	U-0	U-0	U-0	U-0
QEOUT	QECK<2:0>	—	—	—	—
bit 7		bit 0			

bit 15-11 未用: 读作 0

bit 10-9 **IMV<1:0>**: 索引匹配值——当接受到索引脉冲时, POSCNT 寄存器将被复位, 这些位允许用户指定 QEA 和 QEB 输入引脚在这个时候的状态。

在 4X 正交计数模式下:

IMV1 = 索引脉冲匹配所要求的 B 相输入信号的状态

IMV0 = 索引脉冲匹配所要求的 A 相输入信号的状态

在 2X 正交计数模式下:

IMV1 = 为索引状态匹配选择相位输入信号 (0 = A 相, 1 = B 相)

IMV0 = 索引脉冲匹配要求的所选相位输入信号的状态

bit 8 **CEID**: 计数错误中断禁止

1 = 禁止计数错误中断

0 = 使能计数错误中断

bit 7 **QEOUT**: QEA/QEB/INDX 引脚数字滤波器输出使能

1 = 数字滤波器输出使能

0 = 数字滤波器输出禁止 (正常的引脚操作)

bit 6-4 **QECK<2:0>**: QEA/QEB/INDX 数字滤波器时钟分频选择位

111 = 时钟分频比为 1:256

110 = 时钟分频比为 1:128

101 = 时钟分频比为 1:64

100 = 时钟分频比为 1:32

011 = 时钟分频比为 1:16

010 = 时钟分频比为 1:4

001 = 时钟分频比为 1:2

000 = 时钟分频比为 1:1

bit 3-0 未用: 读作 0

注: 选用不同的 dsPIC30F 器件, DFLTCON 寄存器中可用的控制位可能会有所不同。详细信息请参见寄存器 16-2 和寄存器 16-3。

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

y = 在 POR 或 BOR 时由配置位设置的值

16.3 可编程数字噪声滤波器

数字噪声滤波器部件用于滤除输入索引脉冲和正交信号中的噪声。施密特触发器输入和有三个时钟周期延时的滤波器组合使用，用以滤除低电平噪声和通常在易产生噪声的应用（如电机系统应用）中出现的幅度大而短时间持续的尖脉冲噪声。

该滤波器可以确保在三个连续的滤波器周期内都获得同一个稳定值之后，才允许经过滤波的输出信号发生变化。

滤波器时钟的速率决定滤波器的低通带。滤波器时钟越慢，将导致通带滤除的频率越低。滤波器时钟由器件的 F_{CY} 时钟经过可编程分频器分频后得到的。

将 QEOUT 位 (DFLTCON<7>) 置 1，则使能通道 QEA 和 QEB 的滤波器。QECK<2:0> 的位 (DFLTCON<6:4>) 用于指定通道 QEA 和 QEB 所使用的滤波器时钟分频器。将 INDOUT 位 (DFLTCON<3>) 置 1，将使能索引通道的滤波器。INDCK<2:0> 的位 (DFLTCON<2:0>) 用来指定索引通道所使用的滤波器时钟分频器。复位时，所有通道的滤波器被禁止。

有些器件对于 QEx 输入数字滤波器和 INDX 输入数字滤波器没有单独的控制位。对于这些器件，QEOUT 和 QECK<2:0> 控制位设置 QEA/QEB 和 INDX 引脚的数字滤波器特性。更多信息，请参见寄存器 16-2 和寄存器 16-3。

图 16-4 给出了数字噪声滤波器的简化框图。

图 16-4: 简化的数字噪声滤波器框图

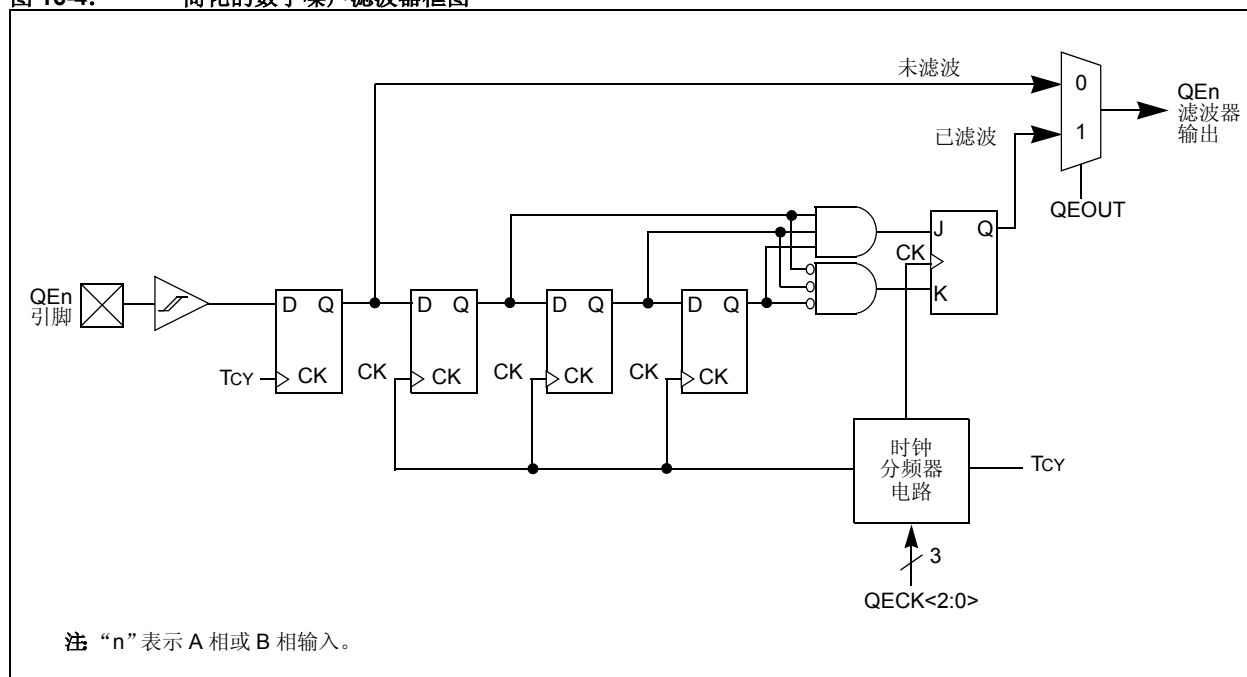
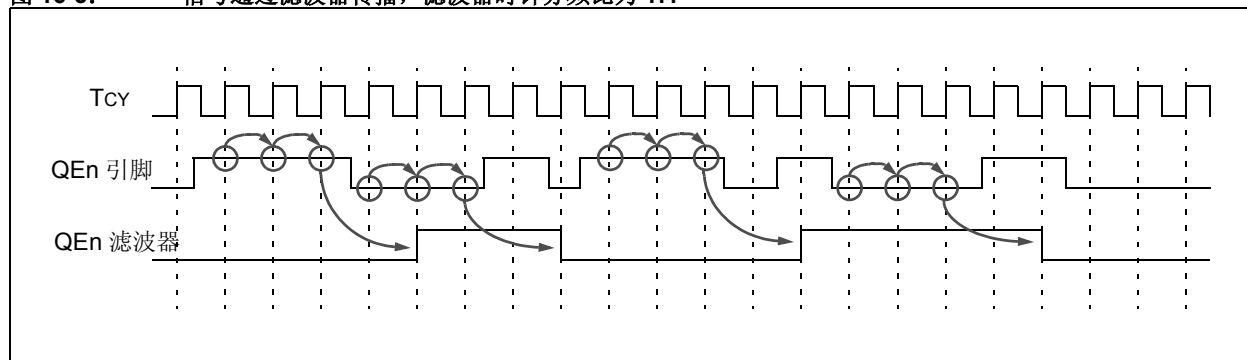


图 16-5: 信号通过滤波器传播，滤波器时钟分频比为 1:1



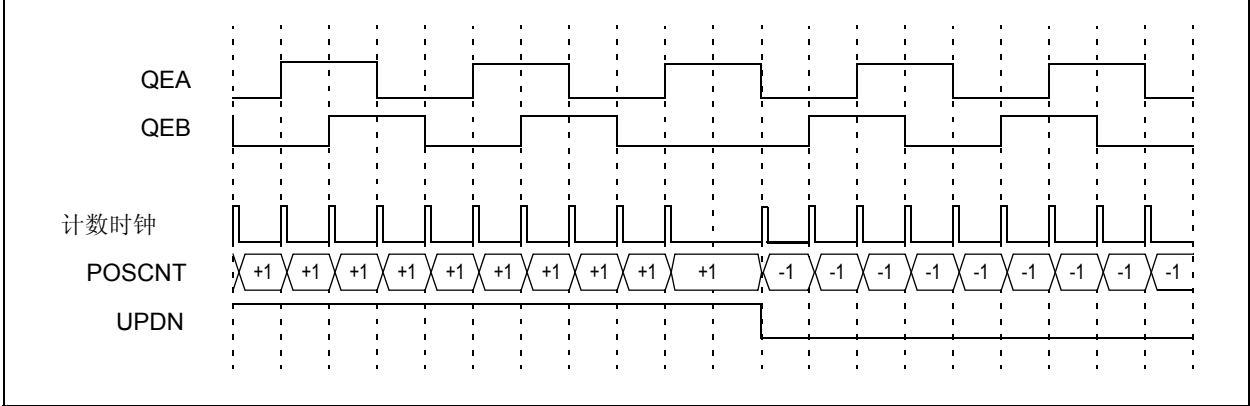
16.4 正交解码器

当 $QEIM2 = 1$ ($QEICON<10>$) 时，选择位置测量模式。

当 $QEIM1 = 1$ ($QEICON<9>$) 时，选择 “x4” 测量模式，且 QEI 逻辑在 A 相和 B 相输入信号的上升沿和下降沿都使位置计数器计数。

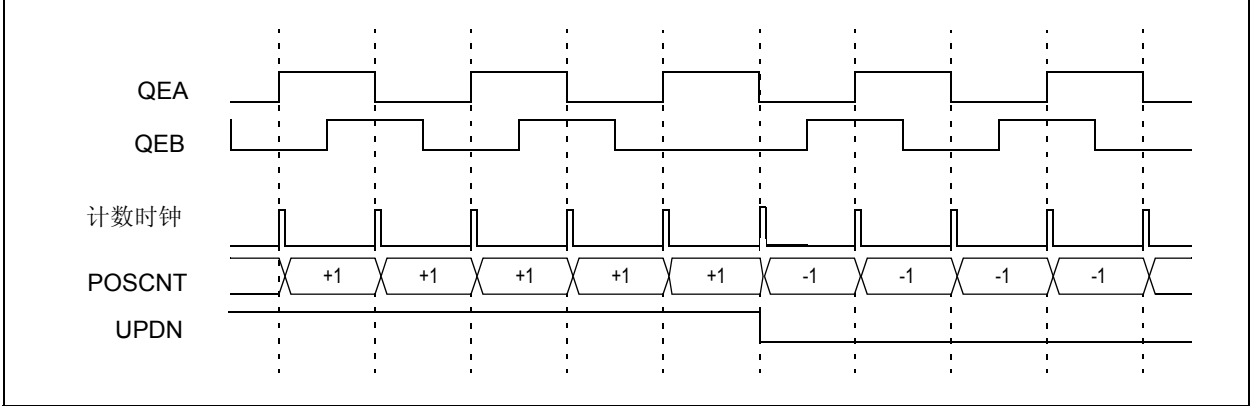
“x4” 测量模式可以为确定编码器位置提供更高精度的数据（更多位置计数）。

图 16-6: 4X 模式的正交解码器信号



当 $QEIM1 = 0$ 时，选择 “x2” 测量模式，且 QEI 逻辑仅根据 A 相的输入的上升沿和下降沿来确定位置计数器递增计数的速率。A 相信号的每个上升沿和下降沿都会使位置计数器进行递增或递减计数。B 相信号仍用于确定计数器的方向，与 x4 测量模式中完全一样。

图 16-7: 2X 模式的正交解码器信号



16.4.1 超前 / 滞后测试说明

由正交解码器逻辑执行超前 / 滞后测试，用来确定 QEA 和 QEB 信号的相位关系，从而确定将递增还是递减 POSCNT 寄存器。表 16-1 中对超前 / 滞后测试进行了说明。

表 16-1: 超前 / 滞后测试的说明

当前转换	前一次转换	条件	操作	
QEA ↑	QEB ↓	QEA 超前于 QEB 通道	UPDN 置 1	POSCNT 递增
	QEB ↑	QEA 滞后于 QEB 通道	UPDN 清零	POSCNT 递减
	QEA ↓	方向变化	UPDN 翻转	POSCNT 递增或递减
QEA ↓	QEB ↓	QEA 滞后于 QEB 通道	UPDN 清零	POSCNT 递减
	QEB ↑	QEA 超前于 QEB 通道	UPDN 置 1	POSCNT 递增
	QEA ↑	方向变化	UPDN 翻转	POSCNT 递增或递减
QEB ↑	QEA ↓	QEA 滞后于 QEB 通道	UPDN 清零	POSCNT 递减
	QEA ↑	QEA 超前于 QEB 通道	UPDN 置 1	POSCNT 递增
	QEB ↓	方向变化	UPDN 翻转	POSCNT 递增或递减
QEB ↓	QEA ↓	QEA 超前于 QEB 通道	UPDN 置 1	POSCNT 递增
	QEA ↑	QEA 滞后于 QEB 通道	UPDN 清零	POSCNT 递减
	QEB ↑	方向变化	UPDN 翻转	POSCNT 递增或递减

16.4.2 计数方向状态

如前面章节所述，QEI 逻辑根据 A 相与 B 相的时间关系产生 UPDN 信号。UPDN 信号可以在 I/O 引脚上输出。

将 PCDOUT 位 (QEICON<6>) 置 1，并将与该引脚相关的相应 TRIS 位清零，将使 UPDN 信号输出到引脚。

除输出引脚外，内部 UPDN 信号的状态被提供给 SFR 位 QEICON<11>，该位作为只读位，用 UPDN 表示。

16.4.3 编码器计数方向

正交计数的方向由 SWPAB 位 (QEICON<7>) 决定。如果 SWPAB = 0，A 相的输入信号送到正交计数器的 A 输入，而 B 相的输入信号则送到正交计数器的 B 输入。所以，当 A 相的信号超前于 B 相的信号时，正交计数器在每个边沿都递增。这种情况 (A 信号超前于 B 信号) 被定义为运动的正方向。将 SWPAB 位 (QEICON<7>) 置为逻辑 1，使 A 相输入送到正交计数器的 B 输入，而 B 相信号则送到正交计数器的 A 输入。因此如果 dsPIC30F 器件引脚上的 A 相信号超前于 B 相信号，那么正交计数器的 A 相输入将落后于 B 相输入。这种情况下，则为反方向旋转，且计数器将在正交脉冲的每个边沿递减。

16.4.4 正交速率

位置控制系统的 RPM 有所不同。RPM 与正交解码器线计数（Line Count）一起决定 QEA 和 QEB 输入信号的频率。可以将正交解码器信号进行解码，从而在每个正交信号的边沿产生计数脉冲。这样就允许角度位置测量的精度最高为解码器线计数的四倍。例如：一个 6,000 RPM 的电机使用 4096 线解码器产生的正交计数速率为： $((6000/60) \times (4096 \times 4)) = 1.6384 \text{ MHz}$ 。同样地，一个使用 8,192 线解码器而转速为 10,000 RPM 的电机产生的正交计数速率为： $((10000/60) \times (8192 \times 4)) = 5.46 \text{ MHz}$ 。

QE1 允许正交频率最大为 $F_{CY}/3$ 。例如，如果 $F_{CY} = 30 \text{ MHz}$ ，QEA 和 QEB 信号的最大频率可达到 10 MHz。更多详细信息，请参见器件数据手册的“电气规范”部分。

16.5 16 位向上 / 向下位置计数器

16 位向上 / 向下计数器对正交解码器逻辑产生的每个计数脉冲进行向上或向下计数。此时计数器充当积分器，其计数值与位置成比例。计数的方向由正交解码器决定。

用户软件可以通过读取 POSCNT 寄存器来检查计数的内容。用户软件还可以通过写 POSCNT 寄存器来初始化计数。

改变 QEIM 位不影响位置计数器寄存器的内容。

16.5.1 位置计数器的使用

系统可以用几种方法之一来使用位置计数器的数据。在某些系统中，位置计数一直累加，并作为一个代表该系统总位置的绝对值。举一个典型的示例，假设正交解码器固定在一台用来控制打印机的打印头的电机上，工作中，初始化系统时，将打印头移动到最左位置并复位 POSCNT 寄存器。当打印头往右移动时，正交解码器将在 POSCNT 寄存器中开始累加计数。当打印头往左移动时，累加计数将递减。当打印头到达最右位置时，应该达到最大位置计数。如果最大计数小于 2^{16} ，就可以用 QE1 模块为整个运动范围编码。

然而，如果最大计数大于 2^{16} ，必须由用户软件获取额外的计数精度。通常，要完成这个工作，该模块应被设置为在最大计数匹配发生时复位的计数器模式。当 QEIM0 = 1 时，可使用 MAXCNT 寄存器将位置计数器复位的模式。当计数器在递增计数时达到预先确定的最大值，或当它在递减计数时达到 0 时，将复位计数并产生中断，以允许用户软件对包含此位置计数的最高有效位的软件计数器进行递增或递减计数。最大计数可达到 0xFFFF，可以允许 QE1 计数器和软件计数器的完整范围，也可以是一些较小的值，比如解码器旋转一次的计数数字。

在其他系统中，位置计数可能是循环的。在由索引脉冲决定的转数范围之内，位置计数仅用作轮子位置的基准。例如，用螺旋杆移动的工具平台使用了固定在螺旋杆上的正交编码器。工作时，螺旋杆可能需要 5.5 转来到达所要求的位置。用户软件将检测 5 个索引脉冲来为整数转数计数，并且使用位置计数来测量剩下的半转。采用这种方法，每次旋转索引脉冲都复位位置计数器，来初始化计数器并为每次旋转产生一次中断。QEIM0 = 0 可以使能这些模式。

16.5.2 使用 MAXCNT 复位位置计数器

当 QEIM0 位为 1 时，在位置计数与预先确定的高、低值匹配时，位置计数器将会复位。不使用索引脉冲复位机制。

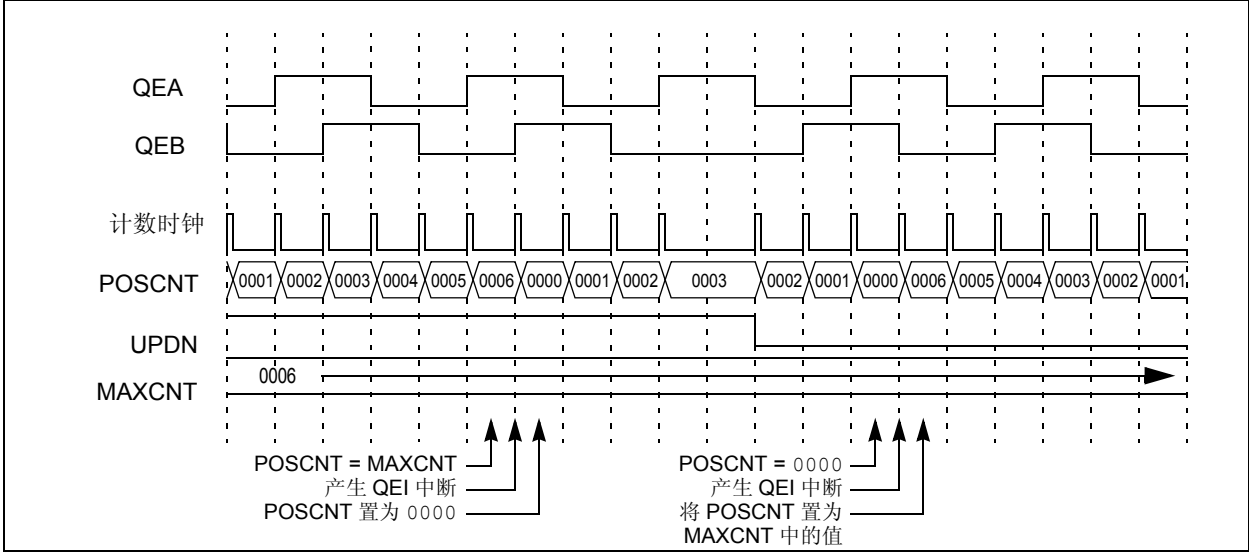
对于该模式，位置计数器的复位机制以如下方式工作：（相关时序的详细信息，参见图 16-8）。

- 如果编码器正向旋转，例如，QEA 超前于 QEB，而且 POSCNT 寄存器中的值与 MAXCNT 寄存器中的值匹配时，POSCNT 将在下一个递增 POSCNT 的正交脉冲沿时复位为 0。在此计满返回事件发生时将产生中断。
- 如果编码器反向旋转，例如，QEB 超前于 QEA，而且 POSCNT 寄存器向下计数至“0”，那么在下一个递减 POSCNT 的正交脉冲沿 MAXCNT 寄存器中的值会被装入 POSCNT。在此下溢事件发生时将产生中断。

当把 MAXCNT 用作位置极限时，请记住位置计数器将以 2X 或 4X 编码器计数模式计数。对于标准的旋转编码器，写入 MAXCNT 的适当值应该是 $4N - 1$ （4x 位置模式）和 $2N - 1$ （2x 位置模式），其中 N 为解码器每转一圈的计数数字。

对于系统范围超出 2^{16} 的绝对位置信息，将值 0xFFFF 装入 MAXCNT 寄存器也是合适的。该模块在计数器发生计满返回或下溢时将会产生中断。

图 16-8: 计满返回 / 下溢复位和向上 / 向下位置计数器

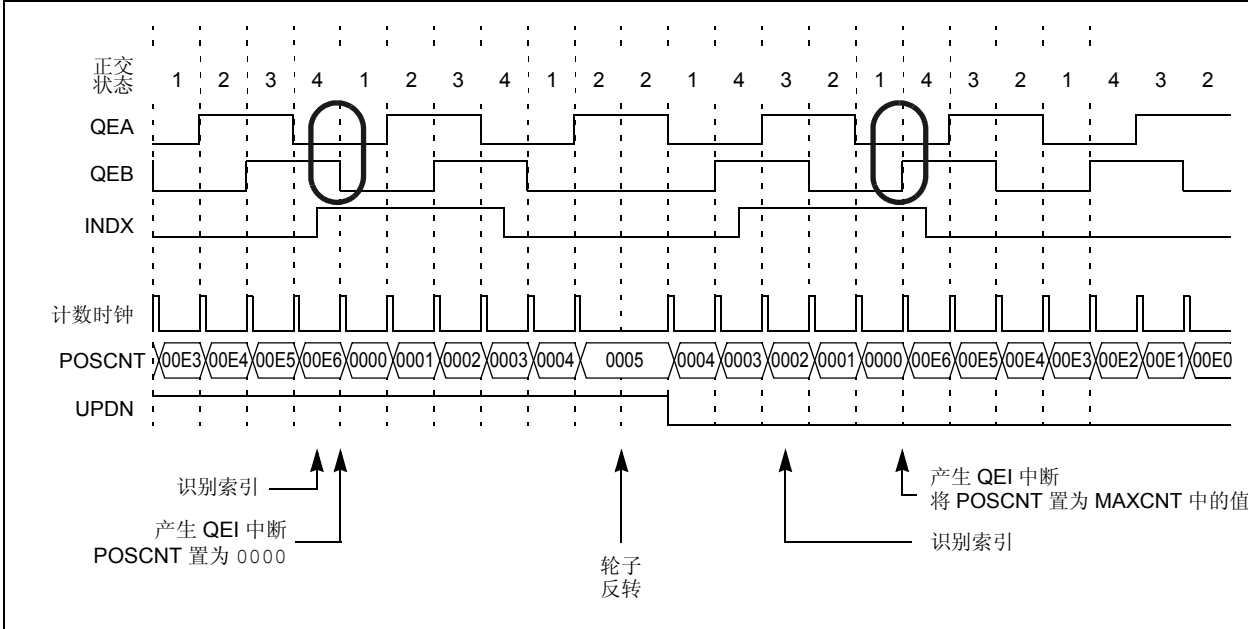


16.5.3 使用索引复位位置计数器

当 $QEIM<0> = 0$ 时，使用索引脉冲复位位置计数器。对于该模式，位置计数器的复位机制以如下方式工作：（相关时序的详细信息，参见图 16-9）。

- 当每次在 INDEX 引脚上接收到索引脉冲时，位置计数复位。
- 如果编码器正向旋转，例如，QEA 超前于 QEB，POSCNT 将复位为“0”。
- 如果编码器反向旋转，例如，QEB 超前于 QEA，MAXCNT 寄存器中的值会被装入 POSCNT。

图 16-9: 索引复位模式——向上 / 向下位置计数器



16.5.3.1 索引脉冲检测标准

来自不同厂商的增量式编码器为索引脉冲使用不同的时序。索引脉冲可以与四种正交状态的任何一种对齐，且脉宽可以是整个周期（四种正交状态）、半个周期（两种正交状态）或四分之一周期（一种正交状态）。脉宽为整个周期或半个周期的索引脉冲通常称为“非门控的”，而脉宽为四分之一周期的索引脉冲通常称为“门控的”。

无论提供的索引脉冲是何种类型，当轮子反向旋转时，QEI 保持计数对称。这意味着轮子以正或反方向旋转时，索引脉冲必须在出现相同的相对正交状态转换时，将位置计数器复位。

例如，在图 16-9 中，第一个索引脉冲被识别到并且当正交状态从 4 转换为 1（如图中标示）时，复位 POSCNT。QEI 锁存这个转换的状态。随后的任何索引脉冲检测将使用这个状态转换来复位。

当轮子反向旋转时，再次产生索引脉冲，然而直到正交状态从 1 转换到 4 时，位置计数器（在图中标示）才发生复位。

注： QEI 索引逻辑确保 POSCNT 寄存器总是在相对于索引脉冲的同一个位置被调整，不考虑旋转的方向。

16.5.3.2 IMV 控制位

IMV<2:0> 控制位在某些带有 QEI 模块的 dsPIC 器件上可用 (参见寄存器 16-3)。这些控制位允许用户选择发生索引脉冲复位所要求的 QEA 和 QEB 信号的状态。

没有这些控制位的器件将在第一次产生索引脉冲期间, 自动选择 QEA 和 QEB 的状态。

16.5.3.3 索引脉冲状态

INDEX 位 (QEICON<12>) 提供了索引引脚上的逻辑状态。当位置控制系统在“自引导 (Homming)”序列中搜索基准位置时, 此状态位很有用。如果该索引位使能, 它表示索引引脚在经过数字滤波器处理之后的状态。

16.5.3.4 使用索引引脚和 MAXCNT 检查错误

当计数器在索引脉冲复位模式下工作时, QEI 还将检测 POSCNT 寄存器的边界条件。在增量式解码器系统中, 可以用来检测系统错误。

例如, 假设轮子编码器有 100 线。在 x4 测量模式下使用并在产生索引脉冲时复位, 计数器应从 0 开始计数到 399 (0x018E) 并复位。如果 POSCNT 寄存器在任何时候达到值 0xFFFF 或 0x0190, 那么已经发生了某种系统错误。

如果向上计数, POSCNT 寄存器的内容将与 MAXCNT+1 作比较; 如果向下计数, 将与 0xFFFF 作比较。如果 QEI 检测到这些值中的一个, 将通过置位 CNTERR 位 (QEICON<15>) 产生一个位置计数错误条件, 而且可以选择产生 QEI 中断。

如果 CEID 控制位 (DFLTCON<8>) 清零 (缺省), 当检测到位置计数错误时, 将产生 QEI 中断。如果 CEID 控制位置 1, 则不产生中断。

在检测到位置计数错误后, 位置计数器继续对编码器的边沿计数。随后的位置计数错误事件将不再产生中断, 直到 CNTERR 被用户清零为止。

16.5.3.5 位置计数器复位使能

当检测到索引脉冲时, 位置计数器复位使能位 POSRES (QEICON<2>) 将使能位置计数器的复位。仅当 QEI 模块被配置为 QEIM<2:0> = 100 或 110 模式时, 此位适用。

如果 POSRES 位置为逻辑 “1”, 那么当检测到索引脉冲时, 位置计数器会如本节所述的那样复位。

如果 POSRES 位置为逻辑 “0”, 那么当检测到索引脉冲时, 位置计数器不复位。位置计数器将继续向上或向下计数, 并在计满返回或下溢情况发生时复位。QEI 继续在检测到索引脉冲时产生中断。

16.6 QEI 用作备用 16 位定时器 / 计数器

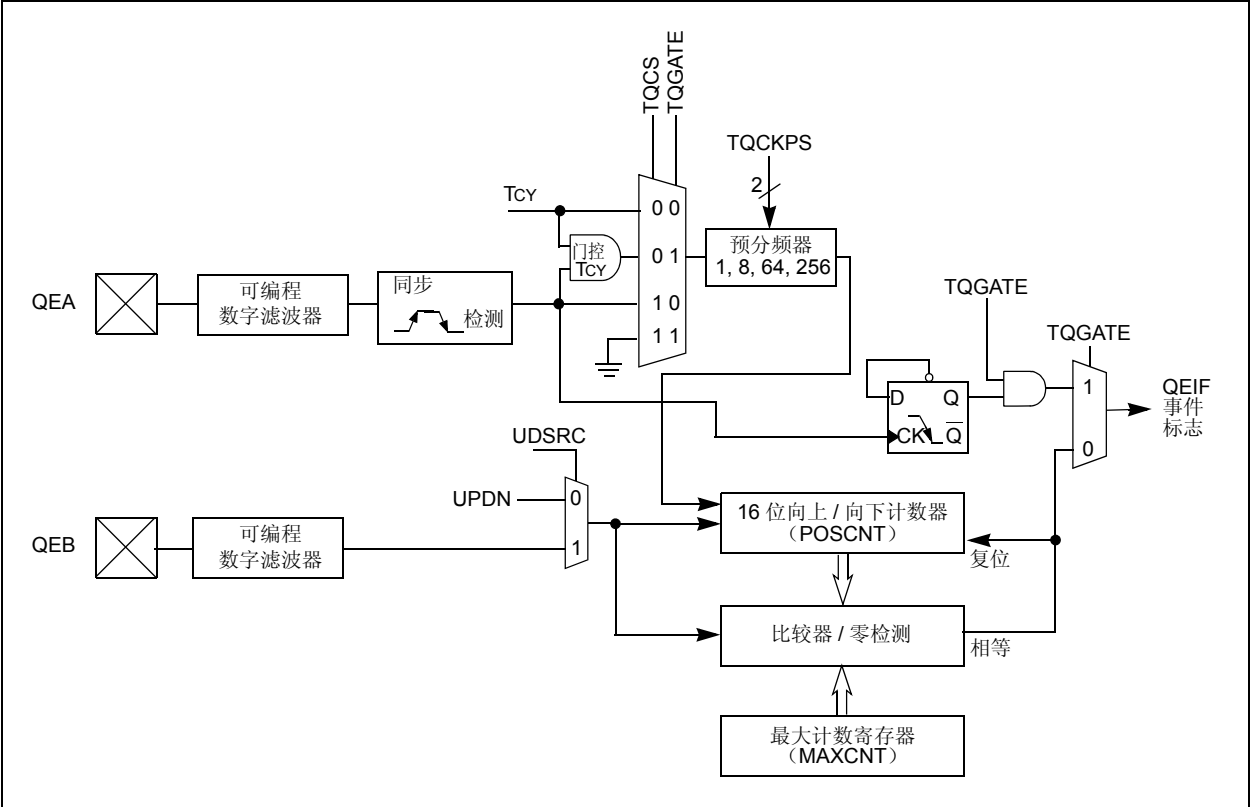
当 QEI 模块被配置为 $QEIM<2:0> = 001$ 时, QEI 功能被禁止, QEI 模块被配置为 16 位定时器 / 计数器。辅助定时器的设置和控制通过 QEICON 寄存器实现。

QEI 定时器的工作方式与其他 dsPIC30F 定时器类似。关于定时器的全面讨论, 请参见第 12 章 “定时器”。

当配置为定时器时, POSCNT 寄存器充当定时器寄存器, 与 GP 定时器的 TMRn 寄存器类似。MAXCNT 寄存器充当周期寄存器, 与 GP 定时器的 PRn 寄存器类似。当定时器 / 周期寄存器发生匹配时, QEIF 标志位置 1。

注: 改变工作模式时, 也就是说, 从 QEI 到定时器或从定时器到 QEI, 不影响定时器 / 位置计数寄存器的内容。

图 16-10: QEI 用作定时器 / 计数器的框图



16.6.1 向上 / 向下定时器的的工作

QEI 定时器可以递增或递减计数。这是其他大部分定时器所不具备的独特功能。

当定时器被配置为向上计数时，定时器 (POSCNT) 将进行递增计数直到计数与周期寄存器 (MAXCNT) 匹配为止。定时器复位为 0 并重新开始递增计数。

当定时器被配置为向下计数时，定时器 (POSCNT) 将进行递减计数直到计数与周期寄存器 (MAXCNT) 匹配为止。定时器复位为 0 并重新开始递减计数。

当定时器被配置为向下计数时，为了正确工作，必须遵循一些总体的工作指导方针。

1. MAXCNT 寄存器将用作周期匹配寄存器，但是因为计数器在进行递减计数，所需的匹配值是 2 倍计数。例如，要对 0x1000 个时钟计数，该周期寄存器必须装入 0xF000。
2. 在匹配条件发生时，定时器复位为 0。

一个 I/O 引脚或 SFR 控制位用于指定计数方向控制。

控制位 UDSRC (QEICON<0>) 决定由谁控制定时器计数方向状态。

当 UDSRC = 1 时，定时器计数方向由 QEB 引脚控制。如果 QEB 引脚为 “1”，计数方向为递增。如果 QEB 引脚为 “0”，计数方向为递减。

当 UDSRC = 0 时，定时器计数方向由 UPDN 位 (QEICON<11>) 控制。当 UPDN = 1 时，定时器递增计数。当 UPDN = 0 时，定时器递减计数。

16.6.2 定时器外部时钟

TQCS 位 (QEICON<1>) 选择内部或外部时钟。当 TQCS 置位时，QEI 定时器可以将 QEA 引脚用作外部时钟输入。QEI 定时器不支持外部异步计数器模式。如果使用外部时钟源，时钟将自动与内部指令周期 (Tcy) 同步。

16.6.3 定时器门控操作

当 TQGATE 位 (QEICON<5>) 置 1 且 TQCS 清零时，QEA 引脚作为定时器的门控。

如果 TQCS 和 TQGATE 同时置位，定时器不进行递增计数且不产生中断。

16.7 正交编码器接口中断

根据 QEI 的模式，QEI 在下列事件发生时将产生中断：

- 工作在匹配复位模式下 (QEIM<2:0> = 111 和 101) 时，在位置计数器发生计满返回 / 下溢时产生中断。
- 当工作在索引脉冲复位模式下 (QEIM<2:0> = 110 和 100) 时，在检测到索引脉冲并选择将 CNTERR 置 1 时，产生中断。
- 当作为定时器 / 计数器工作 (QEIM<2:0> = 001) 时，发生周期匹配事件，或当 TQGATE = 1 时，定时器门控信号出现下降沿时，产生中断。

当 QEI 中断事件发生时，QEIIF 位 (IFS2<8>) 被置 1，若允许中断，将产生中断。QEIIF 位必须用软件清零。

QEI 中断的允许通过对应的使能位 QEIIE (IEC2<8>) 实现。

16.8 I/O 引脚控制

使能 QEI 模块可以使相关的 I/O 引脚受 QEI 的控制，并防止较低优先级的 I/O 功能（如端口功能）影响 I/O 引脚。

根据由 QEIM<2:0>和其他控制位指定的模式，I/O 引脚可以实现不同的功能，如表 16-2 和表 16-3 中所示。

表 16-2: 正交编码器模块 I/O 引脚配置描述

引脚名称	引脚类型	缓冲器类型	描述
QEA	I	ST	正交编码器 A 相输入，或 辅助定时器外部时钟输入或 辅助定时器外部门控输入
	I	ST	
	I	ST	
QEB	I	ST	正交编码器 B 相输入，或 辅助定时器向上 / 向下选择输入
	I	ST	
INDX	I	ST	正交编码器索引脉冲输入
UPDN	O		位置向上 / 向下计数器方向状态， QEI 模式

图注：I = 输入，O = 输出，ST = 施密特触发器

表 16-3: 模块 I/O 模式功能

QEIM<2:0>	PCDOUT	UDSRC	TQGATE	TQCS	QEA 引脚	QEB 引脚	INDX 引脚	UPDN 引脚
000,010,011 模块关闭	N/A	N/A	N/A	N/A				
001 定时器模式	N/A	0	0	0				
		1	0	0		输入 (UPDN)		
		0	1	0	输入 (TQGATE) 端口未禁止			
		1	1	0	输入 (TQGATE) 端口未禁止	输入 (UPDN)		
		0	N/A	1	输入 (TQCKI) 端口未禁止			
		1	N/A	1	输入 (TQCKI) 端口未禁止	输入 (UPDN)		
101,111 QEI 由计数复位	0	N/A	N/A	N/A	输入 (QEA)	输入 (QEB)		
	1	N/A	N/A	N/A	输入 (QEA)	输入 (QEB)		输出 (UPDN)
100,110 QEI 由索引脉冲复位	0	N/A	N/A	N/A	输入 (QEA)	输入 (QEB)	输入 (INDX)	
	1	N/A	N/A	N/A	输入 (QEA)	输入 (QEB)	输入 (INDX)	输出 (UPDN)

注：空格表示引脚在这种配置下未被 QEI 使用，引脚由 I/O 端口逻辑控制。

16.9 低功耗模式下的 QEI 工作

16.9.1 当器件进入休眠模式时

当器件进入休眠模式时，QEI 将停止所有工作。POSCNT 将停止在当前值。QEI 将不会响应 QEA、QEB、INDX 或 UPDN 引脚上的有效信号。

QEICON 寄存器将保持不变。

如果 QEI 被配置为定时器 / 计数器 (QEIM<2:0> = 001)，而且由外部提供时钟 (TQCS = 1)，该模块在休眠模式下也将停止工作。

当该模块被唤醒时，正交编码器将接收 QEA 或 QEB 信号的下一次状态转换，并将其与休眠前的上一次转换作比较，以决定下一步的操作。

16.9.2 当器件进入空闲模式时

在空闲模式下，该模块是否会进入低功耗模式取决于 QEISIDL 位 (QEICON<13>)。

如果 QEISIDL = 1，那么该模块将进入低功耗模式，与进入休眠模式的操作类似。

如果 QEISIDL = 0，那么该模块将不进入低功耗模式。当器件处于空闲模式时，该模块将继续正常工作。

16.10 复位的影响

复位强制所有的模块寄存器进入它们的初始复位状态。与 QEI 模块相关的寄存器的所有初始化和复位条件，参见寄存器 16-1。

正交编码器和 POSCNT 计数器复位为初始状态。

表 16-4: 与 QEI 相关的特殊功能寄存器

名称	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的值
QEICON	CNTERR	未使用	QEISIDL	INDX	UPDN	QEIM2	QEIM1	QEIM0	SWPAB	PCDOUT	TQGATE	TQCKPS1	TQCKPS0	POSRES	TQCS	UDSRC	0000 0000 0000 0000
DFLTCON	—	—	—	—	—	—	—	—	QEOUT	QECK2	QECK1	QECK0	INDOUT	INDCK2	INDCK1	INDCK0	---- ---- ----
POSCNT	位置计数寄存器																0000 0000 0000 0000
MAXCNT	最大计数寄存器																1111 1111 1111 1111
ADPCFG	PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8	PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000 0000 0000 0000
INTCON1	NSTDIS	—	—	—	—	OVATE	OVBT	COVTE	—	—	—	MATHERR	ADDRERR	STKERR	OSCFail	—	0000 0000 0000 0000
INTCON2	ALTIVT	—	—	—	—	—	—	—	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP	0000 0000 0000 0000
IFS2	—	—	—	FLTBIF	FLTAIF	LVDIF	DCIIF	QEIIIF	PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF	0000 0000 0000 0000
IEC2	—	—	—	FLTBIE	FLTAIE	LVDIE	DCIIE	QEIIIE	PWMIIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE	0000 0000 0000 0000
IPC10	—	FLTAIP<2:0>			—	LVDIP<2:0>			—	DCIIP<2:0>			—	QEIIIP<2:0>			0100 0100 0100 0100

注: 所用 dsPIC30F 器件不同, DFLTCON 寄存器中可用的控制位可能也会有所不同。详细信息, 请参见寄存器 16-2 和寄存器 16-3。

注: 在很多器件上, QEI 引脚与模拟输入引脚复用。必须确保使用 ADPCFG 控制寄存器将 QEI 引脚配置为数字引脚。

16.11 设计技巧

问 1: **QEI 已初始化, 但当 QEA/QEB 引脚加上正交信号时, POSCNT 寄存器似乎不变化。**

答: 对于很多器件, QEI 引脚与模拟输入引脚复用。必须确保使用 ADPCFG 控制寄存器将 QEI 引脚配置为数字引脚。

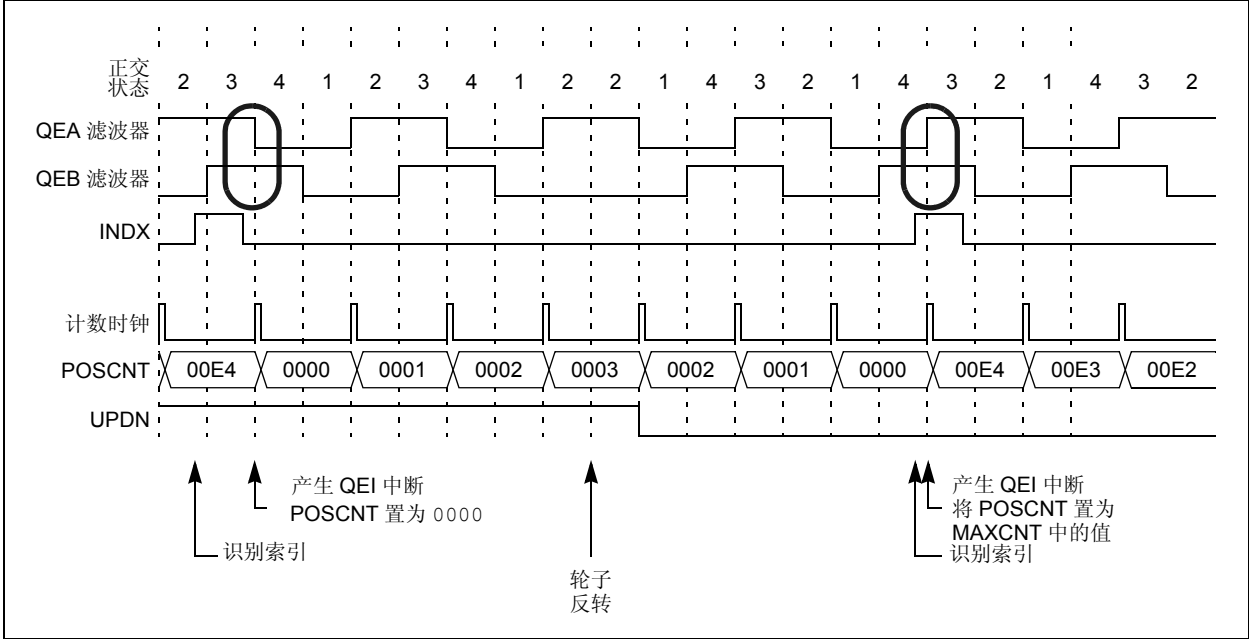
问 2: **正交信号的频率可以有多快?**

答: 这取决于正交信号的滤波器参数设置。不使用滤波器时, QEI 要求正交信号频率必须低于 $F_{CY}/3$; 而使用滤波器时, 要求正交信号频率必须低于滤波器频率的 $1/6$ 。

问 3: **编码器出现一个 90° 索引脉冲, 但计数不能正确复位。**

答: 这取决于计数时钟如何产生, 以及索引脉冲使用哪一种正交状态转换, $1/4$ 周期的索引脉冲可能在所需的转换之前无法被识别。要解决这个问题, 请在正交时钟上使用一个滤波器, 且要求该滤波器的预分频比高于索引脉冲滤波器的预分频比。这将在某种程度上延迟正交时钟, 以便正确检测索引脉冲。

图 16-11: 索引模式复位 (90° 索引脉冲) —— 向上 / 向下位置计数器



16.12 相关应用笔记

本节列出了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相关的，通过适当修改即可使用，但使用中可能会受到一定限制。当前与正交编码器接口（QEI）模块相关的应用笔记有：

标题	应用笔记编号
Servo Control of a DC-Brush Motor	AN532
PIC18CXXX/PIC16CXXX DC Servomotor	AN696
使用 dsPIC30F 实现交流感应电机的矢量控制	AN908

注：如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

16.13 版本历史

版本 A

本文档的初始发行版。

版本 B

此版本提供了有关 dsPIC30F 正交编码器接口 (QEI) 模块的扩展信息。

版本 C

此版本加入了本文档更新时已知的所有勘误。

注:

第 17 章 10 位 A/D 转换器

目录

本章包括下列主题：

17.1	简介	17-2
17.2	控制寄存器	17-4
17.3	A/D 转换结果缓冲器	17-4
17.4	A/D 转换术语和转换过程	17-11
17.5	A/D 模块配置	17-13
17.6	参考电压源的选择	17-13
17.7	A/D 转换时钟的选择	17-13
17.8	采样模拟输入的选择	17-14
17.9	模块使能	17-16
17.10	采样 / 转换过程的说明	17-16
17.11	如何开始采样	17-17
17.12	如何停止采样和开始转换	17-18
17.13	采样 / 转换工作的控制	17-29
17.14	如何将转换结果写入缓冲器的说明	17-30
17.15	转换过程示例	17-31
17.16	A/D 采样要求	17-45
17.17	读取 A/D 转换结果缓冲器	17-46
17.18	传递函数	17-47
17.19	A/D 转换的精度 / 误差	17-47
17.20	连接注意事项	17-47
17.21	初始化	17-48
17.22	A/D 转换速度	17-49
17.23	在休眠和空闲模式下工作	17-55
17.24	复位的影响	17-55
17.25	与 10 位 A/D 转换器相关的特殊功能寄存器	17-56
17.26	设计技巧	17-57
17.27	相关应用笔记	17-58
17.28	版本历史	17-59

17.1 简介

dsPIC30F 10 位 A/D 转换器具有以下主要特点：

- 逐次逼近寄存器（Successive Approximation Register, SAR）转换
- 最大 1 Msps 的转换速度
- 最多 16 个模拟输入引脚
- 外部参考电压输入引脚
- 四个单极性差分采样保持（S/H）放大器
- 对多达四个模拟输入引脚进行同时采样
- 自动通道扫描模式
- 可选转换触发源
- 16 字转换结果缓冲器
- 可选缓冲器填充模式
- 四种结果对齐选择
- 在 CPU 休眠和空闲模式下运行

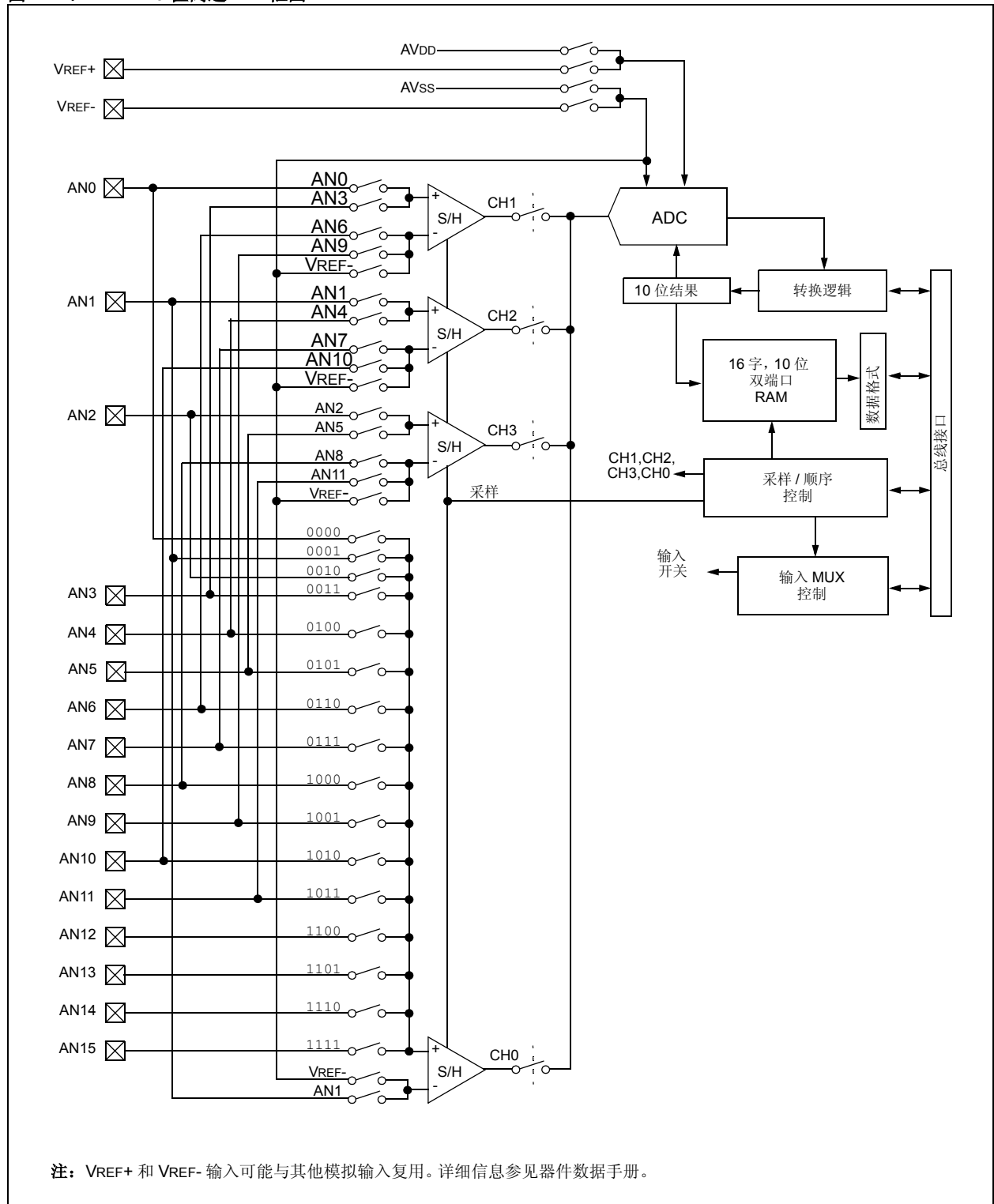
图 17-1 给出了 10 位 A/D 的框图。10 位 A/D 转换器最多可以有 16 个模拟输入引脚，指定为 AN0—AN15。此外，有两个可用于外部参考电压连接的模拟输入引脚。这些参考电压输入可以和其他模拟输入引脚复用。模拟输入引脚的实际数量和外部参考电压输入配置取决于具体的 dsPIC30F 器件。更多详细信息请参见器件数据手册。

模拟输入是通过多路开关连接到四个 S/H 放大器的，指定为 CH0—CH3。可以为采集输入数据使能一个、两个或四个 S/H 放大器。在转换期间，模拟输入多路开关可以在两组模拟输入之间切换。使用某些输入引脚可以在所有通道上实现单极性差分转换（参见图 17-1）。

可以为 CH0 S/H 放大器使能模拟输入扫描模式。控制寄存器用来指定在扫描过程中要包含的模拟输入通道。

10 位 A/D 与 16 字转换结果缓冲器相连。在从缓冲器读取每个 10 位 A/D 转换结果时，转换结果被转换为四种 16 位输出格式之一。

图 17-1: 10 位高速 A/D 框图



17.2 控制寄存器

A/D 模块有六个控制和状态寄存器。这些寄存器为：

- ADCON1: A/D 控制寄存器 1
- ADCON2: A/D 控制寄存器 2
- ADCON3: A/D 控制寄存器 3
- ADCHS: A/D 输入通道选择寄存器
- ADPCFG: A/D 端口配置寄存器
- ADCSSL: A/D 输入扫描选择寄存器

ADCON1、ADCON2 和 ADCON3 寄存器用来控制 A/D 模块的工作。ADCHS 寄存器用于选择连接到 S/H 放大器的输入引脚。ADPCFG 寄存器用于将模拟输入引脚配置为模拟输入或数字 I/O。ADCSSL 寄存器用于选择要被扫描的输入顺序。

17.3 A/D 转换结果缓冲器

该模块包含一个 16 字双端口 RAM，称为 ADCBUF，用于缓存 A/D 结果。16 个缓冲器单元分别称为 ADCBUF0、ADCBUF1、ADCBUF2、……ADCBUFE 和 ADCBUFF。

注： A/D 转换结果缓冲器是只读缓冲器。

寄存器 17-1: ADCON1: A/D 控制寄存器 1

高字节:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
ADON	—	ADSIDL	—	—	—	FORM<1:0>	
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0 HC, HS	R/C-0 HC, HS
SSRC<2:0>			—	SIMSAM	ASAM	SAMP	DONE
bit 7							bit 0

- bit 15 **ADON:** A/D 工作模式位
1 = A/D 转换器模块正在工作
0 = A/D 转换器关闭
- bit 14 **未用:** 读作 0
- bit 13 **ADSIDL:** 空闲模式停止位
1 = 当器件进入空闲模式时, 模块停止工作
0 = 在空闲模式下, 模块继续工作
- bit 12-10 **未用:** 读作 0
- bit 9-8 **FORM<1:0>:** 数据输出格式位
11 = 有符号小数 (DOUT = sddd dddd dd00 0000)
10 = 小数 (DOUT = dddd dddd dd00 0000)
01 = 有符号整数 (DOUT = ssss sssd dddd dddd)
00 = 整数 (DOUT = 0000 00dd dddd dddd)
- bit 7-5 **SSRC<2:0>:** 转换触发源选择位
111 = 通过内部计数器结束采样并开始转换 (自动转换)
110 = 保留
101 = 保留
100 = 保留
011 = 通过电机控制 PWM 间隔结束采样并开始转换
010 = 通用 Timer3 比较结束采样并开始转换
001 = 通过 INTO 引脚的有效转变结束采样并开始转换
000 = 通过清除 SAMP 位结束采样并开始转换
- bit 4 **未用:** 读作 0
- bit 3 **SIMSAM:** 同时采样选择位 (只在 CHPS = 01 或 1x 时适用)
1 = 同时采样 CH0、CH1、CH2 和 CH3 (当 CHPS = 1x 时)
或
同时采样 CH0 和 CH1 (当 CHPS = 01 时)
0 = 按顺序逐个采样多个通道
- bit 2 **ASAM:** A/D 采样自动开始位
1 = 采样在上一次转换结束后立即开始。SAMP 位自动置位
0 = 采样在 SAMP 位置位时开始

寄存器 17-1: **ADCON1: A/D 控制寄存器 1**（续）

- bit 1

SAMP: A/D 采样使能位

1 = 至少一个 A/D 采样 / 保持放大器正在采样

0 = A/D 采样 / 保持放大器正在保持

当 ASAM = 0 时，写 1 到此位将开始采样

当 SSRC = 000 时，写 0 到此位将结束采样并开始转换
- bit 0

DONE: A/D 转换状态位（B 版本或更新版本的芯片）

1 = A/D 转换完成

0 = A/D 转换未完成

此位由软件清零，或新转换开始时清零

将此位清零不影响正在进行的任何操作

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
HC = 硬件清零	HS = 硬件置位	C = 可软件清零	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 17-2: ADCON2: A/D 控制寄存器 2

高字节:							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
VCFG<2:0>			保留	—	CSCNA	CHPS<1:0>	
bit 15			bit 8				

低字节:							
R-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS	—	SMPI<3:0>				BUFM	ALTS
bit 7							
		bit 0					

bit 15-13 **VCFG<2:0>**: 参考电压配置位

	A/D VREFH	A/D VREFL
000	AVDD	AVSS
001	外部 VREF+ 引脚	AVSS
010	AVDD	外部 VREF- 引脚
011	外部 VREF+ 引脚	外部 VREF- 引脚
1XX	AVDD	AVSS

bit 12 **保留**: 用户应在此位写入 0

bit 11 **未用**: 读作 0

bit 10 **CSCNA**: MUX A 输入多路开关设置的 CH0+ S/H 输入的扫描输入选择位
 1 = 扫描输入
 0 = 不扫描输入

bit 9-8 **CHPS<1:0>**: 选择通道使用的位
 1x = 转换 CH0、CH1、CH2 和 CH3
 01 = 转换 CH0 和 CH1
 00 = 转换 CH0
 当 SIMSAM 位 (ADCON1<3>) = 0 时, 多路通道顺序采样
 当 MSAM 位 (ADCON1<3>) = 1 时, 多路通道根据 CHPS<1:0> 的状态采样

bit 7 **BUFS**: 缓冲器填充状态位
 仅在 BUFM = 1 时有效 (ADRES 分成 2 x 8 字的缓冲器)
 1 = A/D 当前在填充缓冲器 0x8—0xF, 用户应该访问 0x0—0x7 的数据
 0 = A/D 当前在填充缓冲器 0x0—0x7, 用户应该访问 0x8—0xF 的数据

bit 6 **未用**: 读作 0

bit 5-2 **SMPI<3:0>**: 每产生一个中断的采样 / 转换过程数选择位
 1111 = 每完成 16 个采样 / 转换过程后产生中断
 1110 = 每完成 15 个采样 / 转换过程后产生中断

 0001 = 每完成 2 个采样 / 转换过程后产生中断
 0000 = 完成每个采样 / 转换过程后产生中断

bit 1 **BUFM**: 缓冲器模式选择位
 1 = 缓冲器配置为两个 8 字缓冲器 ADCBUF (15……8) 和 ADCBUF (7……0)
 0 = 缓冲器配置为一个 16 字缓冲器 ADCBUF (15……0)

bit 0 **ALTS**: 备用输入采样模式选择位
 1 = 为第一个采样使用 MUX A 输入多路开关设置, 然后对所有后续采样在 MUX B 和 MUX A 输入多路开关设置之间轮换
 0 = 总是使用 MUX A 输入多路开关设置

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

dsPIC30F 系列参考手册

寄存器 17-3: **ADCON3: A/D 控制寄存器 3**

高字节:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	SAMC<4:0>				
bit 15			bit 8				

低字节:							
R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADRC	—	ADCS<5:0>					
bit 7		bit 0					

- bit 15-13 未用: 读作 0
- bit 12-8 **SAMC<4:0>**: 自动采样时间位
11111 = 31 TAD
.....
00001 = 1 TAD
00000 = 0 TAD (只有在使用多个 S/H 放大器执行过程转换时才允许)
- bit 7 **ADRC**: A/D 转换时钟源位
1 = A/D 内部 RC 时钟
0 = 时钟由系统时钟产生
- bit 6 未用: 读作 0
- bit 5-0 **ADCS<5:0>**: A/D 转换时钟选择位
111111 = $T_{CY}/2 * (ADCS<5:0> + 1) = 32 * T_{CY}$
.....
000001 = $T_{CY}/2 * (ADCS<5:0> + 1) = T_{CY}$
000000 = $T_{CY}/2 * (ADCS<5:0> + 1) = T_{CY}/2$

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 17-4: ADCHS: A/D 输入选择寄存器

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CH123NB<1:0>		CH123SB	CH0NB	CH0SB<3:0>			
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CH123NA<1:0>		CH123SA	CH0NA	CH0SA<3:0>			
bit 7				bit 0			

bit 15-14 **CH123NB<1:0>**: MUX B 多路开关设置的通道 1、2、3 负输入选择位
与 bit 6-7 的定义相同 (注)

bit 13 **CH123SB**: MUX B 多路开关设置的通道 1、2、3 正输入选择位
与 bit 5 的定义相同 (注)

bit 12 **CH0NB**: MUX B 多路开关设置的通道 0 负输入选择位
与 bit 4 的定义相同 (注)

bit 11-8 **CH0SB<3:0>**: MUX B 多路开关设置的通道 0 正输入选择位
与 bit 3-0 的定义相同 (注)

bit 7-6 **CH123NA<1:0>**: MUX A 多路开关设置的通道 1、2、3 负输入选择位
11 = CH1 负输入为 AN9, CH2 负输入为 AN10, CH3 负输入为 AN11
10 = CH1 负输入为 AN6, CH2 负输入为 AN7, CH3 负输入为 AN8
0x = CH1, CH2, CH3 负输入为 VREF-

bit 5 **CH123SA**: MUX A 多路开关设置的通道 1、2、3 正输入选择位
1 = CH1 正输入为 AN3, CH2 正输入为 AN4, CH3 正输入为 AN5
0 = CH1 正输入为 AN0, CH2 正输入为 AN1, CH3 正输入为 AN2

bit 4 **CH0NA**: MUX A 多路开关选择的通道 0 负输入选择位
1 = 通道 0 负输入为 AN1
0 = 通道 0 负输入为 VREF-

bit 3-0 **CH0SA<3:0>**: MUX A 多路开关设置的通道 0 正输入选择位
1111 = 通道 0 正输入为 AN15
1110 = 通道 0 正输入为 AN14
1101 = 通道 0 正输入为 AN13
||
||
||
0001 = 通道 0 正输入为 AN1
0000 = 通道 0 正输入为 AN0

注: 模拟输入多路开关支持两种输入设置配置, 即 MUX A 和 MUX B。ADCHS<15:8> 决定 MUX B 的设置, ADCHS<7:0> 决定 MUX A 的设置。两组控制位功能相同。

注: 所选器件上可用的 A/D 输入的个数不同, ADCHS 寄存器的描述和功能也有所变化。欲了解此寄存器的其他细节, 请参考具体的器件数据手册。

图注:

R = 可读位	W = 可写位	U = 未用位, 读作 0
-n = POR 时的值	1 = 置位	0 = 清零 x = 未知

dsPIC30F 系列参考手册

寄存器 17-5: ADPCFG: A/D 端口配置寄存器

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 15-0 **PCFG<15:0>**: 模拟输入引脚配置控制位

1 = 模拟输入引脚处于数字模式, 使能端口读取输入, A/D 输入多路开关输入连接到 AVss

0 = 模拟输入引脚处于模拟模式, 禁止端口读取输入, A/D 采样引脚电压

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

寄存器 17-6: ADCSSL: A/D 输入扫描选择寄存器

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0
bit 7							bit 0

bit 15-0 **CSSL<15:0>**: A/D 输入引脚扫描选择位

1 = 选择对 ANx 输入进行扫描

0 = 不对 ANx 输入进行扫描

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

17.4 A/D 转换术语和转换过程

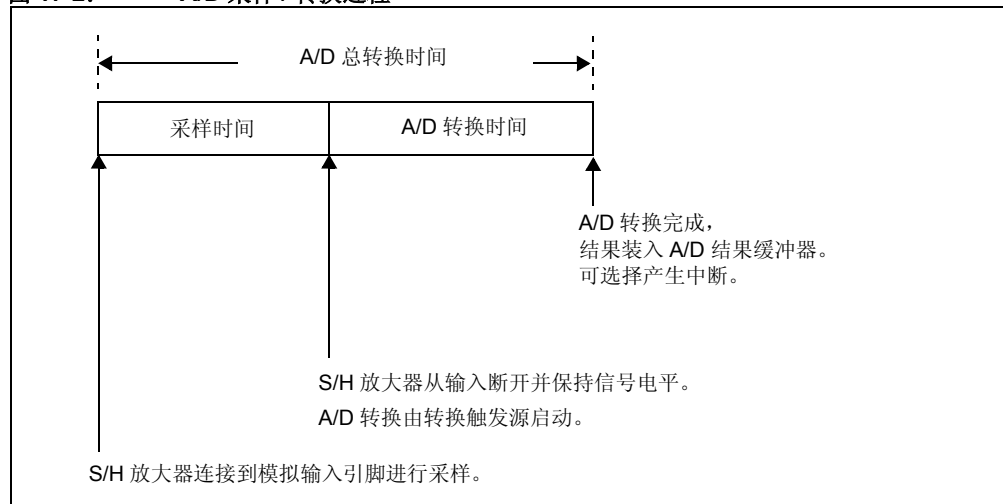
图 17-2 显示了一个基本转换过程及其所使用的术语。模拟输入引脚电压的采样是通过采样和保持 S/H 放大器进行的。S/H 放大器也称为 S/H 通道。10 位 A/D 转换器共有四个 S/H 通道，指定为 CH0—CH3。S/H 通道通过模拟输入多路开关连接到模拟输入引脚。模拟输入多路开关由 ADCHS 寄存器控制。在 ADCHS 寄存器中有两组多路开关控制位，其功能相同。这两组控制位允许设置两种不同的模拟输入多路开关配置（称为 MUX A 和 MUX B）。A/D 转换器在两次转换之间可以在 MUX A 和 MUX B 配置之间切换。A/D 转换器也可以选择对一系列模拟输入进行扫描。

采样时间是 A/D 模块的 S/H 放大器连接到模拟输入引脚的时间。采样时间可通过将 SAMP 位（ADCON1<1>）置位手动开始或由 A/D 转换器硬件自动开始。采样时间可通过在用户软件中将 SAMP 控制位清零手工结束，或由转换触发源自动结束。

转换时间是 A/D 转换器转换 S/H 放大器保持电压所需的时间。A/D 在采样时间结束时从模拟输入引脚断开。除了一个时钟周期外，A/D 转换器还需要一个 A/D 时钟周期（T_{AD}）来转换结果中的每个位。要完成整个转换，总共需要 12 个 T_{AD} 周期。当转换时间结束后，结果装入 16 个 A/D 结果寄存器（ADCBUF0……ADCBUFF）中的一个，S/H 可重新连接到输入引脚，并且可能会产生一个 CPU 中断。

采样时间和 A/D 转换时间之和就是总转换时间。为了确保 S/H 放大器能为 A/D 转换提供需要的精度，需要有一个最小采样时间（参见第 17.16 节“A/D 采样要求”）。此外，A/D 转换器还有多个输入时钟选择。用户必须选择不会违反最小 T_{AD} 规范的输入时钟选择。

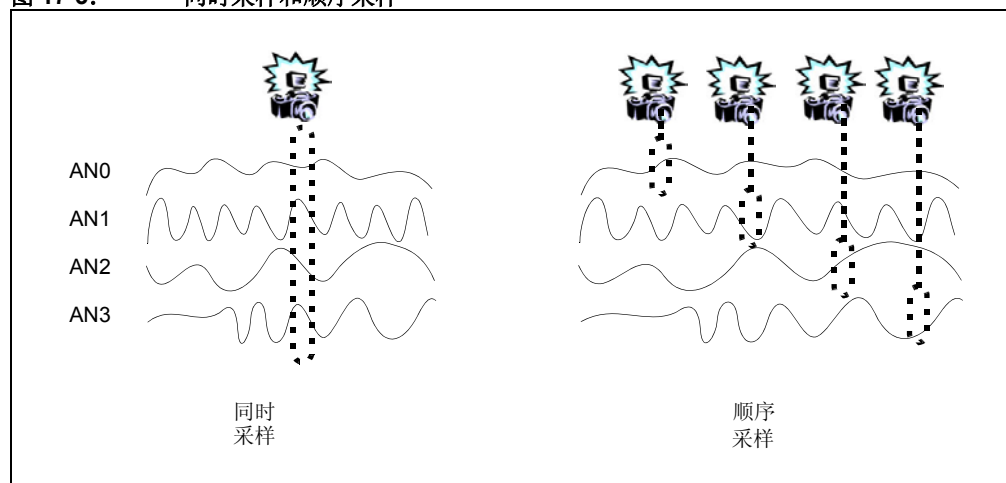
图 17-2: A/D 采样 / 转换过程



10 位 A/D 转换器有多种选择来指定采样 / 转换过程。如图 17-3 所示，采样 / 转换过程可以非常简单。图 17-3 中的示例只使用了一个 S/H 放大器。更精确的采样 / 转换过程使用多个的 S/H 放大器实现多个转换。10 位 A/D 转换器可以使用两个 S/H 放大器在一个采样 / 转换过程中完成两个转换，或使用四个 S/H 放大器在一个采样 / 转换过程中完成四个转换。在每个采样 / 转换过程中，S/H 放大器数量和每个采样的通道数量由 CHPS 控制位决定。

使用多个 S/H 通道的采样 / 转换过程可以同时采样或顺序采样，由 SIMSAM 位 (ADCON1<3>) 控制。同时采样多个信号可以确保所有的模拟输入的快照正好在同一时间发生。顺序采样在每个模拟输入的转换开始之前，获得输入的快照，多个输入的采样之间没有任何关系。

图 17-3: 同时采样和顺序采样



采样开始时间可通过将 SAMP 控制位置位由软件控制。采样开始时间还可以由硬件自动控制。当 A/D 转换器工作在自动采样模式时，当采样 / 转换过程中转换结束时，S/H 放大器会重新连接到模拟输入引脚。自动采样功能由 ASAM 控制位 (ADCON1<2>) 控制。

转换触发源结束采样时间并开始 A/D 转换或采样 / 转换过程。采样触发源由 SSRC 控制位选择。转换触发源可从多种硬件源中选择，或通过软件中将 SAMP 控制位清零来手工控制。转换触发源中的一个自动转换。自动转换之间的时间通过计数器和 A/D 时钟设置。自动采样模式和自动转换触发可以一起使用，提供无需软件参与的无限自动转换功能。

在每个或多个采样 / 转换过程结束时都可能会产生中断，这取决于 SMPI 控制位 ADCON2<5:2> 的值。中断之间的采样 / 转换过程数在 1 到 16 之间变化。用户应该注意，当选择了 SMPI 值时，A/D 转换缓冲器保持了 16 个结果。中断之间的总转换数为每次采样的通道数和 SMPI 值的积。中断之间的总转换数不应超过缓冲器长度。

17.5 A/D 模块配置

执行 A/D 转换时应遵循以下步骤：

1. 配置 A/D 模块
 - 选择端口引脚作为模拟输入 ADPCFG<15:0>
 - 选择参考电压源，以匹配模拟输入的预期范围 ADCON2<15:13>
 - 选择模拟转换时钟，以使预期的数据速率与处理器时钟匹配 ADCON3<5:0>
 - 确定要使用多少个 S/H 通道 ADCON2<9:8> 和 ADPCFG<15:0>
 - 确定采样如何发生 ADCON1<3> 和 ADCSSL<15:0>
 - 确定如何将输入分配给 S/H 通道 ADCHS<15:0>
 - 选择相应的采样 / 转换过程 ADCON1<7:0> 和 ADCON3<12:8>
 - 选择转换结果在缓冲器中的格式 ADCON1<9:8>
 - 选择中断频率 ADCON2<5:9>
 - 打开 A/D 模块 ADCON1<15>
2. 配置 A/D 中断（如果需要）
 - 清除 ADIF 位
 - 选择 A/D 中断优先级

每个配置步骤的选择将在后面的章节中说明。

17.6 参考电压源的选择

A/D 转换的参考电压通过 VCFG<2:0> 控制位（ADCON2<15:13>）选择。参考电压高电平（VREFH）和参考电压低电平（VREFL）可来自于内部 AVDD 和 AVSS 电压或 VREF+ 和 VREF- 输入引脚。

在低引脚数器件上，外部参考电压引脚可能会和 AN0 以及 AN1 输入复用。当这些引脚与 VREF+ 和 VREF- 输入引脚复用时，A/D 转换器仍然可以在这些引脚上执行转换。

加到外部参考电压引脚上的电压必须符合特定的规范。更多详细信息，请参见器件数据手册的“电气规范”部分。

注： 转换速率高于 500 ksp/s 时必须选用外部 VREF+ 和 VREF-。详情请参见第 17.22 节“A/D 转换速度”。

17.7 A/D 转换时钟的选择

A/D 转换器有一个可以完成转换的最大速率。转换时序由模拟模块时钟 T_{AD} 控制。A/D 转换需要 12 个时钟周期（12T_{AD}）。A/D 时钟是由器件指令时钟或内部 RC 时钟源产生。

A/D 转换时钟通过六位计数器软件选择。有 64 种可能的 T_{AD}，由 ADCS<5:0> 位（ADCON3<5:0>）指定。公式 17-1 给出了 T_{AD} 值与 ADCS 控制位，以及器件指令周期 T_{CY} 之间的关系。

公式 17-1: A/D 转换时钟周期

$$T_{AD} = \frac{T_{CY}(ADCS + 1)}{2}$$

$$ADCS = \frac{2T_{AD}}{T_{CY}} - 1$$

为了获得正确的 A/D 转换结果，必须选择 A/D 转换时钟（T_{AD}），以确保 83.33 ns 的最小 T_{AD} 时间（详情请参见第 17.22 节“A/D 转换速度”）。

A/D 转换器有一个专用内部 RC 时钟源，可以用于执行转换。当 dsPIC30F 处于休眠模式时进行 A/D 转换，应使用内部 RC 时钟源。内部 RC 振荡器是通过将 ADRC 位（ADCON3<7>）置位来选择的。当 ADRC 位置位时，ADCS<5:0> 位对 A/D 操作没有影响。

17.8 采样模拟输入的选择

为了选择采样的模拟输入，所有采样和保持放大器在其同相和反相输入上都有模拟多路开关（参见图 17-1）。一旦指定了采样 / 转换顺序，ADCHS 位决定为每个采样要选择的模拟输入。

此外，所选的输入可能会在交替采样的基础上发生变化，或者随采样的重复过程变化。

可将同一模拟输入连接到两个或更多的采样保持通道以提高转换速率。

注： 不同器件对应不同数量的模拟输入。应对照器件数据手册验证模拟输入的可用性。

17.8.1 配置模拟端口引脚

ADPCFG 寄存器指定了用作模拟输入的器件引脚的输入状态。

当相应的 PCFGn 位（ADPCFG<n>）清零时，对应的引脚就被配置为模拟输入。ADPCFG 寄存器在复位时清零，在默认情况下，这使得 A/D 输入引脚在复位时被配置为模拟输入。

当引脚被配置为模拟输入时，相关端口 I/O 数字输入缓冲器被禁止，因此不消耗电流。

ADPCFG 寄存器和 TRISB 寄存器控制 A/D 端口引脚的工作方式。

对于希望作为模拟输入的端口引脚，其对应的 TRIS 位必须置位，来指定端口为输入。如果与 A/D 输入关联的 I/O 引脚被配置为输出，则 TRIS 位被清零，且端口数字输出电平（VOH 或 VOL）将被转换。在器件复位后，所有的 TRIS 位都会置位。

当相应的 PCFGn 位（ADPCFG<n>）置位时，对应的引脚就被配置为数字 I/O。在此配置中，模拟多路开关的输入连接到 AVss。

注 1: 当读取 A/D 端口寄存器时，任何配置为模拟输入的引脚都读作 0。
2: 在任何定义为数字输入引脚（包括 AN15:AN0 引脚）所加的模拟电平都可能导致输入缓冲器消耗超出器件规范中的规定的电流。

17.8.2 通道 0 输入选择

在选择模拟输入时，通道 0 是 4 个 S/H 通道中最灵活的。

用户可以选择最多 16 个模拟输入中的任何一个作为通道的正输入。CH0SA<3:0> 位（ADCHS<3:0>）通常用来选择作为通道 0 正输入的模拟输入。

用户可以选择 VREF- 或 AN1 做为通道的负输入。CH0NA 位（ADCHS<4>）通常用来选择作为通道 0 负输入的模拟输入。

17.8.2.1 指定交替通道 0 输入选择

在顺序采样时，ALTS 位（ADCON2<0>）会使模块在两组选择的输入间切换。

CH0SA<3:0>、CH0NA、CHXSA和CHXNA<1:0>指定的输入统称为MUX A输入。CH0SB<3:0>、CH0NB、CHXSB和CHXNB<1:0>指定的输入统称为MUX B输入。当ALTS位为1时，模块将在一个采样中使用MUX A输入，而在随后一个采样中使用MUX B输入，依次进行轮换。

对于通道0，如果ALTS位为0，则只会选择采样CH0SA<3:0>和CH0NA指定的输入。

如果ALTS位为1，在通道0的第一个采样/转换过程中，则会选择采样CH0SA<3:0>和CH0NA指定的输入。在通道0的下一个采样转换过程中，会选择采样CH0SB<3:0>和CH0NB指定的输入。此模式将在后续采样转换过程中重复。

注意如果指定了多通道（CHPS = 01 或 1x）和同时采样（SIMSAM = 1），由于所有通道在每个采样时间进行采样，交替输入会改变每个采样。如果指定了多通道（CHPS = 01 或 1x）和顺序采样（SIMSAM = 0），交替输入只在特定通道的每个采样上发生改变。

17.8.2.2 使用通道 0 扫描数个输入

通道0能扫描一组选定的输入。CSCNA位（ADCON2<10>）可以使CH0通道扫描选定数量的模拟输入。当CSCNA置位时，会忽略CH0SA<3:0>位。

ADCSSL寄存器指定要扫描的输入。ADCSSL寄存器中的每个位分别对应一个模拟输入。bit 0 对应AN0，bit 1 对应AN1，依此类推。如果ADCSSL寄存器中一个特定位为1，则在扫描过程中将扫描对应的输入。扫描总是在每次发生中断后从第一个所选择的通道开始，从编号较小的输入扫描到编号较大的输入。

注： 如果选择的扫描输入个数大于每次中断的采样数，则编号较大的输入将不会被采样。

ADCSSL位只指定通道正输入的输入源。CH0NA位将选择扫描时通道负输入的输入源。

如果ALTS位为1，扫描只应用于MUX A输入选择。由CH0SB<3:0>指定的MUX B输入选择仍选择轮换的通道0输入。当以这种方式设定输入选择时，通道0输入将在一系列ADCSSL寄存器指定的扫描输入和CH0SB位指定的固定输入之间轮换。

17.8.3 通道 1、2 和 3 输入选择

通道 1、2 和 3 可以对一部分模拟输入引脚的进行采样。通道 1、2 和 3 可以选择两组 3 个输入中的一组。

CHXSA 位 (ADCHS<5>) 用于选择通道 1、2 和 3 的正输入源。

清零 CHXSA 可以选择 AN0、AN1 和 AN2 分别作为通道 1、2 和 3 的正输入源。将 CHXSA 置位可以选择 AN3、AN4 和 AN5 作为模拟输入源。

CHXNA<1:0> 位 (ADCHS<7:6>) 用于选择通道 1、2 和 3 的负输入源。

设定 CHXNA = 0x, 选择 VREF- 为通道 1、2 和 3 的负输入的模拟输入源。设定 CHXNA = 10 选择 AN6、AN7 和 AN8 分别作为通道 1、2 和 3 的负输入模拟输入源。设定 CHXNA = 11 选择 AN9、AN10 和 AN11 作为模拟输入源。

17.8.3.1 为单模拟输入选定多个通道

可将模拟输入复用器配置为将同一输入引脚与两个或更多采样保持通道相连接。A/D 将对保存在某个 S/H 通道中的值进行转换，同时第二个 S/H 通道采样新的输入。

17.8.3.2 指定交替的通道 1、2 和 3 输入选择

与通道 0 输入一样，在通道 1、2 和 3 的顺序采样过程中，ALTS 位 (ADCON2<0>) 使模块在选择的两组输入之间切换。

CHXSA 和 CHXNA<1:0> 指定的 MUX A 输入总是在 ALTS = 0 时选择输入。

当 ALTS = 1 时，由 CHXSB 和 CHXNB<1:0> 指定的 MUX B 输入与 MUX A 输入交替。

17.9 模块使能

当 ADON 位 (ADCON1<15>) 为 1 时，模块处于有效模式，且以全功耗模式工作。

当 ADON 为 0 时，模块被禁止。为了最大限度地节省电流消耗，电路的数字和模拟部分被关闭。

为了从关闭模式返回有效模式，用户必须等待模拟阶段稳定下来。有关稳定时间的信息，请参见器件数据手册的电气特性部分。

注： 当 ADON = 1 时，不应写入 SSRC<2:0>、SIMSAM、ASAM、CHPS<1:0>、SMPI<3:0>、BUFM 和 ALTS 位以及 ADCON3 和 ADCSSL 寄存器，否则会产生无法预料的结果。

17.10 采样 / 转换过程的说明

10 位 A/D 模块有 4 个采样 / 保持放大器和一个 A/D 转换器。此模块可以在每个采样 / 转换过程实现 1、2 或 4 个输入采样和 A/D 转换。

17.10.1 采样 / 保持通道的数量

CHPS<1:0> 控制位 (ADCON2<9:8>) 用于选择 A/D 模块在采样 / 转换过程中使用 S/H 放大器的个数。有以下三种选项：

- 仅 CH0
- CH0 和 CH1
- CH0、CH1、CH2 和 CH3

CHPS 控制位与 SIMSAM (同时采样) 控制位 (ADCON1<3>) 配合工作。

17.10.2 同时采样使能

某些应用可能要求在同一时间采样多个信号。如表 17-1 所示，SIMSAM 控制位（ADCON1<3>）与 CHPS 控制位配合工作来控制多通道采样 / 转换过程。如果 CHPS<1:0> = 00，SIMSAM 控制位不影响模块操作。如果 CHPS 控制位使能多个的 S/H 放大器，且 SIMSAM 位为 0，在两个或四个采样周期内，两个或四个被选的通道就会被顺序采样和转换。如果 SIMSAM 位为 1，两个或四个选择的通道就会在一个采样周期内被同时采样。然后这些通道被顺序转换。

表 17-1: 采样 / 转换控制选项

CHPS<1:0>	SIMSAM	采样 / 转换过程	要完成的采样 / 转换周期数	示例
00	x	采样 CH0, 转换 CH0	1	图 17-4, 图 17-5, 图 17-6, 图 17-7, 图 17-10, 图 17-11, 图 17-14, 图 17-15
01	0	采样 CH0, 转换 CH0 采样 CH1, 转换 CH1	2	
1x	0	采样 CH0, 转换 CH0 采样 CH1, 转换 CH1 采样 CH2, 转换 CH2 采样 CH3, 转换 CH3	4	图 17-9, 图 17-13, 图 17-20
01	1	同时采样 CH0 和 CH1 转换 CH0 转换 CH1	1	图 17-18
1x	1	同时采样 CH0、CH1、CH2 和 CH3 转换 CH0 转换 CH1 转换 CH2 转换 CH3	1	图 17-8 图 17-12, 图 17-16, 图 17-17, 图 17-9

17.11 如何开始采样

17.11.1 手工

将 SAMP 位（ADCON1<1>）置位将使 A/D 开始采样。可以使用几个选项之一来结束采样并完成转换。在 SAMP 位再次置位前，不会重新开始采样。如图 17-4 所示。

17.11.2 自动

只要在该通道上没有进行转换，将 ASAM 位（ADCON1<2>）置位，会导致 A/D 自动开始采样通道。可以使用几个选项之一结束采样并完成转换。如果 SIMSAM 位指定顺序采样，通道上的采样在该通道的转换完成后恢复。如果 SIMSAM 位指定同时采样，则通道上的采样在所有通道的转换完成后恢复。如图 17-5 所示。

17.12 如何停止采样和开始转换

转换触发源将终止采样并开始所选的转换过程。SSRC<2:0> 位 (ADCON1<7:5>) 选择转换触发源。

注： dsPIC30F 器件不同，可用的转换触发源可能也会有所不同。有关可用转换触发源的信息，请参见特定器件数据手册。

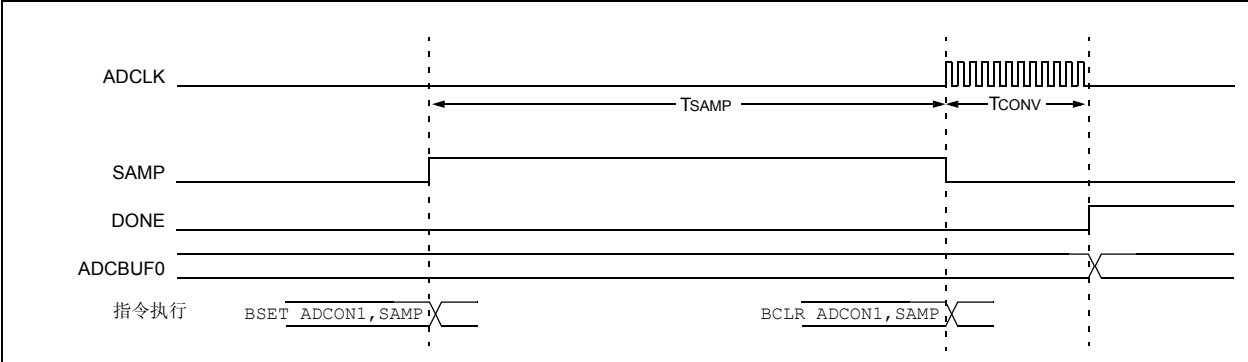
注： A/D 模块使能时，不应修改 SSRC 选择位。如果用户希望修改转换触发源，应首先将 ADON 位 (ADCON1<15>) 清零来禁止 A/D 模块。

17.12.1 手工

当 SSRC<2:0> = 000 时，转换触发处于软件控制下。将 SAMP 位 (ADCON1<1>) 清零将会开始转换过程。

图 17-4 示例说明了置位 SAMP 位开始采样，将 SAMP 位清零终止采样并开始转换的过程。用户软件必须对 SAMP 位置位和清零计时，以确保输入信号有足够的采样时间。代码示例参见例 17-1。

图 17-4: 转换一个通道 (手工采样开始, 手工转换开始)



例 17-1: 转换一个通道 (手工采样开始, 手工转换开始) 的代码示例

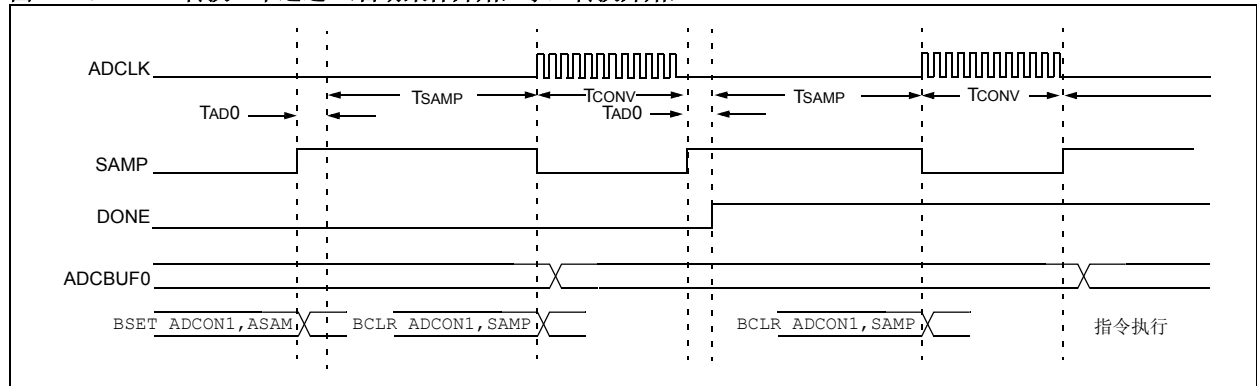
```
ADPCFG = 0xFFFB;           // all PORTB = Digital; RB2 = analog
ADCON1 = 0x0000;           // SAMP bit = 0 ends sampling ...
                             // and starts converting
ADCHS  = 0x0002;           // Connect RB2/AN2 as CH0 input ..
                             // in this example RB2/AN2 is the input

ADCSSL = 0;
ADCON3 = 0x0002;           // Manual Sample, Tad = internal 2 Tcy
ADCON2 = 0;

ADCON1bits.ADON = 1;       // turn ADC ON
while (1)                  // repeat continuously
{
    ADCON1bits.SAMP = 1;    // start sampling ...
    DelayNmSec(100);        // for 100 mS
    ADCON1bits.SAMP = 0;    // start Converting
    while (!ADCON1bits.DONE); // conversion done?
    ADCValue = ADCBUF0;     // yes then get ADC value
}
```

图 17-5 给出了一个示例，说明置位 **ASAM** 位开始自动采样，将 **SAMP** 位清零终止采样并开始转换的过程。转换完成后，模块将自动返回采样状态。在采样间隔开始时，**SAMP** 位会自动置位。用户软件必须对 **SAMP** 位的清零计时，以确保输入信号有充足的采样时间，因为 **SAMP** 位清零的时间中除了采样时间外还包括转换时间。参见例 17-2 的代码示例。

图 17-5: 转换一个通道（自动采样开始，手工转换开始）



例 17-2: 转换一个通道（自动采样开始，手工转换开始）的代码示例

```
ADPCFG = 0xFF7F;           // all PORTB = Digital but RB7 = analog
ADCON1 = 0x0004;           // ASAM bit = 1 implies sampling ..
                             // starts immediately after last
                             // conversion is done
ADCHS  = 0x0007;           // Connect RB7/AN7 as CH0 input ..
                             // in this example RB7/AN7 is the input
ADCSSL = 0;
ADCON3 = 0x0002;           // Sample time manual, Tad = internal 2 Tcy
ADCON2 = 0;

ADCON1bits.ADON = 1;       // turn ADC ON
while (1)                  // repeat continuously
{
    DelayNmSec(100);        // sample for 100 mS
    ADCON1bits.SAMP = 0;    // start Converting
    while (!ADCON1bits.DONE); // conversion done?
    ADCValue = ADCBUF0;     // yes then get ADC value
}                           // repeat
```

17.12.2 对转换触发计时

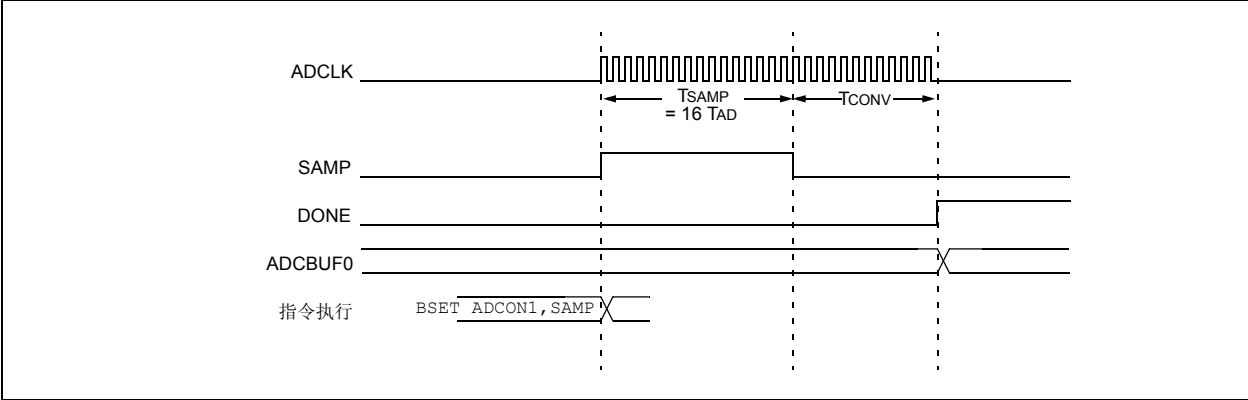
当 $SSRC<2:0> = 111$ 时，转换触发处于 A/D 时钟控制下。SAMC 位（ADCON3<12:8>）选择开始采样和开始转换之间的 T_{AD} 时钟周期数。此触发选项提供了多通道上最快的转换速率。在采样开始后，模块会对 SAMC 位指定的 T_{AD} 时钟周期计数。

公式 17-2: 计时转换触发时间

$$T_{SMP} = SAMC<4:0> * T_{AD}$$

当只使用 1 个 S/H 通道或同时采样时，SAMC 必须始终编程为至少一个时钟周期。当使用多个 S/H 通道进行顺序采样时，将 SAMC 编程为零时钟周期就可得到可能的最快转换速率。参见例 17-3 的代码示例。

图 17-6: 转换一个通道（手工采样开始，基于 T_{AD} 的转换开始）



例 17-3: 转换一个通道（手工采样开始，基于 T_{AD} 的转换开始）的代码示例

```
ADPCFG = 0xEFFF;           // all PORTB = Digital; RB12 = analog
ADCON1 = 0x00E0;           // SSRC bit = 111 implies internal
                             // counter ends sampling and starts
                             // converting.
ADCHS  = 0x000C;           // Connect RB12/AN12 as CH0 input ..
                             // in this example RB12/AN12 is the input
ADCSSL = 0;
ADCON3 = 0x1F02;           // Sample time = 31Tad, Tad = internal 2 Tcy
ADCON2 = 0;

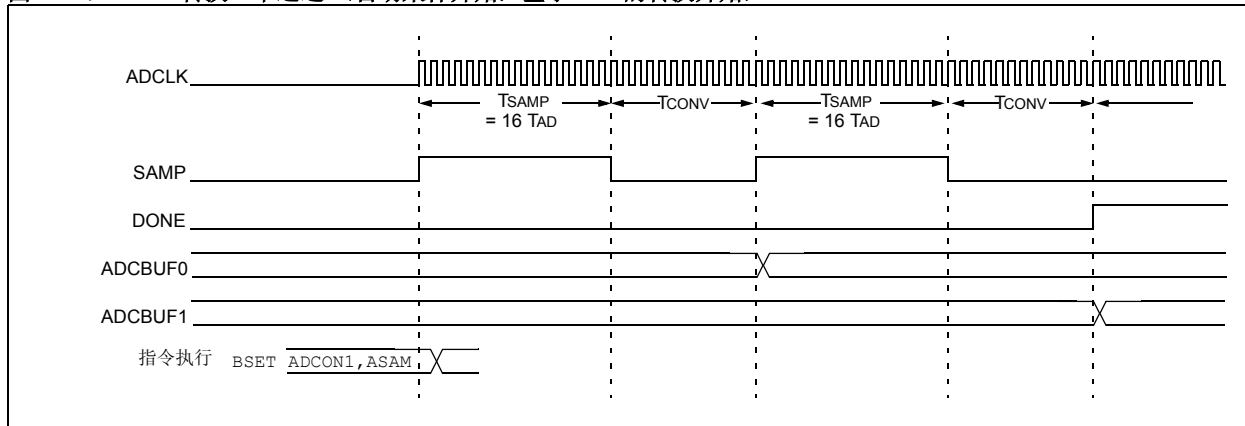
ADCON1bits.ADON = 1;       // turn ADC ON
while (1)                  // repeat continuously
{
    ADCON1bits.SAMP = 1;    // start sampling then ...
                             // after 31Tad go to conversion
    while (!ADCON1bits.DONE); // conversion done?
    ADCValue = ADCBUF0;     // yes then get ADC value
}                           // repeat
```


17.12.2.1 自由运行采样转换过程

如图 17-7 所示，使用自动转换触发模式（SSRC = 111），配合自动采样开始模式（ASAM = 1），可以确定采样 / 转换的过程，而无需用户干预或其他器件资源。此“计时”模式允许在模块初始化后进行连续数据收集。参见例 17-4 的代码示例。

注： 转换速率为 750 ksps 时应使能此 A/D 配置（详情请参见第 17.22 节“A/D 转换速度”）。

图 17-7: 转换一个通道（自动采样开始，基于 TAD 的转换开始）



例 17-4: 转换一个通道（自动采样开始，基于 TAD 的转换开始）的代码示例

```
ADPCFG = 0xFFFFB;           // all PORTB = Digital; RB2 = analog
ADCON1 = 0x00E0;             // SSRC bit = 111 implies internal
                               // counter ends sampling and starts
                               // converting.
ADCHS = 0x0002;              // Connect RB2/AN2 as CH0 input ..
                               // in this example RB2/AN2 is the input

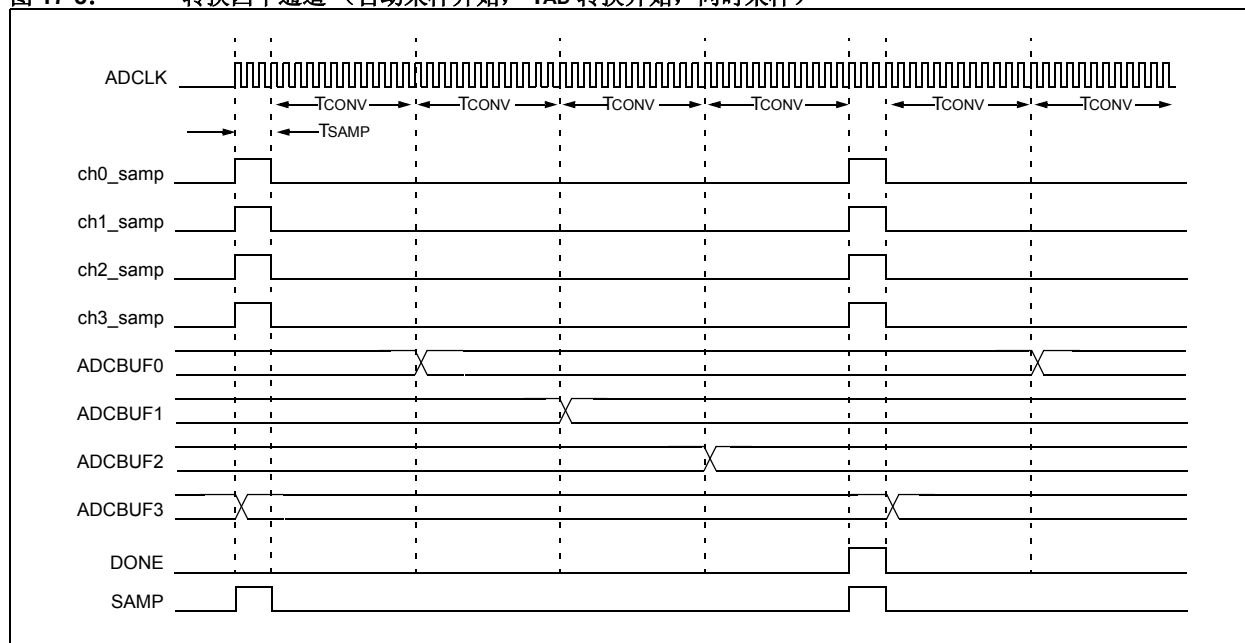
ADCSSL = 0;
ADCON3 = 0x0F00;             // Sample time = 15Tad, Tad = internal Tcy/2
ADCON2 = 0x0004;             // Interrupt after every 2 samples

ADCON1bits.ADON = 1;         // turn ADC ON
while (1)                    // repeat continuously
{
    ADCValue = 0;             // clear value
    ADC16Ptr = &ADCBUF0;     // initialize ADCBUF pointer
    IFS0bits.ADIF = 0;        // clear ADC interrupt flag
    ADCON1bits.ASAM = 1;      // auto start sampling
                               // for 31Tad then go to conversion
    while (!IFS0bits.ADIF);    // conversion done?
    ADCON1bits.ASAM = 0;       // yes then stop sample/convert
    for (count = 0; count < 2; count++) // average the 2 ADC value
        ADCValue = ADCValue + *ADC16Ptr++;
    ADCValue = ADCValue >> 1;
}                             // repeat
```

17.12.2.2 多通道同时采样

如图 17-8 所示，当使用同时采样时，采样时间由 **SAMC** 值指定。在此示例中，**SAMC** 指定 3 T_{AD} 的采样时间。因为自动采样开始有效，在最后一个转换结束时，采样在所有通道上自动开始，并会持续 3 个 A/D 时钟周期。参见例 17-5 的代码示例。

图 17-8: 转换四个通道（自动采样开始， T_{AD} 转换开始，同时采样）



例 17-5: 转换四个通道（自动采样开始， T_{AD} 转换开始，同时采样）的代码示例

```
ADPCFG = 0xFF78;           // RB0,RB1,RB2 & RB7 = analog
ADCON1 = 0x00EC;           // SIMSAM bit = 1 implies ...
                             // simultaneous sampling
                             // ASAM = 1 for auto sample after convert
                             // SSRC = 111 for 3Tad sample time

ADCHS = 0x0007;            // Connect AN7 as CH0 input

ADCSSL = 0;
ADCON3 = 0x0302;           // Auto Sampling 3 Tad, Tad = internal 2 Tcy
ADCON2 = 0x030C;           // CHPS = 1x implies simultaneous ...
                             // sample CH0 to CH3
                             // SMPI = 0011 for interrupt after 4 converts

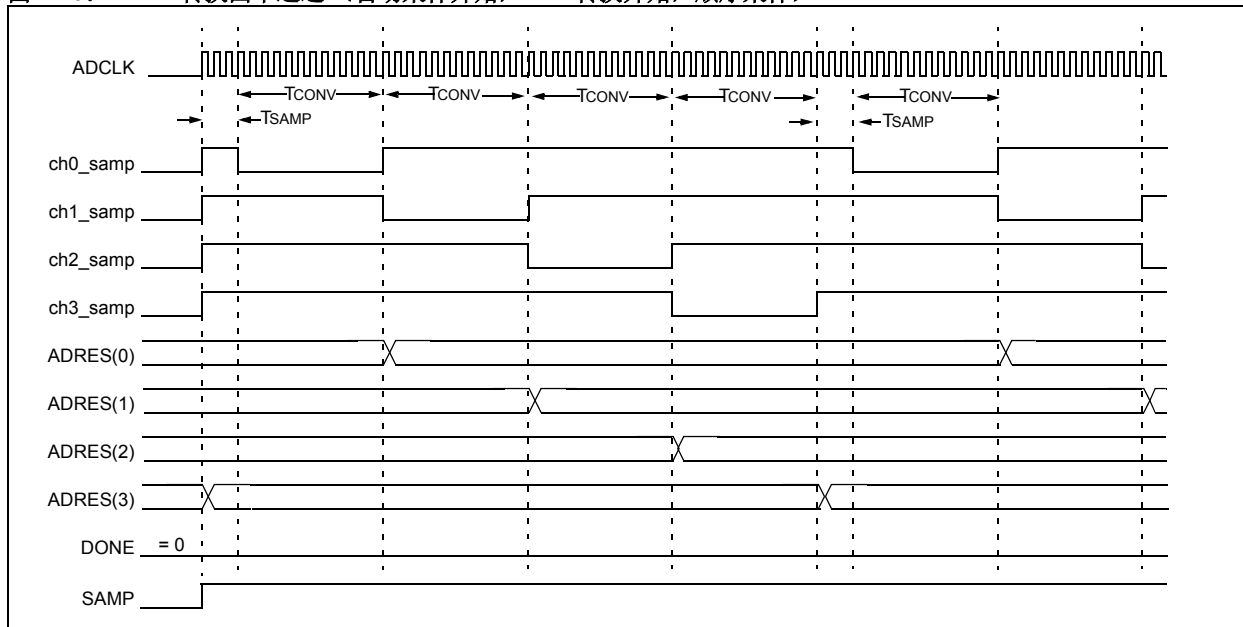
ADCON1bits.ADON = 1;       // turn ADC ON
while (1)                  // repeat continuously
{
    ADC16Ptr = &ADCBUF0;   // initialize ADCBUF pointer
    OutDataPtr = &OutData[0]; // point to first TXbuffer value
    IFS0bits.ADIF = 0;      // clear interrupt
    while (IFS0bits.ADIF);  // conversion done?
    for (count = 0; count < 4; count++) // save the ADC values
    {
        ADCValue = *ADC16Ptr++;
        LoadADC(ADCValue);
    }
}
```

17.12.2.3 多通道顺序采样

如图 17-9 所示，当使用顺序采样时，采样时间先于每个转换时间。在示例中，每个通道的采样时间都增加了 3 T_{AD} 。

注： 转换速率为 1 Msps 和 600 ksps 时应使能此 A/D 配置（详情请参见第 17.22 节“A/D 转换速度”）。

图 17-9: 转换四个通道（自动采样开始， T_{AD} 转换开始，顺序采样）



17.12.2.4 使用计时转换触发和自动采样时采样时间的注意事项

不同的采样 / 转换过程为 S/H 通道提供不同的可用采样时间来采集模拟信号。用户必须确保采样时间超过第 17.16 节“A/D 采样要求”中规定的要求。

假设模块设置为自动采样，并使用计时转换触发，则采样间隔取决于 SAMC 位。

如果 SIMSAM 位指定了同时采样或只有一个有效通道，采样时间为 SAMC 位指定的周期。

公式 17-3: 可用采样时间，同时采样

$$T_{SMP} = SAMC<4:0> * T_{AD}$$

如果 SIMSAM 位指定了顺序采样，用于转换所有通道的总间隔时间等于通道数量乘以采样时间与转换时间之和。单个通道的转换时间等于总间隔时间减去该通道的转换时间。

公式 17-4: 可用采样时间，同时采样

$$\begin{aligned} T_{SEQ} &= \text{每次采样的通道 (CH/S)} * \\ &\quad ((SAMC<4:0> * T_{AD}) + \text{转换时间 (TCONV)}) \\ T_{SMP} &= (T_{SEQ} - T_{CONV}) \end{aligned}$$

注 1: CHPS<1:0> 位指定 CH/S。

注 2: TSEQ 是采样 / 转换过程的总时间。

17.12.3 事件触发转换开始

通常需要将采样结束和转换开始与某个其他时间事件同步。A/D 模块可以使用三个触发源之一作为转换触发事件。

17.12.3.1 外部 INT 引脚触发

当 $SSRC<2:0> = 001$ 时，A/D 转换是由 INT0 引脚上的有效电平转换触发的。INT0 引脚可以编程为上升沿输入或下降沿输入。

17.12.3.2 通用定时器比较触发

通过将 $SSRC<2:0>$ 置为 010，可将 A/D 配置为此触发模式。如果 32 位定时器 TMR3/TMR2 和 32 位组合周期寄存器 PR3/PR2 之间匹配，则 Timer3 会产生一个特殊的 ADC 触发事件信号。TMR5/TMR4 定时器对不具备此功能。更多详细信息可参见第 12 章“定时器”。

17.12.3.3 电机控制 PWM 触发器

PWM 模块有一个事件触发器，允许 A/D 转换与 PWM 时基同步。当 $SSRC<2:0> = 011$ 时，A/D 采样和转换时间发生在 PWM 周期中的任何用户可编程点。特殊事件触发器可以让用户将需要 A/D 转换结果的时间与占空比值更新的时间之间的延迟减到最小。更多详细信息可参见第 15 章“电机控制 PWM”。

17.12.3.4 A/D 工作与内部或外部事件同步

外部事件触发脉冲结束采样，并启动转换（ $SSRC = 001$ 、010 或 011）的模式可以与自动采样（ $ASAM = 1$ ）组合使用，以使 A/D 采样转换事件与触发脉冲源同步。例如，在图 17-11 中， $SSRC = 010$ 和 $ASAM = 1$ 的情况下，A/D 的结束采样和启动转换总是与定时器比较触发事件同步。A/D 将有与定时器比较事件速率对应的采样转换速率。参见例 17-6 的代码示例。

图 17-10： 转换一个通道（手工采样开始，基于转换触发的转换开始）

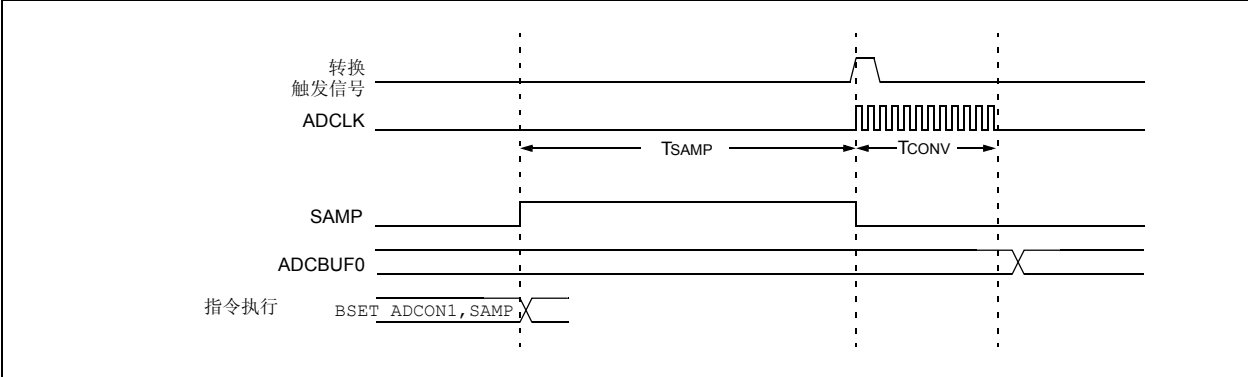
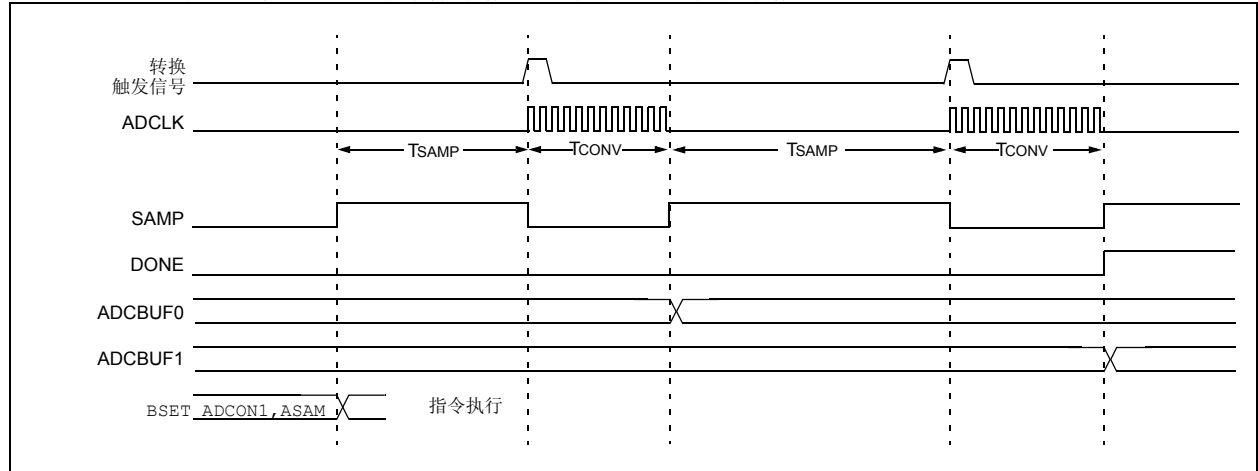


图 17-11: 转换一个通道 (自动采样开始, 基于转换触发的转换开始)



例 17-6: 转换一个通道 (自动采样开始, 基于转换触发转换开始) 的代码示例

```

ADPCFG = 0xFFFF;           // all PORTB = Digital; RB2 analog
ADCON1 = 0x0040;           // SSRC bit = 010 implies GP TMR3
                             // compare ends sampling and starts
                             // converting.
ADCHS  = 0x0002;           // Connect RB2/AN2 as CH0 input ..
                             // in this example RB2/AN2 is the input
ADCSSL = 0;
ADCON3 = 0x0000;           // Sample time is TMR3, Tad = internal Tcy/2
ADCON2 = 0x0004;           // Interrupt after 2 conversions

// set TMR3 to time out every 125 mSecs
TMR3  = 0x0000;
PR3   = 0x3FFF;
T3CON = 0x8010;

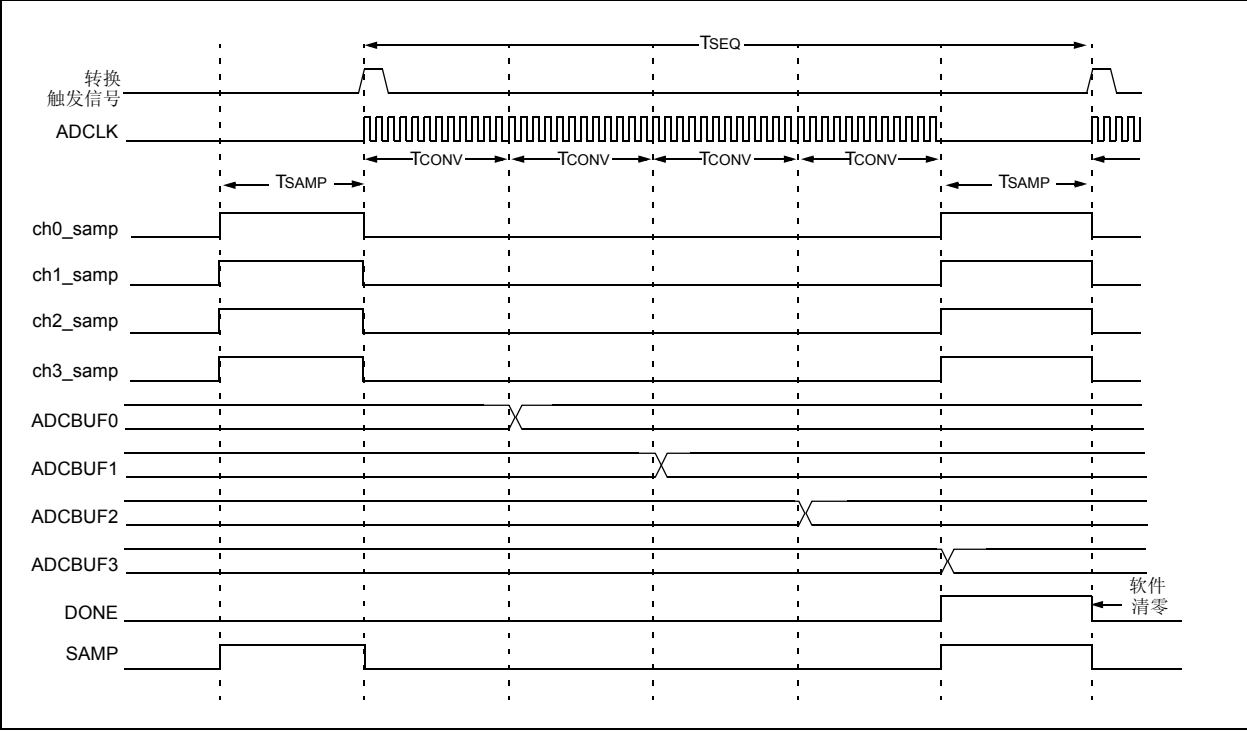
ADCON1bits.ADON = 1;       // turn ADC ON
ADCON1bits.ASAM = 1;       // start auto sampling every 125 mSecs
while (1)                  // repeat continuously
{
    while (!IFS0bits.ADIF); // conversion done?
    ADCValue = ADCBUF0;     // yes then get first ADC value
    IFS0bits.ADIF = 0;     // clear ADIF
}                           // repeat

```

17.12.3.5 多通道同时采样

如图 17-12 所示，当使用同时采样时，采样将在 ASAM 位置位或最后一个转换结束时在所有通道上开始。当转换触发事件发生时，采样停止，转换开始。

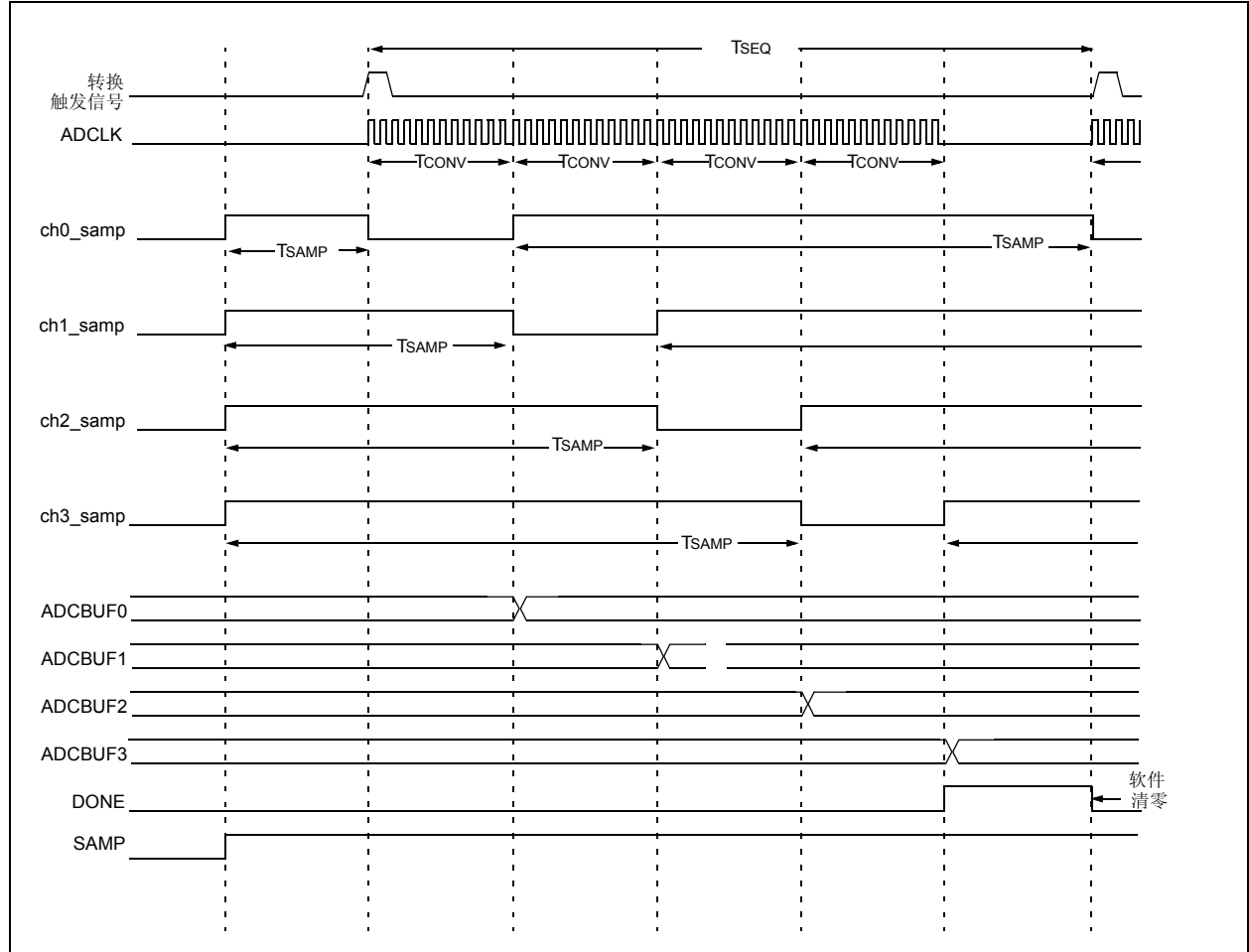
图 17-12: 转换四个通道（自动采样开始，触发转换开始，同时采样）



17.12.3.6 多通道顺序采样

如图 17-13 所示，当使用顺序采样时，特定通道的采样在转换该通道前停止，并在转换停止后恢复。

图 17-13: 转换四个通道（自动采样开始，触发转换开始，顺序采样）



17.12.3.7 自动采样 / 转换过程的采样时间注意事项

不同的采样 / 转换过程为 S/H 通道提供不同的可用采样时间来采集模拟信号。用户必须确保采样时间超过第 17.16 节 “A/D 采样要求” 中的规定。

假设该模块设置为自动采样，并把外部触发脉冲用作转换触发源，则采样间隔是触发脉冲间隔的一部分。

如果 SIMSAM 位指定了同时采样，采样时间等于触发脉冲周期减去完成指定转换所需的时间。

公式 17-5: 可用采样时间，同时采样

$$\begin{aligned} \text{TSMP} &= \text{触发脉冲间隔 (TSEQ)} - \\ &\quad \text{每次采样的通道 (CH/S) * 转换时间 (TCONV)} \\ \text{TSMP} &= \text{TSEQ} - (\text{CH/S} * \text{TCONV}) \end{aligned}$$

注 1: CHPS<1:0> 位指定 CH/S。
2: TSEQ 是触发脉冲间隔的时间。

如果 SIMSAM 位指定了顺序采样，采样时间等于触发脉冲周期减去完成一个转换所需的时间。

公式 17-6: 可用采样时间，顺序采样

$$\begin{aligned} \text{TSMP} &= \text{触发脉冲间隔 (TSEQ)} - \\ &\quad \text{转换时间 (TCONV)} \\ \text{TSMP} &= \text{TSEQ} - \text{TCONV} \end{aligned}$$

注: TSEQ 是触发脉冲间隔时间。

17.13 采样 / 转换工作的控制

应用软件可以查询 SAMP 和 CONV 位的状态，以对 A/D 工作进行跟踪，或者当转换完成时，该模块可能中断 CPU。必要时应用软件还可以中止 A/D 工作。

17.13.1 监视采样 / 转换状态

SAMP (ADCON1<1>) 和 DONE (ADCON1<0>) 位分别表示 A/D 处于采样状态和转换状态。通常，SAMP 位清零表示采样结束，DONE 位自动置位以表示转换结束。如果 SAMP 和 DONE 都为“0”，则 A/D 处于无效状态。在某些工作模式下，SAMP 位也可以开始和终止采样。

17.13.2 产生 A/D 中断

SMPI<3:0> 位控制中断的产生。启动采样之后，若干次采样 / 转换过程后将发生中断，并在每次经过相同次数的采样之后重新发生中断。注意中断是按采样，而不是按转换或缓冲存储器中的数据样本指定的。

如果 SIMSAM 位指定顺序采样，无论 CHPS 位指定的通道数如何，模块会对每个转换和缓冲器中的数据样本进行一次采样。因此，SMPI 位指定的值将对应缓冲器中的数据样本数，最多可达 16 个。

当 SIMSAM 位指定同时采样时，缓冲器中数据样本的数量与 CHPS 位相关。从算法上来说，通道 / 采样乘以采样次数就是缓冲器中数据样本的数量。为了防止缓冲器溢出丢失数据，SMPI 位必须被设置为所需缓冲器大小除以每个采样的通道数。

不能使用 SMPI 位禁止 A/D 中断。要禁止该中断，应将 ADIE 模拟模块的中断使能位清零。

17.13.3 中止采样

在手动采样模式下清零 SAMP 位将终止采样，但如果 SSRC = 000，也可能启动转换。

在自动采样模式下清零 ASAM 位将不会终止正在进行的采样 / 转换过程。然而，在随后的转换完成之后，采样不会自动恢复。

17.13.4 中止转换

在转换过程中，清零 ADON 位将中止当前的转换。不会用部分完成的 A/D 转换样本来更新 A/D 结果寄存器对。即，对应的 ADCBUF 缓冲器单元将仍然保持上一次转换完成后的值（即上一次写入该缓冲器的值）。

17.14 如何将转换结果写入缓冲器的说明

当转换完成时，该模块把转换结果写入 A/D 结果缓冲器。该缓冲器是 16 个 10 位字的 RAM 阵列。可以通过 SFR 空间内的 16 个地址单元访问该缓冲器，这些单元分别名为 ADCBUF0……ADCBUFF。

用户软件可能试图在每个 A/D 转换结果产生时读取它，但这可能会消耗太多的 CPU 时间。通常，为了简化代码，该模块会用结果填充该缓冲器，而后在该缓冲器填满时产生中断。

17.14.1 每次中断前的转换次数

SMPI<3:0> 位 (ADCON2<5:2>) 用来选择在中断 CPU 前将发生多少次 A/D 转换。每个中断前的采样数量可以为 1 到 16 之间的任何一个。每次中断后，A/D 转换器模块总是开始将其转换结果写入缓冲器的起始单元。例如，如果 SMPI<3:0> = 0000，那么转换结果总是会写入 ADCBUF0。在这个例子中，不会用到其他缓冲器单元。

17.14.2 缓冲器大小造成的限制

当 BUFM 位 (ADCON2<1>) 为 0 时，用户对以下情况不能编程，若 CHPS 和 SMPI 位的联合指定每个中断超过 16 个转换，或者当 BUFM 位 (ADCON2<1>) 为 0 时，指定每个中断 8 个转换。BUFM 位的功能描述如下。

17.14.3 缓冲器填充模式

当 BUFM 位 (ADCON2<1>) 为 1 时，此 16 字结果缓冲器 (ADRES) 将被拆分成两个 8 字组。每次中断事件发生后，这两个 8 字缓冲器会交替地接收转换结果。BUFM 置位后首先使用位于 ADCBUF 的低地址的 8 字缓冲器。当 BUFM 为 0 时，所有转换过程将使用完整的 16 字缓冲器。

是否使用 BUFM 功能取决于中断后有多少时间可用来移动缓冲器的内容，这是由应用决定的。如果处理器能够在采样和转换一个通道所需的时间内快速地卸载满载的缓冲器，可以将 BUFM 位设置为 0，从而可以在每次中断前进行最多 16 次转换。在第一个缓冲器单元被改写前，处理器会有一段采样和转换时间。

如果在这段采样和转换时间内处理器不能卸载缓冲器，BUFM 位应设置为 1。例如，如果 SMPI<3:0> = 0111，那么 8 个转换结果将被装入半个缓冲器，随后发生中断。紧接着的 8 个转换结果将被装入该缓冲器的另一半。从而处理器在中断之间有整段时间将 8 个转换结果移出缓冲器。

17.14.4 缓冲器填充状态

使用 BUFM 控制位拆分转换结果寄存器时，BUFS 控制位 (ADCON2<7>) 表示 A/D 转换器当前正在填充的半个缓冲器。如果 BUFS = 0，则 A/D 转换器正在填充 ADCBUF0—ADCBUF7，且用户软件应从 ADCBUF8—ADCBUFF 读取转换值。如果 BUFS = 1，情形正好相反，用户软件应从 ADCBUF0—ADCBUF7 读取转换值。

17.15 转换过程示例

下面的配置示例说明了在不同采样和缓冲配置下的 A/D 工作。在各个示例中，将 ASAM 位置 1 会启动自动采样。转换触发结束采样并启动转换。

17.15.1 示例：单个通道的多次采样和转换

图 17-11 和表 17-2 举例说明了 A/D 的基本配置。在这种情况下，一个 A/D 输入（AN0）将由一个采样和保持通道 CH0 采样，并进行转换。转换结果存储在 ADCBUF 缓冲器中。此过程会重复 16 次，直到缓冲器满为止，模块产生中断。然后重复整个过程。

CHPS 位指定了仅采样 / 保持 CH0 是有效的。ALTS 位清零，只有 MUX A 输入有效。CH0SA 位和 CH0NA 位指定 AN0 到 VREF- 为采样 / 保持通道的输入。不使用所有其他输入选择位。

图 17-14: 转换一个通道 16 次 / 中断

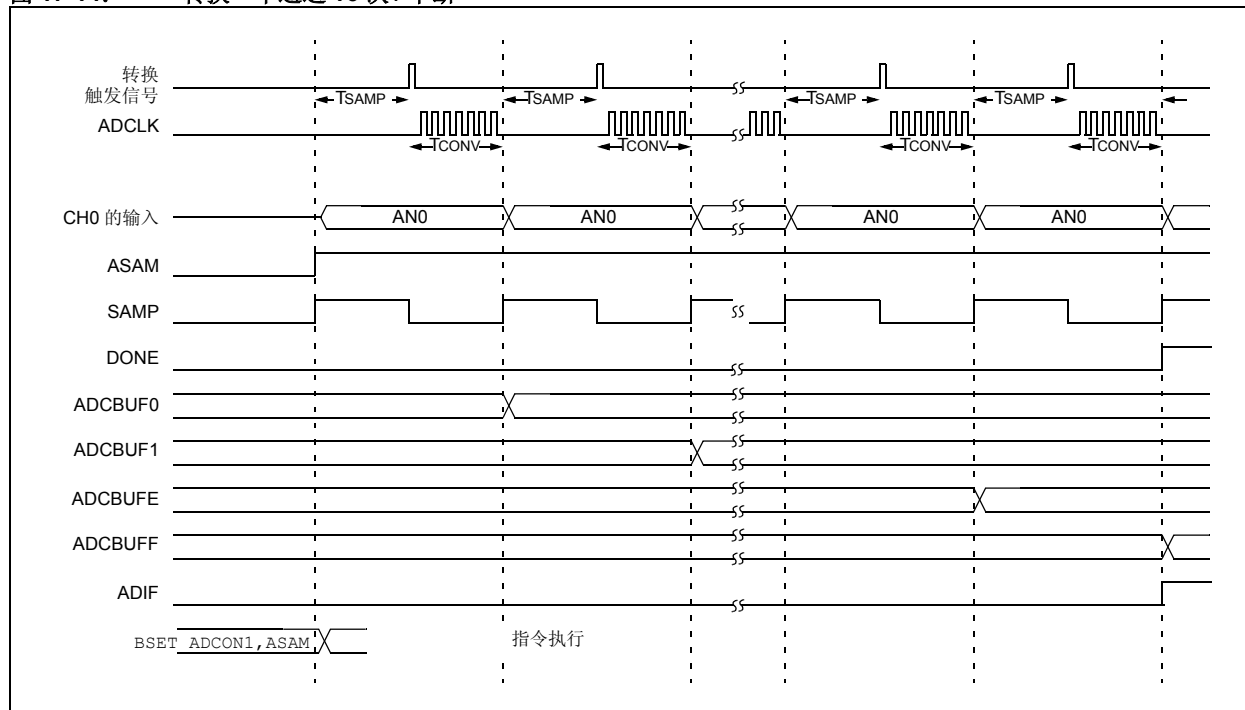


表 17-2: 转换一个通道 16 次 / 中断

控制位 过程选择	
SMPI<2:0> = 1111	第 16 次采样后中断
CHPS<1:0> = 00	采样通道 CH0
SIMSAM = n/a	不适用于单通道采样
BUFM = 0	单个 16 字结果缓冲器
ALTS = 0	总是使用 MUX A 输入选择
MUX A 输入选择	
CH0SA<3:0> = 0000	选择 AN0 作为 CH0+ 输入
CH0NA = 0	选择 VREF- 作为 CH0- 输入
CSCNA = 0	无输入扫描
CSSL<15:0> = n/a	扫描输入选择未使用
CH123SA = n/a	通道 CH1、CH2、CH3 + 输入未使用
CH123NA<1:0> = n/a	通道 CH1、CH2、CH3- 输入未使用
MUX B 输入选择	
CH0SB<3:0> = n/a	通道 CH0+ 输入未使用
CH0NB = n/a	通道 CH0- 输入未使用
CH123SB = n/a	通道 CH1、CH2、CH3 + 输入未使用
CH123NB<1:0> = n/a	通道 CH1、CH2、CH3- 输入未使用

缓冲器 地址	第一次中断 时的缓冲器
ADCBUF0	AN0 采样 1
ADCBUF1	AN0 采样 2
ADCBUF2	AN0 采样 3
ADCBUF3	AN0 采样 4
ADCBUF4	AN0 采样 5
ADCBUF5	AN0 采样 6
ADCBUF6	AN0 采样 7
ADCBUF7	AN0 采样 8
ADCBUF8	AN0 采样 9
ADCBUF9	AN0 采样 10
ADCBUFA	AN0 采样 11
ADCBUFB	AN0 采样 12
ADCBUFC	AN0 采样 13
ADCBUFD	AN0 采样 14
ADCBUFE	AN0 采样 15
ADCBUFF	AN0 采样 16

工作过程	
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0x0
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0x1
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0x2
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0x3
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0x4
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0x5
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0x6
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0x7
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0x8
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0x9
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0xA
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0xB
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0xC
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0xD
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0xE
采样 MUX A 输入: AN0->CH0	
	转换 CH0, 写缓冲器 0xF
	中断
	重复

第二次中断 时的缓冲器
AN0 采样 17
AN0 采样 18
AN0 采样 19
AN0 采样 20
AN0 采样 21
AN0 采样 22
AN0 采样 23
AN0 采样 24
• • •
AN0 采样 25
AN0 采样 26
AN0 采样 27
AN0 采样 28
AN0 采样 29
AN0 采样 30
AN0 采样 31
AN0 采样 32

17.15.2 示例：扫描所有模拟输入时的 A/D 转换

图 17-15 和表 17-3 说明了一个非常典型的设置，其中所有可用的模拟输入通道都由一个采样和保持通道 CH0 采样和转换。置 1CSCNA 位指定对作为 CH0 正输入的 A/D 输入进行扫描。其他情况与第 17.15.1 节类似。

首先，AN0 输入被 CH0 采样，然后转换。转换结果存储在 ADCBUF 缓冲器中。然后 AN1 输入被采样并转换。此扫描输入的过程重复 16 次，直到缓冲器满为止，模块产生中断。然后重复整个过程。

图 17-15: 扫描 16 个输入 / 中断

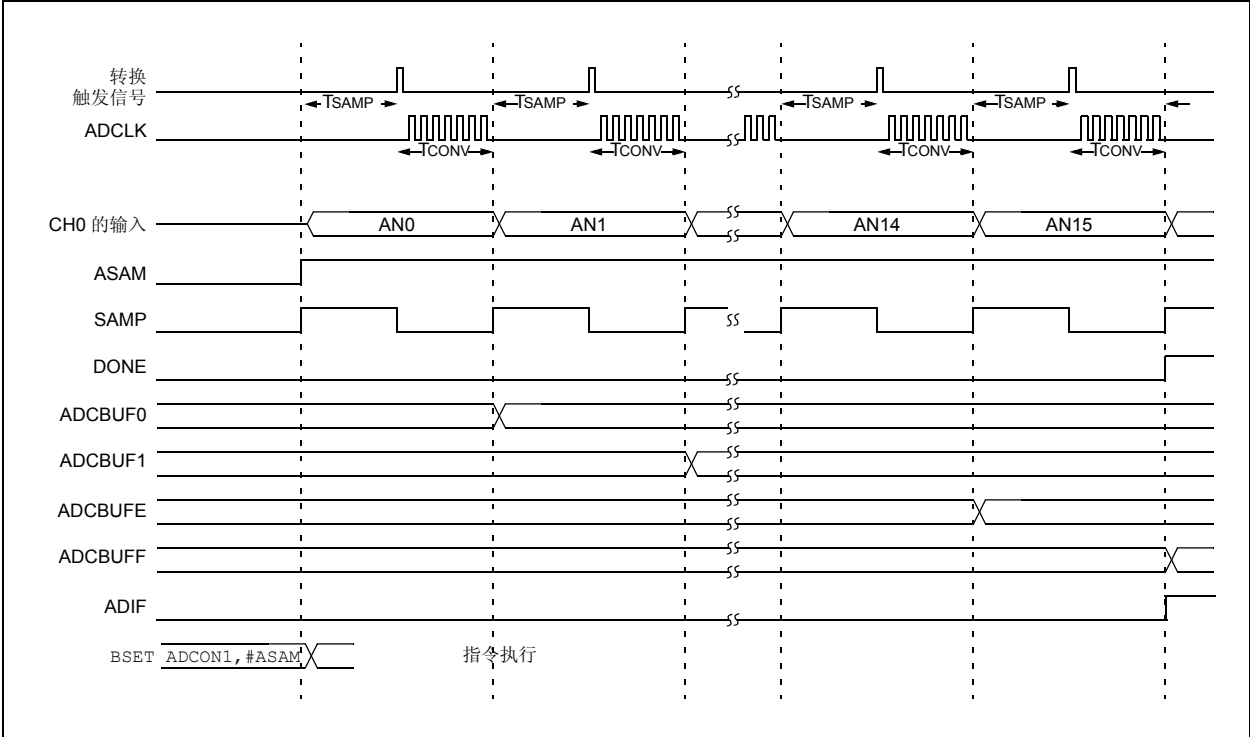


表 17-3: 扫描 16 个输入 / 中断

控制位 过程选择	
SMPI<2:0> = 1111	第 16 次采样后中断
CHPS<1:0> = 00	采样通道 CH0
SIMSAM = n/a	不适用于单通道采样
BUFM = 0	单个 16 字结果缓冲器
ALTS = 0	总是使用 MUX A 输入选择
MUX A 输入选择	
CH0SA<3:0> = n/a	被 CSCNA 覆盖
CH0NA = 0	选择 VREF- 作为 CH0- 输入
CSCNA = 1	扫描 CH0+ 输入
CSSL<15:0> = 1111 1111 1111 1111	扫描输入选择未使用
CH123SA = n/a	通道 CH1、CH2、CH3 + 输入未使用
CH123NA<1:0> = n/a	通道 CH1、CH2、CH3 - 输入未使用
MUX B 输入选择	
CH0SB<3:0> = n/a	通道 CH0+ 输入未使用
CH0NB = n/a	通道 CH0 - 输入未使用
CH123SB = n/a	通道 CH1、CH2、CH3 + 输入未使用
CH123NB<1:0> = n/a	通道 CH1、CH2、CH3 - 输入未使用

缓冲器 地址	第一次中断 时的缓冲器
ADCBUF0	AN0 采样 1
ADCBUF1	AN1 采样 2
ADCBUF2	AN2 采样 3
ADCBUF3	AN3 采样 4
ADCBUF4	AN4 采样 5
ADCBUF5	AN5 采样 6
ADCBUF6	AN6 采样 7
ADCBUF7	AN7 采样 8
ADCBUF8	AN8 采样 9
ADCBUF9	AN9 采样 10
ADCBUFA	AN10 采样 11
ADCBUFB	AN11 采样 12
ADCBUFC	AN12 采样 13
ADCBUFD	AN13 采样 14
ADCBUFE	AN14 采样 15
ADCBUFF	AN15 采样 16

工作过程	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0x0	
采样 MUX A 输入: AN1->CH0	
转换 CH0, 写缓冲器 0x1	
采样 MUX A 输入: AN2->CH0	
转换 CH0, 写缓冲器 0x2	
采样 MUX A 输入: AN3->CH0	
转换 CH0, 写缓冲器 0x3	
采样 MUX A 输入: AN4->CH0	
转换 CH0, 写缓冲器 0x4	
采样 MUX A 输入: AN5->CH0	
转换 CH0, 写缓冲器 0x5	
采样 MUX A 输入: AN6->CH0	
转换 CH0, 写缓冲器 0x6	
采样 MUX A 输入: AN7->CH0	
转换 CH0, 写缓冲器 0x7	
采样 MUX A 输入: AN8->CH0	
转换 CH0, 写缓冲器 0x8	
采样 MUX A 输入: AN9->CH0	
转换 CH0, 写缓冲器 0x9	
采样 MUX A 输入: AN10->CH0	
转换 CH0, 写缓冲器 0xA	
采样 MUX A 输入: AN11->CH0	
转换 CH0, 写缓冲器 0xB	
采样 MUX A 输入: AN12->CH0	
转换 CH0, 写缓冲器 0xC	
采样 MUX A 输入: AN13->CH0	
转换 CH0, 写缓冲器 0xD	
采样 MUX A 输入: AN14->CH0	
转换 CH0, 写缓冲器 0xE	
采样 MUX A 输入: AN15->CH0	
转换 CH0, 写缓冲器 0xF	
中断	
重复	

第二次中断 时的缓冲器
AN0 采样 17
AN1 采样 18
AN2 采样 19
AN3 采样 20
AN4 采样 21
AN5 采样 22
AN6 采样 23
AN7 采样 24
AN8 采样 25
AN9 采样 26
AN10 采样 27
AN11 采样 28
AN12 采样 29
AN13 采样 30
AN14 采样 31
AN15 采样 32

dsPIC30F 系列参考手册

表 17-4: 转换三个输入，四次四个输入、每个中断一次

[illegible]

缓冲器地址	第一次中断时的缓冲器	第二次中断时的缓冲器
ADCBUF0	AN4 采样 1	AN4 采样 5
ADCBUF1	AN0 采样 1	AN0 采样 5
ADCBUF2	AN1 采样 1	AN1 采样 5
ADCBUF3	AN2 采样 1	AN2 采样 5
ADCBUF4	AN5 采样 2	AN5 采样 6
ADCBUF5	AN0 采样 2	AN0 采样 6
ADCBUF6	AN1 采样 2	AN1 采样 6
ADCBUF7	AN2 采样 2	AN2 采样 6
ADCBUF8	AN6 采样 3	AN6 采样 7
ADCBUF9	AN0 采样 3	AN0 采样 7
ADCBUFA	AN1 采样 3	AN1 采样 7
ADCBUFB	AN2 采样 3	AN2 采样 7
ADCBUFC	AN7 采样 4	AN7 采样 8
ADCBUFD	AN0 采样 4	AN0 采样 8
ADCBUFE	AN1 采样 4	AN1 采样 8
ADCBUFF	AN2 采样 4	AN2 采样 8

17.15.4 示例：使用双 8 字缓冲器

图 17-17 和表 17-5 说明了双 8 字缓冲器的使用和交替填充缓冲器。将 BUFM 位置位使能双 8 字缓冲器。BUFM 的设置不影响其他工作参数。首先，转换过程从 ADCBUF0（缓冲器单元 0x0）开始填充缓冲器。第一次中断发生后，缓冲器从 ADCBUF8（缓冲器单元 0x8）开始填充。每次中断后，BUFS 状态位交替置位和清零。在本示例中，所有四个通道同时采样，且每次采样后发生一次中断。

图 17-17： 使用双 8 字缓冲器转换四个输入，每中断一次

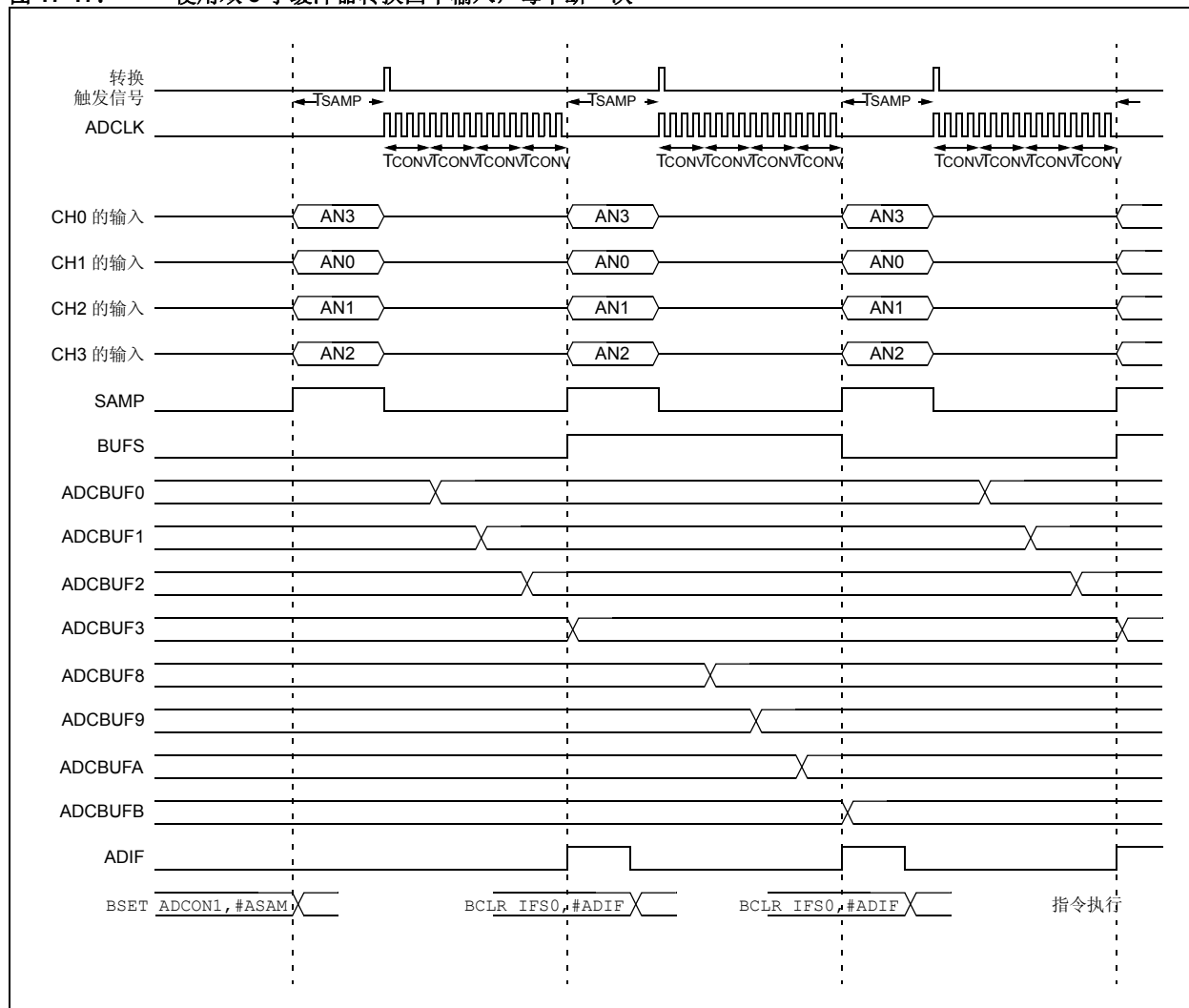


表 17-5: 使用双 8 字缓冲器转换四个输入，每中断一次

控制位 过程选择	工作过程
SMPI<2:0> = 0000 每次采样后中断	采样 MUX A 输入： AN3 -> CH0, AN0 -> CH1, AN1 -> CH2, AN2 -> CH3
CHPS<1:0> = 1x 采样通道 CH1, CH2, CH3, CH0	转换 CH0, 写缓冲器 0x0
SIMSAM = 1 同时采样所有通道	转换 CH1, 写缓冲器 0x1
BUFM = 1 双 8 字结果缓冲器	转换 CH2, 写缓冲器 0x2
ALTS = 0 总是使用 MUX A 输入选择	转换 CH3, 写缓冲器 0x3
MUX A 输入选择	中断; 改变缓冲器
CH0SA<3:0> = 0011 选择 AN3 作为 CH0+ 输入	采样 MUX A 输入： AN3 -> CH0, AN0 -> CH1, AN1 -> CH2, AN2 -> CH3
CH0NA = 0 选择 VREF- 作为 CH0- 输入	转换 CH0, 写缓冲器 0x8
CSCNA = 0 无输入扫描	转换 CH1, 写缓冲器 0x9
CSSL<15:0> = n/a 扫描输入选择未使用	转换 CH2, 写缓冲器 0xA
CH123SA = 0 CH1+ = AN0, CH2+ = AN1, CH3+ = AN2	转换 CH3, 写缓冲器 0xB
CH123NA<1:0> = 0x CH1-, CH2-, CH3- = VREF-	中断; 改变缓冲器
MUX B 输入选择	重复
CH0SB<3:0> = n/a 通道 CH0+ 输入未使用	
CH0NB = n/a 通道 CH0- 输入未使用	
CH123SB = n/a 通道 CH1、CH2、CH3 + 输入未使用	
CH123NB<1:0> = n/a 通道 CH1、CH2、CH3- 输入未使用	

缓冲器 地址	第一次中断 时的缓冲器	第二次中断 时的缓冲器
ADCBUF0	AN3 采样 1	
ADCBUF1	AN0 采样 1	
ADCBUF2	AN1 采样 1	
ADCBUF3	AN2 采样 1	
ADCBUF4		
ADCBUF5		
ADCBUF6		
ADCBUF7		
ADCBUF8		AN3 采样 2
ADCBUF9		AN0 采样 2
ADCBUFA		AN1 采样 2
ADCBUFB		AN2 采样 2
ADCBUFC		
ADCBUFD		
ADCBUFE		
ADCBUFF		

• • •

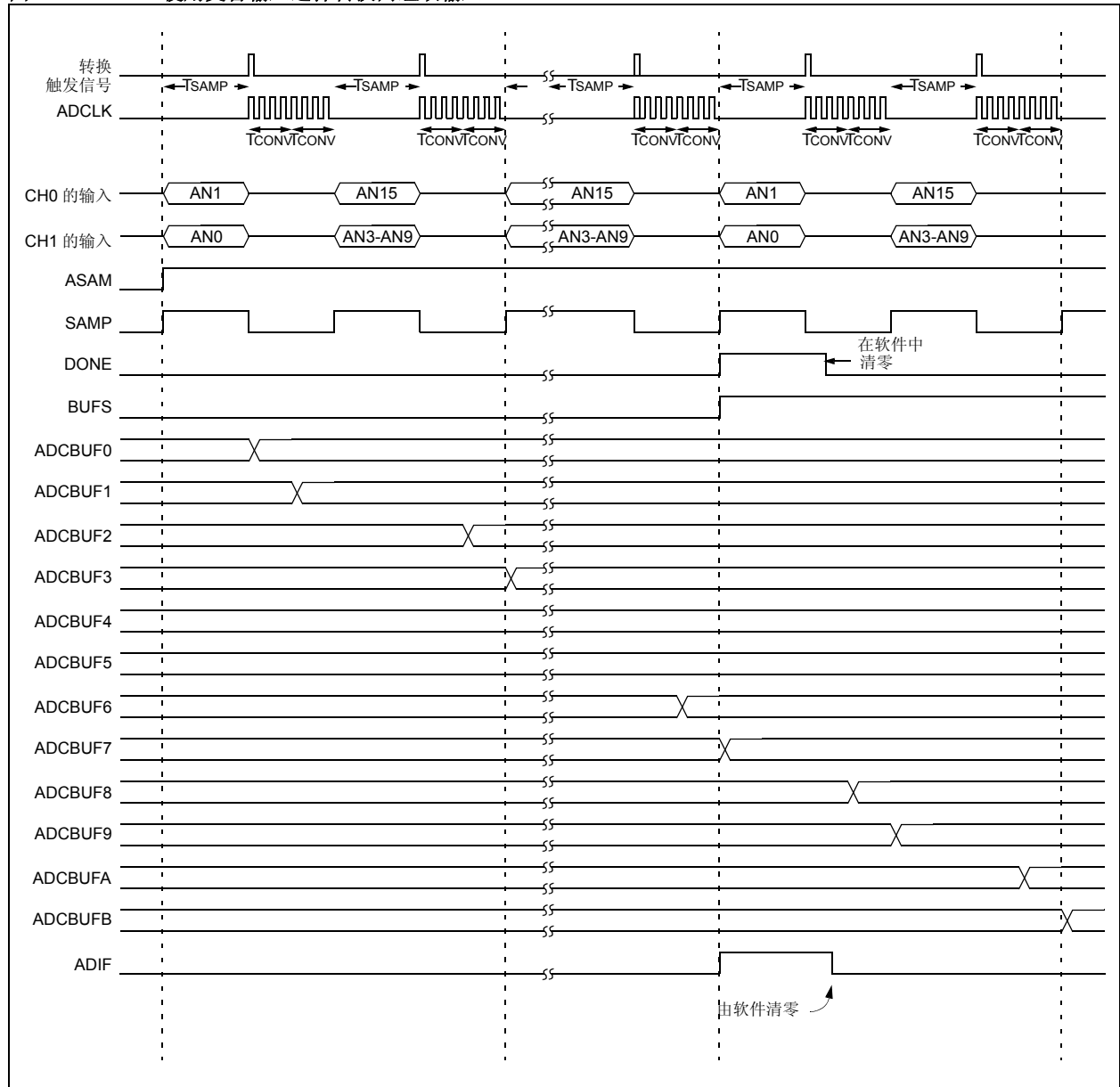
17.15.5 示例：使用交替多路开关 A、多路开关 B 输入选择

图 17-18 和表 17-6 说明了对分配给 MUX A 和 MUX B 的输入进行交替采样。在本例中，使能两个通道同时采样。将 **ALTS** 位置位，交替输入选择。第一次采样使用由 **CH0SA**、**CH0NA**、**CHXSA** 和 **CHXNA** 位指定的 MUX A 输入。下一次采样使用由 **CH0SB**、**CH0NB**、**CHXSB** 和 **CHXNB** 位指定的 MUX B 输入。在本示例中，MUX B 输入规范之一是使用两个模拟输入作为采样 / 保持的差分源，采样（**AN3—AN9**）。

本示例还说明了双 8 字缓冲器的使用。每四次采样后发生一次中断，每次中断时将 8 个字载入到缓冲器。

注意使用 4 个不用交替输入选择的采样 / 保持通道将产生与本示例（使用两个交替输入选择的通道）相同数量的转换。然而，**CH1**、**CH2** 和 **CH3** 通道在模拟输入的选择性上更为有限，所以本示例的方法为输入选择提供了比使用 4 个通道方式更高的灵活性。

图 17-18: 使用交替输入选择转换两组双输入



17.15.6 示例：使用同时采样对八个输入进行采样

第 17.15.6 节和第 17.15.7 节给出了几乎相同的设置，不同之处在于，第 17.15.6 节使用同时采样（SIMSAM = 1），而第 17.15.7 节使用顺序采样（SIMSAM = 0）。两个示例都使用交替输入并为采样 / 保持指定了差分输入。

图 17-19 和表 17-7 说明了同时采样。当转换多个的通道并选择了同时采样时，该模块会采样所有通道，然后依次执行要求的转换。在本示例中，ASAM 位置 1，转换完成后开始采样。

图 17-19： 使用同时采样对八个输入进行采样

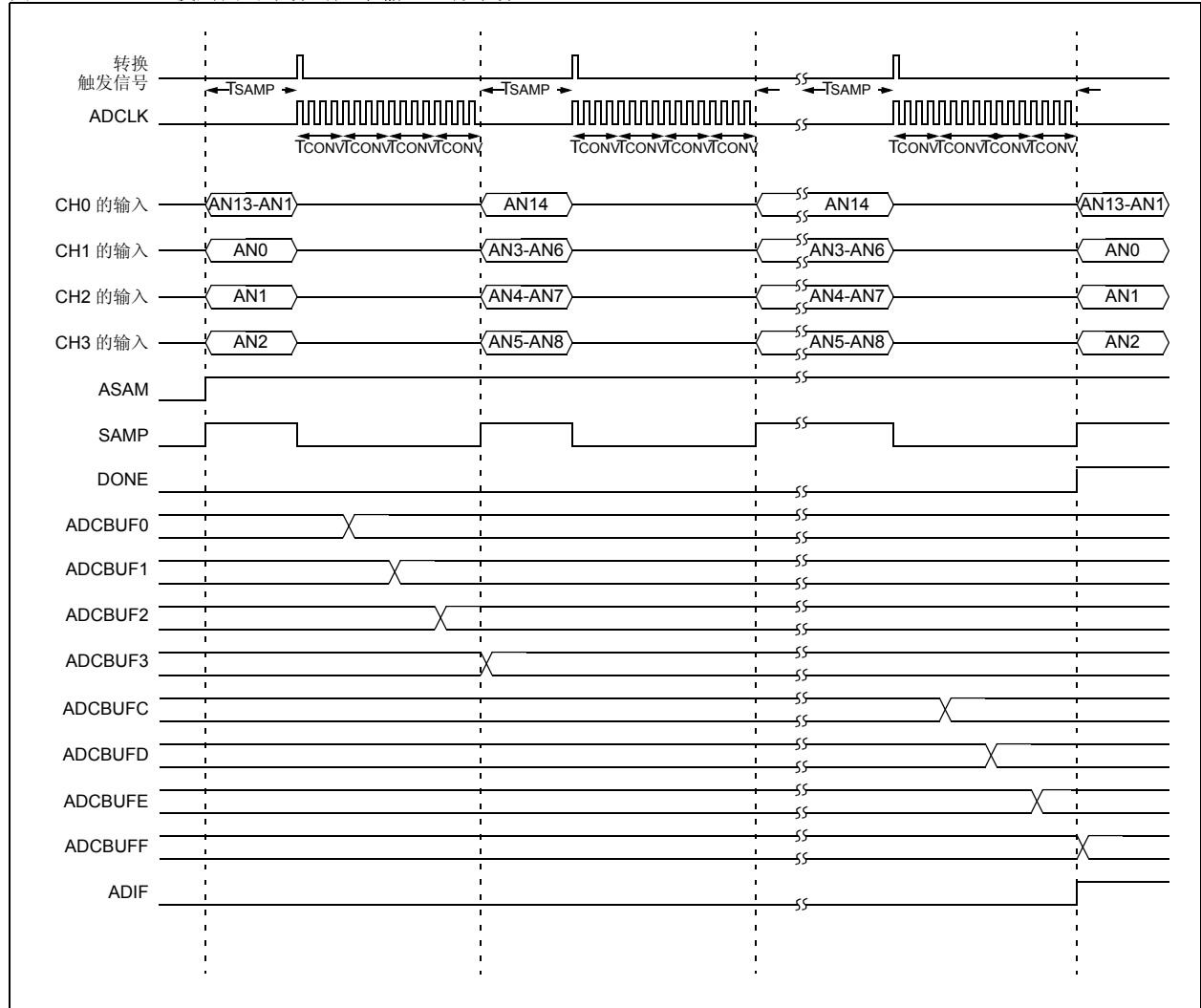


表 17-7: 使用同时采样对八个输入进行采样

控制位	过程选择
SMPI<2:0> = 0011	第 4 次采样时中断
CHPS<1:0> = 1x	采样通道 CH0, CH1, CH2, CH3
SIMSAM = 1	同时采样所有通道
BUFM = 0	单个 16 字结果缓冲器
ALTS = 1	交替 MUX A/MUX B 输入选择
MUX A 输入选择	
CH0SA<3:0> = 1101	选择 AN13 作为 CH0+ 输入
CH0NA = 1	选择 AN1 作为 CH0- 输入
CSCNA = 0	无输入扫描
CSSL<15:0> = n/a	扫描输入选择未使用
CH123SA = 0	CH1+ = AN0, CH2+ = AN1, CH3+ = AN2
CH123NA<1:0> = 0x	CH1-, CH2-, CH3- = VREF-
MUX B 输入选择	
CH0SB<3:0> = 1110	选择 AN14 作为 CH0+ 输入
CH0NB = 0	选择 VREF- 作为 CH0- 输入
CH123SB = 1	CH1+ = AN3, CH2+ = AN4, CH3+ = AN5
CH123NB<1:0> = 10	CH1- = AN6, CH2- = AN7, CH3- = AN8

工作过程

采样 MUX A 输入:
(AN13—AN1) -> CH0, AN0 -> CH1, AN1 -> CH2, AN2 -> CH3
转换 CH0, 写缓冲器 0x0
转换 CH1, 写缓冲器 0x1
转换 CH2, 写缓冲器 0x2
转换 CH3, 写缓冲器 0x3
采样 MUX B 输入:
AN14 -> CH0,
(AN3—AN6) -> CH1, (AN4-AN7) -> CH2, (AN5—AN8) -> CH3
转换 CH0, 写缓冲器 0x4
转换 CH1, 写缓冲器 0x5
转换 CH2, 写缓冲器 0x6
转换 CH3, 写缓冲器 0x7
采样 MUX A 输入:
(AN13—AN1) -> CH0, AN0 -> CH1, AN1 -> CH2, AN2 -> CH3
转换 CH0, 写缓冲器 0x8
转换 CH1, 写缓冲器 0x9
转换 CH2, 写缓冲器 0xA
转换 CH3, 写缓冲器 0xB
采样 MUX B 输入:
AN14 -> CH0,
(AN3—AN6) -> CH1, (AN4-AN7) -> CH2, (AN5—AN8) -> CH3
转换 CH0, 写缓冲器 0xC
转换 CH1, 写缓冲器 0xD
转换 CH2, 写缓冲器 0xE
转换 CH3, 写缓冲器 0xF
中断
重复

缓冲器地址

ADCBUF0
ADCBUF1
ADCBUF2
ADCBUF3
ADCBUF4
ADCBUF5
ADCBUF6
ADCBUF7
ADCBUF8
ADCBUF9
ADCBUFA
ADCBUFB
ADCBUFC
ADCBUFD
ADCBUFE
ADCBUFF

第一次中断时的缓冲器

(AN13—AN1) 采样 1
AN0 采样 1
AN1 采样 1
AN2 采样 1
AN14 采样 2
(AN3—AN6) 采样 2
(AN4—AN7) 采样 2
(AN5—AN8) 采样 2
(AN13—AN1) 采样 3
AN0 采样 3
AN1 采样 3
AN2 采样 3
AN14 采样 4
(AN3—AN6) 采样 4
(AN4—AN7) 采样 4
(AN5—AN8) 采样 4

第二次中断时的缓冲器

(AN13—AN1) 采样 5
AN0 采样 5
AN1 采样 5
AN2 采样 5
AN14 采样 6
(AN3—AN6) 采样 6
(AN4—AN7) 采样 6
(AN5—AN8) 采样 6
(AN13—AN1) 采样 7
AN0 采样 7
AN1 采样 7
AN2 采样 7
AN14 采样 8
(AN3—AN6) 采样 8
(AN4—AN7) 采样 8
(AN5—AN8) 采样 8

• • •

17.15.7 示例：使用顺序采样对八个输入进行采样

图 17-20 和表 17-8 给出了顺序采样。当转换多个的通道并选择了顺序采样时，一旦时机出现，该模块便会开始采样一个通道，然后依次执行要求的转换。在本例中，ASAM 位置位，该通道转换完成后将开始采样下一个通道。

当 ASAM 清零时，转换完成后将不会继续采样，但当 SAMP 位置位时会继续采样。

当使用多个的通道时，顺序采样提供了更多的采样时间，因为一个通道可以在另一个通道发生转换时被采样。

图 17-20： 使用顺序采样对八个输入进行采样

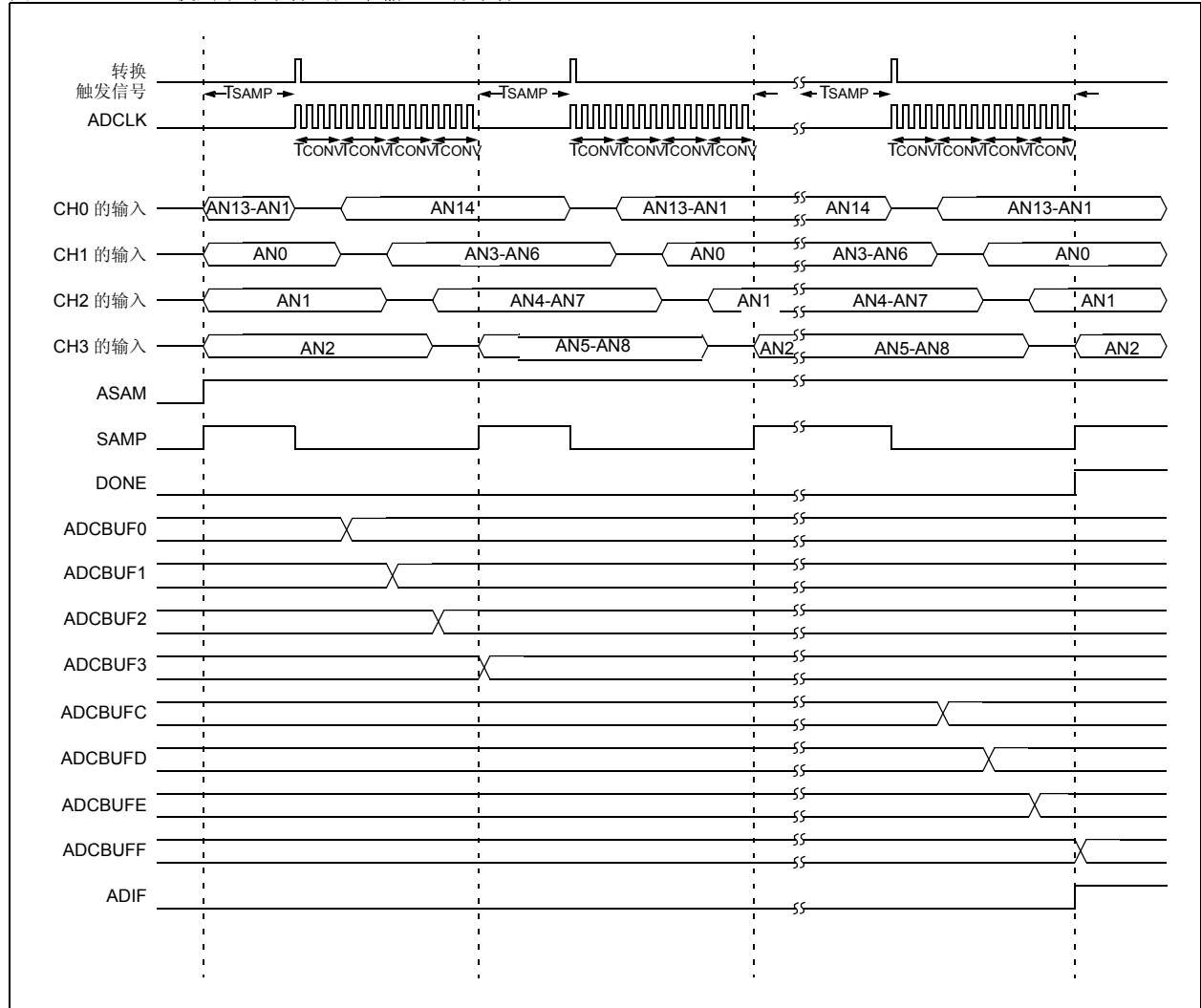


表 17-8: 使用顺序采样对八个输入进行采样

控制位 过程选择		工作过程	
SMPI<2:0> = 1111 第 16 次采样时中断		采样: (AN13—AN1) -> CH0 转换 CH0, 写缓冲器 0x0	
CHPS<1:0> = 1x 采样通道 CH0, CH1, CH2, CH3		采样: AN0 -> CH1 转换 CH1, 写缓冲器 0x1	
SIMSAM = 0 顺序采样所有通道		采样: AN1 -> CH2 转换 CH2, 写缓冲器 0x2	
BUFM = 0 单个 16 字结果缓冲器		采样: AN2 -> CH3 转换 CH3, 写缓冲器 0x3	
ALTS = 1 交替 MUX A/B 输入选择		采样: AN14 -> CH0 转换 CH0, 写缓冲器 0x4	
MUX A 输入选择		采样: (AN3—AN6) -> CH1 转换 CH1, 写缓冲器 0x5	
CH0SA<3:0> = 0110 选择 AN6 作为 CH0+ 输入		采样: (AN4—AN7) -> CH2 转换 CH2, 写缓冲器 0x6	
CH0NA = 0 选择 VREF- 作为 CH0- 输入		采样: (AN5—AN8) -> CH3 转换 CH3, 写缓冲器 0x7	
CSCNA = 0 无输入扫描		采样: (AN13—AN1) -> CH0 转换 CH0, 写缓冲器 0x8	
CSSL<15:0> = n/a 扫描输入选择未使用		采样: AN0 -> CH1 转换 CH1, 写缓冲器 0x9	
CH123SA = 0 CH1+ = AN0, CH2+ = AN1, CH3+ = AN2		采样: AN1 -> CH2 转换 CH2, 写缓冲器 0xA	
CH123NA<1:0> = 0x CH1-, CH2-, CH3- = VREF-		采样: AN2 -> CH3 转换 CH3, 写缓冲器 0xB	
MUX B 输入选择		采样: AN14 -> CH0 转换 CH0, 写缓冲器 0xC	
CH0SB<3:0> = 0111 选择 AN7 作为 CH0+ 输入		采样: (AN3—AN6) -> CH1 转换 CH1, 写缓冲器 0xD	
CH0NB = 0 选择 VREF- 作为 CH0- 输入		采样: (AN4—AN7) -> CH2 转换 CH2, 写缓冲器 0xE	
CH123SB = 1 CH1+ = AN3, CH2+ = AN4, CH3+ = AN5		采样: (AN5—AN8) -> CH3 转换 CH3, 写缓冲器 0xF	
CH123NB<1:0>=0x CH1-, CH2-, CH3- = VREF-		中断	
		重复	

缓冲器 地址	第一次中断 时的缓冲器	第二次中断 时的缓冲器
ADCBUF0	(AN13—AN1) 采样 1	(AN13—AN1) 采样 17
ADCBUF1	AN0 采样 2	AN0 采样 18
ADCBUF2	AN1 采样 3	AN1 采样 19
ADCBUF3	AN2 采样 4	AN2 采样 20
ADCBUF4	AN14 采样 5	AN14 采样 21
ADCBUF5	(AN3—AN6) 采样 6	(AN3—AN6) 采样 22
ADCBUF6	(AN4—AN7) 采样 7	(AN4—AN7) 采样 23
ADCBUF7	(AN5—AN8) 采样 8	(AN5—AN8) 采样 24
ADCBUF8	(AN13—AN1) 采样 9	(AN13—AN1) 采样 25
ADCBUF9	AN0 采样 10	AN0 采样 26
ADCBUFA	AN1 采样 11	AN1 采样 27
ADCBUFB	AN2 采样 12	AN2 采样 28
ADCBUFC	AN14 采样 13	AN14 采样 29
ADCBUFD	(AN3—AN6) 采样 14	(AN3—AN6) 采样 30
ADCBUFE	(AN4—AN7) 采样 15	(AN4—AN7) 采样 31
ADCBUFF	(AN5—AN8) 采样 16	(AN5—AN8) 采样 32

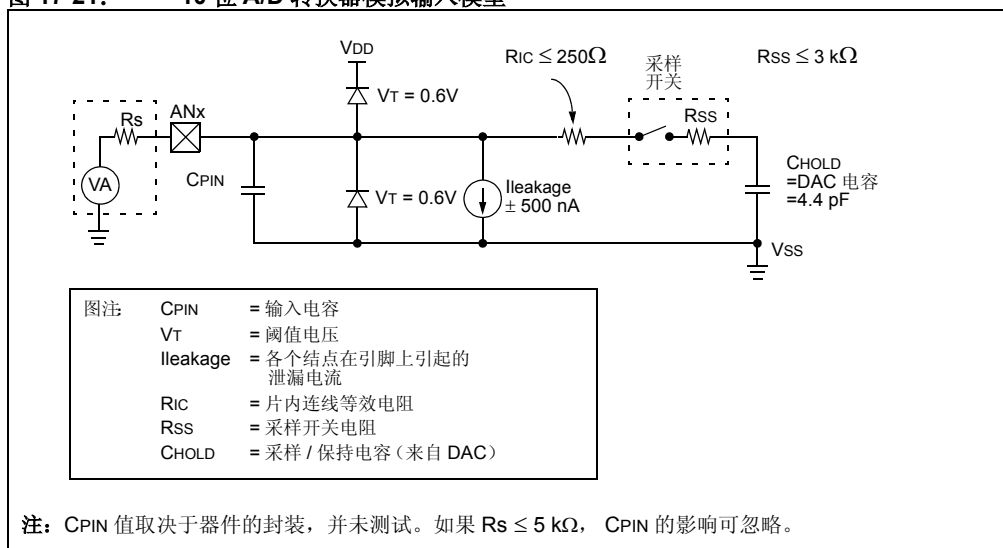
17.16 A/D 采样要求

图 17-21 给出了 10 位 A/D 转换器的模拟输入模型。A/D 的总采样时间是内部放大器稳定时间和保持电容充电时间的函数。

为了使 A/D 转换器达到规定的精度，必须让充电保持电容（CHOLD）充分充电至模拟输入引脚上的电平。模拟输出源阻抗（ R_s ）、内部连线等效阻抗（ R_{ic} ）和内部采样开关（ R_{ss} ）阻抗共同地直接影响电容 CHOLD 充电所需的时间，因此合并的源阻抗应足够小，以便在选择的采样时间内对保持电容充分充电。为了把引脚泄漏电流对 A/D 转换器精度的影响降到最低，最大建议源阻抗 R_s 应为 $5\text{ k}\Omega$ （转换速率最高为 500 ksp/s 时），或最大 500Ω （转换速率最高为 1 Msps 时）。当选择（改变）了模拟输入通道时，采样工作必须在启动转换前完成。在每次采样工作前，内部保持电容将处于放电状态。

两次转换之间应该至少留出 1 TAD 的时间周期作为采样时间。更多详细信息，参见器件电气规范。

图 17-21: 10 位 A/D 转换器模拟输入模型



17.17 读取 A/D 转换结果缓冲器

该 RAM 为 10 位宽，当对该缓冲器执行读操作时，数据自动格式化为四种可选格式中的一种。FORM<1:0> 位（ADCON1<9:8>）用来选择格式。格式化硬件在数据总线上为所有数据格式提供一个 16 位结果。图 17-22 给出了使用 FORM<1:0> 控制位选择的数据输出格式。

图 17-22: A/D 输出数据格式

RAM 内容:							d09	d08	d07	d06	d05	d04	d03	d02	d01	d00
读出到总线:																
整数	0	0	0	0	0	0	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00
有符号整数	$\overline{\text{d09}}$	$\overline{\text{d09}}$	$\overline{\text{d09}}$	$\overline{\text{d09}}$	$\overline{\text{d09}}$	$\overline{\text{d09}}$	$\overline{\text{d09}}$	d08	d07	d06	d05	d04	d03	d02	d01	d00
小数 (1.15)	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0
有符号小数 (1.15)	$\overline{\text{d09}}$	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0

图 17-23: 不同结果代码的等价数值

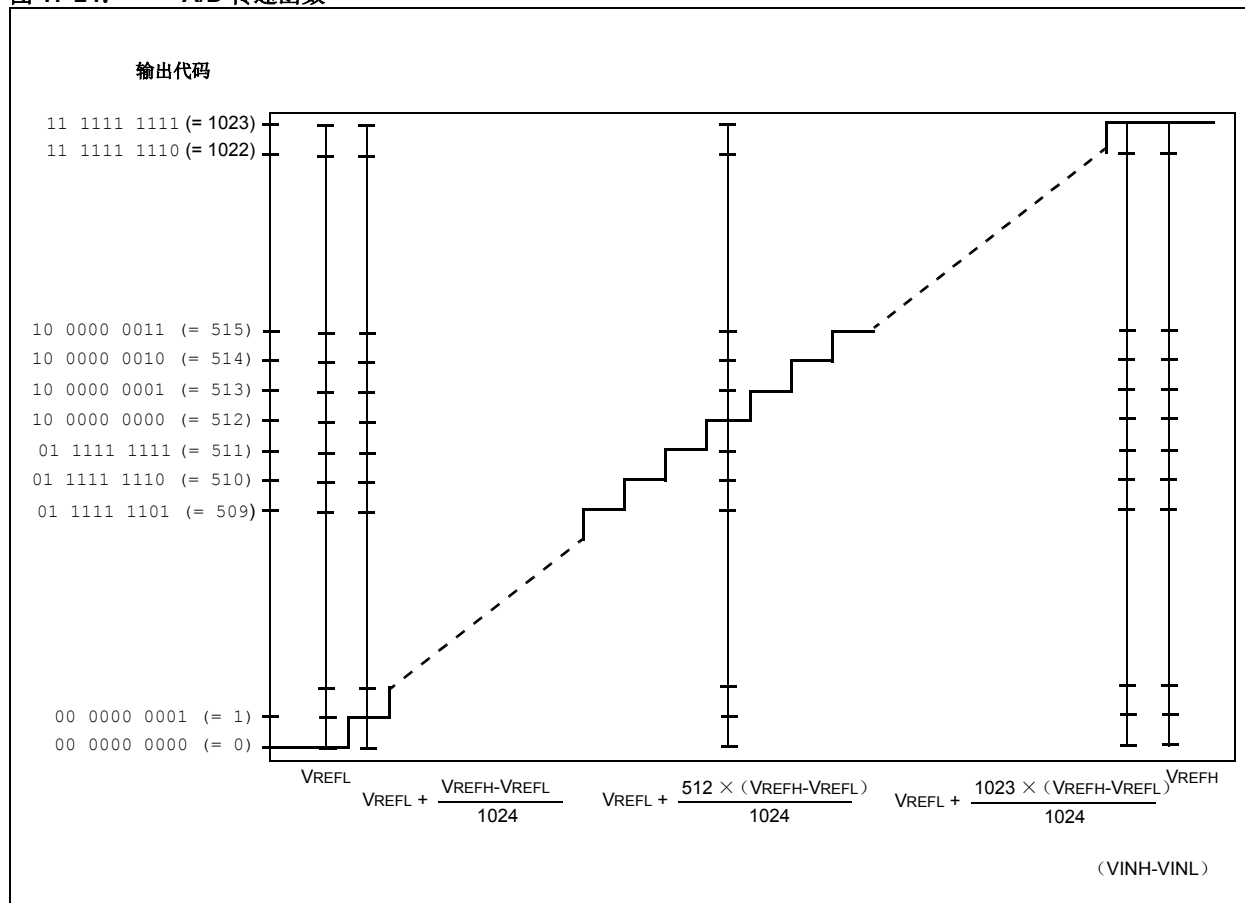
VIN/VREF	10 位 输出码	16 位整数格式	16 位有符号 整数格式	16 位小数格式	16 位有符号 小数格式
1023/1024	11 1111 1111	0000 0011 1111 1111 = 1023	0000 0001 1111 1111 = 511	1111 1111 1100 0000 = 0.999	0111 1111 1100 0000 = 0.499
1022/1024	11 1111 1110	0000 0011 1111 1110 = 1022	0000 0001 1111 1110 = 510	1111 1111 1000 0000 = 0.998	0111 1111 1000 0000 = 0.498
...					
513/1024	10 0000 0001	0000 0010 0000 0001 = 513	0000 0000 0000 0001 = 1	1000 0000 0100 0000 = 0.501	0 000 0000 0100 0000 = 0.001
512/1024	10 0000 0000	0000 0010 0000 0000 = 512	0000 0000 0000 0000 = 0	1000 0000 0000 0000 = 0.500	0000 0000 0000 0000 = 0.000
511/1024	01 1111 1111	0000 0001 1111 1111 = 511	1111 1111 1111 1111 = -1	0111 1111 1100 0000 = .499	1111 1111 1100 0000 = -0.001
...					
1/1024	00 0000 0001	0000 0000 0000 0001 = 1	1111 1110 0000 0001 = -511	0000 0000 0100 0000 = 0.001	1000 0000 0100 0000 = -0.499
0/1024	00 0000 0000	0000 0000 0000 0000 = 0	1111 1110 0000 0000 = -512	0000 0000 0000 0000 = 0.000	1000 0000 0000 0000 = -0.500

17.18 传递函数

图 17-24 给出了 A/D 转换器的理想传递函数。输入电压差值 ($V_{INH} - V_{INL}$) 与参考电压差值 ($V_{REFH} - V_{REFL}$) 作比较。

- 当输入电压为 $(V_{REFH} - V_{REFL})/2048$ 或 0.5 LSb 时发生第一个代码转换。
- 代码 00 0000 0001 被置于 $(V_{REFH} - V_{REFL})/1024$ 或 1.0 LSb 中央。
- 代码 10 0000 0000 被置于 $(512 \times (V_{REFH} - V_{REFL})/1024)$ 中央。
- 小于 $(1 \times (V_{REFH} - V_{REFL})/2048)$ 的输入电压被转换为 00 0000 0000。
- 大于 $(2045 \times (V_{REFH} - V_{REFL})/2048)$ 的输入被转换为 11 1111 1111。

图 17-24: A/D 传递函数



17.19 A/D 转换的精度 / 误差

如要获取讨论 A/D 精度的文档列表，请参见第 17.27 节“相关应用笔记”。

17.20 连接注意事项

因为模拟输入使用 ESD 保护，它们通过二极管连接到 V_{DD} 和 V_{SS} 。这就要求模拟输入电压必须介于 V_{DD} 和 V_{SS} 之间。如果输入电压超出此范围 $0.3V$ 以上（任一方向上），就会有一个二极管正偏，若超过输入电流规范就可能损坏器件。

有时候，可以通过外接 RC 滤波器来对输入信号进行抗混叠滤波。应选择合适的 R 元件，以确保达到采样时间要求。任何通过高阻抗连接到模拟输入引脚上的外部元件（如电容和齐纳二极管等），在引脚上的泄漏电流都应极小。

17.21 初始化

例 17-7 为 A/D 模块的简单初始化代码示例。

在这个特定的配置中，所有 16 个模拟输入引脚（AN0 到 AN15）都被设置为模拟输入。空闲模式下的工作被禁止，输出数据为无符号小数格式，而且 AVDD 和 AVSS 被用作 VREFH 和 VREFL。采样的启动和转换（转换触发信号）的启动通过软件手动执行。CH0 S/H 放大器用于转换。禁止对输入扫描，每个采样 / 转换过程（1 个转换结果）后产生一次中断。A/D 转换时钟为 Tcy/2。

因为在每次转换完成后通过将 SAMP 位（ADCON1<1>）置位手动启动转换，所以自动采样时间位 SAMC<4:0>（ADCON3<12:8>）被忽略。此外，因为转换的启动（即，采样的结束）也是手动触发的，所以每次需要转换新的采样时要清零 SAMP 位。

例 17-7: A/D 初始化代码示例

```

CLR      ADPCFG          ; Configure A/D port,
                        ; all input pins are analog

MOV      #0x2208,W0
MOV      W0,ADCON1       ; Configure sample clock source
                        ; and conversion trigger mode.
                        ;   Unsigned Fractional format,
                        ;   Manual conversion trigger,
                        ;   Manual start of sampling,
                        ;   Simultaneous sampling,
                        ;   No operation in IDLE mode.

CLR      ADCON2          ; Configure A/D voltage reference
                        ; and buffer fill modes.
                        ;   VREF from AVDD and AVSS,
                        ;   Inputs are not scanned,
                        ;   1 S/H channel used,
                        ;   Interrupt every sample

CLR      ADCON3          ; Configure A/D conversion clock

CLR      ADCHS           ; Configure input channels,
                        ;   CH0+ input is AN0.
                        ;   CH0- input is VREFL (AVss)

CLR      ADCSSL          ; No inputs are scanned.

BCLR     IFS0,#ADIF      ; Clear A/D conversion interrupt

; Configure A/D interrupt priority bits (ADIP<2:0>) here, if
; required. (default priority level is 4)

BSET     IEC0,#ADIE      ; Enable A/D conversion interrupt

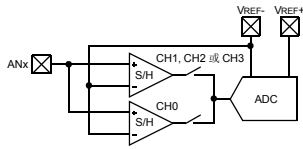
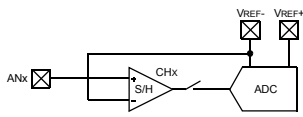
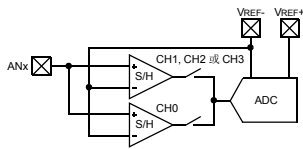
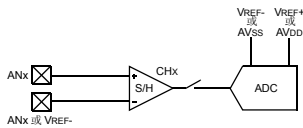
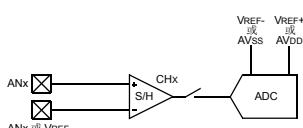
BSET     ADCON1,#ADON     ; Turn on A/D
BSET     ADCON1,#SAMP     ; Start sampling the input
CALL     DELAY            ; Ensure the correct sampling time has
                        ; elapsed before starting conversion.

BCLR     ADCON1,#SAMP     ; End A/D Sampling and start Conversion
:           ; The DONE bit is set by hardware when
:           ; the convert sequence is finished
:           ; The ADIF bit will be set.
    
```

17.22 A/D 转换速度

dsPIC30F 10 位 A/D 转换器规范允许的最大采样速率为 1 Msps。下表对 dsPIC30F 10 位 A/D 转换器的转换速度和所需的工作条件作了小结。

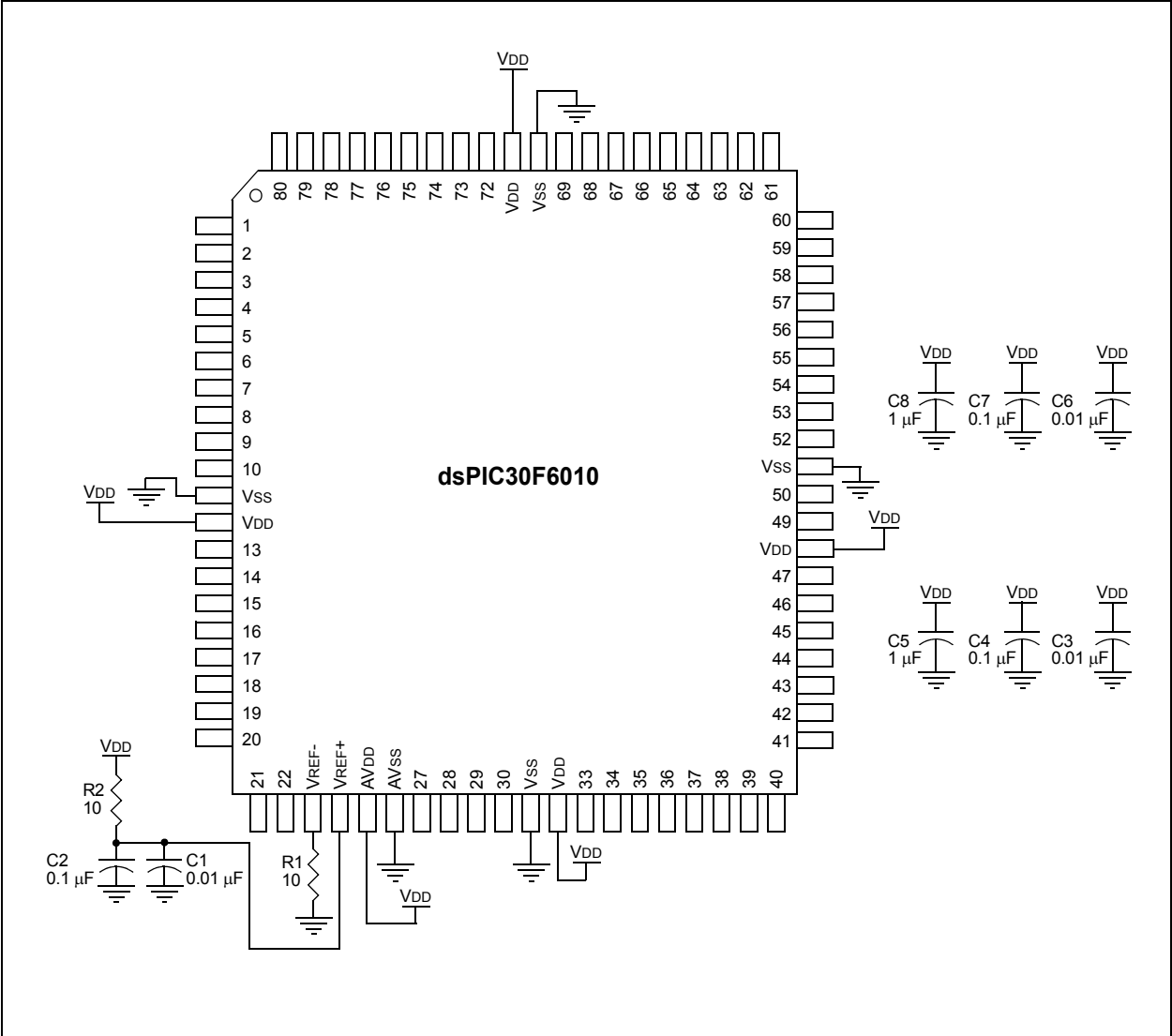
表 17-9: 10 位转换速率参数

dsPIC30F 10 位 A/D 转换器转换速率						
A/D 速率	最小 T _{AD}	最小采样时间	最大 R _s	V _{DD}	温度	A/D 通道配置
小于等于 1 Msps ⁽¹⁾	83.33 ns	12 T _{AD}	500 Ω	4.5V~5.5V	-40°C~+85°C	
小于等于 750 ksps ⁽¹⁾	95.24 ns	2 T _{AD}	500 Ω	4.5V~5.5V	-40°C~+85°C	
小于等于 600 ksps ⁽¹⁾	138.89 ns	12 T _{AD}	500 Ω	3.0V~5.5V	-40°C~+125°C	
小于等于 500 ksps	153.85 ns	1 T _{AD}	5.0 kΩ	4.5V~5.5V	-40°C~+125°C	
小于等于 300 ksps	256.41 ns	1 T _{AD}	5.0 kΩ	3.0V~5.5V	-40°C~+125°C	

注 1: 要正常工作, 应使用外部 VREF- 和 VREF+ 引脚。建议电路见图 17-25。

下图为转换速率高于 500 ksps 时的建议电路。图中以 dsPIC30F6010 为例。

图 17-25: A/D 转换器参考电压原理图



以下配置步骤为转换速度高于 500 ksps 时所需的设置值。

17.22.1 1 Msps 配置指南

A/D 工作于 1 Msps 时，其配置取决于对单一输入引脚进行采样还是对多个输入引脚进行采样。

17.22.1.1 单模拟输入

对单模拟输入进行速率为 1 Msps 的转换时，必须至少使能两个采样保持通道。模拟输入复用器必须配置为同一输入引脚同时与两个采样保持通道相连。A/D 将对保存在某个 S/H 通道的值进行转换，同时第二个 S/H 通道采样新的输入。

17.22.1.2 多模拟输入

A/D 转换器使用多个采样保持通道还可对多个模拟输入进行采样。这种情况下，总转换速率 1 Msps 被划分供不同的信号使用。例如，四个输入信号可分别以 250 ksps 的速率被采样，或者两个输入信号分别以 500 ksps 的速率被采样。在此配置下，必须使用顺序采样以使每个输入有足够的采样时间。

17.22.1.3 1 Msps 配置步骤

必须进行如下配置以达到 1 Msps 的转换速率。

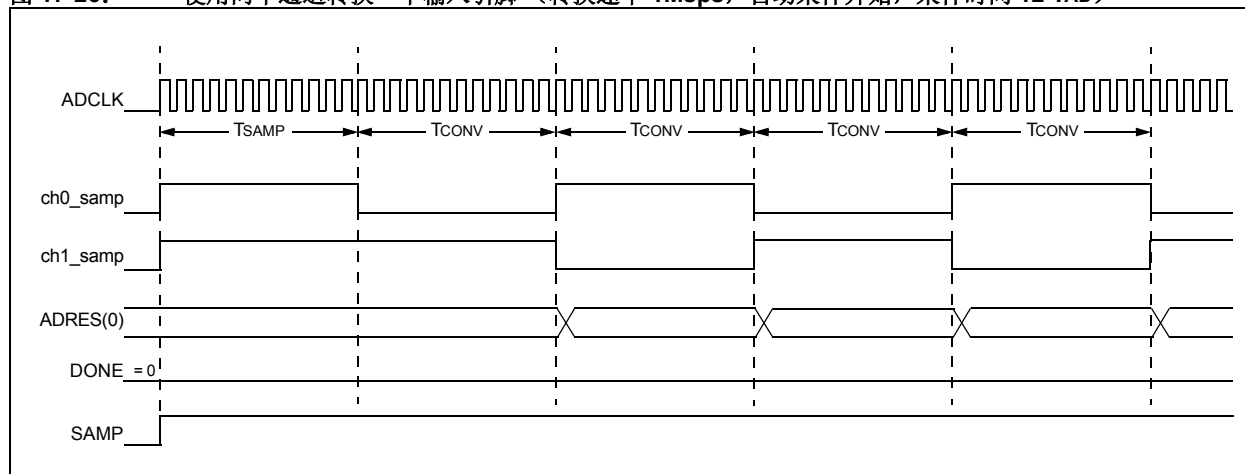
- 符合表 17-9 中所述条件。
- 根据图 17-26 中的建议电路所示连接外部 VREF+ 和 VREF- 引脚。
- 在寄存器 ADCON1 中置 SSRC<2:0> = 111，使能自动转换选项。
- 将寄存器 ADCON1 中的 ASAM 控制位置 1，使能自动采样。
- 将寄存器 ADCON1 中的 SIMSAM 位清零，使能顺序采样。
- 写入寄存器 ADCON2 中的 CHPS<1:0> 控制位，使能至少两个采样保持通道。
- 写入寄存器 ADCON2 中的 SMPI<3:0> 控制位，将中断之间的转换次数配置为最小为 2，因为至少有两个采样保持通道。
- 写入寄存器 ADCON3 中的 ADCS<5:0> 控制位，将 A/D 时钟周期配置为：

$$\frac{1}{12 \times 1,000,000} = 83.33 \text{ ns}$$

- 置 SAMC<4:0> = 00010，将采样时间配置为 2 TAD。
- 写入寄存器 ADCHS，为每个模拟输入引脚选择至少两个通道。

A/D 使用两个采样保持通道转换一个输入引脚的时序图如下所示。所选的 TAD 以及根据上述要求进行的配置将使转换速度达到 1 Msps。代码示例参见例 17-8。

图 17-26: 使用两个通道转换一个输入引脚（转换速率 1Msps，自动采样开始，采样时间 12 TAD）



例 17-8: 转换两个通道（自动采样开始，TAD 转换开始，顺序采样）的代码

```
ADPCFG = 0xFFFB; // all PORTB = Digital; RB2 = analog
ADCON1 = 0x00E0; // SSRC bit = 111 implies internal
// counter ends sampling and starts
// converting.
ADCHS = 0x0002; // Connect RB2/AN2 as CH0 input and also connect
RB2/AN2 // to positive CH1 input.
// in this example RB2/AN2 is the input to two
channels.
ADCSSL = 0;
ADCON3 = 0x0C04; // Sample time = 12Tad = 83.33 ns @ MIPS
// which will give 1 / (12 * 83.33 ns) = 1 Msps
ADCON2 = 0x6104; // Select external VREF+ and VREF- pins, convert CH0
and // CH1, Interrupt after every 2 samples
ADCON1bits.ADON = 1; // turn ADC ON
while (1) // repeat continuously
{
    IFS0bits.ADIF = 0; // clear interrupt
    while (IFS0bits.ADIF); // conversion done?
    ADCValue = ADCBUF0; // save the ADC values
} // repeat
```

17.22.2 750 ksps 配置指南

要达到 750 ksps 的转换速率，必须进行如下配置。此配置假设采样单个模拟输入。

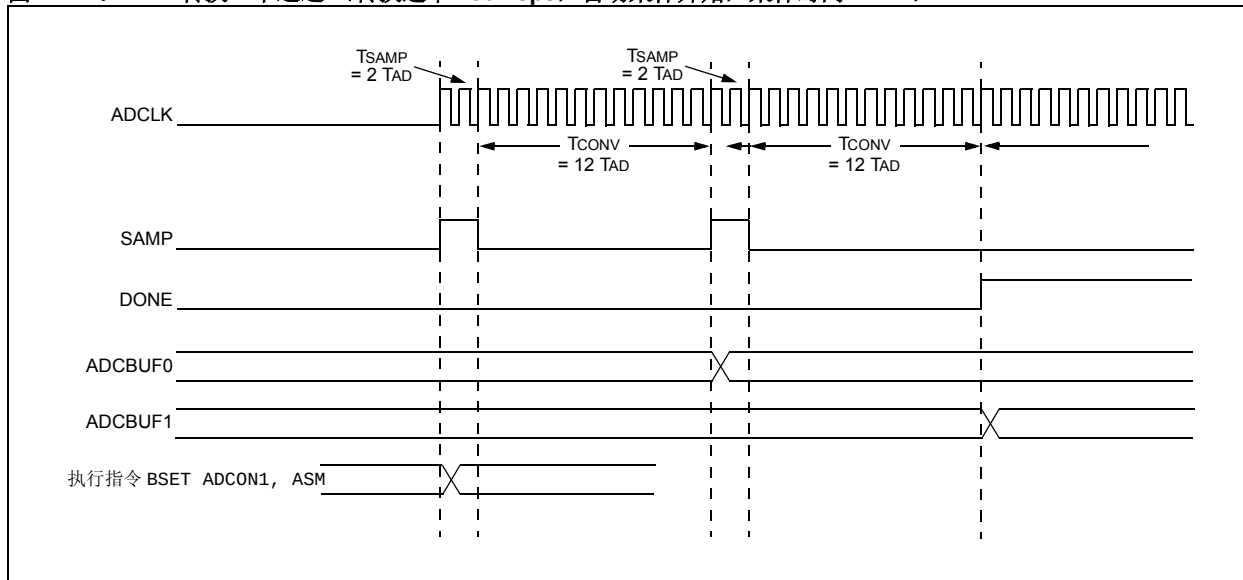
- 符合表 17-9 中所述条件。
- 根据图 17-27 中的建议电路所示连接外部 VREF+ 和 VREF- 引脚。
- 在寄存器 ADCON1 中置 SSRC<2:0> = 111，使能自动转换选项。
- 将寄存器 ADCON1 中的 ASAM 控制位置 1，使能自动采样。
- 在寄存器 ADCON2 中置 CHPS<1:0> = 00，使能一个采样保持通道。
- 在寄存器 ADCON2 中的 SMPI<3:0> 控制位写入所需的中断之间的转换次数。
- 写入寄存器 ADCON3 中的 ADCS<5:0> 控制位，将 A/D 时钟周期配置为：

$$\frac{1}{(12 + 2) \times 750,000} = 95.24 \text{ ns}$$

- 置 SAMC<4:0> = 00010，将采样时间配置为 2 TAD。

下图显示了 A/D 工作速率为 750 ksps 时的时序图。所选的 TAD 以及根据上述要求进行的配置使转换速度可达 750 ksps。代码示例参见例 17-9。

图 17-27: 转换一个通道 (转换速率 750 ksp/s, 自动采样开始, 采样时间 2 TAD)



例 17-9: 转换一个通道 (转换速率 750 ksp/s, 自动采样开始, 采样时间 2 TAD) 的代码示例

```

ADPCFG = 0xFFFF; // all PORTB = Digital; RB2 = analog
ADCON1 = 0x00E0; // SSRC bit = 111 implies internal
// counter ends sampling and starts
// converting.
ADCHS = 0x0002; // Connect RB2/AN2 as CH0 input
// in this example RB2/AN2 is the input
ADCSSL = 0;
ADCON3 = 0x0203; // Sample time = 2Tad, Tad = 95.24 ns @ 21 MIPS
// which will give 1 / (14 * 95.24 ns) = 750 ksp/s
ADCON2 = 0x6004; // Select external VREF+ and VREF- pins
// Interrupt after every 2 samples
ADCON1bits.ADON = 1; // turn ADC ON
while (1) // repeat continuously
{
    ADCValue = 0; // clear value
    ADC16Ptr = &ADCBUF0; // initialize ADCBUF pointer
    IFS0bits.ADIF = 0; // clear ADC interrupt flag
    ADCON1bits.ASAM = 1; // auto start sampling
    // for 31Tad then go to conversion
    while (!IFS0bits.ADIF); // conversion done?
    ADCON1bits.ASAM = 0; // yes then stop sample/convert
    for (count = 0; count < 2; count++) // average the 2 ADC value
        ADCValue = ADCValue + *ADC16Ptr++;
    ADCValue = ADCValue >> 1;
} // repeat

```

17.22.3 600 ksp/s 配置指南

A/D 工作于 600 ksp/s 时, 其配置取决于对一个输入引脚进行采样还是对多个引脚进行采样。

17.22.3.1 单模拟输入

对单模拟输入进行速率为 600 ksp/s 的转换时, 必须使能至少两个采样保持通道。模拟输入复用器必须配置为同一输入引脚同时与两个采样保持通道相连。A/D 将对保存在某个 S/H 通道中的值进行转换, 同时在第二个 S/H 通道采样新的输入。

17.22.3.2 多模拟输入

A/D 转换器使用多采样保持通道还可采样多个模拟输入。这种情况下，总转换速率 600 ksps 被划分供不同输入信号使用。例如，四个输入信号可分别以 150 ksps 的速率被采样，或两个输入信号分别以 300 ksps 的速率被采样。此配置下必须使用顺序采样以使每个输入有足够的采样时间。

17.22.3.3 600 ksps 配置项

要达到 600 ksps 的转换速率，必须进行以下配置。

- 符合表 17-9 中所述的条件。
- 根据图 17-10 中的建议电路所示连接外部 VREF+ 和 VREF- 引脚。
- 在寄存器 ADCON1 中置 SSRC<2:0> = 111，使能自动转换选项。
- 将寄存器 ADCON1 中的 ASAM 控制位置 1，使能自动采样。
- 将寄存器 ADCON1 中的 SIMSAM 位清零，使能顺序采样。
- 写入寄存器 ADCON2 中的 CHPS<1:0> 控制位，使能至少两个采样保持通道。
- 写入寄存器 ADCON2 中的 SMPI<3:0> 控制位，配置中断之间的转换次数为至少为 2，因为至少有两上采样保持通道。
- 写入寄存器 ADCON3 中的 ADCS<5:0>，配置 A/D 时钟周期为：

$$\frac{1}{12 \times 600,000} = 138.89 \text{ ns}$$

- 置 SAMC<4:0> = 00010，配置采样时间为 2 TAD。
- 写入寄存器 ADCHS，为每个模拟输入引脚选择至少两个通道。

600 ksps 扩展速率的时序图与 1 Msps 的相同，如图 17-26 所示。600 ksps A/D 工作的代码示例见例 17-10。

例 17-10: 转换两个通道（自动采样开始，TAD 转换开始，顺序采样）的代码

```
ADPCFG = 0xFFFB; // all PORTB = Digital; RB2 = analog
ADCON1 = 0x00E0; // SSRC bit = 111 implies internal
// counter ends sampling and starts
// converting.
ADCHS = 0x0002; // Connect RB2/AN2 as CH0 input and also connect
RB2/AN2 // to positive CH1 input.
// in this example RB2/AN2 is the input to two
channels.
ADCSSL = 0;
ADCON3 = 0x0C04; // Sample time = 12Tad = 138.89 ns @ 18 MIPS
// which will give 1 / (12 * 138.89 ns) = 600 ksps
ADCON2 = 0x6104; // Select external VREF+ and VREF- pins, convert CH0
and // CH1, Interrupt after every 2 samples
ADCON1bits.ADON = 1; // turn ADC ON
while (1) // repeat continuously
{
    IFS0bits.ADIF = 0; // clear interrupt
    while (IFS0bits.ADIF); // conversion done?
    ADCValue = ADCBUF0; // save the ADC values
} // repeat
```

17.23 在休眠和空闲模式下工作

休眠和空闲模式有助于将转换噪声降至最小，因为 CPU、总线和其他外设的数字活动被减到最少。

17.23.1 不使用 RC A/D 时钟的 CPU 休眠模式

当器件进入休眠模式时，模块的所有时钟源被关闭并保持逻辑“0”状态。

如果在一次转换过程中进入休眠状态，转换会中止，除非 A/D 将内部 RC 时钟发生器作为时钟源。从休眠模式退出时，转换器将不会恢复尚未完成的转换。

器件进入或离开休眠模式不会影响寄存器的内容。

17.23.2 使用 RC A/D 时钟的 CPU 休眠模式

如果将内部 A/D RC 振荡器设置为 A/D 时钟源 ($ADRC = 1$)，A/D 模块就可以在休眠模式下工作。这样做可以消除转换中的数字开关噪声。转换完成后，DONE 位将被置位，转换结果送入 A/D 结果缓冲器 ADCBUF。

如果允许 A/D 中断 ($ADIE = 1$)，A/D 中断发生时器件将从休眠模式唤醒。如果 A/D 中断的优先级高于当前 CPU 的优先级，程序执行将在 A/D 中断服务程序执行后恢复。否则，程序执行将从使器件处于休眠模式的 PWRSAV 指令后的指令继续。

如果禁止 A/D 中断，即使 ADON 保持置位，还是会关闭 A/D 模块。

为了将数字噪声对 A/D 模块工作的影响降至最低，用户应该选择转换触发源以确保 A/D 转换可在休眠模式下进行。自动转换触发源选项可用于休眠模式下的采样和转换 ($SSRC<2:0> = 111$)。要使用自动转换选项，应该在 PWRSAV 指令前的指令中将 ADON 位置位。

注： 为了使 A/D 模块可以在休眠模式下工作，必须将 RC 设置为 A/D 时钟源 ($ADRC = 1$)。

17.23.3 CPU 空闲模式下的 A/D 工作

ADSIDL 位 ($ADCON1<13>$) 选择 A/D 模块在空闲模式下是停止工作还是继续工作。如果 ADSIDL = 0，当器件进入空闲模式时，该模块将继续正常工作。如果允许 A/D 中断 ($ADIE = 1$)，A/D 中断发生时，器件将从空闲模式唤醒。如果 A/D 中断的优先级高于当前 CPU 的优先级，程序执行将在 A/D 中断服务程序执行后恢复。否则，程序将从使器件处于空闲模式的 PWRSAV 指令后的指令继续执行。

如果 ADSIDL = 1，在空闲模式下该模块将停止工作。如果器件在一次转换过程中进入空闲模式，该转换将中止。从空闲模式退出时，该转换器将不会恢复部分完成的转换。

17.24 复位的影响

器件复位强制所有寄存器进入复位状态。同时迫使 A/D 模块关闭并中止任何正在进行的转换。所有与模拟输入复用的引脚将被配置为模拟输入。相应的 TRIS 位将被置位。

上电复位时，不初始化 ADCBUF 寄存器，ADCBUF0……ADCBUFF 中的值不确定。

17.25 与 10 位 A/D 转换器相关的特殊功能寄存器

下表列举了 dsPIC30F 10 位 A/D 转换器特殊功能寄存器，包括它们的地址和格式。所有未用的寄存器和 / 或寄存器中的位读作 0。

表 17-10: ADC 寄存器映射

寄存器名	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位状态
INTCON1	0080	NSTDIS	—	—	—	—	OVATE	OVBTE	COVTE	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000 0000 0000 0000
INTCON2	0082	ALTIVT	—	—	—	—	—	—	—	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP	0000 0000 0000 0000
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0	0000 0000 0000 0000
IEC0	008C	CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IPC2	0098	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			0100 0100 0100 0100
ADCBUF0	0280	ADC 数据缓冲器 0																uuuu uuuu uuuu uuuu
ADCBUF1	0282	ADC 数据缓冲器 1																uuuu uuuu uuuu uuuu
ADCBUF2	0284	ADC 数据缓冲器 2																uuuu uuuu uuuu uuuu
ADCBUF3	0286	ADC 数据缓冲器 3																uuuu uuuu uuuu uuuu
ADCBUF4	0288	ADC 数据缓冲器 4																uuuu uuuu uuuu uuuu
ADCBUF5	028A	ADC 数据缓冲器 5																uuuu uuuu uuuu uuuu
ADCBUF6	028C	ADC 数据缓冲器 6																uuuu uuuu uuuu uuuu
ADCBUF7	028E	ADC 数据缓冲器 7																uuuu uuuu uuuu uuuu
ADCBUF8	0290	ADC 数据缓冲器 8																uuuu uuuu uuuu uuuu
ADCBUF9	0292	ADC 数据缓冲器 9																uuuu uuuu uuuu uuuu
ADCBUFA	0294	ADC 数据缓冲器 10																uuuu uuuu uuuu uuuu
ADCBUFB	0296	ADC 数据缓冲器 11																uuuu uuuu uuuu uuuu
ADCBUFC	0298	ADC 数据缓冲器 12																uuuu uuuu uuuu uuuu
ADCBUFD	029A	ADC 数据缓冲器 13																uuuu uuuu uuuu uuuu
ADCBUFE	029C	ADC 数据缓冲器 14																uuuu uuuu uuuu uuuu
ADCBUFF	029E	ADC 数据缓冲器 15																uuuu uuuu uuuu uuuu
ADCON1	02A0	ADON	ADFRZ	ADSIDL	—	—	—	FORM[1:0]		SSRC[2:0]		—	SIMSAM	ASAM	SAMP	CONV	0000 0000 0000 0000	
ADCON2	02A2	VCFG[2:0]			OFFCAL	—	CSCNA	CHPS[1:0]		BUFS	—	SMP[3:0]				BUFM	ALTS	0000 0000 0000 0000
ADCON3	02A4	—	—	—	SAMC[4:0]					ADRC	—	ADCS[5:0]						0000 0000 0000 0000
ADCHS	02A6	CHXNB[1:0]		CHXSB	CH0NB	CH0SB[3:0]				CHXNA[1:0]		CHXSA	CH0NA	CH0SA[3:0]				0000 0000 0000 0000
ADPCFG	02A8	PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8	PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000 0000 0000 0000
ADCSSL	02AA	ADC 输入扫描选择寄存器																0000 0000 0000 0000

图注: u = 未知

注: 所有中断源及其相关的控制位在某个器件上可能不可用。欲了解详细信息，请参见器件数据手册。

17.26 设计技巧

问 1: 如何优化 A/D 转换器的系统性能?

答:

1. 请确保满足了所有时序规范要求。如果关闭 A/D 模块后再打开, 应等待一个最小延时后再开始采样。如果改变了输入通道, 同样需要等待一个最小延时; 最后是 T_{AD} , 它是所选择的每一位的转换时间。它在 $ADCON3$ 中选择, 且应该在一定范围之内, 满足电气特性的要求。如果 T_{AD} 太短, 转换终止时有可能还未对数据进行完全转换; 如果 T_{AD} 太长, 采样电容上的电压会在转换结束前开始衰减。在器件数据手册的“电气规范”章节中提供了这些时序规范。
2. 模拟输入信号的源阻抗经常很高 (大于 $10\text{ k}\Omega$), 因此为采样电容充电时产生的泄漏电流可能会影响到精度。如果输入信号的变化不是太快, 可以尝试在模拟输入端连接一个 $0.1\text{ }\mu\text{F}$ 的电容。该电容可充电到所采样的模拟电压, 并为 4.4 pF 的内部保持电容提供所需的瞬时充电电流。
3. 在启动 A/D 转换前请让器件进入休眠模式。在休眠模式下的转换需要选择 RC 时钟源。这个方法可以提高精度, 因为来自 CPU 和其他外设的数字噪声被降至最小。

问 2: 您知道有关于 A/D 的好的参考资料吗?

答: Prentice Hall 出版的“Analog-Digital Conversion Handbook”第三版 (ISBN 0-13-03-2848-0) 对理解 A/D 转换很有帮助。

问 3: 当通道/采样和采样/中断组合超过了缓冲器的大小时, 缓冲器会发生什么情况?

答: 不建议这种配置。该缓冲器将包含不确定的转换结果。

17.27 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与 10 位 A/D 转换器模块相关的应用笔记有：

标题	应用笔记编号
Using the Analog-to-Digital (A/D) Converter	AN546
Four Channel Digital Voltmeter with Display and Keyboard	AN557
Understanding A/D Converter Performance Specifications	AN693
dsPIC30F 在无传感器 BLDC 控制中的应用	AN901
使用 dsPIC30F 实现交流感应电机的矢量控制	AN908
使用 dsPIC30F2010 控制带传感器的 BLDC	AN957
使用 dsPIC30F MCU 控制交流感应电机	AN984

注： 如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

17.28 版本历史

版本 A

本文档的首次发行版。

版本 B

此版本对 dsPIC30F 10 位 A/D 转换器模块进行了文字和技术内容修订。

版本 C

此版本加入了本文档更新时已知的所有勘误表。

版本 D

此版本增加了扩展转换速率的指南。

注:

第 18 章 12 位 A/D 转换器

目录

本章包括下列主题：

18.1	简介	18-2
18.2	控制寄存器	18-4
18.3	A/D 转换结果缓冲器	18-4
18.4	A/D 转换术语和转换过程	18-10
18.5	A/D 模块配置	18-11
18.6	参考电压源的选择	18-11
18.7	A/D 转换时钟的选择	18-12
18.8	采样模拟输入的选择	18-12
18.9	模块使能	18-14
18.10	如何开始采样	18-14
18.11	如何停止采样和开始转换	18-14
18.12	采样 / 转换工作的控制	18-19
18.13	如何将转换结果写入缓冲器的说明	18-19
18.14	转换过程示例	18-21
18.15	A/D 采样要求	18-26
18.16	读取 A/D 转换结果缓冲器	18-27
18.17	传递函数	18-28
18.18	A/D 转换的精度 / 误差	18-28
18.19	连接注意事项	18-28
18.20	初始化	18-29
18.21	A/D 转换速度	18-30
18.22	在休眠和空闲模式下工作	18-33
18.23	复位的影响	18-33
18.24	与 12 位 A/D 转换器相关的特殊功能寄存器	18-34
18.25	设计技巧	18-35
18.26	相关应用笔记	18-36
18.27	版本历史	18-37

18.1 简介

dsPIC30F 12 位 A/D 转换器具有以下主要特点：

- 逐次逼近寄存器（SAR）转换
- 最大 200 ksp/s 的转换速度
- 最大 16 个模拟输入引脚
- 外部参考电压输入引脚
- 单极性差分 S/H 放大器
- 自动通道扫描模式
- 可选转换触发源
- 16 字转换结果缓冲器
- 可选缓冲器填充模式
- 四种结果对齐选择
- 在休眠和空闲模式下运行

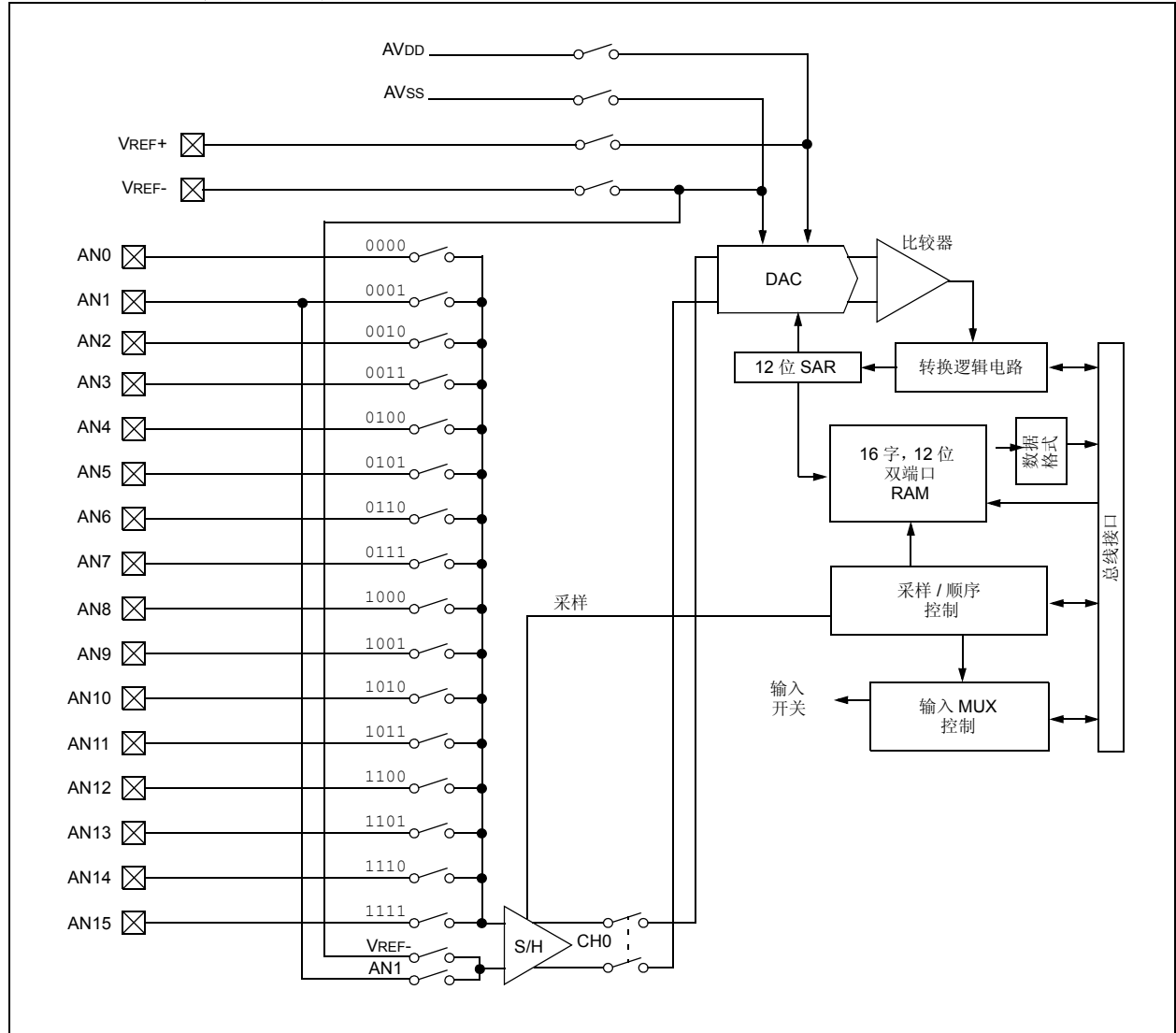
图 18-1 显示了 12 位 A/D 的框图。12 位 A/D 转换器最多可以有 16 个模拟输入引脚，即 AN0—AN15。此外，有两个可用于外部参考电压连接的模拟输入引脚。这两个参考电压输入可以和其他模拟输入引脚复用。实际的模拟输入引脚数和外部参考电压输入配置取决于具体的 dsPIC30F 器件。更多信息请参见 dsPIC30F 器件数据手册（DS70082 和 DS70083）。

模拟输入是通过多路开关连接到 S/H 放大器的，例如 CH0。模拟输入多路开关可以在转换期间在两组模拟输入之间切换。使用某些输入引脚可以进行单极性差分转换（参见图 18-1）。

CH0 S/H 放大器可以使能为模拟输入扫描模式。控制寄存器指定在扫描过程中要包含的模拟输入通道。

12 位 A/D 与 16 字结果缓冲器相连。在从缓冲器读取每个 12 位结果时，它们被转换为四种 16 位输出格式之一。

图 18-1: 12 位高速 A/D 框图



18.2 控制寄存器

A/D 模块有六个控制和状态寄存器。这些寄存器为：

- ADCON1: A/D 控制寄存器 1
- ADCON2: A/D 控制寄存器 2
- ADCON3: A/D 控制寄存器 3
- ADCHS: A/D 输入通道选择寄存器
- ADPCFG: A/D 端口配置寄存器
- ADCSSL: A/D 输入扫描选择寄存器

ADCON1、ADCON2 和 ADCON3 寄存器控制 A/D 模块的工作。ADCHS 寄存器选择连接到 S/H 放大器的输入引脚。ADPCFG 寄存器将模拟输入引脚配置为模拟输入或数字 I/O。ADCSSL 寄存器选择扫描输入的顺序。

18.3 A/D 转换结果缓冲器

该模块包含 16 字双口 RAM，称为 ADCBUF，用于缓冲 A/D 结果。16 个缓冲器单元分别称为 ADCBUF0、ADCBUF1、ADCBUF2、……ADCBUFE 和 ADCBUFF。

注： A/D 结果缓冲器是只读缓冲器。

寄存器 18-1: ADCON1: A/D 控制寄存器 1

高字节:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
ADON	—	ADSIDL	—	—	—	FORM<1:0>	
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/C-0
SSRC<2:0>			—	—	ASAM	SAMP	DONE
bit 7							bit 0

- bit 15 **ADON:** A/D 工作模式位
1 = A/D 转换器模块正在工作
0 = A/D 转换器已关闭
- bit 14 **未用:** 读作 0
- bit 13 **ADSIDL:** 在空闲模式位停止
1 = 当器件进入空闲模式时模块不继续工作
0 = 在空闲模式模块继续工作
- bit 12-10 **未用:** 读作 0
- bit 9-8 **FORM<1:0>:** 数据输出格式位
11 = 有符号小数 (DOUT = sddd dddd dddd 0000)
10 = 小数 (DOUT = dddd dddd dddd 0000)
01 = 有符号整数 (DOUT = ssss sddd dddd dddd)
00 = 整数 (DOUT = 0000 dddd dddd dddd)
- bit 7-5 **SSRC<2:0>:** 转换触发源选择位
111 = 内部计数器结束采样并开始转换 (自动转换)
110 = 保留
101 = 保留
100 = 保留
011 = 电机控制 PWM 间隔结束采样并开始转换
010 = 通用功能 Timer3 比较结束采样并开始转换
001 = INT0 引脚的有效电平变化结束采样并开始转换
000 = 清除 SAMP 位结束采样并开始转换
- bit 4-3 **未用:** 读作 0
- bit 2 **ASAM:** A/D 采样自动开始位
1 = 采样在上一个转换结束后立即开始。SAMP 位自动置位。
0 = 采样在 SAMP 位置位后开始
- bit 1 **SAMP:** A/D 采样使能位
1 = 至少一个 A/D 采样 / 保持放大器正在采样
0 = A/D 采样 / 保持放大器正在保持
当 ASAM = 0 时, 此位写入 1 将开始采样。
当 SSRC = 000 时, 此位写入 0 将结束采样并开始转换。
- bit 0 **DONE:** A/D 转换状态位
1 = A/D 转换完成
0 = A/D 转换未完成
将此位清零不会影响进行中的任何操作。
此位由软件或开始新转换清零。

图注:

R = 可读位	W = 可写位	C = 可软件清零
HC = 硬件清零	HS = 硬件置位	U = 未用位, 读作 0
-n = POR 时的值	1 = 置位	0 = 清零
		x = 未知

寄存器 18-2: ADCON2: A/D 控制寄存器 2

高字节:							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	U-0	U-0
VCFG<2:0>			—	—	CSCNA	—	—
bit 15							bit 8

低字节:							
R-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS	—	SMPI<3:0>				BUFM	ALTS
bit 7							bit 0

bit 15-13 **VCFG<2:0>**: 参考电压配置位

	A/D VREFH	A/D VREFL
000	AVDD	AVSS
001	外部 VREF+ 引脚	AVSS
010	AVDD	外部 VREF- 引脚
011	外部 VREF+ 引脚	外部 VREF- 引脚
1xx	AVDD	AVSS

bit 12 **保留**: 用户应该在此位置写入 0

bit 11 **未用**: 读作 0

bit 10 **CSCNA**: MUX A 输入多路开关设置的 CH0+ S/H 输入的扫描输入选择位
1 = 扫描输入
0 = 不扫描输入

bit 9-8 **未用**: 读作 0

bit 7 **BUFS**: 缓冲器填充状态位
仅在 BUFM = 1 时有效 (ADRES 分成 2 x 8 字的缓冲器)
1 = A/D 正在填充缓冲器 0x8—0xF, 用户应该访问 0x0—0x7 的数据
0 = A/D 正在填充缓冲器 0x0—0x7, 用户应该访问 0x8—0xF 的数据

bit 6 **未用**: 读作 0

bit 5-2 **SMPI<3:0>**: 每产生一次中断的采样 / 转换过程数选择位
1111 = 每完成第 16 个采样 / 转换过程后产生中断
1110 = 每完成第 15 个采样 / 转换过程后产生中断
.....
0001 = 每完成第 2 个采样 / 转换过程后产生中断
0000 = 完成每个采样 / 转换过程后产生中断

bit 1 **BUFM**: 缓冲器模式选择位
1 = 缓冲器配置为两个 8 字缓冲器 ADCBUF (15.....8) 和 ADCBUF (7.....0)
0 = 缓冲器配置为一个 16 字缓冲器 ADCBUF (15.....0)

bit 0 **ALTS**: 交替输入采样模式选择位
1 = 为第一个采样使用 MUX A 输入多路开关设置, 然后对所有后续采样在 MUX B 和 MUX A 输入多路开关设置之间轮换
0 = 总是使用 MUX A 输入多路开关设置

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 18-3: **ADCON3: A/D 控制寄存器 3**

高字节:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	SAMC<4:0>				
bit 15			bit 8				

低字节:							
R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADRC	—	ADCS<5:0>					
bit 7		bit 0					

- bit 15-13 未用：读作 0
- bit 12-8 **SAMC<4:0>**: 自动采样时间位
11111 = 31 TAD
.....
00001 = 1 TAD
00000 = 0 TAD
- bit 7 **ADRC**: A/D 转换时钟源位
1 = A/D 内部 RC 时钟
0 = 时钟由系统时钟产生
- bit 6 未用：读作 0
- bit 5-0 **ADCS<5:0>**: A/D 转换时钟选择位
111111 = Tcy/2 * (ADCS<5:0> + 1) = 32 * Tcy
.....
000001 = Tcy/2 * (ADCS<5:0> + 1) = Tcy
000000 = Tcy/2 * (ADCS<5:0> + 1) = Tcy/2

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 18-4: ADCHS: A/D 输入选择寄存器

高字节:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CH0NB	CH0SB<3:0>			
bit 15				bit 8			

低字节:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CH0NA	CH0SA<3:0>			
bit 7				bit 0			

- bit 15-13 未用: 读作 0
- bit 12 **CH0NB**: MUX B 多路开关设置通道 0 负输入选择位
与 bit <4> 定义相同 (见注)。
- bit 11-8 **CH0SB<3:0>**: MUX B 多路开关设置通道 0 正输入选择位
与 bit <3:0> 定义相同 (见注)。
- bit 7-5 未用: 读作 0
- bit 4 **CH0NA**: MUX A 多路开关设置通道 0 负输入选择位
1 = 通道 0 负输入是 AN1
0 = 通道 0 负输入是 VREF-
- bit 3-0 **CH0SA<3:0>**: MUX A 多路开关设置通道 0 正输入选择位
1111 = 通道 0 正输入是 AN15
1110 = 通道 0 正输入是 AN14
1101 = 通道 0 正输入是 AN13
.....
0001 = 通道 0 正输入是 AN1
0000 = 通道 0 正输入是 AN0

注: 模拟输入多路开关支持两种输入设置配置, 即 MUX A 和 MUX B。ADCHS<15:8> 决定 MUX B 的设置, ADCHS<7:0> 决定 MUX A 的设置。两种设置的控制位功能相同。

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 18-5: ADPCFG: A/D 端口配置寄存器

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 15-0 **PCFG<15:0>**: 模拟输入引脚配置控制位

1 = 模拟输入引脚处于数字模式, 使能端口读取输入, A/D 输入多路开关输入连接到 AVss

0 = 模拟输入引脚处于模拟模式, 禁止端口读取输入, A/D 采样引脚电压

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

寄存器 18-6: ADCSSL: A/D 输入扫描选择寄存器

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0
bit 7							bit 0

bit 15-0 **CSSL<15:0>**: A/D 输入引脚扫描选择位

1 = 选择对 ANx 输入进行扫描

0 = 输入扫描时跳过 ANx

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

18.4 A/D 转换术语和转换过程

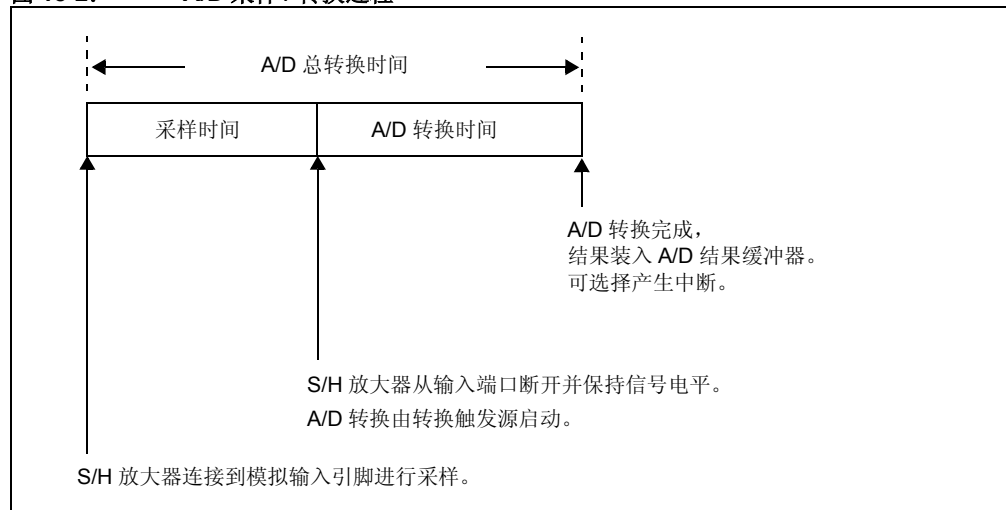
图 18-2 显示了一个基本转换过程以及所使用的术语。模拟输入引脚电压的采样是通过采样和保持 S/H 放大器进行的。S/H 放大器也称为 S/H 通道。12 位 A/D 转换器有一个 S/H 通道，名为 CH0。S/H 通道通过模拟输入多路开关连接到模拟输入引脚。模拟输入多路开关由 ADCHS 寄存器控制。在 ADCHS 寄存器中有两组多路开关控制位，两者功能相同。这两组控制位可以允许设置两种不同的模拟输入多路开关配置（称为 MUX A 和 MUX B）。A/D 转换器也可以在转换过程中在 MUX A 和 MUX B 配置之间切换。A/D 转换器也可以选择对一系列模拟输入进行扫描。

采样时间是 A/D 模块的 S/H 放大器连接到模拟输入引脚的时间。采样时间可通过将 SAMP 位（ADCON1<1>）置位手动开始或由 A/D 转换器硬件自动开始。采样时间可通过在用户软件中将 SAMP 控制位清零手工结束，或由转换触发源自动结束。

转换时间是 A/D 转换器转换 S/H 放大器保持电压所需的时间。A/D 在采样时间结束时从模拟输入引脚断开。除了一个时钟周期外，A/D 转换器还需要一个 A/D 时钟周期（T_{AD}）来转换结果中的每个位。要完整执行转换，总共需要 14 个 T_{AD} 周期。当转换时间结束后，结果装入 16 个 A/D 结果寄存器（ADCBUF0……ADCBUFF）中的一个，S/H 可重新连接到输入引脚，并且可能会产生 CPU 中断。

采样时间和 A/D 转换时间之和就是总转换时间。为了确保 S/H 放大器能为 A/D 转换提供足够的精度，需要有一个最小采样时间（参见第 18.15 节“A/D 采样要求”）。此外，A/D 转换器还有多个输入时钟选择。用户必须选择不会违反最小 T_{AD} 规范的输入时钟选择。

图 18-2: A/D 采样 / 转换过程



采样的开始时间可在软件中将 **SAMP** 控制位置位控制。采样开始时间还可以由硬件自动控制。当 A/D 转换器工作在自动采样模式，在采样 / 转换过程中转换结束时，S/H 放大器会重新连接到模拟输入引脚。自动采样功能由 **ASAM** 控制位控制。

转换触发源结束采样时间并开始 A/D 转换或采样 / 转换过程。采样触发源由 **SSRC** 控制位选择。转换触发源可以从各种硬件源中选择，或者通过在软件中将 **SAMP** 控制位清零来手工控制。转换触发源中有一个是自动转换。自动转换之间的时间是通过计数器和 A/D 时钟设置的。自动采样模式和自动转换触发可以一起使用，以提供无限自动转换而无需软件干预。

在每个或多个采样 / 转换过程结束时可能会产生中断，这取决于 **SMPI** 控制位的值。中断之间的采样 / 转换过程数可以为 1 到 16 之间的任何值。

18.5 A/D 模块配置

执行 A/D 转换时应该遵循以下步骤：

1. 配置 A/D 模块
 - 选择参考电压源以匹配模拟输入的预期范围
 - 选择模拟转换时钟以便使预期的数据速率与处理器时钟匹配
 - 确定采样发生的方式
 - 确定输入分配到 S/H 通道的方式
 - 选择转换结果出现在缓冲器中的方式
 - 选择中断率
 - 打开 A/D 模块
2. 配置 A/D 中断（如果需要）
 - 清除 **ADIF** 位
 - 选择 A/D 中断优先级

每个配置步骤的选择将在后续章节中说明。

注： 当 **ADON = 1** 时，不应该写入 **SSRC<2:0>**、**SIMSAM**、**ASAM**、**CHPS<1:0>**、**SMPI<3:0>**、**BUFM** 和 **ALTS** 位以及 **ADCON3** 和 **ADCSSL** 寄存器。否则会产生无法预料的结果。

18.6 参考电压源的选择

A/D 转换的参考电压是使用 **VCFG<2:0>** 控制位（**ADCON2<15:13>**）选择的。参考电压高电平（**VREFH**）和参考电压低电平（**VREFL**）可以是内部 **AVDD** 和 **AVSS** 电压或 **VREF+** 和 **VREF-** 输入引脚。

在低引脚数器件上，外部参考电压引脚可能会和 **AN0** 以及 **AN1** 输入复用。当这些引脚与 **VREF+** 和 **VREF-** 输入引脚复用时，A/D 转换器仍然可以在这些引脚上执行转换。

加到外部参考引脚上的电压必须符合某些规范。更多详细信息，请参见器件数据手册的“电气规范”部分。

注： 转换速率高于 100 ksp/s 时必须选用外部 **VREF+** 和 **VREF-**。详情请参见第 18.21 节“A/D 转换速度”。

18.7 A/D 转换时钟的选择

A/D 转换器有一个可以完成转换的最大速率。转换时序由模拟模块时钟 TAD 控制。A/D 转换需要 14 个时钟周期（14 TAD）。A/D 时钟是从器件指令时钟产生的。

A/D 转换时钟是使用六位计数器由软件选择的。有 64 种可能的 TAD 选择，由 ADCS<5:0> 位（ADCON3<5:0>）指定。公式 18-1 说明了 TAD 值与 ADCS 控制位以及器件指令周期 Tcy 之间的关系。

公式 18-1: A/D 转换时钟周期

$$T_{AD} = \frac{T_{CY}(ADCS + 1)}{2}$$
$$ADCS = \frac{2T_{AD}}{T_{CY}} - 1$$

为了获得正确的 A/D 转换结果，必须选择 A/D 转换时钟（TAD）以确保 333.33 ns 的最小 TAD 时间（详情请参见第 18.21 节“A/D 转换速度”）。

A/D 转换器有一个专用的内部 RC 时钟源，可以用于执行转换。如果在 dsPIC30F 处于休眠模式时进行 A/D 转换，应该使用内部 RC 时钟源。内部 RC 振荡器是通过将 ADRC 位（ADCON3<7>）置位选择的。当 ADRC 位置位时，ADCS<5:0> 位对 A/D 操作没有影响。

18.8 采样模拟输入的选择

为了选择采样的模拟输入，采样和保持放大器在其同相和反相输入上都有模拟多路开关（见图 18-1）。一旦指定了采样 / 转换过程，ADCHS 位决定为每个采样选择的模拟输入。

此外，所选的输入可能会在交替采样的基础上发生变化，或者随重复的采样过程变化。

注：不同器件会有不同的模拟输入数。应对照器件数据手册确认可用的模拟输入。

18.8.1 配置模拟端口引脚

ADPCFG 寄存器指定了用于模拟输入的器件引脚的输入状态。

当相应的 PCFGn 位（ADPCFG<n>）清零时，对应的引脚就被配置为模拟输入。ADPCFG 寄存器在复位时清零，这使 A/D 输入引脚在默认情况下在复位时被配置为模拟输入。

当配置为模拟输入时，相关端口 I/O 数字输入缓冲器被禁止以免消耗电流。

ADPCFG 寄存器和 TRISB 寄存器控制 A/D 端口引脚的工作方式。

希望作为模拟输入的端口引脚对应的 TRIS 位必须置位，以指定为端口输入。如果与 A/D 输入关联的 I/O 引脚被配置为输出，TRIS 位被清零，该引脚处于模拟模式（ADPCFG<n> = 0），并且端口数字输出电平（VOH 或 VOL）将被转换。在器件复位后，所有的 TRIS 位都会被置位。

当相应的 PCFGn 位（ADPCFG<n>）置位时，引脚就被配置为数字 I/O。在此配置中，模拟多路开关的输入连接到 AVss。

注 1: 当读取端口寄存器时，任何配置为模拟输入的端口都读作 0。
2: 在任何定义为数字输入的引脚（包括 AN15:AN0 引脚）所加的模拟电平可能会导致输入缓冲器消耗超出器件规范的电流。

18.8.2 通道 0 输入选择

用户可以选择最多 16 个模拟输入中的任何一个连接到通道的正输入。CH0SA<3:0> 位 (ADCHS<3:0>) 通常选择作为通道 0 的正输入的模拟输入。

用户可以选择 VREF- 或 AN1 作为通道的负输入。CH0NA 位 (ADCHS<4>) 通常选择作为通道 0 的负输入的模拟输入。

18.8.2.1 指定交替通道 0 输入选择

ALTS 位 (ADCON2<0>) 使模块在连续采样期间可选择在两组输入间切换。

CH0SA<3:0>、CH0NA、CHXSA 和 CHXNA<1:0> 指定的输入统称为 MUX A 输入。CH0SB<3:0>、CH0NB、CHXSB 和 CHXNB<1:0> 指定的输入统称为 MUX B 输入。当 ALTS 位为 1 时, 模块将在一个采样中使用 MUX A 输入, 而在随后一个采样中使用 MUX B 输入, 依次进行轮换。

对于通道 0, 如果 ALTS 位为 0, 则只会选择采样 CH0SA<3:0> 和 CH0NA 指定的输入。

如果 ALTS 位在通道 0 的第一个采样 / 转换过程中为 1, 则会选择采样 CH0SA<3:0> 和 CH0NA 指定的输入。在通道 0 的下一个采样转换过程中, 会选择采样 CH0SB<3:0> 和 CH0NB 指定的输入。此模式将在后续采样转换过程中重复。

18.8.2.2 扫描数个输入

通道 0 能扫描一组选定的输入。CSCNA 位 (ADCON2<10>) 可以使 CH0 通道扫描选定数量的模拟输入。当 CSCNA 置位时, 会忽略 CH0SA<3:0> 位。

ADCSSL 寄存器指定要扫描的输入。ADCSSL 寄存器中的每个位分别对应一个模拟输入。Bit 0 对应 AN0, bit 1 对应 AN1, 依此类推。如果 ADCSSL 寄存器中特定位为 1, 则在扫描过程中将包括对应的输入。扫描总是在每次发生中断后从第一个所选择的通道开始, 从编号较小的输入扫描到编号较大的输入。

注: 如果选择的扫描输入个数大于每次中断的采样数, 则较大编号的输入将不会被采样。

ADCSSL 位只指定通道正输入的输入源。CH0NA 位将选择在扫描时通道负输入的输入源。

如果 ALTS 位为 1, 扫描只应用于 MUX A 输入选择。由 CH0SB<3:0> 指定的 MUX B 输入选择将仍然选择备用的输入。当以这种方式设定输入选择时, 输入将在一系列 ADCSSL 寄存器指定的扫描输入和 CH0SB 位指定的固定输入之间轮换。

18.9 模块使能

当 ADON 位（ADCON1<15>）为 1 时，模块处于有效模式并且全功耗全功能运行。

当 ADON 为 0 时，模块被禁止。为了最大限度地节省电流消耗，电路的数字和模拟部分被关闭。

为了从关闭模式返回有效模式，用户必须等待模拟阶段稳定下来。有关稳定时间的信息，请参见器件数据手册的“电气特性”一章。

18.10 如何开始采样

18.10.1 手工

将 SAMP 位（ADCON1<1>）置位将使 A/D 开始采样。可以使用几个选项之一用于结束采样并完成转换。在 SAMP 位再次置位前，不会恢复采样。如图 18-3 所示。

18.10.2 自动

只要在通道上没有进行转换，将 ASAM 位（ADCON1<2>）置位将导致 A/D 自动开始采样通道。可以使用几个选项之一来结束采样并完成转换。在通道上的转换完成后，通道的采样将会重新开始。如图 18-4 所示。

18.11 如何停止采样和开始转换

转换触发源将终止采样并开始所选的转换过程。SSRC<2:0> 位（ADCON1<7:5>）选择转换触发源。

注： 根据 dsPIC30F 器件的不同，可用的转换触发源可能也会不同。有关可用转换触发源的信息，请参见特定器件数据手册。

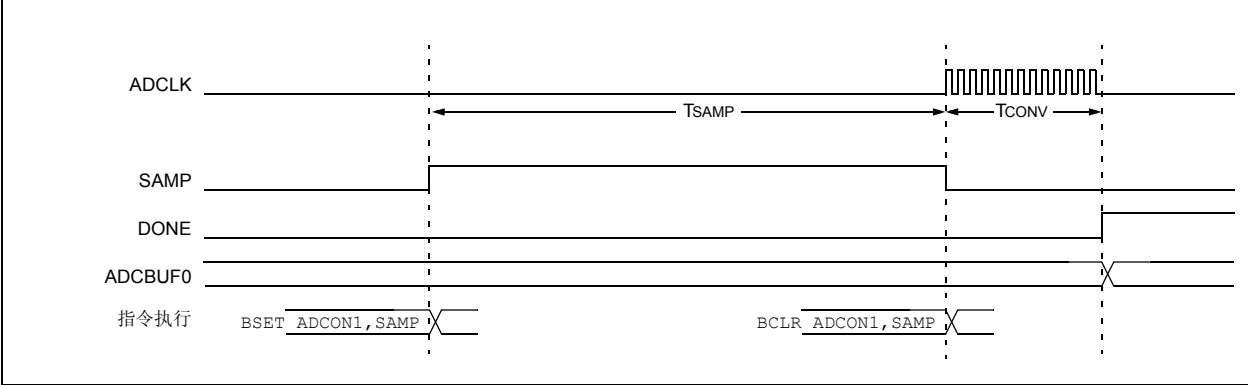
注： 使能了 A/D 模块时，不应该改变 SSRC 选择位。如果用户希望改变转换触发源，应该首先通过将 ADON 位（ADCON1<15>）清零来禁止 A/D 模块。

18.11.1 手工

当 SSRC<2:0> = 000 时，转换触发处于软件控制下。将 SAMP 位（ADCON1<1>）清零将会开始转换过程。

图 18-3 说明置位 SAMP 位开始采样，将 SAMP 位清零终止采样并开始转换的示例。用户软件必须对 SAMP 位的置位和清零计时，以确保输入信号有足够的采样时间。

图 18-3: 转换一个通道（手工采样开始，手工转换开始）



例 18-1: 转换一个通道（手工采样开始，手工转换开始）的代码示例

```

ADPCFG = 0xFFFFB;           // all PORTB = Digital; RB2 = analog
ADCON1 = 0x0000;             // SAMP bit = 0 ends sampling ...
                                // and starts converting
ADCHS = 0x0002;               // Connect RB2/AN2 as CH0 input ..
                                // in this example RB2/AN2 is the input

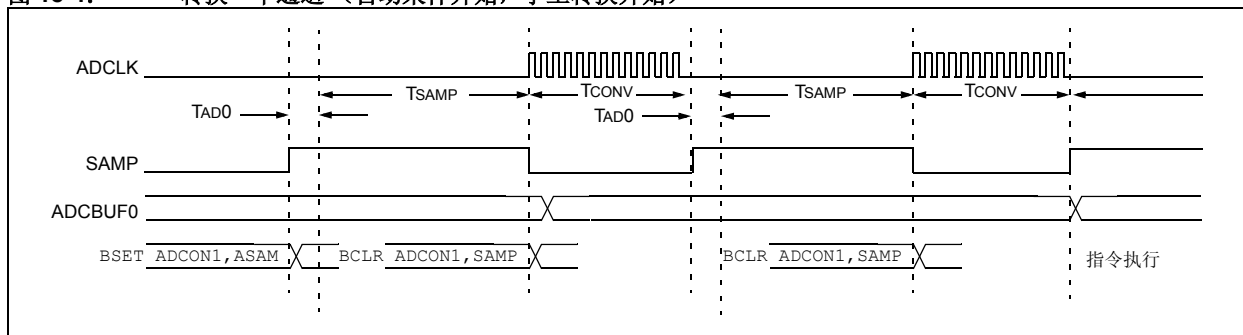
ADCSSL = 0;
ADCON3 = 0x0002;             // Manual Sample, Tad = internal 2 Tcy
ADCON2 = 0;

ADCON1bits.ADON = 1;         // turn ADC ON
while (1)                    // repeat continuously
{
    ADCON1bits.SAMP = 1;      // start sampling ...
    DelayNmSec(100);          // for 100 mS
    ADCON1bits.SAMP = 0;      // start Converting
    while (!ADCON1bits.DONE); // conversion done?
    ADCValue = ADCBUF0;       // yes then get ADC value
}                             // repeat

```

图 18-4 示例说明了置位 **ASAM** 位开始自动采样，将 **SAMP** 位清零终止采样并开始转换。转换完成后，模块将自动返回采样状态。在采样间隔开始时，**SAMP** 位会自动置位。用户软件必须对 **SAMP** 位的清零计时以确保输入信号有充足的采样时间。因为将 **SAMP** 位清零之间的时间除了采样时间外还包括转换时间。

图 18-4: 转换一个通道（自动采样开始，手工转换开始）



18.11.2 对转换触发计时

当 $SSRC<2:0> = 111$ ，转换触发处于 A/D 时钟控制下。SAMC 位（ $ADCON3<12:8>$ ）选择开始采样和开始转换之间的 T_{AD} 时钟周期数。在采样开始时，模块会计数 SAMC 位指定的 T_{AD} 时钟周期数。

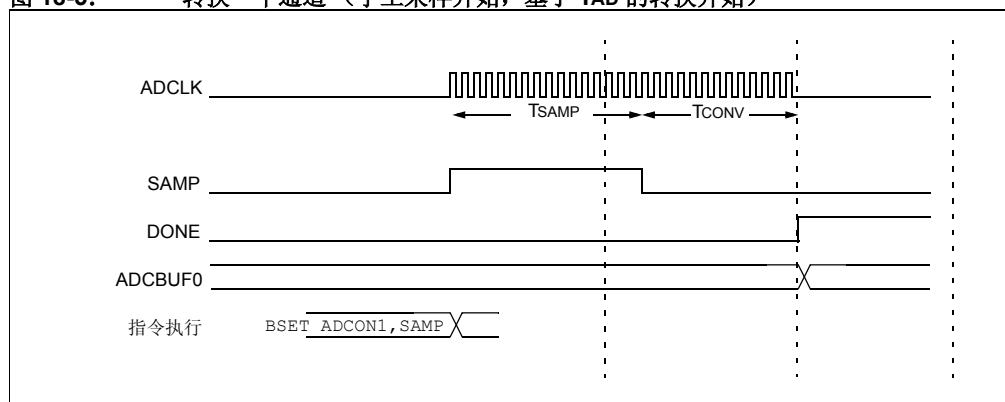
公式 18-2: 计时转换触发时间

$$T_{SMP} = SAMC<4:0> * T_{AD}$$

SAMC 必须总是编程为至少 1 个时钟周期，以确保符合采样要求。

图 18-5 显示了如何在用户软件开始采样时使用计时转换触发。

图 18-5: 转换一个通道（手工采样开始，基于 T_{AD} 的转换开始）



例 18-2: 转换一个通道（手工采样开始，基于 T_{AD} 的转换开始）的代码示例

```
ADPCFG = 0xFFFFB;           // all PORTB = Digital; RB2 = analog
ADCON1 = 0x00E0;             // SSRC bit = 111 implies internal
                               // counter ends sampling and starts
                               // converting.
ADCHS = 0x000C;              // Connect RB12/AN12 as CH0 input ..
                               // in this example RB12/AN12 is the input
ADCSSL = 0;
ADCON3 = 0x1F02;             // Sample time = 31Tad, Tad = internal 2
Tcy
ADCON2 = 0;

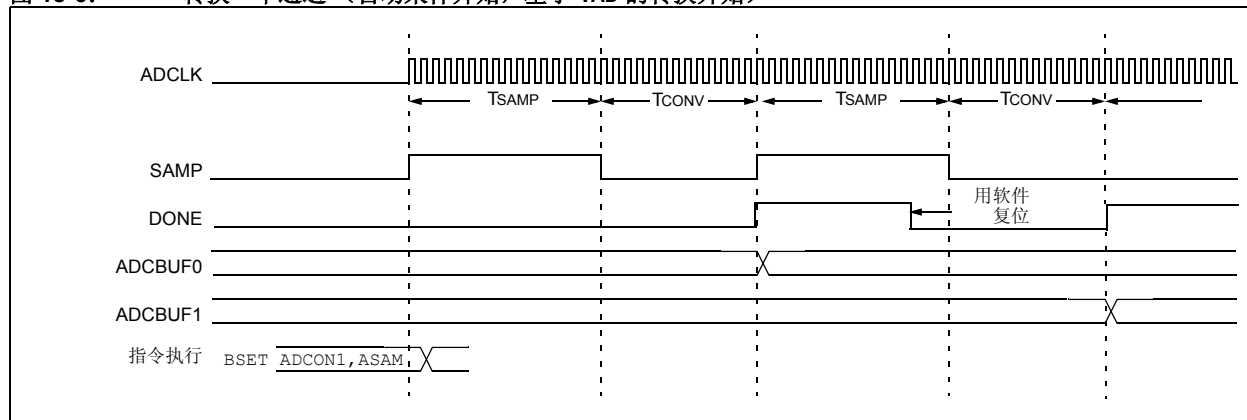
ADCON1bits.ADON = 1;         // turn ADC ON
while (1)                    // repeat continuously
{
    ADCON1bits.SAMP = 1;     // start sampling then ...
                               // after 31Tad go to conversion
    while (!ADCON1bits.DONE); // conversion done?
    ADCValue = ADCBUF0;      // yes then get ADC value
}                             // repeat// repeat
```


18.11.2.1 自由运行采样转换过程

如图 18-6 所示，使用自动转换的转换触发源模式（SSRC = 111），配合自动采样开始模式（ASAM = 1），可以安排采样 / 转换过程，而无需用户干预或其他器件资源。此“计时”模式允许在模块启动后进行连续数据收集。

注：转换速率为 200 ksps 时应使能此 A/D 配置（详情请参见第 18.21 节“A/D 转换速度”）。

图 18-6: 转换一个通道（自动采样开始，基于 TAD 的转换开始）



18.11.2.2 使用计时转换触发和自动采样的采样时间注意事项

用户必须确保采样时间超过第 18.15 节“A/D 采样要求”中规定的采样要求。

假设模块设置为自动采样并使用计时转换触发，采样间隔由 SAMC 位指定。

18.11.3 事件触发转换开始

能够将采样结束和转换开始与某个其他时间事件同步通常是比较理想的。A/D 模块可以使用三个触发源之一作为转换触发事件。

18.11.3.1 外部 INT 引脚触发

如果 SSRC<2:0> = 001 时，A/D 转换是由 INT0 引脚上的有效电平转换触发的。INT0 引脚可以编程为上升沿输入或下降沿输入。

18.11.3.2 通用定时器比较触发

通过将 SSRC<2:0> 置为 010，可将 A/D 配置为此触发模式。如果在 32 位定时器 TMR3/TMR2 和 32 位组合周期寄存器 PR3/PR2 之间发生匹配，则 Timer3 会产生一个特殊的 ADC 触发事件信号。TMR5/TMR4 这对定时器不具备此功能。更多详细信息请参见第 12 章“定时器”。

18.11.3.3 电机控制 PWM 触发器

PWM 模块有一个事件触发器，可以让 A/D 转换与 PWM 时基同步。当 SSRC<2:0> = 011 时，A/D 采样和转换时间发生在 PWM 周期中的任何用户可编程点。特殊事件触发器可以让用户将获得 A/D 转换结果与占空比值更新之间的延时减到最少。更多详细信息请参见第 15 章“电机控制 PWM”。

18.11.3.4 A/D 工作与内部或外部事件同步

发生外部事件触发脉冲结束采样并启动转换（SSRC = 001、010 或 011）的模式可以与自动采样（ASAM = 1）组合使用，以使 A/D 采样转换事件与触发脉冲源同步。例如，在图 18-8 SSRC = 010 和 ASAM = 1 的情况下，A/D 将总是让结束采样和启动转换与定时器比较触发事件同步。A/D 将有与定时器比较事件速率对应的采样转换速率。

图 18-7: 手动采样开始，基于转换触发的转换开始

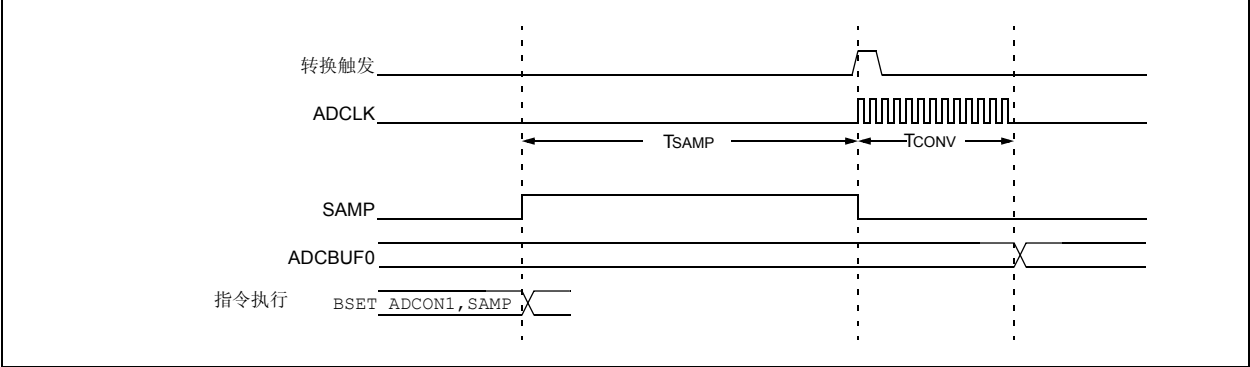
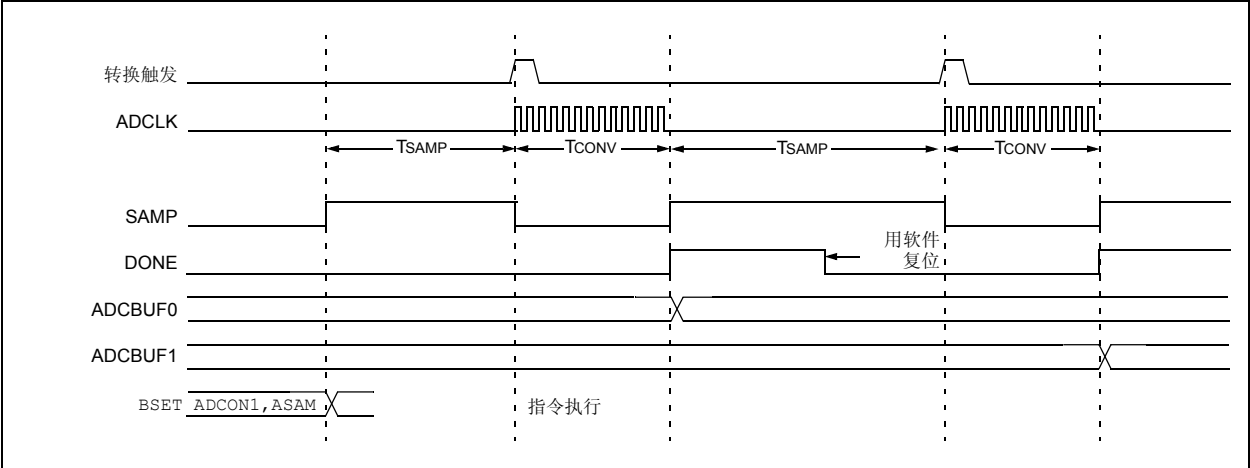


图 18-8: 自动采样开始，基于转换触发的转换开始



18.11.3.5 自动采样 / 转换过程的采样时间注意事项

不同的采样 / 转换过程为 S/H 通道提供不同的采样时间来采集模拟信号。用户必须确保采样时间超过第 18.15 节“A/D 采样要求”中给出的采样要求。

假设该模块设置为自动采样并把外部触发脉冲用作转换触发，采样间隔是触发脉冲间隔的一部分。

采样时间为触发脉冲周期与完成转换所需的时间之差。

公式 18-3: 采样时间，顺序采样

$$TSMP = \text{触发脉冲间隔 (TSEQ)} - \text{转换时间 (TCONV)}$$
$$TSMP = TSEQ - TCONV$$

注：TSEQ 是触发脉冲间隔时间。

18.12 采样 / 转换工作的控制

应用软件可以查询 SAMP 和 CONV 位的状态来跟踪 A/D 工作，或者当转换完成时模块可能中断 CPU。必要时应用软件还可以中止 A/D 工作。

18.12.1 监视采样 / 转换状态

SAMP (ADCON1<1>) 和 CONV (ADCON1<0>) 位分别表示 A/D 的采样状态和转换状态。通常，SAMP 位清零表示采样结束，CONV 位自动置 1 表示转换开始。如果 SAMP 和 CONV 都为“0”，则 A/D 处于无效状态。在有些工作模式下，SAMP 位还可以启动和终止采样，而 CONV 位可以终止转换。

18.12.2 产生 A/D 中断

SMPI<3:0> 位控制中断的产生。启动采样之后，当经过了数个采样 / 转换过程后将发生中断，并在每次经过相同数量的采样之后重新发生中断。

SMPI 位指定的值将对应缓冲器中的数据样本数量，最大可达 16 个。

不能使用 SMPI 位来禁止 A/D 中断。要禁止该中断，请将 ADIE 模拟模块中断使能位清零。

18.12.3 中止采样

在手动采样模式下清零 SAMP 位将终止采样，但是如果 SSRC = 000，还可以启动转换。

在自动采样模式下清零 ASAM 位将不会终止正在进行的采样 / 转换过程，然而，在随后的转换进行之后，采样不会自动恢复。

18.12.4 中止转换

在转换过程中清零 ADON 位将中止当前的转换。不会用部分完成的 A/D 转换样本来更新 A/D 结果寄存器对。即，对应的 ADCBUF 缓冲器单元将仍然保持上一次转换完成后的值（即上一次写入该缓冲器的值）。

18.13 如何将转换结果写入缓冲器的说明

当转换完成之后，模块将转换结果写入 A/D 结果缓冲器中。该缓冲器是 16 个 12 位字的 RAM 阵列。可以通过 SFR 空间内的 16 个地址单元访问该缓冲器，这些单元的名称分别为 ADCBUF0 ADCBUFF。

用户软件可能试图在每个 A/D 转换结果产生时读取它，然而，这可能会消耗太多的 CPU 时间。通常，为了简化代码，模块会用结果填充该缓冲器，然后当该缓冲器满时产生中断。

18.13.1 每次中断前的转换次数

SMPI<3:0> 位 (ADCON2<5:2>) 选择在中断 CPU 前将发生多少次 A/D 转换。每个中断前的采样数量可以为 1 到 16 之间的任何值。每次中断后, A/D 转换器模块总是开始将其转换结果写入缓冲器的起始单元。例如, 如果 SMPI<3:0> = 0000, 那么总是将转换结果写入 ADCBUF0。在这个例子中, 不会用到其他缓冲器单元。

18.13.2 缓冲器大小造成的限制

当 BUFM 位 (ADCON2<1>) 为 1 时, 用户不能把 SMPI 位编程为一个指定在每次中断前进行 8 次以上转换的值。BUFM 位的功能描述如下。

18.13.3 缓冲器填充模式

当 BUFM 位 (ADCON2<1>) 为 1 时, 16 字结果缓冲器 (ADRES) 将拆分成两个 8 字组。每次中断事件发生后, 这两个 8 字缓冲器会交替地接收转换结果。BUFM 置位后首先使用的 8 字缓冲器位于 ADCBUF 的低地址。当 BUFM 为 0 时, 所有转换过程将使用完整的 16 字缓冲器。

使用 BUFM 功能与否取决于中断后有多少时间可用来移动缓冲器的内容, 这是由应用决定的。如果处理器能够在采样和转换一个通道所需的时间内快速地卸载满载的缓冲器, 可以将 BUFM 位设置为 0, 从而可以在每次中断前进行多达 16 次转换。在第一个缓冲器单元被改写前, 处理器会有一段采样和转换时间。

如果在这段采样和转换时间内处理器不能卸载缓冲器, BUFM 位应该设置为 1。例如, 如果 SMPI<3:0> = 0111, 那么 8 个转换结果将被装入半个缓冲器, 随后将发生中断。紧接着的 8 个转换结果将被装入该缓冲器的另一半。从而处理器在中断之间有整段时间将 8 个转换结果移出缓冲器。

18.13.4 缓冲器填充状态

使用 BUFM 控制位拆分转换结果缓冲器时, BUFS 状态位 (ADCON2<7>) 表示 A/D 转换器当前正在填充的半个缓冲器。如果 BUFS = 0, 那么 A/D 转换器正在填充 ADCBUF0 至 ADCBUF7, 并且用户软件应该从 ADCBUF8 至 ADCBUFF 读转换值。如果 BUFS = 1, 那么情形正好相反, 并且用户软件应该从 ADCBUF0 至 ADCBUF7 读转换值。

18.14 转换过程示例

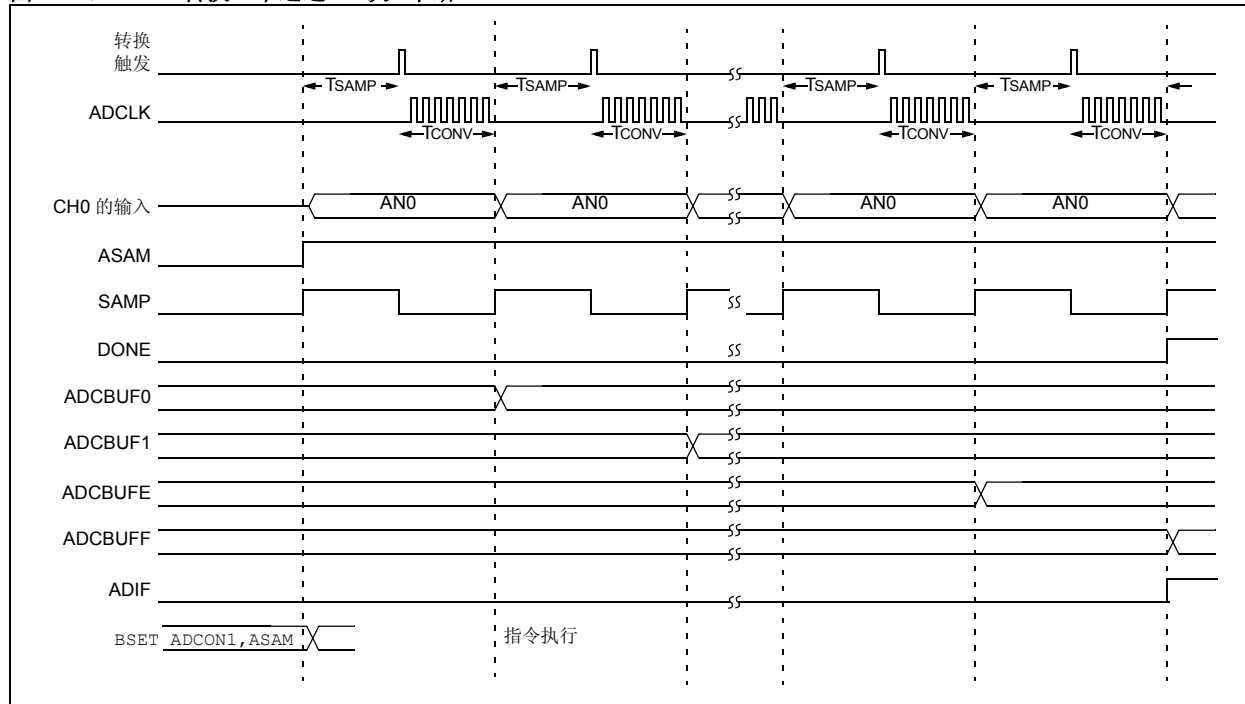
下面的配置示例说明了在不同采样和缓冲配置下的 A/D 工作。在各个示例中，将 ASAM 位置 1 会启动自动采样。转换触发结束采样并启动转换。

18.14.1 示例：单个通道的多次采样和转换

图 18-9 和表 18-1 举例说明了 A/D 的基本配置。在该情形下，将对一个 A/D 输入（AN0）采样和转换。转换结果存储在 ADCBUF 缓冲器中。此过程重复 16 次直到缓冲器满为止，然后模块产生中断。然后将重复整个过程。

ALTS 位清零，只有 MUX A 输入有效。CH0SA 位和 CH0NA 位指定 AN0 到 VREF- 为采样 / 保持通道的输入。不使用所有其他输入选择位。

图 18-9: 转换一个通道 16 次 / 中断



例 18-3: 多次采样和转换单个通道的代码示例

```
ADPCFG = 0xFFFFB;           // all PORTB = Digital; RB2 = analog
ADCON1 = 0x00E0;             // SSRC bit = 111 implies internal
                                // counter ends sampling and starts
                                // converting.
ADCHS = 0x0002;               // Connect RB2/AN2 as CH0 input ..
                                // in this example RB2/AN2 is the input
ADCSSL = 0;
ADCON3 = 0x0F00;             // Sample time = 15Tad, Tad = internal Tcy/2
ADCON2 = 0x003C;             // Interrupt after every 16 samples

ADCON1bits.ADON = 1;         // turn ADC ON
while (1)                    // repeat continuously
{
    ADCValue = 0;             // clear value
    ADC16Ptr = &ADCBUF0;     // initialize ADCBUF pointer
    IFS0bits.ADIF = 0;        // clear ADC interrupt flag
    ADCON1bits.ASAM = 1;       // auto start sampling
                                // for 31Tad then go to conversion
    while (!IFS0bits.ADIF);    // conversion done?
    ADCON1bits.ASAM = 0;       // yes then stop sample/convert
    for (count = 0; count < 16; count++) // average the 16 ADC value
        ADCValue = ADCValue + *ADC16Ptr++;
    ADCValue = ADCValue >> 4;
}                             // repeat
```

表 18-1: 转换一个通道 16 次 / 中断
 控制位
 过程选择

SMPI<2:0> = 1111	第 16 次采样后中断
BUFM = 0	单个 16 字结果缓冲器
ALTS = 0	总是使用 MUX A 输入选择
MUX A 输入选择	
CH0SA<3:0> = 0000	选择 AN0 作为 CH0+ 输入
CH0NA = 0	选择 VREF- 作为 CH0- 输入
CSCNA = 0	无输入扫描
CSSL<15:0> = n/a	扫描输入选择未使用
MUX B 输入选择	
CH0SB<3:0> = n/a	通道 CH0+ 输入未使用
CH0NB = n/a	通道 CH0- 输入未使用

工作过程	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0x0	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0x1	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0x2	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0x3	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0x4	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0x5	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0x6	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0x7	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0x8	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0x9	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0xA	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0xB	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0xC	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0xD	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0xE	
采样 MUX A 输入: AN0->CH0	
转换 CH0, 写缓冲器 0xF	
中断	
重复	

缓冲器地址	第一次中断时的缓冲器
ADCBUF0	AN0 采样 1
ADCBUF1	AN0 采样 2
ADCBUF2	AN0 采样 3
ADCBUF3	AN0 采样 4
ADCBUF4	AN0 采样 5
ADCBUF5	AN0 采样 6
ADCBUF6	AN0 采样 7
ADCBUF7	AN0 采样 8
ADCBUF8	AN0 采样 9
ADCBUF9	AN0 采样 10
ADCBUFA	AN0 采样 11
ADCBUFB	AN0 采样 12
ADCBUFC	AN0 采样 13
ADCBUFD	AN0 采样 14
ADCBUFE	AN0 采样 15
ADCBUFF	AN0 采样 16

第二次中断时的缓冲器
AN0 采样 17
AN0 采样 18
AN0 采样 19
AN0 采样 20
AN0 采样 21
AN0 采样 22
AN0 采样 23
AN0 采样 24
• • •
AN0 采样 25
AN0 采样 26
AN0 采样 27
AN0 采样 28
AN0 采样 29
AN0 采样 30
AN0 采样 31
AN0 采样 32

18.14.2 示例：扫描所有模拟输入时的 A/D 转换

图 18-10 和表 18-2 举例说明了一种典型设置，其中所有可用的模拟输入通道被采样和转换。置 1 的 CSCNA 位指定对作为 CH0 正输入的 A/D 输入进行扫描。其他情况与第 18.14.1 节类似。

首先，AN0 输入被 CH0 采样，然后转换。转换结果存储在 ADCBUF 缓冲器中。然后 AN1 输入被采样并转换。此扫描输入的过程重复 16 次直到缓冲器满为止，然后该模块产生中断。然后将重复整个过程。

图 18-10: 扫描 16 个输入 / 中断

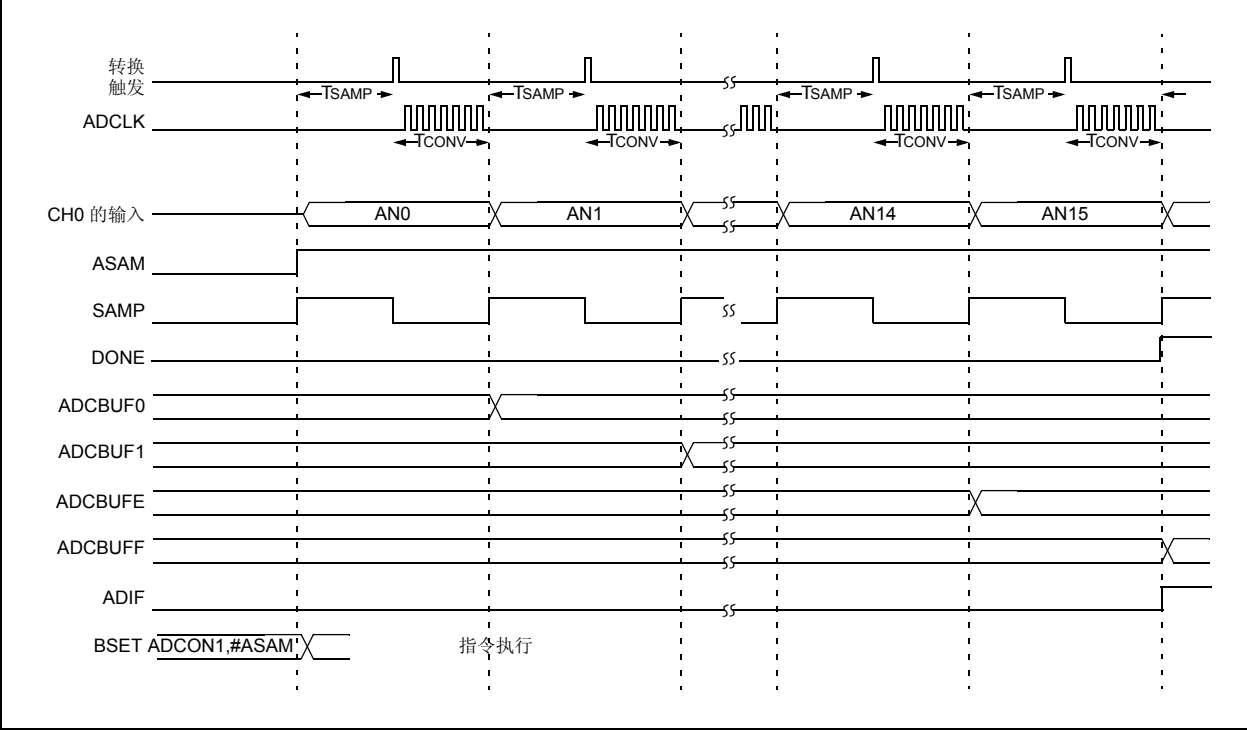


表 18-2: 扫描 16 个输入 / 中断

控制位 过程选择	
SMPI<2:0> = 1111	第 16 次采样后中断
BUFM = 0	单个 16 字结果缓冲器
ALTS = 0	总是使用 MUX A 输入选择
MUX A 输入选择	
CH0SA<3:0> = n/a	被 CSCNA 覆盖
CH0NA = 0	选择 VREF- 作为 CH0- 输入
CSCNA = 1	扫描 CH0+ 输入
CSSL<15:0> = 1111 1111 1111 1111	扫描所有输入
MUX B 输入选择	
CH0SB<3:0> = n/a	通道 CH0+ 输入未使用
CH0NB = n/a	通道 CH0- 输入未使用

工作过程

采样 MUX A 输入: AN0->CH0
转换 CH0, 写缓冲器 0x0
采样 MUX A 输入: AN1->CH0
转换 CH0, 写缓冲器 0x1
采样 MUX A 输入: AN2->CH0
转换 CH0, 写缓冲器 0x2
采样 MUX A 输入: AN3->CH0
转换 CH0, 写缓冲器 0x3
采样 MUX A 输入: AN4->CH0
转换 CH0, 写缓冲器 0x4
采样 MUX A 输入: AN5->CH0
转换 CH0, 写缓冲器 0x5
采样 MUX A 输入: AN6->CH0
转换 CH0, 写缓冲器 0x6
采样 MUX A 输入: AN7->CH0
转换 CH0, 写缓冲器 0x7
采样 MUX A 输入: AN8->CH0
转换 CH0, 写缓冲器 0x8
采样 MUX A 输入: AN9->CH0
转换 CH0, 写缓冲器 0x9
采样 MUX A 输入: AN10->CH0
转换 CH0, 写缓冲器 0xA
采样 MUX A 输入: AN11->CH0
转换 CH0, 写缓冲器 0xB
采样 MUX A 输入: AN12->CH0
转换 CH0, 写缓冲器 0xC
采样 MUX A 输入: AN13->CH0
转换 CH0, 写缓冲器 0xD
采样 MUX A 输入: AN14->CH0
转换 CH0, 写缓冲器 0xE
采样 MUX A 输入: AN15->CH0
转换 CH0, 写缓冲器 0xF
中断
重复

缓冲器 地址	第一次中断时 的缓冲器
ADCBUF0	AN0 采样 1
ADCBUF1	AN1 采样 2
ADCBUF2	AN2 采样 3
ADCBUF3	AN3 采样 4
ADCBUF4	AN4 采样 5
ADCBUF5	AN5 采样 6
ADCBUF6	AN6 采样 7
ADCBUF7	AN7 采样 8
ADCBUF8	AN8 采样 9
ADCBUF9	AN9 采样 10
ADCBUFA	AN10 采样 11
ADCBUFB	AN11 采样 12
ADCBUFC	AN12 采样 13
ADCBUFD	AN13 采样 14
ADCBUFE	AN14 采样 15
ADCBUFF	AN15 采样 16

第二次中断时 的缓冲器
AN0 采样 17
AN1 采样 18
AN2 采样 19
AN3 采样 20
AN4 采样 21
AN5 采样 22
AN6 采样 23
AN7 采样 24
AN8 采样 25
AN9 采样 26
AN10 采样 27
AN11 采样 28
AN12 采样 29
AN13 采样 30
AN14 采样 31
AN15 采样 32

• • •

18.14.3 示例：使用双 8 字缓冲器

想要获取使用双缓冲器的示例，请参见第 17 章 “10 位 A/D 转换器” 中的第 17.15.4 节。

18.14.4 示例：使用交替多路开关 A、多路开关 B 输入选择

想要获取使用多路开关 A 和多路开关 B 输入选择的示例，参见第 17 章 “10 位 A/D 转换器” 中的第 17.15.5 节。

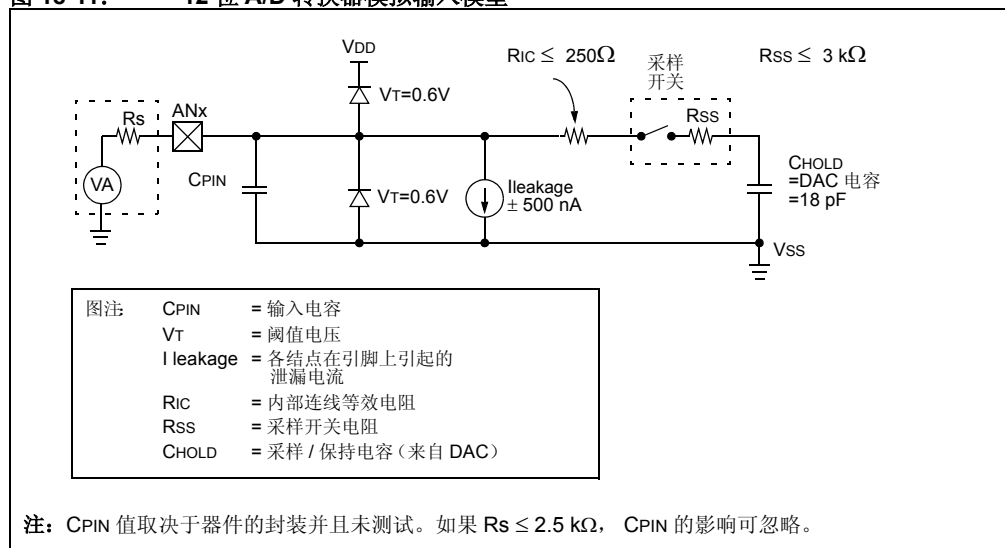
18.15 A/D 采样要求

图 18-11 所示为 12 位 A/D 转换器的模拟输入模型。A/D 的总采样时间是内部放大器稳定时间和保持电容充电时间的函数。

为了使 A/D 转换器达到规定的精度，必须让充电保持电容（CHOLD）充分充电至模拟输入引脚上的电平。源阻抗（ R_s ）、内部连线等效阻抗（ R_{ic} ）和内部采样开关（ R_{ss} ）阻抗共同直接影响电容 CHOLD 充电所需的时间。所以模拟源总的阻抗应该足够小以在选择的采样时间内对保持电容充分充电。为了把引脚泄漏电流对 A/D 转换器精度的影响降到最低，最大建议源阻抗 R_s 为 2.5 k Ω 。当选择（改变）了模拟输入通道后，采样工作必须在启动转换前完成。在每次采样工作前，内部保持电容将处于放电状态。

两次转换之间应该至少留出 1 个 T_{AD} 时间周期作为采样时间。更多详细信息，参见器件电气规范。

图 18-11: 12 位 A/D 转换器模拟输入模型



18.16 读取 A/D 转换结果缓冲器

该 RAM 为 12 位宽，当对该缓冲器执行读操作时，数据自动格式化为四种可选格式中的一种格式。FORM<1:0> 位（ADCON1<9:8>）选择格式。格式化硬件在数据总线上为所有数据格式提供一个 16 位结果。图 18-12 显示了可以使用 FORM<1:0> 控制位选择的数据输出格式。

图 18-12: A/D 输出数据格式

RAM 内容:	<table><tr><td>d11</td><td>d10</td><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td></tr></table>												d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00
d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00													
读出到总线:																								
整数	<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>d11</td><td>d10</td><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td></tr></table>	0	0	0	0	d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00							
0	0	0	0	d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00									
有符号整数	<table><tr><td>$\overline{\text{d11}}$</td><td>$\overline{\text{d11}}$</td><td>$\overline{\text{d11}}$</td><td>$\overline{\text{d11}}$</td><td>$\overline{\text{d11}}$</td><td>d10</td><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td></tr></table>	$\overline{\text{d11}}$	$\overline{\text{d11}}$	$\overline{\text{d11}}$	$\overline{\text{d11}}$	$\overline{\text{d11}}$	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00							
$\overline{\text{d11}}$	$\overline{\text{d11}}$	$\overline{\text{d11}}$	$\overline{\text{d11}}$	$\overline{\text{d11}}$	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00									
小数	<table><tr><td>d11</td><td>d10</td><td>d09</td><td>d08</td><td>d07</td><td>d06</td><td>d05</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0							
d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0									
有符号小数（1.15）	<table><tr><td>$\overline{\text{d11}}$</td><td>d10</td><td>d09</td><td>d08</td><td>d07</td><td>d04</td><td>d03</td><td>d02</td><td>d01</td><td>d00</td><td>d01</td><td>d00</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	$\overline{\text{d11}}$	d10	d09	d08	d07	d04	d03	d02	d01	d00	d01	d00	0	0	0	0							
$\overline{\text{d11}}$	d10	d09	d08	d07	d04	d03	d02	d01	d00	d01	d00	0	0	0	0									

表 18-3: 不同结果数码的等价数值

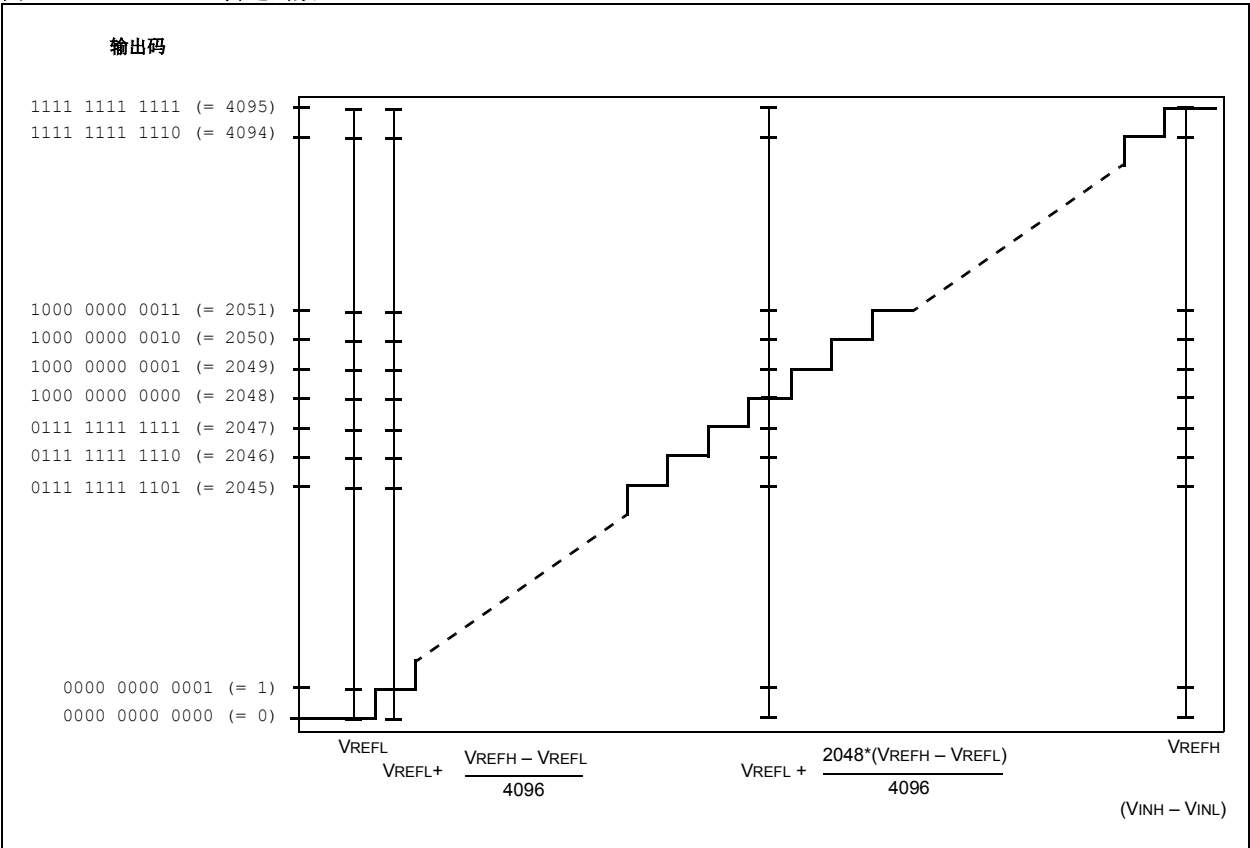
VIN/VREF	12 位 输出码	16 位无符号 整数格式	16 位有符号 整数格式	16 位无符号 小数格式	16 位有符号 小数格式
4095/4096	1111 1111 1111	0000 1111 1111 1111 = 4095	0000 0111 1111 1111 = 2047	1111 1111 1111 0000 = 0.9998	0111 1111 1111 0000 = 0.9995
4094/4096	1111 1111 1110	0000 1111 1111 1110 = 4094	0000 0111 1111 1110 = 2046	1111 1111 1110 0000 = 0.9995	0111 1111 1110 0000 = 0.9990
...					
2049/4096	1000 0000 0001	0000 1000 0000 0001 = 2049	0000 0000 0000 0001 = 1	1000 0000 0001 0000 = 0.5002	0000 0000 0001 0000 = 0.0005
2048/4096	1000 0000 0000	0000 1000 0000 0000 = 2048	0000 0000 0000 0000 = 0	1000 0000 0000 0000 = 0.5000	0000 0000 0000 0000 = 0.0000
2047/4096	0111 1111 1111	0000 0111 1111 1111 = 2047	1111 1111 1111 1111 = -1	0111 1111 1111 0000 = 0.4998	1111 1111 1111 0000 = -0.0005
...					
1/4096	0000 0000 0001	0000 0000 0000 0001 = 1	1111 1000 0000 0001 = -2047	0000 0000 0001 0000 = 0.0002	1000 0000 0001 0000 = -0.9995
0/4096	0000 0000 0000	0000 0000 0000 0000 = 0	1111 1000 0000 0000 = -2048	0000 0000 0000 0000 = 0.0000	1000 0000 0000 0000 = -1.0000

18.17 传递函数

图 18-13 所示为 A/D 转换器的理想传递函数。输入电压差值 ($V_{INH} - V_{INL}$) 与参考电压差值 ($V_{REFH} - V_{REFL}$) 作比较。

- 当输入电压为 $(V_{REFH} - V_{REFL}/8192)$ 或 0.5 LSb 时发生第一个数码转换。
- 数码 00 0000 0001 被置于 $(V_{REFH} - V_{REFL}/4096)$ 或 1.0 LSb 中央。
- 数码 10 0000 0000 被置于 $(2048 \times (V_{REFH} - V_{REFL})/4096)$ 中央。
- 小于 $(1 \times (V_{REFH} - V_{REFL})/8192)$ 的输入电压被转换为 00 0000 0000。
- 大于 $(8192 \times (V_{REFH} - V_{REFL})/8192)$ 的输入被转换为 11 1111 1111。

图 18-13: A/D 传递函数



18.18 A/D 转换的精度 / 误差

如要获取讨论 A/D 精度的文档列表，请参见第 18.26 节“相关应用笔记”。

18.19 连接注意事项

因为模拟输入使用 ESD 保护，它们有连接到 V_{DD} 和 V_{SS} 的二极管。这就要求模拟输入电压必须介于 V_{DD} 和 V_{SS} 之间。如果输入电压超出此范围 $0.3V$ 以上（任一方向上），就会有一个二极管正向偏置，而且如果超过输入电流规范可能会损坏器件。

有时可外接 RC 滤波器来进行输入信号的抗混叠滤波。R 元件的选择要确保满足采样时间要求。任何通过高阻抗连接到模拟输入引脚上的外部元件（如电容和齐纳二极管等），在引脚上的泄漏电流都应极小。

18.20 初始化

例 18-4 为 A/D 模块的简单初始化代码示例。

在这个特定的配置中，所有 16 个模拟输入引脚（AN0 到 AN15）被设置为模拟输入。空闲模式下的工作被禁止，输出数据为无符号小数格式，AVDD 和 AVSS 用作 VREFH 和 VREFL。采样的启动和转换（转换触发）的启动通过软件手动执行。禁止对输入扫描，每次采样 / 转换过程（1 个转换结果）后产生中断。A/D 转换时钟为 Tcy/2；转换 AN0。

因为在每次转换完成后通过将 SAMP 位（ADCON1<1>）置 1 手动启动采样，所以自动采样时间位 SAMC<4:0>（ADCON3<12:8>）被忽略。此外，因为转换的启动也是手动触发的，所以每次需要转换新的采样时需要清零 SAMP 位。

例 18-4: A/D 初始化代码示例

```

CLR      ADPCFG          ; Configure A/D port,
                        ; all input pins are analog

MOV      #0x2200,W0
MOV      W0,ADCON1       ; Configure sample clock source
                        ; and conversion trigger mode.
                        ;   Unsigned Fractional format,
                        ;   Manual conversion trigger,
                        ;   Manual start of sampling,
                        ;   No operation in IDLE mode.

CLR      ADCON2          ; Configure A/D voltage reference
                        ; and buffer fill modes.
                        ;   VREF from AVDD and AVSS,
                        ;   Inputs are not scanned,
                        ;   Interrupt every sample

CLR      ADCON3          ; Configure A/D conversion clock

CLR      ADCHS           ; Configure input channels,
                        ;   CH0+ input is AN0.
                        ;   CH0- input is VREFL (AVss)

CLR      ADCSSL          ; No inputs are scanned.

BCLR     IFS0,#ADIF      ; Clear A/D conversion interrupt flag

; Configure A/D interrupt priority bits (ADIP<2:0>) here, if
; required.(default priority level is 4)

BSET     IEC0,#ADIE      ; Enable A/D conversion interrupt

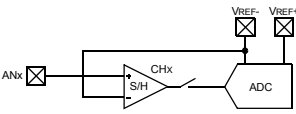
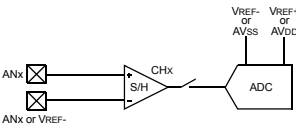
BSET     ADCON1,#ADON     ; Turn on A/D
BSET     ADCON1,#SAMP     ; Start sampling the input
CALL     DELAY            ; Ensure the correct sampling time has
                        ; elapsed before starting conversion.
BCLR     ADCON1,#SAMP     ; End A/D Sampling and start Conversion
:                                     ; The DONE bit is set by hardware when
:                                     ; conversion sequence is complete.
:                                     ; The ADIF bit will be set.

```

18.21 A/D 转换速度

dsPIC30F 12 位 A/D 转换器规范允许的最大采样速率为 200 ksps。下表对 dsPIC30F 12 位 A/D 转换器的转换速度和所需的工作条件作了小结。

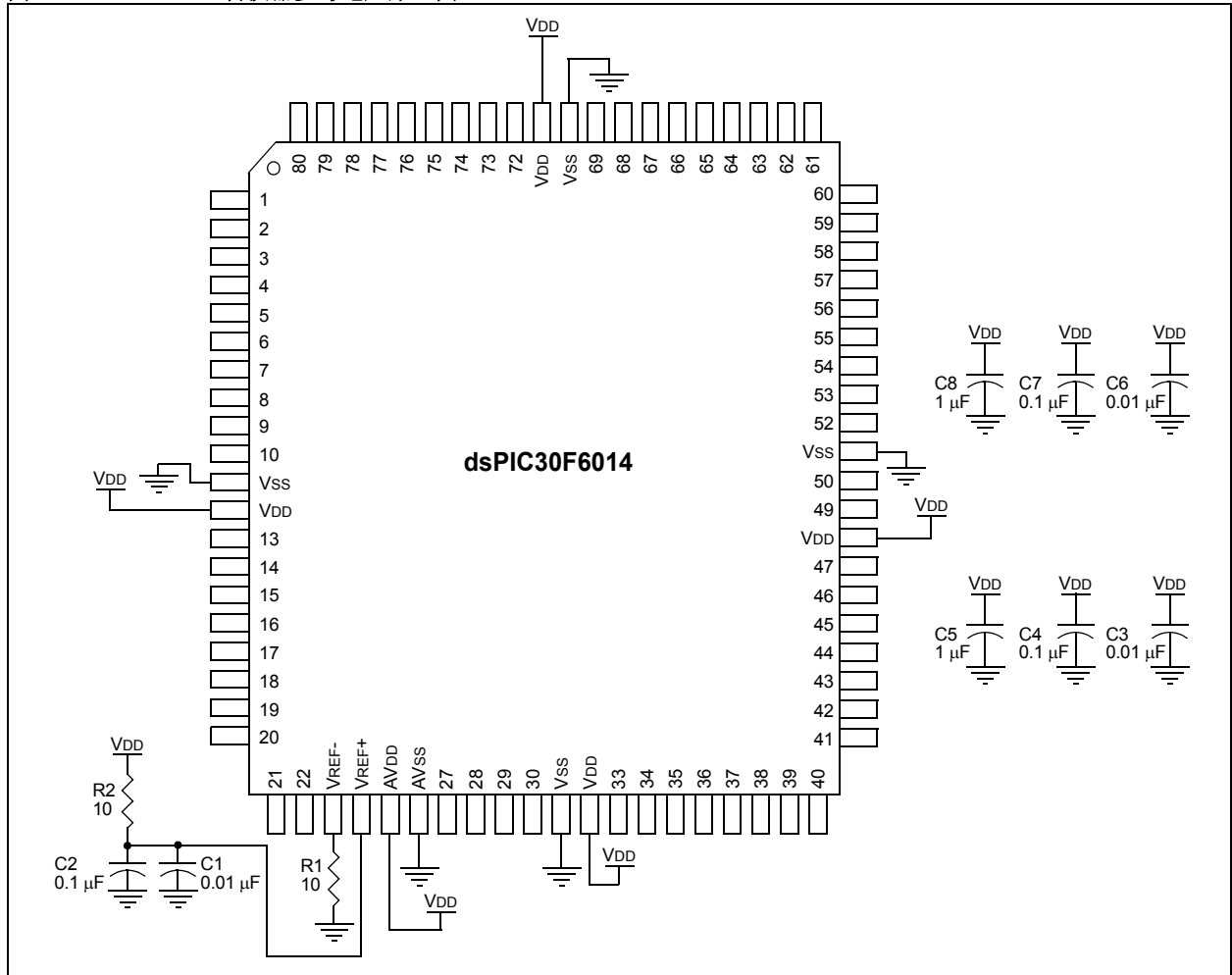
表 18-4: 12 位 A/D 转换器扩展转换速率

dsPIC30F 12 位 A/D 转换器转换速率						
A/D 速度	最小 TAD	最小采样时间	最大 R _s	VDD	温度	A/D 通道配置
小于等于 200 ksps ⁽¹⁾	333.33 ns	1 TAD	2.5 kΩ	4.5V~5.5V	-40°C~+85°C	
小于等于 100 ksps	666.67 ns	1 TAD	2.5 kΩ	3.0V~5.5V	-40°C~+125°C	

注 1: 欲正常工作应使用外部 VREF- 和 VREF+ 引脚。建议电路见图 18-14。

下图为转换速率高于 100 ksp/s 时的建议电路。图中以 dsPIC30F6014 为例。

图 18-14: A/D 转换器参考电压原理图



以下配置步骤为转换速度高于 100 ksp/s 时所需的设置值。

18.21.1 200 ksp/s 配置指南

必须进行如下配置以达到 200 ksp/s 的转换速率。

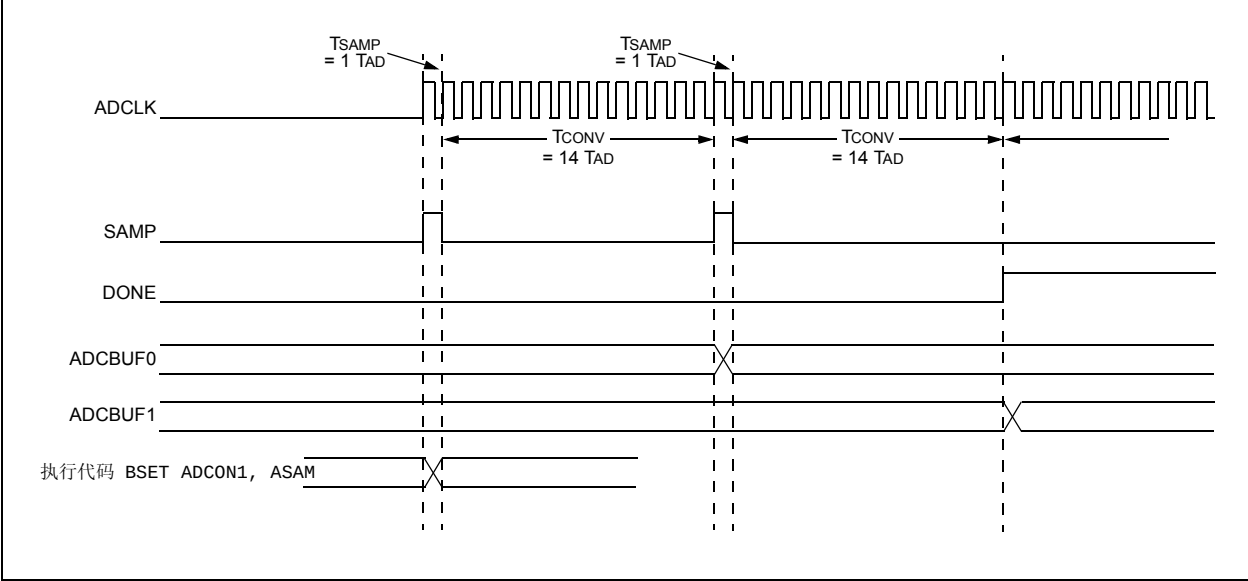
- 符合表 18-4 中所述条件。
- 根据图 18-14 中的建议电路所示连接外部 VREF+ 和 VREF- 引脚。
- 在寄存器 ADCON1 中置 SSRC<2:0> = 111 使能自动转换选项。
- 将寄存器 ADCON1 中的 ASAM 控制位置 1 使能自动采样。
- 在寄存器 ADCON2 中的 SMPI<3:0> 控制位写入所需的中断之间的转换次数。
- 写入寄存器 ADCON3 中的 ADCS<5:0> 控制位，将 A/D 时钟周期配置为：

$$\frac{1}{(14 + 1) \times 200,000} = 333.33 \text{ ns}$$

- 置 SAMC<4:0> = 00001 将采样时间配置为 1 TAD。

下图为 A/D 转换速率为 200 ksp/s 的时序图。所选的 TAD 以及根据上述要求进行的配置将使转换速度达到 200 ksp/s。代码示例见例 18-5。

图 18-15: 转换一个通道（转换速率 200 ksps，自动采样开始，采样时间 1 TAD）



例 18-5: 转换（速率为 200 ksps，自动采样开始，采样时间 1 TAD）的代码示例

```
ADPCFG = 0xFFFB;           // all PORTB = Digital; RB2 = analog
ADCON1 = 0x00E0;           // SSRC bit = 111 implies internal
                             // counter ends sampling and starts
                             // converting.
ADCHS = 0x0002;            // Connect RB2/AN2 as CH0 input
                             // in this example RB2/AN2 is the input
ADCSSL = 0;
ADCON3 = 0x0113;           // Sample time = 1Tad, Tad = 333.33 ns @ 30 MIPS
                             // which will give 1 / (15 * 333.33 ns) = 200 ksps
ADCON2 = 0x6004;           // Select external VREF+ and VREF- pins
                             // Interrupt after every 2 samples
ADCON1bits.ADON = 1;       // turn ADC ON
while (1)                  // repeat continuously
{
    ADCValue = 0;           // clear value
    ADC16Ptr = &ADCBUF0;   // initialize ADCBUF pointer
    IFS0bits.ADIF = 0;     // clear ADC interrupt flag
    ADCON1bits.ASAM = 1;   // auto start sampling
                             // for 31Tad then go to conversion
    while (!IFS0bits.ADIF); // conversion done?
    ADCON1bits.ASAM = 0;   // yes then stop sample/convert
    for (count = 0; count < 2; count++) // average the 2 ADC value
        ADCValue = ADCValue + *ADC16Ptr++;
    ADCValue = ADCValue >> 1;
}
```


18.22 在休眠和空闲模式下工作

休眠和空闲模式有助于将转换噪声降至最小，因为 CPU、总线和其他外设的数字活动被减到最少。

18.22.1 不使用 RC A/D 时钟的 CPU 休眠模式

当器件进入休眠模式时，该模块的所有时钟源被关闭并保持逻辑“0”状态。

如果在一次转换过程中进入休眠状态，转换会中止，除非 A/D 将内部 RC 时钟发生器作为时钟源。从休眠模式退出时，转换器将不会恢复尚未完成的转换。

器件进入或离开休眠模式不会影响寄存器的内容。

18.22.2 使用 RC A/D 时钟的 CPU 休眠模式

如果将内部 A/D RC 振荡器设置为 A/D 时钟源 ($ADRC = 1$)，A/D 模块可以在休眠模式下工作。这可以消除转换中的数字开关噪声。转换完成后，CONV 位将被清零，转换结果送入 A/D 结果缓冲器 ADCBUF。

如果允许 A/D 中断 ($ADIE = 1$)，A/D 中断发生时器件将从休眠模式唤醒。如果 A/D 中断的优先级高于当前 CPU 的优先级，程序执行将在 A/D 中断服务程序执行后恢复。否则，程序将从使器件处于休眠模式的 PWRSAV 指令后的指令继续执行。

如果仍然禁止 A/D 中断，即使 ADON 保持置位，也还是会关闭 A/D 模块。

为了将数字噪声对 A/D 模块工作的影响降至最低，用户应该选择转换触发源以确保 A/D 转换可在休眠模式下进行。自动转换触发选项可用于休眠模式下的采样和转换 ($SSRC<2:0> = 111$)。要使用自动转换选项，应该在 PWRSAV 指令前的指令中将 ADON 位置 1。

注： 为了使 A/D 模块可以在休眠模式下工作，必须将 RC 设置为 A/D 时钟源 ($ADRC = 1$)。

18.22.3 CPU 空闲模式下的 A/D 工作

ADSIDL 位 ($ADCON1<13>$) 选择 A/D 模块在空闲模式下是停止工作还是继续工作。如果 ADSIDL = 0，当器件进入空闲模式时，该模块将继续正常工作。如果允许 A/D 中断 ($ADIE = 1$)，A/D 中断发生时器件将从空闲模式唤醒。如果 A/D 中断的优先级高于当前 CPU 的优先级，程序执行将在 A/D 中断服务程序执行后恢复。否则，程序将从使器件处于空闲模式的 PWRSAV 指令后的指令继续执行。

如果 ADSIDL = 1，在空闲模式下该模块将停止工作。如果器件在一次转换过程中进入空闲模式，该转换将中止。从空闲模式退出时，该转换器将不会恢复部分完成的转换。

18.23 复位的影响

器件复位强制所有寄存器进入复位状态。同时迫使 A/D 模块关闭并中止任何正在进行的转换。所有与模拟输入复用的引脚将被配置为模拟输入。相应的 TRIS 位将被置位。

上电复位时，不会将 ADCBUF 寄存器中的值初始化。ADCBUF0……ADCBUFF 中的值不确定。

18.24 与 12 位 A/D 转换器相关的特殊功能寄存器

下表列举了 dsPIC30F 12 位 A/D 转换器特殊功能寄存器，包括它们的地址和格式。所有未用的寄存器和 / 或寄存器中的位读作 0。

表 18-4: ADC 寄存器映射

寄存器名	ADR	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位状态	
INTCON1	0080	NSTDIS	—	—	—	—	OVATE	OVBTE	COVTE	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000 0000 0000 0000	
INTCON2	0082	ALTIVT	—	—	—	—	—	—	—	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP	0000 0000 0000 0000	
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0	0000 0000 0000 0000	
IEC0	008C	CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000	
IPC2	0098	—	ADIP<2:0>				—	U1TXIP<2:0>			—	U1RXIP<2:0>		—	SPI1IP<2:0>			0100 0100 0100 0100	
ADCBUF0	0280	ADC 数据缓冲器 0																uuuu uuuu uuuu uuuu	
ADCBUF1	0282	ADC 数据缓冲器 1																uuuu uuuu uuuu uuuu	
ADCBUF2	0284	ADC 数据缓冲器 2																uuuu uuuu uuuu uuuu	
ADCBUF3	0286	ADC 数据缓冲器 3																uuuu uuuu uuuu uuuu	
ADCBUF4	0288	ADC 数据缓冲器 4																uuuu uuuu uuuu uuuu	
ADCBUF5	028A	ADC 数据缓冲器 5																uuuu uuuu uuuu uuuu	
ADCBUF6	028C	ADC 数据缓冲器 6																uuuu uuuu uuuu uuuu	
ADCBUF7	028E	ADC 数据缓冲器 7																uuuu uuuu uuuu uuuu	
ADCBUF8	0290	ADC 数据缓冲器 8																uuuu uuuu uuuu uuuu	
ADCBUF9	0292	ADC 数据缓冲器 9																uuuu uuuu uuuu uuuu	
ADCBUFA	0294	ADC 数据缓冲器 10																uuuu uuuu uuuu uuuu	
ADCBUFB	0296	ADC 数据缓冲器 11																uuuu uuuu uuuu uuuu	
ADCBUFC	0298	ADC 数据缓冲器 12																uuuu uuuu uuuu uuuu	
ADCBUFD	029A	ADC 数据缓冲器 13																uuuu uuuu uuuu uuuu	
ADCBUFE	029C	ADC 数据缓冲器 14																uuuu uuuu uuuu uuuu	
ADCBUFF	029E	ADC 数据缓冲器 15																uuuu uuuu uuuu uuuu	
ADCON1	02A0	ADON	—	ADSIDL	—	—	—	FORM[1:0]		SSRC[2:0]			—	—	ASAM	SAMP	CONV	0000 0000 0000 0000	
ADCON2	02A2	VCFG[2:0]			—	—	CSCNA	—	—	BUFS	—	SMPI[3:0]				BUFM	ALTS	0000 0000 0000 0000	
ADCON3	02A4	—	—	—	SAMC[4:0]					ADRC	—	ADCS[5:0]						0000 0000 0000 0000	
ADCHS	02A6	—	—	—	CH0NB	CH0SB[3:0]					—	—	—	CH0NA	CH0SA[3:0]				0000 0000 0000 0000
ADPCFG	02A8	PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8	PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000 0000 0000 0000	
ADCSSL	02AA	ADC 输入扫描选择寄存器																0000 0000 0000 0000	

图注: u= 未知

注: 所有中断源及其相关的控制位在某个器件上可能不可用。详细信息请参见器件数据手册。

18.25 设计技巧

问 1: 如何优化 A/D 转换器的系统性能?

答:

1. 请确保您满足了所有时序规范要求。如果您关闭 A/D 模块后再打开, 应等待一个最小延时后再开始采样。如果改变了输入通道, 同样需要等待一个最小延时; 最后是 T_{AD} , 它是所选择的每一位的转换时间。它在 $ADCON3$ 中选择并且应该在电气特性中指定的一定范围之内。如果 T_{AD} 太短, 转换终止时有可能还未对数据进行完全转换; 如果 T_{AD} 太长, 采样电容上的电压会在转换结束前开始衰减。在器件数据手册的“电气规范”一章中提供了这些时序规范。
2. 模拟输入信号的源阻抗经常很高 (大于 $10\text{ k}\Omega$), 因此泄漏电流和为采样电容充电的电流 (均来自信号源) 可能会影响精度。如果输入信号的变化不是太快, 可以尝试在模拟输入端连接一个 $0.1\text{ }\mu\text{F}$ 的电容。该电容可充电到所采样的模拟电压, 并为内部 18 pF 的保持电容提供所需的瞬时充电电流。
3. 在启动 A/D 转换前请让器件进入休眠模式。在休眠模式下的转换需要选择 RC 时钟源。这个技术提高了精度, 因为来自 CPU 和其他外设的数字噪声会被降至最小。

问 2: 您知道有关于 A/D 的好的参考资料吗?

答: Prentice Hall 出版的“Analog-Digital Conversion Handbook”第三版 (ISBN 0-13-03-2848-0) 对理解 A/D 转换很有帮助。

问 3: 我的通道 / 采样和采样 / 中断组合超过了缓冲器的大小。该缓冲器会发生什么情况?

答: 不建议这种配置。缓冲器将包含不确定的转换结果。

18.26 相关应用笔记

本节列出了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是专门为 dsPIC30F 产品系列编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与 12 位 A/D 转换器模块相关的应用笔记有：

标题	应用笔记编号
Using the Analog-to-Digital (A/D) Converter	AN546
Four Channel Digital Voltmeter with Display and Keyboard	AN557
Understanding A/D Converter Performance Specifications	AN693

注：如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

18.27 版本历史

版本 A

这是本文档的首次发行版。

版本 B

此版本反映了 dsPIC30F 12 位 A/D 转换器模块的编辑排版及技术内容修订。

版本 C

此版本加入了本文档更新时已知的所有勘误表。

版本 D

此版本增加了扩展转换速率的指南。

注:

第 19 章 UART

目录

本章包括下列主题：

19.1 简介	19-2
19.2 控制寄存器	19-3
19.3 UART 波特率发生器 (BRG)	19-8
19.4 UART 配置	19-10
19.5 UART 发送器	19-11
19.6 UART 接收器	19-14
19.7 使用 UART 进行 9 位通信	19-18
19.8 接收中止字符	19-19
19.9 初始化	19-20
19.10 UART 的其他特性	19-21
19.11 UART 在 CPU 休眠和空闲模式下的工作	19-21
19.12 与 UART 模块相关的寄存器	19-22
19.13 设计技巧	19-23
19.14 相关应用笔记	19-24
19.15 版本历史	19-25

19.1 简介

通用异步收发器（Universal Asynchronous Receiver Transmitter, UART）模块是 dsPIC30F 系列器件提供的串行 I/O 模块之一。UART 是可以和外设（例如，个人电脑、RS-232 和 RS-485 接口）通信的全双工异步系统。

UART 模块的主要特性有：

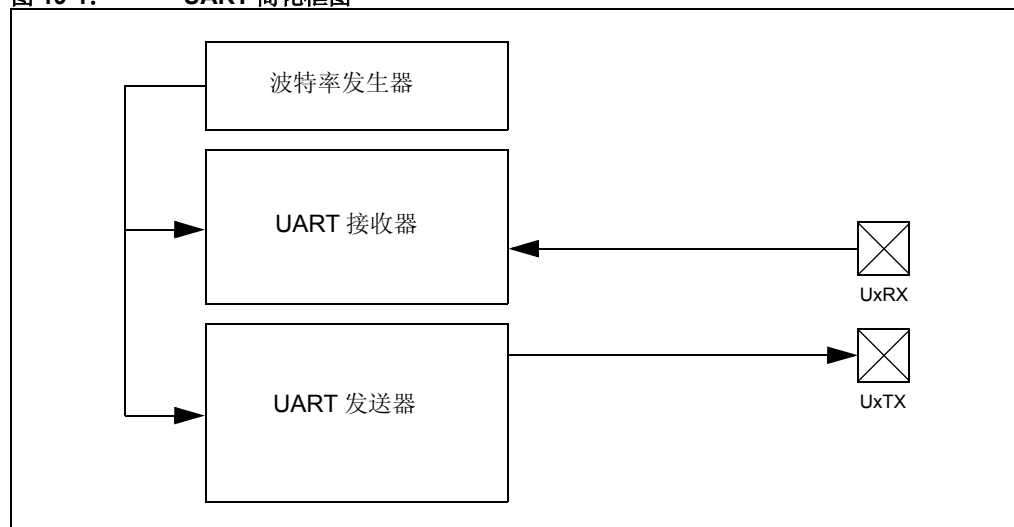
- 通过 UxTX 和 UxRX 引脚进行全双工 8 位或 9 位数据传输
- 偶、奇或无奇偶校验选项（对于 8 位数据）
- 一或两个停止位
- 完全集成的具有 16 位预分频器的波特率发生器
- 当 Fcy 为 30 MHz 时，波特率范围为 29 bps 到 1.875 Mbps
- 4 级深度先进先出（First-In-First-Out, FIFO）发送数据缓冲器
- 4 级深度 FIFO 接收数据缓冲器
- 奇偶、帧和缓冲溢出错误检测
- 支持带地址检测的 9 位模式（第 9 位 = 1）
- 发送和接收中断
- 用于诊断支持的环回模式

注： 不同的 dsPIC30F 器件可以有一个或多个 UART 模块。在引脚、控制 / 状态位和寄存器的名称中，使用“x”表示特定的模块。更多细节请参见特定的器件数据手册。

图 19-1 显示了 UART 的简化框图。UART 模块由以下至关重要的硬件元件组成：

- 波特率发生器
- 异步发送器
- 异步接收器

图 19-1: UART 简化框图



19.2 控制寄存器

寄存器 19-1: UxMODE: UARTx 模式寄存器

高字节:							
R/W-0	U-0	R/W-0	U-0	U-0	R/W-0	U-0	U-0
UARTEN	—	USIDL	—	保留	ALTIO	保留	保留
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
WAKE	LPBACK	ABAUD	—	—	PDSEL<1:0>	STSEL	
bit 7				bit 0			

- bit 15 **UARTEN:** UART 使能位
1 = UART 使能。UEN<1:0> 和 UTXEN 控制位定义了 UART 如何控制 UART 引脚。
0 = UART 禁止。UART 引脚由相应的 PORT、LAT 和 TRIS 位控制。
- bit 14 **未用:** 读作 0
- bit 13 **USIDL:** 在空闲模式停止位
1 = 当器件进入空闲模式时，停止运行
0 = 在空闲模式继续运行
- bit 12 **未用:** 读作 0
- bit 11 **保留:** 在该位置写 0
- bit 10 **ALTIO:** UART 备用 I/O 选择位
1 = UART 通过 UxATX 和 UxARX I/O 引脚通信
0 = UART 通过 UxTX 和 UxRX I/O 引脚通信

注： 不是所有的器件都具有备用 UART I/O 引脚。具体细节参见器件数据手册。
- bit 9-8 **保留:** 在这两位写 0
- bit 7 **WAKE:** 在休眠模式期间检测到启动位唤醒使能位
1 = 使能唤醒
0 = 禁止唤醒
- bit 6 **LPBACK:** UART 环回 (Loopback) 模式选择位
1 = 使能环回模式
0 = 禁止环回模式。
- bit 5 **ABAUD:** 自动波特率使能位
1 = 从 UxRX 引脚输入到捕捉模块
0 = 从 ICx 引脚输入到捕捉模块
- bit 4-3 **未用:** 读作 0
- bit 2-1 **PDSEL<1:0>:** 奇偶校验和数据选择位
11 = 9 位数据，无奇偶校验
10 = 8 位数据，奇校验
01 = 8 位数据，偶校验
00 = 8 位数据，无奇偶校验
- bit 0 **STSEL:** 停止选择位
1 = 2 个停止位
0 = 1 个停止位

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 19-2: UxSTA: UARTx 状态和控制寄存器

高字节:							
R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R-0	R-1
UTXISEL	—	—	—	UTXBRK	UTXEN	UTXBF	TRMT
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R-1	R-0	R-0	R/C-0	R-0
URXISEL<1:0>		ADDEN	RIDLE	PERR	FERR	OERR	URXDA
bit 7				bit 0			

- bit 15 UTXISEL:** 发送中断模式选择位
 1 = 当一个字符被传输到发送移位寄存器，导致缓冲器空的时候，产生中断
 0 = 当一个字符被传输到发送移位寄存器（发送缓冲器中至少还有一个字符）时，产生中断
- bit 14-12 未用:** 读作 0
- bit 11 UTXBRK:** 发送中止位
 1 = 不管发送器处在什么状态，拉低 UxTX 引脚
 0 = UxTX 引脚正常工作
- bit 10 UTXEN:** 发送使能位
 1 = UART 发送器使能，UART 控制 UxTX 引脚（如果 UARTEN = 1）
 0 = UART 发送器禁止，中止所有等待的发送，缓冲器复位。PORT 控制 UxTX 引脚。
- bit 9 UTXBF:** 发送缓冲器满状态位（只读）
 1 = 发送缓冲器满
 0 = 发送缓冲器未满，至少一个或多个数据字可以写入缓冲器。
- bit 8 TRMT:** 发送移位寄存器空位（只读）
 1 = 发送移位寄存器为空，同时发送缓冲器为空（上一个发送已经完成）
 0 = 发送移位寄存器非空，发送在进行中或在发送缓冲器中排队
- bit 7-6 URXISEL<1:0>:** 接收中断模式选择位
 11 = 接收缓冲器满时（即，有 4 个数据字符），中断标志位置位
 10 = 接收缓冲器 3/4 满时（即，有 3 个数据字符），中断标志位置位
 0x = 当接收到一个字符时，中断标志位置位
- bit 5 ADDEN:** 地址字符检测（接收数据的第 8 位 = 1）
 1 = 地址检测模式使能。如果没有选择 9 位模式，这个控制位将无效。
 0 = 地址检测模式禁止
- bit 4 RIDLE:** 接收器空闲位（只读）
 1 = 接收器空闲
 0 = 正在接收数据
- bit 3 PERR:** 奇偶校验错误状态位（只读）
 1 = 检测到当前字符的奇偶校验错误
 0 = 没有检测到奇偶校验错误
- bit 2 FERR:** 帧错误状态位（只读）
 1 = 检测到当前字符的帧错误
 0 = 没有检测到帧错误

寄存器 19-2: UxSTA: UARTx 状态和控制寄存器 (续)

- bit 1 **OERR:** 接收缓冲器溢出错误状态位 (只读 / 清零)
 1 = 接收缓冲器溢出
 0 = 接收缓冲器没有溢出
- bit 0 **URXDA:** 接收缓冲器中是否有数据位 (只读)
 1 = 接收缓冲器中有数据、有至少一个或多个字符可读
 0 = 接收缓冲器为空

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	C = 可清零位
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 19-3: UxRXREG: UARTx 接收寄存器

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	R-0
—	—	—	—	—	—	—	URX8
bit 15							bit 8

低字节:							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
URX<7:0>							
bit 7							bit 0

bit 15-9 未用: 读作 0

bit 8 **URX8**: 接收到的字符的第 8 位数据 (9 位模式下)

bit 7-0 **URX<7:0>**: 接收到的字符的 7-0 位数据

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

寄存器 19-4: UxTXREG: UARTx 发送寄存器 (只写)

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	W-x
—	—	—	—	—	—	—	UTX8
bit 15							bit 8

低字节:							
W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
UTX<7:0>							
bit 7							bit 0

bit 15-9 未用: 读作 0

bit 8 **UTX8**: 将要发送的字符的第 8 位数据 (9 位模式下)

bit 7-0 **UTX<7:0>**: 将要发送的字符的 7-0 位数据

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 的值

1 = 置位

0 = 清零

x = 未知

寄存器 19-5: UxBRG: UARTx 波特率寄存器

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BRG<15:8>							
bit 15						bit 8	

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BRG<7:0>							
bit 7						bit 0	

bit 15-0 **BRG<15:0>**: 波特率除数位

图注:			
R = 可读位	W = 可写位	U= 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

19.3 UART 波特率发生器（BRG）

UART 模块包含一个专用的 16 位波特率发生器（Baud Rate Generator, BRG）。UxBRG 寄存器控制一个自由运行的 16 位定时器的周期。公式 19-1 给出了计算波特率的公式。

公式 19-1: UART 波特率

$$\text{波特率} = \frac{F_{CY}}{16 \cdot (UxBRG + 1)}$$
$$UxBRG = \frac{F_{CY}}{16 \cdot \text{波特率}} - 1$$

注： F_{CY} 表示指令周期时钟频率。

例 19-1 给出了如下条件下的波特率误差计算：

- F_{CY} = 4 MHz
- 目标波特率 = 9600

例 19-1: 波特率误差计算

目标波特率 = F_{CY}/(16 (UxBRG + 1))

UxBRG 值的计算方法:

 UxBRG = ((F_{CY}/ 目标波特率)/16)-1

 UxBRG = ((4000000/9600)/16)-1

 UxBRG = [25.042] = 25

计算出来的波特率 = 4000000/(16 (25 + 1))

 = 9615

误差 = $\frac{(\text{计算波特率} - \text{目标波特率})}{\text{目标波特率}}$

 = (9615-9600)/9600

 = 0.16%

最大的波特率是 F_{CY} / 16 （当 UxBRG = 0 时），最小的波特率是 F_{CY} /(16 * 65536)。

向 UxBRG 寄存器中写新值会导致 BRG 定时器复位（清零）。这保证了 BRG 在产生新的波特率之前不需要等待定时器溢出。

19.3.1 波特率表

表 19-1 给出了普通器件指令周期频率 (Fcy) 的 UART 波特率。同时也给出了每个频率下的最小和最大波特率。

表 19-1: UART 波特率

目标 波特率 (kbps)	Fcy = 30 MHz			25 MHz			20 MHz			16 MHz		
	计算 波特率	% 误差	BRG 值 (十进制)	计算 波特率	% 误差	BRG 值 (十进制)	计算 波特率	% 误差	BRG 值 (十进制)	计算 波特率	% 误差	BRG 值 (十进制)
0.3	0.3	0.0	6249	0.3	+0.01	5207	0.3	0.0	4166	0.3	+0.01	3332
1.2	1.1996	0.0	1562	1.2001	+0.01	1301	1.1996	0.0	1041	1.2005	+0.04	832
2.4	2.4008	0.0	780	2.4002	+0.01	650	2.3992	0.0	520	2.3981	-0.08	416
9.6	9.6154	+0.2	194	9.5859	-0.15	162	9.6154	+0.2	129	9.6154	+0.16	103
19.2	19.1327	-0.4	97	19.2901	0.47	80	19.2308	+0.2	64	19.2308	+0.16	51
38.4	38.2653	-0.4	48	38.1098	-0.76	40	37.8788	-1.4	32	38.4615	+0.16	25
56	56.8182	+1.5	32	55.8036	-0.35	27	56.8182	+1.5	21	55.5556	-0.79	17
115	117.1875	+1.9	15	111.6071	-2.95	13	113.6364	-1.2	10	111.1111	-3.38	8
250							250	0.0	4	250	0.0	3
500										500	0.0	1
最小值	0.0286	0.0	65535	0.0238	0.0	65535	0.019	0.0	65535	0.015	0.0	65535
最大值	1875	0.0	0	1562.5	0.0	0	1250	0.0	0	1000	0.0	0

目标 波特率 (kbps)	Fcy = 12 MHz			10 MHz			8 MHz			7.68 MHz		
	计算 波特率	% 误差	BRG 值 (十进制)	计算 波特率	% 误差	BRG 值 (十进制)	计算 波特率	% 误差	BRG 值 (十进制)	计算 波特率	% 误差	BRG 值 (十进制)
0.3	0.3	0.0	2499	0.3	0.0	2082	0.2999	-0.02	1666	0.3	0.0	1599
1.2	1.2	0.0	624	1.1996	0.0	520	1.199	-0.08	416	1.2	0.0	399
2.4	2.3962	-0.2	312	2.4038	+0.2	259	2.4038	+0.16	207	2.4	0.0	199
9.6	9.6154	-0.2	77	9.6154	+0.2	64	9.6154	+0.16	51	9.6	0.0	49
19.2	19.2308	+0.2	38	18.9394	-1.4	32	19.2308	+0.16	25	19.2	0.0	24
38.4	37.5	+0.2	19	39.0625	+1.7	15	38.4615	+0.16	12			
56	57.6923	-2.3	12	56.8182	+1.5	10	55.5556	-0.79	8			
115			6									
250	250	0.0	2				250	0.0	1			
500							500	0.0	0			
最小值	0.011	0.0	65535	0.010	0.0	65535	0.008	0.0	65535	0.007	0.0	65535
最大值	750	0.0	0	625	0.0	0	500	0.0	0	480	0.0	0

目标 波特率 (kbps)	Fcy = 5 MHz			4 MHz			3.072 MHz			1.8432 MHz		
	计算 波特率	% 误差	BRG 值 (十进制)	计算 波特率	% 误差	BRG 值 (十进制)	计算 波特率	% 误差	BRG 值 (十进制)	计算 波特率	% 误差	BRG 值 (十进制)
0.3	0.2999	0.0	1041	0.3001	0.0	832	0.3	0.0	639	0.3	0.0	383
1.2	1.2019	+0.2	259	1.2019	+0.2	207	1.2	0.0	159	1.2	0.0	95
2.4	2.4038	+0.2	129	2.4038	+0.2	103	2.4	0.0	79	2.4	0.0	47
9.6	9.4697	-1.4	32	9.6154	+0.2	25	9.6	0.0	19	9.6	0.0	11
19.2	19.5313	+1.7	15	19.2308	+0.2	12	19.2	0.0	9	19.2	0.0	5
38.4	39.0625	+1.7	7				38.4	0.0	4	38.4	0.0	2
56												
115												
250												
500												
最小值	0.005	0.0	65535	0.004	0.0	65535	0.003	0.0	65535	0.002	0.0	65535
最大值	312.5	0.0	0	250	0.0	0	192	0.0	0	115.2	0.0	0

19.4 UART 配置

UART 使用标准非归零（Non-Return-to-Zero, NRZ）格式（一个启动位，8 或 9 个数据位，一或两个停止位）。硬件提供奇偶校验，用户可以配置成偶校验，奇校验或者不使用奇偶校验。最普通的数据格式是 8 位，没有奇偶校验位，有一个停止位（用 8、N 或 1 表示），这是缺省的（POR）设置。数据位和停止位的数目，还有奇偶校验在 PDSEL<1:0>（UxMODE<2:1>）和 STSEL（UxMODE<0>）位中指定。片上专用的 16 位波特率发生器可用于根据振荡器产生标准的波特率频率。UART 先发送和接收 LSb。UART 的发送器和接收器从功能上讲是独立的，但使用相同的数据格式和波特率。

19.4.1 使能 UART

UART 模块通过置位 UARTEN（UxMODE<15>）位和 UTXEN（UxSTA<10>）位使能。一旦使能了，UxTX 和 UxRX 引脚被分别配置为输出和输入，改写了相应 I/O 端口引脚的 TRIS 和 PORT 寄存器位的设置。UxTX 引脚在没有传输发生时，状态为逻辑 1。

注： 在 UARTEN 位置位之前，不应该置位 UTXEN 位。否则将无法使能 UART 发送。

19.4.2 禁止 UART

通过清零 UARTEN（UxMODE<15>）位禁止 UART 模块。这是任何复位后的默认状态。如果禁止了 UART，所有的 UART 引脚在相应的 PORT 和 TRIS 位控制下用作端口引脚。

禁止 UART 模块将缓冲器复位为空状态。所有缓冲器中的字符丢失，同时波特率计数器也复位。

当 UART 模块禁止时，所有与之相关的错误和状态标志都复位。URXDA、OERR、FERR、PERR、UTXEN、UTXBRK 和 UTXBF 位被清零而 RIDLE 和 TRMT 被置位。其他的控制位，包括 ADDEN、URXISEL<1:0> 和 UTXISEL，还有 UxMODE 和 UxBRG 寄存器不受影响。

当 UART 处于活动状态时，对 UARTEN 位清零将中止所有有等待的发送和接收，同时如以上定义那样将该模块复位。再次使能 UART 将使用同样的配置重新启动 UART。

19.4.3 备用 UART I/O 引脚

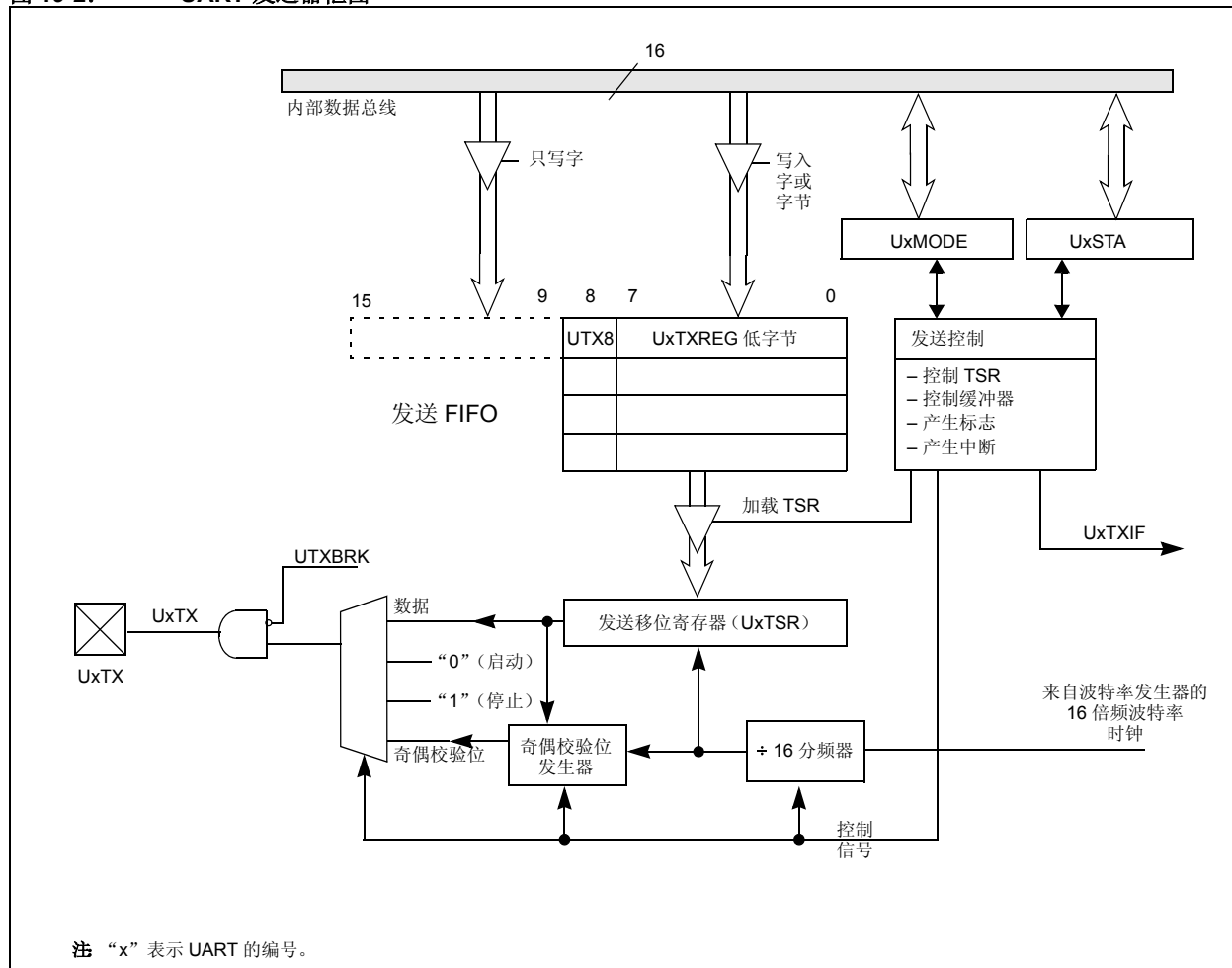
一些 dsPIC30F 器件有可用来通信的备用 UART 发送和接收引脚。当主 UART 引脚和其他外设共用时，可以使用备用 UART 引脚。备用 I/O 引脚通过置位 ALTIO 位（UxMODE<10>）来使能。如果 ALTIO = 1，UART 模块使用 UxTX 和 UxRX 引脚（分别为备用发送引脚和备用接收引脚）代替 UxTX 和 UxRX 引脚。如果 ALTIO = 0，UART 模块使用 UxTX 和 UxRX 引脚。

19.5 UART 发送器

图 19-2 显示了 UART 发送器的框图。发送器的核心是发送移位寄存器 (UxTSR)。移位寄存器从发送 FIFO 缓冲器 UxTXREG 中获得数据。UxTXREG 寄存器使用软件加载数据。直到前一次加载的停止位发送完成后才开始加载 UxTSR 寄存器。停止位一发送, UxTSR 就从 UxTXREG 寄存器加载新的数据 (如果有数据的话)。

注: UxTSR 寄存器没有映射到数据存储空间, 所以用户不能使用它。

图 19-2: UART 发送器框图



通过置位 UTXEN 使能位 (UxSTA<10>) 使能发送。实际的发送要到 UxTXREG 寄存器加载了数据并且波特率发生器 (UxBRG) 产生了移位时钟 (图 19-2) 后才发生。还可以先加载 UxTXREG 寄存器然后置位 UTXEN 使能位来启动发送。一般来说, 第一次开始发送的时候, 由于 UxTSR 寄存器为空, 这样传输数据到 UxTXREG 会导致该数据立即传输到 UxTSR。发送过程中对 UTXEN 位清零, 会导致发送中止并复位发送器。因此, UxTX 引脚将回复到一个高阻抗状态。

为了选择 9 位传输, PDSEL<1:0> 位 (UxMODE<2:1>) 应该设置为 11 并且第 9 位应该写到 UTX9 位 (UxTXREG<8>)。应该向 UxTXREG 执行一个字写操作, 这样可以同时写入所有的 9 位。

注: 在 9 位数据发送的情况下, 不采取奇偶校验。

19.5.1 发送缓冲器 (UxTXB)

每个 UART 有一个 4 级深、9 位宽的 FIFO 发送数据缓冲器。UxTXREG 寄存器提供对下一个可用的缓冲单元的用户访问。用户最多可在缓冲器中写 4 个字。一旦 UxTXREG 的内容被传送到 UxTSR 寄存器，当前缓冲单元就可以写入新的数据，下一个缓冲单元将成为 UxTSR 寄存器的数据源。无论何时，只要缓冲器满了，UTXBF (UxSTA<9>) 状态位就会置位。如果用户试图写满缓冲器，则新数据将不会被 FIFO 接收。

FIFO 在任何器件复位时复位，但当器件进入省电模式或从省电模式唤醒时，FIFO 不受影响。

19.5.2 发送中断

发送中断标志 (UxTXIF) 在相应中断标志状态寄存器 (IFS) 中。UTXISEL 控制位 (UxSTA<15>) 决定 UART 何时产生一个发送中断。

1. 如果 UTXISEL = 0，当一个字从发送缓冲器传输到发送移位寄存器 (UxTSR) 时产生中断。这暗示了发送缓冲器中至少有一个空字。由于中断在每个字发送完成后产生，这个模式对于频繁处理中断是很有用的（也就是说，ISR 在下一个字发送前完成）。
2. 如果 UTXISEL = 1，当将一个字从发送缓冲器传输到发送移位寄存器 (UxTSR) 并且缓冲器为空时产生中断。由于只有在所有 4 个字都发送完后才产生中断，所以如果用户代码不能足够迅速地处理中断（也就是说，ISR 在发送下一个字前完成），这种“块发送”模式是很有用的。

当模块第一次使能时，UxTXIF 位将置位。

用户应该在 ISR 中将 UxTXIF 位清零。

可以在运行时在两个中断模式间切换。

注： 如果 UTXISEL = 0，当 UTXEN 位置位时，UxTXIF 标志位也置位，因为发送缓冲器尚未满（可以向 UxTXREG 寄存器移入待发送数据）。

UxTXIF 标志位指示 UxTXREG 寄存器的状态，而 TRMT 位 (UxSTA<8>) 表明 UxTSR 寄存器的状态。TRMT 状态位是一个只读位，当 UxTSR 寄存器空时置位。没有任何中断逻辑和这个位有关，所以用户必须查询该位来确定 UxTSR 寄存器是否为空。

19.5.3 设置 UART 发送

设置发送时应该遵循的步骤:

- 1. 初始化UxBRG寄存器来获得合适的波特率（第19.3节“UART 波特率发生器（BRG）”）。
- 2. 通过写 PDSEL<1:0>（UxMODE<2:1>）和 STSEL（UxMODE<0>）位来设置数据位数，停止位数和奇偶校验选择。
- 3. 如果需要发送中断，就要置位相应的中断使能控制寄存器（IEC）中的 UxTXIE 控制位。使用相应中断优先级控制寄存器（IPC）中的 UxTXIP<2:0> 控制位来指定发送中断的中断优先级。同样，写 UTXISEL（UxSTA<15>）位来选择发送中断模式。
- 4. 置位 UARTEN（UxMODE<15>）位来使能 UART 模块。
- 5. 置位 UTXEN（UxSTA<10>）位来使能发送，与此同时将置位 UxTXIF 位。在 UART 发送中断服务程序中，UxTXIF 位应该清零。UxTXIF 位的操作受 UTXISEL 控制位控制。
- 6. 向 UxTXREG 寄存器加载数据（开始发送）。如果选择了 9 位发送，则加载一个字。如果选择了 8 位发送，则加载一个字节。数据可以加载到缓冲器，直到 UxTXBF 状态位（UxSTA<9>）置位为止。

注：在 UARTEN 位置位之前不应该置位 UTXEN 位。否则将无法使能 UART 发送。

图 19-3: 发送（8 位或 9 位数据）

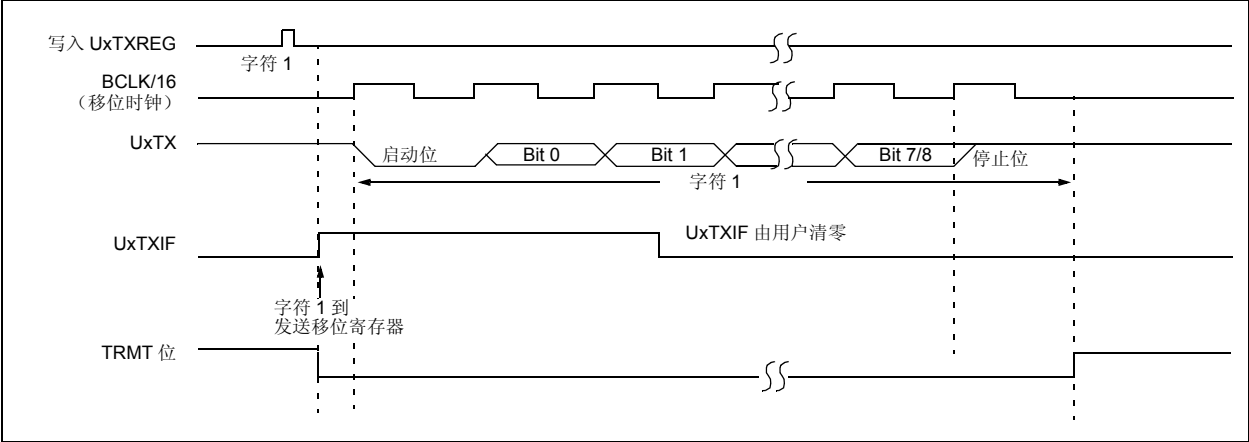
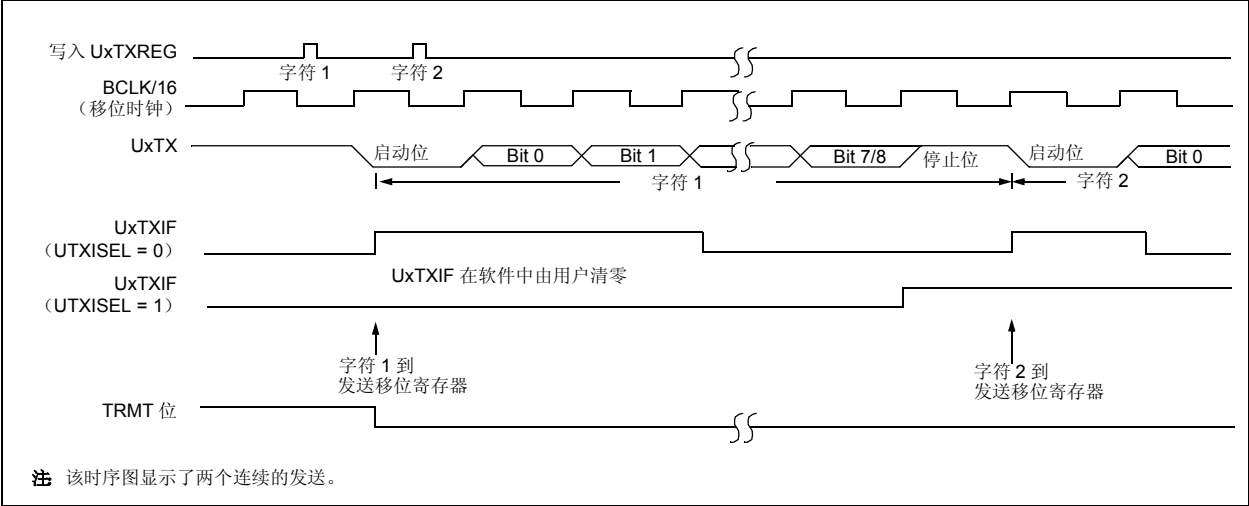


图 19-4: 发送（背靠背）



19.5.4 中止字符的发送

对 **UTXBRK** 位 (**UxSTA<11>**) 置位将强制 **UxTX** 线为 “0”。**UTXBRK** 忽略任何其他的发送器活动。用户应该在置位 **UTXBRK** 之前等待发送器空闲 (**TRMT = 1**)。

为了发送中止字符 (**Break Character**)，必须由软件置位 **UTXBRK** 位，而且保持该位置位至少 13 个波特率时钟。波特率时钟周期在软件中计时。然后通过软件将 **UTXBRK** 位清零，产生停止位。用户在再次加载 **UTXBUF** 或重新开始新的发送活动前，必须等待至少一或两个波特率时钟，来保证产生了有效的停止位。

注： 发送一个中止字符不会产生发送中断。

19.6 UART 接收器

图 19-5 显示了接收器的框图。接收器的核心是接收 (串行) 移位寄存器 (**UxRSR**)。数据在 **UxRX** 引脚上接收，并送到数据恢复区中。数据恢复区以 16 倍波特率运行，而主接收串行移位器以波特率运行。在采集到 **UxRX** 引脚上的停止位后，**UxRSR** 里面的接收到的数据传输到接收 FIFO 中 (如果为空)。

注： **UxRSR** 寄存器没有映射到数据存储空间，所以用户不可使用它。

多数检测电路三次采样 **UxRX** 引脚上的数据，确定该引脚上是高电平还是低电平。图 19-5 显示了采样机理图。

19.6.1 接收缓冲器 (**UxRXB**)

UART 接收器有一个 4 级深、9 位宽的 FIFO 接收数据缓冲器。**UxRXREG** 是一个存储器映射的寄存器可提供对 FIFO 输出的访问。在缓冲器溢出发生以前，可以有 4 个字的数据被接收并传输到 FIFO 中，从第 5 个字开始将数据移位到 **UxRSR** 寄存器中。

19.6.2 接收器错误处理

如果 FIFO 已满 (4 个字符)，而第五个字符已经完全接收到了 **UxRSR** 寄存器，溢出错误位 **OERR** (**UxSTA<1>**) 将会置位。**UxRSR** 中的字得以保留，但是只要 **OERR** 位置位，将禁止向接收 FIFO 传输后续数据。用户必须在软件中将 **OERR** 位清零，以允许更多的数据接收。

如果需要保存溢出前接收到的数据，用户应该先读所有 5 个字符，然后清零 **OERR** 位。如果这 5 个字符可以丢弃，用户只要清零 **OERR** 位即可。这可有效地复位接收 FIFO，同时先前接收到的所有数据都将丢失。

注： 接收 FIFO 中的数据应该在清零 **OERR** 位之前读出。当 **OERR** 清零时，FIFO 复位，这将导致缓冲器中所有的数据丢失。

如果停止位被检测为逻辑低电位，帧错误位 **FERR** (**UxSTA<2>**) 将置位。

如果检测到缓冲器顶部的数据字 (也就是说当前的字) 有奇偶校验错误，奇偶校验错误位 **PERR** (**UxSTA<3>**) 将置位。举例来说，如果奇偶校验设置为偶，但检测出数据中 1 的总数为奇，就产生了奇偶校验错误。**PERR** 位在 9 位模式中是无关的。**FERR** 和 **PERR** 位和相应的字一起被缓冲，并且应该在读取数据字之前读出。

19.6.3 接收中断

UART 接收中断标志 (UxRXIF) 位于相应中断标志状态寄存器 (IFS) 中。URXISEL<1:0> (UxSTA<7:6>) 控制位决定 UART 接收器何时产生中断。

- a) 如果 URXISEL<1:0> = 00 或 01, 每当一个数据字从接收移位寄存器 (UxRSR) 传输到接收缓冲器后就会产生中断。接收缓冲器中可以有一个或多个字符。
- b) 如果 URXISEL<1:0> = 10, 当一个字从移位寄存器 (UxRSR) 传输到了接收缓冲器使得缓冲器中有 3 或 4 个字符时产生中断。
- c) 如果 URXISEL<1:0> = 11, 当一个字从移位寄存器 (UxRSR) 传输到了接收缓冲器使得缓冲器中有 4 个字符 (即, 缓冲器满) 时产生中断。

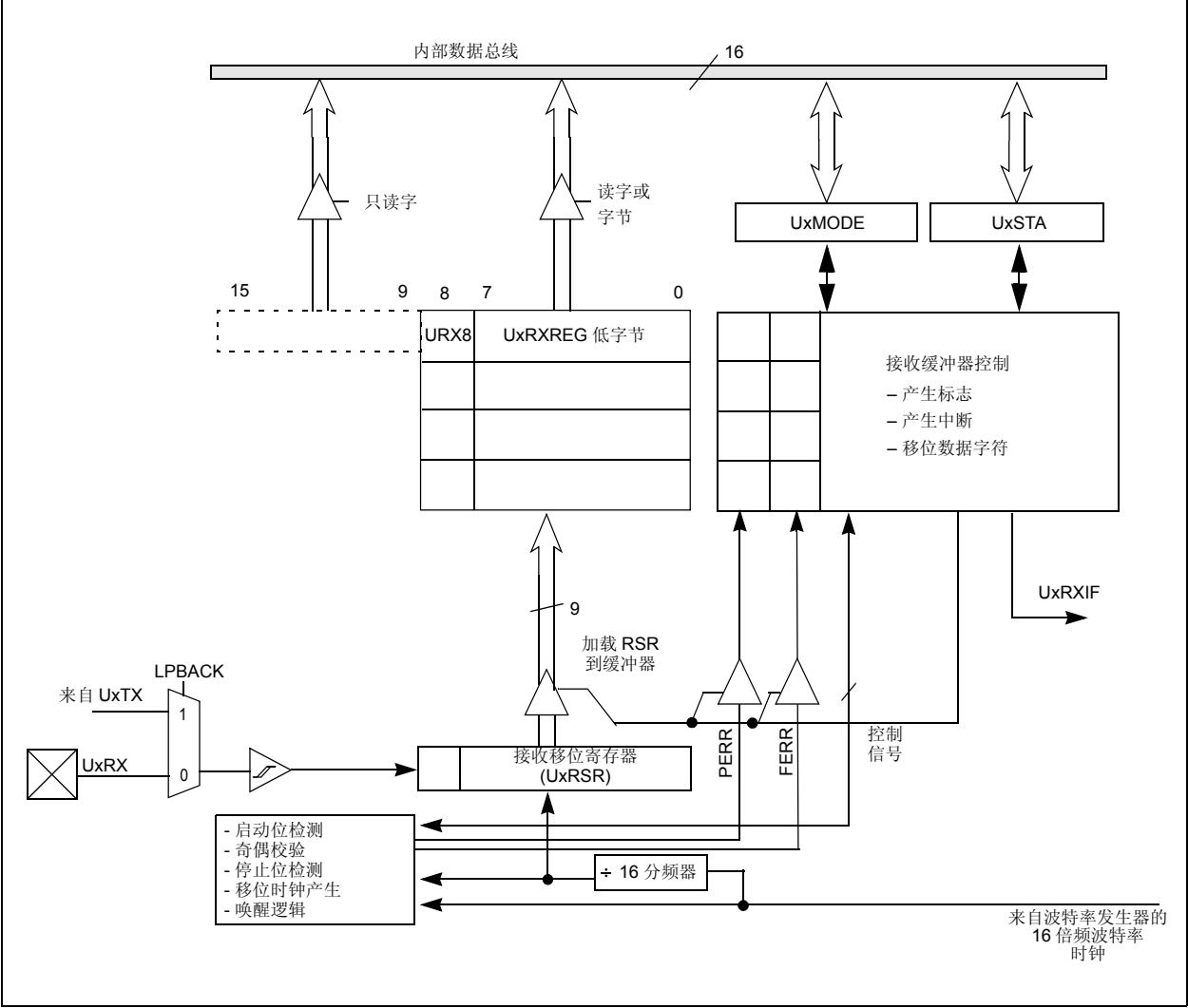
运行时可以在三个中断模式间切换。

URXDA 和 UxRXIF 标志位指示 UxRXREG 寄存器的状态, 而 RIDLE 位 (UxSTA<4>) 表明 UxRSR 寄存器的状态。RIDLE 状态位是一个只读位, 当接收器空闲 (即, UxRSR 寄存器空) 时置位。没有任何中断逻辑和这个位有关, 所以用户必须查询该位来确定 UxRSR 寄存器是否为空闲。

URXDA 位 (UxSTA<0>) 指示了接收缓冲器有数据还是为空。只要接收缓冲器中至少有一个可以读出的字符, 该位就将置位。URXDA 是只读位。

图 19-5 显示了 UART 接收器的框图。

图 19-5: UART 接收器框图



19.6.4 设置 UART 接收

设置接收时应该遵循的步骤：

- 1. 初始化UxBRG寄存器来获得合适的波特率（第19.3节“UART 波特率发生器（BRG）”）。
- 2. 通过写 PDSEL<1:0>（UxMODE<2:1>）和 STSEL（UxMODE<0>）位来设置数据位数，停止位数和奇偶校验选择。
- 3. 如果需要中断，就要置位相应的中断使能控制寄存器（IEC）中的 UxRXIE 位。使用相应中断优先级控制寄存器（IPC）中的 UxTXIP<2:0> 控制位来指定该中断的优先级。同时，通过写 URXISEL<1:0>（UxSTA<7:6>）位选择接收中断的模式。
- 4. 通过置位 UARTEN（UxMODE<15>）位，使能 UART 模块。
- 5. 接收中断取决于 URXISEL<1:0> 控制位的设置。如果没有允许接收中断，用户可以查询 URXDA 位。UxRXIF 位应该在 UART 接收中断服务程序中清零。
- 6. 从接收缓冲器中读取数据。如果选择 9 位传输，读一个字。否则，读一个字节。无论何时，只要缓冲中有数据可读，URXDA 状态位（UxSTA<0>）就将置位。

图 19-6: UART 接收

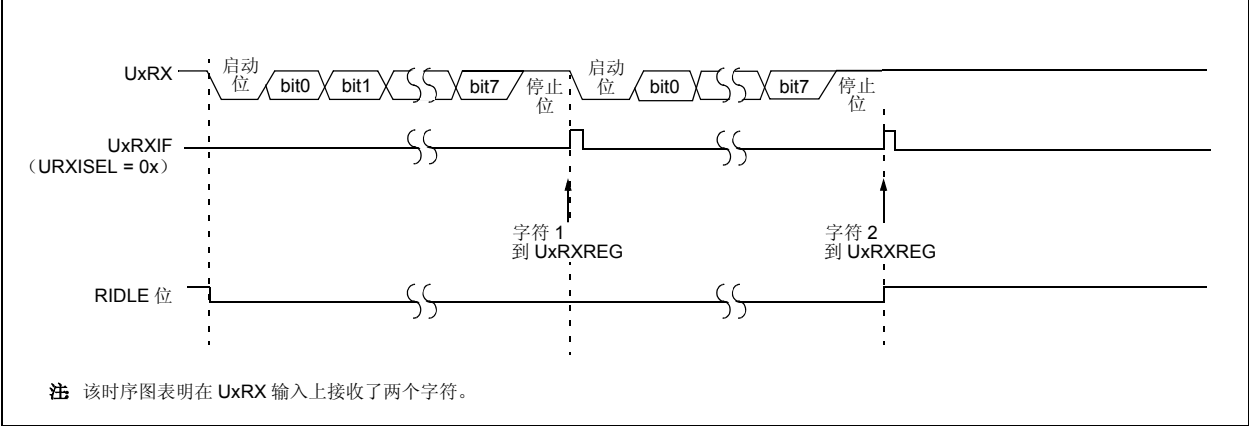
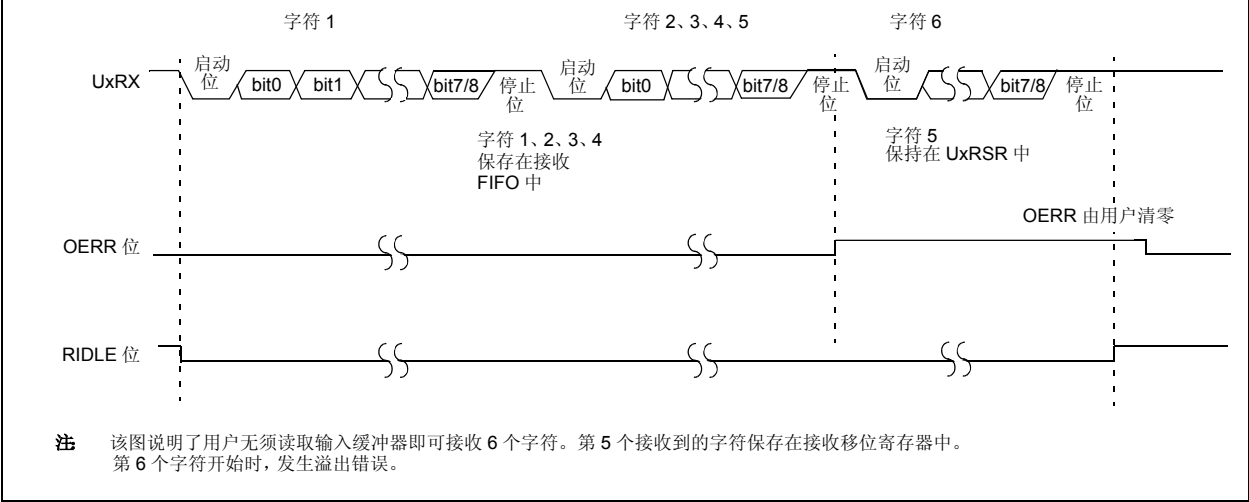


图 19-7: UART 在接收溢出下的接收



19.7 使用 UART 进行 9 位通信

典型的多处理器通信协议会区别数据字节和地址 / 控制字节。一般的方法是使用第 9 个数据位来识别数据字节是地址还是数据信息。如果第 9 位置位，数据就作为地址或控制信息处理。如果第 9 位清零，接收到的数据字就作为和前面的地址 / 控制字节相关的数据处理。

协议操作如下：

- 主器件发送一个第 9 位置位的数据字。数据字包含从器件的地址。
- 通信链中的所有从器件接收地址字并检查从地址值。
- 被寻址的从器件将接收和处理主器件发送的后续数据字节。所有其他的从器件将丢弃后面的数据字节，直到接收到新的地址字（第 9 位置位）。

19.7.1 ADDEN 控制位

UART 接收器有一个地址检测模式，该模式允许接收器忽略第 9 位清零的数据字。这降低了中断开销，因为第 9 位清零的数据字不被缓冲。这个功能通过置位 ADDEN 位（UxSTA<5>）来使能。

使用地址检测模式时，UART 必须配置为 9 位数据。当接收器配置为 8 位数据模式时，ADDEN 位无效。

19.7.2 设置 9 位发送

除了 PDSEL<1:0>（UxMODE<2:1>）应该设为 11 外，设置 9 位发送的过程几乎和设置 8 位发送模式一样（参见第 19.5.3 节“设置 UART 发送”）。

应当对 UxTXREG 寄存器执行写字操作（开始发送）。

19.7.3 设置使用地址检测模式的 9 位接收

除了 PDSEL<1:0> (UxMODE<2:1>) 应该设为 11 外, 设置 9 位接收的过程和设置 8 位接收模式类似 (参见第 19.6.4 节 “设置 UART 接收”)。

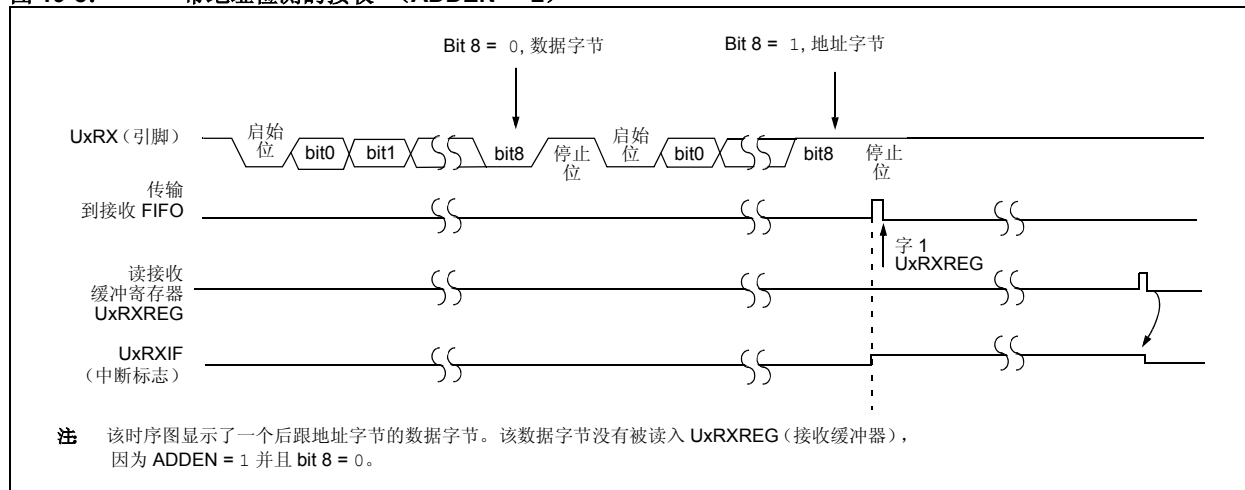
接收中断模式应该通过写 URXISEL<1:0> (UxSTA<7:6>) 位来配置。

注: 如果地址检测模式使能 (ADDEN = 1) 了, URXISEL<1:0> 控制位应该配置成接收到每个字后就产生中断。每个接收到的数据字在接收后必须立即在软件中进行检查, 看是否地址匹配。

使用地址检测模式的过程如下所述:

1. 置位 ADDEN (UxSTA<5>) 位来使能地址检测。必须保证 URXISEL 控制位配置成每接收一个字就产生一个中断。
2. 读 UxRXREG 寄存器, 检查每个 8 位地址, 确定器件是否被寻址。
3. 如果该器件没有被寻址, 就丢弃接收到的字。
4. 如果器件被寻址, 对 ADDEN 位清零, 允许后来的数据字节可以读进接收缓冲器, 并中断 CPU。如果希望长数据包, 则需要改变接收中断模式, 以使中断之间可以缓冲多于一个的数据字节。
5. 如果最后的数据字节接收到了, 置位 ADDEN 位使得只有地址字低功耗被接收。同样, 必须保证 URXISEL 控制位配置成每接收一个字就产生一个中断。

图 19-8: 带地址检测的接收 (ADDEN = 1)



19.8 接收中止字符

接收器会根据编程到 PDSEL (UxMODE<2:1>) 和 STSEL (UxMODE<0>) 位的值, 计数并等待一定的位时间数。

如果中止 (Break) 超过 13 个位时间, 在 PDSEL 和 STSEL 指定的位时间之后, 认为接收完成。URXDA位置位, FERR置位, 接收FIFO中装载零, 同时产生中断 (若允许中断) 且RIDLE位置位。

如果模块接收到了中止信号, 同时接收器检测到了启动位、数据位和无效停止位 (将 FERR 置 1), 接收器在找寻下一个启动位前必须等待有效停止位。接收器不能假定线上的中止条件就是下一个启动位。FERR 位置位且字符包含了全零被认为是中止。中止字符被加载到缓冲器中。只有接收到停止位后才可以接收更多的字符。注意当接收到停止位时 RIDLE 变高。

19.9 初始化

例 19-2 是 8 位模式的发送器 / 接收器的初始化程序。例 19-3 给出了 9 位地址检测模式下可寻址 UART 的初始化。在两个例子中，UxBRG 寄存器中加载的值取决于目标波特率和器件的频率。

注： 在 UARTEN 位置位之前不应该置位 UTXEN 位。否则将无法使能 UART 发送。

例 19-2: 8 位发送 / 接收 (UART1)

```
MOV    #baudrate,W0      ; Set Baudrate
MOV    W0,U1BRG

BSET   IPC2,#U1TXIP2     ; Set UART TX interrupt priority
BCLR   IPC2,#U1TXIP1     ;
BCLR   IPC2,#U1TXIP1     ;
BSET   IPC2,#U1TXIP2     ; Set UART TX interrupt priority
BCLR   IPC2,#U1TXIP1     ;
BCLR   IPC2,#U1TXIP1     ;

CLR    U1STA

MOV    #0x8800,W0        ; Enable UART for 8-bit data,
                        ; no parity, 1 STOP bit,
                        ; no wakeup

MOV    W0,U1MODE

BSET   U1STA,#UTXEN      ; Enable transmit

BSET   IEC0,#U1TXIE      ; Enable transmit interrupts
BSET   IEC0,#U1RXIE      ; Enable receive interrupts
```

例 19-3: 8 位发送 / 接收 (UART1)，地址检测使能

```
MOV    #baudrate,W0      ; Set Baudrate
MOV    W0,U1BRG

BSET   IPC2,#U1TXIP2     ; Set UART TX interrupt priority
BCLR   IPC2,#U1TXIP1     ;
BCLR   IPC2,#U1TXIP0     ;
BSET   IPC2,#U1RXIP2     ; Set UART RX interrupt priority
BCLR   IPC2,#U1RXIP1     ;
BCLR   IPC2,#U1RXIP0     ;

BSET   U1STA,#ADDEN      ; Enable address detect

MOV    #0x8883,W0        ; UART1 enabled for 9-bit data,
                        ; no parity, 1 STOP bit,
                        ; wakeup enabled

MOV    W0,U1MODE

BSET   U1STA,#UTXEN      ; Enable transmit

BSET   IEC0,#U1TXIE      ; Enable transmit interrupts
BSET   IEC0,#U1RXIE      ; Enable receive interrupts
```

19.10 UART 的其他特性

19.10.1 环回模式下的 UART

置位 LPBACK 位来使能这个特殊的模式，在该模式下 UxTX 输出在内部被连到 UxRX 输入。当配置为环回模式时，UxRX 引脚从内部 UART 接收逻辑断开。但是，UxTX 引脚仍然正常工作。要选择该模式：

- 1. 将 UART 配置为所需的工作模式。
- 2. 设置 LPBACK = 1 使能环回模式。
- 3. 像第 19.5 节 “UART 发送器” 定义的那样，使能发送。

环回模式取决于 UEN<1:0> 位的设置，如表 19-2 所示。

表 19-2: 环回模式引脚功能

UEN<1:0>	引脚功能， LPBACK = 1
00	UxRX 输入连接到 UxTX； UxTX 引脚工作； UxRX 引脚忽略； UxCTS/UxRTS 没有使用
01	UxRX 输入连接到 UxTX； UxTX 引脚工作； UxRX 引脚忽略； UxRTS 引脚工作， UxCTS 没有使用
10	UxRX 输入连接到 UxTX； UxTX 引脚工作； UxRX 引脚忽略； UxRTS 引脚工作， UxCTS 输入连接到 UxRTS； UxCTS 引脚忽略
11	UxRX 输入连接到 UxTX； UxTX 引脚工作； UxRX 引脚忽略； BCLK 引脚工作； UxCTS/UxRTS 没有使用

19.10.2 自动波特率支持

为了让系统确定接收字符的波特率，UxRX 输入应该在内部连到所选择的输入捕捉通道。当 ABAUD 位（UxMODE<5>）置位时，UxRX 引脚在内部被连接到输入捕捉通道。ICx 引脚从输入捕捉通道断开。

自动波特率支持所使用的输入捕捉通道是由具体器件决定的。更多的细节请参见器件数据手册。这种模式只有在 UART 使能（UARTEN = 1），同时禁止环回模式（LPBACK = 0）的情况下有效。而且，用户必须对捕捉模块进行设置，使其检测启动位的上升沿和下降沿。

19.11 UART 在 CPU 休眠和空闲模式下的工作

UART 在休眠模式下不工作。如果在发送进行期间进入休眠模式，则将中止发送并且 UxTX 引脚被驱动为逻辑 “1”。同样的，如果接收进行期间进入休眠模式，接收将中止。

UART 可以用来在检测到启动位时选择性地将 dsPIC 器件从休眠模式唤醒。如果 WAKE 位（UxSTA<7>）置位、器件处于休眠模式同时允许接收中断（UxRXIE = 1），则 UxRX 引脚上的下降沿将产生一个接收中断。接收中断选择模式位（URXISEL）对该功能没有影响。为了产生唤醒中断，UARTEN 位必须置位。

USIDL 位（UxMODE<13>）选择当器件进入空闲模式时模块是停止工作，还是在空闲模式下继续正常工作。如果 USIDL = 0，模块将在空闲模式期间继续正常工作。如果 USIDL = 1，模块将在空闲模式下停止工作。正在进行的任何发送或接收操作将中止。

19.12 与 UART 模块相关的寄存器

表 19-3: 与 UART1 相关的寄存器

SFR 名称	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位状态
U1MODE	UARTEN	—	USIDL	—	保留	ALTIO	保留	保留	WAKE	LPBACK	ABAUD	—	—	PDSEL<1:0>		STSEL	0000 0000 0000 0000
U1STA	UTXISEL	—	—	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL<1:0>		ADDEN	RIDLE	PERR	FERR	OERR	URXDA	0000 0001 0001 0000
U1TXREG	—	—	—	—	—	—	—	UTX8	发送寄存器								0000 0000 0000 0000
U1RXREG	—	—	—	—	—	—	—	URX8	接收寄存器								0000 0000 0000 0000
U1BRG	波特率发生器预分频器																0000 0000 0000 0000
IFS0	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0	0000 0000 0000 0000
IEC0	CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IPC2	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			0100 0100 0100 0100

注： 与 UART1 相关的寄存器如上表所示供用户参考。与其他 UART 模块相关的寄存器，请参见器件数据手册。

19.13 设计技巧

问 1: *我用 UART 发送的数据不能正确的接收。这是什么原因啊?*

答: 最普遍接收错误的原因是为 UART 波特率发生器计算了一个错误的值。确保写进 UxBRG 寄存器的值是正确的。

问 2: *尽管 UART 接收引脚上的信号看上去是正确的,但我还是得到了帧错误。可能是什么原因?*

答: 确保下列控制位已被正确设置:

- UxBRG: UART 波特率寄存器
- PDSEL<1:0>: 奇偶校验和数据大小选择位
- STSEL: 停止位选择

19.14 相关应用笔记

本节列出了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相关的，通过适当修改即可使用，但使用中可能会受到一定限制。目前和 UART 模块相关的应用笔记有：

标题

应用笔记编号

Serial Port Utilities Implementing Table Read and Table Write

AN547

注： 如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

19.15 版本历史

版本 A

这是本文档的初始发行版。

版本 B

版本 B 已经过扩充，包含了对 dsPIC30F UART 模块的全面描述。

版本 C

此版本加入了本文档更新时已知的所有勘误表。

注:

第 20 章 串行外设接口 (SPI™)

目录

本章包括下列主题：

20.1 简介	20-2
20.2 状态和控制寄存器	20-4
20.3 工作模式	20-7
20.4 SPI 主控模式时钟频率	20-19
20.5 低功耗模式下的工作	20-20
20.6 与 SPI 模块相关的特殊功能寄存器	20-22
20.7 相关应用笔记	20-23
20.8 版本历史	20-24

20.1 简介

串行外设接口（Serial Peripheral Interface，SPI）模块是一个同步串行接口，可用于与其他外设或者单片机进行通信。这些外设可以是串行 EEPROM、移位寄存器、显示驱动器、A/D 转换器等。SPI 模块与 Motorola 的 SPI 和 SIOP 接口兼容。

dsPIC30F 系列器件在单个器件上提供一个或两个 SPI 模块，具体取决于具体不同的器件。SPI1 和 SPI2 功能相同。很多高引脚数（64 引脚或更多）封装的器件具有 SPI2 模块，而 SPI1 模块则是所有的器件都具有的。

注： 在本章中，SPI 模块统称为 SPIx，或分别称为 SPI1 和 SPI2。特殊功能寄存器也使用类似的符号表示上述类似的意思。例如，SPIxCON 指 SPI1 或 SPI2 模块的控制寄存器。

SPI 串口包含下列特殊功能寄存器（SFR）：

- SPIxBUF：地址位于 SFR 空间，用于缓冲待发送数据和已接收数据。此地址由 SPIxTXB 和 SPIxRXB 寄存器共享。
- SPIxCON：配置模块各种操作模式的控制寄存器。
- SPIxSTAT：显示各种状态条件的状态寄存器。

此外，还有一个 16 位移位寄存器 SPIxSR，此寄存器不映射到存储器空间。该寄存器可用于将数据移入和移出 SPI 端口。

存储器映射的 SFR（SPIxBUF）是 SPI 数据接收 / 发送寄存器。在内部，SPIxBUF 寄存器实际上由两个独立的寄存器（SPIxTXB 和 SPIxRXB）组成。接收缓冲寄存器 SPIxRXB 和发送缓冲寄存器 SPIxTXB 是两个单向 16 位寄存器。这两个寄存器共享名为 SPIxBUF 的 SFR 地址单元。如果用户将需要发送的数据写入了 SPIxBUF 地址单元，该数据会在内部写入 SPIxTXB 寄存器。与此类似，当用户从 SPIxBUF 读取已接收到的数据时，该数据在内部是从 SPIxRXB 寄存器读取的。这种接收和发送操作的双缓冲可以使数据在后台连续传输。发送和接收可同时进行。

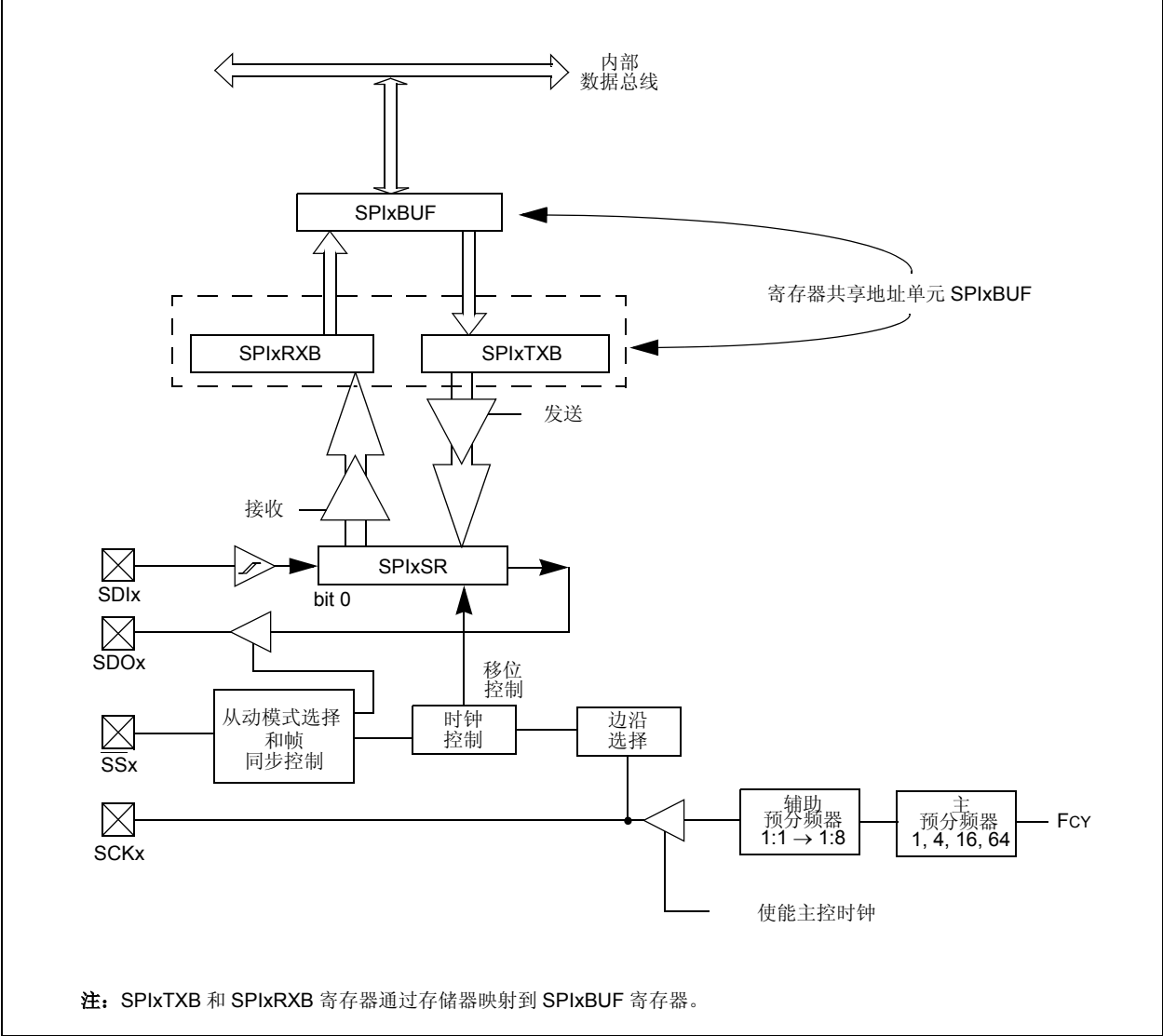
注： 用户无法直接写入 SPIxTXB 寄存器或读取 SPIxRXB 寄存器。所有的读写操作都是在 SPIxBUF 寄存器中进行的。

SPI 串行接口由以下四个引脚组成：

- SDIx：串行数据输入
- SDOx：串行数据输出
- SCKx：移位时钟输入或输出
- SSx：低电平有效从动选择或者帧同步 I/O 脉冲

注： SPI 模块可以配置为使用 3 个或 4 个引脚工作。在 3 引脚模式下，不使用 SSx 引脚。

图 20-1: SPI 模块框图



20.2 状态和控制寄存器

寄存器 20-2: **SPIxSTAT: SPI 状态和控制寄存器**

高字节:							
R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
SPIEN	—	SPISIDL	—	—	—	—	—
bit 15							bit 8

低字节:							
U-0	R/W-0 HS	U-0	U-0	U-0	U-0	R-0	R-0
—	SPIROV	—	—	—	—	SPITBF	SPIRBF
bit 7							bit 0

- bit 15 **SPIEN:** SPI 使能位
1 = 使能模块并将 SCKx、SDOx、SDIx 和 SSx 配置为串口引脚
0 = 禁止模块
- bit 14 **未用:** 读作 0
- bit 13 **SPISIDL:** 在空闲模式停止位
1 = 当器件进入空闲模式时模块不继续工作
0 = 在空闲模式下模块继续工作
- bit 12-7 **未用:** 读作 0
- bit 6 **SPIROV:** 接收溢出标志位
1 = 一个新字节 / 字已完全接收并丢弃。用户软件未读取 SPIxBUF 寄存器中原先的数据。
0 = 没有发生溢出
- bit 5-2 **未用:** 读作 0
- bit 1 **SPITBF:** SPI 发送缓冲器满状态位
1 = 未开始发送, SPIxTXB 满
0 = 发送开始, SPIxTXB 空
当 CPU 写 SPIxBUF 地址单元并装载 SPIxTXB 时, 该位由硬件自动置位。
当 SPIx 模块将数据从 SPIxTXB 传输到 SPIxSR 时, 该位由硬件自动清零。
- bit 0 **SPIRBF:** SPI 接收缓冲器满状态位
1 = 接收完成, SPIxRXB 满
0 = 接收未完成, SPIxRXB 空
当 SPIx 将数据从 SPIxSR 传输到 SPIxRXB 时, 该位由硬件自动置位。
当内核通过读 SPIxBUF 地址单元读 SPIxRXB 时, 该位由硬件自动清零。

图注:

R = 可读位	W = 可写位	U = 未用位, 读作 0
HC = 由硬件清零	HS = 由硬件置位	
-n = 复位时的值	1 = 置位	0 = 清零
		x = 未知

寄存器 20-2: SPIxCON: SPIx 控制寄存器

高字节:							
U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	FRMEN	SPIFSD	—	DISSDO	MODE16	SMP	CKE
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SSEN	CKP	MSTEN	SPRE<2:0>			PPRE<1:0>	
bit 7			bit 0				

- bit 15 未用：读作 0
- bit 14 **FRMEN**: 帧 SPI 支持位
1 = 使能帧 SPI 支持
0 = 禁止帧 SPI 支持
- bit 13 **SPIFSD**: SSx 引脚上的帧同步脉冲方向控制位
1 = 帧同步脉冲输入（从动模式）
0 = 帧同步脉冲输出（主控模式）
- bit 12 未用：读作 0
- bit 11 **DISSDO**: SDOx 引脚禁止位
1 = 模块不使用 SDOx 引脚。该引脚由相关端口寄存器控制。
0 = SDOx 引脚由模块控制
- bit 10 **MODE16**: 字 / 字节通信选择位
1 = 通信为字宽（16 位）
0 = 通信为字节宽（8 位）
- bit 9 **SMP**: SPI 数据输入采样相位位
主控模式:
1 = 输入数据在数据输出时间末尾采样
0 = 输入数据在数据输出时间中间采样
从动模式:
当 SPI 在从动模式下使用时，必须将 SMP 清零。
- bit 8 **CKE**: SPI 时钟沿选择位
1 = 串行输出数据在有效时钟状态转变为空闲时钟状态时变化（参见 bit 6）
0 = 串行输出数据在空闲时钟状态转变为有效时钟状态时变化（参见 bit 6）
注: 在帧 SPI 模式下未使用 CKE 位。在帧 SPI 模式下（FRMEN = 1），用户应将该位编程为 0。
- bit 7 **SSEN**: 从动选择使能（从动模式）位
1 = SS 引脚用于从动模式
0 = 模块不使用 SS 引脚。引脚由端口功能控制。
- bit 6 **CKP**: 时钟极性选择位
1 = 空闲状态时钟信号为高电平；有效状态为低电平
0 = 空闲状态时钟信号为低电平；有效状态为高电平
- bit 5 **MSTEN**: 主控模式使能位
1 = 主控模式
0 = 从动模式

寄存器 20-2: SPIxCON: SPIx 控制寄存器 (续)

- bit 4-2 **SPRE<2:0>**: 辅预分频比 (主控模式) 位
 (支持设置: 1:1、 2:1 到 8:1 全部支持)
 111 = 辅预分频比 1:1
 110 = 辅预分频比 2:1

 000 = 辅预分频比 8:1
- bit 1-0 **PPRE<1:0>**: 主预分频比 (主控模式) 位
 11 = 主预分频比 1:1
 10 = 主预分频比 4:1
 01 = 主预分频比 16:1
 00 = 主预分频比 64:1

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 值	1 = 置位	0 = 清零	x = 未知

20.3 工作模式

以下各节讨论了 SPI 模块灵活的工作模式：

- 8 位和 16 位数据发送 / 接收
- 主控模式和从动模式
- 帧 SPI 模式

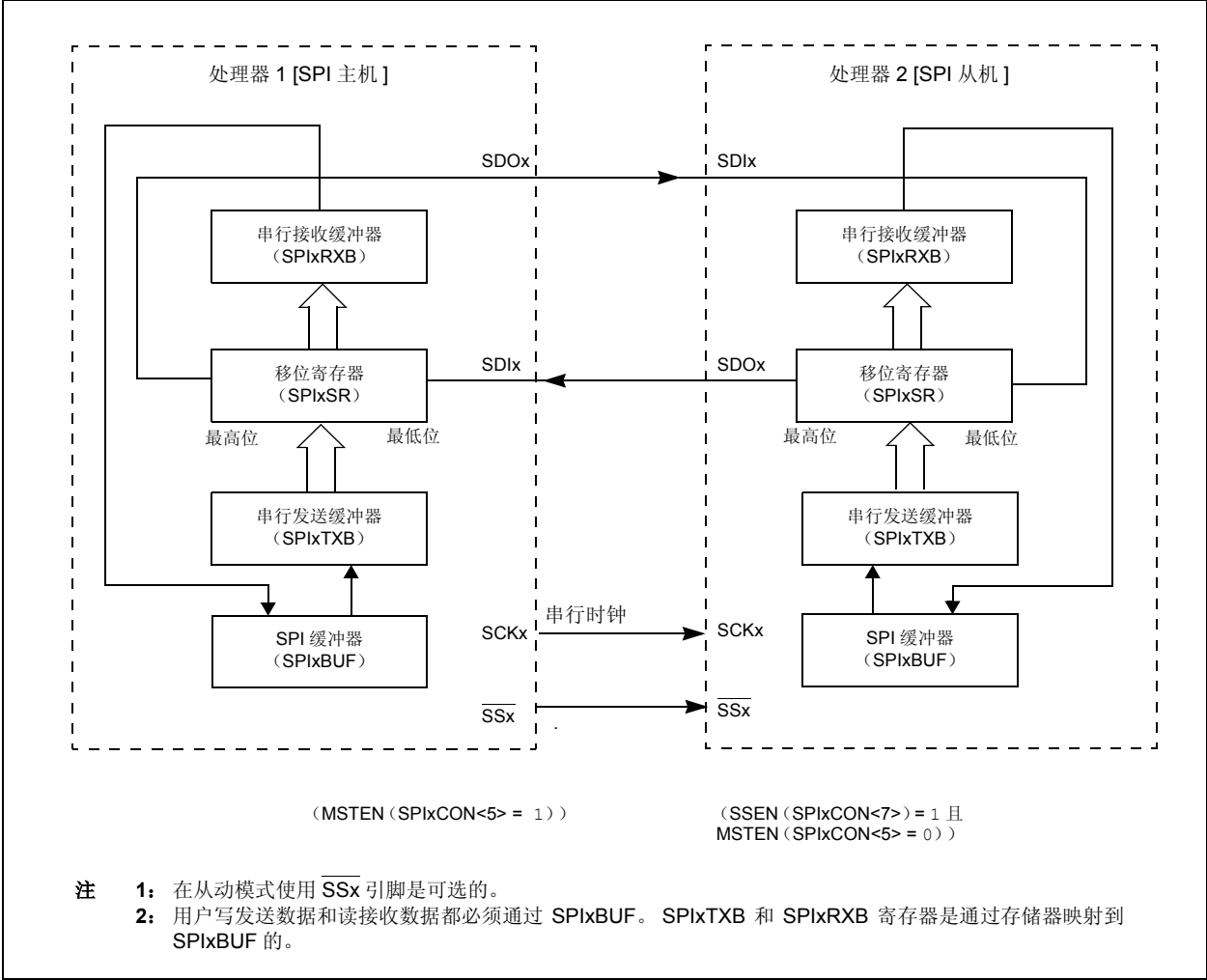
20.3.1 8 位与 16 位工作模式

控制位 MODE16 (SPIxCON<10>) 允许模块在 8 位或 16 位模式下通信。除了接收和发送的位数外，两种模式的功能是相同的。此外，在阅读本文时还应注意以下各项：

- 当 MODE16 (SPIxCON<10>) 位的值变化时，模块会复位。因此，在正常工作过程中不应该改变该位。
- 8 位工作模式下数据是从 SPIxSR 的 bit 7 发送的，而在 16 位工作模式下，则是从 SPIxSR 的 bit 15 发送的。在两种模式下，数据都会移入 SPIxSR 的 bit 0。
- 在 8 位模式下移入 / 移出数据需要 SCKx 引脚上出现 8 个时钟脉冲，而在 16 位模式下则需要 16 个时钟脉冲。

20.3.2 主控模式和从动模式

图 20-2: SPI 主 / 从连接



20.3.2.1 主控模式

遵循以下步骤将 SPI 模块设置为工作在主控模式：

1. 如果使用中断：
 - 清零相应 IFSn 寄存器中的 SPIxIF 位。
 - 置位相应 IECn 寄存器中的 SPIxIE 位。
 - 向相应的 IPCn 寄存器设置 SPIxIP 位。
2. 将所需设置写入 SPIxCON 寄存器，同时 MSTEN (SPIxCON<5>) = 1。
3. 将 SPIROV 位 (SPIxSTAT<6>) 清零。
4. 通过将 SPIEN 位 (SPIxSTAT<15>) 置位使能 SPI 工作。
5. 将待发送数据写入 SPIxBUF 寄存器。数据一写入 SPIxBUF 寄存器，发送（以及接收）就会立即开始。

在主控模式下，系统时钟被预分频，然后作为串行时钟使用。预分频基于 PPRE<1:0> (SPIxCON<1:0>) 和 SPRE<1:0> (SPIxCON<4:2>) 位的设置。串行时钟通过 SCKx 引脚输出到从动器件。仅当有待发送数据时才会产生时钟脉冲。如需了解更多信息，请参见第 20.4 节“SPI 主控模式时钟频率”。

CKP 和 CKE 位确定在哪个时钟沿发送数据。

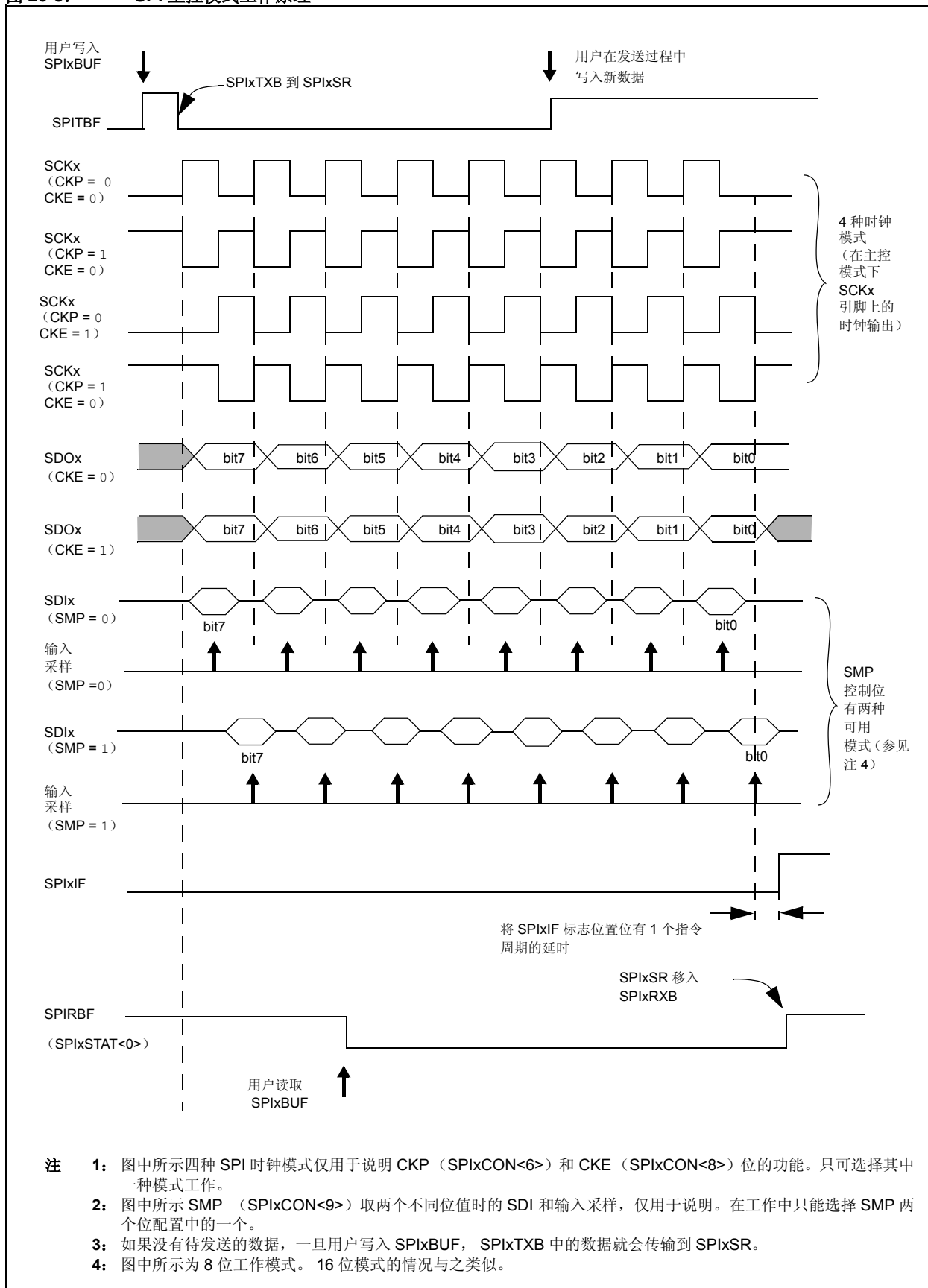
待发送数据和已接收数据都分别向 SPIxBUF 寄存器写入和从该寄存器读取。

以下内容描述了主控模式下 SPI 模块的工作原理。

1. 一旦模块被设置为主控工作模式并使能，待发送数据就会写入 SPIxBUF 寄存器。SPITBF (SPIxSTAT<1>) 位置位。
2. SPIxTXB 的内容移到移位寄存器 SPIxSR，并且模块将 SPITBF 位清零。
3. 一组 8/16 个时钟脉冲将 8/16 位发送数据从 SPIxSR 移出到 SDOx 引脚，同时将 SDIx 引脚的数据移入 SPIxSR。
4. 当传输结束后，会发生以下事件：
 - 中断标志位 SPIxIF 置位。通过将中断使能位 SPIxIE 置位可以允许 SPI 中断。SPIxIF 标志不会被硬件自动清零。
 - 另外，当正在进行的发送和接收操作结束后，SPIxSR 的内容会移到 SPIxRXB 寄存器。
 - SPIRBF (SPIxSTAT<0>) 位由模块置位，表明接收缓冲器满。一旦用户代码读 SPIxBUF 寄存器，硬件就会将 SPIRBF 位清零。
5. 当 SPI 模块需要从 SPIxSR 传输数据到 SPIxRXB 时，如果 SPIRBF 位置位（接收缓冲器满），模块会将 SPIROV (SPIxSTAT<6>) 位置位，表明产生了溢出条件。
6. 只要 SPITBF (SPIxSTAT<1>) 清零，用户软件就可以在任何时候将待发送数据写入 SPIxBUF。写入可以与 SPIxSR 移出前面写入的数据同时发生，因此可以允许连续发送。

注： 用户不能直接写 SPIxSR 寄存器。对 SPIxSR 寄存器的所有写入都是通过 SPIxBUF 寄存器进行的。

图 20-3: SPI 主控模式工作原理



20.3.2.2 从动模式

遵循以下步骤将 SPI 模块设置为从动工作模式：

1. 将 SPIxBUF 寄存器清零。
2. 如果使用中断：
 - 清零相应 IFSn 寄存器中的 SPIxIF 位。
 - 置位相应 IECn 寄存器中的 SPIxIE 位。
 - 向相应的 IPCn 寄存器设置 SPIxIP 位。
3. 将所需设置写入 SPIxCON 寄存器，同时使 MSTEN (SPIxCON<5>) = 0。
4. 将 SMP 位清零。
5. 如果 CKE 位置位，则 SSEN 位必须置位，从而使能 $\overline{\text{SSx}}$ 引脚。
6. 将 SPIROV 位 (SPIxSTAT<6>) 清零，并且
7. 通过将 SPIEN 位 (SPIxSTAT<15>) 置位使能 SPI 工作。

在从动模式下，在外部时钟脉冲出现在 SCKx 引脚时发送和接收数据。CKP (SPIxCON<6>) 和 CKE (SPIxCON<8>) 位决定数据发送发生在哪个时钟沿。

待发送数据和已接收数据分别向 SPIxBUF 寄存器写入和从该寄存器读取。

模块在该模式下的其余工作与在主控模式下相同。

从动模式还提供了一些其他功能，它们是：

从动模式选择同步： $\overline{\text{SSx}}$ 引脚允许同步从动模式。如果 SSEN (SPIxCON<7>) 位置位，只有 $\overline{\text{SSx}}$ 引脚驱动为低电平状态时才会使能从动模式下的发送和接收。为了使 $\overline{\text{SSx}}$ 引脚能作为输入引脚使用，不能驱动端口输出或其他外设输出。如果 SSEN 位置位且 $\overline{\text{SSx}}$ 引脚驱动为高电平，SDOx 引脚将不再被驱动并将呈现为三态，即使模块处于发送过程中也是如此。在下次 $\overline{\text{SSx}}$ 引脚驱动为低电平时，使用保存在 SPIxTXB 寄存器的数据重试上次中止的发送。如果 SSEN 位没有置位， $\overline{\text{SSx}}$ 引脚不会影响从动模式下的模块工作。

SPITBF 状态标志工作原理：SPITBF (SPIxSTAT<1>) 位的功能在从动工作模式下是与主控模式不同的。以下描述了从动工作模式下 SPITBF 的各种设置所对应的功能：

1. 如果 SSEN (SPIxCON<7>) 清零，SPITBF 将在用户代码装入 SPIxBUF 时置位。它将在模块将 SPIxTXB 中的数据传送到 SPIxSR 时清零。这与主控模式下 SPITBF 位的功能类似。
2. 如果 SSEN (SPIxCON<7>) 置位，SPITBF 将在用户代码载入 SPIxBUF 时置位。但是，它只有在 SPIx 模块完成数据发送后才会清零。当 $\overline{\text{SSx}}$ 引脚变为高电平时，发送将被中止并可能在一段时间以后重试。每个数据字都保存在 SPIxTXB 中，直到所有的位都被发送到接收器为止。

注：要符合模块的时序要求，当 CKE = 1 时，在从动模式下必须使能 $\overline{\text{SSx}}$ 引脚（详情请参见图 20-6）。

图 20-4: SPI 从动模式工作原理：禁止从动选择引脚

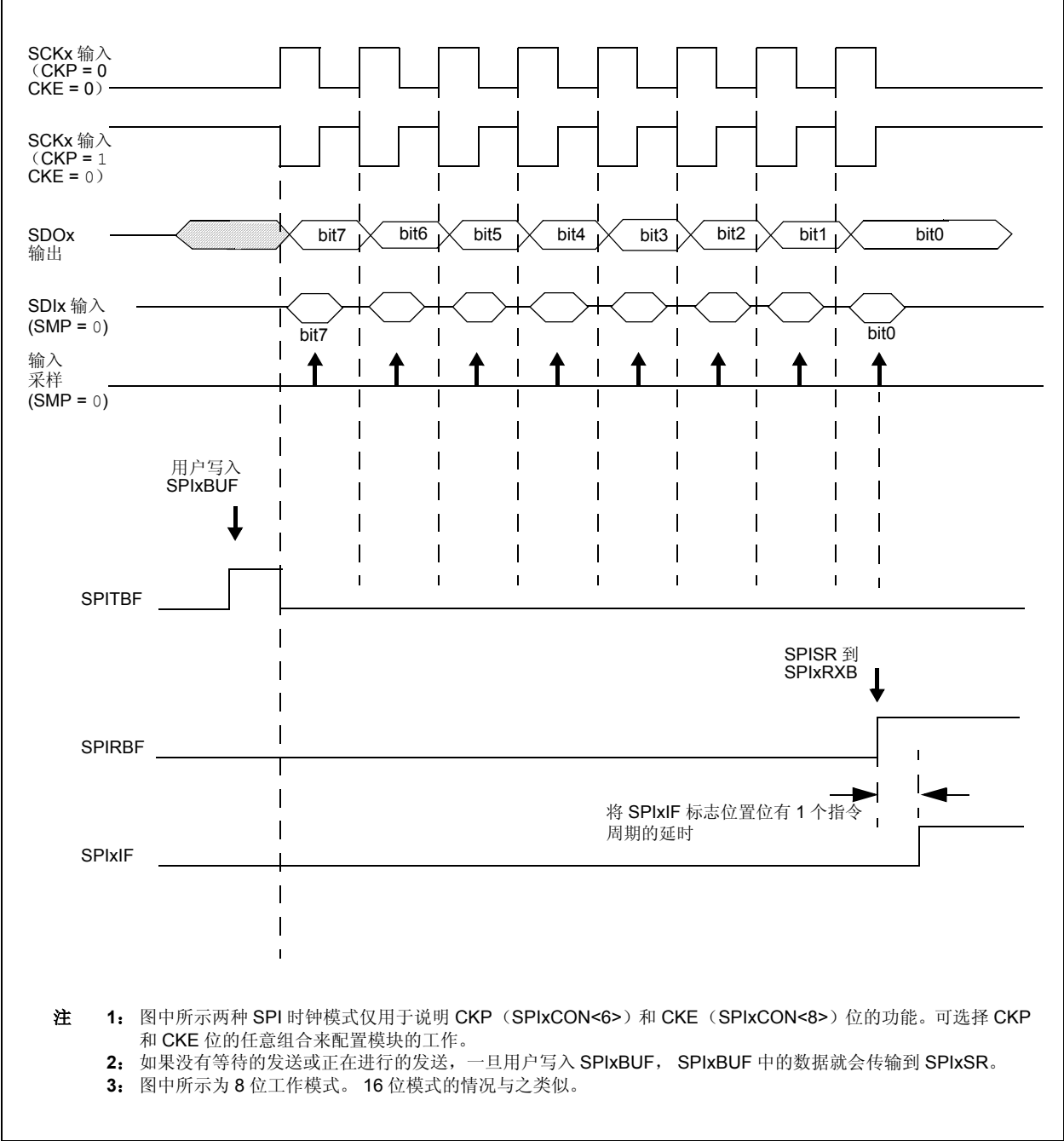
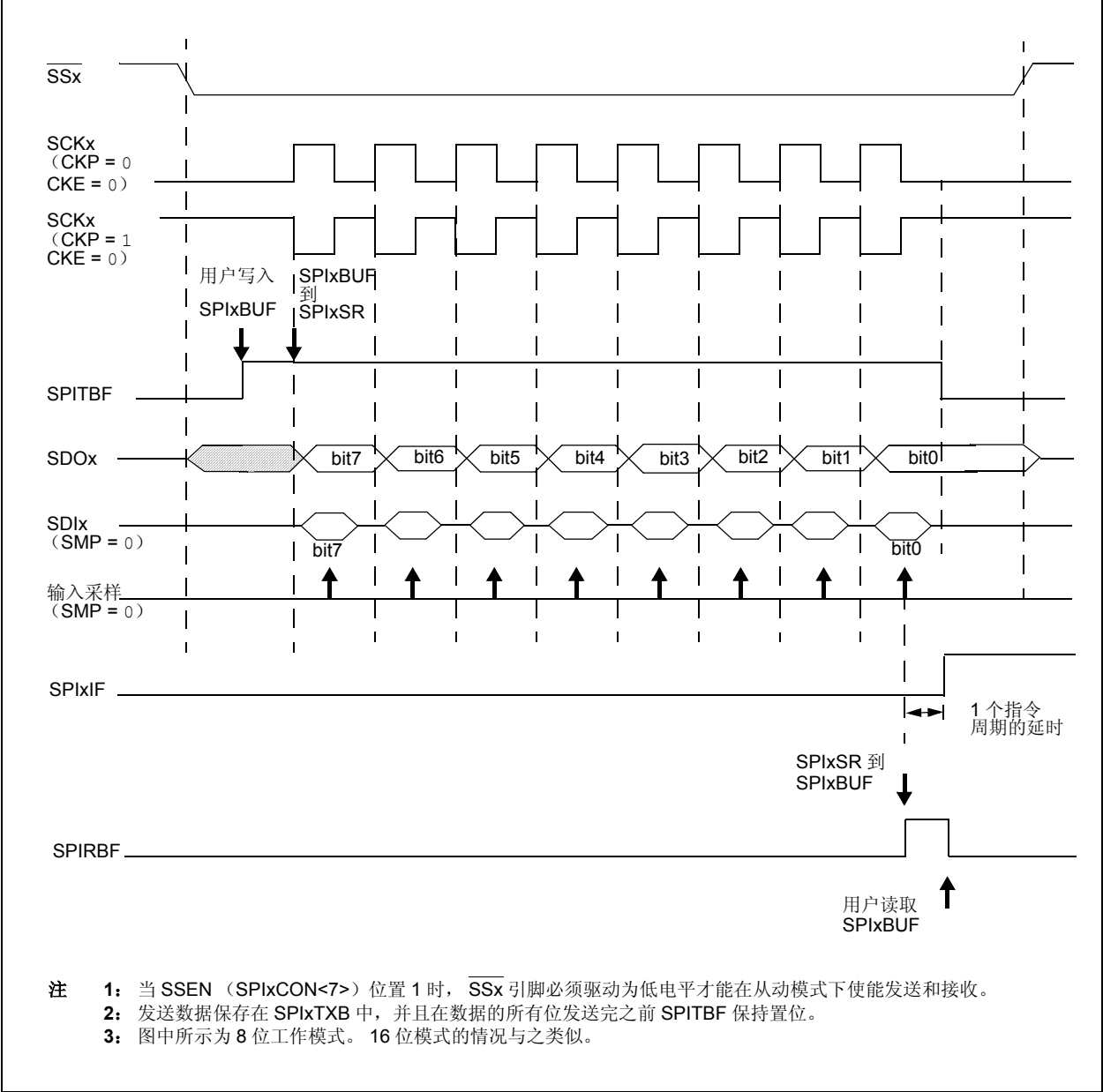
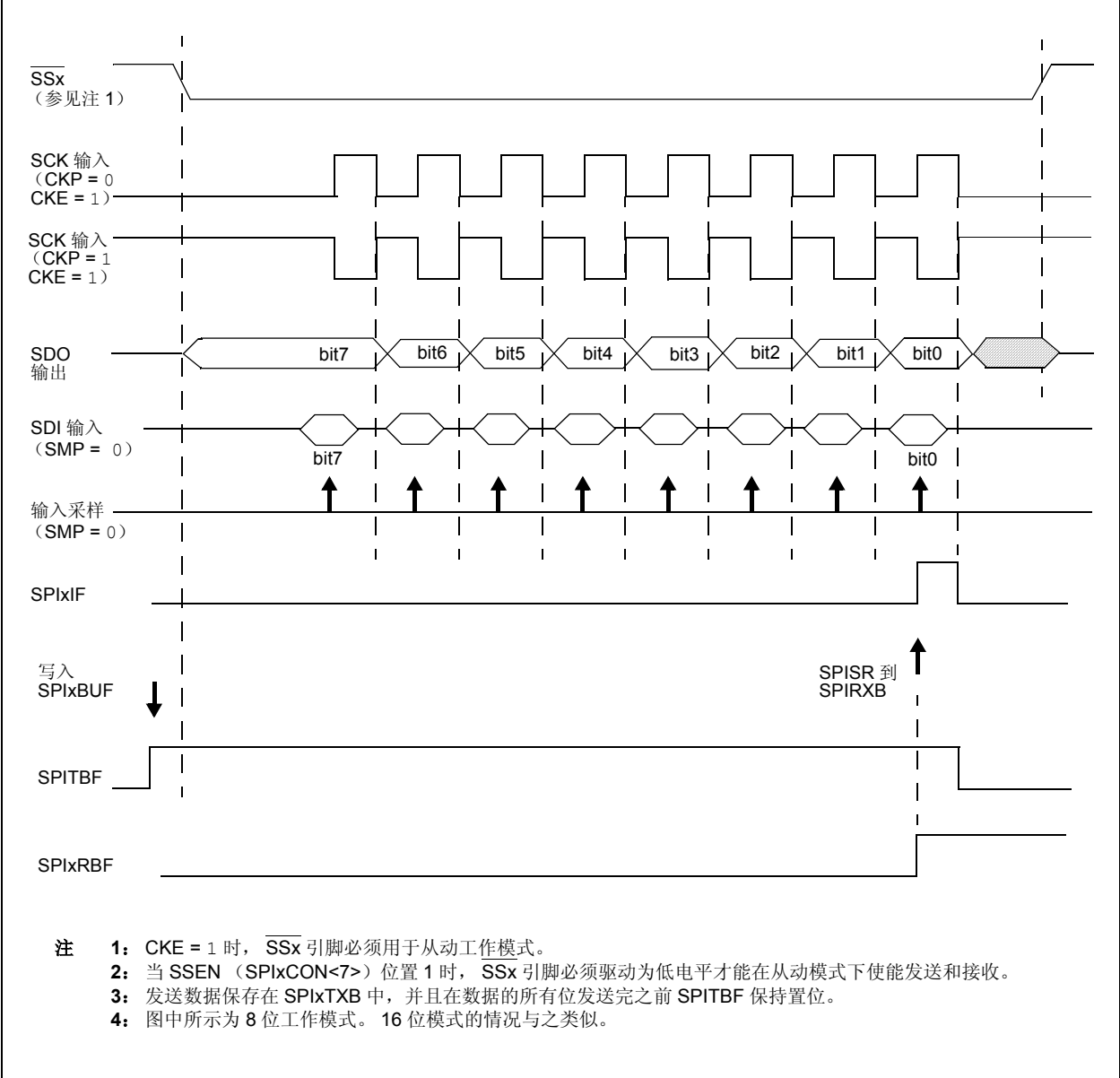


图 20-5: 从动选择引脚使能时的 SPI 从动模式工作原理



- 注
- 1: 当 SSEN (SPIxCON<7>) 位置 1 时, $\overline{\text{SSx}}$ 引脚必须驱动为低电平才能在从动模式下使能发送和接收。
 - 2: 发送数据保存在 SPIxTXB 中, 并且在数据的所有位发送完之前 SPITBF 保持置位。
 - 3: 图中所示为 8 位工作模式。16 位模式的情况与之类似。

图 20-6: SPI 模式时序 (从动模式且 CKE = 1)



20.3.3 SPI 错误处理

当新的数据字移入 SPIxSR 并且用户软件未读取 SPIxRXB 的原先内容时, SPIROV 位 (SPIxSTAT<6>) 将置位。模块不会将接收到的数据从 SPIxSR 传输到 SPIxRXB。在 SPIROV 位清零前, 禁止接收后续数据。SPIROV 位不会被模块自动清零, 它必须由用户软件清零。

20.3.4 SPI 仅启用接收功能的工作原理

将控制位 DISSDO (SPIxCON<11>) 置位, 禁止 SDOx 引脚的发送功能。这样可以使 SPIx 模块配置为仅接收的工作模式。如果 DISSDO 位置位, SDOx 引脚将由相应端口功能控制。

DISSDO 功能适用于所有 SPI 工作模式。

20.3.5 帧 SPI 模式

当工作在主控模式或从动模式时, 模块支持一个非常基本的帧 SPI 协议。在 SPI 模块中提供了以下功能来支持帧 SPI 模式:

- 控制位 FRMEN (SPIxCON<14>) 可启用帧 SPI 模式并使 $\overline{SSx}$ 引脚作为帧同步脉冲输入或输出引脚使用。SSEN (SPIxCON<7>) 的状态会被忽略。
- 控制位 SPIFSD (SPIxCON<13>) 决定 $\overline{SSx}$ 引脚的输入 / 输出方向 (即模块是接收还是产生帧同步脉冲)。
- 帧同步脉冲在一个 SPI 时钟周期内为高电平有效脉冲。

SPI 模块支持以下两种帧 SPI 模式:

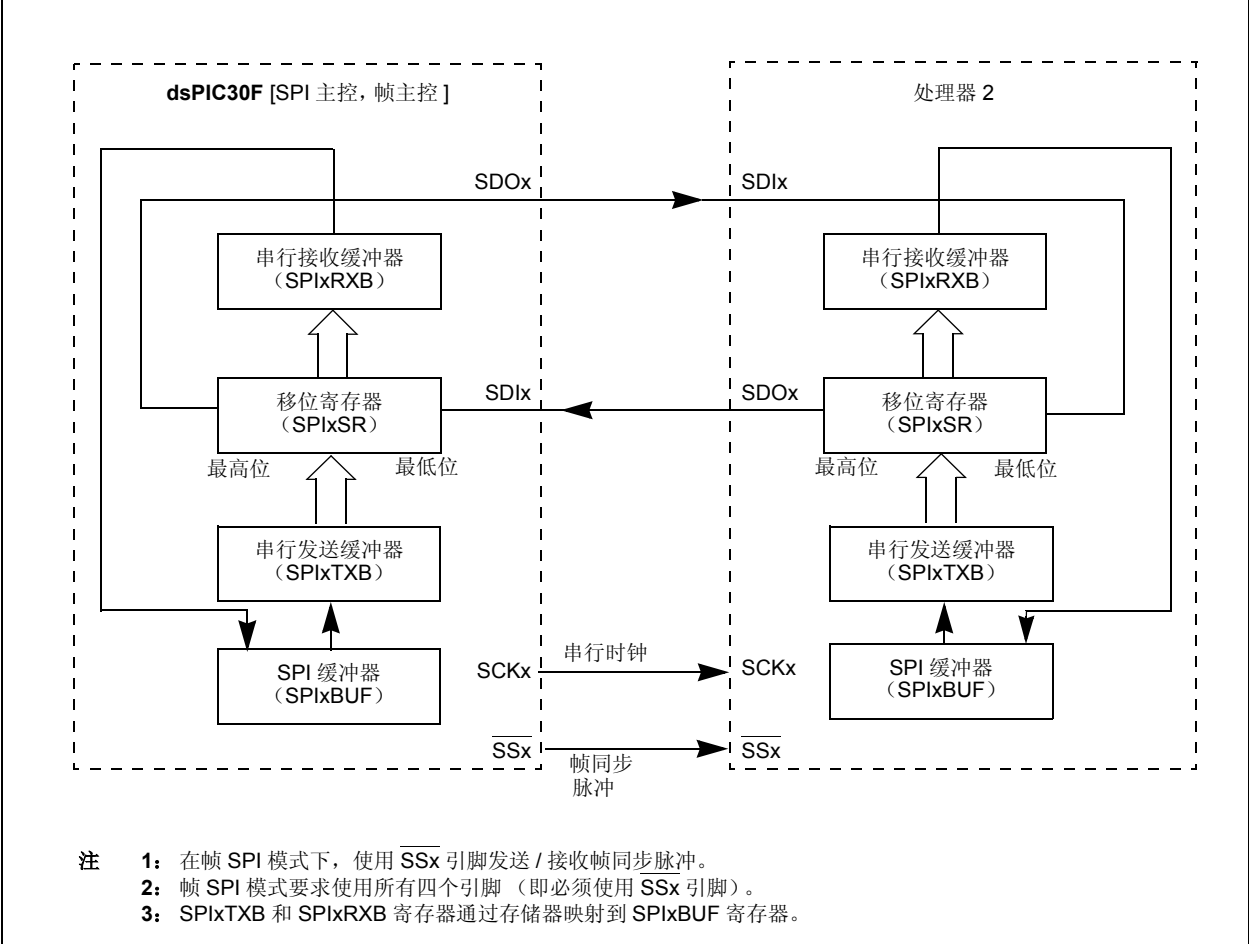
- 帧主控模式: SPI 模块产生帧同步脉冲并在 $\overline{SSx}$ 引脚为其他器件提供此脉冲。
- 帧从动模式: SPI 模块使用在 $\overline{SSx}$ 引脚接收到的帧同步脉冲。

主控模式和从动模式都支持帧 SPI 模式。因此, 用户可以使用以下四种帧 SPI 配置:

- SPI 主控模式和帧主控模式
- SPI 主控模式和帧从动模式
- SPI 从动模式和帧主控模式
- SPI 从动模式和帧从动模式

这四种模式决定 SPIx 模块是否产生串行时钟和帧同步脉冲。

图 20-7: SPI 主控模式、帧主控模式框图



注 1: 在帧 SPI 模式下, 使用 $\overline{SSx}$ 引脚发送 / 接收帧同步脉冲。
2: 帧 SPI 模式要求使用所有四个引脚 (即必须使用 $\overline{SSx}$ 引脚)。
3: SPIxTXB 和 SPIxRXB 寄存器通过存储器映射到 SPIxBUF 寄存器。

20.3.5.1 在帧 SPI 模式下的 SCKx

当 $FRMEN (SPIxCON<14>) = 1$ 且 $MSTEN (SPIxCON<5>) = 1$ 时, SCKx 引脚为输出引脚, 且 SCKx 上的 SPI 时钟成为自由运行时钟。

当 $FRMEN = 1$ 且 $MSTEN = 0$ 时, SCKx 引脚成为输入引脚。假设提供给 SCKx 引脚的源时钟信号是自由运行时钟信号。

时钟的极性由 $CKP (SPIxCON<6>)$ 位选择。CKE (SPIxCON<8>) 位在分帧 SPI 模式下未使用, 应该在用户软件中编程为 0。

当 $CKP = 0$ 时, 帧同步脉冲输出和 SDOx 数据输出在 SCKx 引脚的时钟脉冲上升沿变化。在串行时钟的下降沿, 在 SDIx 输入引脚上采样输入数据。

当 $CKP = 1$ 时, 帧同步脉冲输出和 SDOx 数据输出在 SCKx 引脚的时钟脉冲下降沿变化。在串行时钟的上升沿, 在 SDIx 输入引脚上采样输入数据。

20.3.5.2 在帧 SPI 模式下的 SPIx 缓冲器

当 SPIFSD (SPIxCON<13>) = 0 时, SPIx 模块处于帧主控工作模式。在此模式下, 当用户软件将发送数据写入 SPIxBUF 地址单元 (从而可将发送数据写入 SPIxTXB 寄存器) 时, 模块启动帧同步脉冲。在帧同步脉冲的末尾, SPIxTXB 的数据被传输到 SPIxSR, 同时开始发送/接收数据。

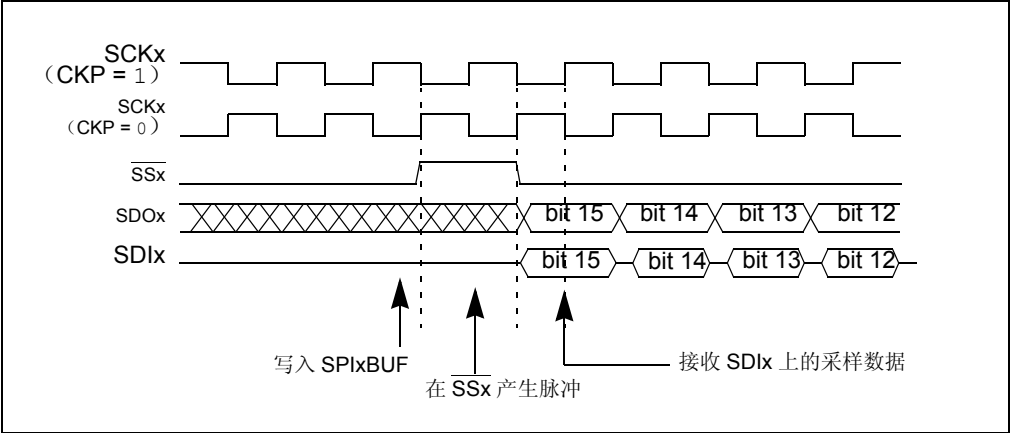
当 SPIFSD (SPIxCON<13>) = 1 时, 模块处于帧从动模式。在此模式下, 帧同步脉冲由外部时钟源提供。当模块采样帧同步脉冲时, 它将把 SPIxTXB 寄存器的内容传输到 SPIxSR, 同时开始发送/接收数据。在接收到帧同步脉冲前, 用户必须确保在 SPIxBUF 中装入了要发送的正确数据。

注: 无论数据是否写入 SPIxBUF, 接收到帧同步脉冲的同时都将开始发送。如果新数据尚未写入, 将发送 SPIxTXB 的原有数据。

20.3.5.3 SPI 主控模式和帧主控模式

通过将 MSTEN (SPIxCON<5>) 和 FRMEN (SPIxCON<14>) 位置 1 并将 SPIFSD (SPIxCON<13>) 位清零可使得此帧 SPI 模式。在此模式下, 无论模块是否正在发送, 串行时钟都将在 SCKx 引脚连续输出。当写入 SPIxBUF 时, SSx 引脚将在 SCKx 时钟的下一个发送沿驱动为高电平。SSx 引脚在一个 SCKx 时钟周期内将为高电平。如图 20-8 所示, 模块将在 SCKx 的下一个发送沿开始发送数据。图 20-7 所示为表明此工作模式的信号方向的连接图。

图 20-8: SPI 主控模式、帧主控模式



20.3.5.4 SPI 主控模式和帧从动模式

通过将 MSTEN、FRMEN 和 SPIFSD 位置 1 可启用此帧 SPI 模式。 $\overline{\text{SSx}}$ 引脚为输入引脚，并在 SPI 时钟的采样沿对其进行采样。如图 20-9 所示，当采样到高电平时，在紧接着的 SPI 时钟发送沿就会发送数据。当发送完成时中断标志 SPIxIF 将置位。在 $\overline{\text{SSx}}$ 引脚接收到信号前，用户必须确保在 SPIxBUF 中装入了正确的待发送数据。图 20-10 所示为表示此工作模式的信号方向的连接图。

图 20-9: SPI 主控模式，帧从动模式

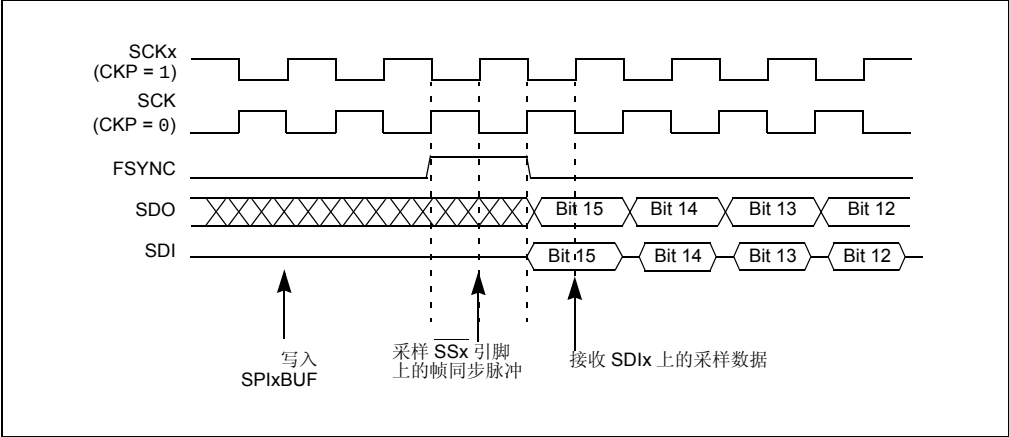
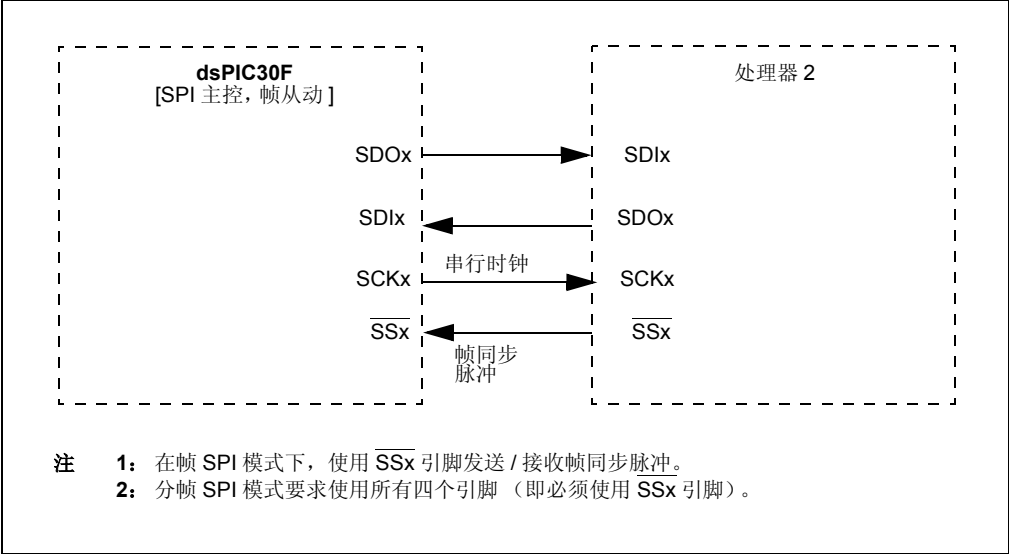


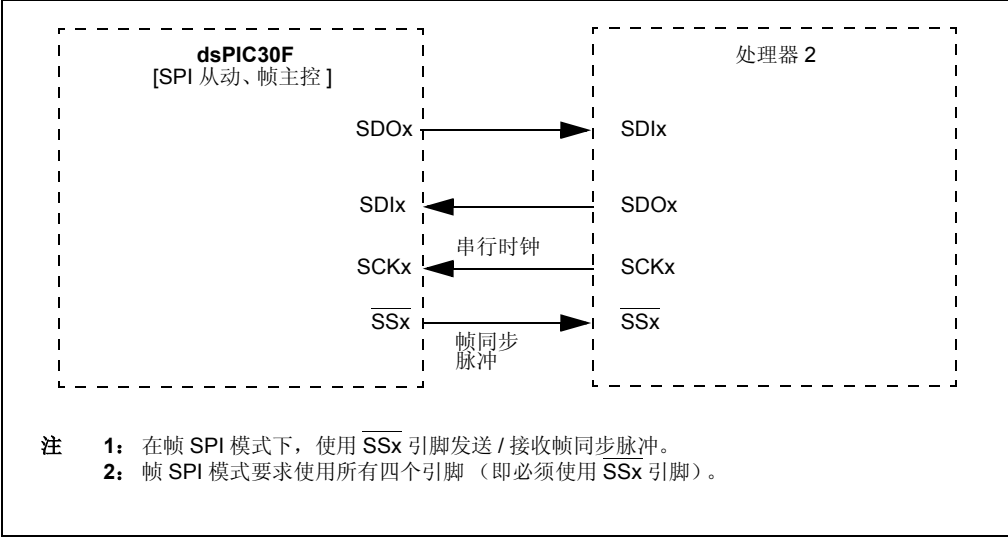
图 20-10: SPI 主控模式、帧从动模式连接图



20.3.5.5 SPI 从动模式和帧主控模式

通过将 MSTEN (SPIxCON<5>) 位清零、FRMEN (SPIxCON<14>) 位置 1 以及 SPIFSD (SPIxCON<13>) 位清零, 可使能此帧 SPI 模式。在从动模式下, 将继续使用输入 SPI 时钟。当 SPIFSD 位为低时, SSx 引脚是输出引脚。因此, 当写入 SPIBUF 时, 模块将在 SPI 时钟的下一个发送沿把 SSx 引脚驱动为高电平。SSx 引脚在一个 SPI 时钟周期内将保持驱动为高电平。将在下一个 SPI 时钟发送沿开始发送数据。图 20-11 所示为表示此工作模式的信号方向的连接图。

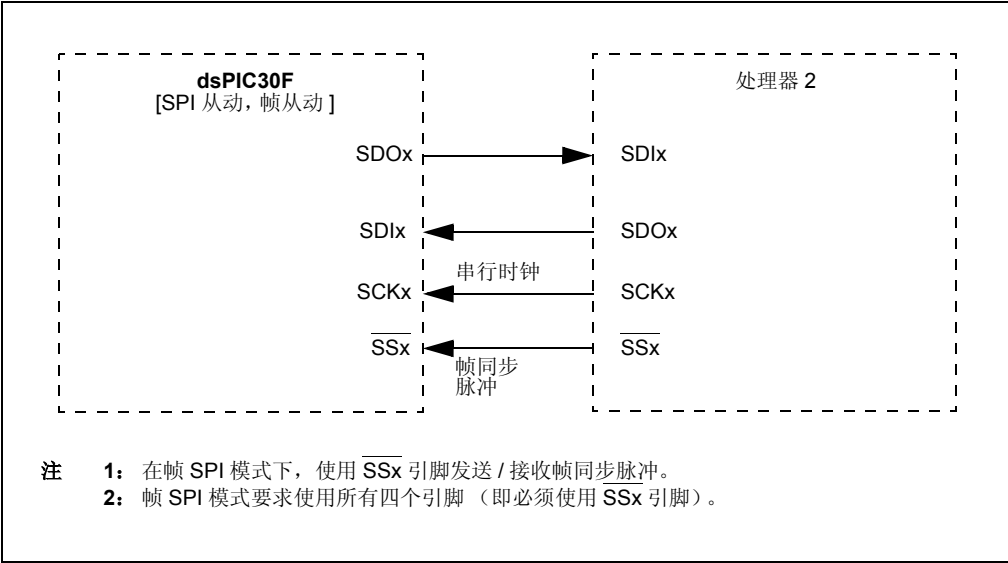
图 20-11: SPI 从动模式、帧主控模式连接图



20.3.5.6 SPI 从动模式和帧从动模式

通过将 MSTEN (SPIxCON<5>) 位清零、FRMEN (SPIxCON<14>) 位置 1 以及 SPIFSD (SPIxCON<13>) 位置 1, 可使能此帧 SPI 模式。因此, SCKx 和 SSx 引脚都将是输入引脚。将在 SPI 时钟的采样沿采样 SSx 引脚。当采样到 SSx 引脚上为高电平时, 将在下一个 SCKx 发送沿发送数据。图 20-12 所示为表示此工作模式的信号方向的连接图。

图 20-12: SPI 从动模式、帧从动模式连接图



20.4 SPI 主控模式时钟频率

在 主控模式下，提供给 SPI 模块的时钟周期就是指令周期（Tcy）。然后将此时钟信号由主预分频器（由 PPRE<1:0>（SPIxCON<1:0>）指定）和辅助预分频器（由 SPRE<2:0>（SPIxCON<4:2>）指定）预分频。经过预分频的指令时钟就变为串行时钟并通过 SCKx 引脚提供给外部器件。

注： 注意 SCKx 信号时钟在正常 SPI 模式下不是自由运行的。它仅在 SPIxBUF 加载了数据后运行 8 或 16 个脉冲时间。但是在分帧模式下，它会连续运行。

公式 20-1 可用于计算 SCKx 时钟频率主预分频器和辅助预分频器设置值的函数。

公式 20-1:

$$F_{SCK} = \frac{F_{CY}}{\text{主预分频比} \times \text{辅助预分频比}}$$

下表列出了部分范例 SPI 时钟频率（单位为 kHz）：

表 20-1: 范例 SCKx 频率

Fcy = 30 MHz		辅助预分频比设置				
		1:1	2:1	4:1	6:1	8:1
主预分频比设置	1:1	30000	15000	7500	5000	3750
	4:1	7500	3750	1875	1250	938
	16:1	1875	938	469	313	234
	64:1	469	234	117	78	59
Fcy = 5 MHz						
主预分频比设置	1:1	5000	2500	1250	833	625
	4:1	1250	625	313	208	156
	16:1	313	156	78	52	39
	64:1	78	39	20	13	10

注： 表中 SCKx 频率的单位为 kHz。

注： 并不支持所有的时钟速率。如需更多信息，参见特定器件数据手册的 SPI 时序规范。

20.5 低功耗模式下的工作

dsPIC30FXXXX 系列器件具有三种能耗模式：

- 工作模式：内核与外设均处于运行状态。
- 低功耗模式：通过执行 PWRSAV 指令可进入该模式。dsPIC30F 系列器件支持两种低功耗模式。在 PWRSAV 指令中可以通过参数来指定具体的模式。这两种模式为：

- 休眠模式：器件时钟源和整个器件都关闭。可通过以下指令实现。

```
;include device p30fxxxx.inc file
PWRSAV #SLEEP_MODE
```

- 空闲模式：器件时钟处于工作状态，CPU 和所选外设关闭。

```
;include device p30fxxxx.inc file
PWRSAV #IDLE_MODE
```

20.5.1 休眠模式

当器件进入休眠模式后，系统时钟会被禁止。

20.5.1.1 主控模式下的工作

以下为将 SPIx 模块配置为主控工作模式时进入休眠模式的后果：

- SPIx 模块的波特率发生器停止并复位。
- 如果 SPIx 模块在发送 / 接收的过程中进入休眠模式，则发送 / 接收将被中止。因为在发送或接收未完成时没有自动的方式能防止 SPIx 模块进入休眠模式，因此用户软件必须将进入休眠与 SPI 模块工作同步以防止传输中止。
- 在休眠模式下发送器和接收器将停止工作。发送器或接收器在被唤醒后不会继续尚未完成的传输。

20.5.1.2 从动模式下的工作

因为在从动模式下，SCKx 的时钟脉冲由外部提供，所以模块在休眠模式下将继续工作。它将在进入到休眠的过渡时间内完成所有事务处理（Transaction）。完成事务处理后，SPIRBF 标志将置位。从而将 SPIxIF 位置位。如果允许 SPI 中断（SPIxIE = 1），器件将从休眠模式唤醒。如果 SPI 中断的优先级高于当前 CPU 优先级，将从 SPIx 中断向量地址处恢复代码执行。否则，将继续执行在进入休眠模式之前执行的 PWRSAV 指令后的代码。如果此模块作为从器件工作，那么在进入休眠模式时它将不会复位。

当 SPIx 模块进入或退出休眠模式时，寄存器内容不受影响。

20.5.2 空闲模式

当器件进入空闲模式时，系统时钟源保持工作。SPISIDL 位（SPIxSTAT<13>）选择模块在空闲模式下是停止工作还是继续工作。

- 如果 SPISIDL = 1，SPI 模块将在进入空闲模式时停止通信。其工作状态将和处于休眠模式时相同。
- 如果 SPISIDL = 0（默认选择），模块将在空闲模式下继续工作。

表 20-2: 与 SPI 模块相关的引脚

引脚名称	引脚类型	缓冲器类型	说明
SCK1	I/O	CMOS	SPI1 模块时钟输入或输出
SCK2	I/O	CMOS	SPI2 模块时钟输入或输出
SDI1	I	CMOS	SPI1 模块数据接收引脚
SDI2	I	CMOS	SPI2 模块数据接收引脚
SDO1	O	CMOS	SPI1 模块数据发送引脚
SDO2	O	CMOS	SPI2 模块数据发送引脚
SS1	I/O	CMOS	SPI1 模块从动选择控制引脚 1) 置位 SSEN（SPI1CON<7> 时，用于在从动模式下使能发送 / 接收 2) 在 FRMEN 和 SPIFSD（SPI1CON<14:13>）置为 11 或 10 时，作为帧同步 I/O 脉冲
SS2	I/O	CMOS	SPI2 模块从动选择控制引脚 1) 置位 SSEN（SPI2CON<7>）时，用于在从动模式下使能发送 / 接收 2) 在 FRMEN 和 SPIFSD（SPI2CON<14:13>）置为 11 或 10 时，作为帧同步 I/O 脉冲

图注： CMOS = CMOS 兼容的输入或输出， ST = CMOS 电平的施密特触发输入，
I = 输入， O = 输出

20.6 与 SPI 模块相关的特殊功能寄存器

表 20-3: SPI1 寄存器映射

SFR 名称	地址	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	复位状态
SPI1STAT	0220	SPIEN	—	SPISIDL	—	—	—	—	—	—	SPIROV	—	—	—	—	SPITBF	SPIRBF	0000 0000 0000 0000
SPI1CON	0222	—	FRMEN	SPIFSD	—	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0	0000 0000 0000 0000
SPI1BUF	0224	SPI1TXB 和 SPI1RXB 寄存器共享的发送和接收缓冲器地址单元																0000 0000 0000 0000

表 20-4: SPI2 寄存器映射

SFR 名称	地址	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	复位状态
SPI2STAT	0226	SPIEN	—	SPISIDL	—	—	—	—	—	—	SPIROV	—	—	—	—	SPITBF	SPIRBF	0000 0000 0000 0000
SPI2CON	0228	—	FRMEN	SPIFSD	—	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0	0000 0000 0000 0000
SPI2BUF	022A	SPI2TXB 和 SPI2RXB 寄存器共享的发送和接收缓冲器地址单元																0000 0000 0000 0000

表 20-5: SPI 模块相关的中断寄存器

SFR 名称	地址	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	复位状态
INTCON1	0080	NSTDIS	—	—	—	—	OVATE	OVBT	COVTE	—	—	—	SWTRAP	OVRFLOW	ADDRERR	STKERR	—	0000 0000 0000 0000
INTCON2	0082	ALTIVT	DISI	—	—	—	—	LEV8F	—	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP	0000 0000 0000 0000
IFS0	0084	CNIF	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0	0000 0000 0000 0000
IFS1	0086	IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IEC0	008C	CNIE	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IPC2	0098	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			0100 0100 0100 0100
IPC6	00A0	—	C1IP<2:0>			—	SPI2IP<2:0>			—	U2TXIP<2:0>			—	U2RXIP<2:0>			0100 0100 0100 0100

20.7 相关应用笔记

本节列出了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品编写的，但是概念是相关的，通过适当修改即可使用，但使用中可能会受到一定限制。当前与串行外设接口（SPI）模块相关的应用笔记有：

标题	应用笔记编号
Interfacing Microchip's MCP41XXX/MCP42XXX Digital Potentiometers to a PIC [®] Microcontroller	AN746
Interfacing Microchip's MCP3201 Analog-to-Digital Converter to the PIC [®] Microcontroller	AN719

注：如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

20.8 版本历史

版本 A

这是本文档的初始发行版。

版本 B

此版本反映了 dsPIC30F 串行外设接口（SPI）模块的编辑修订及技术内容修改。

版本 C

本章未包含任何技术内容修订，然而我们谨更新此节以反映本手册已更新到版本 C。

第 21 章 I²C™ 模块

目录

本章包括下列主题：

21.1 概述	21-2
21.2 I ² C 总线特性	21-4
21.3 控制和状态寄存器	21-7
21.4 使能 I ² C 操作	21-13
21.5 作为主器件在单主机环境下通信	21-15
21.6 作为主器件在多主机环境下通信	21-29
21.7 作为从器件通信	21-32
21.8 I ² C 总线的连接注意事项	21-47
21.9 在 PWRSV 指令执行期间的模块操作	21-49
21.10 复位的影响	21-49
21.11 设计技巧	21-50
21.12 相关应用笔记	21-51
21.13 版本历史	21-52

21.1 概述

I²C™ (Inter-integrated Circuit) 模块是用来与其他外设或单片机通信的串行接口。这些外设可以是串行 EEPROM、显示驱动器和 A/D 转换器等。

I²C 模块可以工作在任何下列 I²C 系统中：

- dsPIC30F 作为从器件的系统
- dsPIC30F 作为主器件的单主机系统
(也可以作为从器件)
- dsPIC30F 作为主 / 从器件的多主机系统
(具有总线冲突检测和仲裁)

I²C 模块包含独立的 I²C 主逻辑和 I²C 从逻辑，两者都会根据相应事件产生中断。在多主机系统中，软件仅被简单地分为主控制器和从控制器。

当 I²C 主逻辑有效时，从逻辑也保持有效，以检测总线状态并可能接收自身的报文（单主机系统）或来自其他主器件的报文（多主机系统）。在多主机总线仲裁时不会丢失报文。

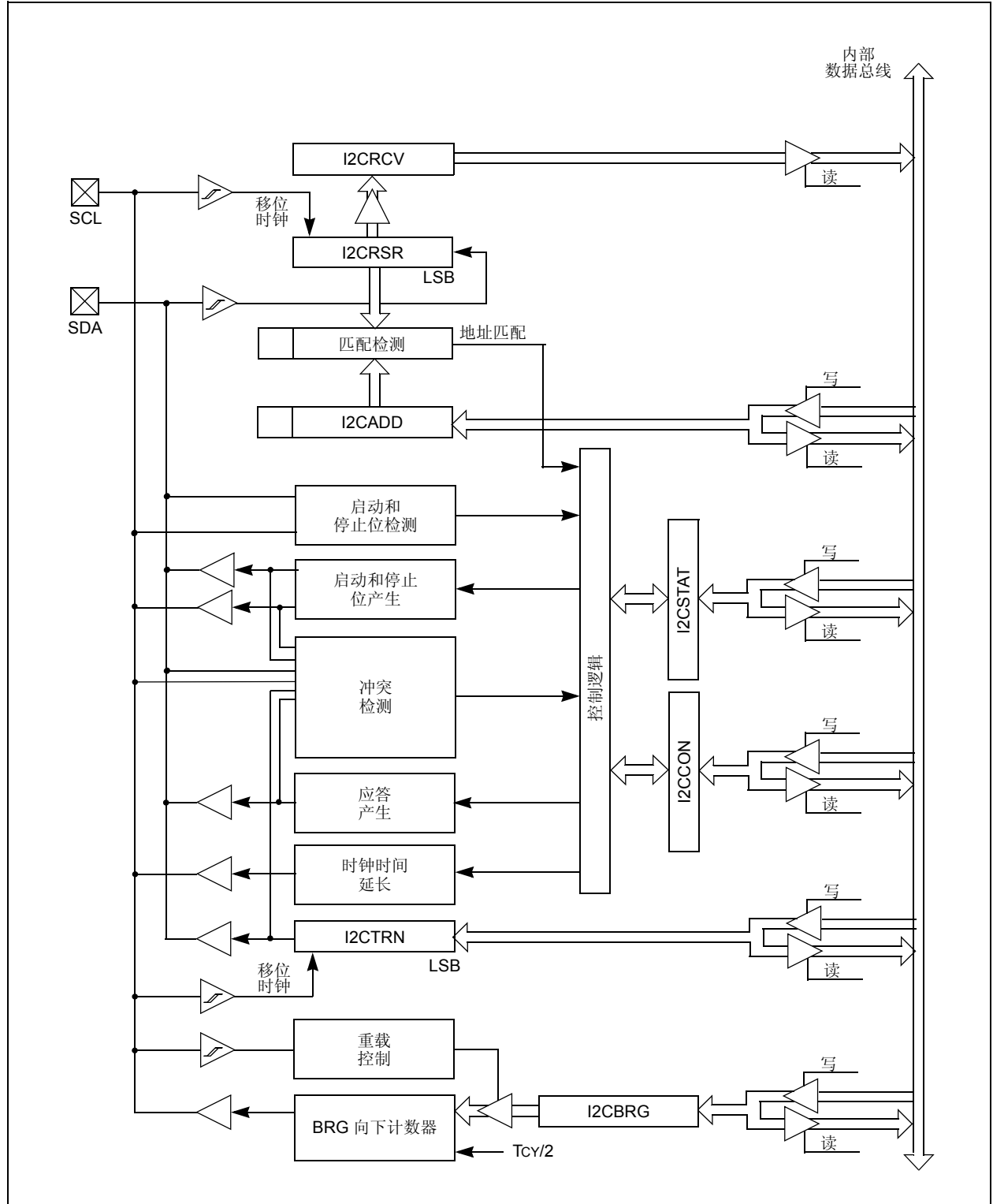
在多主机系统中，与系统中其他主器件冲突的总线冲突会被检测到，模块提供了终止并重新开始报文传输的方法。

I²C 模块包含一个波特率发生器。I²C 波特率发生器不消耗器件中的其他定时器资源。

21.1.1 模块特点

- 独立的主从逻辑
- 支持多主机模式。在仲裁时不丢失报文。
- 检测 7 位和 10 位器件地址
- 按 I²C 协议的定义检测广播呼叫地址
- 总线重发器模式。作为从器件，不论报文地址如何，可接收所有报文。
- 自动 SCL 时钟时间延长可为处理器提供响应从器件数据请求的延时。
- 支持 100 kHz 到 400 kHz 总线规范。

图 21-1 所示为 I²C 模块框图。

图 21-1: I²C™ 框图

21.2 I²C 总线特性

I²C 总线是一个双线串行接口。图 21-2 是 dsPIC30F 器件和 24LC256 I²C 串行 EEPROM 的典型 I²C 连接示意图。

I²C 接口使用了全面的协议以确保可靠的数据发送与接收。在通信时，一个器件作为主器件启动总线上的传输并产生时钟信号来允许传输，而其他器件作为响应传输的从器件。时钟线 “SCL” 是从主器件输出并输入到从器件，虽然从器件偶尔也会驱动 SCL 线。数据线 “SDA” 可以是主器件和从器件两者的输出和输入。

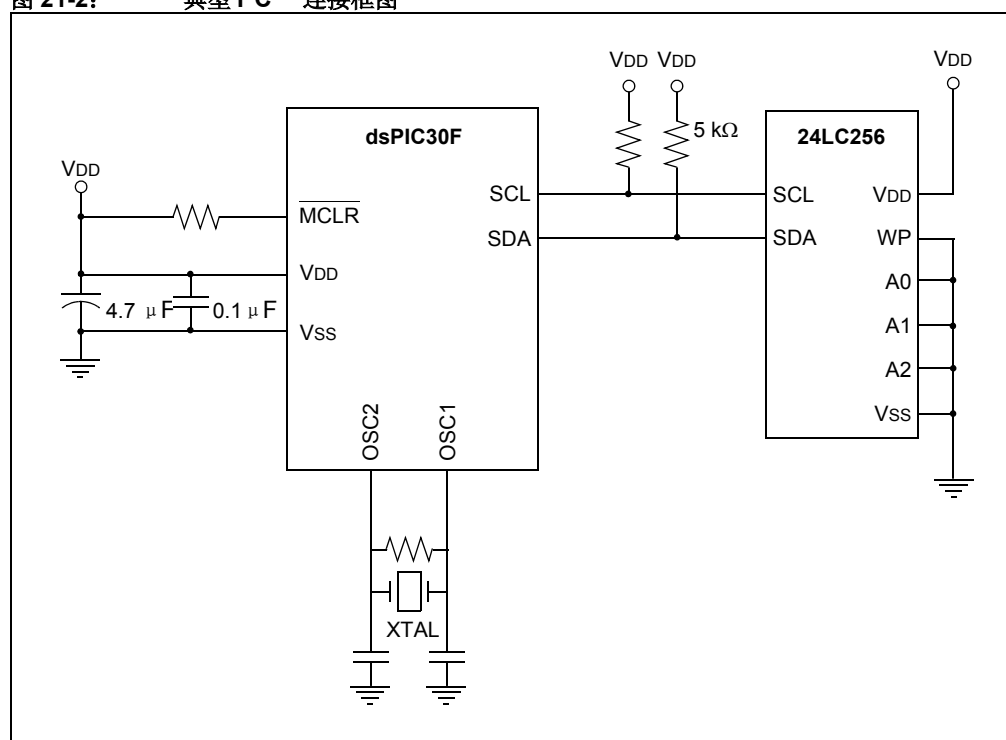
因为 SDA 和 SCL 线是双向的，驱动 SDA 和 SCL 线的器件的输出级必须漏极开路以执行总线的“线与”功能。另外使用了外部上拉电阻，以确保当没有器件将线拉低时能保持高电平。

在 I²C 接口协议中，每个器件都有一个地址。当主器件要开始数据传输时，它首先发送想要“通话”的目标器件地址。所有从器件都会“接听”，以确定是否与其地址匹配。在该地址中，bit 0 指定主器件是希望读还是写从器件。在数据传输时，主器件和从器件总是工作在相反的模式（发送器 / 接收器）下。也就是说主器件和从器件可以工作在下面两种关系之一：

- 主器件——发送器，从器件——接收器
- 从器件——发送器，主器件——接收器

在这两种情况下，都由主器件产生 SCL 时钟信号。

图 21-2: 典型 I²C™ 连接框图



21.2.1 总线协议

定义了下列 I²C 总线协议：

- 只有在总线不忙时才可以启动数据传输。
- 在数据传输时，只要 SCL 时钟线为高电平，数据线就必须保持稳定。当 SCL 时钟线为高电平时数据线发生变化，将被解释为“启动”或“停止”条件。

与此对应，定义了以下总线条件（图 21-3）。

21.2.1.1 启动数据传输（S）

在总线空闲状态后，当时钟（SCL）为高电平时，SDA 线由高电平变为低电平会产生“启动”条件。所有数据传输前必须有一个“启动”条件。

21.2.1.2 停止数据传输（P）

当时钟（SCL）处于高电平时，SDA 线由低电平变为高电平会产生“停止”条件。所有的数据传输必须以“停止”条件结束。

21.2.1.3 重复启动（R）

在“等待”状态后，当时钟（SCL）为高电平时，SDA 线由高电平变为低电平会产生“重复启动”条件。重复启动可以让主器件在不失去总线控制的情况下改变总线方向。

21.2.1.4 数据有效（D）

在启动条件之后，如果 SDA 线在时钟信号的高电平期间保持稳定，则 SDA 线的状态代表有效数据。每个 SCL 时钟都有一位数据。

21.2.1.5 应答（A）或不应答（N）

所有的数据字节传输必须由接收器应答（ACK）或不应答（NACK）。接收器会将 SDA 线拉低发出 ACK 或释放 SDA 线发出 NACK。使用一个 SCL 时钟，应答信号需要一个一位周期。

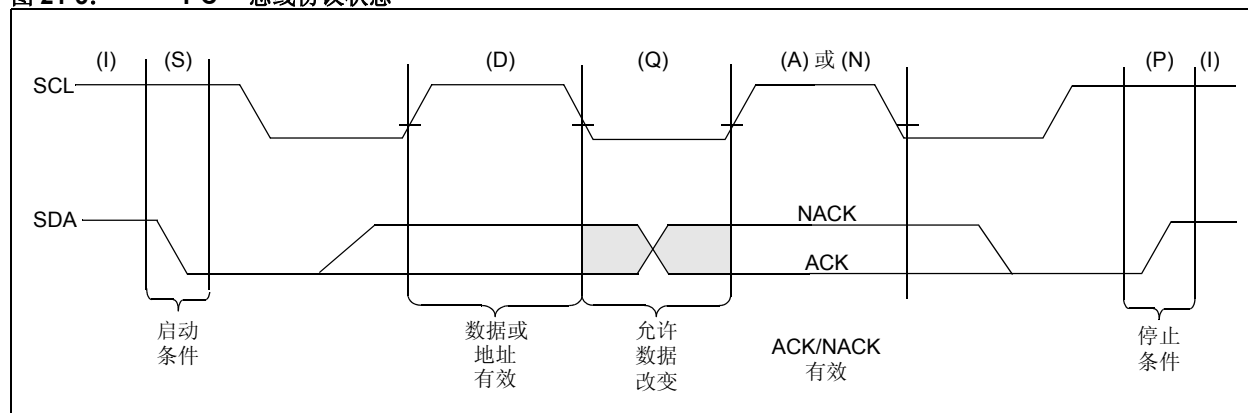
21.2.1.6 等待 / 数据无效（Q）

在时钟信号的低电平周期，必须修改线上数据。通过将 SCL 线拉低，器件可以延长时钟低电平时间，导致总线的“等待”状态。

21.2.1.7 总线空闲（I）

在停止条件后，启动条件前，数据线和时钟线在这些时间段保持高电平。

图 21-3: I²C™ 总线协议状态

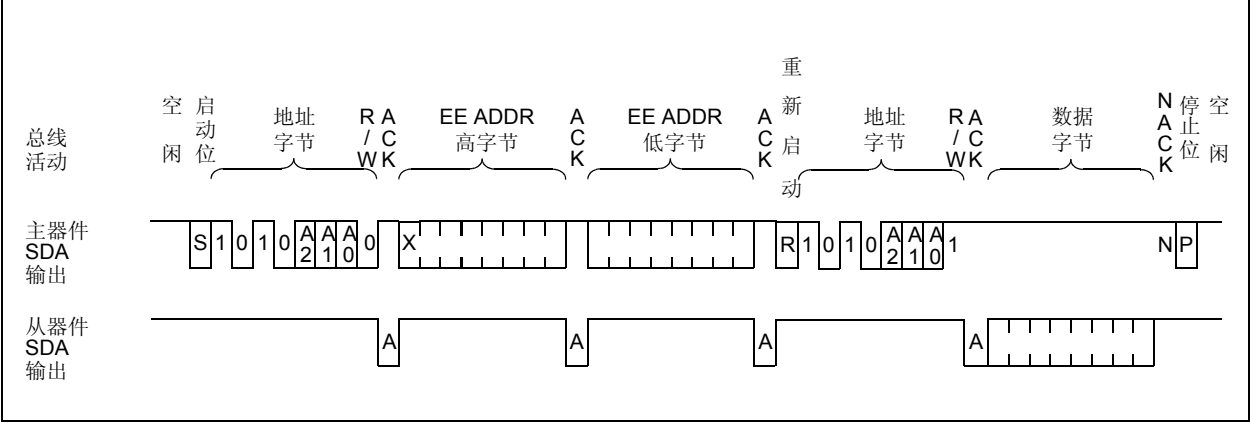


21.2.2 报文协议

图 21-4 所示为典型的 I²C 报文。在此示例中，报文会从 24LC256 的 I²C 串行 EEPROM 读取指定的字节。dsPIC30F 器件将作为主器件，24LC256 器件将作为从器件。

图 21-4 表明由主器件和从器件驱动的数据，注意复合的 SDA 线上是主数据与从数据的“线与”值。主器件控制协议及其时序。从器件只在特别确定的时间驱动总线。

图 21-4: 典型 I²C™ 报文：读取串行 EEPROM（随机寻址模式）



21.2.2.1 启动报文

所有报文都以“启动”条件开始并以“停止”条件终止。在启动和停止条件之间传输的数据字节数取决于主器件。如系统协议所定义的，报文的字节可以有“器件地址字节”或“数据字节”等特殊意义。

21.2.2.2 寻址从器件

如图所示，第一个字节是器件地址字节，任何 I²C 报文的最初部分必须为此字节。它包含一个器件地址和一个 R/W 位。如需有关地址字节格式的更多信息，请参见第 26 章“附录”。注意该第一个地址字节的 R/W = 0，表示主器件将充当发送器，从器件则将是接收器。

21.2.2.3 从器件应答

接收到每个字节后，接收器件必须产生应答信号“ACK”。主器件必须产生一个与该应答位有关的额外的 SCL 时钟信号。

21.2.2.4 主器件发送

主器件发送到从器件的接下去 2 个字节是包含所请求 EEPROM 数据字节位置的数据字节。从器件必须应答每个数据字节。

21.2.2.5 重复启动

此时，从器件 EEPROM 拥有将所请求数据字节返回主器件所必需的地址信息。但是，第一个器件地址字节中的 R/W 位指定了主器件发送，从器件接收。要让从器件向主器件发送数据，总线必须转为另一个方向。

要实现此功能且不终止报文传送，主器件可发送一个“重新启动”信号。“重新启动”后接一个器件地址字节，该字节包含和前面相同的器件地址，但 R/W = 1，以表明从器件发送，主器件接收。

21.2.2.6 从器件回复

现在从器件发送驱动 SDA 线的数据字节，主器件继续产生时钟信号，但是释放其 SDA 驱动。

21.2.2.7 主器件应答

在读取时，主器件必须通过对报文的最后一个字节做出“不应答”（产生一个“NACK”）来终止对从器件的数据请求。

21.2.2.8 停止报文

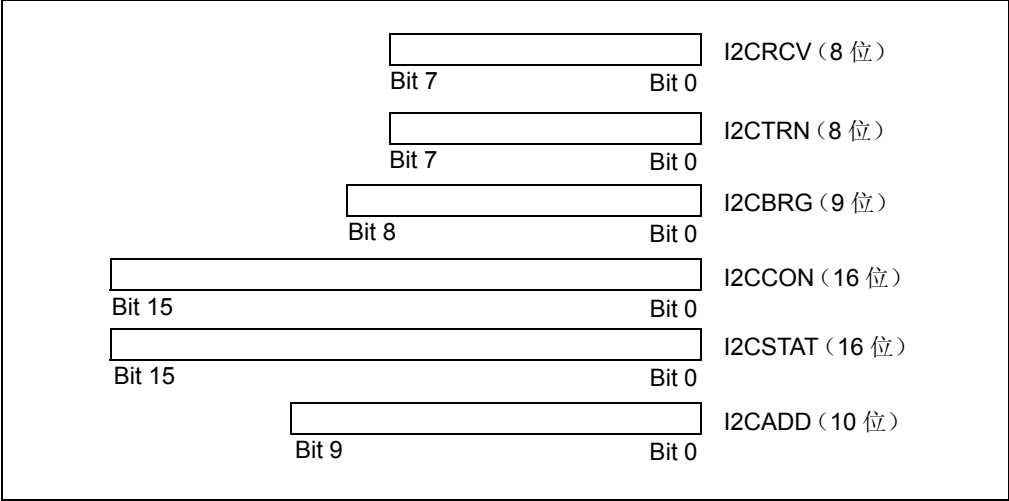
主器件发送停止信号终止报文并将总线恢复到空闲状态。

21.3 控制和状态寄存器

I²C 模块有六个用户可访问的寄存器用于 I²C 操作。这些寄存器可以字节或字模式访问。见图 21-5 并在下面列出：

- 控制寄存器（I2CCON）：此寄存器允许控制 I²C 工作方式。
- 状态寄存器（I2CSTAT）：此寄存器包含表明 I²C 工作过程中模块状态的状态标志。
- 接收缓冲寄存器（I2CRCV）：这是可从中读取数据字节的缓冲寄存器。I2CRCV 寄存器是只读寄存器。
- 发送寄存器（I2CTRN）：这是发送寄存器，在发送操作时，字节会写入此寄存器。I2CTRN 寄存器是读 / 写寄存器。
- 地址寄存器（I2CADD）：I2CADD 寄存器保存从器件地址。
- 波特率发生器重载寄存器（I2CBRG）：保存 I²C 模块波特率发生器的波特率发生器重载值。

图 21-5: I²C™ 编程模型



寄存器 21-1 和寄存器 21-2 定义 I²C 模块控制和状态寄存器 I2CCON 和 I2CSTAT。

I2CTRN 是写入待发送数据的寄存器。当模块作为主器件发送数据到从器件或从器件发送应答数据到主器件时，要使用此寄存器。在报文传输过程中，I2CTRN 寄存器会移出各个位。因此，除非总线处于空闲状态，否则可能无法写入 I2CTRN。在发送当前数据时，可以重载 I2CTRN。

主器件或从器件正在接收的数据被移入一个名为 I2CRSR 的不可访问移位寄存器。当接收到完整的字节时，字节会被传输到 I2CRCV 寄存器。接收时，I2CRSR 和 I2CRCV 会构成一个双重缓冲接收器。这可以允许在读取已接收数据的当前字节前开始接收下一个字节。

如果在软件从 I2CRCV 寄存器读取前一个字节前，模块接收到了另一个完整字节，将会发生接收器溢出，同时置位 I2COV (I2CCON<6>)。I2CRSR 中的字节则会丢失。

I2CADD 寄存器用于保存从器件地址。在 10 位模式中，所有的位都是相关的。在 7 位地址模式中，只有 I2CADD<6:0> 是相关的。A10M (I2CCON<10>) 指定从器件地址的期望模式。

寄存器 21-1: I²CCON: I²C™ 控制寄存器

高字节:							
R/W-0	U-0	R/W-0	R/W-1 HC	R/W-0	R/W-0	R/W-0	R/W-0
I2CEN	—	I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEN
bit 15							bit 8

低字节:								
R/W-0	R/W-0	R/W-0	R/W-0 HC	R/W-0 HC	R/W-0 HC	R/W-0 HC	R/W-0 HC	R/W-0 HC
GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	
bit 7								bit 0

- bit 15 **I2CEN:** I²C 使能位
1 = 使能 I²C 模块并将 SDA 和 SCL 引脚配置为串行端口引脚
0 = 禁止 I²C 模块。所有的 I²C 引脚都由端口功能控制。
- bit 14 **未用:** 读作 0
- bit 13 **I2CSIDL:** 在空闲模式停止位
1 = 当器件进入空闲模式时模块停止工作
0 = 模块在空闲模式继续工作
- bit 12 **SCLREL:** SCL 释放控制位 (作为 I²C 从器件工作时)
1 = 释放 SCL 时钟
0 = 保持 SCL 时钟为低电平 (时钟低电平时间延长)
如果 STREN = 1:
该位是可读写的 (即软件可以写入 0 来启动延长低电平时间或写入 1 来释放时钟)
在从器件发送开始时由硬件清零。
在从器件接收结束时由硬件清零。
如果 STREN = 0:
该位为读 / 置位 (即软件只能写入 1 来释放时钟)
在从器件发送开始时由硬件清零。
- bit 11 **IPMIEN:** 智能外设管理接口 (Intelligent Peripheral Management Interface, IPMI) 使能位
1 = 使能 IPMI 支持模式。已应答所有的地址
0 = 未使能 IPMI 模式
- bit 10 **A10M:** 10 位从器件地址位
1 = I2CADD 是一个 10 位从器件地址
0 = I2CADD 是一个 7 位从器件地址
- bit 9 **DISSLW:** 禁止转换率 (Slew Rate) 控制位
1 = 已禁止转换率控制
0 = 已使能转换率控制
- bit 8 **SMEN:** SMBus 输入电平位
1 = 使能符合 SMBus 规范的 I/O 引脚阈值
0 = 禁止 SMBus 输入阈值
- bit 7 **GCEN:** 广播呼叫使能位 (作为 I²C 从器件工作时)
1 = I2CRSR 接收到广播呼叫地址时允许中断
(已使能模块接收)
0 = 禁止广播呼叫地址
- bit 6 **STREN:** SCL 时钟低电平时间延长使能位 (作为 I²C 从器件工作时)
与 SCLREL 位一起使用。
1 = 使能软件或接收时钟低电平时间延长
0 = 禁止软件或接收时钟低电平时间延长
- bit 5 **ACKDT:** 应答数据位 (作为 I²C 主器件工作时。适用于主器件接收过程。)
当软件启动应答序列时将发送的值。
1 = 在应答时发送 NACK
0 = 在应答时发送 ACK

寄存器 21-1: I2CCON: I²C™ 控制寄存器 (续)

- bit 4 **ACKEN**: 应答序列使能位
 (当作为 I²C 主器件工作时。适用于主器件接收过程。)
 1 = 初始化 SDA 和 SCL 引脚上的应答序列并发送 ACKDT 数据位。
 在主器件应答序列结束时由硬件清零。
 0 = 应答序列不在进行中
- bit 3 **RCEN**: 接收使能位 (作为 I²C 主器件工作时。)
 1 = 使能 I²C 接收模式
 在主器件接收数据字节的第八位结束时由硬件清零。
 0 = 接收序列不在进行中
- bit 2 **PEN**: 停止条件使能位 (作为 I²C 主器件工作时)
 1 = 初始化 SDA 和 SCL 引脚上的停止条件。
 在主器件停止序列结束时由硬件清零。
 0 = 不产生停止条件
- bit 1 **RSEN**: 重复启动条件使能位 (作为 I²C 主器件工作时)
 1 = 初始化 SDA 和 SCL 引脚上的重复启动条件。
 在主器件重复启动序列结束时由硬件清零。
 0 = 不产生重复启动条件
- bit 0 **SEN**: 启动条件使能位 (作为 I²C 主器件工作时)
 1 = 初始化 SDA 和 SCL 引脚上的启动条件
 在主器件启动序列结束时由硬件清零。
 0 = 不产生启动条件

图注:		
R = 可读位	C = 可清零位	U = 未用位, 读作 0
W = 可写位	HS = 由硬件置位	S = 可设置位
HC = 由硬件清零	0 = 在 POR 时位清零	x = 在 POR 时位未知
1 = 在 POR 时位置位		

寄存器 21-2: I²CSTAT: I²C™ 状态寄存器

高字节:							
R-0	R-0	U-0	U-0	U-0	R/C-0	R-0	R-0
HS, HC	HS, HC				HS	HS, HC	HS, HC
ACKSTAT	TRSTAT	—	—	—	BCL	GCSTAT	ADD10
bit 15							bit 8

低字节:							
R/C-0	R/W-0	R-0	R/C-0	R/C-0	R-0	R-0	R-0
HS	HS	HS, HC	HS, HC	HS, HC	HS, HC	HS, HC	HS, HC
IWCOL	I2COV	D_A	P	S	R_W	RBF	TBF
bit 7							bit 0

- bit 15 ACKSTAT:** 应答状态位
(作为 I²C 主器件工作时。适用于主器件发送操作。)
1 = 接收到来自从器件的 NACK
0 = 接收到来自从器件的 ACK
在从器件应答结束时由硬件置位或清零。
- bit 14 TRSTAT:** 发送状态位
(作为 I²C 主器件工作时。适用于主器件发送操作。)
1 = 主器件正在发送过程中 (8 位 +ACK)
0 = 主器件不在发送过程中
在主器件发送开始时由硬件置位。
在从器件应答结束时由硬件清零。
- bit 13-11 未用:** 读作 0
- bit 10 BCL:** 主器件总线冲突检测位
1 = 在主器件工作期间检测到了总线冲突
0 = 未发生冲突
在检测到总线冲突时由硬件置位。
- bit 9 GCSTAT:** 广播呼叫状态位
1 = 收到了广播呼叫地址
0 = 未收到广播呼叫地址
当地址与广播呼叫地址匹配时由硬件置位。
在停止检测时由硬件清零。
- bit 8 ADD10:** 10 位地址状态位
1 = 10 位地址匹配
0 = 10 位地址不匹配
在匹配的 10 位地址的第二个字节匹配时由硬件置位。
在停止检测时由硬件清零。
- bit 7 IWCOL:** 写冲突检测位
1 = 因为 I²C 模块忙, 尝试写 I2CTRN 寄存器失败
0 = 未发生冲突
当总线忙, 写 I2CTRN 发生时由硬件置位 (由软件清零)。
- bit 6 I2COV:** 接收溢出标志位
1 = 当 I2CRCV 寄存器仍然保存有原先的字节时接收到了新字节
0 = 无溢出
尝试从 I2CRSR 传输到 I2CRCV 时由硬件置位 (由软件清零)。
- bit 5 D_A:** 数据 / 地址位 (作为 I²C 从器件工作时)
1 = 表示上次接收的字节是数据
0 = 表示上次接收的字节是器件地址
器件地址匹配时由硬件清零。
通过写 I2CTRN 或接收从器件字节由硬件置位。

寄存器 21-2: I2CSTAT: I²C™ 状态寄存器 (续)

- bit 4 **P:** 停止位
1 = 表明上次检测到了停止位
0 = 表示上次未检测到停止位
当检测到启动、重复启动或停止时由硬件置位或清零。
- bit 3 **S:** 启动位
1 = 表明最后检测到了启动 (或重复启动) 位
0 = 表示最后未检测到启动位
当检测到启动、重复启动或停止时由硬件置位或清零。
- bit 2 **D_W:** 读 / 写位信息 (作为 I²C 从器件工作时)
1 = 读——表示数据传输由从器件输出
0 = 写——表示数据传输输入到从器件
接收到 I²C 器件地址字节后由硬件置位或清零。
- bit 1 **RBF:** 接收缓冲器满状态位
1 = 接收完成, I2CRCV 满
0 = 接收没有完成, I2CRCV 空
当用接收到的字节写 I2CRCV 时由硬件置位。
当软件读 I2CRCV 时由硬件清零。
- bit 0 **TBF:** 发送缓冲器满状态位
1 = 正在发送, I2CTRN 满
0 = 发送完成, I2CTRN 空
当软件写 I2CTRN 时由硬件置位。
在数据发送完成时由硬件清零。

图注:		
R = 可读位	W = 可写位	C = 可清零位
HC = 由硬件清零	HS = 由硬件置位	U = 未用位, 读作 0
1 = POR 时位置位	0 = POR 时位清零	x = POR 时位未知

21.4 使能 I²C 操作

通过将 I2CEN (I2CCON<15>) 位置位可以使能模块。

I²C 模块完全实现了所有主器件和从器件功能。当模块被使能后，主器件和从器件功能同时有效，并且将会按软件或总线事件响应。

在最初使能时，模块将释放 SDA 和 SCL 引脚，将总线置于空闲状态。主器件功能将保持在空闲状态，除非软件将某个控制位置位来启动一个主器件事件。从器件功能将开始监视总线。如果从器件逻辑在总线上检测到启动事件和有效地址，从器件逻辑将开始从器件事务处理。

21.4.1 使能 I²C I/O

总线操作使用了两个引脚。一个是时钟 SCL 引脚，另一个是数据 SDA 引脚。当模块被使能后，假定没有其他具有更高优先级的模块拥有控制权，那么模块会控制 SDA 和 SCL 引脚。模块软件不需要关心这些引脚端口 I/O 的状态，模块会改写端口状态和方向。在初始化时，引脚为三态（释放）。

21.4.2 I²C 中断

I²C 模块会产生两个中断。一个中断被分配给主器件事件，另一个则被分配给从器件事件。这些中断将会把相应的中断标志位置位，并且如果相应的中断允许位置位，且对应中断优先级够高，将会中断软件执行过程。

主器件中断名为 MI2CIF，会在主器件报文事件完成时激活。

下列事件会产生 MI2CIF 中断。

- 启动条件
- 停止条件
- 数据传输字节已发送 / 已接收
- 应答发送
- 重复启动
- 检测到总线冲突事件

从器件中断称为 SI2CIF，在检测到地址为从器件地址的报文时被激活。

- 检测到有效器件地址（包括广播呼叫地址）
- 发送数据的请求
- 接收到数据

21.4.3 当作为总线主器件工作时设置波特率

当作为 I²C 主器件工作时，模块必须产生系统 SCL 时钟。通常，I²C 系统时钟被指定为 100 kHz、400 kHz 或 1 MHz。系统时钟速率被指定为最小 SCL 低电平时间加上最小 SCL 高电平时间。在大部分情况下，这是通过两个 TBRG 间隔定义的。

波特率发生器的重载值是 I2CBRG 寄存器的内容，如图 21-6 所示。当波特率发生器装入该值后，发生器递减计数直至 0，然后停止直到再次装入。发生器计数会在每个指令周期（Tcy）递减两次。波特率发生器在波特率重新启动时会自动重新加载。例如，如果发生了时钟同步，波特率发生器会在 SCL 引脚采样为高电平时重新加载。

注： I2CBRG 值不允许装载为 0x0。

要计算波特率发生器重载值，可使用以下公式。

公式 21-1:

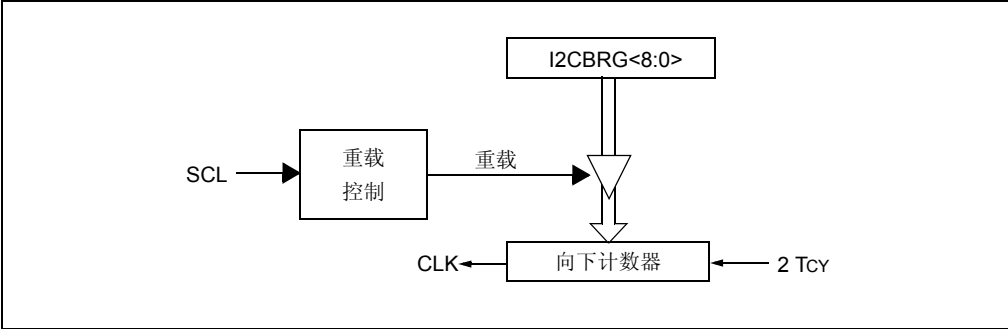
$$I2CBRG = \left(\frac{F_{CY}}{F_{SCL}} - \frac{F_{CY}}{1,111,111} \right) - 1$$

表 21-1: I²C™ 时钟速率

所需的系统 F _{SCL}	F _{CY}	I2CBRG 十进制	I2CBRG 十六进制	实际 F _{SCL}
100 kHz	30 MHz	272	0x110	100 kHz
100 kHz	20 MHz	181	0x0B5	100 kHz
100 kHz	1 MHz	8	0x008	101 kHz
400 kHz	10 MHz	15	0x00F	400 kHz
400 kHz	5 MHz	7	0x007	400 kHz
400 kHz	1 MHz	1	0x001	345 kHz**
1 MHz*	11 MHz	1	0x001	1 MHz*
1 MHz	1 MHz	0	0x000 (无效)	1 MHz

*Fcy=11 MHz 是为获得 FSCL=1 MHz 允许的最小输入时钟频率。
** 对于这个 Fcy 值，这是最接近 400 kHz 的值。

图 21-6: 波特率发生器框图



21.5 作为主器件在单主机环境下通信

系统中典型的 I²C 模块操作是使用 I²C 与 I²C 外设（如 I²C 串行存储器）通信。在 I²C 系统中，主器件控制总线上所有数据通信的时序。在此例中，dsPIC30F 及其 I²C 模块是系统中的惟一主器件。作为惟一的主器件，它要负责产生 SCL 时钟并控制报文协议。

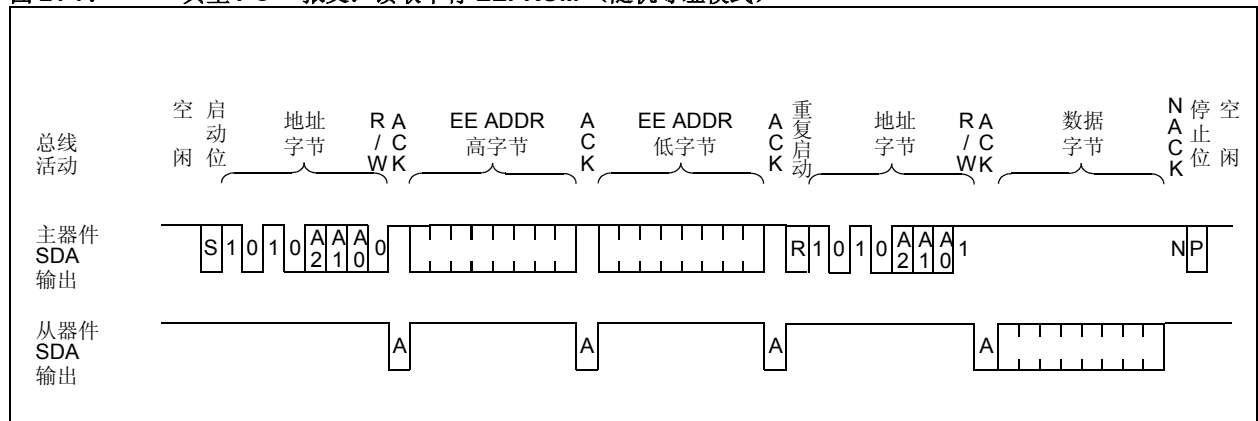
在 I²C 模块中，模块控制 I²C 报文协议的各个部分，但是，控制协议中组件的时序以构成完整的报文是软件的任务。

例如，单主机环境中的一种典型操作可能是从 I²C 串行 EEPROM 读取一个字节。图 21-7 描述了此示例报文。

要完成此报文，软件将通过以下步骤控制时序。

1. 在 SDA 和 SCL 上发出一个启动信号。
2. 发送一个带有写指示的 I²C 器件地址字节到从器件。
3. 等待并验证从器件的应答。
4. 发送串行存储器地址高字节到从器件。
5. 等待并验证从器件的应答。
6. 发送串行存储器地址低字节到从器件。
7. 等待并验证从器件的应答。
8. 在 SDA 和 SCL 上发出一个重复启动信号。
9. 将带有读指示的器件地址字节发送到从器件。
10. 等待并验证从器件的应答。
11. 使能主器件接收以接收串行存储器数据。
12. 在接收的数据字节结束时产生 ACK 或 NACK 条件。
13. 在 SDA 和 SCL 上产生停止条件。

图 21-7: 典型 I²C™ 报文：读取串行 EEPROM（随机寻址模式）



I²C 模块支持主控模式通信，包括启动和停止信号发生、数据字节发送、数据字节接收、应答发生和波特率发生器。

通常，软件会写一个控制寄存器以开始一个特定步骤，然后等待一个中断或查询状态以等待传输完成。

后续各节将详细说明这些操作。

21.5.1 产生启动总线事件

要开始启动事件，软件要将启动使能位 SEN (I2CCON<0>) 置位。在将启动位置位前，软件可以检查 P (I2CSTAT<4>) 状态位以确保总线处于空闲状态。

图 21-8 显示了启动条件的时序。

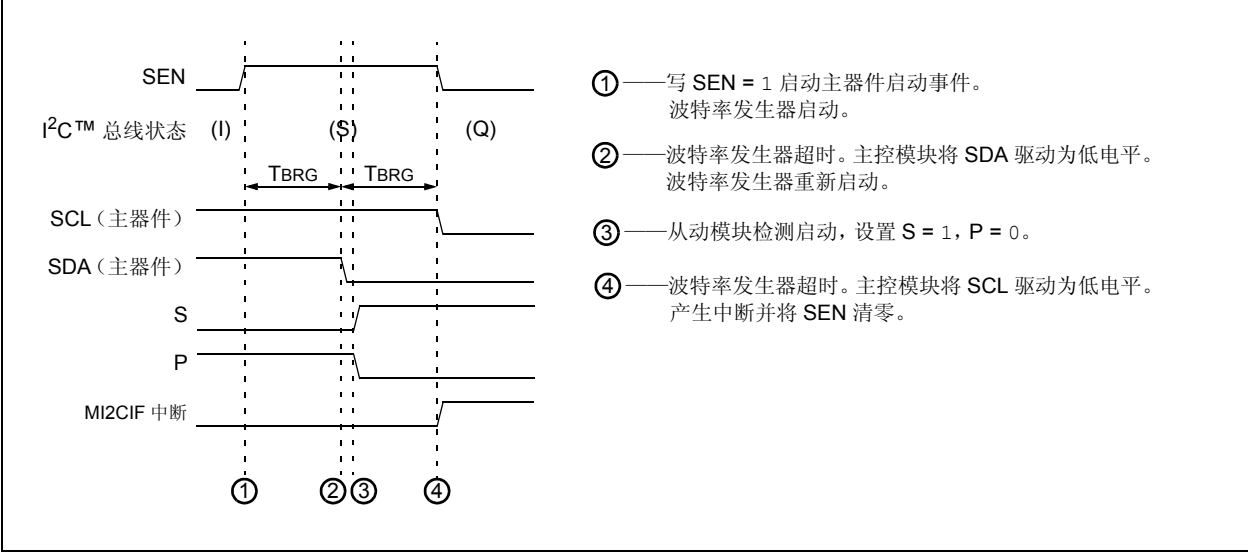
- 从器件逻辑会检测启动条件，将 S 位 (I2CSTAT<3>) 置位并将 P 位 (I2CSTAT<4>) 清零。
- SEN 位会在启动条件结束时自动清零。
- 在启动条件完成时会产生 MI2CIF 中断。
- 在启动条件后，SDA 线和 SCL 线会保持在低电平状态 (Q 状态)。

21.5.1.1 IWCOL 状态标志

在启动序列进行过程中，如果软件写 I2CTRN，则 IWCOL 被置位，同时发送缓冲器内容被忽略。

注： 由于不允许事件排队，在启动条件结束之前，不能对 I2CCON 的低 5 位进行写操作。

图 21-8: 主器件启动时序图



21.5.2 发送数据到从器件

发送数据字节（7 位器件地址字节或 10 位地址的第二个字节）只需通过将适当的值写到 I2CTRN 寄存器来实现。装载此寄存器将会开始以下过程：

- 软件将待发送数据字节装入 I2CTRN。
- 写 I2CTRN 将缓冲器满标志位 TBF（I2CSTAT<0>）置位。
- 数据字节从 SDA 引脚移出，直到发送完所有 8 位。每个地址 / 数据位都将在 SCL 的下降沿后移出到 SDA 引脚上。
- 在第九个 SCL 时钟，模块会将从器件移入 ACK 位并将其值写入 ACKSTAT 位（I2CCON<15>）。
- 模块在第九个 SCL 时钟周期结束时会产生 MI2CIF 中断。

注意模块不会产生或验证数据字节。字节的内容和使用取决于软件所保持的报文协议的状态。

21.5.2.1 将 7 位地址发送到从器件

发送 7 位器件地址涉及向从器件发送 1 个字节。7 位地址字节必须包含 7 位的 I²C 器件地址和一个 R/W 位，该位定义报文是写入从器件（主器件发送，从器件接收）还是由从器件读取（从器件发送，主器件接收）。

21.5.2.2 将 10 位地址发送到从器件

发送 10 位器件地址涉及向从器件发送 2 个字节。第一个字节包含 5 个为 10 位寻址模式保留的 I²C 器件地址位，以及 10 位地址中的 2 位。因为从器件必须接收下一个字节（包含 10 位地址余下的 8 位），第一个字节中的 R/W 必须是“0”，以表明主器件发送，从器件接收。如果报文数据也被定向到从器件，主器件可以继续发送数据。但是，如果主器件希望得到一个来自从器件的应答，R/W 位设为“1”的重新启动序列将把报文的 R/W 状态修改为读取从器件。

21.5.2.3 接收来自从器件的应答

在第 8 个 SCL 时钟的下降沿，TBF 位被清零，主器件将 SDA 引脚拉为高电平，以允许从器件发出一个应答响应。随后主器件会产生第九个 SCL 时钟。

这使被寻址的从器件在发生地址匹配或是数据被正确接收时，将在第 9 位时间以一个 $\overline{\text{ACK}}$ 位作为响应。从器件在识别出其器件地址（包括广播呼叫地址）或正确接收数据后，会发送一个应答信号。

$\overline{\text{ACK}}$ 的状态会在第九个 SCL 时钟的下降沿被写入应答状态位 ACKSTAT（I2CSTAT<15>）。在第九个 SCL 时钟后，模块会产生 MI2CIF 中断并进入空闲状态，直到下一个数据字节被装入 I2CTRN。

21.5.2.4 ACKSTAT 状态标志

当从器件发送了应答（ $\overline{\text{ACK}} = 0$ ）后，ACKSTAT 位（I2CCON<15>）被清零；而当从器件不应答时（ $\overline{\text{ACK}} = 1$ ），该位置位。

21.5.2.5 TBF 状态标志

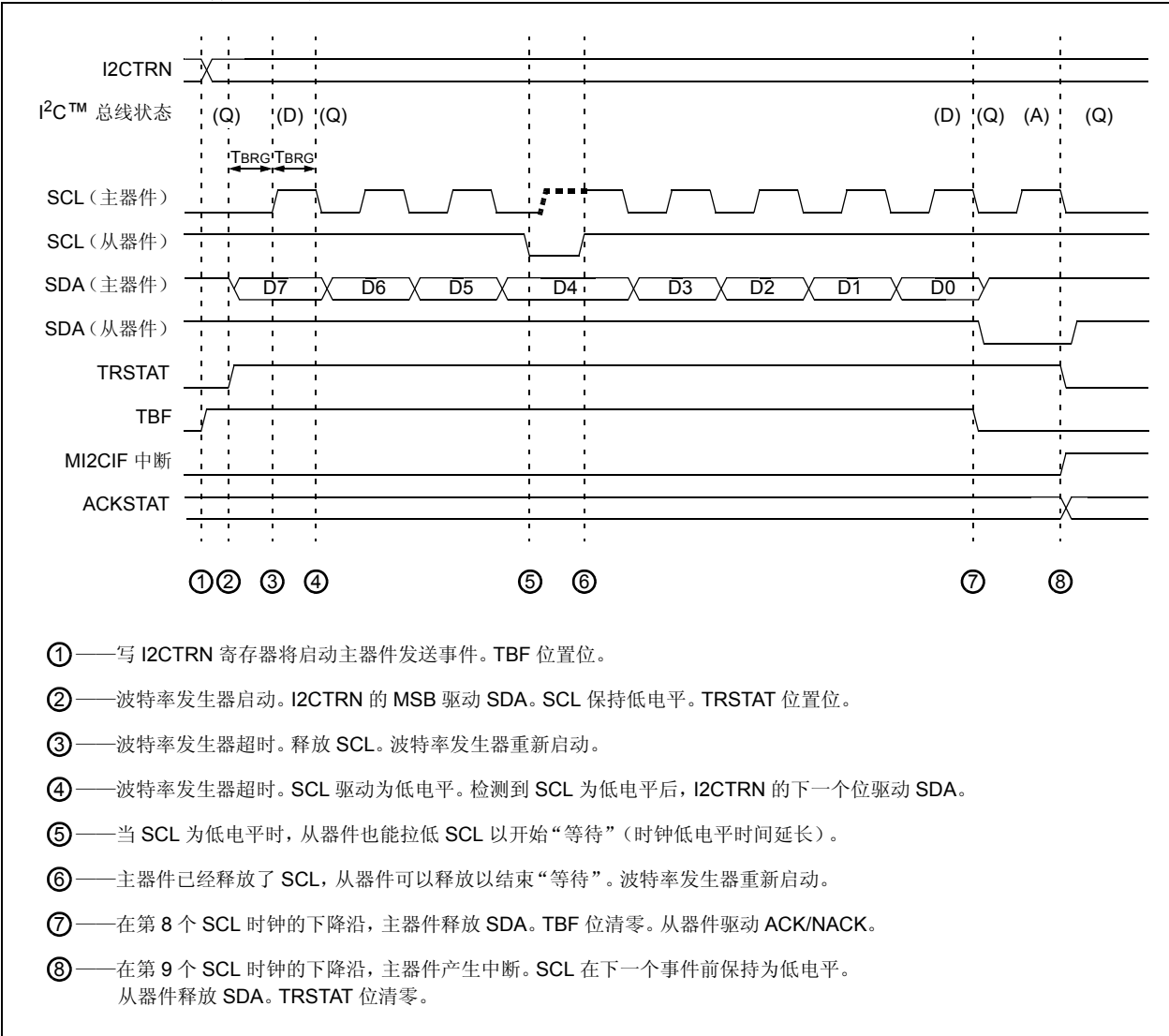
在发送时，当 CPU 写 I2CTRN 后，TBF 位（I2CSTAT<0>）将置位；当所有的 8 个位都被移出后，该位将清零。

21.5.2.6 IWCOL 状态标志

当发送已经进行时（即模块仍在移出一个数据字节），如果用软件写 I2CTRN，则 IWCOL 置位，缓冲器内容被忽略。IWCOL 必须用软件清零。

注： 由于不允许事件排队，在发送条件结束之前，不能对 I2CCON 的低 5 位进行写操作。

图 21-9： 主器件发送时序图



21.5.3 接收来自从器件的数据

将接收使能位 RCEN (I2CCON<3>) 置 1, 使能主器件接收来自从器件的数据。

注: 在尝试将 RCEN 位置 1 前, I2CCON 的低 5 位必须为 0。这确保了主控逻辑处于无效状态。

主控逻辑开始产生时钟, 并在 SCL 的每次下降沿出现之前, 采样 SDA 线并将数据移入 I2CRSR。在第 8 个 SCL 时钟脉冲的下降沿出现之后:

- RCEN 位自动清零。
- I2CRSR 的内容传输到 I2CRCV。
- RBF 标志位置 1。
- 模块产生 MI2CIF 中断。

当 CPU 读缓冲器时, RBF 标志位自动清零。软件可以处理数据然后产生应答序列。

21.5.3.1 RBF 状态标志

接收数据时, 当器件地址或数据字节被从 I2CRSR 装入 I2CRCV 时, RBF 位置 1。当软件读 I2CRCV 寄存器时, RBF 位清零。

21.5.3.2 I2COV 状态标志

如果当 RBF 位保持置 1 且前一个字节保持在 I2CRCV 寄存器中时, I2CRSR 接收到了另一个字节, 那么 I2COV 位置 1 并且 I2CRSR 中的数据将会丢失。

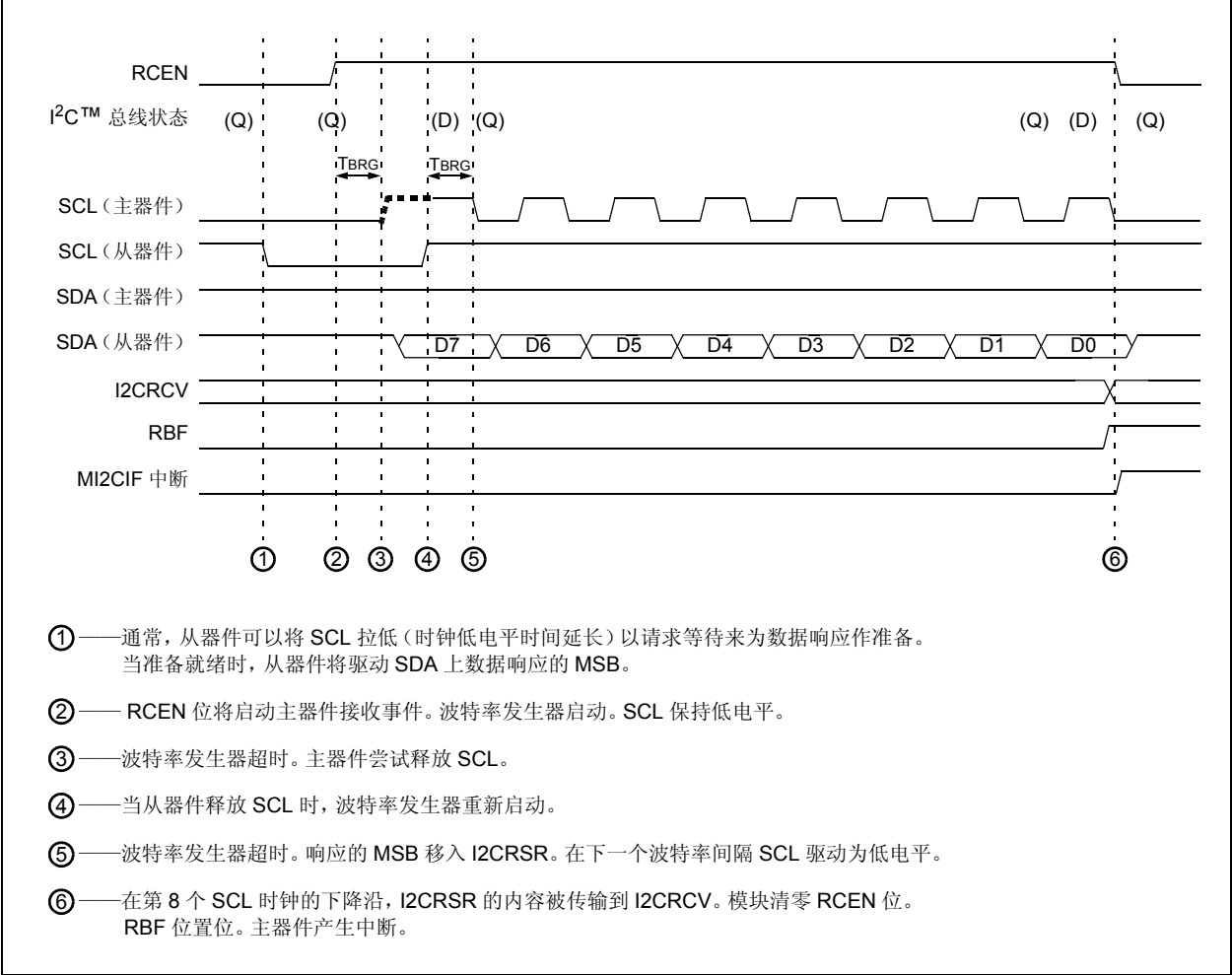
让 I2COV 保持置 1, 并不会阻止继续接收。如果通过读 I2CRCV 将 RBF 清零, 而且 I2CRSR 接收了另一个字节, 该字节将被传输到 I2CRCV。

21.5.3.3 IWCOL 状态标志

如果软件在接收已经在进行时 (即 I2CRSR 仍在移入数据字节时) 写 I2CTRN, 则 IWCOL 位置 1 且缓冲器内容被忽略。

注: 由于不允许事件排队, 在数据接收条件结束之前禁止对 I2CCON 的低 5 位进行写操作。

图 21-10: 主器件接收时序图



21.5.4 应答产生

将应答序列使能位 ACKEN (I2CCON<4>) 置 1, 使能主器件应答序列的产生。

注: 在尝试将 ACKEN 位置 1 前, I2CCON 的低 5 位必须为 “0” (主控逻辑无效)。

图 21-11 所示为 ACK 序列, 图 21-12 所示为 NACK 序列。应答数据位 ACKDT (I2CCON<5>) 用于指定 ACK 或 NACK。

在两个波特率周期后:

- ACKEN 位自动清零。
- 模块产生 MI2CIF 中断。

21.5.4.1 IWCOL 状态标志

如果软件在应答序列已经在进行时写 I2CTRN, 则 IWCOL 置 1 且缓冲器内容被忽略。

注: 由于不允许事件排队, 在应答条件结束之前, 禁止对 I2CCON 的低 5 位进行写操作。

图 21-11: 主器件应答 (ACK) 时序图

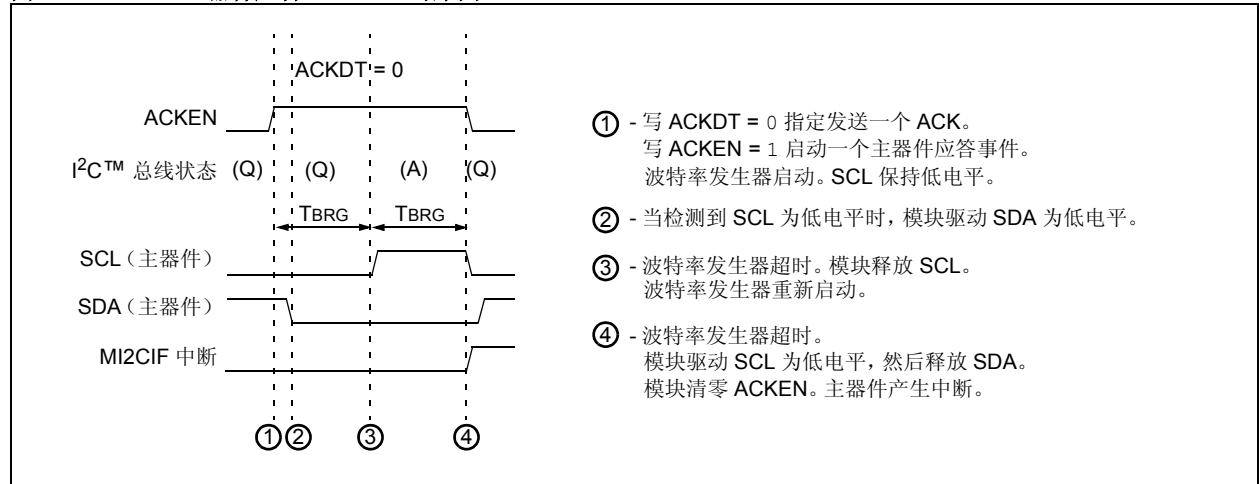
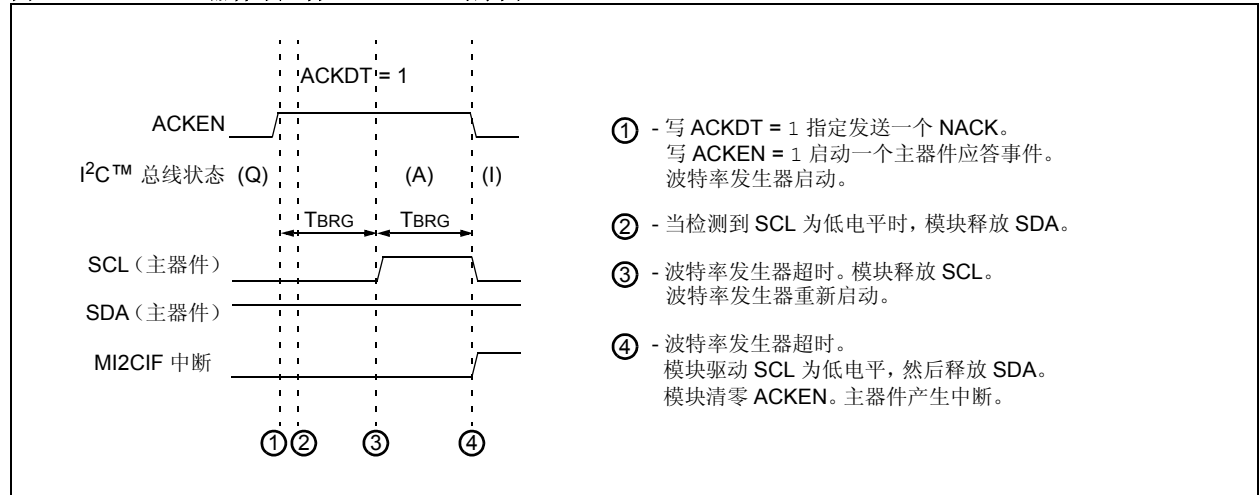


图 21-12: 主器件不应答 (NACK) 时序图



21.5.5 产生停止总线事件

将停止序列使能位 PEN (I2CCON<2>) 置 1，使能主器件停止序列的产生。

注： 在尝试将 PEN 位置 1 前，I2CCON 的低 5 位必须为 “0” (主控逻辑无效)。

当 PEN 位置 1 时，主器件产生停止序列，如图 21-13 所示。

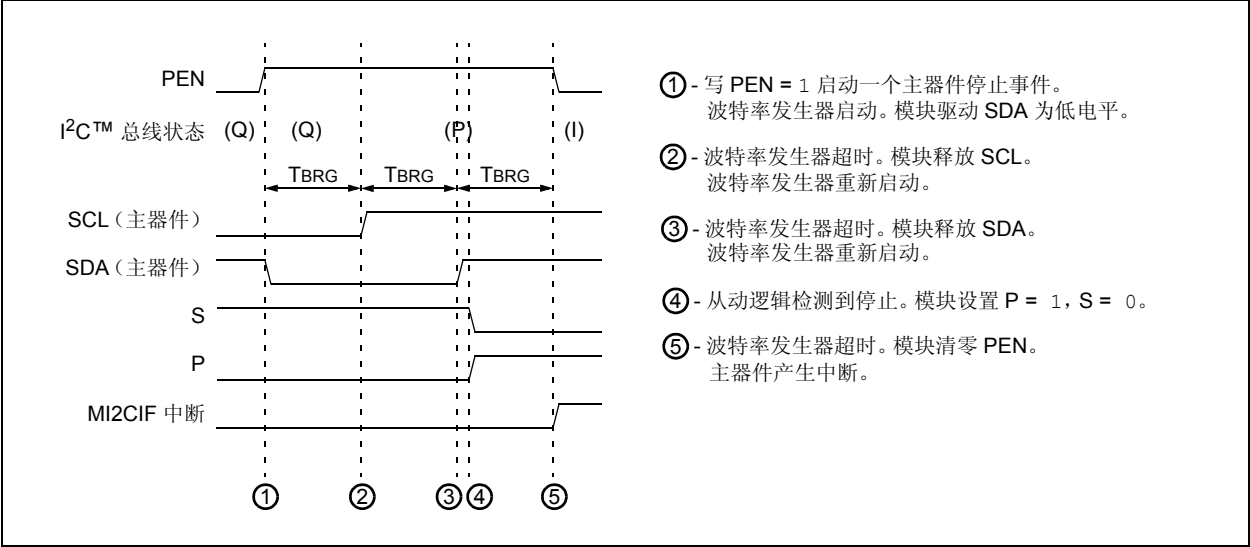
- 从器件检测到停止条件，将 P 位 (I2CSTAT<4>) 置 1 并清零 S 位 (I2CSTAT<3>)。
- PEN 位自动清零。
- 模块产生 MI2CIF 中断。

21.5.5.1 IWCOL 状态标志

如果软件在停止序列已经在进行时写 I2CTRN，则 IWCOL 位置 1 且缓冲器内容被忽略。

注： 由于不允许事件排队，在停止条件结束之前，禁止对 I2CCON 的低 5 位进行写操作。

图 21-13: 主器件停止时序图



21.5.6 产生重复启动总线事件

将重复启动序列使能位 RSEN (I2CCON<1>) 置 1, 使能主器件重复启动序列的产生 (参见图 21-14)。

注: 在尝试将 RSEN 位置 1 前, I2CCON 的低 5 位必须为 0 (主控逻辑无效)。

为了产生重复启动条件, 软件将 RSEN 位 (I2CCON<1>) 置 1。模块将 SCL 引脚置为低电平。当模块采样到 SCL 引脚为低电平时, 模块会将 SDA 引脚释放一个波特率发生器计数周期 (TBRG)。当波特率发生器超时, 如果模块采样到 SDA 为高电平, 模块会将 SCL 引脚拉高。当模块采样到 SCL 引脚为高电平时, 波特率发生器重新装载并开始计数。SDA 和 SCL 采样为高电平的时间必须为一个 TBRG。接下来, 当 SCL 为高电平时, 将 SDA 引脚拉低一个 TBRG 的时间。

下面是重复启动序列:

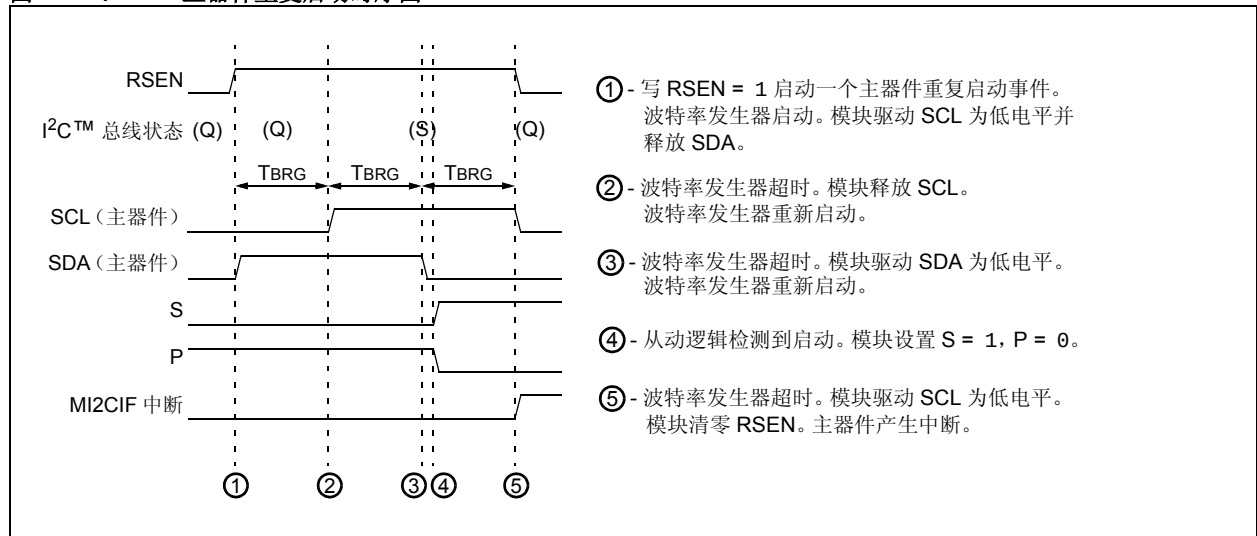
- 从器件检测到启动条件, 将 S 位 (I2CSTAT<3>) 置 1 并清零 P 位 (I2CSTAT<4>)。
- RSEN 位自动清零。
- 模块产生 MI2CIF 中断。

21.5.6.1 IWCOL 状态标志

如果软件在重复启动序列正在进行时写 I2CTRN, 则 IWCOL 置 1 且缓冲器内容被忽略。

注: 由于不允许事件排队, 在重复启动条件结束前, 禁止对 I2CCON 的低 5 位进行写操作。

图 21-14: 主器件重复启动时序图



21.5.7 建立完整的主器件报文

如在第 21.5 节的开头所述，软件负责用正确的报文协议建立报文。模块控制 I²C 报文协议的各个部分，但是，控制协议组件的时序以建立完整的报文是软件的任务。

当使用该模块时，软件可使用查询或中断的方法。给出的示例使用的是中断。

在报文传输过程中，软件可以将 SEN、RSEN、PEN、RCEN 和 ACKEN 位（I2CCON 寄存器的 5 个最低有效位）和 TRSTAT 位用作“状态”标志。例如，表 21-2 显示了与总线状态相关的一些示例状态编号。

表 21-2: 主器件报文协议状态

示例 状态编号	I2CCON<4:0>	TRSTAT (I2CSTAT<14>)	状态
0	00000	0	总线空闲或等待
1	00001	n/a	发送启动事件
2	00000	1	主器件发送
3	00010	n/a	发送重复启动事件
4	00100	n/a	发送停止事件
5	01000	n/a	主器件接收
6	10000	n/a	主器件应答

注： 示例状态编号仅供参考。用户软件可以根据需要分配状态编号。

软件可以通过发出启动命令开始发送报文。软件将记录与启动对应的状态编号。

当每个事件完成并产生中断时，中断处理程序可检查状态编号。所以，对于启动状态，中断处理程序将应答启动序列的执行，然后启动一个主器件发送事件来发送 I²C 器件地址，改变状态编号以与主器件发送相对应。

在下次中断时，中断处理程序会再次检查状态，以确定刚刚完成了主器件发送。中断处理程序将确认数据成功完成发送，然后根据报文的内容转至下一个事件。

以这种方式，每次中断时，中断处理程序将通过报文协议进行，直到发送完整个报文。

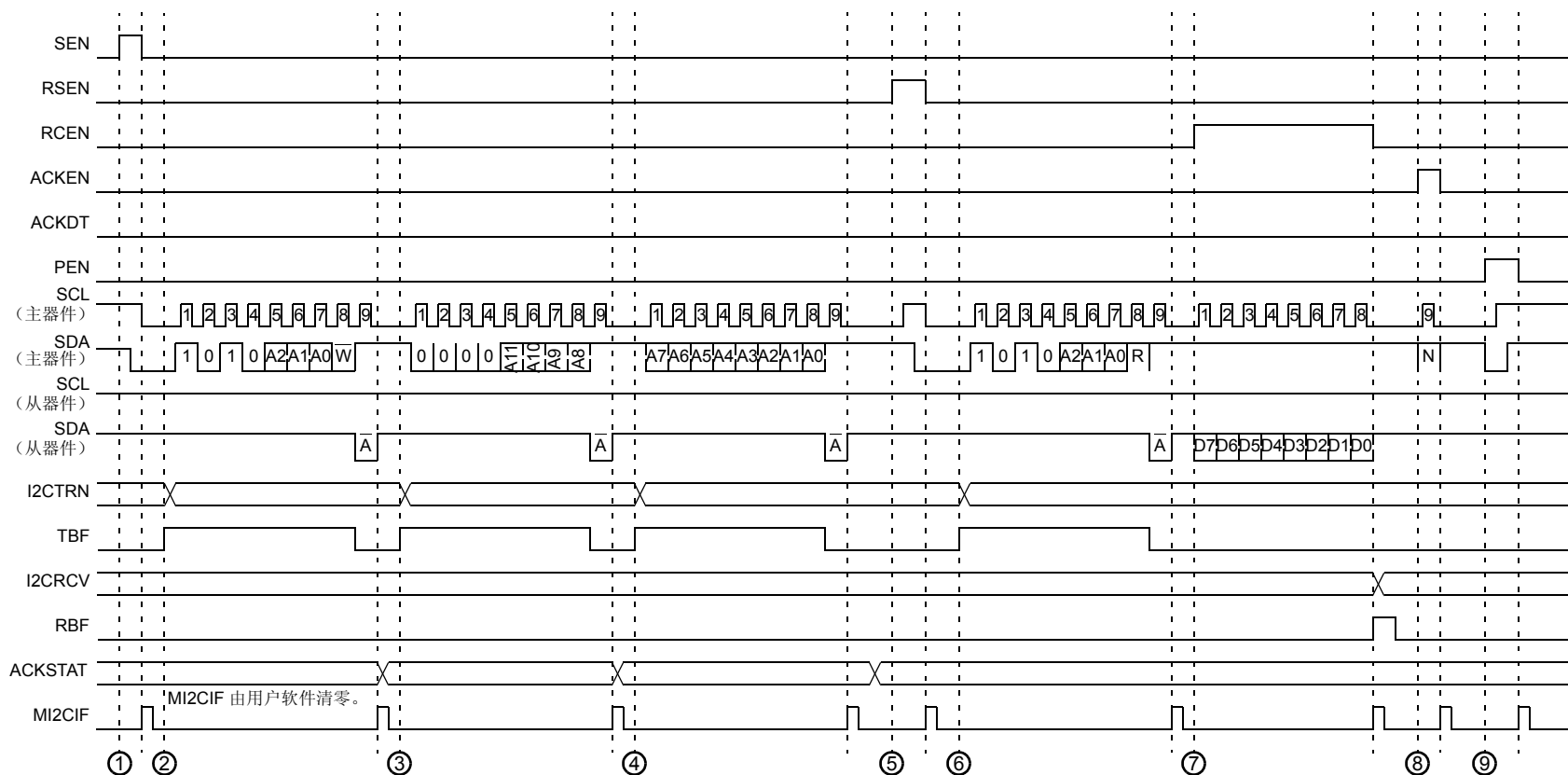
图 21-15 为图 21-17 中的同一个报文序列提供了更详细的检查。

图 21-16 显示了一些使用 7 位寻址格式的报文的简单示例。

图 21-17 所示为发送数据至从器件的 10 位地址格式报文示例。

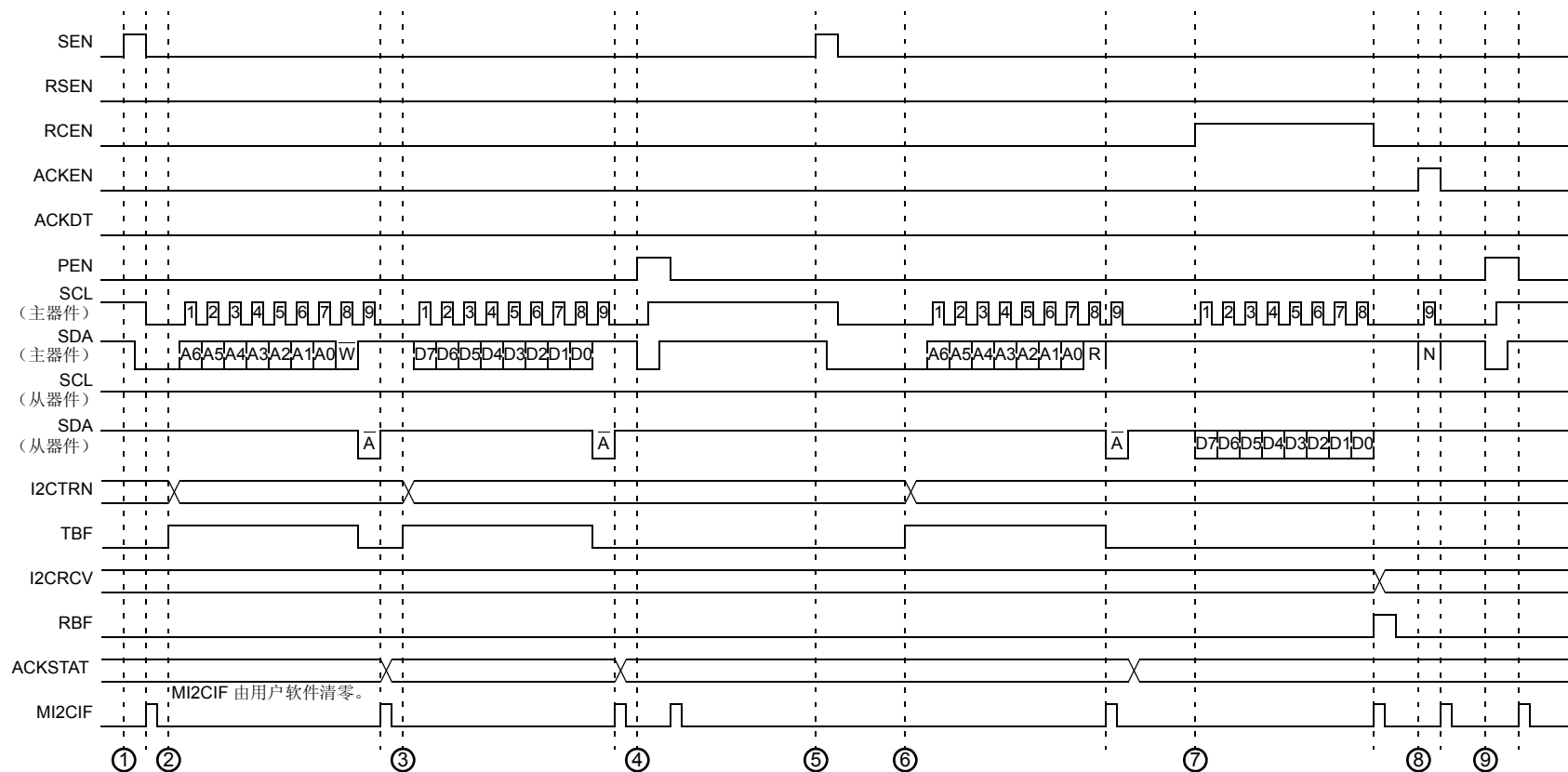
图 21-18 所示为接收来自从器件的数据的 10 位地址格式报文示例。

图 21-15: 主器件报文（典型的 I²C™ 报文：读串行 EEPROM）



- ①——将 SEN 位置 1，启动一个启动事件。
- ②——写 I2CTRN 寄存器启动主器件发送。该数据是串行 EE 器件的地址字节，其 R/W 位清零表示写。
- ③——写 I2CTRN 寄存器启动主器件发送。该数据是 EE 数据地址的第一个字节。
- ④——写 I2CTRN 寄存器启动主器件发送。该数据是 EE 数据地址的第二个字节。
- ⑤——将 RSEN 位置 1，启动重复启动事件。
- ⑥——写 I2CTRN 寄存器启动主器件发送。该数据是重新发送的串行 EE 器件的地址字节，但其 R/W 位置 1 表示读。
- ⑦——将 RCEN 位置 1，启动主器件接收。中断时，软件读 I2CRCV 寄存器，这样做将清零 RBF 标志位。
- ⑧——将 ACKEN 位置 1，启动应答事件。设置 ACKDT = 1 以发送 NACK。
- ⑨——将 PEN 位置 1，启动主器件停止事件。

图 21-16: 主器件报文 (7 位地址: 发送和接收)



- ① — 将 SEN 位置 1, 启动一个启动事件。
- ② — 写 I2CTR_N 寄存器启动主器件发送。该数据是地址字节, 其 R/W 位清零。
- ③ — 写 I2CTR_N 寄存器启动主器件发送。该数据是报文字节。
- ④ — 将 PEN 位置 1, 启动主器件停止事件。
- ⑤ — 将 SEN 位置 1, 启动一个启动事件。

- ⑥ — 写 I2CTR_N 寄存器启动主器件发送。该数据是地址字节, 其 R/W 位置 1。
- ⑦ — 将 RCEN 位置 1, 启动主器件接收。
- ⑧ — 将 ACKEN 位置 1, 启动应答事件。设置 ACKDT = 1 以发送 NACK。
- ⑨ — 将 PEN 位置 1, 启动主器件停止事件。

图 21-17: 主器件报文 (10 位发送)

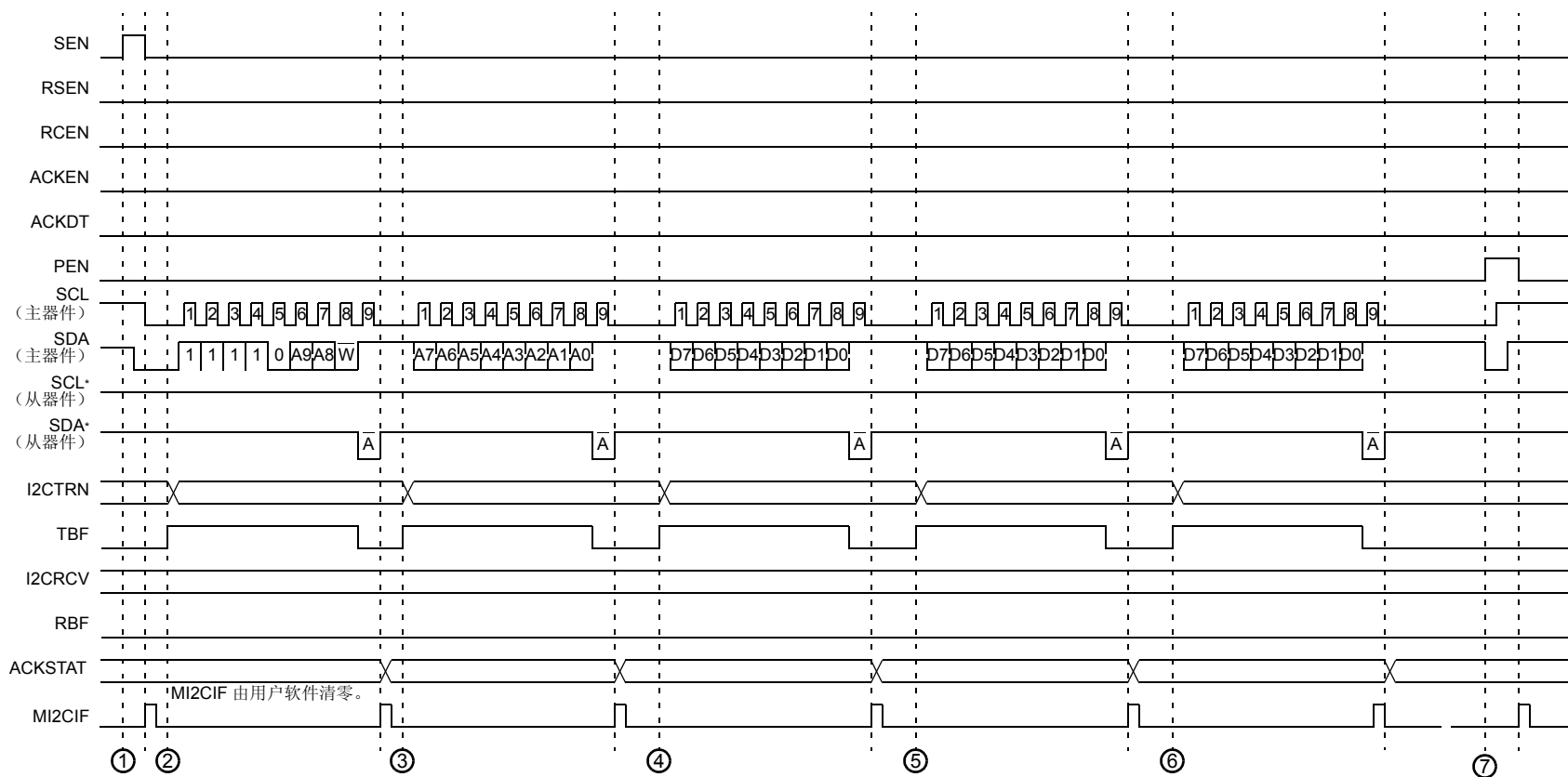
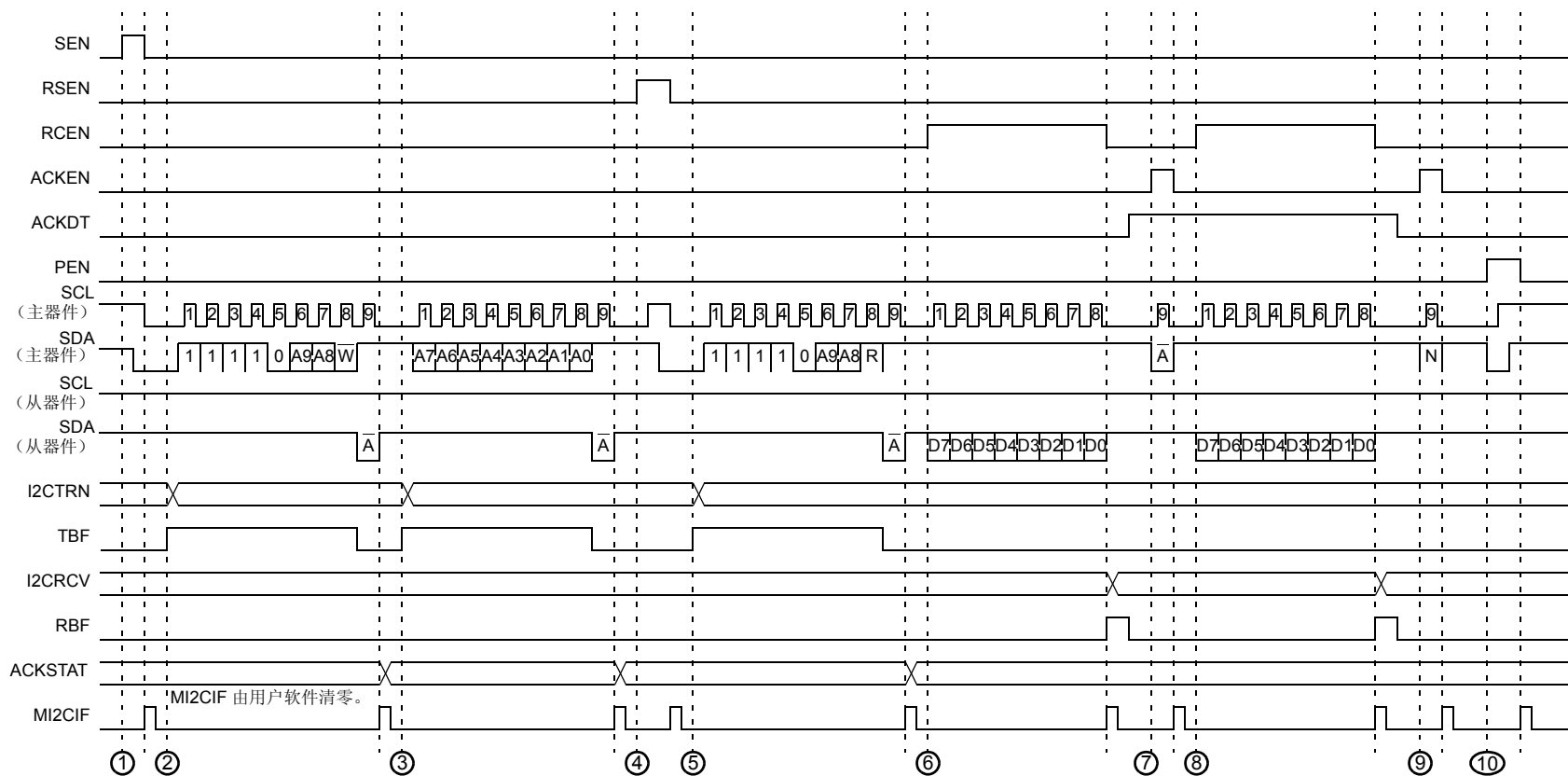


图 21-18: 主器件报文 (10 位接收)



- ① —将 SEN 位置 1, 启动一个启动事件。
- ② —写 I2CTRN 寄存器启动主器件发送。该数据是地址的第一个字节, 其 R/W 位清零。
- ③ —写 I2CTRN 寄存器启动主器件发送。该数据是地址的第二个字节。
- ④ —将 RSEN 位置 1, 启动一个主器件重新启动事件。
- ⑤ —写 I2CTRN 寄存器启动主器件发送。该数据是重发的地址的第一个字节, 其 R/W 位置 1。

- ⑥ —将 RCEN 位置 1, 启动主器件接收。中断时, 软件读 I2CRCV 寄存器, 这样做将清零 RBF 标志位。
- ⑦ —将 ACKEN 位置 1, 启动应答事件。设置 ACKDT = 0 以发送 ACK。
- ⑧ —将 RCEN 位置 1, 启动主器件接收。
- ⑨ —将 ACKEN 位置 1, 启动应答事件。设置 ACKDT = 1 以发送 NACK。
- ⑩ —将 PEN 位置 1, 启动主器件停止事件。

21.6 作为主器件在多主机环境下通信

I²C 协议允许一个以上的主器件挂载在系统总线上。请记住，主器件可以启动报文事务处理并为总线产生时钟，协议有几种方法，可用以解决一个以上的主器件尝试控制总线的情形。时钟同步确保了多个节点能够同步它们的 SCL 时钟以形成 SCL 线上的一个共同的时钟。如果一个以上的节点尝试报文传输，总线仲裁能确保有且仅有一个节点成功完成该报文。其他节点总线仲裁失败并留下一个总线冲突。

21.6.1 多主机工作

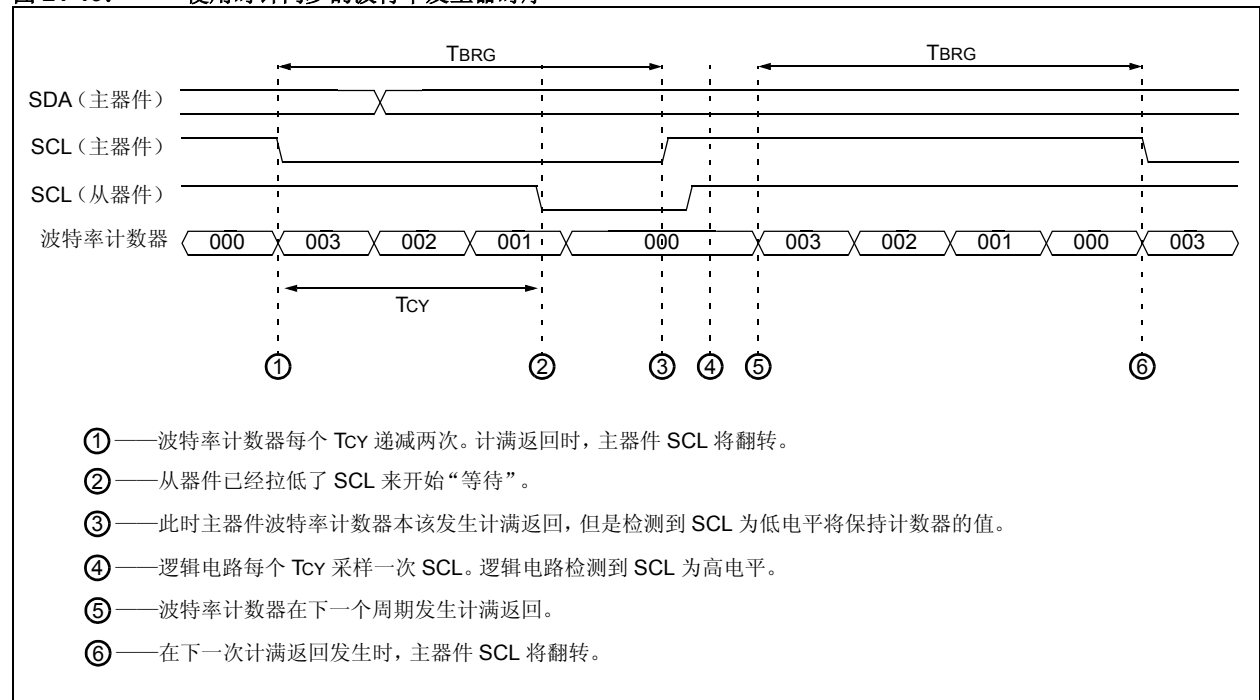
主控模块没有特别的设置来使能多主机工作。该模块始终执行时钟同步和总线仲裁。如果该模块在单主机模式下使用，将只在主器件和从器件之间发生时钟同步，而总线仲裁将不会发生。

21.6.2 主器件时钟同步

在多主机系统中，不同的主器件可能会有不同的波特率。时钟同步将确保当这些主器件尝试争用总线时，它们的时钟将是相同的。

当主器件拉高 SCL 引脚（SCL 试图悬空为高电平）时，发生时钟同步。当释放 SCL 引脚时，波特率发生器（BRG）将暂停计数直到 SCL 引脚被实际采样到高电平为止。当 SCL 引脚被采样到高电平时，波特率发生器重新装载 I2CBRG<8:0> 的内容并开始计数。这可以保证在发生外部器件将时钟保持为低的事件时，SCL 始终至少保持一个 BRG 计满返回计数周期的高电平，如图 21-19 所示。

图 21-19: 使用时钟同步的波特率发生器时序



21.6.3 总线仲裁与总线冲突

总线仲裁支持多主机系统的工作。

SDA 线的“线与”特性允许仲裁。当第一个主器件通过让 SDA 悬空为高电平在 SDA 上输出一个“1”，而且与此同时，第二个主器件通过把 SDA 拉为低电平在 SDA 上输出一个“0”时，发生仲裁。SDA 信号将变低。在这种情况下，第二个主器件赢得了总线仲裁。第一个主器件总线仲裁失败，从而发生了一个总线冲突。

对于第一个主器件，期望 SDA 上的数据是“1”，但是 SDA 上采样到的数据是“0”。这就是总线冲突的定义。

第一个主器件将置位总线冲突位 BCL (I2CSTAT<10>) 并产生主器件中断。该主控模块会将 I²C 端口复位到空闲状态。

在多主机工作中，必须监视 SDA 线来进行仲裁，查看信号电平是否为期望的输出电平。检查由主控模块执行，并将结果放入 BCL 位。

仲裁可能失败的状态是：

- 启动条件
- 重复启动条件
- 地址、数据或应答位
- 停止条件

21.6.4 检测总线冲突和重新发送报文

当发生总线冲突时，该模块置位 BCL 位并产生一个主器件中断。如果在字节发送过程中发生总线冲突，则发送停止，TBF 标志位清零，而且 SDA 和 SCL 引脚被拉高。如果在启动、重复启动、停止或应答条件的执行过程中发生总线冲突，则这种条件被中止，I2CCON 寄存器中的对应控制位清零，并且 SDA 和 SCL 线被拉高。

软件预备在主器件事件完成后发生中断。软件可以检查 BCL 位以确定主器件事件是否成功完成或是否发生了冲突。如果发生了冲突，软件必须中止发送待发报文的其余部分，并准备在总线返回空闲状态后，启动条件发生时开始重新发送整个报文序列。软件可以监视 S 和 P 位以等待总线空闲。当软件处理主器件中断服务程序并且 I²C 总线空闲时，软件可通过发出启动条件恢复通信。

21.6.5 启动条件期间的总线冲突

在发出启动命令前，软件应该使用 S 和 P 状态位验证总线的空闲状态。两个主器件可能尝试在同一个时间点启动报文。通常，这两个主器件将同步它们的时钟并持续争取报文传输直到一个主器件仲裁失败为止。然而，某些条件会引起在启动时发生总线冲突。在这种情况下，在启动位发送期间仲裁失败的主器件会产生一个总线冲突中断。

21.6.6 重复启动条件期间的总线冲突

如果两个主器件在整个地址字节未发生冲突，当一个主器件尝试发出重复启动条件而另一个主器件正在发送数据时，可能产生总线冲突。在这种情况下，产生重复启动条件的主器件将仲裁失败并产生一个总线冲突中断。

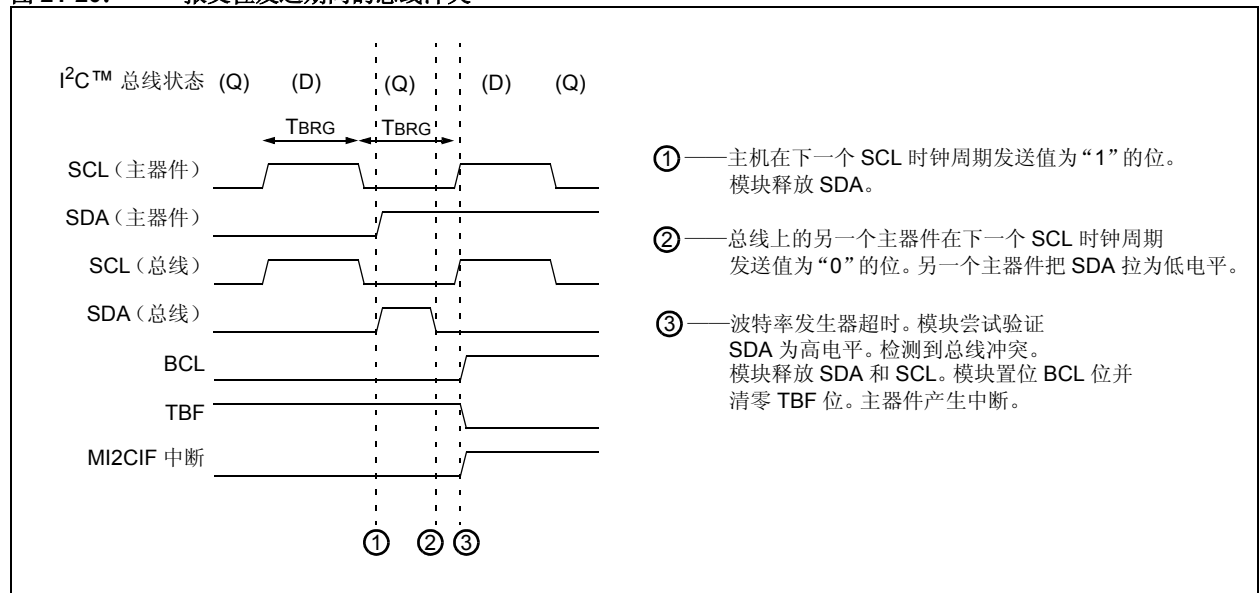
21.6.7 报文位发送期间的总线冲突

当主器件尝试发送器件地址字节、数据字节或应答位时，会发生最典型的数据冲突情况。

如果软件正在正确地检查总线状态，在启动条件发生时不太可能发生总线冲突。然而，因为另一个主器件可以在非常接近的时间，检查总线并开始它的启动条件，此时很有可能发生 SDA 仲裁并同步两个主器件的启动。在此条件下，两个主器件都将开始并持续发送它们的报文直到一个主器件在一个报文位上仲裁失败为止。记住 SCL 时钟同步会保持两个主器件同步直到一个主器件仲裁失败为止。

图 21-20 所示为报文位仲裁的一个示例。

图 21-20: 报文位发送期间的总线冲突



21.6.8 停止条件期间的总线冲突

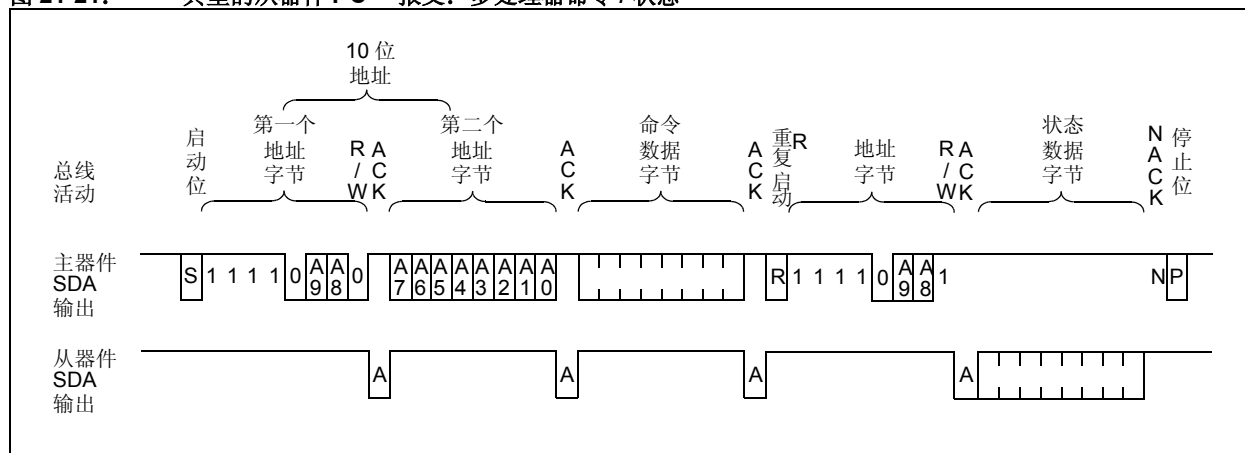
如果主器件软件失去了对 I²C 总线状态的跟踪，有些条件将导致总线冲突在停止条件期间发生。在这种情况下，产生停止条件的主器件将仲裁失败并产生一个总线冲突中断。

21.7 作为从器件通信

在有些系统中，尤其是在有多个处理器互相通信的系统中，dsPIC30F 器件可以作为从器件通信（参见图 21-21）。当该模块使能时，从器件模块有效。从器件不可以开始报文传输，它只能响应由主器件开始的报文序列。主器件请求来自特定从器件的响应，具体是哪个从器件由 I²C 协议中的器件地址字节决定。从动模块在由协议定义的适当时间应答主器件。

对于主器件模块，为应答的协议组件排序是软件的任务。但是，该从器件地址与软件为该从器件指定的地址何时匹配是由从器件检测的。

图 21-21: 典型的从器件 I²C™ 报文：多处理器命令 / 状态



启动条件发生之后，从器件模块将接收并检查器件地址。从器件可以指定 7 位地址或 10 位地址。当匹配器件地址时，该模块将产生一个中断以通知软件它的器件已被选定。根据由主器件发送的 R/W 位，从器件将接收或发送数据。如果从器件将要接收数据，从器件模块会自动产生应答 (ACK)，用接收到的值（当前在 I2CRSR 寄存器中）装载 I2CRCV 寄存器，并通过中断通知软件。如果从器件将要发送数据，软件必须装载 I2CTRN 寄存器。

21.7.1 采样接收的数据

在时钟 (SCL) 线的上升沿采样所有的输入位。

21.7.2 检测启动和停止条件

从动模块将在总线上检测启动和停止条件，并用 S 位 (I2CSTAT<3>) 和 P 位 (I2CSTAT<4>) 表示这些状态。当复位发生时或该模块被禁止时，启动 (S) 和停止 (P) 位清零。当检测到启动或重复启动事件时，S 位置 1 并且 P 位清零。当检测到停止事件后，P 位置 1 并且 S 位清零。

21.7.3 检测地址

一旦该模块被使能，从器件模块将等待启动条件发生。启动条件发生后，根据 A10M 位 (I2CCON<10>)，从器件将尝试检测 7 位或 10 位地址。从器件模块比较接收到的一个字节（对于 7 位地址格式）或接收到的两个字节（对于 10 位地址格式）。7 位地址还包含一个 R/W 位，该位指定该地址后数据传输的方向。如果 R/W = 0，则指定一个写操作，而且从器件将从主器件接收数据。如果 R/W = 1，则指定一个读操作，而且从器件会将数据发送给主器件。10 位地址包含一个 R/W 位，然而按照定义，总是有 R/W = 0，因为从器件必须接收 10 位地址的第二个字节。

表 21-3: I²C™ 模块支持的从地址

0x00	广播呼叫地址或启动字节
0x01—0x03	保留
0x04—0x77	有效 7 位地址
0x78—0x7b	有效 10 位地址（低 7 位）
0x7c—0x7f	保留

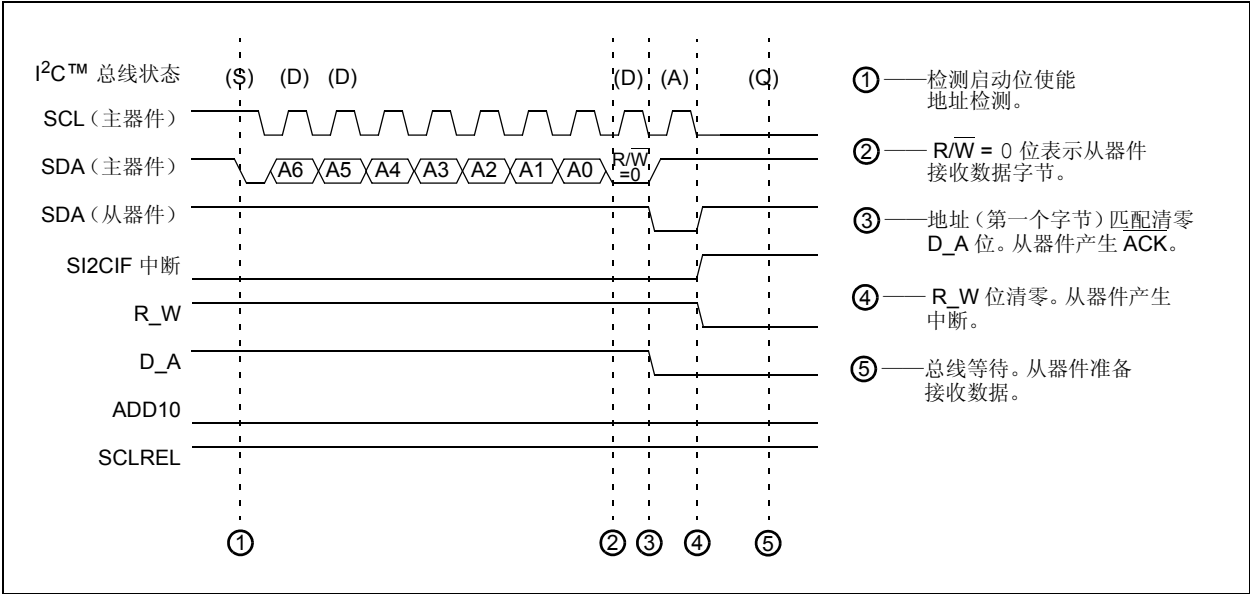
寄存器位字段的说明请参见《dsPIC30F 系列参考手册》(DS70046_CN)。

21.7.3.1 7 位地址和写从器件操作

启动条件发生后，该模块将 8 位移入 I2CRSR 寄存器（参见图 21-22）。寄存器 I2CRSR<7:1> 的值与 I2CADD<6:0> 寄存器的值作比较。在第八个时钟（SCL）脉冲的下降沿，比较器件地址。如果地址匹配，就会发生以下事件：

- 1. 产生一个 $\overline{\text{ACK}}$ 。
- 2. D_A 和 R_W 位被清零。
- 3. 模块在第九个 SCL 时钟的下降沿产生 SI2CIF 中断。
- 4. 模块将等待主器件发送数据。

图 21-22: 写从器件 7 位地址检测时序图



21.7.3.2 7 位地址和读从器件操作

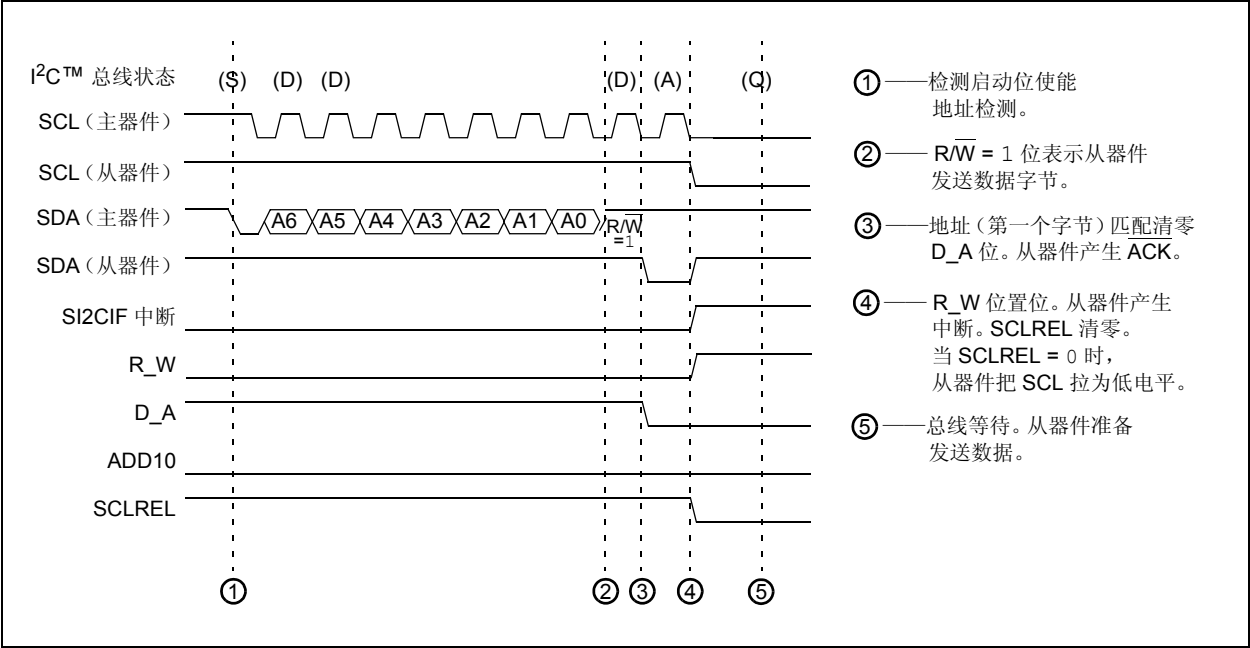
当在 7 位地址字节中通过设置 $\overline{R/W} = 1$ 指定读从器件操作时，检测器件地址的过程与写从器件操作类似（参见图 21-23）。如果地址匹配，就会发生以下事件：

- 1. 产生一个 $\overline{ACK}$ 。
- 2. D_A 位清零且 R_W 位置位。
- 3. 模块在第九个 SCL 时钟的下降沿产生 SI2CIF 中断。

因为此时希望从动模块以数据应答，必须暂停 I²C 总线的工作以允许软件准备响应。这在该模块清零 SCLREL 位时自动完成。SCLREL 为 0，从器件模块将拉低 SCL 时钟线，导致 I²C 总线上的等待。从器件模块和 I²C 总线将保持处于此状态，直到软件用响应数据写 I2CTRN 寄存器并置位 SCLREL 位为止。

注： 检测到读从器件地址后，SCLREL 将自动清零，而不管 STREN 位的状态如何。

图 21-23: 读从器件 7 位地址检测时序图



21.7.3.3 10 位地址

在 10 位地址模式下，从器件必须接收两个器件地址字节（参见图 21-24）。第一个地址字节的五个最高有效位（MSb）指定该地址是 10 位地址。该地址的 R/W 位必须指定为写，使得从器件可接收第二个地址字节。对于一个 10 位地址，第一个字节等于“11110 A9 A8 0”，其中 A9 和 A8 是该地址的两个 MSb。

启动条件发生后，该模块将 8 位移入 I2CRSR 寄存器。寄存器 I2CRSR<2:1> 的值与 I2CADD<9:8> 寄存器的值作比较。I2CRSR<7:3> 的值与“11110”作比较。在第八个时钟（SCL）脉冲的下降沿，比较器件地址。如果地址匹配，就会发生以下事件：

1. 产生一个 $\overline{\text{ACK}}$ 。
2. D_A 和 R_W 位被清零。
3. 模块在第九个 SCL 时钟的下降沿产生 SI2CIF 中断。

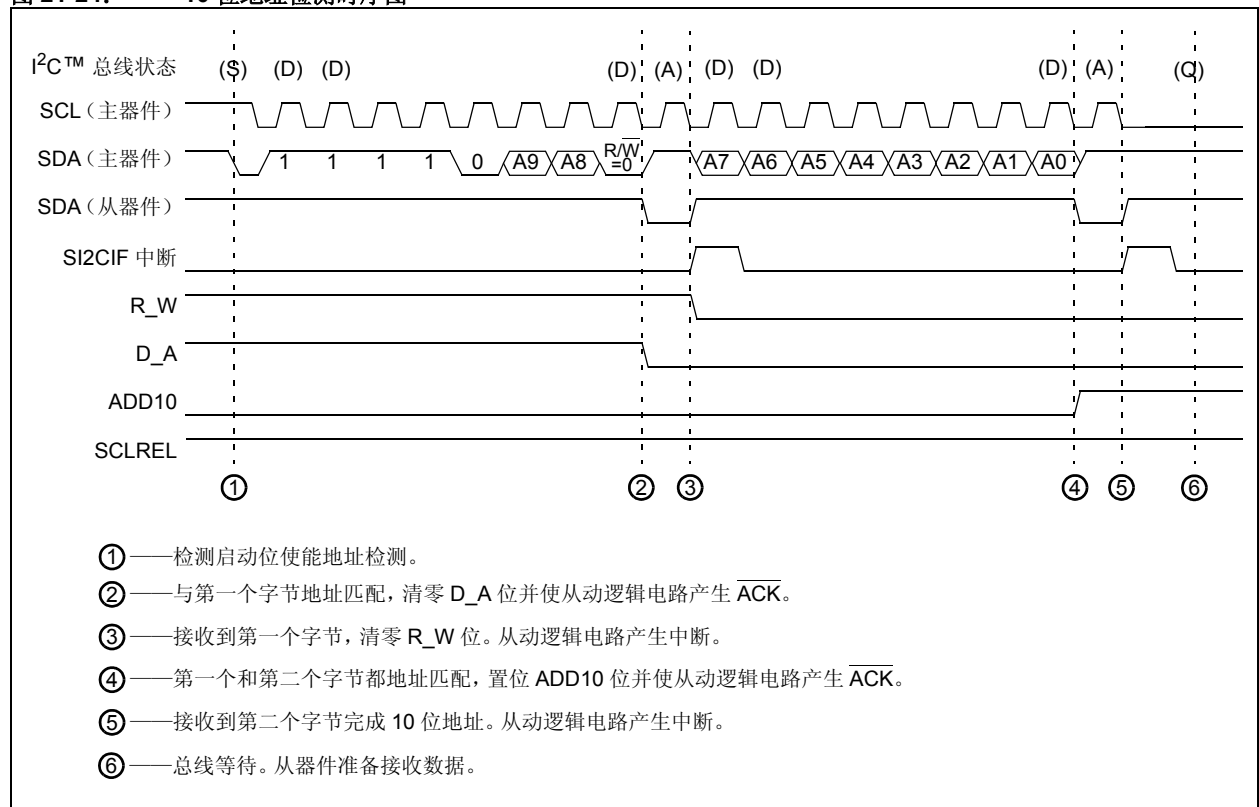
该模块在接收到 10 位地址的第一个字节之后的确会产生中断，但是此中断没什么作用。

该模块将继续接收第二个字节，并将其放入 I2CRSR。此时，I2CRSR<7:0> 与 I2CADD<7:0> 作比较。如果地址匹配，就会发生以下事件：

1. 产生一个 $\overline{\text{ACK}}$ 。
2. ADD10 位被置位。
3. 模块在第九个 SCL 时钟的下降沿产生 SI2CIF 中断。
4. 模块将等待主器件发送数据或开始一个重复启动条件。

注： 在 10 位模式下，重复启动条件发生后，从器件模块只匹配第一个 7 位地址“11110 A9 A8 0”。

图 21-24: 10 位地址检测时序图



21.7.3.4 广播呼叫操作

I²C 总线的寻址过程是启动条件后的第一个字节，通常确定主器件正在寻址哪个从器件。但广播呼叫地址例外，它能寻址所有器件。当使用这个地址时，所有被使能的器件都应该以应答信号作出响应。广播呼叫地址是由 I²C 协议为特定目的保留的八个地址之一。它由 R/W = 0 的全 0 位组成。广播呼叫操作总是写从器件操作。

当广播呼叫使能位 GCEN (I2CCON<7>) 被置位时 (参见图 21-25)，广播呼叫地址被识别。在检测到启动位后，将 8 位移入 I2CRSR 并将地址与 I2CADD 进行比较同时也与广播呼叫地址进行比较。

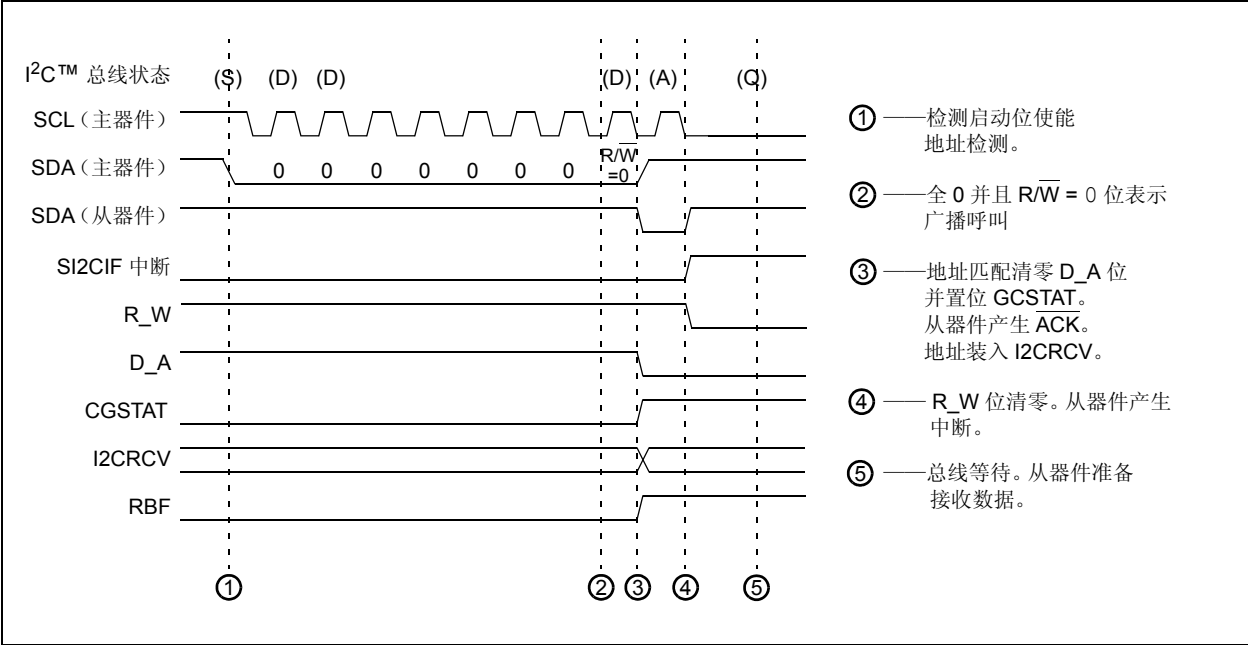
如果广播呼叫地址匹配，就会发生以下事件。

1. 产生一个 ACK。
2. 从模块使 GCSTAT 位 (I2CSTAT<9>) 置位。
3. D_A 和 R_W 位被清零。
4. 模块在第九个 SCL 时钟的下降沿产生 SI2CIF 中断。
5. I2CRSR 被传送到 I2CRCV 且 RBF 标志位被置位 (在第八个位传送的时候)。
6. 模块将等待主器件发送数据。

当响应中断时，通过读 GCSTAT 位的内容可以检测到中断的原因以确定器件地址是器件指定的还是广播呼叫地址。

注意广播呼叫地址是 7 位地址。如果 A10M 位置位，配置从器件模块为 10 位地址但 GCEN 位置位，从器件模块仍将继续检测 7 位广播呼叫地址。

图 21-25: 广播呼叫地址检测时序图 (GCEN = 1)

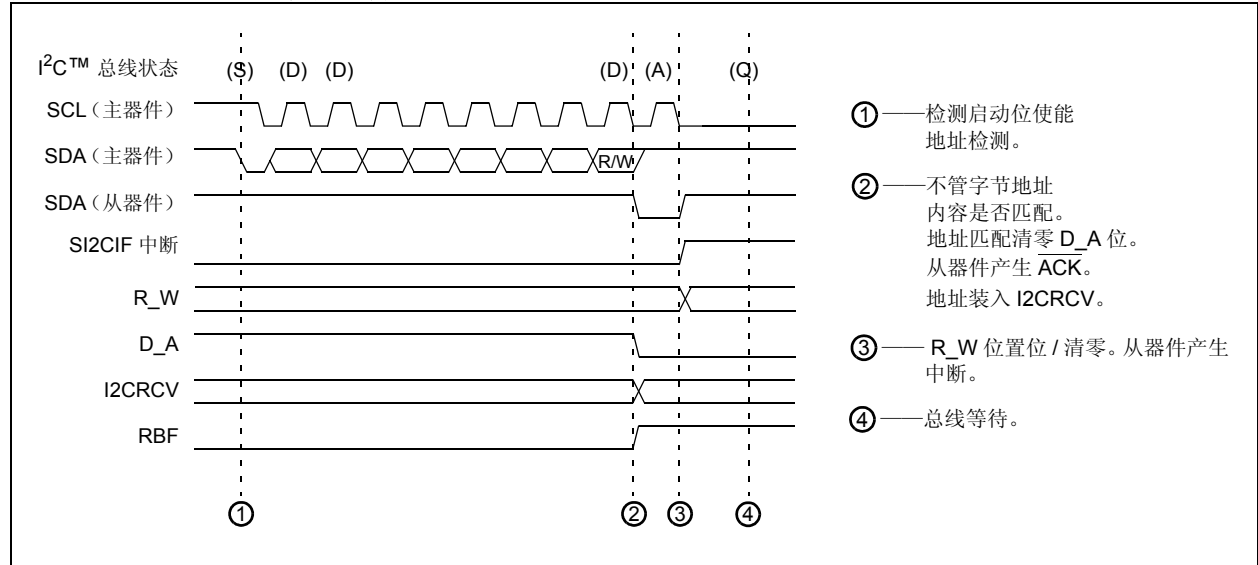


21.7.3.5 接收所有地址 (IPMI 操作)

某些 I²C 系统协议需要从器件响应总线上的所有报文。例如，IPMI（智能外设管理接口）总线使用 I²C 节点作为分布式网络中的报文中继点。要允许节点中继所有报文，从模块必须接收所有报文，不管器件地址是什么。

置位 IPMIEN 位 (I2CCON<11>) 使能此模式（参见图 21-26）。不管 I2CADD 寄存器和 A10M 位以及 GCEN 位的状态如何，所有地址都将被接收。

图 21-26: IPMI 地址检测时序图 (IPMIEN = 1)



21.7.3.6 当地址无效时

如果 7 位地址与 I2CADD<6:0> 的内容不匹配，从模块将回到空闲状态并忽略所有总线活动，直到停止条件以后。

如果 10 位地址与 I2CADD<9:8> 的内容不匹配，从模块将回到空闲状态并忽略所有总线活动，直到停止条件以后。

如果 10 位地址的第一个字节与 I2CADD<9:8> 的内容匹配，但是其第二个字节与 I2CADD<7:0> 不匹配，从模块将回到空闲状态并忽略所有总线活动，直到停止条件以后。

21.7.4 接收来自主器件的数据

当器件地址字节的 $\overline{R\_W}$ 位为零且发生了地址匹配， $\overline{R\_W}$ 位 (I2CSTAT<2>) 将清零。从动模块进入等待主器件发送数据的状态。在器件地址字节之后，数据字节的内容由系统协议定义并仅由从动模块接收。

从动模块将 8 位移入 I2CRSR 寄存器。在第八个时钟 (SCL) 的下降沿，发生以下事件：

1. 模块开始产生 $\overline{ACK}$ 或 NACK。
2. RBF 位置 1 表示接收了数据。
3. I2CRSR 字节被转移到 I2CRCV 寄存器以由软件访问。
4. $\overline{D\_A}$ 位被置位。
5. 产生一个从中断。软件可以检查 I2CSTAT 寄存器的状态以确定事件的原因然后清零 SI2CIF 标志位。
6. 模块将等待下一个数据字节。

21.7.4.1 应答产生

通常情况下，从模块将通过在第九个 SCL 时钟发送 $\overline{\text{ACK}}$ 应答所有接收的字节。如果接收缓冲器溢出，从模块就不会产生该 ACK。在发生以下情况中的一种或两种时就表示溢出：

- 1. 在传输的报文被接收前，缓冲满位 RBF（I2CSTAT<1>）置位。
- 2. 在传输的报文被接收前，溢出位 I2COV（I2CSTAT<6>）置位。

表 21-4 通过给出 RBF 和 I2COV 位的状态，显示了数据传输字节被接收时发生了什么。如果在从动模块尝试发送到 I2CRCV 时 RBF 位已经置位了，这次传输不会发生但是会产生中断并置位 I2COV 位。如果 RBF 和 I2COV 都被置位，从模块的行为与此类似。阴影单元表示软件没有正确清零溢出条件的情况。

读 I2CRCV 清零 RBF 位。通过软件写 0 清零 I2COV。

表 21-4: 数据传输接收字节行为

数据字节接收的状态位		转移 I2CRSR→I2CRCV	产生 ACK	产生 SI2CIF 中断 (中断使能时 产生中断)	设置 RBF	设置 I2COV
RBF	I2COV					
0	0	是	是	是	是	不变
1	0	否	否	是	不变	是
1	1	否	否	是	不变	是
0	1	是	否	是	是	不变

注： 阴影单元表示软件没有正确清零溢出条件的状态。

21.7.4.2 从接收时的等待状态

当从模块接收数据字节时，主器件可以准备立即开始发送下一个字节。这允许软件控制从模块 9 个 SCL 时钟周期以处理前面接收的字节。如果时间还不够，从软件可能需要产生一个总线等待周期。

STREN 位（I2CCON<6>）使能在从接收时产生总线等待。当在一个接收字节的第九个 SCL 时钟下降沿时 STREN = 1，从动模块清零 SCLREL 位。清零 SCLREL 位导致从动模块将 SCL 线拉为低电平，开始一个等待时间。主从模块的 SCL 时钟将同步，如第 21.6.2 节“主器件时钟同步”所示。

当软件准备好恢复接收时，软件置位 SCLREL。这将导致从动模块释放 SCL 线并且主控模块恢复产生时钟信号。

21.7.4.3 从器件接收报文示例

接收从报文是一个相当自动的处理过程。处理从协议的软件使用从中断来使事件同步。

当从器件检测到有效地址时，相关的中断将通知软件准备接收报文。在接收数据的时候，随着每个数据字节传送到 I2CRCV 寄存器，中断会通知软件卸载缓冲器。

图 21-27 所示为一个简单的报文接收。如果是 7 位地址报文，只需要为地址字节产生一个中断。然后，每四个数据字节产生一次中断。

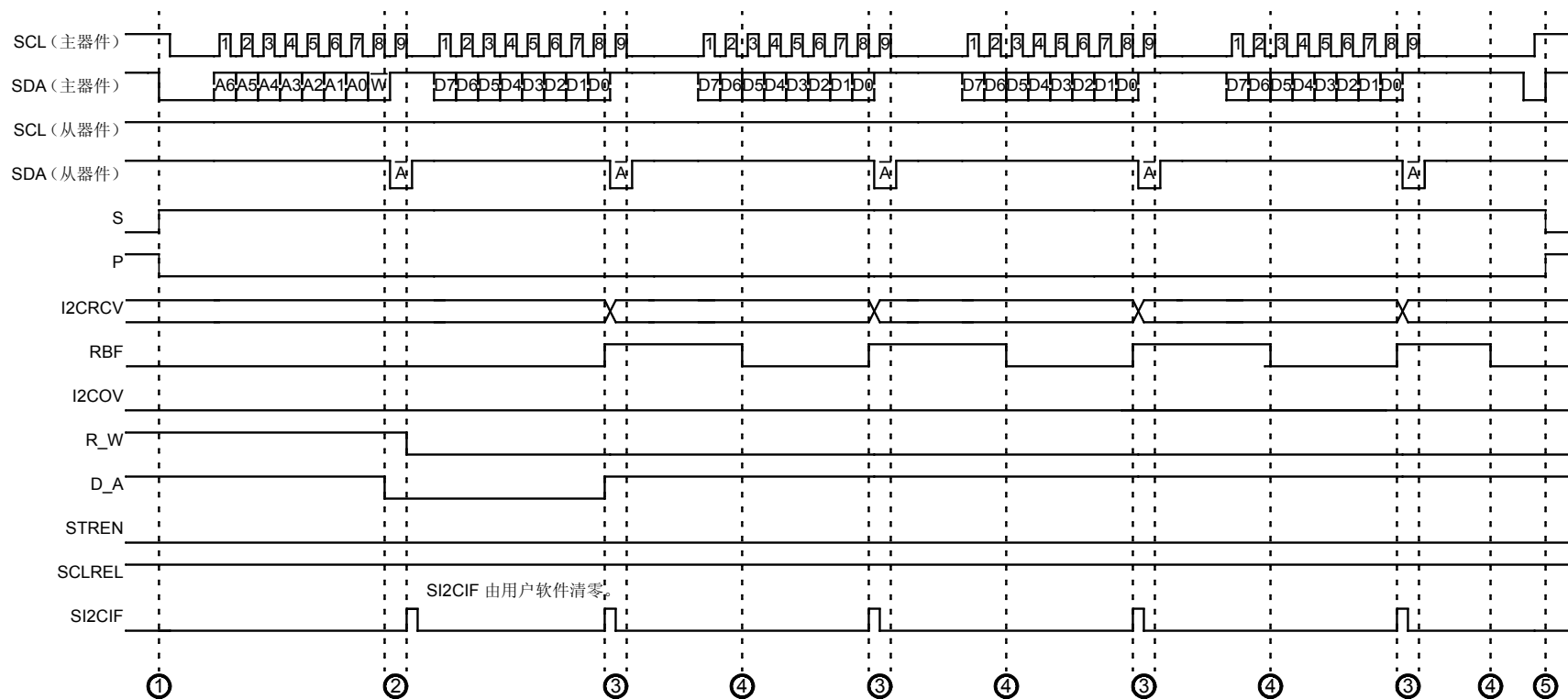
在中断时，软件可能监视 RBF、D_A 和 R_W 位以确定接收字节的状态。

图 21-28 所示为使用 10 位地址的类似报文。在这种情况下，地址需要两个字节。

图 21-29 所示为软件不响应接收字节和缓冲器溢出的情况。在接收第二个字节时，模块将自动 NACK 主器件发送。通常情况下，这会导致主器件再次发送前面的字节。I2COV 位表示缓冲器溢出。I2CRCV 缓冲器将保留第一个字节的内容。在接收第三个字节时，缓冲器仍然为满，模块将再次 NACK 主模块。在此之后，软件最后读缓冲器。读缓冲器将清零 RBF 位，但 I2COV 位保持为置位。软件必须清零 I2COV 位。下一个接收的字节将被移到 I2CRCV 缓冲器并且模块将以 ACK 作出响应。

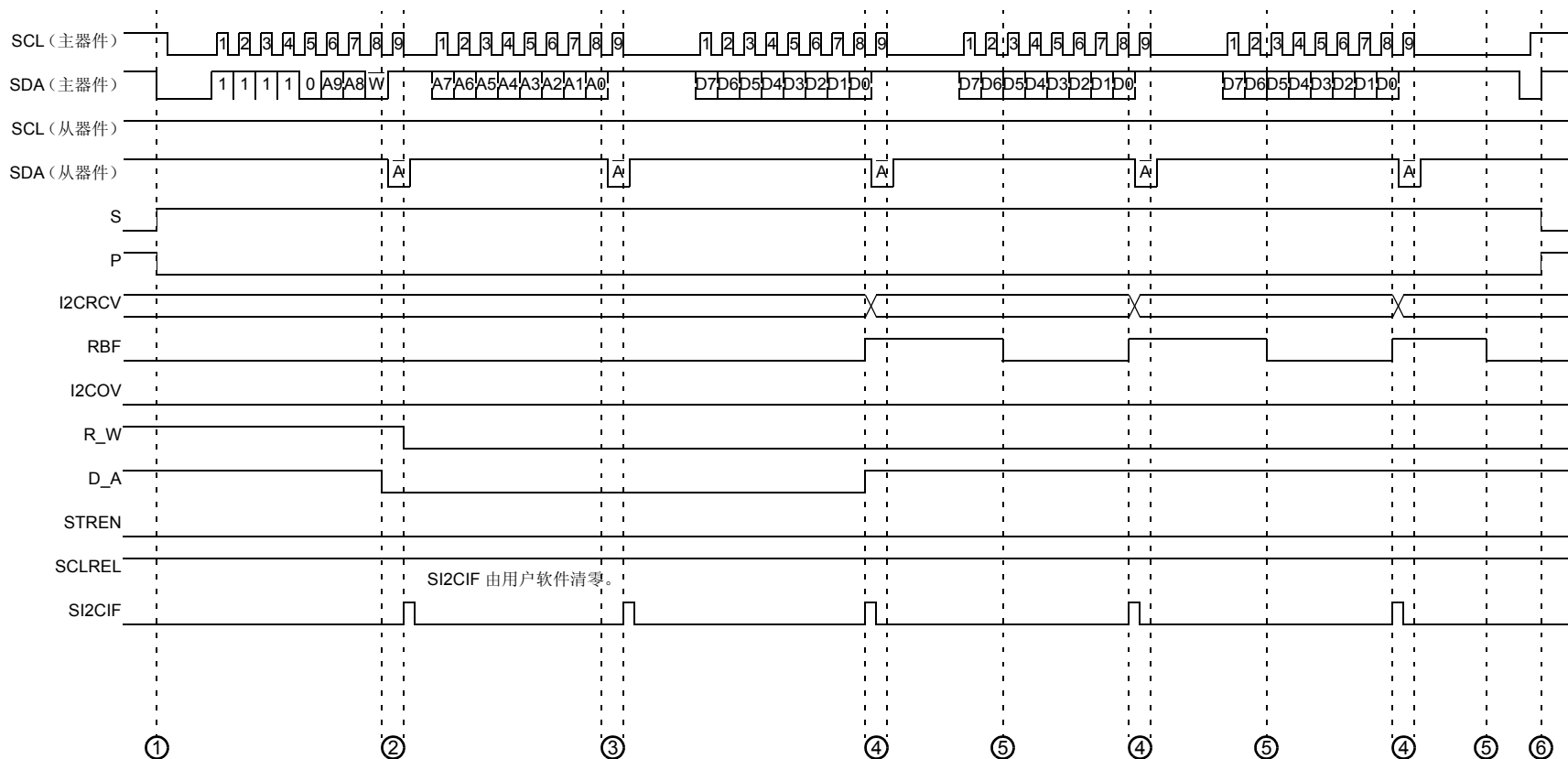
图 21-30 突出说明了接收数据时的时钟延长。注意在前面的示例中，STREN = 0 将禁止接收报文时的时钟延长。在此示例中，软件置位 STREN 以使能时钟延长。当 STREN = 1 时，模块将自动在接收每个数据字节后进行时钟延长，允许软件有更多的时间从缓冲器移走数据。注意如果在第九个时钟的下降沿 RBF = 1，模块将自动清零 SCLREL 位并将 SCL 总线拉为低电平。如第二个接收数据字节所示，如果软件可以在第九个时钟的下降沿之前读取缓冲器并清零 RBF，就不会产生时钟延长。软件也可以在任何时候暂停总线。通过清零 SCLREL 位，模块将在检测到总线 SCL 为低后拉低其 SCL 线。SCL 线将保持为低电平，暂停总线上的事务处理直到 SCLREL 位被置位。

图 21-27: 从报文 (写数据到从器件: 7 位地址, 地址匹配, A10M = 0, GCEN = 0, IPMIEN = 0)



- ① —— 从器件识别启动事件, S 和 P 位相应地置位 / 清零。
- ② —— 从器件接收地址字节。地址匹配。从器件应答并产生中断。地址字节移到 I2CRCV 寄存器且必须被用户软件读取以阻止缓冲器溢出。
- ③ —— 下一个接收的字节是报文数据。字节转移到 I2CRCV 寄存器, 置位 RBF。从器件产生中断。从器件应答接收。
- ④ —— 软件读 I2CRCV 寄存器。RBF 位清零。
- ⑤ —— 从器件识别停止事件, S 和 P 位相应地置位 / 清零。

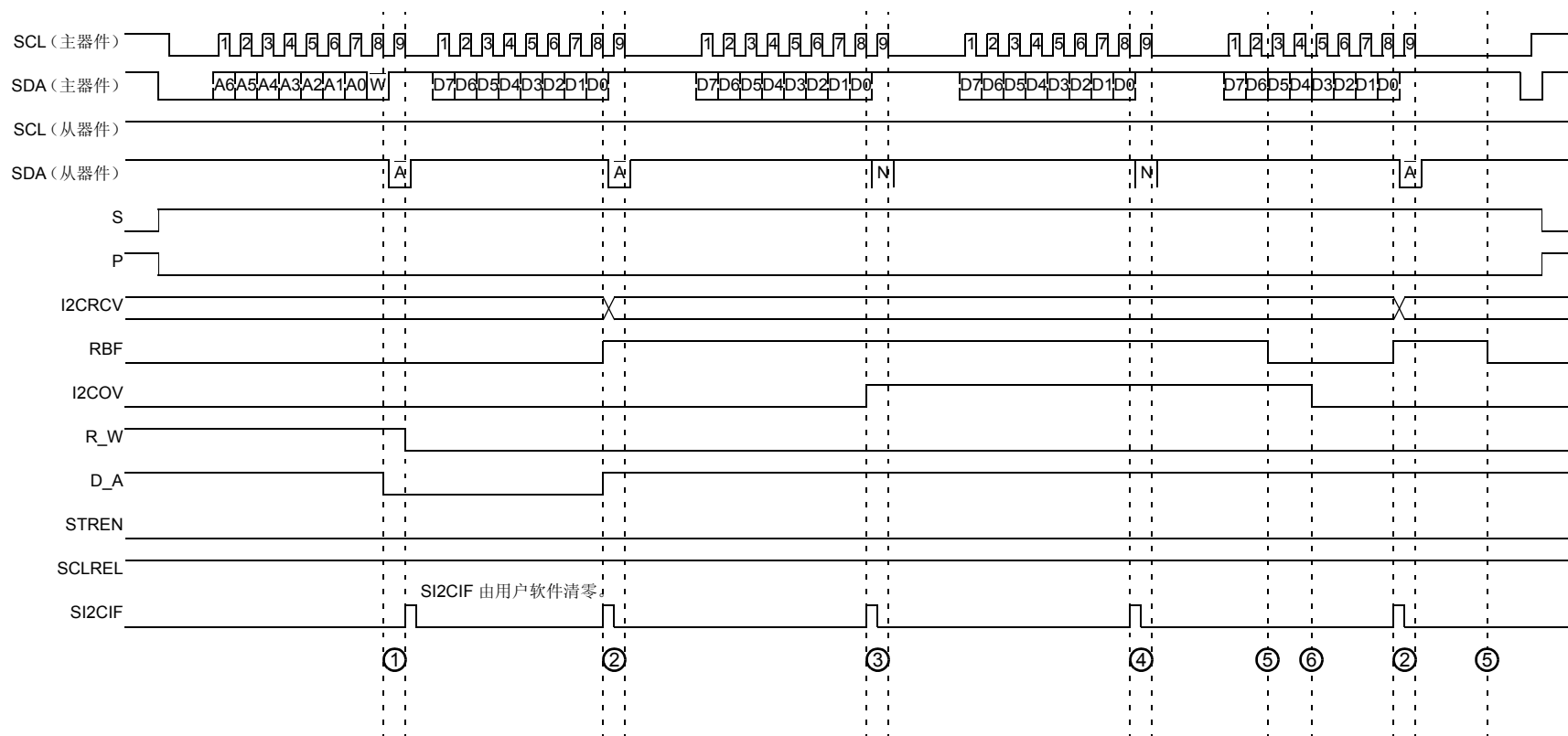
图 21-28: 从报文 (写数据到从模块: 10 位地址, 地址匹配, A10M = 1, GCEN = 0, IPMIEN = 0)



- ① —— 从器件识别启动事件, S 和 P 位相应地置位 / 清零。
- ② —— 从器件接收地址字节。高位地址匹配。从器件应答并产生中断。地址字节没有移动到 I2CRCV 寄存器。
- ③ —— 从器件接收地址字节。低位地址匹配。从器件应答并产生中断。地址字节没有移动到 I2CRCV 寄存器。

- ④ —— 下一个接收的字节是报文数据。字节转移到 I2CRCV 寄存器, 置位 RBF。从器件应答并产生中断。
- ⑤ —— 软件读 I2CRCV 寄存器。RBF 位清零。
- ⑥ —— 从器件识别停止事件, S 和 P 位相应地置位 / 清零。

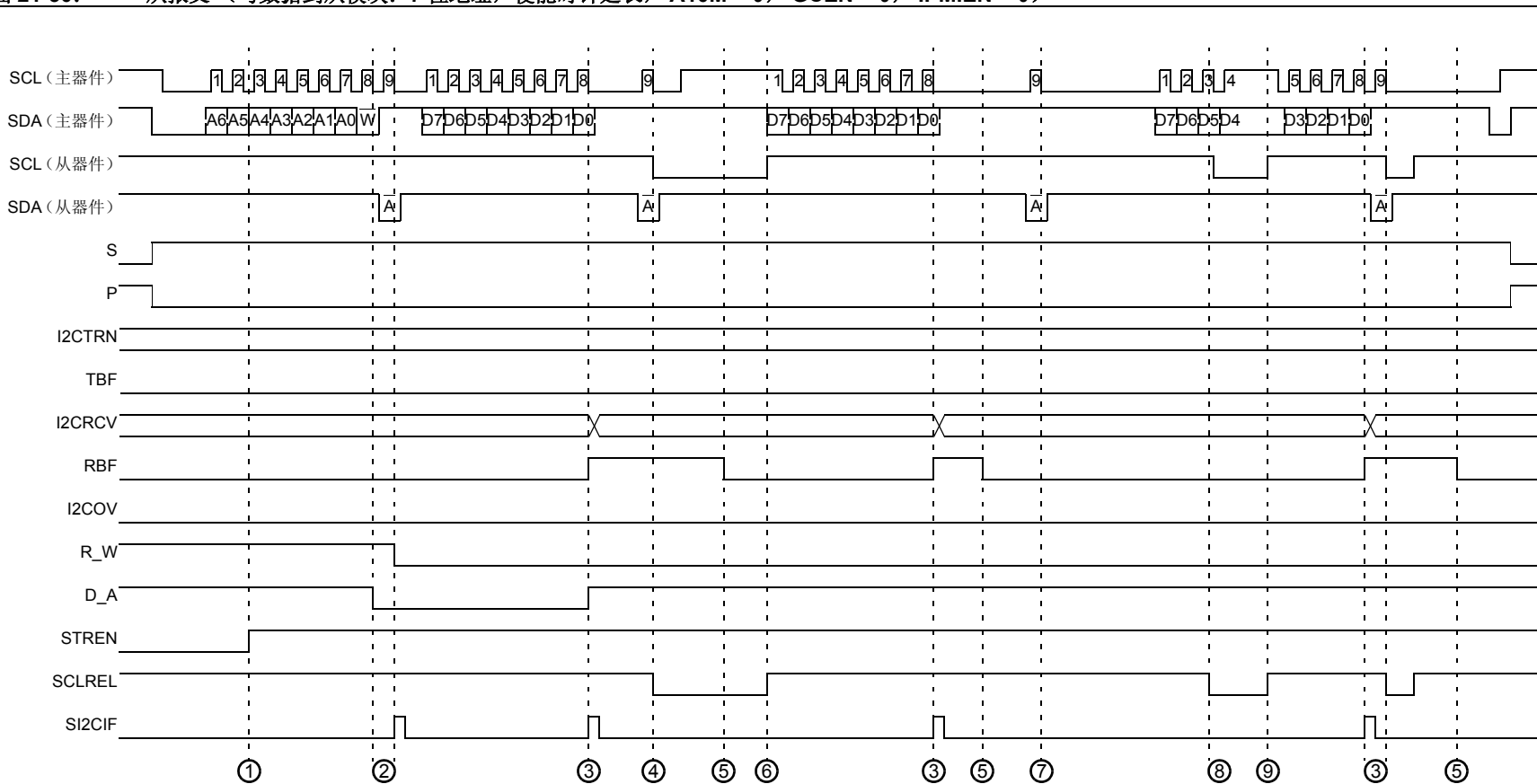
图 21-29: 从报文 (写数据到从模块: 7 位地址, 缓冲器溢出, A10M = 0, GCEN = 0, IPMIEN = 0)



- ① ——从器件接收地址字节。地址匹配。从器件产生中断。
地址字节没有移动到 I2CRCV 寄存器。
- ② ——下一个接收的字节是报文数据。字节转移到 I2CRCV 寄存器, 置位 RBF。
从器件产生中断。从器件应答接收。
- ③ ——在 I2CRCV 被软件读取之前接收的下一个字节。I2CRCV 寄存器未改变。
I2COV 溢出位置位。从器件产生中断。从器件为接收发送 NACK。

- ④ ——下一个字节也在 I2CRCV 被软件读取之前接收。
I2CRCV 寄存器未改变。从器件产生中断。
从器件为接收发送 NACK。
- ⑤ ——软件读 I2CRCV 寄存器。RBF 位清零。
- ⑥ ——软件清零 I2COV 位。

图 21-30: 从报文 (写数据到从模块: 7 位地址, 使能时钟延长, A10M = 0, GCEN = 0, IPMIEN = 0)



21.7.5 发送数据到主器件

当收到的器件地址字节的 $\overline{R/W}$ 位为 1 且发生了地址匹配，则将 R_W 位（ $I2CSTAT<2>$ ）置位。这时，主器件希望从器件通过发送一个数据字节作为响应。此字节的内容由系统协议定义且只可以由从模块发送。

当发生来自地址检测的中断时，软件可以将字节写入 $I2CTRN$ 寄存器以开始数据发送。

从模块置位 TBF 位。8 个数据位会在 SCL 输入的下降沿时移出。这样做，可以保证 SDA 信号在 SCL 高电平时有效。当 8 位全部移出后， TBF 位将被清零。

从模块在第九个 SCL 时钟的上升沿时检测来自主接收器的应答信号。

如果 SDA 线为低表示一个应答（ $\overline{ACK}$ ），主器件需要更多数据，即报文传输未完成。模块产生一个从中断以表示有更多的数据被请求。

在第九个 SCL 时钟的下降沿产生从中断。软件必须检查 $I2CSTAT$ 寄存器的状态并清零 $SI2CIF$ 标志位。

如果 SDA 线为高电平，表示不应答（ $NACK$ ），然后数据传输完成。从模块复位且不产生中断。从模块将等待检测下一个启动位。

21.7.5.1 从器件发送时的等待状态

在从器件发送报文的过程中，主器件希望在检测到 $\overline{R/W} = 1$ 的有效地址后立即返回数据。由于这个原因，不管何时返回数据，从器件都会自动产生总线等待。

在有效器件地址字节的第 9 个 SCL 时钟下降沿，或收到主器件对发送字节产生应答时，（从器件）发生自动等待，表示希望发送更多的数据。

从器件清零 $SCLREL$ 位。清零 $SCLREL$ 位导致从模块将 SCL 线拉为低电平，开始等待。主从器件的 SCL 时钟将同步，如第 21.6.2 节“主器件时钟同步”所示。

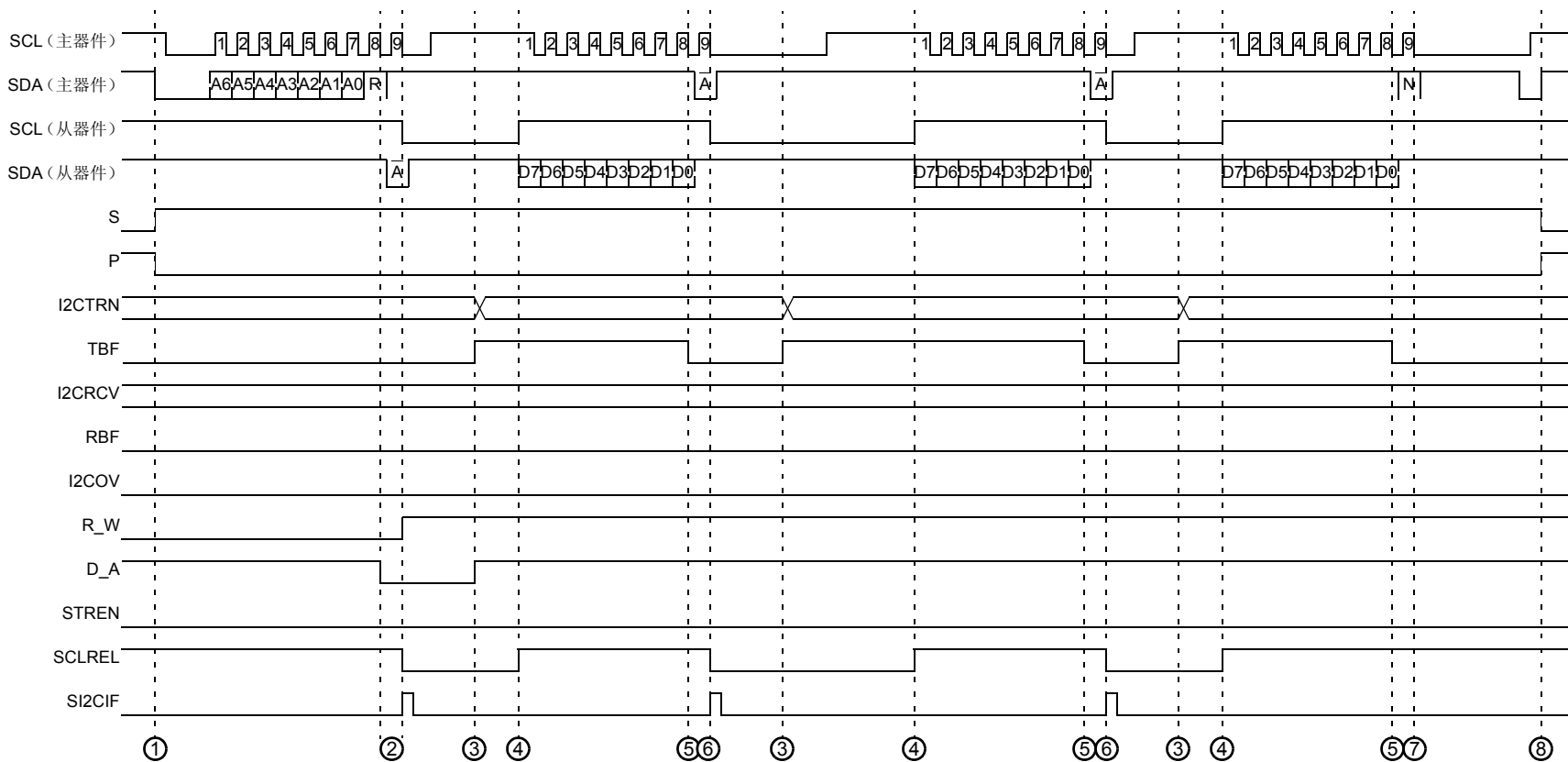
当软件载入 $I2CTRN$ 并准备恢复发送时，软件置位 $SCLREL$ 。这将导致从模块释放 SCL 线并且主模块恢复产生时钟信号。

21.7.5.2 从器件发送报文示例

图 21-31 所示为 7 位地址报文的从发送。当地址匹配且地址的 R/W 位表示从发送时，模块会自动清零 $SCLREL$ 位开始时钟延长并产生中断表示需要响应字节。软件将响应字节写入 $I2CTRN$ 寄存器。当发送结束时，主器件将以应答信号作出响应。如果主器件用一个 ACK 回应，主器件期望更多的数据，而且模块会再次清零 $SCLREL$ 位并产生另一个中断。如果主器件用一个 $NACK$ 响应，表示它不需要其他数据，而且模块不会延长时钟也不会产生中断。

10 位地址报文的从发送需要从器件首先识别 10 位地址。由于主模块必须为地址发送两个字节，地址的第一个字节中的 R/W 位指定一个写操作。要将报文改变为读操作，主器件将发送一个重新启动位并重发地址的第一个字节（但其中的 R/W 位指定为读）。此时，从器件开始发送，如图 21-32 所示。

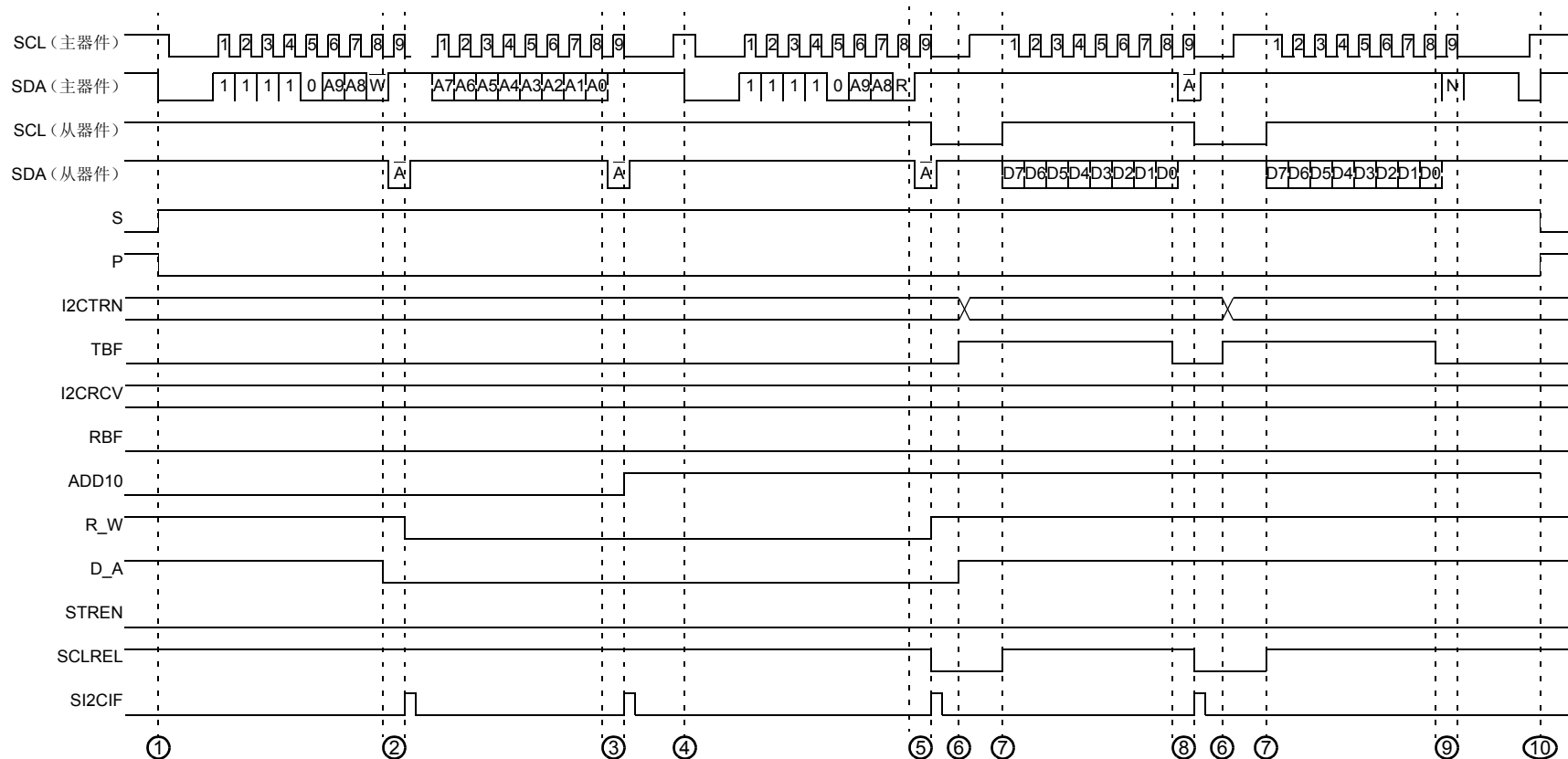
图 21-31: 从报文 (从从动模块读取数据: 7 位地址)



- ① ——从器件识别起始事件, S 和 P 位相应地置位 / 清零。
- ② ——从器件接收地址字节。地址匹配。从器件产生中断。
地址字节没有移动到 I2CRCV 寄存器。R_W = 1 表示从从器件读取。
SCLREL = 0 以暂停主时钟。
- ③ ——软件用响应数据写 I2CTR_N。TBF = 1 表示缓冲器满。
写 I2CTR_N 置位 D_A, 表示是数据字节。
- ④ ——软件置位 SCLREL 位以释放时钟保持。主器件恢复产生时钟并且
从器件发送数据字节。

- ⑤ ——在最后一位传输完之后, 模块清零 TBF 位表示缓冲器对于下一个字节可用。
- ⑥ ——在第九个时钟结束时, 如果主器件发送 ACK, 模块将清零 SCLREL 以暂停时钟。
从器件产生中断。
- ⑦ ——在第九个时钟结束时, 如果主器件发送 NACK, 则不期望接收其他数据。模块不
暂停时钟并产生中断。
- ⑧ ——从器件识别停止事件, S 和 P 位相应地置位 / 清零。

图 21-32: 从报文 (从从动模块读取数据: 10 位地址)



① ——从器件识别起始事件, S 和 P 位相应地置位 / 清零。

② ——从器件接收第一个地址字节。指出为写操作。从器件应答并产生中断。

③ ——从器件接收地址字节。地址匹配。从器件应答并产生中断。

④ ——主器件发送重复启动以重新确定报文传输的方向。

⑤ ——从器件接收再次发送的第一个地址字节。表示读。从器件暂停时钟。

⑥ ——软件用响应数据写 I2CTRN。

⑦ ——软件置位 SCLREL 位以释放时钟保持。主器件恢复产生时钟并且从器件发送数据字节。

⑧ ——在第九个时钟结束时, 如果主器件发送 $\overline{\text{ACK}}$, 模块将清零 SCLREL 以暂停时钟。从器件产生中断。

⑨ ——在第九个时钟结束时, 如果主器件发送 NACK, 则不期望接收其他数据。模块不暂停时钟或产生中断。

⑩ ——从器件识别停止事件, S 和 P 位相应地置位 / 清零。

21.8 I²C 总线的连接注意事项

因为 I²C 总线被定义为一个线与总线连接，所以总线上需要上拉电阻，如图 21-33 中的 R_P。串联电阻（以 R_S 表示）是可选的，用于提高抗 ESD 干扰性。电阻 R_P 和 R_S 的值由以下参数决定：

- 供电电压
- 总线容量
- 连接的器件数（输入电流 + 泄漏电流）

由于器件必须能够通过 R_P 拉低总线电压，R_P 的泄漏电流必须大于 I/O 引脚在器件输出级的 V_{OL}（最大值）= 0.4V 时的最小灌电流 I_{OL} 3 mA。例如，电源电压 V_{DD} = 5V+10% 时：

$$R_P(\text{最小值}) = (V_{DD}(\text{最大值}) - V_{OL}(\text{最大值})) / I_{OL} = (5.5 - 0.4) / 3 \text{ mA} = 1.7 \text{ k}\Omega$$

在 400 kHz 系统中，最小上升时间规范为 300 ns 而在 100 kHz 的系统中，该规范为 1000 ns。

由于 R_P 必须在总电容 C_B 在最大上升时间 300 ns 中将总线电压拉高到 0.7 V_{DD}，R_P 的最大电阻就必须小于：

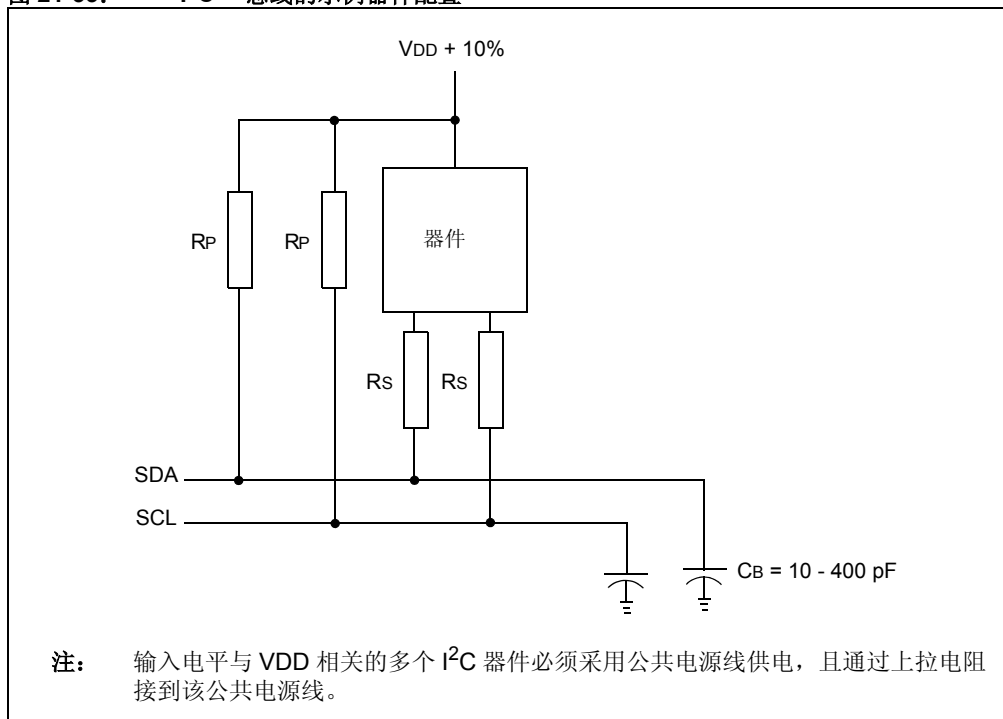
$$R_P(\text{最大值}) = -t_r / C_B * \ln(1 - (V_{IL}(\text{最大值}) - V_{DD}(\text{最大值}))) = -300 \text{ ns} / (100 \text{ pf} * \ln(1 - 0.7)) = 2.5 \text{ k}\Omega$$

R_S 的最大值由所需的小电流噪声容限决定。R_S 不能提供足够的电压降使器件 V_{OL} 与 R_S 两端的电压的和大于最大 V_{IL}。

$$R_S(\text{最大值}) = (V_{IL}(\text{MAX}) - V_{OL}(\text{最小值})) / I_{OL}(\text{最大值}) = (0.3 V_{DD} - 0.4) / 3 \text{ mA} = 366 \Omega$$

SCL 时钟输入必须有最小高电平和低电平时间才能进行正确操作。I²C 规范的高电平和低电平时间以及 I²C 模块的要求如特定的器件数据手册的“电气规范”部分所示。

图 21-33: I²C™ 总线的示例器件配置



21.8.1 集成的信号调节电路

SCL 和 SDA 引脚上都有输入毛刺滤波器。I²C 总线要求在 100 kHz 和 400 kHz 的系统中都有此滤波器。

在 400 kHz 总线上工作时，I²C 规范要求对器件引脚输出进行转换率控制。此转换率控制是集成在器件中的。如果 DISSLW 位 (I2CCON<9>) 被清零，转换率控制被激活。对于其他总线速度，I²C 规范不要求转换率控制并且 DISSLW 应该置位。

某些实现 I²C 总线的系统需要 V_{IL} (最大值) 和 V_{IH} (最小值) 不同的输入电平。

在正常的 I²C 系统中：

V_{IL} (最大值) = 1.5V 和 0.3 V_{DD} 中较小的一个

V_{IH} (最小值) = 3.0V 和 0.7 V_{DD} 中较大的一个

在 SMBus (系统管理总线) 系统中：

V_{IL} (最大值) = 0.2 V_{DD}

V_{IH} (最小值) = 0.8 V_{DD}

SMEN 位 (I2CCON<8>) 控制输入电平。SMEN 置位改变 SMBus 规范的输入电平。

21.9 在 PWRSAV 指令执行期间的模块操作

21.9.1 当器件进入休眠模式时

当器件执行 PWRSAV 0 指令时，器件进入休眠状态。当器件进入休眠模式时，主模块和从动模块将终止所有未处理的报文活动并复位模块的状态。当器件从休眠模式唤醒时所有在处理中的发送 / 接收都不会继续。在器件回到工作模式后，主模块将处于空闲状态等待报文命令而从模块将等待启动条件。在休眠模式中，IWCOL、I2COV 和 BCL 位被清零。此外，由于中止了主器件的功能，SEN、RSEN、PEN、RCEN、ACKEN 和 TRSTAT 位将清零。TBF 和 RBF 被清零且缓冲器在唤醒时可用。

不管发送或接收是激活的还是待进行的，都没有自动的方法可以阻止模块进入休眠模式。软件必须将进入休眠模式与 I²C 操作同步以避免中止报文。

在休眠过程中，从模块不会监视 I²C 总线。因此，不可能根据使用 I²C 模块的 I²C 总线而产生唤醒事件。其他中断输入，比如电平变化中断输入，可以用于检测 I²C 总线上的报文流量并引起器件唤醒。

21.9.2 当器件进入空闲模式时

当器件执行 PWRSAV 1 指令时，器件进入空闲状态。在空闲模式下，该模块会根据 I2CSIDL 位（I2CCON<13>）的设置进入低功耗状态。

如果 I2CSIDL = 1，模块将进入低功耗模式，这与进入休眠模式的行为相类似。

如果 I2CSIDL = 0，模块将不会进入低功耗模式。模块将继续正常工作。

21.10 复位的影响

复位将禁止 I²C 模块并终止任何活动或待处理的报文活动。这些寄存器的复位条件参见 I2CCON 和 I2CSTAT 寄存器定义。

21.11 设计技巧

问 1: *我将模块作为总线主器件运行并发送数据，但是还是发生了从器件和接收中断。*

答: 主器件和从器件电路是独立的。从动模块将从总线上接收主模块发送的事件。

问 2: *我将模块作为从器件运行并写数据到 I2CTRN 寄存器，但数据没有发送。*

答: 在准备发送时从器件进入自动等待状态。确保您置位了 SCLREL 位以释放 I²C 时钟。

问 3: *如何分辨主模块的状态？*

答: 检查 SEN、RSEN、PEN、RCEN、ACKEN 和 TRSTAT 位的状态可以表示主模块的状态。如果所有位都为 0，模块为空闲状态。

问 4: *作为从器件运行时，当 STREN = 0 时接收到一个字节。如果不能在接收下一个字节之前处理该字节，软件应该做些什么？*

答: 由于 STREN 为 0，模块不会在接收到字节时产生自动等待。但是，软件可能在报文的任何时间置位 STREN 然后清零 SCLREL。这将在同步 SCL 时钟的下一个机会产生等待。

问 5: *我的 I²C 系统是一个多主机系统。当我尝试发送报文时，该报文就会被破坏。*

答: 在多主机系统中，其他主器件可能导致总线冲突。在主器件的中断服务程序中，检查 BCL 位以确保操作已完成且没有发生冲突。如果检测到了冲突，报文必须重新从头发送。

问 6: *我的 I²C 系统是一个多主机系统。我如何分辨它何时准备就绪开始报文传输？*

答: 看 S 和 P 位。如果 S = 0 且 P = 0，总线空闲。如果 S = 0 且 P = 1，总线空闲。

问 7: *我尝试在总线上发送启动条件，然后通过写 I2CTRN 寄存器发送一个字节。此字节没有被发送。为什么？*

答: 在开始下一个事件之前您必须等待 I²C 总线上的每个事件都完成。在这种情况下，您应该查询 SEN 位确定启动事件完成的时间，或在数据写入 I2CTRN 之前等待主器件 I²C 中断。

21.12 相关应用笔记

本节列出了与手册的本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能受到一定限制。当前与 I²C 模块相关的应用笔记是：

标题	应用笔记编号
Use of the SSP Module in the I ² C™ Multi-Master Environment	AN578
Using the PIC® SSP for Slave I ² C™ Communication	AN734
Using the PIC® MSSP Module for Master I ² C™ Communications	AN735
An I ² C™ Network Protocol for Environmental Monitoring	AN736

注： 如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

21.13 版本历史

版本 A

这是本文档的初始发行版。

版本 B

此版本已通过扩展包含了 dsPIC30F I²C 模块的全面说明。

版本 C

此版本收录了本文档更新时已知的所有勘误。

第 22 章 数据转换器接口 (DCI)

目录

本章包括下列主题：

22.1 简介	22-2
22.2 控制寄存器说明	22-2
22.3 编解码器接口基础与术语	22-8
22.4 DCI 工作原理	22-10
22.5 使用 DCI 模块	22-17
22.6 低功耗模式下的工作	22-28
22.7 与 DCI 相关的寄存器	22-28
22.8 设计技巧	22-30
22.9 相关应用笔记	22-31
22.10 版本历史	22-32

22.1 简介

dsPIC 数据转换器接口 (Data Converter Interface, DCI) 允许与一些器件进行简单接口, 例如音频编解码器 / 解码器 (编解码器)、A/D 转换器和 D/A 转换器。

支持以下接口:

- 帧同步串行传输 (单通道或多通道)
- Inter-IC Sound (I²S) 接口
- AC-Link 兼容模式

很多用于音频应用的编解码器支持 8 kHz 到 48 kHz 的采样速率并使用上述接口协议之一。DCI 会自动处理与这些编解码器相关的接口时序。在 DCI 发送和 / 或收到所需的数据量前, 不需要 CPU 开销。在 CPU 中断之间最多可以传输 4 个数据字。

要符合 dsPIC30F CPU 的数据长度要求, DCI 数据字的长度最大只能编程至 16 位。但是, 很多编解码器都有大于 16 位的数据字。DCI 能够支持长数据字。DCI 被配置为在多个 16 位时隙 (Slot) 中发送 / 接收长字。这种操作对用户透明, 并且长字储存在连续的寄存器单元中。

DCI 能在一个数据帧内最多支持 16 个时隙, 即最大 256 位的帧大小。在数据帧中的每个时隙都有相应的控制位, 决定 DCI 在该时隙期间是进行发送还是进行接收。

22.2 控制寄存器说明

DCI 具有如下五个控制寄存器和一个状态寄存器:

- DCICON1: DCI 模块使能和模式位。
- DCICON2: DCI 模块字长、数据帧长度和缓冲器设置。
- DCICON3: DCI 模块位时钟发生器设置。
- DCISTAT: DCI 模块状态信息。
- RSCON: 数据接收的有效帧时隙控制。
- TSCON: 数据发送的有效帧时隙控制。

除了这些控制和状态寄存器外, 还有四个发送寄存器 (TXBUF0.....TXBUF3) 和四个接收寄存器 (RXBUF0.....RXBUF3)。

寄存器 22-1: DCICON1

高字节:							
R/W-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
DCIEN	—	DCISIDL	—	DLOOP	CCKD	CCKE	COFSD
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
UNFM	CSDOM	DJST	—	—	—	COFSM<1:0>	
bit 7						bit 0	

- bit 15 **DCIEN:** DCI 模块使能位
1 = 使能模块
0 = 禁止模块
- bit 14 **保留:** 读作 0
- bit 13 **DCISIDL:** DCI 在空闲时停止控制位
1 = 模块在 CPU 空闲模式将停止
0 = 模块在 CPU 空闲模式将继续工作
- bit 12 **保留:** 读作 0
- bit 11 **DLOOP:** 数字环回模式控制位
1 = 使能数字环回模式, CSDI 和 CSDO 引脚是在内部连接在一起。
0 = 禁止数字环回模式
- bit 10 **CCKD:** 采样时钟方向控制位
1 = 当 DCI 模块被使能时, CCK 是输入引脚
0 = 当 DCI 模块被使能时, CCK 是输出引脚
- bit 9 **CCKE:** 采样时钟沿控制位
1 = 在串行时钟下降沿改变数据, 在串行时钟上升沿采样数据。
0 = 在串行时钟上升沿改变数据, 在串行时钟下降沿采样数据。
- bit 8 **COFSD:** 帧同步方向控制位
1 = 当 DCI 模块被使能时, COFS 是输入引脚
0 = 当 DCI 模块被使能时, COFS 是输出引脚
- bit 7 **UNFM:** 下溢模式位
1 = 在发送下溢时发送最后写入发送寄存器的值
0 = 在发送下溢时发送 0
- bit 6 **CSDOM:** 串行数据输出模式位
1 = 当发送时隙被禁止时 CSDO 引脚为三态
0 = 当发送时隙被禁止时 CSDO 引脚为 “0”
- bit 5 **DJST:** DCI 数据对齐控制位
1 = 在产生帧同步脉冲的同一串行时钟周期内开始数据发送 / 接收
0 = 在帧同步脉冲的后一个串行时钟周期开始数据发送 / 接收
- bit 4-2 **保留:** 读作 0
- bit 1-0 **COFSM<1:0>:** 帧同步模式位
11 = 20 位 AC-Link 模式
10 = 16 位 AC-Link 模式
01 = I²S 帧同步模式
00 = 多通道帧同步模式

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

dsPIC30F 系列参考手册

寄存器 22-2: DCICON2

高字节:							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	U-0	R/W-0
—	—	—	—	BLEN<1:0>		—	COFSG3
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
COFSG<2:0>			—	WS<3:0>			
bit 7							bit 0

bit 15-12 保留: 读作 0

bit 11-10 **BLEN<1:0>**: 缓冲器长度控制位
11 = 在中断之间将缓冲四个数据字
10 = 在中断之间将缓冲三个数据字
11 = 在中断之间将缓冲两个数据字
11 = 在中断之间将缓冲一个数据字

bit 9 保留: 读作 0

bit 8-5 **COFSG<3:0>**: 帧同步发生器控制位
1111 = 数据帧有 16 个字
||
0010 = 数据帧有 3 个字
0001 = 数据帧有 2 个字
0000 = 数据帧有 1 个字

bit 4 保留: 读作 0

bit 3-0 **WS<3:0>**: DCI 数据字长位
1111 = 数据字长为 16 位
||
0100 = 数据字长为 5 位
0011 = 数据字长为 4 位
0010 = **无效选择**。请勿使用。否则可能会发生无法预料的结果。
0001 = **无效选择**。请勿使用。否则可能会发生无法预料的结果。
0000 = **无效选择**。请勿使用。否则可能会发生无法预料的结果。

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 22-3: DCICON3

高字节:							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	BCG<11:8>			
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BCG<7:0>							
bit 7				bit 0			

bit 15-12 保留：读作 0。
bit 11-0 BCG<11:0>：DCI 位时钟发生器控制位

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 22-4: DCISTAT

高字节:							
U-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
—	—	—	—	SLOT<3:0>			
bit 15				bit 8			

低字节:							
U-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
—	—	—	—	ROV	RFUL	TUNF	TMPTY
bit 7				bit 0			

- bit 15-12 保留: 读作 0
- bit 11-8 **SLOT<3:0>**: DCI 时隙状态位
1111 = Slot#15 目前有效
||
0010 = Slot#2 目前有效
0001 = Slot#1 目前有效
0000 = Slot#0 目前有效
- bit 7-4 保留: 读作 0
- bit 3 **ROV**: 接收溢出状态位
1 = 至少一个接收寄存器发生了接收溢出
0 = 没有发生接收溢出
- bit 2 **RFUL**: 接收缓冲器满标志位
1 = 在接收寄存器中有新数据
0 = 接收寄存器中为旧数据
- bit 1 **TUNF**: 发送缓冲器下溢状态位
1 = 至少一个发送寄存器发生了发送下溢
0 = 没有发生发送下溢
- bit 0 **TMPTY**: 发送缓冲器空状态位
1 = 发送寄存器空
0 = 发送寄存器非空

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 22-5: RSCON

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RSE15	RSE14	RSE13	RSE12	RSE11	RSE10	RSE9	RSE8
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RSE7	RSE6	RSE5	RSE4	RSE3	RSE2	RSE1	RSE0
bit 7				bit 0			

bit 11 **RSE<15:0>**: 接收时隙使能位
1 = CSDI 数据在单独的时隙 n 接收
0 = 忽略了在单独的时隙 n 内的 CSDI 数据

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 22-6: TSCON

高字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TSE15	TSE14	TSE13	TSE12	TSE11	TSE10	TSE9	TSE8
bit 15							bit 8

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TSE7	TSE6	TSE5	TSE4	TSE3	TSE2	TSE1	TSE0
bit 7				bit 0			

bit 11 **TSE<15:0>**: 发送时隙使能控制位
1 = 发送缓冲器中的内容在独立的时隙 n 期间发送
0 = 根据 CSDOM 位的状态, CSDO 引脚在独立的时隙内为三态或为逻辑 0

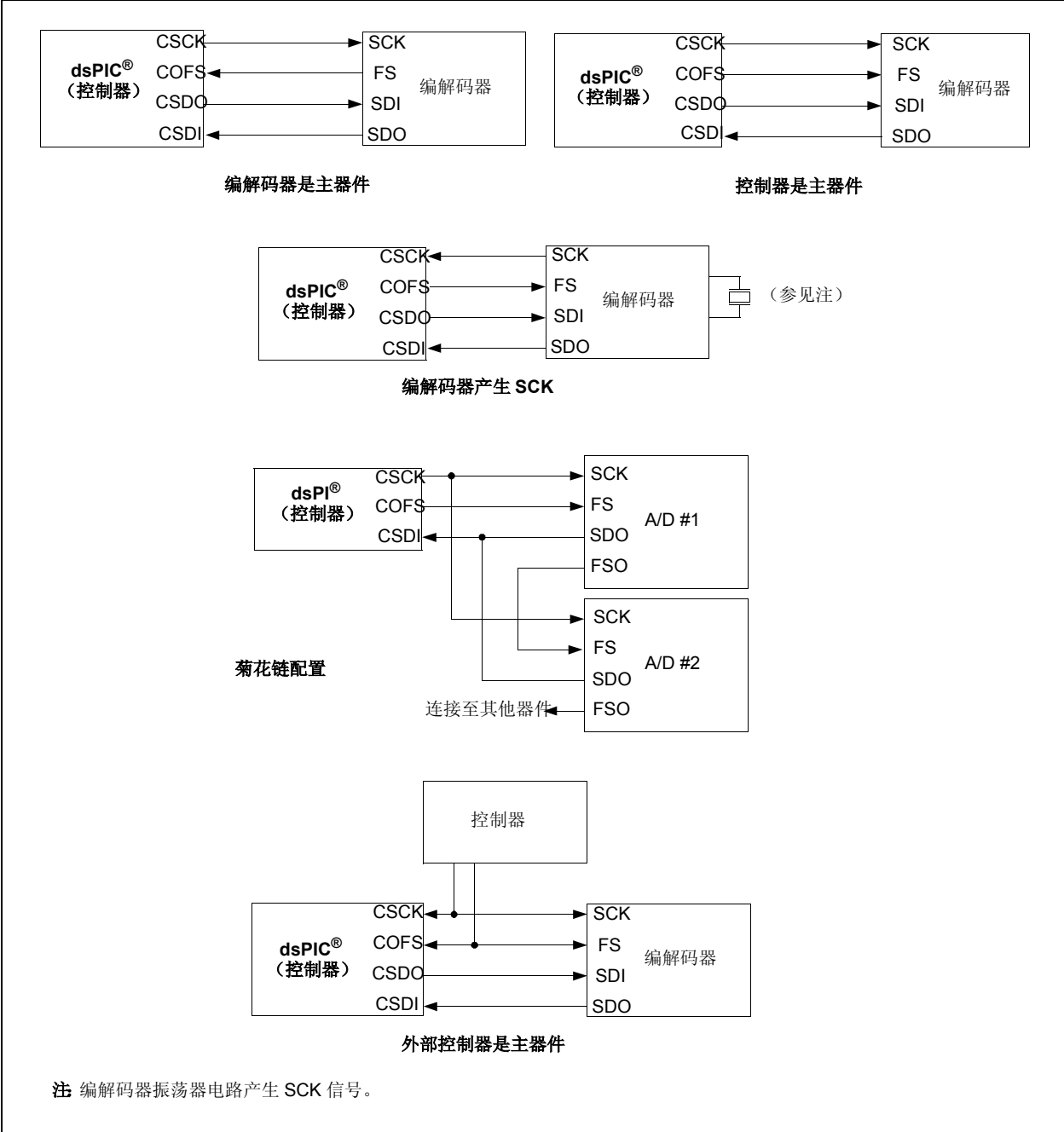
图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

22.3 编解码器接口基础与术语

DCI 支持的接口协议需要使用一个帧同步（Frame Synchronization，FS）信号来启动两个器件之间的数据传输。在大部分情况下，会在 FS 的上升沿开始一次新的数据传输。在任何编解码器应用中，都至少有一个控制器和编解码器件。这两种器件都可以产生 FS。产生 FS 信号的器件是主器件。概念上，主器件并不一定要是发送或接收器件。图 22-1 显示了各种连接的示例。FS 信号的频率通常是系统采样速率 fs。

注： 本节提供的内容并不特指 DCI 模块。本讨论旨在为用户介绍与大部分编解码器件所使用的数字串行接口协议相关的背景知识和术语。

图 22-1: 编解码器连接示例



所有接口都有一个串行传输时钟 **SCK**。**SCK** 信号可以由任何连接到该接口的器件产生，也可以由外部提供。在某些系统中，**SCK** 也称为位时钟。对于提供高保真信号的编解码器，**SCK** 信号通常都是由编解码器件的晶体振荡器产生的。该协议定义了数据采样的 **SCK** 沿。主器件会根据 **SCK** 产生 **FS** 信号。

FS 信号的周期形成一个数据帧。此周期与数据采样周期相同。在数据帧传输过程中发生的 **SCK** 周期数取决于所选择的编解码器类型。**SCK** 频率与系统采样速率的比例可表示为比例 n ，其中 n 是每个数据帧中的 **SCK** 周期数。

使用分帧接口协议的好处是可以在每个采样周期或数据帧中传输多个数据字。数据帧的各个分段称为时隙。时隙可以用于多个编解码器数据通道和 / 或控制信息。此外，多个器件可以复用同一个串行数据引脚。各从器件被编程为在适当的时隙将数据放在串行数据连接引脚上。各个从器件的输出在任何时候都是三态的，以便让其他器件能使用串行总线。

某些器件可以让 **FS** 信号通过帧同步输出 (**Frame Synchronization Output, FSO**) 引脚构成菊花链。图 22-1 所示为一个典型的菊花链配置。当来自第一个从器件的传输结束后，会通过其 **FSO** 引脚发送一个 **FS** 脉冲给链上的第二个器件。此过程会一直继续，直到链上的最后一个器件发出了数据。控制器 (主) 器件应该设置使得数据帧大小能够容纳传输的所有数据字。

图 22-2 所示为典型的数据传输时序。大部分协议在检测到 **FS** 信号后的一个 **SCK** 周期开始数据传输。此例使用了一个 16 **fs** 时钟且每帧传输四个 4 位数据字。

图 22-2: 帧数据传输示例

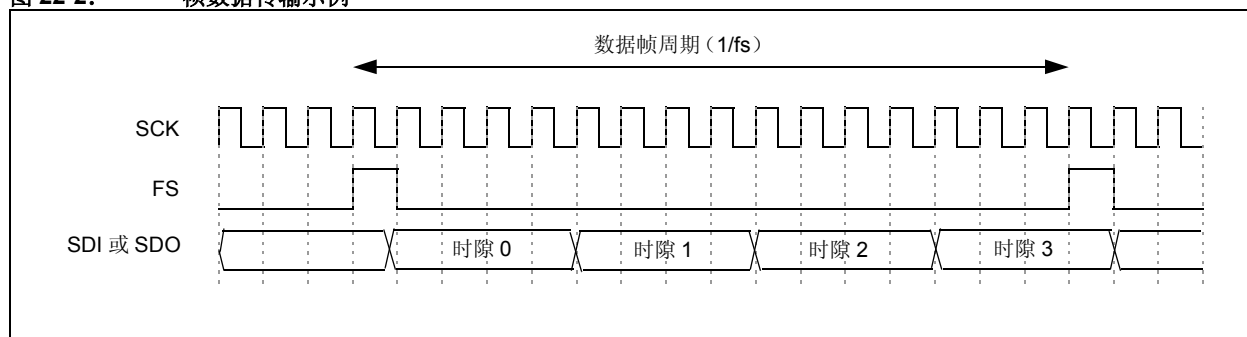
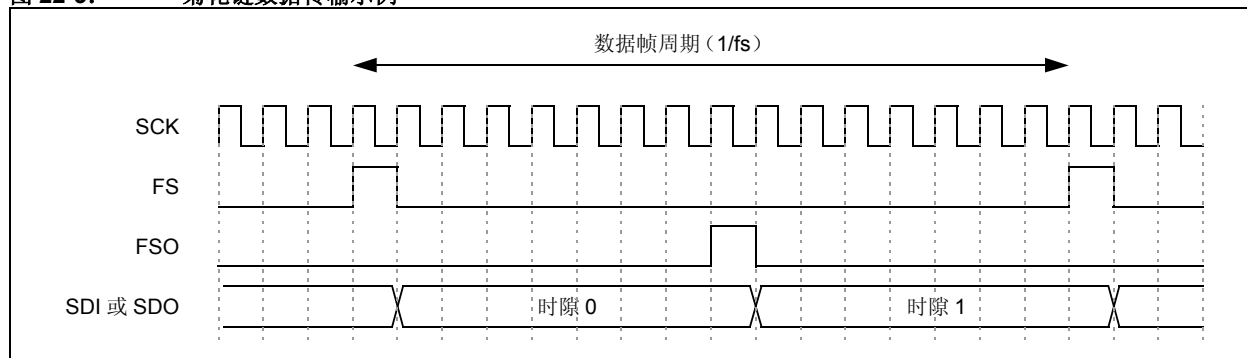


图 22-3 给出了菊花链器件的典型数据传输时序。此示例使用 16 **fs** **SCK** 频率且每帧传输两个 8 位数据字。在检测到 **FS** 脉冲后，菊花链上的第一个器件传输第一个 8 位数据字且在传输结束时产生 **FSO** 信号。**FSO** 信号开始菊花链上第二个器件的第二个数据字的传输。

图 22-3: 菊花链数据传输示例



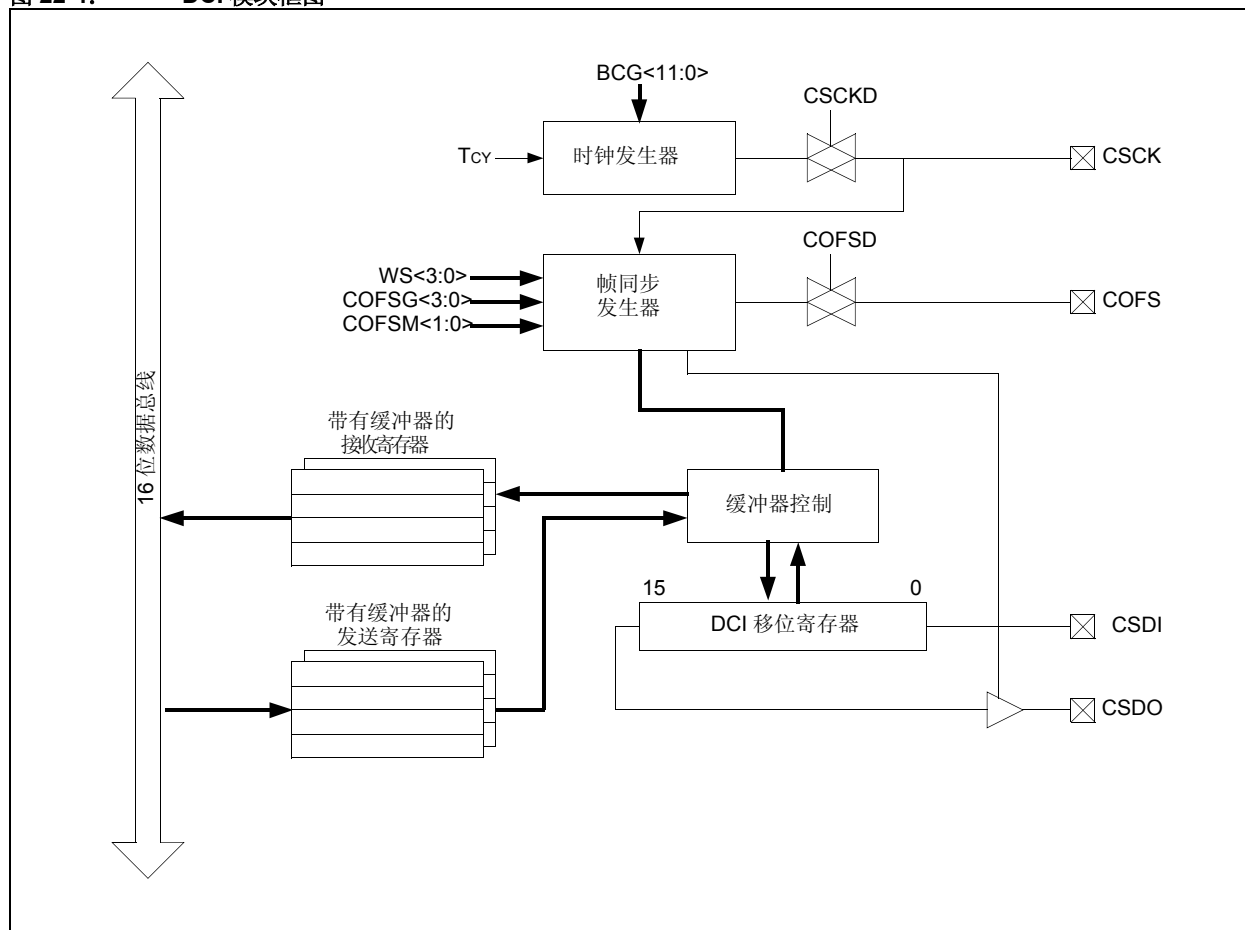
FS 脉冲有一个 SCK 周期的最小有效时间，使得此从器件可以检测到数据帧的起始位。FS 脉冲的占空比可能会随着用于标记数据帧中的特定边界的协议的不同而变化。例如，I²S 协议使用占空比为 50% 的 FS 信号。I²S 协议为两个数据通道（左右声道音频信息）的传输进行了优化。FS 信号的边沿标记了左右声道数据字的边界。AC-Link 协议使用 FS 信号，该信号在 16 SCK 周期为高电平，240 SCK 周期为低电平。AC-Link 协议使用了在 16 SCK 周期为高电平、在 240 SCK 周期为低电平的 FS 信号。

注： 如需编解码器通信协议的更多信息，参见本手册的**第 26 章“附录”**。

22.4 DCI 工作原理

图 22-4 所示为模块的简化框图。模块包含一个发送 / 接收移位寄存器，该寄存器通过缓冲器控制单元连接到一小段存储器缓冲区。这种安排可以让 DCI 支持各种编解码器串行协议。DCI 移位寄存器为 16 位宽。数据的发送和接收都是首先从 DCI 的 MSb 开始的。

图 22-4: DCI 模块框图



22.4.1 DCI 引脚

有四个 I/O 引脚与 DCI 相关。当使能了 DCI 时，DCI 控制这四个引脚的数据方向。

22.4.1.1 CSCK 引脚

CSCK 引脚为 DCI 提供串行时钟连接。使用 CSCKD 控制位 (DCICON1<10>)，CSCK 引脚可以配置为输入或输出引脚。当 CSCK 引脚配置为输出 (CSCKD = 0) 时，串行时钟是 dsPIC30F 系统时钟源产生的，并通过 DCI 提供给外部器件。当 CSCK 引脚配置为输入 (CSCKD = 1) 时，串行时钟必须由外部器件提供。

22.4.1.2 CSDO 引脚

当模块使能时，串行数据输出 CSDO 引脚被配置为只能输出的引脚。只要需要发送数据，CSDO 引脚就会驱动串行总线。在不发送数据的串行时钟周期内，CSDO 引脚可以为三态或驱动为 0，这取决于 CSDOM 控制位 (DCICON1<6>) 的状态。三态选项可以让其他器件与 CSDO 连接引脚复用。

22.4.1.3 CSDI 引脚

当模块使能时，串行数据输入 CSDI 引脚被配置为只能输入的引脚。

22.4.1.4 COFS 引脚

帧同步 (COFS) 引脚用于同步发生在 CSDO 和 CSDI 引脚上的数据传输。COFS 引脚可以配置为输入或输出引脚。COFS 引脚的数据方向是由 COFSD 控制位 (DCICON1<8>) 决定的。当 COFSD 位清零时，COFS 是输出引脚。DCI 模块将产生帧同步脉冲来启动数据传输。DCI 在此配置中为主器件。当 COFSD 位置位时，COFS 是输入引脚。向模块输入同步信号将启动数据传输。当 COFSD 控制位置位时 DCI 是从器件。

22.4.2 模块使能

DCI 模块的使能或禁止是通过置位 / 清零 DCIEN 控制位 (DCICON1<15>) 完成的。将 DCIEN 控制位清零将复位模块。尤其是，所有与串行时钟发生、帧同步和缓冲器控制逻辑相关的计数器都会被复位 (如需更多信息，请参见第 22.5.1.1 节 “DCI 启动和数据缓冲” 和第 22.5.1.2 节 “DCI 禁止”)。

当 DCI 使能时，它控制与模块相关的 CSCK、CSDI、CSDO 和 COFS I/O 引脚的数据方向。当 DCIEN 位置位时，这些 I/O 引脚的 PORT、LAT 和 TRIS 寄存器的值都会被 DCI 模块改写。

当使能了位时钟发生器时，也可能会单独改写 CSCK 引脚的上述寄存器值。这可以无需使能 DCI 模块的其余部分，就能让位时钟发生器工作。

22.4.3 位时钟发生器

DCI 模块有一个产生位时钟的专用 12 位时基。位时钟的速率（周期）是通过写一个非零 12 位值到 BCG<11:0> 控制位（DCICON3<11:0>）设置的。当 BCG<11:0> 位置为零时，位时钟将禁止。

注： 如果 DCIEN 位置位或通过写非零值到 BCG<11:0> 使能位时钟发生器，CSCK I/O 引脚将由 DCI 模块控制。这可以让位时钟发生器的工作独立于 DCI 模块。

当 CSCK 引脚由 DCI 模块控制时，其相应的 PORT、LAT 和 TRIS 控制寄存器值将被改写，并且 CSCK 的数据方向将由 SCKD 控制位（DCICON1<10>）控制。

如果 DCI 的串行时钟由外部器件提供，BCG<11:0> 位应该置为 0，CSCKD 位应置为 1。

如果串行时钟要由 DCI 模块产生，BCG<11:0> 控制位应该置为非零值（参见公式 22-1），且 CSCKD 控制位应清零。

公式 22-1 给出了位时钟频率的计算公式。

公式 22-1: DCI 位时钟发生器值

$$\text{BCG<11:0>} = \frac{f_{CY}}{2 f_{CSCK}} - 1$$

所需的位时钟频率将取决于系统采样速率和帧大小。根据数据转换器和使用的通信协议，典型的位时钟频率范围在 16 倍到 512 倍转换器采样速率之间。

注： 当 CSCK 信号由外部提供时（CSCKD = 1），BCG<11:0> 位对 DCI 模块的工作没有影响。

22.4.4 采样时钟沿选择

CSCKE 控制位（DCICON1<9>）决定串行时钟信号的采样沿。如果 CSCKE 位清零（默认），将在 CSCK 信号的下降沿采样数据。AC-Link 协议和大部分多通道格式要求在 CSCK 信号的下降沿采样数据。如果 CSCKE 位置位，将在 CSCK 上升沿采样数据。I²S 协议要求在串行时钟信号的上升沿采样数据。

22.4.5 帧同步模式控制位

DCI 支持的接口协议类型是使用 COFSM<1:0> 控制位（DCICON1<1:0>）选择的。可选择以下工作模式：

- 多通道模式
- I²S 模式
- AC-Link 模式（16 位）
- AC-Link 模式（20 位）

在后续各节提供了各种协议的具体信息。

22.4.6 字长选择位

WS<3:0> 字长选择位（DCICON2<3:0>）决定各个 DCI 数据字的位数。可以选择 4 到 16 位之间的任何数据长度。

注： WS 控制位只在多通道和 I²S 模式中使用。这些位对 AC-Link 模式没有影响，因为该协议中数据间隙大小是固定的。

22.4.7 帧同步发生器

帧同步发生器 (Frame Sync Generator, FSG) 是一个在数据字中设置帧长度的 4 位计数器。FSG 的周期是通过写 COFSG<3:0> 控制位 (DCICON2<8:5>) 设置的。(串行时钟周期中的) FSG 周期由以下公式决定:

公式 22-2: 帧长度 (在 CSCK 周期内)

$$\text{帧长度} = (\text{WS}<3:0> + 1) \cdot (\text{COFSG}<3:0> + 1)$$

可以选择的帧长度最大为 16 个数据字。串行时钟周期中的帧长度会随选择的字长而变化, 最大可以为 256。

注: COFSG 控制位在 AC-Link 模式中不起作用, 因为该协议中帧长度已经被设置为 256 个串行时钟周期。

22.4.8 发送和接收寄存器

DCI 有四个发送寄存器 (TXBUF0……TXBUF3) 和四个接收寄存器 (RXBUF0……RXBUF3)。所有的发送和接收寄存器都是存储器映射的。

22.4.8.1 缓冲器数据对齐

数据值在 DCI 寄存器中总是以左对齐方式存储, 因为音频 PCM 数据是用一个有符号 2 的补码小数表示的。如果设定的 DCI 字长小于 16 位, 接收寄存器中未使用的 LSB 会被模块置为 0。此外, 发送寄存器中未使用的 LSB 会被模块忽略。

22.4.8.2 发送和接收缓冲器

发送和接收寄存器均有一组用户无法访问的缓冲器。实际上, 每个发送和接收缓冲器单元都是双缓冲的。DCI 从发送缓冲器发送数据, 并将接收到的数据写入接收缓冲器。当 DCI 使用来自缓冲器的数据时, 缓冲器可以让用户读写 RXBUF 和 TXBUF 寄存器。

22.4.9 DCI 缓冲器控制单元

DCI 模块包含一个缓冲器控制单元, 在缓冲器存储器和串行移位寄存器之间传输数据。缓冲器控制单元还可以在缓冲器存储器和 TXBUF 与 RXBUF 寄存器之间传输数据。缓冲器控制单元可以让 DCI 将多个数据字的发送和接收排队而不会产生 CPU 开销。

每次在缓冲器存储器和 TXBUF 与 RXBUF 寄存器之间发生传输数据时, DCI 就会产生一个中断。中断之间缓冲的数据字数量取决于 BLEN<1:0> 控制位 (DCICON2<11:10>)。发送和接收缓冲的大小可以在 1 到 4 个数据字之间变动, 由 BLEN<1:0> 位决定。

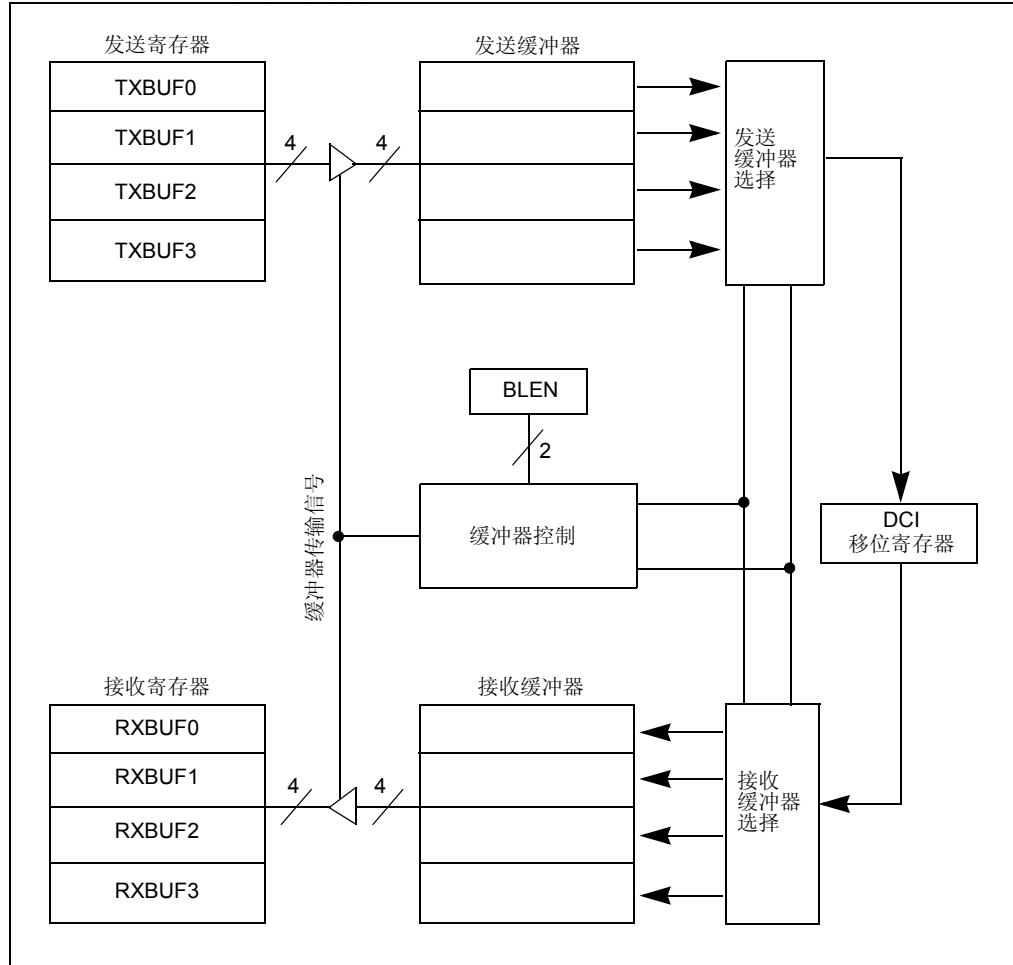
每次在 DCI 移位寄存器和缓冲器存储器之间发生数据传输时, DCI 缓冲器控制单元会递增以指向下一个缓冲器单元。如果发送或接收的数据字数量等于 BLEN 值 + 1, 则会发生以下情况:

- 缓冲器控制单元复位, 以指向第一个缓冲器单元
- 保存在缓冲器中的已接收数据被传输到 RXBUF 寄存器
- TXBUF 寄存器中的数据传输到缓冲器
- 产生一个 CPU 中断

每次到达帧边界时, DCI 缓冲器控制单元也会将缓冲器指针复位到指向第一个缓冲器单元。此操作可以确保缓冲器单元与在数据帧中使能的时隙之间是对齐的。

DCI 缓冲控制单元总是会访问发送和接收缓冲器中相同的相对地址单元。例如, 如果 DCI 正在发送 TXBUF3 中的数据, 则该时隙中接收到的任何数据也都会写入 RXBUF3。

图 22-5: DCI 缓冲器控制单元



22.4.10 发送时隙使能位

TSCON SFR 具有的控制位可用于使能最多 16 个发送时隙。这些控制位为 TSE<15:0> 位。每个时隙的大小由 WS<3:0> 字长选择位决定并可以变化，最多可以为 16 位。

如果通过一个 TSE 位（TSE_x = 1）使能发送时隙，当前发送缓冲器单元中的内容就会被装入 CSDO 移位寄存器，同时 DCI 缓冲器控制单元将递增 1 以指向下一个缓冲器单元。

如果所选的帧大小小于 16 个数据间隙，不是所有的 TSE 控制位都会对模块工作产生影响，即不使用高位的 TSE 控制位。例如，如果 COFSG<3:0> = 0111（每帧 8 个数据间隙），那么 TSE8 到 TSE15 对 DCI 工作就不产生影响。

22.4.10.1 CSDO 模式控制

在禁止发送的时隙中，根据 CSDOM 位（DCICON1<6>）的状态，CSDO 引脚可以为 0 或三态。如果某个发送时隙在 TSCON 寄存器中相应的 TSE_x 位被清零，此发送时隙将被禁止。

如果 CSDOM 位被清零（默认），在禁止时隙周期中 CSDO 引脚将为 0。当只有两个器件（1 个主器件和一个从器件）连接到串行总线时使用此模式。

如果 CSDOM 位被置位，CSDO 引脚在未使用的时隙周期中将为三态。此模式允许多个 dsPIC30F 器件共用同一根 CSDO 线形成一个复用应用系统，此时应将 CSDO 线中的所有器件配置为只在特定帧内发送数据，而不允许有两个器件在同一帧时发送数据。

22.4.11 接收时隙使能位

RSCON SFR 包含用于使能多达 16 个接收时隙的控制位（RSE<15:0>）。每个接收时隙的大小由 WS<3:0> 控制位决定，大小可以为 4 位到 16 位。

如果接收时隙通过一个 RSE 位（RSE_x = 1）使能，移位寄存器内容将被写入当前的 DCI 接收缓冲器单元，同时缓冲器控制逻辑将转到下一个可用的缓冲器单元。

如果所选的字长小于 16 位，数据在接收存储器缓冲单元中不被打包。每个接收时隙数据字都存储在一个单独的 16 位缓冲器单元中。数据总是以左对齐的格式存储在接收存储缓冲器中。

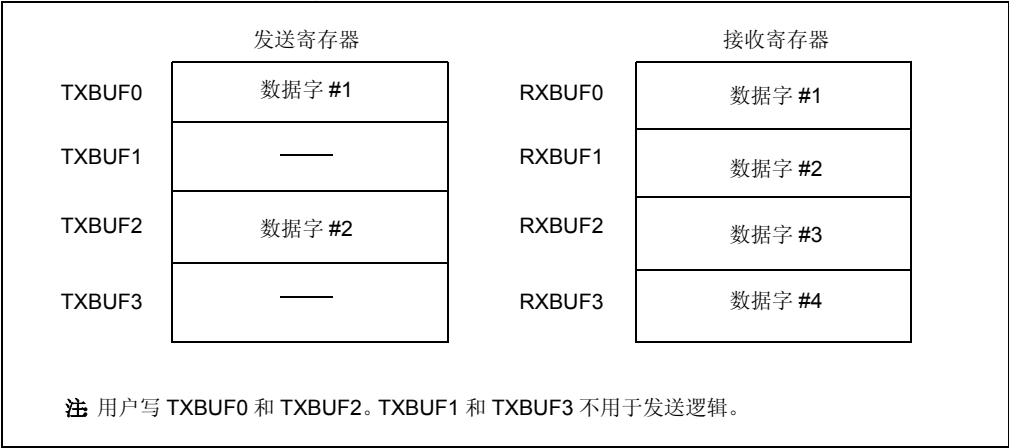
22.4.12 使用缓冲器控制单元进行 TSCON 和 RSCON 操作

除了缓冲控制逻辑以外，TSCON 和 RSCON 寄存器中的时隙使能位是独立工作的。对于数据帧中的每个时隙来说，如果当前时隙的 TSE_x 或 RSE_x 位置位，那么缓冲器单元将转到下一个单元。也就是说，缓冲器控制单元使得发送和接收缓冲同步，这样发送和接收缓冲器单元对于数据帧中的每个时隙来说总是相同的。

如果为数据帧中使用的每个时隙都置位相应的 TSE_x 位和 RSE_x 位，DCI 将发送和接收相等数量的数据。

在某些应用中，一个帧中发送的数据字数量与接收的数据字数量可能不相等。例如，假设 DCI 配置为双字数据帧传输，TSCON = 0x0001 且 RSCON = 0x0003。此配置允许 DCI 每帧发送一个数据字同时接收两个数据字。由于每发送一个数据字接收两个数据字，用户需每隔一个发送缓冲器单元写入一个数据字。明确地说，只有 TXBUF0 和 TXBUF2 可用于发送数据。

图 22-6: DCI 缓冲器操作: TSCON = 0x0001, RSCON = 0x0003, BLEN<1:0> = 11b



22.4.13 接收状态位

有两个接收状态位，RFUL 和 ROV。

接收状态位只表示由模块使能且需要使用的寄存器单元的状态。状态位的值与 BLEN<1:0> 控制位有关。如果缓冲器长度设置为小于四个字，未使用的缓冲器单元将不会影响接收状态位。

RFUL 状态位（DCISTAT<2>）是只读位，且表示接收寄存器中是否存在新数据。当所有使用的 RXBUF 寄存器被用户软件读取时，RFUL 位会自动清零。

ROV 状态位（DCISTAT<3>）是只读的，用来表示至少有一个接收寄存器单元发生了接收溢出。如果用户软件未读取 RXBUF 寄存器，这时有新数据从缓冲存储器传输到此寄存器，就会发生接收溢出。当发生接收溢出时，寄存器中的旧内容将被覆盖。当导致溢出的寄存器被读取时，ROV 状态位被自动清零。

22.4.14 发送状态位

有两个发送状态位，TMPTY 和 TUNF。

发送状态位只表示模块使用的寄存器单元的状态。例如，如果缓冲器长设置为小于四个字，未使用的寄存器单元将不会影响发送状态位。

TMPTY 位 (DCISTAT<0>) 为只读位，当有效 TXBUF 寄存器的内容被传输到发送缓冲寄存器时置位。TMPTY 位可以用软件查询以确定发送寄存器被写入的时间。当发生对任何在使用的 TXBUF 寄存器的写操作时，TMPTY 位由硬件自动清零。

TUNF 位 (DCISTAT<1>) 为只读位，用来表示在使用的发送寄存器至少有一个发生了发送下溢。当 TXBUF 寄存器的内容被传输到发送缓冲存储器，且从上一次缓冲传输完成以后用户没有写所有在使用的 TXBUF 寄存器，TUNF 位置位。当发生下溢的 TXBUF 寄存器被用户软件写入时，TUNF 状态位被自动清零。

22.4.15 SLOT 状态位

SLOT<3:0> 状态位 (DCISTAT<11:7>) 表示数据帧中的当前有效时隙，该状态位在当每个数据帧需要传输的数据超过四个字时很有用处。当发生 DCI 中断时用户可以用软件查询这些状态位以确定哪个时隙数据是最后接收到的以及哪个时隙数据应该被装入 TXBUF 寄存器。

22.4.16 数字环回模式

数字环回模式通过置位 DLOOP 控制位 (DCICON1<11>) 使能。当 DLOOP 位置位时，模块将在内部将 CSDO 信号连入 CSDI 引脚。在数字环回模式中将忽略 CSDI 引脚上实际输入的数据。

22.4.17 下溢模式控制位

当发生发送下溢时，根据 UNFM 控制位 (DCICON1<7>) 的状态模块会执行以下两种操作之一。如果 UNFM 位被清零 (默认)，在有效时隙期间，模块将在 CSDO 引脚上发送 0。在此工作模式中，连接到 DCI 模块的编解码器件将仅输入数字“静音”。如果 UNFM 控制位被置位，模块将发送上一次写入缓冲器单元的数据。该工作模式允许用户发送连续的数据值到编解码器件，而不需软件开销。

22.4.18 数据对齐控制

在大部分应用中，从采样到有效的 FS 信号之后的一个串行时钟周期开始数据传输。这是 DCI 模块默认的。可以通过设置 DJST 控制位 (DCICON1<5>) 来选择一种数据对齐方式。当 DJST = 1 时，在 FS 信号产生的串行时钟周期开始数据传输。

22.4.19 DCI 模块中断

DCI 模块中断的频率取决于有效时隙的数量 (由 TSCON 和 RSCON 寄存器控制)、数据帧长度 (由 WS 和 COFSG 控制位确定) 以及 BLEN 控制位。在以下情况下会产生中断：

- 当达到缓冲器长度时
- 当达到帧边界时

每当上述事件发生时都将发生缓冲存储器传输。缓冲存储器传输被定义为上一次写入 TXBUF 的值被传入发送缓冲存储器，及接收缓冲存储器新接收的值被传入 RXBUF 寄存器的时间。

22.5 使用 DCI 模块

本节说明了在指定类型的数据转换器中如何配置和使用 DCI。

22.5.1 如何使用 DCI 缓冲器、状态位及中断发送和接收数据

根据 **BLEN** 控制位的设置, DCI 可以在 CPU 中断之间最多缓冲四个数据字。根据 **TSCON** 和 **RSCON** 寄存器的设置, 可以在一个单独的数据帧中或通过多个数据帧发送和接收缓冲数据。例如, 假设 **BLEN<1:0> = 00b** (每个中断缓冲一个数据字) 且 **TSCON = RSCON = 0x0001**。此特定配置代表大多数基本设置, 且它会导致 DCI 在每个数据帧开始的时候发送/接收一个数据字。由于 **BLEN<1:0> = 00b**, 在每个数据字被发送/接收后, CPU 将被中断。

对于第二种配置, 假设 **BLEN<1:0> = 11b** (每个中断缓冲四个数据字) 且 **TSCON = RSCON = 0x0001**。此配置会导致 DCI 在每个数据帧的开始发送/接收一个数据字, 但在发送/接收了四个数据字之后才会产生一个 CPU 中断。此配置对于一次处理多个数据采样的块处理很有用处。

对于第三种配置, 假设 **BLEN<1:0> = 11b** (每个中断缓冲四个数据字) 且 **TSCON = RSCON = 0x000F**。此配置会导致 DCI 在每个数据帧开始的时候发送/接收四个数据字。这种情况下由于 DCI 被设置成在一个数据帧中缓冲四个数据字, 所以每个数据帧都会产生一个 CPU 中断。此配置代表典型的多通道缓冲设置。

DCI 也可以配置为每帧缓冲四个以上的数据字。例如, 假设 **BLEN<1:0> = 11b** (每个中断缓冲 4 个数据字) 且 **TSCON = RSCON = 0x00FF**。在此配置中, DCI 将在每个数据帧中发送/接收 8 个数据字。因此, 每个数据帧将产生两次中断。要确定每次中断时在发送/接收寄存器中的数据部分, 用户需要在中断服务程序中检查 **SLOT** 状态位 (**DCISTAT <11:7>**) 以确定当前的数据帧位置。

发送和接收寄存器是双缓冲的, 这样当用户软件处理其他组数据时, DCI 模块可以处理一组发送和接收数据。由于双缓冲, 需要用三个中断周期来接收数据、处理数据和发送处理过的数据。对于每个 DCI 中断来说, CPU 将处理前一个中断周期接收到的数据字并产生将在下一个中断周期发送出去的数据字。**dsPIC** 器件的缓冲和数据处理时间将在经过处理的数据之后插入两个中断周期延时。在大部分情况下, 此数据延时可被忽略。

DCI 状态标志位和 CPU 中断表示发生了缓冲器传输并表明此时 CPU 可以处理更多的数据。在典型应用中, 每当 DCI 数据被处理时都将发生以下步骤:

1. 用户软件读取 **RXBUF** 寄存器。**RFUL** 状态位 (**DCISTAT<2>**) 将由模块置位表示接收寄存器包含新数据。在所有有效接收寄存器被读取后, **RFUL** 位自动清零。
2. 用户软件将处理接收到的数据。
3. 处理过的数据被写入 **TXBUF** 寄存器。**TMPTY** 状态位 (**DCISTAT<0>**) 原来已被置位表示发送寄存器为更多的数据写入做好了准备。

对于配置为发送和接收数据 (**TSCON** 和 **RSCON** 不为零) 的应用, 可以在用户软件中查询 **RFUL** 和 **TMPTY** 状态位确定发生 DCI 缓冲器传输的时间。如果 DCI 只用于发送数据 (**RSCON = 0**), 则可以只查询 **TMPTY** 位表示缓冲器传输。如果 DCI 只用于接收数据 (**TSCON = 0**), 则可以只查询 **RFUL** 位表示缓冲器传输。

DCIIF 状态位 (**IFS2<9>**) 在发生 DCI 缓冲器传输且产生 CPU 中断 (如果允许的话) 的时候置位。**DCIIF** 状态位的值由 **RFUL** 和 **TMPTY** 状态位逻辑“或”生成。

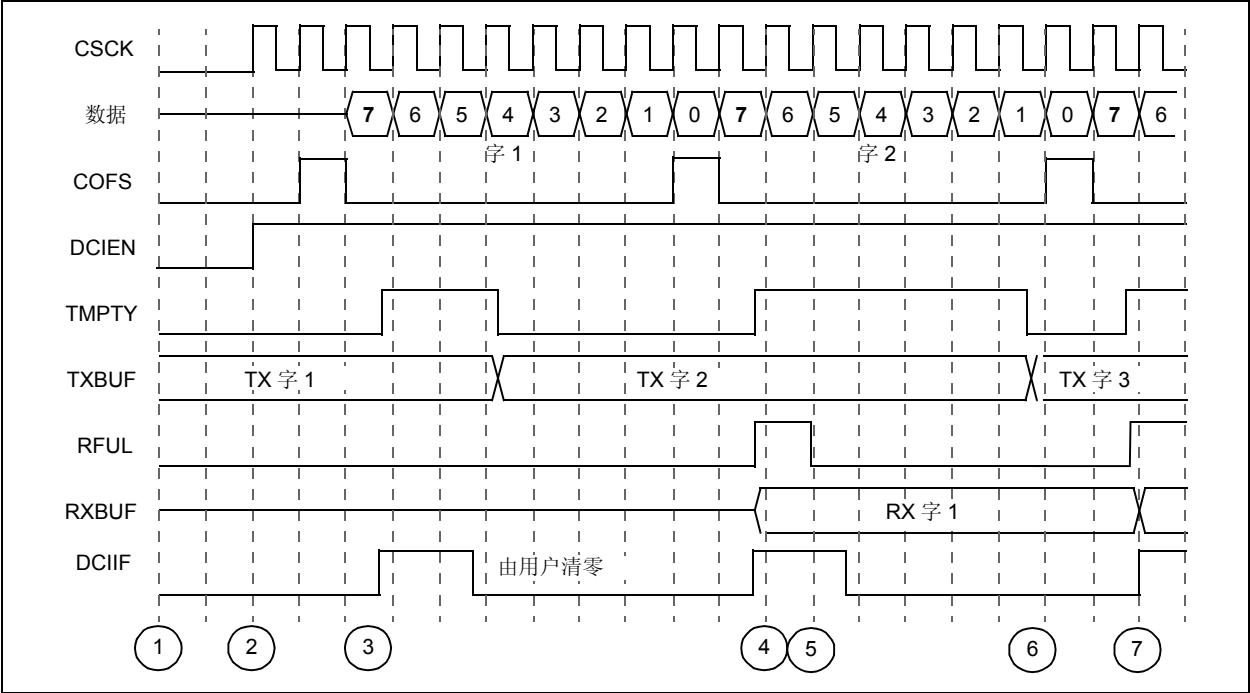
22.5.1.1 DCI 启动和数据缓冲

数据传输始于置位 DCIEN 控制位 (DCICON1<15>)。在此之前, 应该为需要的工作模式初始化 DCI 控制寄存器。(参见第 22.5.4 节“多通道工作”、第 22.5.5 节“I2S 工作”和第 22.5.6 节“AC-Link 工作”)

图 22-7 所示为 DCI 启动的时序图。在此示例中, DCI 配置为 8 位数据字 (WS<3:0> = 0111b) 和 8 位数据帧 (COFSG<3:0> = 0000b) 输出模式。使用多通道模式 (COFSM<1:0> = 00b)。下面描述了发送和接收数据需要的步骤。

1. 应该在模块使能前将要发送的第一个数据预装载到 TXBUF 寄存器中。如果发送数据基于从编解码器接收到的数据, 则用户可以仅清零 TXBUF 寄存器。这将发送数字“静音”直到数据第一次从编解码器接收到 RXBUF 寄存器为止。
2. 置位 DCIEN 位 (DCICON1<15>) 使能 DCI 模块。如果 DCI 为主器件, TXBUF 寄存器中的数据将被传输到发送缓冲器且开始第一个数据帧的发送。否则, TXBUF 数据将被保存在发送缓冲器中直到从主器件接收到一个帧同步信号。
3. 在模块被使能且 DCI 中断 (如果允许的话) 产生后, TMPTY 位将立即置位。此时, 模块准备好让 TXBUF 寄存器装入在第二个数据帧上传输的数据。此时模块不接收数据, 所以如果被发送的数据要由接收到的数据推导出, 那么 TXBUF 寄存器将被再次清零。如果允许中断, 用户应该用软件清零 DCIIF 状态位。
4. 在传输了第一个数据帧之后, TMPTY 位将置位、RFUL 状态位将置位且产生 DCI 中断 (如果允许的话)。这是从连接到 DCI 的器件接收到的第一个数据。
5. 用户读接收寄存器, 自动清零 RFUL 状态位。此时用户软件处理接收到的数据。
6. 把在下一个数据帧中要发送的数据写到发送寄存器。当发生写入时, TMPTY 状态位会自动清零。要写入的数据可能由上一个中断接收的数据推导出。
7. 产生下一个 DCI 中断并重复该过程。

图 22-7: DCI 启动和数据缓冲示例



22.5.1.2 DCI 禁止

通过清零 DCIEN 控制位 (DCICON1<15>) 禁止 DCI 模块。当 DCIEN 位被清零时, 模块将完成正在进行的当前数据帧传输。如果需要在帧结束之前写 / 读缓冲器, 会产生中断。

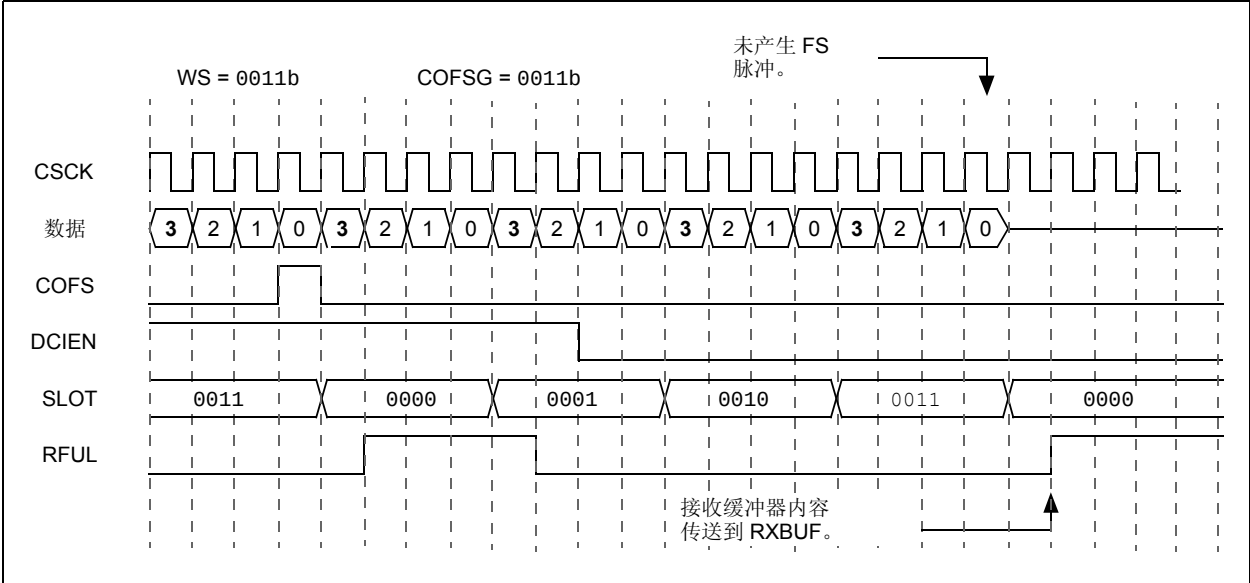
要在当前帧禁止模块, 则 DCIEN 位必须在该帧结束前至少三个 CSCK 周期被清零。如果不是这样的话, 模块将在下一个帧被禁止。

DCI 在 DCIEN 位清零后将不再产生任何帧同步脉冲, 也不会对任何进入的帧同步脉冲作出响应。

当帧同步发生器达到数据帧中的最后时隙时, 所有与 DCI 相关的状态机将复位到空闲状态, 且将释放与该模块相关联的 I/O 引脚的控制。在 DCIEN 位被清零后用户可以查询 SLOT<3:0> 状态位 (DCISTAT<11:7>) 以确定模块进入空闲模式的时间。当 SLOT<3:0> = 0000b 且 DCIEN = 0 时, DCI 进入空闲模式。

当模块进入空闲状态时, 所有接收影子寄存器中的数据都将被传送到 RXBUF 寄存器, 且 RFUL 和 ROV 状态位也将随之受到影响。

图 22-8: DCI 时序, 模块禁止



22.5.2 主控和从动工作模式比较

DCI 可以配置为主控或从动工作模式。主器件产生帧同步信号以启动数据传输。工作模式（主控或从动）由 COFSD 控制位（DCICON1<8>）选择。

当 DCI 模块作为主器件（COFSD = 0）工作时，COFSM 模式位确定帧同步发生器逻辑产生的帧同步脉冲的类型。当帧同步发生器复位时，将产生新的帧同步信号，并在 COFS 引脚上输出。

当 DCI 模块作为帧同步从器件（COFSD = 1）工作时，数据传输由与 DCI 模块相连的器件控制。COFSM 控制位控制 DCI 模块如何响应输入的 FS 信号。

在多通道模式中，在 COFS 引脚采样到高电平之后的一个串行时钟周期将会开始新的数据帧传输。COFS 引脚上的脉冲会复位帧同步发生器逻辑。

在 I²S 模式中，在 COFS 引脚上采样到从低到高或从高到低的电平转变后的一个串行时钟周期，新的数据字会被传输。COFS 引脚的上升或下降沿将复位帧同步发生器逻辑。

在 AC-Link 模式中，当在 COFS 引脚采样到高电平之后的一个串行时钟周期，下一个帧的标记间隙和接下来的数据间隙将被传输。

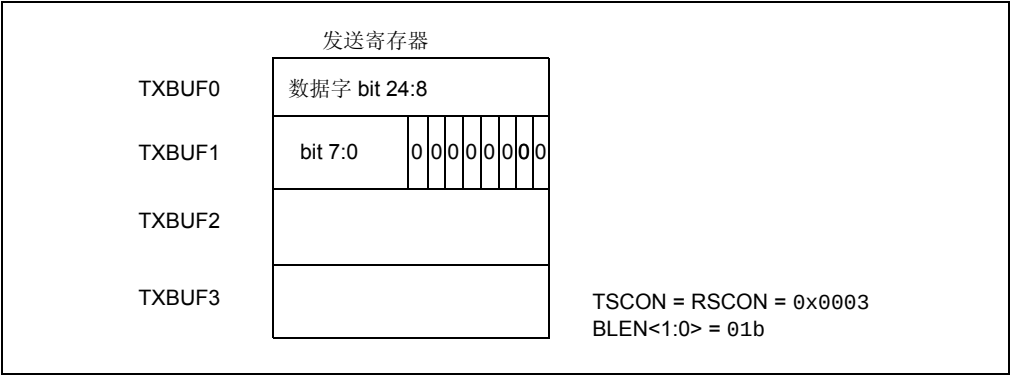
当模块工作在从动模式时，必须配置 COFSG 和 WS 位以提供期望的帧长度。一旦模块在 COFS 引脚上采样到了有效的帧同步脉冲，就会产生一个完整的数据帧传输。在当前数据帧传输彻底完成之前模块都不会响应其他帧同步脉冲。

22.5.3 支持长数据字的数据打包

很多编解码器的数据字长度都超过了 16 位。DCI 本身支持的字长度最多 16 位，但通过使能多个发送和接收时隙并将数据打包到多个发送和接收缓冲单元可以支持更长的字长度。例如，假设一种特定的编解码器可以发送 / 接收 24 位数据字。通过设置 BLEN<1:0> = 01b（每个中断两个数据字）和 TSCON = RSCON = 0x0003，可以发送和接收此数据。这样可以在数据帧的前两个时隙使能发送和接收。要发送数据的高 16 位被写入 TXBUF0。如图 22-9 所示，要发送数据的低 8 位被左对齐写入 TXBUF1。TXBUF1 的 8 Lsb 可被写为 0。从编解码器接收到的 24 位数据将用与发送数据相同的格式被装入 RXBUF0 和 RXBUF1。

任何字长和使能时隙的组合都可以用于在多个发送和接收寄存器中发送和接收长数据字。例如，图 22-9 中所示的例子，通过设置 WS<3:0> = 0111（字长 = 8 位）、BLEN<1:0> = 10（在中断之间缓冲三个字）和 TSCON = RSCON = 0x0007（在数据帧的前三个时隙发送 / 接收），可以在三个连续的寄存器中发送 / 接收 24 位数据字。每个发送和接收寄存器都包含数据字的 8 位。

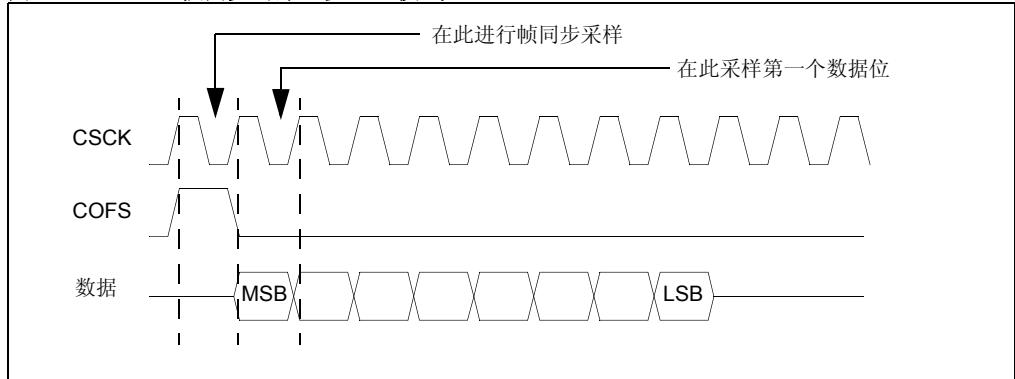
图 22-9: 长数据字的数据打包示例



22.5.4 多通道工作

多通道模式 ($\text{COFSM}<1:0> = 00$) 用于需要帧同步脉冲被驱动为高电平的时间为一个串行时钟周期才能启动数据传输的编解码器。在一个数据帧中, 可以传输一个或多个数据字。连接到 DCI 模块的器件决定连续帧同步脉冲之间的时钟周期数量。图 22-10 所示为多通道模式中的帧同步信号的时序图。图 22-2 还给出了表示四字数据传输的时序示例。

图 22-10: 帧同步时序, 多通道模式



22.5.4.1 多通道设置细节

本节提供了使用多通道模式为编解码器配置 DCI 所需要的步骤。此工作模式可以用于单数据通道或多数据通道编解码器。同样, 这些步骤与通道数无关。

作为设置示例, 假定一个编解码器, 在此设置示例中使用的单通道编解码器将使用 256 fs 的串行时钟频率, 并在每个帧开始的时候发送 16 位数据字。

以下描述了设置和工作所需要的步骤。

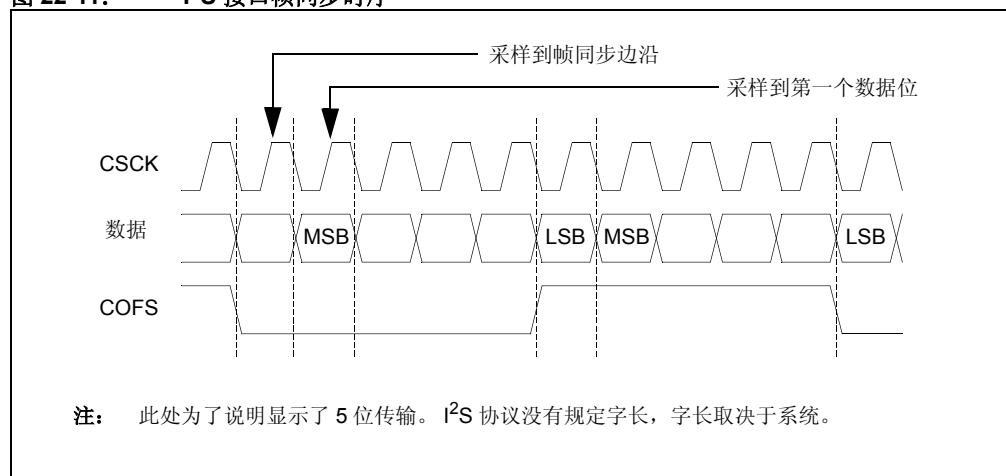
1. 确定编解码器所需要的采样率和数据字长。此示例中假设采样率为 8 kHz。
2. 确定编解码器要求的串行传输时钟频率。大部分编解码器要求串行时钟的频率为采样频率的整数倍。此示例编解码器需要的频率为 256 fs 或 1.024 MHz。因此, 在每 256 个串行时钟周期必须产生一个帧同步脉冲以启动数据传输。
3. 必须配置 DCI 的串行传输时钟。如果 CCLK 信号由 DCI 产生, 清零 CSCDK 控制位 ($\text{DCICON1}<10>$) 并将值写入 DCICON3, 以产生正确时钟频率 (参见第 22.4.3 节 “位时钟发生器”)。如果编解码器或其他外部时钟源产生 CCLK 信号, 置位 CCLKD 控制位并清零 DCICON3 寄存器。
4. 清零 COFSM<1:0> 控制位 ($\text{DCICON1}<1:0>$), 将帧同步信号设置为多通道模式。
5. 如果将由 DCI 产生帧同步信号 (主器件), 则清零 COFSD 控制位 ($\text{DCICON1}<8>$)。如果 DCI 将接收帧同步信号 (从器件), 则置位 COFSD 控制位。
6. 清零 CSCKE 控制位 ($\text{DCICON1}<9>$), 以在 CCLK 下降沿采样输入数据。这是对于大部分编解码器的典型配置。要确保使用了正确的采样边沿, 请参见编解码器数据手册。
7. 写 WS<3:0> 控制位 ($\text{DCICON2}<3:0>$) 以设置需要的数据字长。比如, 对于 16 位数据字长, 本例中的编解码器要求 $\text{WS}<3:0> = 1111\text{b}$ 。

8. 写 COFSG<3:0> 控制位 (DCICON2<8:5>) 设置每帧需要传输的数据字数。WS 和 COFSG 控制位将确定 CSCK 周期中数据帧的长度 (参见第 22.4.7 节“帧同步发生器”)。COFSG<3:0> = 1111b 用于提供示例编解码器需要的 256 位数据帧。
9. 使用 CSDOM 控制位 (DCICON1<6>) 将 CSDO 引脚设置为输出模式。如果只有一个器件连接到 DCI, CSDOM 将被清零。这将强制 CSDO 引脚在未使用的数据间隙内为 0。如果有多个器件连接到 CSDO 引脚, 此时可能需要置位 CSDOM。
10. 写 TSCON 和 RSCON 寄存器以分别确定帧中的哪些数据间隙将被发送和接收。对于单通道编解码器, 使用 TSCON = RSCON = 0x0001 使能数据帧中第一个 16 位间隙中的发送和接收。
11. 设置 BLEN 控制位 (DCICON2<11:10>) 缓冲需要的数据字数。对于单通道编解码器, BLEN = 00 将在每个数据帧提供一次中断。可对此编解码器使用值更高的 BLEN 以便在中断之间缓冲多个采样。
12. 如果需要使用中断, 则清零 DCIIF 状态位 (IFS2<9>) 并置位 DCIIE 控制位 (IEC2<9>)。
13. 如第 22.5.1.1 节“DCI 启动和数据缓冲”中描述的那样开始工作。

22.5.5 I²S 工作

I²S 工作模式用于需要具有 50% 占空比的帧同步信号的编解码器。串行时钟周期中的 I²S 帧同步信号周期由连接到 DCI 模块的编解码器字长来确定。COFS 引脚上的从高到低或从低到高转变沿标记新字开始的边界, 如图 22-11 所示。I²S 编解码器通常为立体的或双通道器件, 一个数据字在帧同步信号的低电平时间传输, 而另一个数据字在高电平时间传输。

图 22-11: I²S 接口帧同步时序



通过将值 01h 写入 DCICON1 SFR 中的 COFSM<1:0> 控制位将 DCI 模块配置为 I²S 模式。在 I²S 模式中工作时, DCI 模块将产生占空比为 50% 的帧同步信号。帧同步信号的每个边沿都将标记新的数据字传输边界。有关 I²S 协议的更多信息, 请参见本手册的附录。用户也必须使用 DCICON2 SFR 中的 COFSG 和 WS 控制位选择帧长度和数据字长。

22.5.5.1 I²S 设置细节

本节中提供了为 I²S 编解码器配置 DCI 所需的步骤。对本设置示例，假定一个 I²S 编解码器。

此设置示例中的 I²S 编解码器将使用 64 fs 串行时钟频率，在每个数据帧中传输两个 16 位数据字。因此，帧长度为 64 个 CSCK 周期，其中 32 周期为 COFS 信号高电平，32 周期为 COFS 信号低电平。第一个数据字将在 COFS 下降沿之后的一个 CSCK 周期发送，第二个数据字将在 COFS 上升沿之后的一个 CSCK 周期发送，如图 22-11 所示。

1. 确定编码器使用的采样率以确定 CSCK 频率。在此示例中假设 fs 为 48 kHz。
2. 确定编解码器要求的串行传输时钟频率。此示例编解码器需要的频率为 64 fs 或 3.072 MHz。
3. 必须配置 DCI 的串行传输时钟。如果将由 DCI 产生 CSCK 信号，则清零 CSCDK 控制位 (DCICON1<10>)，并将值写入 DCICON3 以产生正确的时钟频率（参见第 22.4.3 节“位时钟发生器”）。如果编解码器或其他外部时钟源产生 CSCK 信号，置位 CSCKD 控制位并清零 DCICON3 寄存器。
4. 然后，设置 COFSM<1:0> = 01b 将帧同步信号设置为 I²S 模式。
5. 如果将由 DCI 产生帧同步信号（主器件），则清零 COFSD 控制位 (DCICON1<8>)。如果 DCI 将接收帧同步信号（从器件），则置位 COFSD 控制位。
6. 清零 CSCKE 控制位 (DCICON1<9>) 以在 CSCK 上升沿采样输入数据。这是大部分 I²S 编解码器的典型配置。
7. 写 WS<3:0> 控制位 (DCICON2<3:0>) 以设置需要的数据字长。示例编解码器要求 WS<3:0> = 1111b 以进行 16 位数据字长的传输。
8. 写 COFSG<3:0> 控制位 (DCICON2<8:5>) 以设置每帧需要的数据字数量。WS 和 COFSG 控制位将决定 CSCK 周期（参见第 22.4.7 节“帧同步发生器”）中的数据帧长度。对于此示例编解码器，设置 COFSG<3:0> = 0001b。

注： 在 I²S 模式中，COFSG 位被设置为 1/2 数据帧长度。对于此示例编解码器，设置 COFSG<3:0> = 0001b（每帧两个数据字）以便产生 32 位帧。从而将产生长度为 64 位的 I²S 数据帧。

9. 使用 CSDOM 控制位 (DCICON1<6>) 将 CSDO 引脚设置为输出模式。如果只有一个器件连接到 DCI，CSDOM 将被清零。如果有多个器件连接到 CSDO 引脚，此时可能需要置位 CSDOM。
10. 写 TSCON 和 RSCON 寄存器以分别确定帧中的哪些数据时隙将被发送和接收。对于该编解码器，设置 TSCON = 0x0001 和 RSCON = 0x0001 以便使能在 32 位数据帧的第一个 16 位时隙中的发送和接收。可以使能相邻的时隙以缓冲长于 16 位的数据字。
11. 设置 BLEN<1:0> 控制位 (DCICON2<11:10>) 以缓冲需要的数据字个数。对于双通道 I²S 编解码器，BLEN<1:0> = 01b 将在传输了两个数据字后产生中断。
12. 如果需要使用中断，则清零 DCIIF 状态位 (IFS2<9>) 并置位 DCIIE 控制位 (IEC2<9>)。
13. 如第 22.5.1.1 节“DCI 启动和数据缓冲”中描述的那样开始工作。在 I²S 主控模式中，COFS 引脚将在模块被使能之后驱动为高电平，并开始发送装入 TXBUF0 的数据。

22.5.5.2 确定 I²S 通道对齐的方式

大部分 I²S 编解码器支持两个数据通道，帧同步信号电平表明数据帧传输过程中正在传输的通道。可使用相关的端口寄存器在 DCI 中断服务程序中用软件查询 COFS 引脚以确定 DCI 引脚上的电平。这将指出哪个数据处于接收寄存器中、哪个数据应该写入发送寄存器并在下一个帧传输。

22.5.5.3 I²S 数据对齐

根据 I²S 规范，在默认情况下，数据字传输开始于帧同步信号跳变之后的一个串行时钟周期。可以使用 DJST 控制位（DCICON1<5>）选择“MSb 左对齐”选项。

如果 DJST = 1，则 I²S 数据传输为 MSb 左对齐。在 FS 信号的上升或下降沿的串行时钟周期期间，数据字的 MSb 将出现在 CDSO 引脚上。在数据字被发送后，由 CSDOM（DCICON1<6>）位规定 CDSO 引脚的状态。

左对齐的数据选项允许将两个立体编解码器连接到同一个串行总线上。很多 I²S 兼容的器件都有数据左对齐和右对齐的配置选项。字长选择位将被设置为编解码器字长度的两倍，且数据将以打包的格式读 / 写到 DCI 存储器。图 22-12 所示为双 I²S 编解码器系统的连接细节。

图 22-13 所示为 I²S 模式的时序图。作为参考，这些图假设传输的数据字长为 8 位（WS<3:0> = 0111b）。要得到一个 16 位的子帧（COFSG<3:0> = 0001b），每帧需要两个数据字。图 22-13 中的第三个时序图使用打包的数据格式从两个编解码器中读 / 写数据字。在此示例中，DCI 模块被配置为 16 位数据字传输（WS<3:0> = 1111b）。在 DCI 存储缓冲器中的每个 16 位单元都写入两个打包的 8 位字。

图 22-12: 双 I²S 编解码器接口

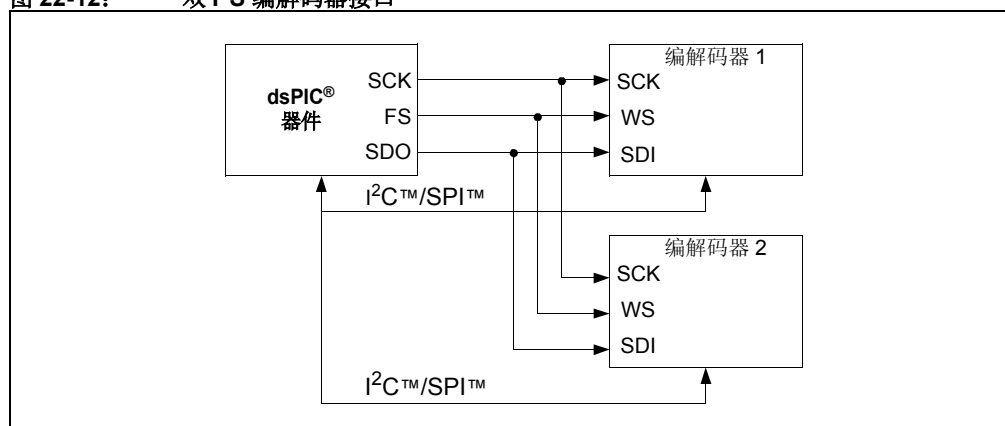
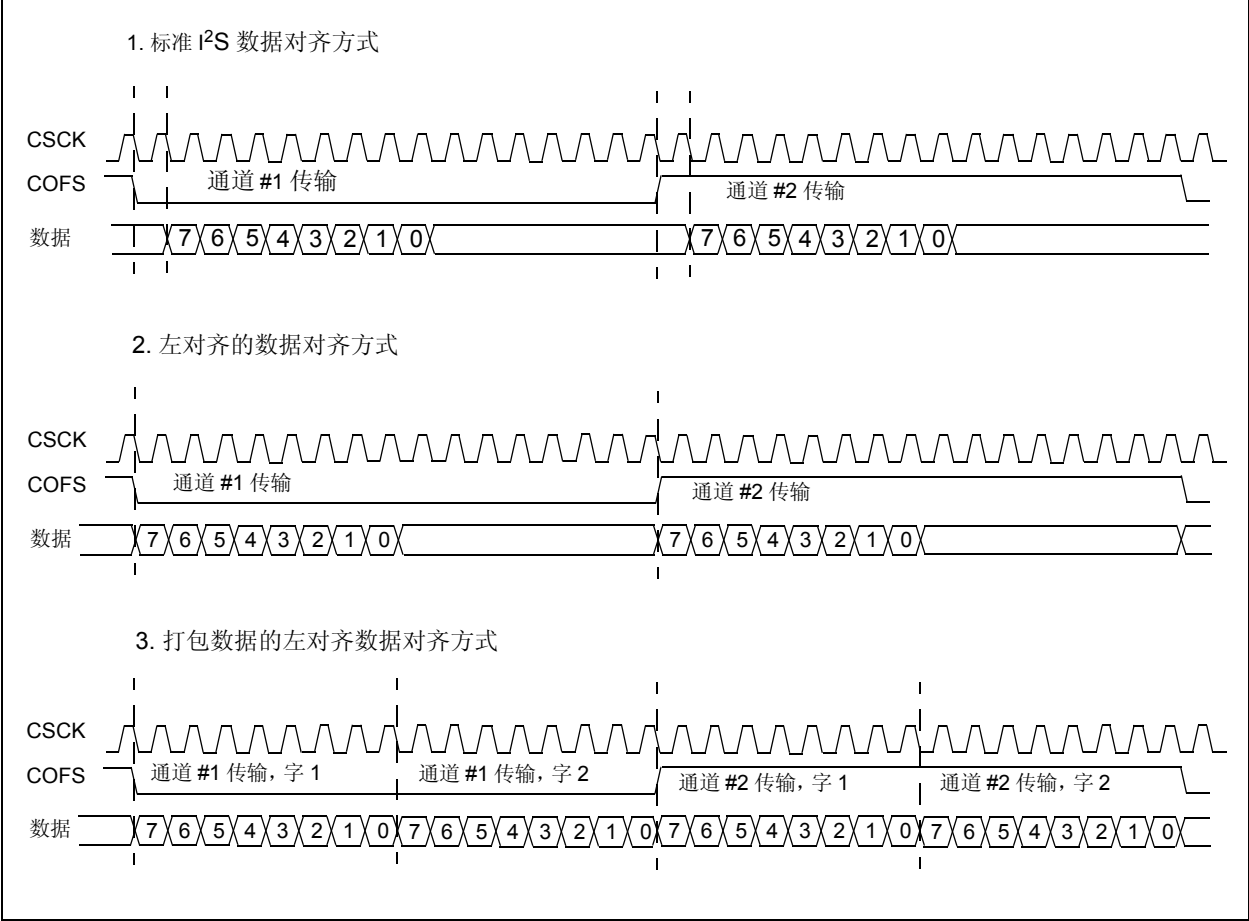


图 22-13: I²S 数据对齐选项



22.5.6 AC-Link 工作

本节介绍如何在 AC-Link 模式中使用 DCI。AC-Link 模式用于与符合 AC '97 的编解码器器件进行通信。

22.5.6.1 AC-Link 数据帧

AC-Link 数据帧是 256 位的，它可以被分成一个 16 位控制间隙和其后的 12 个 20 位控制间隙。AC'97 编解码器通常提供由晶振产生的串行传输时钟信号，如图 22-14 所示。控制器接收串行时钟并产生帧同步信号。默认数据帧频率为 48 kHz。用于 AC-Link 系统的帧同步信号在数据帧开始的时候为 16 个 CSCK 周期的高电平、240 个 CSCK 周期的低电平。如图 22-16 所示，数据传输在帧同步信号上升沿后的一个 CSCK 周期开始。接收器件在 CSCK 的下降沿采样数据。如图 22-15 所示，定义了 AC-Link 中的控制和数据间隙在协议中的使用。AC-Link 协议的完整定义，请参见本手册的附录或 Intel® AC '97 编解码器规范（Intel® AC '97 Codec Specification）2.2 版。

图 22-14: AC-Link 信号连接

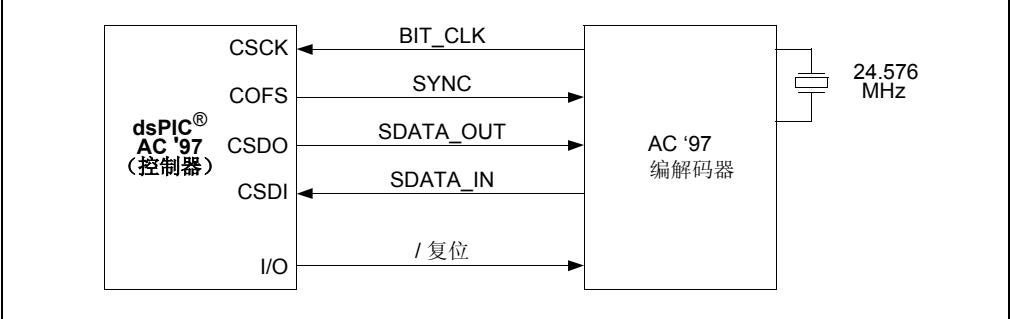


图 22-15: AC-Link 数据帧

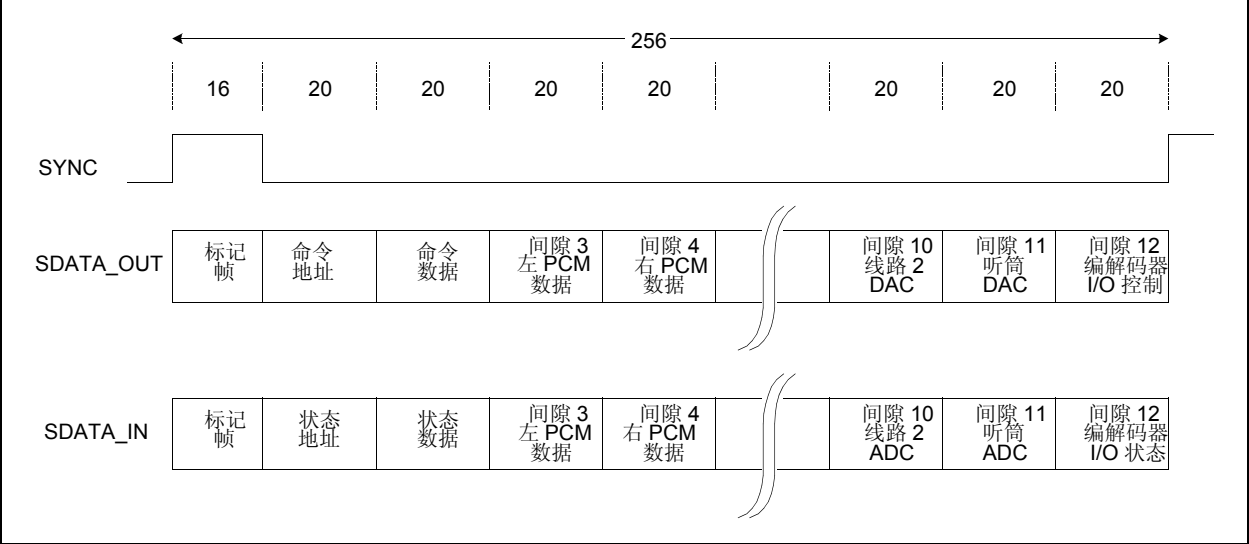
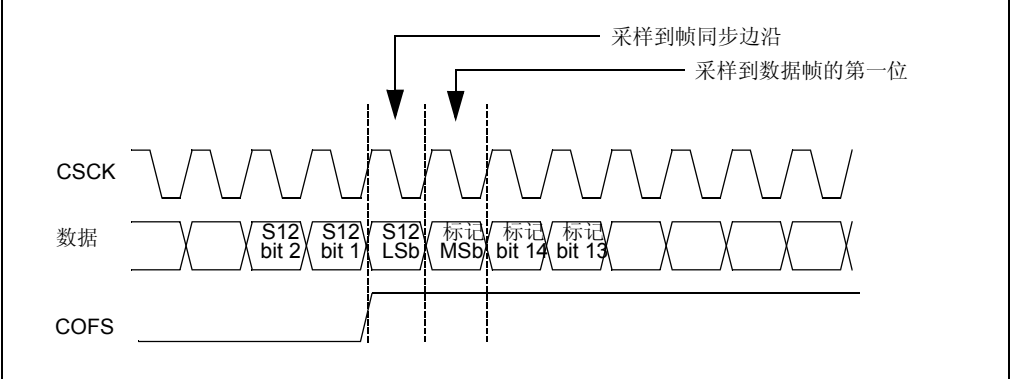


图 22-16: 帧同步时序，AC-Link 帧起始



DCI 模块有两种让 AC-Link 协议容纳 20 位数据时隙的工作模式。这两种工作模式由 COFSM<1:0> 控制位 (DCICON1<1:0>) 选择。第一种 AC-Link 模式称为 “16 位 AC-Link 模式”，并通过设置 COFSM<1:0> = 10b 选定。第二种 AC-Link 模式称为 “20 位 AC-Link 模式”，并通过设置 COFSM<1:0> = 11b 选定。

22.5.6.2 16 位 AC-Link 模式

在 16 位 AC-Link 模式下, 发送和接收数据字长度被限制为 16 位以适合 DCI 发送和接收寄存器。注意此限制仅影响 AC-Link 协议的 20 位数据时隙。对于接收时隙, 输入数据将被截断为 16 位。对于输出时隙, 模块将数据字的 4 个 LSB 置为 0。该工作模式通过把每个时隙当作 16 位时隙处理, 简化了 AC-Link 数据帧。帧同步发生器保持与时隙边界对齐。

22.5.6.3 20 位 AC-Link 模式

20 位 AC-Link 模式允许发送和接收数据时隙中的所有位, 但是数据与 AC-Link 协议中定义的特定时隙边界不保持对齐。

除了产生的帧同步信号的占空比不同之外, 20 位 AC-Link 模式的工作方式类似于 DCI 模块的多通道模式。AC-Link 帧同步信号应该保持 16 个时钟周期的高电平并在接着的 240 个时钟周期保持低电平。

20 位模式把每个 256 位 AC-Link 帧当作 16 个 16 位时隙处理。在 20 位 AC-Link 模式下, 该模块像在 $\text{COFSG}\langle 3:0 \rangle = 1111\text{b}$ 且 $\text{WS}\langle 3:0 \rangle = 1111\text{b}$ 的条件下工作。在该工作模式下, 20 位数据间隙中的数据不保持对齐。例如, 通过将 TSCON 和 RSCON 寄存器中的所有位置 1, 整个 256 位 AC-Link 数据帧能够以打包的方式发送和接收。由于可用缓冲器的总长度为 64 位, 所以传输该 AC-Link 帧需要经过四次连续的中断。应用软件必须通过监视 $\text{SLOT}\langle 3:0 \rangle$ 状态位 ($\text{DCISTAT}\langle 11:7 \rangle$) 时刻跟踪当前 AC-Link 帧段。

22.5.6.4 AC-Link 设置详情

通过将 10h 或 11h 写入 DCICON1 SFR 中的 $\text{COFSM}\langle 1:0 \rangle$ 控制位, 将该模块使能为 AC-Link 模式。字长选择位 ($\text{WS}\langle 3:0 \rangle$) 和帧同步发生器位 ($\text{COFSG}\langle 3:0 \rangle$) 对 16 位和 20 位 AC-Link 模式没有影响, 因为帧和字的大小由协议设定。

大多数 AC'97 编解码器产生控制数据传输的时钟信号。因此, 要在软件中将 CSCKD 控制位置 1。COFSD 控制位会被清零, 因为 DCI 将通过输入时钟信号产生 FS 信号。将 CSCKE 位清零以便在上升沿采样数据。

用户必须决定要将 AC-Link 数据帧中的哪一个时隙缓冲, 并相应设置控制位 TSE 和 RSE。至少必须缓冲发送和接收标记间隙, 所以应该在软件中将 TSCON<0> 和 RSCON<1> 控制位置 1。

注: 只有 TSCON<12:0> 控制位和 RSCON<12:0> 控制位将会对 16 位 AC-Link 模式有影响, 因为 AC-Link 帧有 13 个时隙。

1. 必须配置 DCI 以接受来自 AC'97 编解码器的串行传输时钟。将 CSCKD 控制位置 1, 并清零 DCICON3 寄存器。
2. 然后, 将 $\text{COFSM}\langle 1:0 \rangle$ 控制位 ($\text{DCICON1}\langle 1:0 \rangle$) 设置为 10b 或 11b 以设置所需的 AC-Link 帧同步模式。
3. 清零 COFSD 控制位 ($\text{DCICON1}\langle 8 \rangle$), 从而 DCI 将输出帧同步信号。
4. 清零 CSCKE 控制位 ($\text{DCICON1}\langle 9 \rangle$), 从而在 CSCK 的下降沿采样输入数据。

注: 字长选择位 ($\text{WS}\langle 3:0 \rangle$) 和帧同步发生器位 ($\text{COFSG}\langle 3:0 \rangle$) 对 16 位和 20 位 AC-Link 模式没有影响, 因为帧和字的大小由协议设定。

5. 清零 CSDOM 控制位 (DCICON1<6>)。
6. 写 TSCON 和 RSCON 寄存器以分别确定帧中的哪些数据时隙将被发送和接收。这取决于 AC-Link 协议中哪些数据时隙将要被使用。至少, 在 Slot #0 (标记间隙) 上的通信是必需的。欲获取更多信息, 请参见本手册第 22.5.6.2 节 “16 位 AC-Link 模式”、第 22.5.6.3 节 “20 位 AC-Link 模式” 和第 26 章 “附录” 中的讨论。
7. 设置 BLEN 控制位 (DCICON2<11:10>), 以缓冲所需数量的数据字。对于单通道编解码器, BLEN = 00 会在每个数据帧期间提供一个中断。在此编解码器中可以使用值更高的 BLEN, 以便在中断之间缓冲多个采样。
8. 如果需要使用中断, 则清零 DCIIF 状态位 (IFS2<9>), 并置位 DCIIE 控制位 (IEC2<9>)。
9. 如第 22.5.1.1 节 “DCI 启动和数据缓冲” 中描述的那样开始工作。

22.6 低功耗模式下的工作

22.6.1 CPU 空闲模式

DCI 模块可以选择在 CPU 处于空闲模式时继续工作。DCISIDL 控制位 (DCICON1<13>) 决定 DCI 模块在 CPU 处于空闲模式时是否工作。如果 DCISIDL 控制位清零 (默认), 该模块将在空闲模式下继续正常工作。如果 DCISIDL 位置 1, 当 CPU 进入空闲模式时, 该模块将停止工作。

22.6.2 休眠模式

如果 CSMC 信号来自器件指令时钟 Tcy, 那么当器件处于休眠模式时, DCI 将不工作。

然而, 当 CSMC 信号由外部器件提供 (CSCKD = 1) 时, DCI 模块在器件处于休眠模式时能够工作并唤醒 CPU。必须将 DCI 中断使能位 DCIIE 置位, 以使器件能够从休眠模式唤醒。当 DCI 中断标志位 DCIIF 置 1 时, 器件将从休眠模式唤醒。如果 DCI 中断的优先级高于当前 CPU 的优先级, 将转到 DCI ISR 程序执行。否则, 将继续执行在先前进入休眠模式的 PWRSAV 指令后的指令。

22.7 与 DCI 相关的寄存器

表 22-1 列出了与 DCI 模块相关的寄存器。

表 22-1: DCI 寄存器映射

名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的值	
IFS2	0088	—	—	—	FLTBIF	FLTAIF	LVDIF	DCIIF	QEIIIF	PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF	0000 0000 0000 0000	
IEC2	0090	—	—	—	FLTBIE	FLTAIE	LVDIE	DCIIE	QEIIIE	PWMIIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE	0000 0000 0000 0000	
IPC10	00A8	—	FLTAIP<2:0>			—	LVDIP<2:0>			—	DCIIP<2:0>			—	QEIIIP<2:0>			0100 0100 0100 0100	
DCICON1	240	DCIEN	—	DCISIDL	—	DLOOP	CSCKD	CSCKE	COFSD	UNFM	SDOM	DJST	—	—	—	COFSM<1:0>		000- -000 000- --00	
DCICON2	242	—	—	—	—	BLEN<1:0>		—	COFSG<3:0>				—	WS<3:0>				---- 00-0 000- 0000	
DCICON3	244	—	—	—	—	BCG<11:0>													---- 0000 0000 0000
DCISTAT	246	—	—	—	—	SLOT<3:0>				—	—	—	—	ROV	RFUL	TUNF	TMPTY	---- 0000 ---0 0000	
TSCON	248	TSE15	TSE14	TSE13	TSE12	TSE11	TSE10	TSE9	TSE8	TSE7	TSE6	TSE5	TSE4	TSE3	TSE2	TSE1	TSE0	0000 0000 0000 0000	
RSCON	24C	RSE15	RSE14	RSE13	RSE12	RSE11	RSE10	RSE9	RSE8	RSE7	RSE6	RSE5	RSE4	RSE3	RSE2	RSE1	RSE0	0000 0000 0000 0000	
RXBUF0	250	接收 #0 数据寄存器																uuuu uuuu uuuu uuuu	
RXBUF1	252	接收 #1 数据寄存器																uuuu uuuu uuuu uuuu	
RXBUF2	254	接收 #2 数据寄存器																uuuu uuuu uuuu uuuu	
RXBUF3	256	接收 #3 数据寄存器																uuuu uuuu uuuu uuuu	
TXBUF0	258	发送 #0 数据寄存器																0000 0000 0000 0000	
TXBUF1	25A	发送 #1 数据寄存器																0000 0000 0000 0000	
TXBUF2	25C	发送 #2 数据寄存器																0000 0000 0000 0000	
TXBUF3	25E	发送 #3 数据寄存器																0000 0000 0000 0000	

图注: r= 保留, x= 未知, u= 不变。
注: 灰色的单元表示 SFR 映射中的保留空间, 用于将来模块的扩展。保留单元读作 0。

22.8 设计技巧

问 1: *DCI 能支持大于 16 位的数据字长度吗?*

答: 可以。通过使用多个发送和接收寄存器, 可以发送和接收长数据字。更多详细信息, 参见第 22.5.3 节 “支持长数据字的数据打包”。

22.9 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与数据转换器接口（DCI）模块相关的应用笔记有：

标题	应用笔记编号
目前没有相关的应用笔记。	

注：如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

22.10 版本历史

版本 A

这是本文档的初始发行版。

版本 B

此版本加入了 dsPIC30F 数据转换器接口（DCI）模块的更多技术内容和更改。

版本 C

此版本加入了本文档更新时已知的所有勘误表。

第 23 章 CAN 模块

目录

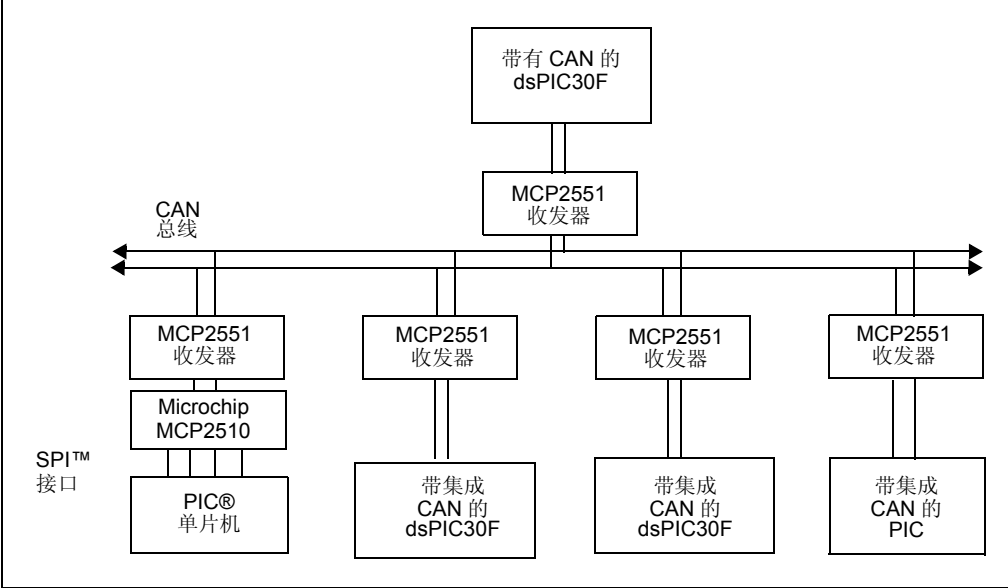
本章包括下列主题：

23.1 简介	23-2
23.2 CAN 模块的控制寄存器	23-2
23.3 CAN 模块特点	23-28
23.4 CAN 模块的实现	23-29
23.5 CAN 模块工作模式	23-36
23.6 报文接收	23-39
23.7 发送	23-49
23.8 错误检测	23-58
23.9 CAN 波特率	23-60
23.10 中断	23-64
23.11 时间标记	23-65
23.12 CAN 模块 I/O	23-65
23.13 CPU 低功耗模式下的工作	23-66
23.14 CAN 协议综述	23-68
23.15 相关应用笔记	23-72
23.16 版本历史	23-73

23.1 简介

控制器局域网（Controller Area Network，CAN）模块是一个串行接口，可用于与其他外设或者单片机之间进行通信。此接口/协议是针对允许在噪声环境下通信而设计的。图 23-1 所示为 CAN 总线网络示例。

图 23-1: CAN 总线网络示例



23.2 CAN 模块的控制寄存器

与 CAN 模块相关的寄存器很多。这些寄存器被分成以下几部分进行说明。它们是：

- 控制和状态寄存器
- 发送缓冲寄存器
- 接收缓冲寄存器
- 波特率控制寄存器
- 中断状态和控制寄存器

注 1: 寄存器标识符中的“i”表示特定的 CAN 模块（CAN1 或 CAN2）。
2: 寄存器标识符中的“n”表示缓冲器、过滤器或屏蔽器编号。
3: 寄存器标识符中的“m”表示特定 CAN 数据字段中的字数。

23.2.1 CAN 控制和状态寄存器

寄存器 23-1: CiCTRL: CAN 模块控制和状态寄存器

高字节:							
R/W-x	U-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0
TSTAMP	—	CSIDL	ABAT	CANCKS	REQOP<2:0>		
bit 15					bit 8		

低字节:							
R-1	R-0	R-0	U-0	R-0	R-0	R-0	U-0
OPMODE<2:0>			—	ICODE<2:0>			—
bit 7				bit 0			

- bit 15
- TSTAMP: CAN 报文接收捕捉使能位

1 = 使能 CAN 捕捉

0 = 禁止 CAN 捕捉

注: TSTAMP 总是可写的, 与 CAN 模块工作模式无关。
- bit 14
- 未用: 读作 0
- bit 13
- CSIDL: 空闲模式停止位

1 = 当器件进入空闲模式时, 停止 CAN 模块工作

0 = 在空闲模式继续 CAN 模块工作
- bit 12
- ABAT: 中止所有等待的发送位

1 = 中止所有发送缓冲器中等待的发送

0 = 不起作用

注: 当所有发送被中止时, 模块会清零此位。
- bit 11
- CANCKS: CAN 主时钟选择位

1 = FCAN 时钟为 Fcy

0 = FCAN 时钟为 4 Fcy
- bit 10-8
- REQOP<2:0>: 请求工作模式位

111 = 设置监听所有报文模式

110 = 保留

101 = 保留

100 = 设置配置模式

011 = 设置监听模式

010 = 设置环回模式

001 = 设置禁止模式

000 = 设置正常工作模式
- bit 7-5
- OPMODE<2:0>: 工作模式位

注: 这些位表示 CAN 模块的当前工作模式。参见 REQOP 位的说明 (CiCTRL<10:8>)。
- bit 4
- 未用: 读作 0

寄存器 23-1: CiCTRL: CAN 模块控制和状态寄存器（续）

bit 3-1 **ICODE<2:0>**: 中断标志编码位
 111 = 唤醒中断
 110 = RXB0 中断
 101 = RXB1 中断
 100 = TXB0 中断
 011 = TXB1 中断
 010 = TXB2 中断
 001 = 错误中断
 000 = 无中断

bit 0 未用: 读作 0

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

23.2.2 CAN 发送缓冲寄存器

以下部分将描述 CAN 发送缓冲寄存器和相关的发送缓冲控制寄存器。

寄存器 23-2: CiTXnCON: 发送缓冲器状态和控制寄存器

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

低字节:							
U-0	R-0	R-0	R-0	R/W-0	U-0	R/W-0	R/W-0
—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI<1:0>	
bit 7					bit 0		

- bit 15-7 未用: 读作 0
- bit 6 TXABT: 报文中止位
1 = 报文被中止
0 = 报文未被中止
注: 当 TXREQ 置位时此位被清零。
- bit 5 TXLARB: 报文丢失仲裁位
1 = 报文在发送过程中失去仲裁
0 = 报文在发送过程中不失去仲裁
注: 当 TXREQ 置位时此位被清零。
- bit 4 TXERR: 发送时的错误检测位
1 = 报文发送时发生总线错误
0 = 报文发送时未发生总线错误
注: 当 TXREQ 置位时此位被清零。
- bit 3 TXREQ: 报文发送请求位
1 = 请求报文发送
0 = 如果 TXREQ 已置位, 将中止报文发送, 否则不会产生影响
注: 当报文发送成功后, 此位会自动清零。
- bit 2 未用: 读作 0
- bit 1-0 TXPRI<1:0>: 报文发送优先级位
11 = 最高报文优先级
10 = 中高报文优先级
10 = 中低报文优先级
00 = 最低报文优先级

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 23-3: CiTXnSID: 发送缓冲器 n 标准标识符

高字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0
SID<10:6>					—	—	—
bit 15				bit 8			

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID<5:0>						SRR	TXIDE
bit 7				bit 0			

bit 15-11 SID<10:6>: 标准标识符位

bit 10-8 未用: 读作 0

bit 7-2 SID<5:0>: 标准标识符位

bit 1 SRR: 替代远程请求控制位

1 = 报文将请求远程发送

0 = 正常报文发送

bit 0 TXIDE: 扩展标识符位

1 = 报文将发送扩展的标识符

0 = 报文将发送标准标识符

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

寄存器 23-4: CiTXnEID: 发送缓冲器 n 扩展标识符

高字节:							
R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0	U-0
EID<17:14>				—	—	—	—
bit 15				bit 8			

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<13:6>							
bit 7				bit 0			

bit 15-12 EID<17:14>: 扩展标识符的 17-14 位

bit 11-8 未用: 读作 0

bit 7-0 EID<13:6>: 扩展标识符的 13-6 位

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

寄存器 23-5: CiTXnDLC: 发送缓冲器 n 数据长度控制

高字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<5:0>						TXRTR	TXRB1
bit 15							bit 8

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-	U-
TXRB0	DLC<3:0>					—	—
bit 7							

- bit 15-10 **EID<5:0>**: 扩展标识符的 5-0 位
- bit 9 **TXRTR**: 远程发送请求位
1 = 报文将请求远程发送
0 = 正常报文发送
- bit 8-7 **TXRB<1:0>**: 保留位
注: 根据 CAN 协议, 用户必须将这些位置为 1。
- bit 6-3 **DLC<3:0>**: 数据长度码位
- bit 2-0 未用: 读作 0

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 23-6: CiTXnBm: 发送缓冲器 n 数据字段字 m

高字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CTXB<15:8>							
bit 15							bit 8

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CTXB<7:0>							
bit 7							bit 0

- bit 15-0 **CTXB<15:0>**: 数据字段缓冲器字位 (2 字节)

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

23.2.3 CAN 接收缓冲寄存器

本节给出了接收缓冲寄存器及其相关的控制寄存器。

寄存器 23-7: C_{IRX0CON}: 接收缓冲器 0 状态和控制寄存器

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15				bit 8			

低字节:							
R/C-0	U-0	U-0	U-0	R-0	R/W-0	R/W-0	R-0
RXFUL	—	—	—	RXRTRRO	DBEN	JTOFF	FILHIT0
bit 7				bit 0			

- bit 15-8 未用位: 读作 0
- bit 7 **RXFUL**: 接收满状态位
1 = 接收缓冲器包含有效的接收报文
0 = 接收缓冲器准备接收新报文
此位由 CAN 模块置位, 应该在读取缓冲器后由软件清零。
- bit 6-4 未用: 读作 0
- bit 3 **RXRTRRO**: 接收到远程传输请求位 (只读)
1 = 接收到远程传输请求
0 = 未接收到远程传输请求
注: 此位反映上次装入接收缓冲器 0 的报文的状态。
- bit 2 **DBEN**: 接收缓冲器 0 双缓冲使能位
1 = 接收缓冲器 0 溢出将写入接收缓冲器 1
0 = 接收缓冲器 0 溢出不写入接收缓冲器 1
- bit 1 **JTOFF**: 跳转表偏移位 (DBEN 的只读备份)
1 = 允许跳转表在 6 和 7 之间偏移
0 = 允许跳转表在 0 和 1 之间偏移
- bit 0 **FILHIT0**: 指出哪个接收过滤器使能报文接收的位
1 = 接收过滤器 1 (RXF1)
0 = 接收过滤器 0 (RXF0)
注: 此位反映上次装入接收缓冲器 0 的报文状态。

图注:			
R = 可读位	W = 可写位	C = 可清零位	U = 未用位, 读作 0
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 23-8: CiRX1CON: 接收缓冲器 1 状态和控制寄存器

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15				bit 8			

低字节:							
R/C-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
RXFUL	—	—	—	RXRTRRO	FILHIT<2:0>		
bit 7				bit 0			

- bit 15-8 未用：读作 0
- bit 7 **RXFUL**: 接收满状态位
1 = 接收缓冲器包含有效的接收报文
0 = 接收缓冲器准备接收新报文
注： 此位由 CAN 模块置位，应该在读取缓冲器后用软件清零。
- bit 6-4 未用：读作 0
- bit 3 **RXRTRRO**: 接收到远程传输请求位（只读）
1 = 接收到远程传输请求
0 = 未接收远程传输请求
注： 此位反映上次装入接收缓冲器 1 的报文的状态。
- bit 2-0 **FILHIT<2:0>**: 指出哪个接收过滤器使能报文接收的位
101 = 接收过滤器 5（RXF5）
100 = 接收过滤器 4（RXF4）
011 = 接收过滤器 3（RXF3）
010 = 接收过滤器 2（RXF2）
001 = 接收过滤器 1（RXF1）（只可能出现在 DBEN 位置位后）
000 = 接收过滤器 0（RXF0）（只可能出现在 DBEN 位置位后）

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 23-9: CiRXnSID: 接收缓冲器 n 标准标识符

高字节:							
U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	SID<10:6>				
bit 15							bit 8

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID<5:0>						SRR	RXIDE
bit 7							bit 0

bit 15-13 未用: 读作 0

bit 12-2 **SID<10:0>**: 标准标识符位

bit 1 **SRR**: 代替远程请求位 (仅当 RXIDE = 1 时可用)

1 = 发生远程传输请求

0 = 未发生远程传输请求

bit 0 **RXIDE**: 扩展标识符标志位

1 = 接收的报文为扩展数据帧, SID<10:0> 为 EID<28:18>

0 = 接收的报文为一个标准数据帧

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

寄存器 23-10: CiRXnEID: 接收缓冲器 n 扩展标识符

高字节:							
U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	—	EID<17:14>			
bit 15							bit 8

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<13:6>							
bit 7							bit 0

bit 15-12 未用: 读作 0

bit 11-0 **EID<17:6>**: 扩展标识符的 17-6 位

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

寄存器 23-11: CiRXnBm: 接收缓冲器 n 数据字段字 m

高字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CRXB<15:8>							
bit 15				bit 8			

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CRXB<7:0>							
bit 7				bit 0			

bit 15-0 CRXB<15:0>: 数据字段缓冲器字位 (2 个字节)

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 23-12: CiRXnDLC: 接收缓冲器 n 数据长度控制

高字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<5:0>						RXRTR	RB1
bit 15				bit 8			

低字节:							
U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	RB0	DLC<3:0>			
bit 7				bit 0			

- bit 15-10 EID<5:0>: 扩展标识符位
- bit 9 RXRTR: 接收远程发送请求位
1 = 远程传输请求
0 = 无远程传输请求
注: 此位反映了上次接收的报文中 RTR 位的状态。
- bit 8 RB1: 保留的位 1
依据 CAN 规范保留且读作 0
- bit 4 RB0: 保留的位 0
依据 CAN 规范保留且读作 0
- bit 3-0 DLC<3:0>: 数据长度码位 (接收缓冲器中的内容)

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

23.2.4 报文接收过滤器

本节介绍报文接收过滤器。

寄存器 23-13: CiRXFnSID: 接收过滤器 n 标准标识符

高字节:							
U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	SID<10:6>				
bit 15							bit 8

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	R/W-x
SID<5:0>						—	EXIDE
bit 7							bit 0

- bit 15-13 未用: 读作 0
- bit 12-2 **SID<10:0>**: 标准标识符位
- bit 1 未用: 读作 0
- bit 0 **EXIDE**: 扩展标识符使能位
如果 MIDE = 1, 则
1 = 使能用于扩展标识符的过滤器
0 = 使能用于标准标识符的过滤器
如果 MIDE = 0, 则忽略 EXIDE 位

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 23-14: CiRXFnEIDH: 接收过滤器 n 扩展标识符高位

高字节:							
U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	—	EID<17:14>			
bit 15							bit 8

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<13:6>							
bit 7							bit 0

- bit 15-12 未用: 读作 0
- bit 11-0 **EID<17:6>**: 扩展标识符的 17-6 位

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 23-15: CiRXFnEIDL: 接收过滤器 n 扩展标识符低位

高字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0
EID<5:0>						—	—
bit 15							bit 8

低字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

bit 15-10 **EID<5:0>**: 扩展标识符位
bit 9-0 未用: 读作 0

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

23.2.5 接收过滤器屏蔽寄存器

寄存器 23-16: CiRXMnSID: 接收过滤器屏蔽器 n 标准标识符

高字节:							
U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	SID<10:6>				
bit 15							bit 8

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	R/W-x
SID<5:0>						—	MIDE
bit 7							bit 0

bit 15-13 未用: 读作 0

bit 12-2 **SID<10:0>**: 标准标识符屏蔽位

1 = 包含过滤器比较中的位

0 = 不包含过滤器比较中的位

bit 1 未用: 读作 0

bit 0 **MIDE**: 标识符模式选择位

1 = 只与过滤器中的 **EXIDE** 位指定的报文类型 (标准或扩展地址) 匹配

0 = 如果过滤器匹配则与标准或扩展地址报文匹配

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

寄存器 23-17: CiRXMnEIDH: 接收过滤器屏蔽器 n 扩展标识符高位

高字节:							
U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	—	EID<17:14>			
bit 15							bit 8

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID<13:6>							
bit 7							bit 0

bit 15-12 未用: 读作 0

bit 11-0 **EID<17:6>**: 扩展标识符屏蔽器的 17-6 位

1 = 包含过滤器比较中的位

0 = 不包含过滤器比较中的位

图注:

R = 可读位

W = 可写位

U = 未用位, 读作 0

-n = POR 时的值

1 = 置位

0 = 清零

x = 未知

寄存器 23-18: CiRXMnEIDL: 接收过滤器屏蔽器 n 扩展标识符低位

高字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0
EID<5:0>						—	—
bit 15						bit 8	

低字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7				bit 0			

bit 15-10 EID<5:0>: 扩展标识符位

bit 9-0 未用: 读作 0

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

23.2.6 CAN 波特率寄存器

本节介绍 CAN 波特率寄存器。

寄存器 23-19: CiCFG1: 波特率配置寄存器 1

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SJW<1:0>		BRP<5:0>					
bit 7				bit 0			

bit 15-8 未用: 读作 0

bit 7-6 **SJW<1:0>**: 同步跳转宽度位
11 = 同步跳转宽度时间为 4 x TQ
10 = 同步跳转宽度时间为 3 x TQ
01 = 同步跳转宽度时间为 2 x TQ
00 = 同步跳转宽度时间为 1 x TQ

bit 5-0 **BRP<5:0>**: 波特率预分频位
11 1111 = $TQ = 2 \times (BRP + 1) / F_{CAN} = 128 / F_{CAN}$
.
.
00 0000 = $TQ = 2 \times (BRP + 1) / F_{CAN} = 2 / F_{CAN}$

注: F_{CAN} 为 F_{CY} 或 4 F_{CY}, 由 CANCKS 位的设置决定。

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

寄存器 23-20: CiCFG2: 波特率配置寄存器 2

高字节:							
U-0	R/W-x	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x
—	WAKFIL	—	—	—	SEG2PH<2:0>		
bit 15					bit 8		

低字节:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SEG2PHT S	SAM	SEG1PH<2:0>			PRSEG<2:0>		
bit 7							bit 0

- bit 15 未用：读作 0
- bit 14 **WAKFIL**：将 CAN 总线滤波器用于唤醒的选择位
1 = 使用 CAN 总线滤波器唤醒
0 = 不使用 CAN 总线滤波器唤醒
- bit 13-11 未用：读作 0
- bit 10-8 **SEG2PH<2:0>**：相位缓冲段 2 位
111 = 长度为 8 × Tq
. . .
000 = 长度为 1 × Tq
- bit 7 **SEG2PHTS**：相位段 2 时间选择位
1 = 可自由编程
0 = SEG1PH 与信息处理时间（3 Tq）中较大的一个（无论两者中哪个较大）
- bit 6 **SAM**：CAN 总线采样位
1 = 总线在采样点被采样三次
0 = 总线在采样点被采样一次
- bit 5-3 **SEG1PH<2:0>**：相位缓冲段 1 位
111 = 长度为 8 × Tq
. . .
000 = 长度为 1 × Tq
- bit 2-0 **PRSEG<2:0>**：广播时间段位
111 = 长度为 8 × Tq
. . .
000 = 长度为 1 × Tq

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

23.2.7 CAN 模块错误计数寄存器

本节介绍 CAN 模块发送 / 接收错误计数寄存器。各种错误状态标志位在 CAN 中断标志寄存器中。

寄存器 23-21: CIEC: 发送 / 接收错误计数

高字节:							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
TERRCNT<7:0>							
bit 15				bit 8			

低字节:							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
RERRCNT<7:0>							
bit 7				bit 0			

bit 15-8 TERRCNT<7:0>: 发送错误计数位

bit 7-0 RERRCNT<7:0>: 接收错误计数位

图注:			
R = 可读位	W = 可写位	U = 未用位, 读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

23.2.8 CAN 中断寄存器

本节介绍与中断相关的 CAN 寄存器。

寄存器 23-22: CIINTE: 中断使能寄存器

高字节:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IVRIE	WAKIE	ERRIE	TX2IE	TX1IE	TX0IE	RX1IE	RX0IE
bit 7				bit 0			

- bit 15-8 未用：读作 0
- bit 7 **IVRIE**: 接收到无效报文的中断使能位
1 = 使能
0 = 禁止
- bit 6 **WAKIE**: 总线唤醒活动中断使能位
1 = 使能
0 = 禁止
- bit 5 **ERRIE**: 错误中断使能位
1 = 使能
0 = 禁止
- bit 4 **TX2IE**: 发送缓冲器 2 中断使能位
1 = 使能
0 = 禁止
- bit 3 **TX1IE**: 发送缓冲器 1 中断使能位
1 = 使能
0 = 禁止
- bit 2 **TX0IE**: 发送缓冲器 0 中断使能位
1 = 使能
0 = 禁止
- bit 1 **RX1IE**: 接收缓冲器 1 中断使能位
1 = 使能
0 = 禁止
- bit 0 **RX0IE**: 接收缓冲器 0 中断使能位
1 = 使能
0 = 禁止

图注:			
R = 可读位	W = 可写位	U = 未用位，读作 0	
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

dsPIC30F 系列参考手册

寄存器 23-23: CILNTE: 中断标志寄存器

高字节:							
R/C-0	R/C-0	R-0	R-0	R-0	R-0	R-0	R-0
RX0OVR	RX1OVR	TXBO	TXEP	RXEP	TXWAR	RXWAR	EWARN
bit 15				bit 8			

低字节:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IVRIF	WAKIF	ERRIF	TX2IF	TX1IF	TX0IF	RX1IF	RX0IF
bit 7				bit 0			

- bit 15 **RX0OVR**: 接收缓冲器 0 溢出位
1 = 接收缓冲器 0 溢出
0 = 接收缓冲器 0 未溢出
- bit 14 **RX1OVR**: 接收缓冲器 1 溢出位
1 = 接收缓冲器 1 溢出
0 = 接收缓冲器 1 未溢出
- bit 13 **TXBO**: 发送器处于错误状态, 总线关闭位
1 = 发送器处于错误状态, 总线关闭
0 = 发送器不处于错误状态, 总线关闭
- bit 12 **TXEP**: 发送器处于错误状态, 总线被动位
1 = 发送器处于错误状态, 总线被动
0 = 发送器不处于错误状态, 总线被动
- bit 11 **RXEP**: 接收器处于错误状态, 总线被动位
1 = 接收器处于错误状态, 总线被动
0 = 接收器不处于错误状态, 总线被动
- bit 10 **TXWAR**: 发送器处于错误状态, 警告位
1 = 发送器处于错误状态, 警告
0 = 发送器未处于错误状态, 警告
- bit 9 **RXWAR**: 接收器处于错误状态, 警告位
1 = 接收器处于错误状态, 警告
0 = 接收器未处于错误状态, 警告
- bit 8 **EWARN**: 发送器或接收器处于错误状态, 警告位
1 = 发送器或接收器处于错误状态, 警告
0 = 发送器和接收器均未处于错误状态
- bit 7 **IVRIF**: 接收到无效报文的中断标志位
1 = 在接收上一条报文过程中发生了某种类型的错误
0 = 未发生接收错误
- bit 6 **WAKIF**: 总线唤醒活动中断标志位
1 = 发生中断请求
0 = 未发生中断请求
- bit 5 **ERRIF**: 错误中断标志位 (CILNTE<15:8> 寄存器中有多个中断源)
1 = 发生中断请求
0 = 未发生中断请求
- bit 4 **TX2IF**: 发送缓冲器 2 中断标志位
1 = 发生中断请求
0 = 未发生中断请求
- bit 3 **TX1IF**: 发送缓冲器 1 中断标志位
1 = 发生中断请求
0 = 未发生中断请求

寄存器 23-23: CiINTF: 中断标志寄存器 (续)

- bit 2 TX0IF: 发送缓冲器 0 中断标志位
1 = 发生中断请求
0 = 未发生中断请求
- bit 1 RX1IF: 接收缓冲器 1 中断标志位
1 = 发生中断请求
0 = 未发生中断请求
- bit 0 RX0IF: 接收缓冲器 0 中断标志位
1 = 发生中断请求
0 = 未发生中断请求

图注:			
R = 可读位	W = 可写位	C = 可清零位	U = 未用位, 读作 0
-n = POR 时的值	1 = 置位	0 = 清零	x = 未知

表 23-1: CAN1 寄存器映射

寄存器名称	地址	bit																复位	
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
C1RXF0SID	300	—	—	—	SID<10:6>				SID<5:0>							—	EXIDE	xxxx	
C1RXF0EIDH	302	—	—	—	—	EID<17:14>				EID<13:6>									xxxx
C1RXF0EIDL	304	EID<5:0>								—	—		—	—	—	—	—	—	xxxx
未使用	306	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx	
C1RXF1SID	308	—	—	—	SID<10:6>				SID<5:0>							—	EXIDE	xxxx	
C1RXF1EIDH	30A	—	—	—	—	EID<17:14>				EID<13:6>									xxxx
C1RXF1EIDL	30C	EID<5:0>								—	—		—	—	—	—	—	—	xxxx
未使用	30E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx	
C1RXF2SID	310	—	—	—	SID<10:6>				SID<5:0>							—	EXIDE	xxxx	
C1RXF2EIDH	312	—	—	—	—	EID<17:14>				EID<13:6>									xxxx
C1RXF2EIDL	314	EID<5:0>								—	—		—	—	—	—	—	—	xxxx
未使用	316	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx	
C1RXF3SID	318	—	—	—	SID<10:6>				SID<5:0>							—	EXIDE	xxxx	
C1RXF3EIDH	31A	—	—	—	—	EID<17:14>				EID<13:6>									xxxx
C1RXF3EIDL	31C	EID<5:0>								—	—		—	—	—	—	—	—	xxxx
未使用	31E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx	
C1RXF4SID	320	—	—	—	SID<10:6>				SID<5:0>							—	EXIDE	xxxx	
C1RXF4EIDH	322	—	—	—	—	EID<17:14>				EID<13:6>									xxxx
C1RXF4EIDL	324	EID<5:0>								—	—		—	—	—	—	—	—	xxxx
未使用	326	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx	
C1RXF5SID	328	—	—	—	SID<10:6>				SID<5:0>							—	EXIDE	xxxx	
C1RXF5EIDH	32A	—	—	—	—	EID<17:14>				EID<13:6>									xxxx
C1RXF5EIDL	32C	EID<5:0>								—	—		—	—	—	—	—	—	xxxx
未使用	32E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx	
C1RXM0SID	330	—	—	—	SID<10:6>				SID<5:0>							—	MIDE	xxxx	
C1RXM0EIDH	332	—	—	—	—	EID<17:14>				EID<13:6>									xxxx
C1RXM0EIDL	334	EID<5:0>								—	—		—	—	—	—	—	—	xxxx
未使用	336	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx	
C1RXM1SID	338	—	—	—	SID<10:6>				SID<5:0>							—	MIDE	xxxx	
C1RXM1EIDH	33A	—	—	—	—	EID<17:14>				EID<13:6>									xxxx
C1RXM1EIDL	33C	EID<5:0>								—	—		—	—	—	—	—	—	xxxx
未使用	33E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx	

表 23-1: CAN1 寄存器映射 (续)

寄存器名称	地址	bit																复位		
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
C1TX2SID	340	SID<10:6>					—	—	—	SID<5:0>						SRR	TX IDE	xxxx		
C1TX2EID	342	EID<17:14>				—	—	—	—	EID<13:6>										xxxx
C1TX2DLC	342	EID<5:0>						TX RTR	TX RB1	TX RB0	DLC<3:0>				—	—	—	xxxx		
C1TX2B1	346	发送缓冲器 0 字节 1								发送缓冲器 0 字节 0										xxxx
C1TX2B2	348	发送缓冲器 0 字节 3								发送缓冲器 0 字节 2										xxxx
C1TX2B3	34A	发送缓冲器 0 字节 5								发送缓冲器 0 字节 4										xxxx
C1TX2B4	34C	发送缓冲器 0 字节 7								发送缓冲器 0 字节 6										xxxx
C1TX2CON	34E	—	—	—	—	—	—	—	—	—	TX ABT	TX LARB	TX ERR	TX REQ	—	TXPRI[1:0]	0000			
C1TX1SID	350	SID<10:6>					—	—	—	SID<5:0>						SRR	TX IDE	xxxx		
C1TX1EID	352	EID<17:14>				—	—	—	—	EID<13:6>										xxxx
C1TX1DLC	352	EID<5:0>						TX RTR	TX RB1	TX RB0	DLC<3:0>				—	—	—	xxxx		
C1TX1B1	356	发送缓冲器 0 字节 1								发送缓冲器 0 字节 0										xxxx
C1TX1B2	358	发送缓冲器 0 字节 3								发送缓冲器 0 字节 2										xxxx
C1TX1B3	35A	发送缓冲器 0 字节 5								发送缓冲器 0 字节 4										xxxx
C1TX1B4	35C	发送缓冲器 0 字节 7								发送缓冲器 0 字节 6										xxxx
C1TX1CON	35E	—	—	—	—	—	—	—	—	—	TX ABT	TX LARB	TX ERR	TX REQ	—	TXPRI[1:0]	0000			
C1TX0SID	360	SID<10:6>					—	—	—	SID<5:0>						SRR	TX IDE	xxxx		
C1TX0EID	362	EID<17:14>				—	—	—	—	EID<13:6>										xxxx
C1TX0DLC	362	EID<5:0>						TX RTR	TX RB1	TX RB0	DLC<3:0>				—	—	—	xxxx		
C1TX0B1	366	发送缓冲器 0 字节 1								发送缓冲器 0 字节 0										xxxx
C1TX0B2	368	发送缓冲器 0 字节 3								发送缓冲器 0 字节 2										xxxx
C1TX0B3	36A	发送缓冲器 0 字节 5								发送缓冲器 0 字节 4										xxxx
C1TX0B4	36C	发送缓冲器 0 字节 7								发送缓冲器 0 字节 6										xxxx
C1TX0CON	36E	—	—	—	—	—	—	—	—	—	TX ABT	TX LARB	TX ERR	TX REQ	—	TXPRI[1:0]	0000			

表 23-1: CAN1 寄存器映射 (续)

寄存器名称	地址	bit																复位			
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
C1RX1SID	370	—	—	—	SID<10:6>					SID<5:0>							SRR	RX IDE	xxxx		
C1RX1EID	372	—	—	—	—	EID<17:14>					EID<13:6>										xxxx
C1RX1DLC	374	EID<0:5>						RX RTR	RX RB1	—	—	—	RX RB0	DLC[3:0]						xxxx	
C1RX1B1	376	接收缓冲器 1 字节 1								接收缓冲器 1 字节 0											xxxx
C1RX1B2	378	接收缓冲器 1 字节 3								接收缓冲器 1 字节 2											xxxx
C1RX1B3	37A	接收缓冲器 1 字节 5								接收缓冲器 1 字节 4											xxxx
C1RX1B4	37C	接收缓冲器 1 字节 7								接收缓冲器 1 字节 6											xxxx
C1RX1CON	37E	—	—	—	—	—	—	—	—	RX FUL	—	—	RX ERR	RX RTR R0	FILHIT[2:0]		0000				
C1RX1SID	380	—	—	—	SID<10:6>					SID<5:0>							SRR	RX IDE	xxxx		
C1RX1EID	382	—	—	—	—	EID<17:14>					EID<13:6>										xxxx
C1RX1DLC	384	EID<0:5>						RX RTR	RX RB1	—	—	—	RX RB0	DLC[3:0]						xxxx	
C1RX0B1	386	接收缓冲器 0 字节 1								接收缓冲器 0 字节 0											xxxx
C1RX0B2	388	接收缓冲器 0 字节 3								接收缓冲器 0 字节 2											xxxx
C1RX0B3	38A	接收缓冲器 0 字节 5								接收缓冲器 0 字节 4											xxxx
C1RX0B4	38C	接收缓冲器 0 字节 7								接收缓冲器 0 字节 6											xxxx
C1RX0CON	38E	—	—	—	—	—	—	—	—	RX FUL	—	—	RX ERR	RX RTR R0	RXB0 DBEN	JTOFF	FIL HIT 0	0000			
C1CTRL	390	CAN CAP	—	C SIDL	ABAT	CAN CKS	REQOP[2:0]			OPMODE[2:0]			—	ICODE[2:0]			—	0480			
C1CFG1	392	—	—	—	—	—	—	—	—	SJW[1:0]S		BRP[5:0]							0000		
C1CFG2	394	—	WAK FIL	—	—	—	SEG2PH[2:0]			SEG2 PHTS	SAM	SEG1PH[2:0]			PRSEG[2:0]			0000			
C1INTF	396	RXB0 OVR	RXB1 OVR	TXB0	TXBP	RXBP	TX WARN	RX WARN	E WARN	IVR IF	WAK IF	ERR IF	TXB2 IF	TXB1 IF	TXB0 IF	RXB1 IF	RXB0 IF	0000			
C1INTE	398	—	—	—	—	—	—	—	—	IVR IE	WAK IE	ERR IE	TXB2 IE	TXB1 IE	TXB0 IE	RXB1 IE	RXB0 IE	0000			
C1EC	39A	发送错误计数器								接收错误计数器											0000
保留	39C 3FE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx			

图注: x = 未知

表 23-2: CAN2 寄存器映射

寄存器名称	地址	bit																复位		
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
C2RXF0SID	3C0	—	—	—	SID<10:6>				SID<5:0>								—	EXIDE	xxxx	
C2RXF0EIDH	3C2	—	—	—	—	EID<17:14>				EID<13:6>										xxxx
C2RXF0EIDL	3C4	EID<5:0>						—	—	—	—	—	—	—	—	—	—	—	xxxx	
未使用	3C6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx		
C2RXF1SID	3C8	—	—	—	SID<10:6>				SID<5:0>								—	EXIDE	xxxx	
C2RXF1EIDH	3CA	—	—	—	—	EID<17:14>				EID<13:6>										xxxx
C2RXF1EIDL	3CC	EID<5:0>						—	—	—	—	—	—	—	—	—	—	xxxx		
未使用	3CE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx		
C2RXF2SID	3D0	—	—	—	SID<10:6>				SID<5:0>								—	EXIDE	xxxx	
C2RXF2EIDH	3D2	—	—	—	—	EID<17:14>				EID<13:6>										xxxx
C2RXF2EIDL	3D4	EID<5:0>						—	—	—	—	—	—	—	—	—	—	xxxx		
未使用	3D6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx		
C2RXF3SIDH	3D8	—	—	—	SID<10:6>				SID<5:0>								—	EXIDE	xxxx	
C2RXF3EID	3DA	—	—	—	—	EID<17:14>				EID<13:6>										xxxx
C2RXF3EIDL	3DC	EID<5:0>						—	—	—	—	—	—	—	—	—	—	xxxx		
未使用	3DE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx		
C2RXF4SID	3E0	—	—	—	SID<10:6>				SID<5:0>								—	EXIDE	xxxx	
C2RXF4EIDH	3E2	—	—	—	—	EID<17:14>				EID<13:6>										xxxx
C2RXF4EIDL	3E4	EID<5:0>						—	—	—	—	—	—	—	—	—	—	xxxx		
未使用	3E6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx		
C2RXF5SID	3E8	—	—	—	SID<10:6>				SID<5:0>								—	EXIDE	xxxx	
C2RXF5EIDH	3EA	—	—	—	—	EID<17:14>				EID<13:6>										xxxx
C2RXF5EIDL	3EC	EID<5:0>						—	—	—	—	—	—	—	—	—	—	xxxx		
未使用	3EE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx		
C2RXM0SID	3F0	—	—	—	SID<10:6>				SID<5:0>								—	MIDE	xxxx	
C2RXM0EIDH	3F2	—	—	—	—	EID<17:14>				EID<13:6>										xxxx
C2RXM0EIDL	3F4	EID<5:0>						—	—	—	—	—	—	—	—	—	—	xxxx		
未使用	3F6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx		
C2RXM1SID	3F8	—	—	—	SID<10:6>				SID<5:0>								—	MIDE	xxxx	
C2RXM1EIDH	3FA	—	—	—	—	EID<17:14>				EID<13:6>										xxxx
C2RXM1EIDL	3FC	EID<5:0>						—	—	—	—	—	—	—	—	—	—	xxxx		
未使用	3FE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx		

表 23-2: CAN2 寄存器映射 (续)

寄存器名称	地址	bit																复位			
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
C2TX2SID	400	SID<10:6>						—	—	—	SID<5:0>						SRR	TX IDE	xxxx		
C2TX2EID	402	EID<17:14>				—	—	—	—	EID<13:6>											xxxx
C2TX2DLC	404	EID<5:0>						TX RTR	TX RB1	TX RB0	DLC<3:0>				—	—	—		xxxx		
C2TX2B1	406	发送缓冲器 0 字节 1								发送缓冲器 0 字节 0											xxxx
C2TX2B2	408	发送缓冲器 0 字节 3								发送缓冲器 0 字节 2											xxxx
C2TX2B3	40A	发送缓冲器 0 字节 5								发送缓冲器 0 字节 4											xxxx
C2TX2B4	40C	发送缓冲器 0 字节 7								发送缓冲器 0 字节 6											xxxx
C2TX2CON	40E	—	—	—	—	—	—	—	—	—	TX ABT	TX LARB	TX ERR	TX REQ	—	TXPRI[1:0]		0000			
C2TX1SID	410	SID<10:6>						—	—	—	SID<5:0>						SRR	TX IDE	xxxx		
C2TX1EID	412	EID<17:14>				—	—	—	—	EID<13:6>											xxxx
C2TX1DLC	414	EID<5:0>						TX RTR	TX RB1	TX RB0	DLC<3:0>				—	—	—		xxxx		
C2TX1B1	416	发送缓冲器 0 字节 1								发送缓冲器 0 字节 0											xxxx
C2TX1B2	418	发送缓冲器 0 字节 3								发送缓冲器 0 字节 2											xxxx
C2TX1B3	41A	发送缓冲器 0 字节 5								发送缓冲器 0 字节 4											xxxx
C2TX1B4	41C	发送缓冲器 0 字节 7								发送缓冲器 0 字节 6											xxxx
C2TX1CON	41E	—	—	—	—	—	—	—	—	—	TX ABT	TX LARB	TX ERR	TX REQ	—	TXPRI[1:0]		0000			
C2TX0SID	420	SID<10:6>						—	—	—	SID<5:0>						SRR	TX IDE	xxxx		
C2TX0EID	422	EID<17:14>				—	—	—	—	EID<13:6>											xxxx
C2TX0DLC	424	EID<5:0>						TX RTR	TX RB1	TX RB0	DLC<3:0>				—	—	—		xxxx		
C2TX0B1	426	发送缓冲器 0 字节 1								发送缓冲器 0 字节 0											xxxx
C2TX0B2	428	发送缓冲器 0 字节 3								发送缓冲器 0 字节 2											xxxx
C2TX0B3	42A	发送缓冲器 0 字节 5								发送缓冲器 0 字节 4											xxxx
C2TX0B4	42C	发送缓冲器 0 字节 7								发送缓冲器 0 字节 6											xxxx
C2TX0CON	42E	—	—	—	—	—	—	—	—	—	TX ABT	TX LARB	TX ERR	TX REQ	—	TXPRI[1:0]		0000			

表 23-2: CAN2 寄存器映射 (续)

寄存器名称	地址	bit																复位			
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
C2RX1SID	430	—	—	—	SID<10:6>					SID<5:0>							SRR	RX IDE	xxxx		
C2RX1EID	432	—	—	—	—	EID<17:14>					EID<13:6>										xxxx
C2RX1DLC	434	EID<0:5>						RX RTR	RX RB1	—	—	—	RX RB0	DLC[3:0]							xxxx
C2RX1B1	436	接收缓冲器 1 字节 1								接收缓冲器 1 字节 0											xxxx
C2RX1B2	438	接收缓冲器 1 字节 3								接收缓冲器 1 字节 2											xxxx
C2RX1B3	43A	接收缓冲器 1 字节 5								接收缓冲器 1 字节 4											xxxx
C2RX1B4	43C	接收缓冲器 1 字节 7								接收缓冲器 1 字节 6											xxxx
C2RX1CON	43E	—	—	—	—	—	—	—	—	RX FUL	—	—	RX ERR	RX RTR R0	FILHIT[2:0]			0000			
C2RX1SID	440	—	—	—	SID<10:6>					SID<5:0>							SRR	RX IDE	xxxx		
C2RX1EID	442	—	—	—	—	EID<17:14>					EID<13:6>										xxxx
C2RX1DLC	444	EID<0:5>						RX RTR	RX RB1	—	—	—	RX RB0	DLC[3:0]							xxxx
C2RX0B1	446	接收缓冲器 0 字节 1								接收缓冲器 0 字节 0											xxxx
C2RX0B2	448	接收缓冲器 0 字节 3								接收缓冲器 0 字节 2											xxxx
C2RX0B3	44A	接收缓冲器 0 字节 5								接收缓冲器 0 字节 4											xxxx
C2RX0B4	44C	接收缓冲器 0 字节 7								接收缓冲器 0 字节 6											xxxx
C2RX0CON	44E	—	—	—	—	—	—	—	—	RX FUL	—	—	RX ERR	RX RTR R0	RXB0 DBEN	JTOFF	FIL HIT 0	0000			
C2CTRL	450	CAN CAP	—	C SIDL	ABAT	CAN CKS	REQOP[2:0]			OPMODE[2:0]			—	ICODE[2:0]			—	0480			
C2CFG1	452	—	—	—	—	—	—	—	—	SJW[1:0]S		BRP[5:0]							0000		
C2CFG2	454	—	WAK FIL	—	—	—	SEG2PH[2:0]			SEG2 PHTS	SAM	SEG1PH[2:0]			PRSEG[2:0]			0000			
C2INTF	456	RXB0 OVR	RXB1 OVR	TXBO	TXBP	RXBP	TX WARN	RX WARN	E WARN	IVR IF	WAK IF	ERR IF	TXB2 IF	TXB1 IF	TXB0 IF	RXB1 IF	RXB0 IF	0000			
C2INTE	458	—	—	—	—	—	—	—	—	IVR IE	WAK IE	ERR IE	TXB2 IE	TXB1 IE	TXB0 IE	RXB1 IE	RXB0 IE	0000			
C2EC	45A	发送错误计数器								接收错误计数器											0000
保留	45C 4FE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	xxxx			

图注: x = 未知

23.3 CAN 模块特点

CAN 模块是实现 BOSCH 规范中定义的 CAN 2.0A/B 协议的通信控制器。模块支持 CAN 协议的 CAN 1.2、CAN 2.0A、CAN 2.0B Passive 和 CAN 2.0B Active 版本。该模块实现的是一个完整的 CAN 系统。

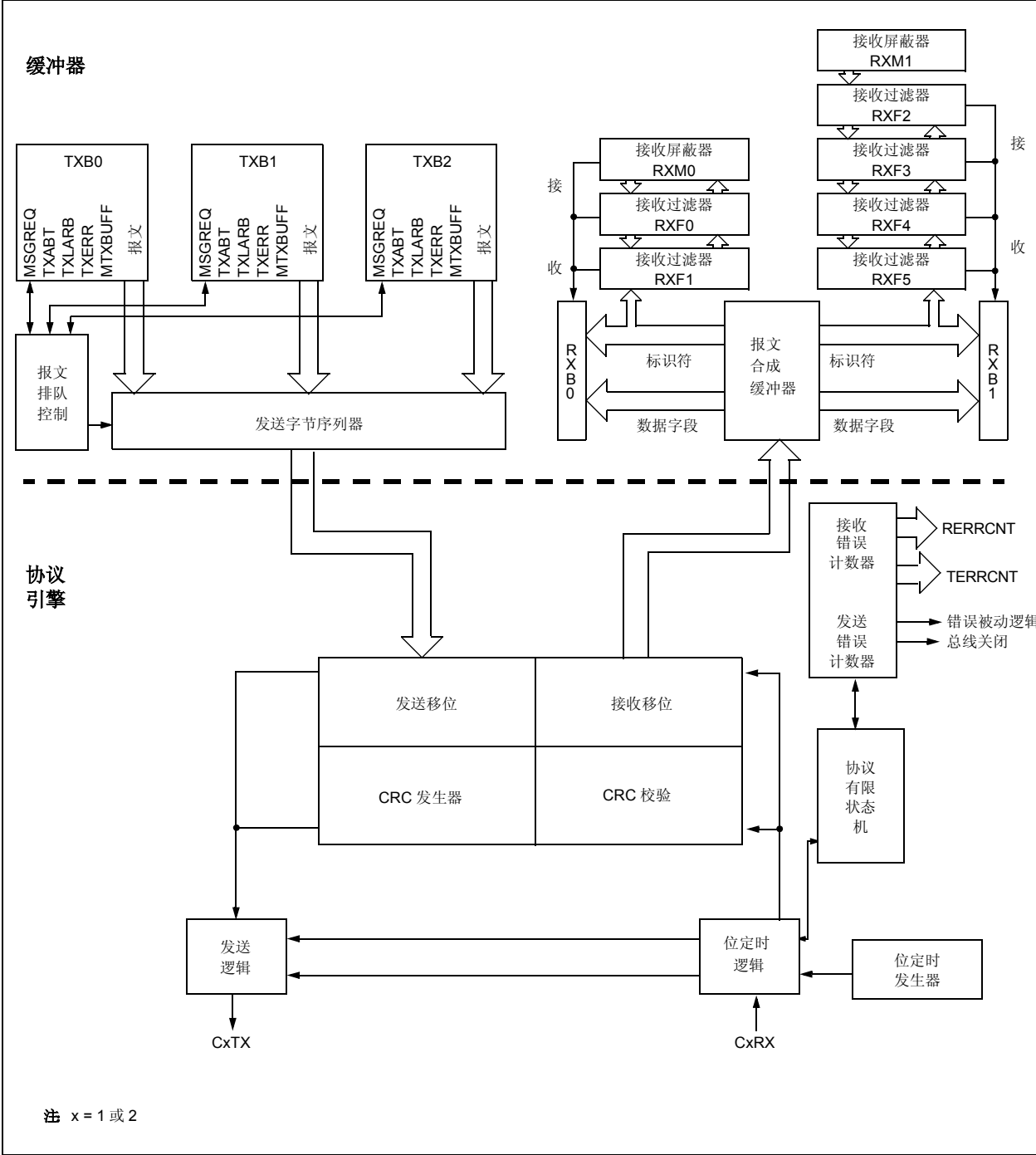
模块特性如下：

- 实现 CAN 协议：CAN 1.2、CAN 2.0A 和 CAN 2.0B
- 标准和扩展的数据帧
- 数据长度为 0 到 8 字节
- 可编程比特率达到 1 Mb/s
- 支持远程数据帧
- 双缓冲的接收器，带两个区分优先级的接收报文存储缓冲器
- 6 个完全（标准 / 扩展标识符）接收过滤器，其中 2 个与高优先级的接收缓冲器相关联，4 个与低优先级接收缓冲器相关联
- 2 个完全接收过滤屏蔽寄存器，其中的一个与高优先级接收缓冲器相关联，另一个与低优先级接收缓冲器相关联
- 三个发送缓冲器，能够由应用指定优先级并具有中止能力
- 集成低通滤波器的可编程唤醒功能
- 支持自检操作的可编程环回模式
- 通过中断功能为所有 CAN 接收器和发送器的错误状态发送信号
- 可编程时钟源
- 可编程的定时器模块链接，以标记时间标记并进行网络同步
- 低功耗休眠模式

23.4 CAN 模块的实现

CAN 总线模块由一个协议引擎与报文缓冲和控制组成。通过确定模块发送和接收的数据帧类型就可以很好的理解协议引擎。这些框图见图 23-2。

图 23-2: CAN 缓冲器和协议引擎框图



23.4.1 CAN 报文格式

CAN 协议引擎处理在 CAN 总线上接收和发送报文的所有功能。报文通过首先装入相应的数据寄存器来发送。可以通过读取相应的寄存器来检查状态和错误。在 CAN 总线上检测到的任何报文会进行错误检查然后与过滤器进行匹配，以确定该报文是否应该被接收并存储在两个接收寄存器之一中。

CAN 模块支持以下类型的帧：

- 标准数据帧
- 扩展数据帧
- 远程帧
- 错误帧
- 帧间间隔

23.4.1.1 标准数据帧

当节点希望发送数据时会产生一个标准数据帧。图 23-3 所示为标准 CAN 数据帧。与其他所有帧相同，标准 CAN 数据帧以帧起始位（SOF 一显性状态）开始，与所有节点进行硬同步。

在 SOF 之后是仲裁字段，由 12 位组成，包括 11 位的标识符（表示报文的内容和优先级）和远程传输请求（Remote Transmission Request, RTR）位。RTR 位用于区分数据帧（RTR 一显性）和远程帧。

下一个字段是控制字段，由 6 位组成。字段的第一位称为标识符扩展（Identifier Extension, IDE）位，该位为显性状态时，说明该帧为标准帧。接下来是 CAN 协议的保留位 RB0，这一位也被定义为显性位。控制字段的其余 4 位为数据长度码（Data Length Code, DLC），规定了报文中包含的数据字节数。

控制字段之后为数据字段，包含正在发送的数据字节，数据字段长度由上述数据长度码 DLC 定义（0 到 8 字节）。

数据字段之后是循环冗余校验（Cyclic Redundancy Check, CRC）字段，用来检测可能的报文传输错误。CRC 字段由一个 15 位的 CRC 序列和一个定界符位组成。报文以帧结束（End-Of-Frame, EOF）字段结束。该字段由七个无位填充隐性位构成。

最后是应答字段。在应答间隙位期间，发送节点发出一个隐性位。任何收到无错误帧的节点会发回一个显性位（无论该节点是否配置为接收该特定报文），应答帧已被正确接收。隐性应答定界符是应答间隙的结束标志，除了发生错误帧外不能被显性位改写。

23.4.1.2 扩展数据帧

如图 23-4 所示，在扩展 CAN 数据帧中，紧随帧起始（SOF）位的是 38 位仲裁字段。仲裁字段的前 11 位为 29 位标识符的 11 个最高位（基本 ID）。紧随这 11 位的是代替远程请求（Substitute Remote Request, SRR）位，它以隐性状态发送。SRR 位后是 IDE 位，该位隐性时表示这是扩展的 CAN 帧。值得注意的是，如果在扩展帧标识符的前 11 位发送完后，总线仲裁无果，而此时参与仲裁的某个节点发出标准 CAN 帧（11 位标识符），那么由于节点发出了显性 IDE 位而使标准 CAN 帧赢得仲裁。另外，扩展 CAN 帧的 SRR 位应为隐性，以允许正在发送标准 CAN 远程帧的节点发出显性 RTR 位。SRR 位和 IDE 位之后是标识符的其余 18 位（扩展 ID）以及一个显性远程发送请求位。

为使标准帧和扩展帧都能在同一网络上发送，应将 29 位的扩展报文标识符拆分成 11 位（最高位）和 18 位（最低位）两部分。拆分时必须确保标识符扩展位（IDE）在标准帧和扩展帧中的位置保持不变。

下一个字段是控制字段，由 6 位组成。控制字段前两位为保留位，为显性状态。控制字段的其余 4 位为数据长度码（DLC），它规定了数据字节数。

扩展数据帧的其他部分（数据字段、CRC 字段、应答字段、帧结束和中断）在结构上与标准数据帧相同。

23.4.1.3 远程帧

数据传输通常是由数据源节点（例如，传感器发送数据帧）自主完成的。但也可能发生目的节点向源节点请求发送数据的情况。要做到这一点，目的节点需要发送一个标识符与所需数据帧的标识符相匹配的“远程帧”。随后相应的数据源节点会发送一个数据帧作为对远程请求的响应。

如图 23-5 所示，远程帧与数据帧有两点不同。第一，远程帧的 RTR 位为隐性状态；第二，远程帧没有数据字段。带有相同标识符的数据帧和远程帧同时发送的情况是很少出现的，这种情况下数据帧将赢得仲裁，这是因为其标识符之后的 RTR 位为显性。这样可使发送远程帧的节点立即收到所需数据。

23.4.1.4 错误帧

错误帧是由检测到总线错误的任一节点产生的。如图 23-6 所示，错误帧包含 2 个字段，即错误标志字段和紧随其后的错误定界符字段。错误定界符由 8 个隐性位组成，允许总线节点在错误发生后立即重新启动总线通信。错误标志字段有两种形式。其具体形式取决于检测到错误的节点的错误状态。

当错误主动节点检测到一个总线错误时，这个节点将通过产生一个主动错误标志，中断当前的报文发送。主动错误标志由 6 个连续的显性位构成。这种位序列有效地打破了位填充规则。所有其他站点在识别到由此产生的位填充错误后，反过来也会产生错误帧，称为错误回送标志（Error Echo Flag）。错误标志字段因此包含 6 到 12 个连续显性位（由 1 个或多个节点产生）。错误帧以错误定界符字段结束。在错误帧发送完毕后，总线活动恢复到正常状态，被中断的节点会尝试重新发送被中止的报文。

当错误被动节点检测到一个总线错误时，该节点将发送一个错误被动标志，后面仍然跟错误定界符字段。错误被动标志由六个连续的隐性位组成。由此可知，除非总线错误被正在发送报文的总线主节点或其他实际上正在发送的错误主动接收器检测到，否则错误被动节点发送错误帧将不会影响网络中任何其他节点。如果总线主节点产生了一个错误被动标志，那么由于位填充规则被打破，将导致其他节点产生错误帧。错误帧发送完毕后，错误被动节点必须等待总线上出现 6 个连续隐性位后，才能尝试重新参与总线通信。

23.4.1.5 帧间间隔

帧间间隔将前一个帧（无论何种类型）与其后的数据帧或远程帧分隔开来。帧间间隔至少由 3 个隐性位构成，也称为间断。间断使接收节点在开始发送 / 接收下一个报文帧之前有时间进行报文的内部处理。在间断之后，CAN 总线将保持隐性状态（总线空闲），直至开始发送下一个帧。

如果发送节点处于错误被动状态，在节点发送任何其他报文前，帧间间隔中会被插入另外 8 个隐性位。此时间段被称为暂停发送段。暂停发送段可以为其他发送节点取得总线的控制权留出更多的延时。

图 23-3: 标准数据帧

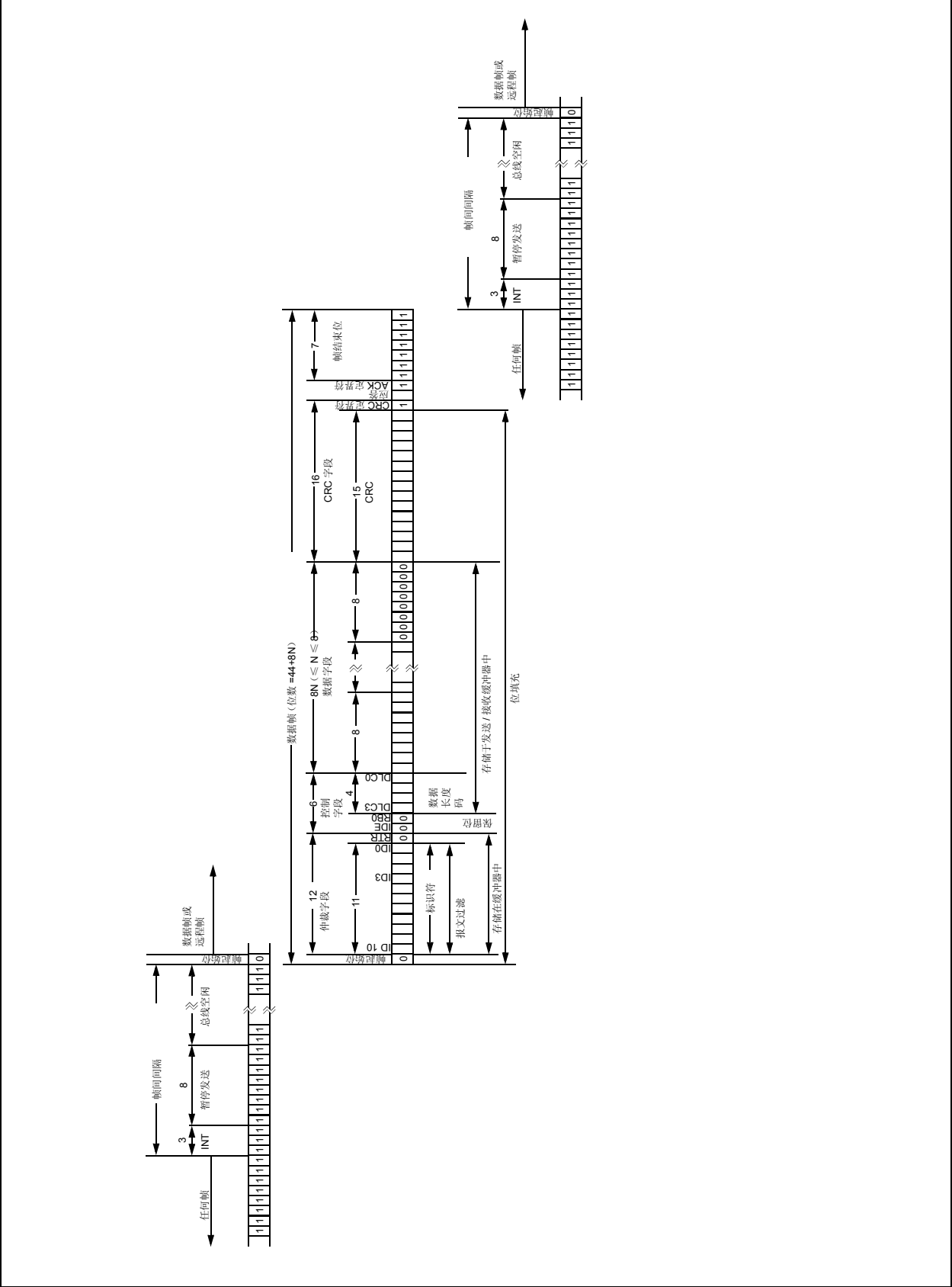
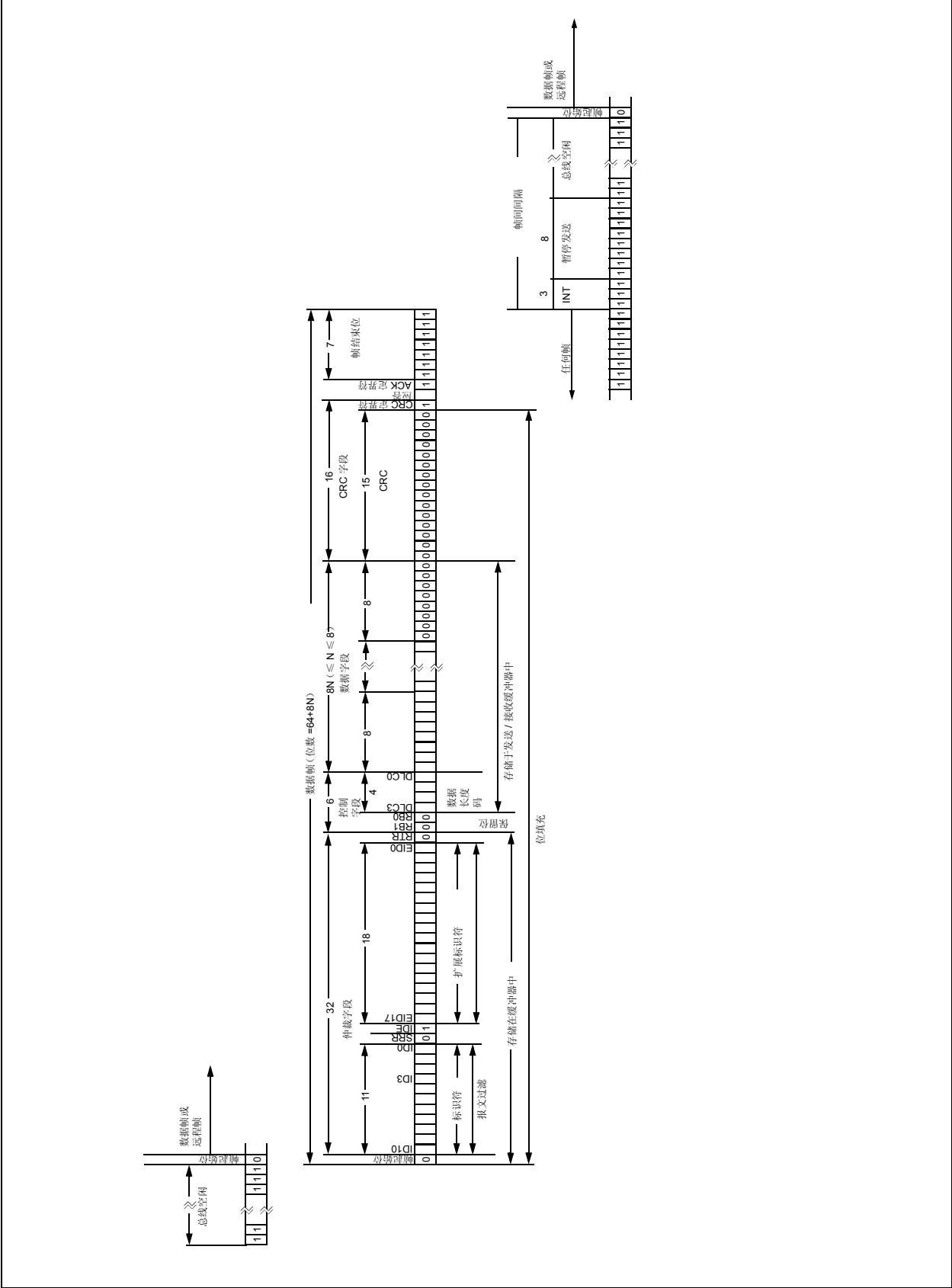


图 23-4: 扩展数据格式



远程数据帧

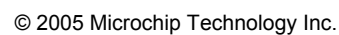
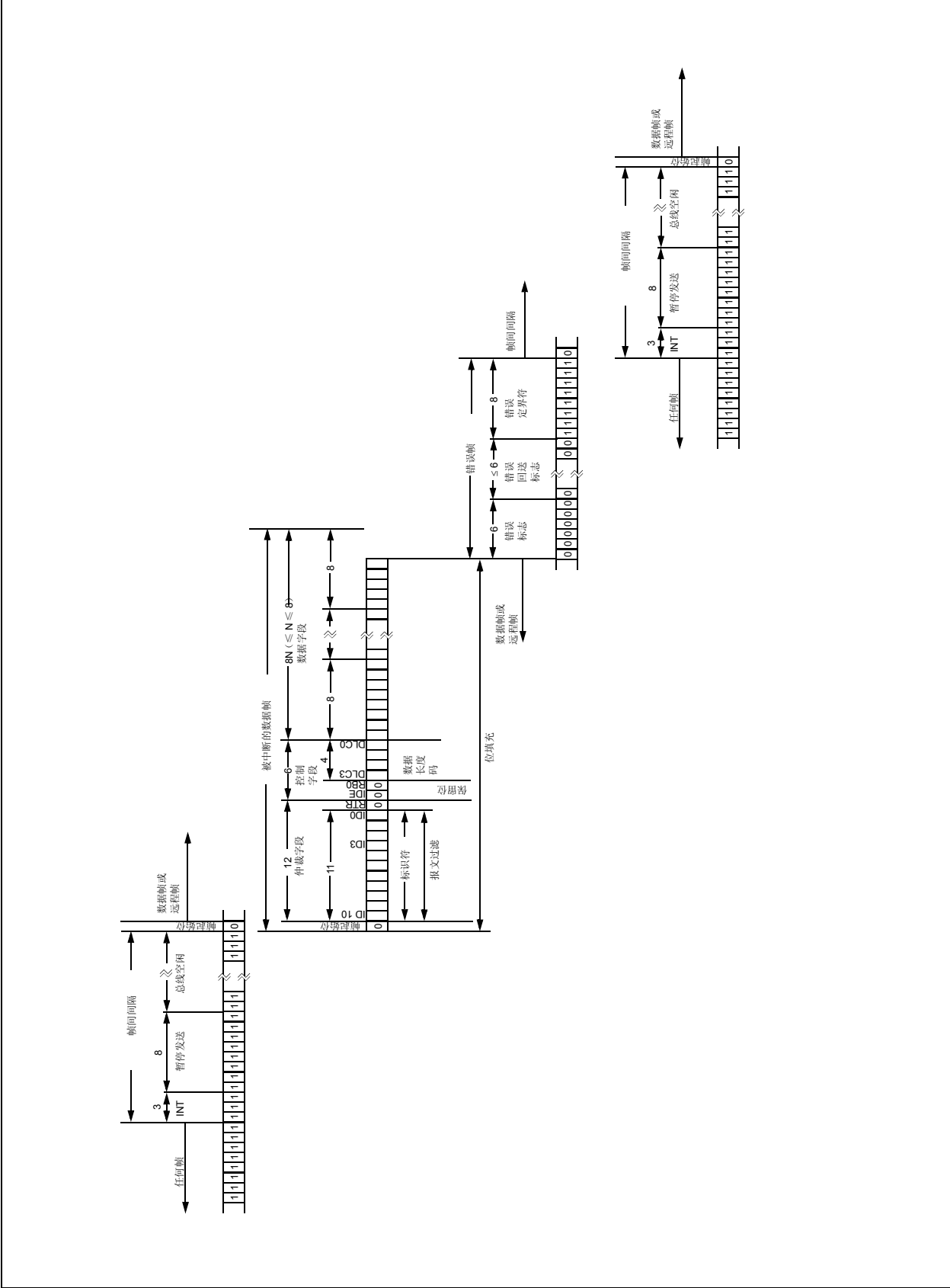


图 23-6: 错误帧



23.5 CAN 模块工作模式

用户可以选择 CAN 模块工作在以下几种工作模式之一。这些模式包括：

- 正常工作模式
- 禁止模式
- 环回模式
- 监听模式
- 配置模式
- 监听所有报文模式

通过设置 REQOP<2:0> 位 (CiCTRL<10:8>) 可选择所需模式。通过监视 OPMODE<2:0> 位 (CiCTRL<7:5>) 可以确认进入的模式。在允许模式改变之前，通常在由至少 11 位连续的隐性位确定的总线空闲时间内，模块不会改变模式和 OPMODE 位。

23.5.1 正常工作模式

当 REQOP<2:0> = 000 时选择正常工作模式，在这个模式下，模块被激活，I/O 引脚将承担 CAN 总线功能。模块发送和接收 CAN 总线报文的方式在后续各节中会有介绍。

23.5.2 禁止模式

在禁止模式中，模块将不会发送或接收。由于总线活动，模块能够置位 WAKIF 位，但是等待处理的中断将继续等待且错误计数器也将保持它们的值不变。

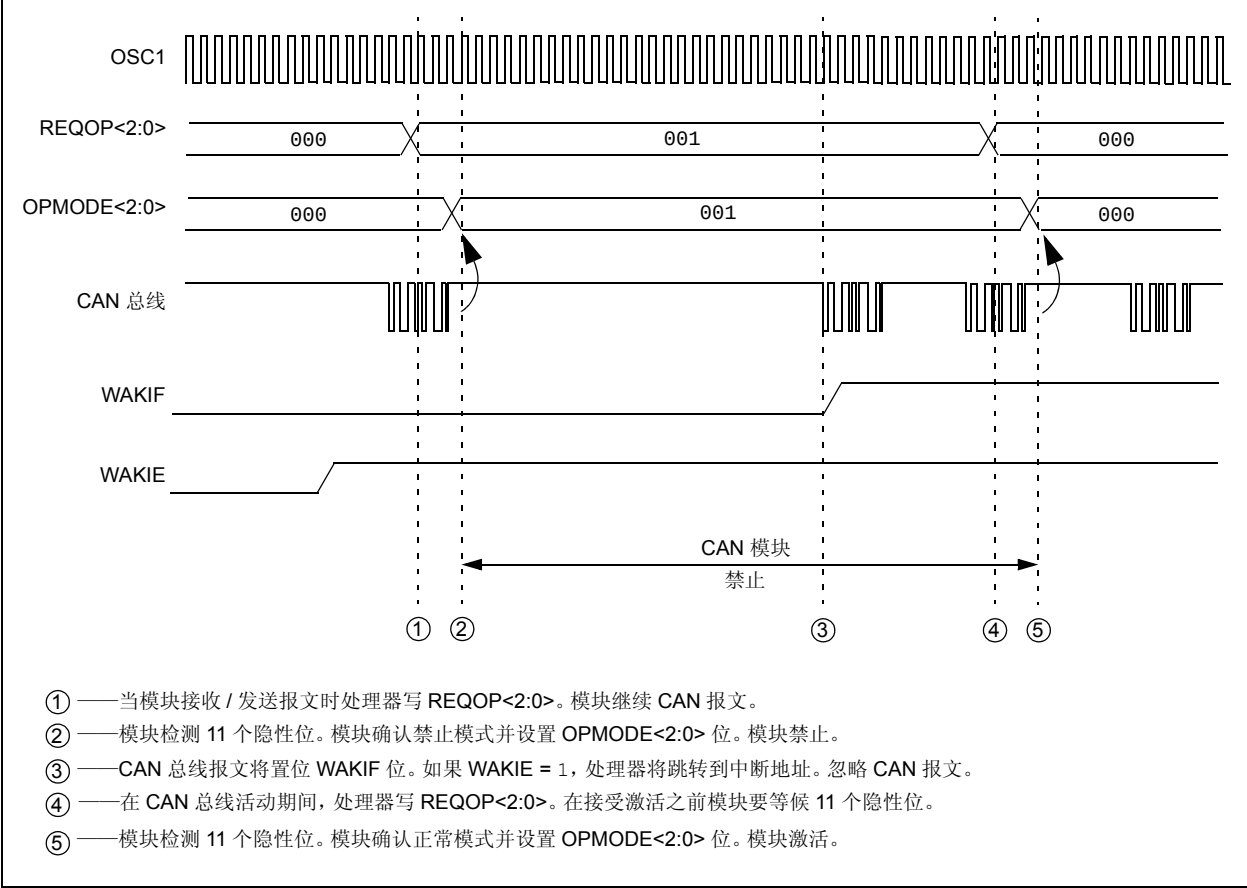
如果 REQOP<2:0> 位 (CiCTRL<10:8>) = 001，模块将进入模块禁止模式。该禁止模式与禁止其他外设模块相同，只要关闭该模块的使能就可以了。除非模块处于活动状态（即在接收或发送报文），否则禁止模块将导致模块内部时钟停止。如果该模块处于活动状态，它将等候 CAN 总线上的 11 位隐性位，检测到空闲总线状态，然后接受模块禁止命令。当 OPMODE<2:0> 位 (CiCTRL<7:5>) = 001，这表示模块成功进入了模块禁止模式（参见图 23-7）。

WAKIF 中断是在模块禁用模式中惟一仍然有效的模块中断。如果 WAKIE 位 (CiINTE<6>) 置位，不管 CAN 总线何时检测到显性状态，处理器都将收到一个中断，就象检测到帧起始 (SOF) 一样。

当模块在模块禁止模式时，I/O 引脚将恢复为普通 I/O 功能。

注：	通常，如果允许 CAN 模块在某种工作模式下发送，并且在 CAN 模块进入该模式后立即被要求发送，则模块将在开始发送前等待总线上出现 11 个连续隐性位。如果用户在此 11 位期间切换到禁止模式，则发送会被中止，同时相应的 TXABT 位置位，TXREQ 位清零。

图 23-7: 进入和退出模块禁止模式



23.5.3 环回模式

如果环回模式被激活，将在模块边沿连接内部发送信号和内部接收信号。发送和接收引脚将转换到它们的 I/O 端口功能。

发送器将为它发送的报文接收应答信号。特殊硬件会为发送器产生应答信号。

23.5.4 监听模式

监听模式和环回模式是正常工作模式的特例，允许系统进行调试。如果激活监听模式，CAN 总线上的模块是被动的。发送器缓冲器恢复为端口 I/O 功能。接收引脚保持为 CAN 模块的输入引脚。对于接收器，不发出错误标志或应答信号。该状态下，错误计数器失效。监听模式可用来检测 CAN 总线上的波特率。要使用它，必须有 2 个以上可以互相通信的节点。通过测试不同的值，可以凭经验检测波特率。此模式下模块也可用作总线监视器而不影响数据通信。

23.5.5 配置模式

在配置模式下，模块既不发送也不接收。此时错误计数器被清零，而且中断标志位保持不变。编程器将访问在其他模式访问受限的配置寄存器。

在器件复位后，CAN 模块处于配置模式（OPMODE<2:0> = 100）。错误计数器被清零且所有寄存器都为其复位值。应该确保初始化在 REQOP<2> 位清零前完成。

CAN 模块必须在激活之前被初始化。这只有在模块处于配置模式时才可能。通过将 REQOP<2> 位置位可以选择配置模式。只有当状态位 OPMODE<2> 为高逻辑级别时，才可以执行初始化。然后配置寄存器、接收屏蔽寄存器和接收过滤寄存器才可以被写入。通过将控制位 REQOP<2:0> 清零可以激活模块。

模块会防止用户因为编程错误而意外违反 CAN 协议。当模块在线时，所有控制模块配置的寄存器都不能被修改。当进行发送的时候，不允许 CAN 模块进入配置模式。配置模式可作为保护以下寄存器的锁。

- 所有模块控制寄存器
- 波特率和中断配置寄存器
- 总线定时寄存器
- 标识符接收过滤寄存器
- 标识符接收屏蔽寄存器

23.5.6 监听所有报文模式

监听所有报文模式是正常工作模式的特例，它允许系统进行调试。如果激活监听模式，CAN 总线上的模块是被动的。发送器缓冲器恢复为端口 I/O 功能。接收引脚保持在输入状态。对于接收器，不发出错误标志或应答信号。该状态下，错误计数器失效。过滤器禁止。接收缓冲器 0 将接收任何在总线上传输的报文。该模式对于将模块作为不影响数据通信的总线监视器，来记录所有的总线通信很有用。

23.6 报文接收

本节说明 CAN 模块的报文接收。

23.6.1 接收缓冲器

CAN 总线模块有 3 个接收缓冲器。但是，其中总是有一个缓冲器用于监视总线是否有进入的报文。这个缓冲器叫做报文合成缓冲器（Message Assembly Buffer，MAB）。因此只有 2 个接收缓冲器可见（RXB0 和 RXB1），基本上可以即时接收来自协议引擎的完整报文。当一个接收缓冲器在接收报文或保持上次接收到的报文时，CPU 仍可以使用另一个接收缓冲器工作。

MAB 保存来自总线的解填充比特流，以允许并行访问整个数据帧或远程帧，以进行接收匹配测试并将帧并行传输到接收缓冲器。MAB 将组合所有接收到的报文。这些报文只有在符合接收过滤器标准时才被传送到 RXBn 缓冲器。当接收到报文时，RXnIF 标志（CiINTF<0> 或 CiINRF<1>）将置位。此位只有在报文被接收时才被模块置位。该位在 CPU 处理完缓冲器中的报文后将由 CPU 清零。该位提供的正向锁定功能确保 CPU 已经完成了报文缓冲器的处理。如果 RXnIE 位（CiINTE<0> 或 CiINTE<1>）置位，当接收到报文时将会产生一个中断。

有 2 个与接收缓冲器相关的可编程的接收过滤屏蔽器，两个缓冲器各有一个。

当接收到报文时，FILHIT 位（接收缓冲器 0 的 CiRX0CON<0> 和接收缓冲器 1 的 CiRX1CON<2:0>）会表明报文的接收标准。除了表明使能接收的接收过滤屏蔽器个数外，还有一个指出所接收的报文是远程传输请求的状态位。

注： 对于接收缓冲器 0，可以使用有限数量的接收过滤器来使能接收。FILHIT0（CiRX0CON<0>）位决定使能报文接收的是两个过滤器中的哪一个（RXF0 还是 RXF1）。

23.6.1.1 接收缓冲器优先级

为了提供灵活性，每个接收缓冲器都有几个对应的接收过滤器。这也就意味着要给接收缓冲器赋予优先级。RXB0 是优先级较高的缓冲器，它有 2 个相关联的报文接收过滤器。RXB1 是优先级较低的缓冲器，它有 4 个相关联的报文接收过滤器。可用的接收过滤器的数量越少使得 RXB0 上的匹配限制越多，并且表明相关的缓冲器的优先级就越高。此外，如果 RXB0 包含有效报文，且有另一个有效报文被接收，RXB0 可以被设置为在这种情况下不会溢出，而是将到 RXB0 的新报文放入 RXB1。图 23-8 所示为接收缓冲器框图。而图 23-9 所示为接收操作的流程图。

图 23-8: 接收缓冲器

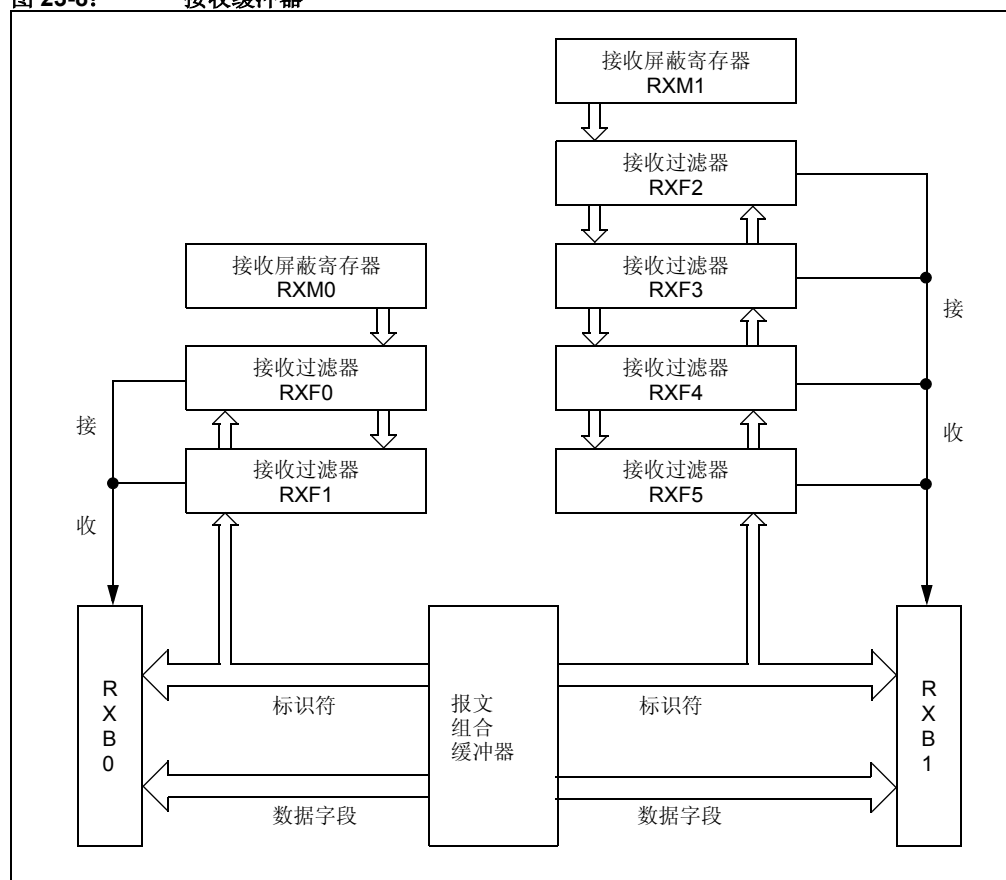
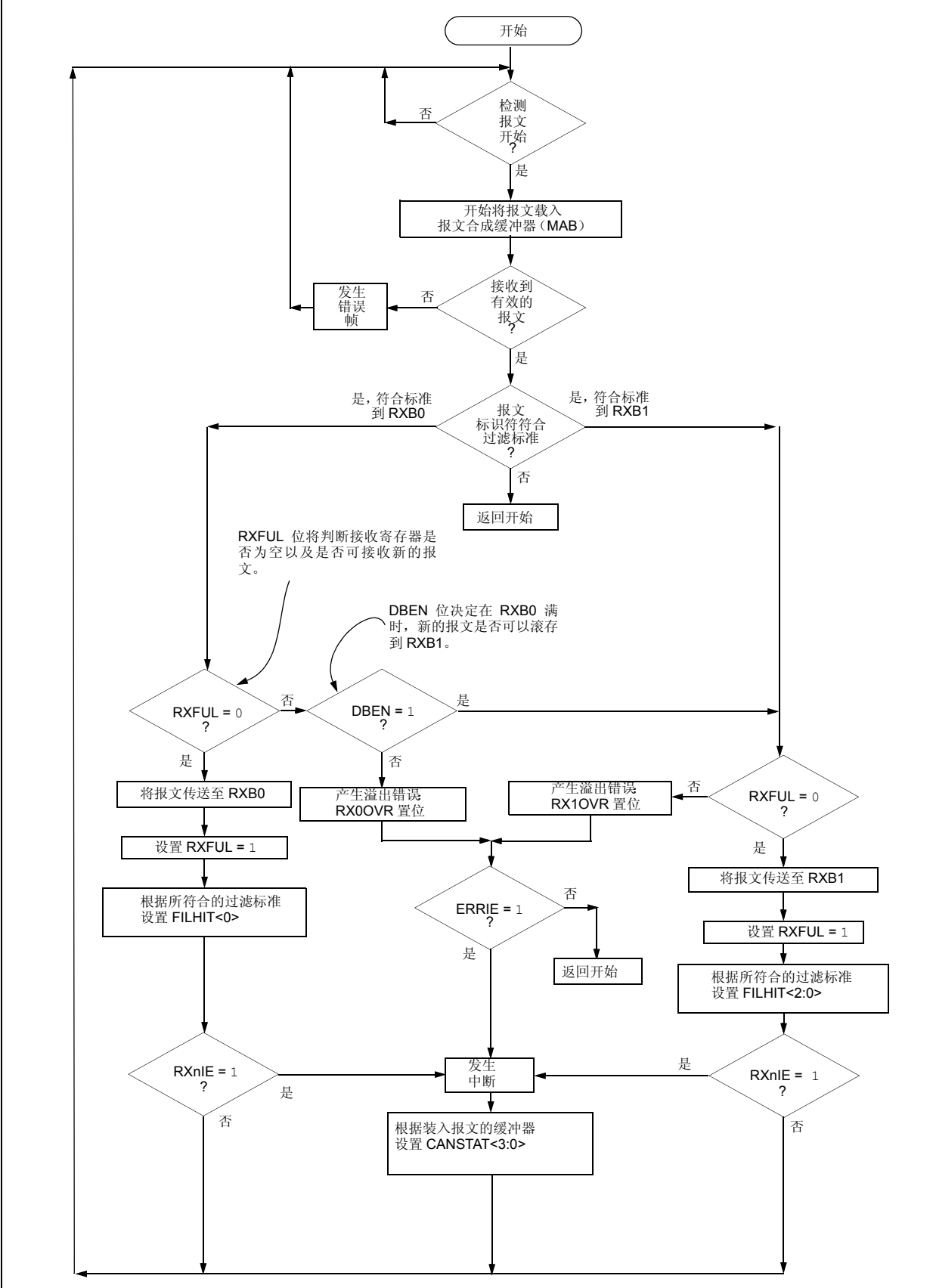


图 23-9: 接收流程图



23.6.2 报文接收过滤器

报文接收过滤器和屏蔽寄存器用于决定报文合成缓冲器中的一条报文是否应该被装入接收缓冲器中的一个。一旦一条有效报文被接收入报文合成缓冲器（MAB），就会将该报文的标识符字段与过滤值进行比较。如果匹配的话，该报文就会被装入相应的接收缓冲器。过滤屏蔽寄存器用于决定标识符中的哪些位将被过滤器检查。表 23-3 为真值表，它指出了标识符中的每个位是如何与屏蔽寄存器和过滤器进行比较来确定报文是否应该被装入接收缓冲器。屏蔽位基本上可以决定过滤器将应用哪些位。如果任何屏蔽位被置为零，无论过滤位为何值该位都会被自动接收。

表 23-3: 过滤 / 屏蔽真值表

屏蔽位 n	过滤位 n	报文标识符位	接收或拒绝位 n
0	x	x	接收
1	0	0	接收
1	0	1	拒绝
1	1	0	拒绝
1	1	1	接收

图注: x = 忽略

23.6.2.1 标识符模式选择

EXIDE 控制位（CiRXFnSID<0>）和 MIDE 控制位（CiRXMnSID<0>）为标准或扩展标识符使能接收过滤器。接收过滤器通过 RXIDE 位检查进入的报文以决定如何比较标识符。如果 RXIDE 位清零，报文是标准帧。如果 RXIDE 位置位，报文是扩展帧。

如果过滤器的 MIDE 控制位置位，则过滤器的标识符类型由过滤器的 EXIDE 控制位决定。如果 EXIDE 控制位清零，则过滤器将接收标准标识符。如果 EXIDE 位置位，则过滤器将接收扩展标识符。大部分 CAN 系统只使用标准标识符或只使用扩展标识符。

当过滤器的 MIDE 控制位清零时，如果过滤器位匹配，过滤器将接收标准和扩展标识符。此模式可用于在同一个总线上同时支持标准和扩展标识符的 CAN 系统。

23.6.2.2 FILHIT 状态位

如接收缓冲器框图图 23-8 所示，带 RXM0 屏蔽寄存器的 RXF0 和 RXF1 过滤器与 RXB0 相关。过滤器 RXF2、RXF3、RXF4、RXF5 和屏蔽寄存器 RXM1 则与 RXB1 相关。当过滤器匹配，报文被装入接收缓冲器时，使能报文接收的过滤器编号将通过 FILHIT 位在 CiRXnCON 寄存器中显示出来。CiRX0CON 寄存器包含一个 FILHIT 状态位，表明使能报文接收的是 RXF0 还是 RXF1 过滤器。CiRX1CON 寄存器包含 FILHIT<2:0> 位。它们的编码如表 23-4 所示。

表 23-4: 接收过滤器

FILHIT<2:0>	接收过滤器	注释
000 ⁽¹⁾	RXF0	仅当 DBEN = 1 时
001 ⁽¹⁾	RXF1	仅当 DBEN = 1 时
010	RXF2	—
011	RXF3	—
100	RXF4	—
101	RXF5	—

注 1: 仅当 DBEN 位被置位时有效。

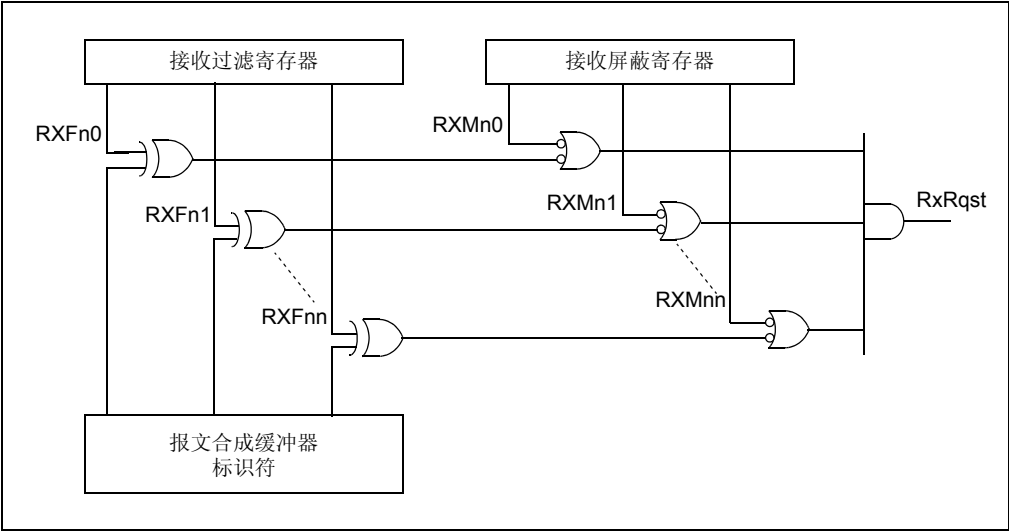
DBEN 位 (CiRX0CON<2>) 可以让 FILHIT 位区分保存在 RXB0 或滚存到 RXB1 中的数据是与过滤器 RXF0 还是与 RXF1 匹配。

- 111 = 接收过滤器 1 (RXF1)
- 110 = 接收过滤器 0 (RXF0)
- 001 = 接收过滤器 1 (RXF1)
- 000 = 接收过滤器 0 (RXF0)

如果 DBEN 位清零, 就会有 6 种编码与 6 个过滤器对应。如果 DBEN 位置位, 就有 6 种编码对应 6 个过滤器再加另外 2 种与滚存到 RXB1 的 RXF0 和 RXF1 过滤器对应的编码。

如果与一个以上的接收过滤器匹配, FILHIT 位中的二进制代码将反映其中编号最小的过滤器。换句话说, 如果过滤器 2 和过滤器 4 同时与接收报文匹配, FILHIT 将编码为过滤器 2。这实际上为编号较小的接收过滤器赋予较高的优先级。图 23-10 所示为报文接收过滤器的框图。

图 23-10: 报文接收过滤器



23.6.3 接收器溢出

当报文合成缓冲器（MAB）组合了一个有效的接收报文后，该报文通过接收过滤器接收，而此时与该过滤器相关的接收缓冲器还未指定将前一次接收的报文清除时，就会发生溢出。

溢出错误标志 RXnOVR（CiINTF<15> 或 CiINTF<14>）和 ERRIF 位（CiINTF<5>）将置位，MAB 中的报文会被丢弃。在溢出状态下，模块将保持与 CAN 总线同步而且能够发送报文，但模块将废弃所有发往溢出缓冲器的进入报文。

如果 DBEN 位清零，RXB1 和 RXB0 独立工作。在这种情况下，即使 RXB0 包含未读报文，为 RXB0 指定的报文也不会发送到 RXB1，但要置位 RX0OVR 位。

如果 DBEN 位置位，处理 RXB0 溢出的方法就不同了。如果接收到一条将要存入 RXB0 的有效报文，且 RXFUL = 1（CiRX0CON<7>）则表示 RXB0 满，如果 RXFUL = 0（CiRX1CON<7>）则表示 RXB1 为空，RXB0 的报文将装入 RXB1。此时不会为 RXB0 产生溢出错误。如果接收到一条将要存入 RXB0 的有效报文且 RXFUL 为 1，表示 RXB0 和 RXB1 位都为满，这样报文将丢弃同时指出 RXB1 溢出。

如果 DBEN 位清零，就会有 6 种编码与 6 个过滤器对应。如果 DBEN 位置位，就有 6 种编码对应 6 个过滤器再加另外 2 个与滚存到 RXB1 的 RXF0 和 RXF1 过滤器对应的编码。表 23-5 列出了这些编码。

表 23-5: 缓冲器接收和溢出真值表

报文 匹配 过滤器 0 或 1	报文 匹配 过滤器 2,3,4,5	RXFUL0 位	RXFUL1 位	DBEN 位	操作	结果
0	0	X	X	X	无	没有接收到报文
0	1	X	0	X	MAB →RXB1	RXB1 的报文，RXB1 可用
0	1	X	1	X	MAB 被丢弃 RX1OVR = 1	RXB1 的报文，RXB1 满
1	0	0	X	X	MAB →RXB0	RXB0 的报文，RXB0 可用
1	0	1	X	0	MAB 被丢弃 RX0OVR = 1	RXB0 的报文，RXB0 满， DBEN 未使能
1	0	1	0	1	MAB →RXB1	RXB0 的报文，RXB0 满， DBEN 使能，RXB1 可用
1	0	1	1	1	MAB 被丢弃 RX1OVR = 1	RXB0 的报文，RXB0 满， DBEN 使能，RXB1 满
1	1	0	X	X	MAB →RXB0	RXB0 和 RXB1 的报文， RXB0 可用
1	1	1	X	0	MAB 被丢弃 RX0OVR = 1	RXB0 和 RXB1 的报文， RXB0 满，DBEN 未使能
0	0	X	X	X	无	没有接收到报文
0	1	X	0	X	MAB →RXB1	RXB1 的报文，RXB1 可用

图注： X= 忽略

23.6.4 复位的影响

发生任何复位时，CAN 模块都必须被初始化。所有寄存器都根据复位值设置。收到的报文内容将被丢弃。第 23.5.5 节“配置模式”中讨论了初始化的过程。

23.6.5 接收错误

CAN 模块将会检测到以下接收错误：

- 循环冗余校验（CRC）错误
- 位填充错误
- 无效报文接收错误

这些接收错误不会产生中断。然而，当发生上述错误之一时，接收错误计数器会加 1。RXWAR 位（CiINTF<9>）指出接收错误计数器已经达到 CPU 警告的上限值 96，接着发出中断。

23.6.5.1 循环冗余校验（CRC）错误

使用循环冗余校验时，发送器为从帧起始位到数据字段结束的位序列计算特殊校验位。CRC 序列在 CRC 字段中发送。接收节点使用同一个公式计算 CRC 序列，并且将计算结果与已接收到的序列比较。如果检测到两者不匹配，说明发生了 CRC 错误，然后产生一个错误帧。报文将被重发。接收错误中断计数器加 1。仅当错误计数器的值超过阈值时发出中断。

23.6.5.2 位填充错误

如果在帧起始位和 CRC 定界符之间检测到 6 个连续的同极性位，那么就违反了位填充规则。发生位填充错误，会产生错误帧。报文将被重发。产生位填充错误将不会产生中断。

23.6.5.3 无效报文接收错误

如果在接收报文期间发生任何类型的错误，错误类型将通过 IVRIF 位（CiINTF<7>）标示出来。当器件处于监听模式时，该位可用于自动波特率检测（中断时也可选用）。该错误标志并不表示需要采取任何动作，而是指出在 CAN 总线上已经发生了错误。

23.6.5.4 修改接收错误计数器的规则

接收错误计数器根据以下规则进行修改：

- 当接收器检测到一个错误时，除非检测到的错误是在发送主动错误标志时的一个位错误，否则接收错误计数器将加 1。
- 如果接收器在发送一个错误标志后检测到的第一个位为“显性”位，接收错误计数器将加 8。
- 如果接收器在发送主动错误标志时检测到位错误，接收错误计数器将加 8。
- 所有的节点在发送了主动错误标志或被动错误标志后最多允许 7 个连续“显性”位。在检测到第 14 个连续的“显性”位（对于主动错误标志）或在被动错误标志后检测到第 8 个连续“显性”位，以及其他的每八个连续“显性”位序列时，每个发送器会将其发送错误计数器加 8，每个接收器也将其接收错误计数器加 8。
- 在成功接收了报文（在 ACK 间隙之前接收无误并成功发送了 ACK 位）后，如果接收错误计数器在 1 到 127 之间，接收错误计数器会减 1。如果接收错误计数器为 0，则保持不变。如果接收错误计数器大于 127，它将变为 119 到 127 之间的一个值。

23.6.6 接收中断

有几个中断与报文接收有关。接收中断可分为两类：

- 接收错误中断
- 接收中断

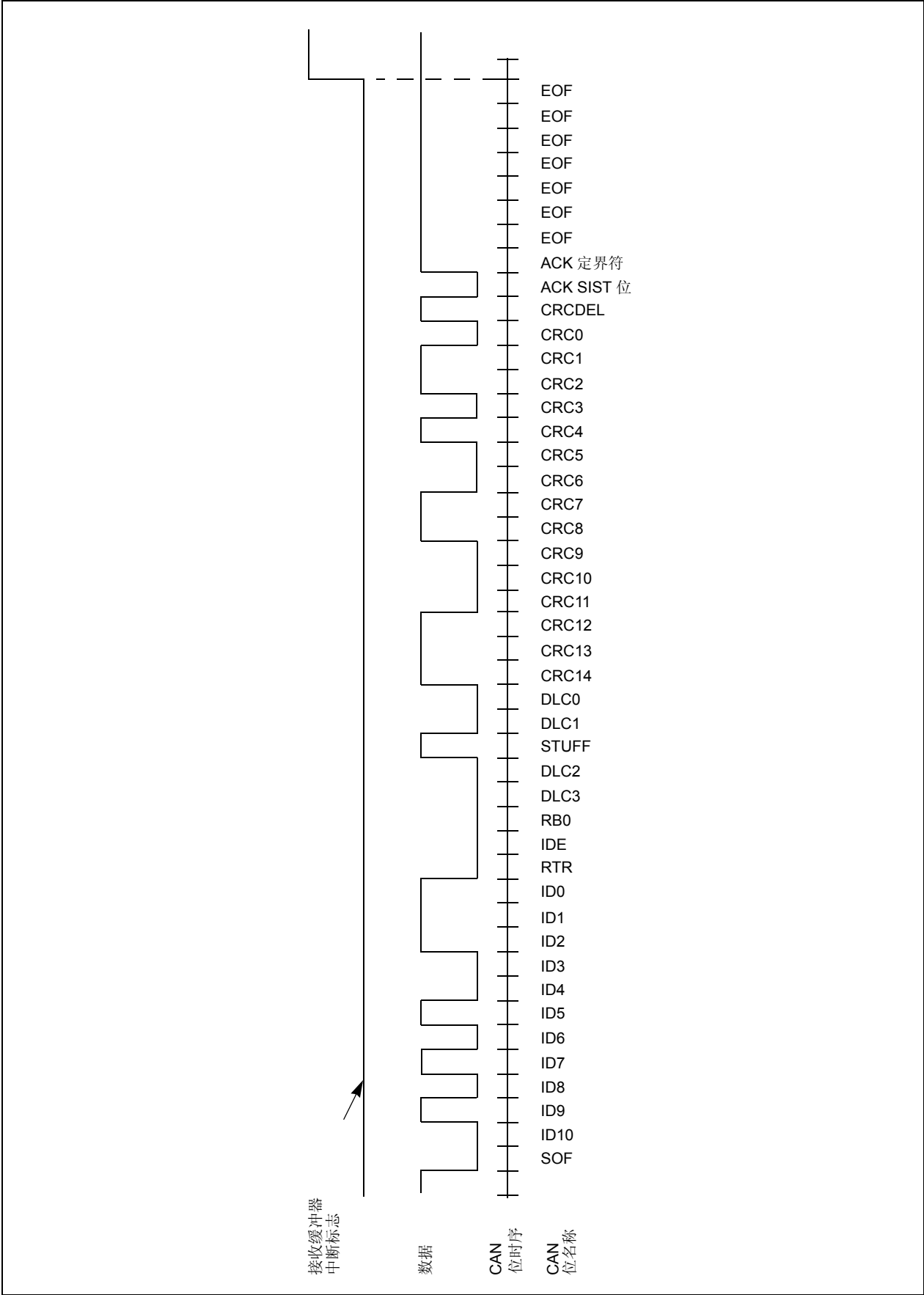
23.6.6.1 接收中断

报文已被成功接收并被装入一个接收缓冲器。接收到帧结束（EOF）字段后，中断立即被激活。读 **RXnIF** 标志位可知哪个接收缓冲器引起了中断。图 23-11 描述了什么时候接收缓冲中断标志位将被置位。

23.6.6.2 唤醒中断

第 23.13.1 节“休眠模式下的工作”中有唤醒中断过程的介绍。

图 23-11: 接收缓冲器中断标志



23.6.6.3 接收错误中断

接收错误中断由 **ERRIF** 位 (**CiINTF<5>**) 标示。该位表示有错误情况发生。通过检查 **CAN** 中断状态寄存器 **CiINTF** 的相应位, 就可以确定错误源。该寄存器中的位与接收和发送错误有关。下面将说明与接收错误相关的标志位。

23.6.6.3.1 无效报文接收中断

如果在接收上一个报文期间发生了任何类型的错误, **IVRIF** 位 (**CiINTF<7>**) 都将指出有错误发生。但具体发生的是何种类型的错误则不得而知。当器件处于监听模式时, 该位可用于自动波特率检测 (中断时可选用)。该错误标志位并不表示需要采取任何动作, 而是表示在 **CAN** 总线上已经发生了错误。

23.6.6.3.2 接收器溢出中断

RXnOVR 位 (**CiINTF<15>** 和 **CiINTF<14>**) 表明接收缓冲器发生了溢出情况。当报文合成缓冲器 (**MAB**) 组合了一个有效的接收报文后, 该报文通过接收过滤器接收, 但是与该过滤器相关的接收缓冲器没有清除上一次接收的报文, 这就发生了溢出的情况。溢出错误中断标志位将被置位, 而报文将被丢弃。而在溢出情况下, 模块仍与 **CAN** 总线保持同步, 并且能够发送和接收报文。

23.6.6.4 接收器警告中断

RXWAR 位 (**CiINTF<8>**) 用于指出接收错误计数器的值已经达到 CPU 警告的极限值 96。当 **RXWAR** 位由 0 变为 1 时, 它会使错误中断标志位 **ERRIF** 置 1。此时, **ERRIF** 位不能手工清零, 因为它应当保留以指出接收错误计数器的值已经达到 CPU 警告极限值 96。这时如果接收错误计数器的值变成小于或等于 95, **RXWAR** 位将自动清零。可以手工清零 **ERRIF** 位以允许在不影响 **RXWAR** 位的情况下, 退出中断服务程序。

23.6.6.5 接收器错误被动

RXBP 位 (**CiINTF<11>**) 表明接收错误计数器已经超过了错误被动的极限值 127, 且该模块已经进入错误被动状态。当 **RXEP** 位由 0 变为 1 时, 将导致错误中断标志置位。**RXEP** 位不能手工清零, 因为此时该位应当保留以指出总线处于错误被动状态。此时当接收错误计数器的值小于或等于 127 时, **RXEP** 位将自动清零。可以手工清零 **ERRIF** 位以允许退出中断服务程序而不影响 **RXBP** 位。

23.7 发送

本节介绍如何使用 CAN 模块来发送 CAN 报文。

23.7.1 实时通信和发送报文缓冲

为了使应用能实时有效地发送报文，CAN 节点必须能够控制和保持总线（假定节点报文有足够高的优先级来赢得总线仲裁）。如果一个节点只有一个发送缓冲器，它必须先发送一个报文，然后在 CPU 重新装载缓冲器时释放总线。如果一个节点有两个发送缓冲器，则可以在一个缓冲器发送报文的同时另一个缓冲器正在被重新装载。但是 CPU 必须保持紧密跟踪总线活动，确保另一个缓冲器在第一缓冲器完成报文发送前重新载入。

典型的应用需要三个发送报文缓冲器。如果有三个缓冲器，第一个正在发送报文时，第二个则准备等到第一个发送完就立即开始发送，而第三个被 CPU 重新载入。这样减轻了用软件维持总线同步的负担（参见图 23-12）。

另外，拥有三个缓冲器可以在一定程度上提高外发报文的优先级。比方说，应用软件有可能当其在第三个缓冲器上工作时将一个报文在第二缓冲器中排队。应用可能要求进入第三缓冲器的报文比已经在第二缓冲器排序的报文更重要。如果只有两个缓冲器可用，那么不得不将已排序的报文删除而用重要性更高的报文取代。删除报文的过程可能意味着失去对总线的控制。有了三个缓冲器，第二和第三缓冲器的报文都可以排序，且模块可以被设置为第三缓冲器中的报文比第二缓冲器报文具有更高的优先级。那么发送完第一个报文后将发送第三个报文然后紧跟着发送第二个报文。

23.7.2 发送报文缓冲器

CAN 模块有三个发送缓冲器。每个缓冲器可容纳 14 字节的数据。其中的 8 个字节用于存放发送的报文（最大 8 个字节）。另外 5 个字节用来存放标准或扩展报文标识符和其他报文仲裁信息。

最后一个字节是与每个报文相关的控制字节。该字节中的信息决定在何种情况下报文将被发送以及表示报文发送的状态。

TXnIF 位（CiINTF<2>、CiINTF<3> 或 CiINTF<4>）将被置位，并且 TXREQ 位（CiTXnCON<3>）清零，表明报文缓冲器完成了发送。随后 CPU 把将要发送的报文内容装入报文缓冲器。至少，必须装载标准标识符寄存器 CiTXnSID。如果报文中含有数据字节，还应装载 TXBnDm 寄存器。如果报文使用扩展标识符，CiTXnEID 寄存器和 EID<5:0> 位（CiTXnDLC<15:10>）会被加载并置位 TXIDE 位（CiTXnSID<0>）。

在发送报文前，用户必须初始化 TXnIE 位（CiINTE<2>、CiINTE<3> 或 CiINTE<4>）以便在发送报文后使能或禁止中断。用户还必须初始化发送优先级。图 23-12 所示是发送缓冲器的框图。

23.7.5 发送报文中止

通过清零与各个报文缓冲器相关的 TXREQ 位，系统能中止报文发送。置位 ABAT 位 (CiCTRL<12>) 将请求中止所有待发送报文 (参见图 23-14)。清零 TXREQ 位中止一个排队的报文。图 23-13 是中止排队报文的图示。如果报文还未开始发送或者报文已开始发送但由于仲裁失败或错误而被中断，那么将中止。当模块置位 TXABT 位 (CiTXnCON<6>) 且 TXnIF 标志位不置位时，表明发生了中止。

如果报文已经开始发送，此时就会试图让当前报文完整发送 (参见图 23-15)。如果当前报文被完整地发送，并且没有仲裁失败或出现错误，那么 TXABT 位将由于报文成功发送而不被置 1。类似地，如果报文在发生中止请求期间发送，且该报文仲裁失败 (参见图 23-16) 或出现了错误，那么该报文将不会被重新发送，而且 TXABT 位将置 1，表示成功中止报文。

图 23-13: 中止排队报文

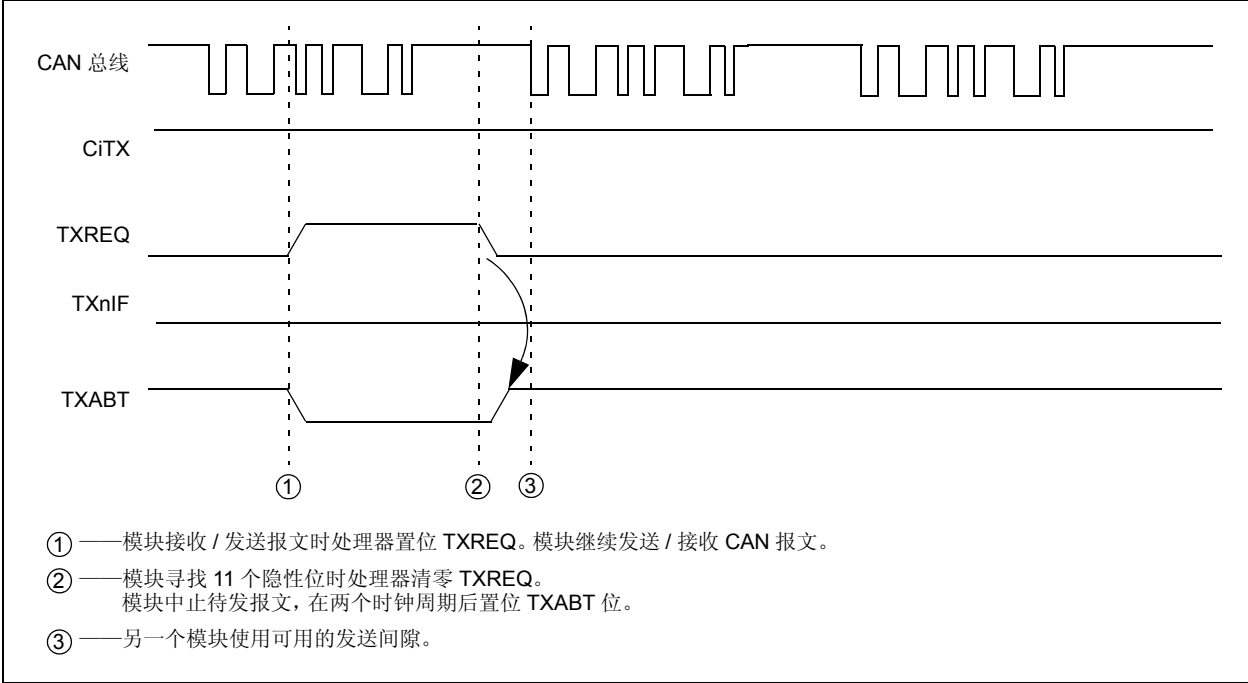


图 23-14: 中止所有报文

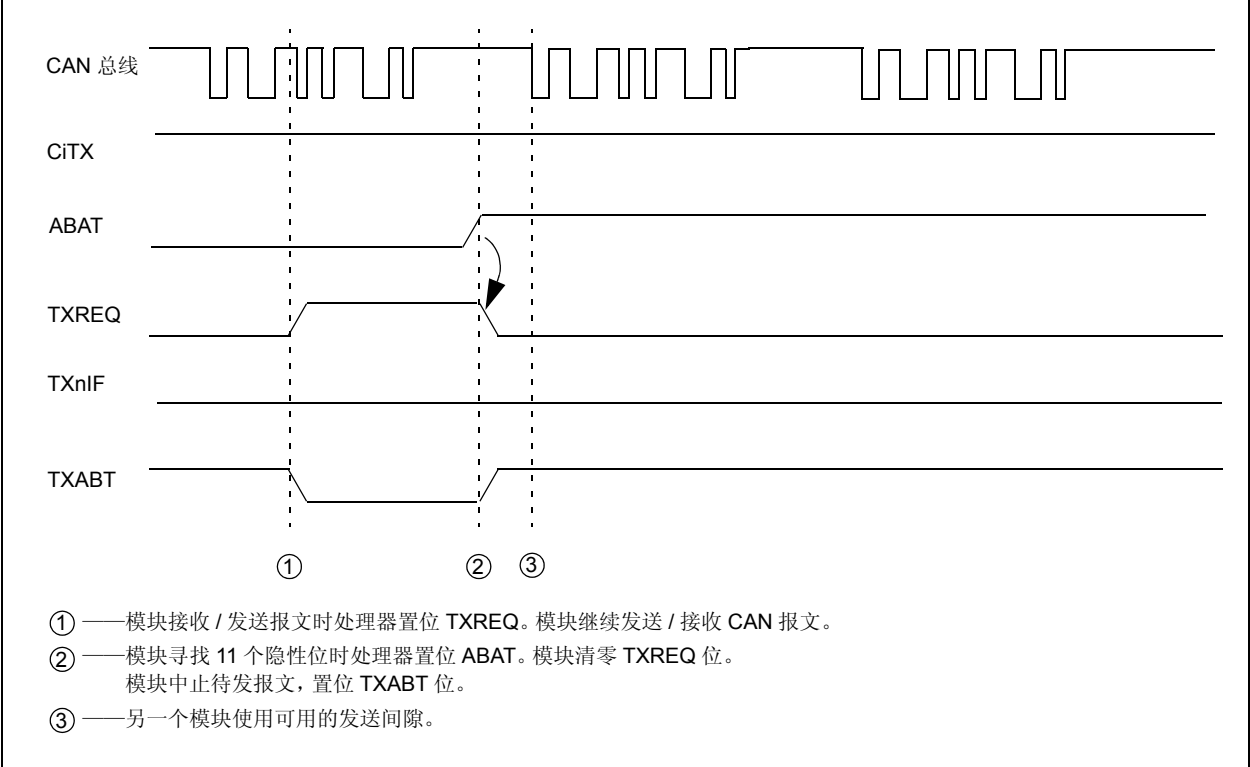


图 23-15: 发送期间失败的中止

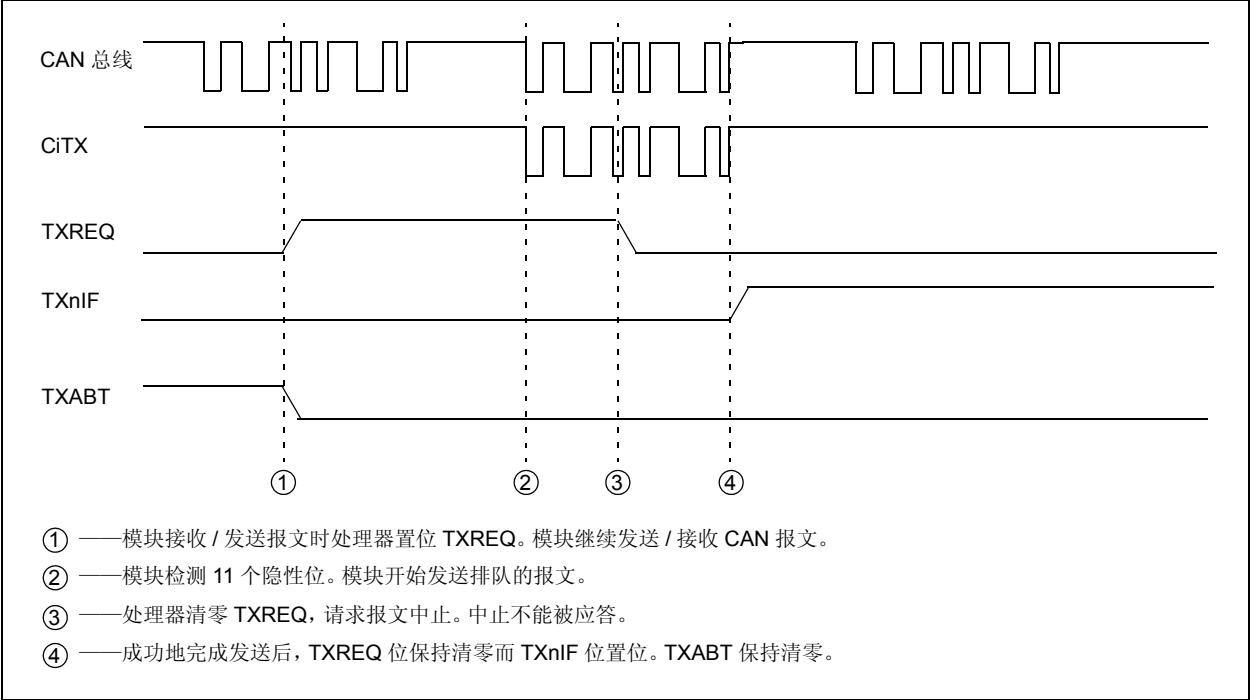


图 23-16: 发送过程中仲裁失败

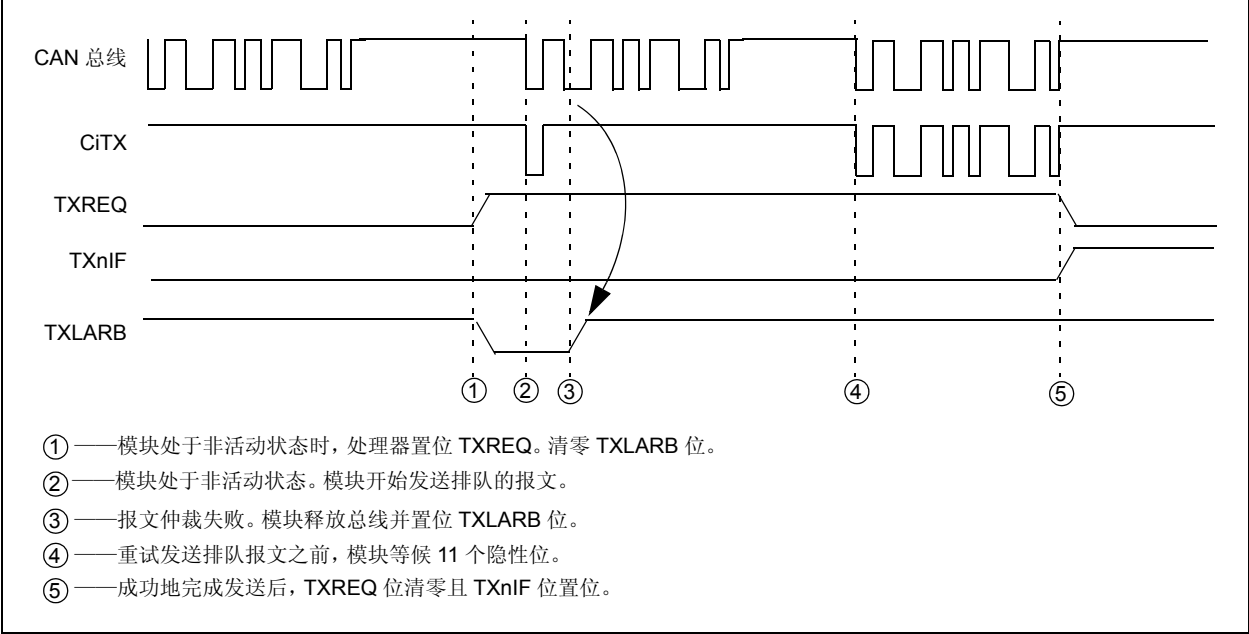
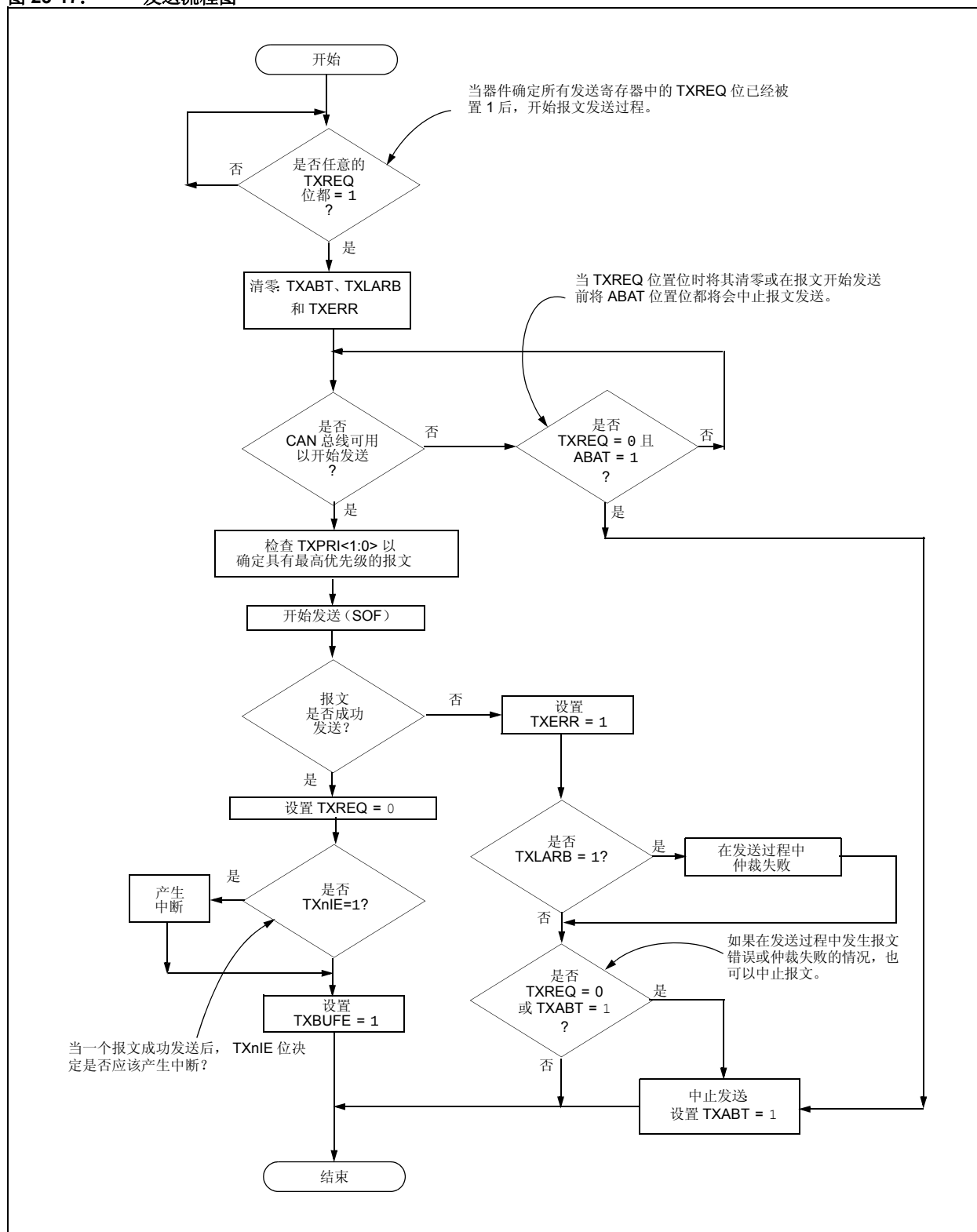


图 23-17: 发送流程图



23.7.6 发送边界条件

模块处理与那些不必与 CAN 总线报文成帧时间同步的发送命令。

23.7.6.1 当报文开始发送时，清零 TXREQ 位

当报文刚开始发送时，就可以清零 TXREQ 位以中止报文发送。如果报文没有正式开始被发送，TXABT 位将被置位，表示成功执行了中止。

当用户清零了 TXREQ 位，且两个周期后，TXABT 位也不被置位，那么报文已经开始发送了。

如果报文正在发送，中止将不会立即执行，而是过一段时间后执行并将 TXnIF 中断标志位或 TXABT 位置 1。如果已经开始发送报文，只有当发生错误或仲裁失败时才会中止报文。

23.7.6.2 当报文开始发送时，置位 TXABT 位

置位 TXABT 位将会中止所有等待发送的缓冲器，并具有清零所有寄存器中 TXREQ 位的功能。边界条件与清零 TXREQ 位相同。

23.7.6.3 当报文完成发送时，清零 TXREQ 位

当报文发送即将成功完成时，可以清零 TXREQ 位。即使数据总线已在报文成功完成发送前一小段时间将 TXREQ 位清零（它本将在成功完成报文发送后清零），TXnIF 标志位由于成功完成发送仍将置位。

23.7.6.4 当报文完成发送时，置位 TXABT 位

边界条件与清零 TXREQ 位相同。

23.7.6.5 当报文发送失败时，清零 TXREQ 位

当报文即将仲裁失败或发生错误时，可以清零 TXREQ 位。

如果 TXREQ 信号在仲裁失败信号或错误信号产生前下降，那么结果如同在报文发送过程中清零 TXREQ 一样。当仲裁失败或出现错误时，TXABT 位将置 1，表示发送时发生了错误，但是 TXREQ 位不置 1。

如果 TXREQ 信号在仲裁信号产生后下降，那么结果如同在非活动发送时间内清零 TXREQ 一样。TXABT 位将置位。

23.7.6.6 当报文发送失败时，置位 TXABT 位

边界条件与清零 TXREQ 位相同。

23.7.7 复位的影响

发生任何复位时，CAN 模块都必须被初始化。根据复位值设置所有寄存器。发送的报文内容将被丢失。第 23.5.5 节“配置模式”中有关于初始化的讨论。

23.7.8 发送错误

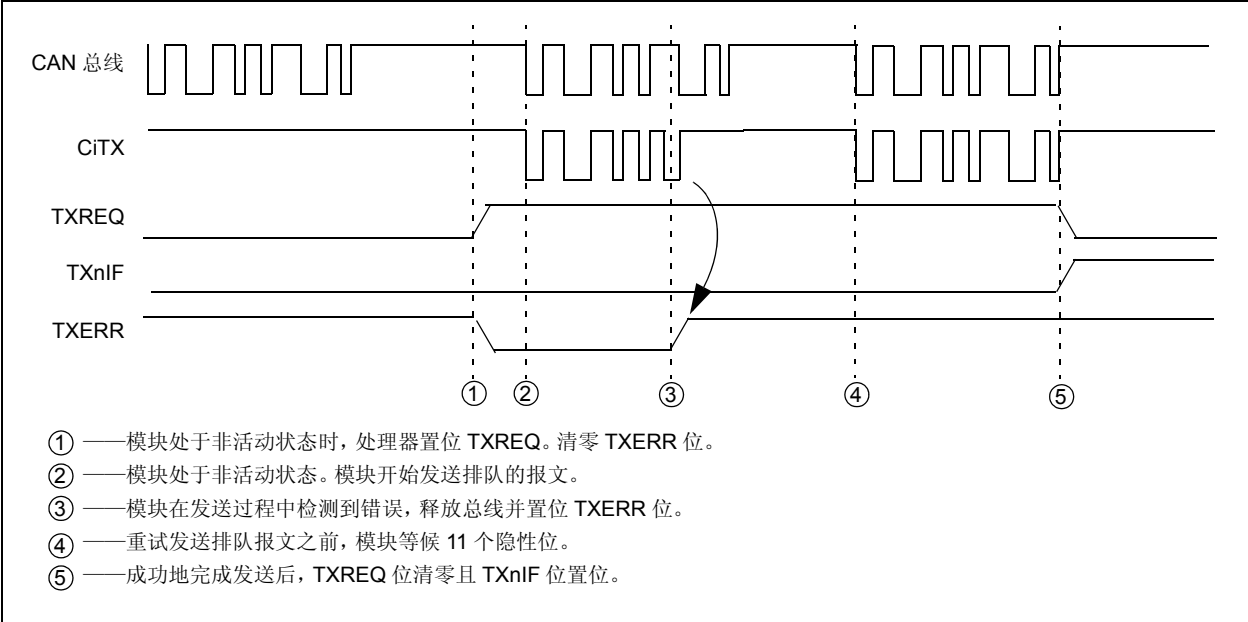
CAN 模块将会检测到以下发送错误：

- 应答错误
- 格式错误
- 位错误

这些错误不一定会产生中断，但是发送错误计数器将会计数以表示有错误发生。每个错误将引起错误计数器的值加 1。一旦错误计数器的值超过 96，ERRIF (CiINTF<5>) 和 TXWAR 位 (CiINTF<10>) 将被置 1。一旦错误计数器的值超过 96，将产生中断并将错误标志寄存器中的 TXWAR 位置 1。

图 23-18 是发送错误示例的图示。

图 23-18: 发送过程中的错误



23.7.8.1 应答错误

在报文的应答字段，发送器检查应答间隙（已作为一个隐性位发送）是否包含一个显性位。如果不是，说明没有其他节点正确接收了该帧。发生了应答错误，报文必须重发。不产生错误帧。

23.7.8.2 格式错误

如果发送器检测到四个段（包括帧结束字段、帧间间隔、应答定界符和 CRC 定界符）之一中有一个显性位，则表示发生了格式错误并将产生一个错误帧。报文将被重发。

23.7.8.3 位错误

如果发送器发送了一个显性位并检测到一个隐性位，则发生了位错误。在发送器发送了一个隐性位，而在仲裁字段和应答间隙期间检测到一个显性位的情况下，不会产生位错误，因为此时正在进行正常的仲裁。

23.7.8.4 修改发送错误计数器的规则

发送错误计数器根据以下规则进行修改：

- 当发送器发送一个错误标志时，发送错误计数器的值加 8，但以下情况例外。在这两种例外情况中，不更改发送错误计数器。
 - 若发送器为“错误被动”，并检测到一个应答错误，此应答错误是由检测不到一个“显性”ACK 以及当发送被动错误标志时检测不到一个“显性”位而引起的。
 - 发送器因为在仲裁期间发生位填充错误而发送错误标志。引起位填充错误的原因是：填充位位于 RTR 位之前，本该是“隐性”并已作为“隐性”发送，但是却被监视为“显性”。
- 当发送一个主动错误标志时，如果发送器检测到一个位错误，发送错误计数器的值加 8。
- 在发送了一个主动错误标志或被动错误标志之后，任何节点最多容许 7 个连续的“显性”位。当检测到第 14 个连续的“显性”位后（发送了一个主动错误标志后）；在检测到第 8 个跟随着被动错误标志的连续的“显性”位以后或在每个具有 8 个其他连续的“显性”位的序列之后，每个发送器将它们的发送错误计数器加 8 且每个接收器将它们的接收错误计数器加 8。
- 当报文成功发送后（得到了应答且直到帧结束字段传输完成都无错误发生），发送错误计数器的值减 1，除非该值已经为 0。

23.7.9 发送中断

有几个中断与报文发送相关联。发送中断可分为两类：

- 发送中断
- 发送错误中断

23.7.9.1 发送中断

三个发送缓冲器中至少有一个为空（未预定）并且可以装入预定发送的报文。读 CiINTF 寄存器中的 TXnIF 标志位可知哪一个发送缓冲器可用并引起了中断。

23.7.9.2 发送错误中断

发送错误中断由 **ERRIF** 标志位表示。该标志位表示发生了错误情况。通过检查 **CAN** 中断状态寄存器 **CiINTF** 中的错误标志，就可以确定错误的起源。该寄存器中的标志位与接收和发送错误有关。

TXWAR 位 (**CiINTF<10>**) 表示发送错误计数器的值已经达到 **CPU** 警告的上限值 **96**。当 **TXWAR** 位由 **0** 变为 **1** 时，它会导致错误中断标志位置 **1**。**TXWAR** 位不可手工清零，因为该位应当保留，以标志发送错误计数器的值已经达到 **CPU** 警告上限值 **96**。如果发送错误计数器的值变成小于或等于 **95**，**TXWAR** 位将自动清零。可以手动清零 **ERRIF** 标志位以允许在不影响 **TXWAR** 位的情况下，退出中断服务程序。

TXEP 位 (**CiINTF<12>**) 用来表示发送错误计数器已经超过了错误被动的上限值 **127**，且该模块已经进入了错误被动状态。当该位从 **0** 变为 **1** 时，将导致错误中断标志位置 **1**。**TXEP** 位不能手动清零，因为该位应当被保留以标志总线处于错误被动状态。当发送错误计数器的值变成小于或等于 **127** 时，**TXEP** 位将自动清零。可以手动清零 **ERRIF** 位以允许在不影响 **TXEP** 位的情况下，退出中断服务程序。

TXBO 位 (**CiINTF<13>**) 表示发送错误计数器的值已经超过了 **255**，且该模块已经进入总线关断状态。当该位从 **0** 变为 **1** 时，将导致错误中断标志位置 **1**。**TXBO** 位不能手动清零，因为该位应当被保留以标志总线关断。可以手动清零 **ERRIF** 标志位以允许在不影响 **TXBO** 位的情况下，退出中断服务程序。

23.8 错误检测

CAN 协议提供了成熟的错误检测机制。能检测到以下错误。这些错误有些是接收错误，有些是发送错误。

接收错误有：

- 循环冗余校验 (**CRC**) 错误 (参见第 23.6.5.1 节 “循环冗余校验 (**CRC**) 错误”)
- 位填充位错误 (参见第 23.6.5.2 节 “位填充错误”)
- 无效报文接收错误 (参见第 23.6.5.3 节 “无效报文接收错误”)

发送错误有：

- 应答错误 (参见第 23.7.8.1 节 “应答错误”)
- 格式错误 (参见第 23.7.8.2 节 “格式错误”)
- 位错误 (参见第 23.7.8.3 节 “位错误”)

23.8.1 错误状态

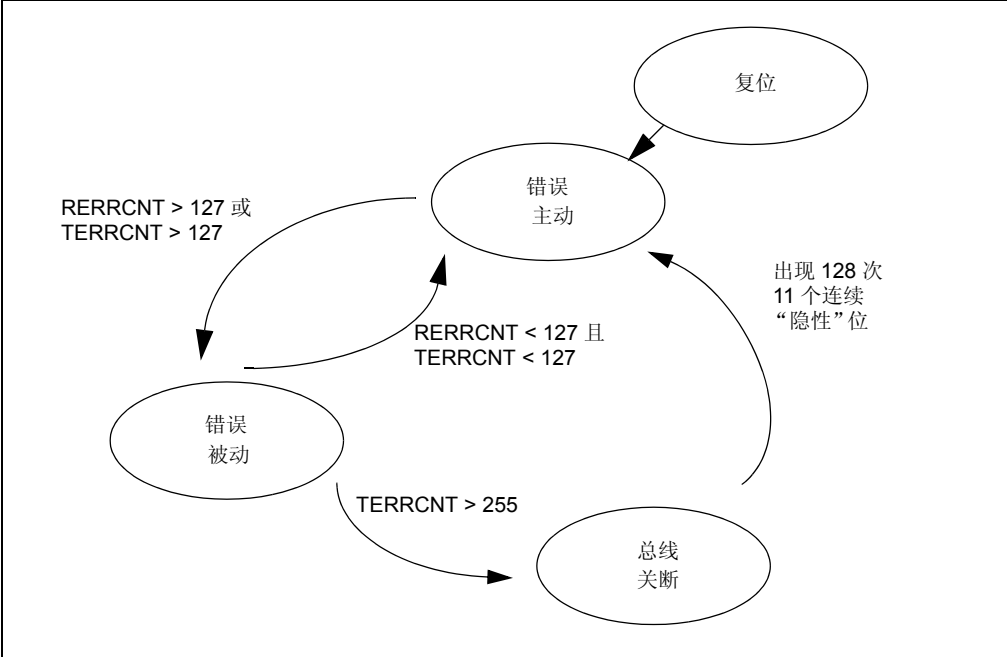
通过发出错误帧将检测到的错误通知所有其他节点。中止错误报文的发送并尽快重发该帧。进一步说，根据内部错误计数器的值，各个 **CAN** 节点处于三种错误状态中的一种：即“错误主动”、“错误被动”或“总线关断”。错误主动状态是常见的状态，在该状态，总线节点能够无限制地发送报文和主动错误帧（由显性位组成）。在错误被动状态，可以发送报文和被动错误帧（由隐性位组成）。总线关断状态使站暂时不能参加总线通信。在该状态，既不能接收报文，也不能发送报文。

23.8.2 错误模式和错误计数器

CAN 控制器包含两个错误计数器，即接收错误计数器（RERRCNT）和发送错误计数器（TERRCNT）。两个计数器的值都能由 CPU 从错误计数寄存器 CiEC 读取。这些计数器的值根据 CAN 总线规范进行递增或递减。

如果两个错误计数器的值都低于错误被动的极限值 128，则 CAN 控制器处于错误主动状态。当至少其中一个错误计数器的值等于或超过 128 时，CAN 控制器处于错误被动状态。若发送错误计数器的值等于或超过总线关断的极限值 256，CAN 控制器进入总线关断状态。器件将保持该状态直到完成总线关断恢复过程，即出现 128 次 11 个连续隐性位。此外，还有一个错误状态警告标志位 EWARN（CiINTF<8>），若至少其中一个错误计数器的值等于或超过错误警告极限值 96，该位置 1。如果两个错误计数器的值都小于错误警告极限值，那么 EWARN 复位。

图 23-19： 错误模式



23.8.3 错误标志寄存器

错误标志寄存器中的值表示出是哪种错误导致错误中断标志置位。RXnOVR 错误标志位（CiINTF<15> 和 CiINTF<14>）与该寄存器中的其他错误标志位相比，有一个不同的功能。为了清零 ERRIF 中断标志位必须清零 RXnOVR 位。当发送错误计数器和接收错误计数器值超过一个特定的阈值时，该寄存器中的其他错误标志位将导致 ERRIF 中断标志位置 1。在上述情况清零 ERRIF 中断标志位将允许退出中断服务程序，而不发生递归中断。当错误计数器的值在阈值附近上下波动时，若出现一些特定中断，最好在它们发生一次中断后加以禁止，以防止器件反复中断。

23.9 CAN 波特率

任意特定 CAN 总线上的所有节点必须有相同的标称比特率。CAN 总线使用 NRZ 编码，该编码不为时钟编码。因此接收器的独立时钟必须通过接收节点恢复并与发送器时钟同步。

为了设置波特率，必须初始化以下位：

- 同步跳转宽度位（参见第 23.9.6.2 节“重新同步”）
- 波特率预分频器位（参见第 23.9.2 节“预分频器设置”）
- 相位段位（参见第 23.9.4 节“相位段”）
- 相位段 2 的长度决定位（参见第 23.9.4 节“相位段”）
- 采样点位（参见第 23.9.5 节“采样点”）
- 传播段位（参见第 23.9.3 节“传播段”）

23.9.1 位时序

由于各节点的振荡器和发送时间可能各不相同，所以接收器必须使某种类型的 PLL 与数据发送沿同步，以同步和保持接收器时钟。因为数据是用 NRZ 码编码的，所以必须包含位填充以确保至少每 6 个位时间出现一次同步沿，以保持数字锁相环（Digital Phase Lock Loop, DPLL）同步。

在位时间帧中执行的总线时序功能，诸如与本地振荡器同步、网络发送延时补偿和采样点定位，由 DPLL 的可编程位时序逻辑定义。

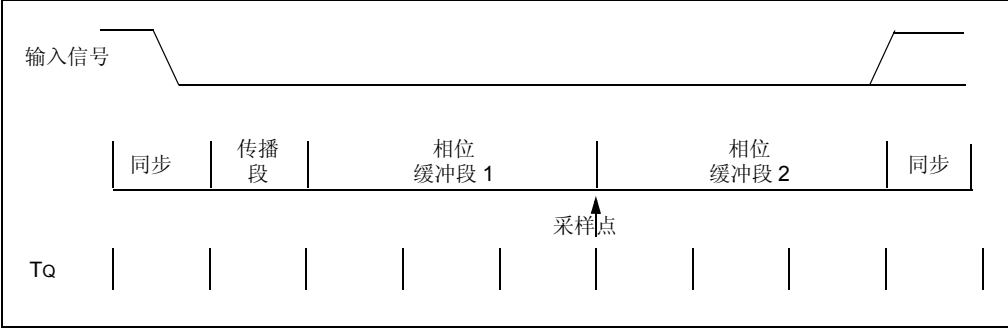
CAN 总线上的所有控制器必须有相同的波特率和位长度。然而，不同的控制器并不要求使用相同的主振荡器时钟。如果各个控制器的时钟频率不同，必须通过调节各个段的时间份额数调节波特率。

可考虑把标称位时间分成几个不重叠的时间段。这些段如图 23-20 中所示。

- 同步段（Sync Seg）
- 传播时间段（Prop Seg）
- 相位缓冲段 1（Phase1 Seg）
- 相位缓冲段 2（Phase2 Seg）

时间段以及标称位时间由整数个时间单元组成，这些单元称作时间份额或 Tq。根据定义，标称位时间最少由 8 个 Tq 组成，最多由 25 个 Tq 组成。同样根据定义，最小标称位时间是 1 μ s，对应最大 1 MHz 的比特率。

图 23-20: CAN 位时序



23.9.2 预分频器设置

有一个可编程预分频器，除了固定的二分频用于时钟发生以外，其整数预分频比范围为 1 至 64。时间份额（TQ）是由输入时钟频率 F_{CAN} 得到的固定时间单元。公式 23-1 中所示为时间份额的定义。

注： F_{CY} 不能超出 30 MHz。如果 CANCKS = 0，那么 F_{CY} 不能超出 7.5 MHz。

公式 23-1： 时钟发生的时间份额

$$TQ = \frac{2(BRP<5:0> + 1)}{F_{CAN}}$$

此处 BRP 是 BRP <5:0> 的二进制值
取决于 CANCKS 位，F_{CAN} 为 F_{CY} 或 4F_{CY}

例 23-1： 比特率计算示例

如果 4F_{CY} = 32 MHz，BRP<5:0> = 0x01 且 CANCKS = 0，那么：

$$TQ = 2 \cdot (BRP + 1) \cdot \frac{TCY}{4} = 2 \times 2 \times (1/32 \times 10^6) = 125 \text{ ns}$$

如果标称位时间 = 8 TQ，那么：
标称比特率 = 1/(8 × 125 × 10⁻⁹) Mbps

例 23-2： 波特率预分频比计算示例

CAN 波特率 = 125 kHz
F_{CY} = 5 MHz，CANCKS = 1

1. 选择每个位时间的 TQ 时钟数（例如，K = 16）。
2. 用波特率计算 TQ:

$$TQ = \frac{1/(BaudRate)}{K} = \frac{1/125 \times 10^3}{16} = 500ns$$

3. 计算 BRP<5:0>:

$$\begin{aligned} TQ &= 2 \cdot (BRP + 1) \cdot TCAN \\ BRP &= (2 TQ / TCY) - 1 \\ &= \frac{2(500 \times 10^{-9})}{1/(5 \times 10^6)} - 1 \\ &= 4 \end{aligned}$$

不同节点内振荡器的频率必须相同以提供一个全系统的特定时间份额。这意味着所有振荡器的 T_{osc} 必须是 TQ 的整除因子。

23.9.3 传播段

这部分的位时间用来补偿网络内的物理延时。这些延时包括总线线路上的信号传播时间以及节点的内部延时。延时计算的是从发送器到接收器的一个往返过程，是信号在总线线路上的传播时间、输入比较器延时与输出驱动器延时之和的两倍。通过设置PRSEG<2:0>位（CiCFG2<2:0>），传播段可以设置为 1 T_Q 到 8 T_Q。

23.9.4 相位段

相位段用于在发送位时间内优化定位接收位的采样。采样点在相位缓冲段 1 与相位缓冲段 2 之间。这两个段可以通过重新同步加长或缩短。相位缓冲段 1 的末尾决定在一个位周期内的采样点。该段可以编程为 1 T_Q 到 8 T_Q。相位缓冲段 2 为下一个发送的数据转变提供一个延时。该段可以编程为 1 T_Q 到 8 T_Q，也可定义为相位缓冲段 1 的时间份额与信息处理时间（3T_Q）两者中的较大者。通过设置 SEG1PH<2:0>（CiCFG2<5:3>）位初始化相位缓冲段 1，通过设置 SEG2PH<2:0>（CiCFG2<10:8>）位初始化相位缓冲段 2。

23.9.5 采样点

采样点是读总线电平并将其解释为对应位的值的一个时间点。它的位置在相位缓冲段 1 的末尾。若位时序较慢而且包含很多 T_Q，可以在同一个采样点指定总线线路的多个采样。由 CAN 总线决定的电平将对应于三值多数决定的结果。多数采样在采样点进行，且前两次采样相隔 T_Q/2。CAN 模块允许选择在同一点采样 3 次或 1 次。这可以通过置位或清零 SAM 位（CiCFG2<6>）做到。

23.9.6 同步

为了补偿不同总线站振荡器频率的相移，每个 CAN 控制器必须能与输入信号的相关信号沿同步。当检测到发送数据中的一个沿时，逻辑会将该沿的位置与预期时间（同步段）比较。然后电路将调节相位缓冲段 1 和相位缓冲段 2 的值。有两种机制用来同步。

23.9.6.1 硬同步

硬同步仅当总线空闲期间，有一个从“隐性”转变到“显性”的边沿时，才被执行，它指示报文传输的开始。硬同步后，位时间计数器从同步段重新开始计数。硬同步强制引起硬同步的沿处于重新开始的位时间同步段之内。根据同步规则，如果完成一个硬同步，在该位时间内将不会再有重新同步。

23.9.6.2 重新同步

重新同步可能使相位缓冲段 1 加长或相位缓冲段 2 缩短。相位缓冲段加长或缩短的量（由 SJW<1:0> 位（CiCFG1<7:6>）指定）有一个上限值，该值由重新同步跳转宽度位给出。同步跳转宽度位的值将被加入相位缓冲段 1 或从相位缓冲段 2 减去。重新同步跳转宽度可以设置为 Tq 到 4 Tq。

时钟信息只能由总线状态从隐性到显性的转变产生。同值连续位的个数只能有一个固定的最大值，该特性确保了在一个帧期间将一个总线单元与比特流重新同步（例如，位填充）。

一个边沿的相位误差由与同步段相关的边沿的位置给出，以时间份额量度。相位误差用 Tq 的幅值定义如下：

- $e = 0$ （如果边沿处于同步段里）。
- $e > 0$ （如果边沿位于采样点之前）。
- $e < 0$ （如果边沿位于前一个位的采样点之后）。

当相位误差的幅值小于或等于重新同步跳转宽度的设定值时，重新同步和硬同步的作用相同。

如果相位误差的幅值大于重新同步跳转宽度，且相位误差为正，则相位缓冲段 1 被加长，加长的量为与重新同步跳转宽度相等的值。

如果相位误差的幅值大于重新同步跳转宽度，且相位误差为负，则相位缓冲段 2 被缩短，缩短的量为与重新同步跳转宽度相等的值。

图 23-21： 加长一个位周期

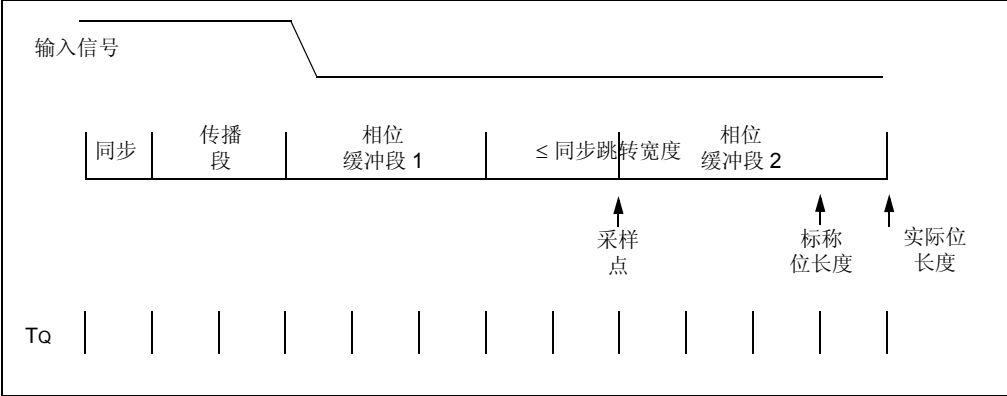
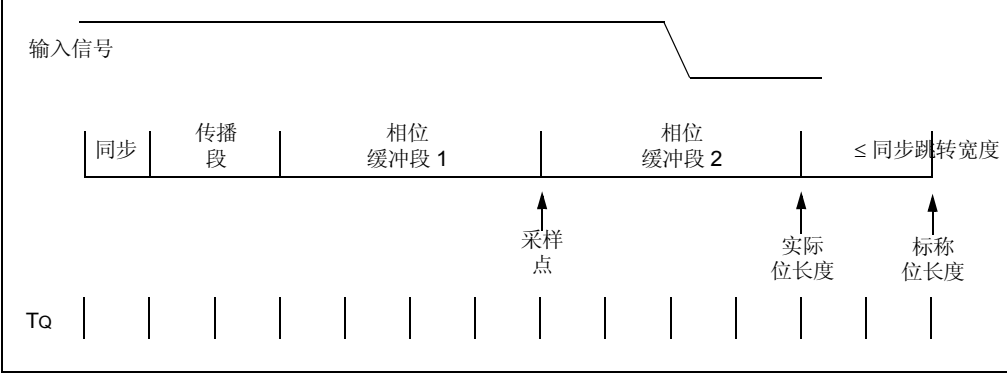


图 23-22： 缩短一个位周期



23.9.7 时间段编程

下面是时间段编程的一些要求：

- 传播段 + 相位缓冲段 1 > 相位缓冲段 2
- 相位缓冲段 2 > 同步跳转宽度

通常，位的采样应当发生在位时间的 60—70% 左右，取决于系统参数。

例 23-2 是一个有 16 T_Q 的位时间。如果我们选择同步段 = 1 T_Q 而且传播段 = 2 T_Q，那么设置相位缓冲段 1 = 7 T_Q 会把采样点安排在首次转变后的 10 T_Q（位时间的 62%）。这样会把 6 T_Q 留给相位缓冲段 2。

由于相位缓冲段 2 的长度为 6，根据规则，SJWS<1:0> 位能够设置的最大值为 4 T_Q。然而，通常只有当不同节点的时钟产生不精确或不稳定时（比如使用陶瓷谐振器），才需要一个大的同步跳转宽度。所以，通常同步跳转宽度为 1 就足够了。

23.10 中断

该模块有几个中断源。这些中断中的每一个都可以单独地使能或禁止。CiINTF 寄存器包含中断标志位。CiINTE 寄存器控制 8 个主要中断的使能。CiCTRL 寄存器中一组特别的只读位（ICODE<2:0>）可以与跳转表一起使用，高效地处理中断。

所有的中断都有一个源，错误中断例外。任何一个错误中断源都可以置位错误中断标志位。可以通过读 CiINTF 寄存器确定错误中断的原因。

中断可分为两类：接收和发送中断。

与接收相关的中断有：

- 接收中断
- 唤醒中断
- 接收器溢出中断
- 接收器警告中断
- 接收器错误被动中断

与发送相关的中断有：

- 发送中断
- 发送器警告中断
- 发送器错误被动中断
- 总线关断中断

23.10.1 中断确认

中断与 CiINTF 寄存器中的一个或多个状态标志位直接相关。只要一个对应的标志位置 1，中断就将处于等待状态。在中断处理程序中必须复位寄存器中的标志位以允许下次中断。若标志位置位的条件仍存在就不能清零，除非是因为某个错误计数寄存器达到了特定值而引起中断。

23.10.2 ICODE 位

ICODE<2:0> 位 (CiCTRL<3:1>) 是一组只读位, 旨在通过跳转表高效地处理中断。ICODE<2:0> 位一次只能显示一个中断, 因为中断位在该寄存器中是复用的。所以, ICODE<2:0> 位反映出具有最高优先级的等待中断或被允许的中断。一旦具有最高优先级的中断标志位被清零, ICODE<2:0> 位将反映出下一个具有最高优先级的中断码。相应中断的中断码仅当中断标志位和中断使能位都被置位时才能够被显示。表 23-6 描述了 ICODE<2:0> 位的操作。

表 23-6: ICODE 位译码表

ICODE<2:0>	布尔表达式
000	$\overline{ERR} \cdot \overline{WAK} \cdot \overline{TX0} \cdot \overline{TX1} \cdot \overline{TX2} \cdot \overline{RX0} \cdot \overline{RX1}$
001	ERR
100	$\overline{ERR} \cdot TX0$
011	$\overline{ERR} \cdot \overline{TX0} \cdot TX1$
010	$\overline{ERR} \cdot \overline{TX0} \cdot \overline{TX1} \cdot TX2$
110	$\overline{ERR} \cdot \overline{TX0} \cdot \overline{TX1} \cdot \overline{TX2} \cdot RX0$
101	$\overline{ERR} \cdot \overline{TX0} \cdot \overline{TX1} \cdot \overline{TX2} \cdot \overline{RX0} \cdot RX1$
111	$\overline{ERR} \cdot \overline{TX0} \cdot \overline{TX1} \cdot \overline{TX2} \cdot \overline{RX0} \cdot \overline{RX1} \cdot WAK$

图注: ERR = ERRIF • ERRIE
TX0 = TX0IF • TX0IE
TX1 = TX1IF • TX1IE
TX2 = TX2IF • TX2IE
RX0 = RX0IF • RX0IE
RX1 = RX1IF • RX1IE
WAK = WAKIF • WAKIE

23.11 时间标记

无论何时接收到一个有效帧, CAN 都将产生一个可以发送给定时器捕捉输入的信号。因为 CAN 规范定义如果在 EOF 字段成功发送之前未发生错误则该帧有效, 所以定时器信号将在 EOF 后立即产生。产生一个位时间的脉冲。

通过 TSTAMP 控制位 (CiCTRL<15>) 使能时间标记。时间标记所使用的捕捉输入取决于不同的器件。更多信息, 请参见特定的器件数据手册。

23.12 CAN 模块 I/O

该 CAN 总线模块通过至多两个 I/O 引脚通信。有一个发送引脚和一个接收引脚。这些引脚与器件的正常数字 I/O 功能复用。

当模块处于配置模式、模块禁止模式或环回模式, I/O 引脚将恢复为端口 I/O 功能。

当该模块处于活动状态时, CiTX 引脚 (i = 1 或 2) 始终专用为 CAN 输出功能。与发送引脚相关的 TRIS 位被 CAN 总线模式忽略。该模块在 CiRX 输入引脚上接收 CAN 输入。

23.13 CPU 低功耗模式下的工作

23.13.1 休眠模式下的工作

器件通过执行 `PWRSV #0` 指令进入休眠模式。这样做将使晶振停振并关闭所有系统时钟。当 CPU 进入休眠模式时，用户应该确保模块不处于活动状态。引脚将恢复为普通 I/O 功能，取决于 TRIS 寄存器内的值。

因为 CAN 总线不能被中断，所以当该模块处于工作模式时，用户决不能执行 `PWRSV #0` 指令。必须通过设置 `REQOP<2:0> = 001` (`CiCTRL<10:8>`)，先将该模块切换到禁止模式。当 `OPMODE<2:0> = 001` (`CiCTRL<7:5>`)，表示模块已进入了禁止模式时，然后才可以使用休眠指令。

图 23-23 描述了当 CPU 进入休眠模式时 CAN 模块的操作，以及该模块如何在发生总线活动时唤醒。当 CAN 总线上的活动导致 CPU 退出休眠模式时，`WAKIF` 标志位 (`CiINTF<6>`) 置 1。

当器件处于休眠模式时，该模块将监视 `CiRX` 线路的活动。

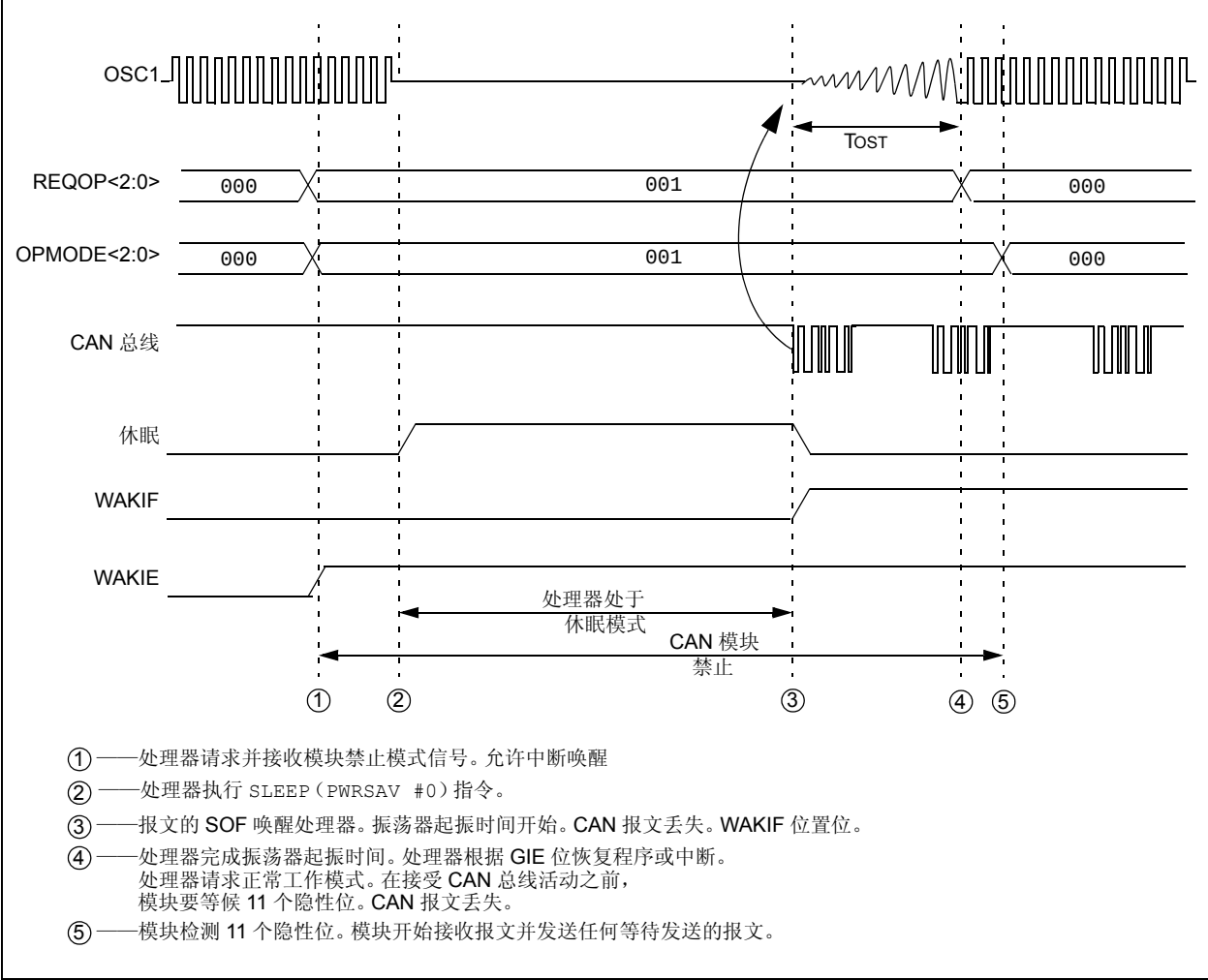
如果器件处于休眠模式且 `WAKIE` 唤醒中断使能位被置位，该模块就会产生中断，唤醒 CPU。由于振荡器和 CPU 启动时的延时，引起唤醒的报文活动将会丢失。

如果模块处于 CPU 休眠模式且 `WAKIE` 没有被置位，就不会产生中断且 CPU 和该 CAN 模块会继续休眠。

如果 CAN 模块处于禁止模式，模块将被唤醒而且根据 `WAKIE` 位的不同状态，还可能产生中断，并且模块可以正确接收引起从休眠模式唤醒的报文。

当模块和 CPU 处于休眠模式时，通过对模块进行编程可以在 `CiRX` 输入线路中加入低通滤波器功能。该功能可用于保护模块不会由于 CAN 总线上的短时尖脉冲干扰而导致误唤醒。这种尖脉冲干扰是由噪声环境下电磁干扰引起的。`WAKFIL` 位 (`CiCFG2<14>`) 使能或禁止该滤波器。

图 23-23: 处理器休眠和 CAN 总线唤醒中断



23.13.2 CPU 空闲模式下的 CAN 模块工作

当执行 CPU 空闲 (PWRSAV #1) 指令后，CAN 模块的工作由 CSIDL 位 (CiCTRL<13>) 的状态决定。

如果 CSIDL = 0，该模块将继续在空闲模式下工作。如果允许 CAN 模块中断，CAN 模块可以将器件从空闲模式唤醒。

如果 CSIDL = 1，该模块在空闲模式下将停止工作。规则和条件同进入休眠模式和从休眠模式唤醒。更多详细信息，请参见第 23.13.1 节“休眠模式下的工作”。

23.14 CAN 协议综述

控制器局域网（CAN）是串行通信协议，它能有效支持高性能的分布式实时控制。Robert Bosch GmbH 在 1991 年的 CAN 规范 V2.0B 中完整定义了 CAN 协议。

CAN 应用涵盖从高速网络到低成本多路复用线路的各种领域。汽车电子设备（即，引擎控制单元、传感器和防滑系统等）均是使用 CAN 以最大 1 Mb/s 的比特率连接的。CAN 网络可以用来取代汽车中的线路连接以有效节约成本。CAN 总线在噪声环境中的可靠性及其故障状态检测和从故障状态恢复的能力使其适用于 DeviceNet、SDS 和其他现场总线协议等工控应用。

CAN 是具有一条逻辑总线线路的异步串行总线系统。它具有开放式线性总线型网络结构，总线节点平均分布。CAN 总线由两个或多个节点组成。总线上的节点数量可以动态改变而不会干扰其他节点的通信。这使得总线节点可方便地连接和断开（如为了增加系统功能、错误恢复或总线监测）。

总线逻辑采用“线与”机制，“隐性”（Recessive）位（通常为逻辑“1”但非必须如此）被“显性”（Dominant）位覆盖（通常逻辑电平为“0”）。只要没有总线节点在发送显性位，总线线路就处于隐性状态，但是只要任一总线节点发送了显性位，就会产生显性总线状态。因此，对于 CAN 总线线路，必须选择能传送两种可能位状态（显性和隐性）的传输介质。其中最常见也最便宜的方法是使用双绞线。此时总线线路被称为“CANH”和“CANL”，并且可以直接或通过连接器与节点相连。CAN 协议中没有定义有关连接器使用的标准。双绞线的端接是通过在总线线路两端连接端接电阻完成的。当总线长度不超过 40 米时，最大总线速度可达 1 Mb。如果总线长度超过 40 米，总线速度必须降低（将总线速度降到 40 Kb 可使总线长度达到 1000 米）。对于 1000 米以上的总线，应该使用特殊的驱动器。无需额外设备，即至少可以连接 20 个节点。由于发送的差分性，即两条总线线路受 EMI 的影响相同，使差分信号不受影响，所以 CAN 本质上并不容易受到辐射电磁能的影响。也可以屏蔽总线线路以降低来自总线的电磁辐射，尤其是在高波特率下。

二进制数据是按 NRZ 码（非归零码，低电平 = 显性状态，高电平 = 隐性状态）编码的。使用位填充以确保所有总线节点时钟同步。这意味着在报文发送时，最多可以有 5 个连续位具有相同的极性。只要发送了相同极性的 5 个连续位，在继续发送后续位前，发送器将在比特流中插入一个极性相反的位。接收器也会检查同极性位的个数，并把填充位从比特流中删除（解填充）。

在 CAN 协议中，不对总线节点寻址。地址信息包含在发送的报文中。这是通过标识符（每个报文的组成部分）实现的，标识符标识了报文内容（如引擎速度和油温等）。另外，标识符还会指出报文的优先级。标识符的二进制值越小，报文的优先级就越高。

对于总线仲裁，采用的是带有 NDA 的 CSMA/CD（带有非破坏性仲裁的载波监听多路访问 / 冲突检测）。如果总线节点 A 要通过网络发送报文，它首先会检查总线是否处于空闲状态（“载波监听”）（即没有正在进行发送的节点）。如果这样（并且没有其他节点希望在这一时刻开始发送），节点 A 就成为总线主控节点并发送其报文。在节点 A 发送第一个发送位（帧起始位）时，所有其他节点都会切换到接收模式。在正确接收了报文后（报文经过每个节点应答），每个总线节点都会检查报文标识符，如果需要的话还将存储报文。否则，就将报文丢弃。

如果两个或多个总线节点同时开始发送（“多路访问”），此时可通过按位仲裁（“冲突检测 / 非破坏性仲裁”加上“线与”机制和“显性”位覆盖“隐性”位）避免报文冲突。每个节点都发送报文的标识符位（先发送 MSb）并监视总线电平。如果节点发送的是隐性标识符位但是读回的是显性标识符位，则它将失去总线仲裁并切换到接收模式。当与之发生竞争的节点的报文标识符有较小的二进制值（显性状态 = 逻辑 0）时，也即竞争的节点发送的报文具有较高的优先级时，就会产生这种情况。这样，具有最高优先级报文的总线节点不需要花费时间重发这一报文就可以赢得仲裁。一旦总线返回到空闲状态，所有其他节点就会自动尝试重新发送。不允许不同的节点采用同一标识符来发送报文，因为这样的话仲裁可能会失败导致发生冲突和差错。

早期的 CAN 规范（1.0、1.2 和 2.0A 版）定义报文标识符为 11 位长，从而给出了 2048 种可能的不同的报文标识符。后来此规范进行了更新（更新到 2.0B 版）就消除了这一限制。CAN 规范 2.0B 版允许的可用报文标识符的长度为 11 和 / 或 29 位（29 位长的标识符给出了超过 5 亿 3 千 6 百万种不同的报文标识符）。CAN 2.0B 版也被称为“扩展 CAN”；而 1.0、1.2 和 2.0A 版则被称为“标准 CAN”。

23.14.1 标准 CAN 与扩展的 CAN 的对比

根据 CAN 规范 V2.0A，只包含 11 位标识符的数据帧和远程帧被称为标准帧。这些帧可以用来标识 2048 条不同报文（标识符为 0 — 2047）。但是，优先级最低的 16 条报文（2032 — 2047）是保留的。根据 CAN 规范 V2.0B，扩展帧具有 29 位的标识符。如前所述，29 位标识符是由 11 位标识符（“标准 ID”）和 18 位扩展标识符（“扩展 ID”）组成的。

CAN V2.0A 规定的 CAN 模块只能根据标准 CAN 协议发送和接收标准帧。发送和接收使用 29 位标识符的报文会产生差错。如果是 CAN V2.0B 规范的器件，则存在两种情况。命名为“Part B Passive”的模块只能发送和接收标准帧，但是在遇到扩展帧时不会产生错误帧。称为“Part B Active”的器件能够发送和接收标准帧和扩展帧。

23.14.2 ISO 模型

ISO/OSI 参考模型用于定义通信系统协议的层，如图 23-24 所示。在最高层，应用程序需要互相通信。在最低层，使用某些物理介质提供电信令。

协议的高层由软件控制。在 CAN 总线规范中，并没有对报文类型或被传输报文的内容及意义进行定义。在有些体系中定义了这些内容，比如 Volcano（Volvo 汽车 CAN 规范）、J1939（美国重型卡车多路布线规范）以及 Allen-Bradly DeviceNet 和 Honeywell SDS（工业协议的两个范例）。

CAN 总线模块定义了整个协议中以下相邻的两层：

- 数据链路层
 - 逻辑链路控制（LLC）子层
 - 介质访问控制（Medium Access Control，MAC）子层
- 物理层
 - 物理信令（PLS）子层

LLC 子层与报文过滤、过载通知和错误恢复管理有关。LLC 子层的功能范围包括：

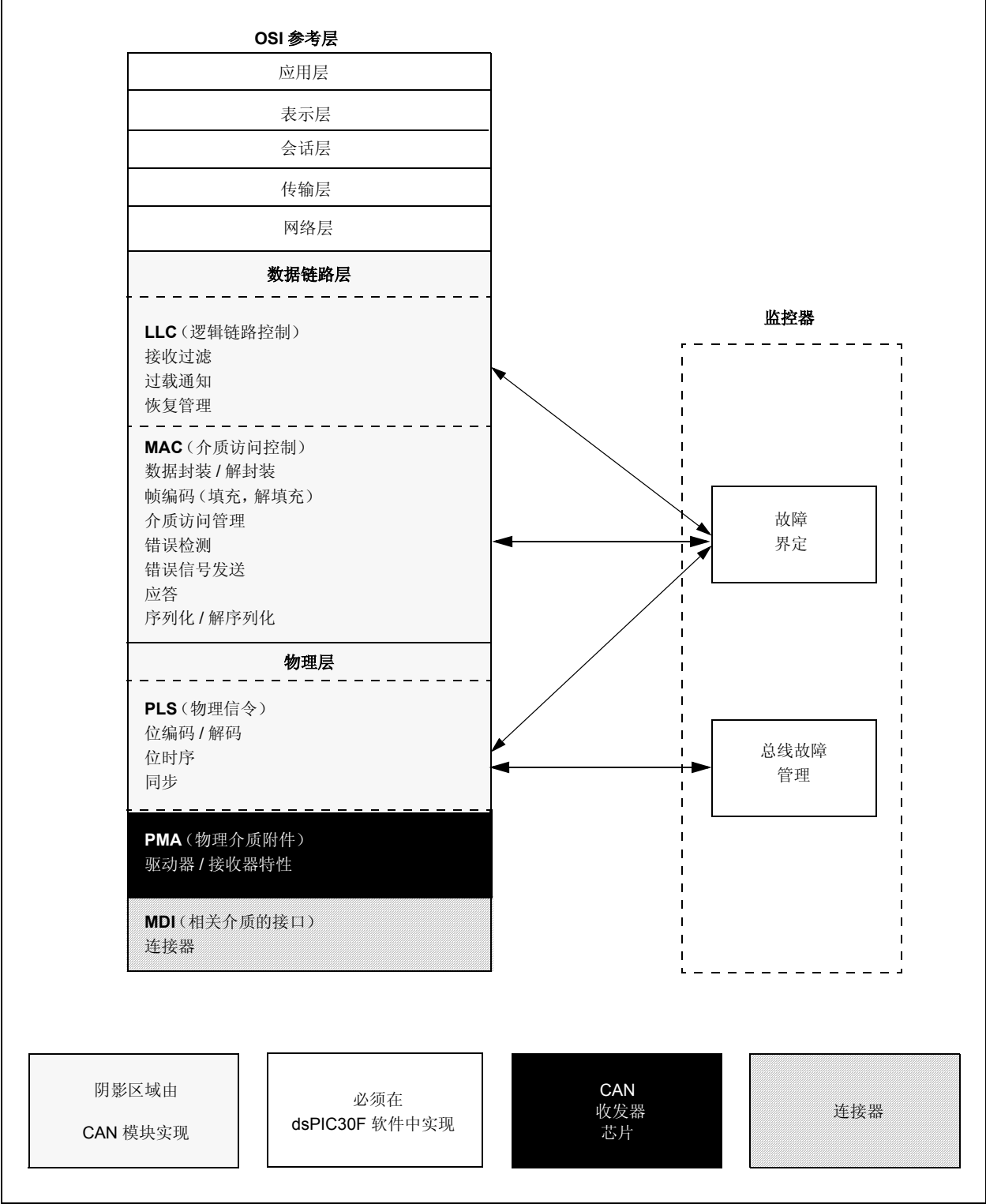
- 为数据传输和远程数据请求提供服务
- 决定 LLC 子层接收的哪些信息将确实被接受。
- 提供错误恢复管理和过载通知的方式

MAC 子层是 CAN 协议的核心。MAC 子层定义了传输协议（即控制帧、执行仲裁、错误检测、错误信令产生以及故障界定）。该层提供从 LLC 子层接收的报文，并接收将要发送到 LLC 子层的报文。判断总线是处于空闲状态以开始新的发送还是一次接收刚刚开始都是在 MAC 子层内进行的。MAC 子层由一个叫做故障界定的管理实体监管，这是一种能够把短时干扰和永久故障区分开来的自检机制。而且，位时序的某些通用功能也被认为是 MAC 子层的一部分。

物理层可以依照所有的电气属性定义不同节点之间的实际位传输。PLS 子层定义信号的实际发送方式，并由此处理位时序、位编码和同步。

协议的较低层在驱动器 / 接收器芯片和实际接口（如双绞线或光纤等）中实现。在一个网络中，所有节点必须使用同一个物理层。物理层的驱动器 / 接收器特性并未在 CAN 规范中定义，这样可使发送介质和信令实现根据应用进行优化。ISO11898 道路车辆多路布线规范就是物理传输介质的最常见的例子。

图 23-24: ISO/OSI 参考模型中的 CAN 总线



23.15 相关应用笔记

本节列举了与手册的本章节相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相近的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与 CAN 模块相关的应用笔记有：

标题

An Introduction to the CAN Protocol

应用笔记编号

AN713

注： 如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

23.16 版本历史

版本 A

这是本文档的初始发行版。

版本 B

此版本添加了有关 dsPIC30F CAN 模块的技术内容。

版本 C

此版本加入了本文档更新时已知的所有勘误表。

注:

第 24 章 器件配置

目录

本章包括下列主题：

24.1 简介	24-2
24.2 器件配置寄存器	24-2
24.3 配置位说明	24-6
24.4 器件标识寄存器	24-7
24.5 相关应用笔记	24-8
24.6 版本历史	24-9

24.1 简介

器件配置寄存器允许每个用户定制器件的某些方面以适应应用的需要。器件配置寄存器是程序存储器映射空间中的非易失性存储单元，在掉电期间它保存 dsPIC 器件的设置。这些配置寄存器保存器件的全局设置信息，诸如振荡器来源、看门狗定时器模式和代码保护设置。

器件配置寄存器映射在程序存储器以地址 0xF80000 开始的单元中，在器件正常工作期间可以访问这些单元。此区域也称为“配置空间”。

可以通过对配置位编程（读为 0）或不编程（读为 1）来选择不同的器件配置。

24.2 器件配置寄存器

虽然每个器件配置寄存器都是 24 位寄存器，但只有它们的低 16 位可用来保存配置数据。有四个器件配置寄存器可供用户使用：

- FOSC (0xF80000)：振荡器配置寄存器（注 2）
- FWDT (0xF80002)：看门狗定时器配置寄存器
- FBORPOR (0xF80004)：BOR 和 POR 配置寄存器
- FGS (0xF8000A)：通用代码段配置寄存器

可以使用运行时自编程（RTSP）、在线串行编程（In-Circuit Serial Programming™，ICSP™）或器件编程器对器件配置寄存器进行编程。

注 1：	后述配置寄存器的器件配置位在某个特定器件上并非都可用。更多信息请参见器件数据手册。
2：	dsPIC30F 器件的通用系列、传感器系列和电机控制系列具有以下三种版本的振荡器系统之一——版本 1、版本 2 和版本 3。欲了解每种版本上的 Fosc 器件配置寄存器配置位信息，请参见第 7 章“振荡器”。

低字节:								
U	U	R/P	R/P	R/P	R/P	R/P	R/P	
		FWPSA<1:0>		FWPSB<3:0>				
bit 7								bit 0

0000 = 1:1

U = 未用位

dsPIC30F 系列参考手册

寄存器 24-2: FBORPOR: BOR 和 POR 配置寄存器

高字节:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23				bit 16			

中间字节:							
R/P	U	U	U	U	R/P	R/P	R/P
MCLREN	—	—	—	—	PWMPIN	HPOL	LPOL
bit 15				bit 8			

低字节:							
R/P	U	R/P	R/P	U	U	R/P	R/P
BOREN	—	BORV<1:0>		—	—	FPWRT<1:0>	
bit 7				bit 0			

bit 23-16 未用: 读为 0

bit 15 **MCLREN**: $\overline{\text{MCLR}}$ 引脚功能使能位

1 = 引脚功能为 $\overline{\text{MCLR}}$ (默认情形)

0 = 引脚禁止

bit 14-11 未用: 读为 0

bit 10 **PWMPIN**: 电机控制 PWM 模块引脚模式位

1 = 器件复位时, PWM 模块引脚由 PORT 寄存器控制 (三态)

0 = 器件复位时, PWM 模块引脚由 PWM 模块控制 (配置为输出引脚)

bit 9 **HPOL**: 电机控制 PWM 模块高端极性位

1 = PWM 模块高端输出引脚的输出极性处于高电平有效状态

0 = PWM 模块高端输出引脚的输出极性处于低电平有效状态

bit 8 **LPOL**: 电机控制 PWM 模块低端极性位

1 = PWM 模块低端输出引脚的输出极性处于高电平有效状态

0 = PWM 模块低端输出引脚的输出极性处于低电平有效状态

bit 7 **BOREN**: PBOR 使能位

1 = PBOR 使能

0 = PBOR 禁止

bit 6 未用: 读为 0

bit 5-4 **BORV<1:0>**: 欠电压选择位

11 = 2.0V

10 = 2.7V

01 = 4.2V

00 = 4.5V

bit 3-2 未用: 读为 0

bit 1-0 **FPWRT<1:0>**: 上电复位延时定时器延迟时间选择位

11 = PWRT 为 64 ms

10 = PWRT 为 16 ms

01 = PWRT 为 4 ms

00 = 上电延时定时器禁止

注: PWMPIN、HPOL 和 LPOL 配置位只在具备电机控制 PWM 模块的器件上才有。

图注:

R = 可读位

P = 编程位

U = 未用位

寄存器 24-3: FGS: 通用代码段配置寄存器

高字节:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23				bit 16			

中间字节:							
U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 15				bit 8			

低字节:							
U	U	U	U	U	U	P	P
—	—	—	—	—	—	GCP	GWRP
bit 7				bit 0			

- bit 23-2 未用: 读为 0
- bit 1 **GCP:** 通用代码段代码保护位
1= 用户程序存储器无代码保护
0= 用户程序存储器有代码保护
- bit 0 **GWRP:** 通用代码段写保护位
1= 用户程序存储器无写保护
0= 用户程序存储器有写保护
- 注: BCP 和 GWRP 配置位只能编程为 “0”。

图注:		
R = 可读位	P = 编程位	U = 未用位

24.3 配置位说明

本节提供各个器件配置位的特定的功能信息。

24.3.1 振荡器配置位

dsPIC30F 器件的通用系列、传感器系列和电机控制系列具有以下三种版本的振荡器系统之一——版本 1、版本 2 和版本 3。欲了解每种版本上的 Fosc 器件配置寄存器配置位信息，请参见第 7 章“振荡器”。

24.3.2 BOR 和 POR 配置位

FBORPOR 配置寄存器中的 BOR 和 POR 配置位可用来为器件设置欠压复位电压，使能欠压复位电路并设置上电延时定时器的延时时间。如需了解有关这些配置位的更多信息，请参见第 8 章“复位”。

24.3.3 电机控制 PWM 模块配置位

电机控制 PWM 模块配置位位于 FBORPOR 配置寄存器中，并且只有在具有 PWM 模块的器件中才存在这些配置位。与 PWM 模块相关的配置位有两个功能：

1. 在器件复位时选择 PWM 引脚的状态（高阻态或输出）。
2. 为 PWM 引脚选择有效的信号极性。可以独立选择高端和低端 PWM 引脚的极性。

如需了解有关这些配置位的更多信息，请参见第 15 章“电机控制 PWM”。

24.3.4 通用代码段配置位

通用代码段配置位位于 FGS 配置寄存器中，用于对用户程序存储器空间进行代码保护或写保护。通用代码段包括除中断向量表空间（0x000000 到 0x0000FE）以外的所有用户程序存储器。

如果通过把 GCP 配置位（FGS<1>）编程为“0”对通用代码段进行代码保护，则不能使用在线串行编程（ICSP）或器件编程器从器件读器件程序存储器。另外，在没有首先擦除整个通用代码段之前，不能进一步对器件编程。

当通用代码段有代码保护时，用户代码仍然能通过表读指令或从数据空间使用程序空间可视性（PSV）访问程序存储器中的数据。

如果 GWRP（FGS<0>）配置位被编程，将禁止对用户存储器空间的所有写操作。

24.3.4.1 通用代码段配置位组

FGS 配置寄存器中的 GCP 和 GWRP 配置位必须作为一个组来编程 / 擦除。如果这两个配置位中的一个或全部被编程为“0”，必须执行完全的芯片擦除以改变两个位中任何一位的状态。

注：	如果代码保护配置熔丝组（FGS<GCP:GWRP>）已被编程，那么仅当电压 $V_{DD} \geq 4.5V$ 时才可以擦除整个有代码保护的器件。

24.4 器件标识寄存器

dsPIC30F 器件有两组提供标识信息的寄存器位于配置空间内。

24.4.1 器件 ID (DEVID) 寄存器

配置存储器空间单元 0xFF0000 和 0xFF0002 用于存储一个只读的器件 ID 编号，该编号在生产器件时设置。此编号指明了 dsPIC30F 器件的型号和硅芯片版本。

用户可使用读表指令读器件 ID 寄存器。

24.4.2 器件 ID 字段

器件 ID 字段位于从地址 0x800600 到地址 0x80063E 的配置存储器空间中。该字段由 32 个程序存储器单元组成，并且可在 Microchip 工厂使用惟一器件信息对其编程。用户不可写或擦除该字段，但可使用表读指令读该字段。

如欲了解更详细信息，请联系 Microchip 技术支持或当地 Microchip 代表。

24.5 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与器件配置模块相关的应用笔记有：

标题	应用笔记编号
dsPIC30F 在无传感器 BLDC 控制中的应用	AN901

注：如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

24.6 版本历史

版本 A

这是本文档的初始发行版。

版本 B

此版本融合了 dsPIC30F 器件配置模块的技术内容修改。

版本 C

此版本包含了本文档更新时已知的所有勘误表。

版本 D

增加了振荡器控制模块的三种版本的说明。将 Fosc 配置寄存器的定义移到 DS70054_CN。

注:

第 25 章 开发工具支持

目录

本章包括下列主题：

25.1	简介	25-2
25.2	Microchip 硬件和语言工具	25-2
25.3	第三方硬件 / 软件工具和应用程序库	25-6
25.4	dsPIC30F 硬件开发板	25-11
25.5	相关应用笔记	25-15
25.6	版本历史	25-16

注： 尽管本章描述的部分开发工具在撰写本文时尚未提供，但它们目前已在开发中。某些产品的细节可能会有所不同。有关各产品的最新信息和供货状况请浏览 **Microchip** 网站或咨询当地的 **Microchip** 销售办事处。

25.1 简介

Microchip 将提供支持 dsPIC® 架构的全面的开发工具包和库。此外，本公司还与许多提供其他 dsPIC 器件支持的第三方供应商合作。

25.2 Microchip 硬件和语言工具

建议使用的 Microchip 工具包括：

- MPLAB® 集成开发环境（IDE）
- dsPIC 语言套件，包括 MPLAB C30 C 编译器、汇编器、链接器和库
- MPLAB SIM 软件模拟器
- MPLAB ICE 4000 在线仿真器
- MPLAB ICD 2 在线调试器
- PRO MATE® II 通用器件编程器
- PICSTART® Plus 开发编程器

25.2.1 MPLAB 6.XX 集成开发环境软件

注： 本产品目前可从 Microchip 网站 www.microchip.com 下载。

MPLAB 集成开发环境（IDE）可免费使用。MPLAB IDE 软件是一个带有用于开发和调试单片机设计应用程序的工具包的桌面开发环境。MPLAB IDE 可以在不同的开发和调试操作之间进行快速切换。本产品设计与 Windows® 操作环境共同使用，它是一款强大的、并不需要昂贵代价的运行开发工具。它也是各种 Microchip 开发系统工具共用的用户界面，包括 MPLAB 编辑器、MPLAB ASM30 汇编器、MPLAB SIM 软件模拟器、MPLAB LIB30 库、MPLAB LINK30 链接器、MPLAB ICE 4000 在线仿真器、PRO MATE II 编程器和在线调试器（ICD 2）。MPLAB IDE 为用户提供了从同一个用户界面编辑、编译和仿真的灵活性。工程师们可以在与设计 PIC® 单片机所使用的相同环境中为 dsPIC 器件设计和开发代码。

MPLAB IDE 是一款基于 32 位 Windows 的应用软件。它通过一个时尚而便利的界面为工程师们提供了许多高级的功能。MPLAB IDE 集成了：

- 功能齐全用颜色区分代码功能的文本编辑器
- 带可视化显示的易于使用的项目管理器
- 源代码调试功能
- 增强型源代码调试功能，可用于调试 C 语言编写的程序
 - （结构和自动变量等）
- 可定制的工具栏和按键映射
- 动态状态栏，“一瞥”即可查看新显示的处理器状况
- 上下文关联的互动在线帮助
- 集成的 MPLAB SIM 指令模拟器
- PRO MATE II 和 PICSTART Plus 器件编程器的用户界面（单独销售）
- MPLAB ICE 4000 在线仿真器的用户界面（单独销售）
- MPLAB ICD 2 在线调试器的用户界面（单独销售）

MPLAB IDE 可以让工程师：

- 用汇编语言或 C 语言编辑源文件
- 点击鼠标一次即可将代码下载到仿真器或模拟器中的 dsPIC 程序存储器中。所有项目信息均被更新。
- 可使用如下各项进行调试：
 - 源文件
 - 机器码
 - 混合模式的源代码和机器代码

由于 MPLAB IDE 能与多个开发和调试目标配合使用，使用户只需最少的再培训即可从低成本的模拟器转而使用全功能的仿真器。

25.2.2 dsPIC 语言套件

注： 本产品目前可从 Microchip 网站 www.microchip.com 下载。MPLAB IDE 中还包括汇编器、链接器和库管理器。如需了解 MPLAB C30 C 编译器的供货情况，请联络当地的 Microchip 销售办事处。

Microchip Technology 的 MPLAB C30 C 编译器是完整而易于使用的语言产品。它让您可以用高级 C 语言编写 dsPIC 应用程序，然后将它们完全转换成机器目标代码为单片机编程。它通过消除代码三角的壁垒简化代码开发并使设计师集中精力于程序流程而不是程序元素。我们提供了几种编译选项，这样用户就可以针对代码特征选择效率最高的编程方式。

MPLAB C30 编译器是完全符合 ANSI 标准的产品，内含 dsPIC 系列单片机的标准库。它使用 dsPIC 器件的许多高级特性来提供高效的汇编代码生成。

MPLAB C30 还提供了能使硬件得到极佳支持的扩展功能，例如，中断和外设。它完全集成在 MPLAB IDE 中，可进行高级源代码调试。其特性包括：

- 16 位原始数据类型
- 高效使用基于寄存器的 3 个操作数指令
- 复杂的寻址模式
- 高效的多位移位操作
- 高效的有符号 / 无符号比较

MPLAB C30 自带汇编器、链接器和库管理器。使用户能写混合模式（C 语言和汇编语言）程序，并将生成的目标文件链接到单个可执行文件中。编译器是单独销售的。汇编器、链接器和库管理器是免费的，它们随 MPLAB IDE 一起提供。

25.2.3 MPLAB SIM 软件模拟器

注： 本产品包含在 MPLAB IDE 中。

MPLAB SIM 软件模拟器可通过在指令级上对 dsPIC 器件进行模拟，从而在 PC 主机环境下进行代码开发。使用任意给定的指令，都可以对数据区域进行检查和修改。指令的执行方式包括单步、执行到断点或跟踪模式。⁽¹⁾

MPLAB SIM 模拟器完全支持使用 MPLAB C30 编译器和汇编器进行符号调试。软件模拟器为在实验室环境以外开发和调试代码提供了灵活性，使其成为一款极佳的多项目软件开发工具。

注 1： 在编写本手册时，某些功能（包括外设支持）尚未用。请浏览 Microchip 网站或向当地 Microchip 销售办事处查询最新信息。

25.2.4 MPLAB ICE 4000 在线仿真器

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能会有所改变。有关本产品的最新信息和供货情况请浏览 [Microchip](#) 网站或联系当地 [Microchip](#) 销售办事处。

MPLAB ICE 4000 在线仿真器为产品开发工程师提供了完整的 dsPIC 器件的硬件设计工具。MPLAB IDE 提供该仿真器的软件控制。

MPLAB ICE 4000 具有增强的跟踪、触发和监控功能，是一款全功能仿真器系统。可更换的处理器模块使系统可以很容易地重新配置，以仿真不同的处理器。

MPLAB ICE 4000 支持扩展的高端 PIC[®] 单片机，PIC18CXXX 和 PIC18FXXX 器件以及 dsPIC 系列的数字信号控制器。MPLAB ICE 4000 在线仿真器的模块化架构使其可扩展到支持新器件。

MPLAB ICE 4000 在线仿真器系统已被设计为一种实时仿真系统，并具有以往较昂贵的开发工具才有的高级功能。它具有如下特性：

- 全速仿真，高达 50 MHz 的总线速度或 200 MHz 的外部时钟速度
- 低压仿真，最低可至 1.8V
- 配置了 2 Mb 程序仿真存储器和高达 16 Mb 的额外模块化存储器
- 64K × 136 位宽的跟踪存储器
- 无限软件断点
- 复杂的暂停、跟踪和触发逻辑
- 多级触发（最多可达 4 级）
- 跟踪特定事件的过滤触发功能
- 用于顺序事件触发的 16 位通过计数器
- 16 位延时计数器
- 48 位时间戳
- 秒表功能
- 事件的时间间隔
- 统计结果分析
- 代码覆盖分析
- 可与 PC 连接的 USB 端口和并行打印机端口

25.2.5 MPLAB ICD 2 在线调试器

注： 该产品现已供货，但是目前不支持 dsPIC30F 器件。请浏览 Microchip 网站查阅产品升级信息。

Microchip 的在线调试器 MPLAB ICD 是一款功能强大的低成本运行时开发工具。此工具基于 PIC 和 dsPIC 闪存器件。

MPLAB ICD 2 将利用不同器件内建的在线调试功能。此功能配合 Microchip 的 In-Circuit Serial Programming™ 协议 (ICSP™)，可通过 MPLAB IDE 的图形用户界面，提供低成本的在线调试。这将使设计者通过观察变量、单步运行和设置断点来开发和调试源代码。其全速运行使测试硬件得以实时进行。它包含以下特性：

- 可在器件允许的范围内全速运行
- 串行或 USB PC 接口
- 外部供电的串行接口
- 来自 PC 接口的 USB 供电
- 低噪声电源 (VPP 和 VDD) 可与模拟和其他噪声敏感应用程序配合工作。
- 工作电压低至 2.0V
- 可用作 ICD 或廉价串行编程器
- 作为 MPLAB ICD 的模块化应用程序接口
- 有限的断点数量
- “Smart watch” 变量窗口
- 需要某些的芯片资源 (RAM、程序存储器和 2 个引脚)

25.2.6 PRO MATE II 通用器件编程器

注： 该产品现已供货，但目前不支持 dsPIC30F 器件。请浏览 Microchip 网站查阅产品升级信息。

PRO MATE II 通用器件编程器是一种功能齐全的编程器，能够在单机模式和 PC 主控模式下运行。

PRO MATE II 器件编程器有可编程的 VDD 和 VPP 电源，允许在编程需要这一功能时，在 VDDMIN 和 VDDMAX 上校验程序存储器以最大限度地提高可靠性。它具有能显示指令和错误消息 LCD 显示器。可更换的插座模块选择将支持所有的封装类型。

在单机模式下，PRO MATE II 器件编程器将能够读、校验或编程 PIC 和 dsPIC30F 器件。还能够在这种模式下设置代码保护。PRO MATE II 功能包括：

- 在 MPLAB IDE 环境下运行
- 可现场升级的固件
- 批量生产的 DOS 命令行界面
- 主机、安全和单击操作
- 自动下载目标文件
- SQTPSM 序列号生成功能为每个编程的器件添加惟一的序列号
- 在线串行编程包 (单独销售)
- 可更换的插座工具模块支持所有封装选项 (单独销售)

25.3 第三方硬件 / 软件工具和应用程序库

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能已经过更新。有关本产品的最新信息和供货情况请浏览 **Microchip** 网站或联系当地 **Microchip** 销售办事处。

Microchip 与主要的第三方工具生产商合作开发高质量的支持 **dsPIC30F** 产品系列的硬件和软件工具。**Microchip** 计划提供这套初始工具和库，使客户能够快速开发基于 **dsPIC30F** 的应用程序。

Microchip 将扩展现有的产品列表，为客户提供增值服务（即经验丰富的 / 通过认证的技术应用工程师、参考设计和软硬开发人员的人才库）。

如需获取有关 **dsPIC30F** 器件系列的第三方支持的最新信息，请浏览 **Microchip** 网站（www.microchip.com）。

dsPIC30F 软件工具和库包含：

- 第三方 C 编译器
- 浮点和双精度数学库
- DSP 算法库
- 数字滤波器设计软件实用程序
- 外设驱动程序库
- CAN 库
- 实时操作系统（Real-Time Operating System，RTOS）
- OSEK 操作系统
- TCP/IP 协议栈
- V.22/V.22bis 和 V.32 ITU 规范

dsPIC30F 硬件开发板工具包括：

- 通用开发板
- 电机控制开发系统
- 网络连接开发板

25.3.1 第三方 C 编译器

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能会有所改变。有关本产品的最新信息和供货状况请浏览 **Microchip** 网站或联系当地 **Microchip** 销售办事处。

除了 **Microchip MPLAB C30 C** 编译器，由 **IAR**、**HI-TECH** 和 **Custom Computer Services (CCS)** 开发的 **ANSI C** 编译器也支持 **dsPIC30F**。

此编译器让您可以用高级 C 语言编写 **dsPIC** 应用程序代码，然后将它们完全转换成机器目标代码以便为这些单片机编程。每个编译器工具都提供几种编译选项，这样用户可以针对新生成代码的特征的编程方法选择具有最高效生成代码的特性的选项。

多种 C 编译器解决方案具有不同价格定位和特性，客户可以选择最适合自己的应用程序需求的编译器。

25.3.2 数学库

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能会有所改变。有关本产品的最新信息和供货情况请浏览 **Microchip** 网站或联系当地 **Microchip** 销售办事处。

该数学库支持几种标准的 C 语言函数，包含但不限于：

- `sin()`, `cos()`, `tan()`
- `asin()`, `acos()`, `atan()`,
- `log()`, `log10()`
- `sqrt()`, `power()`
- `ceil()`, `floor()`
- `fmod()`, `frexp()`

这些数学函数程序将使用 **dsPIC30F** 汇编语言进行开发和优化，并可用汇编语言和 C 语言调用。我们还提供每个函数的浮点和双精度版本。它们支持 **Microchip MPLAB C30** 和 **IAR C** 编译器。

25.3.3 DSP 算法库

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能会有所改变。有关本产品的最新信息和供货情况请浏览 **Microchip** 网站或联系当地 **Microchip** 销售办事处。

DSP 库将支持多种滤波、卷积、向量和矩阵函数。将包含且不限于下列函数：

- 级联无限冲激响应（IIR）滤波器
- 相关
- 卷积
- 有限冲激响应（FIR）滤波器
- 窗口函数
- FFT
- LMS 滤波器
- 向量加和向量减
- 向量点积
- 向量求幂
- 矩阵加和矩阵减
- 矩阵乘

25.3.4 DSP 滤波器设计软件实用程序

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能会有所改变。有关本产品的最新信息和供货情况请浏览 **Microchip** 网站或联系当地 **Microchip** 销售办事处。

Microchip 将提供数字滤波器设计软件，使用户能够通过图形用户界面为低通、高通、带通和带阻 **IIR** 和 **FIR** 滤波器（包括 16 位的小数数据大小的滤波器系数）开发优化的汇编代码。应用程序开发人员输入所需的滤波器频率参数，而这种软件工具则可开发出滤波器代码和系数。该软件还将产生理想的滤波器频率响应和时域图以进行分析。

软件支持多达 513 个抽头的 **FIR** 滤波器和长达 10 个级联部分的 **IIR** 滤波器。

所有生成的 **IIR** 和 **FIR** 程序均为汇编语言产生且可以由汇编语言和 **C** 语言调用。该软件支持 **Microchip** **MPLAB C30 C** 编译器。

25.3.5 外设驱动程序库

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能会有所改变。有关本产品的最新信息和供货情况请浏览 **Microchip** 网站或联系当地 **Microchip** 销售办事处。

Microchip 提供外设驱动程序库，支持 **dsPIC30F** 硬件外设的设置和控制，这些外设包括但不限于：

- 模数转换器
- 电机控制 **PWM**
- 正交编码器接口
- **UART**
- **SPI™**
- 数据转换器接口
- **I²C™**
- 通用定时器
- 输入捕捉
- 输出比较 / 简单的 **PWM**

25.3.6 CAN 库

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能已经过更新。有关本产品的最新信息和供货情况请浏览 [Microchip](#) 网站或联系当地 [Microchip](#) 销售办事处。

Microchip 提供 CAN 驱动程序库，支持 dsPIC30F CAN 外设。支持的 CAN 功能包括：

- 初始化 CAN 模块
- 设置 CAN 操作模式
- 设置 CAN 波特率
- 设置 CAN 屏蔽
- 设置 CAN 过滤器
- 发送 CAN 报文
- 接收 CAN 报文
- 中止 CAN 序列
- 获取 CAN TX 错误计数
- 获取 CAN RX 错误计数

25.3.7 实时操作系统（RTOS）

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能已经过更新。有关本产品的最新信息和供货情况请浏览 [Microchip](#) 网站或联系当地 [Microchip](#) 销售办事处。

我们将提供 dsPIC30F 产品系列的实时操作系统（RTOS）解决方案。这些 RTOS 解决方案将提供必需的功能调用和操作系统子程序，用于给多任务应用场合编写高效的 C 语言和/或汇编代码。另外，RTOS 解决方案将使用在那些程序空间，更重要的是数据存储空间资源有限的应用场合。我们还提供可配置的并经过优化的内核，用于支持各种 RTOS 应用程序的需求。

RTOS 解决方案的范围从全真的抢占式多任务调度程序到合作型调度程序，它们均设计为在 dsPIC30F 器件上高效运行。根据 RTOS 的实现情况，系统内核中提供的函数调用包括：

- 控制任务
- 发送和接收消息
- 处理事件
- 控制资源
- 控制信号
- 以各种方式调整时序
- 提供存储器管理
- 处理中断和交换任务

大部分函数都用 ANSI C 编写，但那些对时间要求严格的函数除外，这些函数用汇编语言，从而缩短执行时间以最大限度地提高代码效率。Microchip MPLAB C30 C 编译器支持 ANSI C 和汇编程序。

随 RTOS 一起提供有电子文档让用户快速理解 RTOS 并在其应用中运用。

25.3.8 OSEK 操作系统

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能已经过更新。有关本产品的最新信息和供货情况请浏览 **Microchip** 网站或联系当地 **Microchip** 销售办事处。

我们将开发符合汽车软件标准 OSEK/VDX 的操作系统以支持 dsPIC30F 产品系列。OSEK 的功能“Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug”（开放系统和相应的汽车电子设备接口）与 VDX “Vehicle Distributed eXecutive” 颁布的 OSEK/VDX 是一致的。

我们还将提供基于标准接口和协议的结构化和模块化软件实现。结构化和模块化软件的实现将为分布式汽车控制单元提供便携和可扩展性。

将提供各种 OSEK COM 模块，如：

- OSEK/COM 标准 API
- OSEK/COM 通信 API
- OSEK/COM 网络 API
- OSEK/COM 标准协议
- OSEK/COM 器件驱动器接口

Microchip 还将提供内部和外部 CAN 驱动器支持。物理层将被集成到通信控制器的硬件中，在 OSEK 规范中不会涉及到它。

大部分模块函数都用 ANSI C 编写，但那些对时间要求严格的函数和外设实用程序除外，这些函数用汇编语言，从而缩短执行时间以最大限度地提高代码效率。该软件支持 Microchip MPLAB C30 C 编译器。

25.3.9 TCP/IP 协议栈

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能已经过更新。有关本产品的最新信息和供货情况请浏览 **Microchip** 网站或联系当地 **Microchip** 销售办事处。

Microchip 将为在 dsPIC30F 系列器件上实现的 Internet 接入端，提供各种传输控制协议 / 互联网协议（Transmission Control Protocol/Internet Protocol, TCP/IP）堆栈层解决方案。我们将提供精简的和完全的协议栈实现，以便让用户为其应用选择最佳的 TCP/IP 协议栈解决方案。

我们还将提供各种协议层（如 FTP、TFTP 和 SMTP）、传输层和网络层（如 TCP、UDP、ICMP 和 IP）以及网络访问层协议（如 PPP、SLIP、ARP 和 DHCP）。我们还提供各种配置，比如最小 UDP/IP 协议栈可用于有限的连接需求。

大多数堆栈协议函数都将使用 Microchip MPLAB C30 C 语言开发和优化。对于特定 dsPIC30F 硬件外设和以太网驱动程序可能用汇编语言编制代码以优化代码大小和执行时间。这些专门用汇编语言开发的程序可以用汇编和 C 语言调用。

随 TCP/IP 协议栈一起提供有电子文档，供用户快速理解协议栈并在实际中应用。

25.3.10 V.22/V.22bis 和 V.32 规范

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能已经过更新。有关本产品的最新信息和供货情况请浏览 **Microchip** 网站或联系当地 **Microchip** 销售办事处。

Microchip 将提供符合 ITU V.22/V.22bis (1200/2400 bps) 和 V.32 (非格式编码速率为 9600 bps) 的调制解调器规范, 以支持 “已连接” 应用。

这些调制解调器规范将给为数众多、各种门类的应用带来好处。以下列出了一些应用场合:

- 通过 Internet 实现家庭安全系统
- 通过 Internet 连接电表、气表和水表
- 通过 Internet 连接自动售货机
- 智能电器
- 工业监控
- POS 终端
- 机顶盒
- 升降盒
- 消防面板

多数 ITU 规范模块都将使用 **Microchip** MPLAB C30 C 语言开发和优化。对于特定的 dsPIC30F 硬件外设和重要的发送器和接收器滤波程序可能用汇编语言编制代码以优化代码大小和执行时间。这些专门用汇编语言开发的程序可以用汇编和 C 语言调用。

随调制解调器库提供有电子文档, 供用户快速理解库函数并在实际中运用。

25.4 dsPIC30F 硬件开发板

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能已经过更新。有关本产品的最新信息和供货情况请浏览 **Microchip** 网站或联系当地 **Microchip** 销售办事处。

Microchip 将首先提供三种硬件开发板, 它们将为应用开发人员提供快速进行样机开发和验证关键设计要求的工具。每块板都具有主要的 dsPIC30F 外设并支持 **Microchip** MPLAB 在线调试器 (ICD 2) 工具, 以用于对 dsPIC30F 器件进行高成本效益的调试和编程。初始提供的三种开发板是:

- 通用开发板
- 电机控制开发系统
- 网络连接开发板

25.4.1 通用开发板

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能已经过更新。有关本产品的最新信息和供货情况请浏览 **Microchip** 网站或联系当地 **Microchip** 销售办事处。

dsPIC30F 通用开发板将为应用程序设计人员提供一款低成本的开发工具，使他们熟悉 dsPIC30F 的 16 位架构、高性能外设和强大的指令集。开发板将作为理想的样机开发工具，可以快速地开发和验证关键设计要求。

通用开发板的一些关键功能和特性包括：

- 支持各种 dsPIC30F 封装
- CAN 通信通道
- RS-232 和 RS-485 通信通道
- 带输入 / 输出插孔的编解码器接口
- 在线调试器接口
- MPLAB ICE 4000 仿真支持
- Microchip 温度传感器
- Microchip 运放电路，支持用户输入信号
- Microchip 数模转换器
- 2x16 LCD
- 通用样机区
- 各种 LED、开关和电位计

随通用开发板提供 9V 电源、RS-232 I/O 线缆、已预编程的 dsPIC30F 器件、示例软件和相关文档，供用户练习开发板演示程序。

25.4.2 电机控制开发系统

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能已经过更新。有关本产品的最新信息和供货情况请浏览 **Microchip** 网站或联系当地 **Microchip** 销售办事处。

dsPIC30F 电机控制开发系统首先为应用程序开发人员提供三个主要组件，用于快速进行 BLDC、PMAC 和 ACIM 应用的样机开发和验证。这三个主要组件是：

- dsPIC30F 电机控制主板
- 三相低压电源模块
- 三相高压电源模块

主控制板将支持 dsPIC30F6010 器件，各种外设接口和一个用户定制的接口主机系统，从而可以连接不同的电机模块。该控制板还包括用于连接机械位置传感器的接口（如增量旋转编码器和霍尔效应传感器）和用于定制电路的实验板区域。由标准插入式变压器为主控板供电。

低压电源模块是为那些要求 DC 总线电压低于 60 V 而输出功率高达 400W 的 3 相电机应用而优化的。三相低压电源模块旨在为 BLDC 和 PMAC 电机供电。

高压电源模块是为那些要求 DC 总线电压高达 400 V 而输出功率高达 1 KW 的 3 相电机应用而优化的。此高压模块具有由 dsPIC30F 器件控制的有源功率因数校正电路。该电源模块旨在用于 AC 感应电机和电源逆变器的应用。

上述两种电源模块还具有自动故障保护和与控制接口电隔离的功能。它们还将为主控制板提供预调节的电压和电流信号。所有与电机控制电路隔离的反馈位置信息的器件，如增量编码器、霍尔效应传感器或转速传感器，均直接连接到主控制板。两种模块都配备了电机制动电路。

25.4.3 网络连接开发板

注： 在编写此手册时本产品尚在开发阶段。产品的某些细节可能已经过更新。有关本产品的最新信息和供货情况请浏览 **Microchip** 网站或联系当地 **Microchip** 销售办事处。

dsPIC30F 网络连接开发板将为应用程序开发人员提供一个基础平台，用于开发和评估各种网络连接解决方案，实现 TCP/IP 协议层以及 V.22/V.22bis 和 V.32（非格式编码）ITU 规范，通过 PSTN 或以太网信道。

网络连接开发板的一些关键功能和特性包括：

- 支持 dsPIC30F6014 器件
- 介质访问控制（MAC）和 PHY 接口
- 带 DAA/AFE 的 PSTN 接口
- RS-232 和 RS-485 通信通道
- 在线调试器接口
- MPLAB ICE 4000 仿真支持
- Microchip 温度传感器
- Microchip 数模转换器
- 2x16 LCD
- 通用样机区
- 各种 LED、开关和电位计

随网络连接开发板提供 9V 电源、RS-232 I/O 线缆、带示例连接软件和相关文档的已预编程的 dsPIC30F 器件，供用户练习开发板连接演示程序。

25.5 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 dsPIC30F 系列产品而编写的，但是概念是相关的，通过适当修改即可使用，但在使用中将受到一定限制。当前与开发工具支持相关的应用笔记有：

标题	应用笔记编号
目前没有相关的应用笔记。	

注：如需获取更多 dsPIC30F 系列器件的应用笔记和代码示例，请访问 Microchip 网站 (www.microchip.com)。

25.6 版本历史

版本 A

这是描述 dsPIC30F 开发工具支持的初始发行版本。

版本 B

手册的本章没有任何技术内容或编辑排版上的修订，然而我们谨更新此节以反映本手册已更新到版本 B。

版本 C

手册的本章没有任何技术内容修订，然而我们谨更新此节以反映本手册已更新到版本 C。

第 26 章 附录

目录

本章包括下列主题：

附录 A: I ² C™ 概述	26-2
附录 B: CAN 概述	26-12
附录 C: 编解码器协议概述	26-25

附录 A: I²C™ 概述

本附录提供 I²C™ 总线的概述，其中第 A.2 节“对 I²C 器件寻址”讨论了在 I²C 模式下 SSP 模块的工作原理。

I²C 总线是双线制串行接口总线。其原始规范（或标准模式）是为最高 100 kbps 的数据传输速率而制定的。I²C 还支持增强型规范（或快速模式，400 kbps）。标准模式器件和快速模式器件连接在同一总线上时，如果总线以快速器件的工作速度运行，则两类器件均可正常工作。

I²C 接口使用了综合的协议以确保数据发送与接收的可靠性。当发送数据时，其中一个器件作为主机，它启动总线上的传输并产生时钟信号以允许该传输，而其他器件则充当从机。除了全局呼叫支持外，从机协议的各部分都由 SSP 模块的硬件实现，而主机协议的各部分则需要 PIC16CXX 的软件中处理。MSSP 模块可以实现所有的 I²C 主机协议、全局呼叫地址和最大速率可达 1 Mbps 的数据传输。Microchip 的某些串行 EEPROM 支持 1 Mbps 的数据传输。表 A-1 定义了 I²C 总线的部分术语。

在 I²C 接口协议中，每个器件都有一个地址。当主机要启动一次数据传输时，它首先发送与之“通话”的器件地址。所有器件都“侦听”该地址，查看是否与自己的地址匹配。在该地址中有一位用于指定主机是希望读还是希望写它寻址的从机。在数据传输过程中，主机和从机总是工作在相反的模式（发送器 / 接收器）下。也就是说主机和从机只能工作在以下两种关系之一：

- 主机——发送器，从机——接收器
- 从机——发送器，主机——接收器

在这两种情况下，都由主机产生时钟信号。

为了实施总线的“线与”功能，时钟线（SCL）和数据线（SDA）的输出都必须采用漏极开路或集电极开路的电路。需要外接上拉电阻，以保证总线在没有器件将其拉低时为高电平。I²C 总线上连接的器件数量仅受到最大 400 pF 的总线负载规范和寻址能力的限制。

A.1 启动和终止数据传输

在总线无数据传输时（空闲时），时钟线（SCL）和数据线（SDA）都通过外部上拉电阻拉为高电平。由启动和停止条件决定数据传输的启动和停止。启动条件定义为当 SCL 为高电平时，SDA 发生由高到低的转变；停止条件定义为当 SCL 为高电平时，SDA 发生由低到高的转变。图 A-1 显示了启动和停止条件。主机产生这些条件以启动和终止数据传输。根据启动和停止条件的定义，当数据正在传输时，只有在 SCL 线为低电平时 SDA 线的状态才允许改变。

图 A-1: 启动和停止条件

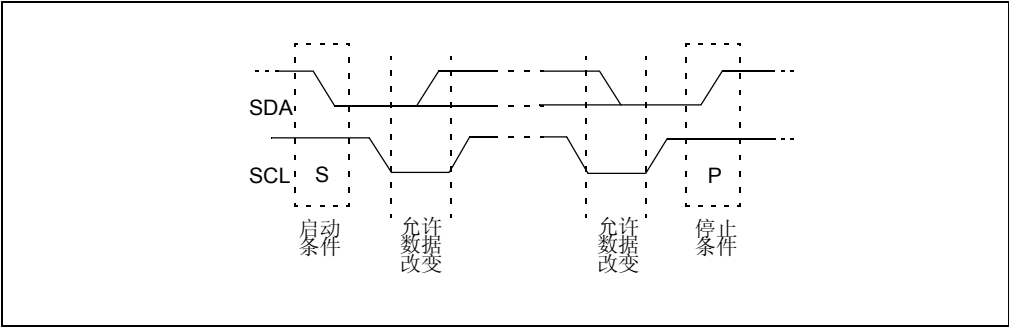


表 A-1: I²C 总线术语表

术语	说明
发送器	向总线发送数据的器件。
接收器	从总线接收数据的器件。
主机	启动传输、产生时钟并终止传输的器件。
从机	被主机寻址的器件。
多主机	系统中有一个以上的主器件。这些主器件可尝试同时对总线进行控制而不会破坏报文。
仲裁	保证只有一个主器件控制总线的过程，可确保传输的数据不会被破坏。
同步	两个或多个器件的时钟信号同步的过程。

A.2 对 I²C 器件寻址

有两种地址格式：最简单的一种是带有 7 位地址格式 $\overline{R/W}$ 位（图 A-2）。另一种是较复杂的带有 10 位地址格式加上一个 $\overline{R/W}$ 位（图 A-3）。对于 10 位地址格式，必须发送两个字节。第一个字节的前五位用来指定当前采用的是 10 位地址格式，发送的第一个字节中包括 5 位用于指定 10 位地址的 5 位以及地址的两个 MSb 和 1 位 $\overline{R/W}$ 位。第二个字节是其余的 8 位地址。

图 A-2: 7 位地址格式

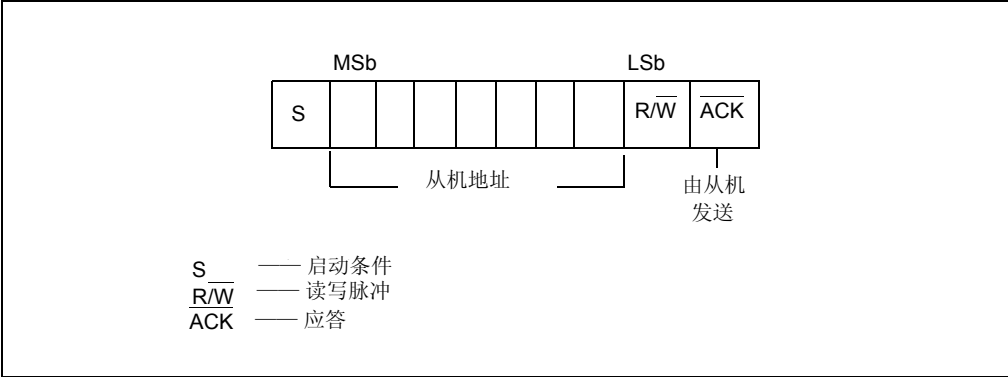
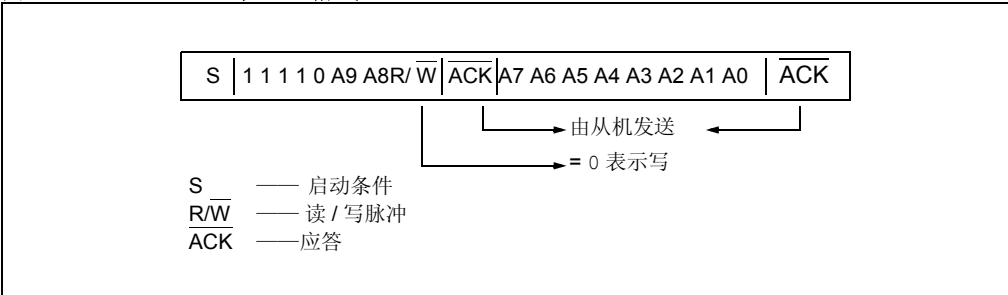


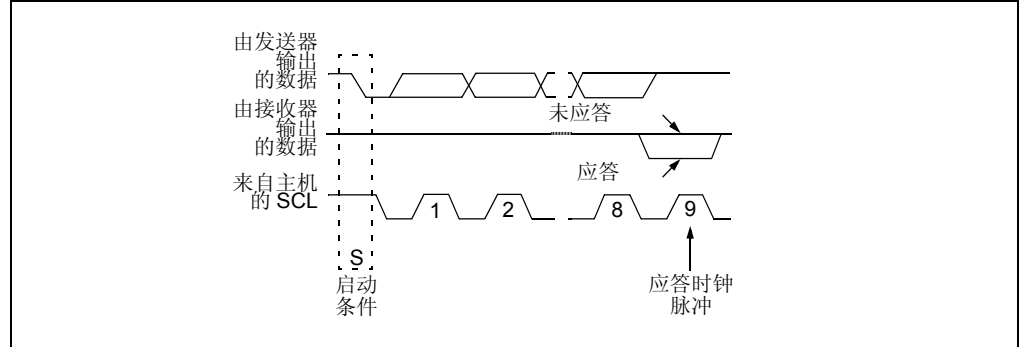
图 A-3: I²C 10 位地址格式



A.3 发送应答

所有数据都必须逐字节发送，每次数据传输所发送的字节数没有限制。接收到每个字节后，从机（接收器）会产生一个应答位（ACK）（图 A-4）。如果从机（接收器）没有对从机地址或已接收的数据作出应答，主机必须中止发送。从机必须保持 SDA 为高电平，以便主机可以产生停止条件（图 A-1）。

图 A-4: 从机——接收器应答



如果主机接收数据（主机作为接收器），则除了最后一个字节外，主机在收到每一个数据字节后都会产生一个应答信号。如果主机没有发出应答信号（未作出确认），则通知从机（发送器）结束数据传输。然后从机释放 SDA 线，以便使主机能够产生停止条件。主机也可以在确认脉冲输出期间产生停止条件，以有效终止数据传输。

如果从机需要延迟下一个字节的传输，可以通过保持 SCL 线为低电平，迫使主机进入等待状态。当从机释放 SCL 线时，将继续数据传输。这就使得从机在允许时钟启动前先移入接收到的数据或取出需要发送的数据。这种等待状态技术也可以用在对齐的处理上，如图 A-5 所示。

图 A-5: 数据传输等待状态

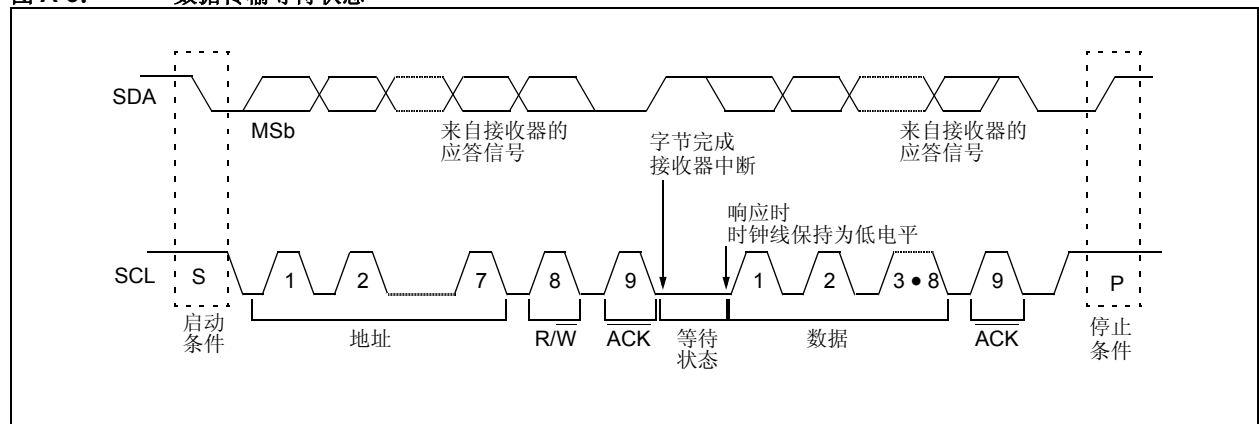


图 A-6 和图 A-7 描述了主机作为发送器和主机作为接收器时的数据传输序列。

图 A-6: 主机作为发送器的序列

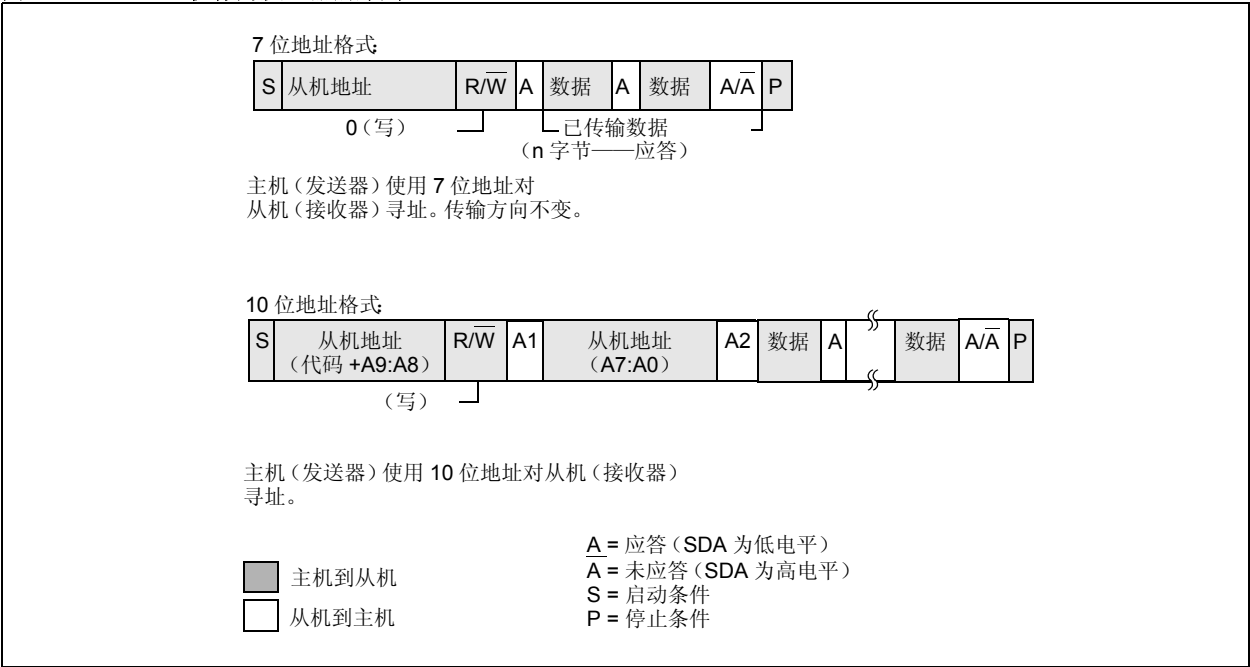
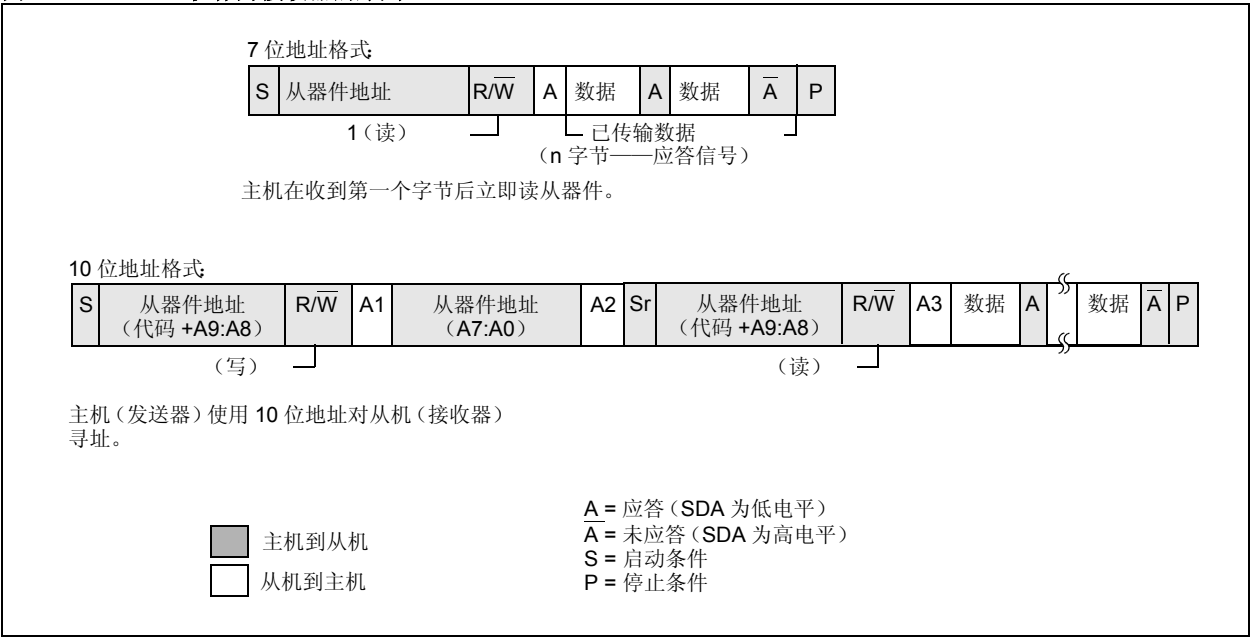
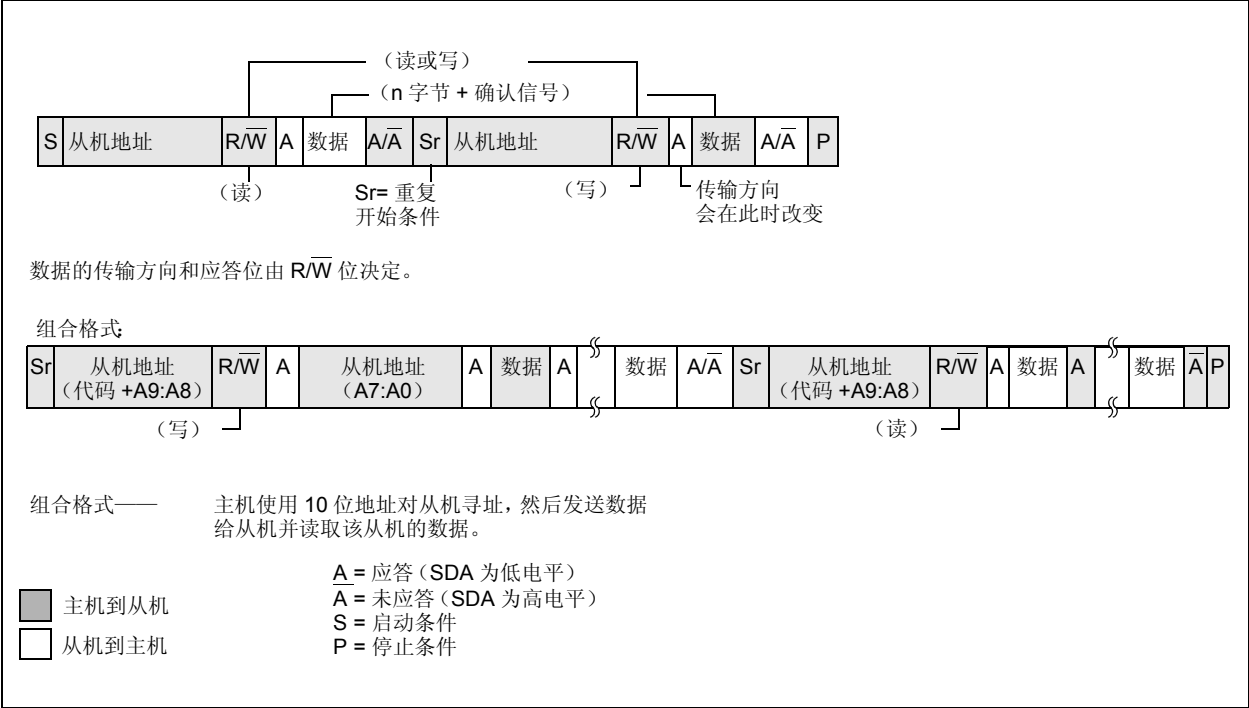


图 A-7: 主机作为接收器的序列



如果主机不想放弃总线控制权（通过产生停止条件），必须产生一个重复启动条件（Sr）。这个条件和启动条件相同（SCL 保持为高时，SDA 由高变低），只是它发生在数据传输应答脉冲产生之后（而不是在总线处于空闲状态时）。这样主机就可以把“命令”发送给从机，然后接收所需要的信息或对另外一个从器件进行寻址。图 A-8 所示为此过程。

图 A-8： 组合格式



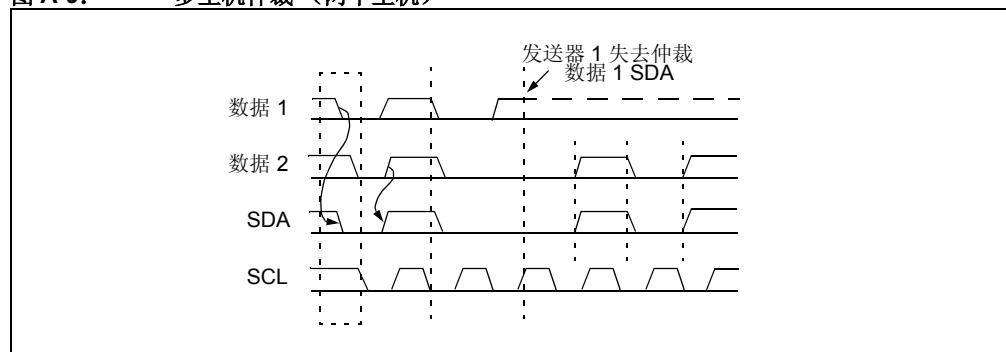
A.4 多主机

I²C 协议允许系统有一个以上的主机，这称为多主机系统。当两个或两个以上的主机试图同时传输数据时，就会产生总线仲裁和同步。

A.4.1 仲裁

当 SCL 线为高电平时，在 SDA 线上进行仲裁。当一个主机在另一个主机发送低电平时发送了高电平，则第一个主机失去仲裁（图 A-9）并关闭其数据输出级。失去仲裁的主机在数据字节结束（即失去仲裁）之前可以产生时钟脉冲。如果多个主器件对同一器件寻址，则会继续进行数据仲裁。

图 A-9: 多主机仲裁（两个主机）



兼有从机功能且失去仲裁的主机，必须立即切换到从机（接收器）模式。这是因为赢得仲裁的主机（发送器）可能会对它进行寻址。

不允许在以下情况进行仲裁：

- 重复启动条件
- 停止条件和数据位之间
- 重复启动条件和停止条件之间

编程时应该注意以确保不会发生这些情况。

A.4.2 时钟同步

时钟同步发生在器件开始仲裁之后。它是由连接到 SCL 线上的“线与”功能完成的。当 SCL 线的电平由高变低时，相关器件就开始对其时钟低电平时间进行计算。一旦器件时钟变为低电平，那么在其时钟线（SCL）变为高电平之前，SCL 线将一直保持为低电平。如果另一个器件的时钟仍处于低电平状态，那么器件时钟由低电平变为高电平不会改变 SCL 线的状态。SCL 线保持低电平的时间由低电平时间最长的器件决定。低电平持续时间较短的器件将进入高电平等待状态，直到 SCL 线变为高电平。此时，所有器件开始计算时钟高电平的时间。第一个变为低电平的器件又会把 SCL 拉低。SCL 线保持高电平的时间由高电平时间最短的器件决定，如图 A-10 所示。

图 A-10: 时钟同步

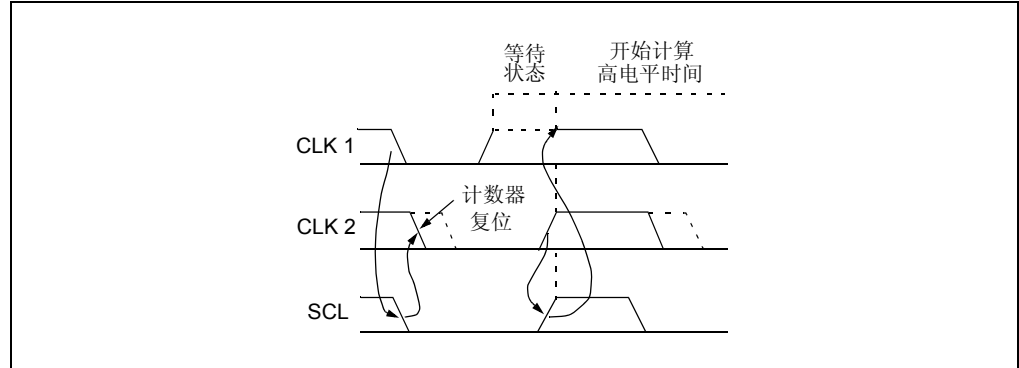


表 A-2 和表 A-3 显示了 I²C 总线规范。“参数编号”一行是为了方便用户在器件数据手册中查找相应的参数。图 A-11 和图 A-12 显示了相应波形的时序。

图 A-11: I²C 总线启动 / 停止位时序规范

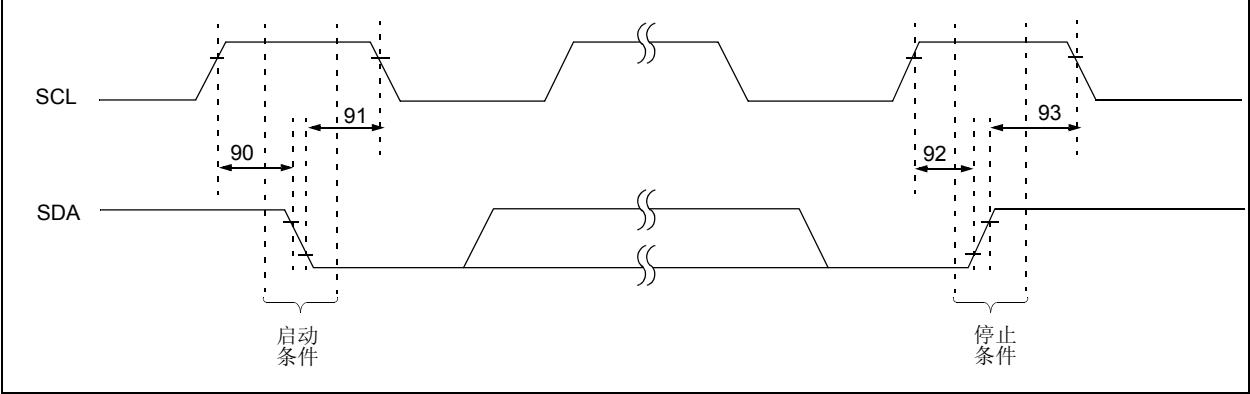


表 A-2: I²C 总线启动 / 停止位时序规范

参数编号	符号	特性		最小值	典型值	最大值	单位	条件
90	TSU:STA	启动条件建立时间	100 kHz 模式	4700	—	—	ns	仅与重复启动条件相关
			400 kHz 模式	600	—	—		
91	THD:STA	启动条件保持时间	100 kHz 模式	4000	—	—	ns	在这段时间后，将产生第一个时钟脉冲
			400 kHz 模式	600	—	—		
92	TSU:STO	停止条件建立时间	100 kHz 模式	4700	—	—	ns	
			400 kHz 模式	600	—	—		
93	THD:STO	停止条件保持时间	100 kHz 模式	4000	—	—	ns	
			400 kHz 模式	600	—	—		

图 A-12: I²C 总线数据时序规范

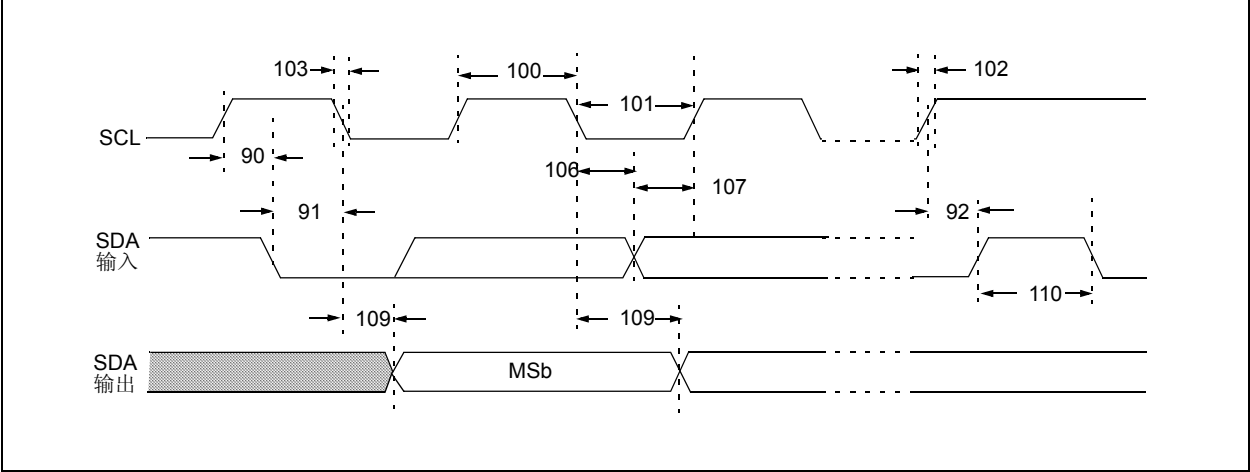


表 A-3: I²C 总线数据时序规范

参数编号	符号	特性	最小值	最大值	单位	条件
100	T _{HIGH}	时钟高电平时间	100 kHz 模式	4.0	—	μs
			400 kHz 模式	0.6	—	μs
101	T _{LOW}	时钟低电平时间	100 kHz 模式	4.7	—	μs
			400 kHz 模式	1.3	—	μs
102	T _R	SDA 和 SCL 上升时间	100 kHz 模式	—	1000	ns
			400 kHz 模式	20+0.1C _b	300	ns 指定 C _b 在 10 到 400 pF 之间
103	T _F	SDA 和 SCL 下降时间	100 kHz 模式	—	300	ns
			400 kHz 模式	20+0.1C _b	300	ns 指定 C _b 在 10 到 400 pF 之间
90	T _{SU:STA}	启动条件建立时间	100 kHz 模式	4.7	—	μs
			400 kHz 模式	0.6	—	μs
91	T _{HD:STA}	启动条件保持时间	100 kHz 模式	4.0	—	μs
			400 kHz 模式	0.6	—	μs
106	T _{HD:DAT}	数据输入保持时间	100 kHz 模式	0	—	ns
			400 kHz 模式	0	0.9	μs
107	T _{SU:DAT}	数据输入建立时间	100 kHz 模式	250	—	ns
			400 kHz 模式	100	—	ns
92	T _{SU:STO}	停止条件建立时间	100 kHz 模式	4.7	—	μs
			400 kHz 模式	0.6	—	μs
109	T _{AA}	时钟输出有效时间	100 kHz 模式	—	3500	ns
			400 kHz 模式	—	1000	ns
110	T _{BUF}	总线空闲时间	100 kHz 模式	4.7	—	μs
			400 kHz 模式	1.3	—	μs
D102	C _b	总线的容性负载	—	400	pF	

注 1: 作为发送器, 为避免意外产生启动或停止条件, 器件必须提供此内部最小延迟时间, 以补偿 SCL 下降沿的未定义区域 (最小值 300 ns)。

2: 快速模式的 I²C 总线器件可以在标准模式 I²C 总线系统中使用, 但必须满足 T_{SU:DAT} ≥ 250 ns 的要求。如果快速模式器件没有延长 SCL 信号的低电压周期, 则必然满足此条件。如果该器件延长了 SCL 信号的低电压周期, 则它必须将下一个数据位输出到 SDA 线, 根据标准模式 I²C 总线规范, 释放 SCL 先之前的时间为: T_{Rmax} + T_{SU: DAT} = 1000 + 250 = 1250 ns。

附录 B: CAN 概述

本附录提供了控制器局域网（CAN）总线的概述。在本参考手册的 CAN 一章中讨论了用于 dsPIC30F 硬件模块的 CAN 协议的实现过程。

B.1 CAN 总线背景知识

控制器局域网（CAN）是串行通信协议，它能够有效支持高性能的分布式实时控制。Robert Bosch GmbH 在 1991 年的 CAN 规范 V2.0B 中完整定义了 CAN 协议。

CAN 应用涵盖从高速网络到低成本多路复用线路的各种领域。汽车电子设备（即，引擎控制单元、传感器和防滑系统等）均是使用 CAN 以最大 1 Mb/s 的比特率连接的。CAN 网络可以用来取代汽车中的线路连接以有效节约成本。CAN 总线在噪声环境中的可靠性及其故障状态检测和从故障状态恢复的能力使其适用于 DeviceNet、SDS 和其他现场总线协议等工控应用。

CAN 是具有一条逻辑总线线路的异步串行总线系统。它具有开放式线性总线型网络结构，总线节点平均分布。CAN 总线由两个或多个节点组成。总线上的节点数量可以动态改变而不会干扰其他节点的通信。这使得总线节点可方便地连接和断开（如为了增加系统功能、错误恢复或总线监测）。

总线逻辑采用“线与”机制，“隐性”位（通常为逻辑 1 但非必须如此）被“显性”位覆盖（通常逻辑电平为 0）。只要没有总线节点在发送显性位，总线线路就处于隐性状态，但是只要任一总线节点发送了显性位，就会产生显性总线状态。因此，对于 CAN 总线线路，必须选择能传送两种可能位状态（显性和隐性）的传输介质。其中最常见也最便宜的方法是使用双绞线。此时总线线路被称为“CANH”和“CANL”，并且可以直接或通过连接器与节点相连。CAN 协议中没有定义有关连接器使用的标准。双绞线的端接是通过在总线线路两端连接端接电阻完成的。当总线长度不超过 40 米时，最大总线速度可达 1 Mb。如果总线长度超过 40 米，总线速度必须降低（将总线速度降到 40 Kb 可使总线长度达到 1000 米）。对于 1000 米以上的总线，应该使用特殊的驱动器。无需额外设备，即至少可以连接 20 个节点。由于发送的差分性，即两条总线线路受 EMI 的影响相同，使差分信号不受影响，所以 CAN 本质上并不容易受到辐射电磁能的影响。也可以屏蔽总线线路以降低来自总线的电磁辐射，尤其是在高波特率下。

二进制数据是按 NRZ 码（非归零码，低电平 = 显性状态，高电平 = 隐性状态）编码的。使用位填充以确保所有总线节点时钟同步。这意味着在报文发送时，最多可以有 5 个连续位具有相同的极性。只要发送了相同极性的 5 个连续位，在继续发送后续位前，发送器将在比特流中插入一个极性相反的位。接收器也会检查同极性位的个数，并把填充位从比特流中删除（解填充）。

在 CAN 协议中，不对总线节点寻址。地址信息包含在发送的报文中。这是通过标识符（每个报文的组成部分）实现的，标识符标识了报文内容（如引擎速度和油温等）。另外，标识符还会指出报文的优先级。标识符的二进制值越小，报文的优先级就越高。

对于总线仲裁，采用的是带有 NDA 的 CSMA/CD（带有非破坏性仲裁的载波侦听多路访问 / 冲突检测）。如果总线节点 A 要通过网络发送报文，它首先会检查总线是否处于空闲状态（“载波侦听”）（即没有正在进行发送的节点）。如果是这样（并且没有其他节点希望在此刻开始发送），节点 A 就成为总线主控节点并发送其报文。在节点 A 发送第一个发送位（帧起始位）时，所有其他节点都会切换到接收模式。在正确接收了报文后（每个节点均对报文进行了接收应答），每个总线节点都会检查报文标识符，如果需要的话还将存储报文。否则，就将报文丢弃。

如果两个或多个总线节点同时开始其发送（“多路访问”），此时可通过按位仲裁（“冲突检测 / 非破坏性仲裁”加上“线与”机制和“显性”位覆盖“隐性”位）避免报文冲突。每个节点都发送报文的标识符位（先发送 MSb）并监视总线电平。如果节点发送的是隐性标识符位但是读回的是显性标识符位，则它将失去总线仲裁并切换到接收模式。当与之发生冲突的节点的报文标识符有较低的二进制值（显性状态 = 逻辑 0）时就会产生这种情况，即冲突节点发送的报文具有较高的优先级。这样，具有最高优先级报文的总线节点不需要花费时间重发这一报文就可以赢得仲裁。一旦总线返回到空闲状态，其他节点就会自动尝试重新发送。不允许不同的节点采用同一标识符来发送报文，因为这样的话仲裁可能会失败导致发生冲突和错误。

早期的 CAN 规范（1.0、1.2 和 2.0A 版）定义报文标识符为 11 位长，从而给出了 2048 个种报文标识符。后来此规范进行了更新（到 2.0B 版）后消除了这一限制。CAN 规范 2.0B 版允许的可用报文标识符的长度为 11 和 / 或 29 位（29 位长的标识符可提供超过 536,000,000 种不同的报文标识符）。CAN 2.0B 版也被称为“扩展 CAN”；而 1.0、1.2 和 2.0A 版则被称为“标准 CAN”。

B.2 不同的 CAN 实现方法

B.2.1 标准 CAN 和扩展 CAN

根据 CAN 规范 V2.0A，只包含 11 位标识符的数据帧和远程帧被称为标准帧。这些帧可以用来标识 2048 种不同的报文（标识符为 0 — 2047）。但是，优先级最低的 16 条报文（2032 — 2047）是保留不用的。根据 CAN 规范 V2.0B，扩展帧具有 29 位的标识符。如前所述，29 位标识符是由 11 位标识符（“基本 ID”）和 18 位扩展标识符（“扩展 ID”）组成的。

CAN V2.0A 规定的 CAN 模块只能根据标准 CAN 协议发送和接收标准帧。发送和接收使用 29 位标识符的报文会产生错误。如果是 CAN V2.0B 规定的器件，另外还有一种区别。称为“Part B Passive”的模块只能发送和接收标准帧，但是在遇到扩展帧时不会产生错误帧。称为“Part B Active”的器件能够发送和接收标准帧和扩展帧。

B.3 基本 CAN 和完全 CAN

还有一个 CAN 特性与 CAN 模块和主机 CPU 之间的接口有关，该特性将 CAN 芯片分为“基本 CAN”和“完全 CAN”器件。这与所使用的协议（标准或扩展 CAN）没有关系，可以允许在同一网络中同时使用基本 CAN 和完全 CAN 器件。

在基本 CAN 器件中，硬件只实现了协议的基本功能（如比特流的产生和检查）。判断接收到的报文是否需要被存储（接收过滤）以及整个报文管理都必须由软件完成（即由主机 CPU 完成）。CAN 芯片通常只提供一个发送缓冲器和一到两个接收缓冲器。这样使用基本 CAN 模块的主机 CPU 负担会非常重，因此这些器件只能使用在低波特率和只允许几条不同报文同时传输的低总线负荷的场合使用。基本 CAN 的优势在于芯片体积小，可以降低器件成本。

完全 CAN 器件用硬件实现整个总线协议，包括接收过滤和报文管理。它们包含几个所谓的报文对象，可以处理标识符、数据、方向（接收或发送）以及标准 CAN/ 扩展 CAN 信息。在器件的初始化过程中，主机 CPU 决定哪些报文要发送、哪些报文要接收。如果某个接收到的报文的标识符与已编程的（接收）报文对象之一相匹配，中断就会通知主机 CPU。这样，就减轻了 CPU 的负担。使用完全 CAN 器件可以处理高波特率和允许很多报文同时传输的高总线负荷。但这些芯片比基本 CAN 器件昂贵。

许多完全 CAN 芯片还提供“基本 CAN 功能”。可以编程某个报文对象，使之将每个报文存储起来而不与其他报文对象相匹配。这对于许多应用都很有帮助。

B.4 ISO 模型

如图 B-1 所示，ISO/OSI 参考模型用于定义通信系统协议的层。在最高层，应用程序需要互相通信。在最低层，使用某些物理介质提供电信令。

协议的高层由软件控制。通常硬件只能实现到应用层。在 CAN 总线规范中，并没有对报文类型或被传输报文的内容及意义进行定义。这些内容在有些系统中定义，比如 Volcano（Volvo 汽车 CAN 规范）、J1939（美国重型卡车多路布线规范）以及 Allen-Bradly DeviceNet 和 Honeywell SDS（工业协议的两个范例）。

CAN 总线模块定义包含整个协议中的两个级别如下。

- 数据链路层
 - 逻辑链路控制（LLC）子层
 - 介质访问控制（MAC）子层
- 物理层
 - 物理信令（PLS）子层

LLC 子层与报文过滤、过载通知和错误恢复管理有关。LLC 子层的功能范围包括：

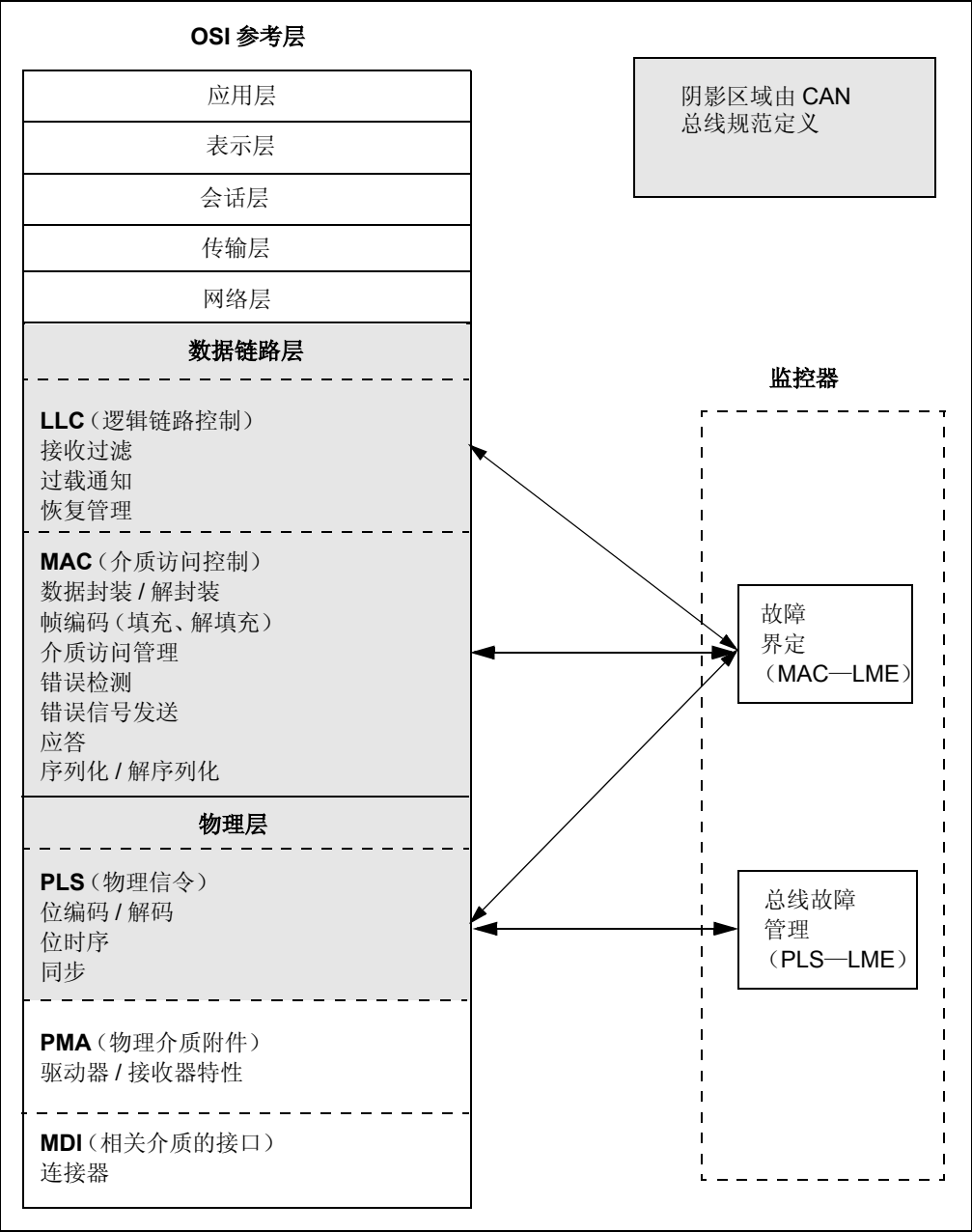
- 为数据传输和远程数据请求提供服务
- 决定 LLC 子层接收的哪些信息将被实际接收
- 提供错误恢复管理和过载通知的方式

MAC 子层是 CAN 协议的核心。MAC 子层定义了传输协议（即控制帧、执行仲裁、错误检测、错误信号产生以及故障界定）。该层提供从 LLC 子层的接收的报文，并接收将要发送到 LLC 子层的报文。判断总线是处于空闲状态以开始新的发送还是刚刚开始接收都是在 MAC 子层内完成的。MAC 子层由一个叫做故障界定的管理实体监管，这是一种能够把短时干扰和永久失效区分开来的自检机制。而且，位时序的某些通用功能也被认为是 MAC 子层的一部分。

物理层定义了所有电气属性下不同节点之间的实际的位传输。PLS 子层定义信号的实际发送方式，并由此对位时序、位编码和同步进行描述。

协议的较低层在驱动器 / 接收器芯片和实际接口（如双绞线或光纤等）中实现。在同一个网络中，所有节点必须使用同一个物理层。物理层的驱动器 / 接收器特性没有定义，这样可使发送介质和信令的实现根据应用进行优化。ISO11898 道路车辆多路布线规范（英文）中对最常见的例子给出了定义。

图 B-1: ISO/OSI 参考模型中的 CAN 总线



B.5 CAN 总线功能

CAN 具有以下属性：

- 报文优先级控制
 - 确保响应时间
 - 配置灵活
 - 带时间同步的组播接收
 - 全系统数据一致性
 - 多主机
 - 错误检测和信令
 - 损坏报文的自动重发
 - 区分节点的暂时错误和永久无效并自动关闭出错节点
1. 报文：总线上的信息是以固定格式的报文形式发送的，报文的长度不同但有限制。当总线空闲时，任何连接到总线的部件都可以开始发送新的报文。
 2. 信息路由：在 CAN 系统中，CAN 节点不会使用系统配置（如站点地址）的任何信息。
 3. 系统灵活性：可以向 CAN 网络添加节点而无需在节点和应用层的软硬件上进行任何修改。
 4. 报文路由：报文的内容是以标识符命名的。标识符没有指出报文的目的地，而是说明了数据的意义，因此网络中的所有节点都能够通过报文过滤判断它们是否需要对其数据进行处理。
 5. 组播：作为报文过滤的一个结果，任意数量的节点都能接收并同时处理一个报文。
 6. 数据一致性：在 CAN 网络中确保报文被所有节点同时接收或不被任何节点接收。这样，就能通过组播和错误处理保证系统的数据一致性。
 7. 比特率：在不同系统中 CAN 的速度可能会不同。但是，在一个给定的系统中，比特率是统一且固定的。
 8. 优先级控制：标识符会在总线访问期间确定一个静态报文的优先级。
 9. 远程数据请求：通过发送远程帧，请求数据的节点可要求另一个节点发送相应的数据帧。数据帧和相应的远程帧由相同的标识符命名。
 10. 多主机：当总线空闲时，任何部件都可开始发送报文。具有较高优先级的待发送报文的部件会获得总线的访问权。
 11. 仲裁：只要总线空闲，任何部件就可开始发送报文。如果两个或多个部件同时开始发送报文就需要使用标识符进行按位仲裁以解决总线访问冲突。仲裁机制可以确保不会丢失信息或损失时间。如果使用相同标识符的数据帧和远程帧同时开始发送，则数据帧优先于远程帧。在仲裁时，每个发送器都会将待发送位的优先级与在总线上监视到的优先级对比。如果优先级相等，则该发送器可继续发送。如果发送了“隐性”级别但是监视到“显性”级别，则该器件失去仲裁，必须撤回请求并不再发送位。
 12. 安全性：为了实现数据传输的最高安全性，在每个 CAN 节点都实施了强大的错误检测、信令和自检机制。
 13. 错误检测：为了检测错误，采取了以下措施：
 - 监视（发送器将待发送位的优先级与在总线上监视到的优先级对比）
 - 循环冗余校验
 - 位填充
 - 报文帧校验

错误检测机制具有以下属性：

- 检测所有全局错误
 - 检测发送器的所有本地错误
 - 最大可检测到一个报文中的 5 个随机分布错误
 - 检测报文中长度小于 15 的突发错误
 - 检测报文中任何数量为奇数的错误。
14. 错误信令和恢复时间：任何检测到错误的节点都会标记损坏的报文。此类报文发送被中止并将自动重发。如果没有发生更多错误，从检测到错误到开始下一个报文最长需要 31 个位时间。
 15. 故障界定：CAN 节点能够区别短期干扰和无效故障。出错的节点会被关闭。
 16. 连接：CAN 串行通信链路是可连接一定数量器件的总线。理论上，此数量没有限制。实际上，器件总数受到总线时延和 / 或总线上的电气负载的限制。
 17. 单通道：总线由载有位的单个通道组成。信息可通过经过该通道传输的数据再同步的过程获得。本规范中并不规定该通道的实现方式（即单线（接地）、两条差分线路和光纤等）。
 18. 总线值：总线可以有两个互补的逻辑值，即“显性”或“隐性”。同时发送“显性”和“隐性”位，总线值将是“显性”的。例如，当执行总线的“线与”操作时，逻辑“0”代表“显性”，逻辑“1”代表“隐性”。在规范中未说明代表逻辑值的物理状态（例如电压和光）。
 19. 应答：所有的接收器都会检查待接收报文的一致性，并会对一致的报文作出应答并标志不一致的报文。
 20. 休眠模式及唤醒：要降低系统的功耗，可将 CAN 器件设置进入休眠模式，此模式下没有任何内部活动并且其总线驱动器也未与总线相连。任何总线活动或系统内部条件唤醒器件，休眠模式都会结束。在唤醒时，内部活动重新开始，但是 MAC 子层将等待系统振荡器稳定下来，并在总线驱动器再次置为“在线”前，继续等待直到其状态与总线活动同步（通过检测 11 个连续的“隐性”位）。

B.6 帧类型

B.6.1 标准数据帧

当节点希望发送数据时它会产生一个数据帧。图 B-2 所示为标准 CAN 数据帧。与其他所有帧相同，数据帧也是以帧起始（SOF，显性状态）位开始与所有节点进行硬同步。

在 SOF 之后是仲裁字段，由 12 位组成，包括 11 位的标识符（反映报文的内容和优先级）和 RTR 位（远程传输请求位）。RTR 位用于区分数据帧（RTR，显性）和远程帧。

下一个字段是控制字段，由 6 个位组成。字段的第一位称为 IDE 位（标识符扩展位），该位为显性状态，指定该帧为标准帧。接下来是保留位 RB0，这一位也被定义为显性位。控制字段的其余 4 位为数据长度码（DLC），它规定了报文中包含的数据字节数。

控制字段之后为数据字段，包含正在发送的数据字节，数据字段长度由上述数据长度码 DLC 定义（1 到 8 字节）。

数据字段之后是循环冗余字段（CRC），用来检测可能的报文传输错误。CRC 字段由一个 15 位的 CRC 序列组成，以隐性 CRC 定界符位结束。

最后是应答字段。在应答间隙位（ACK Slot bit）产生期间，发送节点发出一个隐性位。任何收到无错误帧的节点会发回一个显性位（无论该节点是否配置为接收该特定报文），确认帧已正确接收。从这一点上可以看出，CAN 属于“位回应”类（In-bit-response）协议。应答间隙以隐性确认定界符结束，该字符不能被显性位改写。

B.7 扩展数据帧

如图 B-3 所示，在扩展 CAN 数据帧中，紧随帧起始（SOF）位的是 38 位仲裁字段。仲裁字段的前 11 位为 29 位标识符的 11 个最高位（基本 ID）。紧随这 11 位的是代理远程请求（SRR）位，它以隐性状态发送。SRR 位后是 IDE 位，该位是隐性的，表示这是扩展的 CAN 帧。值得注意的是，如果在扩展帧标识符的前 11 位发送完后，总线仲裁无果，而此时参与仲裁的某个节点发出标准 CAN 帧（11 位标识符），那么由于该节点发出了显性的 IDE 位而使标准 CAN 帧赢得仲裁。另外，扩展 CAN 帧的 SRR 位应为隐性，以允许正在发送标准 CAN 远程帧的节点发出显性 RTR 位。SRR 位和 IDE 位之后是标识符的其余 18 位（扩展 ID）以及一个远程发送请求位。

为使标准帧和扩展帧都能在它们共享的网络上发送，应将 29 位的扩展报文标识符拆分成 11 位（最高位）和 18 位（最低位）两部分。拆分后可确保标识符扩展位（IDE）在标准帧和扩展帧中各位的位置保持不变。

下一个字段是控制字段，由 6 个位组成。控制字段前两位为保留位，为显性状态。控制字段的其余 4 位为数据长度码（DLC），它规定了数据字节数。

扩展数据帧的其他部分（数据字段、CRC 字段、应答字段、帧结束和间断）在结构上与标准数据帧相同。

B.8 远程帧

通常，数据发送是由数据源节点自主完成的（例如，传感器发送数据帧）。但也可能发生宿节点向源节点请求发送数据的情况。要做到这一点，宿节点需要发送一个标识符与所需数据帧的标识符相匹配的“远程帧”。随后相应的数据源节点会发送一个数据帧作为对该远程请求的响应。

如图 B-4 所示，远程帧与数据帧有两点不同。第一，远程帧的 RTR 位为隐性状态；第二，远程帧没有数据字段。带有相同标识符的数据帧和远程帧同时发送的情况是很少出现的，这种情况下数据帧将赢得仲裁，这是因为其标识符之后的 RTR 位为显性。这样可使发送远程帧的节点立即收到所需数据。

B.9 错误帧

错误帧是由检测到总线错误的任何节点产生的。如图 B-5 所示，错误帧包含 2 个字段，即错误标志字段和紧随其后的错误定界符字段。错误定界符由 8 个隐性位组成，可以让总线节点在错误发生后立即重新启动总线通信。错误标志字段有两种形式。其具体形式取决于检测到错误的节点的错误状态。

当错误主动节点检测到一个总线错误时，这个节点将通过产生一个主动错误标志，中断当前的报文发送。主动错误标志由 6 个连续的显性位构成。这种位序列有效地打破了位填充规则。所有其他站点在识别到由此产生的位填充错误后，反过来也会产生错误帧，称为错误回送标志。错误标志字段因此包含 6 到 12 个连续显性位（由 1 个或多个节点产生）。错误帧以错误定界符字段结束。在错误帧发送完毕后，总线活动恢复正常状态，被中断的节点会尝试重新发送被中止的报文。

当错误被动节点检测到一个总线错误时，该节点将发送一个错误被动标志，后面仍然紧随错误定界符字段。错误被动标志由 6 个连续隐性位组成，因此错误被动节点的错误帧由 14 个隐性位组成。由此可知，除非总线错误被正在发送报文的总线主节点检测到，否则错误被动节点发送的错误帧将不会影响网络中任何其他节点。如果总线主节点产生了一个错误被动标志，那么由于位填充规则被打破，将导致其他节点产生错误帧。错误帧发送完毕后，错误被动节点必须等待总线上出现 6 个连续隐性位后，才能尝试重新参与总线通信。

B.10 帧间间隔

帧间间隔将前一个帧（无论何种类型）与其后的数据帧或远程帧分隔开来。帧间间隔至少由 3 个隐性位构成，也称为间断。间断使节点在开始发送 / 接收下一个报文帧之前有时间进行内部处理。在间断之后，CAN 总线将保持隐性状态（总线空闲），直至开始发送下一个帧。

标准数据帧



图 B-4: 远程数据帧

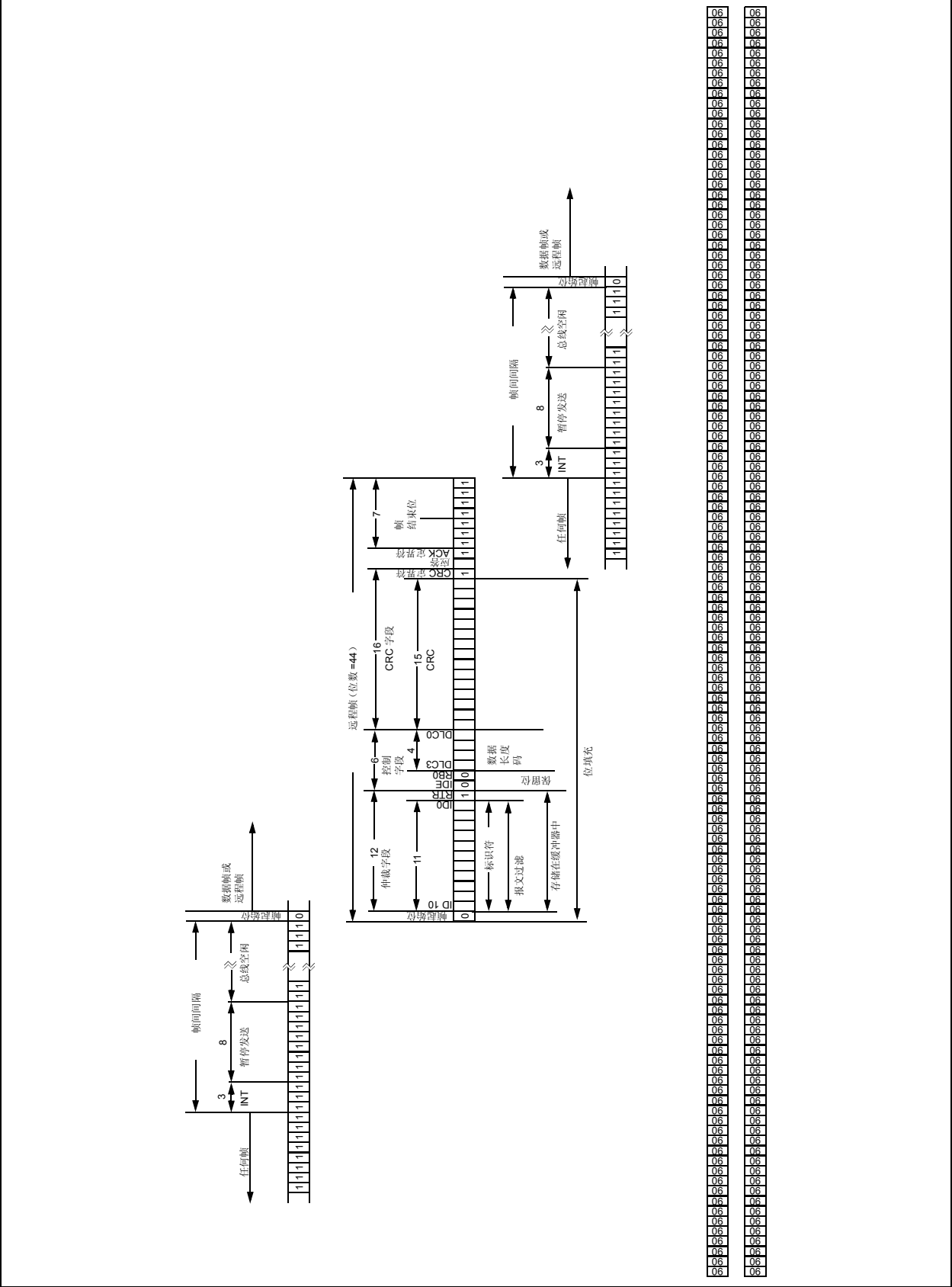
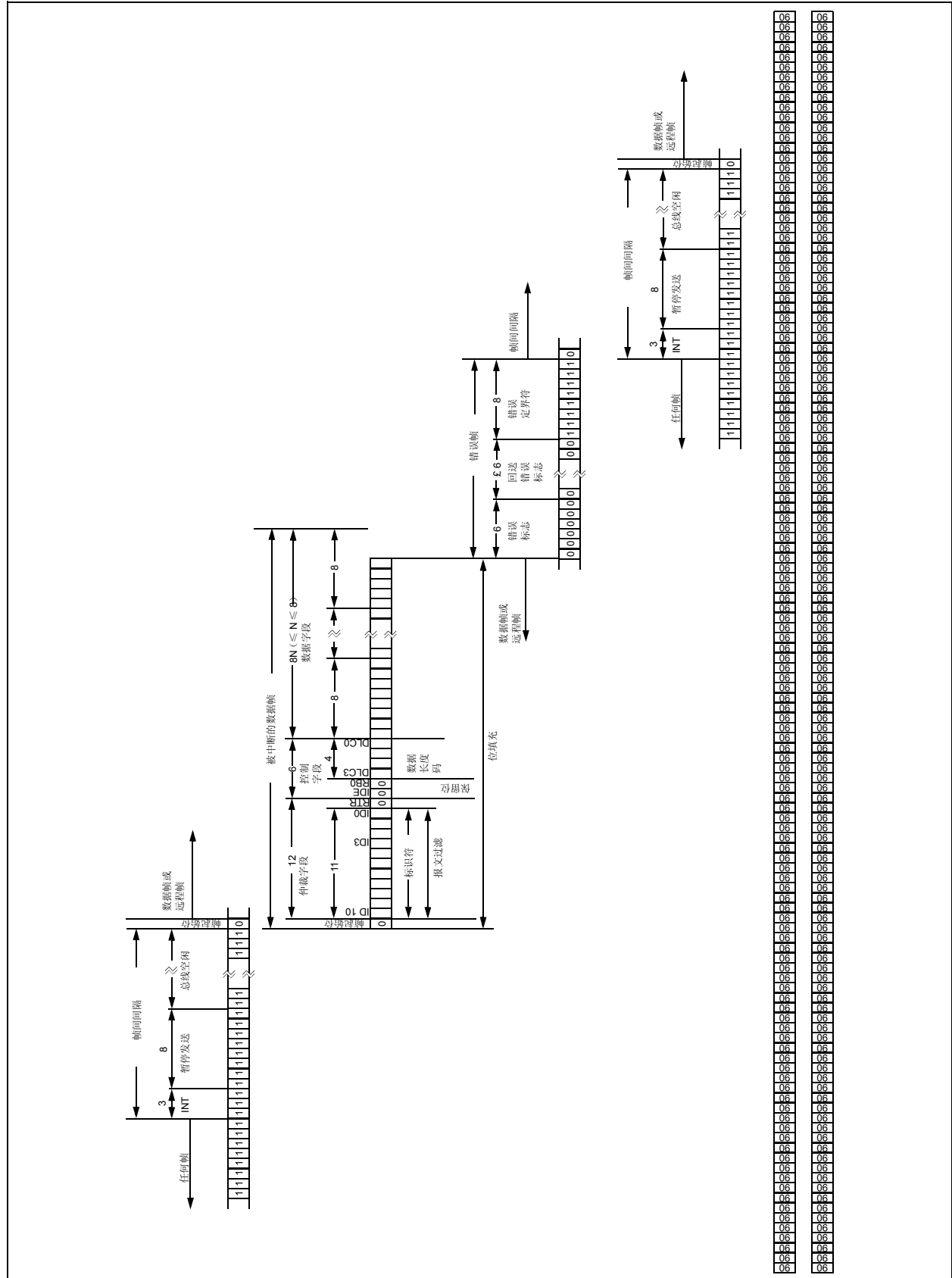


图 B-5: 错误帧



B.11 参考文档

标题	文档
Road Vehicles; Interchange of Digital Information, Controller Area Network Bosch CAN Specification Version 2.0	ISO11898

附录 C： 编解码器协议概述

本附录概括了 Inter-IC Sound (I²S) 和 AC-Link 兼容模式接口的音频编码器 / 解码器 (编解码器, Codec) 协议。很多倾向于在音频应用中使用的编解码器支持 8 kHz 到 48 kHz 的采样率, 通常使用前面提到的接口协议中的一种。数据转换器接口 (DCI) 模块自动处理与这些编解码器有关的接口时序。在 DCI 发送和 / 或接收被请求的数据量前, 无需 CPU 开销。在两次 CPU 中断之间最多可以传输 4 个数据字。

C.1 I²S 协议说明

Inter-IC Sound (I²S) 是一种简单的三线总线接口, 用于在以下设备之间传输数字音频数据:

- DSP 处理器
- A/D 和 D/A 转换器
- 数字输入 / 输出接口

本附录信息旨在对 Philips, Inc 发布的 I²S Protocol Specification[®] 作一个补充。

I²S 总线是时分复用的并可传输两个通道的数据。这两个数据通道通常是数字音频流的左右声道。

I²S 总线具有以下连接引脚:

- SCK: I²S 串行时钟线
- SDx: I²S 串行数据线 (输入或输出)
- WS: I²S 字选择线

图 C-2 所示为数据传输的时序图。串行数据在 I²S 总线上是首先发送 MSb 的 2 的补码格式发送的。因此协议允许发送和接收不同数据字长度, 所以必须先发送 MSb。如果接收器收到了比它能接受的数据字更多的位, 则将 LSb 忽略。如果接收器收到了比原始字长更少的位, 则它必须在内部将余下的 LSb 置为 0。

WS 线指出正在发送的数据通道。使用以下标准:

- WS = 0: 通道 1 或左声道
- WS = 1: 通道 2 或右声道

WS 线由接收器在 SCK 的上升沿采样, 下一个数据字的 MSb 在 WS 改变后的下一个 SCK 周期发送。WS 改变后一个周期的延时可以让接收器有时间存储以前发送给它的字, 并准备好接收下一个字。发送器发送的串行数据在 SCK 的下降沿放到总线上, 并在 SCK 的上升沿由接收器锁存。

任何器件都可以在 I²S 系统中充当系统主机。系统主机产生 SCK 和 WS 信号。通常, 由发送器充当系统主机, 但是接收器或其他器件也可以作为主机。图 C-1 显示了可能的 I²S 总线配置。虽然在图 C-1 中没有特别说明, 但两个相连的器件可以有数据发送和数据接收两种连接方式。

图 C-1: I²S 总线连接

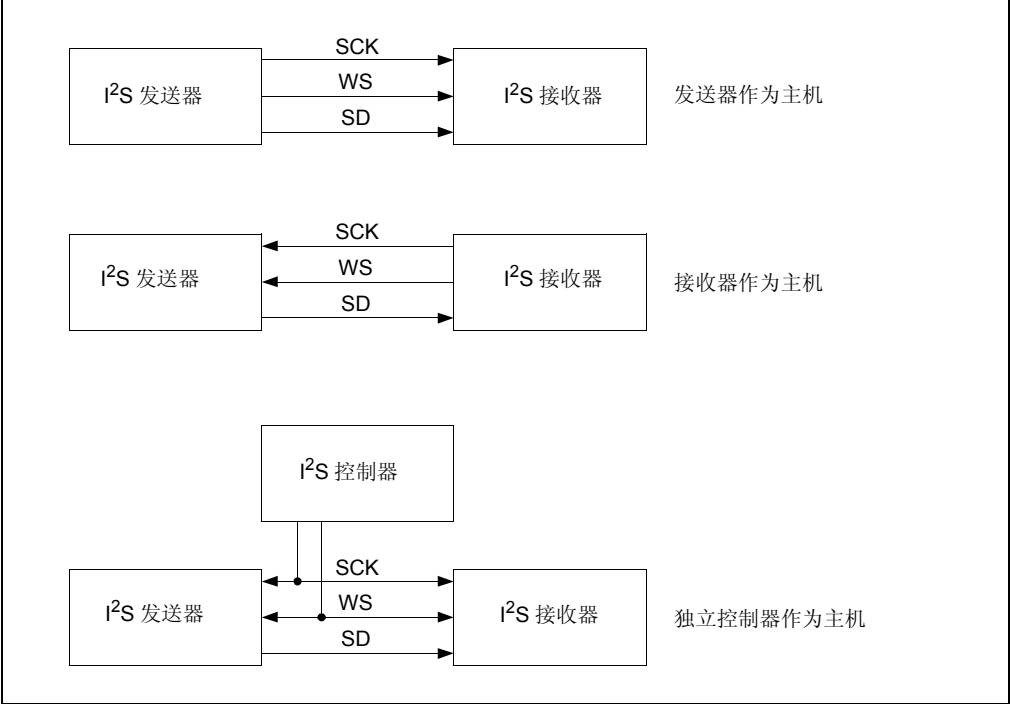
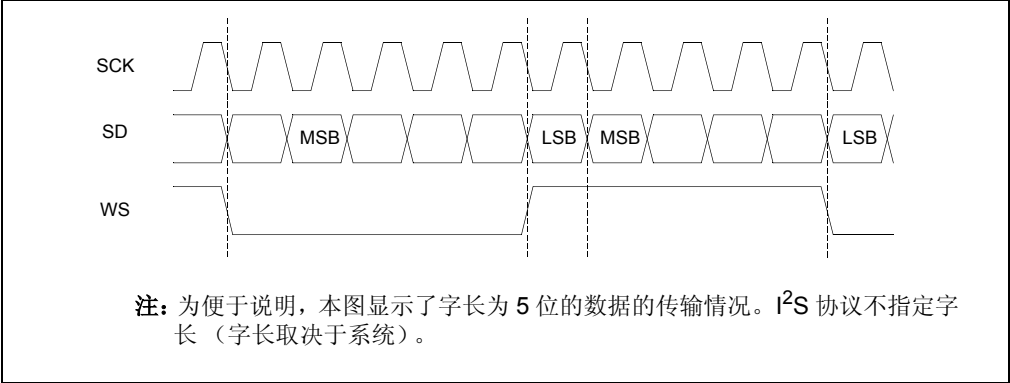


图 C-2: I²S 接口时序图



C.2 AC' 97 协议

音频编解码器' 97（Audio Codec '97，AC '97）规范定义了用于 PC 平台的音频编解码器的标准架构和数字接口协议。符合 AC '97 的编解码器的数字接口协议称为 AC-Link，它是本文讨论的重点。本文不说明 AC '97 控制器设备的具体要求和功能。

本附录信息旨在对 Intel 公司发布的 AC '97 组件规范文档作一个补充。

C.3 AC-Link 信号说明

所有 AC-Link 信号都是由 AC '97 主时钟源产生的。建议使用连接到 AC '97 编解码器的时钟源为 24.576 MHz 晶振，以使时钟抖动减至最小。24.576 MHz 时钟也可以由 AC'97 控制器或外部源提供。

所有的 AC-Link 信号名称都涉及到 AC '97 控制器，而非 AC '97 编解码器。控制器是产生 SYNC 信号以启动数据传输的器件。下面各节对每个信号进行了说明。

C.3.1 位时钟 (BIT_CLK)

系统中的主 AC '97 编解码器中提供一个 12.288 MHz BIT_CLK 信号。BIT_CLK 信号是 AC '97 控制器和系统中最多三个从动 AC '97 编解码器设备的输入信号。所有 AC-Link 数据在 BIT_CLK 的上升沿传送，并且由接收设备在 BIT_CLK 的下降沿采样。

C.3.2 串行数据输出 (SDO)

SDO 是发送到 AC '97 编解码器的时分复用数据流。

C.3.3 串行数据输入 (SDI)

SDI 是 AC '97 编解码器发出的时分复用数据流。

C.3.4 SYNC

SYNC 是由 AC '97 控制器发送到 AC '97 编解码器的 48 kHz 固定速率的采样同步信号。SYNC 信号是通过将 BIT_CLK 信号进行 256 分频得到的。SYNC 信号的电平保持 16 个 BIT_CLK 周期然后变为低电平并保持 240 个 BIT_CLK 周期。SYNC 信号只在 BIT_CLK 的上升沿改变，并且其周期确定了一个音频数据帧的边界。

C.3.5 复位

$\overline{\text{RESET}}$ 信号是系统中每个 AC '97 编解码器的输入信号，它会复位编解码器硬件。

C.4 AC-Link 协议

C.4.1 AC-Link 串行接口协议

AC-Link 串行数据流采样采用 256 位数据帧的时分复用 (TDM) 机制。每个数据帧都分成 13 个时隙，编号为 Slot #0 到 Slot #12。Slot #0 是一个特殊时隙，包含 16 个位。其余 12 个时隙都是 20 位宽的。

图 C-4 显示了一个 AC-Link 帧示例。该帧在 SYNC 信号的上升沿（与 BIT_CLK 的上升沿重合）开始传输。AC '97 编解码器在紧随在该上升沿之后的 BIT_CLK 下降沿采样 SYNC 信号。此下降沿标记编解码器和控制器发现新帧起始位的时间。在 BIT_CLK 的下一个上升沿，编解码器产生 SDATA_IN 的 MSb 并产生 SDATA_OUT 的第一个边沿。此过程可以确保接收和发出的数据流的数据电平跳变点和随后的采样点在时间上是对齐的。

Slot #0、Slot #1 和 Slot #2 在 AC-Link 协议中对状态和控制有特殊用途。其余时隙都分配给某些特定类型的数字音频数据。Slot #3 到 Slot #12 的数据分配取决于所选择的 AC '97 编解码器，因此这里只能简要说明时隙用法。如需更多时隙用法的信息，请参见 AC '97 组件规范。

C.4.2 Slot #0, 标记帧

Slot #0 通常称为“标记帧”(Tag Frame)。标记帧对 AC-Link 协议的每个数据时隙都有一个对应的位位置。这些位用于指定帧中的哪些时隙可供控制器使用。Slot #0 特定位位置中的“1”表示当前音频帧中对应的时隙已经分配给一个数据流并且此时隙包含有效数据。如果某个间隙标记为无效,则应由数据源(对于输入流为 AC '97 编解码器、对于输出流为 AC '97 控制器)在该时隙的有效时间内用 0 填充所有位置。

在标记帧中也有特殊位。对于 SDATA_OUT, 标记帧的 MSb 是“帧有效”(Frame Valid)状态位。帧有效位是编解码器的全局指示符,表明在帧里至少有一个时隙包含有效数据。如果整个帧都标记为无效,编解码器会忽略标志帧后续的所有时隙。此功能用于实现非 48 kHz 的采样率。

SDATA_OUT 标记帧的两个 LSb 指出编解码器的地址。系统中最多可连接四个 AC '97 编解码器。如果系统只使用了一个编解码器,这些位保持为 0。

SDATA_IN 的 MSb 是“编解码器就绪”(Codec Ready)状态位。如果此位为 0,则表示编解码器已关闭和/或尚未就绪。如果“编解码器就绪”位置位,则应由控制器查询编解码器中的状态寄存器,以确定哪个编解码器可使用。

C.4.3 Slot #1 (命令地址) 和 Slot #2 (命令数据)

Slot #1 和 Slot #2 在 AC-Link 协议中也有特殊用途。当读取或写入 AC '97 编解码器控制寄存器时,这些时隙用于放置地址和数据值。这些时隙必须在 Slot #0 中标记为有效才能读写控制寄存器。AC '97 组件规范允许在编解码器中存在 64 个 16 位控制寄存器。在 AC-Link 协议中提供了 7 个地址位,但是只使用偶数地址位。奇数地址位的值是保留的。

SDATA_OUT 线的 Slot #1 和 Slot #2 分别称为命令地址和命令数据。SDATA_OUT 线上的命令地址间隙用于指定编解码器寄存器地址,并指定寄存器访问是读取还是写入。SDATA_OUT 上的命令数据时隙包含将要写入编解码器的某个控制寄存器的 16 位值。如果对编解码器寄存器执行了读操作,命令数据位将设置为零。

SDATA_IN 线的 Slot #1 和 Slot #2 分别称为状态地址和状态数据时隙。状态地址时隙回送先前发送到编解码器的寄存器地址。如果该值为 0,表示先前向编解码器发送的是无效地址。

状态地址时隙还有 10 个时隙请求位。时隙请求位可以由编解码器控制用于具有可变采样率的应用。

状态数据时隙返回从编解码器控制/状态寄存器读取的 16 位数据。

C.4.4 Slot #3 (PCM 左声道)

SDATA_OUT 信号中的 Slot #3 用于合成的数据音频左声道回放流。对于声卡应用,该回放流通常是复合的 .WAV 音频和 MIDI 合成器输出相结合而成的。

SDATA_IN 信号中的 Slot #3 是从 AC '97 编解码器输入混音器获取的左声道录音数据。

C.4.5 Slot #4 (PCM 右声道)

SDATA_OUT 信号中的 Slot #4 用于组合的数字音频右声道回放流。对于声卡应用,该回放流通常是复合的 .WAV 音频和 MIDI 合成器输出。

SDATA_IN 信号中的 Slot #4 是从 AC '97 编解码器输入混音器获取的右声道录音数据。

C.4.6 Slot #5 (调制解调器线 1)

SDATA_OUT 信号中的 Slot #5 用于调制解调器 DAC 数据。兼容调制解调器的 AC '97 编解码器的缺省精度为 16 位。与所有时隙一样，此时隙中未使用的位都置为 0。

SDATA_IN 信号中的 Slot #5 用于调制解调器 ADC 数据。

C.4.7 Slot #6

SDATA_OUT 信号中的 Slot #6 用于 4 声道或 6 声道声音配置中的 PCM 中间声道 DAC 数据。

SDATA_IN 信号中的 Slot #6 专门用于专用麦克风录音数据。此时隙中的数据允许在应用中使用话筒回声消除算法用于话筒应用中。

C.4.8 Slot #7

SDATA_OUT 信号中的 Slot #7 用于 4 声道或 6 声道声音配置中的 PCM 左声道 DAC 数据。

SDATA_IN 信号中的 Slot #7 在 AC '97 组件规范中保留供未来使用。

C.4.9 Slot #8

SDATA_OUT 信号中的 Slot #8 用于 4 声道或 6 声道声音配置中的 PCM 右声道 DAC 数据。

SDATA_IN 信号中的 Slot #8 在 AC '97 组件规范中保留供未来使用。

C.4.10 Slot #9

SDATA_OUT 信号中的 Slot #9 用于 6 声道声音配置中的 PCM LFE DAC 数据。

SDATA_IN 信号中的 Slot #9 在 AC '97 组件规范中保留供未来使用。

C.4.11 Slot #10 (调制解调器线 2)

Slot #10 用于调制解调器兼容设备中的调制解调器线 2 的 ADC 和 DAC 数据。

C.4.12 Slot #11 (调制解调器听筒)

Slot #11 用于调制解调器兼容设备中的调制解调器听筒的 ADC 和 DAC 数据。

C.4.13 Slot #12 (GPIO 控制 / 状态)

Slot #12 中的位用于读取和写入 AC '97 编解码器中的 GPIO 引脚。GPIO 引脚用于调制解调器兼容设备上的调制解调器控制功能。

图 C-3: AC-Link 信号连接

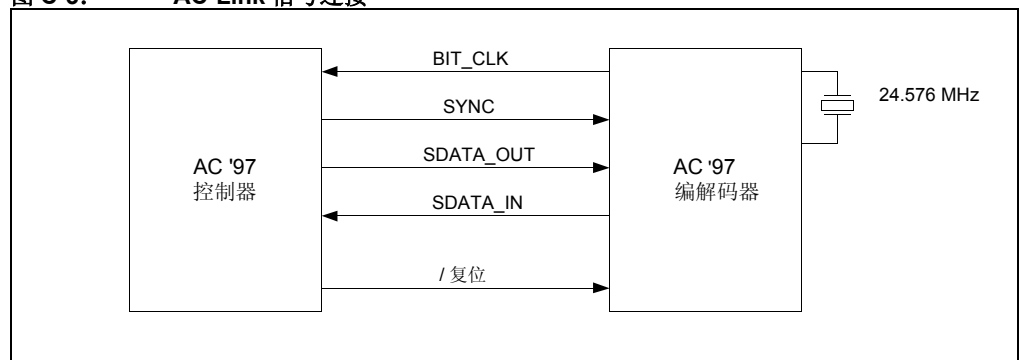


图 C-4: AC-Link 数据帧

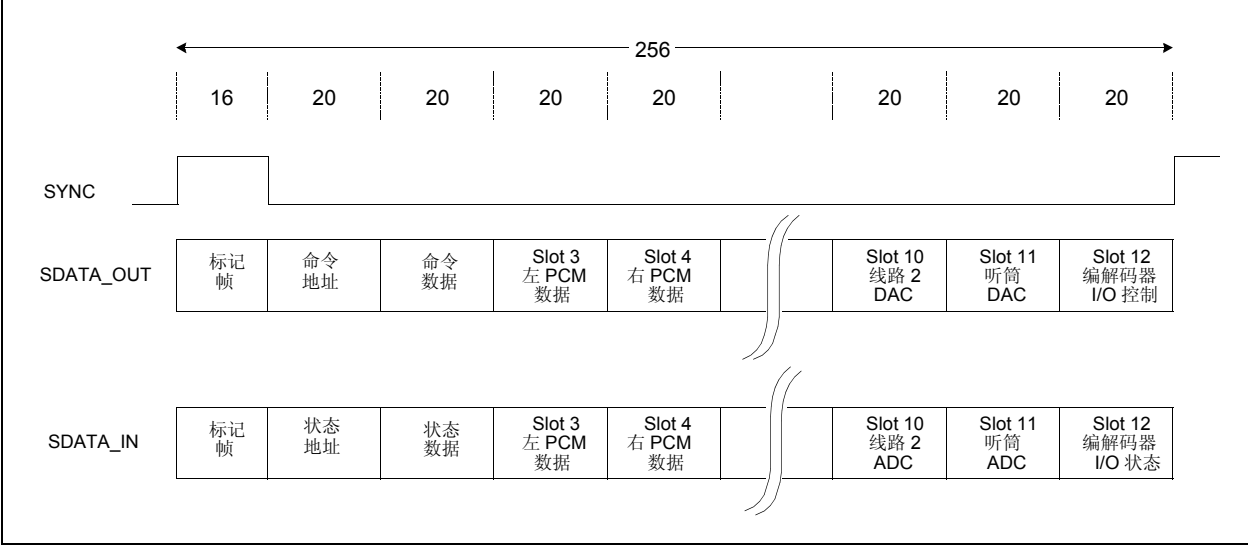


图 C-5: SLOT#0、SLOT#1 和 SLOT#2 在 SDATA_IN 中的位置

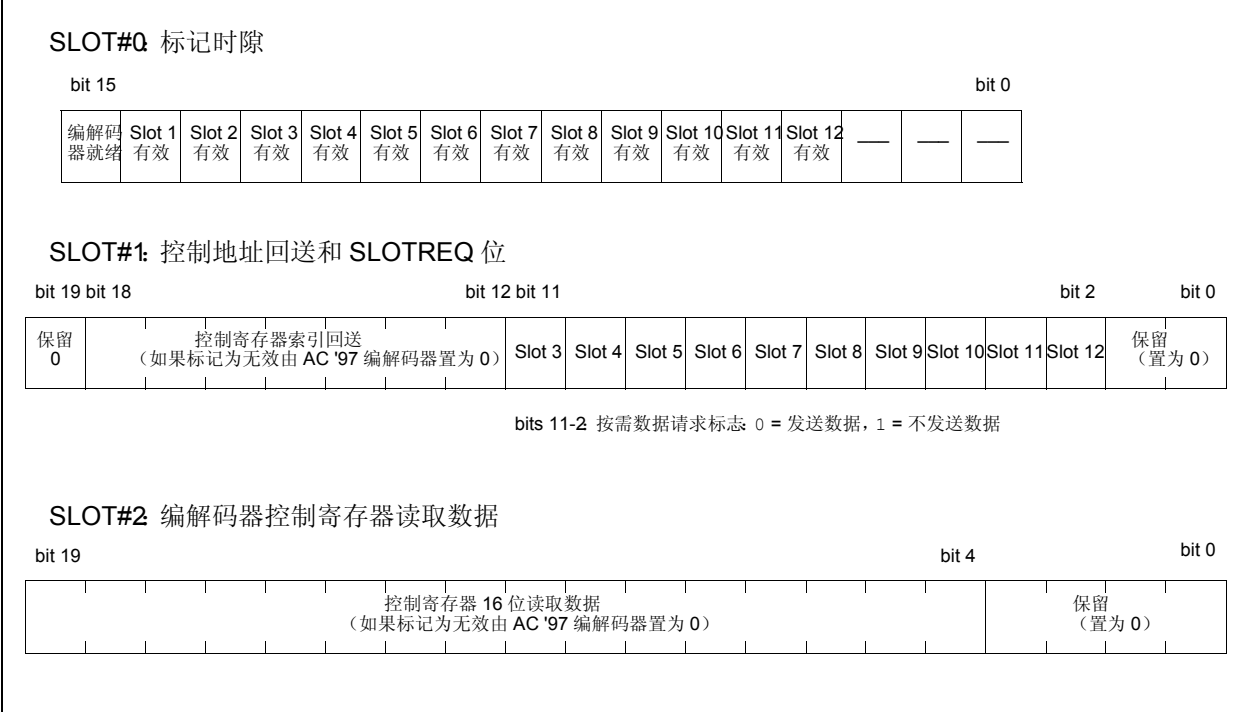
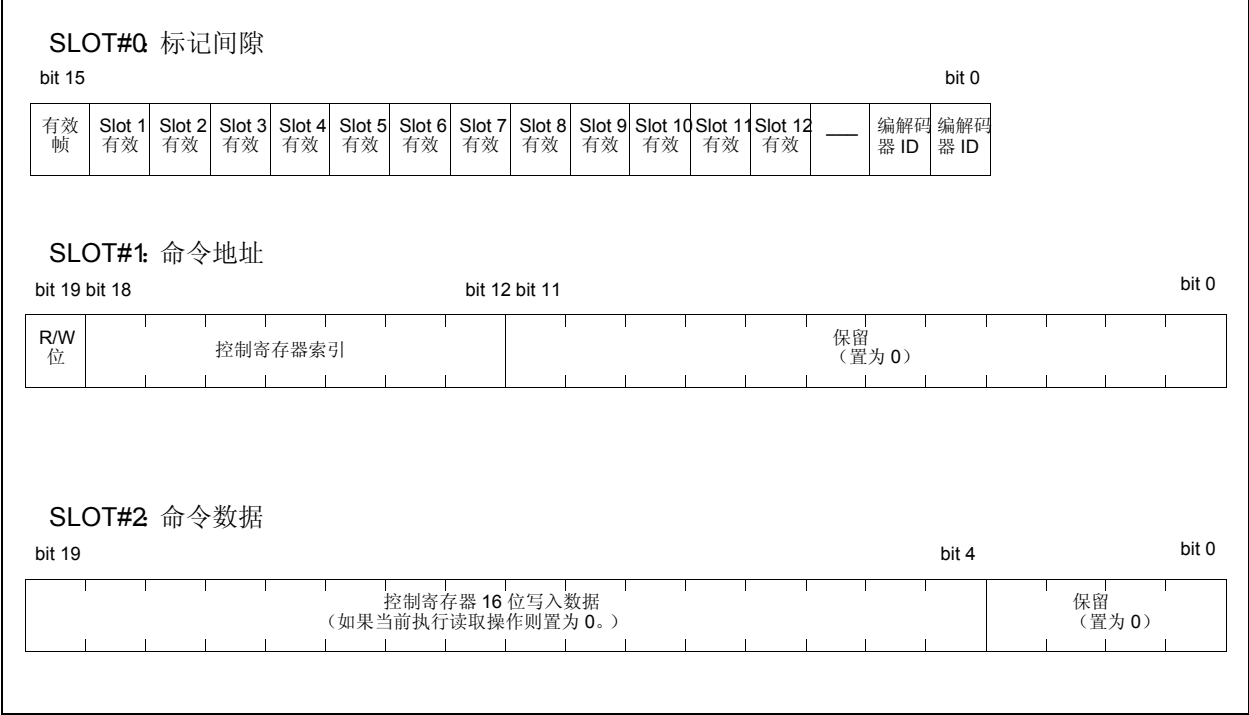


图 C-6: SLOT#0、SLOT#1 和 SLOT#2 在 SDATA_OUT 中的位置



注:

数字

10 位 A/D	
ADCHS	17-4
ADPCFG	17-4
CPU 休眠模式下的工作	17-55
在 CPU 空闲模式下的工作	17-55
10 位地址模式	21-35
12 位 A/D	
ADCHS	18-4
ADPCFG	18-4
不使用 RC A/D 时钟的 CPU 休眠模式	18-33
CPU 空闲模式下的工作	18-33
使用 RC A/D 时钟的 CPU 休眠模式	18-33
16 位向上 / 向下位置计数器	16-12
32 位定时器配置	12-16

A

A/D	
ADCON0 寄存器	17-4, 18-4
边沿检测模式	13-8
采样要求	17-45, 18-26
读 A/D 结果缓冲器	17-46
复位的影响	16-19, 17-55
复位的影响 (12 位)	18-33
精度 / 误差	17-47, 18-28
配置模拟端口引脚	17-14, 18-12
如何开始采样	17-17
如何停止采样和开始转换	17-18
使能模块	17-16
使能模块 (12 位)	18-14
传递函数	17-47, 18-28
传递函数 (12 位)	18-28
A/D 采样要求 (12 位)	18-26
A/D 精度 / 误差 (12 位)	18-28
A/D 模块配置	17-13
A/D 模块配置 (12 位)	18-11
A/D 特殊事件触发信号	12-22
A/D 转换器外部转换请求	6-10
A/D 转换速度	17-49
A/D 转换术语和转换过程	17-11
A/D 转换术语和转换过程 (12 位)	18-10
ACK	21-38
ACKSTAT	21-17
A 类型定时器	12-3

B

B 类型定时器	12-4
饱和度和溢出模式	2-23
备用向量列表	6-2
编程器模型	2-3, 2-4
寄存器描述	2-4
编解码器接口基础与术语	22-8
表指令操作	5-2
波特率	
表	19-9
发生器 (BRG)	19-8
捕获缓冲操作	13-8
不可屏蔽陷阱	6-6
地址错误	6-6
堆栈错误	6-6
算术错误	6-6
振荡器故障	6-6
不同的定时器	12-3

C

CAN	
缓冲器接收和溢出真值表	23-44
CAN 库	25-9
C 类型定时器	12-5
CN	
电平变化通知引脚	11-7
控制寄存器	11-8
配置和操作	11-8
休眠和空闲模式下的工作	11-8
CPU	
版本历史	2-41
寄存器映射	2-38
相关应用笔记	2-40
CPU 寄存器描述	2-11
CPU 优先级状态	6-5
采样 / 转换工作的控制 (12 位)	18-19
采样 / 转换过程的说明	17-16
采样模拟输入的选择 (12 位)	18-12
乘法器	2-20
乘法指令	2-22
程序存储器	
表地址的生成	4-6
表指令综述	4-5
低位字访问	4-6
地址映射	4-2
高位字访问	4-7
计数器	4-4
来自数据空间的程序空间可视性	4-8
数据存储	4-7
相关应用笔记	4-12
除法支持	2-27
初始化	17-48
初始化 (12 位)	18-29
从程序存储器	
存取数据	4-4
从休眠和空闲模式唤醒	6-10
从中断返回	6-13
D	
DCI	
工作原理	22-10
缓冲器控制单元	22-13
控制寄存器说明	22-2
使用模块	22-17
dsPIC30F 硬件开发板	25-11
dsPIC 语言套件	25-3
DSP 滤波器设计软件实用程序	25-8
DSP 算法库	25-7
DSP 引擎	2-18
DSP 引擎模式选择	2-26
DSP 引擎陷阱事件	2-26
代码示例	
16 位选通时间累加模式的初始化代码	12-13
32 位选通时间累加模式的初始化代码	12-20
8 位发送 / 接收 (UART1)	19-20
9 位发送 / 接收 (UART1), 地址检测使能	19-20
比较模式电平交替翻转设置和中断服务程序	14-8
比较模式交替模式引脚状态设置	14-8
单输出脉冲设置和中断服务程序	14-12
单行编程	5-14
读 32 位定时器	12-21
连续输出脉冲设置和中断服务程序	14-16
PWM 模式脉冲设置和中断服务程序	14-22
使用外部时钟输入的 16 位同步计数器模式的初始化代码	12-10

使用外部时钟输入的 16 位异步计数器模式的初始化代码	12-11
使用外部时钟输入的 32 位同步计数器模式的初始化代码	12-19
使用系统时钟的 16 位定时器的初始化代码	12-9
使用指令周期作为输入时钟的 32 位定时器的初始化代码	12-18
写配置寄存器	5-15
预分频捕捉	13-7
低功耗 32 kHz 晶体振荡器	7-31
低功耗 32 kHz 晶振输入	12-15
低功耗模式	10-2
低功耗指令与中断同时发生	10-5
第三方 C 编译器	25-6
第三方硬件 / 软件工具和应用程序库	25-6
地址发生器单元和 DSP 类指令	3-6
地址寄存器相关性	2-35
电机控制开发系统	25-13
定时器	
版本历史	12-26
设计技巧	12-24
相关应用笔记	12-25
定时器的的工作模式	12-9
32 位定时器	12-18
定时器模式	12-9
使用外部时钟输入的 A 类型定时器异步计数器模式	12-11
使用外部时钟输入的同步计数器	12-10
定时器的外部时钟输入引脚作为中断引脚	12-22
定时器工作模式	
使用快速外部时钟源	12-12
选通时间累加	12-12
定时器模块	
相关的特殊功能寄存器	12-23
定时器选择	13-4
定时器预分频器	12-14
定时器中断	12-14
读和写 16 位定时器模块寄存器	12-15
读和写 32 位定时器	12-21
独立 PWM 输出模式	15-30
读取 A/D 转换结果缓冲器 (12 位)	18-27
端口 (I/O 端口) 寄存器	11-3
多主机模式	21-29
F	
FSCM 和器件复位	8-12
复位	
非法操作码	8-9
设计技巧	8-17
未初始化的 W 寄存器	8-9
陷阱冲突	8-9
复位顺序	6-2
G	
公式	
递减缓冲区的模起始地址	3-9
递增缓冲器的模结束地址	3-9
计算 PWM 周期	14-19
计算最大 PWM 精度	14-20
WDT 超时周期	10-7
工作寄存器阵列	2-6
W0 和数据寄存器指令	2-6
W 寄存器存储器映射	2-6
W 寄存器和字节模式指令	2-6
工作模式	14-4
比较模式电平交替翻转输出	14-7
比较模式输出驱动为低电平	14-6
比较模式输出驱动为高电平	14-5
单比较匹配	14-4
双比较匹配	14-9
双比较, 产生连续输出脉冲	
特殊情况 (列表)	14-17
双比较, 单输出脉冲	14-9
特殊情况 (列表)	14-13
双比较, 连续输出脉冲	14-14
故障保险时钟监视器 (FSCM)	7-32
和缓慢振荡器起振	7-33
和 WDT	7-33
延时	7-32
H	
互补 PWM 输出模式	15-26
I	
I/O 端口	
版本历史	11-12
相关应用笔记	11-11
I/O 端口控制寄存器	11-3
I/O 引脚控制	12-22, 13-10, 14-23, 16-18
I/O 与多个外设复用	11-4
I ² C	
报文位发送期间的总线冲突	21-31
产生启动总线事件	21-16
产生停止总线事件	21-22
产生重复启动总线事件	21-23
发送数据到从器件	21-17
发送数据到主器件	21-44
检测地址	21-32
检测启动和停止条件	21-32
检测总线冲突和重新发送报文	21-30
建立完整的主器件报文	21-24
接收来自从器件的数据	21-19
接收来自主器件的数据	21-37
启动	26-3
启动和终止数据传输	26-3
启动条件期间的总线冲突	21-31
使能操作	21-13
使能 I/O	21-13
停止	26-3
停止条件期间的总线冲突	21-31
应答产生	21-21
在 PWSAV 指令执行期间的模块操作	21-49
中断	21-13
重复启动条件期间的总线冲突	21-31
主器件报文协议状态	21-24
总线连接注意事项	21-47
总线仲裁与总线冲突	21-30
作为从器件通信	21-32
作为主器件在多主机环境下通信	21-29
I ² C 模块	
支持的从地址	21-33
I ² C 模块	
10 位地址模式	21-35
多主机模式	21-29
I2COV	21-19
IWCOL	21-18, 21-19, 21-21, 21-22
J	
寄存器	
10 位 A/D 转换器特殊功能	17-56
12 位 A/D 转换器特殊功能	18-34
6 输出 PWM 模块	15-41

8 输出 PWM 模块	15-40	IEC0 (中断允许控制寄存器 0)	6-24
A/D 输入扫描选择寄存器 (ADCSSL)	17-10	IEC1 (中断允许控制寄存器 1)	6-26
A/D 输入扫描选择 (ADCSSL - 12 位)	18-9	IEC2 (中断允许控制寄存器 2)	6-28, 6-29
ADCHS A/D 输入选择	17-9	IFS0 (中断标志状态 0)	6-18
ADCHS A/D 输入选择 (12 位)	18-8	IFS1 (中断标志状态 1)	6-20
ADCHS(A/D 输入选择) 寄存器	17-9, 18-8	IFS2 (中断标志状态 2)	6-22, 6-23
ADCON1 A/D 控制器 1	17-5, 17-6	INTCON1 (中断控制 1)	6-16
ADCON1 A/D 控制 1 (12 位)	18-5	INTCON2 (中断控制 2)	6-17
ADCON1 (A/D 控制) 寄存器 1	17-5, 18-5	IPC0 (中断优先级控制寄存器 0)	6-30
ADCON2 A/D 控制器 2	17-7	IPC10 (中断优先级控制寄存器 10)	6-40
ADCON2 A/D 控制 2 (12 位)	18-6	IPC11 (中断优先级控制寄存器 11)	6-41
ADCON2 (A/D 控制) 寄存器 2	17-7, 18-6	IPC1 (中断优先级控制寄存器 1)	6-31
ADCON3 A/D 控制器 3	17-8	IPC2 (中断优先级控制寄存器 2)	6-32
ADCON3 A/D 控制 3 (12 位)	18-7	IPC3 (中断优先级控制寄存器 3)	6-33
ADCON3 (A/D 控制) 寄存器 3	17-8, 18-7	IPC4 (中断优先级控制寄存器 4)	6-34
ADPCFG A/D 端口配置	17-10	IPC5 (中断优先级控制寄存器 5)	6-35
ADPCFG A/D 端口配置 (12 位)	18-9	IPC6 (中断优先级控制寄存器 6)	6-36
ADPCFG (A/D 端口配置) 寄存器	17-10, 18-9	IPC7 (中断优先级控制寄存器 7)	6-37
CiCFG1 (波特率配置寄存器)	23-16	IPC8 (中断优先级控制寄存器 8)	6-38
CiCFG2 (波特率配置寄存器 2)	23-17	IPC9 (中断优先级控制寄存器 9)	6-39
CiCTRL (CAN 模块控制和状态寄存器)	23-3	控制和状态	16-4
CiEC (发送 / 接收错误计数)	23-18	控制寄存器	15-4
CiINTE (中断使能寄存器)	23-19	MODCON (模和位反转寻址控制)	3-19
CiINTF (中断标志寄存器)	23-20	NVMADRU (非易失性存储器地址)	5-9
CiRX0CON (接收缓冲器 0 状态和控制寄存器)	23-8	NVMADR (非易失性存储器地址)	5-8
CiRX1CON (接收缓冲器 1 状态和控制寄存器)	23-9	NVMCON (非易失性存储器控制)	5-7
CiRX1FnSID (验收滤波器 n 标准标识符)	23-12	NVMKEY (非易失性存储器密钥)	5-9
CiRXFnEIDH (验收滤波器 n 扩展标识符高位)	23-12	OCxCON (输出比较 x 控制)	14-3
CiRXFnEIDL (验收滤波器 n 扩展标识符低位)	23-13	OVDCON 改写控制	15-13
CiRXMnEIDH (验收滤波器屏蔽 n 扩展标识符高位)	23-14	PDC1 PWM 占空比 1	15-13
CiRXMnEIDL (验收滤波器屏蔽 n 扩展标识符低位)	23-15	PDC2 PWM 占空比 2	15-14
CiRXMnSID (验收滤波器屏蔽 n 标准标识符)	23-14	PDC3 PWM 占空比 3	15-14
CiRXnBm (接收缓冲器 n 数据字段字 m)	23-11	PDC4 PWM 占空比 4	15-15
CiRXnDLC (接收缓冲器 n 数据长度控制)	23-11	PTCON PWM 时基控制	15-5
CiRXnEID (发送缓冲器 n 扩展标识符)	23-10	PTMR PWM 时基	15-6
CiRXnSID (接收缓冲器 n 标准标识符)	23-10	PTPER PWM 时基周期	15-6
CiTnBm (发送缓冲器 n 数据字段字 m)	23-7	PWMCON1 PWM 控制 1	15-7
CiTnCON (发送缓冲器状态和控制寄存器)	23-5	PWMCON2 PWM 控制 2	15-8
CiTnDLC (发送缓冲器 n 数据长度控制)	23-7	QEICON QEI 控制	16-5, 16-6
CiTnEID (发送缓冲器 n 扩展标识符)	23-6	QEI 特殊功能	16-20
CiTnSID (发送缓冲器 n 标准标识符)	23-6	RCON (复位控制)	8-3, 9-4
CNEN1 (输入变化通知中断使能 1)	11-9	RSCON	22-7
CNEN2 (输入变化通知中断使能 2)	11-9	SEVTCMP 特殊事件比较	15-7
CNPU1 (输入变化通知上拉使能 1)	11-10	SR (CPU 中的状态)	6-15
CNPU2 (输入变化通知上拉使能 2)	11-10	SR (CPU 状态)	2-12
CORCON (内核控制)	2-14, 6-15	TSCON	22-7
DCICON1	22-3	TxCON (A 类型时基的定时器控制)	12-6
DCICON2	22-4	TxCON (B 类型时基的定时器控制)	12-7
DCICON3	22-5	TxCON (C 类型时基的定时器控制)	12-8
DCISTAT	22-6	UxBRG (UARTx 波特率)	19-7
DFLTCON 数字滤波器控制	16-7, 16-8	UxMODE (UARTx 模式)	19-3
DTCON1 死区时间控制 1	15-9	UxRXREG (UARTx 接收)	19-6
DTCON2 死区时间控制 2	15-10	UxSTA (UARTx 状态和控制)	19-4
FBORPOR BOR 和 POR 器件配置	15-15	UxTXREG (UARTx 发送寄存器)	19-6
FBORPOR (BOR 和 POR 配置寄存器)	24-4	XBREV (X 写 AGU 位反转寻址控制)	3-22
FGS (通用代码段配置寄存器)	24-5	XMODEND (X AGU 模寻址结束)	3-20
FLTACON 故障 A 控制	15-11	XMODSRT (X AGU 模寻址起始)	3-20
FLTBCON 故障 B 控制	15-12	YMODEND (Y AGU 模寻址结束)	3-21
FOSC (振荡器配置寄存器)	7-9, 7-11	YMODSRT (Y AGU 模寻址起始)	3-21
FWDT (看门狗定时器配置寄存器)	24-3	简单捕捉事件	13-4
I2CCON (I ² C 控制) 寄存器	21-9, 21-10	简介	
I2CSTAT (I ² C 状态) 寄存器	21-11	版本历史	1-7
ICxCON (输入捕捉 x 控制)	13-3	节能模式下的 QEI 工作	16-19
		节能状态下的定时器工作	12-21
		节能状态下的输出比较工作	14-23
		空闲模式	14-23

休眠模式	14-23
禁止中断指令	6-8
晶体振荡器 / 陶瓷谐振器	7-22
晶振和 PLL 时钟源的 FSCM 延时	8-12

K

看门狗超时溢出复位 (WDTR)	8-7
看门狗定时器	10-6
复位	10-8
工作	10-7
软件控制的 WDT	10-6
使能和禁止	10-6
预分频器	10-7
在休眠和空闲模式的工作	10-8
周期选择	10-7
可编程数字噪声滤波器	16-9
可编程振荡器后分频器	7-33
空闲模式	10-4
唤醒时的延时	10-5
在复位时唤醒	10-5
在 WDT 超时时唤醒	10-5
中断时唤醒	10-5
控制采样 / 转换工作	17-29
控制寄存器	12-6, 17-4, 18-4
的中断分配	6-14
控制寄存器说明	3-18
框图	
A 类型定时器	12-3
B 类型定时器	12-4
B 类型 - C 类型定时器对 (32 位定时器)	12-17
C 类型定时器	12-5
dsPIC30F CPU 内核	2-3
DSP 引擎	2-19
低电压检测 (LVD)	9-3
复位系统	8-2
共用的端口结构	11-4
输出比较模块	14-2
输入变化通知	11-7
输入捕捉	13-2
UART	19-2
UART 发送器	19-11
UART 接收器	19-16
WDT	10-6
外部上电复位电路 (用于 VDD 上升时间缓慢的情况) ..	8-7
专用端口结构	11-2

L

LAT (I/O 锁存) 寄存器	11-3
LP 振荡器	
间断工作	7-31
连续工作	7-31
使能	7-31
用于 Timer1 的操作	7-31
LVD	
初始化步骤	9-5
工作原理	9-5
控制位	9-3
设计技巧	9-6
跳变点选择	9-3
相关应用笔记	9-7
运行时的电流消耗	9-5
在休眠和空闲模式下运行	9-5
累加器 “回写”	2-25
连接注意事项	17-47
连通性开发板	25-14

M

Microchip 硬件和语言工具	25-2
MPLAB 6.XX 集成开发环境软件	25-2
MPLAB ICD 2 在线调试器	25-5
MPLAB ICE 4000 在线仿真器	25-4
MPLAB SIM 软件模拟器	25-3
脉宽调制模式	14-18
带有故障保护输入引脚	14-19
占空比	14-20
周期	14-19
模起始和结束地址选择	
XMODEND 寄存器	3-8
XMODSRT 寄存器	3-8
YMODEND 寄存器	3-8
YMODSRT 寄存器	3-8
模寻址	3-7
的适用性	3-11
递减缓冲区的初始化	3-13
递增模数缓冲区的初始化	3-12
计算	3-9
起始和结束地址选择	3-8
W 地址寄存器选择	3-9

N

内部参考电压	9-3
内部低功耗 RC (LPRC) 振荡器	7-32
使能	7-32
内部快速 RC 振荡器 (FRC)	7-31

O

OSEK 操作系统	25-10

P

POR 和长振荡器起振时间	8-12
PRO MATE II 通用器件编程器	25-5
PSV 的配置	4-8
PSV 配置	
时序	4-10
在 REPEAT 循环中使用 PSV	4-10
指令停顿	4-10
X 和 Y 数据空间的 PSV 映射	4-8
PWM 时基	15-16
PWM 更新锁定	15-37
PWM 故障引脚	15-34
PWM 输出改写	15-31
PWM 输出和极性控制	15-34
PWM 特殊事件触发器	15-38
PWM 占空比较单元	15-20
配置位说明	24-6
BOR 和 POR	24-6
电机控制 PWM 模块	24-6
通用代码段	24-6

Q

器件标识寄存器	24-7
器件 ID (DEVID)	24-7
组件 ID 字段	24-7
器件从休眠 / 空闲中唤醒	13-10
器件复位时间	8-11
器件配置	
寄存器说明	24-2
器件起振时间线	8-13
其他 dsPIC30F CPU 控制寄存器	2-16
DISCNT	2-16
MODCON	2-16

PSVPAG	2-16
TBLPAG	2-16
XBREV	2-16
XMODSRT, XMODEND	2-16
YMODSRT, YMODEND	2-16
欠压复位 (BOR)	8-8
配置	8-9
运行时的电流消耗	8-9

R

R/W 位	21-35, 26-4
RBF	21-19
如何将转换结果写入缓冲器的说明 (12 位)	18-19
如何将转换结果写入缓冲器 (10 位) 的说明	17-30
如何开始采样 (12 位)	18-14
如何停止采样和开始转换 (12 位)	18-14
软件堆栈	
示例	2-9
W14 堆栈帧指针	2-10
指针	2-8
指针上溢	2-10
指针下溢	2-10
软件复位指令 (SWR)	8-7
软陷阱	6-6
算术错误 (级别 11)	6-7

S

闪存程序存储器	
编程算法	5-11
擦除一行	5-12
装入写锁存器	5-13
闪存和数据 EEPROM 编程	
控制寄存器	5-5
NVMADR	5-6
NVMCON	5-5
NVMKEY	5-6
上电复位 (POR)	8-5
使用	8-7
上电延时时序 (PWRT)	8-7
舍入逻辑	2-25
设置产生连续输出脉冲	14-15
实时操作系统 (RTOS)	25-9
时序图	
边沿检测模式	13-8
从中断返回	6-13
带地址检测的接收 (ADDEN=1)	19-19
单比较模式 (在比较匹配事件发生时强制 OCx 为低电平)	14-6
单比较模式 (在比较匹配事件发生时设置 OCx 为高电平)	14-5
单比较模式 (在比较匹配事件发生时输出电平翻转, PR2=OCxR)	14-7
单比较模式 (在比较匹配事件发生时输出翻转, PR2=OCxR)	14-7
定时器周期匹配时的中断时序	12-14
发送 (8 位或 9 位数据)	19-13
发送 (背靠背)	19-13
后分频器更新	7-34
简单捕捉事件, 时基预分频值 =1:1	13-5
简单捕捉事件, 时基预分频值 =1:4	13-5
PWM 输出	14-18, 14-21
器件复位延时 (EC 或 RC 时钟, 禁止 PWRT)	8-16
器件复位延时 (使用 EC 和 PLL 时钟, 使能 PWRT)	8-15
器件复位延时 (使用晶振和 PLL 时钟源, 禁止 PWRT)	8-13

器件复位延时 (使用晶振和 PLL 时钟源, 使能 PWRT)	8-14
欠压状态	8-8
SPI 模式时序 (无 SS 控制)	20-12, 20-13
时钟转换	7-35
数据空间访问	2-35, 3-6
双比较模式	14-10
(连续输出脉冲, PR2 = OCxRS)	14-14, 14-15
双比较模式 (单输出脉冲, OCxRS > PR2)	14-10
双周期指令的中断时序	6-12
死区时间	15-28
UART 接收	19-17
UART 在接收溢出下的接收	19-17
VDD 上升时的 POR 模块	8-6
选通定时器模式工作	12-13
中断时序, 中断在双周期指令的第一个周期发生	6-12
使用定时器模块的外设	12-22
使用读表指令	5-3
字节模式	5-3
字模式	5-3
使用 RCON 状态位	8-10
使用写表指令	5-4
保持锁存器	5-4
字节模式	5-5
字模式	5-4
时钟切换	
工作原理	7-34
技巧	7-36
建议代码序列	7-36
使能	7-34
中止	7-36
时钟切换过程中	
进入休眠模式	7-36
输出比较	
版本历史	14-28
设计技巧	14-26
相关的寄存器映射	14-24
相关应用笔记	14-27
输出比较寄存器	14-3
数据存储	
Near	3-4
映射	3-3
数据对齐方式	3-7
数据 EEPROM 编程	5-15
擦除一个字	5-17
擦除一行	5-19
单字算法	5-16
读存储器	5-21
写一个字	5-18
写一行	5-20
行算法	5-16
数据空间地址	
发生器单元 (AGU)	3-5
X 地址发生器单元	3-5
Y 地址发生器单元	3-5
数据空间写饱和度	2-24
数据累加器	2-20
状态位	2-23
数据累加器加法器 / 减法器	2-23
输入捕捉	
缓冲器非空 (ICBNE)	13-9
设计技巧	13-12
相关的特殊功能寄存器	13-11
相关应用笔记	13-13
溢出 (ICOV)	13-9
在节能状态工作	13-10

索引

中断	13-9
控制位	13-9
输入捕捉 / 输出比较的时基	12-22
输入捕捉寄存器	13-3
输入捕捉事件模式	13-4
数学库	25-7
双周期指令的	
中断响应时间	6-12
死区时间控制	15-27
算术逻辑单元 (ALU)	2-17
锁相环 (PLL)	
频率范围	7-30
锁定状态	7-30
在上电复位过程中失去锁定	7-30
在时钟切换的过程中失去锁定	7-30
在正常的器件工作过程中失去锁定	7-30

T

TBF	21-18
TCP/IP 协议堆栈	25-10
调整振荡电路	7-23
TRIS (数据方向) 寄存器	11-3
特殊功能寄存器复位状态	8-16
桶形移位器	2-26
通用开发板	25-12

U

UART	
ADDEN 控制位	19-18
备用 I/O 引脚	19-10
波特率发生器	19-8
发送器	19-11
缓冲器 (UxTXB)	19-12
设置	19-13
中断	19-12
中止符的发送	19-14
接收器	19-14
错误处理	19-14
缓冲器 (UxRXB)	19-14
设置接收	19-17
中断	19-15
禁止	19-10
控制寄存器	19-3
配置	19-10
其他特性	19-21
其它特性	
CPU 休眠和空闲模式下的工作	19-21
环回模式	19-21
自动波特率支持	19-21
设计技巧	19-23
设置 9 位发送	19-18
使能	19-10
使用 UART 进行 9 位通信	19-18
相关的寄存器	19-22
相关应用笔记	19-24
UART 自动波特率支持	13-9
USART	
版本历史	19-25
初始化	19-20
简介	19-2
接收中止字符	19-19
使用地址检测模式的 9 位接收的设置	19-19

V

V.22/V.22bis 和 V.32 规范	25-11

W

WDT 和节能模式	
版本历史	10-12
设计技巧	10-10
相关的应用笔记	10-11
外部复位 (EXTR)	8-7
外部 RC 振荡器	7-26
带 I/O 使能	7-27
工作频率	7-27
起振	7-27
外部时钟输入	7-25
外部中断支持	6-10
外设模块禁止 (PMD) 寄存器	10-9
外设驱动程序库	25-8
未初始化的 W 寄存器复位	2-7
位反转寻址	3-14
操作	3-15
代码示例	3-18
和模寻址	3-15
简介	3-14
修改器值	3-16
为晶振、时钟模式、C1、C2 和 Rs 确定最佳的值	7-24

X

先写后读相关性规则	2-36
写程序存储器	4-10
写器件配置寄存器	5-14
写入算法	5-14
休眠和空闲模式下的工作	17-55
休眠模式	10-2
从休眠模式唤醒时的时钟选择	10-2
和 FSCM 延时	10-3
唤醒时的延时	10-3
退出的延时	10-3
在复位时唤醒	10-4
在看门狗定时器超时唤醒	10-4
在使用晶体振荡器或 PLL 的休眠模式唤醒	10-3
中断时从休眠模式唤醒	10-4
选择 A/D 转换时钟	17-13
选择 A/D 转换时钟 (12 位)	18-12
选择采样的模拟输入	17-14
选择参考电压源	17-13
选择参考电压源 (12 位)	18-11
循环结构	2-29
DO	2-32
REPEAT	2-30

Y

应答脉冲	21-38
硬陷阱	6-7
地址错误 (级别 13)	6-8
堆栈错误 (级别 12)	6-6
优先级和冲突	6-7
振荡器故障 (级别 14)	6-8
影子寄存器	2-6
DO 循环	2-7
PUSH.S 和 POP.S	2-7
用于器件仿真的特殊功能	15-39
预分频器捕捉事件	13-6
运行时自编程 (RTSP)	5-10
工作原理	5-10
闪存操作	5-10

Z

在休眠和空闲模式下工作 (12 位)	18-33

振荡器	
版本历史	7-40
配置	
时钟切换模式配置位	7-6
设计技巧	7-38
系统功能综述	7-2
相关应用笔记	7-39
谐振器起振	7-22
振荡器从休眠模式起振	7-23
振荡器缓慢起振	10-3
振荡器模式选择指导方针	7-21
振荡器起振定时器 (OST)	7-31
正交编码器接口中断	16-17
正交译码器	16-10
指令流类型	2-27
指令停顿周期	2-36
中断	
版本历史	6-46
设计技巧	6-44
相关应用笔记	6-45
中断操作	6-9
从中断返回	6-9
嵌套	6-10
中断处理时序	6-11
中断控制和状态寄存器	6-14
CORCON	6-14
IECx	6-14
IFSx	6-14
INTCON1、INTCON2	6-14
IPCx	6-14
SR	6-14
中断控制器	
相关的特殊功能寄存器	6-43
中断设置流程	6-42
初始化	6-42
陷阱服务程序	6-42
中断服务程序	6-42
中断禁止	6-42
中断矢量表	6-2
中断响应时间	
单周期指令	6-11
中断响应时间的特殊条件	6-13
中断优先级	6-5
主振荡器	7-20
转换过程示例	17-31
转换过程示例 (12 位)	18-21
字节到字的转变	2-17

全球销售及服务中心

美洲

公司总部 **Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277

技术支持:
<http://support.microchip.com>
网址: www.microchip.com

亚特兰大 Atlanta
Alpharetta, GA
Tel: 1-770-640-0034
Fax: 1-770-640-0307

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

科科莫 Kokomo
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

加拿大多伦多 Toronto
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 **Asia Pacific Office**
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 - 北京
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

中国 - 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 - 福州
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

中国 - 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 - 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 - 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 - 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 - 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 - 顺德
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

中国 - 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 - 西安
Tel: 86-29-8833-7250
Fax: 86-29-8833-7256

台湾地区 - 高雄
Tel: 886-7-536-4818
Fax: 886-7-536-4803

台湾地区 - 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

台湾地区 - 新竹
Tel: 886-3-572-9526
Fax: 886-3-572-6459

亚太地区

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

日本 Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

韩国 Korea - Gumi
Tel: 82-54-473-4301
Fax: 82-54-473-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Penang
Tel: 60-4-646-8870
Fax: 60-4-646-5086

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

英国 UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820