



CMS89F52x 用户手册

增强型闪存 8 位 CMOS 单片机

Rev. 1.7.0

请注意以下有关CMS知识产权政策

* 中微半导体（深圳）股份有限公司（以下简称本公司）已申请了专利，享有绝对的合法权益。与本公司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本公司专利权的公司、组织或个人，本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，并追讨本公司因侵权行为所受的损失、或侵权者所得的不法利益。

* 中微半导体（深圳）股份有限公司的名称和标识都是本公司的注册商标。

* 本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。然而本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，本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。本公司的产品不授权适用于救生、维生器件或系统中作为关键器件。本公司拥有不事先通知而修改产品的权利，对于最新的信息，请参考官方网站 www.mcu.com.cn

目录

1. 产品概述	7
1.1 功能特性	7
1.2 系统结构框图	8
1.3 管脚分布	9
1.3.1 CMS89F5233 引脚图	9
1.3.2 CMS89F526 引脚图	10
1.4 系统配置寄存器	11
1.5 在线串行编程	12
2. 中央处理器 (CPU)	13
2.1 内存	13
2.1.1 程序内存	13
2.1.2 数据存储器	18
2.2 寻址方式	23
2.2.1 直接寻址	23
2.2.2 立即寻址	23
2.2.3 间接寻址	23
2.3 堆栈	24
2.4 工作寄存器 (ACC)	25
2.4.1 概述	25
2.4.2 ACC 应用	25
2.5 程序状态寄存器 (STATUS)	26
2.6 预分频器 (OPTION_REG)	27
2.7 程序计数器 (PC)	29
2.8 看门狗计数器 (WDT)	30
2.8.1 WDT 周期	30
2.8.2 看门狗定时器控制寄存器 WDTCON	30
3. 系统时钟	31
3.1 概述	31
3.2 系统振荡器	32
3.2.1 内部 RC 振荡	32
3.3 起振时间	32
3.4 振荡器控制寄存器	33
4. 复位	34
4.1 上电复位	34
4.2 掉电复位	35
4.2.1 掉电复位概述	35
4.2.2 掉电复位的改进办法	36
4.3 看门狗复位	36

5. 休眠模式	37
5.1 进入休眠模式	37
5.2 从休眠状态唤醒	37
5.3 使用中断唤醒	37
5.4 休眠模式应用举例	38
5.5 休眠模式唤醒时间	38
6. I/O 端口	39
6.1 I/O 口结构图	40
6.2 PORTA	41
6.2.1 PORTA 数据及方向控制	41
6.2.2 PORTA 模拟选择控制	42
6.2.3 PORTA 上拉电阻	42
6.3 PORTB	43
6.3.1 PORTB 数据及方向	43
6.3.2 PORTB 模拟选择	44
6.3.3 PORTB 上拉电阻	44
6.4 PORTE	45
6.4.1 PORTE 数据及方向	45
6.4.2 PORTE 上拉电阻	45
6.4.3 PORTE 模拟选择	46
6.5 I/O 使用	47
6.5.1 写 I/O 口	47
6.5.2 读 I/O 口	47
6.6 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	53
7.3 中断现场的保护方法	55
7.4 中断的优先级, 及多中断嵌套	55
8. 定时计数器 TIMER0	56
8.1 定时计数器 TIMER0 概述	56
8.2 TIMER0 的工作原理	57
8.2.1 8 位定时器模式	57
8.2.2 8 位计数器模式	57
8.2.3 软件可编程预分频器	57
8.2.4 在 TIMER0 和 WDT 模块间切换预分频器	57

8.2.5	TIMER0 中断.....	58
8.3	与 TIMER0 相关寄存器.....	59
9.	定时计数器 TIMER1.....	60
9.1	TIMER1 概述.....	60
9.2	TIMER1 的工作原理.....	61
9.3	TIMER1 预分频器.....	61
9.4	TIMER1 中断.....	61
9.5	TIMER1 相关寄存器.....	62
10.	定时计数器 TIMER2.....	63
10.1	TIMER2 概述.....	63
10.2	TIMER2 的工作原理.....	64
10.3	TIMER2 相关的寄存器.....	65
11.	模数转换 (ADC)	66
11.1	ADC 概述.....	66
11.2	ADC 配置.....	67
11.2.1	端口配置.....	67
11.2.2	通道选择.....	67
11.2.3	ADC 参考电压.....	67
11.2.4	转换时钟.....	67
11.2.5	ADC 中断.....	68
11.3	ADC 工作原理.....	69
11.3.1	启动转换.....	69
11.3.2	完成转换.....	69
11.3.3	终止转换.....	69
11.3.4	ADC 在休眠模式下的工作原理.....	69
11.3.5	A/D 转换步骤.....	70
11.4	ADC 相关 RAM.....	71
12.	PWM 模块.....	73
12.1	PWM 特性.....	73
12.2	PWM 相关寄存器.....	73
13.	捕捉模块 CCP	76
13.1	捕捉 CCP 寄存器.....	76
13.2	捕捉模式.....	77
13.2.1	CCP 引脚配置.....	77
13.2.2	TIMER1 模式选择.....	77
13.2.3	软件中断.....	77
14.	主控同步串行端口 (MSSP) 模块	78
14.1	主控 SSP (MSSP) 模块概述.....	78
14.2	SPI 模式.....	78

14.2.1	SPI 相关寄存器	79
14.2.2	SPI 工作原理	81
14.2.3	使能 SPI I/O	82
14.2.4	主控模式	82
14.2.5	从动模式	83
14.2.6	从动选择同步	84
14.2.7	休眠操作	86
14.2.8	复位的影响	86
14.3	I ² C 模块	87
14.3.1	相关寄存器说明	88
14.3.2	主控模式	91
14.3.3	I ² C 主控模式支持	91
14.3.4	波特率发生器	93
14.3.5	I ² C 主控模式发送	94
14.3.6	I ² C 主控模式接收	95
14.3.7	I ² C 主控模式启动条件时序	97
14.3.8	I ² C 主控模式重复启动条件时序	98
14.3.9	应答序列时序	99
14.3.10	停止条件序列	100
14.3.11	时钟仲裁	101
14.3.12	多主机模式	101
14.3.13	多主机通信、总线冲突与总线仲裁	102
14.3.14	从动模式	102
14.3.15	SSP 屏蔽寄存器	105
14.3.16	休眠模式下的操作	105
14.3.17	复位的影响	105
15.	可编程脉冲发生器 PPG	106
15.1	PPG 工作原理	106
15.2	与 PPG 相关的引脚	107
15.3	PPG 工作模式	108
15.3.1	单次输出模式	109
15.3.2	同步输出模式	109
15.4	比较器	110
15.4.1	同步比较器 COMP1	110
15.4.2	过压比较器 COMP2 跟浪涌比较器 COMP4/COMP5	112
15.4.3	过压比较器 1- COMP3	115
15.4.4	比较器调零	117
15.4.5	比较器与 PPG 内部结构图	119
16.	数据 EEPROM 控制	120
16.1	数据 EEPROM 概述	120

16.2	相关寄存器	121
16.2.1	EEADR 寄存器	121
16.2.2	EECON1 和 EECON2 寄存器	121
16.3	读数据 EEPROM 存储器	123
16.4	写数据 EEPROM 存储器	124
16.5	EEPROM 操作注意事项	125
16.5.1	写校验	125
16.5.2	避免误写的保护	125
17.	运算放大器(OPA).....	126
17.1	运算放大器简介	126
17.2	跟运放相关寄存器	127
18.	电气参数	129
18.1	直流电气特性	129
18.2	OPA 电气特性	130
18.3	COMP 电气特性	130
18.4	交流电气特性	131
18.5	内部 RC 振荡特性	132
18.5.1	内部 RC 振荡电压特性	132
18.5.2	内部 RC 振荡温度特性	132
19.	指令	133
19.1	指令一览表	133
19.2	指令说明	135
20.	封装	151
20.1	DIP16	151
20.2	SOP16	152
20.3	DIP20	153
20.4	SOP20	154
21.	版本修订说明	155

1. 产品概述

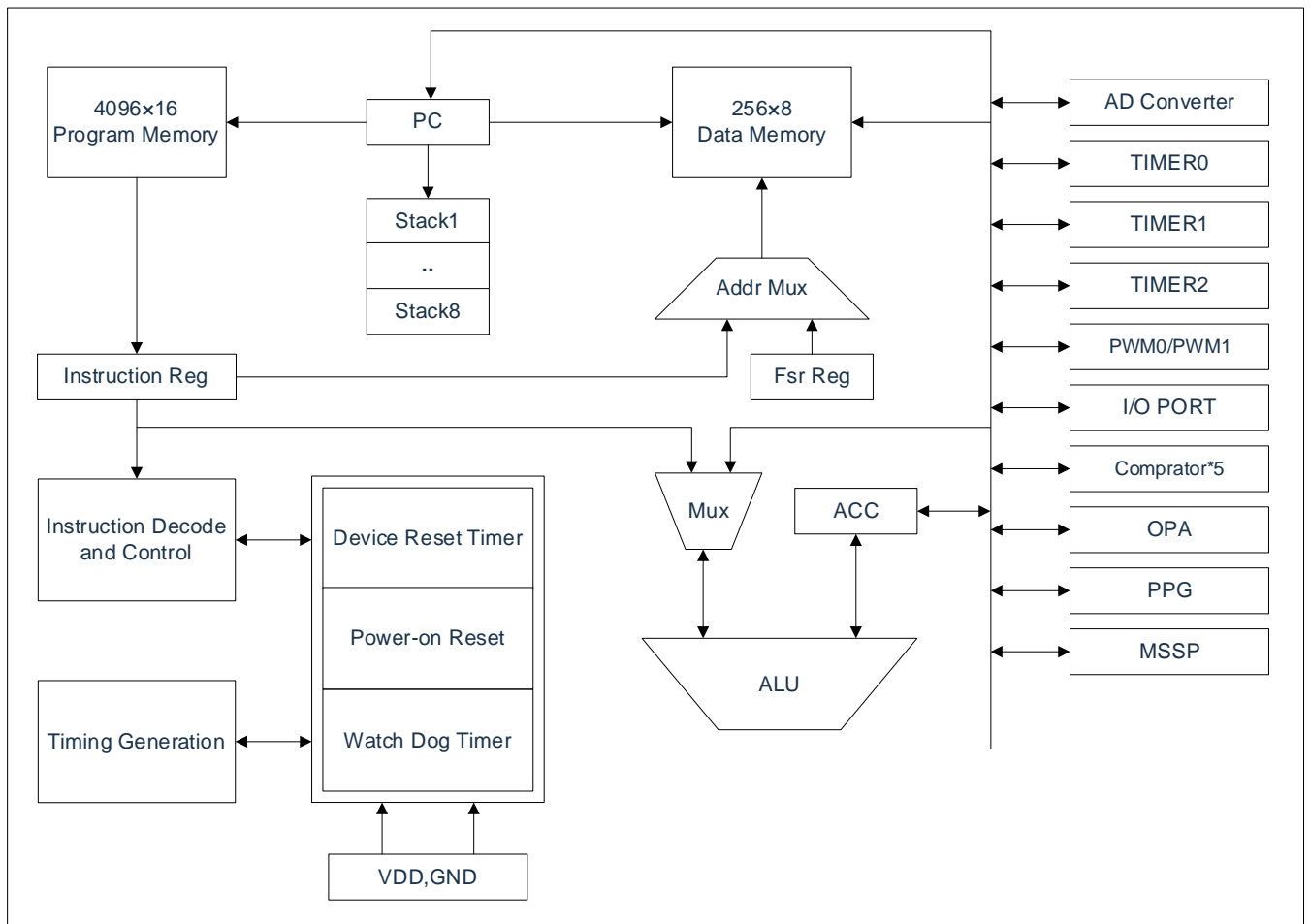
1.1 功能特性

- ◆ 内存
 - FLASH: 4K×16Bit
 - 通用 RAM: 256×8Bit
 - EEPROM: 32×16Bit
- ◆ 8 级堆栈缓存器
- ◆ 简洁实用的指令系统（68 条指令）
- ◆ 内置低压侦测电路
- ◆ 中断源：
 - 3 个定时中断
 - 外部中断
 - 其它外设中断
- ◆ 定时器：
 - 8 位定时器 TIMER0, TIMER2
 - 16 位定时器 TIMER1
- ◆ 2 路 8Bit PWM, 周期和占空比均可调节
- ◆ 查表功能
- ◆ 工作电压范围：3.5V~5.5V @ 8MHz
3.5V~5.5V @ 4MHz
- ◆ 工作温度范围：-40°C~85°C
- ◆ 振荡方式
 - 内部 RC 振荡：设计频率 8MHz/16MHz
- ◆ 指令周期（单指令或双指令）
- ◆ 内置 WDT 定时器
- ◆ 高精度 10 位 ADC
- ◆ MSSP 通信模块（SPI/I²C）
- ◆ PPG 控制模块：
 - 1 个三端接端口的运放 OPA，正/负端可选择内部接地
 - 5 个比较器 CMP，同步/过压/电压浪涌/电流浪涌比较器
 - 10 位 PPG 定时器
 - PPGWDT 定时器
 - 过压自减 PPG_TIMER

型号说明

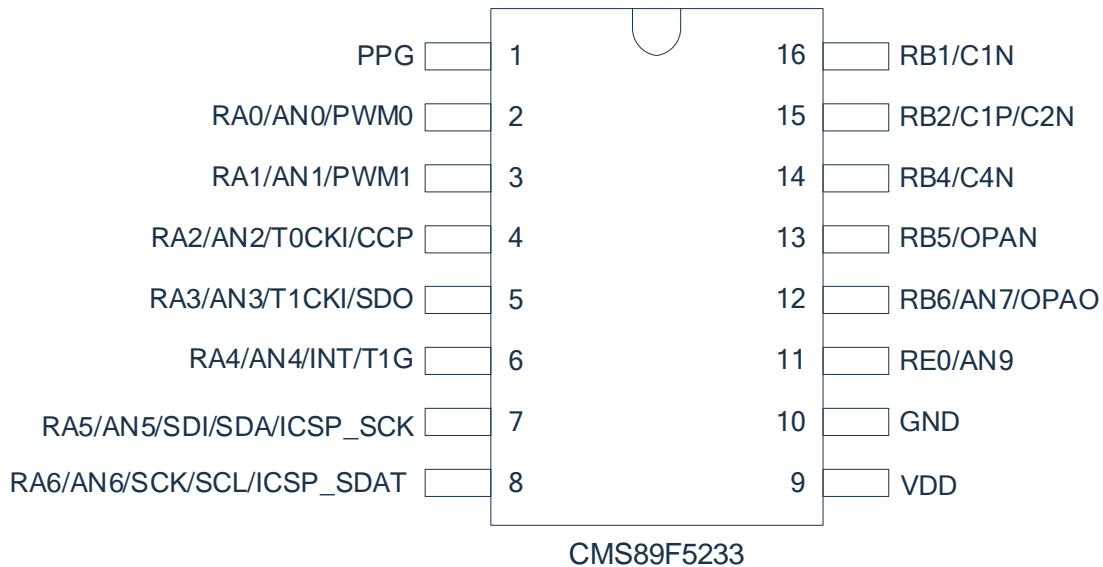
PRODUCT	FLASH	RAM	EEPROM	I/O	ADC	COMP	OPA	PACKAGE
CMS89F5233	4K×16Bit	256	32×16Bit	14	10Bit×9	3	1	DIP16/SOP16
CMS89F526	4K×16Bit	256	32×16Bit	18	10Bit×11	5	1	DIP20/SOP20

1.2 系统结构框图



1.3 管脚分布

1.3.1 CMS89F5233 引脚图



CMS89F5233 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-RA6	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
RB1-RB2/ RB4-RB6	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
RE0	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
ICSP_SCK	I	编程时钟输入脚
ICSP_SDAT	I/O	编程数据输入/输出脚
AN0-AN3/AN5-AN7/ AN9-AN10	I	AD 通道输入脚
INT	I	外部中断输入脚
T1G	I	TIMER1 门控输入脚
T0CKI/T1CKI	I	TIMER0/1 外部时钟输入脚
CCP	I	捕捉模式输入脚
SDI/SDA/SCK/SCL/SDO/SS	I/O	I ² C/SPI 数据/时钟/控制脚
C1P	I	同步比较器正端输入脚
C1N	I	同步比较器负端输入脚
C2N	I	过压比较器负端输入脚
C4N	I	电流浪涌功比较器负端输入脚
OPAN/OPAO	I/O	运放负端输入/输出脚
PPG	O	IGBT 输出控制脚（开漏输出）

1.3.2 CMS89F526 引脚图



CMS89F526 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-RA6	I/O	可编程为输入脚, 推挽输出脚, 带上拉电阻功能
RB0-RB6	I/O	可编程为输入脚, 推挽输出脚, 带上拉电阻功能
RE0/RE1	I/O	可编程为输入脚, 推挽输出脚, 带上拉电阻功能
ICSP_SCK	I	编程时钟输入脚
ICSP_SDAT	I/O	编程数据输入/输出脚
INT	I	外部中断输入脚
AN0-AN10	I	AD 通道输入脚
T0CKI/T1CKI	I	TIMER0/1 外部时钟输入脚
T1G	I	TIMER1 门控输入脚
CCP	I	捕捉模式输入脚
SDI/SDA/SCK/SCL/SDO/SS	I/O	I ² C/SPI 数据/时钟/控制脚
C1P	I	同步比较器正端输入
C1N	I	同步比较器负端输入
C2N	I	过压比较器负端输入
C3N	I	可选电压浪涌功能或过压功能
C4N	I	电流浪涌功比较器负端输入脚
C5N	I	电压浪涌功比较器负端输入脚
OPAN/OPAO	I/O	运放负端输入/输出脚
PPG	O	IGBT 输出控制脚 (开漏输出)

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，用户不能访问及操作。它包含了以下内容：

1. OSC（振荡方式选择）

◆ INTRC 内部 RC 振荡

2. WDT（看门狗选择）

◆ ENABLE 打开看门狗定时器

◆ DISABLE 关闭看门狗定时器

3. PROTECT（加密）

◆ DISABLE FLASH 代码不加密

◆ ENABLE FLASH 代码加密，加密后读出来的值将不确定

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。

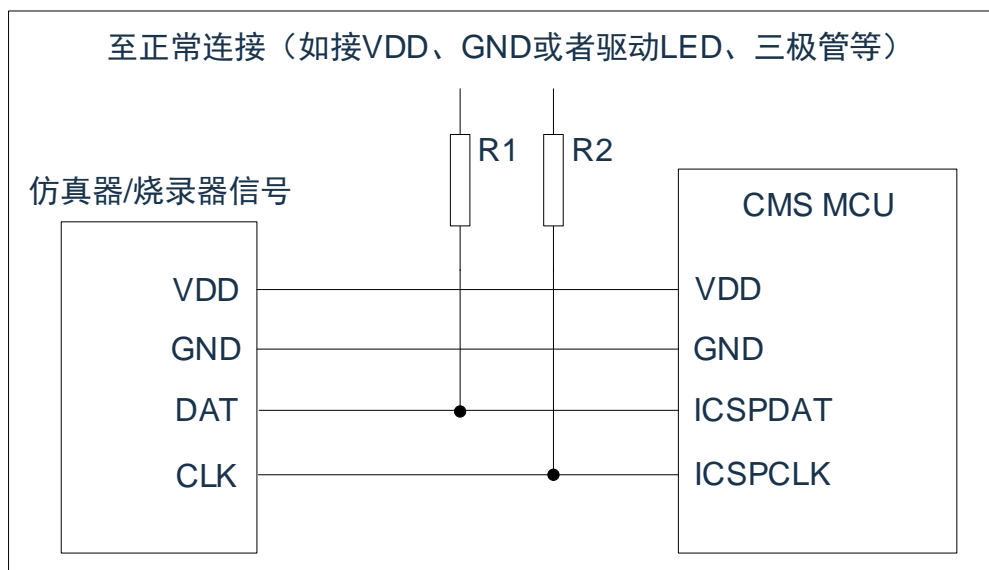


图 1-1：典型的在线串行编程连接方法

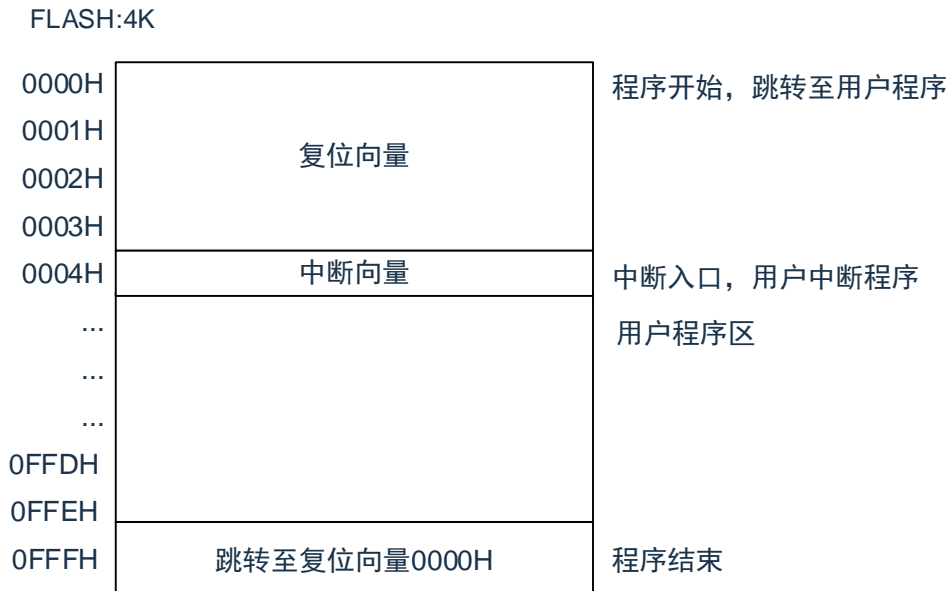
上图中，R1、R2 为电气隔离器件，常以电阻代替，其阻值如下： $R1 \geq 4.7K$ 、 $R2 \geq 4.7K$ 。

2. 中央处理器（CPU）

2.1 内存

2.1.1 程序内存

CMS89F5231/5232/5233/526 程序存储器空间



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 3 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	
	ORG	0004H	;用户程序起始
INT_START:	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 A 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：

- 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) 可擦写寄存器，用来指明表格高 5 位地址。
- TABLE_SPL(111H) 可擦写寄存器，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。

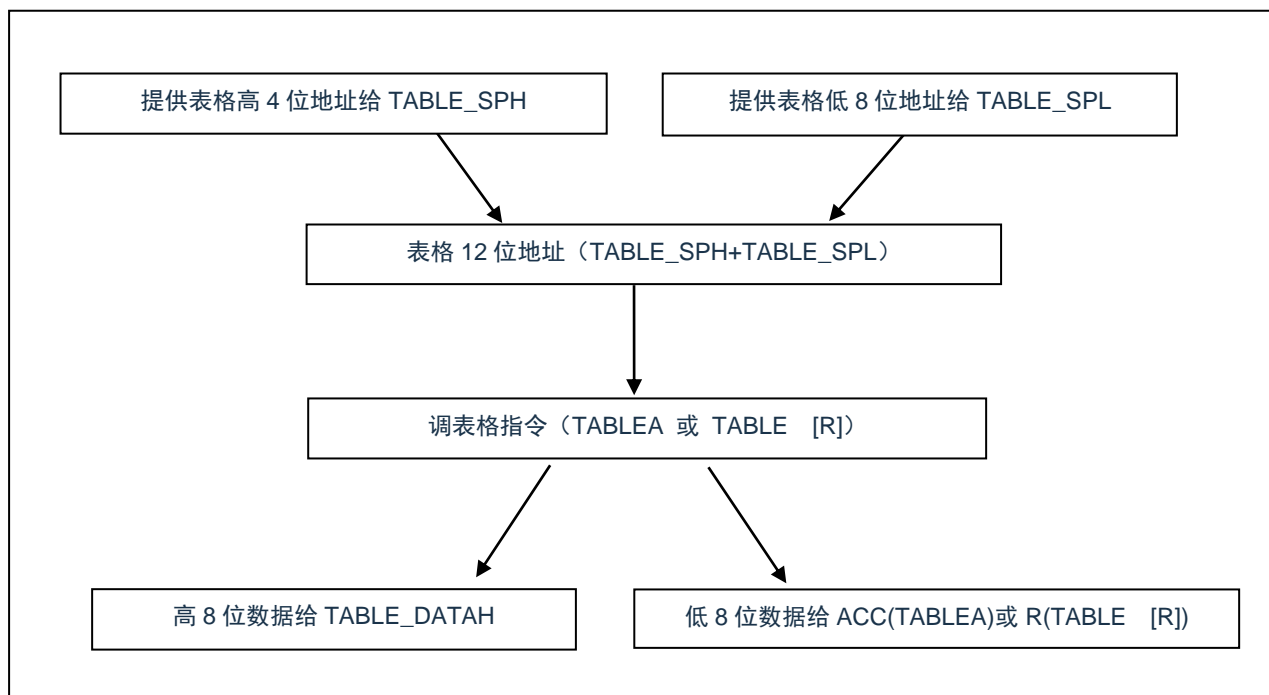


图2-1：表格调用的流程图

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 数据存储器的

CMS89F5231/5232/5233/526 数据存储器列表

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
---	07h	---	87h	OPAADJ	107h	PAANSEL	187h
---	08h	---	88h	OPACON	108h	PBANSEL	188h
PORTE	09h	TRISE	89h	OPACON1	109h	PEANSEL	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	EEDADR	10Dh	EECON2	18Dh
TMR1L	0Eh	---	8Eh	EEDATH	10Eh	CCPRL	18Eh
TMR1H	0Fh	OSCCON	8Fh	---	10Fh	CCPRH	18Fh
T1CON	10h	OSCTUNE	90h	TABLE_SPH	110h	CCPCON	190h
TMR2	11h	---	91h	TABLE_SPL	111h	SSPADD/SSPMSK	191h
T2CON	12h	PR2	92h	TABLE_DATAH	112h	SSPBUF	192h
---	13h	CM1CNT	93h	CM1ADJ	113h	SSPSTAT	193h
PPGTMR1	14h	WPUA	94h	CM2ADJ	114h	SSPCON	194h
PPGTMRH	15h	WPUB	95h	CM3ADJ	115h	SSPCON2	195h
PPGDLY	16h	WPUE	96h	CM4ADJ	116h	---	196h
PPGCON	17h	CM1CON	97h	CM5ADJ	117h	---	197h
PWM0DR	18h	CM2CON	98h	---	118h		198h
PWM0PR	19h	CM2CON1	99h	---	119h		199h
PWM0CR	1Ah	CM3CON	9Ah	---	11Ah		19Ah
PWM1DR	1Bh	CM3CON1	9Bh	---	11Bh		19Bh
PWM1PR	1Ch	CM4CON	9Ch	---	11Ch		19Ch
PWM1CR	1Dh	CM5CON	9Dh	---	11Dh		19Dh
---	1Eh	ADRESL	9Eh	---	11Eh		19Eh
ADCON0	1Fh	ADRESH	9Fh	---	11Fh	---	19Fh
	20h		A0h		120h		1A0h
通用寄存器 96 字节		通用寄存器 80 字节		通用寄存器 80 字节			
	6Fh		EFh		16Fh		1EFh
	70h		F0h	快速存储区 70h-7Fh	170h	快速存储区 70h-7Fh	1F0h
	--		--		--		--
	7Fh		FFh		17Fh		1FFh
BANK0		BANK1		BANK2		BANK3	

数据存储器由 512×8 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。

CMS89F52x 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	Tmr0数据寄存器								xxxx xxxx
02h	PCL	程序计数器低字节								0000 0000
03h	STATUS	IRP	----	----	TO	PD	Z	DC	C	0--1 1xxx
04h	FSR	间接数据存储器地址指针								xxxx xxxx
05h	PORTA	----	RA6	RA5	RA4	RA3	RA2	RA1	RA0	-xxx xxxx
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx
09h	PORTE	----	----	----	----	----	----	RE1	RE0	---- -xx
0Ah	PCLATH	----	---	----	程序计数器高5位的写缓冲器				---	0 0000
0Bh	INTCON	GIE	PEIE	TOIE	INTE	----	TOIF	INTF	----	0000 -0-
0Ch	PIR1	EEIF	ADIF	SSPIF	BCLIF	CCPIF	----	TMR2IF	TMR1IF	0000 0-00
0Dh	PIR2	----	----	C5IF	C4IF	C3IF	C2IF	C1IF	PPGWDTIF	--00 0000
0Eh	TMR1L	16位TMR1寄存器低字节的数据寄存器								xxxx xxxx
0Fh	TMR1H	16位TMR1寄存器高字节的数据寄存器								xxxx xxxx
10h	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	----	----	TMR1CS	TMR1ON	0000 --00
11h	TMR2	TIMER2模块寄存器								0000 0000
12h	T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000
14h	PPGTMR_L	PPGTMR 低8位								0000 0000
15h	PPGTMR_H	----	----	----	----	----	----	PPGTMR高2位		---- --00
16h	PPGDLY	----	----	----	----	PPGDLY				---- 0000
17h	PPGCON	DETC5F	DETC4F	----	RELOAD_EN	----	----	PPGMD	PPG_ON	11-0 --00
18h	PWM0DR	PWM0占空比数据寄存器								xxxx xxxx
19h	PWM0PR	PWM0周期数据寄存器								xxxx xxxx
1Ah	PWM0CR	PWM0EN	PWM0MOD	----	PWM0POL	PWM0CKS[3:0]				00-0 0000
1Bh	PWM1DR	PWM1占空比数据寄存器								xxxx xxxx
1Ch	PWM1PR	PWM1周期数据寄存器								xxxx xxxx
1Dh	PWM1CR	PWM1EN	PWM1MOD	----	PWM1POL	PWM1CKS[3:0]				00-0 0000
1Fh	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON	0000 0000

CMS89F52x 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	T0CS	T0SE	PSA	PS2	PS1	PS0	-111 1111
82h	PCL	程序计数器（PC）的低字节								0000 0000
83h	STATUS	IRP	----	----	TO	PD	Z	DC	C	0—1 1xxx
84h	FSR	间接数据存储器地址指针								xxxx xxxx
85h	TRISA	----	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	-111 1111
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111
89h	TRISE	----	----	----	----	----	----	TRISE1	TRISE0	---- -11
8Ah	PCLATH	----	----	----	程序计数器高5位的写缓冲器				---	0 0000
8Bh	INTCON	GIE	PEIE	T01E	INTE	----	T0IF	INTF	----	0000 -0-
8Ch	PIE1	EEIE	ADIE	SSPIE	BCLIE	CCPIE	----	TMR2IE	TMR1IE	0000 0-00
8Dh	PIE2	----	----	C5IE	C4IE	C3IE	C2IE	C1IE	PPGWDTIE	--00 0000
8Fh	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	----	-110 ----
90h	OSCTUNE	----	----	----	TUN4	TUN3	TUN2	TUN1	TUN0	---0 0000
92h	PR2	TIMER2周期寄存器								1111 1111
93h	CM1CNT	CM1OF	CM1CNT[6:0]							0000 0000
94h	WPUA	----	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	-000 0000
95h	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	0000 0000
96h	WPUE	----	----	----	----	----	----	WPUE1	WPUE0	---- --00
97h	CM1CON	CM1EN	CM1COFM	CM1CEN	CM1CLR	CM1NSL	----	----	----	0000 0---
98h	CM2CON	CM2EN	CM2COFM	CM2DBSEL[1:0]		CM2PVSL[3:0]				0000 0000
99h	CM2CON1	ATPEN	----	----	----	CM2COF	CM2COS[2:0]			0--- 0000
9Ah	CM3CON	CM3EN	CM3COFM	CM3DBSEL[1:0]		CM3PVSL[3:0]				0000 0000
9Bh	CM3CON1	CM3M1	CM3M0	----	CM3CIS	CM3COF	CM3COS[2:0]			00-0 0000
9Ch	CM4CON	CM4EN	CM4COFM	CM4DBSEL[1:0]		CM4PVSL[3:0]				0000 0000
9Dh	CM5CON	CM5EN	CM5COFM	CM5DBSEL[1:0]		CM5PVSL[3:0]				0000 0000
9Eh	ADRESL	A/D结果寄存器的低8位								xxxx xxxx
9Fh	ADRESH	A/D结果寄存器的高2位								---- --xx

CMS89F52x 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	----	----	TO	PD	Z	DC	C	0--1 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	OPAADJ	OPAOUT	OPARS	--	OPAADJS[4:0]				00-1 0000	
108h	OPACON	OPAEN	OPAFM	OPAFE	----	OPAPS1	OPAPS0	OPANS1	OPANS0	001- 0000
109h	OPACON1	----	----	----	----	OPO2ADE	-----	ANRS1	ANRS0	---- 0-00
10Ah	PCLATH	----	----	----	程序计数器高5位的写缓冲器				---0 0000	
10Bh	INTCON	GIE	PEIE	T01E	INTE	----	T0IF	INTF	----	0000 -00-
10Ch	EEDATA	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	0000 0000
10Dh	EEADR	----	----	----	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	---0 0000
10Eh	EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	--00 0000
110h	TABLE_SPH	表格高位指针								---x xxxx
111h	TABLE_SPL	表格低位指针								xxxx xxxx
112h	TABLE_DATAH	表格高位数据								xxxx xxxx
113h	CM1ADJ	CM1OUT	CM1CRS	CM1ADJ[5:0]				0010 0000		
114h	CM2ADJ	CM2OUT	CM2CRS	CM2ADJ[5:0]				0010 0000		
115h	CM3ADJ	CM3OUT	CM3CRS	CM3ADJ[5:0]				0010 0000		
116h	CM4ADJ	CM4OUT	CM4CRS	CM4ADJ[5:0]				0010 0000		
117h	CM5ADJ	CM5OUT	CM5CRS	CM5ADJ[5:0]				0010 0000		

CMS89F52x 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	1111 1111
182h	PCL	程序计（PC）的低字节								0000 0000
183h	STATUS	IRP	----	----	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
187h	PAANSEL	----	PAANS6	PAANS5	PAANS4	PAANS3	PAANS2	PAANS1	PAANS0	-000 0000
188h	PBANSEL	PBANS7	PBANS6	PBANS5	PBANS4	PBANS3	PBANS2	PBANS1	PBANS0	0000 0000
189h	PEANSEL	----	----	----	----	----	----	PEANS1	PEANS0	---- -00
18Ah	PCLATH	----	----	----	程序计数器高5位的写缓冲器				----	---0 0000
18Bh	INTCON	GIE	PEIE	T01E	INTE	----	T01F	INTF	----	0000 0000
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	CCPRL	捕捉寄存器的低字节								xxxx xxxx
18Fh	CCPRH	捕捉寄存器的高字节								xxxx xxxx
190h	CCPCON	CCPEN	----	----	CCPIS	CCPES	CPTM2	CPTM1	CPTM0	0--00000
191H	SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	11111111
191h	SSPADD	同步串行端口(I ² C模式)地址寄存器								0000 0000
192h	SSPBUF	同步串行端口接收缓冲器/ 发送寄存								xxxx xxxx
193h	SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	0000 0000
194h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000
195h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	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。

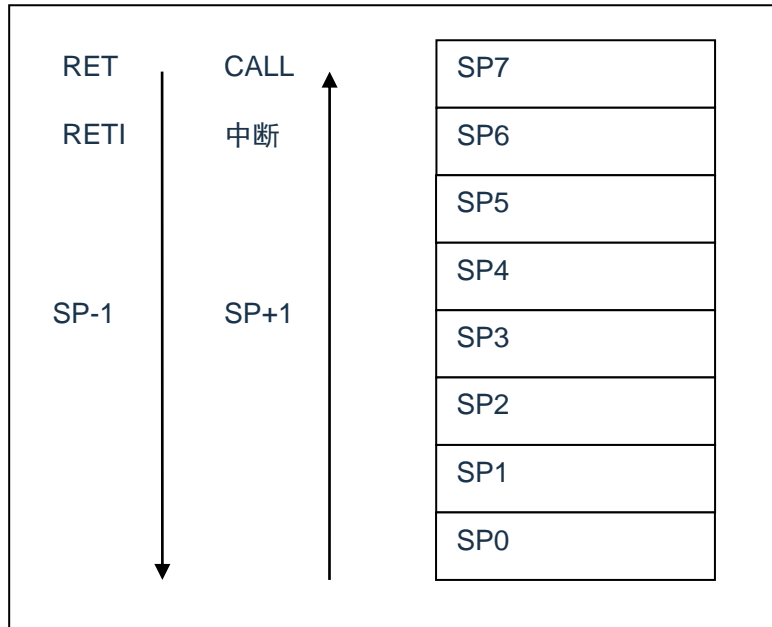


图 2-2：堆栈缓存器工作原理

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 8-Bit 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，CLRSTATUS 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	---	---	TO	PD	Z	DC	C
读写	R/W	---	---	R/W	R/W	R/W	R/W	R/W
复位值	0	---	---	1	1	X	X	X

Bit7	IRP: 寄存器存储器选择位 (用于间接寻址) ; 1= Bank2和Bank3 (100h-1FFh) ; 0= Bank0和Bank1 (00h-FFh) 。
Bit6~Bit5	禁用
Bit4	TO: 超时位; 1= 上电或执行了CLRWDT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CLRWDT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= 发生了结果的第4低位向高位进位; 0= 结果的第4低位没有向高位进位。
Bit0	C: 进位/借位位; 1= 结果的最高位发生了进位; 0= 结果的最高位没有发生进位。

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	0	按键唤醒休眠 MCU
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(181H)

181H	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 禁用

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: TIMER0 时钟源选择位。

0= 内部指令周期时钟 ($F_{osc}/4$)。

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，CLRWDT 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。它可以动态改变。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 12-Bit 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 $4K \times 16\text{Bit}$ 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。

“CLRWDT”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“CLRWDT”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“CLRWDT”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“CLRWDT”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，否则无法检测到主程序“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，以避免出现不必要的 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~Bit5 未用，读为 0。

Bit5~Bit4 保留（不要操作）。

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

时钟信号从由内部振荡器产生，经过 CONFIG 分频和寄存器分频产生系统工作时钟（F_{sys}）后，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。

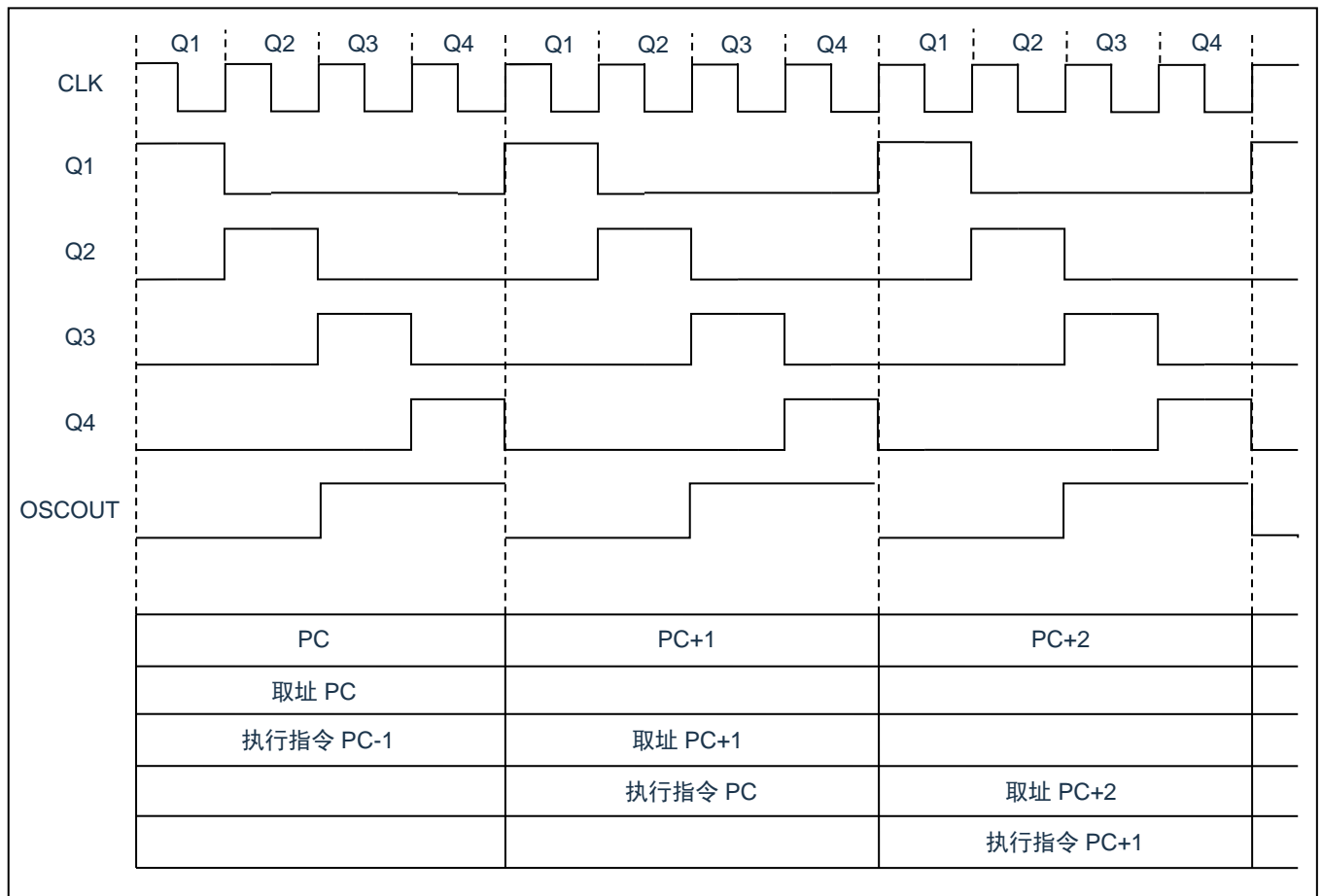


图 3-1：时钟与指令周期时序图

下面列出系统工作频率与指令速度的关系：

系统工作频率(F _{sys})	双指令周期	单指令周期
1MHz	8μs	4μs
2MHz	4μs	2μs
4MHz	2μs	1μs
8MHz	1μs	500ns
16MHz	500ns	250ns

3.2 系统振荡器

芯片只有 1 种振荡方式，内部 RC 振荡。

3.2.1 内部 RC 振荡

芯片默认的振荡方式为内部 RC 振荡，其振荡频率 F_{osc} 为 8M/16M 可通过 OSCCON 寄存器设置芯片工作频率。振荡频率在出厂时校正，其误差在 $\pm 3\%$ 以内。

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，其设计值约为 18ms@5V。

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>: 内部振荡器频率选择位。
 111= 8MHz。
 110= 4MHz（默认）。
 101= 2MHz。
 100= 1MHz。
 011= 500KHz。
 010= 250KHz。
 001= 125KHz。
 000= 31KHz（LFINTOSC）。
 Bit3~Bit0 禁用。

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 未用。
 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。

注：振荡控制寄存器均影响系统工作的时钟 SystemClock（Tsys 为系统工作时钟周期=1/Fsys）。

4. 复位

芯片可用如下 4 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；
- ◆ 休眠模式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 掉电复位概述

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）。

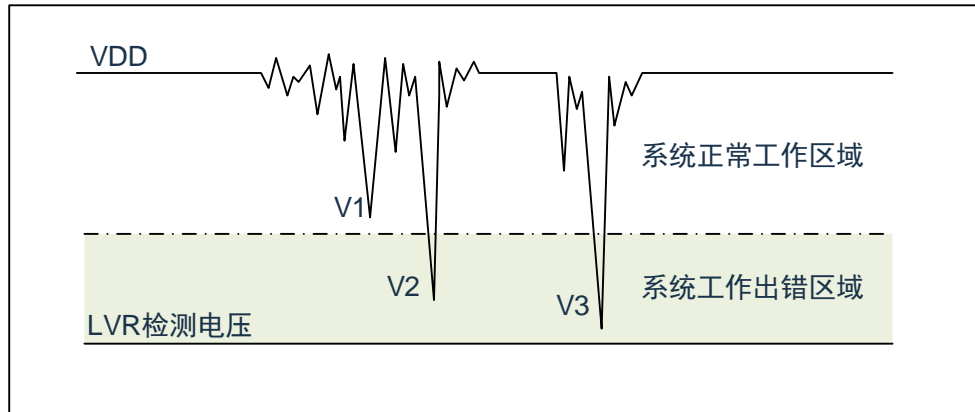


图4-1：掉电复位示意图

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：

- ◆ 开启 MCU 的低压侦测功能；
- ◆ 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，可由烧写 CONFIG 控制，详见 1.5 章关于烧写 CONFIG 选择说明。开启 LVR 功能时，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 指令可进入掉电模式。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，确保每一个输入口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	TRISE,A		
...			;关闭其它功能
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 3 个 I/O 端口：PORTA、PORTB、PORTE（最多 17 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，内部弱上拉，AN0，PWM0	I/O
	1	施密特触发输入，推挽式输出，内部弱上拉，AN1，PWM1	I/O
	2	施密特触发输入，推挽式输出，内部弱上拉，AN2，TMR0 时钟输入，CCP 输入	I/O
	3	施密特触发输入，推挽式输出，内部弱上拉，AN3，TMR1 时钟输入，SPI 数据输出口	I/O
	4	施密特触发输入，推挽式输出，内部弱上拉，AN4，外部中断输入，TMR1 门控输入	I/O
	5	施密特触发输入，推挽式输出，内部弱上拉，AN5，在线烧写和仿真数据口，I ² C 通讯数据口	I/O
	6	施密特触发输入，推挽式输出，内部弱上拉，AN6，在线烧写和仿真时钟口，I ² C 通讯时钟口	I/O
PORTB	0	施密特触发输入，推挽式输出，内部弱上拉，电压浪涌比较器负端输入	I/O
	1	施密特触发输入，推挽式输出，内部弱上拉，同步比较器负端输入	I/O
	2	施密特触发输入，推挽式输出，内部弱上拉，同步比较器正端输入，过压比较器负端输入	I/O
	3	施密特触发输入，推挽式输出，内部弱上拉，可选电压浪涌或过压比较器负端输入	I/O
	4	施密特触发输入，推挽式输出，内部弱上拉，电流浪涌比较器负端输入	I/O
	5	施密特触发输入，推挽式输出，内部弱上拉，运放负端输入	I/O
	6	施密特触发输入，推挽式输出，内部弱上拉，AN7，运放输出	I/O
	7	施密特触发输入，推挽式输出，内部弱上拉，AN8，运放正端输入	I/O
PORTE	0	施密特触发输入，推挽式输出，内部弱上拉，AN9	I/O
	1	施密特触发输入，推挽式输出，内部弱上拉，AN10	I/O

<表 6-1：端口配置总概>

6.1 I/O 口结构图

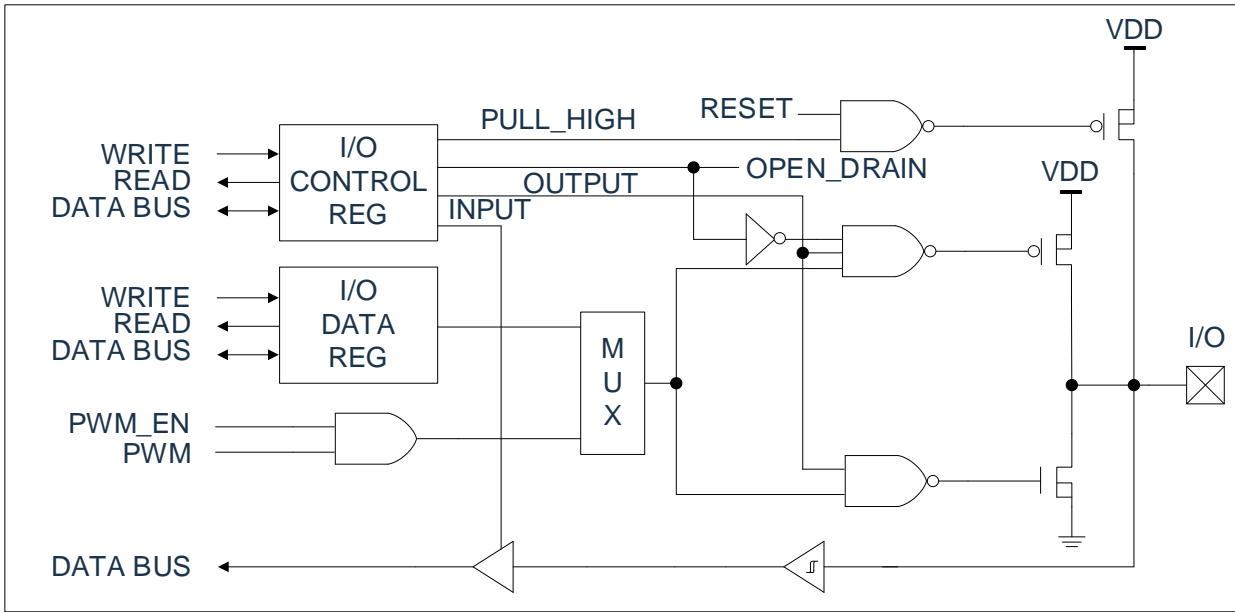


图 6-1: I/O 口结构图 (1)

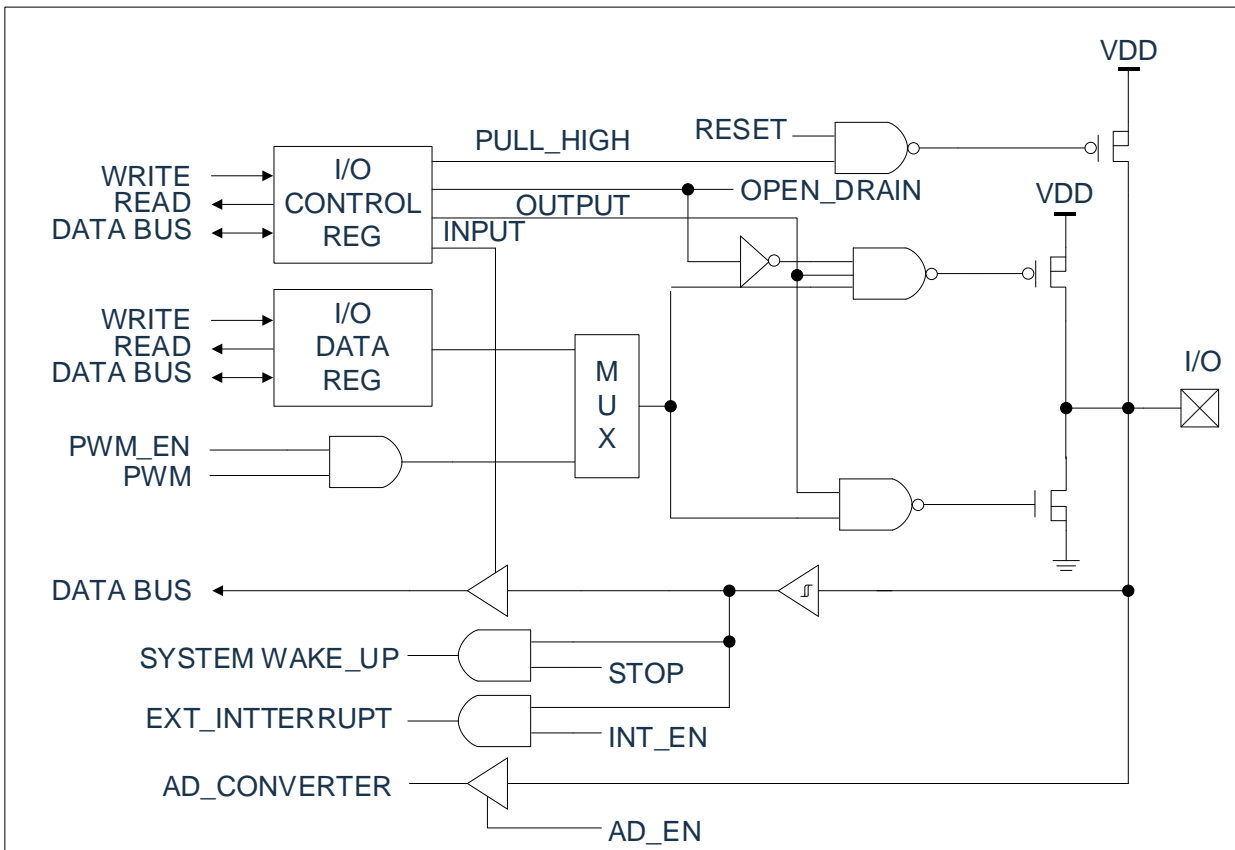


图 6-2: I/O 口结构图 (2)

6.2 PORTA

6.2.1 PORTA 数据及方向控制

PORTA 是 7Bit 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。

注：必须初始化 ANSEL 寄存器以将模拟通道配置为数字输入。配置为模拟输入的引脚将读为 0。

与 PORTA 口相关寄存器有 PORTA、TRISA、WPUA、PAANSEL 等。

PORTA 数据寄存器 PORTA (05H)

05H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTA	---	RA6	RA5	RA4	RA3	RA2	RA1	RA0
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

Bit6~Bit0 PORTA<6:0>: 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	---	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
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

Bit6~Bit0 TRISA<6:0>: 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

CLR	PAANSEL	;设置所有PORTA口为数字I/O口
LDIA	B'11110000'	;设置PORTA<3:0>为输出口, PORTA<6:4>为输入口
LD	TRISA,A	
LDIA	03H	;PORTA<1:0>输出高电平, PORTA<3:2>输出低电平
LD	PORTA,A	;由于PORTA<6:4>为输入口, 所以赋0或1都没影响

6.2.2 PORTA 模拟选择控制

PAANSEL 寄存器用于将 I/O 引脚的输入模式配置为模拟模式。将 PAANSEL 中适当的位置 1 将导致对相关引脚的所有数字读操作返回 0，并使引脚的模拟功能正常工作。PAANSEL 位的状态对数字输出功能没有影响。TRIS 清零且 PAANSEL 置 1 的引脚仍作为数字输出，但输入模式将成为模拟模式。这会导致在受影响的端口上执行读—修改—写操作时产生不可预计的结果。

PORTA 模拟选择寄存器 PAANSEL(187H)

187H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PAANSEL	---	PAANS6	PAANS5	PAANS4	PAANS3	PAANS2	PAANS1	PAANS0
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

Bit6~Bit0 PAANS<6:0>: 模拟选择位，分别选择引脚 PORTA<6:0>的模拟或数字功能。

1= 模拟输入，引脚被分配为模拟输入。

0= 数字 I/O，引脚被分配给端口或特殊功能。

6.2.3 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<6:0>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(94H)

94H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUA	---	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
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

Bit6~Bit0 WPUA<6:0>: 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 TRISB。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、PBANSEL、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	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
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 模拟选择

PBANSEL 寄存器用于将 I/O 引脚的输入模式配置为模拟模式。将 PBANSEL 中适当的位置 1 将导致对相关引脚的所有数字读操作返回 0，并使引脚的模拟功能正常工作。PBANSEL 位的状态对数字输出功能没有影响。方向寄存器清零且 PBANSEL 置 1 的引脚仍作为数字输出，但输入模式将成为模拟模式。这会导致在受影响的端口上执行读—修改—写操作时产生不可预计的结果。

PORTB 模拟选择寄存器 PBANSEL(188H)

188H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PBANSEL	PBANS7	PBANS6	PBANS5	PBANS4	PBANS3	PBANS2	PBANS1	PBANS0
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 PBANS<7:0>: 模拟选择位，分别选择引脚 PORTB<7:0>的模拟或数字功能。
 1= 模拟输入，引脚被分配为模拟输入。
 0= 数字I/O，引脚被分配给端口或特殊功能。

6.3.3 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(95H)

95H	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

Bit7~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 PORTE

6.4.1 PORTE 数据及方向

PORTE 是一个 2Bit 宽的双向端口。对应的数据方向寄存器为 TRISE。将 TRISE 中的某个位置 1 (=1) 可以使对应的 PORTE 引脚作为输入引脚。将 TRISE 中的某个位清零 (=0) 将使对应的 PORTE 引脚作为输出引脚。

读 PORTE 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。

PORTE 数据寄存器 PORTE(09H)

09H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTE	---	---	---	---	---	---	RE1	RE0
R/W	---	---	---	---	---	---	R/W	R/W
复位值	---	---	---	---	---	---	X	X

Bit1~Bit0 PORTE<1:0>: PORTEI/O 引脚位。
 1= 端口引脚为高电平。
 0= 端口引脚为低电平。

PORTE 方向寄存器 TRISE(89H)

89H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISE	---	---	---	---	---	---	TRISE1	TRISE0
R/W	---	---	---	---	---	---	R/W	R/W
复位值	---	---	---	---	---	---	1	1

Bit1~Bit0 TRISE<1:0>: PORTE 三态控制位。
 1= PORTE 引脚被配置为输入（三态）。
 0= PORTE 引脚被配置为输出。

6.4.2 PORTE 上拉电阻

每个 PORTE 引脚都有可单独配置的内部弱上拉。控制位 WPUE<1:0>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。

PORTE 上拉电阻寄存器 WPUE(96H)

96H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUD	---	---	---	---	---	---	WPUE1	WPUE0
R/W	---	---	---	---	---	---	R/W	R/W
复位值	---	---	---	---	---	---	0	0

Bit1~Bit0 WPUE<1:0>: 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。

6.4.3 PORTE 模拟选择

PEANSEL 寄存器用于将 I/O 引脚的输入模式配置为模拟模式。将 PEANSEL 中适当的位置 1 将导致对相关引脚的所有数字读操作返回 0，并使引脚的模拟功能正常工作。PEANSEL 位的状态对数字输出功能没有影响。方向寄存器清零且 PEANSEL 置 1 的引脚仍作为数字输出，但输入模式将成为模拟模式。这会导致在受影响的端口上执行读—修改—写操作时产生不可预计的结果。

PORTE 模拟选择寄存器 PEANSEL(189H)

189H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PEANSEL	---	---	---	---	---	---	PEANS1	PEANS0
R/W	---	---	---	---	---	---	R/W	R/W
复位值	---	---	---	---	---	---	0	0

- Bit1~Bit0 PEANS<1:0> 模拟选择位,分别选择引脚 PORTE<1:0>的模拟或数字功能。
 1= 模拟输入。引脚被分配为模拟输入。
 0= 数字 I/O。引脚被分配给端口或特殊功能。

6.5 I/O 使用

6.5.1 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 口置零
CLR	PORTA	;PORTA 口清零
SET	PORTA	;PORTA 所有输出口置 1
SETB	PORTB,1	;PORTB.1 口置 1

6.5.2 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。

6.6 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。

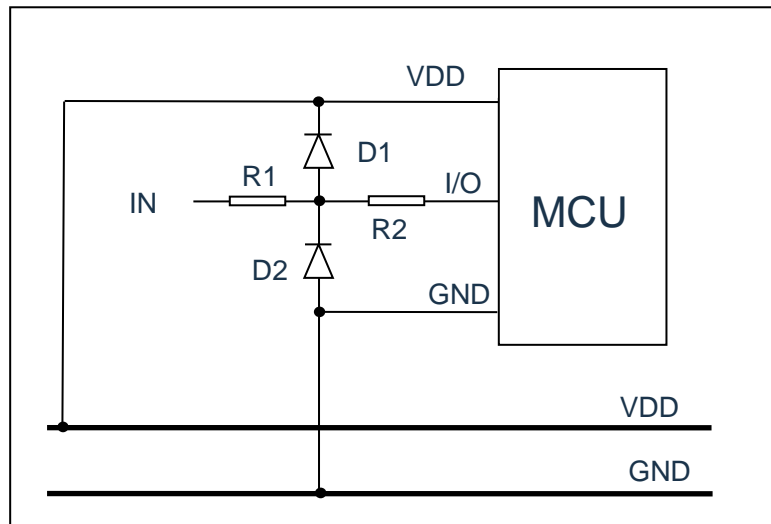


图 6-3: 输入电压不在规定范围内采用电路

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；
- ◆ A/D 中断；
- ◆ 比较器中断；
- ◆ MSSP 中断；
- ◆ TIMER1 溢出中断；
- ◆ INT 中断；
- ◆ CCP 中断；
- ◆ PPGWDT 溢出中断；
- ◆ 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。

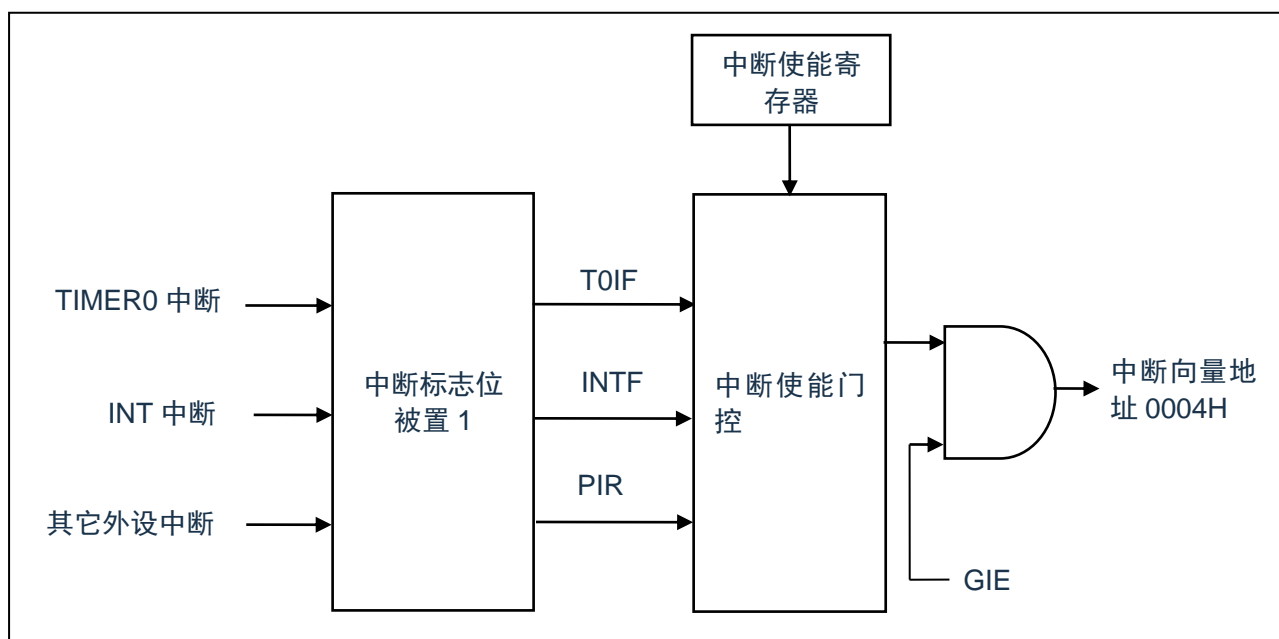


图 7-1: 中断原理示意图

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、PORTB 端口电平变化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	EEIE	ADIE	SSPIE	BCLIE	CCPIE	---	TMR2IE	TMR1IE
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	EEIE: EEPROM写操作中断允许位; 1= 允许EEPROM写操作中断; 0= 禁止EEPROM写操作中断。
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	SSPIE: 主同步串行端口 (MSSP) 中断允许位; 1= 允许MSSP中断; 0= 禁止MSSP中断。
Bit4	BCLIE: 总线冲突中断允许位; 1= 允许总线冲突中断; 0= 禁止总线冲突中断。
Bit3	CCPIE: CCP中断允许位; 1= 允许CCP中断; 0= 禁止CCP中断。
Bit2	未用。
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	TMR1IE: TIMER1溢出中断允许位; 1= 允许TIMER1溢出中断; 0= 禁止TIMER1溢出中断。

外设中断允许寄存器 PIE2(8DH)

8DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE2	---	---	C5IE	C4IE	C3IE	C2IE	C1IE	PPGWDTIE
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	---	0	0	0	0	0	0

Bit7~Bit6	未用
Bit5	C5IE: 比较器C5中断允许位; 1= 允许比较器C5中断; 0= 禁止比较器C5中断。
Bit4	C4IE: 比较器C4中断允许位; 1= 允许比较器C4中断; 0= 禁止比较器C4中断。
Bit3	C3IE: 比较器C3中断允许位; 1= 允许比较器C3中断; 0= 禁止比较器C3中断。
Bit2	C2IE: 比较器C2中断允许位; 1= 允许比较器C2中断; 0= 禁止比较器C2中断。
Bit1	C1IE: 比较器C1中断允许位; 1= 允许比较器C1中断; 0= 禁止比较器C1中断。
Bit0	PPGWDTIE: PPGWDT溢出中断允许位; 1= 允许PPGWDT溢出中断; 0= 禁止PPGWDT溢出中断。

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	EEIF	ADIF	SSPIF	BCLIF	CCPIF	---	TMR2IF	TMR1IF
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	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。
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	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: <ul style="list-style-type: none"> - 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。
Bit4	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。
Bit3	CCP1IF: CCP1中断标志位。 捕捉模式: 1= 发生了TMR1寄存器的捕捉 (必须由软件清零); 0= 没有发生TMR1寄存器的捕捉。 比较模式: 1= 发生了TMR1寄存器的比较匹配 (必须由软件清零); 0= 没有发生TMR1寄存器的比较匹配。 PWM模式: 在此模式下未用。
Bit2	禁用。
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	TMR1IF: TIMER1溢出中断标志位。 1= TMR1寄存器溢出 (必须由软件清零); 0= TMR1寄存器未溢出。

外设中断请求寄存器 PIR2(0DH)

0DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR2	---	---	C5IF	C4IF	C3IF	C2IF	C1IF	PPGWDTIF
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	---	0	0	0	0	0	0

Bit7~Bit6

禁用。

Bit5

C5IF: 比较器C5中断标志位;

1= 比较器输出 (C5OUT位) 发生了改变 (必须由软件清零);

0= 比较器输出 (C5OUT位) 不变。

Bit4

C4IF: 比较器C4中断标志位;

1= 比较器输出 (C4OUT位) 发生了改变 (必须由软件清零);

0= 比较器输出 (C4OUT位) 不变。

Bit3

C3IF: 比较器C3中断标志位;

1= 比较器输出 (C3OUT位) 发生了改变 (必须由软件清零);

0= 比较器输出 (C3OUT位) 不变。

Bit2

C2IF: 比较器C2中断标志位;

1= 比较器输出 (C2OUT位) 发生了改变 (必须由软件清零);

0= 比较器输出 (C2OUT位) 不变。

Bit1

C1IF: 比较器C1中断标志位;

1= 比较器输出 (C1OUT位) 发生了改变 (必须由软件清零);

0= 比较器输出 (C1OUT位) 不变。

Bit0

PPGWDTIF: PPGWDT溢出中断标志位;

1= 发生了PPGWDT溢出 (必须由软件清零);

0= 未发生PPGWDT溢出。

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。

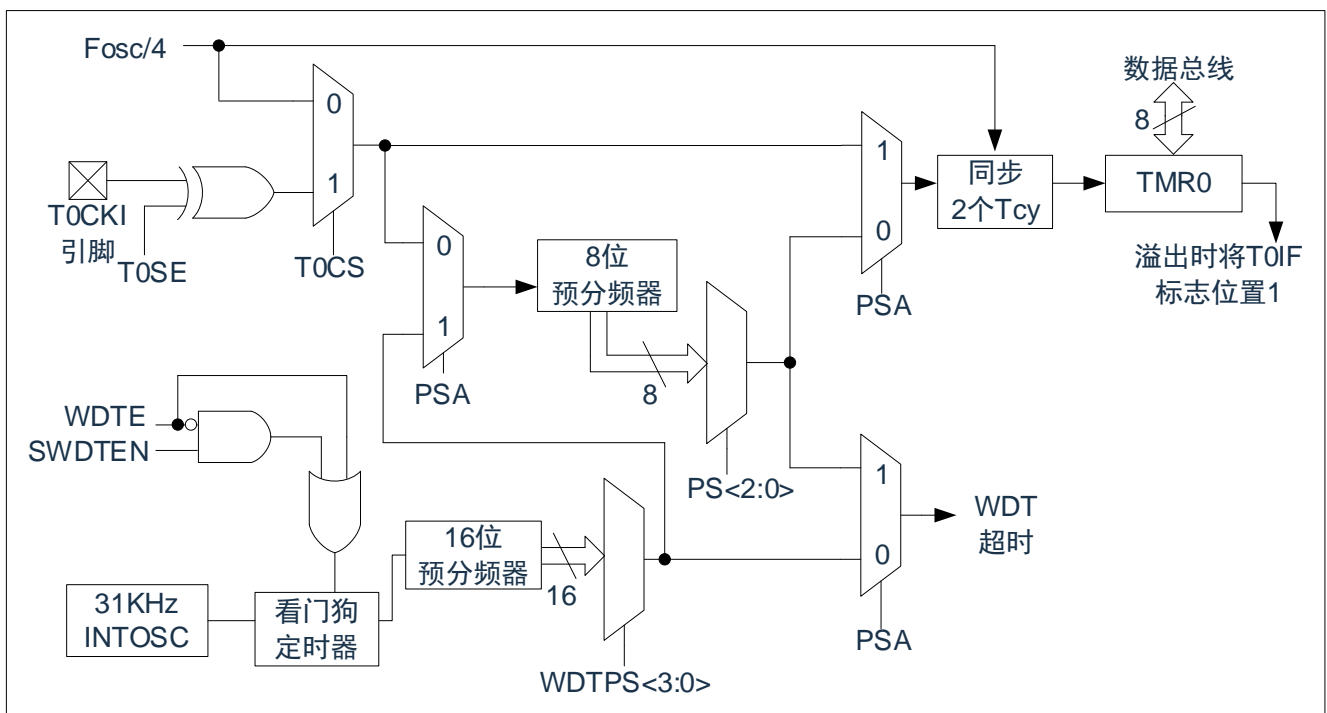


图 8-1: TIMER0/WDT 模块结构图

注：

1. T0SE、T0CS、PSA、PS<2:0>为OPTION_REG寄存器中的位。
2. SWDTEN和WDTPS<3:0>为WDTCON寄存器中的位。
3. WDTE位在配置字寄存器1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，必须执行如下所示的指令序列。

更改预分频器（TMR0-WDT）

CLRWDT		
CLR	TMR0	
SETB	OPTION_REG,PSA	;选择 WDT
CLRWDT		
LDIA	B'11111000'	
AND	OPTION_REG	;低 3 位置 0
OR	B'00000101'	;低 3 位置成 101，其余位不变
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		
LDIA	B'11110000'	
AND	OPTION_REG	;低 4 位置 0
OR	B'00000101'	;低 4 位置成 0101，其余位不变
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 TMR0 相关，8 位定时器/计数器（TMR0），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 TMR0 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(181H)

181H	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	禁用							
Bit6	INTEDG: 中断边沿选择位。							
	1= INT 引脚的上升沿触发中断。							
	0= INT 引脚的下降沿触发中断。							
Bit5	T0CS: TMR0 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	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		

9. 定时计数器 TIMER1

9.1 TIMER1 概述

TIMER1 模块是一个 16 位定时器/计数器，具有以下特性：

- ◆ 16 位定时器/计数器寄存器 (TMR1H:TMR1L)
- ◆ 3 位预分频器
- ◆ 溢出时唤醒 (仅外部时钟异步模式)
- ◆ 特殊事件触发功能 (带有 ECCP)
- ◆ 通过 T1G 引脚门控 TIMER1 (使能计数)
- ◆ 溢出中断
- ◆ 捕捉/比较功能的时基

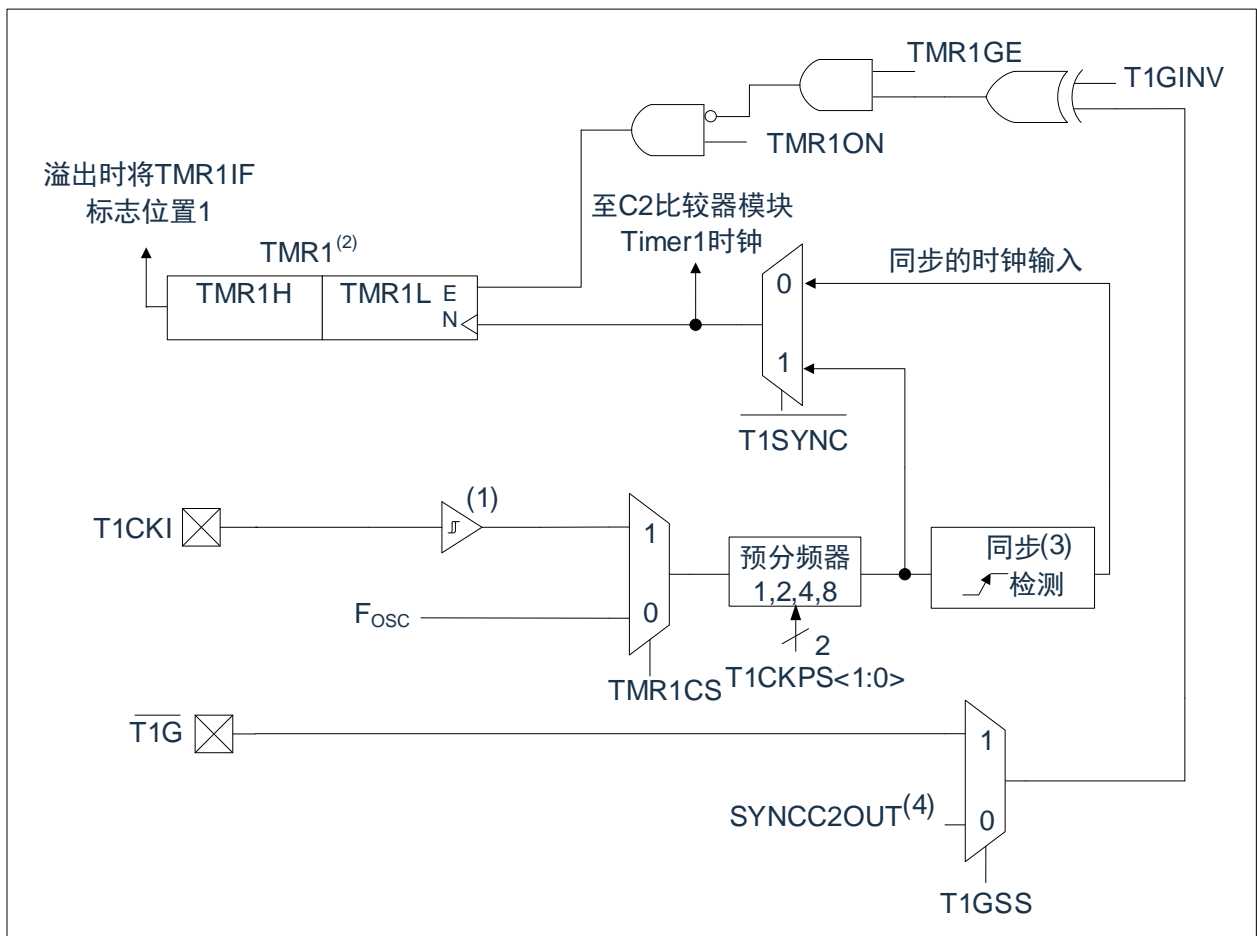


图9-1: TIMER1结构图

注：

1. ST 缓冲器在使用 LP 振荡器时处于低功耗模式，而在使用 T1CKI 时处于高速模式。
2. Timer1 寄存器在上升沿递增。
3. 休眠时不进行同步。
4. 当CM2CON1寄存器的C2SYNC位置1时，SYNCC2OUT同步。

9.2 TIMER1 的工作原理

TIMER1 模块是一个通过一对寄存器 TMR1H:TMR1L 访问的 16 位递增计数器。写入 TMR1L 将只写入到内部缓冲寄存器，写入 TMR1H 将把内部缓冲寄存器加载到 TIMER1 计数器，因此在对 TMR1L 和 TMR1H 进行写操作时，必须先写 TMR1L 寄存器，再写 TMR1H 寄存器。

当 TIMER1 工作时，TMR1H:TMR1L 寄存器将以 F_{osc} 的倍数为频率递增，具体倍数由 TIMER1 预分频器决定。

9.3 TIMER1 预分频器

TIMER1 具有四种预分频比选择，允许对时钟输入进行 1、2、4 或 8 分频。T1CON 寄存器的 T1CKPS 位控制预分频计数器。不能直接对预分频计数器进行读或写操作。

9.4 TIMER1 中断

一对 TIMER1 寄存器 (TMR1H:TMR1L) 递增计数到 FFFFh 后，将溢出返回 0000h。当 TIMER1 溢出时，PIR1 寄存器的 TIMER1 中断标志位被置 1。要允许该溢出中断，用户应将以下位置 1：

- PIE1 寄存器中的 TIMER1 中断允许位；
- INTCON 寄存器中的 PEIE 位；
- INTCON 寄存器中的 GIE 位；
- 在中断服务程序中将 TMR1IF 位清零可以清除该中断。

注：再次允许该中断前，应将 TMR1H:TMR1L 这对寄存器以及 TMR1IF 位清零。由于在休眠状态下定时器是关闭的，所以 TIMER1 中断无法唤醒处理器。

9.5 TIMER1 相关寄存器

TIMER1 主要由 3 个 RAM 控制：TIMER1 控制寄存器 T1CON、数据寄存器 TMR1L、TMR1H。数据寄存器在赋值的时候必须先赋值低位 TMR1L，再赋值 TMR1H。

TIMER1 数据低位寄存器 TMR1L(0EH)

0EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR1L								
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

TIMER1 数据高位寄存器 TMR1H(0FH)

0FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR1H								
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

TIMER1 控制寄存器 T1CON(10H)

10H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	---	---	TMR1CS	TMR1ON
读写	R/W	R/W	R/W	R/W	---	---	R/W	R/W
复位值	0	0	0	0	---	---	0	0

- Bit7 T1GINV: TIMER1门控信号极性位(1);
 1= TIMER1门控信号高电平有效（当门控信号为高电平时TIMER1计数）；
 0= TIMER1门控信号低电平有效（当门控信号为低电平时TIMER1计数）。
- Bit6 TMR1GE: TIMER1门控使能位(2)。
 如果TMR1ON= 0: 此位被忽略。
 如果TMR1ON =1:
 1= TIMER1计数由TIMER1门控功能控制；
 0= TIMER1始终计数。
- Bit5~Bit4 T1CKPS<1:0>: TIMER1输入时钟预分频比选择位；
 11= 1:8预分频比；
 10= 1:4预分频比；
 01= 1:2预分频比；
 00= 1:1预分频比。
- Bit3~Bit2 禁用
- Bit1 TMR1CS: TIMER1时钟源选择位；
 1= 来自T1CKI引脚的外部时钟源（上升沿触发）；
 0= 内部时钟源（F_{osc}/4）。
- Bit0 TMR1ON: TIMER1使能位；
 1= 使能TIMER1；
 0= 禁止TIMER1。

注：

1. T1GINV 位可使 TIMER1 门控信号的逻辑电平反相，而不管门控信号源如何。
2. TMR1GE 位必须置 1，以使用 T1G 引脚作为 TIMER1 的门控信号源。

10. 定时计数器 TIMER2

10.1 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)。

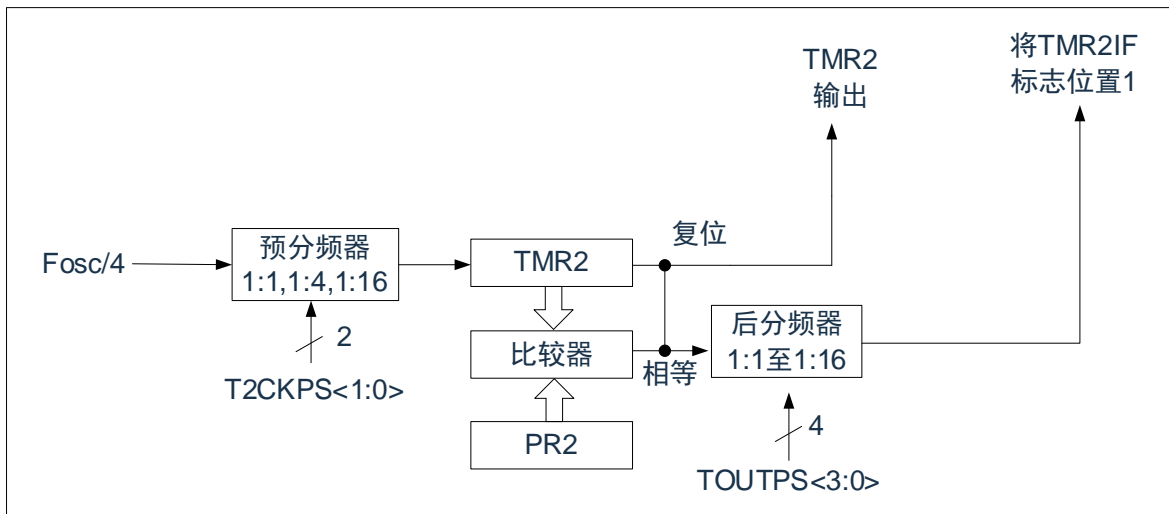


图 10-1: TIMER2 框图

10.2 TIMER2 的工作原理

TIMER2 模块的时钟输入是系统指令时钟 (FOSC/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。

10.3 TIMER2 相关的寄存器

有 2 个寄存器与 TIMER2 相关，分别是数据存储器 TMR2，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，读为 0
- 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。

11. 模数转换 (ADC)

11.1 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 10 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 10 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。

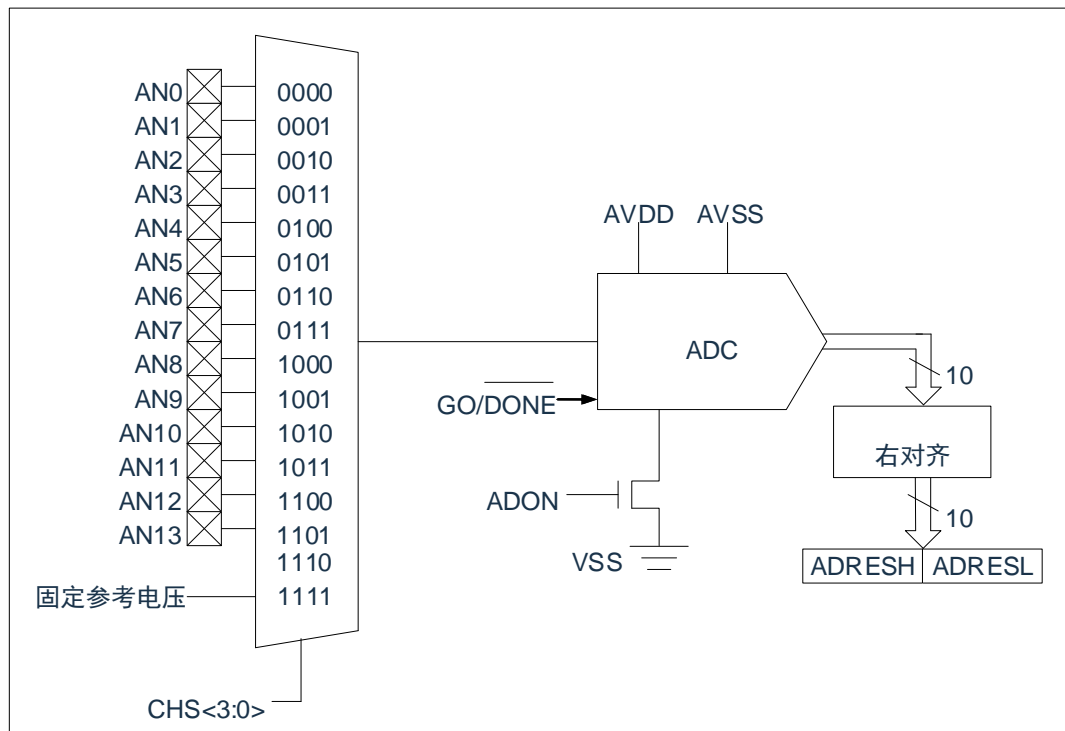


图 11-1: ADC 框图

11.2 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。

11.2.1 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 和 ANSEL 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。

11.2.2 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”。

11.2.3 ADC 参考电压

ADC 的参考电压始终是由芯片的 VDD 和 GND 提供。

11.2.4 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：

- ◆ $F_{osc}/8$
- ◆ $F_{osc}/32$
- ◆ $F_{osc}/16$
- ◆ F_{RC} （专用内部振荡器）

完成一位转换的时间定义为 TAD。一个完整的 10 位转换需要 41 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，下表为正确选择 ADC 时钟的示例。

注：除非使用 F_{RC} ，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
$F_{osc}/8$	00	49.0 μ s	98.0 μ s	392.0 μ s
$F_{osc}/16$	01	98.0 μ s	196.0 μ s	784.0 μ s
$F_{osc}/32$	10	196.0 μ s	392.0 μ s	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。

11.2.5 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。

11.3 ADC 工作原理

11.3.1 启动转换

要使能 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。

11.3.2 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。

11.3.3 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。

11.3.4 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 F_{RC} 选项。如果选择了 F_{RC} 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 F_{RC}，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。

11.3.5 A/D 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 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 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'10000000'
LD            ADCON1,A
SETB         TRISA,0           ;设置 PORTA.0 为输入口
SETB         ANSEL,0          ;设置 PORTA.0 为模拟口
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
    
```

11.4 ADC 相关 RAM

主要有 3 个 RAM 与 AD 转换相关，分别是控制寄存器 ADCON、数据寄存器 ADRESH 和 ADRESL。

AD 控制寄存器 ADCON(1FH)

1FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/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= Fosc/8
 01= Fosc/16
 10= Fosc/32
 11= FRC (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= AN8
 1001= AN9
 1010= AN10
 1011= ----
 1100= ----
 1101= ----
 1110= CVREF
 1111= 固定参考电压 (0.6V固定参考电压)
- Bit1 GO/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 数据寄存器高位 ADRESH(9FH)

9FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	----	----	----	----	----	----	ADRES9	ADRES8
读写	----	----	----	----	----	----	R	R
复位值	----	----	----	----	----	----	X	X

Bit7~Bit0 ADRES<9:8>: ADC结果寄存器位。
10位转换结果的高2位。

AD 数据寄存器低位 ADRESL(9EH)

9EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2	ADRES1	ADRES0
读写	R	R	R	R	R	R	R	R
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0 ADRES<7:0>: ADC结果寄存器位。
10位转换结果的低8位。

12. PWM 模块

12.1 PWM 特性

- ◆ 8Bit 精度；
- ◆ 输出极性可以选择；
- ◆ 计数频率可选；
- ◆ 可选择中心对齐或边缘对齐输出方式。

CMS89F52x 内建两路 8 位 PWM 模块。PWM 模块可以产生周期和占空比分别可以调整的脉宽调制波形。两路 PWM 产生的波形分别由 RA0 和 RA1 口输出。

12.2 PWM 相关寄存器

PWM 功能相关寄存器有：控制寄存器 PWM0CR，PWM1CR；周期设置寄存器 PWM0PR，PWM1PR；占空比设置寄存器 PWM0DR，PWM1DR。

PWM0 占空比设置寄存器 PWM0DR(18H)

18H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM0DR	PWM0DR[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

PWM0 周期设置寄存器 PWM0PR(19H)

19H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM0PR	PWM0PR[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

PWM0 控制寄存器 PWM0CR(1AH)

1AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM0CR	PWM0EN	PWM0MOD	---	PWM0POL	PWM0CKS[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	PWM0EN:	PWM0 使能位	
	0:	禁止	
	1:	使能 (PWM0 口设置为输出, 且输出 PWM0 波形)	
Bit6	PWM0MOD:	PWM0 模式选择位	
	0:	正常模式	
	1:	back-to-back 模式	
Bit5		禁用	
Bit4	PWM0POL:	PWM0 输出极性选择位	
	0:	正常输出	
	1:	反相输出	
Bit3~Bit0	PWM0CKS[3:0]:	PWM0 时钟频率选择位	
	0000:	Fosc	1000: Fosc/256
	0001:	Fosc/2	1001: Fosc/512
	0010:	Fosc/4	1010: Fosc/1024
	0011:	Fosc/8	1011: Fosc/2048
	0100:	Fosc/16	1100: Fosc/4096
	0101:	Fosc/32	1101: Fosc/8192
	0110:	Fosc/64	111x: Fosc/8192
	0111:	Fosc/128	

PWM1 占空比设置寄存器 PWM1DR(1BH)

1BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1DR	PWM1DR[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

PWM1 周期设置寄存器 PWM1PR(1CH)

1CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1PR	PWM1PR[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

PWM1 控制寄存器 PWM1CR(1DH)

1DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1CR	PWM1EN	PWM1MOD	---	PWM1POL	PWM1CKS[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	PWM1EN:	PWM1 使能位	
	0:	禁止	
	1:	使能 (PWM1 口设置为输出, 且输出 PWM1 波形)	
Bit6	PWM1MOD:	PWM1 模式选择位	
	0:	正常模式	
	1:	back-to-back 模式	
Bit5		禁用	
Bit4	PWM1POL:	PWM1 输出极性选择位	
	0:	正常输出	
	1:	反相输出	
Bit3~Bit0	PWM1CKS[3:0]:	PWM1 时钟频率选择位	
	0000:	Fosc	1000: Fosc/256
	0001:	Fosc/2	1001: Fosc/512
	0010:	Fosc/4	1010: Fosc/1024
	0011:	Fosc/8	1011: Fosc/2048
	0100:	Fosc/16	1100: Fosc/4096
	0101:	Fosc/32	1101: Fosc/8192
	0110:	Fosc/64	111x: Fosc/8192
	0111:	Fosc/128	

- PWM0/PWM1 的相关参数: $n=0, 1$
- PWMn 周期: $T_{pwm} = (PWMnPR[7:0] + 1) \times 2^{PWMnCKS[3:0]} \times T_{sys}$ (最长周期: $262Ms @ F_{osc} = 8MHz$)
- PWMn 高电平脉冲时间: $T_{pwmh} = PWMnDR[7:0] \times 2^{PWMnCKS[3:0]} \times T_{sys}$
- PWMn 占空比: $T_{pwmh} / T_{pwm} = PWMnDR[7:0] / (PWMnPR[7:0] + 1)$ (设置范围: 0%-100%, 最大精度 1/256)
- PWMn back-to-back 模式占空比: $(2 * PWMnDR[7:0] + 1) / 2 * (PWMnPR[7:0] + 1)$
- 若 PWMnDR 或 PWMnPR 的值为 0, 则占空比为: 0%; 若 $PWMnDR[7:0] \geq PWMnPR[7:0] + 1$, 则占空比为: 100%。

13. 捕捉模块 CCP

13.1 捕捉 CCP 寄存器

捕捉是允许用户定时和控制不同事件的外设。在捕捉模式下，该外设能对事件的持续时间计时。当 CCP 用在捕捉模式下，需要用到定时器 TIMER1。

CCP 控制寄存器 CCPCON(190H)

190H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CCPCON	CCPEN	----	----	CCPIS	CCPES	CCPM2	CCPM1	CCPM0
读写	R/W	----	----		R/W	R/W	R/W	R/W
复位值	0	----	----	0	0	0	0	0

Bit7	CCPEN: 捕捉功能使能位; 0= 禁止捕捉功能; 1= 使能捕捉功能。
Bit6~Bit5	禁用
Bit4	CCPIS: 捕捉时钟源选择位; 0= 时钟源来自 RA2 口输入; 1= 时钟源来自同步比较器输出。
Bit3	CCPES: 捕捉时钟沿选择位; 0= 在时钟下降沿发生捕捉; 1= 在时钟上升沿发生捕捉。
Bit2~Bit0	CCPM<2:0>: 捕捉模式选择位; 000= 每 1 个时钟发生捕捉; 001= 每 2 个时钟发生捕捉; 010= 每 4 个时钟发生捕捉; 011= 每 8 个时钟发生捕捉; 100= 每 16 个时钟发生捕捉; 101= 每 32 个时钟发生捕捉; 110= 每 64 个时钟发生捕捉; 111= 每 128 个时钟发生捕捉。

13.2 捕捉模式

在捕捉模式下，当对应的 CCP 引脚发生事件时，CCPRH:CCPRL 这对寄存器捕捉 TMR1 寄存器的 16 位值。

通过模式选择位 CCPM2:CCPM0 (CCPCON<2:0>) 选择事件类型。当一个捕捉发生时，中断请求标志位 PIR1 寄存器中的 CCPIF 置 1；它必须用软件清零。如果在 CCPRH 和 CCPRL 这对寄存器中的值被读取之前发生另一次捕捉，那么之前捕捉的值将被新捕捉的值覆盖（见图 13-1）。

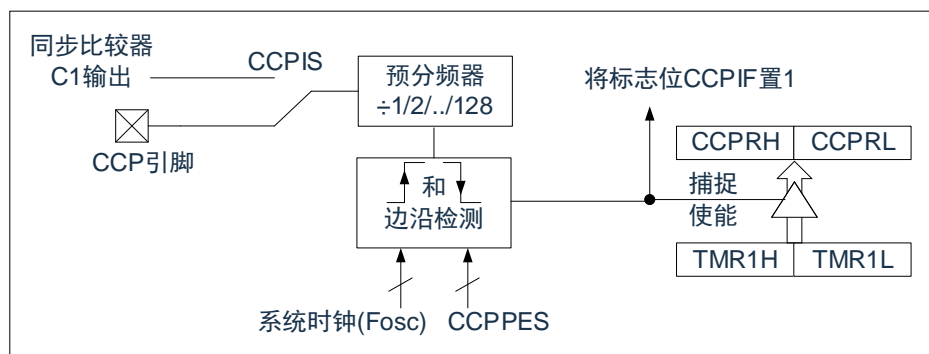


图 13-1: 捕捉模式工作框图

13.2.1 CCP 引脚配置

在捕捉模式下，应通过将对应的 TRIS 控制位置 1 来将相应的 CCP 引脚配置为输入。

注：如果 CCP 引脚被配置为输出，对该端口的写操作可能引发一个捕捉事件。

13.2.2 TIMER1 模式选择

TIMER1 必须运行在定时器模式或同步计数器模式下 CCP 模块才能使用捕捉功能。在异步计数器模式下无法进行捕捉操作。

13.2.3 软件中断

当捕捉模式改变时，可能会产生错误的捕捉中断。用户应该保持 PIE1 寄存器中的 CCPIE 中断允许位清零以避免产生误中断。在操作模式发生任何改变之后也应清零 PIR1 寄存器中的中断标志位 CCPIF。

14. 主控同步串行端口（MSSP）模块

14.1 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）。
 - 全主控模式。
 - 从动模式（支持广播地址呼叫）。
- I²C。

I²C 接口在硬件上支持下列模式：

- 主控模式。
- 多主机模式。
- 从动模式。

14.2 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）——RA3/SDO
- 串行数据输入（SDI）——RA5/SDI/SDA
- 串行时钟（SCK）——RA6/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）——RE1/AN10/SS

14.2.1 SPI 相关寄存器

SSPSTAT: SSP 状态寄存器(193H)

193H	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 1(194H)

194H	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主控模式, 时钟= $F_{Osc}/4$; 0001= SPI主控模式, 时钟= $F_{Osc}/16$; 0010= SPI主控模式, 时钟= $F_{Osc}/64$; 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= I ² C从动模式, 7位地址; 0111= I ² C从动模式, 10位地址; 1000= I ² C主控模式, 时钟= $F_{Osc}/(4 * (SSPADD+1))$; 1001= 禁止装载功能; 1010= 保留; 1011= 保留; 1100= 保留; 1101= 保留; 1110= I ² C从动模式, 7位地址, 并允许起始位和停止位中断; 1111= I ² C从动模式, 10位地址, 并允许起始位和停止位中断。

14.2.2 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）

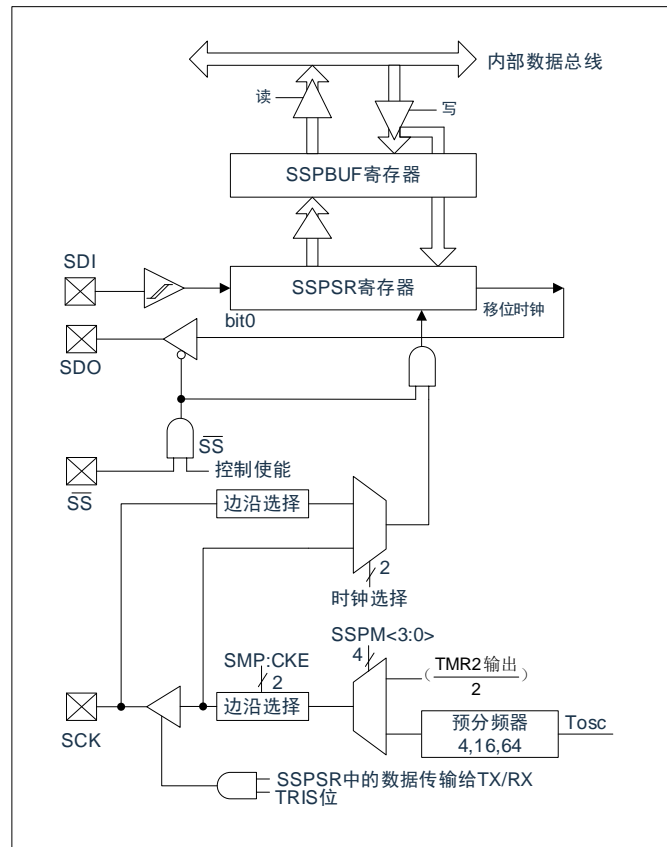


图 14-1：在 SPI 模式下 MSSP 模块的框图

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。

14.2.3 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，要先将 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<3>清零；
- 必须将 SCK（主控模式）的 TRISA<6>位清零；
- 必须将 SCK（从动模式）的 TRISA<6>位置 1；
- 必须将 SS 的 TRISE<1>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。

14.2.4 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 14-2、图 14-4 和图 14-5 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：

- $F_{Osc}/4$ （或 TCY）
- $F_{Osc}/16$ （或 4.TCY）
- $F_{Osc}/64$ （或 16.TCY）
- TIMER2 输出/2

图 14-2 给出了主控模式的波形图。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。

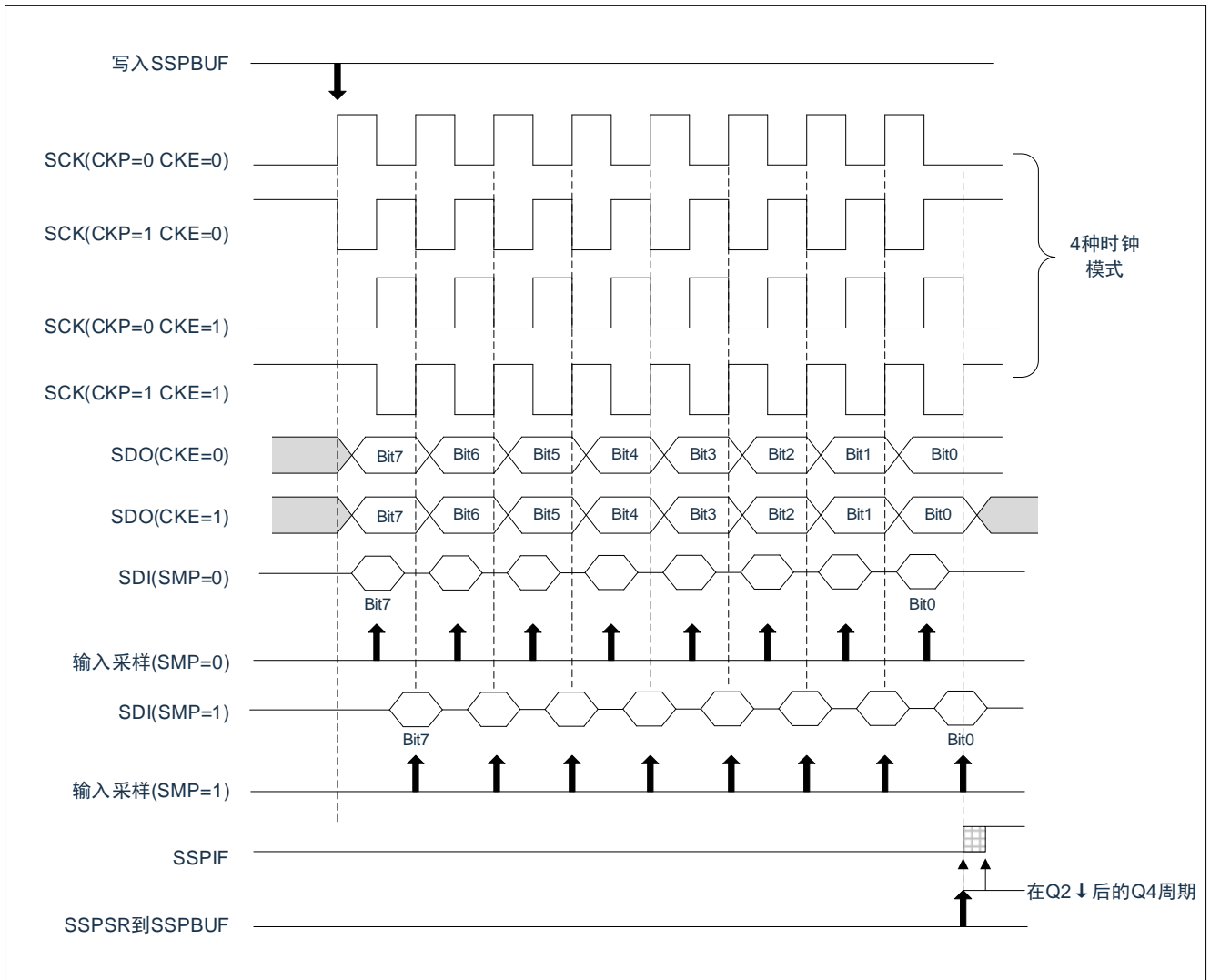


图 14-2: SPI 模式的波形 (主控模式)

14.2.5 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。

14.2.6 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。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总线制通信。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。

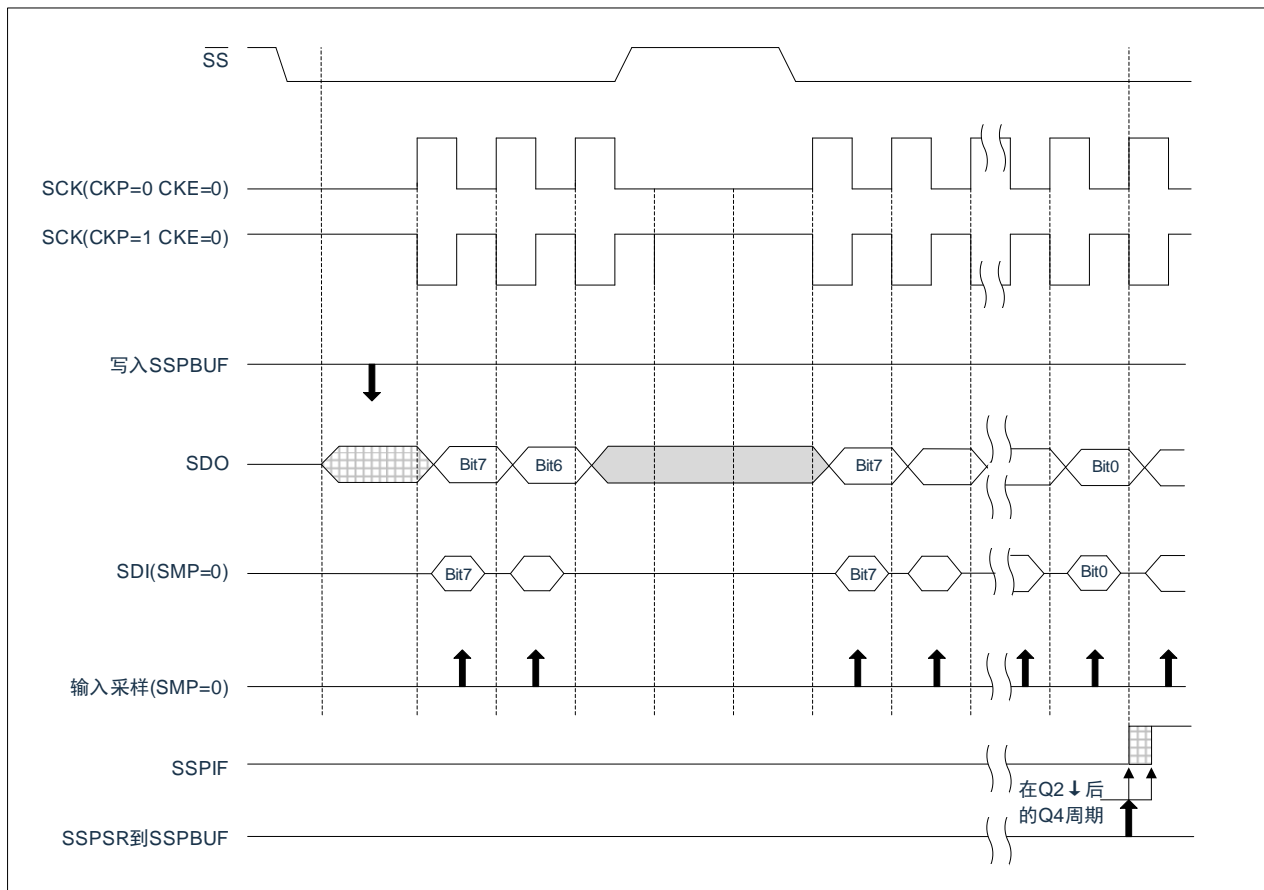


图 14-3: 从动同步波形

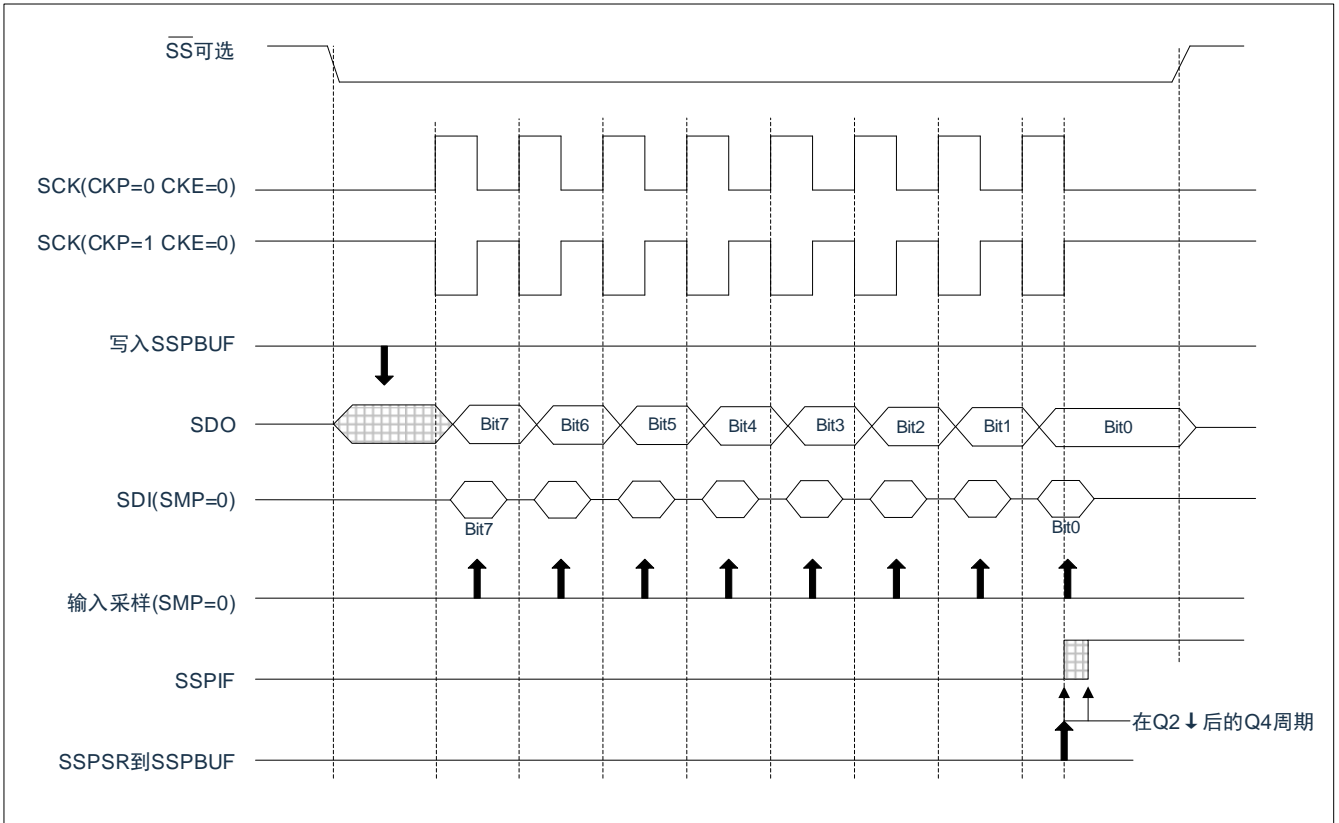


图 14-4: SPI 模式波形 (从动模式, CKE=0)

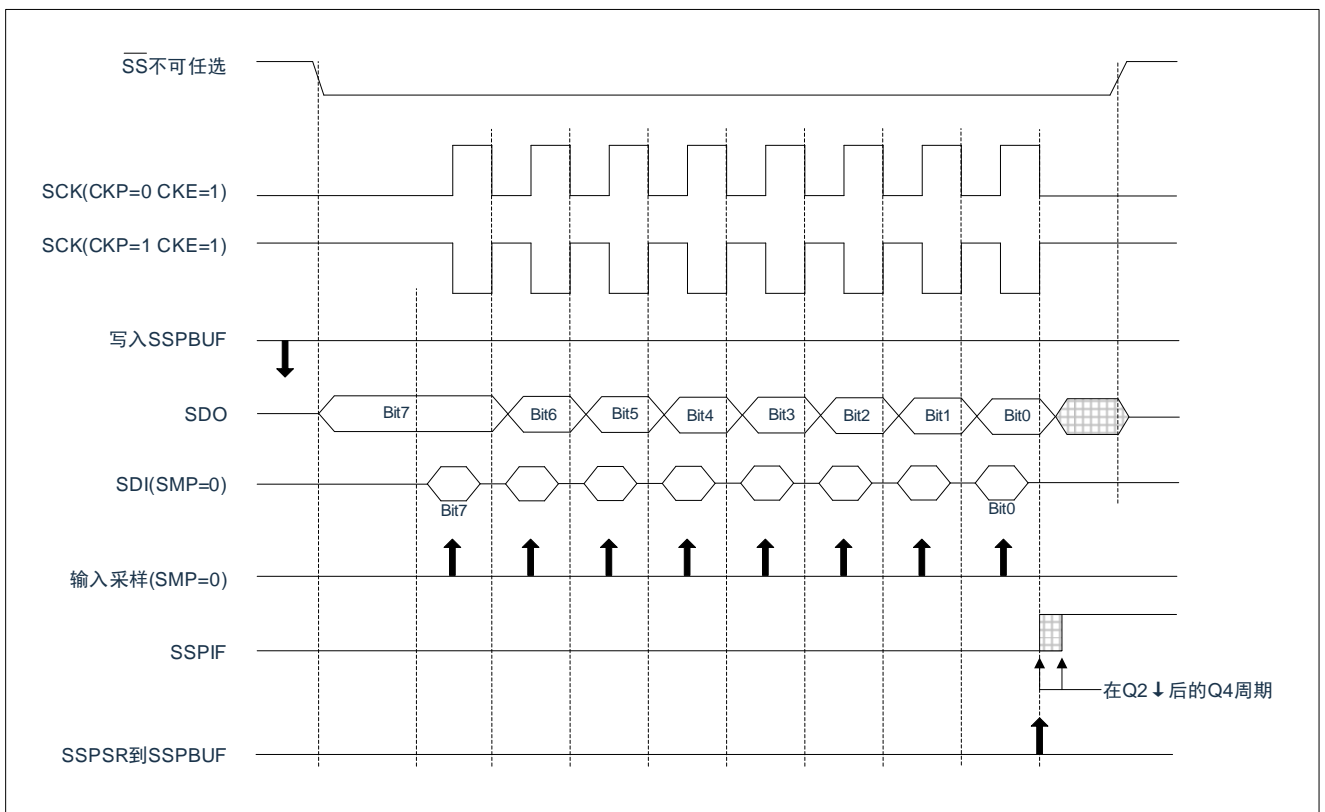


图 14-5: SPI 模式波形 (从动模式, CKE=1)

14.2.7 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。

14.2.8 复位的影响

复位会禁止 MSSP 模块并终止当前的传输。

14.3 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）——RA6/SCK/SCL 引脚，和数据引脚（SDA）——RA5/SDI/SDA 引脚。用户必须通过 TRISA<6:5>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。

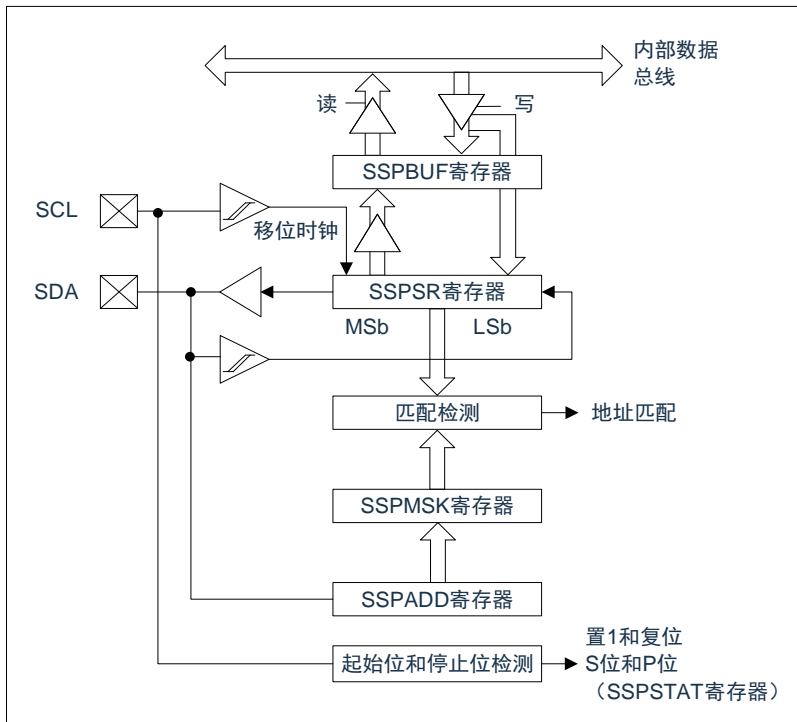


图 14-6: MSSP 框图 (I²C 模式)

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。

14.3.1 相关寄存器说明

SSPSTAT: SSP 状态寄存器(193H)

193H	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	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	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 1(194H)

194H	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	<p>WCOL: 写冲突检测位。</p> <p>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。</p> <p>从动模式:</p> <p>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。</p>
Bit6	<p>SSPOV: 接收溢出指示位。</p> <p>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。</p>
Bit5	<p>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)。</p> <p>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。</p>
Bit4	<p>CKP: 时钟极性选择位。</p> <p>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)。</p> <p>在I²C主控模式下: 在此模式下未使用。</p>
Bit3~Bit0	<p>SSPM<3:0>: 同步串行端口模式选择位。</p> <p>0000= SPI主控模式, 时钟= F_{Osc}/4。 0001= SPI主控模式, 时钟= F_{Osc}/16。 0010= SPI主控模式, 时钟= F_{Osc}/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_{Osc}/(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。</p>

SSPCON2: SSP 控制寄存器 2(195H)

195H	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。

14.3.2 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

14.3.3 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。

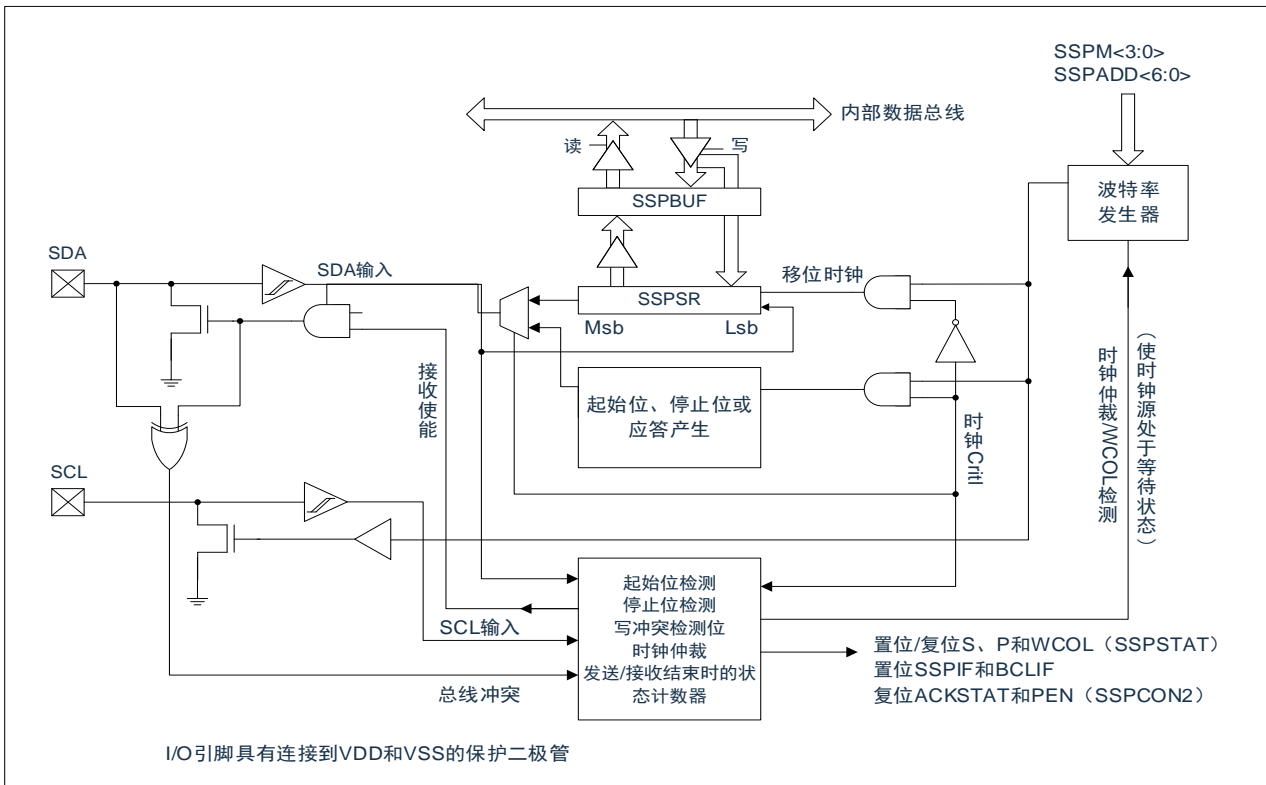


图14-7: MSSP框图 (I²C™主控模式)

注：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。

14.3.3.1 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。

在接收器模式下，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。

14.3.4 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 14-8）。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 14-9）。

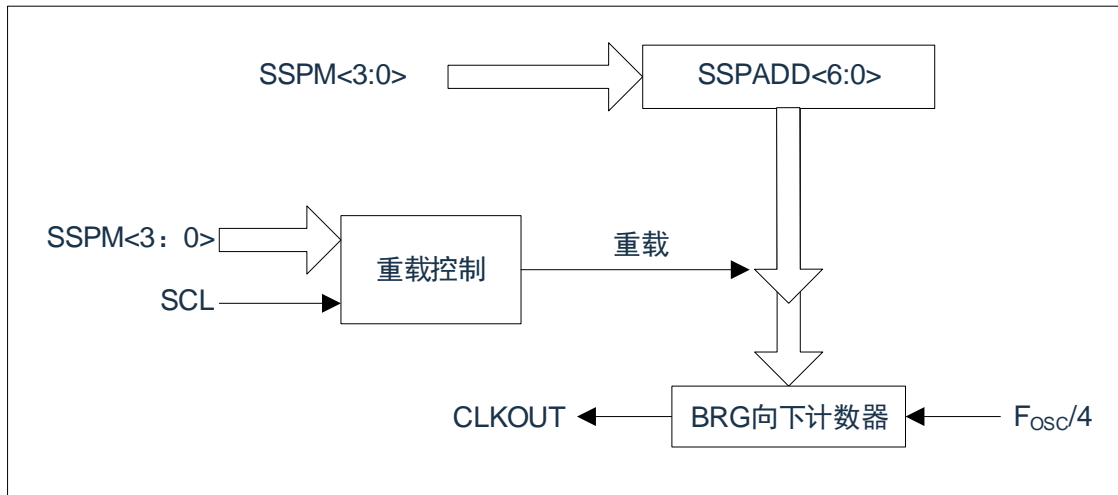


图14-8：波特率发生器框图

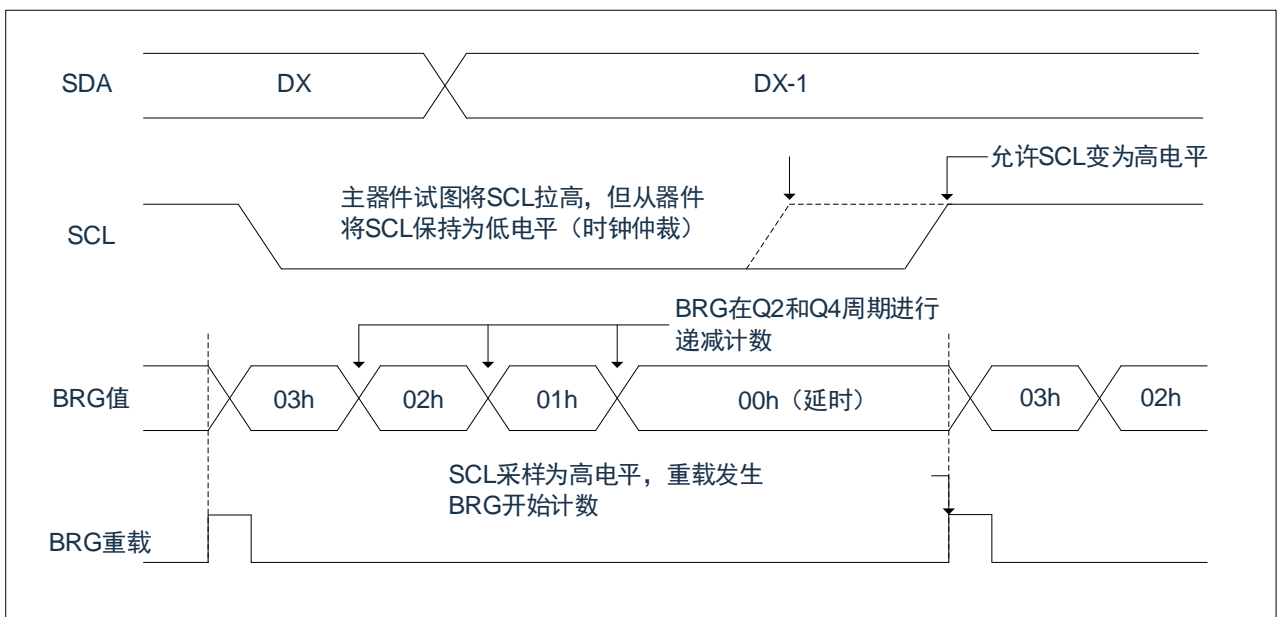


图14-9：带有时钟仲裁的波特率发生器时序

14.3.5 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 (T_{BRG}) 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 T_{BRG} 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。

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。

14.3.5.1 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。

14.3.5.2 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。

14.3.5.3 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。

14.3.6 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。

14.3.6.1 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。

14.3.6.2 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。

14.3.6.3 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)。

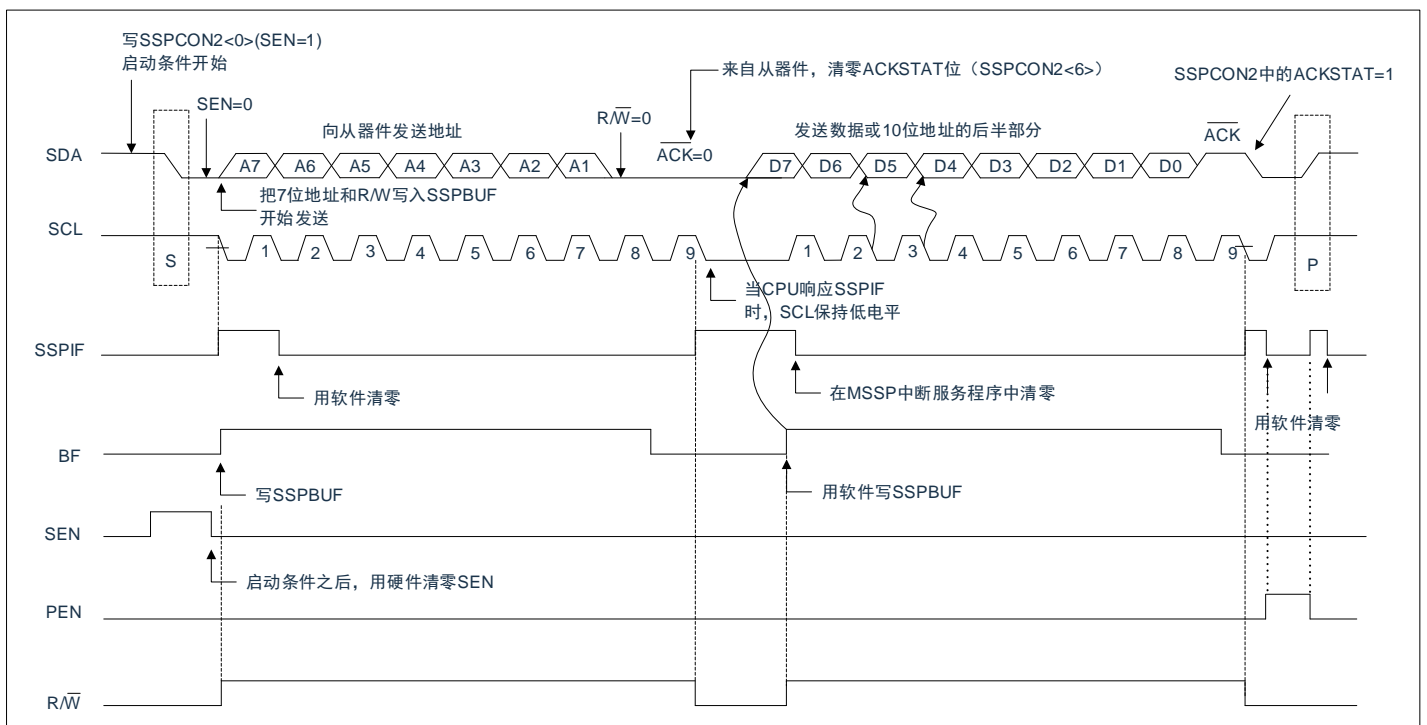
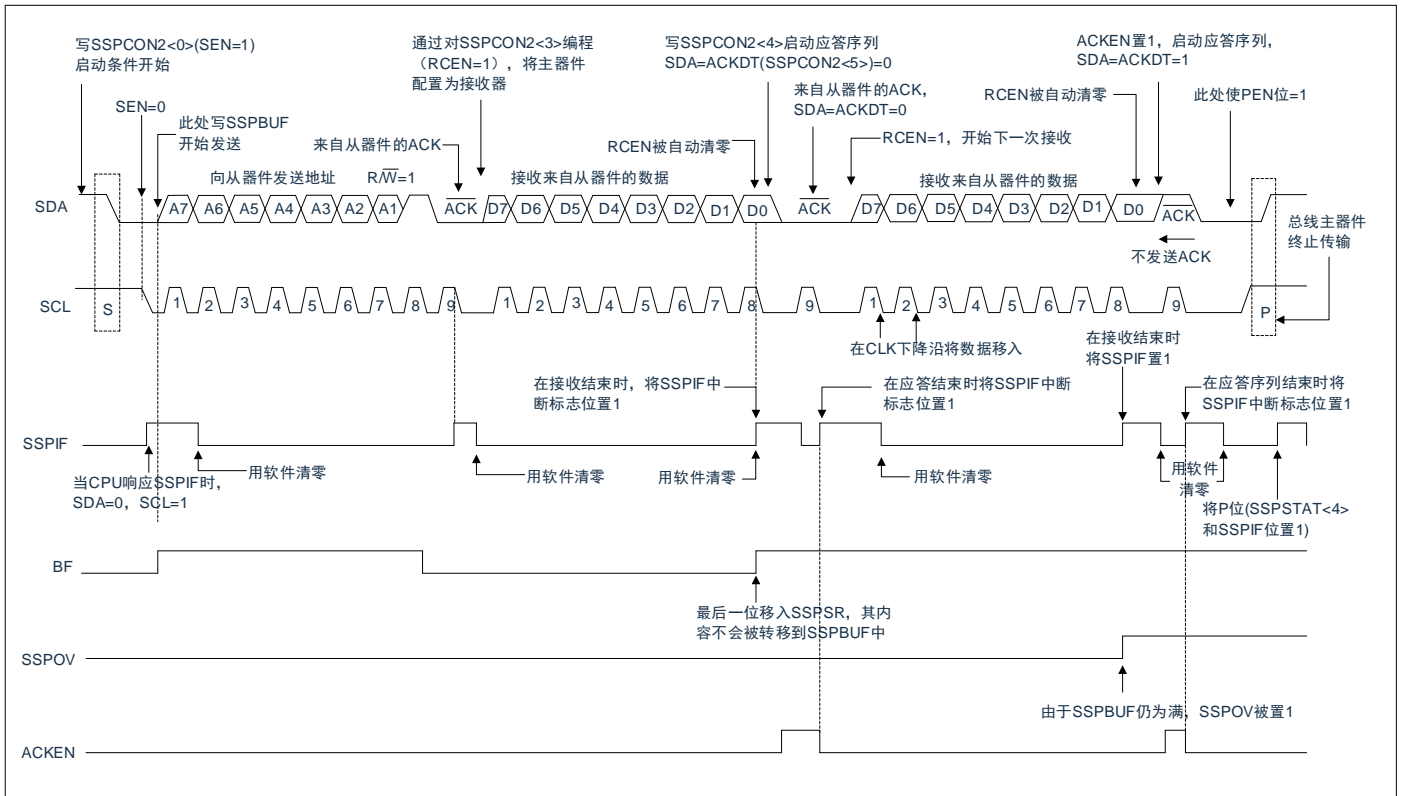


图14-10: I²C™主控模式发送时序 (7位或者10位地址)


 图14-11: I²C™主控模式接收时序 (7位地址)

14.3.7 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 (T_{BRG}) 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 (T_{BRG}) 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。

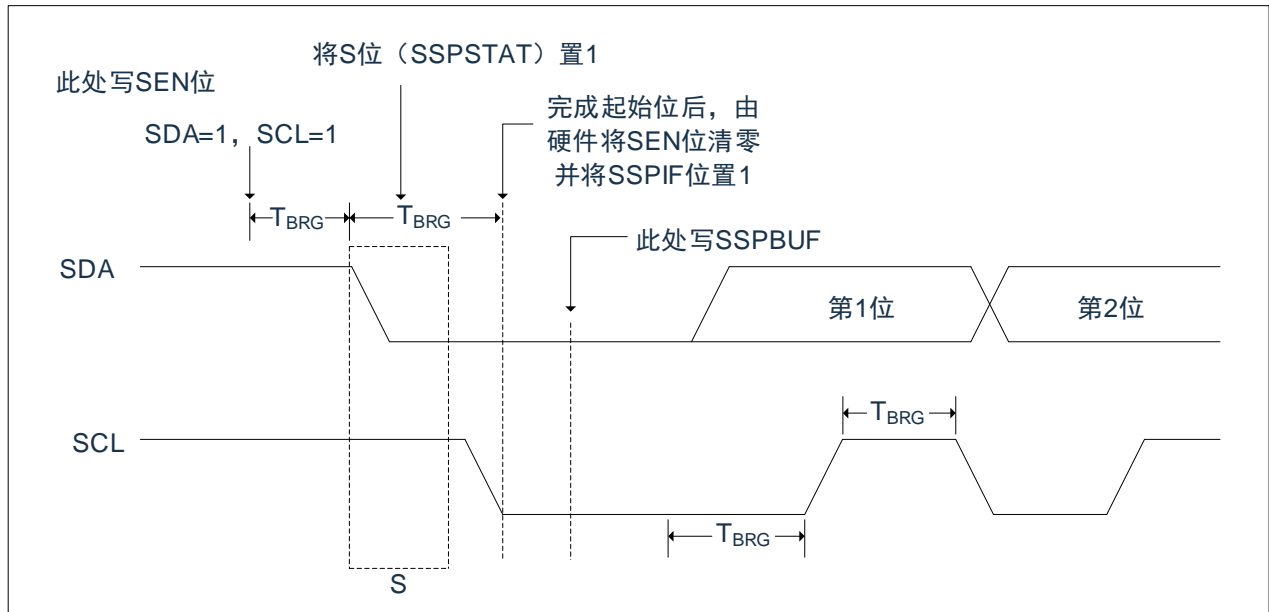


图14-12: 第一个启动位时序

14.3.7.1 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。

14.3.8 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 (T_{BRG}) 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 T_{BRG} 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 T_{BRG} , 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 1 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)。

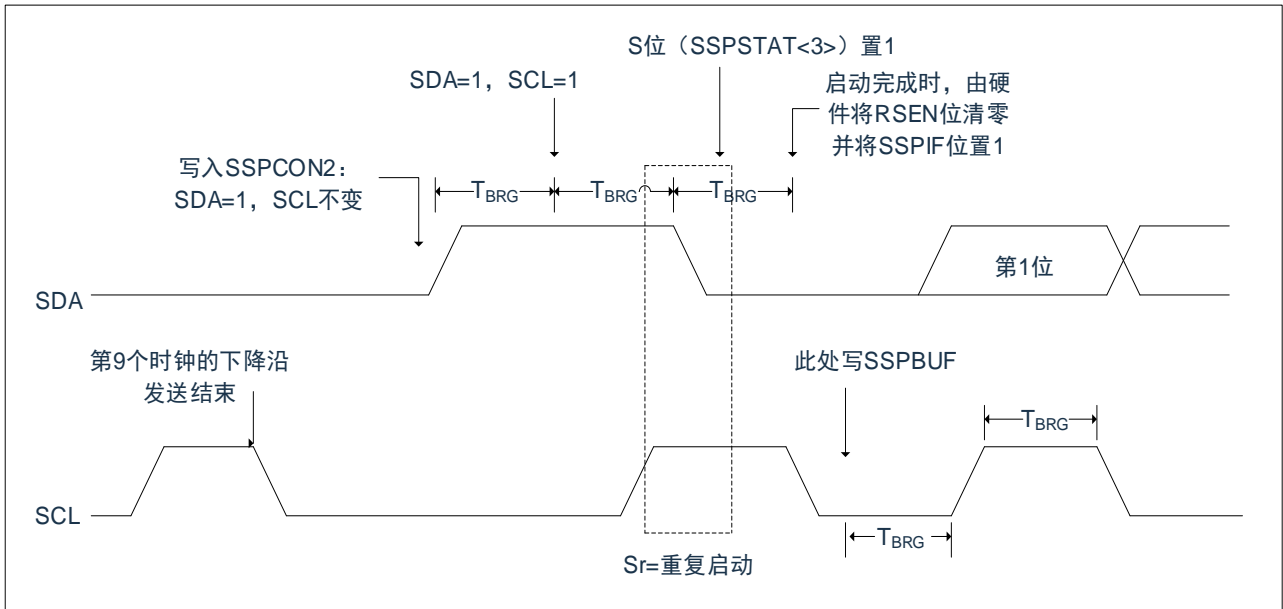


图14-13: 重复启动条件时序波形

14.3.8.1 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。

14.3.9 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 (T_{BRG}) 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 T_{BRG} 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。

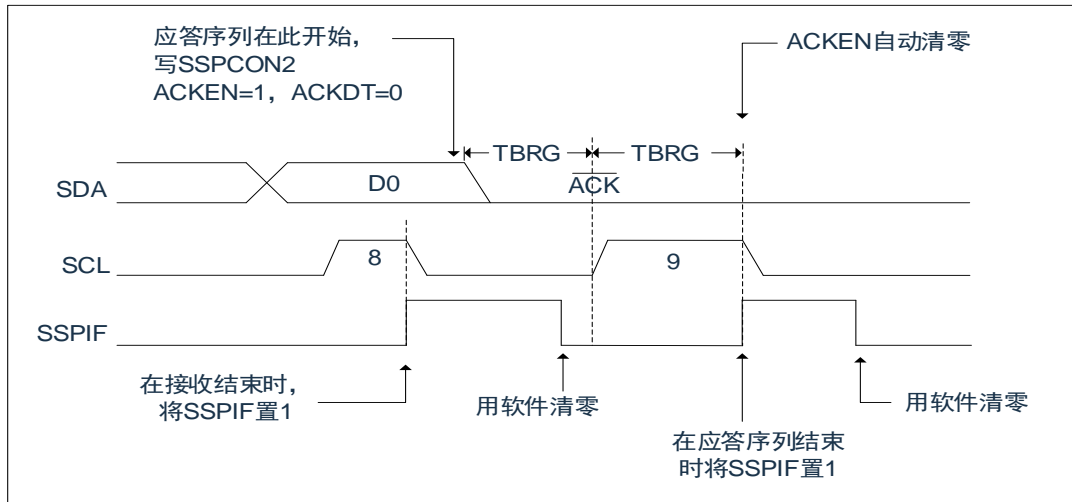


图 14-14: 应答序列时序波形

注: T_{BRG} = 一个波特率发生器周期。

14.3.9.1 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)。

14.3.10 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，主控制器将 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时，SCL 引脚被拉到高电平，且一个 T_{BRG} （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 T_{BRG} 周期后，PEN 位清零且 SSPIF 位置 1。

14.3.10.1 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）。

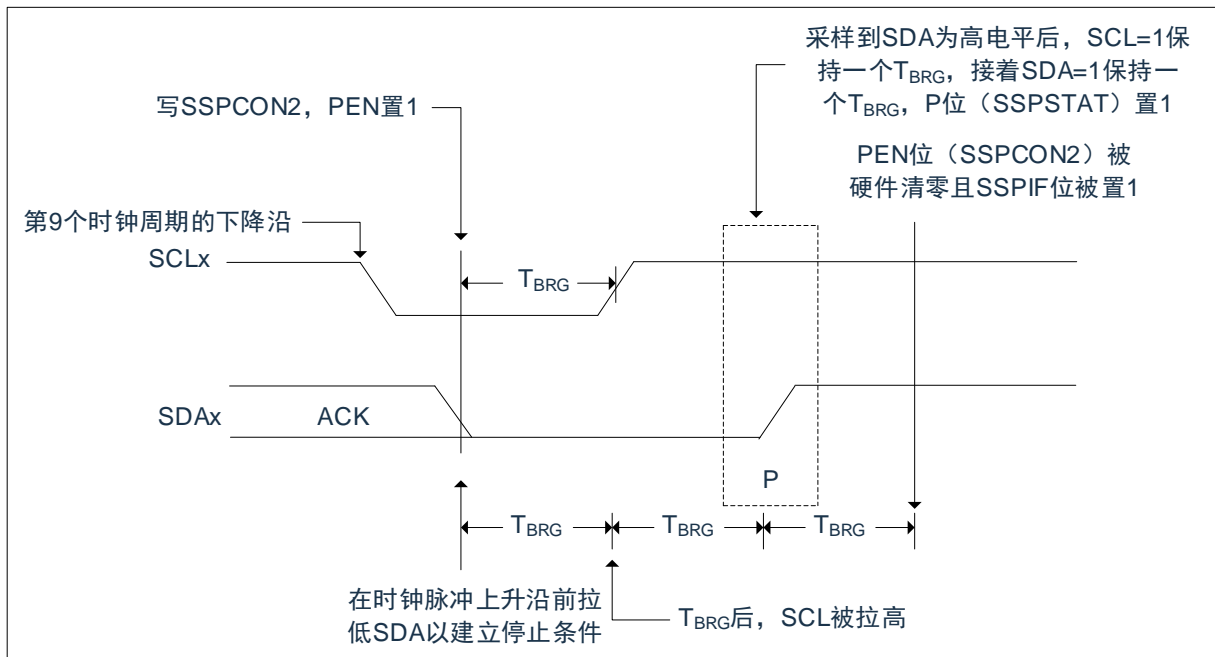


图 14-15: 停止条件接收或发送模式

注: T_{BRG} =一个波特率发生器周期。

14.3.11 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。

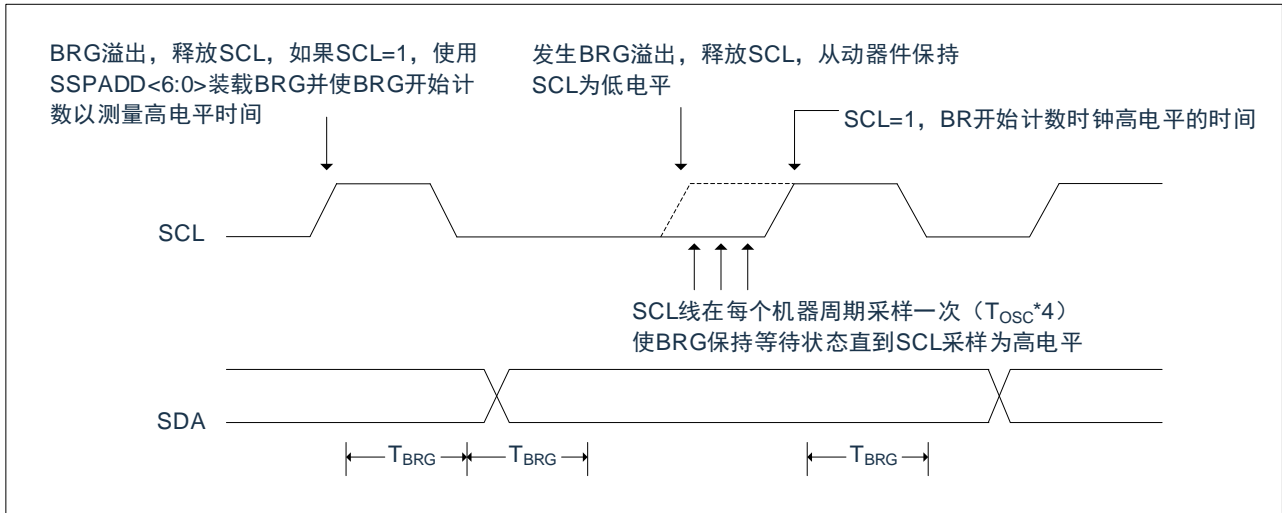


图 14-16: 主控发送模式下的时钟仲裁时序

14.3.12 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

14.3.13 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。

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。

14.3.14 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 (TRISA<6:5>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。

14.3.14.1 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 11110A9 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,其中步骤 7-9 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。

14.3.14.2 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。

14.3.14.3 发送

当接收的地址字节的 R/W 位置 1 并发生地址匹配时，SSPSTAT 寄存器的 R/W 位置 1。接收到的地址被装入 SSPBUF 寄存器。ACK 脉冲在第 9 位上发送，同时 SDA 引脚保持低电平。传输的数据必须装入 SSPBUF 寄存器，同时也被装入 SSPSR 寄存器。随后应通过将 CKP 位（SSPCON 寄存器）置 1 使能 SCL 引脚。在发送另一个时钟脉冲前，主控制器必须监视 SCL 引脚。从动器件可以通过延长时钟，暂停与主控制器的数据传输。8 个数据位在 SCL 输入的下降沿移出。这可确保在 SCL 为高电平期间 SDA 信号是有效的。

每个数据传输字节都会产生一个 MSSP 中断。SSPIF 标志位必须由软件清零，SSPSTAT 寄存器用于确定字节的状态。SSPIF 位在第 9 个时钟脉冲的下降沿被置 1。来自主接收器的 ACK 脉冲将在 SCL 输入第 9 个脉冲的上升沿锁存。如果 SDA 线为高电平（无 ACK），那么表示数据传输已完成。在这种情况下，如果从器件锁存了 ACK，将复位从动逻辑（复位 SSPSTAT 寄存器），同时从器件监视下一个起始位的出现。如果 SDA 线为低电平（ACK），则必须将接下去要发送的数据装入 SSPBUF 寄存器，这也将装载 SSPSR 寄存器。应将 CKP 置 1 使能 RB1/SCK/SCL。

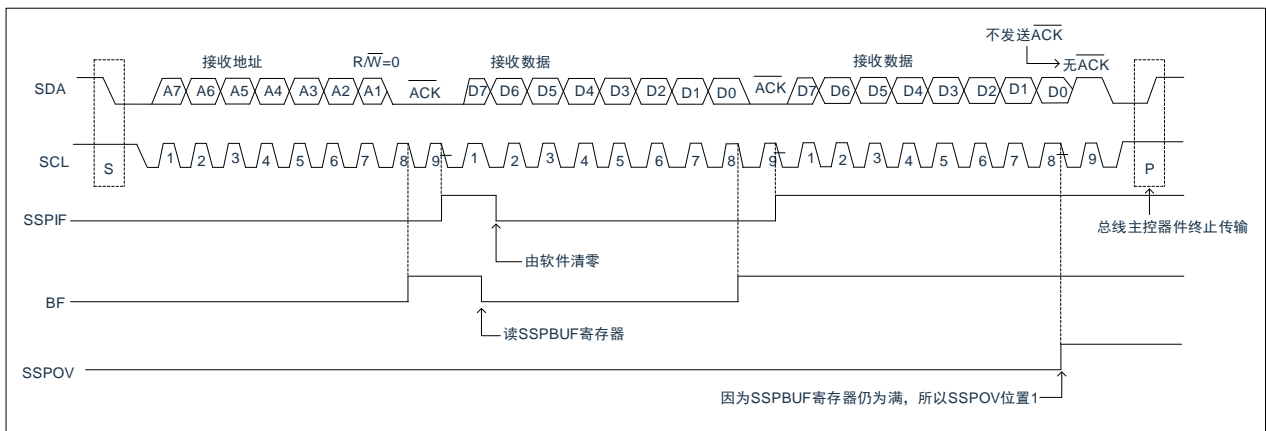


图 14-17: I²C™ 从动模式接收时序 (7 位地址)

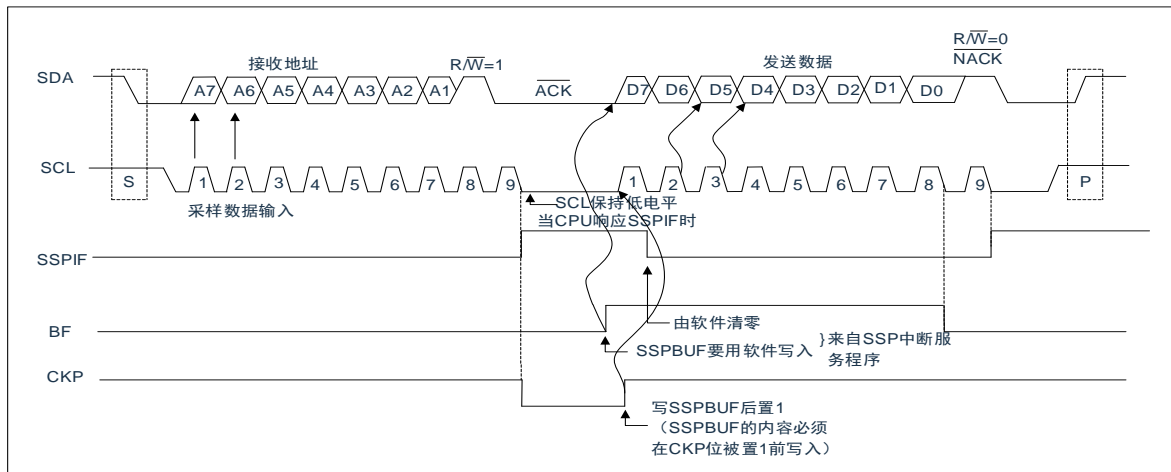


图 14-18: I²C™ 从动模式发送时序 (7 位地址)

14.3.15 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(191H) (1)

191H	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SSPADDSFR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。

14.3.16 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）。

14.3.17 复位的影响

复位会禁止 MSSP 模块并终止当前的传输。

15. 可编程脉冲发生器 PPG

15.1 PPG 工作原理

针对电磁炉方案的应用, CMS89F52x 内部集成了一个可编程脉冲发生器(Programmable Pulse Generator, 以下简称 PPG), 该模块由 1 个 10 位的定时器 PPGTMR, 5 个高精度比较器: 同步比较器 (COMP1)、过压比较器 (COMP2)、过压比较器 1 (COMP3)、电流浪涌比较器 (COMP4)、电流压浪涌比较器 (COMP5), 1 个独立的看门狗计数器 PPGWDT 组成。

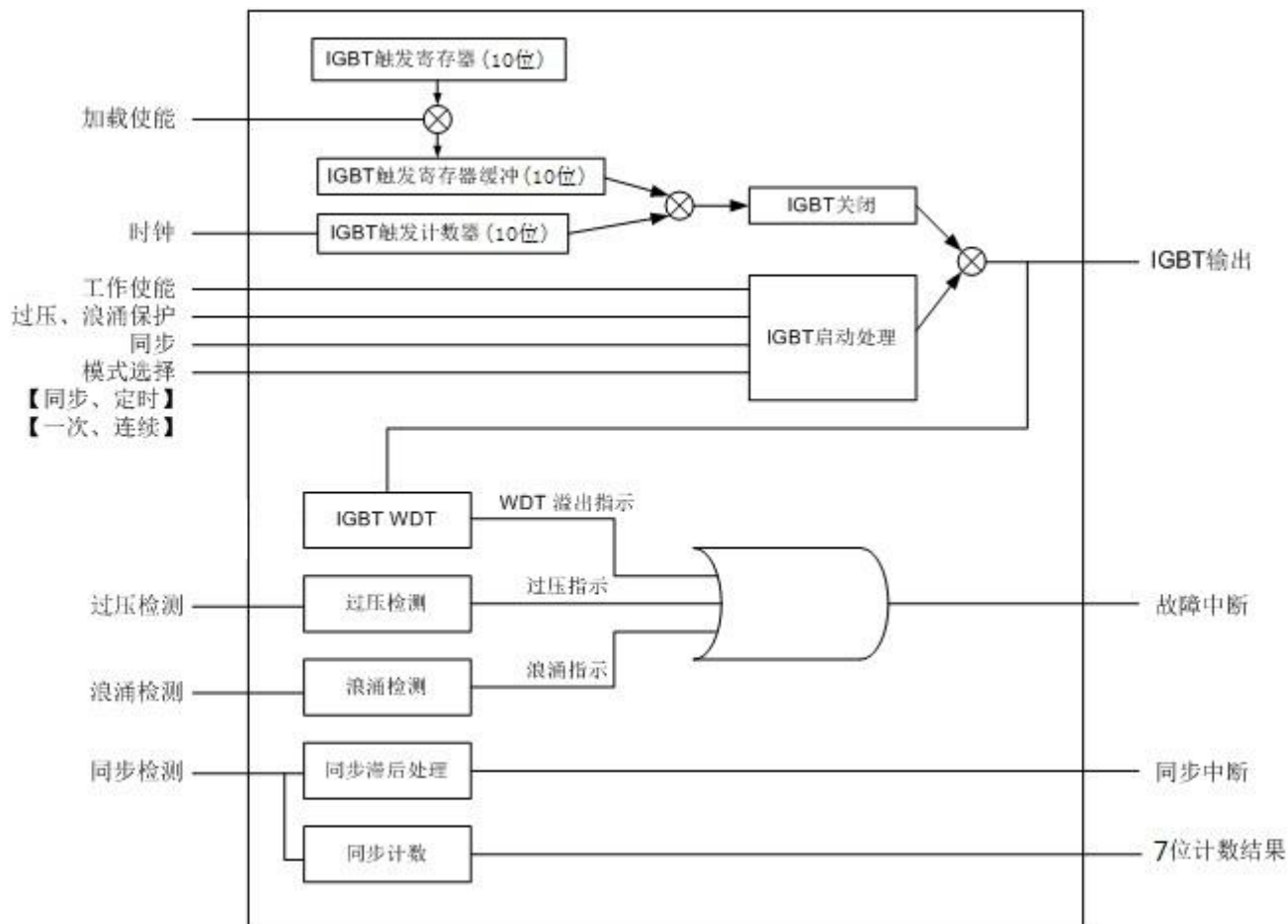


图 15-1: PPG 工作原理

PPG 输出信号是一个只能输出低电平或者高阻态的信号，当 PPG 功能关闭时，它作为高阻态。

PPGTMR 是一个 10 位的定时器，其低 8 位保存在寄存器 PPGTMRL(14H)中，高 2 位保存在寄存器 PPGTMRH(15H)中。当 PPG 输出关断时，内部 10 位计数器清零，当 PPG 输出打开时，计数器开始计数，每个振荡周期自动加 1，当计数器加至等于 PPGTMR 值时，PPG 输出自动关断，计数器自动清零。

PPGTMR 低 8 位寄存器 PPGTMRL(14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PPGTMRL	PPGTMR低8位							
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

PPGTMR 高 2 位寄存器 PPGTMRH(15H)

15H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PPGTMRH	----	----	----	----	----	----	PPGTMR高2位	
R/W	----	----	----	----	----	----	R/W	R/W
复位值	----	----	----	----	----	----	0	0

15.2 与 PPG 相关的引脚

有 6 个管脚与 PPG 相关，如下表：

管脚名称	IO 类型	管脚说明
C1N	I	比较器 1 负端输入
C1P	I	比较器 1 正端输入
C2N	I	比较器 2 负端输入
C3N	I	比较器 3 负端输入
C4N	I	比较器 4 负端输入
C5N	I	比较器 5 负端输入
PPG_OUT	O	PPG 输出

15.3 PPG 工作模式

CMS89F52x 的 PPG 模块具有 2 种工作模式，分别是：

- 单次输出模式；
- 同步输出模式；

与 PPG 状态相关的控制寄存器如下：

PPG 控制寄存器 PPGCON(17H)

17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PPGCON	DETC5F	DETC4F	----	RELOAD_EN	DETC5EN	DETC4EN	PPGMD	PPG_ON
R/W	R/W	R/W	----	R/W	R/W	R/W	R/W	R/W
复位值	1	1	----	0	0	0	0	0

Bit7	DETC5F: 比较器 5 状态位 (PPG 状态位);
	0: 清零比较器 5 的 0->1 翻转标志, (若 DETC5EN=1, 则 PPG 重新打开);
	1: 有比较器 5 的 0->1 翻转, 写 1 无效, (若 DETC5EN=1, 则 PPG 关闭)。
Bit6	DETC4F: 比较器 4 状态位 (PPG 状态位);
	0: 清零比较器 4 的 1->0 翻转标志, (若 DETC4EN=1, 则 PPG 重新打开);
	1: 有比较器 4 的 1->0 翻转, 写 1 无效, (若 DETC4EN=1, 则 PPG 关闭)。
Bit5	禁用
Bit4	RELOAD_EN: PPG TMR 加载使能;
	0: 允许加载 (自动降功率模式时, 同时允许 PPG TMR 自-1);
	1: 禁止加载 (自动降功率模式时, 同时禁止 PPG TMR 自-1)。
Bit3	DETC5EN: 比较器 5 关闭 PPG 使能位;
	0: 禁止;
	1: 使能。
Bit2	DETC4EN: 比较器 4 关闭 PPG 使能位;
	0: 禁止;
	1: 使能。
Bit1	PPGMD: PPG 输出模式;
	0: 根据比较器 1 的同步输出;
	1: 单次输出。
Bit0	PPG_ON: PPG 输出使能位;
	0: 禁止;
	1: 使能。

15.3.1 单次输出模式

当系统寄存器 PPGCON 的第 1 位被置“1”时，PPG 将处在单次输出的工作模式下。这时将 PPG 使能位（PPGCON.0）置 1，则 PPG 在输出 1 个 PPGTMR 时间的低电平后，将重新变为高阻态，并且 PPG 使能位自动清零，PPG 停止工作。单次输出模式一般情况下用来检测电磁炉有没有放锅。

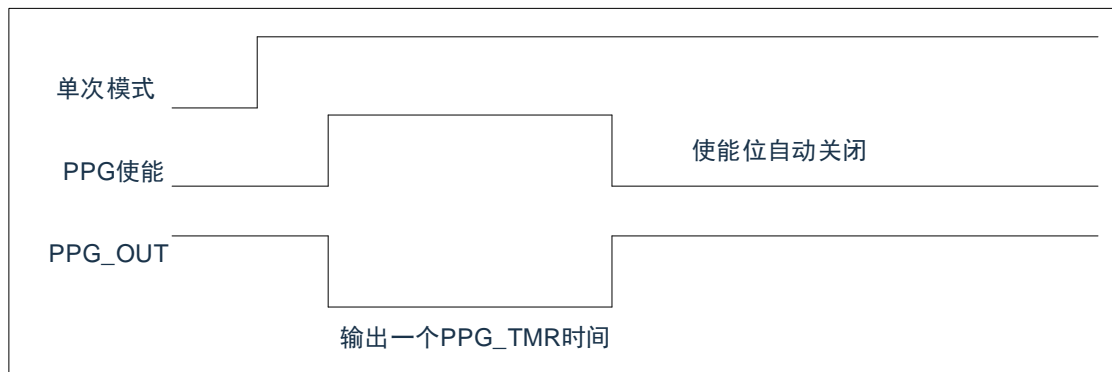


图 15-2: PPG 单次输出模式时序

15.3.2 同步输出模式

同步输出指的是 PPG 输出跟比较器 1 翻转同步。当系统寄存器 PPGCON 的第 1 位被置“0”时，PPG 模块将处在同步输出模式，在该模式下，使能信号从“0”到“1”变化时，PPG 输出一个 PPGTMR 时间的低电平后关闭输出，以后每次根据比较器 1 输出从“1”至“0”翻转的时候自动连续输出。

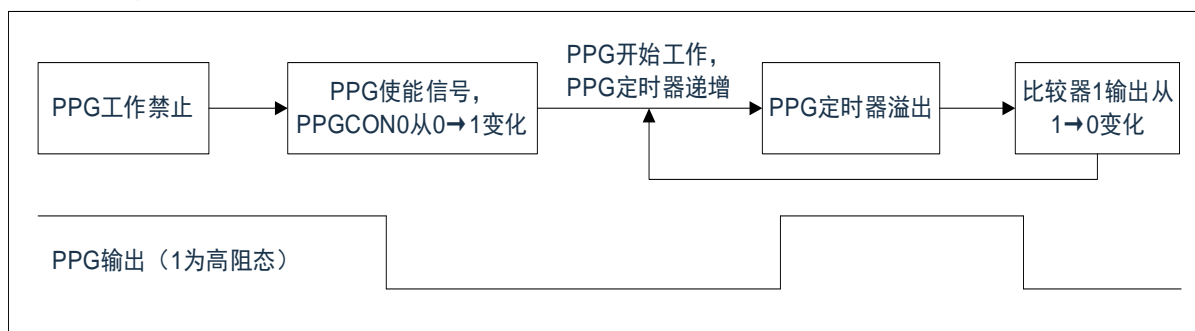


图 15-3: PPG 同步输出模式

15.4 比较器

PPG 模块共有 5 个比较器：同步比较器（COMP1）、过压比较器（COMP2）、过压比较器 1（COMP3）、电流浪涌比较器（COMP4）、电压浪涌比较器（COMP5）。

15.4.1 同步比较器 COMP1

同步比较器的作用是给 PPG 提供同步信号。当 PPG 工作在同步模式下，PPG_OUT 只有在比较器的输出从“1”至“0”变化时才能输出低电平，持续一个 PPGTMR 周期后关闭，等待下次比较器的翻转，重新输出低电平。

可以设置同步比较器翻转后再延时一段时间再导通 PPG，该时间最短为 $0 \cdot T_{sys}$ ，最长为 $64 \cdot T_{sys}$ ，该时间由 PPGDLY 的低 4 位控制。

比较器 1 控制寄存器 CM1CON (97H)

97H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM1CON	CM1EN	CM1COFM	CM1CEN	CM1CLR	CM1NSL	----	----	----
R/W	R/W	R/W	R/W	R/W	R/W	----	----	----
复位值	0	0	0	0	0	----	----	----

- Bit7 CM1EN: 比较器 1 使能位；
 0: 禁止，比较器不工作，输出 0；
 1: 使能，COMP1+、COMP1-为比较器输入端。
- Bit6 CM1COFM: 比较器 1 调节模式选择；
 0: 正常工作模式；
 1: 调节模式。
- Bit5 CM1CEN: 比较器 1 翻转计数使能；
 0: 禁止计数；
 1: 使能计数。
- Bit4 CM1CLR: 比较器 1 计数清零；
 0: 清零；
 1: 正常计数。
- Bit3 CM1NSL: 比较器 1 负端内部接地选择，仅在调节模式生效；
 0: COMP1-接 I/O 口（若 CM1EN=1，使能 C1N 通道口为比较器 1 负端输入）；
 1: COMP1-接 GND（C1N 通道口为普通 I/O 口）。
- Bit2~Bit0 禁用

PPG 延时时间控制寄存器 PPGDLY (16H)

16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PPGDLY	----	----	----	----	PPGDLY			
R/W	----	----	----	----	R/W	R/W	R/W	R/W
复位值	----	----	----	----	0	0	0	0

Bit7~Bit4 禁用
 Bit3~Bit0 PPGDLY: PPG 延时输出;
 0000: 不延时;
 0001: 4-5* T_{sys} ;
 0010: 8-9* T_{sys} ;

 1111: 64-65* T_{sys} 。

同步比较器具有翻转计数功能，可以记录其输出翻转的次数。要使该功能有效，功能必须将 CM1CON 的第 5 位置“1”。PPG 计数功能开启时，当同步比较器的输出从“1”至“0”变化时，计数器自动加 1，最高加至 128。计数结果保存至 CM1CNT 寄存器，其低 7 位表示计数值，最高位表示是否计数溢出，当计数超过 128 时，最高位为 1，计数停止。可以通过对 CM1CON 的第 4 位置“0”来清零计数器。

比较器 1 翻转计数寄存器 CM1CNT(93H)

93H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM1CNT	CM1OF	CM1COUNT[6:0]						
R/W	R	R	R	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0

Bit7 CM1OF: 比较器 1 翻转计数溢出标志;
 0: 没有溢出;
 1: 有溢出。
 Bit6~Bit0 CM1COUNT[6:0]: 比较器 1 的 1->0 翻转计数器，仅读。

15.4.2 过压比较器 COMP2 跟浪涌比较器 COMP4/COMP5

过压比较器跟浪涌比较器的作用都是用来限制 PPG 的输出，从而保护 IGBT。过压比较器的负端从 RB2 口输入，电压浪涌比较器的负端从 RB0 口输入，电流浪涌比较器的负端从 RB4 口输入。它们的正端都在芯片内部，通过软件调节电阻分压。

当过压比较器的负端电压从低于正端电压到高于正端电压的变化，我们称为 IGBT 的过压。当过压次数满足软件设定的次数时，PPG 将禁止输出或者降低输出时间（通过软件设置），并且产生中断标志位，必须将该中断标志位清零，PPG 才能恢复正常。

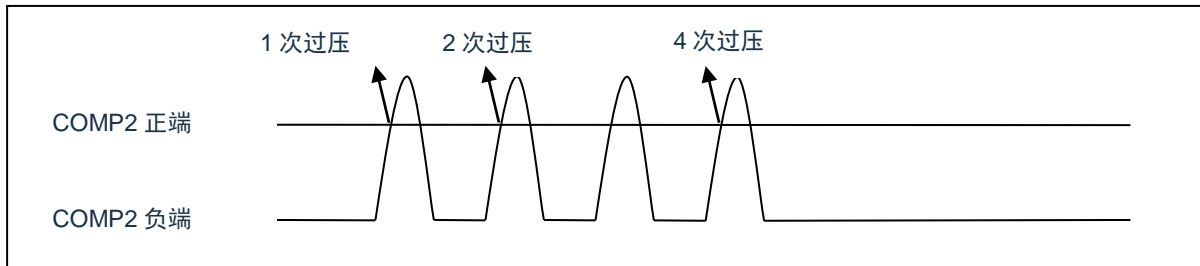


图 15-4：过压比较器 COMP2

当浪涌比较器满足设置的有效电平，并且时间上满足软件设定的时间时，PPG 将禁止输出，并且产生中断标志位，必须将该中断标志位清零，PPG 才能恢复正常。浪涌比较器的正端电压可以选择接地。

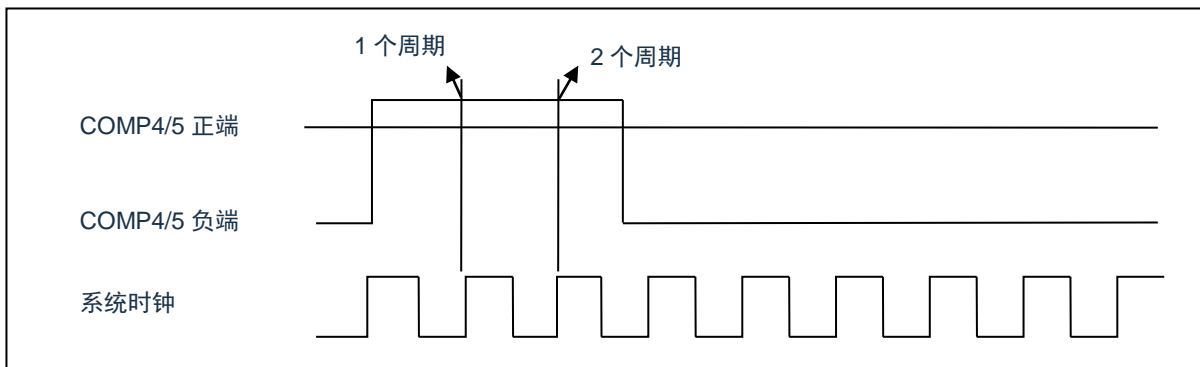


图 15-5：输出低电平有效的示意图

与 COMP2, COMP4, COMP5 相关寄存器:

比较器 2 控制寄存器 CM2CON (98H)

98H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM2CON	CM2EN	CM2COFM	CM2DBSEL		CM2PVSL			
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 CM2EN: 比较器 2 使能位;
 0: 禁止, 比较器 2 不工作, 输出 0;
 1: 使能, 比较器 2 计数器的溢出将影响 PPG 或进中断。
- Bit6 CM2COFM: 比较器 2 调节模式选择;
 0: 正常工作模式;
 1: 调节模式。
- Bit5~Bit4 CM2DBSEL: 比较器 2 滤波时间选择;
 00: $\leq 1T_{sys}$ (能滤掉的脉冲宽度);
 01: $\leq 4T_{sys}$;
 10: $\leq 8T_{sys}$;
 11: $\leq 16T_{sys}$ 。
- Bit3~Bit0 CM2PVSL: 比较器 2 内部正端电压选择。
 0000-1111: 0.4VDD-0.775VDD (共 16 档, 每档 0.025VDD)。

比较器 2 控制寄存器 1CM2CON1 (99H)

99H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM2CON1	ATPEN	----	----	----	CM2COF	CM2CNT		
R/W	R/W	----	----	----	R/W	R/W	R/W	R/W
复位值	0	----	----	----	1	0	0	0

- Bit7 ATPEN: 比较器 2 计数器溢出自动降低 PPG_TMR 使能位;
 0: 禁止, (比较器 2 计数器清零后需要清 CM2COF 才能开始计数);
 1: 使能, 每次检测到溢出时, PPT_TMR 的值自动减 1
 (比较器 2 计数器清零后自动重新计数)。
- Bit6~Bit4 禁用
- Bit3 CM2COF: 比较器 2 计数器溢出标志位, 可通过软件清零;
 0: 比较器 2 计数器无溢出, 写 0 清掉, (若 ATPEN=0, 比较器 2 计数器使能计数);
 1: 比较器 2 计数器有溢出, 写 1 无效, (若 ATPEN=0, 则比较器 2 计数器保持清零状态)。
- Bit2~Bit0 CM2COS[2:0]: 比较器 2 计数器溢出需要的脉冲个数选择位(比较器 2 输出 1->0 跳变为计数触发输入);
 000: 1 次就溢出, 或产生中断, 且计数器清零;
 001: 2 次;
 010: 4 次;
 ...
 111: 128 次。

比较器 4 控制寄存器 CM4CON (电流浪涌) (9CH)

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM4CON	CM4EN	CM4COFM	CM4DBSEL		CM4PVSL			
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 CM4EN: 比较器 4 使能位 (输出高有效);
 0: 禁止, 比较器 4 不工作, 输出 0;
 1: 使能, 比较器 4 的 0->1 翻转将影响 PPG 或进中断。
- Bit6 CM4COFM: 比较器 4 调节模式选择;
 0: 正常工作模式;
 1: 调节模式。
- Bit5~Bit4 CM4DBSEL: 比较器 4 滤波时间选择;
 00: $\leq 1T_{sys}$ (能滤掉的脉冲宽度);
 01: $\leq 4T_{sys}$;
 10: $\leq 8T_{sys}$;
 11: $\leq 16T_{sys}$ 。
- Bit3~Bit0 CM4PVSL: 比较器 4 内部正端电压选择;
 0000: 选择内部接 GND;
 0001-1111: 0.050VDD-0.400VDD (共 16 档, 每档 0.025VDD)。

比较器 5 控制寄存器 CM5CON (电压浪涌) (9DH)

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM5CON	CM5EN	CM5COFM	CM5DBSEL		CM5PVSL			
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 CM5EN: 比较器 5 使能位 (输出高有效);
 0: 禁止, 比较器 5 不工作, 输出 0;
 1: 使能, 比较器 5 的 0->1 翻转将影响 PPG 或进中断。
- Bit6 CM5COFM: 比较器 5 调节模式选择;
 0: 正常工作模式;
 1: 调节模式。
- Bit5~Bit4 CM5DBSEL: 比较器 5 滤波时间选择;
 00: $\leq 1T_{sys}$ (能滤掉的脉冲宽度);
 01: $\leq 4T_{sys}$;
 10: $\leq 8T_{sys}$;
 11: $\leq 16T_{sys}$ 。
- Bit3~Bit0 CM5PVSL: 比较器 5 内部正端电压选择;
 0000-1111: 0.4VDD-0.775VDD (共 16 档, 每档 0.025VDD)。

15.4.3 过压比较器 1- COMP3

过压比较器 1 负端从 RB3 口输入，正端都在芯片内部，可通过软件调节电阻分压。

与 COMP3 相关寄存器：

比较器 3 控制寄存器 CM3CON(9AH)

9AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM3CON	CM3EN	CM3COFM	CM3DBSEL		CM3PVSL			
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 CM3EN: 比较器 3 使能位（输出高有效）；
 0: 禁止，比较器 3 不工作，输出 0；
 1: 使能，比较器 3 计数器溢出将影响 PPG 或进中断。
- Bit6 CM3COFM: 比较器 3 调节模式选择；
 0: 正常工作模式；
 1: 调节模式。
- Bit5~Bit4 CM3DBSEL 比较器 3 滤波时间选择；
 00: $\leq 1T_{sys}$ （能滤掉的脉冲宽度）；
 01: $\leq 4T_{sys}$ ；
 10: $\leq 8T_{sys}$ ；
 11: $\leq 16T_{sys}$ 。
- Bit3~Bit0 CM3PVSL: 比较器 3 内部正端电压选择；
 00001-11111: $0.4V_{DD}$ - $0.775V_{DD}$ （共 16 档，每档 $0.025V_{DD}$ ）。

比较器 3 控制寄存器 CM3CON1(9BH)

9BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM3CON1	CM3M1	CM3M0	----	CM3CIS	CM3COF	CM3COS[2:0]		
R/W	R/W	R/W	----	R/W	R/W	R/W	R/W	R/W
复位值	0	0	----	0	1	0	0	0

- Bit7~Bit6 CM3M1-CM3M0: 比较器 3 功能选择;
- 00: 比较器 3 不影响 PPG;
 - 01: 比较器 3 关闭 PPG (输出低有效);
 - 10: 比较器 3 降低 PPG_TMR (与比较器 2 共用降低 PPG_TMR 模块);
 - 11: 比较器 3 不影响 PPG。
- Bit5 禁用
- Bit4 CM3CIS: 比较器 3 计数器触发边沿选择位;
- 0: 比较器输出 1->0 跳变为计数触发输入;
 - 1: 比较器输出 0->1 跳变为计数触发输入。
- Bit3 CM3COF: 比较器 3 计数器溢出标志位, 可通过软件清零;
- 0: 比较器 3 计数器无溢出, 写 0 清掉, (若 ATPEN=0, 比较器 3 计数器开始计数);
 - 1: 比较器 3 计数器有溢出, 写 1 无效, (若 ATPEN=0, 则比较器 3 计数器保持清零状态)。
- Bit2~Bit0 CM3COS[2:0]: 比较器 3 计数器溢出需要的脉冲个数选择位
(计数器触发输入边沿由 CM3CIS 决定);
- 000: 1 次就溢出, 或产生中断, 且计数器清零;
 - 001: 2 次;
 - 010: 4 次;
 - ...
 - 111: 128 次。

15.4.4 比较器调零

由于生产工艺的误差，在实际使用中，芯片可能存在较大的比较器失调电压。为了解决这一问题 CMS69F52x 内部集成了比较器调零功能。相关 RAM 如下：

比较器 1 调零寄存器 CM1ADJ(113H)

113H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM1ADJ	CM1OUT	CM1CRS	CM1ADJ[5:0]					
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	1	0	0	0	0	0

- Bit7 CM1OUT: 比较器 1 输出，只读，写操作不产生影响；
 0: 比较器输出 0；
 1: 比较器输出 1。
- Bit6 CM1CRS: 调节模式输入端选择；
 0: 负端输入；
 1: 正端输入。
- Bit5~Bit0 CM1ADJ[5:0]: 比较器 1 失调电压调节；
 000000: 调节负端至最小（一般情况下失调电压为正）；
 111111: 调节正端至最小（一般情况下失调电压为负）。

比较器 2 调零寄存器 CM2ADJ(114H)

114H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM2ADJ	CM2OUT	CM2CRS	CM2ADJ[5:0]					
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	1	0	0	0	0	0

- Bit7 CM2OUT: 比较器 2 输出，只读，写操作不产生影响；
 0: 比较器输出 0；
 1: 比较器输出 1。
- Bit6 CM2CRS: 调节模式输入端选择；
 0: 负端输入；
 1: 正端输入。
- Bit5~Bit0 CM2ADJ[5:0]: 比较器 2 失调电压调节；
 000000: 调节负端至最小（一般情况下失调电压为正）；
 111111: 调节正端至最小（一般情况下失调电压为负）。

比较器 3 调零寄存器 CM3ADJ(115H)

115H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM3ADJ	CM3OUT	CM3CRS	CM3ADJ[5:0]					
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	1	0	0	0	0	0

- Bit7 CM3OUT: 比较器 3 输出, 只读, 写操作不产生影响;
 0: 比较器输出 0;
 1: 比较器输出 1。
- Bit6 CM3CRS: 调节模式输入端选择;
 0: 负端输入;
 1: 正端输入。
- Bit5~Bit0 CM3ADJ[5:0]: 比较器 3 失调电压调节;
 000000: 调节负端至最小 (一般情况下失调电压为正);
 111111: 调节正端至最小 (一般情况下失调电压为负)。

比较器 4 调零寄存器 CM4ADJ(116H)

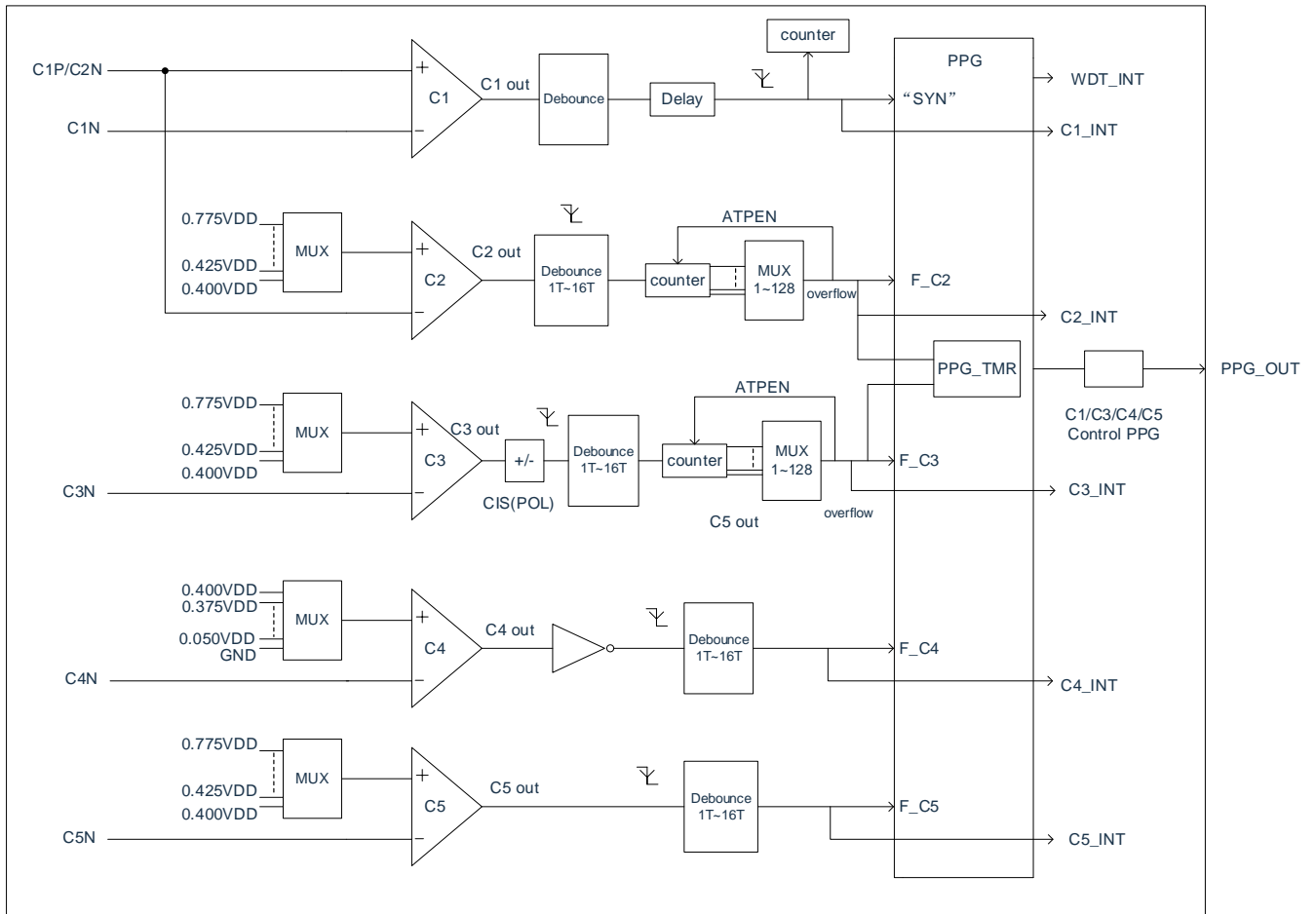
116H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM4ADJ	CM4OUT	CM4CRS	CM4ADJ[5:0]					
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	1	0	0	0	0	0

- Bit7 CM4OUT: 比较器 4 输出, 只读, 写操作不产生影响;
 0: 比较器输出 0;
 1: 比较器输出 1。
- Bit6 CM4CRS: 调节模式输入端选择;
 0: 负端输入;
 1: 正端输入。
- Bit5~Bit0 CM4ADJ[5:0]: 比较器 4 失调电压调节;
 000000: 调节负端至最小 (一般情况下失调电压为正);
 111111: 调节正端至最小 (一般情况下失调电压为负)。

比较器 5 调零寄存器 CM5ADJ(117H)

117H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM5ADJ	CM5OUT	CM5CRS	CM5ADJ[5:0]					
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	1	0	0	0	0	0

- Bit7 CM5OUT: 比较器 5 输出, 只读, 写操作不产生影响;
 0: 比较器输出 0;
 1: 比较器输出 1。
- Bit6 CM5CRS: 调节模式输入端选择;
 0: 负端输入;
 1: 正端输入。
- Bit5~Bit0 CM5ADJ[5:0]: 比较器 5 失调电压调节;
 000000: 调节负端至最小 (一般情况下失调电压为正);
 111111: 调节正端至最小 (一般情况下失调电压为负)。

15.4.5 比较器与 PPG 内部结构图


16. 数据 EEPROM 控制

16.1 数据 EEPROM 概述

数据 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 字的数据 EEPROM，地址范围为 0H 到 01FH。

EEPROM 数据存储器允许字节读写。字节写操作可自动擦除目标单元并写入新数据（在写入前擦除）。

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 仍可继续读写数据 EEPROM。代码保护时，器件编程器将不再能访问数据。

16.2 相关寄存器

16.2.1 EEADR 寄存器

EEADR 寄存器能寻址最大 32 字节数据 EEPROM。

16.2.2 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器。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，允许对数据 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 寄存器仅在执行数据 EEPROM 写序列时使用。

EEPROM 数据寄存器 EEDAT(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>: 对数据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>: 对数据EEPROM进行读取或写入的数据的高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: 数据EEPROM使能位;
 1= 允许操作数据EEPROM;
 0= 禁止操作数据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= 禁止写入数据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= 数据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。

EEPROM 控制寄存器 EECON1(18DH)

18DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON2	---							
读写	W							

EECON2 不是物理寄存器。读 EECON2 得到的是全 0。

EECON2 寄存器仅在执行数据 EEPROM 写序列时使用。

16.3 读数据 EEPROM 存储器

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。

读数据 EEPROM

LD	A,EE_ADD	;将要读取的地址放入 EEADR 寄存器
LD	EEADR,A	
SETB	EECON1,EEPGD	;使能数据 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

16.4 写数据 EEPROM 存储器

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。

写数据 EEPROM 存储器

LD	A,ADDR	;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

16.5 EEPROM 操作注意事项

16.5.1 写校验

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

16.5.2 避免误写的保护

有些情况下，用户可能不希望向数据 EEPROM 存储器写入数据。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

17. 运算放大器(OPA)

芯片内置一路运算放大器。

17.1 运算放大器简介

CMS89F52x 内置一运算放大器，其正端或负端可以通过软件设置内部直接接地或者通过一个电阻接地，输出可程序设置通过内部 RC 滤波接 AN9 或直接接 AN7 相应 AD 转换通道进行检测。其原理如下图：

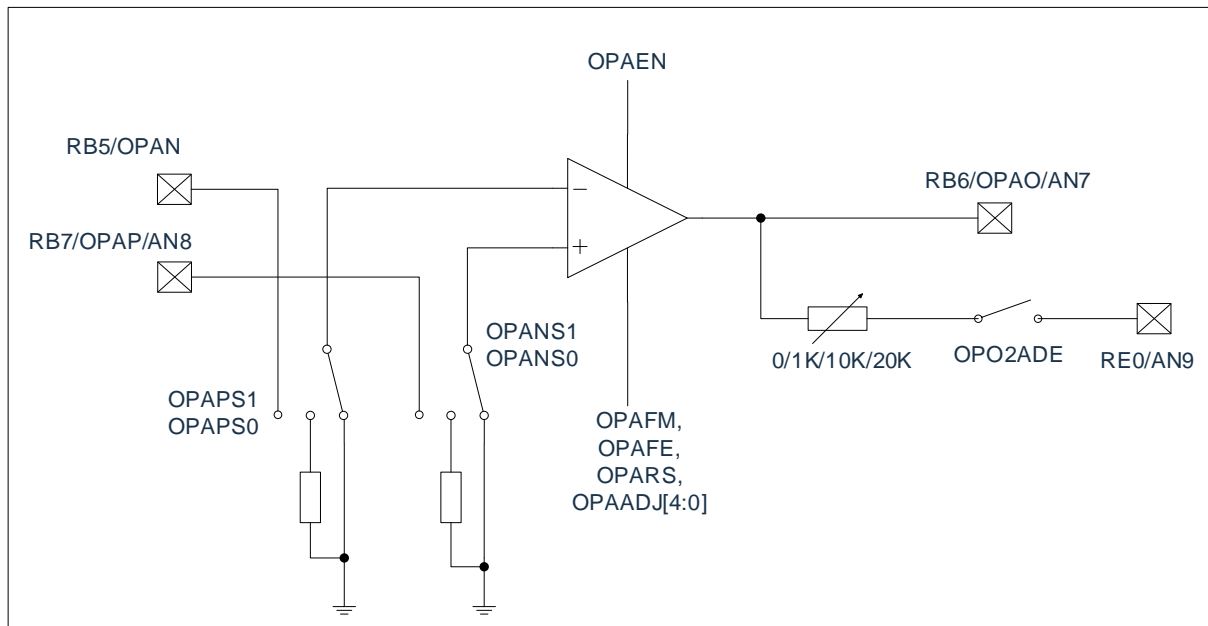


图 17-1：运算放大器工作原理

相关引脚说明

管脚名称	IO 类型	管脚说明
OPAN	I	运放负端输入
OPAP	I	运放正端输入
OPAO	O	运放输出
AN7	I	运放可内部接至该端口 AD 通道
AN9	I	运放可内部接至该端口 AD 通道，可在该端口外接电容滤波

17.2 跟运放相关寄存器

有 3 个寄存器跟运放相关，分别是 OPACON、OPACON1 和 OPAADJ。

运放控制寄存器 OPACON(108H)

108H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPACON	OPAEN	OPAFM	OPAFE	----	OPAPS1	OPAPS0	OPANS1	OPANS0
R/W	R/W	R/W	R/W	----	R/W	R/W	R/W	R/W
复位值	0	0	1	----	0	0	0	0

Bit7	OPAEN:	运放使能位; 0: 运放关闭; 1: 运放使能。
Bit6	OPAFM:	运放调节模式使能; 0: 正常模式; 1: 调节模式。
Bit5	OPAFE:	运放输出滤波使能; 0: 禁止; 1: 使能。
Bit4		禁用
Bit3~Bit2	OPAPS1-OPAPS0:	运放正端输入选择位; 00: 接 GND; 01: 接 1K 下拉电阻; 1x: 接 OPP 口 (若 OPAEN=1, 使能 OPP 口为运放正端输入)。
Bit1~Bit0	OPANS1-OPANS0:	运放负端输入选择位; 00: 接 GND; 01: 接 1K 下拉电阻; 1x: 接 OPN 口 (若 OPAEN=1, 使能 OPN 口运放负端输入)。

运放控制寄存器 OPACON1(109H)

109H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPACON1	----	----	----	----	OPO2ADE	----	ANRS1	ANRS0
R/W	----	----	----	----	R/W	----	R/W	R/W
复位值	----	----	----	----	0	----	0	0

Bit7~Bit4		禁用
Bit3	OPO2ADE:	运放输出口 OPAO 接到 CAP*口使能位; 0: 禁止; 1: 使能。
Bit2		禁用
Bit1~Bit0	ANRS1-ANRS0:	运放输出口 OPAO 接 CAP*口电阻选择位 (OPO2ADE=1 时生效); 00: 运放输出直接接至 CAP 口; 01: 运放输出接 1K 电阻至 CAP 口; 10: 运放输出接 10K 电阻至 CAP 口; 11: 运放输出接 20K 电阻至 CAP 口。

运放调节寄存器 OPAADJ(107H)

107H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPAADJ	OPADOUT	OPARS	----	OPAADJS[4:0]				
R/W	R	R/W	----	R/W	R/W	R/W	R/W	R/W
复位值	0	0	----	1	0	0	0	0

Bit7 OPADOUT: 运放调节模式下输出，只读。
 Bit6 OPARS: 运放调节模式输入端选择；
 0: 负端输入；
 1: 正端输入。
 Bit5 禁用
 Bit4~Bit0 OPAADJ[4:0]: 运放调节失调电压调节位。

18. 电气参数

18.1 直流电气特性

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	3.5	-	5.5	V
		-	4M	3.5	-	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	----		0.1		mA
		3V	----		0.1		mA
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

18.2 OPA 电气特性

(VDD=5.0V, TA= 25°C, 除非另有说明)

符号	参数	测试条件	最小值	典型值	最大值	单位
DC 电气特性						
VDD	工作电压	VDD=3.5~5.5V	3.5		5.5	V
I _{DD}	工作电流	VDD=5.0V, V _{CM} =0V		0.85	1.5	mA
I _{OFF}	关断电流	VDD=5.0V, V _{CM} =0V		3	100	nA
V _{OPOS}	输入失调电压	调零后 VDD=5V V _{CM} =0V	-4		4	mV
V _{CM}	共模电压范围		0		VDD-1.5V	V
V _{OH}	最大输出电压	I _{LOAD} =1mA			VDD-0.1	V
V _{OL}	最小输出电压	I _{LOAD} =1mA	0.1			V
PSRR	电源电压抑制比*	V _{CM} =0V		50		dB
CMRR	共模抑制比*	VDD=5V V _{CM} =0~VDD-1.5V		85		dB
AC 电气特性						
A _{OL}	开环增益*			70		dB
GBW	增益带宽*	R _L =1MΩ, C _L =100pF		0.4		MHz

*表示由设计保证, 未批量测试。

18.3 COMP 电气特性

(VDD=5.0V, TA= 25°C, 除非另有说明)

符号	参数	测试条件	最小值	典型值	最大值	单位
DC 电气特性						
VDD	工作电压	VDD=3.5~5.5V	3.5		5.5	V
I _{DD}	工作电流	VDD=5.0V, V _{CM} =0.1V		0.25	0.35	mA
I _{OFF}	关断电流	VDD=5.0V, V _{CM} =0.1V		3	100	nA
V _{OPOS}	输入失调电压	调零后 VDD=5V V _{CM} =0.1V	-4		4	mV
V _{CM}	共模电压范围		0		VDD-1.5V	V
PSRR	电源电压抑制比*	V _{CM} =0.1V		100		dB
CMRR	共模抑制比*	VDD=5V V _{CM} =0~VDD-1.5V		90		dB
AC 电气特性						
A _{OL}	开环增益*			85		dB
BW	带宽*			120		MHz
瞬态特性						
T _{STB}	稳定时间*	VDD=5.0V, V _{CM} =0.1V			1	us
T _{PGD}	响应时间*	V _{COMP-} = 1V, V _{COMP+} = V _{COMP-} ±0.1V		40	100	ns

*表示由设计保证, 未批量测试。

18.4 交流电气特性

符号	参数	测试条件		最小	典型	最大	单位
		VDD	条件				
TWDT	WDT 复位时间	5V	---	-	18	-	ms
		3V	---	-	36	-	ms
T _{AD}	AD 转换时间	5V	---	-	41	-	CLK
		3V	---	-	41	-	CLK
T _{EEPROM}	EEPROM 写入时间	5V	---	-	2.5	-	ms
		3V	---	-	2.5	-	ms

18.5 内部 RC 振荡特性

18.5.1 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

18.5.2 内部 RC 振荡温度特性

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

19. 指令

19.1 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

助记符	操作	指令周期	标志
DECR [R]	递减数据存储器 R，结果放入 R	1	Z
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

19.2 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
HSUBC   R01     ;执行结果: R01=(80H-77H-C)=09H(C=0)
R                          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

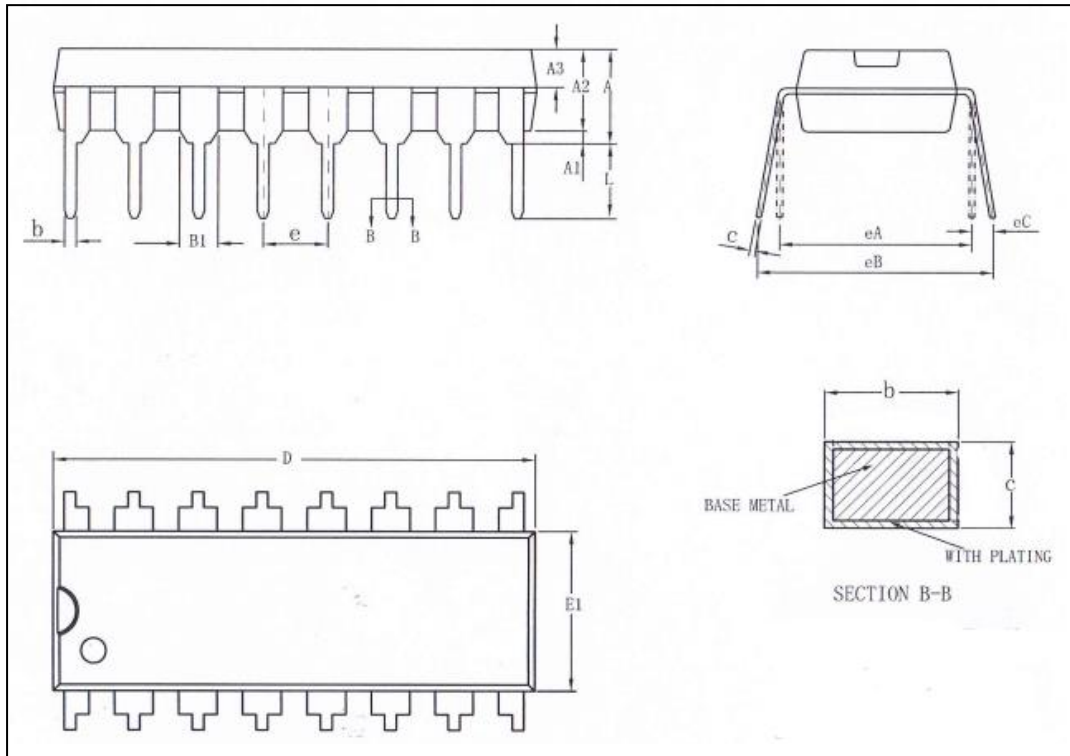
LD R01,A ;ACC 值赋给 R01,R01=0AH

LDIA 0FH ;ACC 赋值 0FH

XORR R01 ;执行结果: R01=05H

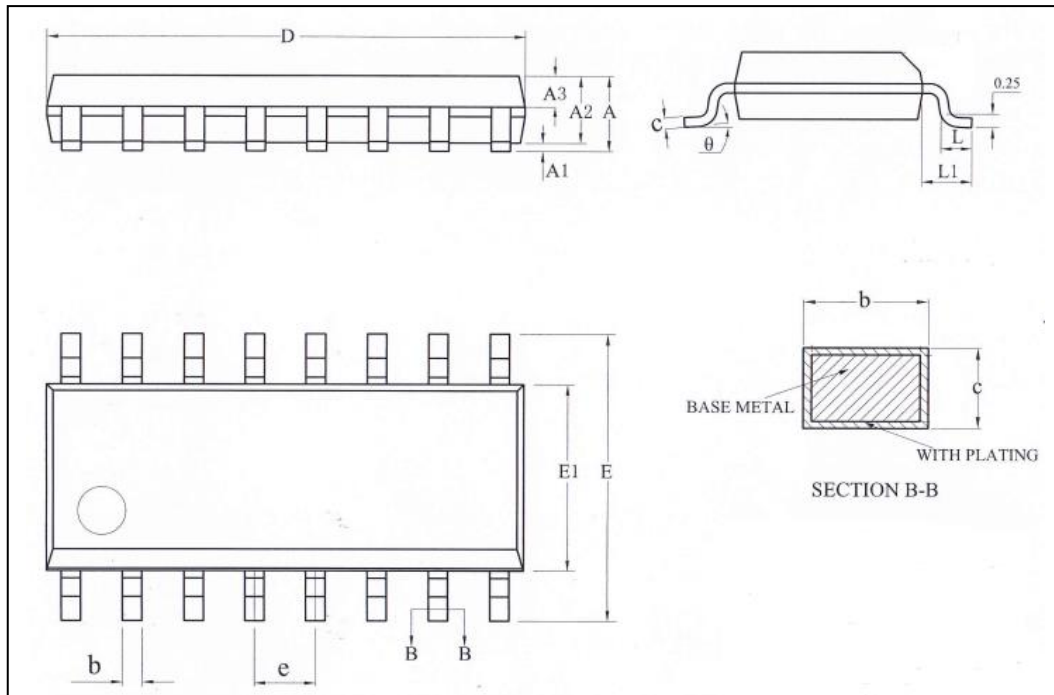
20. 封装

20.1 DIP16



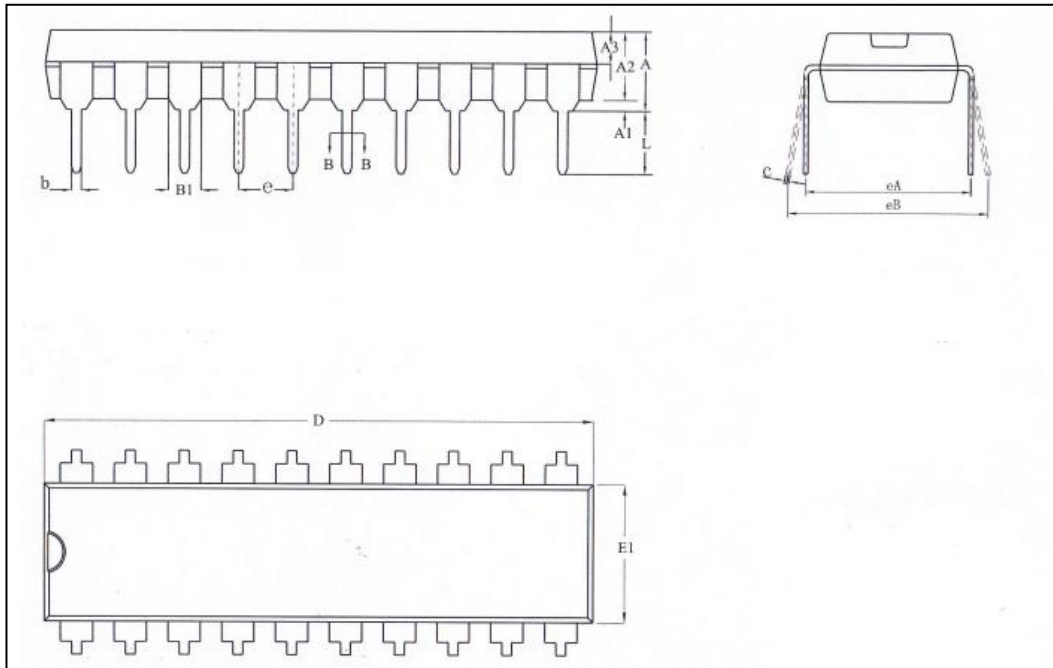
Symbol	Millimeter		
	Min	Nom	Max
A	3.60	-	4.80
A1	0.50	-	-
A2	3.05	-	3.45
A3	1.40	-	1.60
b	0.38	-	0.55
B1	1.52REF		
c	0.21	-	0.35
D	19.00	-	19.40
E1	6.25	6.35	6.45
e	2.54BSC		
eA	7.62REF		
eB	7.62	-	10.90
eC	0	-	1.52
L	2.92	-	-

注意：封装尺寸不包括模的毛边凸起或门毛刺。

20.2 SOP16


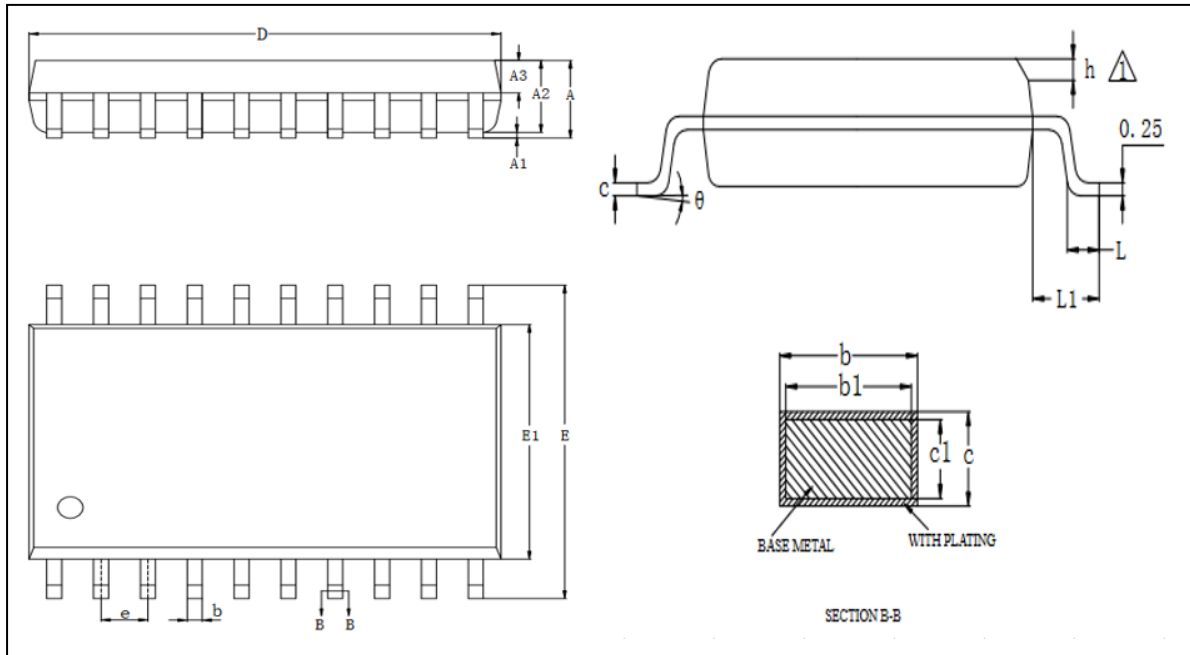
Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.85
A1	0.05	-	0.25
A2	1.30	1.40	1.60
A3	0.60	0.65	0.71
b	0.35	-	0.51
c	0.19	-	0.26
D	9.70	9.90	10.10
E	5.80	6.00	6.20
E1	3.70	3.90	4.10
e	1.27BSC		
L	0.40	-	0.81
L1	1.05REF		
θ	0	-	8°

注意：封装尺寸不包括模的毛边凸起或门毛刺。

20.3 DIP20


Symbol	Millimeter		
	Min	Nom	Max
A	3.60	3.80	4.00
A1	0.51	-	-
A2	3.20	3.30	3.40
A3	1.47	1.52	1.58
b	0.44	-	0.52
B1	1.52REF		
D	25.80	-	26.33
E1	6.35	-	6.65
e	2.54BSC		
eA	7.62REF		
eB	7.62	-	9.30
L	3.00	-	-
θ	0	-	8°


注意：封装尺寸不包括模的毛边凸起或门毛刺。

20.4 SOP20


Symbol	Millimeter		
	Min	Nom	Max
A	-	-	2.65
A1	0.10	-	0.30
A2	2.24	-	2.44
A3	0.97	1.02	1.07
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.25	-	0.30
c1	0.24	0.25	0.26
D	12.65	-	12.90
E	10.10	10.30	10.50
E1	7.40	7.50	7.60
e	1.27BSC		
h	0.50REF		
L	0.70	-	1.01
L1	1.40REF		
θ	0	-	8°

注意：封装尺寸不包括模的毛边凸起或门毛刺。

21. 版本修订说明

版本号	时间	修改内容																																																																																																										
V1.0	2016年07月	整个文档里的 CCP2IN2 改为 CCP2IN0 (P26、P95)																																																																																																										
V1.1	2016年07月	预分频器应用增加 2 行指令，确保不会造成复位 <pre> 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 </pre>																																																																																																										
	2016年07月	添加 SSPMSK 地址。																																																																																																										
V1.2	2016年08月	1、章节 16.4 原文档： 例 16-2 写数据 EEPROM 存储器  修改方案： 选择的部分 4 行，前两行改为： <pre>LD A, ADDR ;写地址 LD EEADR, A</pre> 后两行删除。																																																																																																										
		2、章节 16.2.2 EEPROM 地址寄存器 EEADR(10DH) <table border="1" data-bbox="798 1220 1380 1310"> <thead> <tr> <th>10DH</th> <th>Bit7</th> <th>Bit6</th> <th>Bit5</th> <th>Bit4</th> <th>Bit3</th> <th>Bit2</th> <th>Bit1</th> <th>Bit0</th> </tr> </thead> <tbody> <tr> <td>EEADR</td> <td>---</td> <td>---</td> <td>---</td> <td>EEADR4</td> <td>EEADR3</td> <td>EEADR2</td> <td>EEADR1</td> <td>EEADR0</td> </tr> <tr> <td>读写</td> <td>---</td> <td>---</td> <td>---</td> <td>R/W</td> <td>R/W</td> <td>R/W</td> <td>R/W</td> <td>R/W</td> </tr> <tr> <td>复位值</td> <td>---</td> <td>---</td> <td>---</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table> Bit 7-6 --- (无关) Bit 4-0 EEADR<4:0> ：指定EEPROM读取/写入操作的地址 EEPROM 数据寄存器 EEDATH(10EH) <table border="1" data-bbox="798 1400 1380 1489"> <thead> <tr> <th>10EH</th> <th>Bit7</th> <th>Bit6</th> <th>Bit5</th> <th>Bit4</th> <th>Bit3</th> <th>Bit2</th> <th>Bit1</th> <th>Bit0</th> </tr> </thead> <tbody> <tr> <td>EEDATH</td> <td>EEDATH7</td> <td>EEDATH6</td> <td>EEDATH5</td> <td>EEDATH4</td> <td>EEDATH3</td> <td>EEDATH2</td> <td>EEDATH1</td> <td>EEDATH0</td> </tr> <tr> <td>读写</td> <td>R/W</td> <td>R/W</td> <td>R/W</td> <td>R/W</td> <td>R/W</td> <td>R/W</td> <td>R/W</td> <td>R/W</td> </tr> <tr> <td>复位值</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table> Bit 7-0 EEDATH<7:0> ：对数据EEPROM 进行读取或写入数据的高8 位 EEPROM 控制寄存器 EECON1 (18CH) <table border="1" data-bbox="798 1556 1380 1646"> <thead> <tr> <th>18CH</th> <th>Bit7</th> <th>Bit6</th> <th>Bit5</th> <th>Bit4</th> <th>Bit3</th> <th>Bit2</th> <th>Bit1</th> <th>Bit0</th> </tr> </thead> <tbody> <tr> <td>EECON1</td> <td>EEPGD</td> <td>---</td> <td>---</td> <td>---</td> <td>WRERR</td> <td>WREN</td> <td>WR</td> <td>RD</td> </tr> <tr> <td>读写</td> <td>R/W</td> <td>---</td> <td>---</td> <td>---</td> <td>R/W</td> <td>R/W</td> <td>R/W</td> <td>R/W</td> </tr> <tr> <td>复位值</td> <td>0</td> <td>---</td> <td>---</td> <td>---</td> <td>x</td> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table> Bit 7-6 EEPGD ：数据EEPROM 使能位 1 = 允许操作数据EEPROM 0 = 禁止操作数据 EEPROM 未用，设为 0 Bit 6-4 WRERR ，EEPROM 错误标志位 Bit 3 WREN ，EEPROM 写使能位 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 Bit 2 WR ，EEPROM 写使能位 1 = 写操作完成 Bit 1 RD ，EEPROM 读使能位 1 = 读操作完成 Bit 0 RD ，EEPROM 读使能位 1 = 读操作完成 1 处： EEDATA1 改为 EEADR1， EEDATA0 改为 EEADR0， 2 处： Bit7-6 改为 Bit7	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	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	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
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																																																																																																				
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																																																																																																				
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																																																																																																				

3、章节 2.1.2

表 2-2 CMS89F52X 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	Tmr0数据寄存器									xxxx xxxx
02h	PCL	程序计数器低字节									0000 0000
03h	STATUS	IRP	---	---	TO	PD	Z	DC	C	0001 1xxx	
04h	FSR	间接数据存储器地址指针									xxxx xxxx
05h	PORTA	---	RA6	RA5	RA4	RA3	RA2	RA1	RA0	-xxx xxxx	
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	
09h	PORTE	---	---	---	---	---	---	RE1	RE0	--- -xx	
0Ah	PCLATH	---	---	---	---	---	---	---	---	-0 0000	
0Bh	INTCON	GIE	PEIE	TOIE	INTE	---	TOIF	INTF	---	0000 -00	
0Ch	PIR1	EEIF	ADIF	SSPIF	BCLIF	CCPIF	---	TMR2IF	TMR1IF	0000 0-00	
0Dh	PIR2	---	---	CSIF	CAIF	C3IF	C2IF	C1IF	PPGIF	-00 0000	
0Eh	TMR1L	16位TMR1寄存器低字节的数据寄存器									xxxx xxxx
0Fh	TMR1H	16位TMR1寄存器高字节的数据寄存器									xxxx xxxx
10h	T1CON	T1GNV	TMR1GE	T1CKPS1	T1CKPS0	TOOSCEN	T1SYNC	TMR1CS	TMR1ON	0000 0000	
11h	TMR2	TIMER2寄存器									0000 0000
12h	T2CON	---	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2CN	T2CKPS1	T2CKPS0	-000 0000	
14h	PPGMTMR										
15h	PPGMTMRH										
16h	PPGDLY										

1 处: 改为“PPGWDTIF”

2 处: 改为“----”与“----”

4、章节 2.1.2

表 2-3 CMS89F52X 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	RBPU	INTEG	TOCS	TOSE	PSA	PS2	PS1	PS0	1111 1111	
82h	PCL	程序计数器(PC)的低字节									0000 0000
83h	STATUS	IRP	---	---	TO	PD	Z	DC	C	0-1 1xxx	
84h	FSR	间接数据存储器地址指针									xxxx xxxx
85h	TRISA	---	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	-111 1111	
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	---	1111 1111	
89h	TRISE	---	---	---	---	TRISE3	TRISE2	TRISE1	TRISE0	--- 1111	
8Ah	PCLATH	---	---	---	---	---	---	---	---	-0 0000	
8Bh	INTCON	GIE	PEIE	TOIE	INTE	---	TOIF	INTF	---	0000 -00	
8Ch	PIE1	EEIE	ADIE	SSPIE	BCLIE	CCPIE	---	TMR2IE	TMR1IE	0000 0-00	
8Dh	PIE2	---	---	CSIE	CAIE	C3IE	C2IE	C1IE	PPGIE	-00 0000	
8Fh	OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	---	---	-110 ---	
90h	OSCTUNE	---	---	TUN4	TUN3	TUN2	TUN1	TUN0	---	-0 0000	
92h	PR2	TIMER2周期寄存器									1111 1111
93h	CM1CNT	CM1OF	CM1CNT[6:0]							---	0000 0000
94h	WPUA	---	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	-000 0000	
95h	WPUB	---	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	0000 0000	

1 处: 改为“----”

2 处: 改为“TRISB0”

3 处: 改为“----”与“----”

4 处: 改为“ADIE”

5 处: 改为“PPGWDTIE”

5、章节 2.1.2

表 2-5 CMS89F52X 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	RBPU	INTEG	TOCS	TOSE	PSA	PS2	PS1	PS0	1111 1111	
182h	PCL	程序计数器(PC)的低字节									0000 0000
183h	STATUS	IRP	---	---	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	
187h	PAANSEL	---	PAANS6	PAANS5	PAANS4	PAANS3	PAANS2	PAANS1	PAANS0	-000 0000	
188h	PBANSEL	PBANST	PBANST	PBANST	PBANST	PBANST	PBANST	PBANST	PBANST	0000 0000	
189h	PEANSEL	---	---	---	---	---	---	PEANS1	PEANS0	--- 00	
18Ah	PCLATH	---	---	---	---	---	---	---	---	-0 0000	
18Bh	INTCON	GIE	PEIE	TOIE	INTE	---	TOIF	INTF	---	0000 0000	
18Ch	EEDCON1	EEPGD	---	---	---	WRERR	WREN	WR	RD	0-1000	
18Dh	EEDCON2	EEPROM控制寄存器2(不是物理寄存器)									---
18Eh	CCPRL	捕捉寄存器的低字节									xxxx xxxx
18Fh	CCPRH	捕捉寄存器的高字节									xxxx xxxx
190h	CCPCON	CCPEN	---	---	CCPS1	CCPS0	CPTCS2	CPTCS1	CPTCS0	0-0000	
191h	SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	11111111	
191h	SSPADD	同步串行接口(C)模式地址寄存器									0000 0000
192h	SSPBUF	同步串行接口接收缓冲器/发送寄存器									xxxx xxxx

1 处: 改为“----”

2 处: “CPTCS2”改为“CCPM2”, “CPTCS1”改为“CCPM1”, “CPTCS0”改为“CCPM0”

		<h3>6、章节 15.3</h3> <p>PPG 控制寄存器 PPGCON(17H)</p> <table border="1"> <thead> <tr> <th>17H</th> <th>Bit7</th> <th>Bit6</th> <th>Bit5</th> <th>Bit4</th> <th>Bit3</th> <th>Bit2</th> <th>Bit1</th> <th>Bit0</th> </tr> </thead> <tbody> <tr> <td>PPGCON</td> <td>DETC5F</td> <td>DETC4F</td> <td>----</td> <td>RELOAD_EN</td> <td>DETC4EN</td> <td>DETC3EN</td> <td>PPGMD</td> <td>PPG_ON</td> </tr> <tr> <td>RW</td> <td>RW</td> <td>RW</td> <td>----</td> <td>RAW</td> <td>RAW</td> <td>RW</td> <td>RW</td> <td>RAW</td> </tr> <tr> <td>复位值</td> <td>1</td> <td>1</td> <td>----</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table> <p>Bit7 DETC5F: 比较器 5 状态位 (PPG 状态位) 0: 清零比较器 5 的 0->1 翻转标志, (若 DETC4EN=1, 则 PPG 重新打开) 1: 有比较器 5 的 0->1 翻转, 写 1 无效, (若 DETC4EN=1, 则 PPG 关闭)</p> <p>Bit6 DETC4F: 比较器 4 状态位 (PPG 状态位) 0: 清零比较器 4 的 1->0 翻转标志, (若 DETC3EN=1, 则 PPG 重新打开) 1: 有比较器 4 的 1->0 翻转, 写 1 无效, (若 DETC3EN=1, 则 PPG 关闭)</p> <p>Bit5 禁用</p> <p>Bit4 RELOAD_EN: PPG TMR 加载使能 0: 允许加载 (自动降功率模式时, 同时允许 PPG TMR 自-1) 1: 禁止加载 (自动降功率模式时, 同时禁止 PPG TMR 自-1)</p> <p>Bit3 DETC5EN: 比较器 5 关闭 PPG 使能位 0: 禁止 1: 使能</p> <p>Bit2 DETC4EN: 比较器 4 关闭 PPG 使能位</p> <p>1 处: “DETC4EN”改为“DETC5EN”, “DETC3EN”改为“DETC4EN”</p>	17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	PPGCON	DETC5F	DETC4F	----	RELOAD_EN	DETC4EN	DETC3EN	PPGMD	PPG_ON	RW	RW	RW	----	RAW	RAW	RW	RW	RAW	复位值	1	1	----	0	0	0	0	0
17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0																														
PPGCON	DETC5F	DETC4F	----	RELOAD_EN	DETC4EN	DETC3EN	PPGMD	PPG_ON																														
RW	RW	RW	----	RAW	RAW	RW	RW	RAW																														
复位值	1	1	----	0	0	0	0	0																														
V1.3	2017 年 11 月	修改文中多处表述错误																																				
V1.4	2020 年 04 月	1、增加 OPA, COMP 电气参数 2、更正封装图中的一些错误																																				
V1.5	2021 年 08 月	更新管脚信息																																				
V1.6	2022 年 02 月	更改为新格式																																				
V1.7.0	2024 年 09 月	修改 DIP16/SOP16/DIP20/SOP20 封装尺寸信息																																				