

CMS89FT6xx

用户手册

增强型闪存 8 位 CMOS 单片机 V1.7

请注意以下有关CMS知识产权政策

* 中微半导体公司已申请了专利，享有绝对的合法权益。与中微半导体公司MCU或其他产品有关的专利权并未被同意授权使用，任何经由不当手段侵害中微半导体公司专利权的公司、组织或个人，中微半导体公司将采取一切可能的法律行动，遏止侵权者不当的侵权行为，并追讨中微半导体公司因侵权行为所受的损失、或侵权者所得的不法利益。

* 中微的名称和标识都是中微半导体公司的注册商标。

* 中微半导体公司保留对规格书中产品在可靠性、功能和设计方面的改进作进一步说明的权利。然而中微半导体公司对于规格内容的使用不负责任。文中提到的应用其目的仅仅是用来做说明，中微半导体公司不保证和不表示这些应用没有更深入的修改就能适用，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。中微半导体公司产品不授权适用于救生、维生器件或系统中作为关键器件。中微半导体公司拥有不事先通知而修改产品的权利，对于最新的信息，请参考我们的网站<http://www.mcu.com.cn>

目录

CMS89FT6XX	1
1. 产品概述	1
1.1 功能特性.....	1
1.2 系统结构框图.....	2
1.3 管脚分布.....	3
1.3.1 CMS89FT623 引脚图.....	3
1.3.2 CMS89FT6231 引脚图.....	4
1.3.3 CMS89FT626 引脚图.....	5
1.3.4 CMS89FT627 引脚图.....	6
1.3.5 CMS89FT628 引脚图.....	7
1.4 系统配置寄存器.....	8
1.5 在线串行编程.....	9
2. 中央处理器（CPU）	10
2.1 内存.....	10
2.1.1 程序内存.....	10
2.1.2 数据存储器.....	15
2.2 寻址方式.....	20
2.2.1 直接寻址.....	20
2.2.2 立即寻址.....	20
2.2.3 间接寻址.....	20
2.3 堆栈.....	21
2.4 工作寄存器（ACC）.....	22
2.4.1 概述.....	22
2.4.2 ACC 应用.....	22
2.5 程序状态寄存器（STATUS）.....	23
2.6 预分频器（OPTION_REG）.....	25
2.7 程序计数器（PC）.....	27
2.8 看门狗计数器（WDT）.....	28
2.8.1 WDT 周期.....	28
2.8.2 看门狗定时器控制寄存器 WDTCON.....	28
3. 系统时钟	29
3.1 概述.....	29
3.2 系统振荡器.....	30
3.2.1 内部 RC 振荡.....	30
3.3 起振时间.....	30
3.4 振荡器控制寄存器.....	31
4. 复位	32

4.1	上电复位	32
4.2	掉电复位	33
4.2.1	概述	33
4.2.2	掉电复位的改进办法	34
4.3	看门狗复位	34
5.	休眠模式	35
5.1	进入休眠模式	35
5.2	从休眠状态唤醒	35
5.3	使用中断唤醒	35
5.4	休眠模式应用举例	36
5.5	休眠模式唤醒时间	36
6.	I/O 端口	37
6.1	I/O 口结构图	38
6.2	PORTA	39
6.2.1	PORTA 数据及方向控制	39
6.2.2	PORTA 上拉电阻	40
6.3	PORTB	41
6.3.1	PORTB 数据及方向	41
6.3.2	PORTB 上拉电阻	42
6.4	PORTC	43
6.4.1	PORTC 数据及方向	43
6.4.2	PORTC 上拉电阻	44
6.5	PORTD	45
6.5.1	PORTD 数据及方向	45
6.5.2	PORTD 上拉电阻	46
6.6	I/O 使用	47
6.6.1	写 I/O 口	47
6.6.2	读 I/O 口	47
6.7	I/O 口使用注意事项	48
7.	中断	49
7.1	中断概述	49
7.2	中断控制寄存器	50
7.2.1	中断控制寄存器	50
7.2.2	外设中断允许寄存器	51
7.2.3	外设中断请求寄存器	52
7.3	中断现场的保护方法	54
7.4	中断的优先级, 及多中断嵌套	54
8.	定时计数器 TIMERO	55
8.1	定时计数器 TIMERO 概述	55

8.2	TIMER0 的工作原理	56
8.2.1	8 位定时器模式	56
8.2.2	8 位计数器模式	56
8.2.3	软件可编程预分频器	56
8.2.4	在 TIMER0 和 WDT 模块间切换预分频器	57
8.2.5	TIMER0 中断	57
8.3	与 TIMER0 相关寄存器	58
9.	定时计数器 TIMER2	59
9.1	TIMER2 概述	59
9.2	TIMER2 的工作原理	60
9.3	TIMER2 相关的寄存器	61
10.	模数转换 (ADC)	62
10.1	ADC 概述	62
10.2	ADC 配置	63
10.2.1	端口配置	63
10.2.2	通道选择	63
10.2.3	ADC 参考电压	63
10.2.4	转换时钟	63
10.2.5	ADC 中断	64
10.2.6	结果格式化	64
10.3	ADC 工作原理	65
10.3.1	启动转换	65
10.3.2	完成转换	65
10.3.3	终止转换	65
10.3.4	ADC 在休眠模式下的工作原理	65
10.3.5	A/D 转换步骤	66
10.4	ADC 相关寄存器	67
11.	主控同步串行端口 (MSSP) 模块	70
11.1	主控 SSP(MSSP)模块概述	70
11.2	SPI 模式	70
11.2.2	SPI 相关寄存器	71
11.2.3	SPI 工作原理	73
11.2.4	使能 SPI I/O	74
11.2.5	主控模式	74
11.2.6	从动模式	75
11.2.7	从动选择同步	76
11.2.8	休眠操作	78
11.2.9	复位的影响	78
11.3	I ² C 模块	79

11.3.1	相关寄存器说明	80
11.3.2	主控模式	83
11.3.3	I ² C 主控模式支持	84
11.3.4	波特率发生器	86
11.3.5	I ² C 主控模式发送	87
11.3.6	I ² C 主控模式接收	88
11.3.7	I ² C 主控模式启动条件时序	90
11.3.8	I ² C 主控模式重复启动条件时序	91
11.3.9	应答序列时序	92
11.3.10	停止条件序列	93
11.3.11	时钟仲裁	94
11.3.12	多主机模式	94
11.3.13	多主机通信、总线冲突与总线仲裁	95
11.3.14	从动模式	95
11.3.15	SSP 屏蔽寄存器	98
11.3.16	休眠模式下的操作	98
11.3.17	复位的影响	98
12.	LCD/LED 驱动模块	99
12.1	LCD/LED 功能使能	99
12.2	LCD/LED 功能管脚设置	99
12.3	LCD/LED 功能 COM 口设置	100
12.4	LCD/LED 功能的 SEG 口设置	100
12.5	LCD/LED 功能的数据设置	101
12.6	LCD/LED 相关寄存器	102
13.	程序 EEPROM 控制	106
13.1	程序 EEPROM 概述	106
13.2	相关寄存器	107
13.2.1	EEADR 寄存器	107
13.2.2	EECON1 和 EECON2 寄存器	107
13.3	读程序 EEPROM	109
13.4	写程序 EEPROM	110
13.5	EPROM 操作注意事项	111
13.5.1	写校验	111
13.5.2	避免误写的保护	111
14.	触摸按键功能	112
14.1	触摸按键结构	112
14.2	触摸按键寄存器描述	112
14.3	触摸按键操作	118
14.4	触摸按键中断	120

15. 电气参数	121
15.1 DC 特性	121
15.2 AC 特性	121
15.3 内部 RC 振荡特性	122
15.3.1 内部 RC 振荡电压特性	122
15.3.2 内部 RC 振荡温度特性	122
16. 指令	123
16.1 指令一览表	123
16.2 指令说明	125
17. 封装	141
17.1 SOP 16	141
17.2 SOP 20	143
17.3 SOP24	144
17.4 SKY24	145
17.5 SOP28	146
18. 版本修订说明	147

1. 产品概述

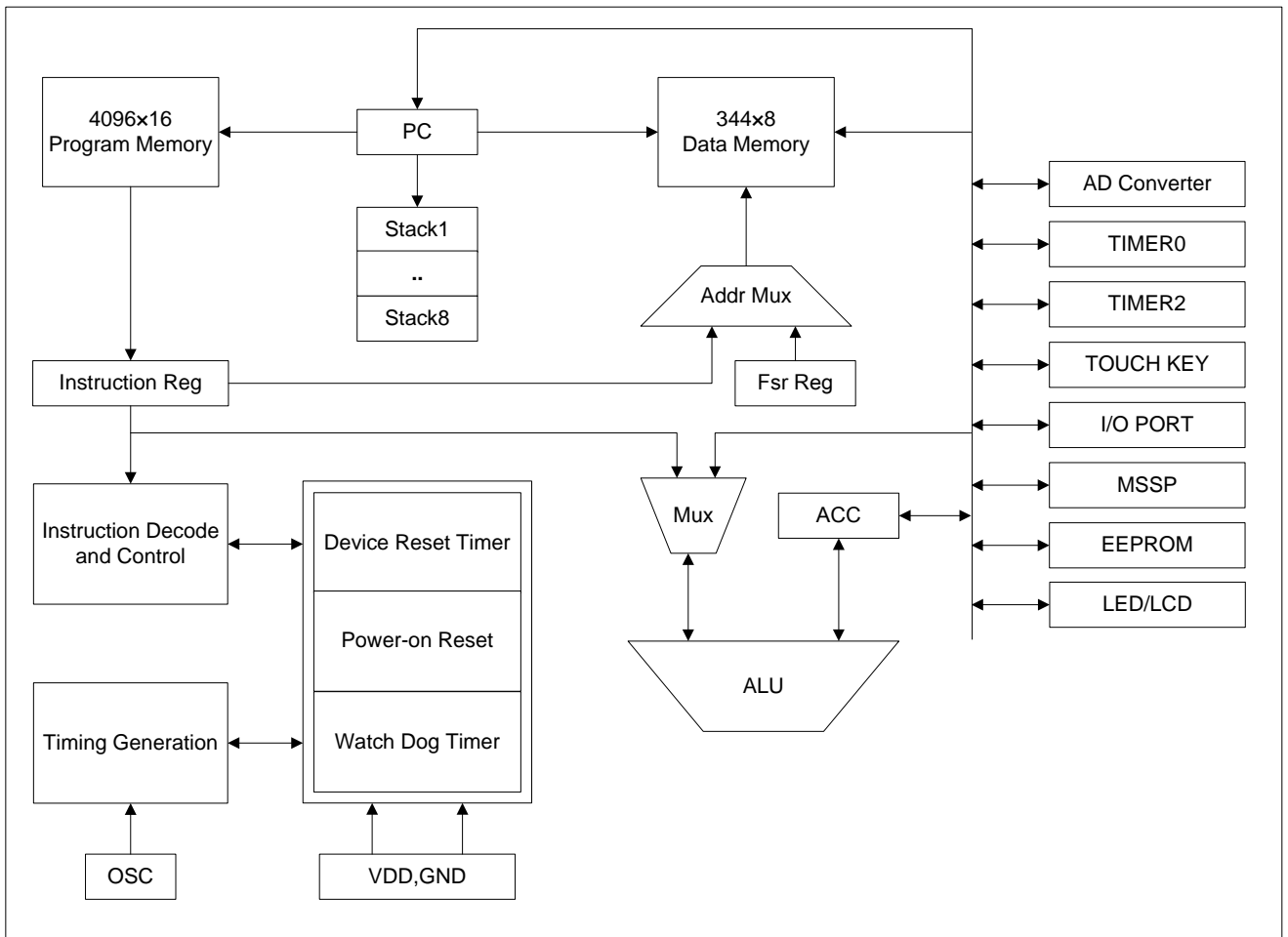
1.1 功能特性

- ◆ 内存
 - Flash: 4K×16
 - 通用 RAM: 344×8
- ◆ 8 级堆栈缓存器
- ◆ 简洁实用的指令系统 (68 条指令)
- ◆ 高精度 12 位 ADC
- ◆ 内置低压侦测电路
- ◆ 中断源
 - 2 个定时中断
 - 其它外设中断
- ◆ 内置 32×16 程序 EEPROM
- ◆ 查表功能
- ◆ 工作电压范围: 3.0V~5.5V @ 8MHz
2.5V~5.5V @ 4MHz
- ◆ 工作温度范围: -40℃~85℃
- ◆ 一种振荡方式
 - 内部 RC 振荡: 设计频率 8MHz
- ◆ 指令周期 (单指令或双指令)
- ◆ 内置 WDT 定时器
- ◆ MSSP 通信模块 (SPI/I²C)
- ◆ 定时器
 - 8 位定时器 TIMER0, TIMER2
- ◆ 内置触摸按键模块
- ◆ 内置 LED/LCD 驱动模块

型号说明:

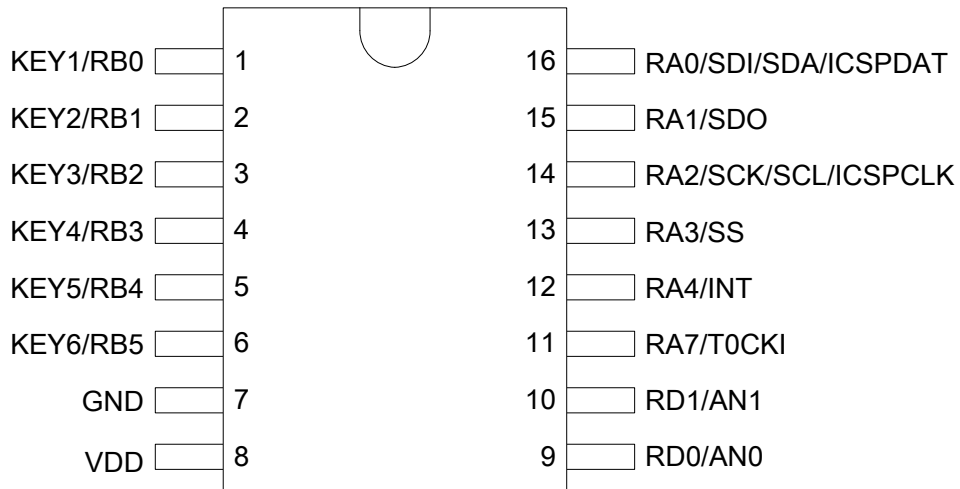
PRODUCT	Flash	RAM	PRO EE	LED	LCD	I/O	ADC	Touch	PACKAGE
CMS89FT623	4K×16	344×8	32×16	----	----	14	12Bit×2	6	SOP16
CMS89FT6231	4K×16	344×8	32×16	----	----	14	12Bit×3	8	SOP16
CMS89FT626	4K×16	344×8	32×16	7×3	15×3	18	12Bit×4	8	SOP20
CMS89FT627	4K×16	344×8	32×16	7×7	15×7	22	12Bit×8	8	SKY24/SOP24
CMS89FT628	4K×16	344×8	32×16	11×7	19×7	26	12Bit×8	12	SOP28

1.2 系统结构框图



1.3 管脚分布

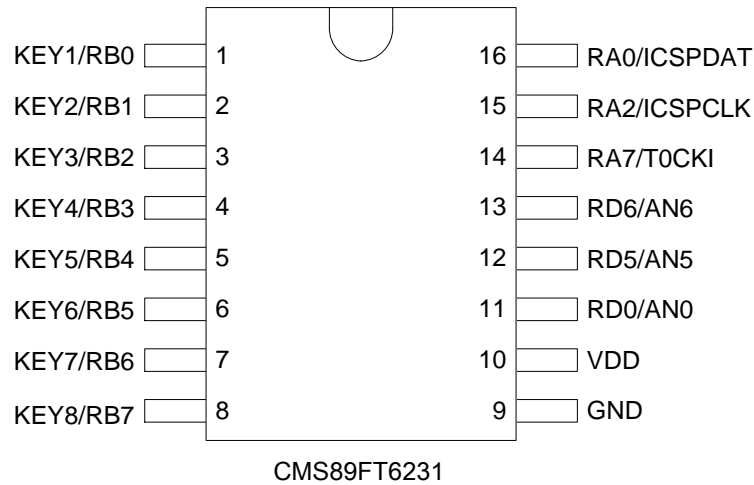
1.3.1 CMS89FT623 引脚图



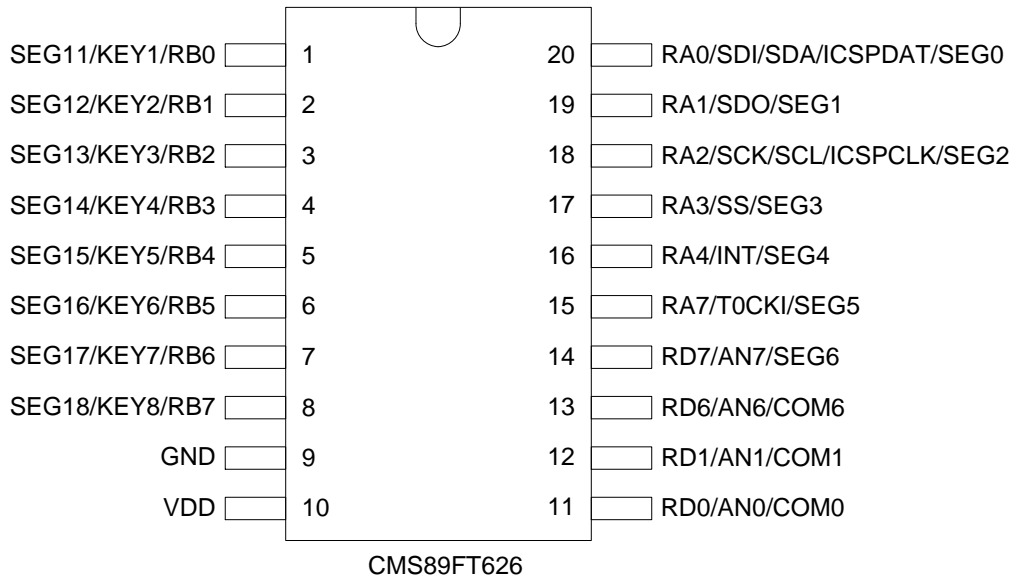
CMS89FT623

CMS89FT623 引脚说明:

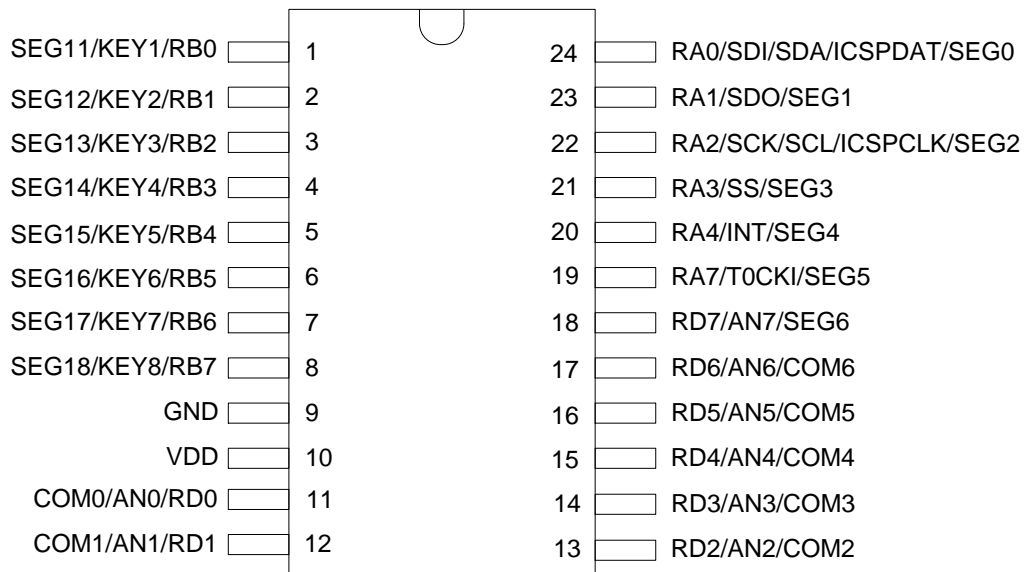
管脚名称	IO 类型	管脚说明
VDD,GND	P	电源电压输入脚，接地脚
RA0-RA4,RA7	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
RB0-RB5	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
RD0,RD1	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
ICSPCLK/ICSPDAT	I	编程时钟输入脚
AN0,AN1	I	12 位 ADC 输入脚
KEY1-KEY6	O	触摸按键输入脚
SCK	I/O	SPI 时钟
SDI	I	SPI 数据输入
SDO	O	SPI 数据输出
SS	I	SPI 从动选择
SCL	I/O	I ² C 时钟
SDA	I/O	I ² C 数据输入/输出
INT	I	外部中断输入脚
T0CKI	I	TIMER0 外部时钟输入脚

1.3.2 CMS89FT6231 引脚图

CMS89FT6231 引脚说明:

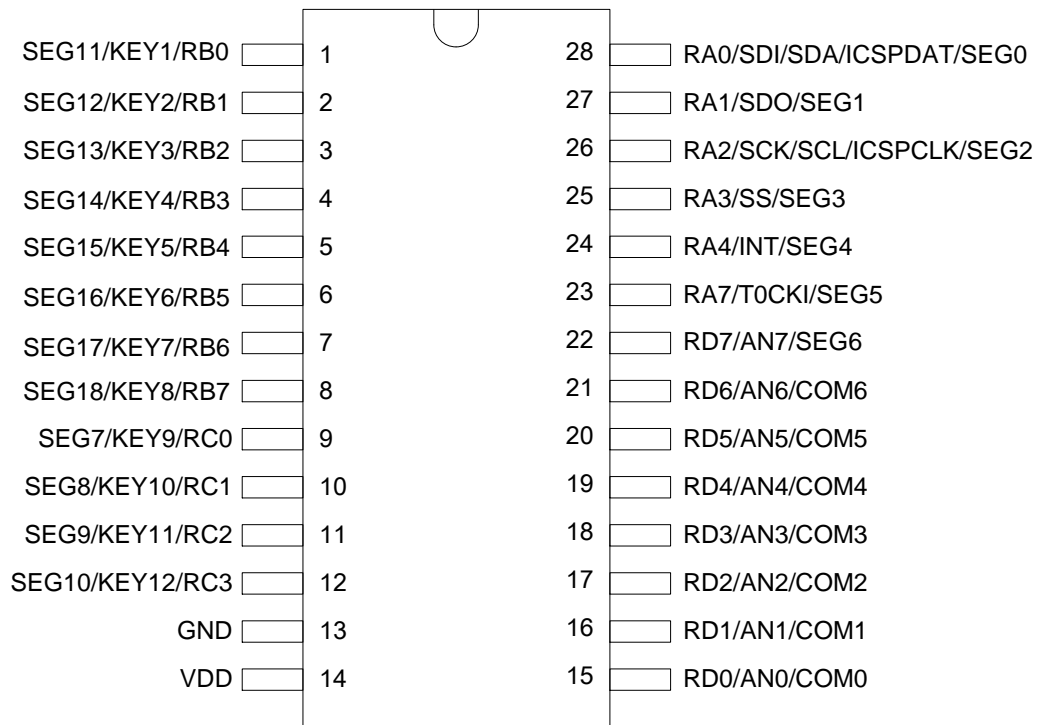
管脚名称	IO 类型	管脚说明
VDD, GND	P	电源电压输入脚, 接地脚
RA0, RA2, RA7	I/O	可编程为输入脚, 推挽输出脚, 带上拉电阻功能
RB0-RB7	I/O	可编程为输入脚, 推挽输出脚, 带上拉电阻功能
RD0, RD5, RD6	I/O	可编程为输入脚, 推挽输出脚, 带上拉电阻功能
ICSPCLK/ICSPDAT	I	编程时钟输入脚
AN0, AN5, AN6	I	12 位 ADC 输入脚
KEY1-KEY8	O	触摸按键输入脚

1.3.3 CMS89FT626 引脚图

CMS89FT626 引脚说明:

管脚名称	IO 类型	管脚说明
VDD,GND	P	电源电压输入脚，接地脚
RA0-RA4,RA7	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
RB0-RB7	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
RD0,RD1,RD6,RD7	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
ICSPCLK/ICSPDAT	I	编程时钟输入脚
AN0,AN1,AN6,AN7	I	12 位 ADC 输入脚
KEY1-KEY8	O	触摸按键输入脚
COM0,COM1,COM6	O	LED/LCD COM 口
SEG0-SEG6	O	LED/LCD SEG 口
SEG11-SEG18	O	LCD SEG 口
SCK	I/O	SPI 时钟
SDI	I	SPI 数据输入
SDO	O	SPI 数据输出
SS	I	SPI 从动选择
SCL	I/O	I ² C 时钟
SDA	I/O	I ² C 数据输入/输出
INT	I	外部中断输入脚
T0CKI	I	TIMER0 外部时钟输入脚

1.3.4 CMS89FT627 引脚图

CMS89FT627
CMS89FT627 引脚说明:

管脚名称	IO 类型	管脚说明
VDD,GND	P	电源电压输入脚，接地脚
RA0-RA4,RA7	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
RB0-RB7	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
RD0-RD7	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
ICSPCLK/ICSPDAT	I	编程时钟输入脚
AN2-AN7	I	12 位 ADC 输入脚
KEY1-KEY8	O	触摸按键输入脚
COM0-COM6	O	LED/LCD COM 口
SEG0-SEG6	O	LED/LCD SEG 口
SEG11-SEG18	O	LCD SEG 口
SCK	I/O	SPI 时钟
SDI	I	SPI 数据输入
SDO	O	SPI 数据输出
SS	I	SPI 从动选择
SCL	I/O	I ² C 时钟
SDA	I/O	I ² C 数据输入/输出
INT	I	外部中断输入脚
T0CKI	I	TIMER0 外部时钟输入脚

1.3.5 CMS89FT628 引脚图

CMS89FT628
CMS89FT628 引脚说明:

管脚名称	IO 类型	管脚说明
VDD,GND	P	电源电压输入脚，接地脚
RA0-RA4,RA7	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
RB0-RB7	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
RC0-RC3	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
RD0-RD7	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
ICSPCLK/ICSPDAT	I	编程时钟输入脚
AN0-AN7	I	12 位 ADC 输入脚
KEY1-KEY12	O	触摸按键输入脚
COM0-COM6	O	LED/LCD COM 口
SEG0-SEG10	O	LED/LCD SEG 口
SEG11-SEG18	O	LCD SEG 口
SCK	I/O	SPI 时钟
SDI	I	SPI 数据输入
SDO	O	SPI 数据输出
SS	I	SPI 从动选择
SCL	I/O	I ² C 时钟
SDA	I/O	I ² C 数据输入/输出
INT	I	外部中断输入脚
T0CKI	I	TIMER0 外部时钟输入脚

1.4 系统配置寄存器

系统配置寄存器（CONFIG）是 MCU 初始条件的 FLASH 选项。它只能被 CMS 烧写器烧写，用户不能通过程序访问及操作。它包含了以下内容：

1. WDT（看门狗选择）
 - ◆ ENABLE 打开看门狗定时器
 - ◆ DISABLE 关闭看门狗定时器
2. PROTECT（加密）
 - ◆ DISABLE Flash 代码不加密
 - ◆ ENABLE Flash 代码加密，加密后烧写/仿真器读出来的值将不确定
3. EE_PROTECT（加密）
 - ◆ DISABLE 程序 EEPROM 代码不加密
 - ◆ ENABLE 程序 EEPROM 代码加密，加密后烧写/仿真器读出来的值将不确定
4. LVR_SEL（低压侦测电压选择）
 - ◆ 2.5V
 - ◆ 3.0V

1.5 在线串行编程

可在最终应用电路中对单片机进行串行编程。编程可以简单地通过以下 4 根线完成：

- 电源线
- 接地线
- 数据线
- 时钟线

这使用户可使用未编程的器件制造电路板，而仅在产品交付前才对单片机进行编程。从而可以将最新版本的固件或者定制固件烧写到单片机中。

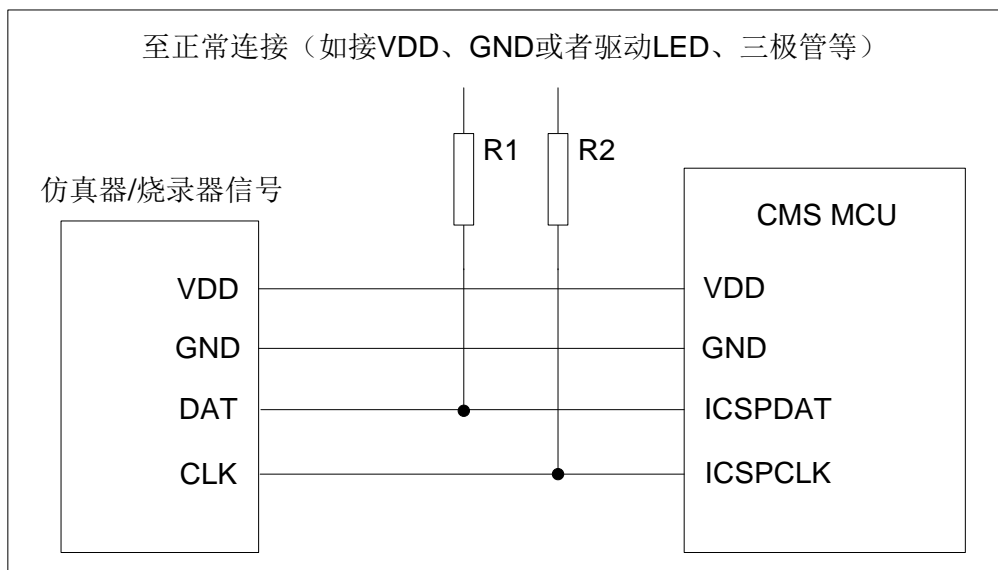


图 1-1：典型的在线串行编程连接方式

上图中，R1、R2 为电气隔离器件，常以电阻代替，其阻值如下： $R1 \geq 4.7K$ 、 $R2 \geq 4.7K$ 。

2. 中央处理器（CPU）

2.1 内存

2.1.1 程序内存

FLASH:4K		
0000H	复位向量	程序开始，跳转至用户程序
0001H		
0002H		
0003H		
0004H	中断向量	中断入口，用户中断程序
...		用户程序区
...		
...		
0FFDH		
0FFEH		
0FFFH	跳转至复位向量0000H	程序结束

2.1.1.1 复位向量（0000H）

单片机具有一个字长的系统复位向量（0000H）。具有以下三种复位方式：

- ◆ 上电复位
- ◆ 看门狗复位
- ◆ 低压复位（LVR）

发生上述任一种复位后，程序将从 0000H 处重新开始执行，系统寄存器也都将恢复为默认值。根据 STATUS 寄存器中的 PD 和 TO 标志位的内容可以判断系统复位方式。下面一段程序演示了如何定义 FLASH 中的复位向量。

例：定义复位向量

	ORG	0000H	;系统复位向量
	JP	START	
	ORG	0010H	;用户程序起始
START:			
	...		;用户程序
	...		
	END		;程序结束

2.1.1.2 中断向量

中断向量地址为 0004H。一旦有中断响应，程序计数器 PC 的当前值就会存入堆栈缓存器并跳转到 0004H 开始执行中断服务程序。所有中断都会进入 0004H 这个中断向量，具体执行哪个中断将由用户根据中断请求标志位寄存器的位决定。下面的示例程序说明了如何编写中断服务程序。

例：定义中断向量，中断程序放在用户程序之后

	ORG	0000H	;系统复位向量
	JP	START	
INT_START:	ORG	0004H	;用户程序起始
	CALL	PUSH	;保存 ACC 跟 STATUS
	...		;用户中断程序
	...		
INT_BACK:	CALL	POP	;返回 ACC 跟 STATUS
	RETI		;中断返回
START:	...		;用户程序
	...		
	END		;程序结束

注：由于单片机并未提供专门的出栈、压栈指令，故用户需自己保护中断现场。

例：中断入口保护现场

PUSH:			
	LD	ACC_BAK,A	;保存 ACC 至自定义寄存器 ACC_BAK
	SWAPA	STATUS	;状态寄存器 STATUS 高低半字节互换
	LD	STATUS_BAK,A	;保存至自定义寄存器 STATUS_BAK

例：中断出口恢复现场

POP:			
	SWAPA	STATUS_BAK	;将保存至 STATUS_BAK 的数据高低半字节互换给 ACC
	LD	STATUS,A	;将 ACC 的值给状态寄存器 STATUS
	SWAPR	ACC_BAK	;将保存至 ACC_BAK 的数据高低半字节互换
	SWAPA	ACC_BAK	;将保存至 ACC_BAK 的数据高低半字节互换给 ACC

2.1.1.3 查表

芯片具有查表功能，FLASH 空间的任何地址都可做为查表使用。

相关指令：

相关指令：

- TABLE [R] 把表格内容的低字节送给寄存器 R，高字节送到寄存器 TABLE_DATAH。
- TABLEA 把表格内容的低字节送给累加器 ACC，高字节送到寄存器 TABLE_DATAH。

相关寄存器：

- TABLE_SPH(110H) 可读写寄存器，用来指明表格高 4 位地址。
- TABLE_SPL(111H) 可读写寄存器，用来指明表格低 8 位地址。
- TABLE_DATAH(112H) 只读寄存器，存放表格高字节内容。

注：在查表之前要先把表格地址写入 TABLE_SPH 和 TABLE_SPL 中。如果主程序和中断服务程序都用到查表指令，主程序中的 TABLE_SPH 的值可能会因为中断中执行的查表指令而发生变化，产生错误。也就是说要避免在主程序和中断服务程序中都使用查表指令。但如果必须这样做的话，我们可以在查表指令前先将中断禁止，在查表结束后再开放中断，以避免发生错误。

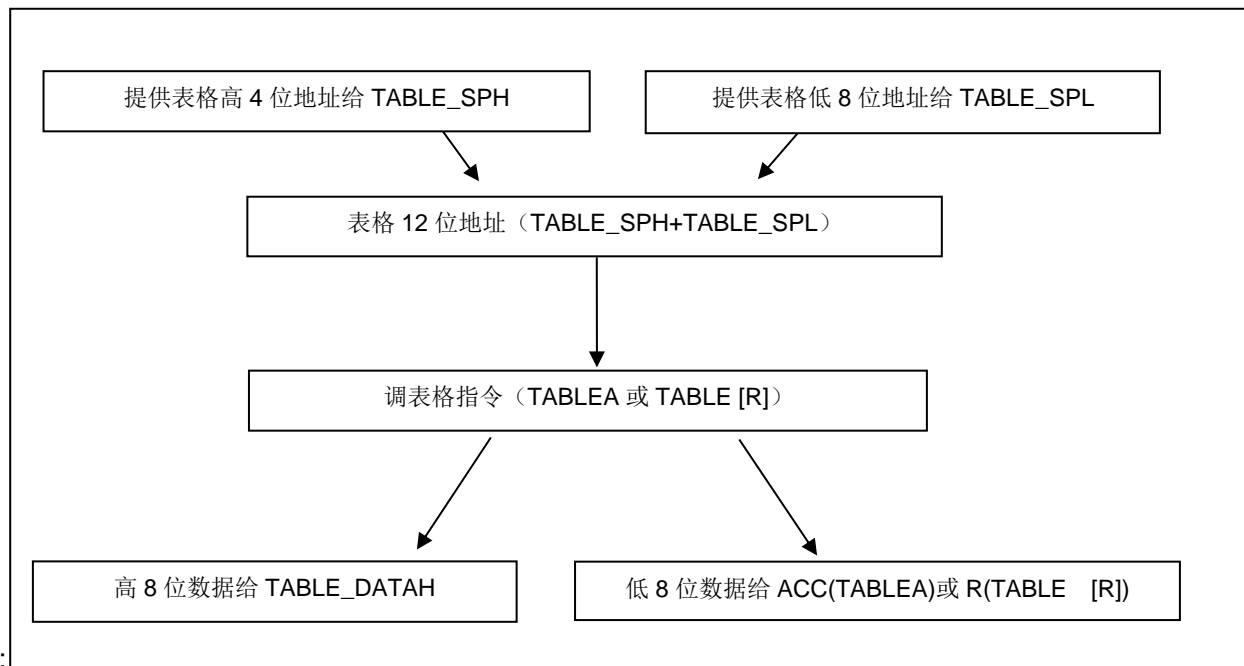


图2-1：表格调用的流程图

下面例子给出了如何在程序中调用表格。

...			;上接用户程序
LDIA	02H		;表格低位地址
LD	TABLE_SPL,A		
LDIA	06H		;表格高位地址
LD	TABLE_SPH,A		
TABLE	R01		;表格指令，将表格低 8 位(56H)给自定义寄存器 R01
LD	A,TABLE_DATAH		;将查表结果的高 8 位(34H)给累加器 ACC
LD	R02,A		;将 ACC 值(34H)给自定义寄存器 R02
...			;用户程序
ORG	0600H		;表格起始地址
DW	1234H		;0600H 地址表格内容
DW	2345H		;0601H 地址表格内容
DW	3456H		;0602H 地址表格内容
DW	0000H		;0603H 地址表格内容

2.1.1.4 跳转表

跳转表能够实现多地址跳转功能。由于 PCL 和 ACC 的值相加即可得到新的 PCL，因此，可以通过对 PCL 加上不同的 ACC 值来实现多地址跳转。ACC 值若为 n，PCL+ACC 即表示当前地址加 n，执行完当前指令后 PCL 值还会自加 1，可参考以下范例。如果 PCL+ACC 后发生溢出，PC 不会自动进位，故编写程序时应注意。这样，用户就可以通过修改 ACC 的值轻松实现多地址的跳转。

PCLATH 为 PC 高位缓冲寄存器，对 PCL 操作时，必须先对 PCLATH 进行赋值。

例：正确的多地址跳转程序示例

FLASH 地址	LDIA	01H	
	LD	PCLATH,A	;必须对 PCLATH 进行赋值
	...		
0110H:	ADDR	PCL	;ACC+PCL
0111H:	JP	LOOP1	;ACC=0, 跳转至 LOOP1
0112H:	JP	LOOP2	;ACC=1, 跳转至 LOOP2
0113H:	JP	LOOP3	;ACC=2, 跳转至 LOOP3
0114H:	JP	LOOP4	;ACC=3, 跳转至 LOOP4
0115H:	JP	LOOP5	;ACC=4, 跳转至 LOOP5
0116H:	JP	LOOP6	;ACC=5, 跳转至 LOOP6

例：错误的多地址跳转程序示例

FLASH 地址	CLR	PCLATH	
	...		
00FCH:	ADDR	PCL	;ACC+PCL
00FDH:	JP	LOOP1	;ACC=0, 跳转至 LOOP1
00FEH:	JP	LOOP2	;ACC=1, 跳转至 LOOP2
00FFH:	JP	LOOP3	;ACC=2, 跳转至 LOOP3
0100H:	JP	LOOP4	;ACC=3, 跳转至 0000H 地址
0101H:	JP	LOOP5	;ACC=4, 跳转至 0001H 地址
0102H:	JP	LOOP6	;ACC=5, 跳转至 0002H 地址

注：由于 PCL 溢出不会自动向高位进位，故在利用 PCL 作多地址跳转时，需要注意该段程序一定不能放在 FLASH 空间的分页处。

2.1.2 数据存储

CMS89FT6xx 数据存储列表

地址		地址		地址		地址	
INDF	00h	INDF	80h	INDF	100h	INDF	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	WDTCON	105h	----	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h	WPUB	107h	----	187h
PORTD	08h	TRISD	88h	WPUA	108h	----	188h
WPUC	09h	WPUD	89h	----	109h	----	189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
----	0Eh	----	8Eh	EEDATH	10Eh	TKM216DL	18Eh
----	0Fh	OSCCON	8Fh	----	10Fh	TKM216DH	18Fh
----	10h	OSCTUNE	90h	TABLE_SPH	110h	TKM2ROL	190h
TMR2	11h	SSPCON2	91h	TABLE_SPL	111h	TKM2ROH	191h
T2CON	12h	PR2	92h	TABLE_DATAH	112h	TKM2C1	192h
TKM016DL	13h	SSPAD	93h	TK16DL	113h	TKM2C0	193h
TKM016DH	14h	SSPSTAT	94h	TK16DH	114h	----	194h
TKM0ROL	15h	SSPCON	95h	TKTMR	115h	----	195h
TKM0ROH	16h	SSPBUF	96h	TKC0	116h	----	196h
TKM0C1	17h	----	97h	TKC1	117h	----	197h
TKM0C0	18h	TKM116DL	98h	LCDCON0	118h	通用寄存器 88 字节	198h
----	19h	TKM116DH	99h	LCDCON1	119h		199h
----	1Ah	TKM1ROL	9Ah	LCDADD	11Ah		19Ah
----	1Bh	TKM1ROH	9Bh	LCDDATA	11Bh		19Bh
----	1Ch	TKM1C1	9Ch	SEGEN2	11Ch		19Ch
----	1Dh	TKM1C0	9Dh	SEGEN1	11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh	SEGEN0	11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh	COMEN	11Fh		19Fh
通用寄存器 96 字节	20h	通用寄存器 80 字节	A0h	通用寄存器 88 字节	120h		1A0h
	6Fh		EFh		16Fh		1EFh
	70h		F0h		170h	1F0h	
	--		--		--	--	
7Fh	7Fh		FFh		17Fh	1FFh	
BANK0	BANK1		BANK2		BANK3		
快速存储区 70h-7Fh			快速存储区 70h-7Fh				

数据存储器由 344×8 位组成，分为两个功能区间：特殊功能寄存器和通用数据存储器。数据存储器单元大多数是可读/写的，但有些只读的。特殊功能寄存器地址为从 00H-1FH，80-9FH，100-117H，180-197H。

CMS89FT6xx 特殊功能寄存器汇总 Bank0

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
00h	INDF	寻址该单元会使用FSR 的内容寻址数据存储器（不是物理寄存器）								xxxx xxxx
01h	TMR0	TIMER0数据寄存器								xxxxxxxx
02h	PCL	程序计数器低字节								0000 0000
03h	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxx
04h	FSR	间接数据存储器地址指针								xxxx xxxx
05h	PORTA	RA7	----	----	RA4	RA3	RA2	RA1	RA0	xxxx xxxx
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx
07h	PORTC	----	----	----	----	RC3	RC2	RC1	RC0	---- xxxx
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx
09h	WPUC	----	----	----	----	WPUC3	WPUC2	WPUC1	WPUC0	---- 0000
0Ah	PCLATH	----	---	----	----	程序计数器高4位的写缓冲器				---- 0000
0Bh	INTCON	GIE	PEIE	T0IE	INTE	----	T0IF	INTF	----	0000 -00-
0Ch	PIR1	----	ADIF	TKIF	----	----	BCLIF	TMR2IF	----	-00--00-
0Dh	PIR2	----	----	----	EEIF	----	----	----	SSPIF	--0--0
11h	TMR2	TIMER2模块寄存器								0000 0000
12h	T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000
13h	TKM016DL	触摸按键模块0 16位计数器低字节内容								0000 0000
14h	TKM016DH	触摸按键模块0 16位计数器高字节内容								0000 0000
15h	TKM0ROL	触摸按键模块0 内建电容低8位选择								0000 0000
16h	TKM0ROH	触摸按键模块0 内建电容高2位选择								-----00
17h	TKM0C1	M0TSS	----	M0ROEN	M0KOEN	M0K4IO	M0K3IO	M0K2IO	M0K1IO	0-00 0000
18h	TKM0C0	M0MXS1	M0MXS0	M0DFEN	M0FILEN	M0SOFC	M0SOF2	M0SOF1	M0SOF0	0000 0000
1Eh	ADRESH	A/D结果寄存器的高字节								xxxx xxxx
1Fh	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ DONE	ADON	0000 0000

CMS89FT6xx 特殊功能寄存器汇总 Bank1

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
80h	INDF	寻址该地址单元会使用FSR的内容寻址数据存储器（不是物理寄存器）								xxxx xxxx
81h	OPTION_REG	----	INTEDG	TOCS	T0SE	PSA	PS2	PS1	PS0	-111 1111
82h	PCL	程序计数器（PC）的低字节								0000 0000
83h	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxx
84h	FSR	间接数据存储器地址指针								xxxx xxxx
85h	TRISA	TRISA7	----	----	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1--1 1111
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111
87h	TRISC	----	----	----	----	TRISC3	TRISC2	TRISC1	TRISC0	---- 1111
88h	TRISD	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	1111 1111
89h	WPUD	WPUD7	WPUD6	WPUD5	WPUD4	WPUD3	WPUD2	WPUD1	WPUD0	0000 0000
8Ah	PCLATH	----	----	----	----	程序计数器高4位的写缓冲器				---- 0000
8Bh	INTCON	GIE	PEIE	T0IE	INTE	----	T0IF	INTF	----	0000 -00-
8Ch	PIE1	----	ADIE	TKIE	----	----	BCLIE	TMR2IE	----	-00--00-
8Dh	PIE2	----	----	----	EEIE	----	----	----	SSPIE	--0--0
8Fh	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	SCS	-110 ---0
90h	OSCTUNE	----	----	----	TUN4	TUN3	TUN2	TUN1	TUN0	--0 0000
91H	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000
92h	PR2	TIMER2周期寄存器								1111 1111
93H	SSPADD	同步串行端口(I ² C模式)地址寄存器								0000 0000
93H	SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	1111 1111
94H	SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	0000 0000
95h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000
96h	SSPBUF	同步串行端口接收缓冲器/ 发送寄存器								xxxx xxxx
98h	TKM116DL	触摸按键模块1 16位计数器低字节内容								0000 0000
99h	TKM116DH	触摸按键模块1 16位计数器高字节内容								0000 0000
9Ah	TKM1ROL	触摸按键模块1 内建电容低8位选择								0000 0000
9Bh	TKM1ROH	触摸按键模块1 内建电容高2位选择								-----00
9Ch	TKM1C1	M1TSS	----	M1ROEN	M1KOEN	M1K4IO	M1K3IO	M1K2IO	M1K1IO	0-00 0000
9Dh	TKM1C0	M1MXS1	M1MXS0	M1DFEN	M1FILEN	M1SOFC	M1SOF2	M1SOF1	M1SOF0	0000 0000
9Eh	ADRESL	A/D结果寄存器的低字节								xxxx xxxx
9Fh	ADCON1	ADFM	----	----	----	----	----	----	----	0--- ----

CMS89FT6xx 特殊功能寄存器汇总 Bank2

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值	
100h	INDF	寻址该地址单元会使用FSR的内容寻址数据存储器（不是物理寄存器）								xxxx xxxx	
101h	TMR0	TIMER0模块寄存器								xxxx xxxx	
102h	PCL	程序计数器（PC）的低字节								0000 0000	
103h	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxx	
104h	FSR	间接数据存储器地址指针								xxxx xxxx	
105h	WDTCON	----	----	----	----	----	----	----	SWDTEN	---- --0	
106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	
107h	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	0000 0000	
108h	WPUA	WPUA7	----	----	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	0--0 0000	
10Ah	PCLATH	----	----	---	---	程序计数器高4位的写缓冲器				---- 0000	
10Bh	INTCON	GIE	PEIE	TOIE	INTE	----	TOIF	INTF	----	0000 -0-	
10Ch	EEDATA	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	xxxxxxxx	
10Dh	EEADR	----	----	----	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	---0 0000	
10Eh	EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	0000 0000	
110h	TABLE_SPH	表格高位指针								---- xxxx	
111h	TABLE_SPL	表格低位指针								xxxx xxxx	
112h	TABLE_DATAH	表格高位数据								xxxx xxxx	
113h	TK16DL	触摸按键模块16位计数器低字节内容								0000 0000	
114h	TK16DH	触摸按键模块16位计数器高字节内容								0000 0000	
115h	TKTMR	触摸按键8位定时/计数器寄存器								0000 0000	
116h	TKC0	----	TKRCOV	TKST	TKCFOV	TK16OV	TSCS	TK16S1	TK16S0	-000 0000	
117h	TKC1	----	----	----	----	----	----	TKFS1	TKFS0	---- --11	
118h	LCDCON0	LCDEN	LEDEN	COMSEL[1:0]		LCDCLK[3:0]				00000000	
119h	LCDCON1	VADJEN	----	SEGOUT[1:0]		VLCD[3:0]				00000000	
11Ah	LCDADD	LCDCS	B2RES	COMEL[2]	LCDADD[4:0]					00000000	
11Bh	LCDDATA	LCDDATA[7:0]								00000000	
11Ch	SEGEN2	SEGDR1[3:0]				----	SEG18EN	SEG17EN	SEG16EN		0000-000
11Dh	SEGEN1	SEG15EN	SEG14N	SEG13EN	SEG12EN	SEG11EN	SEG10EN	SEG9EN	SEG8EN	00000000	
11Eh	SEGEN0	SEG7EN	SEG6EN	SEG5EN	SEG4EN	SEG3EN	SEG2EN	SEG1EN	SEG0EN	00000000	
11Fh	COMEN	COM7EN	COM6EN	COM5EN	COM4EN	COM3EN	COM2EN	COM1EN	COM0EN	00000000	

CMS89FT6xx 特殊功能寄存器汇总 Bank3

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
180h	INDF	寻址该地址单元会使用FSR的内容寻址数据存储器（不是物理寄存器）								xxxx xxxx
181h	OPTION_REG	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	-111 1111
182h	PCL	程序计（PC）的低字节								0000 0000
183h	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxx
184h	FSR	间接数据存储器地址指针								xxxx xxxx
186h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111
18Ah	PCLATH	----	----	----	----	程序计数器高4位的写缓冲器				---- 0000
18Bh	INTCON	GIE	PEIE	T01E	INTE	----	T01F	INTF	----	0000 -00-
18Ch	EECON1	EEPGD	----	----	----	WRERR	WREN	WR	RD	0--- x000
18Dh	EECON2	EEPROM控制寄存器2（不是物理寄存器）								---- ----
18EH	TKM216DL	触摸按键模块2 16位计数器低字节内容								0000 0000
18FH	TKM216DH	触摸按键模块2 16位计数器高字节内容								0000 0000
190h	TKM2ROL	触摸按键模块2 内建电容低8位选择								0000 0000
191h	TKM2ROH	触摸按键模块2 内建电容高2位选择								0000 0000
192h	TKM2C1	M2TSS	----	M2ROEN	M2KOEN	M2K4IO	M2K3IO	M2K2IO	M2K1IO	0-00 0000
193h	TKM2C0	M2MXS1	M2MXS0	M2DFEN	M2FILEN	M2SOFC	M2SOF2	M2SOF1	M2SOF0	0000 0000

2.2 寻址方式

2.2.1 直接寻址

通过工作寄存器（ACC）来对 RAM 进行操作。

例：ACC 的值送给 30H 寄存器

LD	30H,A
----	-------

例：30H 寄存器的值送给 ACC

LD	A,30H
----	-------

2.2.2 立即寻址

把立即数传给工作寄存器（ACC）

例：立即数 12H 送给 ACC

LDIA	12H
------	-----

2.2.3 间接寻址

数据存储器能被直接或间接寻址。通过 INDF 寄存器可间接寻址，INDF 不是物理寄存器。当对 INDF 进行存取时，它会根据 FSR 寄存器内的值（低 8 位）和 STATUS 寄存器的 IRP 位（第 9 位）作为地址，并指向该地址的寄存器，因此在设置了 FSR 寄存器和 STATUS 寄存器的 IRP 位后，就可把 INDF 寄存器当作目的寄存器来存取。间接读取 INDF（FSR=0）将产生 00H。间接写入 INDF 寄存器，将导致一个空操作。以下例子说明了程序中间接寻址的用法。

例：FSR 及 INDF 的应用

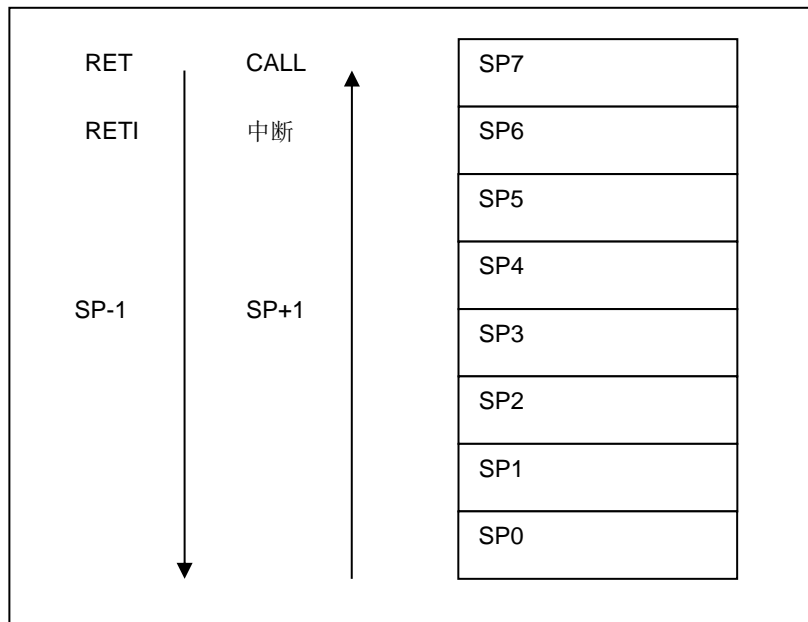
LDIA	30H	
LD	FSR,A	;间接寻址指针指向 30H
CLRB	STATUS,IRP	;指针第 9 位清零
CLR	INDF	;清零 INDF 实际是清零 FSR 指向的 30H 地址 RAM

例：间接寻址清 RAM(20H-7FH)举例:

LDIA	1FH	
LD	FSR,A	;间接寻址指针指向 1FH
CLRB	STATUS,IRP	
LOOP:		
INCR	FSR	;地址加 1, 初始地址为 30H
CLR	INDF	;清零 FSR 所指向的地址
LDIA	7FH	
SUBA	FSR	
SNZB	STATUS,C	;一直清零至 FSR 地址为 7FH
JP	LOOP	

2.3 堆栈

芯片的堆栈缓存器共 8 层，堆栈缓存器既不是数据存储器的—部分，也不是程序内存的一部分，且既不能被读出，也不能被写入。对它的操作通过堆栈指针（SP）来实现，堆栈指针（SP）也不能读出或写入，当系统复位后堆栈指针会指向堆栈顶部。当发生子程序调用及中断时的程序计数器（PC）值被压入堆栈缓存器，当中断或子程序返回时将数值返回给程序计数器（PC），下图说明其工作原理。



堆栈缓存器的使用将遵循一个原则“先进后出”。

注：堆栈缓存器只有 8 层，如果堆栈已满，并且发生不可屏蔽的中断，那么只有中断标志位会被记录下来，而中断响应则会被抑制，直到堆栈指针发生递减，中断才会被响应，这个功能可以防止中断使堆栈溢出，同样如果堆栈已满，并且发生子程序调用，那么堆栈将会发生溢出，首先进入堆栈的内容将会丢失，只有最后 8 个返回地址被保留，故用户在写程序时应注意此点，以免发生程序走飞。

2.4 工作寄存器（ACC）

2.4.1 概述

ALU 是 8Bit 宽的算术逻辑单元，MCU 所有的数学、逻辑运算均通过它来完成。它可以对数据进行加、减、移位及逻辑运算；ALU 也控制状态位（STATUS 状态寄存器中），用来表示运算结果的状态。

ACC 寄存器是一个 8Bit 的寄存器，ALU 的运算结果可以存放在此，它并不属于数据存储器的一部分而是位于 CPU 中供 ALU 在运算中使用，因此不能被寻址，只能通过所提供的指令来使用。

2.4.2 ACC 应用

例：用 ACC 做数据传送

LD	A,R01	;将寄存器 R01 的值赋给 ACC
LD	R02,A	;将 ACC 的值赋给寄存器 R02

例：用 ACC 做立即寻址目标操作数

LDIA	30H	;给 ACC 赋值 30H
ANDIA	30H	;将当前 ACC 的值跟立即数 30H 进行“与”操作， ;结果放入 ACC
XORIA	30H	;将当前 ACC 的值跟立即数 30H 进行“异或”操作， ;结果放入 ACC

例：用 ACC 做双操作数指令的第一操作数

HSUBA	R01	;ACC-R01，结果放入 ACC
HSUBR	R01	;ACC-R01，结果放入 R01

例：用 ACC 做双操作数指令的第二操作数

SUBA	R01	;R01-ACC，结果放入 ACC
SUBR	R01	; R01-ACC，结果放入 R01

2.5 程序状态寄存器（STATUS）

STATUS 寄存器如寄存器下表所示，包含：

- ALU 的算术状态；
- 复位状态；
- 数据存储器（GPR 和 SFR）的存储区选择位。

与其他寄存器一样，STATUS 寄存器可以是任何指令的目标寄存器。如果一条影响 Z、DC 或 C 位的指令以 STATUS 寄存器作为目标寄存器，则不能写这 3 个状态位。这些位根据器件逻辑被置 1 或清零。而且也不能写 TO 和 PD 位。因此将 STATUS 作为目标寄存器的指令可能无法得到预期的结果。

例如，CLR STATUS 会清零高 3 位，并将 Z 位置 1。这样 STATUS 的值将为 000u u1uu（其中 u = 不变）。因此，建议仅使用 CLRB、SETB、SWAPA、SWAPR 指令来改变 STATUS 寄存器，因为这些指令不会影响任何状态位。

程序状态寄存器 STATUS（03H）

03H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	1	1	X	X	X

Bit7 IRP: 寄存器存储区选择位（用于间接寻址）；
 1= Bank 2和Bank 3（100h-1FFh）；
 0= Bank 0和Bank 1（00h-FFh）。

Bit6~Bit5 RP[1:0]: 存储区选择位；
 00: 选择Bank 0；
 01: 选择Bank 1；
 10: 选择Bank 2；
 11: 选择Bank 3。

Bit4 TO: 超时位；
 1= 上电或执行了CLRWDWT指令或STOP指令；
 0= 发生了WDT超时。

Bit3 PD: 掉电位；
 1= 上电或执行了CLRWDWT指令指令或STOP指令；
 0= 执行了STOP指令。

Bit2 Z: 结果为零位；
 1= 算术或逻辑运算的结果为零；
 0= 算术或逻辑运算的结果不为零。

Bit1 DC: 半进位/借位位；
 1= 发生了结果的低4位向高位进位；
 0= 结果的低4位没有向高位进位。

Bit0 C: 进位/借位位；
 1= 结果的最高位发生了进位或没有发生借位；
 0= 结果的最高位没有发生进位或发生了借位。

TO 和 PD 标志位可反映出芯片复位的原因，下面列出影响 TO、PD 的事件及各种复位后 TO、PD 的状态。

事件	TO	PD
电源上电	1	1
WDT 溢出	0	X
STOP 指令	1	0
CLRWDT 指令	1	1
休眠	1	0

影响 PD、TO 的事件表

TO	PD	复位原因
0	0	WDT 溢出唤醒休眠 MCU
0	1	WDT 溢出非休眠态
1	1	电源上电

复位后 TO/PD 的状态

2.6 预分频器 (OPTION_REG)

OPTION_REG 寄存器是可读写的寄存器，包括各种控制位用于配置：

- ◆ TIMER0/WDT 预分频器；
- ◆ TIMER0。

预分频器 OPTION_REG (81H)

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
读写	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	1	1	1	1	1	1	1

Bit7 未用。

Bit6 INTEDG: 触发中断的边沿选择位。
 1= INT 引脚上升沿触发中断。
 0= INT 引脚下降沿触发中断。

Bit5 T0CS: TMR0 时钟源选择位。
 0: 内部指令周期时钟 ($F_{SYS}/4$)。
 1: T0CKI 引脚上的跳变沿。

Bit4 T0SE: TIMER0 时钟源边沿选择位。
 0: 在 T0CKI 引脚信号从低电平跳变到高电平时递增。
 1: 在 T0CKI 引脚信号从高电平跳变到低电平时递增。

Bit3 PSA: 预分频器分配位。
 0: 预分频器分配给 TIMER0 模块。
 1: 预分频器分配给 WDT。

Bit2~Bit0 PS2~PS0: 预分配参数配置位。

PS2	PS1	PS0	TMR0 分频比	WDT 分频比
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

预分频寄存器实际上是一个 8 位的计数器，用于监视寄存器 WDT 时，是作为一个后分频器；用于定时器/计数器时，作为一个预分频器，通常统称作预分频器。在片内只有一个物理的分频器，只能用于 WDT 或 TIMER0，两者不能同时使用。也就是说，若用于 TIMER0，WDT 就不能使用预分频器，反之亦然。

当用于 WDT 时，CLRWDW 指令将同时对预分频器和 WDT 定时器清零。

当用于 TIMER0 时，有关写入 TIMER0 的所有指令（如：CLR TMR0,SETB TMR0,1 等）都会对预分频器清零。

由 **TIMER0** 还是 **WDT** 使用预分频器，完全由软件控制，可以动态改变。为了避免出现不该有的芯片复位，当从 **TIMER0** 换为 **WDT** 使用时，应该执行以下指令。

CLR	TMR0	;TMR0 清零
CLRWDT		;WDT 清零
LDIA	B'00xx1111'	;必要步骤，必须执行
LD	OPTION_REG,A	;必要步骤，必须执行
LDIA	B'00xx1xxx'	;设置新的预分频器
LD	OPTION_REG,A	

将预分频器从分配给 **WDT** 切换为分配给 **TIMER0** 模块，应该执行以下指令。

CLRWDT		;WDT 清零
LDIA	B'00xx0xxx'	;设置新的预分频器
LD	OPTION_REG,A	

注：要使 **TIMER0 获取 1:1 的预分频比配置，可通过将选项寄存器的 **PSA** 位置 1 将预分频器分配给 **WDT**。**

2.7 程序计数器（PC）

程序计数器（PC）控制程序内存 FLASH 中的指令执行顺序，它可以寻址整个 FLASH 的范围，取得指令码后，程序计数器（PC）会自动加一，指向下一个指令码的地址。但如果执行跳转、条件跳转、向 PCL 赋值、子程序调用、初始化复位、中断、中断返回、子程序返回等操作时，PC 会加载与指令相关的地址而不是下一条指令的地址。

当遇到条件跳转指令且符合跳转条件时，当前指令执行过程中读取的下一条指令将会被丢弃，且会插入一个空指令操作周期，随后才能取得正确的指令。反之，就会顺序执行下一条指令。

程序计数器（PC）是 12Bit 宽度，低 8 位通过 PCL（02H）寄存器用户可以访问，高 4 位用户不能访问。可容纳 4K×16Bit 程序地址。对 PCL 赋值将会产生一个短跳转动作，跳转范围为当前页的 256 个地址。

注：当程序员在利用 PCL 作短跳转时，要先对 PC 高位缓冲寄存器 PCLATH 进行赋值。

下面给出几种特殊情况的 PC 值

复位时	PC=0000;
中断时	PC=0004(原来的 PC+1 会被自动压入堆栈);
CALL 时	PC=程序指定地址(原来的 PC+1 会被自动压入堆栈);
RET、RETI、RET i 时	PC=堆栈出来的值;
操作 PCL 时	PC[11:8]不变, PC[7:0]=用户指定的值;
JP 时	PC=程序指定的值;
其它指令	PC=PC+1;

2.8 看门狗计数器（WDT）

看门狗定时器（Watch Dog TIMER）是一个片内自振式的 RC 振荡定时器，无需任何外围组件，即使芯片的主时钟停止工作，WDT 也能保持计时。WDT 计时溢出将产生复位。

2.8.1 WDT 周期

WDT 与 TIMER0 共用 8 位预分频器。在所有复位后，WDT 溢出周期 18ms，假如你需要改变的 WDT 周期，可以设置 OPTION_REG 寄存器。WDT 的溢出周期将受到环境温度，电源电压等参数影响。

“CLRWDW”和“STOP”指令将清除 WDT 定时器以及预分频器里的计数值（当预分频器分配给 WDT 时）。WDT 一般用来防止系统失控，或者可以说是用来防止单片机程序失控。在正常情况下，WDT 应该在其溢出前被“CLRWDW”指令清零，以防止产生复位。如果程序由于某种干扰而失控，那么不能在 WDT 溢出前执行“CLRWDW”指令，就会使 WDT 溢出而产生复位。使系统重启而不至于失去控制。若是 WDT 溢出产生的复位，则状态寄存器（STATUS）的“TO”位会被清零，用户可根据此位来判断复位是否是 WDT 溢出所造成的。

注：

1. 若使用 WDT 功能，一定要在程序的某些地方放置“CLRWDW”指令，以保证在 WDT 溢出前能被清零。否则会使芯片不停的复位，造成系统无法正常工作。
2. 不能在中断程序中对 WDT 进行清零，否则无法侦测到主程序“跑飞”的情况。
3. 程序中应在主程序中有一次清 WDT 的操作，尽量不要在多个分支中清零 WDT，这种架构能最大限度发挥看门狗计数器的保护功能。
4. 看门狗计数器不同芯片的溢出时间有一定差异，所以设置清 WDT 时间时，应与 WDT 的溢出时间有较大的冗余，避免出现不必要的 WDT 复位。

2.8.2 看门狗定时器控制寄存器 WDTCON

看门狗定时器控制寄存器 WDTCON(105H)

105H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDTCON	---	---	---	---	---	---	---	SWDTEN
R/W	---	---	---	---	---	---	---	R/W
复位值	---	---	---	---	---	---	---	0

Bit7~Bit1

未用。

Bit0

SWDTEN: 软件使能或禁止看门狗定时器位

1= 使能 WDT

0= 禁止 WDT（复位值）

注：如果 CONFIG 中 WDT 配置位=1，则 WDT 始终被使能，而与 SWDTEN 控制位的状态无关。如果 CONFIG 中 WDT 配置位=0，则可以使用 SWDTEN 控制位使能或禁止 WDT。

3. 系统时钟

3.1 概述

时钟信号由内部振荡产生，在片内产生 4 个非重叠正交时钟信号，分别称作 Q1、Q2、Q3、Q4。在 IC 内部每个 Q1 使程序计数器（PC）增量加一，Q4 从程序存储单元中取出该指令，并将其锁存在指令寄存器中。在下一个 Q1 到 Q4 之间对取出的指令进行译码和执行，也就是说 4 个时钟周期才会执行一条指令。下图表示时钟与指令周期执行时序图。

一个指令周期含有 4 个 Q 周期，指令的执行和获取是采用流水线结构，取指占用一个指令周期，而译码和执行占用另一个指令周期，但是由于流水线结构，从宏观上看，每条指令的有效执行时间是一个指令周期。如果一条指令引起程序计数器地址发生改变（例如 JP）那么预取的指令操作码就无效，就需要两个指令周期来完成该条指令，这就是对 PC 操作指令都占用两个时钟周期的原因。

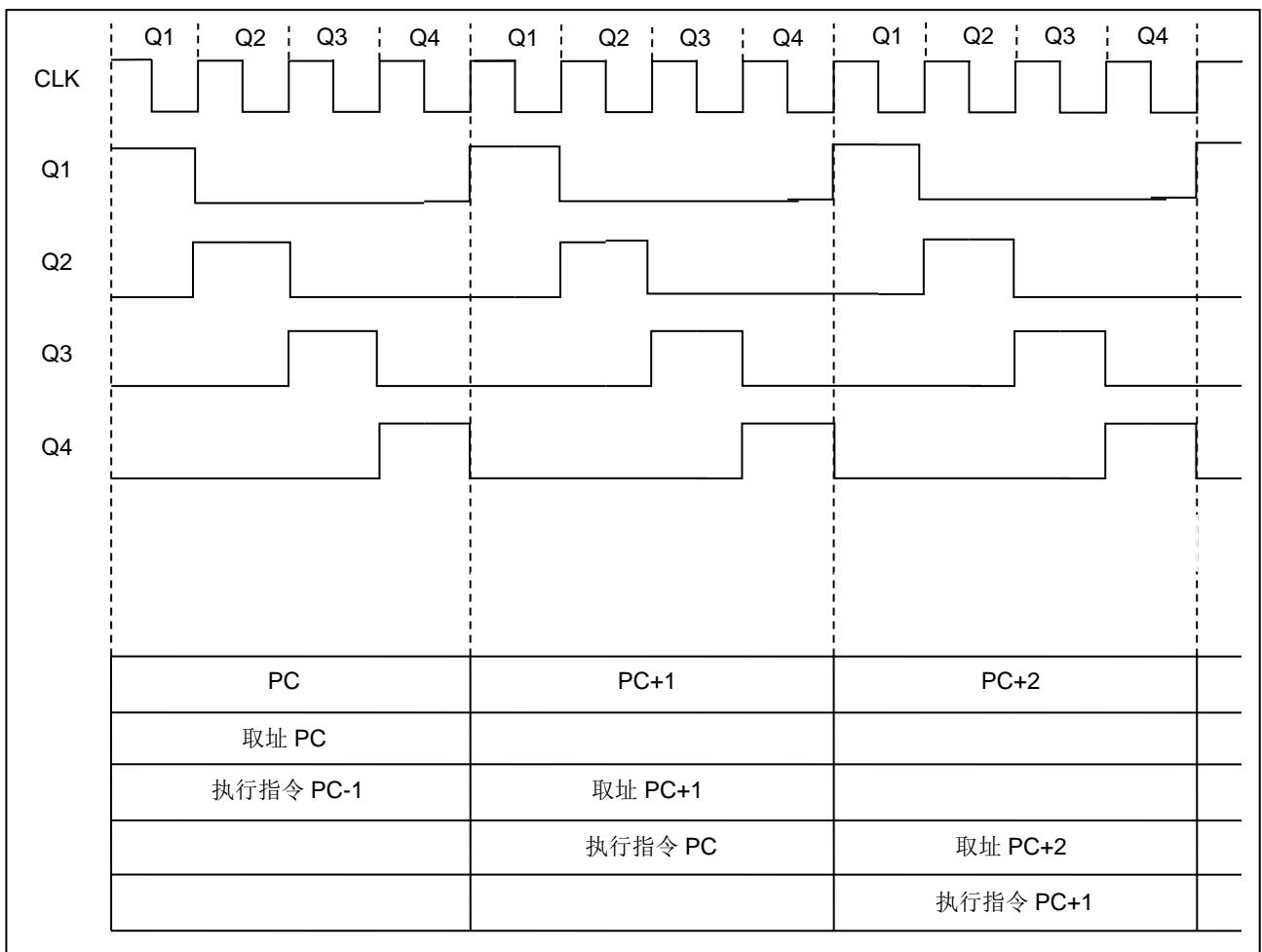


图 3-1：时钟与指令周期时序图

下面列出振荡频率与指令速度的关系

频率	双指令周期	单指令周期
1MHz	8μs	4μs
2MHz	4μs	2μs
4MHz	2μs	1μs
8MHz	1μs	500ns

3.2 系统振荡器

芯片有 1 种振荡方式，内部 RC 振荡。

3.2.1 内部 RC 振荡

芯片的振荡方式为内部 RC 振荡，其振荡频率为 8M 可通过 OSCCON 寄存器设置芯片工作频率。

3.3 起振时间

起振时间（Reset Time）是指从芯片复位到芯片振荡稳定这段时间，其设计值为 18ms。

注：无论芯片是电源上电复位，还是其它原因引起的复位，都会存在这个起振时间。

3.4 振荡器控制寄存器

振荡器控制（OSCCON）寄存器控制系统时钟和频率选择，振荡器调节寄存器 OSCTUNE 可以用软件调节内部振荡频率。

振荡器控制寄存器 OSCCON(8FH)

8FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	---	---
R/W	---	R/W	R/W	R/W	---	---	---	---
复位值	---	1	1	0	---	---	---	---

Bit7 未用，读为 0。

Bit6~Bit4 IRCF<2:0>: 内部振荡器分频选择位。

111= $F_{SYS} = F_{OSC} / 1$

110= $F_{SYS} = F_{OSC} / 2$ (默认)

101= $F_{SYS} = F_{OSC} / 4$

100= $F_{SYS} = F_{OSC} / 8$

011= $F_{SYS} = F_{OSC} / 16$

010= $F_{SYS} = F_{OSC} / 32$

001= $F_{SYS} = F_{OSC} / 64$

000= $F_{SYS} = 32\text{kHz}$ (LFINTOSC)。

Bit3~Bit0 未用。

振荡器调节寄存器 OSCTUNE(90H)

90H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCTUNE	---	---	---	TUN4	TUN3	TUN2	TUN1	TUN0
R/W	---	---	---	R/W	R/W	R/W	R/W	R/W
复位值	---	---	---	0	0	0	0	0

Bit7~Bit5 禁用，读为 0。

Bit4~Bit0 TUN<4:0>: 频率调节位。

01111= 最高频率。

01110=

.

.

.

00001=

00000= 振荡器模块以厂家校准后的频率运行。

11111=

.

.

.

10000= 最低频率。

4. 复位

芯片可用如下 3 种复位方式：

- ◆ 上电复位；
- ◆ 低电压复位；
- ◆ 正常工作下的看门狗溢出复位。

上述任意一种复位发生时，所有的系统寄存器将恢复默认状态，程序停止运行，同时程序计数器 PC 清零，复位结束后程序从复位向量 0000H 开始运行。STATUS 的 TO 和 PD 标志位能够给出系统复位状态的信息，（详见 STATUS 的说明），用户可根据 PD 和 TO 的状态，控制程序运行路径。

任何一种复位情况都需要一定的响应时间，系统提供完善的复位流程以保证复位动作的顺利进行。

4.1 上电复位

上电复位与 LVR 操作密切相关。系统上电的过程呈逐渐上升的曲线形式，需要一定时间才能达到正常电平值。下面给出上电复位的正常时序：

- 上电：系统检测到电源电压上升并等待其稳定；
- 系统初始化：所有的系统寄存器被置为初始值；
- 振荡器开始工作：振荡器开始提供系统时钟；
- 执行程序：上电结束，程序开始运行。

4.2 掉电复位

4.2.1 概述

掉电复位针对外部因素引起的系统电压跌落情形（例如，干扰或外部负载的变化）。

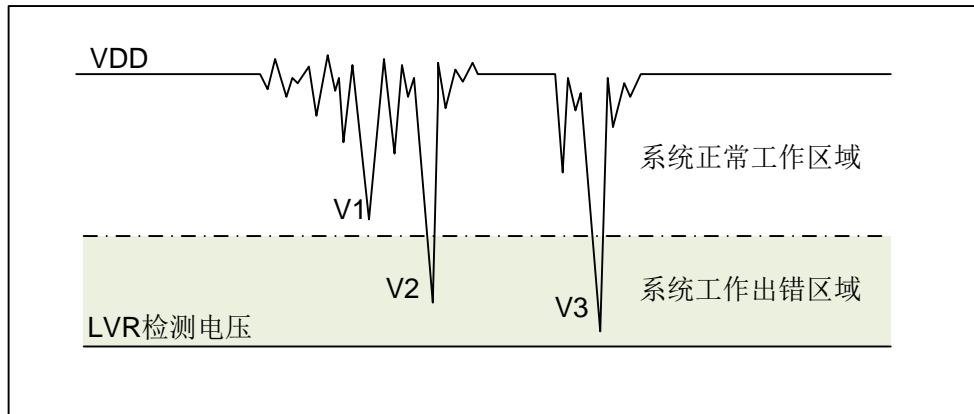


图4-1：掉电复位示意图

上图是一个典型的掉电复位示意图。图中，VDD 受到严重的干扰，电压值降的非常低。虚线以上区域系统正常工作，在虚线以下的区域内，系统进入未知的工作状态，这个区域称作死区。当 VDD 跌至 V1 时，系统仍处于正常状态；当 VDD 跌至 V2 和 V3 时，系统进入死区，则容易导致出错。

以下情况系统可能进入死区：

- DC 运用中：
 - DC 运用中一般都采用电池供电，当电池电压过低或单片机驱动负载时，系统电压可能跌落并进入死区。这时，电源不会进一步下降到 LVD 检测电压，因此系统维持在死区。
- AC 运用中：
 - 系统采用 AC 供电时，DC 电压值受 AC 电源中的噪声影响。当外部负载过高，如驱动马达时，负载动作产生的干扰也影响到 DC 电源。VDD 若由于受到干扰而跌落至最低工作电压以下时，则系统将有可能进入不稳定工作状态。
 - 在 AC 运用中，系统上、下电时间都较长。其中，上电时序保护使得系统正常上电，但下电过程却和 DC 运用中情形类似，AC 电源关断后，VDD 电压在缓慢下降的过程中易进入死区。

如上图所示，系统正常工作电压区域一般高于系统复位电压，同时复位电压由低电压检测（LVR）电平决定。当系统执行速度提高时，系统最低工作电压也相应提高，但由于系统复位电压是固定的，因此在系统最低工作电压与系统复位电压之间就会出现一个电压区域，系统不能正常工作，也不会复位，这个区域即为死区。

4.2.2 掉电复位的改进办法

如何改进系统掉电复位性能，以下给出几点建议：

- ◆ 选择较高的 LVR 电压，有助于复位更可靠；
- ◆ 开启看门狗定时器；
- ◆ 降低系统的工作频率；
- ◆ 增大电压下降斜率。

开启 MCU 的低压侦测功能

芯片内部集成了低压侦测（LVR）功能，当系统电压跌至低于 LVR 电压时，LVR 被触发，系统复位。

看门狗定时器

看门狗定时器用于保证程序正常运行，当系统进入工作死区或者程序运行出错时，看门狗定时器会溢出，系统复位。

降低系统的工作速度

系统工作频率越快，系统最低工作电压越高。从而增大了工作死区的范围，降低系统工作速度就可以降低最低工作电压，从而有效的减小系统工作在死区电压的机率。

增大电压下降斜率

此方法可用于系统工作在 AC 供电的环境，一般 AC 供电系统，系统电压在掉电过程中下降很缓慢，这就造成芯片较长时间工作在死区电压，此时若系统重新上电，芯片工作状态可能出错，建议在芯片电源与地线间加一个放电电阻，以便让 MCU 快速通过死区，进入复位区，避免芯片上电出错可能性。

4.3 看门狗复位

看门狗复位是系统的一种保护设置。在正常状态下，由程序将看门狗定时器清零。若出错，系统处于未知状态，看门狗定时器溢出，此时系统复位。看门狗复位后，系统重启进入正常状态。

看门狗复位的时序如下：

- 看门狗定时器状态：系统检测看门狗定时器是否溢出，若溢出，则系统复位；
- 初始化：所有的系统寄存器被置为默认状态；
- 振荡器开始工作：振荡器开始提供系统时钟；
- 程序：复位结束，程序开始运行。

关于看门狗定时器的应用问题请参看 2.8WDT 应用章节。

5. 休眠模式

5.1 进入休眠模式

执行 STOP 指令可进入休眠模式。如果 WDT 使能，那么：

- ◆ WDT 将被清零并继续运行。
- ◆ STATUS 寄存器中的 PD 位被清零。
- ◆ TO 位被置 1。
- ◆ 关闭振荡器驱动器。
- ◆ I/O 端口保持执行 STOP 指令之前的状态（驱动为高电平、低电平或高阻态）。

在休眠模式下，为了尽量降低电流消耗，所有 I/O 引脚都应该保持为 VDD 或 GND，没有外部电路从 I/O 引脚消耗电流。为了避免输入引脚悬空而引入开关电流，应在外部将高阻输入的 I/O 引脚拉为高电平或低电平。为了将电流消耗降至最低，还应考虑芯片内部上拉电阻的影响。

5.2 从休眠状态唤醒

可以通过下列任一事件将器件从休眠状态唤醒：

1. 看门狗定时器唤醒（WDT 强制使能）；
2. 外设中断。

STATUS 寄存器中的 TO 和 PD 位用于确定器件复位的原因。PD 位在上电时被置 1，而在执行 STOP 指令时被清零。TO 位在发生 WDT 唤醒时被清零。

当执行 STOP 指令时，下一条指令(PC + 1) 被预先取出。如果希望通过中断事件唤醒器件，则必须将相应的中断允许位置 1（允许）。唤醒与 GIE 位的状态无关。如果 GIE 位被清零（禁止），器件将继续执行 STOP 指令之后的指令。如果 GIE 位被置 1（允许），器件执行 STOP 指令之后的指令，然后跳转到中断地址（0004h）处执行代码。如果不想执行 STOP 指令之后的指令，用户应该在 STOP 指令后面放置一条 NOP 指令。器件从休眠状态唤醒时，WDT 都将被清零，而与唤醒的原因无关。

5.3 使用中断唤醒

当禁止全局中断（GIE 被清零）时，并且有任一中断源将其中断允许位和中断标志位置 1，将会发生下列事件之一：

- 如果在执行 STOP 指令之前产生了中断，那么 STOP 指令将被作为一条 NOP 指令执行。因此，WDT 及其预分频器和后分频器（如果使能）将不会被清零，并且 TO 位将不会被置 1，同时 PD 也不会被清零。
- 如果在执行 STOP 指令期间或之后产生了中断，那么器件将被立即从休眠模式唤醒。STOP 指令将在唤醒之前执行完毕。因此，WDT 及其预分频器和后分频器（如果使能）将被清零，并且 TO 位将被置 1，同时 PD 也将被清零。即使在执行 STOP 指令之前检查到标志位为 0，它也可能在 STOP 指令执行完毕之前被置 1。要确定是否执行了 STOP 指令，可以测试 PD 位。如果 PD 位置 1，则说明 STOP 指令被作为一条 NOP 指令执行了。在执行 STOP 指令之前，必须先执行一条 CLRWDT 指令，来确保将 WDT 清零。

5.4 休眠模式应用举例

系统在进入休眠模式之前，若用户需要获得较小的休眠电流，请先确认所有 I/O 的状态，若用户方案中存在悬空的 I/O 口，把所有悬空口都设置为输出口，确保每一个 I/O 都有一个固定的状态，以避免 I/O 为输入状态时，口线电平处于不定态而增大休眠电流；关断 AD 等其它外设模块；根据实际方案的功能需求可禁止 WDT 功能来减小休眠电流。

例：进入休眠的处理程序

```

SLEEP_MODE:
    CLR          INTCON          ;关断中断使能
    LDIA        B'00000000'
    LD          TRISA,A
    LD          TRISB,A          ;所有 I/O 设置为输出口
    LD          TRISC,A
    LD          TRISD,A
    ...
    LDIA        0A5H
    LD          SP_FLAG,A        ;置休眠状态记忆寄存器(用户自定义)
    CLRWDT      ;清零 WDT
    STOP        ;执行 STOP 指令
    
```

5.5 休眠模式唤醒时间

当 MCU 从休眠态被唤醒时，需要等待一个振荡稳定时间（ResetTime），这个时间标称值为 18ms。

6. I/O 端口

芯片有四个 I/O 端口：PORTA、PORTB、PORTC、PORTD（最多 26 个 I/O）。可读写端口数据寄存器可直接存取这些端口。

端口	位	管脚描述	I/O
PORTA	0	施密特触发输入，推挽式输出，编程数据输入/输出	I/O
	1	施密特触发输入，推挽式输出	I/O
	2	施密特触发输入，推挽式输出，编程时钟输入	I/O
	3	施密特触发输入，推挽式输出	I/O
	4	施密特触发输入，推挽式输出	I/O
	7	施密特触发输入，推挽式输出	I/O
PORTB	0	施密特触发输入，推挽式输出，触摸按键口	I/O
	1	施密特触发输入，推挽式输出，触摸按键口	I/O
	2	施密特触发输入，推挽式输出，触摸按键口	I/O
	3	施密特触发输入，推挽式输出，触摸按键口	I/O
	4	施密特触发输入，推挽式输出，触摸按键口	I/O
	5	施密特触发输入，推挽式输出，触摸按键口	I/O
	6	施密特触发输入，推挽式输出，触摸按键口	I/O
	7	施密特触发输入，推挽式输出，触摸按键口	I/O
PORTC	0	施密特触发输入，推挽式输出，触摸按键口	I/O
	1	施密特触发输入，推挽式输出，触摸按键口	I/O
	2	施密特触发输入，推挽式输出，触摸按键口	I/O
	3	施密特触发输入，推挽式输出，触摸按键口	I/O
PORTD	0	施密特触发输入，推挽式输出，AN0	I/O
	1	施密特触发输入，推挽式输出，AN1	I/O
	2	施密特触发输入，推挽式输出，AN2	I/O
	3	施密特触发输入，推挽式输出，AN3	I/O
	4	施密特触发输入，推挽式输出，AN4	I/O
	5	施密特触发输入，推挽式输出，AN5	I/O
	6	施密特触发输入，推挽式输出，AN6	I/O
	7	施密特触发输入，推挽式输出，AN7	I/O

表 6-1：端口配置总概

6.1 I/O 口结构图

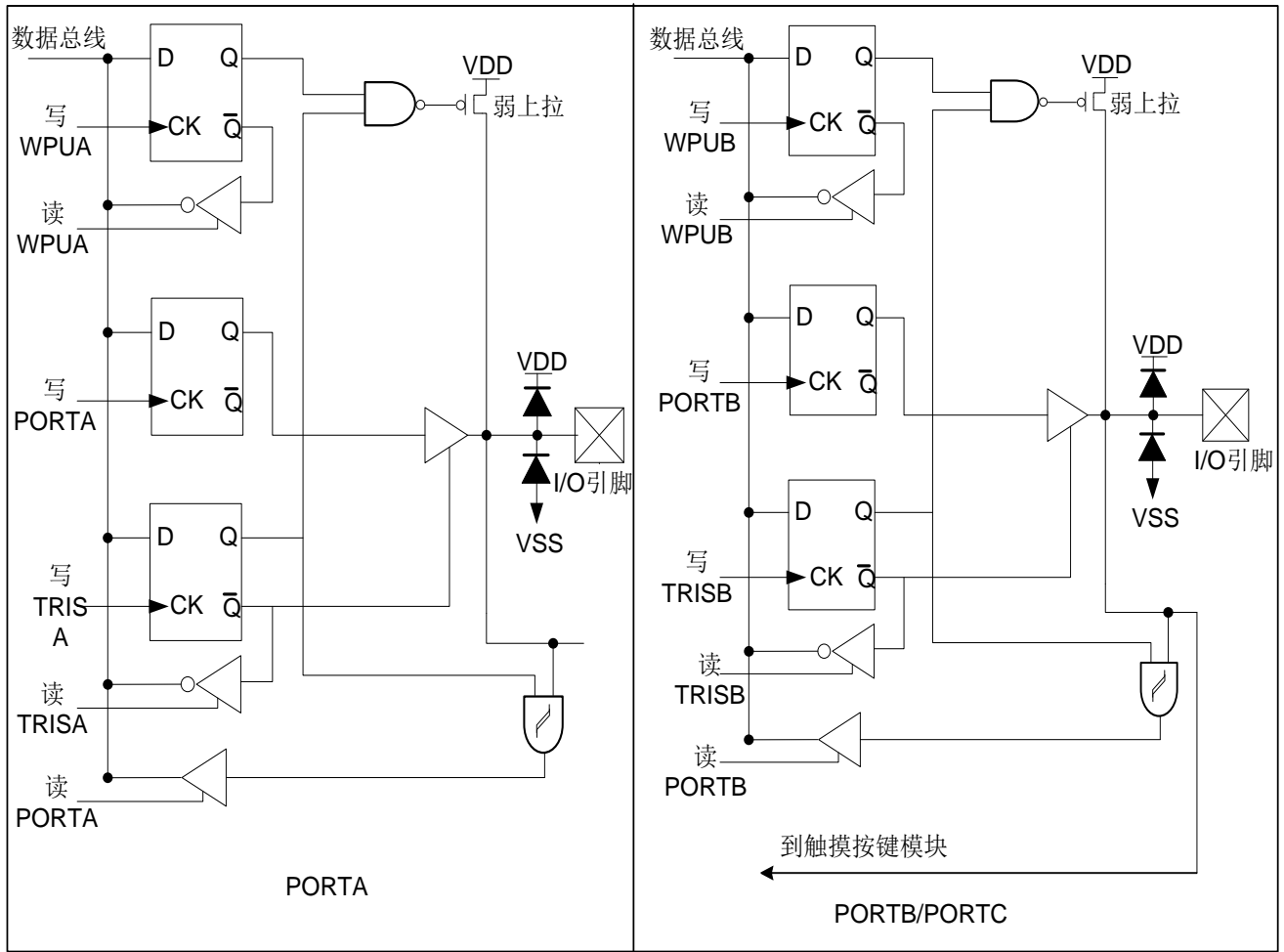


图 6-1: I/O 结构图

6.2 PORTA

6.2.1 PORTA 数据及方向控制

PORTA 是 6Bit 宽的双向端口。它所对应的数据方向寄存器是 TRISA。将 TRISA 的一个位置 1 (=1) 可以将相应的引脚配置为输入。清零 TRISA 的一个位 (=0) 可将相应的 PORTA 引脚配置为输出。

读 PORTA 寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读—修改—写操作。因此，写一个端口就意味着先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。即使在 PORTA 引脚用作模拟输入时，TRISA 寄存器仍然控制 PORTA 引脚的方向。当将 PORTA 引脚用作模拟输入时，用户必须确保 TRISA 寄存器中的位保持为置 1 状态。配置为模拟输入的 I/O 引脚总是读为 0。

与 PORTA 口相关寄存器有 PORTA、TRISA、WPUA 等。

PORTA 数据寄存器 PORTA(05H)

05H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTA	RA7	---	---	RA4	RA3	RA2	RA1	RA0
R/W	R/W	---	---	R/W	R/W	R/W	R/W	R/W
复位值	X	---	---	X	X	X	X	X

Bit7 PORTA<7> PORTA I/O 引脚位；
 1= 端口引脚电平>V_{IH}；
 0= 端口引脚电平<V_{IL}。

Bit6~ Bit5 未用。

Bit7~Bit0 PORTA<4:0>: PORTA I/O 引脚位；
 1= 端口引脚电平>V_{IH}；
 0= 端口引脚电平<V_{IL}。

PORTA 方向寄存器 TRISA(85H)

85H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISA	TRISA7	---	---	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
R/W	R/W	---	---	R/W	R/W	R/W	R/W	R/W
复位值	1	---	---	1	1	1	1	1

Bit7 TRISA<7>: PORTA 三态控制位；
 1= PORTA 引脚被配置为输入（三态）；
 0= PORTA 引脚被配置为输出。

Bit6~ Bit5 未用。

Bit7~Bit0 TRISA<4:0>: PORTA 三态控制位；
 1= PORTA 引脚被配置为输入（三态）；
 0= PORTA 引脚被配置为输出。

例：PORTA 口处理程序

LDIA	B'1110000'	;设置PORTA<4:0>为输出口，PORTA<7>为输入口
LD	TRISA,A	
LDIA	03H	;PORTA<1:0>输出高电平，PORTA<4:2>输出低电平
LD	PORTA,A	;由于PORTA<7>为输入口，所以赋0或1都没影响

6.2.2 PORTA 上拉电阻

每个 PORTA 引脚都有可单独配置的内部弱上拉。控制位 WPUA<7:0>使能或禁止每个弱上拉。当将端口引脚配置为输出时，其弱上拉会自动切断。

PORTA 上拉电阻寄存器 WPUA(108H)

108H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUA	WPUA7	---	---	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
R/W	R/W	---	---	R/W	R/W	R/W	R/W	R/W
复位值	0	---	---	0	0	0	0	0

Bit7~Bit0 WPUA<7:0>: 弱上拉寄存器位;
 1= 使能上拉;
 0= 禁止上拉。

注：如果引脚被配置为输出，将自动禁止弱上拉。

6.3 PORTB

6.3.1 PORTB 数据及方向

PORTB 是一个 8Bit 宽的双向端口。对应的数据方向寄存器为 PORTB。将 TRISB 中的某个位置 1 (=1) 可以使对应的 PORTB 引脚作为输入引脚。将 TRISB 中的某个位清零 (=0) 将使对应的 PORTB 引脚作为输出引脚。

读 PORTB 寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读—修改—写操作。因此，写一个端口就意味着先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。即使在 PORTB 引脚用作模拟输入时，TRISB 寄存器仍然控制 PORTB 引脚的方向。当将 PORTB 引脚用作模拟输入时，用户必须确保 TRISB 寄存器中的位保持为置 1 状态。配置为模拟输入的 I/O 引脚总是读为 0。

与 PORTB 口相关寄存器有 PORTB、TRISB、WPUB 等。

PORTB 数据寄存器 PORTB(06H)

06H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0 PORTB<7:0>: PORTB I/O 引脚位;
 1= 端口引脚电平>V_{IH};
 0= 端口引脚电平<V_{IL}。

PORTB 方向寄存器 TRISB(86H)

86H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISB								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

Bit7~Bit0 TRISB<7:0>: PORTB 三态控制位;
 1= PORTB 引脚被配置为输入 (三态);
 0= PORTB 引脚被配置为输出。

例: PORTB 口处理程序

CLR	PORTB	;清数据寄存器
LDIA	B'00110000'	;设置 PORTB<5:4>为输入口, 其余为输出口
LD	TRISB,A	

6.3.2 PORTB 上拉电阻

每个 PORTB 引脚都有可单独配置的内部弱上拉。控制位 WPUB<7:0>使能或禁止每个弱上拉。当将端口引脚配置为输出时，其弱上拉会自动切断。

PORTB 上拉电阻寄存器 WPUB(107H)

107H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit 7~Bit0 WPUB<7:0>: 弱上拉寄存器位;
 1= 使能上拉;
 0= 禁止上拉。

注：如果引脚被配置为输出，将自动禁止弱上拉。

6.4 PORTC

6.4.1 PORTC 数据及方向

PORTC 是一个 4Bit 宽的双向端口。对应的数据方向寄存器为 TRISC。将 TRISC 中的某个位置 1 (=1) 可以使对应的 PORTC 引脚作为输入引脚。将 TRISC 中的某个位清零 (= 0) 将使对应的 PORTC 引脚作为输出引脚。

读 PORTC 寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读—修改—写操作。因此，写一个端口就意味着先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。

PORTC 数据寄存器 PORTC(07H)

07H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTC	---	---	---	---	RC3	RC2	RC1	RC0
R/W	---	---	---	---	R/W	R/W	R/W	R/W
复位值	---	---	---	---	X	X	X	X

Bit7~Bit4 未用。
 Bit3~Bit0 PORTC<3:0>: PORTC I/O 引脚位;
 1 = 端口引脚电平 > V_{IH};
 0 = 端口引脚电平 < V_{IL}。

PORTC 方向寄存器 TRISC(87H)

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISC	---	---	---	---				
R/W	---	---	---	---	R/W	R/W	R/W	R/W
复位值	---	---	---	---	1	1	1	1

Bit7~Bit4 未用。
 Bit3~Bit0 TRISC<3:0>: PORTC 三态控制位;
 1= PORTC 引脚被配置为输入 (三态);
 0= PORTC 引脚被配置为输出。

例: PORTC 口处理程序

CLR	PORTC	;清数据寄存器
LDIA	B'00000011'	;设置 PORTC<3:2>为输出口, PORTC<1:0>为输入口
LD	TRISC,A	

6.4.2 PORTC 上拉电阻

每个 PORTC 引脚都有可单独配置的内部弱上拉。控制位 WPUC<3:0>使能或禁止每个弱上拉。当将端口引脚配置为输出时，其弱上拉会自动切断。

PORTC 上拉电阻寄存器 WPUC(09H)

09H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUC	---	---	---	---	WPUC3	WPUC2	WPUC1	WPUC0
R/W	---	---	---	---	R/W	R/W	R/W	R/W
复位值	---	---	---	---	0	0	0	0

Bit7~Bit4 未用。

Bit3~Bit0 WPUC<3:0>: 弱上拉寄存器位;

1= 使能上拉;

0= 禁止上拉。

注：如果引脚被配置为输出，将自动禁止弱上拉。

6.5 PORTD

6.5.1 PORTD 数据及方向

PORTD 是一个 8Bit 宽的双向端口。对应的数据方向寄存器为 TRISD。将 TRISD 中的某个位置 1 (=1) 可以使对应的 PORTD 引脚作为输入引脚。将 TRISD 中的某个位清零 (= 0) 将使对应的 PORTD 引脚作为输出引脚。

读 PORTD 寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读—修改—写操作。因此，写一个端口就意味着先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。

PORTD 数据寄存器 PORTD(08H)

08H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0 PORTD<7:0>: PORTD I/O 引脚位;
 1= 端口引脚电平>V_{IH};
 0= 端口引脚电平<V_{IL}。

PORTD 方向寄存器 TRISD(88H)

88H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISD	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

Bit7~Bit0 TRISD<7:0>: PORTD 三态控制位;
 1= PORTD 引脚被配置为输入 (三态);
 0= PORTD 引脚被配置为输出。

例: PORTD 口处理程序

CLR	PORTD	;清数据寄存器
LDIA	B'01110000'	;设置 PORTD<3:0>为输出口, PORTD<6:4>为输入口
LD	TRISD,A	

6.5.2 PORTD 上拉电阻

每个 PORTD 引脚都有可单独配置的内部弱上拉。控制位 WPUD<7:0>使能或禁止每个弱上拉。当将端口引脚配置为输出时，其弱上拉会自动切断。

PORTD 上拉电阻寄存器 WPUD(89H)

89H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUD	WPUD7	WPUD6	WPUD5	WPUD4	WPUD3	WPUD2	WPUD1	WPUD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 WPUD<7:0>: 弱上拉寄存器位;

1= 使能上拉;

0= 禁止上拉。

注：如果引脚被配置为输出，将自动禁止弱上拉。

6.6 I/O 使用

6.6.1 写 I/O 口

芯片的 I/O 口寄存器和一般通用寄存器一样，可以通过数据传输指令，位操作指令等进行写操作。

例：写 I/O 口程序

LD	PORTA,A	;ACC 值赋给 PORTA 口
CLRB	PORTB,1	;PORTB.1 口清零
CLR	PORTC	;PORTC 口清零
SET	PORTA	;PORTA 所有输出口置 1
SETB	PORTB,1	;PORTB.1 口置 1

6.6.2 读 I/O 口

例：读 I/O 口程序

LD	A,PORTA	;PORTA 的值赋给 ACC
SNZB	PORTA,1	;判断 PORTA,1 口是否为 1，为 1 跳过下一条语句
SZB	PORTA,1	;判断 PORTA,1 口是否为 0，为 0 跳过下一条语句

注：当用户读一个 I/O 口状态时，若此 I/O 口为输入口，则用户读回的数据将是此口线外部电平的状态，若此 I/O 口为输出口那么读出的值将会是此口线内部输出寄存器的数据。

6.7 I/O 口使用注意事项

在操作 I/O 口时，应注意以下几个方面：

1. 当 I/O 从输出转换为输入时，要等待几个指令周期的时间，以便 I/O 口状态稳定。
2. 若使用内部上拉电阻，那么当 I/O 从输出转换为输入时，内部电平的稳定时间，与接在 I/O 口上的电容有关，用户应根据实际情况，设置等待时间，以防止 I/O 口误扫描电平。
3. 当 I/O 口为输入口时，其输入电平应在“VDD+0.7V”与“GND-0.7V”之间。若输入口电压不在此范围内可采用如下图所示方法。

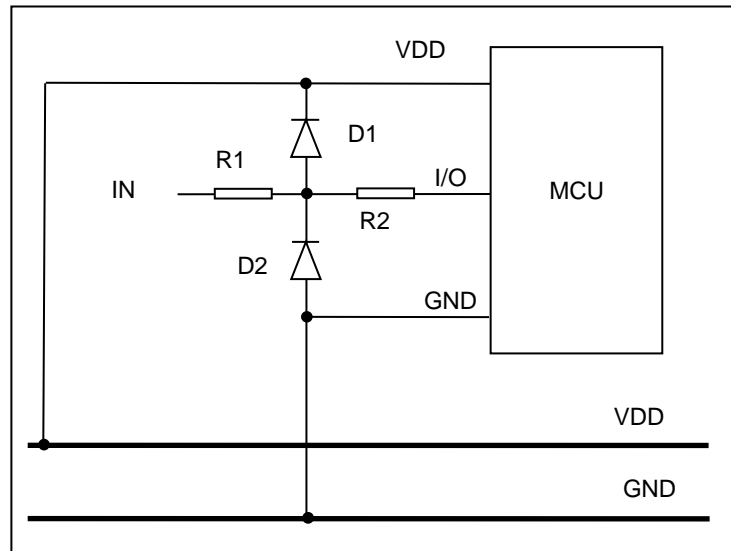


图 6-2: 输入电压不在规定范围内采用电路

4. 若在 I/O 口所在线串入较长的连接线，请在靠近芯片 I/O 的地方加上限流电阻以增强 MCU 抗 EMC 能力。

7. 中断

7.1 中断概述

芯片具有以下多种中断源：

- ◆ TIMER0 溢出中断；
- ◆ TIMER2 匹配中断；
- ◆ INT 中断；
- ◆ A/D 中断；
- ◆ MSSP 中断；
- ◆ EEPROM 写操作中断。

中断控制寄存器 (INTCON) 和外设中断请求寄存器 (PIR1、PIR2) 在各自的标志位中记录各种中断请求。INTCON 寄存器还包括各个中断允许位和全局中断允许位。

全局中断允许位 GIE (INTCON<7>) 在置 1 时允许所有未屏蔽的中断，而在清零时，禁止所有中断。可以通过 INTCON、PIE1、PIE2 寄存器中相应的允许位来禁止各个中断。复位时 GIE 被清零。

执行“从中断返回”指令 RETI 将退出中断服务程序并将 GIE 位置 1，从而重新允许未屏蔽的中断。

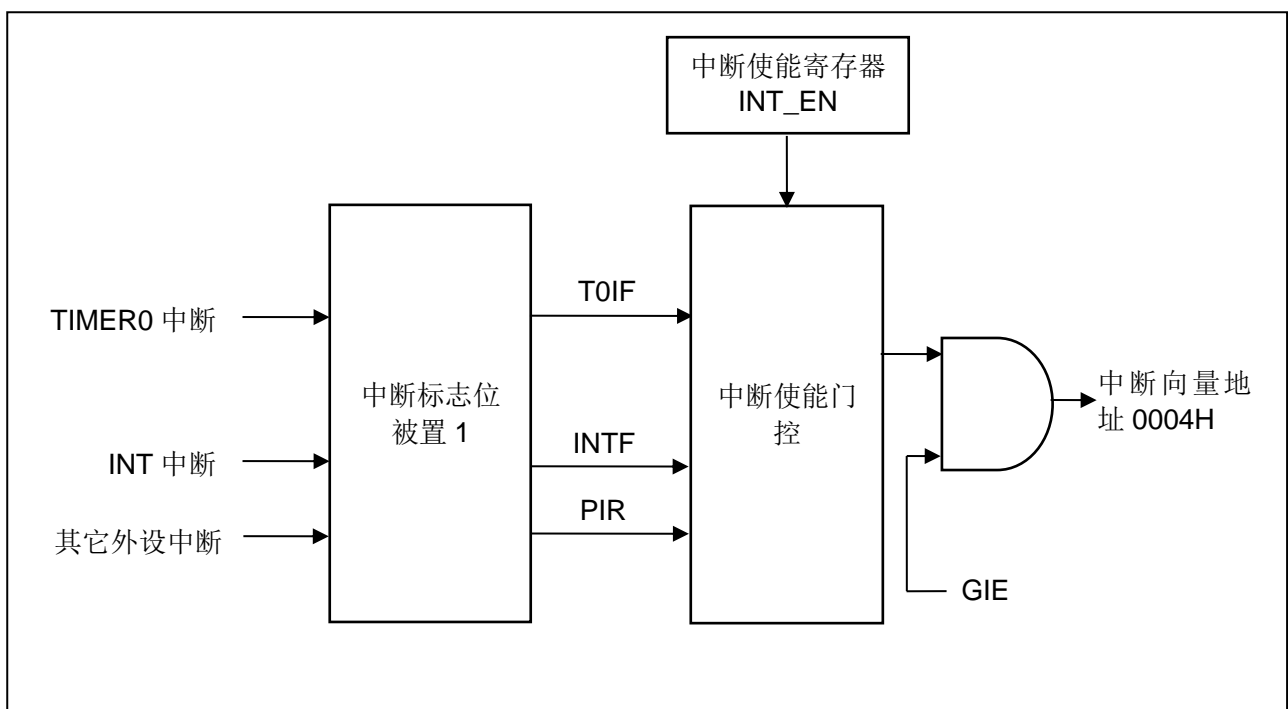


图 7-1：中断执行示意图

7.2 中断控制寄存器

7.2.1 中断控制寄存器

中断控制寄存器 INTCON 是可读写的寄存器，包含 TMR0 寄存器溢出、外部中断等的允许和标志位。

当有中断条件产生时，无论对应的中断允许位或（INTCON 寄存器中的）全局允许位 GIE 的状态如何，中断标志位都将置 1。用户软件应在允许一个中断之前，确保先将相应的中断标志位清零。

中断控制寄存器 INTCON（0BH）

0BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCON	GIE	PEIE	TOIE	INTE	---	TOIF	INTF	---
R/W	R/W	R/W	R/W	R/W	---	R/W	R/W	---
复位值	0	0	0	0	---	0	0	---

Bit7	GIE: 全局中断允许位; 1= 允许所有未被屏蔽的中断; 0= 禁止所有中断。
Bit6	PEIE: 外设中断允许位; 1= 允许所有未被屏蔽的外设中断; 0= 禁止所有外设中断。
Bit5	TOIE: TIMER0溢出中断允许位; 1= 允许TIMER0中断; 0= 禁止TIMER0中断。
Bit4	INTE: INT外部中断允许位; 1= 允许INT外部中断; 0= 禁止INT外部中断。
Bit3	未用。
Bit2	TOIF: TIMER0溢出中断标志位(1); 1= TMR0寄存器已经溢出（必须由软件清零）; 0= TMR0寄存器未发生溢出。
Bit1	INTF: INT外部中断标志位; 1= 发生INT外部中断（必须由软件清零）; 0= 未发生INT外部中断。
Bit0	未用。

注：TOIF 位在 TMR0 计满归 0 时置 1。复位不会使 TMR0 发生改变，应在将 TOIF 位清零前对其进行初始化。

7.2.2 外设中断允许寄存器

外设中断允许寄存器有 PIE1 和 PIE2, 在允许任何外设中断前, 必须先将 INTCON 寄存器的 PEIE 位置 1。

外设中断允许寄存器 PIE1(8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	---	ADIE	TKIE	---	---	BCLIE	TMR2IE	---
R/W	---	R/W	R/W	---	---	R/W	R/W	---
复位值	---	0	0	---	---	0	0	---

- Bit7 未用。
- Bit6 ADIE: A/D转换器 (ADC) 中断允许位;
1= 允许ADC中断;
0= 禁止ADC中断。
- Bit5 TKIE: 触摸按键中断允许位;
1= 允许触摸按键中断;
0= 禁止触摸按键中断。
- Bit4~ Bit3 未用。
- Bit2 BCLIE: 主同步串行端口 (MSSP) 总线中断允许位;
1= 允许总线中断;
0= 禁止总线中断。
- Bit1 TMR2IE: TIMER2与PR2匹配中断允许位;
1= 允许TMR2与PR2匹配中断;
0= 禁止TMR2与PR2匹配中断。
- Bit0 未用。

外设中断允许寄存器 PIE2(8DH)

8DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE2	---	---	---	EEIE	---	---	---	SSPIE
R/W	---	---	---	R/W	---	---	---	R/W
复位值	---	---	---	0	---	---	---	0

- Bit7~ Bit5 未用。
- Bit4 EEIE: EEPROM写操作中中断允许位;
1= 允许EEPROM写操作中中断;
0= 禁止EEPROM写操作中中断。
- Bit3~Bit1 未用。
- Bit 0 SSPIE: MSSP 中断允许位;
1= 允许 MSSP 中断;
0= 禁止 MSSP 中断。

7.2.3 外设中断请求寄存器

外设中断请求寄存器为 PIR1 和 PIR2。当有中断条件产生时，无论对应的中断允许位或全局允许位 GIE 的状态如何，中断标志位都将置 1。用户软件应在允许一个中断之前，确保先将相应的中断标志位清零。

外设中断请求寄存器 PIR1(0CH)

0CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR1	---	ADIF	TKIF	---	---	BCLIF	TMR2IF	---
R/W	---	R/W	R/W	---	---	R/W	R/W	---
复位值	---	0	0	---	---	0	0	---

Bit7	未用。
Bit6	ADIF: A/D转换器中断标志位; 1= A/D转换完成（必须由软件清零）; 0= A/D转换未完成或尚未启动。
Bit5	TKIF: 触摸中断标志位; 1= 触摸中断产生; 0= 触摸中断未产生。
Bit4~ Bit3	未用。
Bit2	BCLIF: 总线冲突中断标志位; 1= 当配置为I ² C主控模式时，MSSP中发生了总线冲突; 0= 未发生总线冲突。
Bit1	TMR2IF: TIMER2与PR2匹配中断标志位; 1= 发生了TIMER2与PR2匹配（必须由软件清零）; 0= TIMER2与PR2不匹配。
Bit0	未用。

外设中断请求寄存器 PIR2(0DH)

0DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR2	---	---	---	EEIF	---	---	---	SSPIF
R/W	---	---	---	R/W	---	---	---	R/W
复位值	---	---	---	0	---	---	---	0

Bit7~ Bit5 未用。

Bit4 EEIF: EE写操作中中断标志位;
 1= 写操作完成(必须由软件清零);
 0= 写操作未完成或尚未启动。

Bit3~ Bit1 未用。

Bit0 SSPIF: 主同步串行端口(MSSP)中断标志位。
 1 = 产生了MSSP中断条件, 在从中断服务程序返回前必须由软件清零。使该位置1的条件有:
 - SPI;
 - 发生发送/接收;
 - I²C从动/主控;
 - 发生发送/接收;
 - I²C主控;
 - 发生的启动条件由MSSP模块完成;
 - 发生的停止条件由MSSP模块完成;
 - 发生的重新启动条件由MSSP模块完成;
 - 发生的应答条件由MSSP模块完成;
 - 当MSSP模块空闲时发生启动条件(多主机系统);
 - 当MSSP模块空闲时发生停止条件(多主机系统)。
 0 = 没有产生MSSP中断条件

7.3 中断现场的保护方法

有中断请求发生并被响应后，程序转至 0004H 执行中断子程序。响应中断之前，必须保存 ACC、STATUS 的内容。芯片没有提供专用的入栈保存和出栈恢复指令，用户需自己保护 ACC 和 STATUS 的内容，以避免中断结束后可能的程序运行错误。

例：对 ACC 与 STATUS 进行入栈保护

```

                ORG    0000H
                JP     START      ;用户程序起始地址
                ORG    0004H
                JP     INT_SERVICE ;中断服务程序
                ORG    0008H

START:
    ...
    ...

INT_SERVICE:
    PUSH:
                ;中断服务程序入口，保存 ACC 及 STATUS
                LD     ACC_BAK,A   ;保存 ACC 的值（ACC_BAK 需自定义）
                SWAPA STATUS
                LD     STATUS_BAK,A ;保存 STATUS 的值，（STATUS_BAK 需自定义）
                ...
                ...

    POP:
                ;中断服务程序出口，还原 ACC 及 STATUS
                SWAPA STATUS_BAK
                LD     STATUS,A    ;还原 STATUS 的值
                SWAPR ACC_BAK      ;还原 ACC 的值
                SWAPA ACC_BAK
                RETI
    
```

7.4 中断的优先级，及多中断嵌套

芯片的各个中断的优先级是平等的，当一个中断正在进行的时候，不会响应另外一个中断，只有执行“RETI”指令后，才能响应下一个中断。

多个中断同时发生时，MCU 没有预置的中断优先级。首先，必须预先设定好各中断的优先权；其次，利用中断使能位和中断控制位，控制系统是否响应该中断。在程序中，必须对中断控制位和中断请求标志进行检测。

8. 定时计数器 TIMER0

8.1 定时计数器 TIMER0 概述

TIMER0 由如下功能组成：

- ◆ 8 位定时器/计数器寄存器（TMR0）；
- ◆ 8 位预分频器（与看门狗定时器共用）；
- ◆ 可编程内部或外部时钟源；
- ◆ 可编程外部时钟边沿选择；
- ◆ 溢出中断。

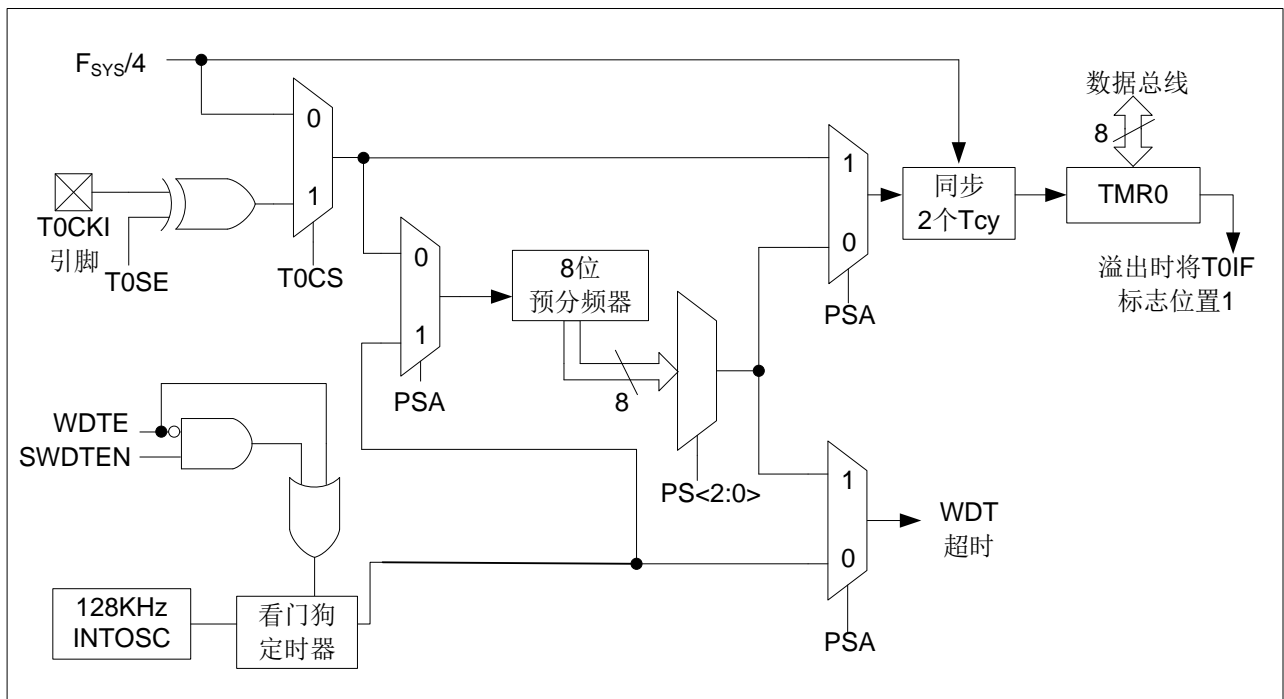


图 8-1: TIMER0/WDT 结构图

注：

1. T0SE、T0CS、PSA、PS<2:0>为 OPTION_REG 寄存器中的位。
2. SWDTEN 为 WDTCON 寄存器中的位。
3. WDTE 位在配置字寄存器中。

8.2 TIMER0 的工作原理

TIMER0 模块既可用作 8 位定时器也可用作 8 位计数器。

8.2.1 8 位定时器模式

用作定时器时，TIMER0 模块将在每个指令周期递增（不带预分频器）。通过将 OPTION_REG 寄存器的 T0CS 位清 0 可选择定时器模式。如果对 TMR0 寄存器执行写操作，则在接下来的两个指令周期将禁止递增。可调整写入 TMR0 寄存器的值，使得在写入 TMR0 时计入两个指令周期的延时。

8.2.2 8 位计数器模式

用作计数器时，TIMER0 模块将在 T0CKI 引脚的每个上升沿或下降沿递增。递增的边沿取决于 OPTION_REG 寄存器的 T0SE 位。通过将 OPTION_REG 寄存器的 T0CS 位置 1 可选择计数器模式。

8.2.3 软件可编程预分频器

TIMER0 和看门狗定时器（WDT）共用一个软件可编程预分频器，但不能同时使用。预分频器的分配由 OPTION_REG 寄存器的 PSA 位控制。要将预分频器分配给 TIMER0，PSA 位必须清 0。

TIMER0 模块具有 8 种预分频比选择，范围为 1:2 至 1:256。可通过 OPTION_REG 寄存器的 PS<2:0>位选择预分频比。要使 TIMER0 模块具有 1:1 的预分频比，必须将预分频器分配给 WDT 模块。

预分频器不可读写。当预分频器分配给 TIMER0 模块时，所有写入 TMR0 寄存器的指令都将使预分频器清零。当预分频器分配给 WDT 时，CLRWDT 指令将同时清零预分频器和 WDT。

8.2.4 在 TIMER0 和 WDT 模块间切换预分频器

将预分频器分配给 TIMER0 或 WDT 后，在切换预分频比时可能会产生无意的器件复位。要将预分频器从分配给 TIMER0 改为分配给 WDT 模块时，必须执行如下指令序列。

更改预分频器（TMR0-WDT）

CLR	TMR0	;TMR0 清零
CLRWDT		;WDT 清零
LDIA	B'00xx1111'	;必要步骤，必须执行
LD	OPTION_REG,A	;必要步骤，必须执行
LDIA	B'00xx1xxx'	;设置新的预分频器
LD	OPTION_REG,A	

要将预分频器从分配给 WDT 改为分配给 TIMER0 模块，必须执行以下指令序列。

更改预分频器（WDT-TMR0）

CLRWDT		;WDT 清零
LDIA	B'00xx0xxx'	;设置新的预分频器
LD	OPTION_REG,A	

8.2.5 TIMER0 中断

当 TMR0 寄存器从 FFh 溢出至 00h 时，产生 TIMER0 中断。每次 TMR0 寄存器溢出时，不论是否允许 TIMER0 中断，INTCON 寄存器的 TOIF 中断标志位都会置 1。TOIF 位必须在软件中清零。TIMER0 中断允许位是 INTCON 寄存器的 TOIE 位。

注：由于在休眠状态下定时器是关闭的，所以 TIMER0 中断无法唤醒处理器。

8.3 与 TIMER0 相关寄存器

有两个寄存器与 TIMER0 相关，8 位定时器/计数器 (TMR0) 和 8 位可编程控制寄存器 (OPTION_REG)。

TMR0 为一个 8 位可读写的定时/计数器，OPTION_REG 为一个 8 位只写寄存器，用户可改变 OPTION_REG 的值，来改变 TIMER0 的工作模式等。请参看 2.6 关于预分频寄存器 (OPTION_REG) 的应用。

8 位定时器/计数器 TMR0 (01H)

01H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR0								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

OPTION_REG 寄存器 (81H)

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	---	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
读写	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	1	1	1	1	1	1	1

Bit7	未用。		
Bit6	INTEDG: 中断边沿选择位;		
	1= INT引脚的下降沿触发中断;		
	0= INT引脚的上升沿触发中断。		
Bit5	T0CS: TIMER0 时钟源选择位;		
	1= T0CKI 引脚上的跳变沿;		
	0= 内部指令周期时钟 ($F_{osc}/4$)。		
Bit4	T0SE: TIMER0 时钟源边沿选择位;		
	1= 在 T0CKI 引脚信号从高电平跳变到低电平时递增;		
	0= 在 T0CKI 引脚信号从低电平跳变到高电平时递增。		
Bit3	PSA: 预分频器分配位;		
	1= 预分频器分配给 WDT;		
	0= 预分频器分配给 TIMER0 模块。		
Bit2~Bit0	PS2~PS0: 预分配参数配置位。		
		PS2-PS1-PS0	TMR0 分频比
		000	1:2
		001	1:4
		010	1:8
		011	1:16
		100	1:32
		101	1:64
		110	1:128
		111	1:256
			WDT 分频比
			1:1
			1:2
			1:4
			1:8
			1:16
			1:32
			1:64
			1:128

9. 定时计数器 TIMER2

9.1 TIMER2 概述

TIMER2 模块是一个 8 位定时器/计数器，具有以下特性：

- ◆ 8 位定时器寄存器 (TMR2)；
- ◆ 8 位周期寄存器 (PR2)；
- ◆ TMR2 与 PR2 匹配时中断；
- ◆ 软件可编程预分频比 (1:1、1:4 和 1:16)；
- ◆ 软件可编程后分频比 (1:1 至 1:16)。

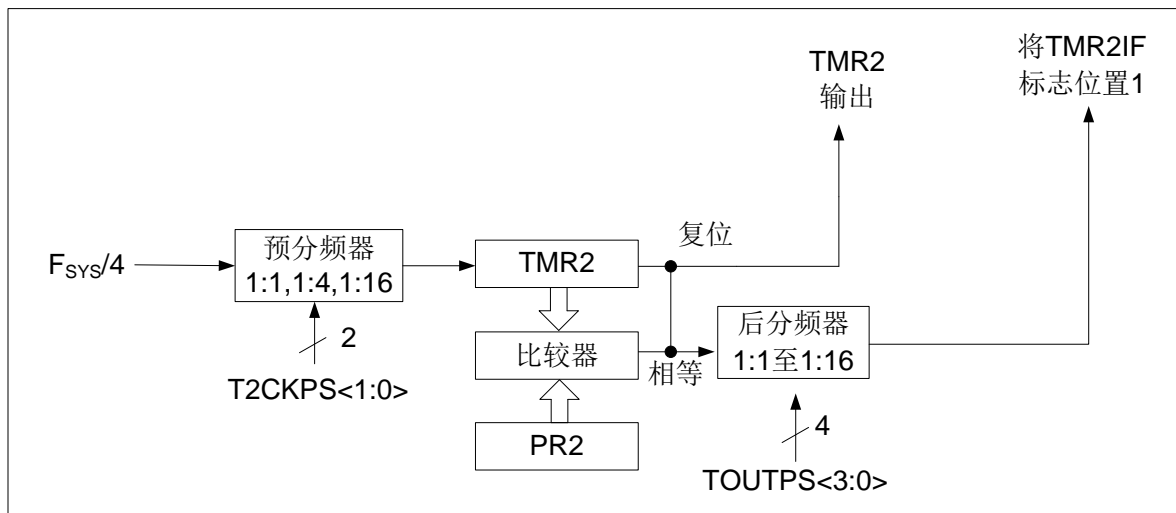


图 9-1: TIMER2 框图

9.2 TIMER2 的工作原理

TIMER2 模块的输入时钟是系统指令时钟 ($F_{SYS}/4$)。时钟被输入到 TIMER2 预分频器, 有如下几种分频比可供选择: 1:1、1:4 或 1:16。预分频器的输出随后用于使 TMR2 寄存器递增。

持续将 TMR2 和 PR2 的值做比较以确定它们何时匹配。TMR2 将从 00h 开始递增直至与 PR2 中的值匹配。匹配发生时, 会发生以下两个事件:

- TMR2 在下一递增周期被复位为 00h;
- TIMER2 后分频器递增。

TIMER2 与 PR2 比较器的匹配输出随后输入给 TIMER2 的后分频器。后分频器具有 1:1 至 1:16 的预分频比可供选择。TIMER2 后分频器的输出用于使 PIR1 寄存器的 TMR2IF 中断标志位置 1。

TMR2 和 PR2 寄存器均可读写。任何复位时, TMR2 寄存器均被设置为 00h 且 PR2 寄存器被设置为 FFh。通过将 T2CON 寄存器的 TMR2ON 位置 1 使能 TIMER2; 通过将 TMR2ON 位清零禁止 TIMER2。

TIMER2 预分频器由 T2CON 寄存器的 T2CKPS 位控制; TIMER2 后分频器由 T2CON 寄存器的 TOUTPS 位控制。

预分步器和后分步器计数器在以下情况下被清零:

- 对 TMR2 寄存器执行写操作;
- 对 T2CON 寄存器执行写操作;
- 发生任何器件复位 (上电复位、看门狗定时器复位或欠压复位)。

注: 写 T2CON 不会将 TMR2 清零。

9.3 TIMER2 相关的寄存器

有 2 个寄存器与 TIMER2 相关，分别是数据存储寄存器 TMR2 和控制寄存器 T2CON。

TIMER2 数据寄存器 TMR2 (11H)

11H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR2								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

TIMER2 控制寄存器 T2CON (12H)

12H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T2CON	---	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
读写	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	0	0	0	0	0	0	0

- Bit7 未用。
- Bit6~Bit3 TOUTPS<3:0>: TIMER2 输出后分频比选择位。
 - 0000= 1:1 后分频比;
 - 0001= 1:2 后分频比;
 - 0010= 1:3 后分频比;
 - 0011= 1:4 后分频比;
 - 0100= 1:5 后分频比;
 - 0101= 1:6 后分频比;
 - 0110= 1:7 后分频比;
 - 0111= 1:8 后分频比;
 - 1000= 1:9 后分频比;
 - 1001= 1:10 后分频比;
 - 1010= 1:11 后分频比;
 - 1011= 1:12 后分频比;
 - 1100= 1:13 后分频比;
 - 1101= 1:14 后分频比;
 - 1110= 1:15 后分频比;
 - 1111= 1:16 后分频比。
- Bit2 TMR2ON: TIMER2 使能位;
 - 1= 使能 TIMER2;
 - 0= 禁止 TIMER2。
- Bit1~Bit0 T2CKPS<1:0>: TIMER2 时钟预分频比选择位;
 - 00= 预分频值为 1;
 - 01= 预分频值为 4;
 - 1x= 预分频值为 16。

10. 模数转换（ADC）

10.1 ADC 概述

模数转换器（ADC）可以将模拟输入信号转换为表示该信号的一个 12 位二进制数。器件使用的模拟输入通道共用一个采样保持电路。采样保持电路的输出与模数转换器的输入相连。模数转换器采用逐次逼近法产生一个 12 位二进制结果，并将该结果保存在 ADC 结果寄存器（ADRESH 和 ADRESL）中。

ADC 参考电压始终为内部产生，ADC 在转换完成之后可以产生一个中断，下图为 ADC 框图。

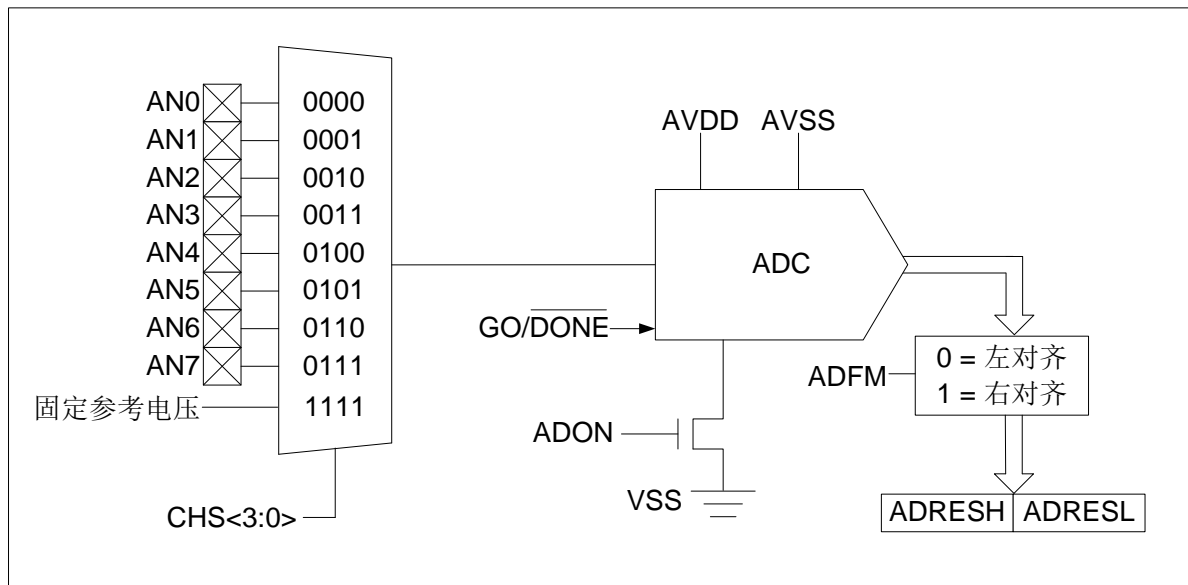


图 10-1: ADC 框图

10.2 ADC 配置

配置和使用 ADC 时，必须考虑如下因素：

- ◆ 端口配置；
- ◆ 通道选择；
- ◆ ADC 转换时钟源；
- ◆ 中断控制；
- ◆ 结果的存储格式。

10.2.1 端口配置

ADC 既可以转换模拟信号，又可以转换数字信号。当转换模拟信号时，应该通过将相应的 TRIS 位置 1，将 I/O 引脚配置为模拟输入引脚。更多信息请参见相应的端口章节。

注：对定义为数字输入的引脚施加模拟电压可能导致输入缓冲器出现过电流。

10.2.2 通道选择

由 ADCON0 寄存器的 CHS 位决定将哪个通道连接到采样保持电路，如果更改了通道，在下一次转换开始前需要一定的延迟。更多信息请参见第 11.3 节“ADC 工作原理”。

10.2.3 ADC 参考电压

ADC 的参考电压始终是由芯片的 VDD 和 GND 提供。

10.2.4 转换时钟

可以通过软件设置 ADCON0 寄存器的 ADCS 位来选择转换的时钟源。有以下 4 种可能的时钟频率可供选择：

- ◆ $F_{SYS}/8$
- ◆ $F_{SYS}/32$
- ◆ $F_{SYS}/16$
- ◆ F_{RC} （专用内部低速振荡器）

完成一位转换的时间定义为 TAD。一个完整的 12 位转换需要 49 个 TAD 周期。

必须符合相应的 TAD 规范，才能获得正确的转换结果。下表给出了正确选择 ADC 时钟的示例。

注：除非使用 FRC，否则系统时钟频率的任何改变都会改变 ADC 时钟的频率，从而对 ADC 转换结果产生负面影响。

ADC 时钟周期（TAD）与器件工作频率的关系（VDD=5.0V）。

ADC 时钟周期		器件频率		
ADC 时钟源	ADCS<1:0>	8MHz	4MHz	1MHz
$F_{SYS}/8$	00	49.0μs	98.0μs	392.0μs
$F_{SYS}/16$	01	98.0μs	196.0μs	784.0μs
$F_{SYS}/32$	10	196.0μs	392.0μs	1.5ms
F_{RC}	11	1-3ms	1-3ms	1-3ms

图注：建议不要使用阴影单元内的值。

10.2.5 ADC 中断

ADC 模块允许在完成模数转换后产生一个中断。ADC 中断标志位是 PIR1 寄存器中的 ADIF 位。ADC 中断允许位是 PIE1 寄存器中的 ADIE 位。ADIF 位必须用软件清零。每次转换结束后 ADIF 位都会被置 1，与是否允许 ADC 中断无关。

不管器件处于工作模式还是休眠模式都可以产生中断。如果器件处于休眠模式，该中断可将器件唤醒。当将器件从休眠状态唤醒后，总是执行 STOP 指令后的下一条指令。如果用户尝试使器件从休眠模式唤醒并按顺序恢复代码执行，则必须禁止全局中断。如果允许全局中断，程序将跳转到中断服务程序处执行。

10.2.6 结果格式化

12 位 A/D 转换的结果可采用两种格式：左对齐或右对齐。由 ADCON0 寄存器的 ADFM 位控制输出格式。

- 当 ADFM=0 时，AD 转换结果左对齐，AD 转换结果为 12Bit;
- 当 ADFM=1 时，AD 转换结果右对齐，AD 转换结果为 10Bit。

10.3 ADC 工作原理

10.3.1 启动转换

要启用 ADC 模块，必须将 ADCON0 寄存器的 ADON 位置 1，将 ADCON0 寄存器的 GO/DONE 位置 1 开始模数转换。

注：不能用开启 A/D 模块的同一指令将 GO/DONE 位置 1。

10.3.2 完成转换

当转换完成时，ADC 模块将：

- 清零 GO/DONE 位；
- 将 ADIF 标志位置 1；
- 用转换的新结果更新 ADRESH:ADRESL 寄存器。

10.3.3 终止转换

如果必须要在转换完成前终止转换，则可用软件清零 GO/DONE 位。不会用尚未完成的模数转换结果更新 ADRESH:ADRESL 寄存器。因此，ADRESH:ADRESL 寄存器将保持上次转换所得到的值。此外，在 A/D 转换终止以后，必须经过 2 个 TAD 的延时才能开始下一次采集。延时过后，将自动开始对选定通道的输入信号进行采集。

注：器件复位将强制所有寄存器进入复位状态。因此，复位会关闭 ADC 模块并且终止任何待处理的转换。

10.3.4 ADC 在休眠模式下的工作原理

ADC 模块可以工作在休眠模式下。此操作需要将 ADC 时钟源设置为 FRC 选项。如果选择了 FRC 时钟源，ADC 在开始转换之前要多等待一个指令周期。从而允许执行 STOP 指令，以降低转换中的系统噪声。如果允许 ADC 中断，当转换结束时，将使器件从休眠模式唤醒。如果禁止 ADC 中断，即使 ADON 位保持置 1，则转换结束后也还是会关闭 ADC 模块。如果 ADC 时钟源不是 FRC，即使 ADON 位仍保持置 1，执行 STOP 指令还是会中止当前的转换并关闭 A/D 模块。

10.3.5 AD 转换步骤

如下步骤给出了使用 ADC 进行模数转换的示例：

1. 端口配置。
 - 禁止引脚输出驱动器（见 TRIS 寄存器）。
 - 将引脚配置为模拟输入引脚。
2. 配置 ADC 模块。
 - 选择 ADC 转换时钟。
 - 选择 ADC 输入通道。
 - 选择结果的格式。
 - 启动 ADC 模块。
3. 配置 ADC 中断（可选）。
 - 清零 ADC 中断标志位。
 - 允许 ADC 中断。
 - 允许外设中断。
 - 允许全局中断。
4. 等待所需的采集时间。
5. 将 $\overline{GO/DONE}$ 置 1 启动转换。
6. 由如下方法之一等待 ADC 转换结束。
 - 查询 $\overline{GO/DONE}$ 位。
 - 等待 ADC 中断（允许中断）。
7. 读 ADC 结果。
8. 将 ADC 中断标志位清零（如果允许中断的话，需要进行此操作）。

注：如果用户尝试在使器件从休眠模式唤醒后恢复顺序代码执行，则必须禁止全局中断。

例：AD 转换

LDIA	B'1000000'	
LD	ADCON1,A	
SETB	TRISD,0	;设置 PORTD.0 为输入口
LDIA	B'11000001'	
LD	ADCON0,A	
CALL	DELAY	;延时一段时间
SETB	ADCON0,GO	
SZB	ADCON0,GO	;等待 AD 转换结束
JP	\$-1	
LD	A,ADRESH	;保存 AD 转换结果高位
LD	RESULTH,A	
LD	A,ADRESL	;保存 AD 转换结果低位
LD	RESULTL,A	

10.4 ADC 相关寄存器

主要有 4 个寄存器与 AD 转换相关，分别是控制寄存器 ADCON0 和 ADCON1，数据寄存器 ADRESH 和 ADRESL。

AD 控制寄存器 ADCON0(1FH)

1FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ $\overline{\text{DONE}}$	ADON
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

- Bit7~Bit6 ADCS<1:0>: A/D转换时钟选择位。
- 00= $F_{\text{SYS}}/8$
 - 01= $F_{\text{SYS}}/16$
 - 10= $F_{\text{SYS}}/32$
 - 11= F_{RC} （由专用的内部振荡器产生频率最高为500KHz的时钟）
- Bit5~Bit2 CHS<3:0>: 模拟通道选择位。
- 0000= AN0
 - 0001= AN1
 - 0010= AN2
 - 0011= AN3
 - 0100= AN4
 - 0101= AN5
 - 0110= AN6
 - 0111= AN7
 - 1000= 未用
 - 1001= 未用
 - 1010= 未用
 - 1011= 未用
 - 1100= 未用
 - 1101= 未用
 - 1110= 未用
 - 1111= 0.6V（内部基准电压）
- Bit1 GO/ $\overline{\text{DONE}}$: A/D 转换状态位。
- 1= A/D转换正在进行。将该位置1启动A/D转换。当A/D转换完成以后，该位由硬件自动清零。
 - 0= A/D转换完成/或不在进行中。
- Bit0 ADON: ADC使能位。
- 1= 使能ADC。
 - 0= 禁止ADC，不消耗工作电流。

AD 控制寄存器 ADCON1(9FH)

9FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	---	---	---	---	---	---	---
读写	R/W	---	---	---	---	---	---	---
复位值	0	---	---	---	---	---	---	---

Bit7 ADFM: A/D转换结果格式选择位;
 1= 右对齐;
 0= 左对齐。

Bit6~Bit0 未用。

AD 数据寄存器高位 ADRESH(1EH), ADFM= 0

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	ADRES11	ADRES10	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4
读写	R	R	R	R	R	R	R	R
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0 ADRES<11:4>: ADC 结果寄存器位。12 位转换结果的高 8 位。

AD 数据寄存器低位 ADRESL(9EH), ADFM= 0

9EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES3	ADRES2	ADRES1	ADRES0	----	----	----	----
读写	R	R	R	R	----	----	----	----
复位值	X	X	X	X	----	----	----	----

Bit7~Bit4 ADRES<3:0>: ADC结果寄存器位。12位转换结果的低4位。

Bit3~Bit0 未用。

AD 数据寄存器高位 ADRESH(1EH), ADFM=1

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	---	---	---	---	---	---	ADRES11	ADRES10
读写	---	---	---	---	---	---	R	R
复位值	---	---	---	---	---	---	X	X

Bit7~Bit2 未用。

Bit1~Bit0 ADRES<11:10>: ADC结果寄存器位。12位转换结果的高2位。

AD 数据寄存器低位 ADRESL(9EH), ADFM=1

9EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2
读写	R	R	R	R	R	R	R	R
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0 ADRES<9:2>: ADC 结果寄存器位。12 位转换结果的第 2-9 位。

注: 在 ADFM=1 的情况下, AD 转换结果只保存 12 位结果的高 10 位, 其中 ADRESH 保存高 2 位, ADRESL 保存第 2 位至第 9 位。

11. 主控同步串行端口（MSSP）模块

11.1 主控 SSP(MSSP)模块概述

主控同步串行端口（Master Synchronous Serial Port, MSSP）模块是用于同其他外设或单片机进行通信的串行接口。这些外设器件可以是串行 EEPROM、移位寄存器、显示驱动器或 A/D 转换器等。MSSP 模块有下列两种工作模式：

- 串行外设接口（SPI）。
 - I²C。
 - 全主控模式；
 - 从动模式（支持广播地址呼叫）。
- I²C 接口在硬件上支持下列模式。
 - 主控模式
 - 主机模式
 - 从动模式

11.2 SPI 模式

SPI 模式允许同时同步发送和接收 8 位数据。支持 SPI 的所有 4 种模式。通常使用以下三个引脚来完成通信：

- 串行数据输出（SDO）——RA1/SDO
- 串行数据输入（SDI）——RA0/SDI/SDA
- 串行时钟（SCK）——RA2/SCK/SCL

此外，在任意一种从动工作模式下均可使用第 4 个引脚：

- 从动选择（SS）——RA3/SS

11.2.2 SPI 相关寄存器
SSPSTAT: SSP 状态寄存器(94H)

94H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPSTAT	SMP	CKE	---	---	---	---	---	---
读写	R/W	R/W	---	---	---	---	---	---
复位值	0	0	---	---	---	---	---	---

Bit7 **SMP:** 采样位。

SPI 主控模式:

1 = 在数据输出时间的末尾采样输入数据;

0 = 在数据输出时间的中间采样输入数据。

SPI 从动模式: 当使用SPI的从动模式时, 必须将SMP清零。

Bit 6

CKE: SPI时钟边沿选择位。

CKP= 0

1= 在SCK引脚的上升沿发送数据;

0= 在SCK引脚的下降沿发送数据。

CKP = 1

1 = 在SCK引脚的下降沿发送数据;

0 = 在SCK引脚的上升沿发送数据。

Bit5~Bit0

SPI模式下未用。

SSPCON: SSP 控制寄存器 0(95H)

95H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPCON	---	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
读写	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	0	0	0	0	0	0	0

- Bit7 未用。
- Bit6 **SSPOV:** 接收溢出指示位。
 1= **SSPBUF**仍保持前一数据时, 又收到一个新的字节。出现溢出时, **SSPSR**中的数据会丢失。溢出只会从从动模式下发生。在从动模式中, 即使仅发送数据, 用户也必须读**SSPBUF**以避免溢出。在主动模式中, 溢出位不被置1, 因为每次接收或发送新数据, 都要通过写**SSPBUF**寄存器来启动(该位必须由软件清零)。
 0= 没有溢出。
- Bit5 **SSPEN:** 同步串行端口使能位。
 1= 使能串行端口并将**SCK**、**SDO**、**SDI**和**SS**配置为串行端口引脚。
 0= 禁止串行端口并将这些引脚配置为I/O端口引脚。
- Bit4 **CKP:** 时钟极性选择位。
 1= 时钟空闲状态为高电平。
 0= 时钟空闲状态为低电平。
- Bit3~Bit0 **SSPM<3:0>:** 同步串行端口模式选择位;
 0000= **SPI**主动模式, 时钟= $F_{SYS} / 4$;
 0001= **SPI**主动模式, 时钟= $F_{SYS} / 16$;
 0010= **SPI**主动模式, 时钟= $F_{SYS} / 64$;
 0011= **SPI**主动模式, 时钟= **TMR2**输出/2;
 0100= **SPI**从动模式, 时钟= **SCK**引脚, 使能**SS**引脚控制;
 0101= **SPI**从动模式, 时钟= **SCK**引脚, 禁止**SS**引脚控制, **SS**可用作I/O引脚;
 0110= **I²C**从动模式, 7位地址;
 0111= **I²C**从动模式, 10位地址;
 1000= **I²C**主动模式, 时钟= $F_{SYS} / (4 * (SSPADD+1))$;
 1001= 禁止装载功能;
 1010= 保留;
 1011= 保留;
 1100= 保留;
 1101= 保留;
 1110= **I²C**从动模式, 7位地址, 并允许起始位和停止位中断;
 1111= **I²C**从动模式, 10位地址, 并允许起始位和停止位中断。

11.2.3 SPI 工作原理

当初始化 SPI 时，需要指定几个选项。可以通过对相应的控制位（SSPCON<5:0>和 SSPSTAT<7:6>）编程来指定。这些控制位用于指定以下选项：

- ◆ 主控模式（SCK 作为时钟输出）
- ◆ 从动模式（SCK 作为时钟输入）
- ◆ 时钟极性（SCK 的空闲状态）
- ◆ 输入数据的采样相位（数据输出时间的中间或末尾）
- ◆ 时钟速率（仅限主控模式）
- ◆ 时钟边沿（在 SCK 的上升沿/下降沿输出数据）
- ◆ 从动选择模式（仅限于从动模式）

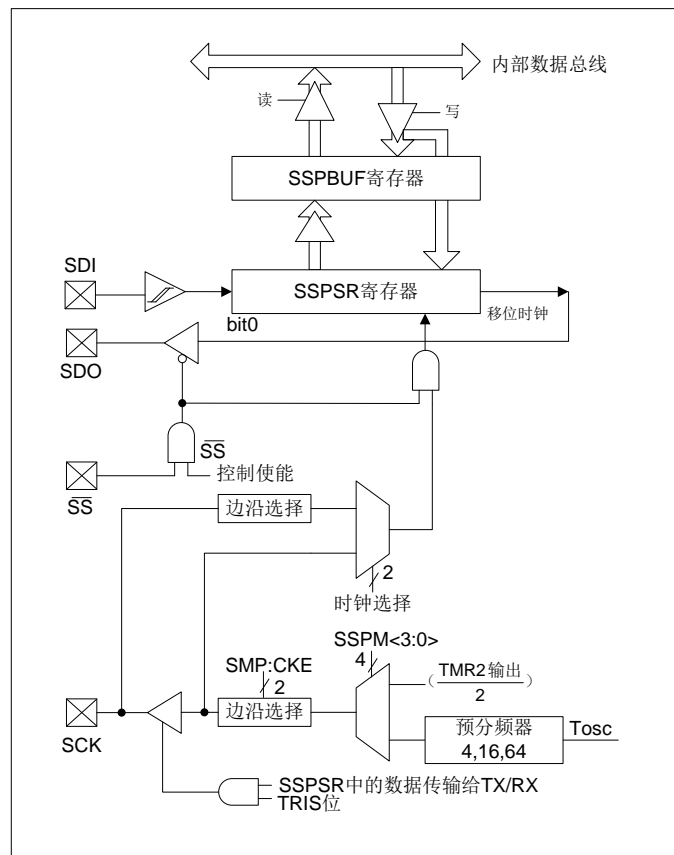


图 11-1: SPI 模式下 MSSP 模块的框图

注：I/O 引脚具有对 VDD 和 VSS 的二极管保护。

MSSP 模块由一个发送/接收移位寄存器（SSPSR）和一个缓冲寄存器（SSPBUF）组成。SSPSR 将数据移入和移出器件，最高有效位在前。SSPBUF 保存上次写入 SSPSR 的数据直到新接收到的数据就绪为止。一旦 8 位数据接收完毕，该字节就被移入 SSPBUF 寄存器。然后，PIR1 寄存器的中断标志位 SSPIF 被置 1。这种双重缓冲数据接收方式（SSPBUF）允许在读取刚接收的数据之前，就开始接收下一个字节。在数据发送/接收期间，任何试图写 SSPBUF 寄存器的操作都会被忽略，并将 SPCON 寄存器的写冲突检测位 WCOL 置 1。此时用户必须用软件将 WCOL 位清零，否则无法判别下一次对 SSPBUF 的写操作是否成功完成。

当应用软件等待接收有效数据时，应在下一个要传输的数据字节写入 SSPBUF 之前，将 SSPBUF 中的前一个数据读出。缓冲器满标志位 BF（SSPSTAT 寄存器）用于表示何时 SSPBUF 已经载入了接收到的数据（发送完成）。如果 SPI 仅仅作为发送器，则不必理会接收的数据。通常可用 MSSP 中断来判断发送或接收何时完成。如果不使用中断来处理数据的收发，用软件查询方法同样可确保不会发生写冲突。

11.2.4 使能 SPI I/O

要使能串行端口，SSPCON 寄存器的 MSSP 使能位 SSPEN 必须置 1。要复位或重新配置 SPI 模式，需先将 SSPEN 位清零，重新初始化 SSPCON 寄存器，然后将 SSPEN 位置 1。这将把 SDI、SDO、SCK 和 SS 引脚配置为串行端口引脚。要将这些引脚用作串行端口，还必须将其数据方向位（在 TRIS 寄存器中）正确编程，方法如下：

- SDI 由 SPI 模块自动控制；
- 必须将 SDO 的 TRISA<1>清零；
- 必须将 SCK（主控模式）的 TRISA<2>位清零；
- 必须将 SCK（从动模式）的 TRISA<2>位置 1；
- 必须将 SS 的 TRISA<3>置 1。

对于任何不想要的串行端口功能，可通过将对应的数据方向（TRIS）寄存器设置为相反值来跳过。

11.2.5 主控模式

主器件控制 SCK，因此可以随时启动数据传输。主器件根据软件协议确定从器件应在何时广播数据。

在主控模式下，数据一旦写入 SSPBUF 寄存器就开始发送或接收。如果 SPI 仅作为接收器，则可以禁止 SDO 输出（将其编程设定为输入）。SSPSR 寄存器按设置的时钟速率对 SDI 引脚上的信号进行连续移位输入。每个字节接收完后，都会被当作普通的接收字节装入 SSPBUF 寄存器（相应的中断和状态位置 1）。这可以在接收器应用中作为“线路活动监控（Line Activity Monitor）”模式，是很有用的。

可通过对 SSPCON 寄存器的 CKP 位进行相应的编程来选择时钟极性。图 11-2、图 11-4 和图 11-5 给出了 PI 通信的波形图，其中 MSb 被首先发送。在主控模式下，SPI 时钟速率（比特率）可由用户编程设定为下列速率之一：

- $F_{SYS} / 4$ （或 TCY）
- $F_{SYS} / 16$ （或 $4 \cdot TCY$ ）
- $F_{SYS} / 64$ （或 $16 \cdot TCY$ ）
- TIMER2 输出/2

图 11-2 为主控模式的波形图。当 SSPSTAT 寄存器的 CKE 位置 1 时，SDO 数据在 SCK 上出现时钟边沿前就有效。图中所示输入采样的变化由 SSPSTAT 寄存器的 SMP 位的状态决定。图中指出了将接收到的数据装入 SSPBUF 的时间。

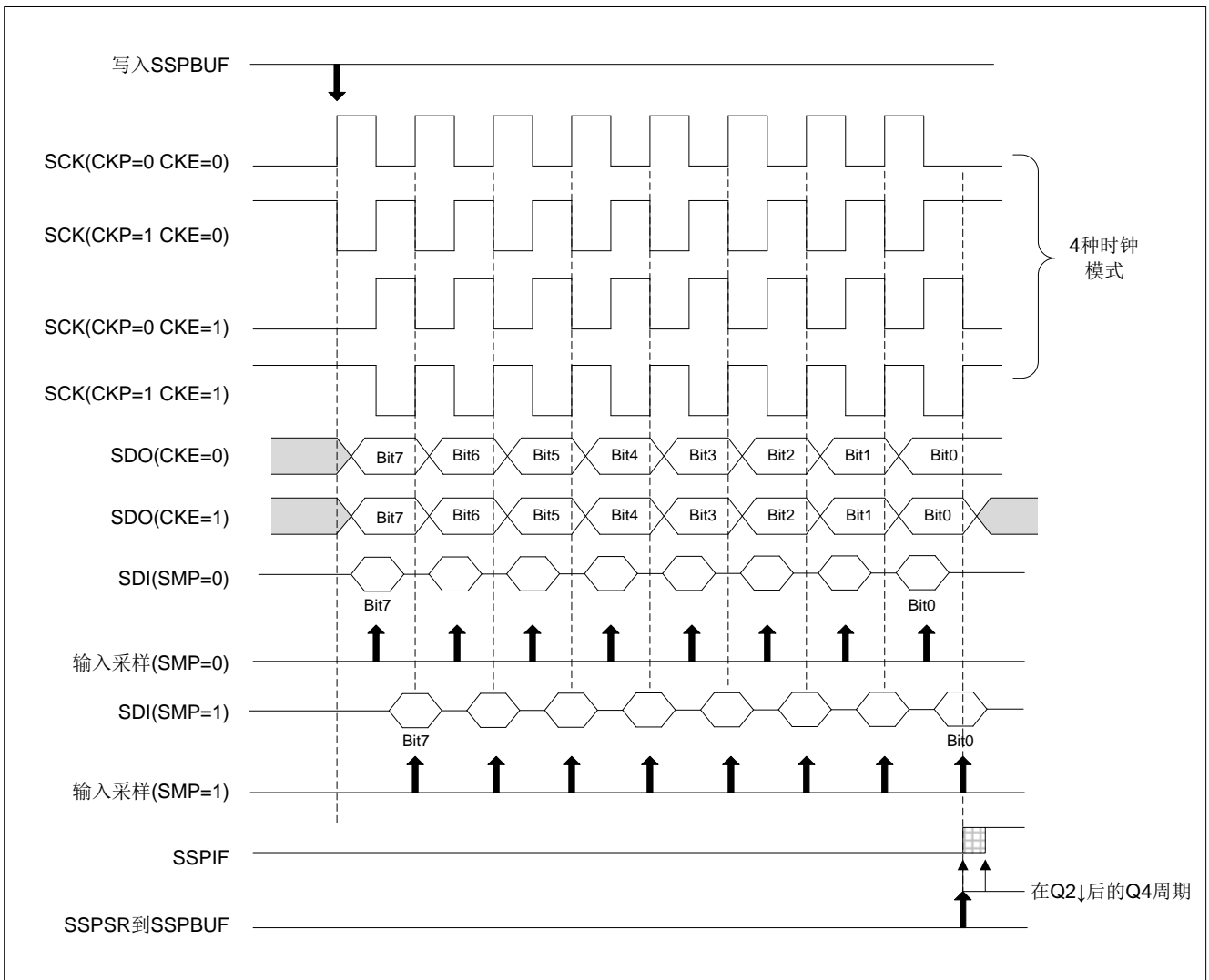


图 11-2: SPI 模式的波形（主控模式）

11.2.6 从动模式

在从动模式下，当 SCK 引脚上出现外部时钟脉冲时，发送和接收数据。当最后一位数据被锁存后，PIR1 寄存器的 SSPIF 中断标志位置 1。

在从动模式下，时钟由 SCK 引脚上的外部时钟源提供。外部时钟必须满足电气规范中规定的高电平和低电平的最短时间要求。

在休眠状态下，从器件仍可发送/接收数据。当收到一个字节时，器件从休眠状态被唤醒。

11.2.7 从动选择同步

SS 引脚允许器件工作在同步从动模式。SPI 必须工作在从动模式下，并使能 SS 引脚控制 SSPxCON1<3:0>=04h)。要使 SS 引脚用作输入引脚，不能将该引脚驱动为低电平。当 SS 引脚为低电平时，使能数据的发送和接收，同时 SDO 引脚被驱动。当 SS 引脚为高电平时，即使是在数据的发送过程中，SDO 引脚也不再被驱动，而是变成悬空输出。根据应用的需要，可外接上拉/下拉电阻。

当 SPI 模块复位后，位计数器被强制归 0。这可以通过强制将 SS 引脚拉为高电平或将 SSPEN 位清零来实现。将 SDO 引脚和 SDI 引脚相连可以仿真双线制通信。当 SPI 需要作为接收器工作时，SDO 引脚可以被配置为输入。这样就禁止了从 SDO 发送数据。因为 SDI 不会引起总线冲突，因而总是可以将其保留为输入（SDI 功能）。

注：

1. 当 SPI 工作在从动模式下，并且 SS 引脚控制使能（SSPxCON1<3:0> = 0100）时，如果 SS 引脚置为 VDD 电平，SPI 模块将被复位。
2. 如果在 CKE 置 1（SSPSTAT 寄存器）的从动模式下使用 SPI，则必须使能 SS 引脚控制。

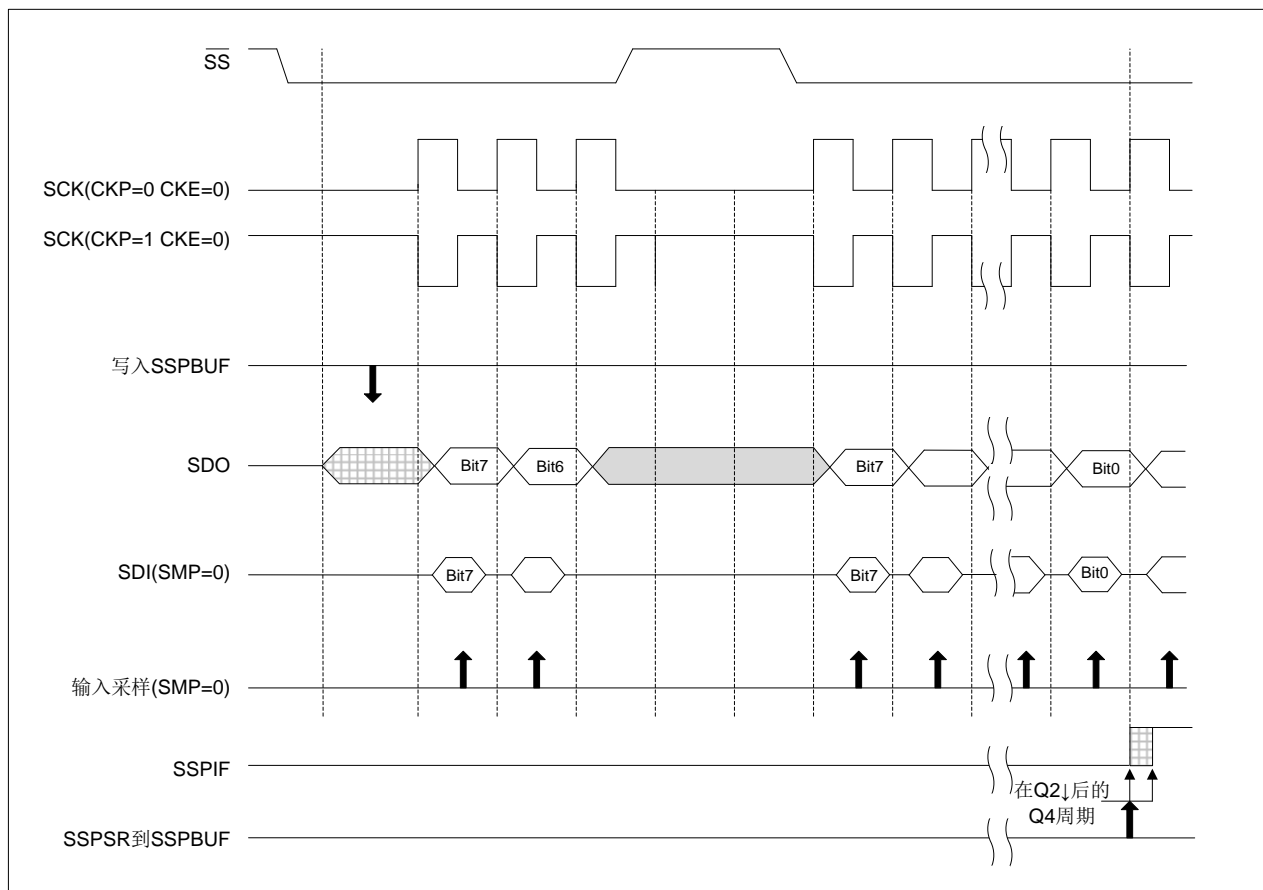


图 11-3: 从动同步波形

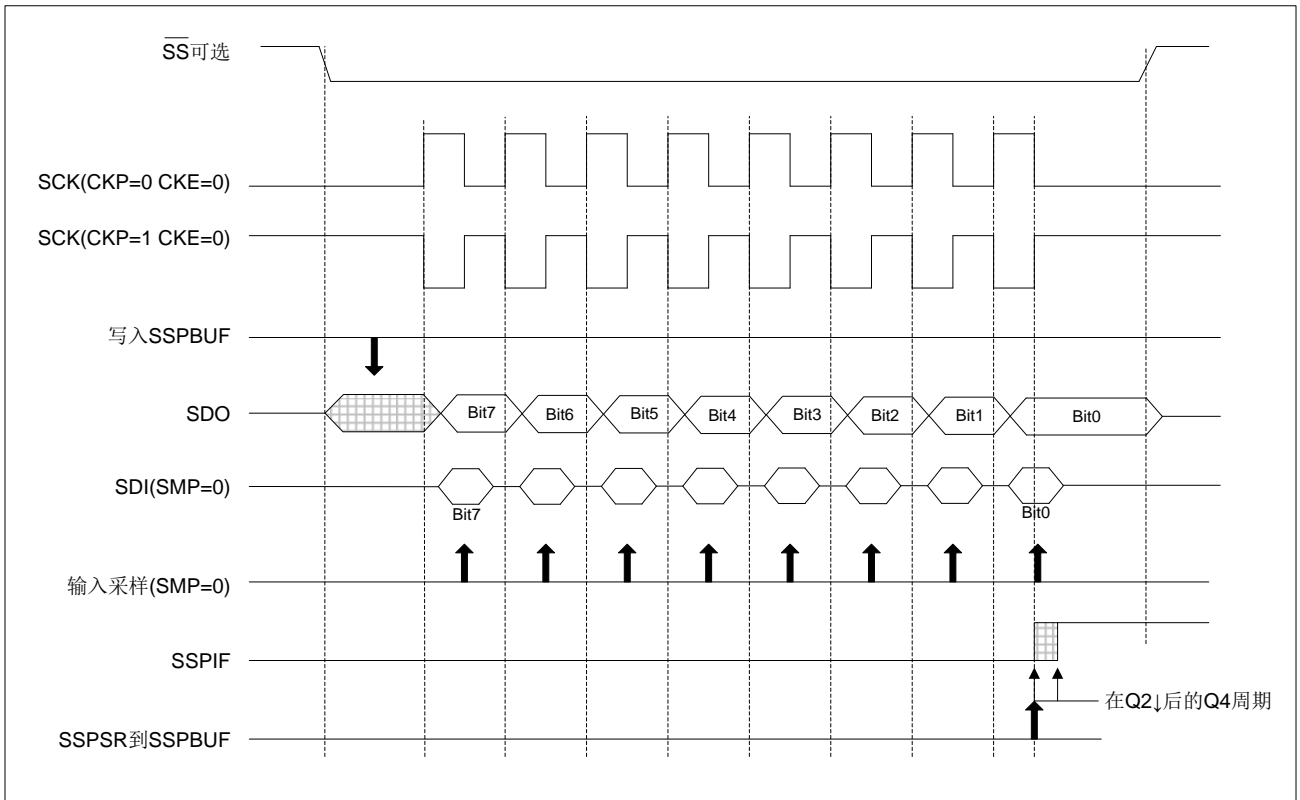


图 11-4: SPI 模式波形 (从动模式, CKE=0)

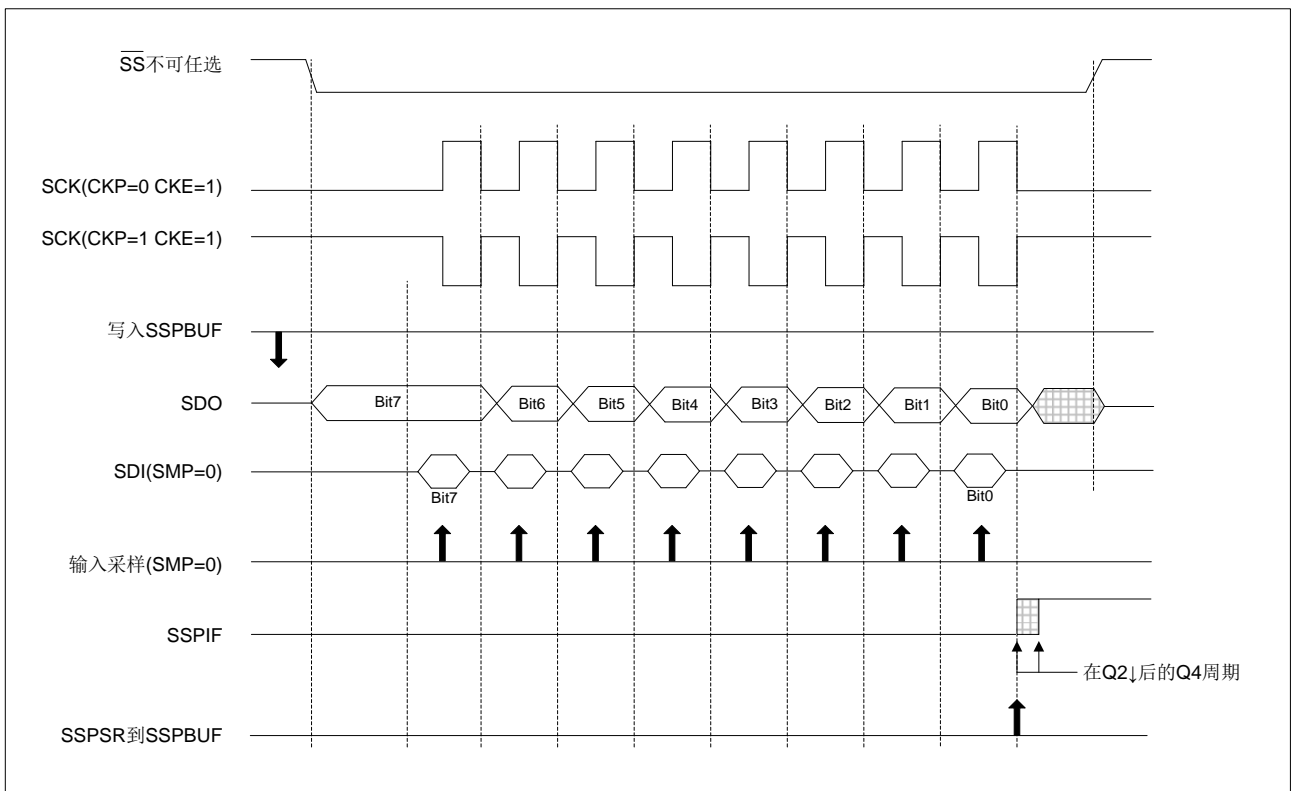


图 11-5: SPI 模式波形 (从动模式, CKE=1)

11.2.8 休眠操作

在主动模式下，如果选择了休眠模式，所有模块的时钟都将停止，并且在器件被唤醒前，发送/接收将保持此停滞状态。当器件返回到运行模式后，该模块将恢复发送和接收数据。

在从动模式下，SPI 发送/接收移位寄存器与器件异步工作。这可以使器件处于休眠模式下，而且数据仍可被移入 SPI 发送/接收移位寄存器。当 8 位数据全部接收到后，MSSP 中断标志位将置 1，并且如果允许中断的话，将唤醒器件。

11.2.9 复位的影响

复位会禁止 MSSP 模块并终止当前的传输。

11.3 I²C 模块

MSSP 模块工作在 I²C 模式时，可以实现所有的主控和从动功能（包括广播呼叫支持），并且用硬件提供起始位和停止位的中断来判断总线何时空闲（多主机功能）。MSSP 模块实现了标准模式规范，以及 7 位和 10 位寻址。

有两个引脚用于数据传输。它们是时钟引脚（SCL）——RA2/SCK/SCL 引脚，和数据引脚（SDA）——RA0/SDI/SDA 引脚。用户必须通过 TRISA<3:0>位将这些引脚配置为输入或输出引脚。通过将 SSPCON 寄存器的 MSSP 使能位 SSPEN 置 1，使能 MSSP 模块的功能。

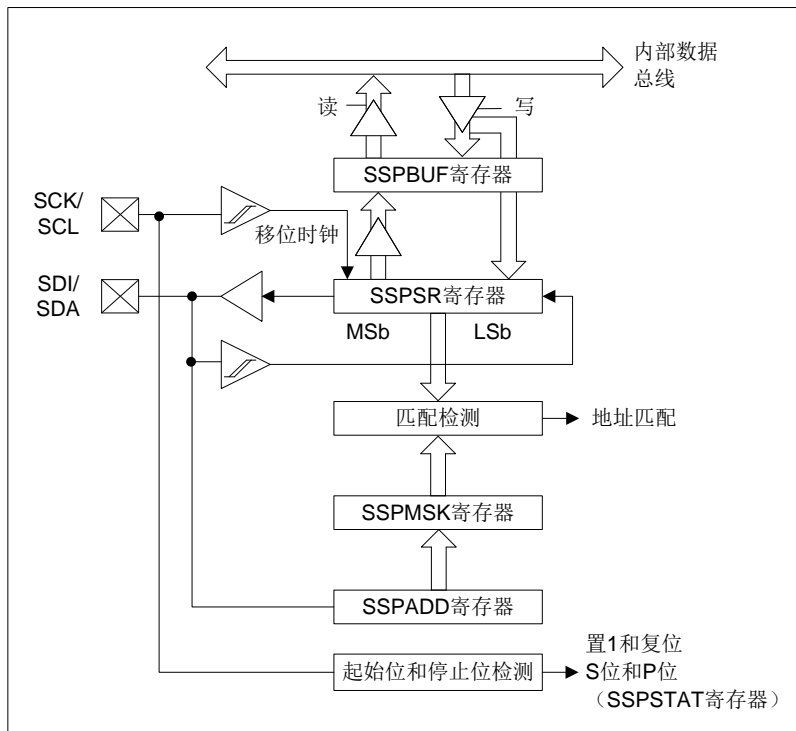


图 11-6: MSSP 框图 (I²C 模式)

注：I/O 引脚具有连接到 VDD 和 VSS 的保护二极管。

MSSP 模块具有 7 个用于 I²C 操作的寄存器。它们是：

- ◆ MSSP 控制寄存器 1 (SSPCON)
- ◆ MSSP 控制寄存器 2 (SSPCON2)
- ◆ MSSP 状态寄存器 (SSPSTAT)
- ◆ 串行接收/发送缓冲寄存器 (SSPBUF)
- ◆ MSSP 移位寄存器 (SSPSR)：不能直接访问
- ◆ MSSP 地址寄存器 (SSPADD)
- ◆ MSSP 屏蔽寄存器 (SSPMSK)

可使用 SSPCON 寄存器控制 I²C 的操作。可使用 SSPM<3:0>模式选择位 (SSPCON 寄存器) 选择以下 I²C 模式之一：

- ◆ I²C 从动模式 (7 位地址)
- ◆ I²C 主控模式，时钟=OSC/4 (SSPADD+1)
- ◆ I²C 从动模式 (10 位地址)
- ◆ I²C 从动模式，7 位地址，允许起始位和停止位中断
- ◆ I²C 固件控制的主控操作，从器件空闲
- ◆ I²C 从动模式，10 位地址，允许起始位和停止位中断

如果已将 SCL 和 SDA 引脚编程为输入引脚（将相应的 TRISA 位置 1），选择任何 I²C 模式且 SSPEN 位置 1 将强制 SCL 和 SDA 引脚为漏极开路。

11.3.1 相关寄存器说明
SSPSTAT: SSP 状态寄存器 (94H)

94H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF
读写	R/W	R/W	R	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0

Bit7	SMP: 1= 禁止变化率控制, 标准速度模式 (100KHz和1MHz)。 0= 使能变化率控制, 高速模式 (400KHz)。
Bit6	未用
Bit5	D/A: 数据/地址位。 1= 表示最后接收或发送的字节是数据。 0= 表示最后接收或发送的字节是地址。
Bit4	P: 停止位 (禁止MSSP模块 (SSPEN清零) 时此位被清零)。 1= 表示最后检测到了停止位 (复位时该位为0)。 0= 表示最后未检测到停止位。
Bit3	S: 起始位 (禁止MSSP模块 (SSPEN 清零) 时此位被清零)。 1= 表示最后检测到了起始位 (复位时该位为0)。 0= 最后未检测到起始位。
Bit2	R/W: 读/写位信息。 该位用来保存在最后一次地址匹配后的R/W位信息。该位仅在从地址匹配开始到下一个起始位、停止位或非ACK位时有效。 在I ² C从动模式下: 1= 读。 0= 写。 在I ² C主控模式下: 1= 正在发送。 0= 不在进行发送。 此位与SEN、RSEN、PEN、RCEN或ACKEN做逻辑或运算的结果将指示MSSP是否在空闲模式下。
Bit1	UA: 更新地址位 (仅限10位I ² C模式)。 1= 表示用户需要更新SSPADD 寄存器中的地址。 0= 不需要更新地址。
Bit0	BF: 缓冲器满状态位。 接收: 1= 接收完成, SSPBUF满。 0= 接收未完成, SSPBUF空。 发送: 1 = 数据正在发送 (不包括ACK和停止位), SSPBUF满。 0 = 数据发送完成 (不包括ACK和停止位), SSPBUF空。

SSPCON: SSP 控制寄存器 0(95H)

95H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

- Bit7** **WCOL:** 写冲突检测位。
主控模式: 1= 在I²C不满足开始发送数据的条件下, 试图对SSPBUF寄存器进行写操作。
 0= 未发生冲突。
从动模式:
 1= 正在发送前一个字时, 又对SSPBUF寄存器进行写操作(必须由软件清零)。
 0= 未发生冲突。
- Bit6** **SSPOV:** 接收溢出指示位。
 1= SSPBUF寄存器仍保持前一数据时, 又接收到一个新的字节。在发送模式下
 SSPOV位可为任意值(该位必须由软件清零)。
 0= 没有溢出。
- Bit5** **SSPEN:** 同步串行端口使能位(必须正确配置这些引脚为输入或输出引脚)。
 1= 使能串行端口并将SDA和SCL引脚配置为串行端口引脚。
 0= 禁止串行端口并将这些引脚配置为I/O端口引脚。
- Bit4** **CKP:** 时钟极性选择位。
在I²C从动模式下: **SCK**释放控制。
 1 = 使能时钟。
 0 = 保持时钟线为低电平(时钟延长)(用于确保数据建立时间)。
在I²C主控模式下: 在此模式下未使用。
- Bit3~Bit0** **SSPM<3:0>:** 同步串行端口模式选择位。
 0000= SPI主控模式, 时钟= F_{sys}/4。
 0001= SPI主控模式, 时钟= F_{sys}/16。
 0010= SPI主控模式, 时钟= F_{sys}/64。
 0011= SPI主控模式, 时钟= TMR2输出/2。
 0100= SPI从动模式, 时钟= SCK引脚, 使能SS引脚控制。
 0101= SPI从动模式, 时钟= SCK引脚, 禁止SS引脚控制, SS可用作I/O引脚。
 0110= I²C从动模式, 7位地址。
 0111= I²C从动模式, 10位地址。
 1000= I²C主控模式, 时钟= F_{sys}/(4 * (SSPADD+1))。
 1001= 禁止装载功能。
 1010= 保留。
 1011= 保留。
 1100= 保留。
 1101= 保留。
 1110= I²C从动模式, 7位地址, 并允许起始位和停止位中断。
 1111= I²C从动模式, 10位地址, 并允许起始位和停止位中断。

SSP 控制寄存器 SSPCON2(91H)

91H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPCON2	GCEM	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
读写	R/W	R/W	R	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0

- Bit7** **GCEM:** 广播呼叫使能位（仅限I²C从动模式）。
 1= 允许在SSPSR中接收到广播呼叫地址（0000h）时产生中断。
 0= 禁止广播呼叫地址。
- Bit6** **ACKSTAT:** 应答状态位（仅限于I²C主控模式）。
 在**主控发送模式**下：
 1 = 未接收到来自从动器件的应答。
 0 = 已接收到来自从动器件的应答。
- Bit5** **ACKDT:** 应答数据位（仅限于I²C主控模式）。
 在**主控接收模式**下：用户在接收完成后发送的应答序列的值。
 1 = 不应答。
 0 = 应答。
- Bit4** **ACKEN:** 应答序列使能位（仅限I²C主控模式）。
 在**主控接收模式**下：
 1 = 在SDA和SCL引脚启动应答序列，发送ACKDT数据位。由硬件自动清零。
 0 = 应答序列空闲。
- Bit3** **RCEN:** 接收使能位（仅限I²C主控模式）。
 1= 使能I²C接收模式。
 0= 接收空闲。
- Bit2** **PEN** 停止条件使能位（仅限于I²C主控模式）。
SCK释放控制:
 1 = 在SDA和SCL引脚启动停止条件。由硬件自动清零。
 0 = 停止条件空闲。
- Bit1** **RSEN:** 重复启动条件使能位（仅限I²C主控模式）。
 1= 在SDA和SCL引脚启动重复启动条件。由硬件自动清零。
 0= 重复启动条件空闲。
- Bit0** **SEN:** 启动条件使能位（仅限I²C主控模式）。
 在**主控模式**下：
 1 = 在SDA和SCL引脚启动启动条件。由硬件自动清零。
 0 = 启动条件空闲。
 在**从动模式**下：
 1 = 从发送和接收都会使能时钟延长（使能时钟延长）。
 0 = 禁止时钟延长。

11.3.2 主控模式

主控模式通过在检测到启动和停止条件时产生中断来工作。停止（P）位和起始（S）位在复位时或禁止 MSSP 模块时清零。当 P 位置 1 时，可以取得 I²C 总线的控制权；否则总线处于空闲状态，且 P 位和 S 位都为零。

在主控模式中，SCL 和 SDA 线由 MSSP 硬件操纵。下列事件会使 MSSP 中断标志位 SSPIF 置 1（如果允许 MSSP 中断，则产生中断）：

- ◆ 启动条件
- ◆ 数据传输字节已发送/已接收
- ◆ 重复启动条件
- ◆ 停止条件
- ◆ 应答发送

11.3.3 I²C 主控模式支持

通过将 SSPCON 中相应的 SSPM 位置 1 或清零并将 SSPEN 位置 1 可使能主控模式。一旦使能主控模式，用户即可选择以下 6 项操作：

1. 在 SDA 和 SCL 上发出一个启动条件。
2. 在 SDA 和 SCL 上发出一个重复启动条件。
3. 写入 SSPBUF 寄存器，开始数据/地址的发送。
4. 在 SDA 和 SCL 上产生停止条件。
5. 将 I2C 端口配置为接收数据。
6. 在接收到数据字节后产生应答条件。

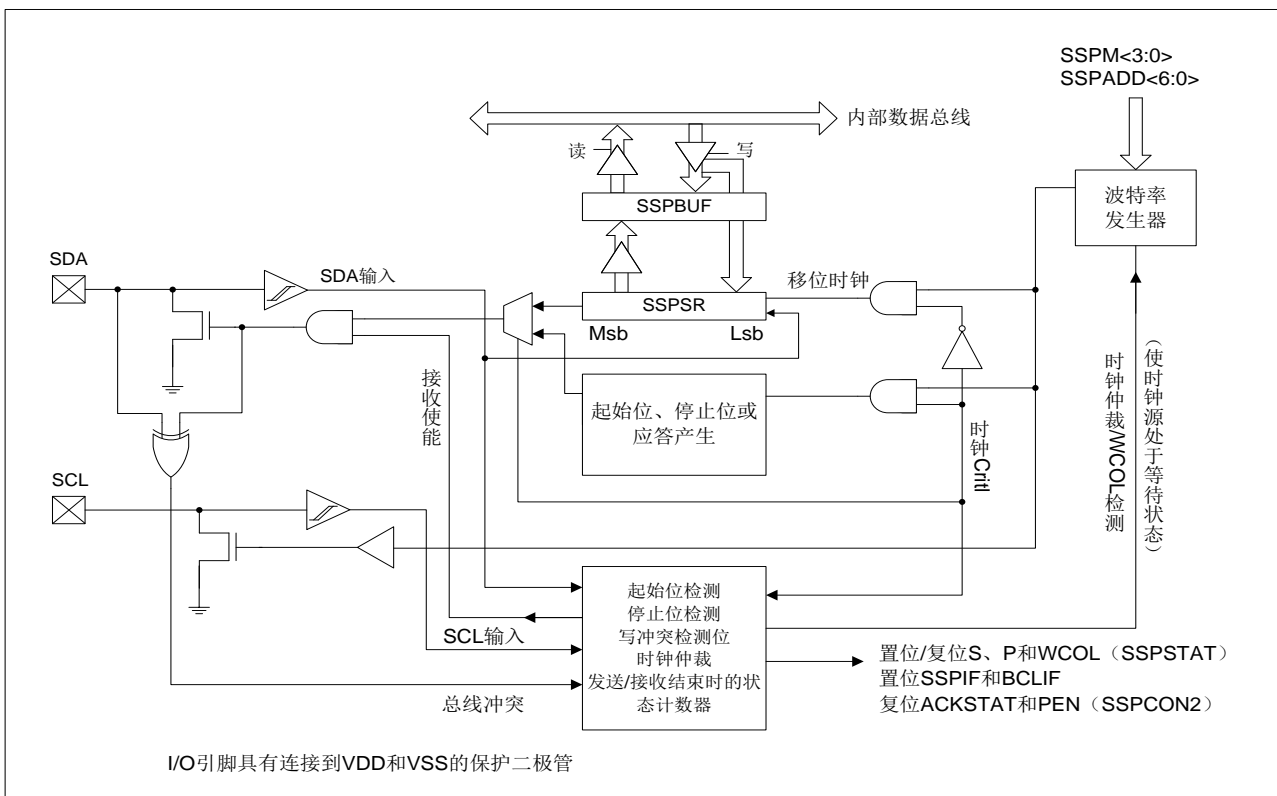


图 11-7: MSSP 框图 (I²C™ 主控模式)

注：

1. I/O 引脚具有连接到 VDD 和 VSS 的保护二极管。
2. 当配置为 I²C 主控模式时，MSSP 模块不允许事件排队。例如，在启动条件结束前，不允许用户发出另一个启动条件并立即写 SSPBUF 寄存器以发起传输。这种情况下，将不会写入 SSPBUF，WCOL 位将被置 1，这表明没有发生对 SSPBUF 的写操作。

11.3.3.1 I²C 主控模式操作

所有串行时钟脉冲和启动/停止条件均由主器件产生。停止条件或重复启动条件能结束传输。因为重复启动条件也是下一次串行传输的开始，因此不会释放 I²C 总线。在主控发送器模式下，串行数据通过 SDA 输出，而串行时钟由 SCL 输出。发送的第一个字节包括接收器件的地址（7 位）和读/写（R/W）位。在这种情况下，R/W 位将是逻辑 0。串行数据每次发送 8 位。每发送一个字节，会收到一个应答位。启动和停止条件的输出表明串行传输的开始和结束。

在主控接收模式下，发送的第一个字节包括发送器件的地址（7 位）和 R/W 位。在这种情况下，R/W 位将是逻辑 1。因此，发送的第一个字节是一个 7 位从器件地址，后面跟 1 表示接收。串行数据通过 SDA 接收，而串行时钟由 SCL 输出。每次接收 8 位串行数据。每接收到一个字节，都会发送一个应答位。启动和停止条件分别表明发送的开始和结束。

在 I²C 模式下，在 SPI 模式中使用的波特率发生器被用于将 SCL 时钟频率设置为 100kHz、400kHz 或 1MHz。波特率发生器的重载值位于 SSPADD 寄存器的低 7 位。当发生对 SSPBUF 的写操作时，波特率发生器将自动开始计数。如果指定操作完成（即发送的最后一个数据位后面跟着 ACK），内部时钟将自动停止计数，SCL 引脚将保持在其最后的状态。

下面是一个典型的发送事件序列：

- 用户通过将启动使能位 SEN（SSPCON2 寄存器）置 1 产生启动条件。
- SSPIF 位置 1。在进行任何其他操作前，MSSP 模块将等待所需的启动时间。
- 用户将从器件地址装入 SSPBUF 进行发送。
- 地址从 SDA 引脚移出，直到发送完所有 8 位为止。
- MSSP 模块移入来自从器件的 ACK 位，并将它的值写入 SSPCON2 寄存器的 ACKSTAT 位。
- MSSP 模块在第 9 个时钟周期的末尾将 SSPIF 位置 1，产生一个中断。
- 用户将 8 位数据装入 SSPBUF。
- 数据从 SDA 引脚移出，直到发送完所有 8 位为止。
- MSSP 模块移入来自从器件的 ACK 位，并将它的值写入 SSPCON2 寄存器的 ACKSTAT 位。
- MSSP 模块在第 9 个时钟的末尾将 SSPIF 位置 1，产生一个中断。
- 用户通过将停止使能位（PEN）位（SSPCON2 寄存器）置 1 产生停止条件。
- 一旦停止条件完成，将产生一个中断。

11.3.4 波特率发生器

在 I2C 主控模式下，波特率发生器的重载值位于 SSPADD 寄存器的低 7 位（图 11-8）。当装载了该值后，波特率发生器将自动开始计数并递减至 0，然后停止直到下次重载为止。BRG 会在每个指令周期（TCY）中的 Q2 和 Q4 时钟周期上进行两次减计数。在 I2C 主控模式下，会自动重载 BRG。例如，在发生时钟仲裁时，BRG 将在 SCL 引脚采样到高电平时重载（图 11-9）。

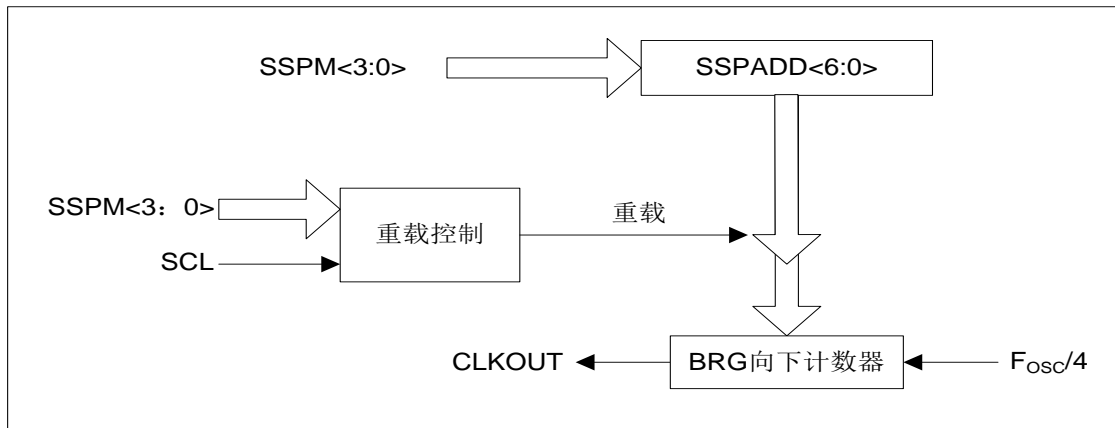


图11-8：波特率发生器框图

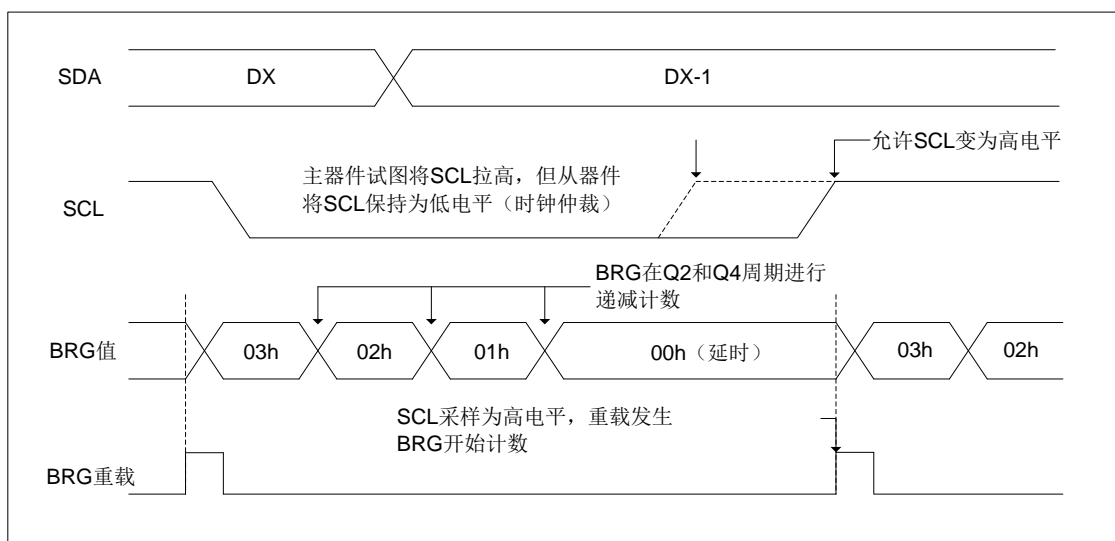


图11-9：带有时钟仲裁的波特率发生器时序

11.3.5 I²C 主控模式发送

发送一个数据字节、一个 7 位地址或一个 10 位地址的另一半，都可以直接通过写一个值到 SSPBUF 寄存器来实现。该操作将使缓冲器满标志位 BF 置 1，并且波特率发生器开始计数，同时启动下一次发送。在 SCL 的下降沿有效后，地址/数据的每一位将被移出至 SDA 引脚。在一个波特率发生器计满返回计数周期 (TBRG) 内，SCL 保持低电平。数据应该在 SCL 释放为高电平前保持有效。当 SCL 引脚被释放为高电平时，它将在整个 TBRG 中保持高电平状态。在此期间以及下一个 SCL 下降沿之后的一段时间内，SDA 引脚上的数据必须保持稳定。在第 8 位被移出（第 8 个时钟周期的下降沿）之后，BF 标志位清零，同时主器件释放 SDA。

此时如果发生地址匹配或是数据被正确接收，被寻址的从器件将在第 9 位的时间以一个 ACK 位响应。ACK 的状态在第 9 个时钟周期的下降沿写入 ACKDT 位。主器件接收到应答之后，应答状态位 ACKSTAT 会被清零；如果未收到应答，则该位被置 1。第 9 个时钟之后，SSPIF 位会置 1，主控时钟（波特率发生器）暂停，直到下一个数据字节装入 SSPBUF 为止，SCL 引脚保持低电平，SDA 保持不变（图 11-13）。

在写 SSPBUF 之后，地址的每一位在 SCL 的下降沿被移出，直至地址的所有 7 位和 R/W 位都被移出为止。在第 8 个时钟的下降沿，主器件将 SDA 引脚拉为高电平，以允许从器件发出应答响应。在第 9 个时钟的下降沿，主器件通过采样 SDA 引脚来判断地址是否被从器件识别。ACK 位的状态被装入 ACKSTAT 状态位（SSPCON2 寄存器）。在发送地址的第 9 个时钟下降沿之后，SSPIF 置 1，BF 标志位清零，波特率发生器关闭直到下一次写 SSPBUF，且 SCL 引脚保持低电平，允许 SDA 引脚悬空。

11.3.5.1 BF 状态标志

在发送模式下，BF 位（SSPSTAT 寄存器）在 CPU 写 SSPBUF 时置 1，在所有 8 位数据移出后清零。

11.3.5.2 WCOL 状态标志位

如果用户在发送过程中（即 SSPSR 仍在移出数据字节时）写 SSPBUF，则 WCOL 置 1 且缓冲器的内容保持不变（未发生写操作）。WCOL 必须由软件清零。

11.3.5.3 ACKSTAT 状态标志

在发送模式下，当从器件发送应答响应（ACK=0）时，ACKSTAT 位（SSPCON2 寄存器）清零；当从器件没有应答（ACK=1）时，该位置 1。从器件在识别出其地址（包括广播呼叫地址）或正确接收数据后，会发送一个应答。

11.3.6 I²C 主控模式接收

通过编程接收使能位 RCEN（SSPCON2 寄存器）使能主控模式接收。波特率发生器开始计数，每次计满返回时，SCL 引脚的状态都发生改变（由高变低或由低变高），且数据被移入 SSPSR。第 8 个时钟的下降沿之后，接收使能标志位自动清零，SSPSR 的内容装入 SSPBUF，BF 标志位置 1，SSPIF 标志位置 1，波特率发生器暂停计数，SCL 保持为低电平。此时 MSSP 处于空闲状态，等待下一条命令。当 CPU 读缓冲器时，BF 标志位将自动清零。通过将应答序列使能位 ACKEN（SSPCON2 寄存器）置 1，用户可以在接收结束后发送应答位。

11.3.6.1 BF 状态标志

接收时，当将地址或数据字节从 SSPSR 装入 SSPBUF 时，BF 位置 1；在读 SSPBUF 寄存器时 BF 位清零。

11.3.6.2 SSPOV 状态标志

接收时，当 SSPSR 接收到 8 位数据时，SSPOV 位置 1，BF 标志位已经在上一次接收时置 1。

11.3.6.3 WCOL 状态标志

如果用户在接收过程中（即 SSPSR 仍在移入数据字节时）写 SSPBUF，则 WCOL 位置 1，缓冲器内容不变（未发生写操作）。

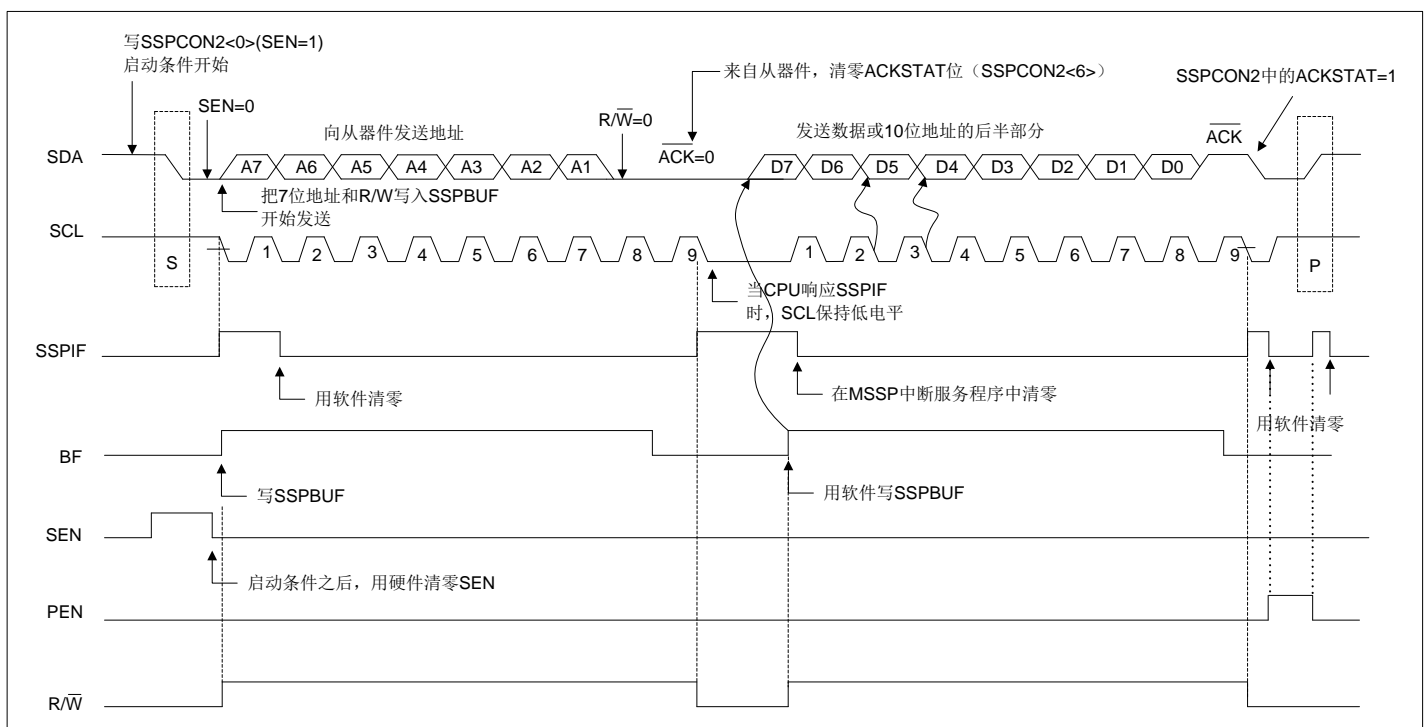


图11-10: I²C™主控模式发送时序（7位或者10位地址）

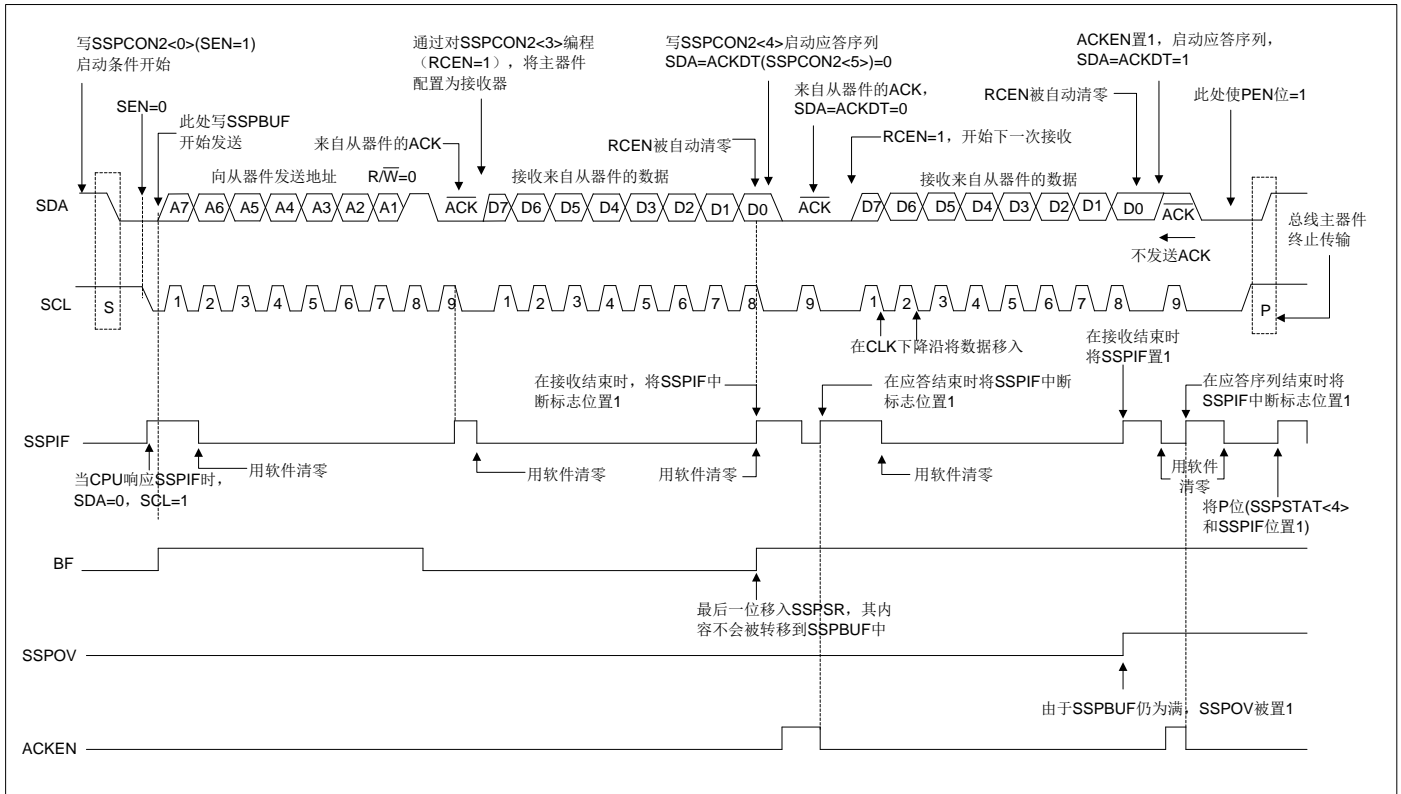


图11-11: I²C™主控模式接收时序 (7位地址)

11.3.7 I²C 主控模式启动条件时序

要发起启动条件，用户应将 SSPCON2 寄存器的启动条件使能位 SEN 置 1。当 SDA 和 SCL 引脚都采样为高电平时，波特率发生器重新装入 SSPADD<6:0>的内容并开始计数。当波特率发生器发生超时（TBRG）时，如果 SCL 和 SDA 都采样为高电平，则 SDA 引脚被驱动为低电平。当 SCL 为高电平时，将 SDA 驱动为低电平就是启动条件，将使 S 位（SSPSTAT 寄存器）置 1。随后波特率发生器重新装入 SSPADD<6:0>的内容并恢复计数。当波特率发生器超时（TBRG）时，SSPCON2 寄存器的 SEN 位将自动被硬件清零。波特率发生器暂停工作，SDA 线保持低电平，启动条件结束。

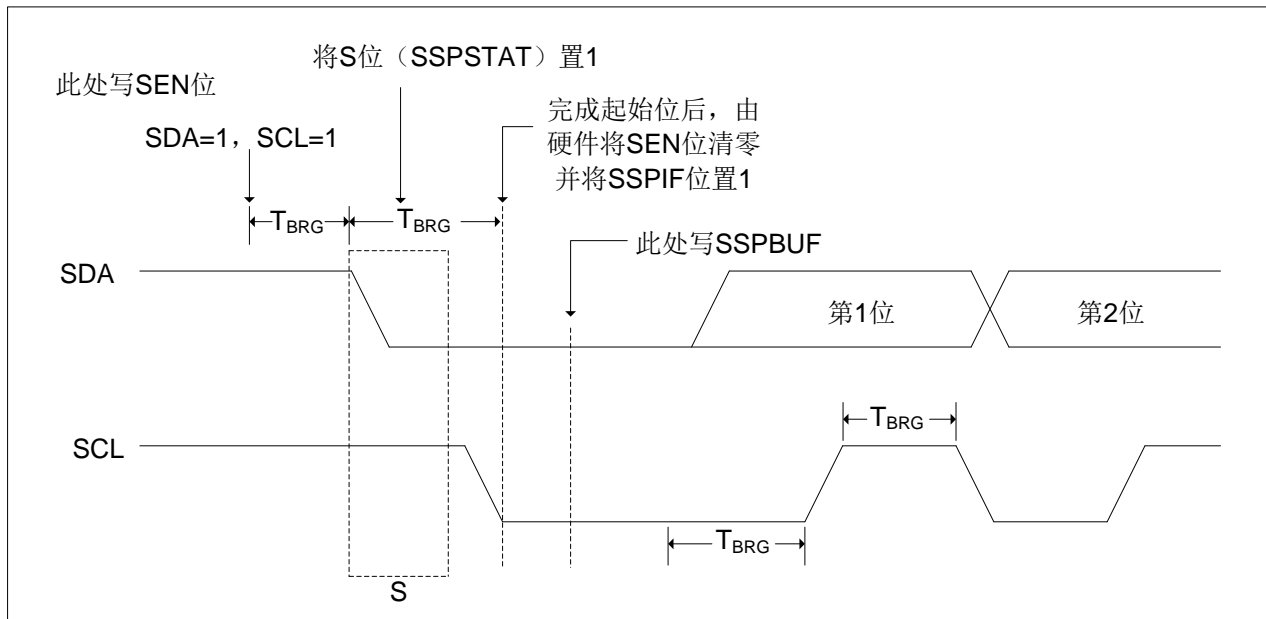


图 11-12: 第一个启动位时序

11.3.7.1 WCOL 状态标志

当启动序列进行时，如果用户写 SSPBUF，则 WCOL 被置 1，同时缓冲器内容不变（未发生写操作）。

注：由于不允许事件排队，在启动条件结束之前，不能对 SSPCON2 的低 5 位进行写操作。

11.3.8 I²C 主控模式重复启动条件时序

将 RSEN 位 (SSPCON2 寄存器) 编程为高电平, 并且 I2C 逻辑模块处于空闲状态时, 就会产生重复启动条件。当 RSEN 位置 1 时, SCL 引脚被拉为低电平。当 SCL 引脚采样为低电平时, 波特率发生器装入 SSPADD<6:0>的内容, 并开始计数。在一个波特率发生器计数周期 (TBRG) 内 SDA 引脚被释放 (其引脚电平被拉高)。当波特率发生器超时时, 如果 SDA 采样为高电平, SCL 引脚将被拉高。当 SCL 引脚采样为高电平时, 波特率发生器将被重新装入 SSPADD<6:0>的内容并开始计数。SDA 和 SCL 必须在一个计数周期 TBRG 内采样为高电平。随后将 SDA 引脚拉为低电平 (SDA = 0) 并保持一个计数周期 TBRG, 同时 SCL 为高电平。然后 RSEN 位 (SSPCON2 寄存器) 将自动清零, 波特率发生器不会重载, SDA 引脚保持低电平。一旦在 SDA 和 SCL 引脚上检测到启动条件, S 位 (SSPSTAT 寄存器) 将被置 1。直到波特率发生器超时时, SSPIF 位才会置 1。

一旦 SSPIF 位被置 1, 用户便可以在 7 位地址模式下将 7 位地址写入 SSPBUF, 或者在 10 位地址模式下写入默认的第一个地址字节。当发送完第一个 8 位并接收到一个 ACK 后, 用户可以发送另外 8 位地址 (10 位地址模式下) 或 8 位数据 (7 位地址模式下)。

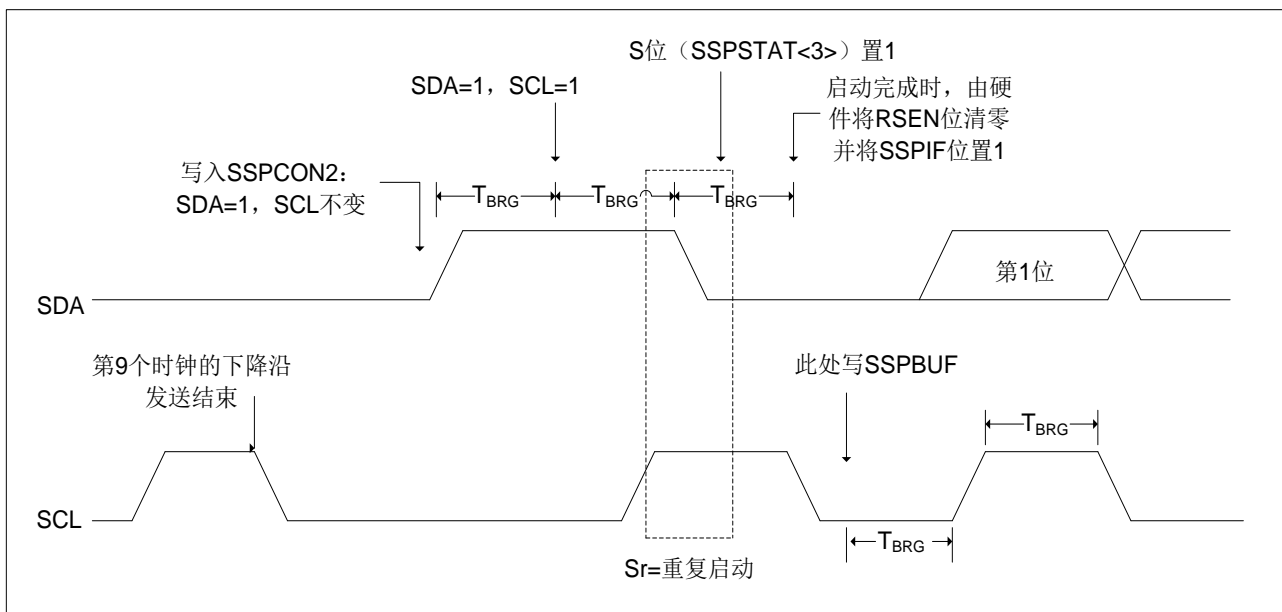


图 11-13: 重复启动条件时序波形

11.3.8.1 WCOL 状态标志

当重复启动序列进行时, 如果用户写 SSPBUF, 则 WCOL 被置 1, 同时缓冲器内容不变 (未发生写操作)。

注: 由于不允许事件排队, 在重复启动条件结束之前, 不能对 SSPCON2 的低 5 位进行写操作。

11.3.9 应答序列时序

将应答序列使能位 **ACKEN** (**SSPCON2** 寄存器) 置 1 即可使能应答序列。当该位被置 1 时, **SCL** 引脚被拉低, 应答数据位的内容出现在 **SDA** 引脚上。如果用户希望产生一个应答, 则应该将 **ACKDT** 位清零; 否则, 用户应该在应答序列开始前将 **ACKDT** 位置 1。然后波特率发生器进行一个计满返回周期 (**TBRG**) 的计数, 随后 **SCL** 引脚电平被拉高。当 **SCL** 引脚采样为高电平时 (时钟仲裁), 波特率发生器再进行一个 **TBRG** 周期的计数。然后 **SCL** 引脚被拉低。在这之后, **ACKEN** 位自动清零, 波特率发生器关闭, **MSSP** 模块进入空闲模式 (图 11-14)。

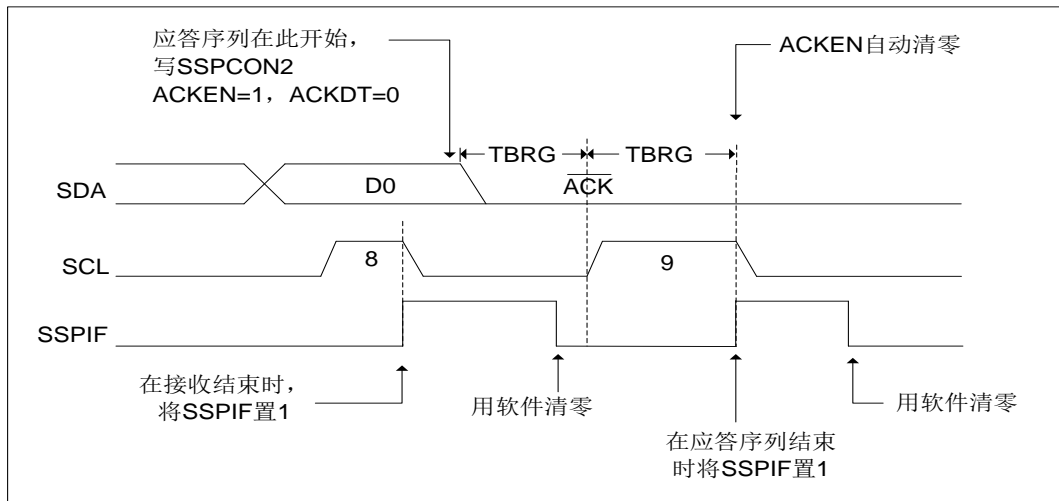


图 11-14: 应答序列时序波形

11.3.9.1 WCOL 状态标志位

如果用户在应答序列正在进行时写 **SSPBUF**, **WCOL** 将被置 1 且缓冲器的内容保持不变 (未发生写操作)。

11.3.10 停止条件序列

在接收/发送结束时，通过置停止序列的使能位，PEN（SSPCON2 寄存器），SDA 引脚将产生一个停止位。在接收/发送结束时，SCL 引脚在第 9 个时钟的下降沿后保持低电平。当 PEN 位置 1 时，主控器件将 SDA 置为低电平。当 SDA 线采样为低电平时，波特率发生器被重新装入值并递减计数至 0。波特率发生器发生超时，SCL 引脚被拉到高电平，且一个 TBRG（波特率发生器计满回零）后，SDA 引脚被重新拉到高电平。当 SDA 引脚采样为高电平且 SCL 也是高电平时，P 位（SSPSTAT 寄存器）置 1。一个 TBRG 周期后，PEN 位清零且 SSPIF 位置 1（图 11-14）。

11.3.10.1 WCOL 状态标志

如果用户在停止序列进行过程中试图写 SSPBUF，则 WCOL 位将置 1，缓冲器的内容不会改变（未发生写操作）。

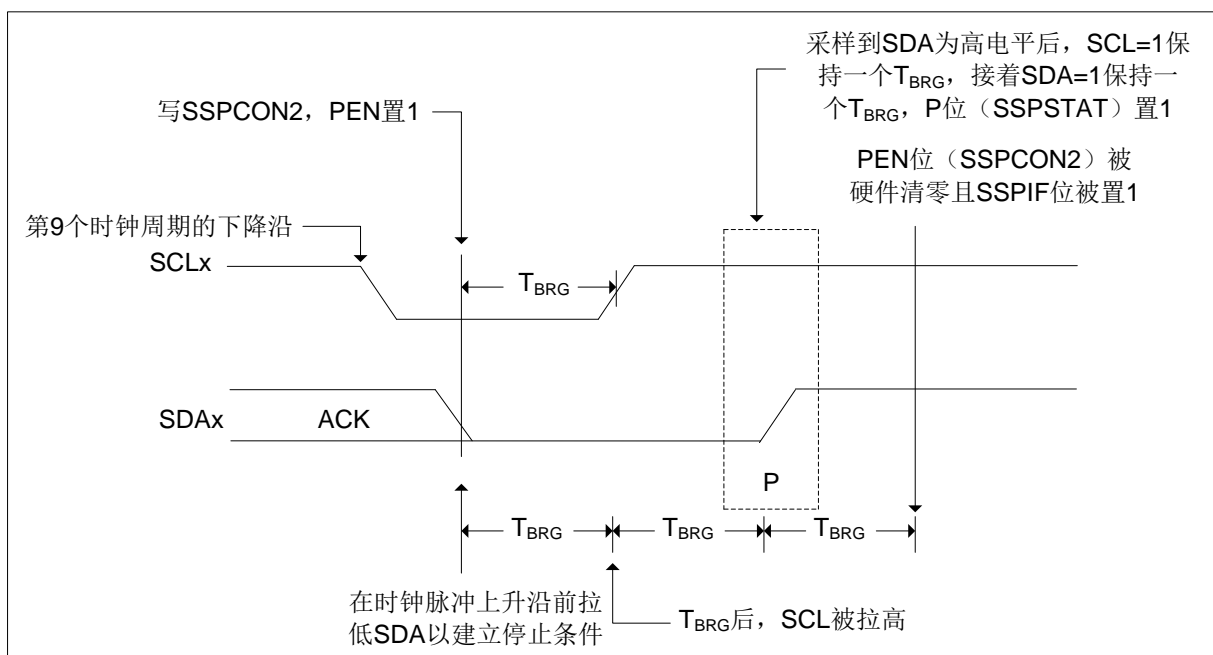


图11-15: 停止条件接收或发送模式

注: T_{BRG}=一个波特率发生器周期。

11.3.11 时钟仲裁

如果在任何接收、发送或重复启动/ 停止条件期间，主器件拉高了 SCL 引脚（允许 SCL 引脚悬空为高电平），就会发生时钟仲裁。如果允许 SCL 引脚悬空为高电平，波特率发生器（BRG）将暂停计数，直到实际采样到 SCL 引脚为高电平为止。当 SCL 引脚采样为高电平时，波特率发生器中将被重新装入 SSPADD<6:0>的内容并开始计数。这可以保证当外部器件将时钟拉低时，SCL 始终保持至少一个 BRG 计满返回周期的高电平（图 11-16）。

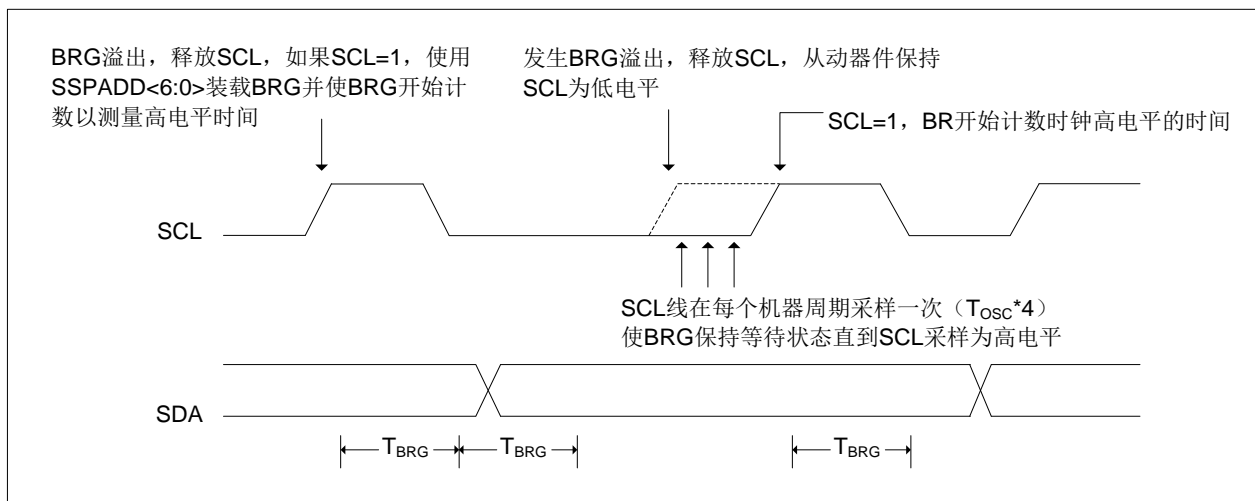


图 11-16：主控发送模式下的时钟仲裁时序

11.3.12 多主机模式

在多主机模式下，通过在检测到启动和停止条件时产生中断可以确定总线何时空闲。停止（P）位和启动（S）位在复位时或禁止 MSSP 模块时清零。当 P 位置 1 时，可以取得 I²C 总线的控制权；否则总线处于空闲状态，且 P 位和 S 位清零。当总线忙时，如果出现停止条件，则将产生中断（若允许 MSSP 中断）。

在多主机模式下工作时，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。此检查由硬件完成，其结果放在 BCLF 位。

在以下状态下仲裁可能失败：

- ◆ 地址传输
- ◆ 启动条件
- ◆ 应答条件
- ◆ 数据传输
- ◆ 重复启动条件

11.3.13 多主机通信、总线冲突与总线仲裁

多主机模式是通过总线仲裁来支持的。当主器件将地址/数据位输出到 SDA 引脚时，如果一个主器件通过将 SDA 引脚悬空为高电平以在 SDA 上输出 1，而另一个主器件输出 0，就会发生总线仲裁。如果 SDA 引脚上期望的数据是 1，而实际在 SDA 引脚上采样到的数据是 0，则发生了总线冲突。主器件将把总线冲突中断标志位 BCL1F 置 1，并将 I²C 端口复位到空闲状态（图 11-16）。

如果在发送过程中发生总线冲突，则发送停止，BF 标志位清零，SDA 和 SCL 线被拉高，并且允许对 SSPBUF 进行写操作。当执行完总线冲突中断服务程序后，如果 I²C 总线空闲，用户可通过发出启动条件恢复通信。如果在启动、重复启动、停止或应答条件的进行过程中发生总线冲突，则该条件被中止，SDA 和 SCL 线被拉高，SSPCON2 寄存器中的对应控制位清零。当执行完总线冲突中断服务程序后，如果 I²C 总线空闲，用户可通过发出启动条件恢复通信。主器件将继续监视 SDA 和 SCL 引脚。如果出现停止条件，SSPIF 位将被置 1。无论发生总线冲突时发送的进度如何，写 SSPBUF 都会从第一个数据位开始发送数据。

在多主机模式下，通过在检测到启动和停止条件时产生中断可以确定总线何时空闲。P 位置 1 时，可以获取 I²C 总线的控制权，否则总线空闲且 S 和 P 位清零。

11.3.14 从动模式

在从动模式下，SCL 引脚和 SDA 引脚必须被配置为输入（TRISC<4:3>置 1）。需要时（如从发送器）MSSP 模块将用输出数据改写输入状态。

当地址匹配时或在地址匹配后传输的数据被接收时，硬件会自动产生一个应答（ACK）脉冲，并把当时 SSPSR 寄存器中接收到的数据装入 SSPBUF 寄存器。

只要满足下列条件之一，MSSP 模块就不会产生此 ACK 脉冲：

- 缓冲器满标志位 BF（SSPCON 寄存器）在接收到传输的数据前置 1。
- 在接收到传输的数据之前，溢出标志位 SSPOV（SSPCON 寄存器）已被置 1。

在这种情况下，SSPSR 寄存器的值不会载入 SSPBUF，但是 PIR1 寄存器的 SSPIF 位会置 1。BF 位是通过读取 SSPBUF 寄存器清零的，而 SSPOV 位是通过软件清零的。

为确保正常工作，SCL 时钟输入必须满足最小高电平时间和最小低电平时间要求。

11.3.14.1 寻址

一旦使能了 MSSP 模块,它就会等待启动条件产生。在启动条件出现后,8 位数据被移入 SSPSR 寄存器。在时钟 (SCL) 线的上升沿采样所有的输入位。寄存器 SSPSR<7:1>的值会和 SSPADD 寄存器的值比较,该比较是在第 8 个时钟脉冲 (SCL) 的下降沿进行的。如果地址匹配,并且 BF 位和 SSPOV 位为零,会发生下列事件:

- SSPSR 寄存器的值被装入 SSPBUF 寄存器。
- 缓冲器满标志位 BF 置 1。
- 产生 ACK 脉冲。
- 在第 9 个 SCL 脉冲的下降沿,PIR1 寄存器的 MSSP 中断标志位 SSPIF 置 1 (如果允许中断则产生中断)。在 10 位地址模式下,从器件需要接收两个地址字节。第一个地址字节的高 5 位指定这是否为 10 位地址。R/W (SSPSTAT 寄存器) 必须指定写操作,这样从器件才能接收到第二个地址字节。对于 10 位地址,第一个字节应该是 11110 A9 A8 0,其中 A9 和 A8 是该地址的两个最高有效位。

10 位地址的工作时序如下,其中 7-9 步是针对从动发送器而言的:

1. 接收地址的第一 (高) 字节 (PIR1 寄存器的 SSPIF 位和 SSPSTAT 寄存器的 BF 和 UA 位置 1)。
2. 用地址的第二个 (低) 字节更新 SSPADD 寄存器 (UA 位清零并释放 SCL 线)。
3. 读 SSPBUF 寄存器 (BF 位清零) 并将标志位 SSPIF 清零。
4. 接收地址的第二个 (低) 字节 (SSPIF 位、BF 位和 UA 位置 1)。
5. 用地址的第一 (高) 字节更新 SSPADD 寄存器。如果匹配,释放 SCL 线,并将 UA 位清零。
6. 读 SSPBUF 寄存器 (BF 位清零) 并将标志位 SSPIF 清零。
7. 接收重复启动条件。
8. 接收地址的第一个 (高) 字节 (SSPIF 位和 BF 位置 1)。
9. 读 SSPBUF 寄存器 (BF 位清零) 并将标志位 SSPIF 清零。

11.3.14.2 接收

当地址字节的 R/W 位清零并发生地址匹配时,SSPSTAT 寄存器的 R/W 位清零。接收到的地址被装入 SSPBUF 寄存器。

当存在地址字节溢出条件时,则不会产生应答脉冲(ACK)。溢出条件是指 BF 位 (SSPSTAT 寄存器) 置 1,或者 SSPOV 位 (SSPCON 寄存器) 置 1。每个数据传输字节都会产生一个 MSSP 中断。必须用软件将 PIR1 寄存器的中断标志位 SSPIF 清零。SSPSTAT 寄存器用于确定该字节的状态。

11.3.15 SSP 屏蔽寄存器

在 I2C 从动模式下，SSP 屏蔽（SSPMSK）寄存器用于在地址比较操作下屏蔽 SSPSR 寄存器中的值。SSPMSK 寄存器中某位为 0 会使 SSPSR 寄存器中相应的位成为“无关位”。

此寄存器在任何复位条件发生时均复位为全 1，因此，在写入屏蔽值前，它对标准 SSP 操作没有影响。必须在通过设置 SSPM<3:0>位以选择 I2C 从动模式（7 位或 10 位地址）之前对此寄存器进行初始化。只有通过 SSPCON 的 SSPM<3:0>位选择了适当的模式后才可访问此寄存器。

SSP 屏蔽寄存器在以下情况下有效：

- 7 位地址模式：与 A<7:1>进行地址比较。
- 10 位地址模式：仅与 A<7:0>进行地址比较。

SSP 屏蔽在接收到地址的第一个（高）字节期间无效。

SSPMSK: SSP 屏蔽寄存器 SSPMSK(93H)(1)

93H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0 ⁽²⁾
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

Bit7~Bit1

MSK<7:1>: 屏蔽位。

1= 接收到的地址的Bit n 与SSPADD<n>比较以检测I²C的地址匹配情况。

0= 接收到的地址的Bit n 不用于检测I²C的地址匹配情况。

Bit 0

MSK<0>: I²C从动模式10 位地址的屏蔽位(2)。

I²C从动模式，10位地址

(SSPM<3:0> = 0111) :

1 = 接收到的地址的Bit0与SSPADD<0>比较以检测I²C的地址匹配情况。

0 = 接收到的地址的Bit0不用于检测I²C的地址匹配情况。

注：

1. 当 SSPCON 位 SSPM<3:0> = 1001 时，任何对 SSPADD SFR 地址的读或写操作都通过 SSPMSK 寄存进行。
2. 在所有其他 SSP 模式下，此位无效。

11.3.16 休眠模式下的操作

在休眠模式下，I²C 模块能够接收地址或数据。并且在地址匹配或字节传输完成后，将唤醒处理器（如果允许了 MSSP 中断）。

11.3.17 复位的影响

复位会禁止 MSSP 模块并终止当前的传输。

12. LCD/LED 驱动模块

芯片内置 LCD/LED 驱动模块，它们共用控制寄存器。具体的 SEG 和 COM 如下表所示：

	CMS89FT626	CMS89FT627	CMS89FT628
LCD	15SEG×3COM	15SEG×7COM	19SEG×7COM
LED	7SEG×3COM	7SEG×7COM	11SEG×7COM

CMS89FT62x 可驱动 1/2Bias、1/3Bias、1/4Bias 的 LCD，并且在设置完 LCD 数据并使能 LCD 控制位后，芯片可自动输出驱动 LCD。

12.1 LCD/LED 功能使能

将 LCDCON0(118H)的第 7 位 LCDEN 置 1，第 6 位 LEDEN 清 0，允许 LCD 驱动功能；

将 LCDCON0(118H)的第 6 位 LEDEN 置 1，第 7 位 LCDEN 清 0，允许 LED 驱动功能；

将 LCDEN 和 LEDEN 都置 0，关闭 LCD/LED 模块。

注：请不要将 LCDEN 和 LEDEN 同时置 1。

12.2 LCD/LED 功能管脚设置

若使能 LCD 驱动功能，必须设置相应的 SEG 口和 COM 口为输入态，即将相应的 TRIS 位置“1”；

若使能 LED 驱动功能，必须设置相应的 SEG 口和 COM 口为输出态，并输出“0”，即将相应的 TRIS 位和 PORT 位清“0”；

12.3 LCD/LED 功能 COM 口设置

LED 的 COM 口设置方式如下：

设置 I/O 口方向和数据寄存器，其中 LCD 功能设置相应管脚为输入态，LED 功能设置相应管脚为输出态并输出低电平。

COMSEL[2:0]	COM 口个数(包括 LCD/LED)	Bias(仅 LCD)
000	4	1/3
001	5	1/3
010	6	1/3
011	8	1/4
100	2	1/2
101	3	1/2
110	3	1/3
111	4	1/2

设置 COM_EN 寄存器，将相应管脚设置为 LED 功能的 COM 口。

如果用户在使用过程中，COM 口不按照顺序排列。例如将 COM3-COM6 作为 LED 功能的 COM 口，COM0-COM2 作为普通 I/O 口，可以做以下设置：

- 将 COM 口个数设置成 8 个 COM，COMSEL= “011”；
- 将 COM_EN 寄存器的 COM3-COM6 置 1，COM0-COM2、COM7 清 0。

此时，COM3-COM6 为 LED 功能的 COM 口，其输出占空比为 1/8。而 COM0-COM2、COM7 可作为普通 I/O 口。

12.4 LCD/LED 功能的 SEG 口设置

使能 LED 功能的 SEG 口必须满足以下条件：

1. 设置相应管脚状态，LCD 功能设置管脚为输入态，LED 功能设置相应管脚为输出态并输出“0”；
2. 设置 SEGEN0、SEGEN1、SEGEN2 寄存器中相应管脚为 LCD/LED 驱动功能，其中 SEG0-SEG10 为 LCD/LED 共用 SEG，而 SEG11-SEG18 只能作为 LCD 功能使用；
3. 设置 SEGEN2 寄存器中 SEG 口输出电流（仅 LED 功能）。

12.5 LCD/LED 功能的数据设置

设置 LCD/LED 显示数据需以下步骤：

1. 设置 LCDCON1 寄存器的 SEGOUT[1:0]位为 “1x”；
2. 设置 LCDADD 寄存器的第 7 位 LEDCS=1，允许读写数据；
3. 设置 LCDADD 的 0-6 位数据地址；
4. 设置 LCDDATA 数据（没有用作 LCD/LED 功能用的管脚，其相应的 LCDDATA 位需设置为 “0”）；
5. 重复步骤 3-4 设置其它地址数据；
6. 设置完成后关闭数据读写位 LCDCS=0。

LCD 地址和数据对应关系如下表：

LCDADD	LCDDATA								
00H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG0
01H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG1
.
.
.
.
11H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG17
12H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG18
	COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0	

12.6 LCD/LED 相关寄存器

LCD/LED 驱动功能相关寄存器有：控制寄存器 LCDCON0、LCDCON1；地址寄存器 LCDADD；数据寄存器 LCDDATA；口线设置寄存器 COMEN、SEGEN0、SEGEN1、SEGEN2。

LCD 控制寄存器 LCDCON0(118H)

118H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LDCON0	LCDEN	LEDEN	COMSEL[1:0]		LCDCLK[3:0]			
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7 LCDEN: LCD 模块使能;

0: 禁止 LCD 模块;

1: 使能 LCD 模块。

Bit6 LEDEN: LED 模块使能;

0: 禁止 LED 模块;

1: 使能 LED 模块。

Bit5~Bit4 COMSEL[1:0]: LCD/LED 模块 COM 口个数选择 (COMSEL[2]在 LCDADD 寄存器第 5 位);

000: 4COM, 1/3Bias;

001: 5COM, 1/3Bias;

010: 6COM, 1/3Bias;

011: 8COM, 1/4Bias;

100: 2COM, 1/2Bias;

101: 3COM, 1/2Bias;

110: 3COM, 1/3Bias;

111: 4COM, 1/2Bias。

Bit3~Bit0 LCDCLK[3:0]: LCD/LED 频率选择

0000: Fosc/64

0001: Fosc/128

0010: Fosc/256

0011: Fosc/512

0100: Fosc/1024

0101: Fosc/2048

0110: Fosc /4096

0111: Fosc/8192

1x00: Fosc/16384

1x01: Fosc/32768

1x10: Fosc/65536

1x11: Fosc/131072

LCD 控制寄存器 LCDCON1(119H)

119H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCON1	VADJEN	----	SEGOUT[1:0]		VLCD[3:0]			
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

- Bit7 VADJEN: VLCD 电压选择;
 0: VDD;
 1: $VDD \times (VLCD[3:0] + 13) / 28$ 。
- Bit6 保留 需写 0
- Bit5~Bit4 SEGOUT: SEG 口输出模式选择;
 00: SEG 口输出全为 0;
 01: SEG 口输出全为 1;
 1x: SEG 口输出为 LCDDATA 中的数据。
- Bit3~Bit0 VLCD: VLCD 参考电压, 当 VADJEN=1 的时候有效。

LCD 地址寄存器 LCDADD(11AH)

11AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDADD	LCDCS	B2RES	COMSEL[2]	LCDADD[4:0]				
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

- Bit7 LCDCS: LCD 数据读写使能;
 0: 禁止读写 LCD 数据;
 1: 允许读写 LCD 数据。
- Bit6 B2RES: LCD 功能 1/2bias 内部分压电阻选择;
 0: R=45K;
 1: R=90K。
- Bit5 COMSEL[2]: COM 选择最高位, COMSEL[2:0]设置 COM 口数量和 Bias;
- Bit4~Bit0 LCDADD[4:0]: LCD/LED 地址选择;
 LCD 地址范围 00H-17H
 LED 地址范围 00H-0AH

LCD 数据寄存器 LCDDATA(11BH)

11BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDDATA	LCDDATA[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

- Bit7~Bit0 LCDDATA[7:0]: LCD 数据设置, 写入 LCDADD 对应地址的数据。

LCD 功能 COM 口控制寄存器 COMEN(11FH)

11FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
COMEN	COM7EN	COM6EN	COM5EN	COM4EN	COM3EN	COM2EN	COM1EN	COM0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 COMxEN: COM 口功能设置;
 0: 对应 COMx 口为普通 I/O 口(x=0-7);
 1: 对应 COMx 口为 LCD/LED 功能的 COM 口(x=0-7)。

LCD 功能 SEG 口控制寄存器 SEGEN0(11EH)

11EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGEN0	SEG7EN	SEG6EN	SEG5EN	SEG4EN	SEG3EN	SEG2EN	SEG1EN	SEG0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 SEGxEN: SEG 口功能设置;
 0: 对应 SEGx 口为普通 I/O 口(x=0-7);
 1: 对应 SEGx 口为 LCD/LED 功能的 SEG 口(x=0-7)。

LED 功能 SEG 口控制寄存器 SEGEN1(11DH)

11DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGEN1	SEG15EN	SEG14EN	SEG13EN	SEG12EN	SEG11EN	SEG10EN	SEG9EN	SEG8EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 SEGxEN: SEG 口功能设置;
 0: 对应 SEGx 口为普通 I/O 口(x=8-15);
 1: 对应 SEGx 口为 LCD/LED 功能的 SEG 口(x=8-15)。

LED 功能 SEG 口控制寄存器 SEGEN2(11CH)

11CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGEN2	SEGDR1[3:0]				----	SEG18EN	SEG17EN	SEG16EN
R/W	R/W	R/W	R/W	R/W	----	R/W	R/W	R/W
复位值	0	0	0	0	----	0	0	0

Bit7~Bit4 SEGDR1[3:0]: SEG 口驱动电流设置; (以 VDD=5V $V_O=0.7V_{DD}$ 计算)

0000: SEG 口驱动电流为 0;

0001: SEG 口驱动电流为 2mA;

0010: SEG 口驱动电流为 4mA;

0011: SEG 口驱动电流为 6mA;

...

1110: SEG 口驱动电流为 28mA;

1111: SEG 口驱动电流为 30mA。

Bit3 未用。

Bit2~Bit0 SEGxEN: SEG 口功能设置;

0: 对应 SEGx 口为普通 I/O 口(x=16-18);

1: 对应 SEGx 口为 LCD 功能的 SEG 口(x=16-18)。

13. 程序 EEPROM 控制

13.1 程序 EEPROM 概述

程序 EEPROM 在正常工作状态下是可读写的。这些存储器并不直接映射到寄存器文件空间，而是通过特殊功能寄存器（SFR）对其进行间接寻址。共有 5 个 SFR 寄存器用于访问这些存储器：

- EECON1
- EECON2
- EEDAT
- EEDATH
- EEADR

当与数据存储器模块接口时，EEDAT 和 EEDATH 寄存器形成一个双字节字用于保存要读/写的 16 位数据，而 EEADR 寄存器存放被访问的 EEDAT 单元的地址。该系列中的器件具有 32 字的程序 EEPROM，地址范围为 00H 到 1FH。

写入时间由片上定时器控制。写入和擦除电压是由片上电荷泵产生的，此电荷泵额定工作在器件的电压范围内，用于进行字节或字操作。

当器件受代码保护时，CPU 仍可继续读写程序 EEPROM。代码保护时，器件编程器将不再能访问数据。

13.2 相关寄存器

13.2.1 EEADR 寄存器

EEADR 寄存器能寻址最大 32 字程序 EEPROM。

13.2.2 EECON1 和 EECON2 寄存器

EECON1 是访问 EE 存储器的控制寄存器。

控制位 EEPGD 需要置 1 后才能操作数据存储。该位被清零时，和复位时一样，任何后续操作都是无效的。

控制位 RD 和 WR 分别启动读和写。用软件只能将这些位置 1 而无法清零。在读或写操作完成后，由硬件将它们清零。由于无法用软件将 WR 位清零，从而可避免意外地过早终止写操作。

- 当 WREN 置 1 时，允许对程序 EEPROM 执行写操作。上电时，WREN 位被清零。当正常的写入操作被 LVR 复位或 WDT 超时复位中断时，WRERR 位会置 1。在这些情况下，复位后用户可以检查 WRERR 位并重写相应的单元。
- 当写操作完成时 PIR2 寄存器中的中断标志位 EEIF 被置 1。此标志位必须用软件清零。

EECON2 不是物理寄存器。读 EECON2 得到的是全 0。

EECON2 寄存器仅在执行程序 EEPROM 写序列时使用。

EEPROM 数据寄存器 EEDATA(10CH)

10CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 EEDAT<7:0>: 对程序EEPROM进行读取或写入的数据的低8位。

EEPROM 地址寄存器 EEADR(10DH)

10DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADR	---	---	---	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0
读写	---	---	---	R/W	R/W	R/W	R/W	R/W
复位值	---	---	---	0	0	0	0	0

Bit7~Bit5 ---- (无关)

Bit4~Bit0 EEADR<4:0>: 指定EEPROM读取/写入操作的地址。

EEPROM 数据寄存器 EEDATH(10EH)

10EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 EEDATH<7:0>: 对程序EEPROM进行读取或写入数据的高8位。

EEPROM 控制寄存器 EECON1 (18CH)

18CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON1	EEPGD	---	---	---	WRERR	WREN	WR	RD
读写	R/W	---	---	---	R/W	R/W	R/W	R/W
复位值	0	---	---	---	X	0	0	0

- Bit7 EEPGD: 程序EEPROM使能位;
 1= 允许操作程序EEPROM;
 0= 禁止操作程序EEPROM。
- Bit6~Bit4 未用, 读为0。
- Bit3 WRERR: EEPROM错误标志位;
 1= 写操作过早终止 (正常工作期间的任何WDT复位或欠压复位);
 0= 写操作完成。
- Bit2 WREN: EEPROM写使能位;
 1= 允许写周期;
 0= 禁止写入程序EEPROM。
- Bit1 WR: 写控制位;
 1= 启动写周期 (写操作一旦完成由硬件清零该位, 用软件只能将WR位置1, 但不能清零);
 0= 程序EEPROM写周期完成。
- Bit0 RD: 读控制位;
 1= 启动存储器读操作 (由硬件清零RD, 用软件只能将RD位置1, 但不能清零);
 0= 不启动存储器读操作。

13.3 读程序 EEPROM

要读取数据存储器单元,用户必须将地址写入 EEADR 寄存器,将 EECON1 寄存器的 EEPGD 控制位置 1,然后将控制位 RD 置 1。一旦设置好读控制位,数据存储器控制器将使用第二个指令周期来读数据。这会导致紧随“SETB EECON1,RD”指令的第二条指令被忽略(1)。在紧接着的下一个周期 EEDAT 寄存器中就有数据了。EEDAT 将保存此值直至下一次用户向该单元读取或写入数据时为止。

注: 程序存储器读操作后的两条指令必须为 NOP。这可阻止用户在 RD 位置 1 后的下一条指令执行双周期指令。

读程序 EEPROM

LD	A,EE_ADD	;将要读取的地址放入 EEADR 寄存器
LD	EEADR,A	
SETB	EECON1,EEPGD	;使能程序 EEPROM
SETB	EECON1,RD	;使能读信号
NOP		;这里读取数据, 必须加 NOP 指令
NOP		
LD	A,EEDATA	;读取数据到 ACC

13.4 写程序 EEPROM

要写 EEPROM 数据存储单元，用户应首先将该单元的地址写入 EEADR 寄存器并将数据写入 EEDATA 寄存器。然后用户必须按特定顺序开始写入每个字节。

如果没有完全按照下面的指令顺序（即首先将 55h 写入 EECON2，随后将 Aah 写入 EECON2，最后将 WR 位置 1）写每个字节，将不会启动写操作。在该代码段中应禁止中断。

此外，必须将 EECON1 中的 WREN 位置 1 以使能写操作。这种机制可防止由于代码执行错误（异常）（即程序跑飞）导致误写 EEPROM。在不更新 EEPROM 时，用户应该始终保持 WREN 位清零。WREN 位不能被硬件清零。

一个写过程启动后，将 WREN 位清零将不会影响此写周期。除非 WREN 位置 1，否则 WR 位将无法置 1。写周期完成时，WR 位由硬件清零并且 EE 写完成中断标志位 (EEIF) 置 1。用户可以允许此中断或查询此位。EEIF 必须用软件清零。

执行了 SETB EECON1,WR 指令之后，处理器需要 2 个指令周期以设置擦除/写操作。用户必须在将 WR 位置 1 的指令后放置两条 NOP 指令。执行完写操作指令后，处理器会使内部操作暂停 4ms（可选）时间。因为时钟和外设仍继续工作，所以这并不是休眠模式。写周期结束后，处理器将从 EECON1 写指令后的第三条指令恢复工作。

写程序 EEPROM

LD	A,ADDR1	;写地址
LD	EEADR,A	
LD	A,DATAL	;写数据
LD	EEDAT,A	
LD	A,DATAH	
LD	EEDATH,A	
SETB	EECON1,EEPGD	;允许操作 EEPROM
SETB	EECON1,WREN	;使能写信号
CLRB	INTCON,GIE	;关闭中断
SZB	INTCON,GIE	;确认中断关闭
JP	\$_2	
LDIA	055H	;给 EECON2 寄存器写 55H 跟 0AAH
LD	EECON2,A	
LDIA	0AAH	
LD	EECON2,A	
SETB	EECON1,WR	;开始写程序存储器
NOP		;写缓冲需要延时
NOP		
CLRB	EECON1,WREN	
SETB	INTCON,GIE	;打开中断
SZB	EECON1,WR	;判断写操作是否完成,写过程中 WREN 位必须保持 1
JP	\$_1	
CLRB	EECON1,WREN	;写结束, 关闭写使能位

13.5 EPROM 操作注意事项

13.5.1 写校验

根据具体的应用，好的编程习惯一般要求将写入程序 EEPROM 的值对照期望值进行校验。

13.5.2 避免误写的保护

有些情况下，用户可能不希望向程序 EEPROM 写入数据。为防止误写 EEPROM，芯片内嵌了各种保护机制。上电时清零 WREN 位。而且，上电延时定时器（延迟时间为 18ms）会防止对 EEPROM 执行写操作。

写操作的启动序列以及 WREN 位将共同防止在以下情况下发生误写操作：

- 欠压
- 电源毛刺
- 软件故障

14. 触摸按键功能

该系列单片机提供了多个触摸按键功能。该按键功能内建于单片机内，不需外接元件，通过内部寄存器对其进行简单的操作。

14.1 触摸按键结构

触摸按键与 RB 和 RC 的 I/O 引脚共用。通过寄存器的位来选择此功能。按键被分成若干个模块，M0~M2。每个模块有 4 个按键，每个按键有自己的振荡器。每个模块都具有自己的控制逻辑电路和设置寄存器。寄存器的名称和它对应的模块编号相关联。

单片机	按键数(个)	触摸按键模块	触摸按键	共用 I/O 口
CMS89FT626/CMS89FT627	8	M0	KEY1-KEY4	RB0~RB3
		M1	KEY5-KEY8	RB4~RB7
CMS89FT628	12	M0	KEY1-KEY4	RB0~RB3
		M1	KEY5-KEY8	RB4~RB7
		M2	KEY9-KEY12	RC0~RC3

14.2 触摸按键寄存器描述

触摸按键模块包含 4 个触摸按键功能，且都有自己相匹配的寄存器。以下表格显示了触摸按键模块的寄存器设置。寄存器名中的 Mn 代表触摸按键模块的编号。CMS89FT626 和 CMS89FT627 的是 M0~M1，CMS89FT628 的是 M0~M2。

触摸按键寄存器描述

名称	作用
TKTMR	触摸按键 8 位定时/计数器寄存器
TKC0	计数器开关和清零控制/ 参考时钟控制/TKST 开始位
TK16DH	触摸按键 16 位 C/F 高字节计数器
TK16DL	触摸按键 16 位 C/F 低字节计数器
TKC1	触摸按键振荡频率选择
TKMn16DH	模块 n 16 位 C/F 高字节计数器
TKMn16DL	模块 n 16 位 C/F 低字节计数器
TKMnROL	参考振荡器内建电容选择
TKMnROH	参考振荡器内建电容选择
TKMnC0	控制寄存器 0，复用按键选择
TKMnC1	控制寄存器 1，按键选择，I/O 引脚或触控引脚选择

TKTMR 寄存器 (115H)

115H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 触摸按键8 位定时/ 计数器寄存器
 时隙计数器溢出设定的时间为 $(256-TKTMR[7:0]) \times 32$

TKC0 寄存器 (116H)

116H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKC0	---	TKRCOV	TKST	TKCFOV	TK16OV	TSCS	TK16S1	TK16S0
读写	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	0	0	0	0	0	0	0

- Bit7 未用。
- Bit6 TKRCOV: 时隙计数器溢出标志位。
 0: 无溢出。
 1: 溢出。
 如果触摸模块的时隙计数器溢出，则触摸按键中断请求标志位将会被置位(TKMF)且所有模块按键振荡器和参考振荡器自动停止。模块n的16位C/F计数器、5位时隙计数器、8位时隙定时/计数器和触摸模块16位计数器都会自动关闭。
- Bit5 TKST: 开启触摸按键检测控制位。
 0: 停止。
 0→1: 开启。
 当该位为“0”时，模块n的16位C/F计数器、5位时隙计数器和触摸模块16位计数器会自动清零（8位可编程时隙计数器不清，由用户设定溢出时间）。当该位由0→1时，16位C/F计数器、5位时隙计数器、8位时隙定时/计数器和触摸模块16位计数器都会自动开启，并使能按键振荡器和参考振荡器输出时钟输入到这些计数器。
- Bit4 TKCFOV 触摸按键模块16位C/F计数器溢出标志位。
 0: 无溢出。
 1: 溢出。
 该位必须通过软件清零。
- Bit3 TK16OV: 触摸按键模块16位计数器溢出标志位。
 0: 无溢出。
 1: 溢出。
 该位必须通过软件清零。
- Bit2 TSCS: 触摸按键时隙计数器选择。
 0: 每个模块使用自己的时隙计数器。
 1: 所有模块使用模块0的时隙计数器。
- Bit1~Bit0 TK16S1~TK16S0: 触摸按键模块16位计数器时钟选择位。
 00: F_{SYS} 。
 01: $F_{SYS}/2$ 。
 10: $F_{SYS}/4$ 。
 11: $F_{SYS}/8$ 。

TKC1 寄存器 (117H)

117H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKC1	---	---	---	---	---	---	TKFS1	TKFS0
读写	---	---	---	---	---	---	R/W	R/W
复位值	---	---	---	---	---	---	1	1

Bit7~Bit2

未用。

Bit1~Bit0

TKFS1~TKFS0: 触摸按键振荡器频率选择位。

00: 500KHz。

01: 1000KHz。

10: 1500KHz。

11: 2000KHz。

TK16DL 寄存器(113H)

113H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
读写	R	R	R	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 触摸按键模块 16 位计数器低字节内容。

TK16DH 寄存器 (114H)

114H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TK16DH	D7	D6	D5	D4	D3	D2	D1	D0
读写	R	R	R	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 触摸按键模块 16 位计数器高字节内容。

TKMn16DL 寄存器

---	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKMn16DL	D7	D6	D5	D4	D3	D2	D1	D0
读写	R	R	R	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 模块 n 16 位计数器低字节内容。

TKMn16DH 寄存器

---	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKMn16DH	D7	D6	D5	D4	D3	D2	D1	D0
读写	R	R	R	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 模块 n 16 位计数器高字节内容。

TKMnROL 寄存器

---	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKMnROL	D7	D6	D5	D4	D3	D2	D1	D0
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 参考振荡器内建电容选择。
振荡器内建电容选择为(TKMnRO[9:0] × 32pF) / 1024。

TKMnROH 寄存器

---	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKMnROH	---	---	---	---	---	---	D9	D8
读写	---	---	---	---	---	---	R/W	R/W
复位值	---	---	---	---	---	---	0	0

Bit7~Bit0 参考振荡器内建电容选择。

TKMnC0 寄存器

---	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKMnC0	MnMXS1	MnMXS0	MnDFEN	MnFILEN	MnSOFC	MnSOF2	MnSOF1	MnSOF0
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

- Bit7~Bit6 MnMXS1,MnMXS0: 复用按键选择。
00: KEY1。
01: KEY2。
10: KEY3。
11: KEY4。
- Bit5 MnDFEN: 倍频功能控制。
0: 除能。
1: 使能。
- Bit4 MnFILEN: 滤波器功能控制。
0: 除能。
1: 使能。
- Bit3 MnSOFC: C-F 振荡器跳率功能选择。
0: 由软件处理跳频功能, MnSOF2~ MnSOF0位决定C-F振荡器微调频率。
1: 由硬件处理跳频功能, MnSOF2~ MnSOF0位无作用。
- Bit2~Bit0 MnSOF2~MnSOF0: 由软件选择按键振荡器或参考振荡器频率作为C-F振荡器频率。
000: 1380KHz。
001: 1500KHz。
010: 1670KHz。
011: 1830KHz。
100: 2000KHz。
101: 2230KHz。
110: 2460KHz。
111: 2740KHz。

上述频率会依外挂或内建电容值的不同而变化。按键振荡器频率选择为 2MHz，用户选择其它频率时可依比例调整。

TKMnC1 寄存器

---	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TKMnC1	MnTSS	---	MnROEN	MnKOEN	MnK4IO	MnK3IO	MnK2IO	MnK1IO
读写	R/W	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	---	0	0	0	0	0	0

- Bit7 MnTSS: 时隙计数器选择。
 0: 参考振荡器。
 1: $F_{sys}/4$ 。
- Bit6 未用
- Bit5 MnROEN: 参考振荡器控制。
 0: 除能。
 1: 使能。
- Bit4 MnKOEN: 按键振荡器控制。
 0: 除能。
 1: 使能。
- Bit3 MnK4IO: I/O引脚和触摸按键4功能选择。
 0: I/O引脚。
 1: 触摸按键。
- Bit2 MnK3IO: I/O引脚和触摸按键3功能选择。
 0: I/O引脚。
 1: 触摸按键。
- Bit1 MnK2IO: I/O引脚和触摸按键2功能选择。
 0: I/O引脚。
 1: 触摸按键。
- Bit0 MnK1IO: I/O引脚和触摸按键1功能选择。
 0: I/O引脚。

MnK4OEN	M0	M1	M2
	RB3/KEY4	RB7/KEY8	RC3/KEY12
0	I/O		
1	触摸按键输入		

MnK3OEN	M0	M1	M2
	RB2/KEY3	RB6/KEY7	RC2/KEY11
0	I/O		
1	触摸按键输入		

MnK2OEN	M0	M1	M2
	RB1/KEY2	RB5/KEY6	RC1/KEY10
0	I/O		
1	触摸按键输入		

MnK1OEN	M0	M1	M2
	RB0/KEY1	RB4/KEY5	RC0/KEY9
0	I/O		
1	触摸按钮输入		

14.3 触摸按键操作

手指接近或接触到触控面板时，面板的电容量会增大，电容量的变化会轻微改变内部感应振荡器的频率，通过测量频率的变化可以感知触控动作。参考时钟通过内部可编程分频器能够产生一个固定的时间周期。在这个时间周期内，通过对感应振荡器产生的时钟周期计数，可确定触摸按键的动作。

在参考时钟固定的时间间隔内，感应振荡器产生的时钟周期数是可以测量的。这个周期数可以用于判断触摸动作是否发生。触摸按键模块包含 4 个与 I/O 引脚共用的触摸按键。通过寄存器可设置相应引脚功能。

通过 TKC0 寄存器的 TSCS 位可选择模块 0 时隙计数器作为所有模块的时隙计数器。所有模块启动信号相同。TSCS 位清零时，所有模块的 16 位 C/F 计数器、16 位计数器、5 位时隙计数器和 8 位时隙定时/计数器会自动清零除了 8 位可编程时隙计数器，由用户设定溢出时间。该位从低电平变为高电平时，16 位 C/F 计数器、16 位计数器和 5 位时隙计数器和 8 位时隙定时/计数器会自动开启。

- 当 5 位时隙计数器溢出时，所有模块的按键振荡器和参考振荡器会自动停止且 16 位 C/F 计数器、16 位计数器、5 位时隙计数器和 8 位时隙定时/计数器会自动停止。5 位时隙计数器和 8 位时隙定时/计数器时钟来自参考振荡器或 $F_{sys}/4$ 。通过 TKMnC1 寄存器中的 MnROEN 和 MnKOEN 位可使能参考振荡器和按键振荡器。
- 当所有使能的按键模块或模块 0 的时隙计数器溢出时，中断产生。

每个触摸按键模块包含 4 个按键，模块 0 包含 KEY1~KEY4，模块 1 包含 KEY5~KEY8，模块 2 包含 KEY9~KEY12。每个触摸按键模块都有相同的结构。

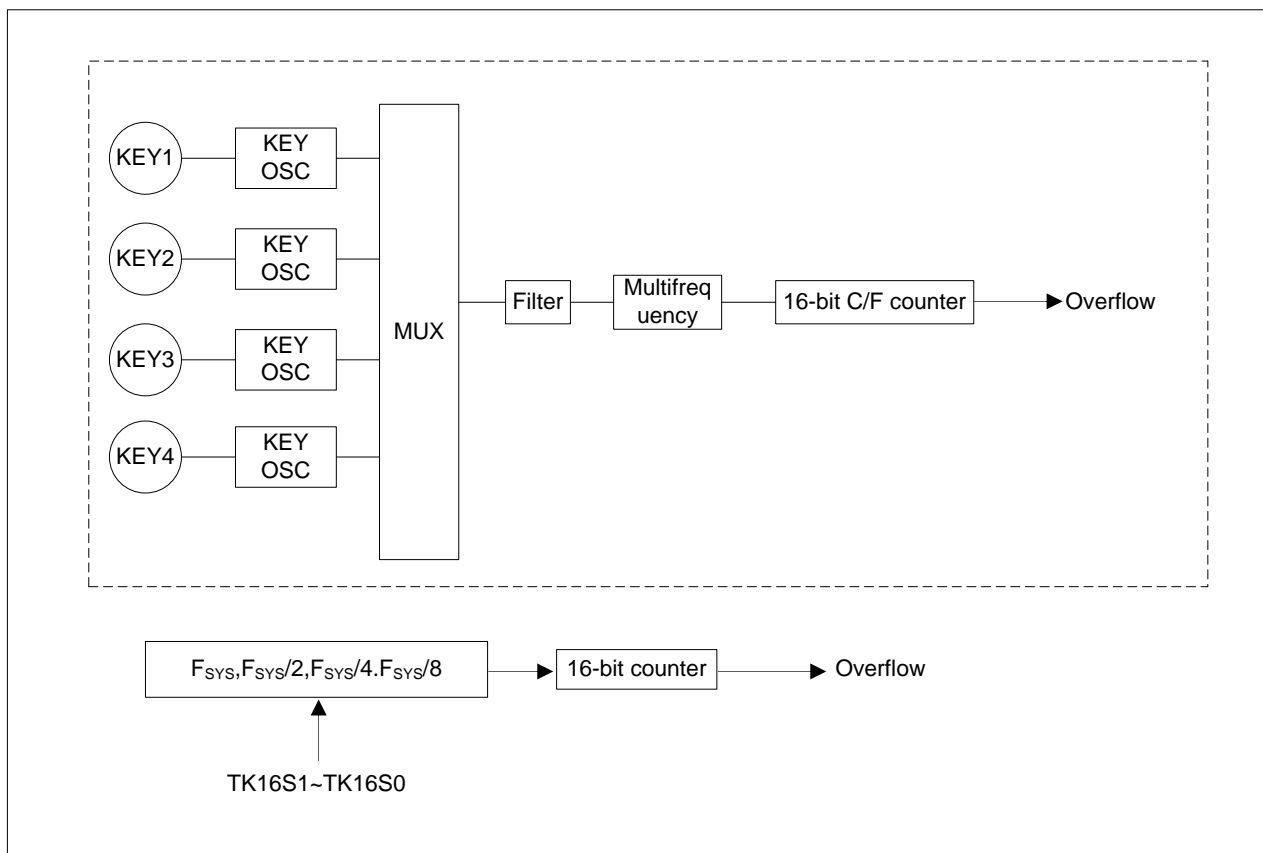


图 14-1: 触摸按键模块结构

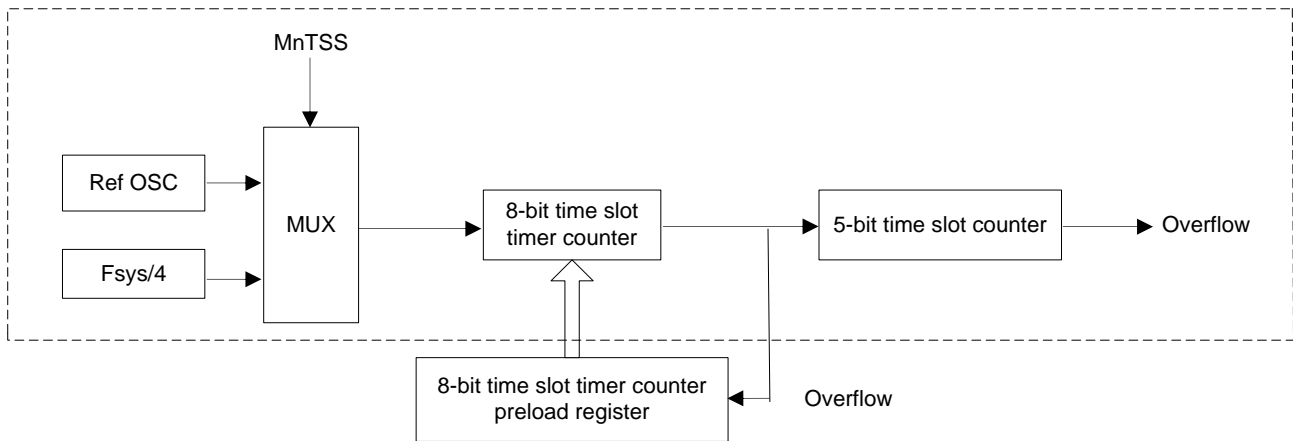


图 14-2: 触摸按键模块方框图

注：虚线部分是每个触摸按键都单独有的部分。

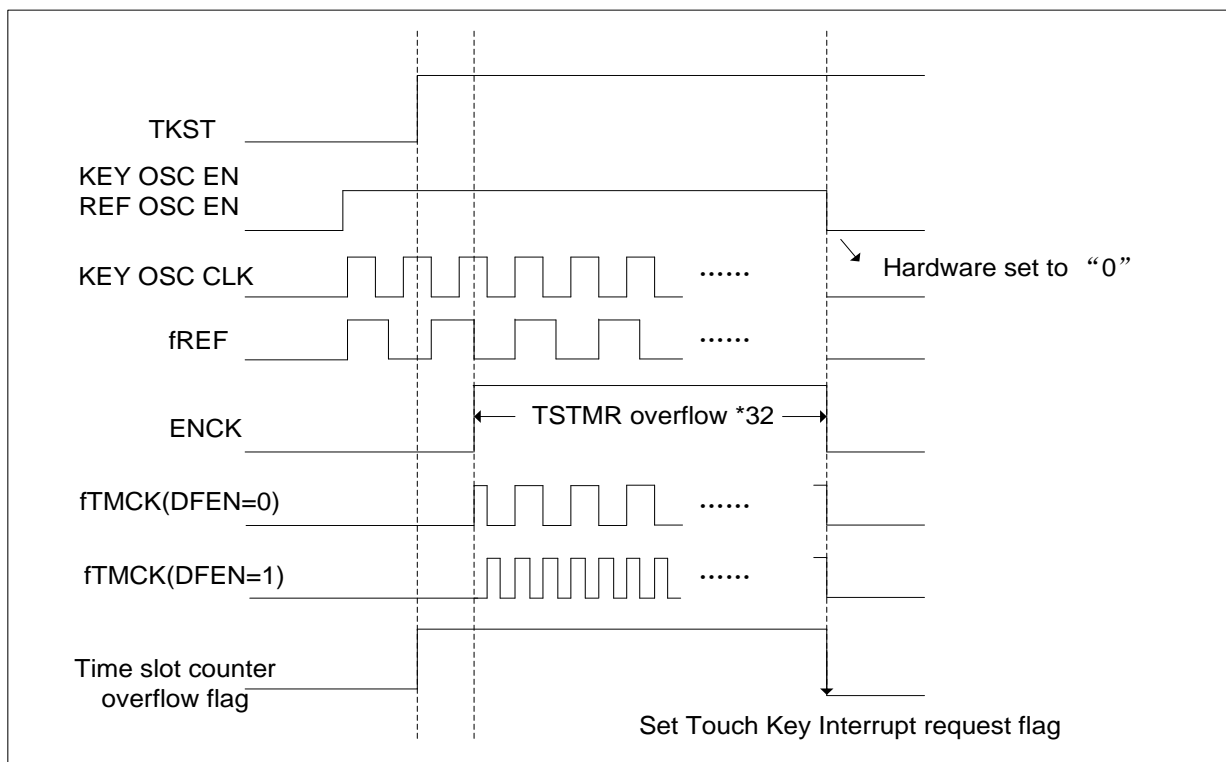


图 14-3: 触摸按键感应振荡器和参考振荡器时序方框图

14.4 触摸按键中断

触摸按键只有一个中断。所有触摸按键模块或模块 0 的时隙计数器溢出时，才产生中断。此时，所有模块的 16 位 C/F 计数器、16 位计数器、5 位时隙计数器和 8 位时隙计数器会自动清零。只有使能的按键都溢出时才产生中断。

触摸按键模块的 16 位 C/F 计数器溢出就会把 16 位 C/F 计数器溢出标志位 TKCFOV 设为“1”，此中断标志位不会自动复位，必须通过应用程序将其复位。

触摸模块只包含一个 16 位计数器。触摸按键模块的 16 位计数器溢出就会把 16 位计数器溢出标志位 TK16OV 设为“1”，此中断标志位不会自动复位，必须通过应用程序将其复位。

编程注意事项：

1. 相关寄存器设置后，TKST 位由低电平变为高电平，触摸按键检测程序初始化。此时所有相关的振荡器将使能并同步。时隙计数器标志位 TKRCOV 将变为高电平直到计数器溢出。计数器溢出发生时，将会产生一个中断信号。
2. 当外部触摸按键的大小和布局确定时，其相关的电容将决定感应振荡器的频率。

15. 电气参数

15.1 DC 特性

符号	参数	测试条件		最小	典型	最大	单位
		VDD	条件				
VDD	工作电压	-	8M	3.0	-	5.5	V
		-	4M	2.5	-	5.5	V
I _{DD}	工作电流	5V	ADC 使能	-	3	-	mA
		3V	ADC 使能	-	2	-	mA
I _{STB}	静态电流	5V	----	-	0.5	10	uA
		3V	----	-	0.1	5	uA
V _{IL}	低电平输入电压	-	----	-	-	0.3VDD	V
V _{IH}	高电平输入电压	-	----	0.7VDD	-	-	V
V _{OH}	高电平输出电压	-	不带负载	0.9VDD	-	-	V
V _{OL}	低电平输出电压	-	不带负载	-	-	0.1VDD	V
V _{ADI}	AD 口输入电压	-	----	0	-	VDD	V
V _{AD}	AD 模块工作电压	-	----	2.7	-	5.5	V
V _{EEPROM}	EEPROM 模块工作电压	-	----	3.0	-	5.5	V
E _{AD}	AD 转换误差	-	----	-	±2	-	-
R _{PH}	上拉电阻阻值	5V	----	-	35	-	K
		3V	----	-	65	-	K
I _{OL}	输出口灌电流	5V	V _{OL} =0.3VDD	-	60	-	mA
		3V	V _{OL} =0.3VDD	-	25	-	mA
I _{OH}	输出口拉电流	5V	V _{OH} =0.7VDD	-	15	-	mA
		3V	V _{OH} =0.7VDD	-	10	-	mA

15.2 AC 特性

符号	参数	测试条件		最小	典型	最大	单位
		VDD	条件				
T _{WDT}	WDT 复位时间	5V	---	-	18	-	ms
		3V	---	-	36	-	ms
T _{AD}	AD 转换时间	5V	---	-	49	-	CLK
		3V	---	-	49	-	CLK
T _{EEPROM}	EEPROM 写入时间	5V	---	-	2.5	-	ms
		3V	---	-	2.5	-	ms

15.3 内部 RC 振荡特性

15.3.1 内部 RC 振荡电压特性

测试条件	振荡频率（典型值）(Hz)
2.5V	8.0M
2.6V	8.1M
2.8V	8.2M
3.0V	8.3M
3.2V	8.3M
3.4V	8.2M
3.6V	8.2M
3.8V	8.2M
4.0V	8.1M
4.2V	8.1M
4.4V	8.1M
4.6V	8.0M
4.8V	8.0M
5.0V	8.0M
5.2V	8.0M
5.4V	7.9M
5.5V	7.8M

15.3.2 内部 RC 振荡温度特性

测试条件	-20℃	25℃	40℃	60℃	85℃
振荡频率（典型值）(Hz)	7.9M	8.0M	8.0M	8.1M	8.1M

16. 指令

16.1 指令一览表

助记符	操作	指令周期	标志
控制类			
NOP	空操作	1	None
STOP	进入休眠模式	1	TO,PD
CLRWDT	清零看门狗计数器	1	TO,PD
数据传送			
LD [R],A	将 ACC 内容传送到 R	1	NONE
LD A,[R]	将 R 内容传送到 ACC	1	Z
TESTZ [R]	将数据存储器内容传给数据存储器	1	Z
LDIA i	立即数 i 送给 ACC	1	NONE
逻辑运算			
CLRA	清零 ACC	1	Z
SET [R]	置位数据存储器 R	1	NONE
CLR [R]	清零数据存储器 R	1	Z
ORA [R]	R 与 ACC 内容做“或”运算，结果存入 ACC	1	Z
ORR [R]	R 与 ACC 内容做“或”运算，结果存入 R	1	Z
ANDA [R]	R 与 ACC 内容做“与”运算，结果存入 ACC	1	Z
ANDR [R]	R 与 ACC 内容做“与”运算，结果存入 R	1	Z
XORA [R]	R 与 ACC 内容做“异或”运算，结果存入 ACC	1	Z
XORR [R]	R 与 ACC 内容做“异或”运算，结果存入 R	1	Z
SWAPA [R]	R 寄存器内容的高低半字节转换，结果存入 ACC	1	NONE
SWAPR [R]	R 寄存器内容的高低半字节转换，结果存入 R	1	NONE
COMA [R]	R 寄存器内容取反，结果存入 ACC	1	Z
COMR [R]	R 寄存器内容取反，结果存入 R	1	Z
XORIA i	ACC 与立即数 i 做“异或”运算，结果存入 ACC	1	Z
ANDIA i	ACC 与立即数 i 做“与”运算，结果存入 ACC	1	Z
ORIA i	ACC 与立即数 i 做“或”运算，结果存入 ACC	1	Z
移位操作			
RRCA [R]	数据存储器带进位循环右移一位，结果存入 ACC	1	C
RRCR [R]	数据存储器带进位循环右移一位，结果存入 R	1	C
RLCA [R]	数据存储器带进位循环左移一位，结果存入 ACC	1	C
RLCR [R]	数据存储器带进位循环左移一位，结果存入 R	1	C
RLA [R]	数据存储器不带进位循环左移一位，结果存入 ACC	1	NONE
RLR [R]	数据存储器不带进位循环左移一位，结果存入 R	1	NONE
RRA [R]	数据存储器不带进位循环右移一位，结果存入 ACC	1	NONE
RRR [R]	数据存储器不带进位循环右移一位，结果存入 R	1	NONE
递增递减			
INCA [R]	递增数据存储器 R，结果放入 ACC	1	Z
INCR [R]	递增数据存储器 R，结果放入 R	1	Z
DECA [R]	递减数据存储器 R，结果放入 ACC	1	Z
DECR [R]	递减数据存储器 R，结果放入 R	1	Z

助记符	操作	指令周期	标志
位操作			
CLRB	[R],b 将数据存储器 R 中某位清零	1	NONE
SETB	[R],b 将数据存储器 R 中某位置一	1	NONE
查表			
TABLE	[R] 读取 FLASH 内容结果放入 TABLE_DATAH 与 R	2	NONE
TABLEA	读取 FLASH 内容结果放入 TABLE_DATAH 与 ACC	2	NONE
数学运算			
ADDA	[R] ACC+[R]→ACC	1	C,DC,Z,OV
ADDR	[R] ACC+[R]→R	1	C,DC,Z,OV
ADDCA	[R] ACC+[R]+C→ACC	1	Z,C,DC,OV
ADDCR	[R] ACC+[R]+C→R	1	Z,C,DC,OV
ADDIA	i ACC+i→ACC	1	Z,C,DC,OV
SUBA	[R] [R]-ACC→ACC	1	C,DC,Z,OV
SUBR	[R] [R]-ACC→R	1	C,DC,Z,OV
SUBCA	[R] [R]-ACC-C→ACC	1	Z,C,DC,OV
SUBCR	[R] [R]-ACC-C→R	1	Z,C,DC,OV
SUBIA	i i-ACC→ACC	1	Z,C,DC,OV
HSUBA	[R] ACC-[R]→ACC	1	Z,C,DC,OV
HSUBR	[R] ACC-[R]→R	1	Z,C,DC,OV
HSUBCA	[R] ACC-[R]- \overline{C} →ACC	1	Z,C,DC,OV
HSUBCR	[R] ACC-[R]- \overline{C} →R	1	Z,C,DC,OV
HSUBIA	i ACC-i→ACC	1	Z,C,DC,OV
无条件转移			
RET	从子程序返回	2	NONE
RET	i 从子程序返回, 并将立即数 I 存入 ACC	2	NONE
RETI	从中断返回	2	NONE
CALL	ADD 子程序调用	2	NONE
JP	ADD 无条件跳转	2	NONE
条件转移			
SZB	[R],b 如果数据存储器 R 的 b 位为“0”, 则跳过下一条指令	1 or 2	NONE
SNZB	[R],b 如果数据存储器 R 的 b 位为“1”, 则跳过下一条指令	1 or 2	NONE
SZA	[R] 数据存储器 R 送至 ACC, 若内容为“0”, 则跳过下一条指令	1 or 2	NONE
SZR	[R] 数据存储器 R 内容为“0”, 则跳过下一条指令	1 or 2	NONE
SZINCA	[R] 数据存储器 R 加“1”, 结果放入 ACC, 若结果为“0”, 则跳过下一条指令	1 or 2	NONE
SZINCR	[R] 数据存储器 R 加“1”, 结果放入 R, 若结果为“0”, 则跳过下一条指令	1 or 2	NONE
SZDECA	[R] 数据存储器 R 减“1”, 结果放入 ACC, 若结果为“0”, 则跳过下一条指令	1 or 2	NONE
SZDECR	[R] 数据存储器 R 减“1”, 结果放入 R, 若结果为“0”, 则跳过下一条指令	1 or 2	NONE

16.2 指令说明

ADDA [R]

操作: 将 R 加 ACC, 结果放入 ACC

周期: 1

影响标志位: C, DC, Z, OV

举例:

LDIA	09H	;给 ACC 赋值 09H
LD	R01,A	;将 ACC 的值 (09H) 赋给自定义寄存器 R01
LDIA	077H	;给 ACC 赋值 77H
ADDA	R01	;执行结果: ACC=09H + 77H =80H

ADDR [R]

操作: 将 R 加 ACC, 结果放入 R

周期: 1

影响标志位: C, DC, Z, OV

举例:

LDIA	09H	;给 ACC 赋值 09H
LD	R01,A	;将 ACC 的值 (09H) 赋给自定义寄存器 R01
LDIA	077H	;给 ACC 赋值 77H
ADDR	R01	;执行结果: R01=09H + 77H =80H

ADDCA [R]

操作: 将 R 加 ACC 加 C 位, 结果放入 ACC

周期: 1

影响标志位: C, DC, Z, OV

举例:

LDIA	09H	;给 ACC 赋值 09H
LD	R01,A	;将 ACC 的值 (09H) 赋给自定义寄存器 R01
LDIA	077H	;给 ACC 赋值 77H
ADDCA	R01	;执行结果: ACC= 09H + 77H + C=80H (C=0) ACC= 09H + 77H + C=81H (C=1)

ADDCR [R]

操作: 将 R 加 ACC 加 C 位, 结果放入 R

周期: 1

影响标志位: C, DC, Z, OV

举例:

LDIA	09H	;给 ACC 赋值 09H
LD	R01,A	;将 ACC 的值 (09H) 赋给自定义寄存器 R01
LDIA	077H	;给 ACC 赋值 77H
ADDCR	R01	;执行结果: R01 = 09H + 77H + C=80H (C=0) R01 = 09H + 77H + C=81H (C=1)

ADDIA **i**

操作: 将立即数 i 加 ACC, 结果放入 ACC

周期: 1

影响标志位: C, DC, Z, OV

举例:

```
LDIA            09H                    ;给 ACC 赋值 09H
ADDIA           077H                  ;执行结果: ACC = ACC(09H) + i(77H)=80H
```

ANDA **[R]**

操作: 寄存器 R 和 ACC 进行逻辑与运算, 结果放入 ACC

周期: 1

影响标志位: Z

举例:

```
LDIA            0FH                    ;给 ACC 赋值 0FH
LD              R01,A                  ;将 ACC 的值(0FH)赋给寄存器 R01
LDIA            77H                    ;给 ACC 赋值 77H
ANDA            R01                    ;执行结果: ACC=(0FH and 77H)=07H
```

ANDR **[R]**

操作: 寄存器 R 和 ACC 进行逻辑与运算, 结果放入 R

周期: 1

影响标志位: Z

举例:

```
LDIA            0FH                    ;给 ACC 赋值 0FH
LD              R01,A                  ;将 ACC 的值(0FH)赋给寄存器 R01
LDIA            77H                    ;给 ACC 赋值 77H
ANDR            R01                    ;执行结果: R01=(0FH and 77H)=07H
```

ANDIA **i**

操作: 将立即数 i 与 ACC 进行逻辑与运算, 结果放入 ACC

周期: 1

影响标志位: Z

举例:

```
LDIA            0FH                    ;给 ACC 赋值 0FH
ANDIA           77H                    ;执行结果: ACC =(0FH and 77H)=07H
```

CALL **add**

操作: 调用子程序

周期: 2

影响标志位: 无

举例:

```
CALL            LOOP                  ;调用名称定义为"LOOP"的子程序地址
```

CLRA

操作: ACC 清零
周期: 1
影响标志位: Z
举例:

CLRA ;执行结果: ACC=0

CLR [R]

操作: 寄存器 R 清零
周期: 1
影响标志位: Z
举例:

CLR R01 ;执行结果: R01=0

CLRB [R],b

操作: 寄存器 R 的第 b 位清零
周期: 1
影响标志位: 无
举例:

CLRB R01,3 ;执行结果: R01 的第 3 位为零

CLRWDT

操作: 清零看门狗计数器
周期: 1
影响标志位: TO, PD
举例:

CLRWDT ;看门狗计数器清零

COMA [R]

操作: 寄存器 R 取反, 结果放入 ACC
周期: 1
影响标志位: Z
举例:

LDIA 0AH ;ACC 赋值 0AH
LD R01,A ;将 ACC 的值(0AH)赋给寄存器 R01
COMA R01 ;执行结果: ACC=0F5H

COMR [R]

操作: 寄存器 R 取反, 结果放入 R
 周期: 1
 影响标志位: Z
 举例:

```
LDIA    0AH           ;ACC 赋值 0AH
LD      R01,A        ;将 ACC 的值(0AH)赋给寄存器 R01
COMR   R01           ;执行结果: R01=0F5H
```

DECA [R]

操作: 寄存器 R 自减 1, 结果放入 ACC
 周期: 1
 影响标志位: Z
 举例:

```
LDIA    0AH           ;ACC 赋值 0AH
LD      R01,A        ;将 ACC 的值(0AH)赋给寄存器 R01
DECA   R01           ;执行结果: ACC=(0AH-1)=09H
```

DECR [R]

操作: 寄存器 R 自减 1, 结果放入 R
 周期: 1
 影响标志位: Z
 举例:

```
LDIA    0AH           ;ACC 赋值 0AH
LD      R01,A        ;将 ACC 的值(0AH)赋给寄存器 R01
DECR   R01           ;执行结果: R01=(0AH-1)=09H
```

HSUBA [R]

操作: ACC 减 R, 结果放入 ACC
 周期: 1
 影响标志位: C,DC,Z,OV
 举例:

```
LDIA    077H          ;ACC 赋值 077H
LD      R01,A        ;将 ACC 的值(077H)赋给寄存器 R01
LDIA    080H          ;ACC 赋值 080H
HSUBA   R01          ;执行结果: ACC=(80H-77H)=09H
```

HSUBR [R]

操作: ACC 减 R, 结果放入 R
 周期: 1
 影响标志位: C,DC,Z,OV
 举例:

```
LDIA    077H    ;ACC 赋值 077H
LD      R01,A   ;将 ACC 的值(077H)赋给寄存器 R01
LDIA    080H    ;ACC 赋值 080H
HSUBR   R01     ;执行结果: R01=(80H-77H)=09H
```

HSUBCA [R]

操作: ACC 减 R 减 C, 结果放入 ACC
 周期: 1
 影响标志位: C,DC,Z,OV
 举例:

```
LDIA    077H    ;ACC 赋值 077H
LD      R01,A   ;将 ACC 的值(077H)赋给寄存器 R01
LDIA    080H    ;ACC 赋值 080H
HSUBCA  R01     ;执行结果: ACC=(80H-77H-C)=09H(C=0)
                          ACC=(80H-77H-C)=08H(C=1)
```

HSUBCR [R]

操作: ACC 减 R 减 C, 结果放入 R
 周期: 1
 影响标志位: C,DC,Z,OV
 举例:

```
LDIA    077H    ;ACC 赋值 077H
LD      R01,A   ;将 ACC 的值(077H)赋给寄存器 R01
LDIA    080H    ;ACC 赋值 080H
HSUBCR  R01     ;执行结果: R01=(80H-77H-C)=09H(C=0)
                          R01=(80H-77H-C)=08H(C=1)
```

INCA [R]

操作: 寄存器 R 自加 1, 结果放入 ACC
 周期: 1
 影响标志位: Z
 举例:

```
LDIA    0AH     ;ACC 赋值 0AH
LD      R01,A   ;将 ACC 的值(0AH)赋给寄存器 R01
INCA    R01     ;执行结果: ACC=(0AH+1)=0BH
```

INCR
[R]

操作: 寄存器 R 自加 1, 结果放入 R
 周期: 1
 影响标志位: Z
 举例:

```
LDIA    0AH           ;ACC 赋值 0AH
LD      R01,A        ;将 ACC 的值(0AH)赋给寄存器 R01
INCR   R01           ;执行结果: R01=(0AH+1)=0BH
```

JP
add

操作: 跳转到 add 地址
 周期: 2
 影响标志位: 无
 举例:

```
JP      LOOP         ;跳转至名称定义为"LOOP"的子程序地址
```

LD
A,[R]

操作: 将 R 的值赋给 ACC
 周期: 1
 影响标志位: Z
 举例:

```
LD      A,R01        ;将寄存器 R0 的值赋给 ACC
LD      R02,A        ;将 ACC 的值赋给寄存器 R02, 实现了数据从 R01→R02 的移动
```

LD
[R],A

操作: 将 ACC 的值赋给 R
 周期: 1
 影响标志位: 无
 举例:

```
LDIA    09H           ;给 ACC 赋值 09H
LD      R01,A        ;执行结果: R01=09H
```

LDIA
i

操作: 立即数 i 赋给 ACC
 周期: 1
 影响标志位: 无
 举例:

```
LDIA    0AH           ;ACC 赋值 0AH
```


NOP

操作: 空指令
周期: 1
影响标志位: 无
举例:

NOP
NOP

ORIA **i**

操作: 立即数与 ACC 进行逻辑或操作, 结果赋给 ACC
周期: 1
影响标志位: Z
举例:

LDIA 0AH ;ACC 赋值 0AH
ORIA 030H ;执行结果: ACC =(0AH or 30H)=3AH

ORA **[R]**

操作: 寄存器 R 跟 ACC 进行逻辑或运算, 结果放入 ACC
周期: 1
影响标志位: Z
举例:

LDIA 0AH ;给 ACC 赋值 0AH
LD R01,A ;将 ACC(0AH)赋给寄存器 R01
LDIA 30H ;给 ACC 赋值 30H
ORA R01 ;执行结果: ACC=(0AH or 30H)=3AH

ORR **[R]**

操作: 寄存器 R 跟 ACC 进行逻辑或运算, 结果放入 R
周期: 1
影响标志位: Z
举例:

LDIA 0AH ;给 ACC 赋值 0AH
LD R01,A ;将 ACC(0AH)赋给寄存器 R01
LDIA 30H ;给 ACC 赋值 30H
ORR R01 ;执行结果: R01=(0AH or 30H)=3AH

RET

操作: 从子程序返回
 周期: 2
 影响标志位: 无
 举例:

```
CALL    LOOP    ;调用子程序 LOOP
NOP     ;RET 指令返回后将执行这条语句
...     ;其它程序
```

LOOP:

```
...     ;子程序
RET     ;子程序返回
```

RET
i

操作: 从子程序带参数返回, 参数放入 ACC
 周期: 2
 影响标志位: 无
 举例:

```
CALL    LOOP    ;调用子程序 LOOP
NOP     ;RET 指令返回后将执行这条语句
...     ;其它程序
```

LOOP:

```
...     ;子程序
RET     35H     ;子程序返回,ACC=35H
```

RETI

操作: 中断返回
 周期: 2
 影响标志位: 无
 举例:

```
INT_START    ;中断程序入口
...          ;中断处理程序
RETI         ;中断返回
```

RLCA
[R]

操作: 寄存器 R 带 C 循环左移一位, 结果放入 ACC
 周期: 1
 影响标志位: C
 举例:

```
LDIA    03H    ;ACC 赋值 03H
LD      R01,A  ;ACC 值赋给 R01,R01=03H
RLCA    R01    ;操作结果: ACC=06H(C=0);
                    ACC=07H(C=1)
                    C=0
```

RLCR
[R]

操作: 寄存器 R 带 C 循环左移一位, 结果放入 R

周期: 1

影响标志位: C

举例:

```
LDIA    03H           ;ACC 赋值 03H
LD      R01,A        ;ACC 值赋给 R01,R01=03H
RLCR    R01          ;操作结果: R01=06H(C=0);
                          R01=07H(C=1);
                          C=0
```

RLA
[R]

操作: 寄存器 R 不带 C 循环左移一位, 结果放入 ACC

周期: 1

影响标志位: 无

举例:

```
LDIA    03H           ;ACC 赋值 03H
LD      R01,A        ;ACC 值赋给 R01,R01=03H
RLA     R01          ;操作结果: ACC=06H
```

RLR
[R]

操作: 寄存器 R 不带 C 循环左移一位, 结果放入 R

周期: 1

影响标志位: 无

举例:

```
LDIA    03H           ;ACC 赋值 03H
LD      R01,A        ;ACC 值赋给 R01,R01=03H
RLR     R01          ;操作结果: R01=06H
```

RRCA
[R]

操作: 寄存器 R 带 C 循环右移一位, 结果放入 ACC

周期: 1

影响标志位: C

举例:

```
LDIA    03H           ;ACC 赋值 03H
LD      R01,A        ;ACC 值赋给 R01,R01=03H
RRCA    R01          ;操作结果: ACC=01H(C=0);
                          ACC=081H(C=1);
                          C=1
```

RRCR [R]

操作: 寄存器 R 带 C 循环右移一位, 结果放入 R

周期: 1

影响标志位: C

举例:

```
LDIA    03H           ;ACC 赋值 03H
LD      R01,A        ;ACC 值赋给 R01,R01=03H
RRCR   R01           ;操作结果: R01=01H(C=0);
                          R01=81H(C=1);
                          C=1
```

RRA [R]

操作: 寄存器 R 不带 C 循环右移一位, 结果放入 ACC

周期: 1

影响标志位: 无

举例:

```
LDIA    03H           ;ACC 赋值 03H
LD      R01,A        ;ACC 值赋给 R01,R01=03H
RRA     R01           ;操作结果: ACC=81H
```

RRR [R]

操作: 寄存器 R 不带 C 循环右移一位, 结果放入 R

周期: 1

影响标志位: 无

举例:

```
LDIA    03H           ;ACC 赋值 03H
LD      R01,A        ;ACC 值赋给 R01,R01=03H
RRR     R01           ;操作结果: R01=81H
```

SET [R]

操作: 寄存器 R 所有位置 1

周期: 1

影响标志位: 无

举例:

```
SET     R01           ;操作结果: R01=0FFH
```

SETB [R],b

操作: 寄存器 R 的第 b 位置 1

周期: 1

影响标志位: 无

举例:

```
CLR     R01           ;R01=0
SETB    R01,3        ;操作结果: R01=08H
```

STOP

操作: 进入休眠状态
周期: 1
影响标志位: TO, PD
举例:

STOP ;芯片进入省电模式, CPU、振荡器停止工作, IO 口保持原来状态

SUBIA i

操作: 立即数 i 减 ACC, 结果放入 ACC
周期: 1
影响标志位: C,DC,Z,OV
举例:

LDIA 077H ;ACC 赋值 77H
SUBIA 80H ;操作结果: ACC=80H-77H=09H

SUBA [R]

操作: 寄存器 R 减 ACC, 结果放入 ACC
周期: 1
影响标志位: C,DC,Z,OV
举例:

LDIA 080H ;ACC 赋值 80H
LD R01,A ;ACC 的值赋给 R01, R01=80H
LDIA 77H ;ACC 赋值 77H
SUBA R01 ;操作结果: ACC=80H-77H=09H

SUBR [R]

操作: 寄存器 R 减 ACC, 结果放入 R
周期: 1
影响标志位: C,DC,Z,OV
举例:

LDIA 080H ;ACC 赋值 80H
LD R01,A ;ACC 的值赋给 R01, R01=80H
LDIA 77H ;ACC 赋值 77H
SUBR R01 ;操作结果: R01=80H-77H=09H

SUBCA [R]

操作: 寄存器 R 减 ACC 减 C, 结果放入 ACC

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
LDIA    080H           ;ACC 赋值 80H
LD      R01,A         ;ACC 的值赋给 R01, R01=80H
LDIA    77H           ;ACC 赋值 77H
SUBCA   R01           ;操作结果: ACC=80H-77H-C=09H(C=0);
                          ACC=80H-77H-C=08H(C=1);
```

SUBCR [R]

操作: 寄存器 R 减 ACC 减 C, 结果放入 R

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
LDIA    080H           ;ACC 赋值 80H
LD      R01,A         ;ACC 的值赋给 R01, R01=80H
LDIA    77H           ;ACC 赋值 77H
SUBCR   R01           ;操作结果: R01=80H-77H-C=09H(C=0)
                          R01=80H-77H-C=08H(C=1)
```

SWAPA [R]

操作: 寄存器 R 高低半字节交换, 结果放入 ACC

周期: 1

影响标志位: 无

举例:

```
LDIA    035H           ;ACC 赋值 35H
LD      R01,A         ;ACC 的值赋给 R01, R01=35H
SWAPA   R01           ;操作结果: ACC=53H
```

SWAPR [R]

操作: 寄存器 R 高低半字节交换, 结果放入 R

周期: 1

影响标志位: 无

举例:

```
LDIA    035H           ;ACC 赋值 35H
LD      R01,A         ;ACC 的值赋给 R01, R01=35H
SWAPR   R01           ;操作结果: R01=53H
```

SZB [R],b

操作: 判断寄存器 R 的第 b 位, 为 0 间跳, 否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```

SZB      R01,3      ;判断寄存器 R01 的第 3 位
JP       LOOP      ;R01 的第 3 位为 1 才执行这条语句, 跳转至 LOOP
JP       LOOP1     ;R01 的第 3 位为 0 时间跳, 执行这条语句, 跳转至 LOOP1
    
```

SNZB [R],b

操作: 判断寄存器 R 的第 b 位, 为 1 间跳, 否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```

SNZB     R01,3     ;判断寄存器 R01 的第 3 位
JP       LOOP     ;R01 的第 3 位为 0 才执行这条语句, 跳转至 LOOP
JP       LOOP1    ;R01 的第 3 位为 1 时间跳, 执行这条语句, 跳转至 LOOP1
    
```

SZA [R]

操作: 将寄存器 R 的值赋给 ACC, 若 R 为 0 则间跳, 否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```

SZA      R01      ;R01→ACC
JP       LOOP     ;R01 不为 0 时执行这条语句, 跳转至 LOOP
JP       LOOP1    ;R01 为 0 时间跳, 执行这条语句, 跳转至 LOOP1
    
```

SZR [R]

操作: 将寄存器 R 的值赋给 R, 若 R 为 0 则间跳, 否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```

SZR      R01      ;R01→R01
JP       LOOP     ;R01 不为 0 时执行这条语句, 跳转至 LOOP
JP       LOOP1    ;R01 为 0 时间跳执行这条语句, 跳转至 LOOP1
    
```

SZINCA [R]

操作: 将寄存器 R 自加 1, 结果放入 ACC, 若结果为 0, 则跳过下一条语句, 否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```
SZINCA    R01           ;R01+1→ACC
JP        LOOP         ;ACC 不为 0 时执行这条语句, 跳转至 LOOP
JP        LOOP1        ;ACC 为 0 时执行这条语句, 跳转至 LOOP1
```

SZINCR [R]

操作: 将寄存器 R 自加 1, 结果放入 R, 若结果为 0, 则跳过下一条语句, 否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```
SZINCR    R01           ;R01+1→R01
JP        LOOP         ; R01 不为 0 时执行这条语句, 跳转至 LOOP
JP        LOOP1        ; R01 为 0 时执行这条语句, 跳转至 LOOP1
```

SZDECA [R]

操作: 将寄存器 R 自减 1, 结果放入 ACC, 若结果为 0, 则跳过下一条语句, 否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```
SZDECA    R01           ;R01-1→ACC
JP        LOOP         ;ACC 不为 0 时执行这条语句, 跳转至 LOOP
JP        LOOP1        ;ACC 为 0 时执行这条语句, 跳转至 LOOP1
```

SZDECR [R]

操作: 将寄存器 R 自减 1, 结果放入 R, 若结果为 0, 则跳过下一条语句, 否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```
SZDECR    R01           ;R01-1→R01
JP        LOOP         ; R01 不为 0 时执行这条语句, 跳转至 LOOP
JP        LOOP1        ; R01 为 0 时执行这条语句, 跳转至 LOOP1
```


TABLE [R]

操作: 查表, 查表结果低 8 位放入 R, 高位放入专用寄存器 TABLE_SPH

周期: 2

影响标志位: 无

举例:

```
LDIA    01H           ;ACC 赋值 01H
LD      TABLE_SPH,A ;ACC 值赋给表格高位地址, TABLE_SPH=1
LDIA    015H         ;ACC 赋值 15H
LD      TABLE_SPL,A ;ACC 值赋给表格地位地址, TABLE_SPL=15H
TABLE   R01          ;查表 0115H 地址, 操作结果: TABLE_DATAH=12H, R01=34H
...
ORG     0115H
DW      1234H
```

TABLEA

操作: 查表, 查表结果低 8 位放入 ACC, 高位放入专用寄存器 TABLE_SPH

周期: 2

影响标志位: 无

举例:

```
LDIA    01H           ;ACC 赋值 01H
LD      TABLE_SPH,A ;ACC 值赋给表格高位地址, TABLE_SPH=1
LDIA    015H         ;ACC 赋值 15H
LD      TABLE_SPL,A ;ACC 值赋给表格地位地址, TABLE_SPL=15H
TABLEA          ;查表 0115H 地址, 操作结果: TABLE_DATAH=12H, ACC=34H
...
ORG     0115H
DW      1234H
```

TESTZ [R]

操作: 将 R 的值赋给 R,用以影响 Z 标志位

周期: 1

影响标志位: Z

举例:

```
TESTZ   R0           ;将寄存器 R0 的值赋给 R0, 用于影响 Z 标志位
SZB     STATUS,Z    ;判断 Z 标志位, 为 0 间跳
JP      Add1        ;当寄存器 R0 为 0 的时候跳转至地址 Add1
JP      Add2        ;当寄存器 R0 不为 0 的时候跳转至地址 Add2
```

XORIA **i**

操作: 立即数与 ACC 进行逻辑异或运算, 结果放入 ACC

周期: 1

影响标志位: Z

举例:

```
LDIA            0AH                    ;ACC 赋值 0AH
XORIA           0FH                    ;执行结果: ACC=05H
```

XORA **[R]**

操作: 寄存器 R 与 ACC 进行逻辑异或运算, 结果放入 ACC

周期: 1

影响标志位: Z

举例:

```
LDIA            0AH                    ;ACC 赋值 0AH
LD              R01,A                  ;ACC 值赋给 R01,R01=0AH
LDIA            0FH                    ;ACC 赋值 0FH
XORA            R01                    ;执行结果: ACC=05H
```

XORR **[R]**

操作: 寄存器 R 与 ACC 进行逻辑异或运算, 结果放入 R

周期: 1

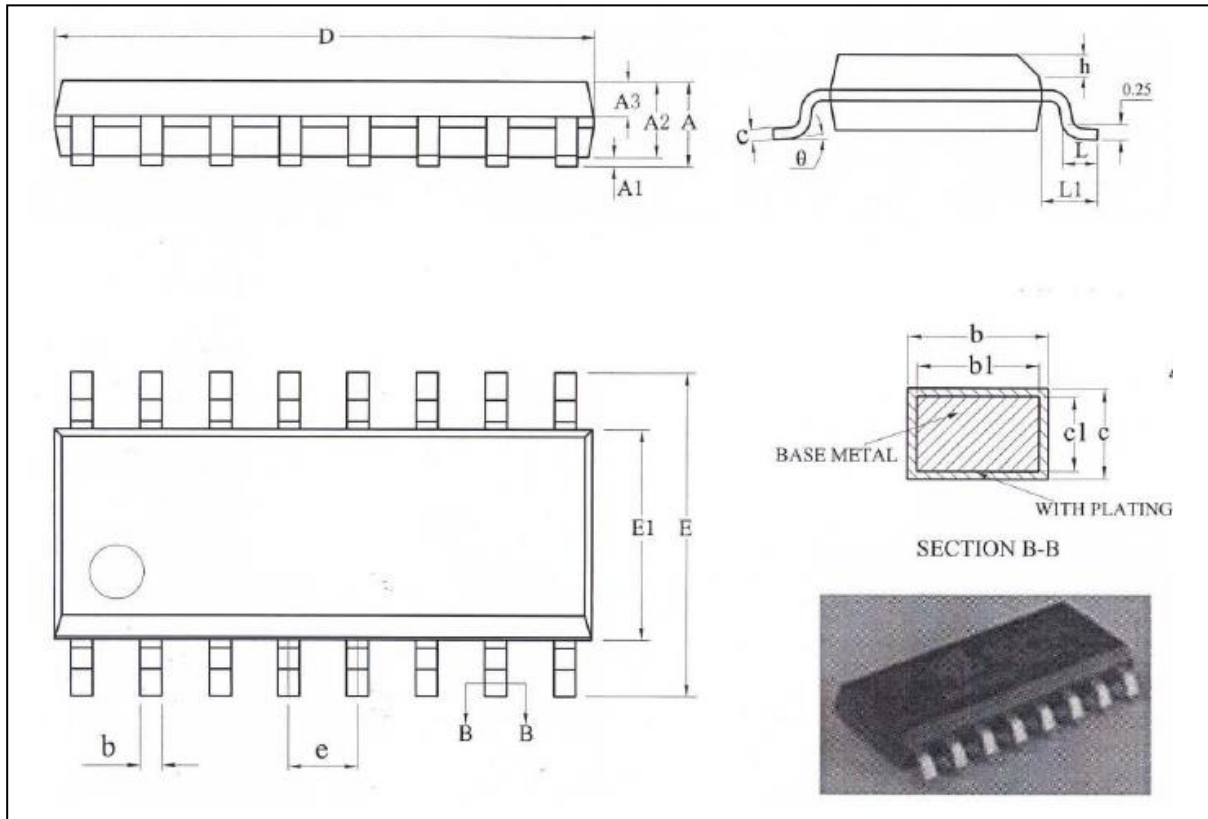
影响标志位: Z

举例:

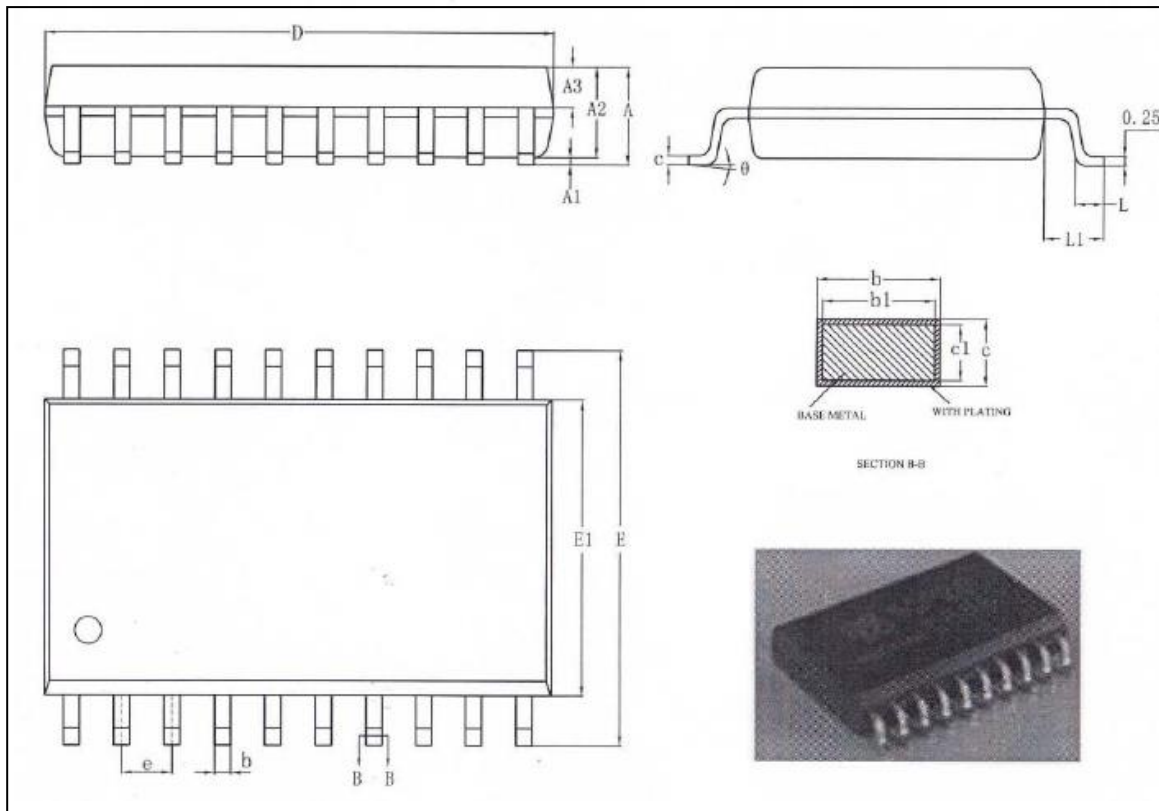
```
LDIA            0AH                    ;ACC 赋值 0AH
LD              R01,A                  ;ACC 值赋给 R01,R01=0AH
LDIA            0FH                    ;ACC 赋值 0FH
XORR            R01                    ;执行结果: R01=05H
```

17. 封装

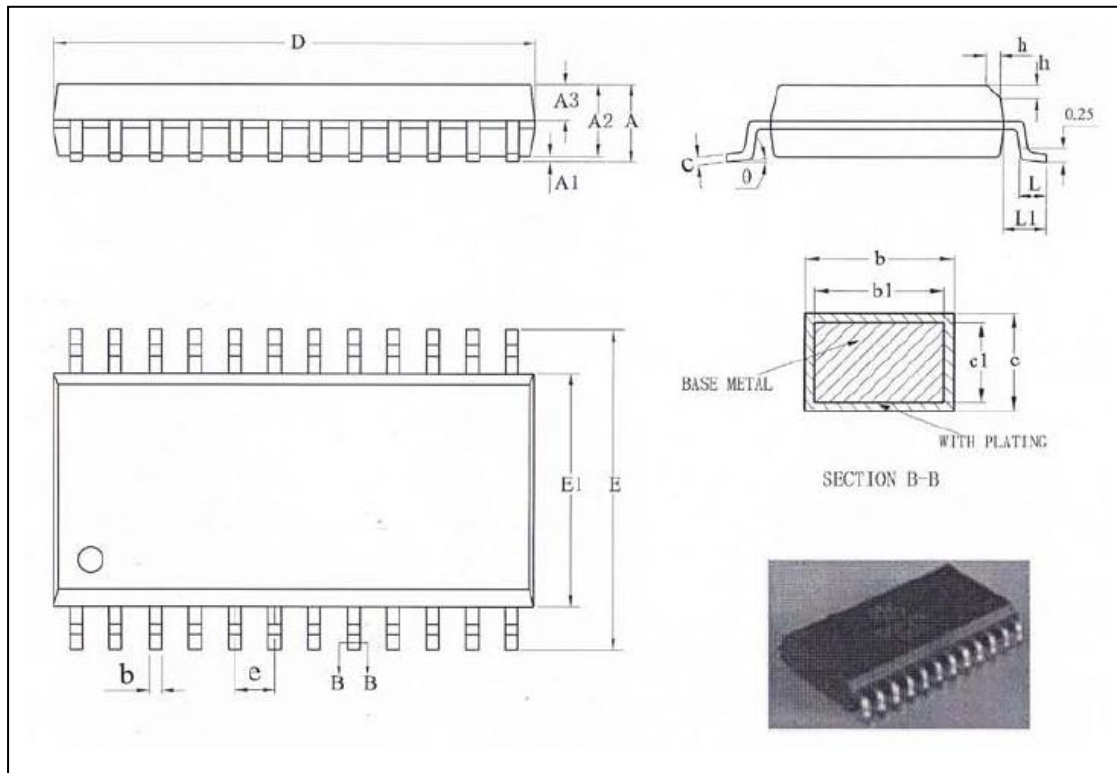
17.1 SOP 16



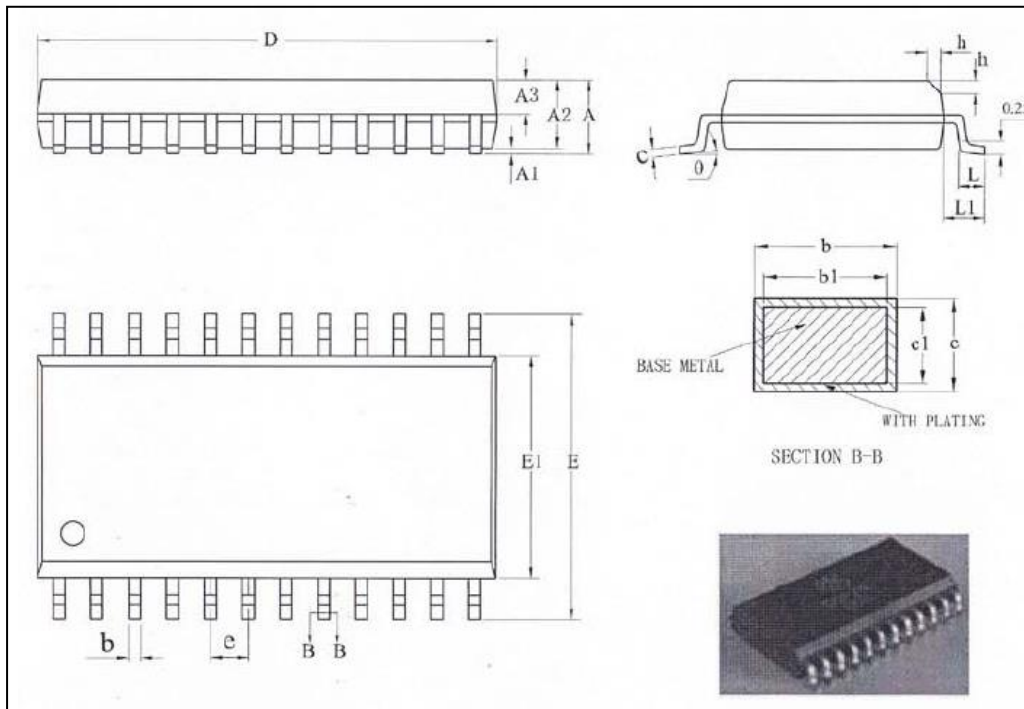
Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.75
A1	1.10	-	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.20	-	0.24
c1	0.19	0.20	0.21
D	9.80	9.90	10.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
h	0.25	-	0.50
L	0.5	-	0.80
L1	1.05REF		
θ	0	-	8°

17.2 SOP 20


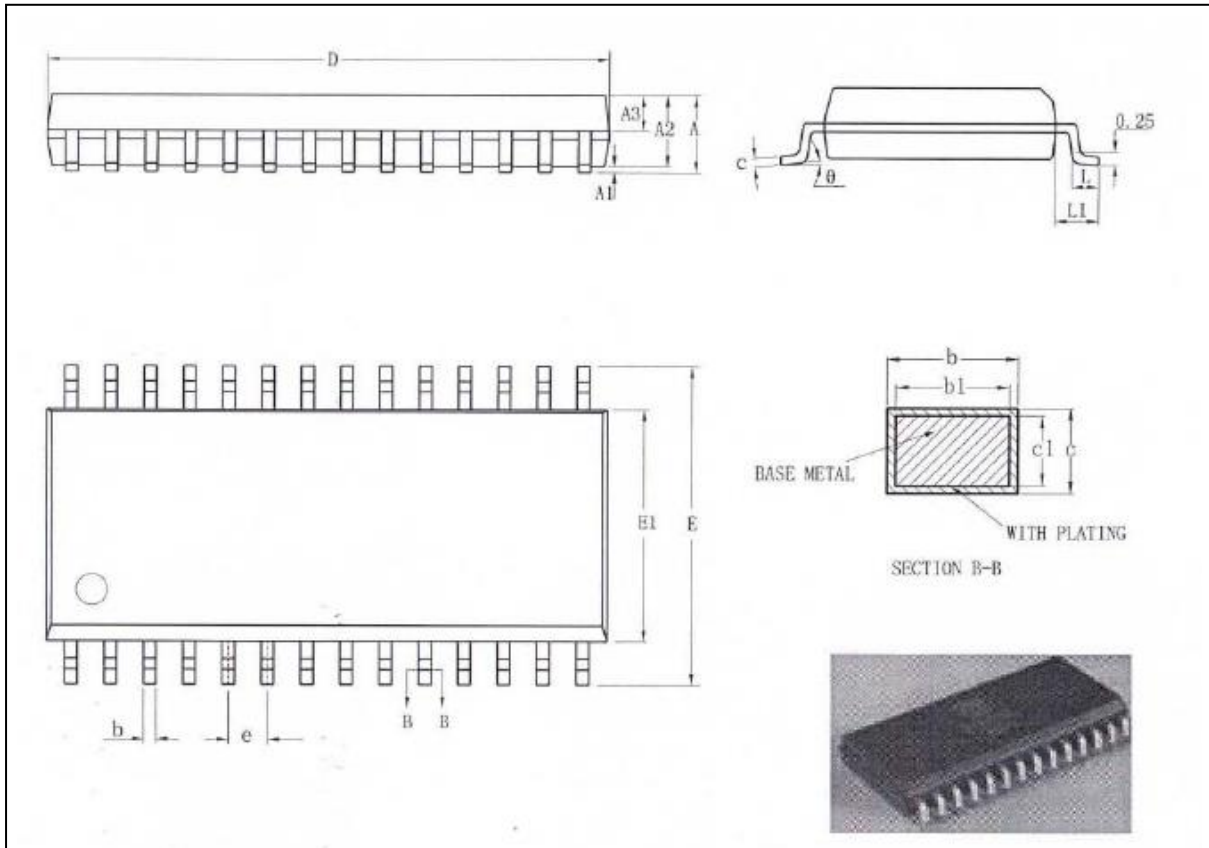
Symbol	Millimeter		
	Min	Nom	Max
A	-	-	2.65
A1	1.10	-	0.30
A2	2.25	2.30	2.35
A3	0.97	1.02	1.07
b	0.35	-	0.43
b1	0.34	0.37	0.40
c	0.25	-	0.29
c1	0.24	0.25	0.26
D	12.70	12.80	12.90
E	10.10	10.30	10.50
E1	7.40	7.50	7.60
e	1.27BSC		
L	0.70	-	1.00
L1	1.40REF		
θ	0	-	8°

17.3 SOP24


Symbol	Millimeter		
	Min	Nom	Max
A	2.36	2.54	2.64
A1	0.10	0.20	0.30
A2	2.26	2.30	2.35
A3	0.97	1.02	1.07
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.25	-	0.29
c1	0.24	0.25	0.26
D	15.30	15.40	15.50
E	10.10	10.30	10.50
E1	7.40	7.50	7.60
e	1.27BSC		
L	0.70	-	1.00
L1	1.40REF		
h	0.25	-	0.75
θ	0	-	8°

17.4 SKY24


Symbol	Millimeter		
	Min	Nom	Max
A	2.36	2.54	2.64
A1	0.10	0.20	0.30
A2	2.26	2.30	2.35
A3	0.97	1.02	1.07
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.25	-	0.29
c1	0.24	0.25	0.26
D	15.30	15.40	15.50
E	10.10	10.30	10.50
E1	7.40	7.50	7.60
e	1.27BSC		
L	0.70	-	1.00
L1	1.40REF		
h	0.25	-	0.75
θ	0	-	8°

17.5 SOP28


Symbol	Millimeter		
	Min	Nom	Max
A	-	-	2.64
A1	0.10	-	0.30
A2	2.25	2.30	2.35
A3	0.97	1.02	1.07
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.25	-	0.29
c1	0.24	0.25	0.26
D	17.90	18.00	18.10
E	10.10	10.30	10.50
E1	7.40	7.50	7.60
e	1.27BSC		
L	0.70	-	1.00
L1	1.40REF		
θ	0	-	8°

18. 版本修订说明

版本号	时间	修改内容																		
V1.0	2016 年 7 月 20	整个文档里的 CCP2IN2 改为 CCP2IN0 (P29、P98)																		
V1.1	2016 年 7 月 22	<p>预分频器应用增加 2 行指令，确保不会造成复位</p> <table border="1" style="margin-left: 40px;"> <tr> <td>CLR</td> <td>TMR0</td> <td>;TMR0 清零</td> </tr> <tr> <td>CLRWDT</td> <td></td> <td>;WDT 清零</td> </tr> <tr> <td>LDIA</td> <td>B'00xx1111'</td> <td>;</td> </tr> <tr> <td>LD</td> <td>OPTION_REG,A</td> <td></td> </tr> <tr> <td>LDIA</td> <td>B'00xx1xxx'</td> <td>;设置新的预分频器</td> </tr> <tr> <td>LD</td> <td>OPTION_REG,A</td> <td></td> </tr> </table>	CLR	TMR0	;TMR0 清零	CLRWDT		;WDT 清零	LDIA	B'00xx1111'	;	LD	OPTION_REG,A		LDIA	B'00xx1xxx'	;设置新的预分频器	LD	OPTION_REG,A	
CLR	TMR0	;TMR0 清零																		
CLRWDT		;WDT 清零																		
LDIA	B'00xx1111'	;																		
LD	OPTION_REG,A																			
LDIA	B'00xx1xxx'	;设置新的预分频器																		
LD	OPTION_REG,A																			
V1.2	2016 年 8 月 6	<ol style="list-style-type: none"> 1、修改 RAM 352 344; 2、修改 EEPROM 64*8Bit==》32*16bit; 3、修改第 12 章，删除“闪存程序存储器控制”相关说明; 4、禁止使用寄存器地址：118H—11FH; 																		
V1.3	2016 年 8 月 6	新增加管脚型号：CMS89FT623 说明与管脚图																		
V1.4	2017 年 1 月 6	新增加管脚型号：CMS89FT6231 说明与管脚图																		
V1.5	2017 年 10 月 30	<ol style="list-style-type: none"> 1、章节 2.6 中，修改预分频器指令范例。 2、修改部分有误内容，修改内容详情请查阅“CMS89FT6xx 资料勘误表” 																		
V1.6	2018 年 8 月 13	增加了第 12 章 LCD、LED 功能模块说明																		
V1.7	2019 年 9 月	删除 DIP 封装形式																		