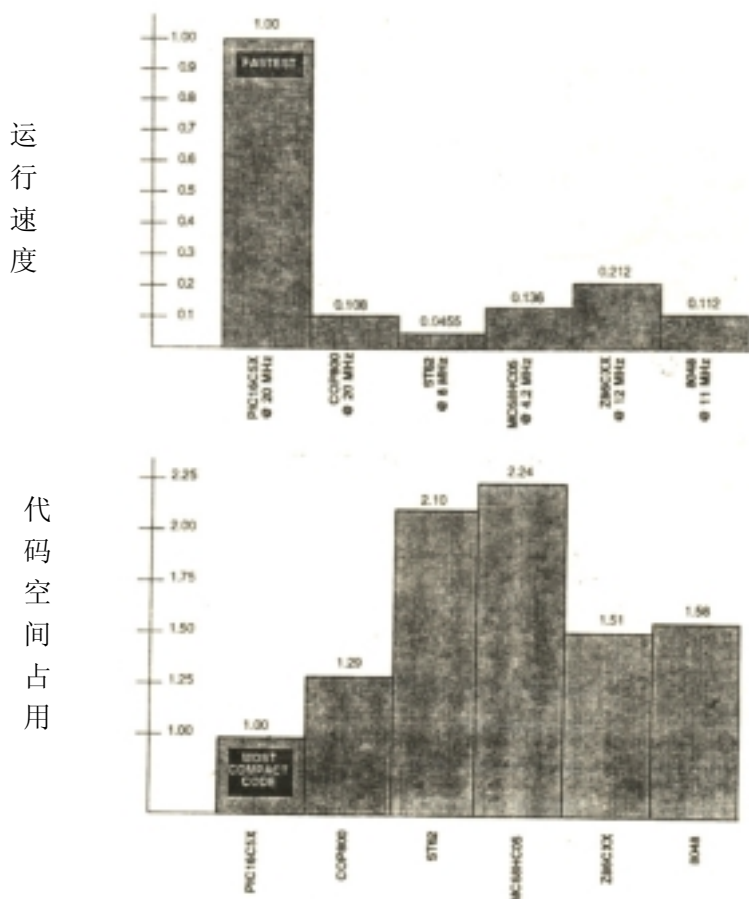


概 述

美国 Microchip 公司推出的 PIC8 位单片机系列是业内首先采用 RISC 结构的高性价比嵌入式控制器，其高速度、低电压工作，低功耗，强大驱动能力，低价 OTP 技术，体积小巧等都体现了单片微控制器工业的新趋势。PIC 单片机从覆盖市场的角度出发，发展出三层次系列多种型号的产品来满足不同的产品设计需求。所以在全球都可以看到 PIC 单片机在从电脑鼠标器、家电控制、电讯通信、智能仪器仪表到汽车电子，金融电子各领域的广泛应用。PIC 单片机已经是世界上最有影响力的主要嵌入式微控制器之一。

PIC 单片机总线结构采取数据总线和指令线分离独立的哈佛(Harvard)结构,具有很高的流水处理速度。它的精简指令集结构(RISC)使的它所有的指令都是单字节, 因此其程序空间的效率比一般单片机高很多。下面二图是 PIC 单片机和其他一些单片机在处理速度和代码紧凑性方面的比较。



PIC 单片机三个层次的系列产品如下表所示：

系 列		主 要 特 性	名 称	工艺及特点	型 号
高 级	PIC17CXXX 8 位单片机	<ul style="list-style-type: none"> • 16 位指令系统；8 位数据线 • 多种中断 • DC~25MHz 时钟 • 最快 160ns 指令周期 • 硬件乘法器，一个指令周期完成 8 位乘法运算 • 高性价比,可取代某些 DSP 	PIC17C4X	OTP/EPROM	PIC17C42 PIC17C43 PIC17C44
中 级	PIC16CXXX 8 位单片机	<ul style="list-style-type: none"> • 14 位指令系统,8 位数据线 • 多种中断 • DC~20MHz 时钟 • 最快 200ns 指令周期 • 8 位 A/D (PIC16C7X) • 电压比较器 (PIC16C62X) • 复位锁定 • E²PROM (PIC16C8X) • LCD 驱动 (PIC16C9XX) • 混合信号处理 (PIC14000) • 低价格 	PIC16C6X	OTP/EPROM	PIC16C61 PIC16C62 PIC16C63 PIC16C64 PIC16C65
			PIC16CR6X	ROM (掩膜)	PIC16CR62 PIC16CR64
			PIC16C62X	OTP/EPROM 含比较器	PIC16C620 PIC16C621 PIC16C622
			PIC16C7X	OTP/EPROM 含 8 位 A/D	PIC16C70 PIC16C71 PIC16C72 PIC16C73 PIC16C74
			PIC16C8X	E ² PROM 程序存储器和 数据存储器	PIC16C83 PIC16C84
			PIC16CR8X	ROM (掩膜) 含 E ² PROM 数据存储器	PIC16CR83 PIC16CR84
			PIC16C9XXX	OTP/EPROM 含 LCD 驱动	PIC16C923 PIC16C924
			PIC14000	OTP/EPROM 含 A/D, D/A 和温度传感器	PIC14000
基 本 级	PIC16C5XX PIC12C5XXX 8 位单片机	<ul style="list-style-type: none"> • 12 位指令系统,8 位数据线 • DC~20MHz 时钟 • 最快 200ns 指令周期 • 8 脚封装(PIC12C5XX) • 极低价格 	PIC16C5X	OTP/EPROM	PIC16C52 PIC16C54 PIC16C55 PIC16C56 PIC16C57 PIC16C58
			PIC16CR5X	ROM (掩膜)	PIC16CR54 PIC16CR57 PIC16CR58
			PIC12C5XX	OTP/EPROM 8 脚封装	PIC12C508 PIC12C509

☆基本级产品——PIC16C5X

PIC16C5X 是最早发展的系列。它的特点是低价,适用于各种对成本要求严格的嵌入式控制。从应用范围到采用的数量, PIC16C5X 都是首屈一指的。而 PIC12C5XX 是世界第一个 8 脚低价单片机,其小巧低价足以使其应用在很多以前不能用单片机的地方,前景广阔。

☆中级产品——PIC16CXX

PIC16CXX 是品种最丰富的系列。它在 PIC16C5X 的基础上进行了很多改进,并保持了很高的兼容性。从 18 脚到 44 脚各种形式的封装, PIC16CXX 可应用于各种高、中、低档电子产品设计中。它的特点是在保持低价格的前提下具有很高的性能,如带 A/D,内部 E²PROM 双时钟工作,捕捉输入, PWM 输出, I²C 和 SPI 接口,异步串行通讯(USART),模拟电压比较器及 LCD 驱动等等,使之成为业界瞩目的机种,已被广泛应用在各种电子产品中,表现极佳。

☆高级产品——PIC17CXX

PIC17CXX 是目前世界上 8 位单片机中运行速度最快的,它具备了一个指令周期内(最短 160ns)完成 8 位×8 位二进制乘法运算的能力,可以在一些需要高速数字运算的应用场合中取代 DSP(数字讯号处理器)。再加上 PIC17CXX 还具备了丰富的 I/O 控制功能,并可以外接扩展 EPROM 和 RAM,使它成为目前 8 位单片机中性能最高的机种之一,被广泛应用于各种高、中档电子设备中。

PIC 系列单片机还具有非常优秀的微处理特性,如多种复位方式,多种中断功能,低功耗睡眠功能,掉电复位锁定等等。在 PIC 单片机的内部还集成有上电复位电路(POR),看门狗电路,I/O 口弱上拉等,可以大大减少外围器件,节省用户的空间和成本。

PIC 单片机的功耗是目前世界上单片机中较低的一种,而驱动能力又大大强于别的单片机。更值得称道的是它的保密技术,目前尚无办法对其直接进行解密拷贝,可以最大限度保护用户的程序版权。

PIC 单片机所有型号都有商用级(0℃~+70℃)、工业级(-40℃~+85℃)和汽车工业级(-40℃~+125℃)芯片,可以适应各种环境温度。

Microchip 公司的这三个层次的 PIC 单片机具有很高的代码兼容性。用户很容易将代码从某型号转移到另一个型号上,所以用户总是可以选择一种最适合于自己产品的型号,使自己的投资得到最好的回报。

第一章 PIC16C6X 单片机

PIC16C6X 目前各型号主要功能配置如下表所示：

型号	振荡	EPROM	RAM	定时器	CCP 模块	串行口	并行口	中断源	电压范围	I/O	封装	复位 锁定
16C61	DC~20M	1K×14	36×8	1	—	—	—	3	3.0-6.0V	13	18 脚	无
16C62A	DC~20M	2K×14	128×8	3	1	SPI/I2C	—	7	2.5-6.0V	22	28 脚	有
16C62B	DC~20M	2K×14	128×8	3	1	SPI/I2C	—	7	2.5-6.0V	22	28 脚	有
16C63A	DC~20M	4K×14	192×8	3	2	SPI/I2C USART	—	11	2.5-6.0V	22	28 脚	有
16C64A	DC~20M	2K×14	128×8	3	1	SPI/I2C	有	8	2.5-6.0V	33	40 脚	有
16C65A	DC~20M	4K×14	192×8	3	2	SPI/I2C USART	有	11	3.0-6.0V	33	40 脚	有
16C65B	DC~20M	4K×14	192×8	3	2	SPI/I2C USART	有	11	2.5-6.0V	33	40 脚	有
16C66	DC~20M	8K×14	368×8	3	2	SPI/I2C USART	—	10	2.5-6.0V	22	28 脚	有
16C67	DC~20M	8K×14	368×8	3	2	SPI/I2C USART	有	11	2.5-6.0V	33	40 脚	有

表 1.1 PIC16C6X 型号功能表

§ 1.1 主要功能特点

一、高性能 RISC 结构 CPU

- 精简指令集，仅 35 条单字节指令，易学易用
- 除地址分支跳转指令（GOTO、CALL）为双周期指令，其余皆为单周期指令
- 执行速度

时钟振荡	指令周期
40HZ	100 μ S
1MHZ	4 μ S
4MHZ	1 μ S
10MHZ	400ns
20MHZ	200ns

- 八级硬件堆栈
- 直接、间接、相对三种寻址方式

二、功能部件特性

- 高驱动电流，I/O 脚可直接驱动数码管（LED）显示
 - 每个 I/O 引脚最大拉电流 25MA
 - 每个 I/O 引脚最大灌电流 20MA
- 双向可独立编程设置 I/O 引脚
- 8 位定时器/计数器 TMR0，带 8 位预分频
- 有 1~2 路捕捉输入/比较输出/PWM 输出
- 16 位定时器/计数器 TMR1，睡眠中仍可计数
- 8 位定时器/计数器 TMR2，带有 8 位的周期寄存器及预分频和后分频
- 并行口操作
- 同步串行口 I²C/SPI 总线操作
- 同步通讯接口 SCI/USART 操作

三、微控制器特性

- 内置上电复位电路（POR）
- 上电定时器，保障工作电压的稳定建立
- 振荡定时器，保障振荡的稳定建立
- 断电复位锁定（16C62A/63/64A/65A）
- 内置自振式（RC 振荡）看门狗
- 程序保密位，可防程序代码的非法拷贝
- 四种可选振荡方式
 - 低成本阻容：RC
 - 标准晶体/陶瓷：XT
 - 高速晶体/陶瓷：HS
 - 低频晶体：LP
- 多种硬件中断方式

四、CMOS 工艺特性

- 低功耗
 - <2MA @5V, 4MHZ
 - <15UA @3V, 32KHZ
 - <1UA 低功耗 Sleep 模式下
- 全静态设计
- 宽工作电压：2.5V~6.0V
- 宽工作温度范围
 - 商用级：0℃~+70℃
 - 工业级：-40℃~+85℃
 - 汽车级：-40℃~+125℃

由于 PIC16C6X 在一个芯片内集成了众多的功能模块并具备优秀微处理器特性和 CMOS 工艺特点，因此它可以减少外部器件、提高产品可靠性和降低成本，另外它的低功耗及宽工作电平、宽工作温度和各种小巧封装，使得它在几乎每个电子产品领域都能得到应用。

§ 1.2 芯片类型

PIC16C6X 目前有 7 种型号，备有各种封装，包括未封装半导体芯（Dies）。同时还提供以下几种形式的芯片以满足单片应用从开发到中小批量试产直至大批量投放的要求。

一、可重擦写型（UV）

系陶瓷封装，中间开有窗口，可以用紫外光擦除重新烧写，是开发中理想的选用芯片。

二、一次性编程型（OTP）

这是 PIC16CXX 最具代表性的芯片，系塑料封装，允许用户用编程器对其进行一次性编程。这个特点使它成为最具商用意义的型号。在多变竞争激烈的电子产品市场，OTP 型芯片具有三大意义：1、产品快速上市；2、可随时修改程序，免除掩膜风险；3、减轻库存压力。

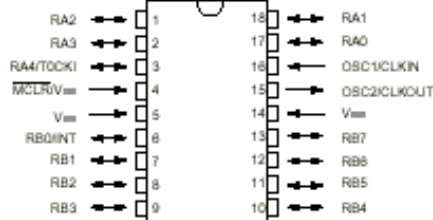
三、掩膜型（MASK）

当用户的程序代码稳定、并具有一定的市场后，可以选择掩膜型芯片以降低成本。

§ 1.3 引脚介绍

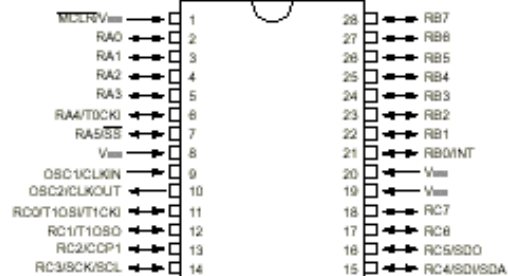
PIC16C6X 芯片具有引脚如下：

PDIP, SOIC, Windowed CERDIP



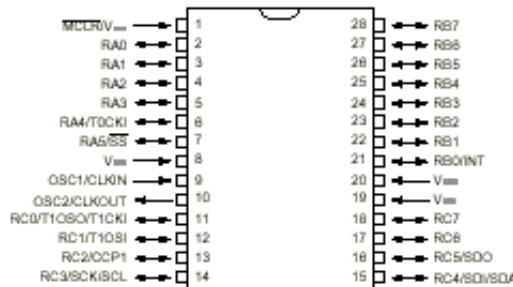
PIC16C61

SDIP, SOIC, SSOP, Windowed CERDIP (300 mil)



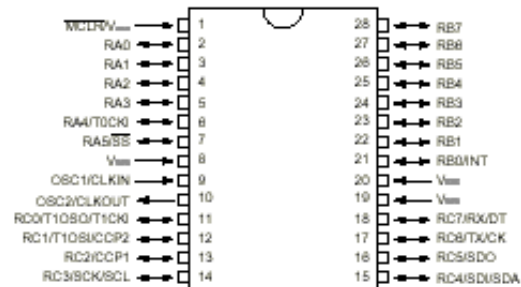
PIC16C62

SDIP, SOIC, SSOP, Windowed CERDIP (300 mil)



**PIC16C62A
PIC16CR62**

SDIP, SOIC, Windowed CERDIP (300 mil)

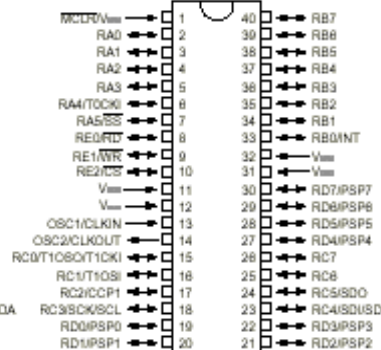


**PIC16C63
PIC16CR63
PIC16C66**

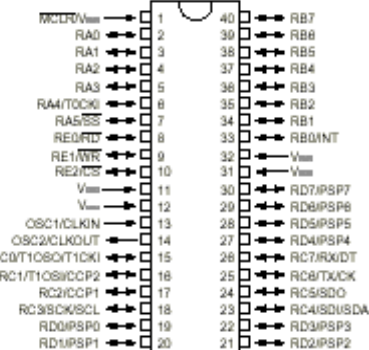
PDIP, Windowed CERDIP



PIC16C64



**PIC16C64A
PIC16CR64**



**PIC16C65
PIC16C65A
PIC16CR65
PIC16C67**

各种型号的引脚意义如下表所示：

引 脚 名	I/O 特性	电 平	功 能
OSC1/CLKIN	输入	CMOS	振荡输入脚
OSC2/CLKOUT	输出	—	振荡输出脚
MCLR	输入	斯密特输入	复位输入脚，低电平有效
RA0	I/O	TTL	PORTA 数字 I/O 口，双向可编程 也可作为 TMR0 计数器外部时钟输入端 也可作为同步串行口的从属器选择输入
RA1	I/O	TTL	
RA2	I/O	TTL	
RA3	I/O	TTL	
RA4/T0CKI	I/O	斯密特输入	
RA5/SS	I/O	TTL	
RB0/INT	I/O	TTL/斯密特	PORTB 数字 I/O 口，双向可编程，并带可编程弱上拉。 也可作为外部中断信号输入 具电平变化中断功能 具电平变化中断功能 具电平变化中断功能 具电平变化中断功能
RB1	I/O	TTL	
RB2	I/O	TTL	
RB3	I/O	TTL	
RB4	I/O	TTL	
RB5	I/O	TTL	
RB6	I/O	TTL	
RB7	I/O	TTL	
RC0/T1OSI/T1CKI	I/O	斯密特输入	PORTC 数字 I/O 口，亦可作为一些特殊功能。 亦可作 TMR1 振荡输入/TMR1 时钟输入 亦可作 TMR1 振荡输出 亦可作 CCP1 输入输出 亦可作同步串行时钟 亦可作 SPI 通讯数据输入线或 I ² C 之数据线 亦可作 SPI 通讯数据输出线
RC1/T1OSO	I/O	斯密特输入	
RC2/CCP1	I/O	斯密特输入	
RC3/SCK/SCL	I/O	斯密特输入	
RC4/SDI/SDA	I/O	斯密特输入	
RC5/SDO	I/O	斯密特输入	
RC6	I/O	斯密特输入	
RC7	I/O	斯密特输入	
RD0/PSP0	I/O	斯密特/TTL	PORTD 数字 I/O 口，双方可编程，亦可作为并行口，当作为并行口时具有 TTL 输入，作为一般 I/O 口时为斯密特输入。
RD1/PSP1	I/O	斯密特/TTL	
RD2/PSP2	I/O	斯密特/TTL	
RD3/PSP3	I/O	斯密特/TTL	
RD4/PSP4	I/O	斯密特/TTL	
RD5/PSP5	I/O	斯密特/TTL	
RD6/PSP6	I/O	斯密特/TTL	
RD7/PSP7	I/O	斯密特/TTL	
RE0/RD	I/O	斯密特/TTL	PORTE I/O 口，双向可编程，亦可作为并行口的控制线。 RD：读 WR：写 CS：片选
RE1/WR	I/O	斯密特/TTL	
RE2/CS	I/O	斯密特/TTL	
VSS	—	—	地
VDD	—	—	电源
NC	—	—	未用

注：16C62 没有 PORTD 和 PORTE。

a. 16C62/62A/64/64A

引脚名	I/O 特性	电 平	功 能
OSC1/CLKIN	输入	CMOS	振荡输入脚
OSC2/CLKOUT	输出	—	振荡输出脚
MCLR	输入	斯密特输入	复位输入脚，低电平有效
RA0 RA1 RA2 RA3 RA4/T0CKI RA5/SS	I/O I/O I/O I/O I/O I/O	TTL TTL TTL TTL 斯密特输入 TTL	PORTA 数字 I/O 口，双向可编程 也可作为 TMR0 计数器外部时钟输入端 也可作为同步串行口的从属器选择输入
RB0/INT RB1 RB2 RB3 RB4 RB5 RB6 RB7	I/O I/O I/O I/O I/O I/O I/O I/O	TTL/斯密特 TTL TTL TTL TTL TTL TTL TTL	PORTB 数字 I/O 口，双向可编程，并带可编程弱上拉，也可作为外部中断信号输入 具电平变化中断功能 具电平变化中断功能 具电平变化中断功能 具电平变化中断功能
RC0/T1OSO /T1CKI RC1/T1OSI /CCP2 RC2/CCP1 RC3/SCK/SCL RC4/SDI/SDA RC5/SDO RC6/TX/CK RC7/RX/DT	I/O I/O I/O I/O I/O I/O I/O I/O	斯密特输入 斯密特输入 斯密特输入 斯密特输入 斯密特输入 斯密特输入 斯密特输入 斯密特输入	PORTC 数字 I/O 口，双向可编程，亦可作一些特殊功能 亦可作为 TMR1 振荡输出/TMR1 时钟输入 亦可作为 TMR1 振荡输入/CCP2 输入输出 亦可作为 CCP1 输入输出 亦可作为同步串行通讯时钟 亦可作为 SPI 通讯之数据输入线或 I ² C 之数据线 亦可作为 SPI 通讯之数据输出线 亦可作为异步发送线或 SCI 同步传输的时钟线 亦可作为异步接收线或 SCI 同步传输的数据线
RD0/PSP0 RD1/PSP1 RD2/PSP2 RD3/PSP3 RD4/PSP4 RD5/PSP5 RD6/PSP6 RD7/PSP7	I/O I/O I/O I/O I/O I/O I/O I/O	斯密特/TTL 斯密特/TTL 斯密特/TTL 斯密特/TTL 斯密特/TTL 斯密特/TTL 斯密特/TTL 斯密特/TTL	PORTD 数字 I/O 口，双向可编程，亦可作为并行口，当作为并行口时具有 TTL 输入，作为一般 I/O 口时为斯密特输入。
RE0/RD RE1/WR RE2/CS	I/O I/O I/O	斯密特/TTL 斯密特/TTL 斯密特/TTL	RD: 读 PORTE 数字 I/O 口，双向可编程， WR: 写 亦可作为并行口的控制线。 CS: 片选
VSS	—	—	地
VDD	—	—	电源
NC	—	—	未用

注：1.16C64 和 16C65 除 RC0~RC7 有区别外，其余完全一样，请参考 a。

2.16C63 没有 PORTD 和 PORTE。

b. 16C63/65/65A

引脚名	I/O 特性	电 平	功 能
OSC1/CLKIN	输入	CMOS	振荡输入脚
OSC2/CLKOUT	输出	—	振荡输出脚
MCLR	输入	ST	复位输入脚，低电平有效
RA0 RA1 RA2 RA3 RA4/ T0CKI	I/O I/O I/O I/O I/O	TTL TTL TTL TTL ST	PORTA 数字 I/O 口，双向可编程 亦可作为 TMR0 外部时钟输入

RB0/INT RB1 RB2 RB3 RB4 RB5 RB6 RB7		TTL/ST TTL TTL TTL TTL TTL TTL TTL	PORTB 数字 I/O 口，双向可编程，亦可作为外部中断信号输入（此时为 ST 输入） 具电平变化中断功能 具电平变化中断功能 具电平变化中断功能 具电平变化中断功能
VSS	—	—	地
VDD	—	—	电源

ST：斯密特输入

c. 16C61

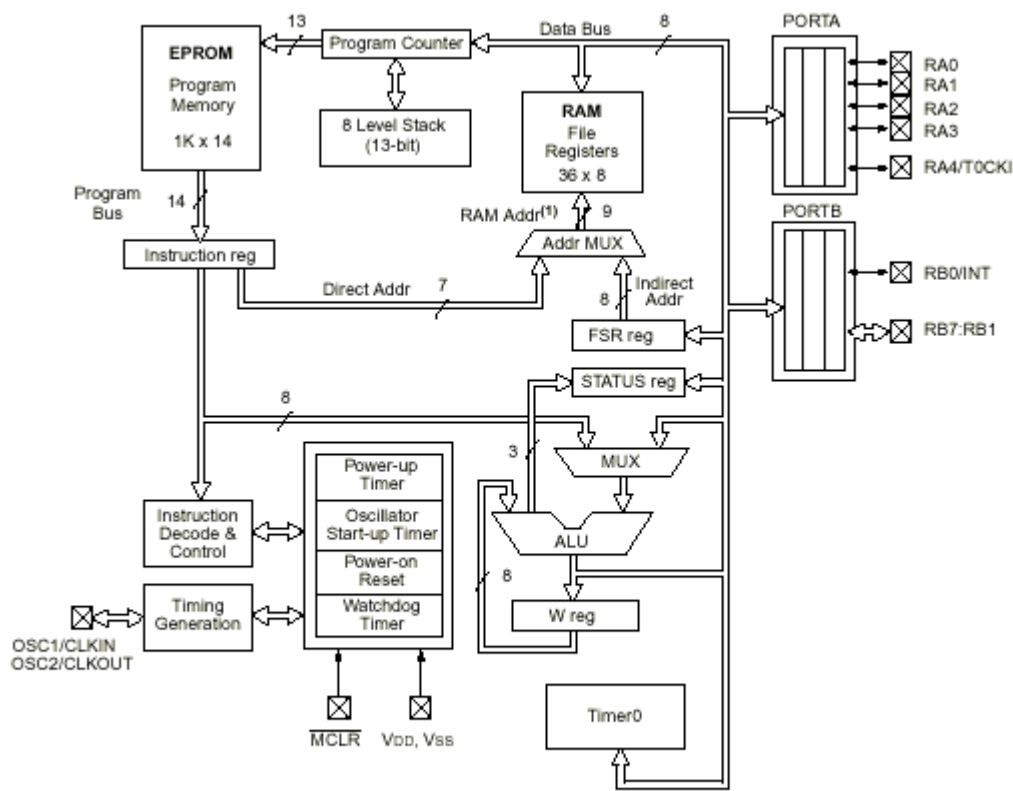
表 1.2 PIC16C6X 引脚功能表

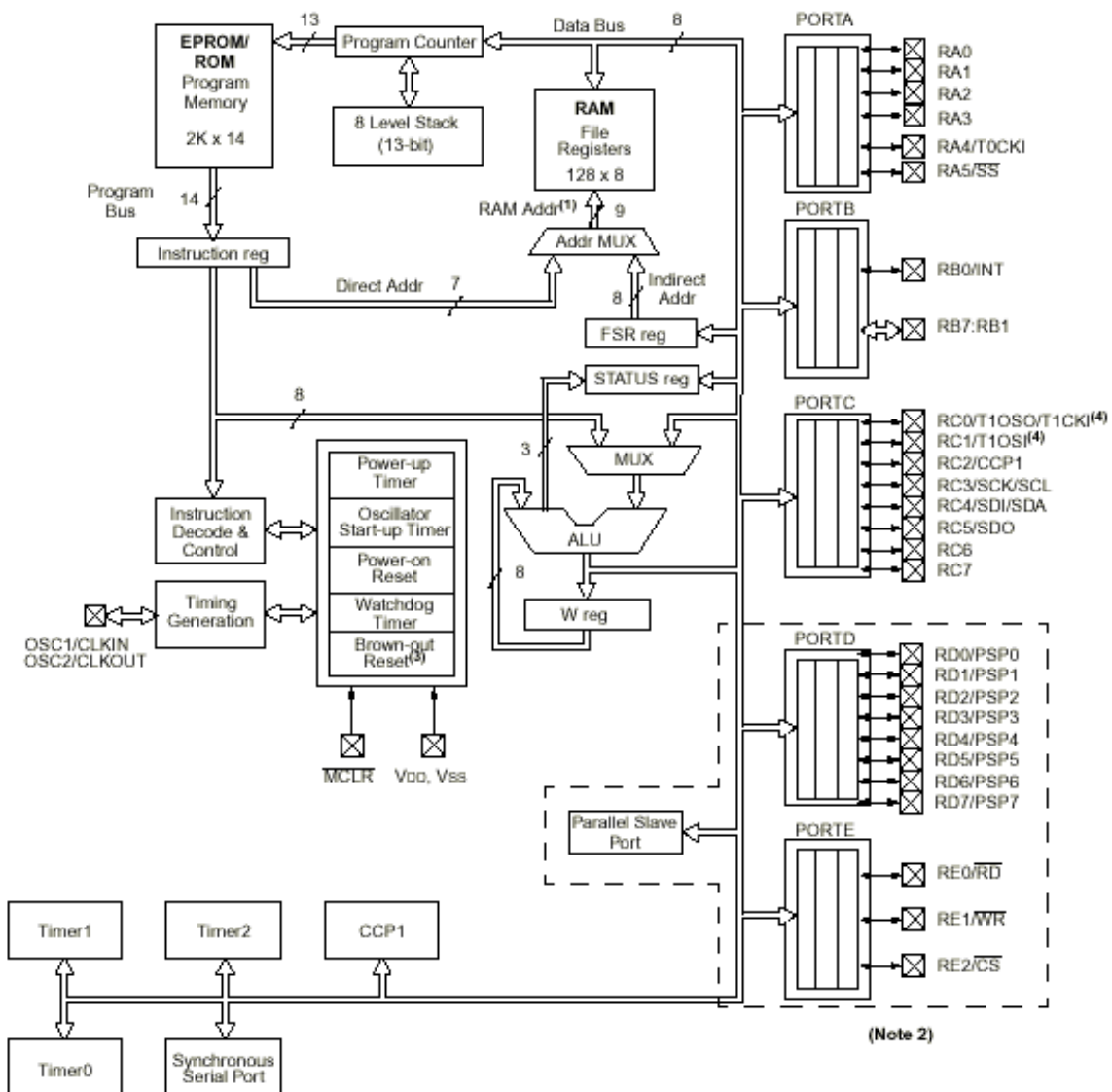
§ 1.4 内部结构

PIC16C6X 内部采用独立分离的 8 位数据总线和 14 位指令总线的“哈佛”结构，它是一种“精简指令集”(RISC) 的 CPU 设计，所以可以达到很高的运行速度。8 位的算术逻辑单元 ALU 可以完成加减、移位和各种布尔逻辑运算，另外它还集成了众多的功能模块如 I/O 、定时器、CCP 模块、并行口、SSP/SCI 串行口以及上电复位电路、看门狗电路、上电/起振延时器等。

在 PIC16C6X 片内带有 1K~4K 的 14 位宽程序存储器（ROM），36~192 个 8 位的数据寄存器（RAM），所有特殊寄存器包括程序计数器、I/O 寄存器等都直接映射到 RAM 单元中，所以程序编码非常简洁高效

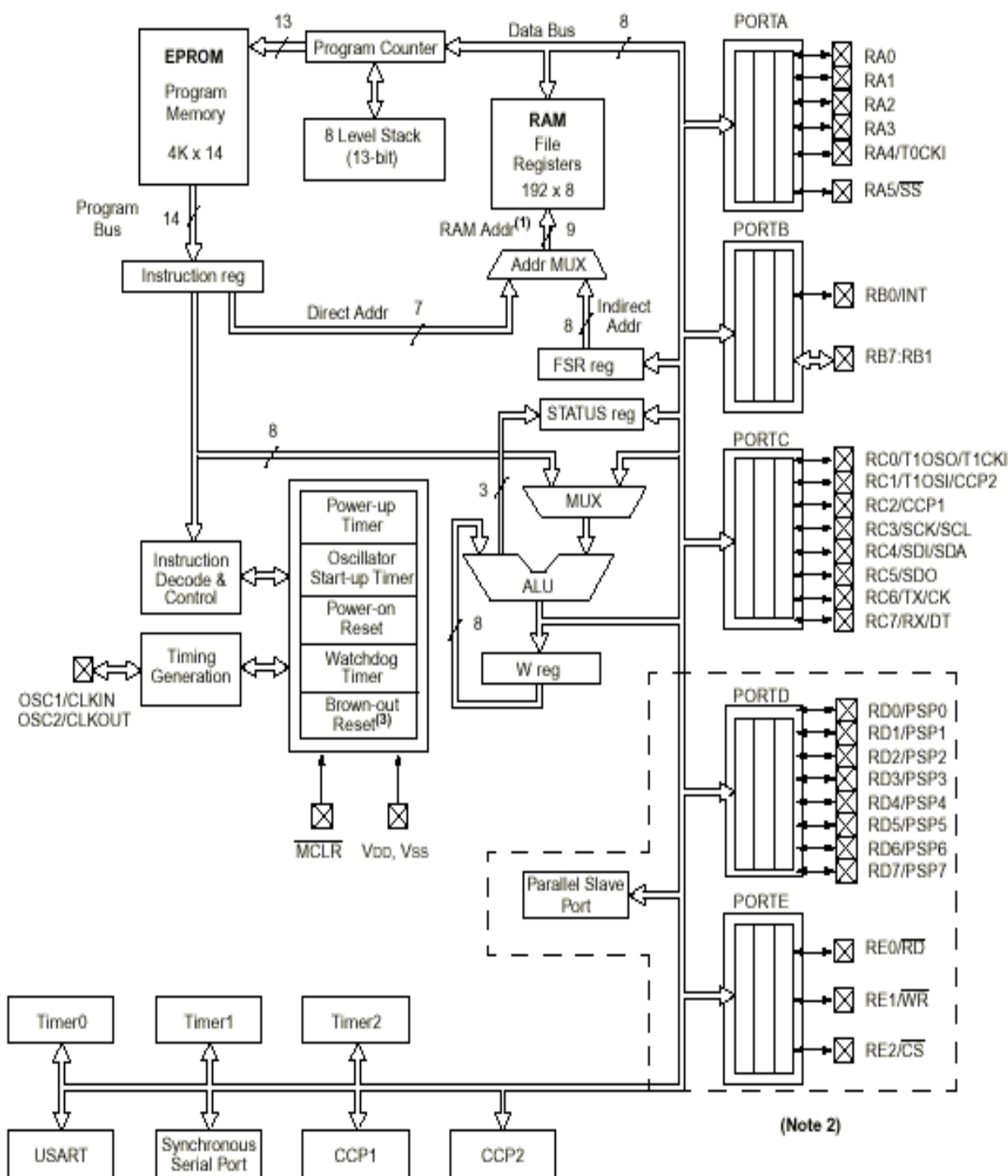
a. PIC16C61 内部结构图





- Note 1: Higher order bits are from the STATUS register.
 Note 2: PORTD, PORTE and the Parallel Slave Port are not available on the PIC16C62/62A/R62.
 Note 3: Brown-out Reset is not available on the PIC16C62/64.
 Note 4: Pin functions T1OSI and T1OSO are swapped on the PIC16C62/64.

b. PIC16C62/62A/64/64A 内部结构图

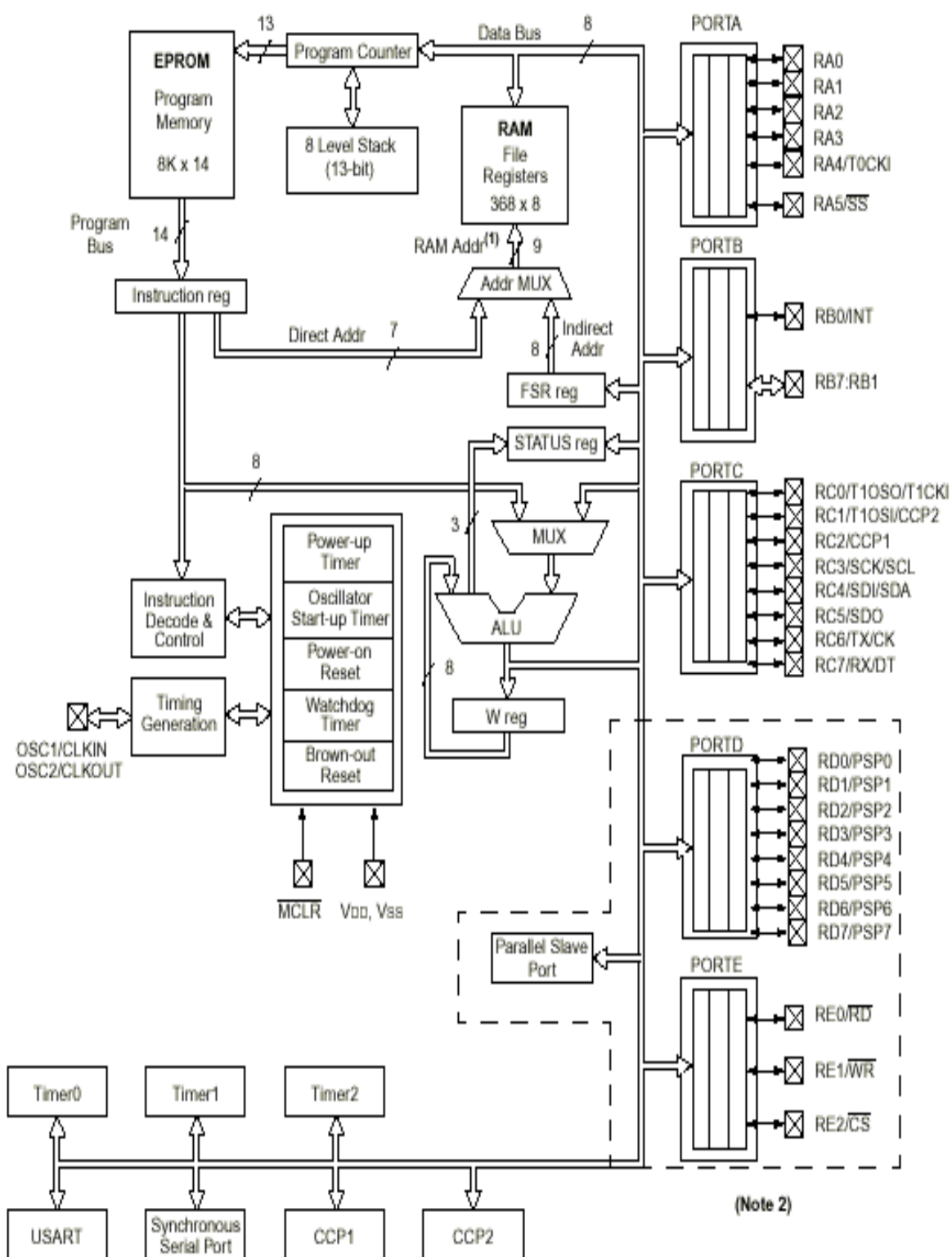


Note 1: Higher order bits are from the STATUS register.

2: PORTD, PORTE and the Parallel Slave Port are not available on the PIC16C63/R63.

3: Brown-out Reset is not available on the PIC16C65.

c. PIC16C63/R63/65/65A/R65 内部结构图



Note 1: Higher order bits are from the STATUS register.

Note 2: PORTD, PORTE and the Parallel Slave Port are not available on the PIC16C66.

d. PIC16C66/67 内部结构图

图 1.2 PIC16C6X 内部结构图

§ 1.5 指令时序和流水作业

从 PIC16CXX 振荡输入端 OSC1 输入的振荡信号经内部 4 分频后形成 4 个不重叠的方波序列 Q1~Q4。程序计数器 PC 随 Q1 节拍增 1，指令代码是在 Q4 节拍取出并放入指令寄存器。指令代码的译码和执行贯穿 Q1~Q4 节拍。对于涉及到 PC 值的指令（如 CALL、GOTO 等），则需要二个指令周期来完成指令的执行，其他指令仅需一个指令周期即可，见下图 1.3。

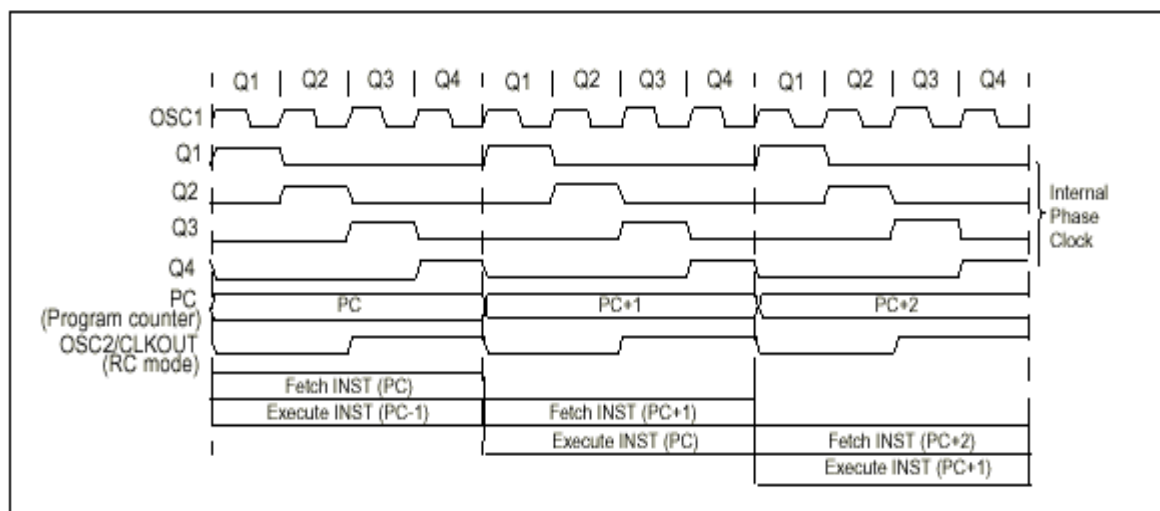


图 1.3 PIC16CXX 指令周期

由于内部采用哈佛结构，使得它在执行一条指令的同时可以取一条指令准备执行，见下图：

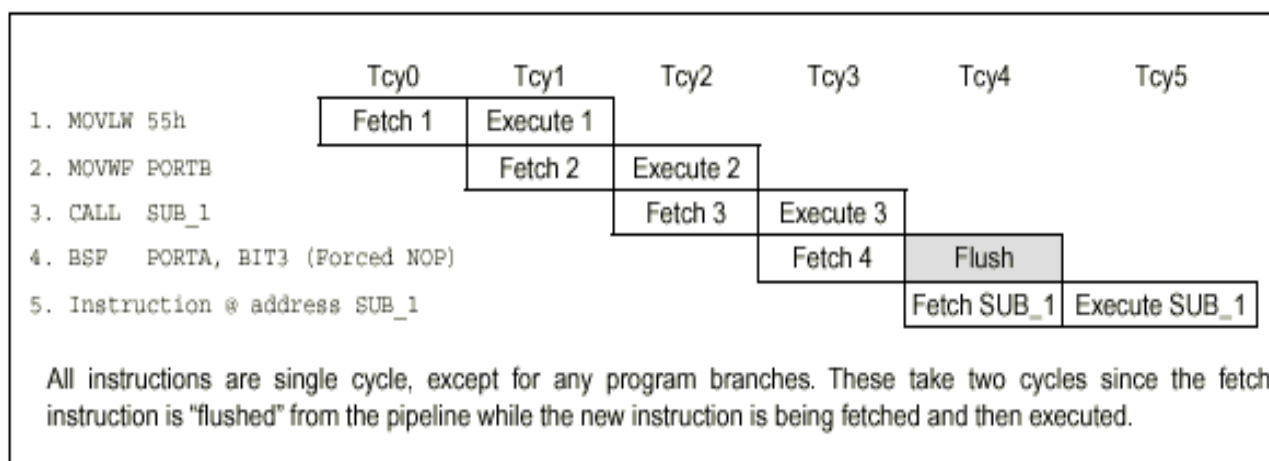
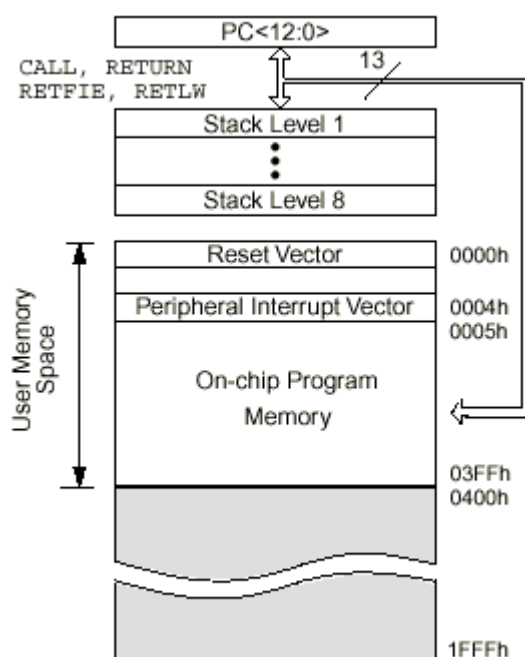


图 1.4 指令流水作业

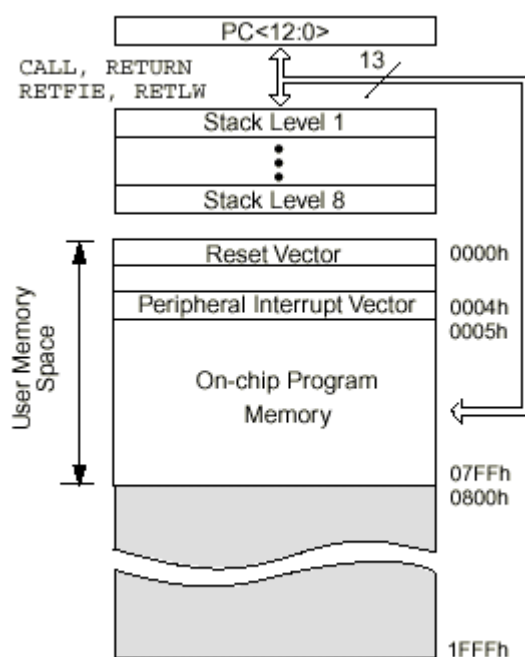
§ 1.6 程序存储器和堆栈

PIC16CXX 有一个 13 位宽的程序计数器 PC，最大可寻址 8K 的程序存储器空间。但是对于 PIC16C65/63，仅使用了头 4K 的空间，PIC16C62/64 仅使用了头 2K，而 PIC16C61 仅使用了头 1K。超出这些空间的指令寻址将导致在物理空间的循环回绕。

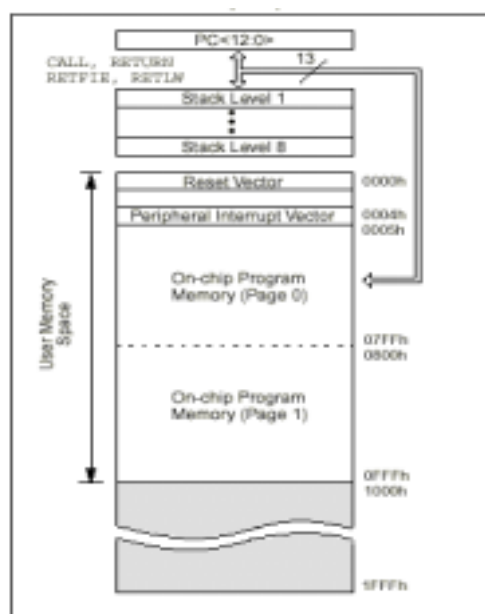
PIC16C6X 的堆栈具有 13×8 的独立空间，不占有程序存储器。



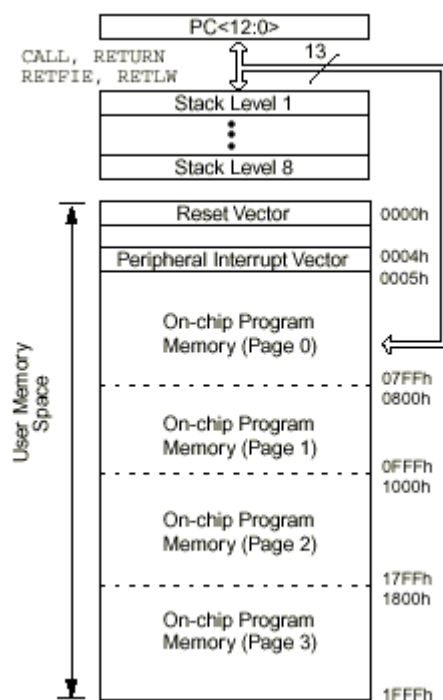
a. 16C61 程序存储器和堆栈



b. 16C62/62A/R62/64/64A/R64 程序存储器和堆栈



c. 16C63/R63/65/65A/R65 程序存储器和堆栈



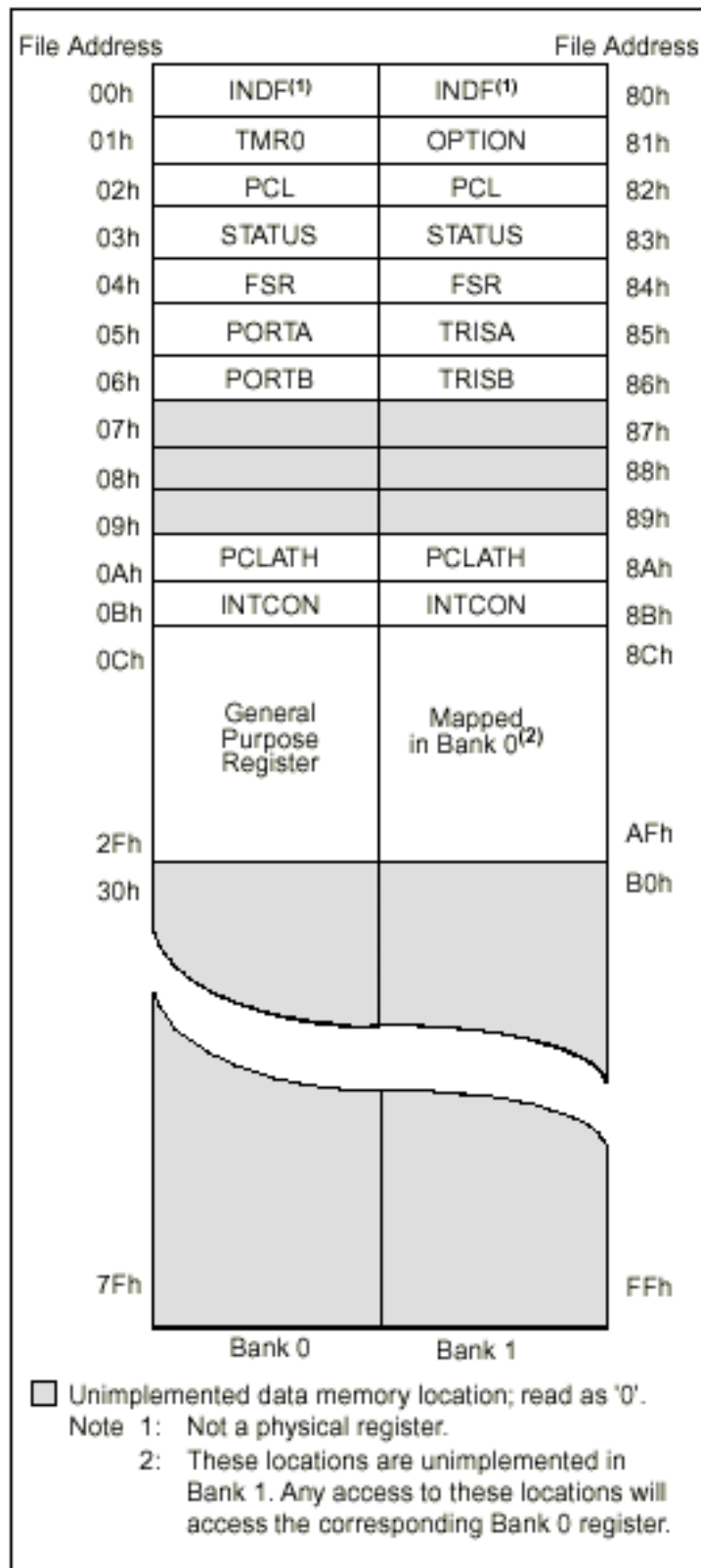
d. 16C66/67 程序存储器和堆栈

图 1.5 PIC16C6X 程序存储器和堆栈

PIC16C63/65/65A 的 4K 程序空间被分为 2 个页面 (page)，各为 2K (0000h~07FFh: page0, 0800h~1FFFh: page1)，请参阅 PCLATH 寄存器的描述。

§ 1.7 数据寄存器

PIC16C6X 数据存储器被分成二个体 (Bank)，包含了特殊功能寄存器和通用寄存器。在状态寄存器 STATUS 中的 RP0 位决定选中 1 体 (RP0=1) 或 0 体 (RP0=0)。每个体最多包含 128 个字节空间。有一些寄存器在 0 体和 1 体之间是相互映射的 (即实际上是同一个物理寄存器)，详见下图。



a. PIC16C61 数据寄存器结构

File Address		File Address	
00h	INDF ⁽¹⁾	INDF ⁽¹⁾	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	PORTC	TRISC	87h
08h	PORTD ⁽²⁾	TRISD ⁽²⁾	88h
09h	PORTE ⁽²⁾	TRISE ⁽²⁾	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	PIR1	PIE1	8Ch
0Dh			8Dh
0Eh	TMR1L	PCON	8Eh
0Fh	TMR1H		8Fh
10h	T1CON		90h
11h	TMR2		91h
12h	T2CON	PR2	92h
13h	SSPBUF	SSPADD	93h
14h	SSPCON	SSPSTAT	94h
15h	CCPR1L		95h
16h	CCPR1H		96h
17h	CCP1CON		97h
18h			98h
1Fh			9Fh
20h		General Purpose Register	A0h
	General Purpose Register		BFh
			C0h
7Fh			FFh

Bank 0 Bank 1

□ Unimplemented data memory location; read as '0'.

Note 1: Not a physical register.

Note 2: PORTD and PORTE are not available on the PIC16C62/62A/R62.

File Address		File Address	
00h	INDF ⁽¹⁾	INDF ⁽¹⁾	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	PORTC	TRISC	87h
08h	PORTD ⁽²⁾	TRISD ⁽²⁾	88h
09h	PORTE ⁽²⁾	TRISE ⁽²⁾	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	PIR1	PIE1	8Ch
0Dh	PIR2	PIE2	8Dh
0Eh	TMR1L	PCON	8Eh
0Fh	TMR1H		8Fh
10h	T1CON		90h
11h	TMR2		91h
12h	T2CON	PR2	92h
13h	SSPBUF	SSPADD	93h
14h	SSPCON	SSPSTAT	94h
15h	CCPR1L		95h
16h	CCPR1H		96h
17h	CCP1CON		97h
18h	RCSTA	TXSTA	98h
19h	TXREG	SPBRG	99h
1Ah	RCREG		9Ah
1Bh	CCPR2L		9Bh
1Ch	CCPR2H		9Ch
1Dh	CCP2CON		9Dh
1Eh			9Eh
1Fh			9Fh
20h	General Purpose Register	General Purpose Register	A0h
7Fh			FFh

Bank 0 Bank 1

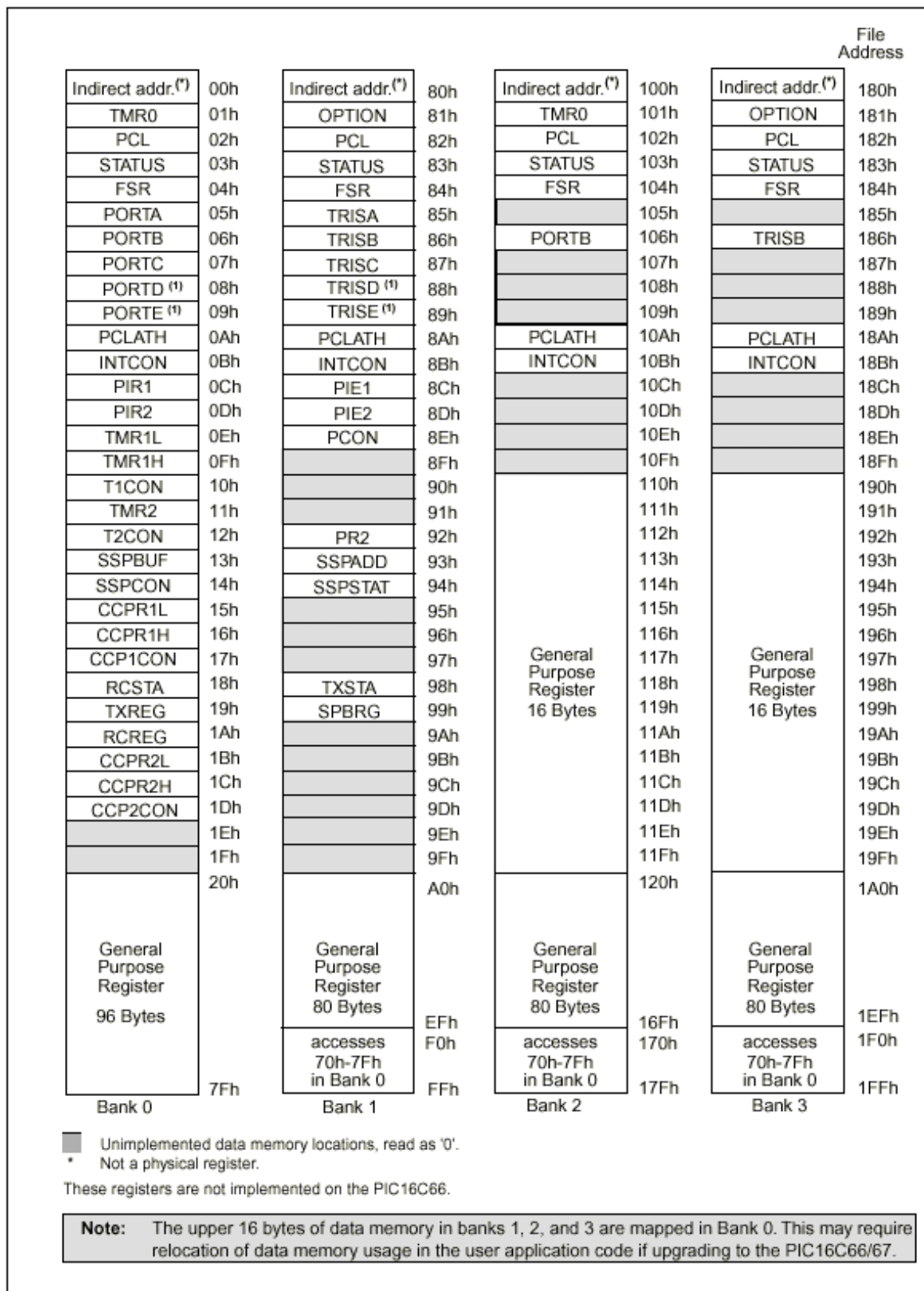
□ Unimplemented data memory location; read as '0'.

Note 1: Not a physical register.

Note 2: PORTD and PORTE are not available on the PIC16C63/R63.

b. PIC16C62/R62/64/64A/R64 数据寄存器结构

c. PIC16C63/R63/65/65A/R65 数据寄存器结构



d. PIC16C66/67 数据寄存器结构

图 1.6 PIC16C6X 数据寄存器结构

§ 1.7.1 通用数据寄存器

通用寄存器即是可用于存储各种数据的寄存器，这些寄存器的内容在单片机上电复位后是随机不定的，在非上

电复位后则保持复位前的内容不变。

PIC16C63/65/65A:20H~7FH (0 体)PIC16C62/62A/64/64A:20H~7FH (0 体)

A0H~FFH (1 体)A0H~BFH (1 体)

PIC16C61:0CH~2FH (0 体)

8CH~AFH (映射到 0 体), 物理上不存在

§ 1.7.2 特殊功能寄存器

特殊功能寄存器被用以控制单片机 CPU 及功能部件的操作。所以一般把它们分成二类：有关 CPU 操作的在本节介绍，另外一类和特定功能部件有关的将在相应的章节介绍。下表列出了 PIC16C6X 各种型号的特殊功能寄存器以及它们在加电复位后和非加电复位后的值。

地 址	名 称	功 能 说 明	上电复位值	其他复位值
Bank0 (0 体)				
00h	INDF	间接寻址逻辑寄存器 (物理上不存在)	0000 0000	0000 0000
01h	TMR0	TIMER0 模块寄存器	xxxx xxxx	uuuu uuuu
02h	PCL	程序计数器 PC 的低 8 位	0000 0000	0000 0000
03h	STATUS	状态寄存器	0001 1xxx	000q quuu
04h	FSR	间接寻址寄存器	xxxx xxxx	uuuu uuuu
05h	PORTA	— — — PORTA 口寄存器	---x xxxx	---u uuuu
06h	PORTB	PORTB 寄存器	xxxx xxxx	uuuu uuuu
07h	—	—	—	—
08h	—	—	—	—
09h	—	—	—	—
0Ah	PCLATH	— — — PC 高 5 位之写入器	---0 0000	---0 0000
0Bh	INTCON	中断控制寄存器	0-00 000x	0-00 000u
Bank1 (1 体)				
80h	INDF	间接寻址逻辑寄存器 (物理上不存在)	0000 0000	0000 0000
81h	OPTION	系统功能定义寄存器	1111 1111	1111 1111
82h	PCL	程序计数器 PC 的低 8 位	0000 0000	0000 0000
83h	STATUS	状态寄存器	0001 1xxx	000q quuu
84h	FSR	间接寻址寄存器	xxxx xxxx	uuuu uuuu
85h	TRISA	— — — PORTA 方向寄存器	---1 1111	---1 1111
86h	TRISB	PORTB 方向寄存器	1111 1111	1111 1111
87h	—	—	—	—
88h	—	—	—	—
89h	—	—	—	—
8Ah	PCLATH	— — — PC 高 5 位之写入器	---0 0000	---0 0000
8Bh	INTCON	中断控制寄存器	0-00 000X	0-00 000u

注：x=不定， u=不变， q=取决于某条件， -=未用 (读为 0)

a. 16C61 特殊功能寄存器表

地 址	名 称	功 能 说 明	上电复位值	其他复位值
Bank0 (0 体)				
00h	INDF	间接寻址逻辑寄存器 (物理上不存在)	0000 0000	0000 0000
01h	TMR0	TIMER0 模块寄存器	xxxx xxxx	uuuu uuuu
02h	PCL	程序计数器 PC 的低 8 位	0000 0000	0000 0000
03h	STATUS	状态寄存器	0001 1xxx	000q quuu
04h	FSR	间接寻址寄存器	xxxx xxxx	uuuu uuuu
05h	PORTA	— — — PORTA 口寄存器	--xx xxxx	--uu uuuu
06h	PORTB	PORTB 口寄存器	xxxx xxxx	uuuu uuuu
07h	PORTC	PORTC 口寄存器	xxxx xxxx	uuuu uuuu
08h	—	—	—	—

09h	—	—				—	—
0Ah	PCLATH	—	—	—	PC 高 5 位写入器	---0 0000	---0 0000
0Bh	INTCON	中断控制寄存器				0000 000x	0000 000u
0Ch	PIR1	某些中断标志位寄存器				00-- 0000	00-- 0000
0Dh	—	—				—	—
0Eh	TMR1L	TIMER1 模块寄存器低 8 位				xxxx xxxx	uuuu uuuu
0Fh	TMR1H	TIMER1 模块寄存器高 8 位				xxxx xxxx	uuuu uuuu
10h	T1CON	TIMER1 控制寄存器				--00 0000	--uu uuuu
11h	TMR2	TIMER2 模块寄存器				xxxx xxxx	uuuu uuuu
12h	T2CON	TIMER2 控制寄存器				-000 0000	-000 0000
13h	SSPBuF	同步串行口（SSP）发送/接收寄存器				xxxx xxxx	uuuu uuuu
14h	SSPCON	同步串行口（SSP）控制寄存器				0000 0000	0000 0000
15h	CCPR1L	CCP1 模块寄存器（低 8 位）				xxxx xxxx	uuuu uuuu
16h	CCPR1H	CCP1 模块寄存器（高 8 位）				xxxx xxxx	uuuu uuuu
17h	CCP1CON	CCP1 控制寄存器				--00 0000	--00 0000
18h~1Fh	—	—				—	—
Bank1（1 体）							
80h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000 0000	0000 0000
81h	OPTION	TIMER0 模块寄存器				xxxx xxxx	uuuu uuuu
82h	PCL	程序计数器 PC 的低 8 位				0000 0000	0000 0000
83h	STATUS	状态寄存器				0001 1xxx	000q quuu
84h	FSR	间接寻址寄存器				xxxx xxxx	uuuu uuuu
85h	TRISA	PORTA 口方向寄存器				--11 1111	--11 1111
86h	TRISB	PORTB 口方向寄存器				1111 1111	1111 1111
87h	TRISC	PORTC 口方向寄存器				1111 1111	1111 1111
88h	—	—				—	—
89h	—	—				—	—
8Ah	PCLATH	—	—	—	PC 高 5 位写入器	---0 0000	---0 0000
8Bh	INTCON	中断控制寄存器				0-00 000x	0-00 000u
8Ch	PIE1	某些中断允许位寄存器				00-- 0000	00-- 0000
8Dh	—	—				—	—
8Eh	PCON	上电复位（POR）和掉电复位（BOR）标志位				---- --qq	---- --uu
8Fh~91h	—	—				—	—
92h	PR2	TIMER2 周期寄存器				1111 1111	1111 1111
93h	SSPADDD	同步串行口（SSP）I ² C 模式下之地址寄存器				0000 0000	0000 0000
94h	SSPSTAT	同步串行口（SSP）之状态寄存器				0000 0000	0000 0000
95h~9Fh	—	—				—	—

注：x=不定， u=不变， q=取决于某条件， -=未用（读为 0）

b. 16C62 特殊功能寄存器表

地 址	名 称	功 能 说 明				上电复位值	其他复位值
Bank0（0 体）							
00h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000 0000	0000 0000
01h	TMR0	TIMER0 模块寄存器				xxxx xxxx	uuuu uuuu
02h	PCL	程序计数器 PC 的低 8 位				0000 0000	0000 0000
03h	STATUS	状态寄存器				0001 1xxx	000q quuu
04h	FSR	间接寻址寄存器				xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	PORTA 口寄存器		--xx xxxx	--uu uuuu
06h	PORTB	PORTB 口寄存器				xxxx xxxx	uuuu uuuu
07h	PORTC	PORTC 口寄存器				xxxx xxxx	uuuu uuuu
08h	—	—				—	—
09h	—	—				—	—
0Ah	PCLATH	—	—	—	PC 高 5 位写入器	---0 0000	---0 0000
0Bh	INTCON	中断控制寄存器				0000 000x	0000 000u

0Ch	PIR1	某些中断标志位寄存器				00-- 0000	00-- 0000
0Dh	PIR2	CCP2 中断标志位				---- --0	---- --0
0Eh	TMR1L	TIMER1 模块寄存器低 8 位				xxxx xxxx	uuuu uuuu
0Fh	TMR1H	TIMER1 模块寄存器高 8 位				xxxx xxxx	uuuu uuuu
10h	T1CON	TIMER1 控制寄存器				--00 0000	--uu uuuu
11h	TMR2	TIMER2 模块寄存器				xxxx xxxx	uuuu uuuu
12h	T2CON	TIMER2 控制寄存器				-000 0000	-000 0000
13h	SSPBuF	同步串行口（SSP）发送/接收寄存器				xxxx xxxx	uuuu uuuu
14h	SSPCON	同步串行口（SSP）控制寄存器				0000 0000	0000 0000
15h	CCPR1L	CCP1 模块寄存器（低 8 位）				xxxx xxxx	uuuu uuuu
16h	CCPR1H	CCP1 模块寄存器（高 8 位）				xxxx xxxx	uuuu uuuu
17h	CCP1CON	CCP1 控制寄存器				--00 0000	--00 0000
18h	RCSTA	SCI 接收状态和控制寄存器				0000 -00x	0000 -00x
19h	TXREG	USART 发送寄存器				0000 0000	0000 0000
1Ah	RCREG	USART 接收寄存器				0000 0000	0000 0000
1Bh	CCPR2L	CCP2 模块寄存器（低 8 位）				xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	CCP2 模块寄存器（高 8 位）				xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	CCP2 控制寄存器				--00 0000	--00 0000
1Eh~1Fh	—	——				—	—
Bank1（1 体）							
80h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000 0000	0000 0000
81h	OPTION	TIMER0 模块寄存器				xxxx xxxx	uuuu uuuu
82h	PCL	程序计数器 PC 的低 8 位				0000 0000	0000 0000
83h	STATUS	状态寄存器				0001 1xxx	000q quuu
84h	FSR	间接寻址寄存器				xxxx xxxx	uuuu uuuu
85h	TRISA	PORTA 口方向寄存器				--11 1111	--11 1111
86h	TRISB	PORTB 口方向寄存器				1111 1111	1111 1111
87h	TRISC	PORTC 口方向寄存器				1111 1111	1111 1111
88h	—	——				—	—
89h	—	——				—	—
8Ah	PCLATH	—	—	—	PC 高 5 位写入器	---0 0000	---0 0000
8Bh	INTCON	中断控制寄存器				0000 000x	0000 000u
8Ch	PIE1	某些中断允许位寄存器				0000 0000	0000 0000
8Dh	PIE2	CCP2 中断使能位寄存器				---- --0	---- --0
8Eh	PCON	上电复位（POR）和掉电复位（BOR）标志位				---- --qq	---- --uu
8Fh~91h	—	——				—	—
92h	PR2	TIMER2 周期寄存器				1111 1111	1111 1111
93h	SSPADD	同步串行口（SSP）I ² C 模式下之地址寄存器				0000 0000	0000 0000
94h	SSPSTAT	同步串行口（SSP）之状态寄存器				0000 0000	0000 0000
95h~97h	—	——				—	—
98h	TXSTA	SCI 发送状态及控制寄存器				0000 -010	0000 -010
99h	SPBRG	波特率发生器寄存器				0000 0000	0000 0000
9Ah~9Fh	—	——				—	—

注：x=不定， u=不变， q=取决于某条件， -=未用（读为 0）

c. 16C63 特殊功能寄存器表

地 址	名 称	功 能 说 明			上电复位值	其他复位值
Bank0（0 体）						
00h	INDF	间接寻址逻辑寄存器（物理上不存在）			0000 0000	0000 0000
01h	TMR0	TIMER0 模块寄存器			xxxx xxxx	uuuu uuuu
02h	PCL	程序计数器 PC 的低 8 位			0000 0000	0000 0000
03h	STATUS	状态寄存器			0001 1xxx	000q quuu
04h	FSR	间接寻址寄存器			xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	PORTA 口寄存器	--xx xxxx	--uu uuuu

06h	PORTB	PORTB 口寄存器					xxxx	xxxx	uuuu	uuuu			
07h	PORTC	PORTC 口寄存器					xxxx	xxxx	uuuu	uuuu			
08h	PORTD	PORTD 口寄存器					xxxx	xxxx	uuuu	uuuu			
09h	PORTE	—	—	—	—	—	PORTE 寄存器		----	-xxx	----	uuuu	
0Ah	PCLATH	—	—	—	PC 高 5 位写入器			---	0	0000	---	0	0000
0Bh	INTCON	中断控制寄存器					0000	000x	0000	000u			
0Ch	PIR1	某些中断标志位寄存器					00--	0000	00--	0000			
0Dh	—	—					—		—				
0Eh	TMR1L	TIMER1 模块寄存器低 8 位					xxxx	xxxx	uuuu	uuuu			
0Fh	TMR1H	TIMER1 模块寄存器高 8 位					xxxx	xxxx	uuuu	uuuu			
10h	T1CON	TIMER1 控制寄存器					--00	0000	--uu	uuuu			
11h	TMR2	TIMER2 模块寄存器					xxxx	xxxx	uuuu	uuuu			
12h	T2CON	TIMER2 控制寄存器					-000	0000	-000	0000			
13h	SSPBuF	同步串行口（SSP）发送/接收寄存器					xxxx	xxxx	uuuu	uuuu			
14h	SSPCON	同步串行口（SSP）控制寄存器					0000	0000	0000	0000			
15h	CCPR1L	CCP1 模块寄存器（低 8 位）					xxxx	xxxx	uuuu	uuuu			
16h	CCPR1H	CCP1 模块寄存器（高 8 位）					xxxx	xxxx	uuuu	uuuu			
17h	CCP1CON	CCP1 控制寄存器					--00	0000	--00	0000			
18h~1Fh	—	—					—		—				
Bank1（1 体）													
80h	INDF	间接寻址逻辑寄存器（物理上不存在）					0000	0000	0000	0000			
81h	OPTION	TIMER0 模块寄存器					xxxx	xxxx	uuuu	uuuu			
82h	PCL	程序计数器 PC 的低 8 位					0000	0000	0000	0000			
83h	STATUS	状态寄存器					0001	1xxx	000q	quuu			
84h	FSR	间接寻址寄存器					xxxx	xxxx	uuuu	uuuu			
85h	TRISA	PORTA 口方向寄存器					--11	1111	--11	1111			
86h	TRISB	PORTB 口方向寄存器					1111	1111	1111	1111			
87h	TRISC	PORTC 口方向寄存器					1111	1111	1111	1111			
88h	TRISD	PORTD 口方向寄存器					1111	1111	1111	1111			
89h	TRISE	PORTE 方向寄存器及 SPI 状态寄存器					0000	-111	0000	-111			
8Ah	PCLATH	—	—	—	PC 高 5 位写入器			---	0	0000	---	0	0000
8Bh	INTCON	中断控制寄存器					0-00	000x	0-00	000u			
8Ch	PIE1	某些中断允许位寄存器					00--	0000	00--	0000			
8Dh	—	—					—		—				
8Eh	PCON	上电复位（POR）和掉电复位（BOR）标志位					----	--qq	----	--uu			
8Fh~91h	—	—					—		—				
92h	PR2	TIMER2 周期寄存器					1111	1111	1111	1111			
93h	SSPADDD	同步串行口（SSP）I ² C 模式下之地址寄存器					0000	0000	0000	0000			
94h	SSPSTAT	同步串行口（SSP）之状态寄存器					0000	0000	0000	0000			
95h~9Fh	—	—					—		—				

注：x=不定， u=不变， q=取决于某条件， -=未用（读为 0）

d. 16C64 特殊功能寄存器表

地 址	名 称	功 能 说 明	上电复位值	其他复位值
Bank0（0 体）				
00h	INDF	间接寻址逻辑寄存器（物理上不存在）	0000 0000	0000 0000
01h	TMR0	TIMER0 模块寄存器	xxxx xxxx	uuuu uuuu
02h	PCL	程序计数器 PC 的低 8 位	0000 0000	0000 0000
03h	STATUS	状态寄存器	0001 1xxx	000q quuu
04h	FSR	间接寻址寄存器	xxxx xxxx	uuuu uuuu
05h	PORTA	— — PORTA 口寄存器	--XX xxxx	--uu uuuu
06h	PORTB	PORTB 口寄存器	xxxx xxxx	uuuu uuuu
07h	PORTC	PORTC 口寄存器	xxxx xxxx	uuuu uuuu
08h	PORTD	PORTD 口寄存器	xxxx xxxx	uuuu uuuu

09h	PORTE	—	—	—	—	—	PORTE 寄存器	---- -xxx	---- uuuu
0Ah	PCLATH	—	—	—	PC 高 5 位写入器			---0 0000	---0 0000
0Bh	INTCON	中断控制寄存器						0000 000x	0000 000u
0Ch	PIR1	某些中断标志位寄存器						00-- 0000	00-- 0000
0Dh	—	—						—	—
0Eh	TMR1L	TIMER1 模块寄存器低 8 位						xxxx xxxx	uuuu uuuu
0Fh	TMR1H	TIMER1 模块寄存器高 8 位						xxxx xxxx	uuuu uuuu
10h	T1CON	TIMER1 控制寄存器						--00 0000	--uu uuuu
11h	TMR2	TIMER2 模块寄存器						xxxx xxxx	uuuu uuuu
12h	T2CON	TIMER2 控制寄存器						-000 0000	-000 0000
13h	SSPBuF	同步串行口（SSP）发送/接收寄存器						xxxx xxxx	uuuu uuuu
14h	SSPCON	同步串行口（SSP）控制寄存器						0000 0000	0000 0000
15h	CCPR1L	CCP1 模块寄存器（低 8 位）						xxxx xxxx	uuuu uuuu
16h	CCPR1H	CCP1 模块寄存器（高 8 位）						xxxx xxxx	uuuu uuuu
17h	CCP1CON	CCP1 控制寄存器						--00 0000	--00 0000
18h	RCSTA	SCI 接收状态和控制寄存器						0000 -00x	0000 -00x
19h	TXREG	USART 发送寄存器						0000 0000	0000 0000
1Ah	RCREG	USART 接收寄存器						0000 0000	0000 0000
1Bh	CCPR2L	CCP2 模块寄存器（低 8 位）						xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	CCP2 模块寄存器（高 8 位）						xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	CCP2 控制寄存器						--00 0000	--00 0000
1Eh~1Fh	—	—						—	—
Bank1（1 体）									
80h	INDF	间接寻址逻辑寄存器（物理上不存在）						0000 0000	0000 0000
81h	OPTION	TIMER0 模块寄存器						xxxx xxxx	uuuu uuuu
82h	PCL	程序计数器 PC 的低 8 位						0000 0000	0000 0000
83h	STATUS	状态寄存器						0001 1xxx	000q quuu
84h	FSR	间接寻址寄存器						xxxx xxxx	uuuu uuuu
85h	TRISA	PORTA 口方向寄存器						--11 1111	--11 1111
86h	TRISB	PORTB 口方向寄存器						1111 1111	1111 1111
87h	TRISC	PORTC 口方向寄存器						1111 1111	1111 1111
88h	TRISD	PORTD 口方向寄存器						1111 1111	1111 1111
89h	TRISE	PORTE 方向寄存器及 SPI 状态寄存器						0000 -111	0000 -111
8Ah	PCLATH	—	—	—	PC 高 5 位写入器			---0 0000	---0 0000
8Bh	INTCON	中断控制寄存器						0-00 000x	0-00 000u
8Ch	PIE1	某些中断允许位寄存器						00-- 0000	00-- 0000
8Dh	PIE2	CCP2 中断使能位寄存器						---- ---0	---- ---0
8Eh	PCON	上电复位（POR）和掉电复位（BOR）标志位						---- --qq	---- --uu
8Fh~91h	—	—						—	—
92h	PR2	TIMER2 周期寄存器						1111 1111	1111 1111
93h	SSPADDD	同步串行口（SSP）I ² C 模式下之地址寄存器						0000 0000	0000 0000
94h	SSPSTAT	同步串行口（SSP）之状态寄存器						0000 0000	0000 0000
95h~97h	—	—						—	—
98h	TXSTA	SCI 发送状态及控制寄存器						0000 -010	0000 -010
99h	SPBRG	波特率发生器寄存器						0000 0000	0000 0000
9Ah~9Fh	—	—						—	—

注：x=不定， u=不变， q=取决于某条件， -=未用（读为 0）

e. 16C65 特殊功能寄存器表

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets ⁽³⁾
Bank 0											
00h ⁽¹⁾	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	0000 0000
01h	TMR0	Timer0 module's register								xxxx xxxx	uuuu uuuu
02h ⁽¹⁾	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	0000 0000
03h ⁽¹⁾	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxx	000q quuu
04h ⁽¹⁾	FSR	Indirect data memory address pointer								xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--xx xxxx	--uu uuuu
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	uuuu uuuu
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	uuuu uuuu
08h ⁽⁵⁾	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	uuuu uuuu
09h ⁽⁵⁾	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
0Ah ^(1,2)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	---0 0000
0Bh ⁽¹⁾	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF ⁽⁶⁾	(4)	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
0Dh	PIR2	—	—	—	—	—	—	—	CCP2IF	---- --0	---- --0
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu
11h	TMR2	Timer2 module's register								0000 0000	0000 0000
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
15h	CCPR1L	Capture/Compare/PWM1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
19h	TXREG	USART Transmit Data Register								0000 0000	0000 0000
1Ah	RCREG	USART Receive Data Register								0000 0000	0000 0000
1Bh	CCPR2L	Capture/Compare/PWM2 (LSB)								xxxx xxxx	uuuu uuuu
1Ch	CCPR2H	Capture/Compare/PWM2 (MSB)								xxxx xxxx	uuuu uuuu
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000
1Eh-1Fh	—	Unimplemented								—	—

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented location read as '0'.

Shaded locations are unimplemented, read as '0'.

Note 1: These registers can be addressed from any bank.

2: The upper byte of the Program Counter (PC) is not directly accessible. PCLATH is a holding register for the PC whose contents are transferred to the upper byte of the program counter. (PC<12:8>)

3: Other (non power-up) resets include external reset through MCLR and the Watchdog Timer reset.

4: PIE1<6> and PIR1<6> are reserved on the PIC16C66/67, always maintain these bits clear.

5: PORTD, PORTE, TRISD, and TRISE are not implemented on the PIC16C66, read as '0'.

6: PSPIF (PIR1<7>) and PSPIE (PIE1<7>) are reserved on the PIC16C66, maintain these bits clear.

f. 16C66/67 特殊功能寄存器表

一、状态寄存器 STATUS

状态寄存器包含了 ALU 的算术状态、复位 (RESET) 状态及数据寄存器体选择位。状态寄存器 STATUS 象其他寄存器一样，可以做为指令的目的操作数 (即被写)，其中的某些状态位根据写操作的结果而设定，但是 TO 和 PD 这两位则不能被写，所以，当你执行一条对 STATUS 操作的指令后，STATUS 的结果可能出乎你的意料。

例： CLRF STATUS ; 清 STATUS 寄存器为零

操作结果是 STATUS=000UU100（U 表示不变），而不是想象中的全零。UU 两位是 PD 和 TO，它们维持不变，而“Z”位由于清零操作被置成“1”，所以如果你想改变 STATUS 的内容，建议你使用位操作指令（BCF、BSF）或传送指令（MOVWF），因为它们的执行不影响任何状态位。下图是 STATUS 中各位的意义。

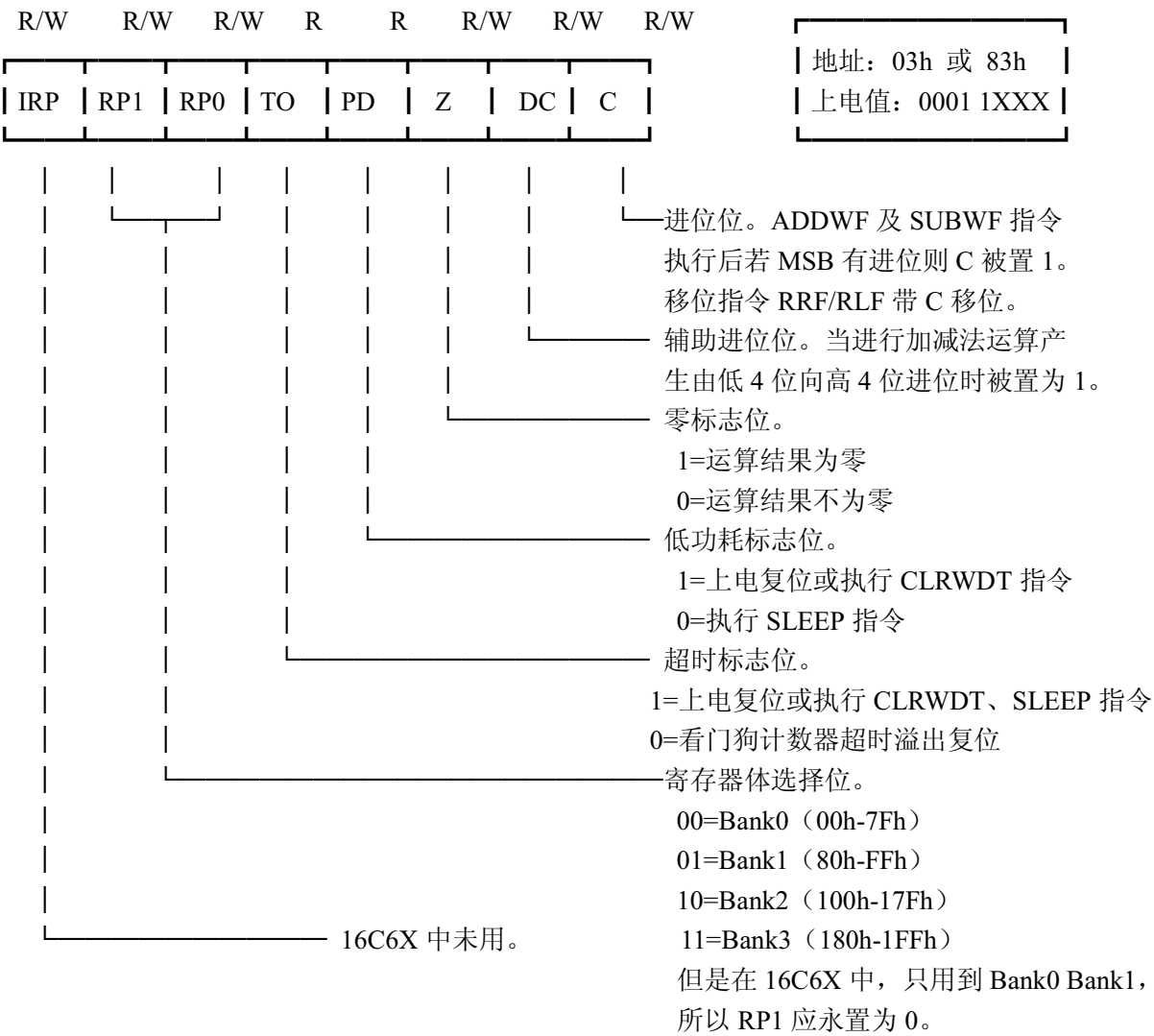


图 1.7 状态寄存器

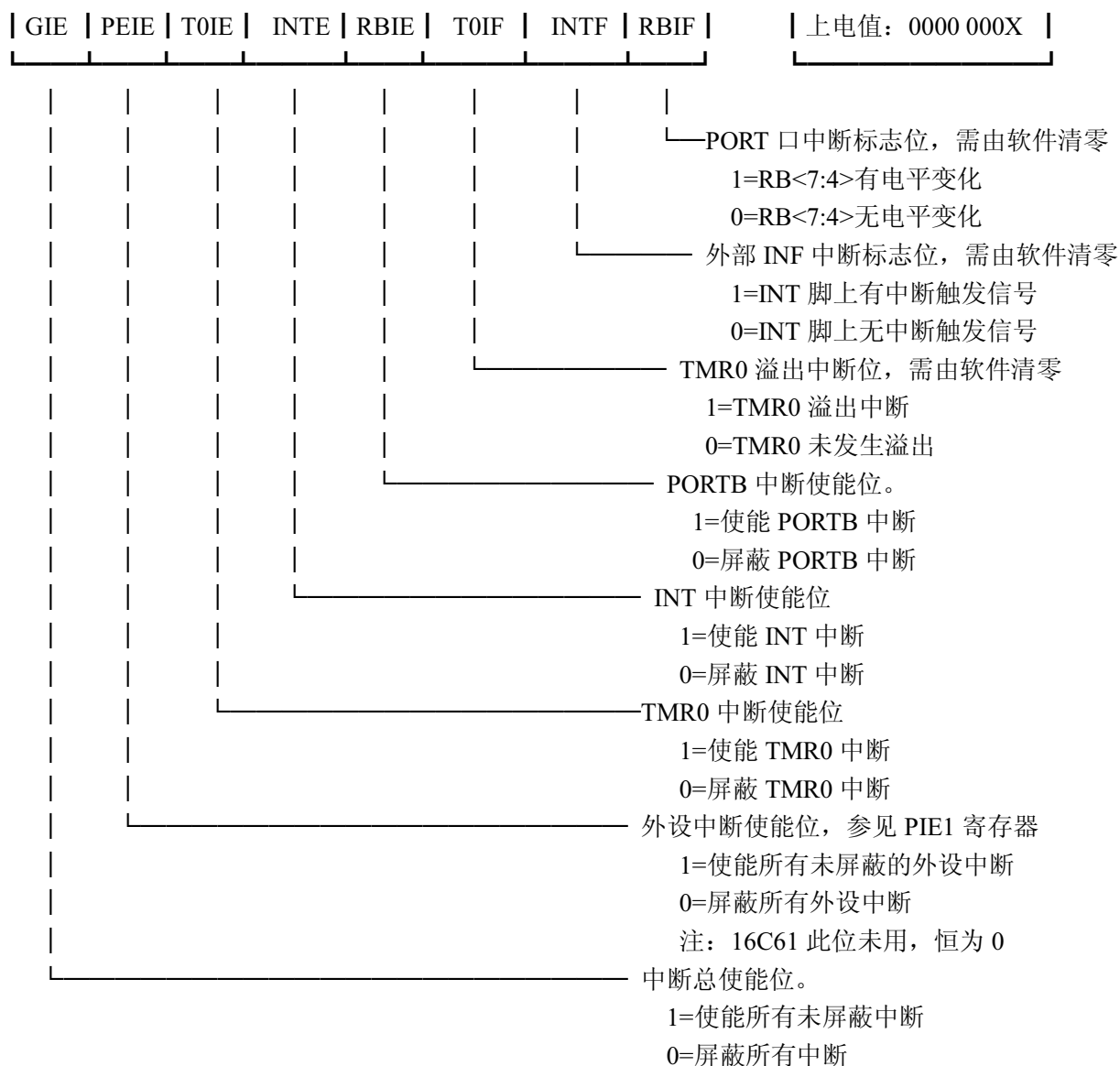
PD 和 TO 两位可用来判断复位的原因：

影响 PD/TO 的事件				RESET 后的 PD、TO 状态		
事 件	TO	PD	注	TO	PD	复 位 源
电源上电	1	1		0	0	WDT 溢时唤醒 SLEEP
WDT 溢出	0	X	不影响 PD	0	1	WDT 溢时（非在 SLEEP 状态）
SLEEP 指令	1	0		1	0	MCLR 加低电平唤醒 SLEEP
CLRWDT 指令	1	1		1	1	电源上电
				X	X	MCLR 端加低电平（非在 SLEEP 状态时）

注：X 表示视情况而定。

二、寄存器 OPTION

OPTION 是一可读/写的寄存器，其包含的控制位如下图所示：



注意：如果中断事件发生，则其相应的标志位都会被置为“1”，最终会不会发生中断，则要看相应的中断允许位是否有效。

涉及型号								
62A	62B	63	63A	64A	65	65A	66	67

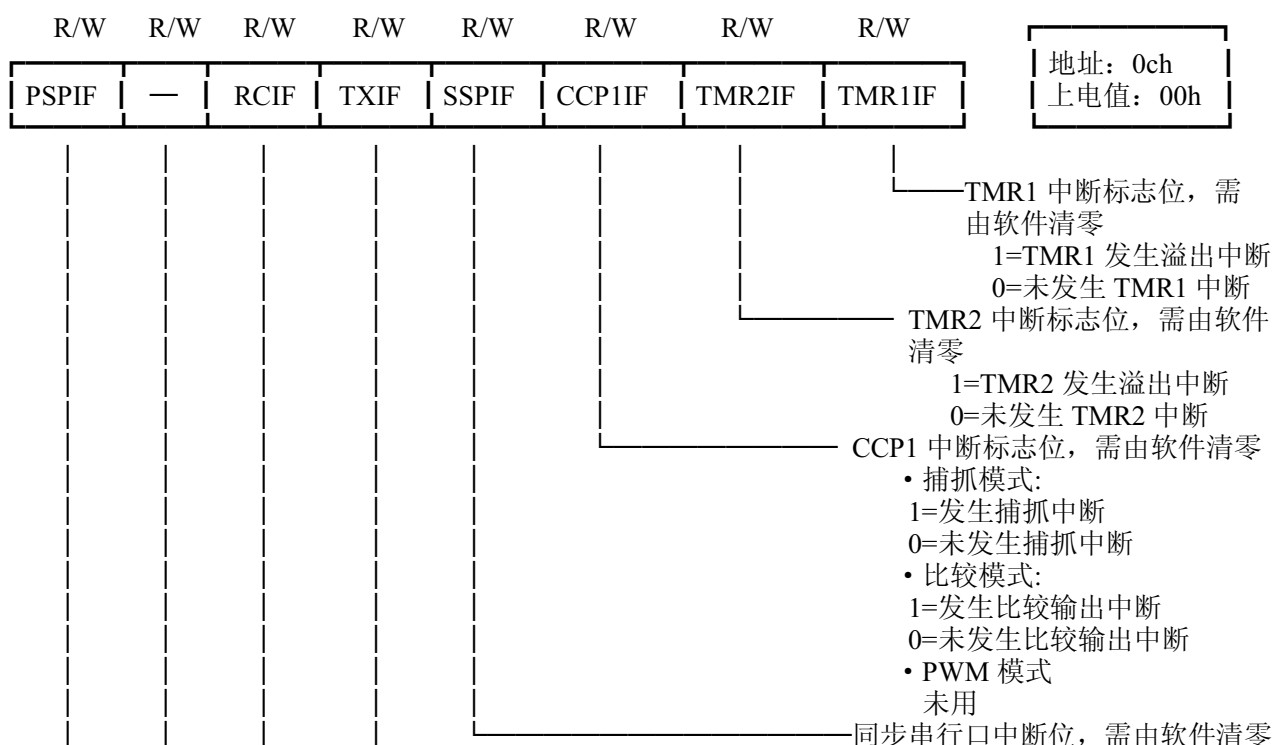


图 1.10 PIE1 寄存器

五、寄存器 PIR1

涉及型号								
62A	62B	63	63A	64A	65	65A	66	67

该寄存器包含外设中断的中断标志位。



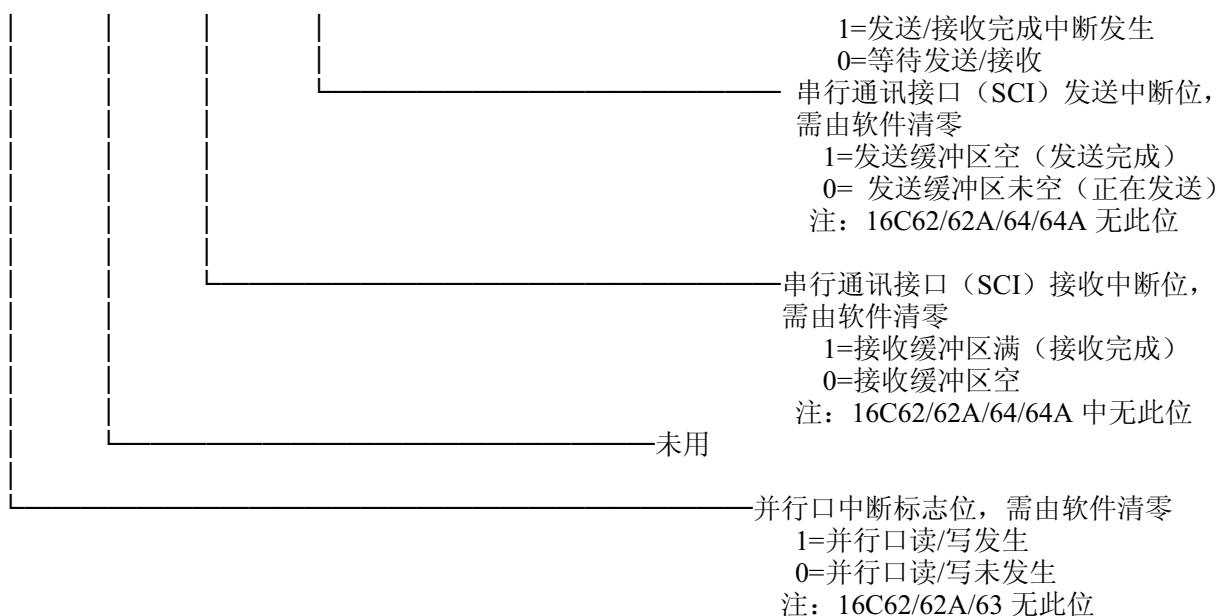


图 1.11 PIR1 寄存器

六、寄存器 PIE2

涉及型号				
63	63A	65	65A	67

该寄存器包含了 CCP2 的中断使能位, 关于 CCP 功能, 请参考有关章节。

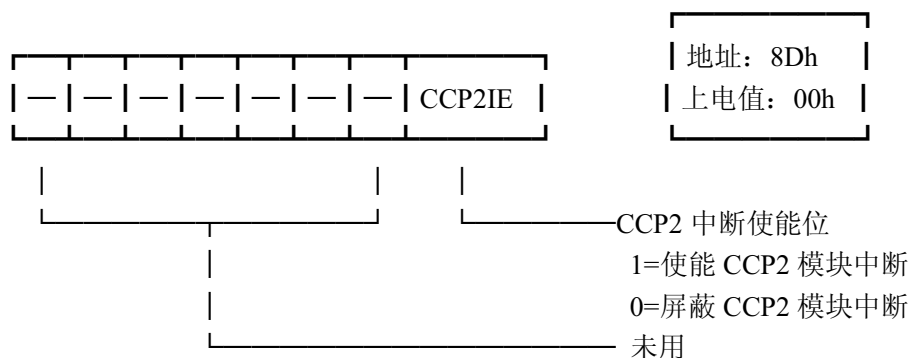
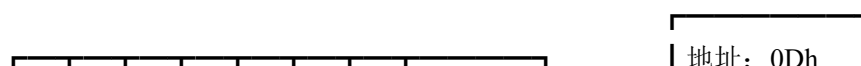


图 1.12 PIE2 寄存器

七、寄存器 PIR2

涉及型号				
63	63A	65	65A	67

该寄存器包含了 CCP2 的中断标志位。



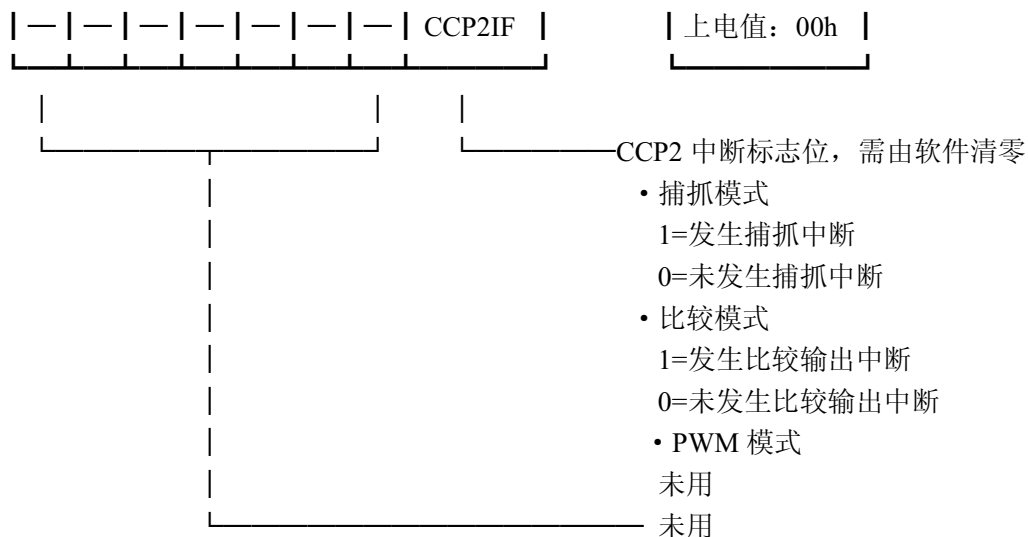
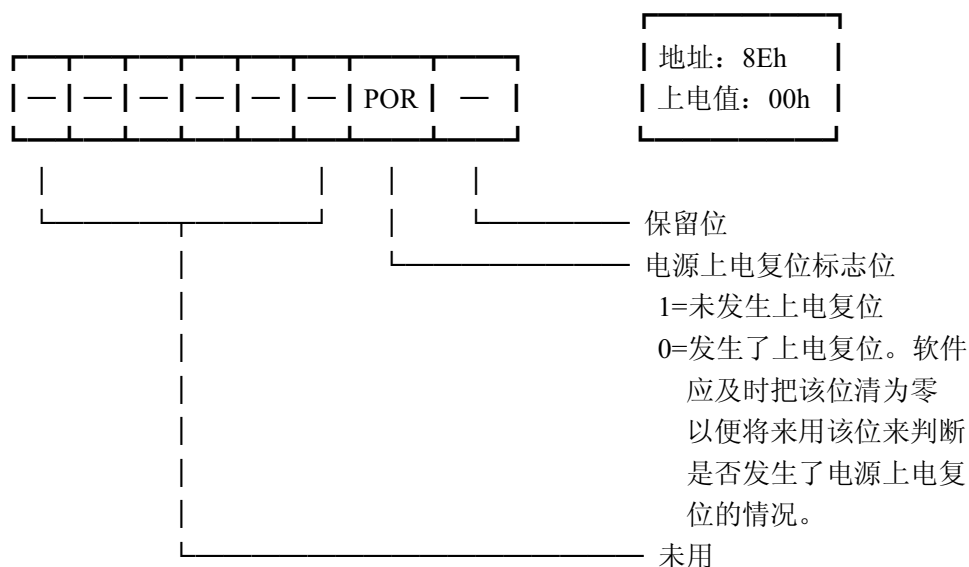


图 1.13 PIR2 寄存器

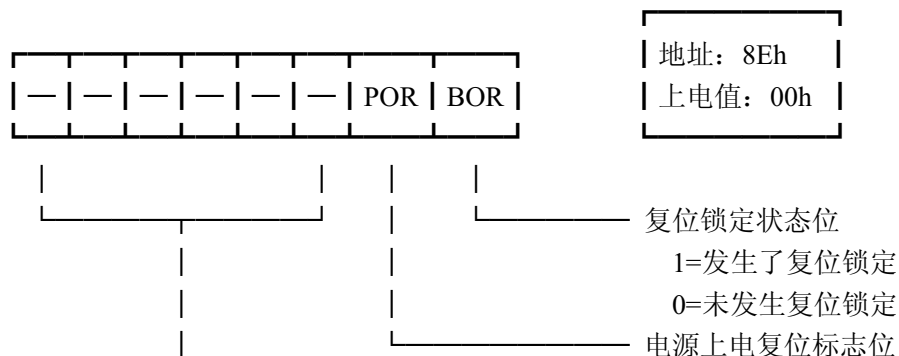
八、寄存器 PCON

涉及型号								
62A	62B	63	63A	64A	65	65A	66	67

该寄存器包含一个称为“上电复位”标志位，用来区别上电复位和别的复位（如 MCLR 拉低或 WDT 超时等造成的复位）。



a. PIC16C62/64/65



在用 MPASM 汇编时，它会发出警告信息。这时可根据提示信息修改编辑汇编程序，作相应的页面预置处理，然后再重新进行汇编。

对于 PIC16C61/62/64，则不存在程序页面选择的问题。

十、寄存器 INDF 和 FSR

INDF 寄存器（地址 0）不是一个物理上存在的寄存器，是一个虚拟的逻辑寄存器，它用以实现间接寻址。当寻址 INDF 时，实际上是访问 FSR 寄存器（地址 4h）内容所指的单元。

以下程序采用间接寻址方式将 20h-2Fh 的寄存器(RAM)单元清零。

```

MOV LW    0X20
MOV F     FSR           ; 起始地址→FSR
NEXT: CLR F     INDF     ; 清 FSR 内容所指的单元（20H-2FH）
      INC F     FSR      ; FSR 内容增 1
      BTFSS    FSR, 4    ; 到 2FH; 否
      GOTO     NEXT      ; 循环
GOON: ...                ; 完成清零
```

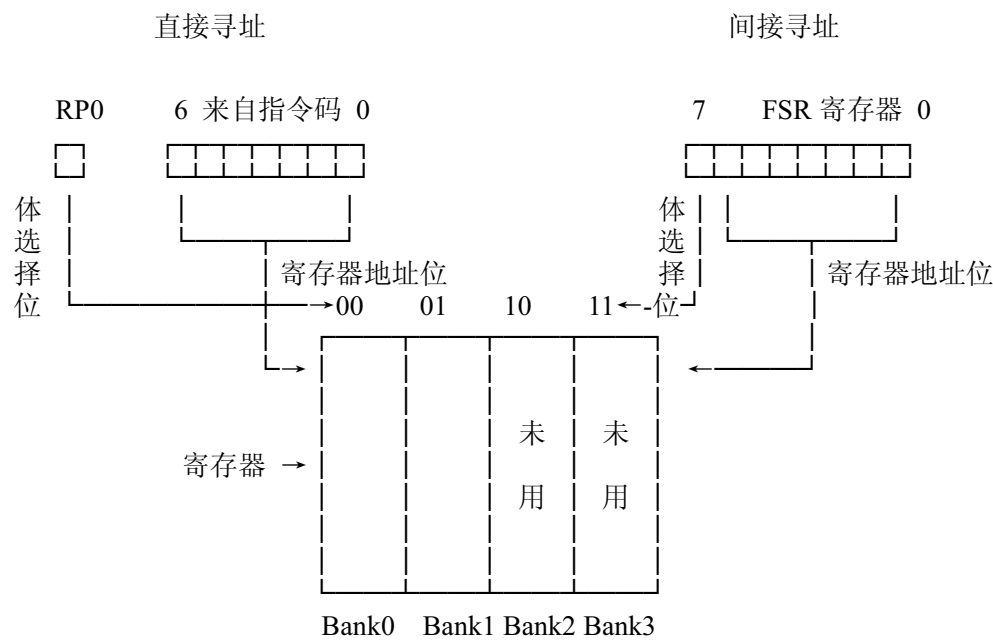


图 1.16 直接或间接寻址方式图

注意:关于寄存器体（Bank0/Bank1）选择，在直接寻址和间接寻址两种方式下，方法是不同的，见上图。

A、直接寻址

此时 Bank0/Bank1 由 STATUS 寄存器中的 RP0:RP1 来选择，见 STATUS 寄存器的描述。

B、间接寻址

此时 Bank0/Bank1 是由间接寻址寄存器 FSR<7>来选择：

FSR<7>=0，Bank0

FSR<7>=1，Bank1

例程：用间接寻址方式往寄存器体 Bank1 中的 A0h-A5h 送数据 88h。

```

MOV LW    0XA0
MOV WF    FSR
LOOP: MOV LW    88h
      MOV WF    INDF
      MOV LW    0XA5
      XOR WF    FSR, 0
```

```

BTFSF    STATUS, Z
GOTO     LOOP
...
...

```

§ 1.8 I/O 口

PIC16C64/65有5个I/O口,PIC16C62/63有3个I/O口,PIC16C61则只有2个I/O 口。 这些I/O管脚有的和某些外部功能部件复用,即可以作为一般的I/O引脚, 也可以作为某些特殊功能的输入/输出。PIC16CXX把I/O口都作为寄存器来处理,编程寻址非常方便。

§ 1.8.1 PORTA和TRISA

对于PIC16C62/63/64/65,PORTA是6位宽的I/O口;而对于PIC16C61而言,PORTA 则只有5位。RA4具有斯密特输入和集电极开路输出,其他所有的RA脚都具有TTL输入和CMOS输出,其方向可由寄存器TRISA相应的位来定义,“1”定义为输入,“0”则为输出,见下图。

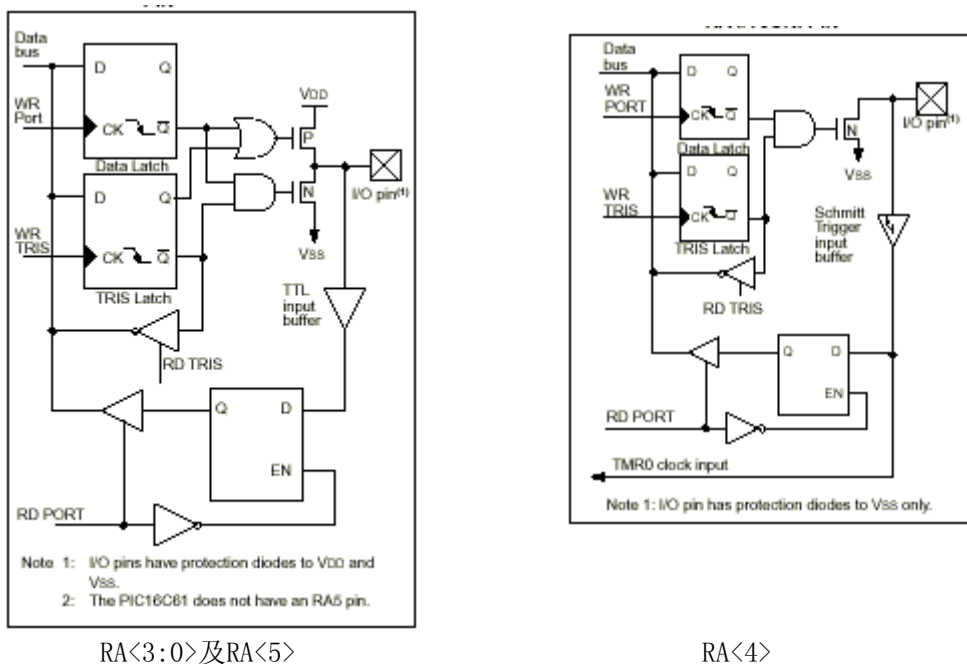


图1.17 PORTA口结构图

I/O脚定义为输入时即为高阻态,定义为输出时则其输出锁存器的输出电平就是I/O 脚输出电平。

读PORTA寄存器的结果是读取I/O管脚上的电平,而写PORTA寄存器的结果是写入I/O 锁存器。所有的 写I/O口的操作都是一个“读入/修改/ 写入”的过程,即先读入I/O脚电平,然后由程序修改(指定一个值),最后再置入I/O锁存器。

例: PORTA始化

```

CLRF    PORTA      ;
BSF     STATUS, RP0 ; 选bank1以便置TRISA寄存器。
MOVLW   0XCF       ; 11001111方向值。
MOVWF   TRISA       ; 置RA<3:0>为输入。
                        RA<5:4>为输出。
                        TRISA<7:6>未用。

```

下表是和PORTA有关的寄存器

寄存器	功 能	地址	上电复位值
PORTA	I/O脚电平或I/O锁器	05h	--XXXXXX

TRISA	PORTA方向寄存器	85h	--111111
-------	------------	-----	----------

注：--未用 X=未定

表1.4 PORTA相关寄存器

§ 1.8.2 PORTB和TRISB

PORTB是一个8位, 双向可编程的I/O口, 相应的方向寄存器为TRISB(地址:86h)。

例: PORTB初始化

```
CLRF    PORTB
BSF     STATUS RPO      ; 选bank1, 准备置TRISB寄存器
MOVLW   0X0F            ; 方向值取00001111
MOVWF   TRISB           ; RB<0:3>为输入
                        ; RB<4:7>为输出
```

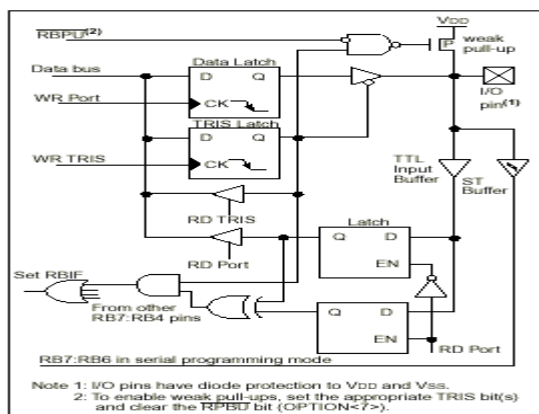
PORTB每个管脚都有可编程之弱上拉, 设置RBPu位(OPTION<7>)可以打开/关闭这些弱上拉。当I/O管脚被设成输出时, 则其弱上拉自动关闭。当芯片上电复位后, RBPu=1, 所有弱上拉被关闭, 可由用户程序将它们打开(置RBPu=0)。

另外PORTB还有一个重要特性, 即RB<7:4>这4根I/O线具有“电平变化中断”功能, 就是当RB<7:4>管脚上的电平发生变化, 可以引起CPU中断, 但仅当I/O 脚定义为输入时这个功能才有效。

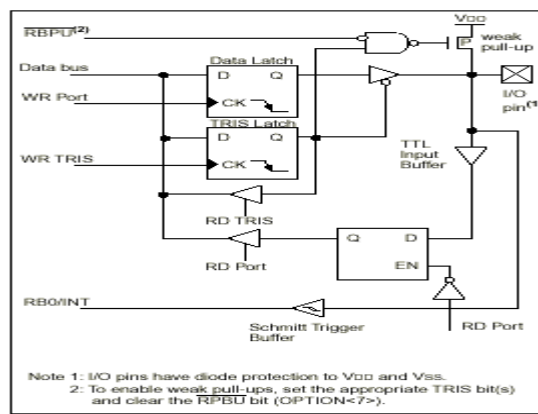
这种中断可以把CPU从“睡眠”(SLEEP)状态中唤醒。

PORTB 的“电平变化中断”以及可编程弱上拉特性使之可以非常方便地构成一个键盘矩阵, 特别是对于那些希望以键盘按动来唤醒CPU工作的设计, 最理想不过, 如手持式仪器, 遥控器, 计算器等。在不用的时候, CPU处于睡眠状态(低功耗), 而当一旦有按键按下, 即可唤醒CPU进行工作。这种设计实例请参考后面的第七章设计范例。

RB<7:4>和RB<3:0>的结构图如下:



a. RB<7:4>



b. RB<3:0>

图1.18 PORTB结构图

下表是和PORTB有关的寄存器

寄存器	功 能	地址	上电复位值
PORTB	I/O脚电平(读), I/O锁 存器(写)	06h	XXXX XXXX
TRISB	PURTb口方向寄存器 “1”=输入 “0”=输出	86h	1111 1111

CPTION	弱上拉开/关控制	81h	1111 1111
--------	----------	-----	-----------

注：X=不定

表1.5 PORTB相关寄存器

§ 1.8.3 PORTC和TRISC

涉及型号						
62	62A	63	64	64A	65	65A

PORTC是一个8位双向I/O口。它的方向由TRISC定义。PORTC 口有输入斯密特触发器缓冲。如下图所示：

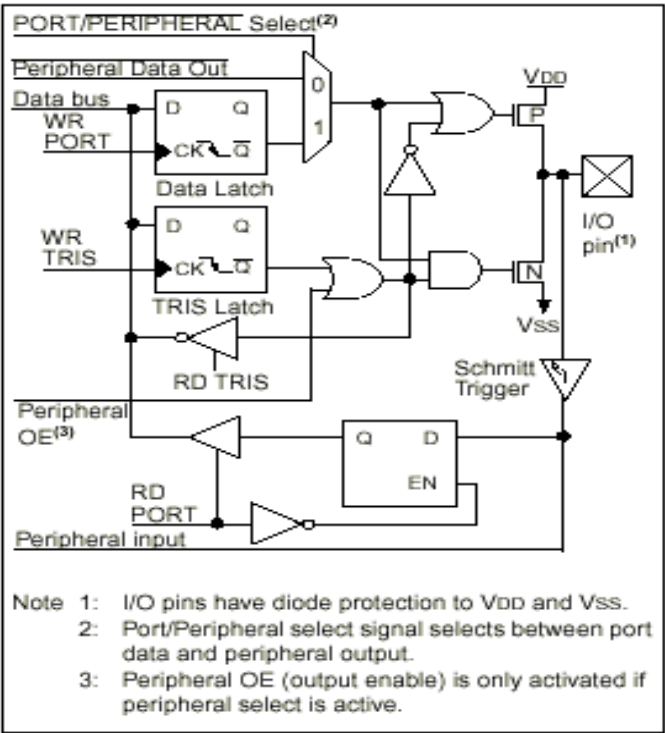


图1.19 PORTC结构图

例：CLRF PORTC
 BSF STATUS, RP0 ; 准备置TRISC
 MOVLW 0X0F ; 0000 1111
 MOVWF TRISC ; 输出输入
 和PORTC有关的寄存器如下：

寄存器	功 能	地址	上电复位值
PORTC	I/O脚电平(读), I/O锁存器(写)	07h	XXXX XXXX
TRISC	PORTC方向控制。 “1”=输入 “0”=输出	87h	1111 1111

注：X=不定

表1.6 PORTC相关寄存器

PORTC的I/O脚还可以作为其他一些外部功能的引脚, 这些将在相应的章节介绍。

§ 1.8.4 PORTD和TRISD

作为并行口控制线,这在 § 1. 8. 7介绍, 请参考。

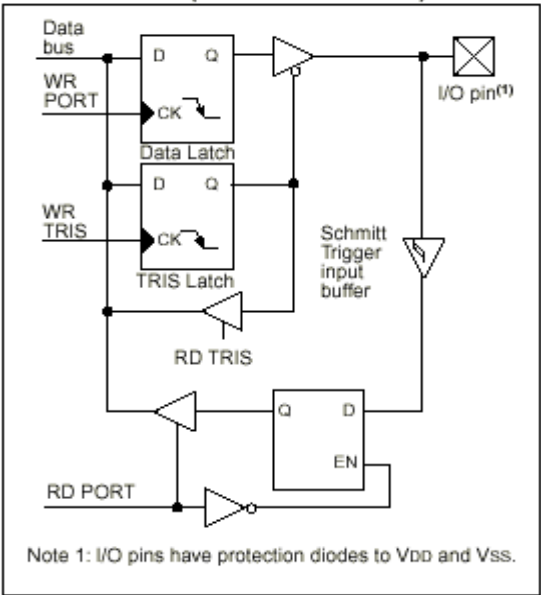
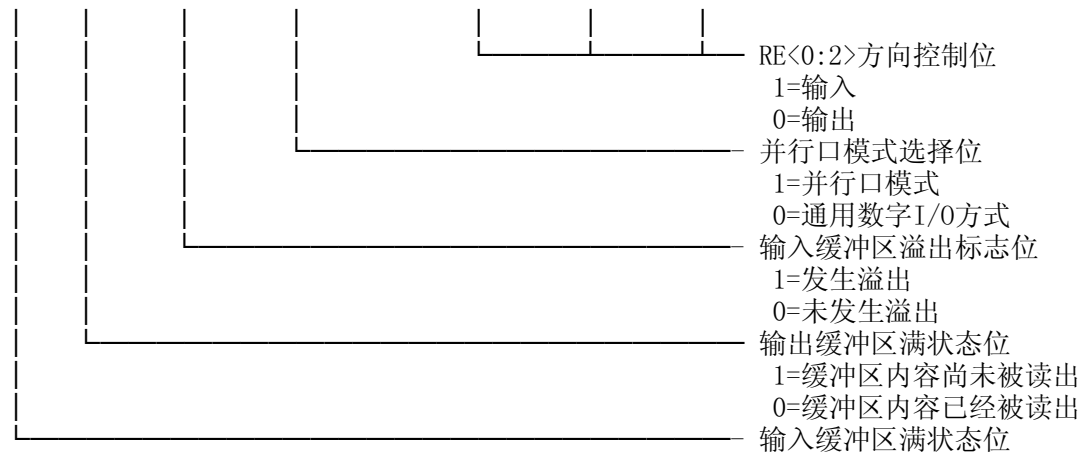


图1. 21 PORTC结构图

TRISE除三位<0:2>用于PORTC方向控制外, 另外还有五位<3:7>用于其他功能的控制, 见下图。

地 址:89H
上电值:0000 -111

R	R	R/W	R/W	U	R/W	R/W	R/W
IBF	OBF	IBOV	PSPMODE		TRISE2	TRISE1	TRISE0



1=输入缓冲区已有数据
0=输入缓冲区尚无数据

图1. 22 TRISE寄存器

和PORTC有关的寄存器如下表：

寄存器	功 能	地址	上电复位值
PORTC	I/O脚电平(读), I/O锁存器(写)	09h	---- -XXX
TRISE	PORTC方向控制。“1”=输入 “0”=输出	88h	0000 -111

注：X=不定

表1. 8 PORTC相关寄存器

§ 1.8.6 I/O编程注意事项

I/O口使用注意事项:

一、I/O方向转置的问题

有的时候我们可能需要一个I/O口一会做输入,一会又做输出,这就是I/O方向的转置。在编写这种I/O转置程序时你必须注意,有些指令写I/O口时先从I/O读入其状态,修改后再重新写回I/O当前状态读入CPU,执行位操作后再将结果写回去覆盖原来的内容(输出的结果放在I/O口的数据锁存器)。举个例说:BSF 6,5这条指令的目的是要把B口的第6位置为高电平“1”。执行这条指令时,先把整个B口当前的状态内容读入到CPU,把第6位置成“1”后再把结果(8个位)重新输出到B口。如果B口中的有一个I/O腿是需要方向转置的(比如说bit1),而这时是处于输入态,那么B口的状态值重新写入后,B口的数据锁存器1(见图1.15相对于B口bit1的锁存器)的锁存值就是当前B口Bit1的状态。这可能与先前Bit1作为输出时所锁存的值不同,所以当Bit1再转置成输出态时,出现在bit1端的状态就可能和先前的输出态不同了。

二、I/O的“线或”和“线与”

从图1.15看出:PIC I/O脚输出电路为CMOS互补推挽输出电路。因此与其他这类电路一样,当某个PIC I/O脚设置为输出状态时,不能与其他电路的输出脚接成“线或”或“线与”的形式,否则可能引起输出电流过载,烧坏PIC。如需要与其他电路接成“线或”电路时,PIC I/O必须置于“0”状态或输入状态并外接上拉电阻。如需要接成“线与”电路时,则PIC I/O必须置于“1”状态或输入状态,并外接下拉电阻,电阻的阻值根据实际电路和PIC I/O脚最大电流来选定。

三、I/O口的连续操作

一条写I/O的指令,对I/O真正写操作是发生在指令的后半周期。而读I/O的指令却是在指令的周期开始就读取I/O端状态。所以当你连续对一个I/O端写入再读出时,必须要让I/O端上的写入电平有一个稳定的时间,否则读入的可能是前一个状态,而不是最新的状态值。一般推荐在两条连续的写,读指令间至少加一条NOP指令。

例: MOVWF 6 ; 写I/O
NOP ; 稳定I/O电平
MOVF 6,W ; 读I/O

四、噪声环境下的I/O操作

在噪声环境下(如静电火花),I/O控制寄存器可能因受干扰而变化。比如I/O口可能会从输入态自己变成输出态,对于这种情形,WDT也是无法检测出来的。因此如果你的应用环境是较恶劣的,建议你每隔一定的间隔,都重新定义一下I/O控制寄存器。

§ 1.8.7 并行口

涉及型号			
64	64A	65	65A

在PIC64/65中,PORTD除作为一般的双向I/O口,还可以用作8位的并行口。把PSPMODE位(TRISE<4>)置为“1”,则PORTD工作在并行口方式,它可以由外部控制器进行异步读写。并行口PORTD可直接和外部的8位微控制器的数据线相连,这时该微控制器即可把PORTD作为一个数据锁存器来读写。PIC64/65的这个特性使它可以方便地和别的8位微控制器进行并行通讯。

当PSPMODE置为“1”后,PORTD的三根I/O线即分别成为并行口PORTD的三根控制线:RE0作为读控制线RD,RE1作为写控制线WE,RE2则作为片选线CS。这时TRISE<0:2>还必须置为“111”以使RE0~RE2成为输入线。

并行口中实际上有二个8位锁存器,分别对应于数据进(写入PIC)和数据出(从PIC读出),

如图示。

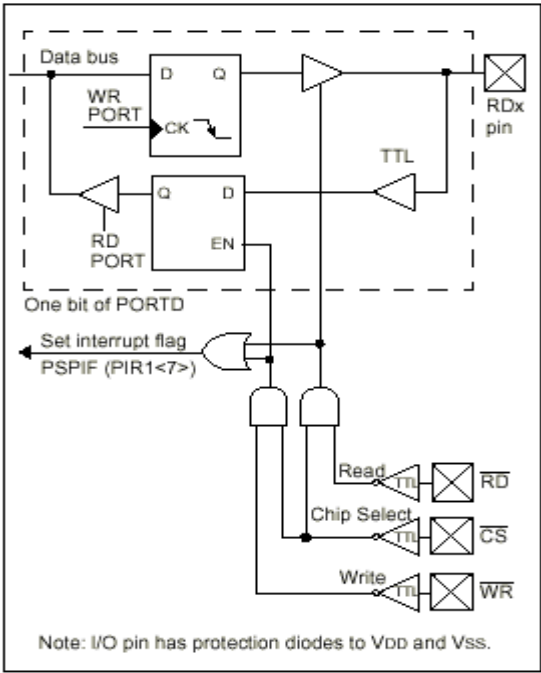


图1.23 并行口结构

对于PIC单片机这方面来说,当写操作时,是把8位数据写入到PORTD数据锁存器,读操作则是从PORTD管脚锁存器读入数据。但这二个物理上不同的寄存器在逻辑地址上是一样的,实际的操作对象则取决于RD(读)和WR(写)信号线的状态。在PORTD作为并行口操作时,TRISD对PORTD的方向控制则不再起作用,此时由外部的微控制器通过RD和WR来控制数据的方向。

当PORTD从外部接收到一个字节数据并等待PIC的CPU来读取时,IFB标志位(TRISE<7>)被置为“1”,一旦这个数据被CPU读取后,IFB位即清为“0”。IFB是一个只读状态位,当数据被CPU写入到PORTD等待外部控制器读取时,IOBF标志位(TRISE<6>)被硬件置为“1”,一旦这个数据被外部控制器读取后,IOBF位即被清为“0”。当PORTD从外部控制器接收到一个数据,CPU还未读取前外部控制器又置入一个数据,则原先的数据会被新数据复盖,并且IOBV标志位(TRISE<5>)被置为“1”。IOBV是一个可读/写的位,必须由程序方能将其清为“0”。当PORTD不作为并行口时(PSPMODEQ=“0”),IFB和IOBF都保持为“0”。此时如果IOBV之前被置为“1”,则应由软件将其清为“0”。当PORTD并行口完成一个读/写操作后,就有一个中断请求产生。这时其相应的中断标志位PSPIF(PIR1<7>)被置为“1”,PSPIF位须由软件来清“0”。PSPIE(PIE1<7>)为该中断的屏蔽位,置PSPIE=“0”可以屏蔽该中断。

下表为和并行口有关的寄存器

寄存器	功 能	地址	上电复位值
PORTD	8位并行口寄存器	08h	XXXX XXXX
PORTE	并行口控制线RD, WR, CS	09h	---- -XXX
TRISE	并行口功能控制寄存器	89h	0000 -111
PIR1	并行口中断标志寄存器	0ch	0000 0000
PIE1	并行口中断使能/屏蔽寄存器	8ch	0000 0000

注: -=未用, X=不定

表1.9 并行口相关寄存器

§ 1.9 定时器/计数器

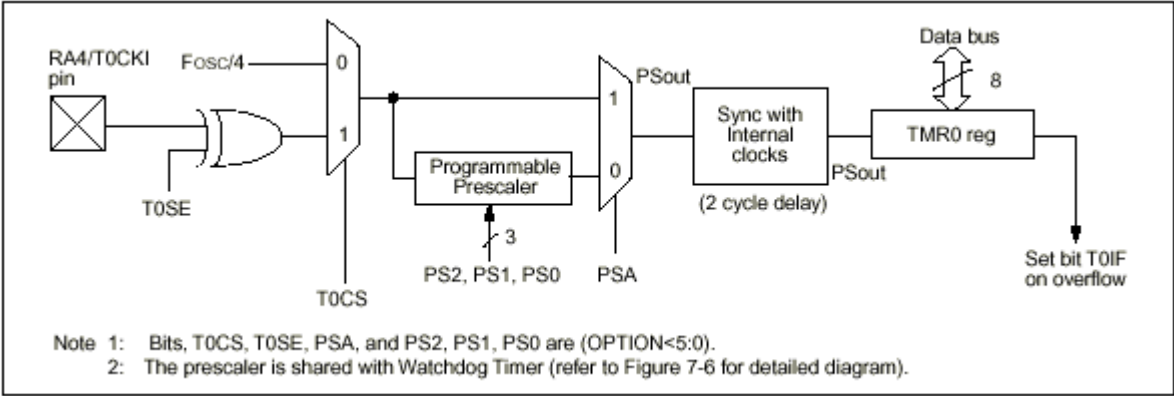
PIC16C61有一个定时器TIMER0,其余PIC16C6X则有3个定时器TIMER0, TIMER1, TIMER2。每个定时器都可以产生中断请求,另外定时器也和一些别的模块配合来完成捕捉/PWM等功能。

§ 1.9.1 TIMER0定时器/计数器

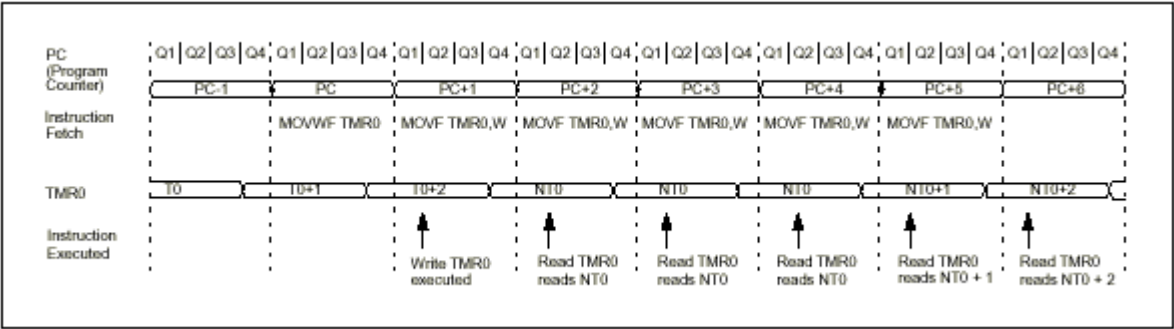
PIC16C6X都有定时器/计数器TIMER0(在PIC15C5X中被称为RTCC), 具有如下特点:

- 8位, 可循环溢出的定时器/计数器。
- 用户可读/写。
- 具8位可编程预分频器。
- 内部/外部信号触发。使用外部信号时可由程序上升/下降沿触发。
- 溢出中断 (从FFh-00h)

下面是TMR0的方块图及时序:



a. TMR0逻辑方块图



b. TMR0时序图

图1. 24 TMR0逻辑方块图及时序

T0CS (OPTION<5>)决定TMR0的工作模式:

T0CS	TMR0工作模式	工作方式
0	定时器	每个指令时钟增1(无预分频器时)
1	计数器	每个RA4/T0CK1脚电平上升/下降沿增1

表1. 10 TMR0工作模式

当工作在计数器模式时, T0SE (OPTION<4>) 决定外部输入信号 (RA4/T0CK1) 的触发沿: T0SE=1, 下降沿触发; T0SE=0, 上升沿触发。

TMR0和看门狗WDT计时器使用同一个分频器, 当注意该预分频器不能同时为TMR0和WDT所用, 在同一时刻只能为其中之一所用。见下表:

PSA (OPTION <3>)	预分频器分配对象
0	TMR0
1	WDT

表1. 11 预分频器分配表

未分配到分频倍数的将以固定的1:1分频率工作. 参见OPTION寄存器描述。

一、TMR0中断

当TMR0发生溢出 (从FFh-00h), 将会置T0IF (INTCON <2>) 置“1”, 同时产生中断请求。在退出TMR0中断服务程序之前, 应用软件把T0IF清为“0”, 以免发生重复中断。

由于CPU处于“睡眠” (SLEEP) 时TMR0不能工作, 所以TMR0中断不能用来唤醒处于“睡眠”中的CPU。

通过设置T0IE（INTCON〈5〉）为“0”可以屏蔽TMR0中断。

二、TMR0预分频器

如前面所述，TMR0可以由软件设置来获得8位预分频器，即一个从1-256的分频倍率。控制位PS2-PS0（OPTION〈2:0〉）给出8种不同的分频倍率，详见OPTION寄存器描述。

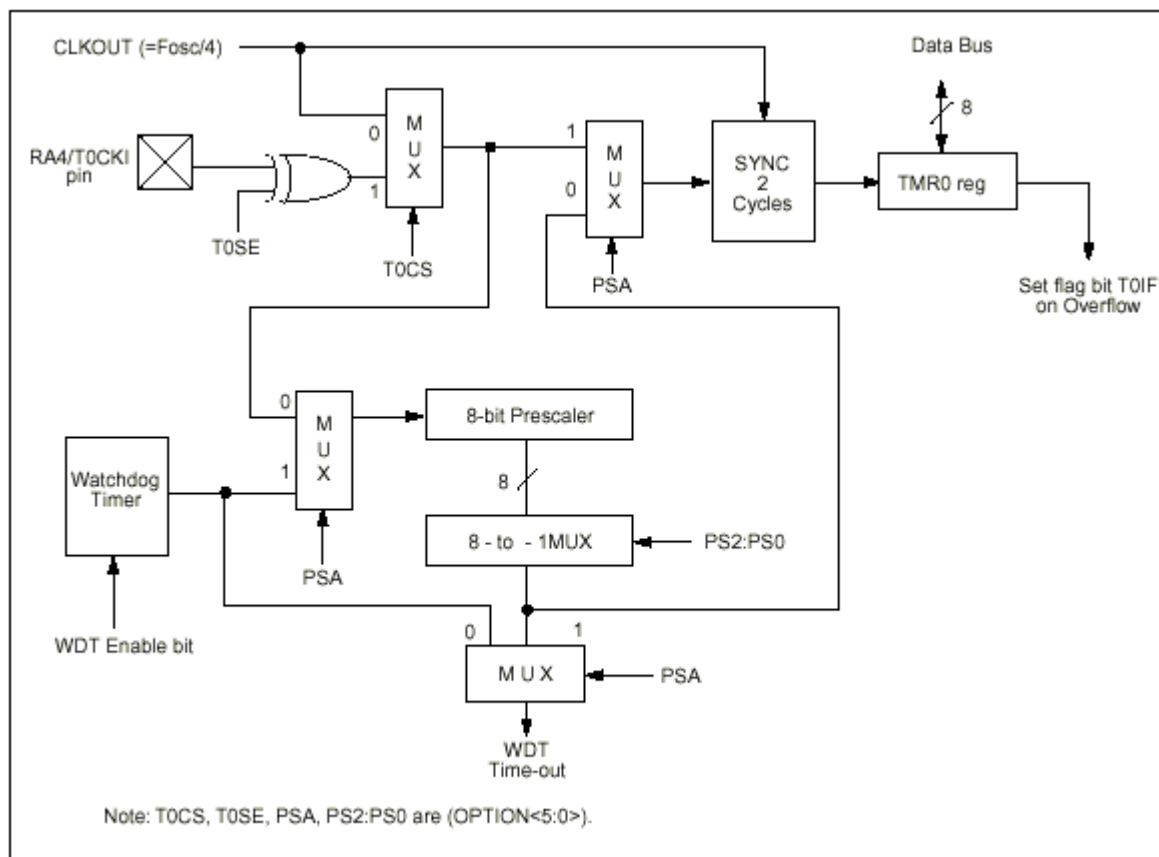


图1.25 TMR0/WDT预分频器方块图

从上图可看出，该预分频器为TMR0和WDT所共用，但某一时刻只能分配给其中之一使用，这完全由程序控制。你可以在程序中变换预分频器的分配对象，下面两段程序是推荐的转换程序。

1. 从TMR0到WDT

```
BCF    STATUS, RP0
CLRF   TMR0           ; 清TMR0和预分频器
BSF    STATUS, RP0
CLRWDT           ; 清WDT
MOVLW  B' XXXX1XXX'
MOVWF  OPTION        ; PSA=1
BCF    STATUS, RP0
```

2. 从WDT到TMR0

```
CLRWDT           ; 清WDT和预分频器
BSF    STATUS, RP0
MOVLW  B' XXXX0XXX'
MOVWF  OPTION        ; PSA=0
BCF    STATUS, RP0
```

当预分频器给TMR0时，任何操作到TMR0的指令（如CLR1，MOVWF1等）都会清零预分

频器。同理, 当预分频器给WDT时, 一条清WDT指令(CLRWDT)将会同时清零其预分频器。 这里应注意, 指的是预分频器, 而不是别的(如OPTION寄存器) 即分频倍率和其分配对象并不会改变。

预分频器不能由软件读/写。

地 址	名 称	Bit7	Bit6	Bit 5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值	其他复位值
01h	TMR0	TIMER0寄存器								XXXX XXXX	uuuu uuuu
0Bh/8Bh	INTCON	GIE	PEIE	TOI E	INTE	RBIE	TOIF	INTF	RBIF	0000 000X	0000 000u
81h	OPTION	RBPu	INTEDG	TOC S	TOCE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
81h	TRISA			TRI SA5	TRISA 4	TRISA 3	TRISA 2	TRISA 1	TRIS A0	--11 1111	--11 1111

注：X=不定，u=不变，-=未用，阴影部分为TIMER0无关位。

(1) TRISA<5>和PEIE位在16C61中未用。

表1. 12 TIMER0相关寄存器

§ 1. 9. 2 TIMER1定时器/计数器

涉及型号						
62	62A	63	64	64A	65	65A

TIMER1为16位宽, 分别由 2个8位可读/写的寄存器TIMER1H和时TIMER1L组成。

TIMER1可从0000增1计数到FFFFh, 并且溢出回到0。当溢出发生后, 置TIMER1IF (PIR1<0>) 为1, 并产生中断请求, 其中断屏蔽位为TIMER1IE (PIE1<0>), 详见PIE1寄存器描述。

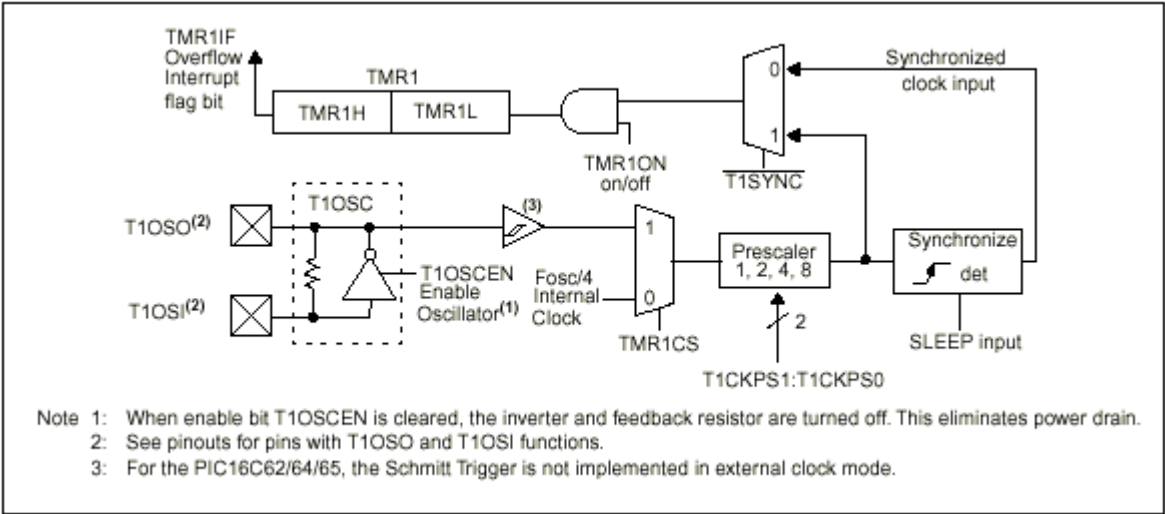


图1. 26 TIMER1方块图

TMR1可工作在两种模式，见下表：

TMR1CS (T1CON)	工作模式	时钟源
0	定时器	内部指令周期
1	计数器	外加于RC0/T1CK1脚上的外部信号

表1.13 TMR1工作模式

T1CON寄存器用来控制时TMR1的种种操作, 包括打开/关闭TMR1, 详见下图:

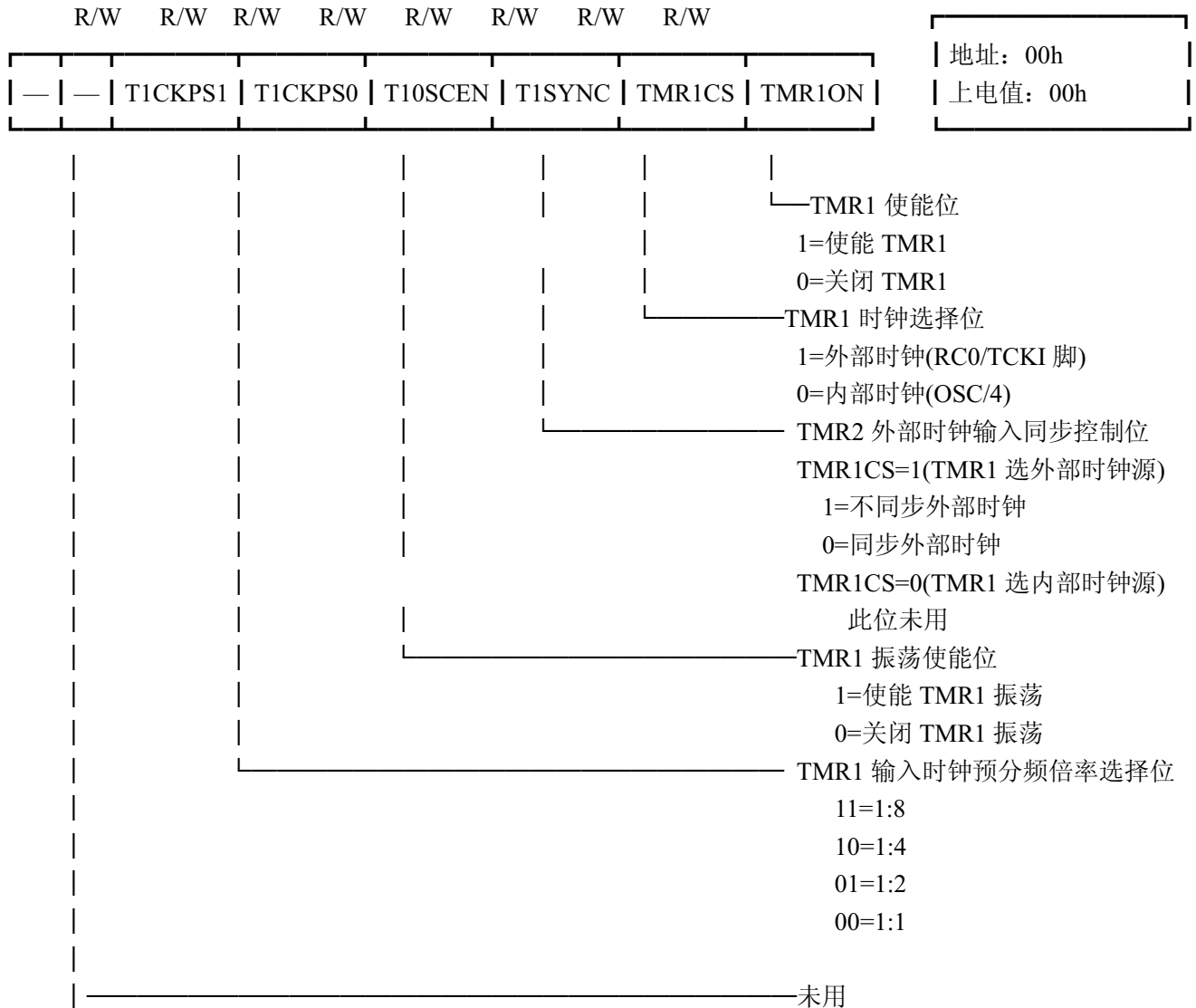


图 1.27 T1CON 控制寄存器

TMR还可以被内部复位, 详见有关CCP1和CCP2(捕捉/比较/PWM) 章节。

一、TMR1定时器模式

当TMR1CS (T1CON <1>) =0, TMR1工作在定时器模式, 这时TMR1的时钟源内部的指令周期信号 (OSC/4) , 此时同步控制位T1NSYNC (T1CON <2>) 无效, 因为内部时钟总是同步的。

二、TMR1计数器模式

当TMR1CS=1, 则TMR1工作在计数器模式, 此时TMR1的时钟源是加到T1CKI脚上的信号, 且上升沿触发计数(增1)。TMR1在计数器模式下再分同步和异步两种不同工作模式。

(一)、同步方式

当TMR1CS=0, TMR1工作在同步计数器模式, 外部时钟输入须和内部的时钟进行相位同步, 再输入到TMR1进行计数。在这种方式下, 如果CPU进入“睡眠”状态, 由于同步电路已经关闭(以便省电), 所以TMR1停止计数, 不理睬加在外部输入端上的触发信号。当然, 其预分频器仍会继续工作(增1)。

由于外部输入信号需和内部时钟同步, 所以它必须一定的条件。而且, 由于同步工作, TMR1的计数会有点延迟。

- a. 当预分频倍数选择1: 1，在指令周期的Q1和Q4之间采样预分频器的输出（此时这个输出即是外部时钟信号的输入）来进行同步，所以要求T1CKI端上的信号高、低电平时间至少各需要2Tosc(另加20ns的RC延迟)。
- b. 当预分频倍数大于1: 1，外部输入信号要经预分频器分频,所以对T1CK1端上信号的要求是其周期至少是4Tosc预分频倍数，至于高、低电平的最短时间不可短于10ns的最小宽度。

（二）、异步方式

当T1SYNC=1，TMR1工作在异步计数器方式，即TMR1的计数不须和内部指令同步。在这种工作模式下，即使CPU处于“睡眠”状态，TMR1将会继续计数，如果发生溢出，可以产生中断并唤醒CPU。

在异步计数器方式下，外部时钟输入的高、低电平时间不得小于10ns的最短宽度。

上面提过，TMR1是由二个8位的寄存器组成：TMR1H和TMR1L。当用户读TMR1这个16位的寄存器时，是分二次分别读TMR1H和TMR1L，下面是一段推荐的读TMR1的程序：

```
BCF      INTCON, GIE      ; 关闭所有中断
MOVF     TMR1H, W
MOVWF    TMPH              ; 读TMR1H存入TMPH
MOVF     TMR1L, W
MOVWF    TMPL              ; 读TMR1L存入TMPL
MOVF     TMR1H, W          ; 再读TMR1H——W
SUBWF    TMPH, W           ; 和第一次读的TMR1H内容作比较
BTFSC    STATUS, Z        ; 相等（相减为0）否？
GOTO     CONTINUE         ; 相等，未发生从TMR1L到TMR4H的进位，OK
MOVF     TMR1H, W          ; 不等，发生进位，则重读
MOVWF    TMPH              ; TMR1寄存器
MOVF     TMR1L, W
MOVWF    TMPR1L, W
MOVWF    TMPL
BSF      INTCON, GIE      ; 重开中断
CONTINUE
:
:
```

三、TMR1振荡

在T10SI（输入）和T10S0（输出）之间有一晶体振荡电路，用户通过置T10SCEN（T1CON〈3〉）=1来启动该振荡电路，这时T10SI脚被硬件自动设置为输入态，而不理会其方向控制位的值。它须是一个低功耗的振荡，频率最高为200KHZ。在单片机进入睡眠状态后，TMR1振荡可以保持继续工作（此时 OSC1和OSC2之间的主振荡停止了），从而使单片机继续工作在低频省电状态下。

下表列出TMR1振荡频率和外接电容的关系：

Osc Type	Freq	C1	C2
LP	32 kHz	33 pF	33 pF
	100 kHz	15 pF	15 pF
	200 kHz	15 pF	15 pF
These values are for design guidance only.			
Crystals Tested:			
32.768 kHz	Epson C-001R32.768K-A	± 20 PPM	
100 kHz	Epson C-2 100.00 KC-P	± 20 PPM	
200 kHz	STD XTL 200.000 kHz	± 20 PPM	
Note 1: Higher capacitance increases the stability of oscillator but also increases the start-up time.			
2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.			

表1. 14 TIMER1振荡参数表

由于TIMER1振荡是低频振荡(LP), 用户程序应考虑一定的延时以便让TIMER1 建立起稳定的振荡。

四、TIMER1的复位(清零)

TIMER1寄存器不会随芯片复位而清零, 而其控制寄存器T1CON则会随芯片上电复位而清零, 其他芯片复位则不影响它的值。

当CCP1和CCP2模块被定义为比较模式(CCP1M<3:0>=1011)则一旦发生比较触发输出时清零TIMER1寄存器, 请参阅有关CCP的章节。

下表是和TIMER1有关的寄存器。

地 址	名 称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电 复位 值	其它 复位 值
0Bh/8 Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF INTF		RBIF	0000 000X	0000 000u
0Ch	PIR1	PSP1F	(3)	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE	(3)	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Eh	TMR1L	TIMER1寄存器(低8位)								XXXX XXXX	uuuu uuuu
0Fh	TMR1H	TIMER1寄存器(高8位)								XXXX XXXX	uuuu uuuu
10h	T1CON		—	T1CKPS 1	T1CKP S0	T1OSCE N	T1SYNC	TMR1CS	TMR1O N	—00 0000	—uu uuuu

注：X=不定，u=不变，-=未用，阴影为TIMER1模块无关位。

- (1) USART仅16C63/65中有。
- (2) PSPIE和PSPIF仅16C64/65中有。
- (3) 保留位, 保持为'0'。

表1. 15 TIMER1相关的寄存器

§ 1. 9. 3 TIMER2定时器

涉及型号						
62	62A	63	64	64A	65	65A

TIMER2是一个8位的定时器, 具有预分频器和后分频器。它最主要是用于作为PWM 输出的时基, 请参阅有关CCP模块作为PWM输出的章节。

TIMER2是可读/写的寄存器, 随芯片复位而清零。TIMER2还有一个8位的周期寄存器PR2。PR2可以预先置一个值, 一旦TIMER2定时器的值和PR2的值相等时, TIMER2 即复位清零重新计数。PR2也是一个可读/写的寄存器, 芯片复位后自动置为全1 (FFh)。

TIMER2的输入为内部指令时钟(OSC/4), 可以有3种预分频率(1:1, 1:4或1:16), 见下面T2CON寄存器的描述。

TIMER2溢时可以产生中断请求输出, 并且这个中断请求输出可以有16种后分频率(1: 1 ~1:16), 在后分频器之前的TIMER2溢时输出则进入同步串行口模块, 作为串行移位的时钟, 详见下面TMR2方块图。

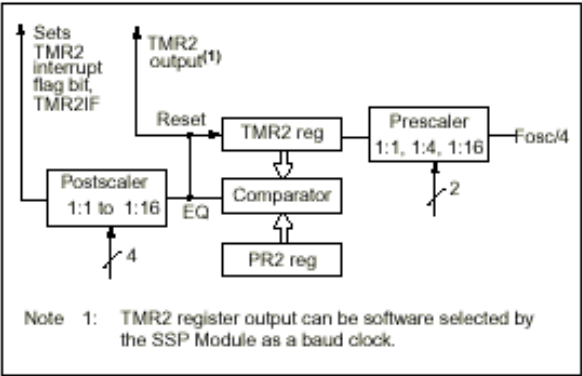


图1.28 TIMER2方块图

TIMER2控制寄存器T2CON如下图所示：

地址:12h
上电值:00h

	R/W	R/W	R/W	R/W	R/W	R/W	R/W
-	TOUTPS 3	TOUTRS 2	TOUTPS 1	TOUTPS 0	TMR20 N	T2CKPS 1	T2CKPS 0

TMR2时钟输入预分频率
00=1 : 1
01=1 : 4
1x=1 : 16

TMR2 使能位
1=使能 TMR2
0=关闭 TMR2

TMR2 输出后分频率
0000=1 : 1
0001=1 : 2
0010=1 : 3
0011=1 : 4
0100=1 : 5
0101=1 : 6
0110=1 : 7
0111=1 : 8
1000=1 : 9
1001=1 : 10
1010=1 : 11
1011=1 : 12
1100=1 : 13
1101=1 : 14
1110=1 : 15
1111=1 : 16

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
0Bh,8Bh 10Bh,18Bh	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF ⁽²⁾	(3)	RCIF ⁽¹⁾	TXIF ⁽¹⁾	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE ⁽²⁾	(3)	RCIE ⁽¹⁾	TXIE ⁽¹⁾	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
11h	TMR2	Timer2 module's register								0000 0000	0000 0000
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
92h	PR2	Timer2 Period register								1111 1111	1111 1111

注：X=不定，u=不变，-=未用，阴影为TIMER2模块无关位。

(1) USART仅16C63/65中有。

(2) PSPIE和PSPIF仅16C64/65中有。

(3) 保留位, 保持为'0'。

表1. 16 TIMER2相关寄存器

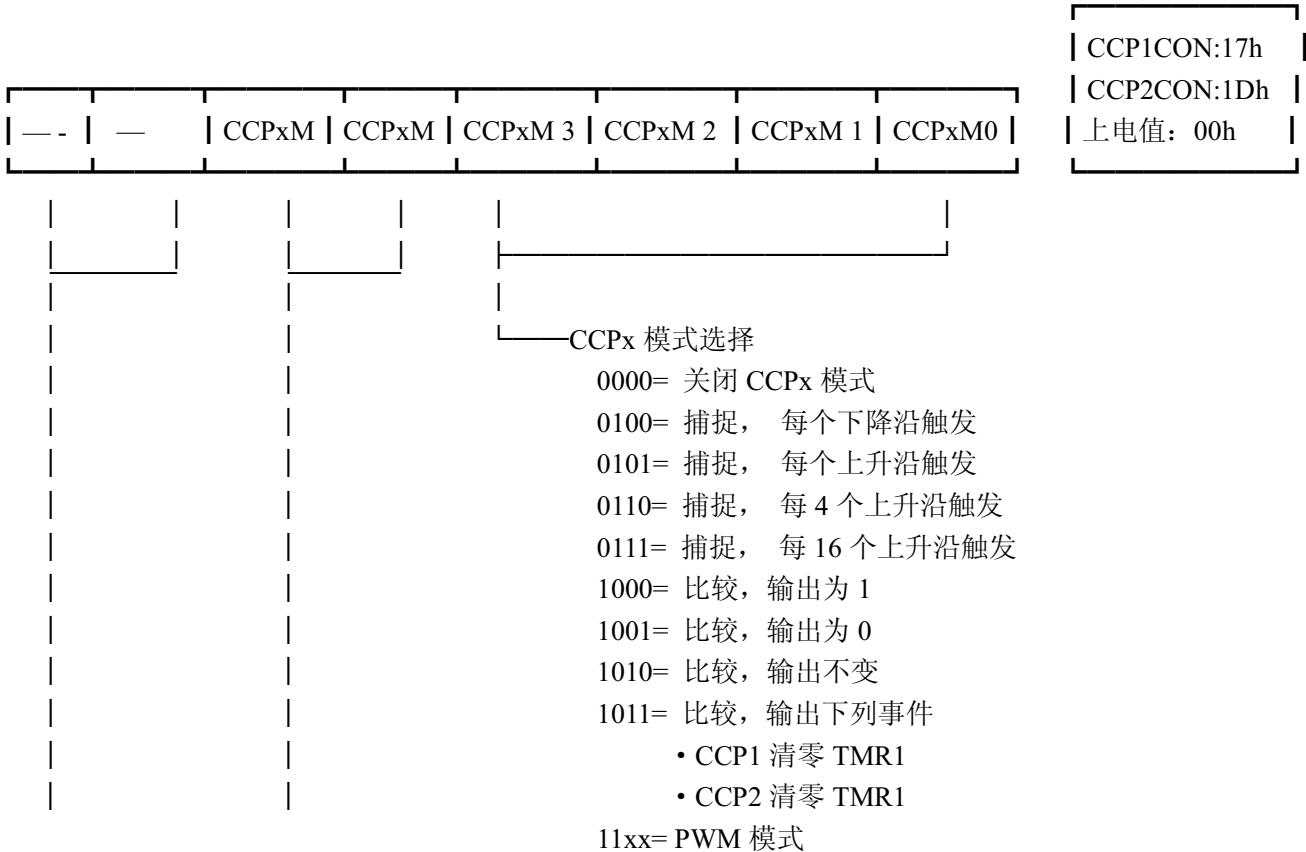
§ 1. 10 CCP模块

涉及型号						
62	62A	63	64	64A	65	65A

CCP模块的含义是捕捉/比较/脉宽调制(Capture/Compare/PWM)。PIC16C63/65 有2 个CCP模块:CCP1和CCP2;PIC16C62/64有一个CCP模块即CCP1, 而PIC16C61没有CCP模块。

CCP模块由一个16位的可读/写寄存器组成, 读寄存器可作为16位的捕捉寄存器或16 位的比较寄存器或PWM (脉宽调制) 输出。CCP1和CCP2的结构及功能完全一样, 区别仅是输出脚 (CCP1脚和CCP2脚) 及寄存器 (CCPR1 和CCPR2) 不一样, 下面仅叙述CCP1模块, CCP2同理因而省略。

下图是CCP1/CCP2模块控制寄存器



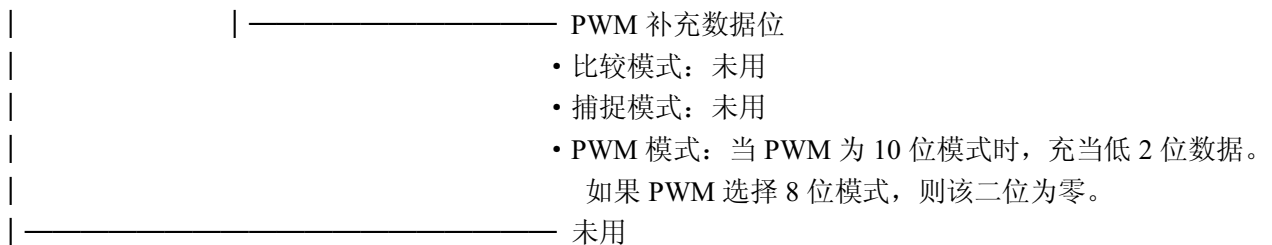


图 1.30 CCP1CON/CCP2CON 寄存器

1.10.1 捕捉模式 (Capture)

当 CCP1 工作在捕捉模式时，一旦有下列事件在 RC2/CCP1 脚上发生时，CCPR1 寄存器（有两个 8 位的 CCPR1H 和 CCPR1L 组成）即捕捉记录下这时 TMR1 寄存器的值：

1. 脉冲下降沿
2. 脉冲上升沿
3. 每 4 个上升沿
4. 每 16 个下降沿

这些触发控制的选择由控制位 CCP1M0—CCP1M3 (CCP1CON <3: 0>) 设置而定。当一个捕捉事件发生后，硬件自动将 CCP1IF 位 (PIR1 <2>) 置为“1”，产生中断请求。CCP1IF 位必须由软件来重新清零。当寄存器 CCPR1 中的值还未被 CPU 读取，而又有另一个新的捕捉事件发生，原来值即被新的值覆盖。

在捕捉模式下，RC2/CCP1 脚必须由相应的 I/O 方向寄存器 (TRISC <2>) 置为输入态。如果 RC2/CCP1 是被置为输出态，则每个写该口的动作就会产生一个捕捉事件。注意，当 CCP1 从捕捉模式改变成其他模式时，会产生一个错误的捕捉中断，故在改变捕捉模式时用户必须事先清零 CCP1IE (PIE1 <0>) 来屏蔽 CCP1 中断，并且在捕捉模式改变后清零 CCP1IF (PIR1 <2>) 位。如果用户程序改变（设置）捕捉预分频率，也会产生一个错误的中断请求，并且预分频器还不会清为零，所以第一次捕捉可能是从一个非零的预分频值开始的。下面例程可以清零预分频器并且不会引起错误的中断请求。

例：	CLRF	CCP1CON	； 关闭 CCP1 模块
	MOVLW	NEW-CAPT-PS	； 选择新的预分频率
	MOVWF	CCP1CON	； 置入 CCP1CON 寄存器中

当 CCP1 工作在捕捉模式时，TMR1 必须工作在定时器或同步计数器模式下，当 TMR1 工作异步计数器模式下时，CCP1 不能工作在捕捉模式。

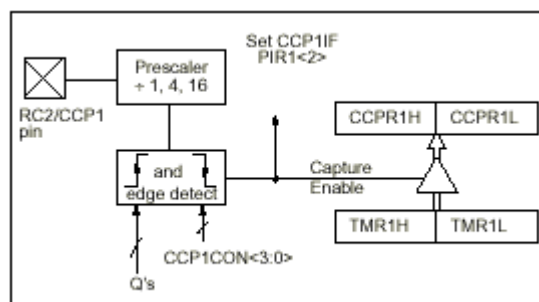


图1. 31 捕捉模块结构

Add	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh, 8Bh 10Bh, 18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF ⁽²⁾	(3)	RCIF ⁽¹⁾	TXIF ⁽¹⁾	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
0Dh ⁽⁴⁾	PIR2	—	—	—	—	—	—	—	CCP2IF	---- --0	---- --0
8Ch	PIE1	PSPIE ⁽²⁾	(3)	RCIE ⁽¹⁾	TXIE ⁽¹⁾	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
8Dh ⁽⁴⁾	PIE2	—	—	—	—	—	—	—	CCP2IE	---- --0	---- --0
87h	TRISC	PORTC Data Direction register								1111 1111	1111 1111
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu
15h	CCPR1L	Capture/Compare/PWM1 (LSB)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM1 (MSB)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
1Bh ⁽⁴⁾	CCPR2L	Capture/Compare/PWM2 (LSB)								xxxx xxxx	uuuu uuuu
1Ch ⁽⁴⁾	CCPR2H	Capture/Compare/PWM2 (MSB)								xxxx xxxx	uuuu uuuu
1Dh ⁽⁴⁾	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

注：阴影部分是本模式无关的位

表 1.7 和捕捉输入有关的寄存器

§ 1.10.2 比较模式 (Compare)

当 CCP1 工作在比较模式下，16 位的 CCPR1 寄存器不停地和 TMR1 寄存器做比较，当两者的值相等时，在 RC2/CCP1 的脚上就可以出现：

1. 高电平，可用于驱动某外部部件。
2. 低电平，可用于驱动某外部部件。
3. 原电平保持不变，软件中断模式。

这 3 种选择由 CCP1CON<3:0>位来决定，请参阅 CCP1CON 寄存器的描述。一旦发生比较相符时还会产生一个比较输出中断请求。在比较模式下，用户程序必须把 RC2/CCP1 置为输出态以便产生比较输出。注意，清零 CCP1CON 寄存器将会引起 RC2/CCP1 脚输出低电平，这并非是正常的比较输出。

TMR1 必须工作在定时器或同步计数器模式下，CCP1 才能作为比较模式工作。当 TMR1 运行于异步计数器模式下时，CCP1 不能作比较模式。

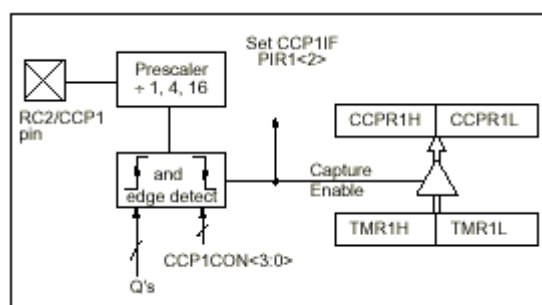


图 1.32 比较模块结构

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值	其他复位值
0Bh/8Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSP1 F	(3)	RCIF	TXIF	SSPIF	CCP1 IF	TMR 2IF	TMR 1IF	0000 0000	0000 0000
0Dh(4)	PIR2	—	—	—	—	—	—	—	CCP2 IF	---- --0	---- --0

8Ch	PIE1	PSPIE	(3)	RCIE	TXIE	SSPIE	CCP1 IE	TMR 2IE	TMR 1IE	0000 0000	0000 0000
8Dh(4)	PIE2	—	—	—	—	—	—	—	CCP2 IE	---- --0	---- --0
0Eh	TMR1L	TIMER1 寄存器(低 8 位)								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	TIMER1 寄存器(高 8 位)								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CK PS1	T1CK PS0	T1OS CEN	T1SY NC	TMR 1CS	TMR 1ON	--00 0000	--uu uuuu
15h	CCPR1L	CCP1 寄存器(低 8 位)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	CCP1 寄存器(高 8 位)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1 X	CCP1 Y	CCP1 M3	CCP1 M2	CCP1 M1	CCP1 M0	--00 0000	--00 0000
1Bh(4)	CCPR2L	CCP2 寄存器(低 8 位)								xxxx xxxx	uuuu uuuu
1Ch(4)	CCPR2H	CCP2 寄存器(高 8 位)								xxxx xxxx	uuuu uuuu
1Dh(4)	CCP2CON	—	—	CCP2 X	CCP2 Y	CCP2 M3	CCP2 M2	CCP2 M1	CCP2 M0	--00 0000	--00 0000

注：x=不定， u=不变， -=未用， 阴影为 TIMER1 模块无关位。

(1) USART 仅 16C63/65 中有。

(2) PSPIE 和 PSPIF 仅 16C64/65 中有。

(3) 保留位，保持为'0'。

(4) 和 CCP2 相关，仅 16C63/65 有。

表 1.17 捕捉输入/比较输出相关寄存器

§ 1.10.3 脉宽调制模式（PWM）

当 CCP1 工作在 PWM 模式下，RC2/CCP1 可输出高达 10 位的脉宽调制波形，这时 RC2/CCP1 必须设置为输出态（通过置 TRISC<2>=1）。在 PWM 输出模式下，用户把 8 位的频宽（duty cycle）置入 CCPR1 寄存器的低 8 位，即 CCPR1L 寄存器。CCPR1 的高 8 位寄存器 CCPR1H 则作为 CCPR1L 的从属寄存器，即 8 位频宽数据从 CCPR1L 再载入 CCPR1H 然后再和时基寄存器 TMR2 进行比较，PWM 输出的周期则由 TMR2 的周期寄存器 PR2 决定，见图 1.33。

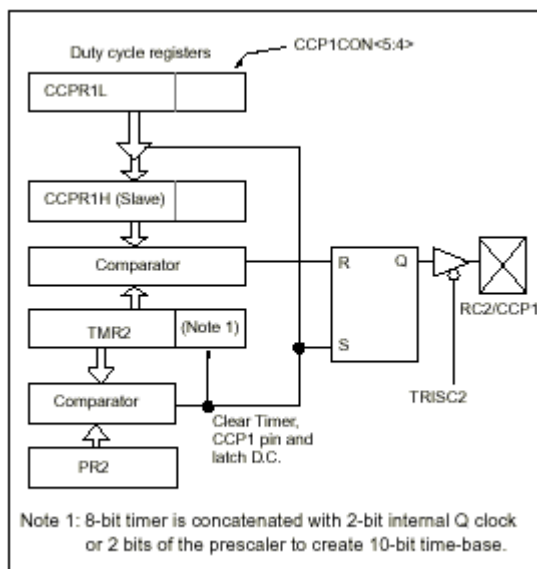


图 1.33 PWM 模块结构

一、PWM 周期

PWM 输出周期由 PR2 寄存器值决定，如下列公式所示：

$$\text{PWM 周期} = \frac{[(\text{PR2}) + 1] \cdot 4T_{\text{osc}} \cdot (\text{TMR2 预分频数})}{1}$$

$$\text{PWM 频率} = \frac{1}{\text{PWM 周期}}$$

当 TMR2=PR2，则会发生如下事件：

1. TMR2 清零；
2. RC2/CCP1 脚置“1”；
3. PWM 之频宽（duty cycle）从 CCPR1L 载入 CCPR1H。

如下图所示：

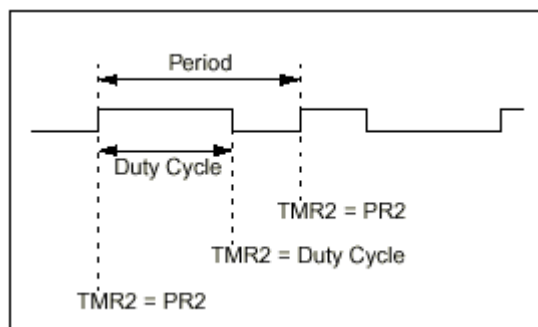


图 1.34 PWM 输出

二、PWM 频宽

PWM 频宽由 CCPR1L 和 CCP1CON<5:4>组成，所示最大可达 10 位，其计算公式如下：

$$\text{PWM 频宽} = (\text{CCPR1L} : \text{CCP1CON} \langle 5:4 \rangle) \cdot T_{\text{osc}} \cdot (\text{TMR2 预分频数})$$

CCPR1L 和 CCP1CON<5:4>的值可以随时写入，但只有当 PR2=TMR2 时，它的值才会载入到 CCPR1H 中。在 PWM 模式下，CCPR1H 是只读寄存器。

例：设需 PWM 频率为 31.25KHZ，

Fosc=8MHZ，

TMR2 预分频数=1，

求 PR2 值。

$$\frac{1}{31.25\text{KHZ}} = \frac{[(\text{PR2}) + 1] \cdot 4 \cdot 1/8\text{MHZ} \cdot 1}{1}$$

$$32 \mu\text{S} = [(\text{PR2}) + 1] \cdot 4 \cdot 125\text{ns} \cdot 1$$

$$\text{PR2} = 63$$

三、PWM 分辨率

$$\log\left(\frac{F_{\text{osc}}}{F_{\text{PWM}}}\right)$$

$$\text{最大 PWM 分辨率} = \frac{\log(2)}{\log(2)} \text{ (位)}$$

对于上例来说，最大分辨率计算如下：

$$\frac{1}{1} = 2 \text{ 最大 PWM 分辨率} \cdot 1/8\text{MHZ} \cdot 1$$

31.25KHZ
最大 PWM 分辨率=8 (位)

所以可以取设最大 PWM 分辨率为 8 位。如果要取得更高的分辨率，则必须降低 PWM 的频率；反之如果要取得更高的 PWM 频率，则需降低分辨率。
下表列出了 PWM 频率和分辨率的关系。

最高分辨率 (高分模式)	频 率		
	TIMER2 预分频=1	TIMER2 预分频=4	TIMER2 预分频=16
10 位	19.53KHZ	4.88KHZ	1.22KHZ
9 位	39.06KHZ	9.77KHZ	2.44KHZ
8 位	78.13KHZ	19.53KHZ	4.88KHZ

a.

PWM 频率	1.22KHZ	4.88KHZ	19.53KHZ	78.12KHZ	156.3KHZ	208.3KHZ
预分频率	1:16	1:4	1:1	1:1	1:1	1:1
PR2 值	0XFF	0XFF	0XFF	0X3F	0X1F	0X5F
分辨率（高分模式）	10 位	10 位	10 位	8 位	7 位	6.5 位
分辨率（低分模式）	8 位	8 位	8 位	6 位	5 位	4.5 位

b.
表 1.18 PWM 频率和分辨率（20MHZ 主频下）

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值	其他复位值
0Bh/8Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSP1 F	(3)	RCIF	TXIF	SSPIF	CCP1 IF	TMR 2IF	TMR 1IF	0000 0000	0000 0000
0Dh(4)	PIR2	—	—	—	—	—	—	—	CCP2 IF	---- ---0	---- ---0
8Ch	PIE1	PSPIE	(3)	RCIE	TXIE	SSPIE	CCP1 IE	TMR 2IE	TMR 1IE	0000 0000	0000 0000
8Dh(4)	PIE2	—	—	—	—	—	—	—	CCP2 IE	---- ---0	---- ---0
11h	TMR2	TIMER2 寄存器								0000 0000	0000 0000
92h	PR2	TIMER2 周期寄存器								1111 1111	1111 1111
12h	T2CON	—	TOUT PS3	TOUT PS2	TOUT PS1	TOUT PS0	TMR 2ON	T2CK PS1	T2CK PS0	-000 0000	-000 0000
15h	CCPR1L	CCP1 寄存器(低 8 位)								xxxx xxxx	uuuu uuuu
16h	CCPR1H	CCP1 寄存器(高 8 位)								xxxx xxxx	uuuu uuuu
17h	CCP1CON	—	—	CCP1 X	CCP1 Y	CCP1 M3	CCP1 M2	CCP1 M1	CCP1 M0	--00 0000	--00 0000
1Bh(4)	CCPR2L	CCP2 寄存器(低 8 位)								xxxx xxxx	uuuu uuuu
1Ch(4)	CCPR2H	CCP2 寄存器(高 8 位)								xxxx xxxx	uuuu uuuu

1Dh(4)	CCP2CON	—	—	CCP2 X	CCP2 Y	CCP2 M3	CCP2 M2	CCP2 M1	CCP2 M0	--00 0000	--00 0000
--------	---------	---	---	-----------	-----------	------------	------------	------------	------------	-----------	-----------

注：x=不定， u=不变， -=未用， 阴影为本模式无关位。

- (1) USART 仅 16C63/65 中有。
- (2) PSPIE 和 PSPIF 仅 16C64/65 中有。
- (3) 保留位，保持为'0'。
- (4) 和 CCP2 相关，仅 16C63/65 有。

表 1.19 和 PWM 输出有关的寄存器

§ 1.11 同步串行口（SSP）模块

涉及型号								
62A	62B	63	63A	64A	65	65A	66	67

除 PIC16C61 外，其余 PIC16C6X 都有同步串行口模块（以下简称 SSP）用来和外围串行器件或其他微处理器进行通讯，这些外围器件可以是串行 E2PROM、移位寄存器、显示器、A/D 转换器等。SSP 有以下二种工作模式：

1. 串行外围接口（简称 SPI）
2. I²C 总线

下图是 SSP 的状态寄存器和控制寄存器，用来对 SSP 进行控制并记录其各种工作状态。

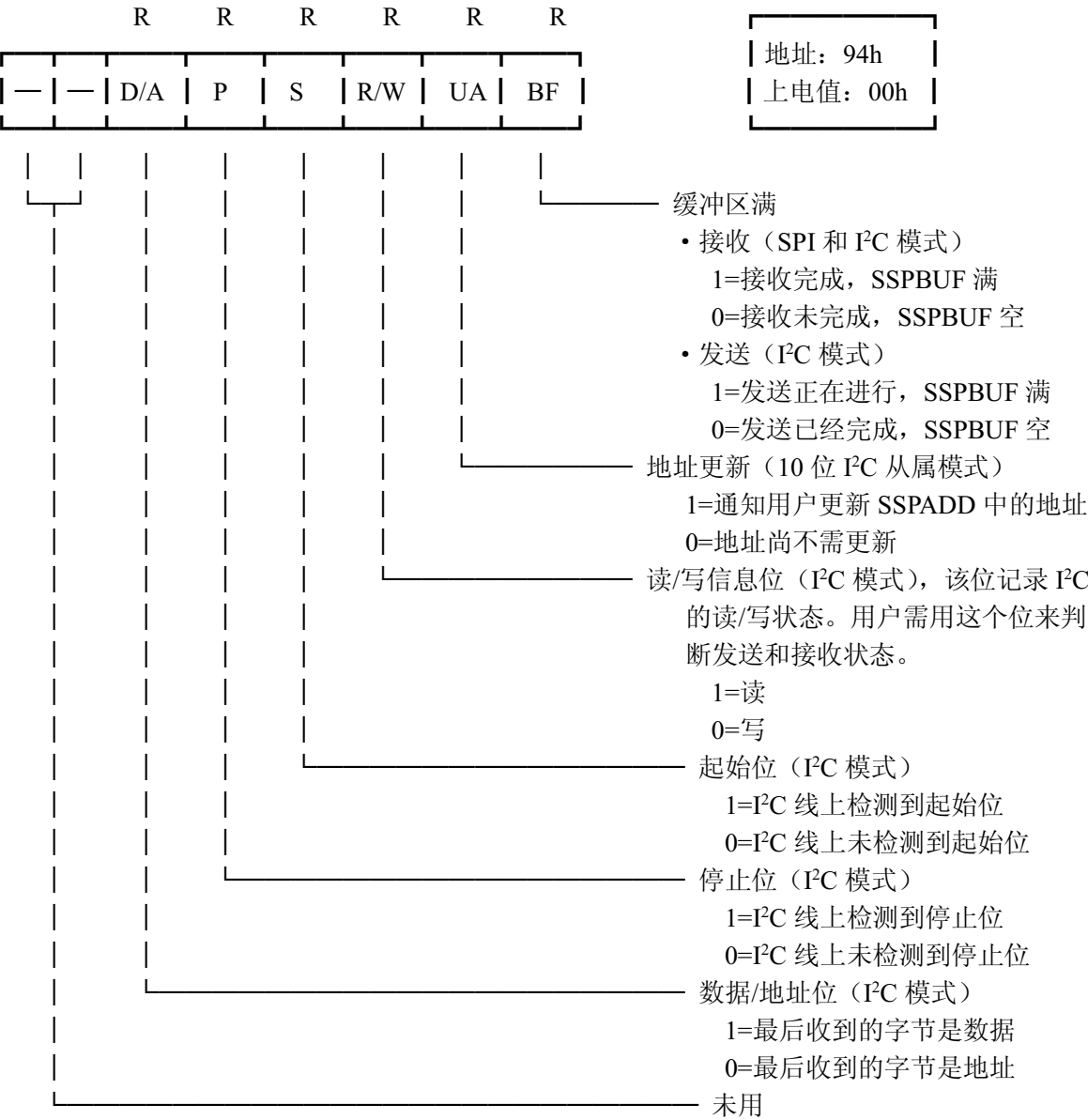


图 1.35 SSP 状态寄存器 SSPSTAT



图 1.36 SSP 控制寄存器 SSPCON

SPI 是 Motorola 公司推出的一种同步串行传输方式,可同步传送或接收 8 位数据,由三个引脚来完成通讯工作。

另外当 SPI 处于“从属操作”模式时，可能还需要第四个脚。

在初始化 SPI 工作状态时，必须通过设置其控制寄存器 SSPCON<5:0>来确定以下工作方式：

同步串行模块 SSP 由传送/接收移位寄存器 SSPSR 和缓冲器 SSPUF 组成。SSPSR 寄存器用于数据传输的移位，而 SSPBUF 用于保存置入 SSPSR 的数据，实际上是起到第二缓冲器的作用。当 CPU 收到一个 8 位数据时，就将其存到 SSPBUF，并且置缓冲区满标志 BF (SSPSTAT<0>) = 1 及中断请求位 SSPIF (PIR1<3>) = 1。由于 SSPBUF 起双缓冲区作用，故在第一个接收到的数据未被 CPU 读取时，SSPSR 寄存器即可进行第二个数据的接收。当进行数据传输/接收时，任何试图写 SSPBUF 的操作都无效，并且将造成写无效标志位 WCOL (SSPCON<7>) = 1。此时用户必须将 WCOL 位重新清为零，以便使其能标志后面的 SSPBUF 写入是否成功。SSPBUF 所存放的接收到的数据必须及时读取走，否则可能会被后来的数据覆盖掉。如果发出数据覆盖则溢出标志位 SPOV (SSPCON<6>) 被置为 1。BF 位用来标志 SSPBUF 是否已经载入了接收数据，当 SSPBUF 中的数据被读取后，BF 位即被清为零。SSP 中断会通知 CPU 数据传输已经完成。如果用户不愿用中断方式，可用软件查询方式来读取和写入 SSPBUF 中的数据，下例给出对 SSPBUF 操作的程序：

下图是 SSP 模块在 SPI 方式下的结构图。从图中可看到 SSPSR 寄存器不能被直接读写，用户需通过 SSPBUF 寄存器来存取它。

图 1.37 SPI 结构图

```

BCF      SSPCON, SSPEN    ; 先将 SSPEN 清 0
MOVLW    SSPDATA          ; 初始化数据→W
MOVWF    SSPCON           ; 初始化 SSPCON
BSF      SSPCON, SSPEN    ; 激活串行口

```

SPI 工作模式下还应定义 SPI、SDO、SCK 和 SS 的方向（通过 TRISC 寄存器）：

1. SDI 置为输入（TRISC<4>=1）
2. SDO 置为输出（TRISC<5>=0）
3. SCK 方向有二种情况：
 - a. 主控方式下置为输出（TRISC<3>=0）
 - b. 从属方式下置为输入（TRISC<3>=1）
4. SS 置为输入（TRISA<5>=1）

任何串行口功能在未用时皆可由设置其相应的方面寄存器而另作它用。例如在主控模式下，你只想发送数据，则可将未用到的 SDI 和 SS 端当做一般的 I/O 口线使用。

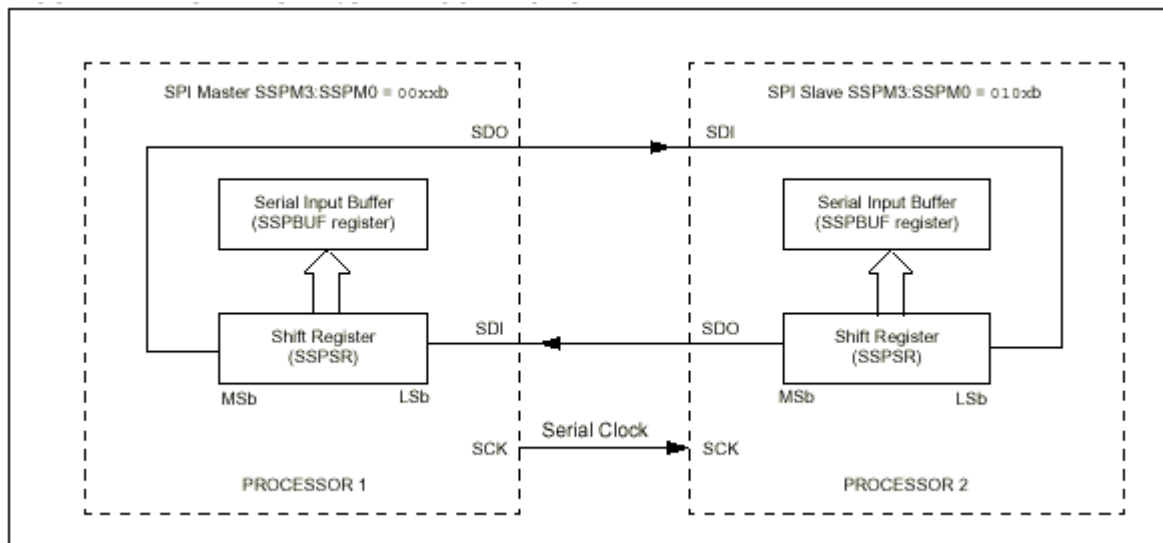


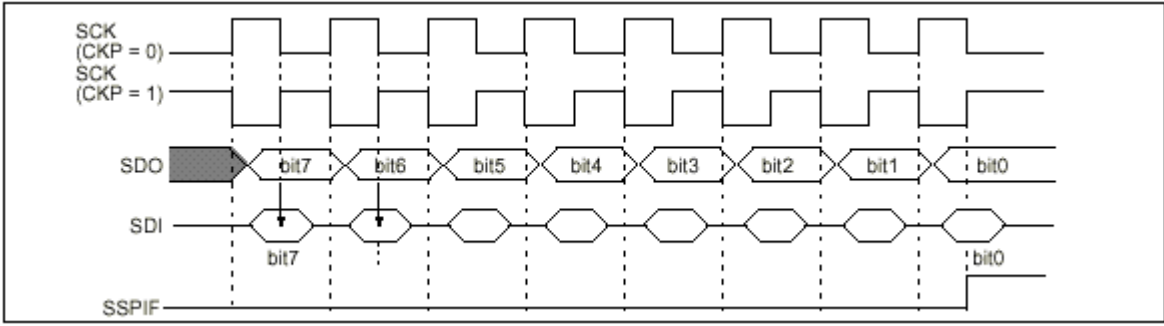
图 1.38 SPI 主控/从属联接

上图给出两个微处理器相连的典型例子。主控器通过发 SCK 信号来启动数据传输，数据通过移位寄存器在各自选定的时钟边沿传送，并在下个边沿被锁存，两个处理器必须以相同的时钟极性进行工作，同时发送和接收数据。传送的数据是否有用则由软件来选择，这有以下三种可能：

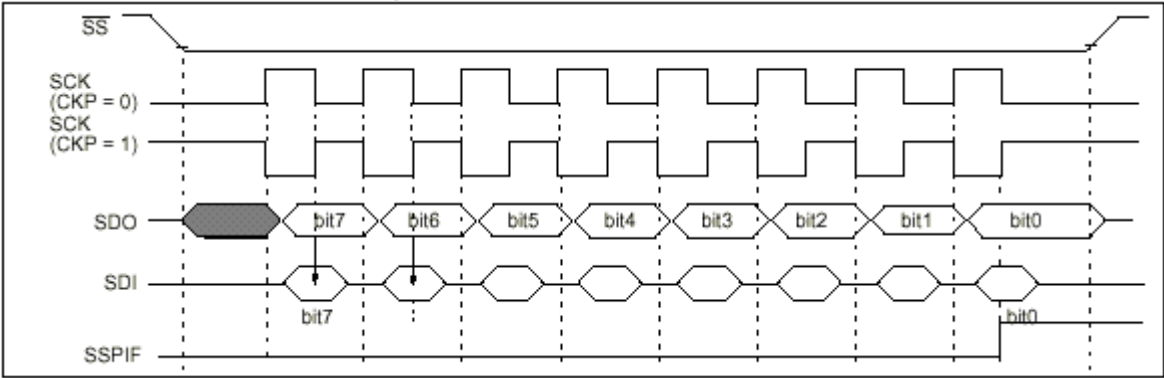
1. 主控器发送有效数据 — 从属器发送无效数据
2. 主控器发送有效数据 — 从属器发送有效数据
3. 主控器发送有效数据 — 从属器发送有效数据

主控器由于控制着 SCK 信号，故可在任何时候启动数据传输。在主控器方面，数据一旦置入 SSPBUF 就可开始读取/传送。此时如果 SPI 仅做接收工作，则 SCK 可以省略不要，SSPSR 将不停地将其 SDI 脚上的信号按其选定的时钟节拍移入，收完一字节（8 位）后即送入 SSPBUF 供 CPU 读取。在从属器方面，数据传送/接收是按 SCK 脚上的时钟节拍进行，当一字节数据接收完成后，中断请求标志 SSPIF=1，发出中断请求。时钟极性可由 CKP（SSPCON<4>）设定。

下面二图给出 SPI 通讯时序。



a. 主控方式或不由 SS 控制的从属方式



b. SS 控制的从属方式
图 1.39 SPI 方式时序

SPI 时钟频率由主控器方面软件设定，有如下四种选择，详见 SSPCON 寄存器描述：

1. OSC/4
2. OSC/16
3. OSC/64
4. TMR2 输出/2

在振荡为 20MHz 时，最高时钟即可达 5MHz。在从属模式下，外来的时钟频率必须满足最短周期的限制。在睡眠（Sleep）状态下，从属处理器亦可传送和接收数据并通过中断请求将 CPU 唤醒。通过 SS 脚还可以设定一种同步从属方式，这时 SPI 须置成从属模式（SSPCON<3:0>= 04h，TRISA<5>=1）。当 SS 处于低电平时，可进行传送/接收，SDO 脚输出可被驱动为高或低电平。当 SS 为高时，则 SDO 成为浮态输出，可以根据需要外接上拉或下拉电阻。如果要仿真二线式通讯，可以把 SDO 和 SDI 直接相连，当 SPI 要作接收时，将 SDO 设为输入，而 SDI 总是设置为输入。

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值	其他复位值
0Bh/8Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBF	0000 000x	0000 000u
0Ch	PIR1	PSP1F	(3)	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE	(3)	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
13h	SSPBUF	SSP 接收/发送寄存器								xxxx xxxx	uuuu uuuu
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
85h	TRISA	—	—	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	--11 1111	--11 1111

94h	SSPSTAT	—	—	D/A	P	S	R/W	UA	BF	--00 0000	--00 0000
-----	---------	---	---	-----	---	---	-----	----	----	-----------	-----------

注：x=不定， u=不变， -=未用， 阴影为本模式无关位。

- (1) USART 仅 16C63/65 中有。
- (2) PSPIE 和 PSPIF 仅 16C64/65 中有。
- (3) 保留位，保持为“0”。

表 1.20 和 SPI 相关的寄存器

§ 1.11.2 I²C 模式

I²C 总线是由 philips 公司设计的两线式串行交互通讯方式，在其标准模式下数据传输速率可达 100Kbps，在快速模式下则可达 400Kbps，两种模式可在同一总线上交互使用。

一、I²C 总线概述

I²C 接口进行数据传输时，须有一个主控器（产生时钟节拍）和一个从属器。下表是有关 I²C 的一些术语。

术 语	解 释
传送器	传送数据到总线的装置
接收器	接收总线上数据的装置
主控器	启动传输，产生时钟和中断传输的装置
受控器	由主控器寻址操作的装置
多主控器	在同一总线中有多个主控器，可同时工作
仲 裁	保证在某刻只有一个主控器控制总线
同 步	同步多个装置的时钟节拍

表 1.21 I²C 总线术语

在 I²C 总线中，每个连接部件都有一个地址。当一主控器要进行数据传输时，它首先发出通讯部件的地址，总线上的其他部件接收到读主控器发出的地址信息并判断是否是自己的地址。在主控器发出的地址信息中，有一个位是用来告诉其通讯对象它将要读出数据还是要写入数据，在数据传输中主控器和从控器总是工作在两个相反的状态：

1. 主控器为发送器 — 从属器为接收器
2. 主控器为接收器 — 从属器为发送器

在任何一种方式中，都是由主控器发时钟节拍信号。SCL（时钟输出端）在其输出时，以及 SDA（数据端）都必须是开漏（集电极开始输出）以便在总线上进行“线与”，所以一般需要加上拉电阻。

下面讲述 I²C 通讯的几个主要方面。

（一）、启动和结束数据传输

I²C 总线在没有数据传输（空闲）时，SCL 线和 SDA 线都由上拉电阻拉为高电平。数据传输的启动和结束时序如下图所示：

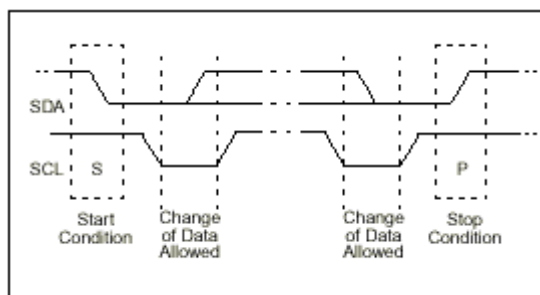


图 1.40 I²C 传输的启动和结束

从上图中可看出，当 SCL 保持高而 SDA 由高变低时，作为启动，当 SCL 保持高而 SDA 由低变高时作为结束。

数据传输过程中，SDA 线只能在 SCL 为低电平时产生高低电平的变化。

（二）、寻址 I²C 部件

I²C 有二种地址传送格式，一种是简单的 7 位地址码加上一位读/写位，另一种则复杂些，由 10 位地址码加一位读/写位。它们的传送格式如下：

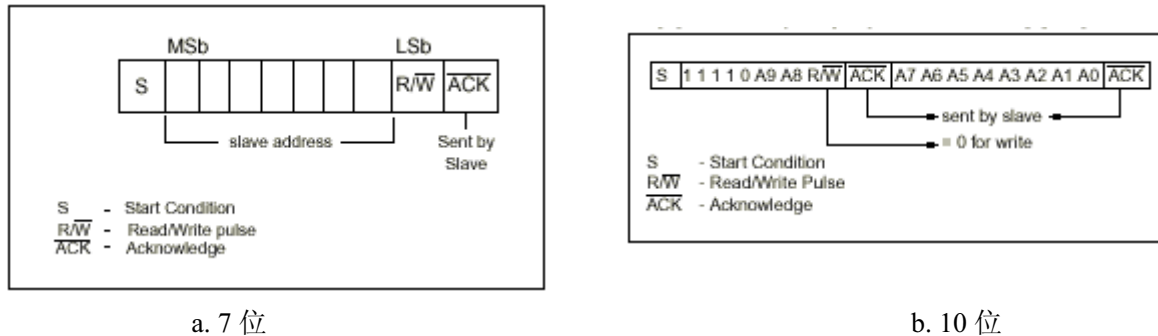
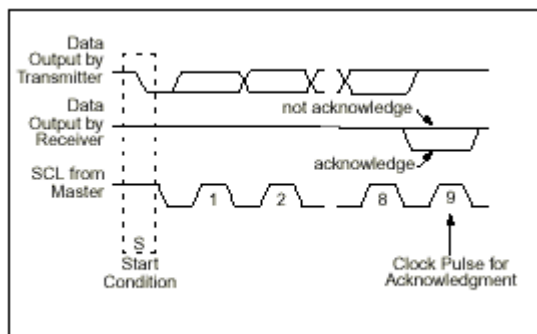


图 1.41 I²C 部件地址传输格式

（三）、传输确认

I²C 总线上所有数据都以字节进行传输，而每次传输的字节数没有限制。从属接收器每收到一个字节的的数据后，都要发出一个确认位（ACK），如图 1.39 所示。如果从属接收器未发 ACK 信号，则主控器必须终止传输。

图 1.42 从接收器确认时序



如果主控器作为接收器，则它收到每一字节数据后也须发一 ACK 确认信号。不过对于接收到的最后一个字节，主控器不发 ACK 信号，并以结束信号（STOP）告诉从属传送器数据传输结束。

从属器这时应释放 SDA 线，以便让主控器产生一个传输结束信号（STOP）。

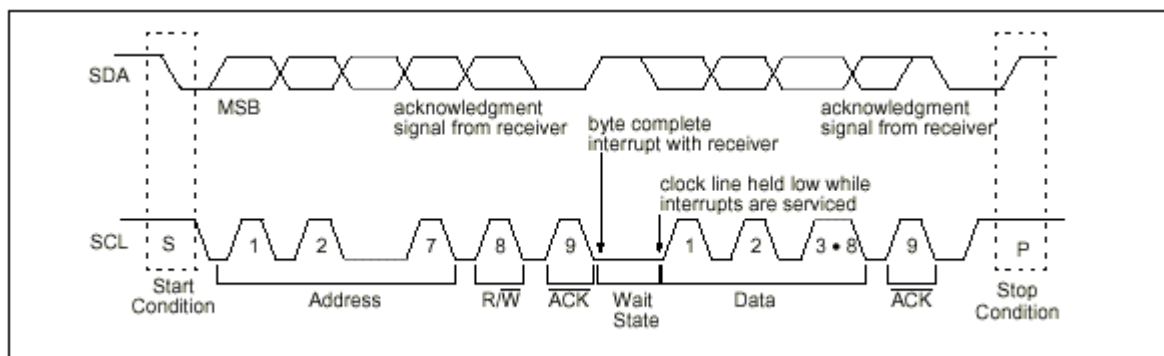


图 1.43 数据传输时序

如果从属器需要延长下一个字节数据的传输，可以通过保持 SCL 为低电平来强制主控器进入等待状态。一旦从属器释放 SCL 线，数据传输则继续下去。这样就允许从属器有足够时间转移收到的数据或取得要传送的数据。

关于 I²C 更详细的技术描述，请读者参阅有关 I²C 总线专著。

二、SSP 模块 I²C 操作模式

PIC16C6X 的 SSP 模块工作在 I²C 模式下时，硬件可完成所有从属器的功能，通过软件支持配合也可完成主控

器功能。SSP 模块并且支持 I²C 的标准和快速传输以及 7 位和 10 位的地址寻址。PIC16C6X 的两个脚用于 I²C 数据传输:RC3/SCK/SCL（时钟）和 RC4/SDI/SDA（数据）。用户须由 TRISC<4:3>设定其输入/输出方向。

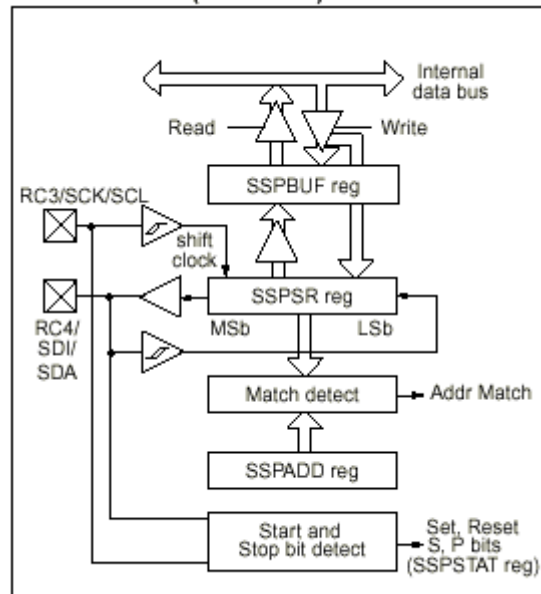


图 1.44 SSP 在 I²C 模式下结构图

SSP 模块有 5 个寄存器用于 I²C 操作。

1. SSP 控制寄存器 SSPCON
2. SSP 状态寄存器 SSPSTAT
3. 串行传输/接收缓冲器 SSBUF
4. SSP 移位寄存器 SSPSR（程序不可直接读取）
5. 地址寄存器 SSPADD

SSPCON 控制 I²C 操作方式，用户程序通过设置 SSPCON<3:0>可选择以下几种方式：

- I²C 从属器模式（7 位地址）
- I²C 从属器模式（10 位地址）
- I²C 从属器模式（7 位地址）并支持主控制器模式
- I²C 从属器模式（10 位地址）并支持主控制器模式
- I²C 主控制器模式，从属器模式关闭

SSPSTAT 寄存器包含数据传输的状态，如 I²C 传输的启动（START）和结束（STOP），收到的字节是数据还是地址以及数据传输的方向（读或写）。详见图 1.34 和图 1.35 有关 SSPCON 和 SSPSTAT 的描述。

SSPBUF 缓冲器和 SSPSR 移位寄存器我们在 § 1.11.1 已有详述，请参阅。8 位的 SSPADD 寄存器存放从属器的地址。在 10 位地址方式下，用户还须发送高位地址（11110A9A80），然后再装入低 8 位地址（A7~A0）。

（一）、从属器模式

在从属模式下，SCL 与 SDA 必须由 TRIS（<4:3>）置为输入态。如果需要（如在从属发送器情况下），SSP 模块将会强行改变其方向为输出态并送出数据。当收到的地址匹配，或从一匹配地址发来的数据收到后，硬件会自动发出一确认脉冲（ACK），并把收到的数据（在 SSPSR 中）装入 SSPBUF 供 CPU 读取。

下面的情况会使 SSP 模块不发出 ACK：

- 在接收时，缓冲区满标志位 BF=1
- 在接收时，溢出标志位 SSPOV=1

在以上情况下，移位寄存器 SSPSR 的值将不被装入 SSPBUF 中，但发出中断请求（SSPIF=1）。

下表列出当收到一个字节后可能发生的各种情况。

数据收到后的状态位值		SSPSR→SSPBUF	发出 ACK 脉冲	置 SSPIF=1 (请求中断)
BF	SSPOV			
0	0	Yes	Yes	Yes
1	0	No	No	Yes
1	1	No	No	Yes

0	1	No	No	Yes
---	---	----	----	-----

表 1.22 从控制器收到一字节后的情况

读 SSBUF 将会清零标志位 BF，而 SSPOV 必须由软件清零。

下面叙述从属模式下的几项操作：

(1) 寻址

当 SSP 模块被激活后，SSP 即等待 START 信号的出现，一旦检测到 START 信号，即把 8 位的接收数据移入 SSPSR。所有的输入位信号都是在 SCL 时钟的上升沿采样。接着把 SSPSR<7:1> 的地址寄存器 SSPADD 作比较（该比较发生在 SCL 时钟的第 8 个脉冲的下降沿）。如果地址匹配，则 BF 位和 SSPOV 位被清零，接着做下面的事：

1. SSPSR 值装入 SSPBUF
2. BF=1
3. 产生 ACK 信号
4. 在 SCL 的第 9 个脉冲的下降沿置 SSP 中断标志位 SSPIF=1（发出中断请求）

如果在 10 位地址方式下，从属器还需要接收第二个地址字节，如图 1.39 所示。第一地址字节的高 5 位（11110）标识出是 10 位地址方式，并且 R/W=0 表示是写状态，这样从属器就会接收第二个地址字节。在 10 位地址方式下，高字节地址总是“11110A9A80”并且 SSP 做如下事（其中 7~9 项是针对从属发送器而言的）：

- ① 接收地址高位字节（SSPIF、BF 和 UA 被置为 1）。
- ② 将低位地址置入 SSPADD 中（UA 被清零，释放 SCL 线）。
- ③ CPU 读取 SSBUF（BF 被清零）并清零 SSPIF 位。
- ④ 接收地址低 8 位字节（SSPIF、BF 和 UA 被置为 1）。
- ⑤ 把高位地址置入 SSPADD 中（UA 被清零。如果地址匹配，则释放 SCL 线）。
- ⑥ 读 SSPBUF（BF 被清零）并清零 SSPIF 位。
- ⑦ 接收重复的启动（START）信号。
- ⑧ 接收高位地址（SSPIF 和 BF 被置 1）。
- ⑨ 读 SSPBUF（清 BF）并清 SSPIF。

(2) 接收

当地址字节后的读写标志位 R/W=0，且地址匹配时，状态寄存 SSPSTAT 中的 R/W 位清零，并且把接收到的地址数据装入 SSPBUF。如果发生地址字节接收溢出，则从属器不会发出确认 ACK。

SSP 每传输一个字节，都会使 SSPIF 置 1 而发出中断请求，在中断服务程序中必须把 SSPIF 清零。

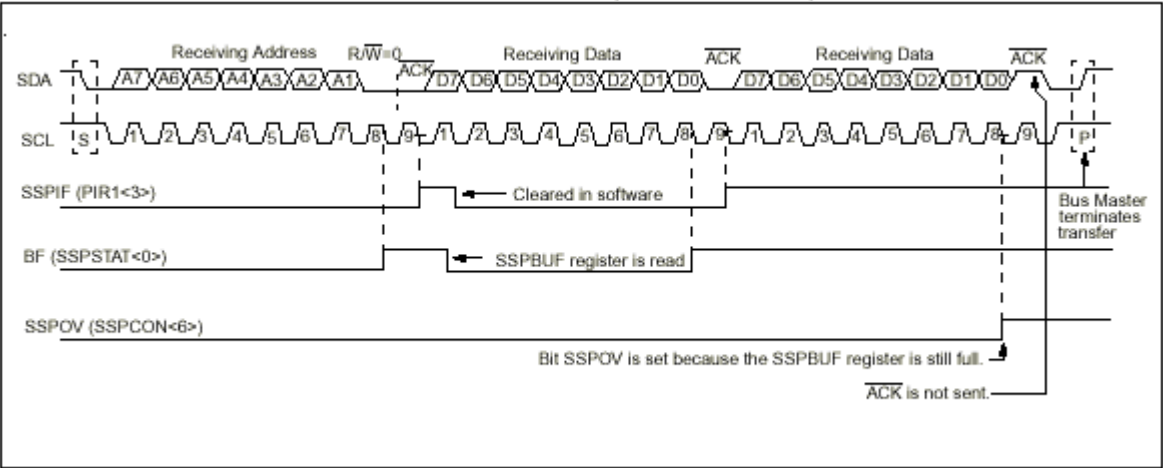


图 1.45 从属器 7 位地址方式的接收时序

(3) 发送

当地址字节后的读写标志位 R/W=1、且地址匹配时，SSPSTAT 中的 R/W 被置 1，接收到的地址数据装入 SSPBUF。从属器在 SCL 的第 9 个脉冲发出 ACK 信号并使 SCL 线保持为低。这时把欲发送的数据送入 SSPBUF（同时也送入了 SSPSR 中），然后 SCL 线被释放（通过置 SSPCON<4>=1），则 8 位的字节数据在 SCL 的下降沿被移位串行输出。SSP 每发送完一个字节，都会发中断请求（SSPIF=1），SSPIF 必须由软件再清为零。如果作为从属发送器，则从主控接收器发出的 ACK 信号在 SCL 线上的第 9 个脉冲上升沿被锁存。若 SDA 线为高电平，则表示无 ACK 信号，数据传输即告完成，而从属器即再检测另一次开始（START）信号。如果 SDA 线为低电平，即有 ACK 信号，发送数据则应置入 SSPUF 寄存器，然后再释放 SCL 线（通过置 SSPCON<4>=1），准备再发送。

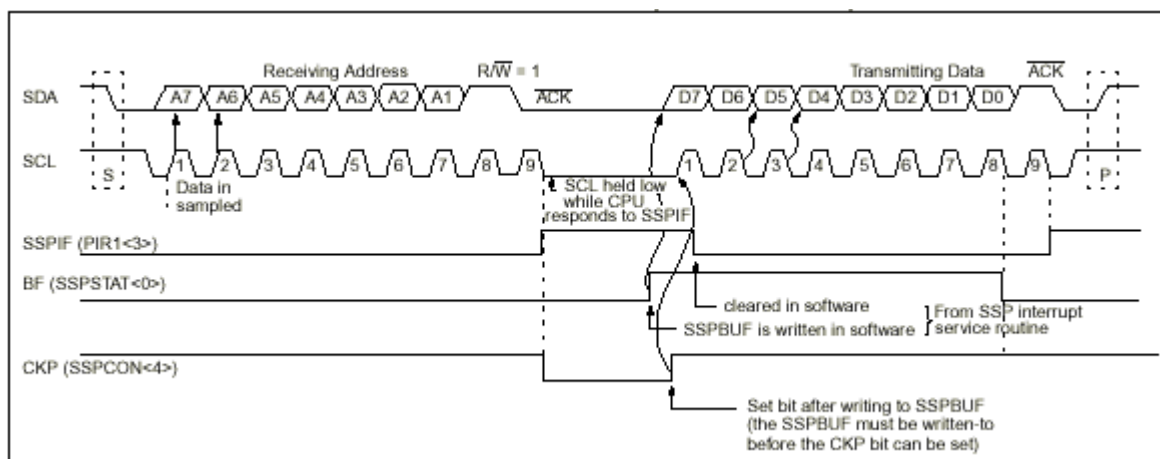


图 1.46 从属器 7 位地址方式的发送时序

（二）、主控器模式

主控器操作是通过检测 START 和 STOP 信号来进行的，在 SSPSTAT 寄存器中有 S 位和 P 位来标志检测的情况。复位或关闭 SSP 模块将会清零 S 位和 P 位。当 P 位置为 1 时，进入 I²C 总线控制操作，当 P 和 S 都为零时，I²C 总线处于闲置状态。在主机方式下，SCL 线和 SDA 线的电平状态是通过改变其相应的方向控制位 TRISC<4:3>来实现的。当置为输出态时，输出电平总为低而不理会 I/O 口寄存器的值。所以当传输“1”，应把其设为输入态（TRISC<4:3>=1），当传输“0”时，则把其设为输出态（TRIS<4:3>=0）。

下列情况会引起中断请求（SSPIF=1）：

1. START
2. STOP
3. 数据传输完一字节

选择主控器模式时，从属器模式可以关闭也可打开，详见 SSPCON<3:0>的描述。如果选择两者同时有效（SSPCON<3:0>=1110）则发生 SSP 中断请求时，软件还需判断其中断源。

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值	其他复位值
0Bh/8Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSP1F	(3)	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE	(3)	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
13h	SSPBUF	SSP 接收/发送寄存器								xxxx xxxx	uuuu uuuu
93h	SSPADD	SSP(I ² C 模式)地址寄存器								0000 0000	0000 0000
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000

87h	TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	1111 1111	1111 1111
94h	SSPSTAT	—	—	D/A	P	S	R/W	UA	BF	--00 0000	--00 0000

注： x=不定， u=不变， -=未用， 阴影为本模式无关位。

- (1) USART 仅 16C63/65 中有。
- (2) PSPIE 和 PSPIF 仅 16C64/65 中有。
- (3) 保留位，保持为“0”。

表 1.23 和 I²C 操作有关的寄存器

§ 1.12 串行通讯接口（SCI）模块

涉及型号				
63	63A	65	65A	67

在 PIC16C6X 中串行通讯接口模块简称 SCI 模块。SCI 是一种二线式的串行通讯方式，它可定义为如下工作方式：

1. 全双工异步方式---和 CRT 终端，PC 机等通讯。
2. 半双工同步（主控器或从属器）---便于和 A/D、D/A、串行 E²PROM 等通讯。

SCI 发送状态寄存器 TXSTA 如下：

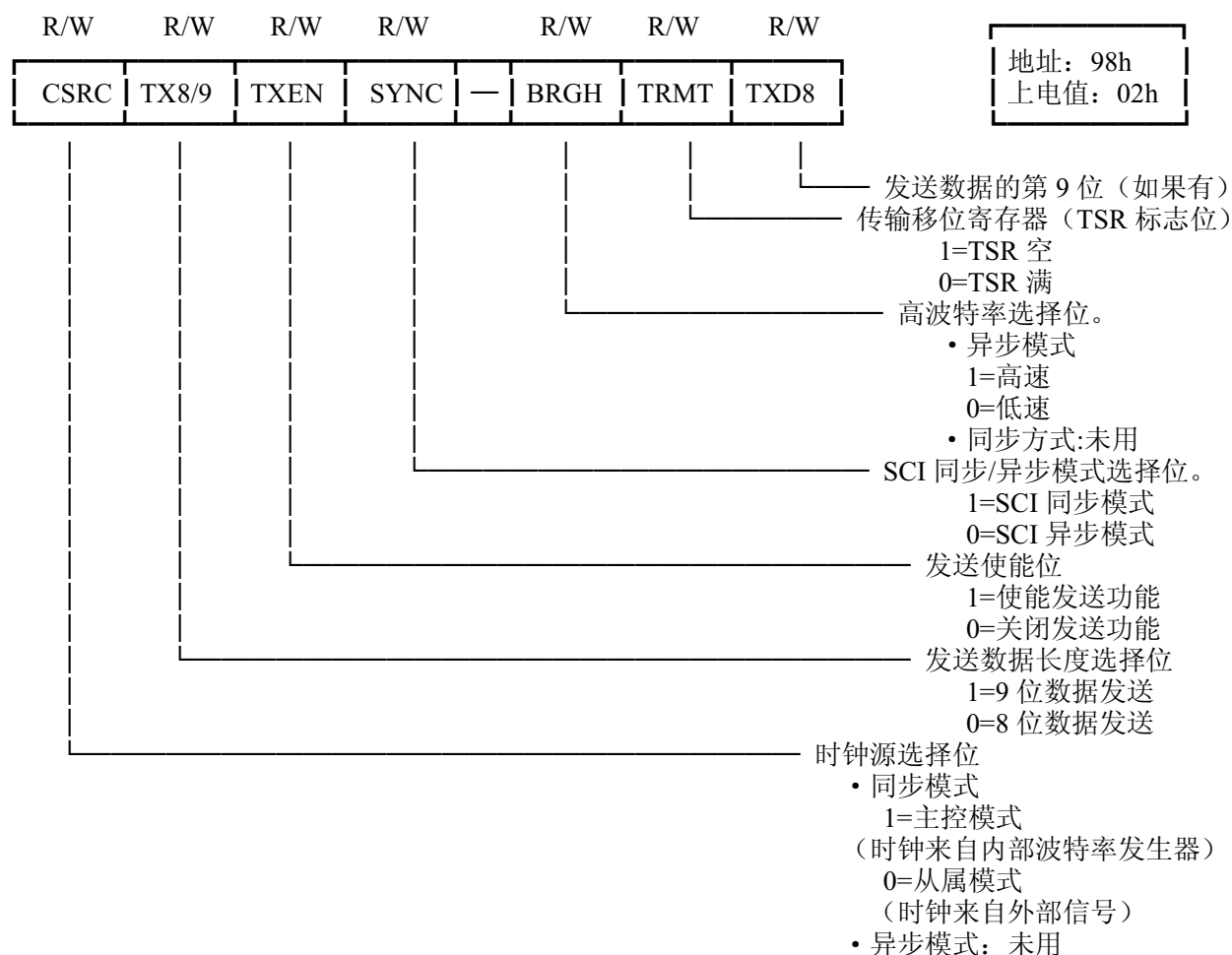


图 1.47 SCI 发送状态和控制寄存器 TXSTA

另外 SCI 还有一个接收状态和控制寄存器 RCSTA 如下：

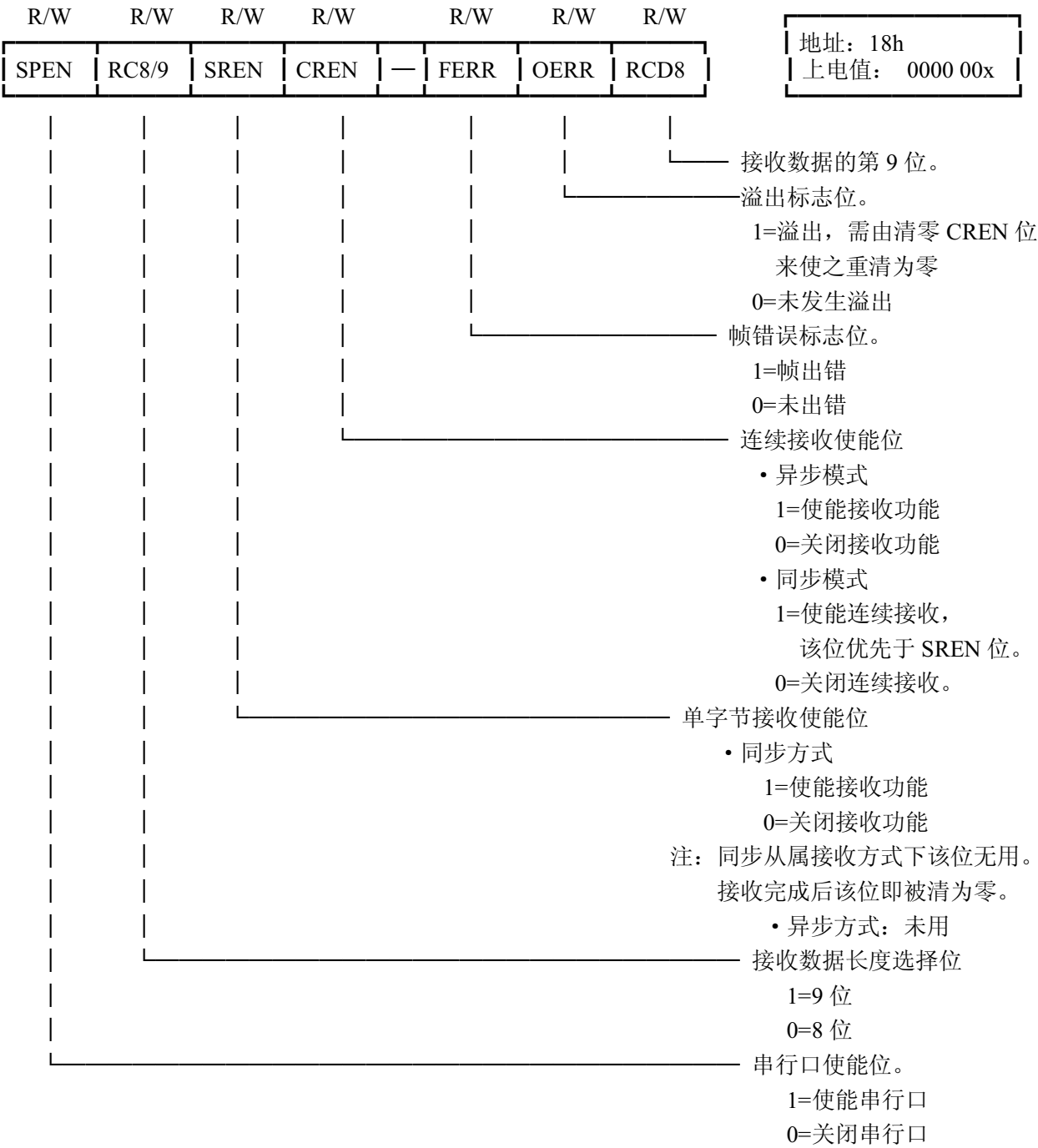


图 1.48 SCI 接收状态和控制寄存器 RCSTA

为使 RC6/TX/CK 和 RC7/RX/DT 工作在 SCI 下，必须置 SPEN（RCSTA<7>）=1。

§ 1.12.1 SCI 波特率产生器（BRG）

SCI 具有 8 位的波特率产生器，以下简称 BRG，支持 SCI 模式下的同步和异步工作方式。SPBRG 的值（8 位）决定波特率的值，见下表：

SYNC	BRGH=0（低速）	BRGH=1（高速）
0	（异步）波特率=Fosc/64（X+1）	波特率=Fosc/16（X+1）
1	（同步）波特率=Fosc/4（X+1）	无

表 1.24 波特率计算表（主控器模式）

表 1.25 和 BRG 有关的寄存器

300	—	—	—	—	—	—	—	—	—	—	—	
500	—	—	—	—	—	—	—	—	—	—	—	
HIGH	79.2	—	0	55.93	—	0	15.63	—	0	0.512	—	0
LOW	0.3094	—	255	0.2185	—	255	0.0610	—	255	0.0020	—	255

a. 同步方式下

BAUD RATE (K)	Fosc=20MHz				16MHz				10MHz				7.15909MHz			
	SPBRG 值				SPBRG 值				SPBRG 值				SPBRG 值			
	波特率	%误差	(十进制)		波特率	%误差	(十进制)		波特率	%误差	(十进制)		波特率	%误差	(十进制)	
0.3	0.31	+3.13	255		0.301	+0.23	185		0.300	+0.16	185		0.256	-14.67	1	
1.2	1.2	0	65		1.190	+0.83	46		1.202	+0.16	46		—	—	—	
0.3	—	—	—		—	—	—		—	—	—		—	—	—	
1.2	—	—	—		—	—	—		—	—	—		—	—	—	
2.4	—	—	—		—	—	—		—	—	—		—	—	—	
9.6	—	—	—		—	—	—		9.766	+1.73	255		9.622	+0.23	185	
19.2	19.53	+1.73	255		19.23	+0.16	207		19.23	+0.16	129		19.24	+0.23	92	
76.8	76.92	+0.16	64		76.92	+0.16	51		75.76	-1.36	32		77.82	+1.32	22	
96	96.15	+0.16	51		95.24	-0.79	41		96.15	+0.16	25		94.20	-1.88	18	
300	294.1	-1.96	16		307.69	+2.56	12		312.5	+4.17	7		298.3	-0.57	5	
500	500	0	9		500	0	7		500	0	4		—	—	—	
HIGH	5000	—	0		4000	—	0		2500	—	0		1789.8	—	0	
LOW	19.53	—	255		15.625	—	255		9.766	—	255		6.991	—	255	

BAUD RATE (K)	Fosc=5.0688MHz				3.579545MHz z				1MHz				32.768MHz			
	SPBRG 值				SPBRG 值				SPBRG 值				SPBRG 值			
	波特率	%误差	(十进制)		波特率	%误差	(十进制)		波特率	%误差	(十进制)		波特率	%误差	(十进制)	
0.3	—	—	—		—	—	—		—	—	—		0.303	+1.14	26	
1.2	—	—	—		—	—	—		1.202	+0.16	207		1.170	-2.48	6	
2.4	—	—	—		—	—	—		2.404	+0.16	103		—	—	—	
9.6	9.6	0	131		9.622	+0.23	92		9.615	+0.16	25		—	—	—	
19.2	19.2	0	65		19.04	-0.83	46		19.24	+0.16	12		—	—	—	
76.8	79.2	+3.13	15		74.57	-2.90	11		83.34	+8.51	2		—	—	—	
96	97.48	+1.54	12		99.43	+3.57	8		—	—	—		—	—	—	
300	316.8	+5.60	3		298.3	-0.57	2		—	—	—		—	—	—	
500	—	—	—		—	—	—		—	—	—		—	—	—	
HIGH	1267	—	0		394.9	—	0		250	—	0		8.192	—	0	
LOW	4.950	—	255		3.496	—	255		0.9766	—	255		0.032	—	255	

b. 异步方式下 (BRGH=0)

BAU D RATE (K)	Fosc=20MHz				16MHz				10MHz				7.15909MHz			
	SPBRG 值				SPBRG 值				SPBRG 值				SPBRG 值			
	波特率	%误差	(十进制)		波特率	%误差	(十进制)		波特率	%误差	(十进制)		波特率	%误差	(十进制)	
9.6	9.615	+0.16	129		9.615	+0.16	103		9.615	+0.16	64		9.520	-0.83	46	
19.2	19.230	+0.16	64		19.230	+0.16	51		18.939	-1.36	32		19.454	+1.32	22	
38.4	37.878	-1.36	32		38.461	+0.16	25		39.062	+1.7	15		37.286	-2.90	11	
57.6	56.818	-1.36	21		58.823	+2.12	16		56.818	-1.36	10		55.930	-2.90	7	
115.2	113.636	-1.36	10		111.111	-3.55	8		125	+8.51	4		111.860	-2.90	3	
250	250	0	4		250	0	3		—	—	—		—	—	—	
625	625	0	1		—	—	—		625	0	0		—	—	—	

1250	1250	0	0	—	—	—	—	—	—	—	—	—
------	------	---	---	---	---	---	---	---	---	---	---	---

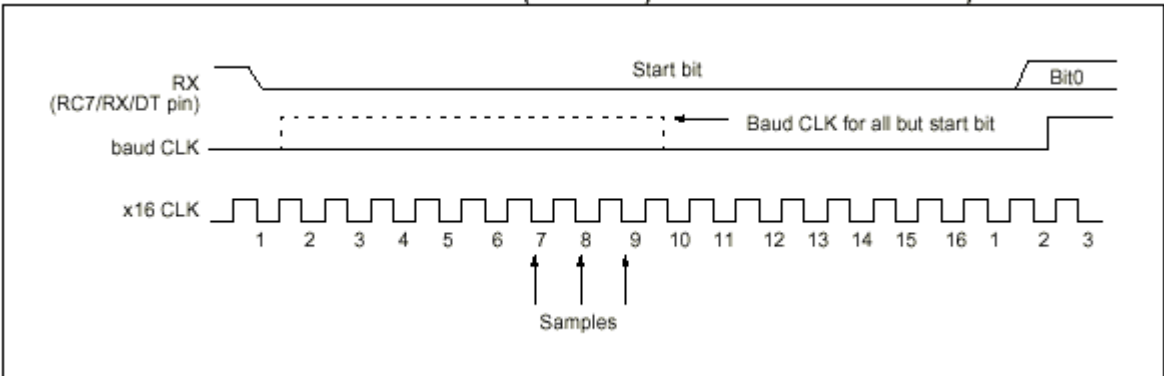
BAUD RATE (K)	Fosc=5.0688MHz			3.579545MHz z			1MHz			32.768MHz		
	SPBRG 值			SPBRG 值			SPBRG 值			SPBRG 值		
	波特率	%误差	(十进制)	波特率	%误差	(十进制)	波特率	%误差	(十进制)	波特率	%误差	(十进制)
9.6	9.6	0	32	9.727	+1.32	22	8.928	-6.99	6	—	—	—
19.2	18.645	-2.94	16	18.643	-2.90	11	20.833	+8.51	2	—	—	—
38.4	39.6	+3.12	7	37.286	-2.90	5	31.25	-18.61	1	—	—	—
57.6	52.8	-8.33	5	55.930	-2.90	3	62.5	+8.51	0	—	—	—
115.2	105.6	-8.33	2	111.860	-2.90	1	—	—	—	—	—	—
250	—	—	—	223.721	-10.51	0	—	—	—	—	—	—
625	—	—	—	—	—	—	—	—	—	—	—	—
1250	—	—	—	—	—	—	—	—	—	—	—	—

c. 异步方式下（BRGH=1）

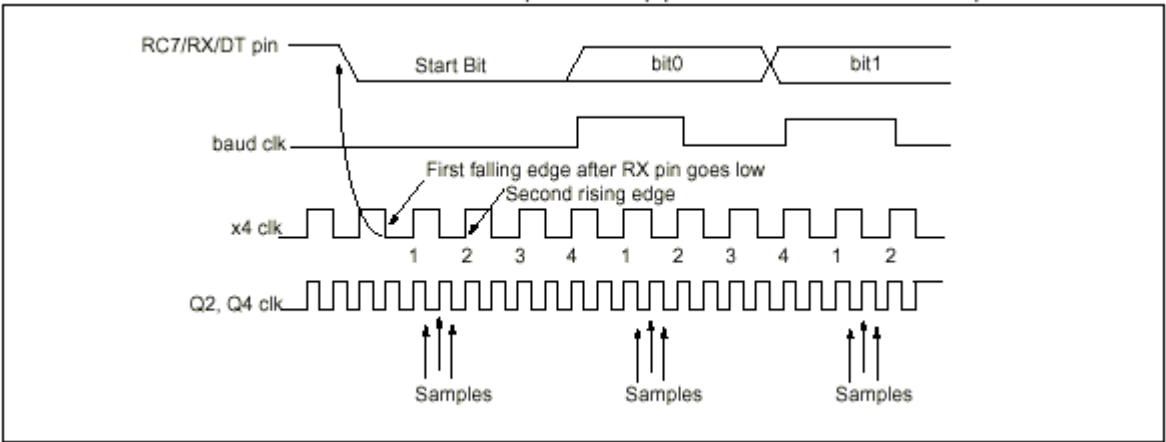
表 1.26 SCI 波特率表

§ 1.12.2 采 样

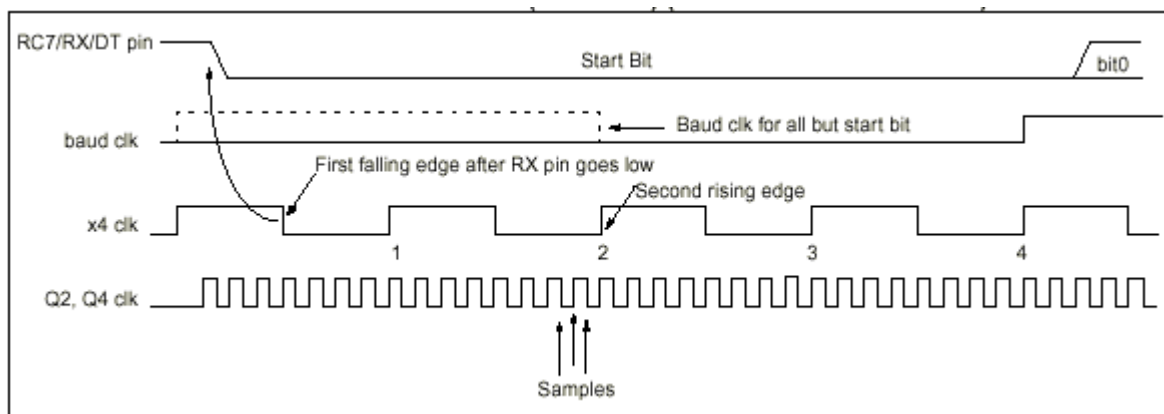
RX 线上的信号由一个多点采样电路采三次样，如下图所示：



a. BRGH=0



b. BRGH=1



c. BRGH=1

图 1.49 RX 脚上的采样时序

§ 1.12.3 SCI 异步方式

在读方式下，SCI 传输的数据格式是：一位起始位，8 或 9 位数据位以及一位停止位（最常用的格式数据是 8 位）。波特率发生器 BRG 产生用户所需的波特率。SCI 接收和发送的顺序是由低位（LSB）到高位（MSB）。发送和接收的数据格式一样，所使用的 BRG 也是同一个。SCI 硬件不直接支持奇偶校验位（Parity），但也以通过软件来实现（如作为第 9 位数据位）。在 CPU 处于睡眠（Sleep）下，波特率发生器 BRG 停止工作，所以 SCI 异步方式在睡眠状态下不能工作。

通过置 SYNC (TXSTA<4>) = 0 可选择异步工作方式。SCI 异步方式包括以下几项重要元素：

1. 波特率发生器 BRG
2. 采样电路
3. 异步发送器
4. 异步接收器

一、SCI 异步发送器

发送器的核心是“发送移位寄存器” TSR 和“发送缓冲器” TXREG。TXREG 是软件可读/写的寄存器，用户程序把要发送的数据送入 TXREG，然后再由 TXREG 置入 TSR 发送出去。TSR 要一直等到把目前正在发送的数据的停止位发出去后，才会从 TXREG 读入新的发送数据。一旦 TXREG 把数据送入 TSR 后，即发出中断请求（置 TXIF=1）。注意，TXIF 标志位不能由软件清零，只有当新的发送数据置入 TXREG 后，才由硬件清为零。TSR 的状态则由 TRMT (TXSTA<1>) 位标识，当 TRMT=1 表示 TSR 寄存器空（没有要发送的数据）。TRMT 位不会产生中断请求，用户程序必须用程序查询该位的值来判断 TSR 的情况。另外 TSR 寄存器是不可寻址的，即用户程序不能直接访问它（只能通过 TXREG 对它置数）。

置 TXEN (TXSTA<5>) = 1 可激活发送，不过真正的发送工作要等到 TXREG 已放有发送数据以及波特率发生器 BRG 发出移位脉冲后才开始，见下图。

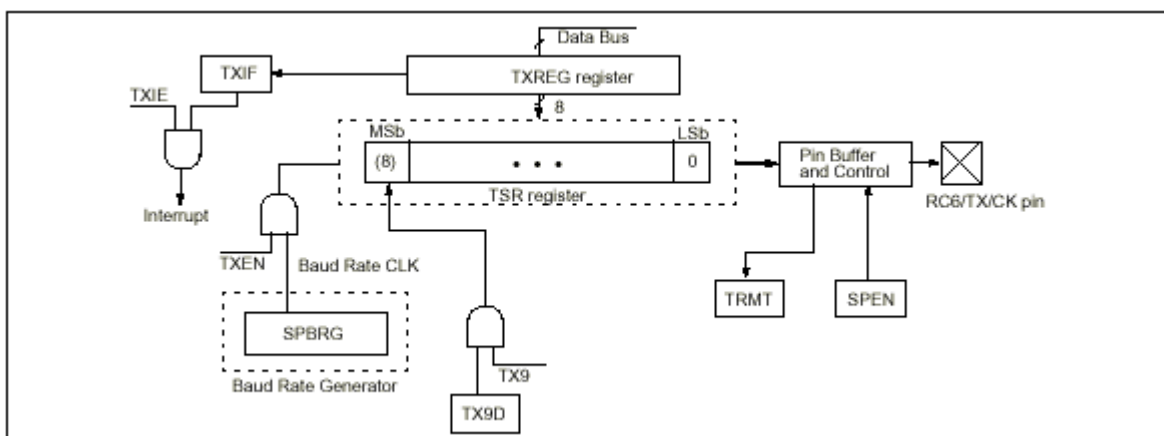
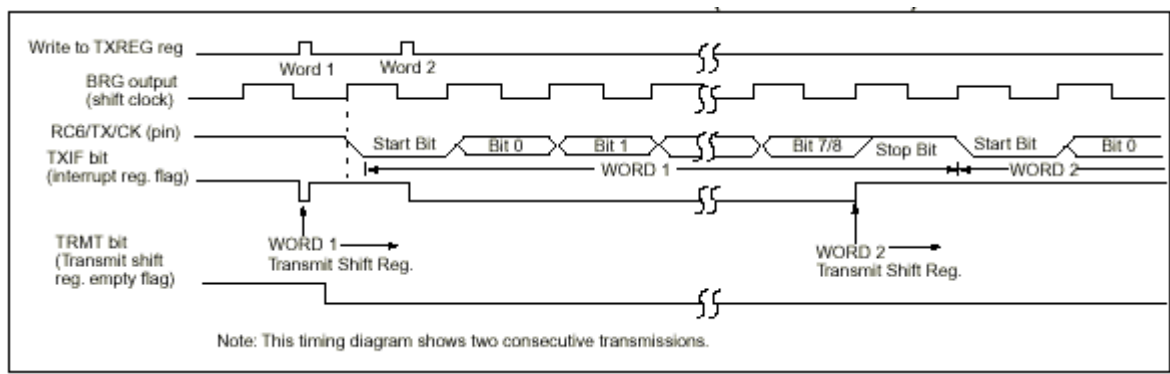


图 1.50 异步发送器结构图

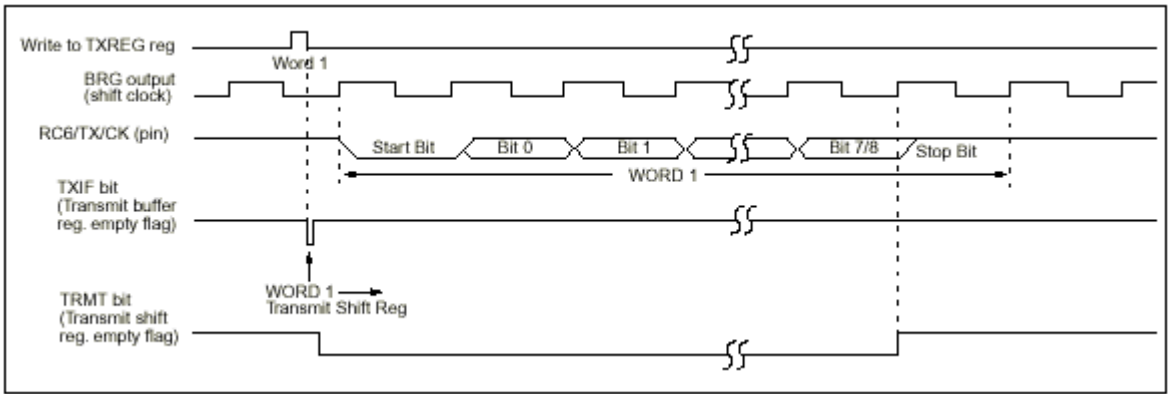
当然用户也可以先把发送数据送入 TXREG，然后再置 TXEN 为 1 来开始发送工作。一般当发送启动时，TSR 都是空的，一旦用户程序把数据置入 TXREG 后，即马上转送入 TSR，然后 TXREG 又是空的，所以二个数据可以连续发送，见图 1.49 所示。

如果在数据发送中用户程序清零 TXEN 位，则会造成发送工作中断，TX 线恢复为高阻态。如果要发送 9 位数据，须先设置 TX8/9 (TXSTA<6>) = 1，然后再把第 9 位数据置入 TXD8 (TXSTA<0>)。注意该第 9 位必须在前 8

位置入 TXREG 之前置入 TXD8，否则第 9 位数可能会出错。



a. 单个数发送



b. 两个数连续发送

图 1.51 异步主控器发送时序

综上所述，一旦选择异步发送方式，需做下面几点：

- 1. 选择波特率，然后把相应的值置入 SPBRG，如果是高速波特率，应置 BRGH=1。
- 2. 置 SYNC=0 及 SPEN=1。
- 3. 如果需要中断，置 TXIE (PIE1<4>) =1。
- 4. 如果要传送 9 位数据，置 TX8/9=1。
- 5. 置 TXEN=1，激活发送器。
- 6. 如果传送 9 位数据，这时要把第 9 位置入 TXD8。
- 7. 把发送数据（8 位）送入 TXREG。
- 8. 硬件开始自动发送（TX 线上有串行信号输出）。

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值	其他复位值
0Ch	PIR1	PSPIF	(2)	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00X	0000 -00X
19h	TXREG	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	0000 0000	0000 0000
8Ch	PIE1	PSPIE	(2)	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYN C	—	BRGH	TRMT	TX9D	0000 -010	0000 -010

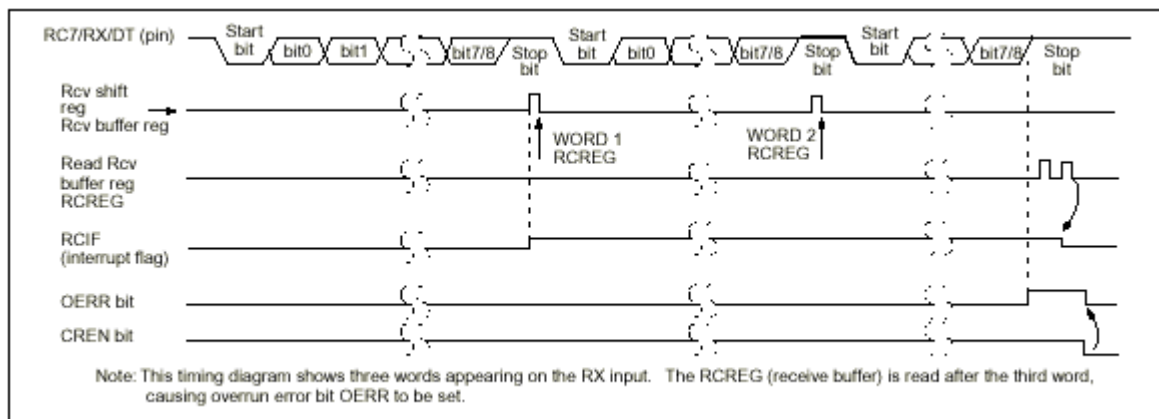


图 1.53 SCI 异步接收时序

综上所述，一旦选择异步接收方式，需做下面几点：

1. 选择波特率，然后算出 SPBRG 中应有的值并将其置入 SPBRG 中。如果是高速波特率，还应置 BRGH=1。
2. SYNC=0 及 SPEN=1。
3. 如需中断，置 RCIE (PIE1<5>) =1。
4. 如需接收 9 位数据，置 RX8/9=1。
5. 置 CREN=1，激活接收器。
6. 当一个字节接收完成后，发中断请求 (RCIF=1)。
7. 读 RCSTA 以便读取数据第 9 位（如果需要的话）以及判断是否发生任何接收上的错误。
8. 读 RCREG。
9. 如果发生任何接收错误，置 CREN=0 以清除错误。

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值	其他复位值
0Ch	PIR1	PSPIF	(2)	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00X	0000 -00X
1Ah	RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	0000 0000	0000 0000
8Ch	PIE1	PSPIE	(2)	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNCH	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	波特率发生器								0000 0000	0000 0000

注： x=不定， u=不变， -=未用， 阴影为本模式无关位。

(1) PSPIF 和 PSPIE 在 16C63 中未用。

(2) PIR1<6>和 PIE1<6>为保留位，恒为“0”。

表 1.28 和异步接收器有关的寄存器

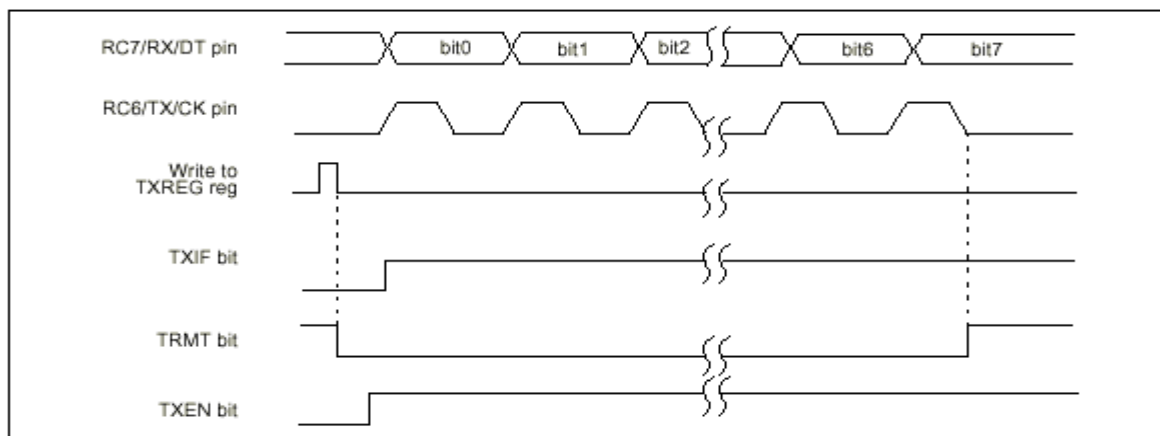
§ 1.12.4 SCI 同步主控模式

在同步主控器方式下，数据的传输是以半双工的方式进行。通过置 SYNC (TXSTA<1>) = 1 可选择该工作模式。另外还应置 SPEN (RCSTA<7>) =1，以使 RC6 脚和 RC7 脚可分别作为时钟线 CK 和数据线 DT，主控模式下由 CPU 发出时钟信号。

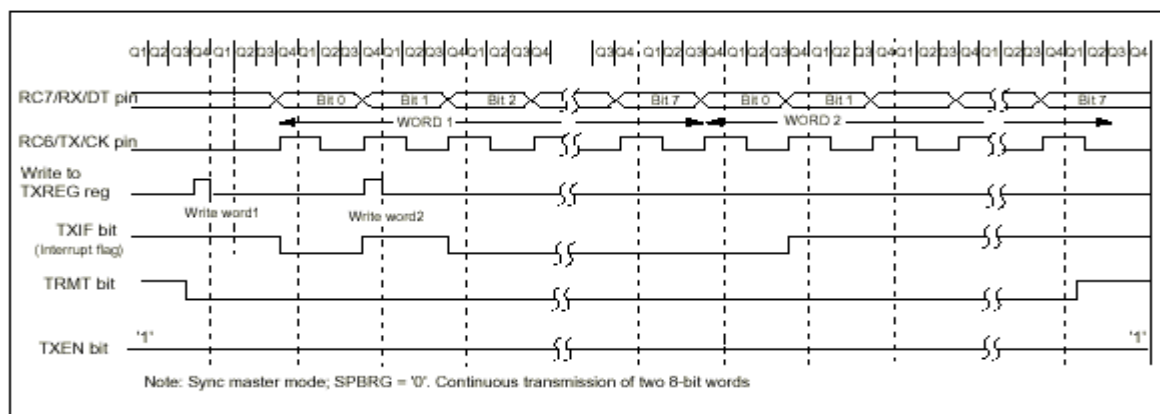
置 CSRC (TXSTA<7>) =1 即进入 SCI 同步主控方式。

一、SCI 同步主控发送

SCI 发送器结构见图 1.47。移位寄存器 TSR 从 TXREG 取得要发送的数据，一旦 TXREG 把数据传给 TSR 后，中断位 TXIF (PIR1(4))=1 发出中断请求。当新的数据由程序载入 TXREG 后，TXIF 清为零。TXIF 位表示了 TXREG 的状态（空或满），而 TRMT 位 (TXSTA<1>) 表示 TSR 的状态。当 TSR 中的数据移位送出后，TRMT=1，但是注意这并不会产生中断请求，所以用户程序必须靠程序查询方式来判断 TSR 是否为空，另外 TSR 不对用户程序开放，程序不能对它直接读/写。设置 TXEN (TXSTA<5>)=1 启动发送。但实际上要 TXREG 载入数据后才会开始发送。第一个数据位将在下一个时钟 (CK) 上升沿移出发送，并在下降沿后稳定下来，如下图所示。



a.



b

图 1.54 SCI 同步发送时序

用户也可以先把发送数据载入 TXREG，然后再置 TXEN=1 来启动发送，这样对于低波特率传送和连续传送会更好些。

在传送过程中如果用户程序清 TXEN=0 则会使发送中断并复位发送器。DT 线和 CK 线变成高阻态（输入态）。如果在数据发送过程中，接收使能位 CREN (RCSTA<4>) 和 SREN (RCSTA<5>) 被程序置为 1，则 DT 线变成输入态而 CK 线保持为输出态，发送器不会被复位。如果这时用户要复位发送器就要清 TXEN=0。

如果用户希望中止正在进行的数据传送转而接收一个外部送来的数，可以置 SREN=1。

当这个新数据接收完毕后 SREN 恢复为 0。同时由于这时 TXEN 位仍保持为 1，就会使 SCI 重新回到发送状态，DT 线将马上从输入态再转回到输出态。如果用户不希望回到发送状态，则应及时清 TXEN=0。

为了发送 9 位数据，TX8/9 (TXSTA<6>) 必须置为 1，并且把第 9 位数载入 TXD8 (TXSTA<0>)。注意，应先把第 9 位数据载入 TXD8，再把前 8 位载入 TXREG。这是因为一旦把前 8 位数置入 TXREG 后，要发送的 9 位数马上会转载入 TSR 中准备移位发送，所以如果先载前 8 位数进 TXREG，则 TSR 中载入的第 9 位数将是上次的第 9 位数而不是最新的。

综上所述，建立 SCI 同步主控发送需如下步骤：

1. 初始化波特率寄存器 SPBRG 以获得想要的波特率；
2. 使能同步主控串行口（置 SYNC=1、SPEN=1 和 CSRC=1）；
3. 如果需要中断功能，则置 TXIE=1；
4. 如果传送的数是 9 位，置 TX8/9=1；
5. 使能发送模式（置 TXEN=1）；

6. 如果传送 9 位数，把第 9 位数载入 TXD8；

7. 把 8 位数载入 TXREG，启动发送。

下表是和同步主控发送有关的寄存器。

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值	其他复位值
0Ch	PIR1	PSPIF	(2)	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00X	0000 -00X
19h	TXREG	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	0000 0000	0000 0000
8Ch	PIE1	PSPIE	(2)	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNCH	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	波特率发生器寄存器								0000 0000	0000 0000

注： X=不定， u=不变， —=未用， 阴影为本模式无关位。

(1) PSPIF 和 PSPIE 在 16C63 中未用。

(2) PIR1<6>和 PIE1<6>为保留位，恒为“0”。

表 1.29 SCI 同步主控发送相关的寄存器

二、SCI 同步主控接收

置 SREN (RCSTA<5>) =1 或 CRENR (STA<4>) =1 即进入同步主控接收，CPU 在 CK 的下降沿采样 DT 线上的信号。如果 SREN=1，将仅接收一个字节，如果 CRENR=1，则连续地接收数据，直至 CRENR 被清为 0。如果 SREN 和 CRENR 都置为 1，则以 CRENR 有效而进行连续接收。

一个数的所有位都被采样后，CPU 即把接收移位寄存器 RSR 的内容（收到的数）置入 RCREG 寄存器，然后置 RCIF (PIR1<5>) =1 而发出中断请求。当用户程序把 RCREG 寄存器中的数取走后，RCIF 位即清为 0，以准备下一个数的接收。实际上 RCREG 是一个二层的先进先出缓冲寄存器，所以它允许存放二个接收到的数据，同时还有第三个数还在接收中，用户程序可以连续读二次 RCREG 寄存器，把二个数取走。如果第三个数接收完毕，而 RCREG 中仍有二个数未被读取，则标志位 OERR (RCSTA<1>) =1，这会禁止 RSR 中的数据转载入 RCREG，这样第三个数据（在 RSR 中）可能会弄失。OERR 位是只读位，程序必须通过清 CRENR=0 来清零 OERR 位。

如果接收的是 9 位数，则第 9 位数放在 RCD8 (RCSTA<0>) 中，用户必须先读 RCD8 以取得第 9 位，然后再读 RCREG 取得前 8 位。

综上所述，SCI 主控接收应按以下步骤：

1. 初始化 SPBRG 以获得想要的波特率；
2. 使能同步主控串行口（置 SYNC=1、SPEN=1 和 CSRC=1）；
3. CRENR=SREN=0；
4. 如需要中断功能，置 RXIE=1；
5. 如要接收 9 位数，置 RX8/9=1；
6. 如只要接收一个数，置 SREN=1；如果要连续接收数据，则置 CRENR=1；
7. 一个数接收完成，置 RCIF=1，发出中断请求；
8. 读 OERR (RCSTA<1>)，看是否发生错误；
9. 如果接收 9 位数，则读取 RCD8 (RCSTA<0>)；

10. 如果发生错（OERR=1），则清 CREN=0 以清零 OERR 位。

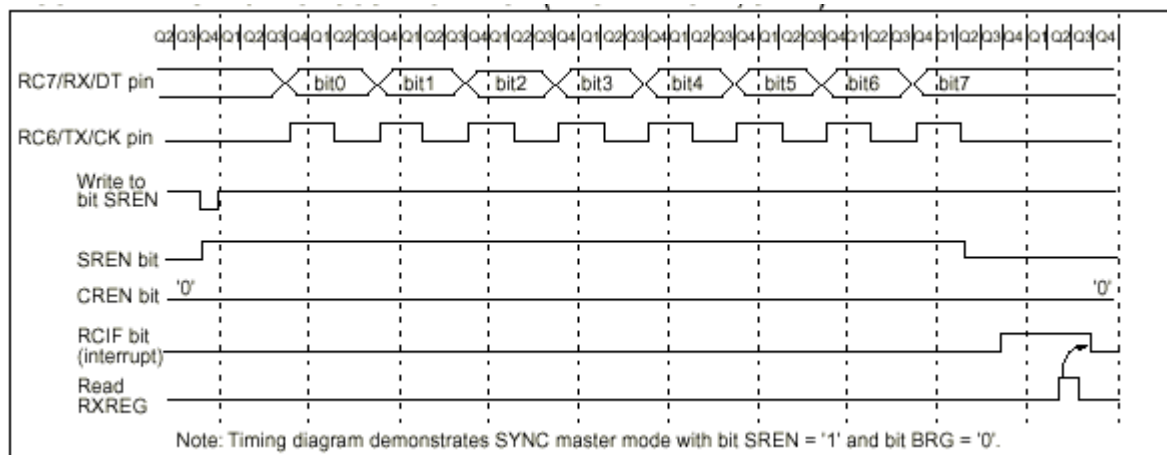


图 1.55 SCI 同步主控接收时序

下表列出了和同步主控接收有关的寄存器。

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值	其他复位值
0Ch	PIR1	PSPIF	(2)	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00X	0000 -00X
1Ah	RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	0000 0000	0000 0000
8Ch	PIE1	PSPIE	(2)	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNCH	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	波特率发生器								0000 0000	0000 0000

注：X=不定，u=不变，-=未用，阴影为本模式无关位。

(1) PSPIF 和 PSPIE 在 16C63 中未用。

(2) PIR1<6>和 PIE1<6>为保留位，恒为“0”。

表 1.30 SCI 同步主控接收相关寄存器

§ 1.12.5 SCI 同步从属模式

同步从属模式和主控模式的不同之处是其时钟信号（CK）是由外部提供的。这样即使 CPU 在睡眠（Sleep）状态下仍然可以发送或接收数据。

清 CSRC（TXSTA<7>）=0，则进入同步从属模式。

一、SCI 同步从属发送

同步从属发送和同步主控发送基本上是一样的，只有一点不同，那就是从属方式下时钟信号由外部供给而非使用单片机内部的时钟信号，所以在单片机睡眠状态下，从属发送仍然可以进行。

假设目前已有二个要发送的数放在 TXREG 寄存器中，此时如果执行 Sleep 指令而使单片机进入睡眠，则 TXREG 中的第一个数据将马上转置入发送移位寄存器 TSR 中进行发送，第二个数据仍保留在 TXREG 中。这时候 TXIF 不像平常那样被置成 1，即不发中断请求。一直等到第一个数发送完毕，TXREG 中的第二个数据再转置入 TSR 中发送，此时就会置 TXIF=1，发出中断请求。如果该中断是允许的（TXIE=1），这个中断请求将唤醒睡眠中的单片机，并跳到中断向量入口地址（0004H）处执行中断例程。

下面是建立同步从控发送的步骤：

1. 置 SYNC=1，SPEN=1 和 CSRC=0；
2. CREN=SREN=0；
3. 如果需要中断，置 TXIE=1；
4. 如果要发送 9 位数，置 TX8/9=1；

5. 置 TXEN=1，使能传送功能；
6. 如果发送 9 位数，将第 9 位置入 TXD8；
7. 把发送的数置入 TXREG，开始发送。

下表列出了和同步从属发送有关的寄存器。

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值	其他复位值
0Ch	PIR1	PSP1F	(2)	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00X	0000 -00X
19h	TXREG	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	0000 0000	0000 0000
8Ch	PIE1	PSPIE	(2)	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	波特率产生器寄存器								0000 0000	0000 0000

注：X=不定，u=不变，—=未用，阴影为本模式无关位。

(1) PSPIF 和 PSPIE 在 16C63 中未用。

(2) PIR1<6>和 PIE1<6>为保留位，恒为“0”。

表 1.31 同步从属发送相关寄存器

二、SCI 同步从属接收

同步从属接收和同步主控接收基本上也是一样，也是只在单片进入睡眠状态时它们才有区别。另外，在从属模式下，SREN 位不起作用。

假设在执行 SLEEP 指令前单片机处于接收状态 (CREN=1)，则单片机进入睡眠后仍可以接收一个外部送来的数据。当该数据接收完成后，RSR 中的数据将转置入 RCREG 并产生中断请求 (RCIF=1)，如果中断是允许的 (RCIE=1)，那么这个中断请求就会唤醒单片机，并转入中断例程。

进行同步从属接收的步骤如下：

1. 置 SYNC=1，SPEN=1 和 CSRC=0；
2. 如果需要中断，置 RCIE=1；
3. 如果要接收 9 位数，置 RX8/9=1；
4. 置 CREN=1 使能接收功能；
5. 接收完成后，置 RCIF=1，发中断请求；
6. 如果允许中断(RCIE=1)则进入中断例程；
7. 如果是 9 位数，读入第 9 位 RCD8；
8. 读取 RCREG 中的 8 位数据；
9. 如果 OERR=1，清 CREN=0 以清零 OERR 位。

下表列出了和同步从属接收有关的寄存器：

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	上电复位值	其他复位值
0Ch	PIR1	PSP1F	(2)	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00X	0000 -00X

1Ah	RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	0000 0000	0000 0000
8Ch	PIE1	PSPIE	(2)	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNCH	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	波特率发生器寄存器								0000 0000	0000 0000

注： X=不定， u=不变， -=未用， 阴影为本模式无关位。

- (1) PSPIF 和 PSPIE 在 16C63 中未用。
- (2) PIR1<6>和 PIE1<6>为保留位，恒为“0”。

表 1.32 同步从属接收相关寄存器

§ 1.13 CPU 的特性

PIC16CXX 单片机集成了一系列优良的微处理器特性，以使其应用更经济、更方便、更可靠。下面就各种特性逐一细述。

§ 1.13.1 系统定义字（Configuration）

在 PIC16CXX 中有一个 13 位长的字节，内含系统定义位，用来定义单片机的一些系统性能，见下图所示：

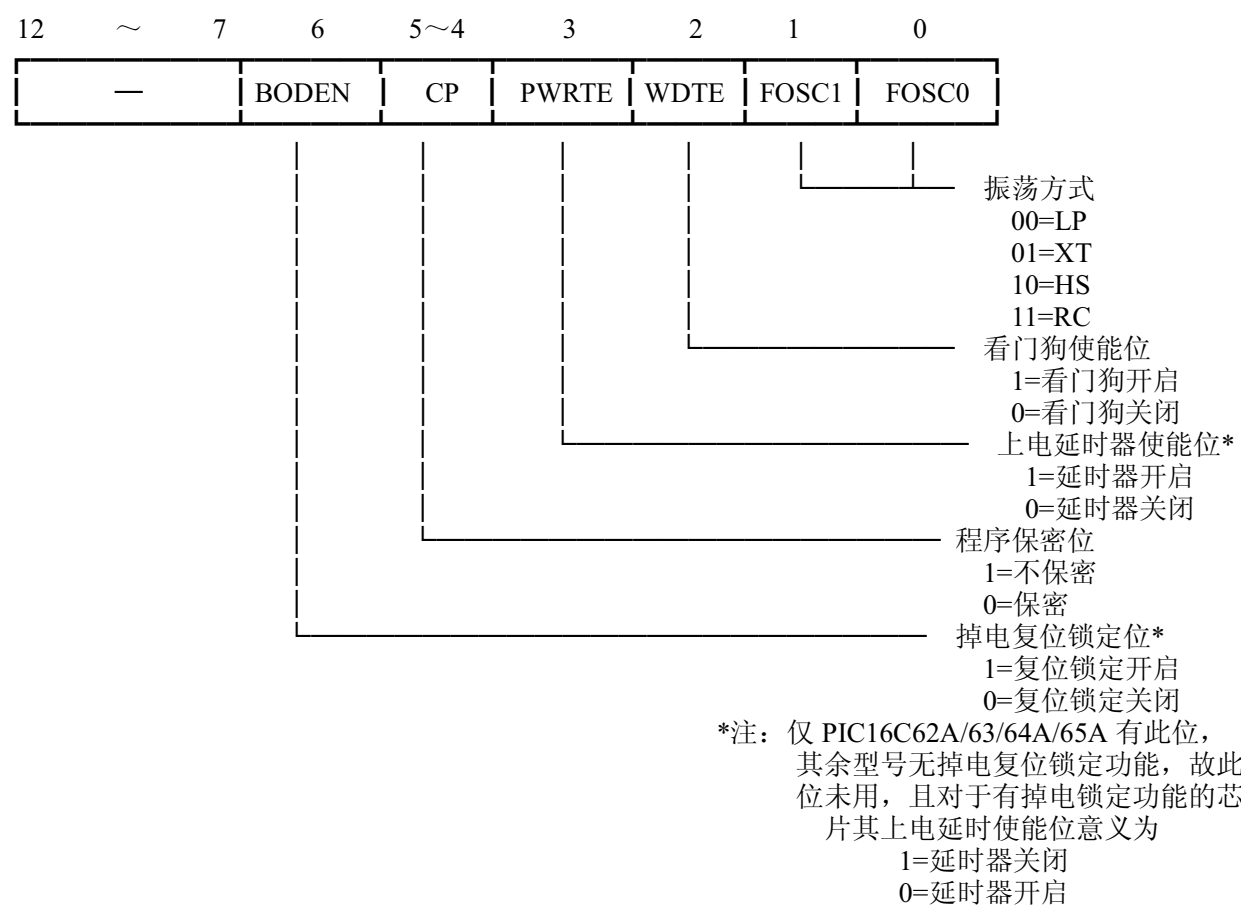


图 1.56 系统定义字（Configuration）

系统定义字的地址在 2007H，不占单片机的程序存储器空间，属于特殊的测试/定义空间，不能由程序指针（PC）

所访问，用户可以用烧写器对其进行编程，参见开发工具篇的描述。
关于系统定义字节各种定义的意义，下面分别还要详细描述。

§ 1.13.2 振 荡

一、振荡类型

PIC16CXX 可以运行在 4 种类型的振荡方式：

- 1. LP — 低频晶体振荡
- 2. XT — 标准晶体/陶瓷振荡
- 3. HS — 高速晶体/陶瓷振荡
- 4. RC — 阻容振荡

二、连接方式

1. 晶体和陶瓷振荡的连接如下图所示：

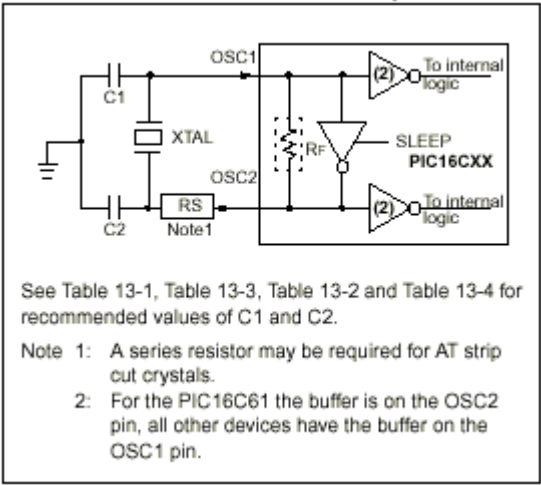


图 1.57 晶体/陶瓷振荡连接

下表列出了在各种情况下 C1 和 C2 的值。

振 荡	频 率	C1	C2
XT	455KHz	68-100P	68-100P
	2.0MHz	15-68P	15-68P
	4.0MHz	15-68P	15-68P
HS	8.0MHz	10-68P	10-68P
	16.0MHz	10-22P	10-22P

a. 陶瓷振荡下

振 荡	频 率	C1	C2
LP	32KHz	33-69P	33-68P
	200KHz	15-47P	15-47P
XT	100KHz	47-100P	47-100P
	500KHz	20-68P	20-68P
	1MHz	15-68P	15-68P
	2MHz	15-47P	15-47P
	4MHz	15-33P	15-33P
HS	8MHz	15-47P	15-47P
	20MHz	15-47P	15-47P

b. 晶体振荡下

表 1.33 C1 和 C2 推荐值

电容值大有利振荡的稳定，但却延长了起振时间。上表的数据供设计时参考，可满足一般的要求。

2. 外部振荡

外部振荡信号（仅限于 HS、XT 和 LP）可以从 OSC1 端输入，如下图所示：

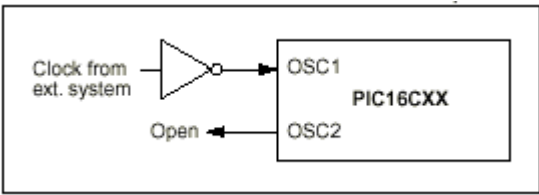


图 1.58 外部振荡电路

3. RC 振荡

这种振荡成本最低，但频率的精确性较差，易受工作环境的影响，适用于时序精度要求不高的应用场合。RC 振荡的频率是 VDD、RC 值及环境湿度的函数，并且每个芯片也会有所不同。

RC 振荡是在 OSC1 所连一个串联的电阻电容电路，如下图所示：

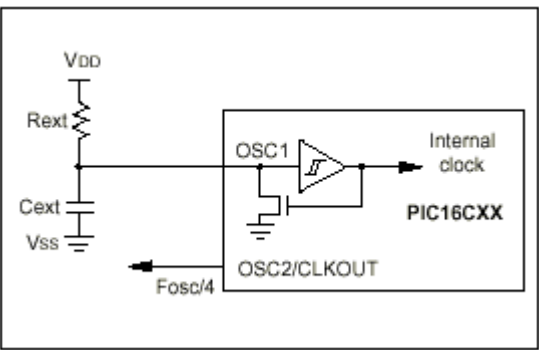


图 1.59 RC 振荡电路

电阻如果低于 2.2KΩ，振荡会不稳定，甚至不能振荡。但是如果高于 1MΩ，则振荡又易受干扰。所以电阻值最好取 5K~100K 之间。尽管电容 C 值为 0 时，电路也能振荡，但易受干扰且不稳定。RC 值和频率关系如下表所示：

C	R	频率（5V，25℃）	误差率
20P	4.7K	4.52MHz	±17%
	10K	2.47MHz	±10%
	100K	290.86KHz	±12%
100P	3.3K	4.52MHz	±9.5%
	4.7K		±9.8%
	10K		±11%
	100K		±16%
300P	3.3K	726.89KHz	±11%
	100K	33.82KHz	±11%

表 1.34 RC 频率表

另外采用 RC 振荡时，OSC2 端可输出 OSC1 的 4 分频信号。

§ 1.13.3 复 位

PIC16C6X 片内都集成有“上电复位”电路（POR），对于一般的应用，只要把 MCLR 端接高电位即可。对于某些特殊的需要，也可以在 MCLR 端外部增加外部上电复位电路。

一、复位的条件和原因

1. 芯片上电；
2. 程序运行中 MCLR 端加低电平；
3. 芯片处于睡眠时 MCLR 端加低电平；
4. 程序运行中看门狗（WDT）超时溢出；
5. 掉电锁定复位。

有一些寄存器的值不受任何一种复位的影响，当芯片上电复位后，它们的值是随机不定的，而在其他形式的复位后其值则保持不变。另一些寄存器在以上描述的前 4 种复位后都会等于一个固定的“复位值”，但第 5 种情形的复位仍然不会改变它们的值，因为这种复位是一种系统恢复继续运行下去的方式，故不应使任何寄存器的值产生变化。状态寄存器 STATUS 中的 TD 和 PD 位则根据不同的复位方式有不同的值，请参阅图 1.6。

二、内部上电复位（POR）

PIC16CXX 内置上电复位电路 POR，当芯片上电后 VDD 上升到一定值（一般在 1.6V~1.8V），POR 即产生复位脉冲，见下图 POR 结构。

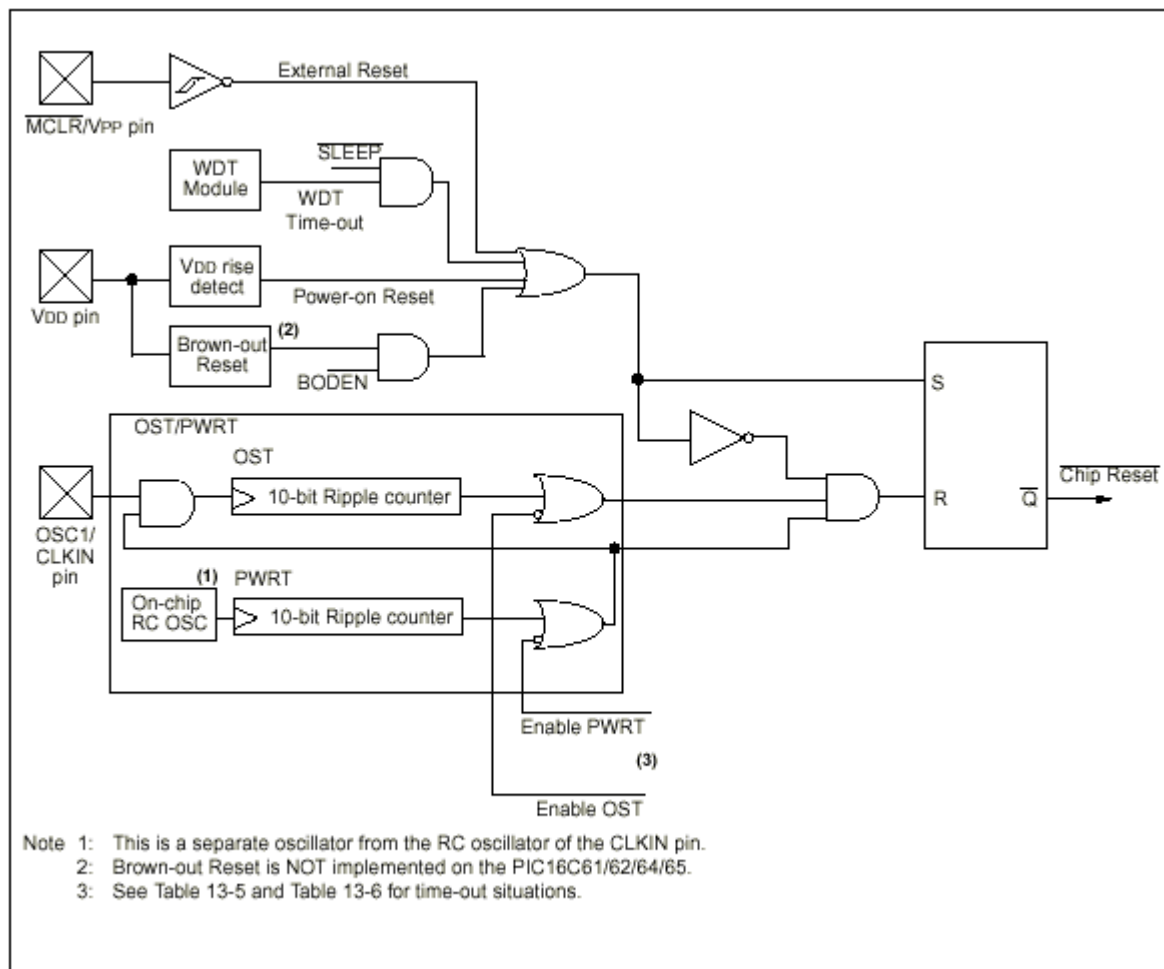


图 1.60 内部复位电路

对于一般应用，可把 MCLR 直接（或通过上拉电阻）连到 VDD 上即可，这样可以节省外围电路。

三、上电延时器（PWRT）

PWRT 提供固定的 72ms 的上电定时延迟，为的是使 VDD 可以有足够时间上升到一个对芯片合适的电压值。在系统定义字中的 PWRTE 位可以使能或关闭这个延时器。

四、振荡起振定时器（OST）

在 PWRT 延迟之后，OST 另提供 1024Tosc 的延迟，目的是让振荡电路有足够的时间建立稳定的振荡。但仅在 XT、LP 和 HS 振荡方式下，并且是上电复位或是从睡眠中唤醒的复位才会启动 OST 定时器计数。

五、上电复位延时时序

上电复位的延时时序如下：首先是 PWRT 延时 72ms，然后是 OST 启动延时 1024Tosc。当然 PWRT 用户可编程选择是否起作用的，另外在 RC 振荡方式下，OST 也不起作用（关闭），所以上电复位的延时由一系列因素共同决定，见下表：

振荡方式	电 源 上 电 复 位		睡眠唤醒复位
	PWRT=1	PWRT=0	
XT、HS、LP	72ms+1024Tosc	1024Tosc	1024Tosc
RC	72ms	—	—

表 1.35 上电复位延时

上电复位延时电路都是由 POR 复位脉冲启动的（即当 VDD 上升到一定值时内部复位电路发出的复位脉冲）。如果 MCLR 端直接接 VDD，则整个复位时序图如下：

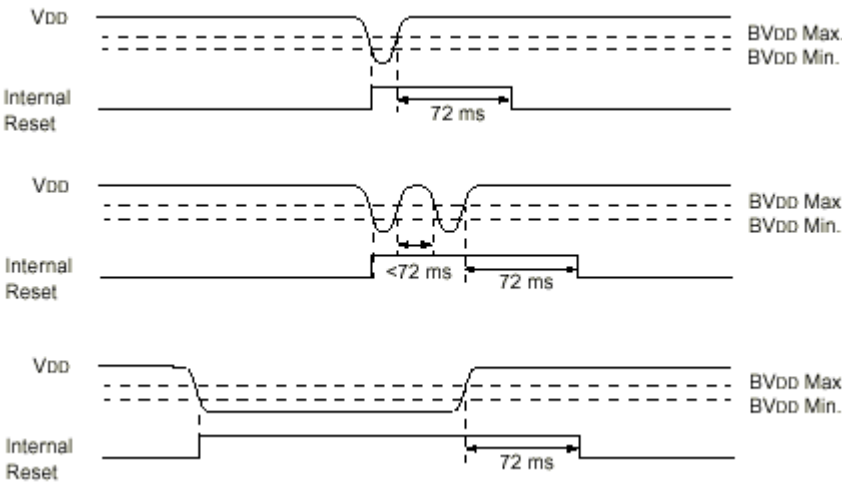


图 1.61 上电复位时序（MCLR 联到 VDD）

在某些应用中我们可能需要延长复位的时间，这时可以在 MCLR 端外接复位电路，下面是一个例图。

- 注：①二极管 D 使电容能在 VDD 掉电时快速放电。
②取 $R < 40K\Omega$ ，以保证其压降不大于 0.2V。
③R1 取 $100\Omega \sim 1K\Omega$ ，用来限制静电造成的电容 C 放电电流。

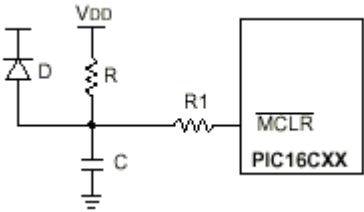
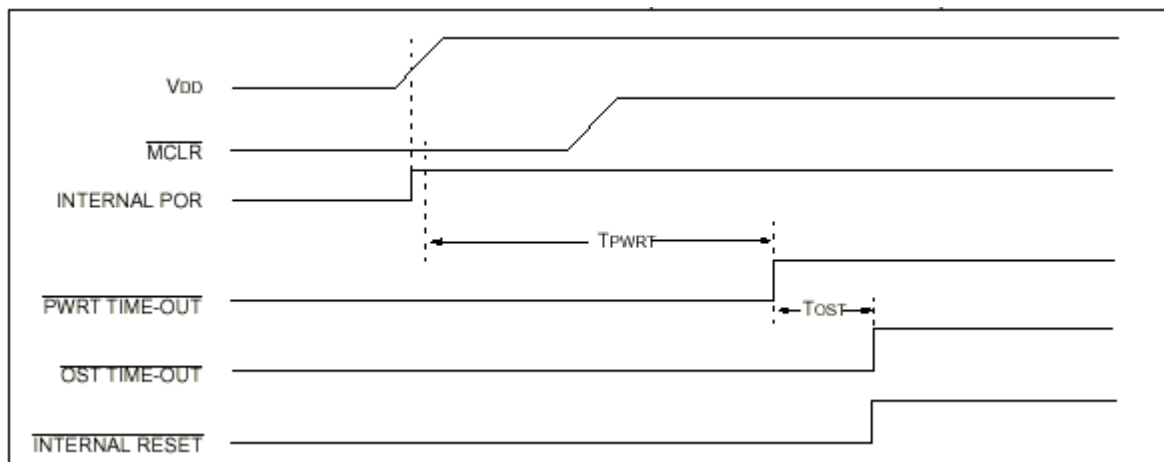
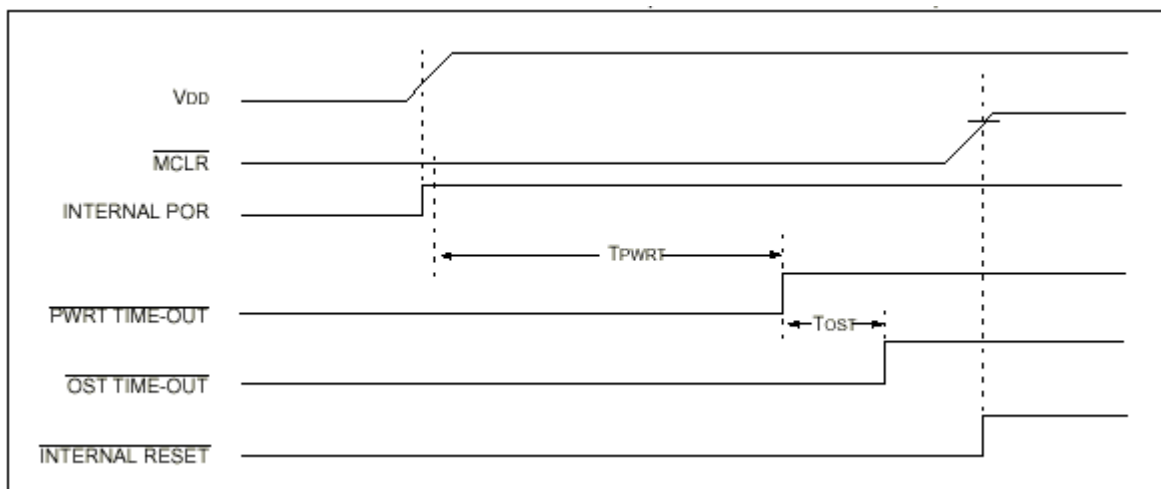


图 1.62 外部上电复位电路

这个电路一般是针对 VDD 上升较慢的应用，它可以保障复位完成后，VDD 已上升到合适的电压值。过程是这样的：当 MCLR 升到高电平后，PWRT 和 OST 延时已经完成，VDD 也升到合适的值，这样芯片将可以正常地进入运行状态，如下图所示：



a.



b.

图 1.63 上电复位时序（MCLR 外接延时电路）

六、复位标志位

我们在 § 1.7.2 讲述了特殊功能寄存器，其中状态寄存器 STATUS 和上电标志寄存器 PCON 含有复位标志位 TO、PD 和 POR，请参阅前面的叙述。PCON 在 16C61 中没有。当芯片上电复位后，POR 位被清为 0，表示发生了上电过程。程序应将其置为 1，以便它能标识下一次的上电复位。下表是各标志位在各种复位后的情况：

POR	BOR	TO	PD	复 位 原 因
0	X	1	1	上电复位
0	X	0	X	非法组合
0	X	X	0	非法组合
1	0	X	X	掉电锁定复位
1	1	0	1	看门狗复位
1	1	0	0	看门狗唤醒
1	1	1	1	非睡眠下的 MCLR 端拉低复位上电复位
1	1	1	0	睡眠下的 MCLR 端拉低复位上电复位

注：POR 位在 16C61 中没有

表 1.36 复位标志位

七、复位后的寄存器值

各种复位后特殊功能寄存器的值或变化情况如下表所示：

复 位 / 唤 醒	程序计数器	状态寄存器	PCON 寄存器
上电复位	000h	0001 1xxx	---- --0-
正常运行时 MCLR 拉低复位	000h	000u uuuu	---- --u-
睡眠时 MCLR 拉低复位	000h	0001 0uuu	---- --u-
看门狗溢时复位	000h	0000 1uuu	---- --u-
睡眠时看门狗溢时唤醒	PC+1	uuu0 0uuu	---- --u-
睡眠时中断唤醒	PC+1 (1)	uuu1 0uuu	---- --u-

注 1：如果 GIE=1，则 PC=0004h（中断向量）

a. 16C61/62/64/65

复 位 / 唤 醒	程序计数器	状态寄存器	PCON 寄存器
上电复位	000h	0001 1xxx	---- --0x
正常运行时 MCLR 拉低复位	000h	000u uuuu	---- --uu
睡眠时 MCLR 拉低复位	000h	0001 0uuu	---- --uu
看门狗溢时复位	000h	0000 1uuu	---- --uu
掉电复位锁定	000h	0001 1uuu	---- --u0
睡眠时看门狗溢时唤醒	PC+1	uuu0 0uuu	---- --uu
睡眠时中断唤醒	PC+1 (1)	uuu1 0uuu	---- --uu

注 1：如果 GIE=1，则 PC=0004h（中断向量）

b. 16C62A/63/64A/65A

Register	Applicable Devices														Power-on Reset Brown-out Reset	MCLR Reset during: – normal operation – SLEEP WDT Reset	Wake-up via interrupt or WDT Wake-up
W	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	N/A	N/A	N/A
TMR0	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000h	0000h	PC + 1 ⁽²⁾
STATUS	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0001 1xxx	000q quuu ⁽³⁾	uuuq quuu ⁽³⁾
FSR	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	--x xxxx	--u uuuu	--u uuuu
	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	--xx xxxx	--uu uuuu	--uu uuuu
PORTB	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTD	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTE	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	---- -xxx	---- -uuu	---- -uuu
PCLATH	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	---0 0000	---0 0000	---u uuuu
INTCON	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000 000x	0000 000u	uuuu uuuu ⁽¹⁾
PIR1	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	00-- 0000	00-- 0000	uu-- uuuu ⁽¹⁾
	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000 0000	0000 0000	uuuu uuuu ⁽¹⁾
PIR2	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	---- --0	---- --0	---- --u ⁽²⁾
TMR1L	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1H	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	--00 0000	--uu uuuu	--uu uuuu
TMR2	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000 0000	0000 0000	uuuu uuuu
T2CON	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	-000 0000	-000 0000	-uuu uuuu
SSPBUF	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPCON	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000 0000	0000 0000	uuuu uuuu
CCPR1L	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1H	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	--00 0000	--00 0000	--uu uuuu
RCSTA	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000 -00x	0000 -00x	uuuu -uuu
TXREG	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000 0000	0000 0000	uuuu uuuu
RCREG	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000 0000	0000 0000	uuuu uuuu
CCPR2L	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2H	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2CON	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000 0000	0000 0000	uuuu uuuu
OPTION	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	1111 1111	1111 1111	uuuu uuuu
TRISA	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	---1 1111	---1 1111	---u uuuu
	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	--11 1111	--11 1111	--uu uuuu
TRISB	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	1111 1111	1111 1111	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit read as '0', q = value depends on condition.

Note 1: One or more bits in INTCON, PIR1 and/or PIR2 will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the global enable bit, GIE is set, the PC is loaded with the interrupt vector (0004h) after execution of PC + 1.

3: See Table 13-10 and Table 13-11 for reset value for specific conditions.

c. 各种复位后特殊寄存器的值

Register	Applicable Devices														Power-on Reset Brown-out Reset	MCLR Reset during: – normal operation – SLEEP WDT Reset	Wake-up via interrupt or WDT Wake-up
TRISC	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	1111 1111	1111 1111	uuuu uuuu
TRISD	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	1111 1111	1111 1111	uuuu uuuu
TRISE	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000 -111	0000 -111	uuuu -uuu
PIE1	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	00-- 0000	00-- 0000	uu-- uuuu
	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000 0000	0000 0000	uuuu uuuu
PIE2	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	---- --0	---- --0	---- --u
PCON	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	---- --0u	---- --uu	---- --uu
	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	---- --0-	---- --u-	---- --u-
PR2	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	1111 1111	1111 1111	1111 1111
SSPADD	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	--00 0000	--00 0000	--uu uuuu
TXSTA	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000 -010	0000 -010	uuuu -uuu
SPBRG	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67	0000 0000	0000 0000	uuuu uuuu

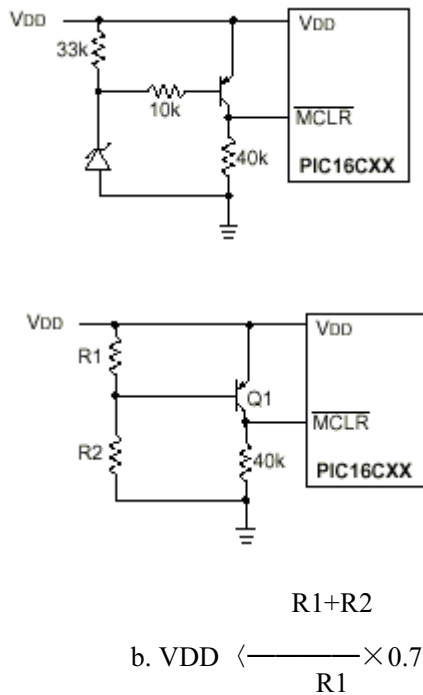
Legend: u = unchanged, x = unknown, - = unimplemented bit read as '0', q = value depends on condition.
Note 1: One or more bits in INTCON, PIR1 and/or PIR2 will be affected (to cause wake-up).
2: When the wake-up is due to an interrupt and the global enable bit, GIE is set, the PC is loaded with the interrupt vector (0004h) after execution of PC + 1.
3: See Table 13-10 and Table 13-11 for reset value for specific conditions.

c. 各种复位后特殊寄存器的值（续）
表 1.37 复位后的特殊寄存器值

八、外部掉电复位锁定

在一些应用中我们可能需要对芯片的供电电压 VDD 随时进行检测，一旦发现 VDD 电压下降（一般是系统掉电引起）到一个门槛值时就使芯片复位以免系统失控，下面是二个典型电路：

a. VDD < Vz+0.7V 时复位



b. VDD < $\frac{R1+R2}{R1} \times 0.7V$ 时复位

图 1.64 掉电复位锁定电路

九、内部掉电复位锁定

涉及型号								
62A	62B	63	63A	64A	65	65A	66	67

上面我们谈到 VDD 掉电复位锁定。在 PIC16C62A/63/64A/65A 等型号中，芯片中已集成有掉电复位锁定电路，请参见 PCON 寄存器和“系统定义字”（Configuration）中有关位的描述。当 VDD 掉到 BVDD（典型值为 4.0V）以下时，复位锁定电路将使芯片保持复位状态，当 VDD 上升到 BVDD 以上时，PWRT 延时器将启动计数 72ms，然后脱离复位状态恢复进入运行状态，见下图：

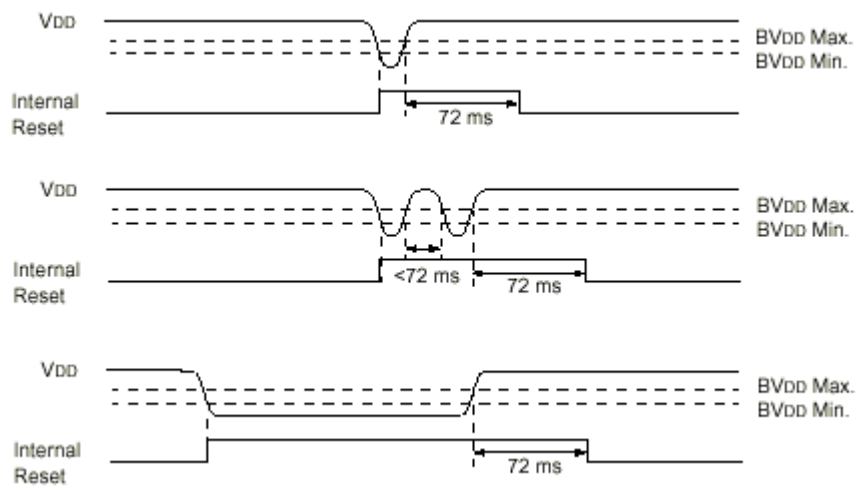


图 1.65 掉电复位锁定时序

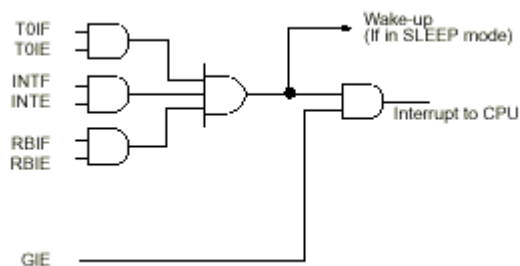
§ 1.13.4 中 断

PIC16C6X 有多种中断源，最多的是 PIC16C65，有 11 种中断，PIC16C61 则有 3 种中断，见下表：

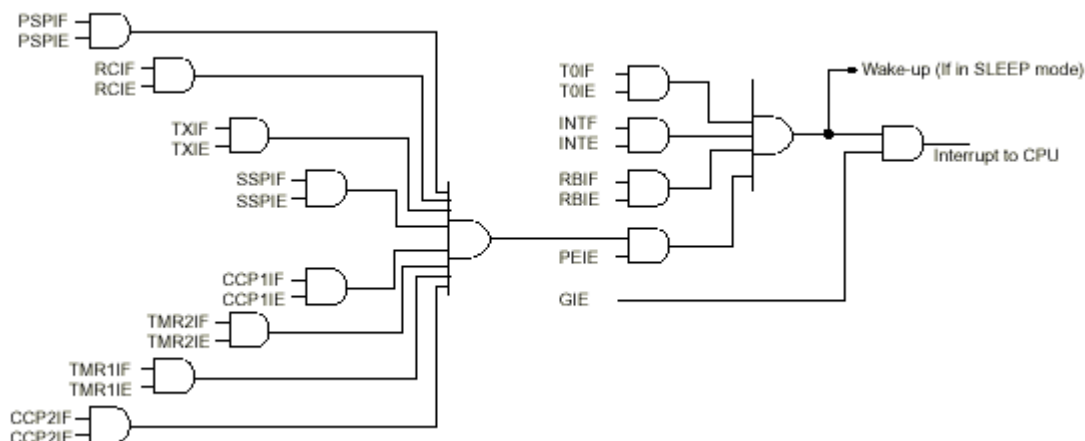
中 断 源	标志位	使能位	65/65A	63	64/64A	62/62A	61
外部触发中断 INT	INTF	INTE	√	√	√	√	√
TMR0 溢出中断	T0IF	T0IE	√	√	√	√	√
PORTB<7:4>中断	RBIF	RBIE	√	√	√	√	√
TMR1 中断	TMR1IF	TMR1IE	√	√	√	√	—
TMR2 中断	TMR2IF	TMR2IE	√	√	√	√	—
CCP1 中断	CCP1IF	CCP1IE	√	√	√	√	—
CCP2 中断	CCP2IF	CCP2IE	√	√	—	—	—
SCI 同步发送中断	TXIF	TXIE	√	√	—	—	—
SCI 同步接收中断	RCIF	RCIE	√	√	—	—	—
SSP 中断	SSPIF	SSPIE	√	√	√	√	—
并行口中断	PSPIF	PSPIE	√	—	√	—	—

表 1.38 PIC16C6X 中断源

在有关中断的寄存器 INTCON、PIE1、PIE2、PIR1、PIR2 中包括了各种中断的使能位和标志位以及所有中断使能位 GIE（INTCON<7>）。芯片复位后硬件自动置 GIE=0，而中断返回指令“RETFIE”执行后将置 GIE=1 以重新开中断。当 CPU 响应中断后，硬件自动清 GIE=0 以关闭所有中断以免发生重复中断，然后把当前的 PC 值（地址）压入堆栈，PC 寄存器置以中断向量地址（0004H）。进入中断服务程序后，程序必须检查中断源，这可以通过检测中断标志位来进行。一旦查到中断源，即用软件把该中断标志位清为 0，因为执行中断返回指令 RETFIE 会重开中断，这时如果有中断标志位为 1，会引起重复中断。中断请求逻辑电路如下：



a. 16C61



b. 16C6X

图 1.66 中断逻辑图

对于外部触发中断，如 INT 和 RB 口中断，中断延迟时间约有 3~4 个指令周期，这取决于中断请求发生的时机，见图 1.60 所示。

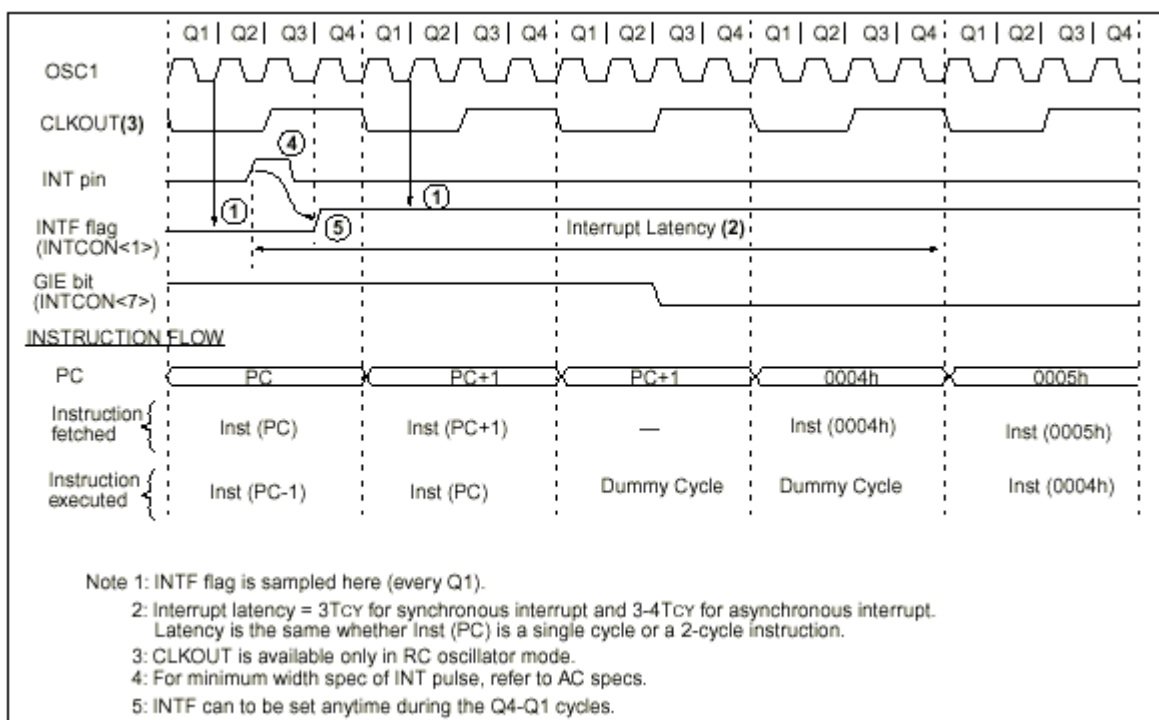


图 1.67 INT 中断时序

注意，不管各种中断使能位或所有中断使能位 GIE 是什么状态（开或关），中断条件满足时都会发中断请求，相应的中断标志位亦会被置成 1，但 CPU 是否响应则根据其使能位状态而定。另一点要注意的是，如果用户程序中要关闭所有中断位，请用以下推荐的程序以确保关闭：

```

LOOP   BCF     INTCON, GIE      ; 置 GIE=0
        BTFSC  INTCON, GIE      ; 成功否
        GOTO   LOOP            ; 否, 继续置 0

```

一、各类中断功能

RB0/INT 脚上的中断信号是边沿触发的, 其极性 (上升或下降) 由 INTEDG (OPTION<6>) 决定。其标志位是 INTF (INTCON<1>), 使能位是 INTE (INTCON<4>)。如果芯片进入睡眠前 INTE=0, 则 INT 中断会唤醒睡眠中的 CPU, 请参见 § 1.13.5 有关 Sleep 的描述。

RB<7:4>中断标志位是 RBIF (INTCON<0>), 使能位是 RBIE (INTCON<4>), 详见 § 1.8.2 描述。

其他各种中断请参阅有关章节, 不再赘述。

二、中断现场保存

中断现场保存是中断技术一个很重要的部分, 由于在 PIC16CXX 指令系统中没有 PUSH (入栈) 和 POP (出栈) 指令, 所以要用一段软件来实现。因为是用一段程序来实现现场保存, 而程序操作是会可能影响 W 寄存器和 STATUS 寄存器, 所以要首先把这二个寄存器保护起来, 然后再保存其他用户认为应保留的寄存器。并且在 PIC16CXX 中, 中断现场数据不是保留到芯片堆栈中, 而是保留在一些用户自己选择的寄存器中, 一般应选择通用寄存器来保留现场。下面是二段例程, 分别是 PIC16C61 和其他 PIC16C6X 的中断现场保护例程。

(1) PIC16C61 中断保护:

```

MOVWF   W_TEMP          ; 保护 W 和 STATUS
SWAPF   STATUS, W
MOVWF   STATUS_TEMP
...
中断服务程序
...
SWAPF   STATUS_TEMP, W   ; 恢复 W 和 STATUS
MOVWF   STATUS
SWAPF   W_TEMP, F
SWAPF   W_TEMP, W

```

(2) 其他 PIC16C6X 中断保护:

```

MOVWF   W_TEMP          ; 保存 W 和 STATUS
SWAPF   STATUS, W
BCF      STATUS, RP0
MOVWF   STATUS_TEMP
...
中断服务程序
...
SWAPF   STATUS_TEMP, W   ; 恢复 W 和 STATUS
MOVWF   STATUS
SWAPF   W_TEMP, F
SWAPF   W_TEMP, W

```

上面这二段例程的区别在于 PIC16C61 寄存器组的 Bank1 是完全映像到 Bank0 的, 即物理上是完全相同的, RP0 位 (STATUS<7>) 未用。而其他的 PIC16C6X 寄存器体由于有 Bank0 和 Bank1 的区别, 所以在其例程中有二点要注意: 一是 W_TEMP 必须同时定义在 Bank0 和 Bank1, 例如 W_TEMP 定义在 0X20, 则 0XA0 也必须分配给它。二是 STATUS_TEMP 必须定义在 Bank0。

以上这二段例程只保存了 W 和 STATUS, 如果用户程序中另有一些寄存器需要保存, 可以自己加上去。

§ 1.13.5 看门狗（WDT）

看门狗计时器（Watch Dog Timer）是一个片内自振式的 RC 振荡计时器，无需任何的外接元件。这意味着即使芯片 OSC1/OSC2 上振荡停止了（例如执行指令 SLEEP 后），WDT 照样保持计时。在正常运行下，WDT 计时溢出将产生复位，而如果芯片是处在睡眠下，WDT 溢出将唤醒 CPU 并沿着原来的路线继续执行。在 PIC16CXX 芯片内的“系统定义字”（Configuration EPROM）中的一个位是用于控制 WDT 的。可以将其置“0”来抑制 WDT 使之永远不起作用。这将在烧写器介绍部分详细说明。

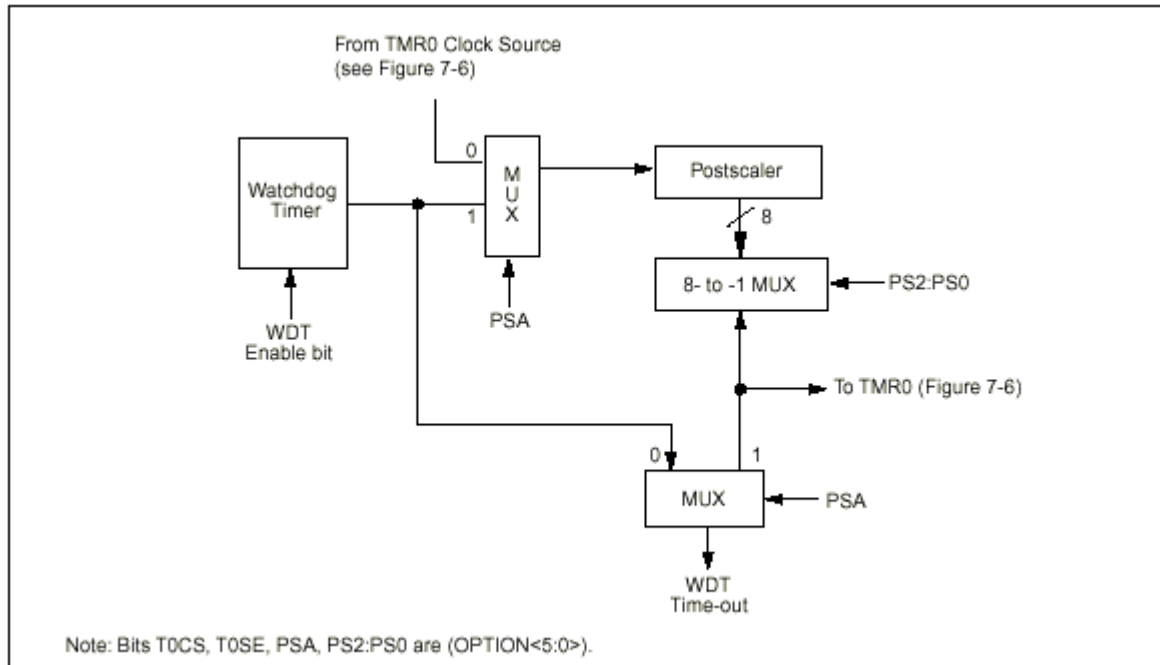


图 1.68 看门狗电路结构图

1.WDT 周期

WDT 有一个基本的溢出周期 18ms（当无预分频倍数时），如果需要更长的 WDT 周期，可以把预分频倍数分配给 WDT，最大分频比可达 1:128，这时的 WDT 溢出周期约为 2.5S。WDT 溢出周期和环境温度、VDD 等参数有关系，请参阅附录的图表。

“CLRWD T”和“SLEEP”指令将清除 WDT 计时器以及预分频器（当预分频器分配给 WDT 时）。WDT 一般用来防止系统失控或者说防止单片机程序“失控”。在正常情况下，WDT 应在计时溢出前被 CLRWD T 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 溢出前执行一条 CLRWD T 指令，就会使 WDT 溢出而产生复位，使系统重新启动运行而不至失去控制。若 WDT 溢出产生复位，则状态寄存器 F3 的“TO”位会被清零，用户可藉此判断复位是否由 WDT 溢时所造成。

2.WDT 编程注意事项

如果使用 WDT，一定要仔细在程序中的某些地方放一条“CLRWD T”指令，以保证在 WDT 溢出前能被清零。否则会造成芯片不停地产生复位，使系统无法正常工作。

在噪声工作环境下，OPTION 寄存器可能会因受干扰而改变，所以最好每隔一段时间就将其重新设置一下。

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
2007h	Config.bits	—	BODEN	CP1	CP0	PWRTE	WDTE	FOSC1	FOSC0
81h	OPTION	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

表 1.39 和看门狗有关的寄存器

§ 1.13.6 睡眠（SLEEP）

一、进入睡眠

执行一条“SLEEP”指令即可进入低功耗模式。当进入 SLEEP 后，WDT 被清零，然后重新开始计数。状态寄存器 STASTUS 中的 PD 位被置成“0”，TO 位置成“1”，同时振荡停止（指 OSC1 端的振荡电路）。所有的 I/O 口保持原来的状态。这种工作模式功耗最低。

为使耗电最小，进入 SLEEP 前，应使所有的 I/O 口处于高电平 VDD 或低电平 VSS，而不应使其处于高阻态，以免产生开关电流损耗。可以在 I/O 口加上拉或下拉电阻，或者把 I/O 口都置成输出态来避免其处于高阻态（浮态）。

RTCC 端亦应置为 VDD 或 VSS（通过上拉或下拉）。

MCLR 必须处于高电平状态。

二、唤醒睡眠

PIC16C6X 可通过以下事件唤醒处于睡眠状态下的 CPU：

1. MCLR 端加低电平；
2. 看门狗超时溢出；
3. 外部中断（INT）；
4. RB 口电平变化中断；
5. TMR1 中断（条件是 TMR1 必须工作在同步计数器方式）；
6. SSP 中断（检测到超始位/停止位）；
7. CCP 捕捉模式中断；
8. 并行口读写操作中断。

细心的读者会发现还有一些中断这里未提到，那是因为那些中断部件在睡眠状态下会关闭不工作，即它们需要单片机在运行状态下才会发生中断请求，所以不能用来唤醒睡眠下的单片机。

注意，PIC16C6X 看门狗超时溢出可唤醒处于睡眠中的芯片，但不会引起芯片复位，而是 PC+1 沿着 Sleep 指令后面的指令继续运行下去。这点和 PIC16C5X 截然不同，在 PIC16C5X 中，WDT 溢出是通过使芯片复位来唤醒原本处于睡眠状态下的 CPU。其他的 PIC16C7X/8X/62X 皆和 PIC16C6X 相同。

如果以上提到的各种中断要用来唤醒睡眠中的单片机，则在进入睡眠之前该中断使能位必须置为 1（开启），至于全部中断使能位 GIE 的状态则不会影响中断唤醒这个功能，但却会影响中断唤醒后单片机的动作走向：

1. GIE=0 — 单片机被中断唤醒后，沿原来的“Sleep”指令后继续执行下去。

2. GIE=1 — 单片机被中断唤醒后，执行完“Sleep”后的下一条指令，然后就跳到中断向量入口处（0004H）去执行中断服务程序。

还有一点，无论何种中断唤醒单片机后，看门狗计数器都会被清零重新计数。注意，只是计数器内容复位清零，而其分频倍率和分配对象并不会改变。

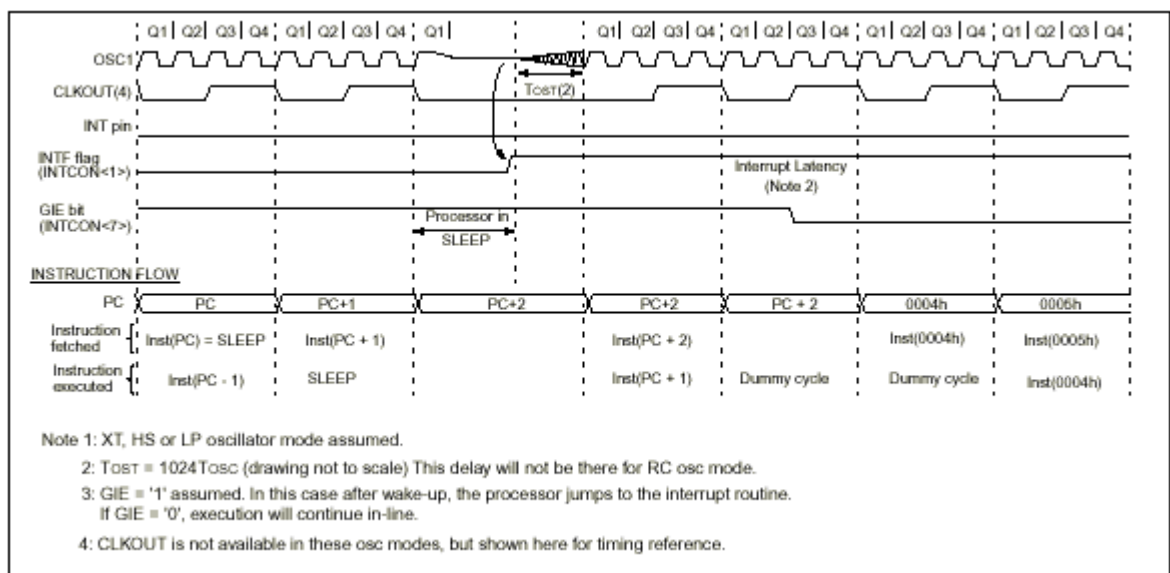


图 1.69 中断唤醒睡眠时序

§ 1.13.7 程序保密位（Protect Fuse）

当选择将芯片的程序保密位熔丝熔断（写入 0）后，程序存储区 ROM 中的程序代码（14 位宽）的高 10 位将被遮没。具体地说，就是当再去读 ROM 的程序代码时，每一个代码都呈现 000XH 的形式。这样高 10 位被用 0 替

代了，低 4 位也是经过加密运算后的值，别人就无法恢复这些被加密的代码，也无法进行代码复制、拷贝了。但单片机的功能不受影响，加密后的程序码并不影响其在单片机内的运行，只是不能被还原读出来。目前尚无办法能够有效地解密 PIC 芯片。

注意:当芯片被选择为保密方式后，程序存储区 40H 以上的空间即不能再被编程，而 0h~003FH 之间的空间还能编程。在程序存储区中，“1”可被烧写成“0”，反之则不可。

§ 1.13.8 用户识别码 (ID Code)

在 PIC16C6X 内部还有一个 16 位的特殊 EPROM (不包括在程序存储区内)，可让用户烧入 4 个十六进制码，以作为芯片标识。这个识别码只起识别作用，对芯片的功能无任何影响。用户可在烧写器上将其烧入和读出验识。

第二章 PIC16C7X 单片机

PIC16C7X 是带 A/D 输入的型号，目前共有下表中所列的几种芯片：

型号	振荡	EPROM	RAM	A/D (8 位)	定时 器	CCP 模块	串行口	并行 口	中断 源	电压范围	I/O	封装	复位 锁定
16C71	DC~ 20M	1K×14	36×8	4	1	—	—	—	4	3.0-6.0V	13	18 脚	无
16C711	DC~ 20M	1K×14	68×8	4	1	—	—	—	4	3.0-6.0V	13	18 脚	有
16C72 16C72A	DC~ 20M	2K×14	128×8	5	3	1	SPI/I2C	—	8	3.0-6.0V	22	28 脚	有
16C73A 16C73B	DC~ 20M	4K×14	192×8	5	3	2	SPI/I2C SCI	—	11	3.0-6.0V	22	28 脚	有
16C74A 16C74B	DC~ 20M	4K×14	192×8	8	3	2	SPI/I2C SCI	有	12	3.0-6.0V	33	40 脚	有
16C76	DC~ 20M	8K×14	368×8	5	3	2	SPI/I2C SCI	—	11	3.0-6.0V	22	28 脚	有
16C77	DC~ 20M	8K×14	368×8	8	3	2	SPI/I2C SCI	有	12	3.0-6.0V	33	40 脚	有

表 2.1 PIC16C7X 型号功能表

从这张表中我们可以看出，实际上 PIC16C7X 就是在 PIC16C6X 的基础上加上 A/D 转换功能，下表是它们之间的对座关系：

PIC16C6X 系列	PIC16C7X 系列
PIC16C61	PIC16C71
PIC16C62	PIC16C72
PIC16C63	PIC16C73
PIC16C65	PIC16C74
PIC16C66	PIC16C76
PIC16C67	PIC16C77
PIC16C64	——

表 2.2 PIC16C6X 和 PIC16C7X 对应表

以上几对 PIC16C6X 和 PIC16C7X 的型号芯片，它们的引脚封装、内部 ROM、I/O 口个数等都是是一样的，不同的只是 16C7X 的型号在某些数字 I/O 脚上还带有可编程的 A/D 输入功能，在内部寄存器（RAM）上则多了几个有关 A/D 转换的特殊功能寄存器，其余完全一样。这个特点是 PIC 系列单片机的一大优点，即用户从某种型号过渡转变到另一种型号往往是非常容易的，不仅硬件可做最小的改动，连软件也仅需很小的变动即可。有时它们之间可以代用，详见附录 I。

有鉴于 PIC 单片机的这种高度兼容特性，下面我们对 PIC16C7X 的介绍将不再重复和 6X 相同的部分，而着重叙述 PIC16C7X 特有的部分，这些特有的部份都是由于带 A/D 功能才具备的。

§ 2.1 主要功能特点

一、高性能 RISC 结构 CPU

- 精简指令集，仅 35 条单字节指令，易学易用
- 除地址分支跳转指令（GOTO、CALL）为双周期指令，其余皆为单周期指令
- 执行速度

时钟振荡	指令周期
40HZ	100 μ S
1MHZ	4 μ S
4MHZ	1 μ S
10MHZ	400ns
20MHZ	200ns

- 八级硬件堆栈
- 多种硬件中断
- 直接、间接、相对三种寻址方式

二、功能部件特性

- 带 8 位 A/D 转换输入

型 号	A/D
16C71/711	4
16C72/72A/73A/73B/76	5
16C74A/74B/77	8

- 高驱动电流，I/O 脚可直接驱动数码管（LED）显示
 - 每个 I/O 引脚最大拉电流 25MA
 - 每个 I/O 引脚最大灌电流 20MA
- 双向可独立编程设置 I/O 引脚
- 8 位定时器/计数器 TMR0，带 8 位预分频
- 有 1~2 路捕抓输入/比较输出/PWM 输出（CCP）
- 16 位定时器/计数器 TMR1，睡眠中仍可计数
- 8 位定时器/计数器 TMR2，带有 8 位的周期寄存器及预分频和后分频
- 并行口操作
- 同步串行口 I2C/SPI 总线操作
- 同步通讯接口 SCI/USART 操作

三、微控制器特性

- 内置上电复位电路（POR）
- 上电定时器，保障工作电压的稳定建立
- 振荡定时器，保障振荡的稳定建立
- 断电复位锁定（16C70/71A/72/73A/74A）
- 内置自振式（RC 振荡）看门狗
- 程序保密位，可防程序代码的非法拷贝
- 四种可选振荡方式
 - 低成本阻容：RC
 - 标准晶体/陶瓷：XT
 - 高速晶体/陶瓷：HS
 - 低频晶体：LP

四、CMOS 工艺特性

- 低功耗
 - <2MA @5V, 4MHZ
 - <15UA @3V, 32KHZ
 - <1UA 低功耗 Sleep 模式下
- 全静态设计
- 宽工作电压：2.0V~6.0V
- 宽工作温度范围：
 - 商用级：0℃~+70℃
 - 工业级：-40℃~+85℃
 - 汽车级：-40℃~+125℃

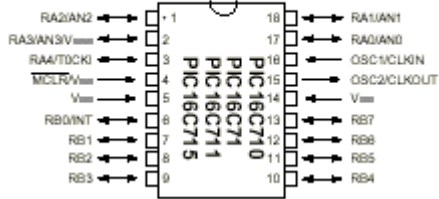
PIC16C7X 是片内带 A/D 的芯片所以它在测量仪器仪表、工业控制、汽车电子、家用电器及通讯等众多方面应用广泛。而它所拥有的高性能，如 CCP 模块、并行口、I²C/SPI、SCI 通讯等等使它能适合于各种应用要求。

§ 2.2 芯片类型

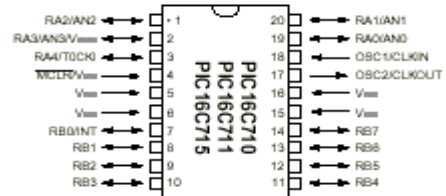
关于 PIC16C7X 的芯片类型，请参阅 § 1.2，和 PIC16C6X 完全一致。

§ 2.3 引脚介绍

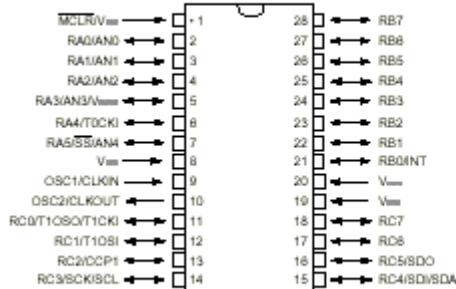
PDIP, SOIC, Windowed Cerdip



SSOP

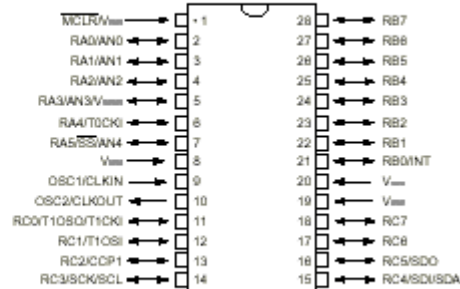


SDIP, SOIC, Windowed Side Brazed Ceramic



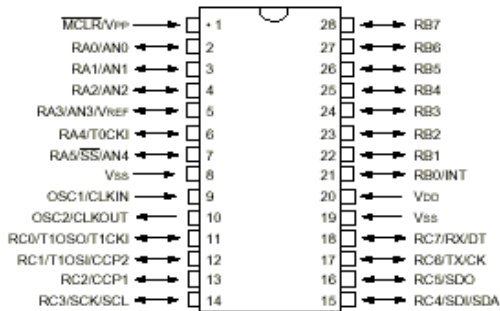
PIC16C72

SSOP



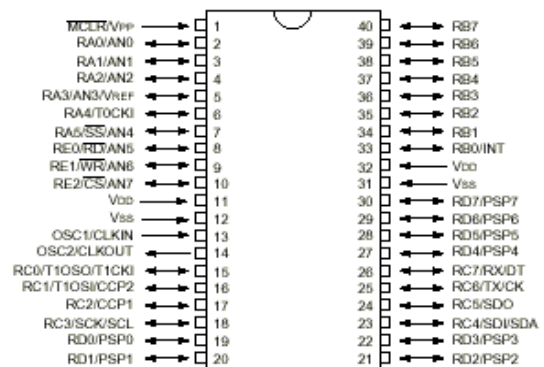
PIC16C72

SDIP, SOIC, Windowed Side Brazed Ceramic



PIC16C73
PIC16C73A
PIC16C76

PDIP, Windowed Cerdip



PIC16C74
PIC16C74A
PIC16C77

图 2.1 PIC16C7X 引脚图

各型号的引脚意义如下表所示：

引脚名	I/O 特性	电平	功能
OSC1/CLKIN	输入	CMOS	振荡输入脚
OSC2/CLKOUT	输出	—	振荡输出脚
MCLR	输入	ST	复位输入脚，低电平有效
RA0/AN0	I/O	TTL	PORTA 数字 I/O 口，双向可编程，并可作 A/D 输入
RA1/AN1	I/O	TTL	A/D 输入通道 0
RA2/AN2	I/O	TTL	A/D 输入通道 1
RA3/AN3/VREF	I/O	TTL	A/D 输入通道 2
RA4/T0CKI	I/O	ST	A/D 输入通道 3 / 参考电压输入 VREF
RB0/INT			可作为 TMR0 外部时钟输入
RB1			PORTB I/O 口，双向可编程
		TTL/ST	亦可作为外部中断信号输入脚（此时为 ST 输入）

RB2		TTL	
RB3		TTL	
RB4		TTL	具有电平变化中断功能
RB5		TTL	具有电平变化中断功能
RB6		TTL	具有电平变化中断功能
RB7		TTL	具有电平变化中断功能
VSS	—	—	地
VDD	—	—	电源

ST：斯密特输入

a. PIC16C71/711

引 脚 名	I/O 特性	电 平	功 能
OSC1/CLKIN	输入	CMOS	振荡输入脚
OSC2/CLKOUT	输出	—	振荡输出脚
MCLR	输入	ST	复位输入脚，低电平有效
RA0/AN0	I/O	TTL	PORTA 数字 I/O 口，双向可编程，并可作 A/D 输入
RA1/AN1	I/O	TTL	A/D 输入通道 0
RA2/AN2	I/O	TTL	A/D 输入通道 1
RA3/AN3/VREF	I/O	TTL	A/D 输入通道 2
RA4/T0CKI	I/O	斯密特输入	A/D 输入通道 3
RA5/AN4/SS	I/O	TTL	可作为 TMR0 外部时钟输入
			亦可做 A/D 输入通道 4，或同步串行口的从属器选择输入
RB0/INT	I/O	TTL/斯密特	PORTB I/O 口，双向可编程，并带可编程弱上拉。
RB1	I/O	TTL	也可作为外部中断信号输入
RB2	I/O	TTL	
RB3	I/O	TTL	
RB4	I/O	TTL	具电平变化中断功能
RB5	I/O	TTL	具电平变化中断功能
RB6	I/O	TTL	具电平变化中断功能
RB7	I/O	TTL	具电平变化中断功能
RC0/T1OSI/T1CKI	I/O		亦可作 TIMER1 振荡输入/TIMER1 时钟输入
RC1/T1OSO	I/O	斯密特输入	亦可作 TIMER1 振荡输出
RC2/CCP1	I/O	斯密特输入	亦可作 CCP1 输入输出
RC3/SCK/SCL	I/O	斯密特输入	亦可作同步串行时钟
RC4/SDI/SDA	I/O	斯密特输入	亦可作 SPI 通讯数据输入线或 I2C 之数据线
RC5/SDO	I/O	斯密特输入	亦可作 SPI 通讯数据输出线
RC6	I/O	斯密特输入	

RC7	I/O	斯密特输入	
VSS	—	—	地
VDD	—	—	电源

b. 16C72/72A

引脚名	I/O 特性	电 平	功 能
OSC1/CLKIN	输入	CMOS	振荡输入脚
OSC2/CLKOUT	输出	—	振荡输出脚
MCLR	输入	斯密特输入	复位输入脚，低电平有效
RA0/AN0 RA1/AN1 RA2/AN2 RA3/AN3/VREF RA4/T0CKI RA5/AN4/SS	I/O I/O I/O I/O I/O I/O	TTL TTL TTL TTL 斯密特输入 TTL	PORTA 数字 I/O 口，双向可编程，并可作 A/D 输入 A/D 输入通道 0 A/D 输入通道 1 A/D 输入通道 2 A/D 输入通道 3 可作为 TMR0 外部时钟输入 亦可做 A/D 输入通道 4，或同步串行口的从属器选择输入
RB0/INT RB1 RB2 RB3 RB4 RB5 RB6 RB7	I/O I/O I/O I/O I/O I/O I/O	TTL/斯密特 TTL TTL TTL TTL TTL TTL	PORTB I/O 口，双向可编程，并带可编程弱上拉 也可作为外部中断信号输入 具电平变化中断功能 具电平变化中断功能 具电平变化中断功能 具电平变化中断功能
RC0/T1OSI/T1CKI RC1/T1OSI /CCP2 RC2/CCP1 RC3/SCK/SCL	I/O I/O I/O I/O	斯密特输入 斯密特输入 斯密特输入 斯密特输入	PORTC I/O 口，双向可编程 亦可作 TIMER1 振荡输入/TIMER1 时钟输入 亦可作为 TIMER1 振荡输入/CCP2 输入输出 亦可作为 CCP1 输入输出 亦可作为同步串行通讯时钟
RC4/SDI/SDA RC5/SDO RC6/TX/CK RC7/RX/DT	I/O I/O I/O I/O	斯密特输入 斯密特输入 斯密特输入 斯密特输入	亦可作为 SPI 通讯之数据输入线或 I2C 数据线 亦可作为 SPI 通讯之数据输出线 亦可作为异步发送线或 SCI 同步传输的时钟线 亦可作为异步接收线或 SCI 同步传输的数据线
RD0/PSP0 RD1/PSP1 RD2/PSP2 RD3/PSP3 RD4/PSP4 RD5/PSP5 RD6/PSP6 RD7/PSP7	I/O I/O I/O I/O I/O I/O I/O	斯密特/TTL 斯密特/TTL 斯密特/TTL 斯密特/TTL 斯密特/TTL 斯密特/TTL 斯密特/TTL	PORTD I/O 口，双向可编程，亦可作为并行口，作为并行口时为 TTL 输入，作为一般 I/O 口时为斯密特输入
RE0/RD/AN5 RE1/WR/AN6 RE2/CS/AN7	I/O I/O I/O	斯密特/TTL 斯密特/TTL 斯密特/TTL	PORTE I/O 口，双向可编程，亦可作为并行口的控制线或 A/D 输入。 RD：并行口读信号线或 A/D 输入通道 5 WR：并行口写信号线或 A/D 输入通道 6 CS：并行口片选线或 A/D 输入通道 7
VSS	—	—	地
VDD	—	—	电源
NC	—	—	未用

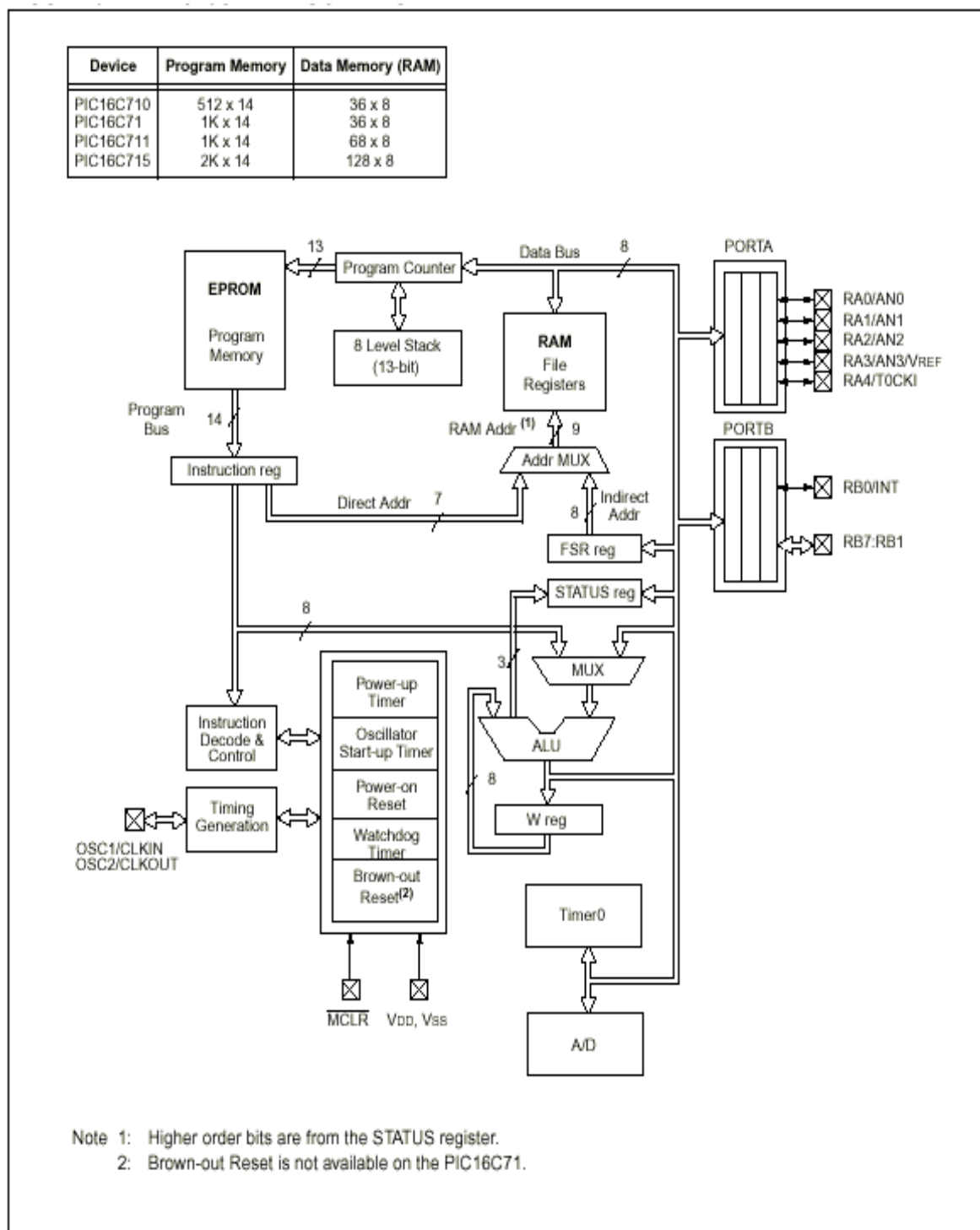
注：16C73A/73B 没有 PORTD 和 PORTE。

c. 16C73A/73B/74A/74B

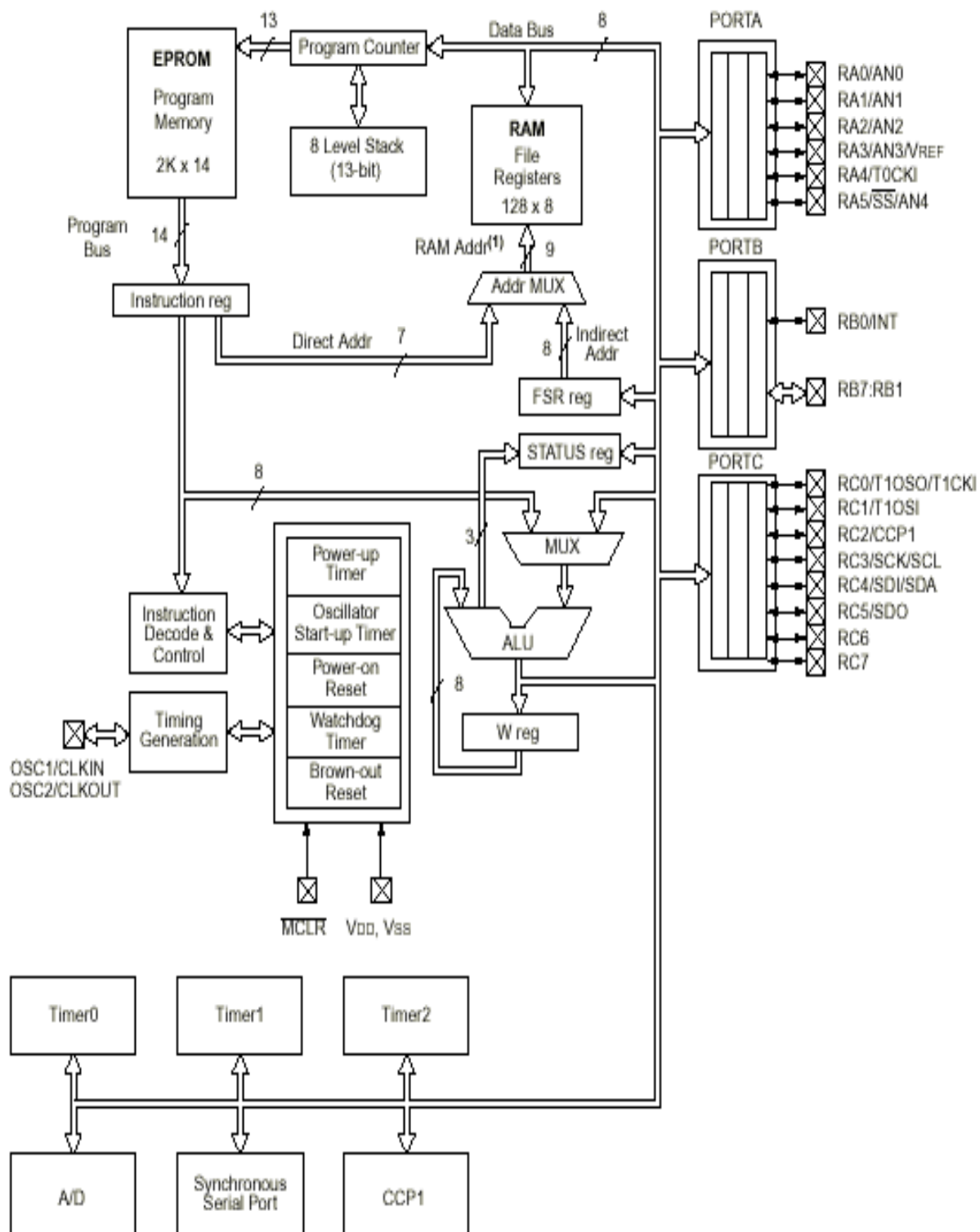
表 2.3 PIC16C7X 引脚功能表

§ 2.4 内部结构

参阅 § 1.4, PIC16C7X 在 PIC16C6X 的基础上加上 A/D 部件, 如下图所示:

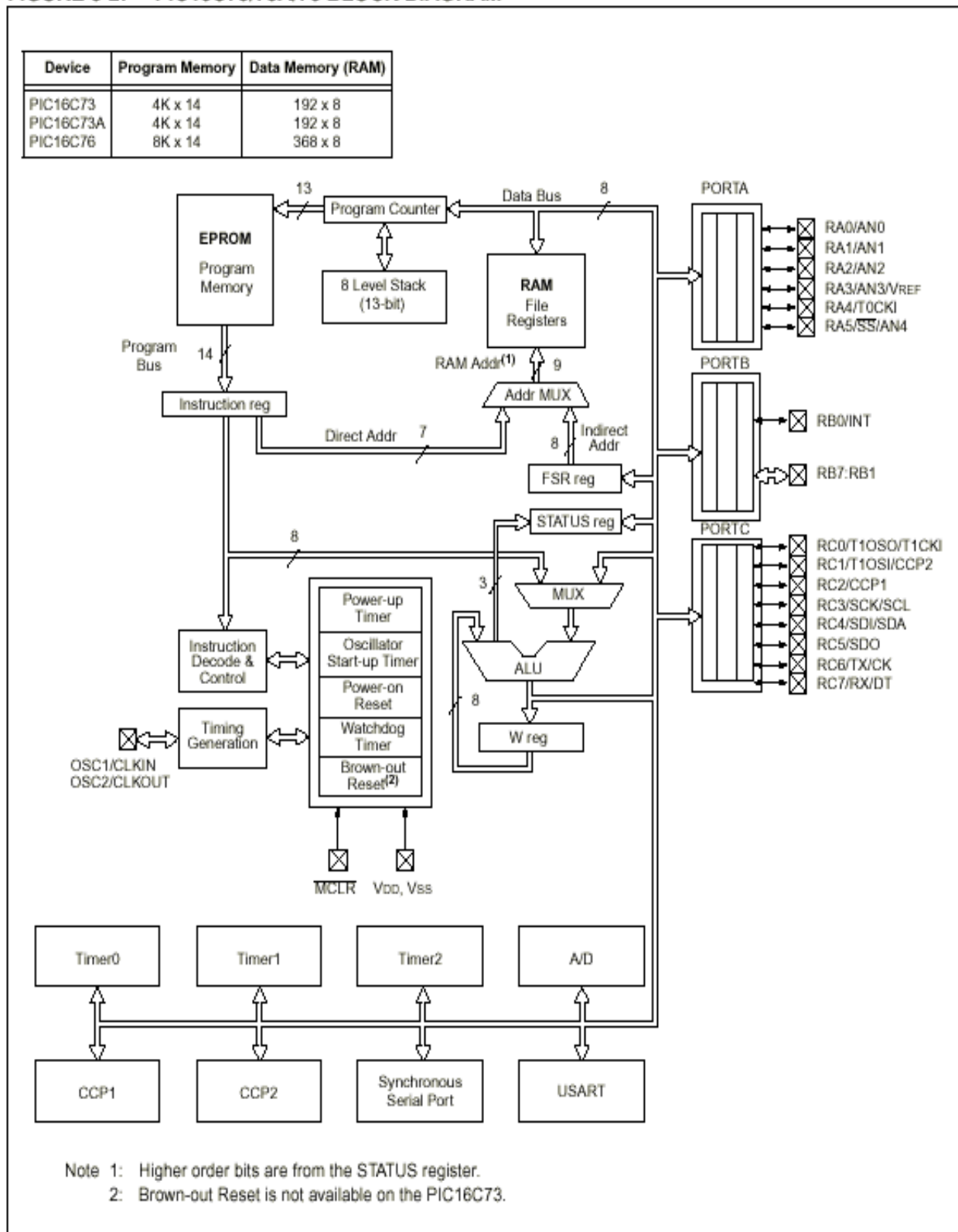


a. PIC16C71/710/715



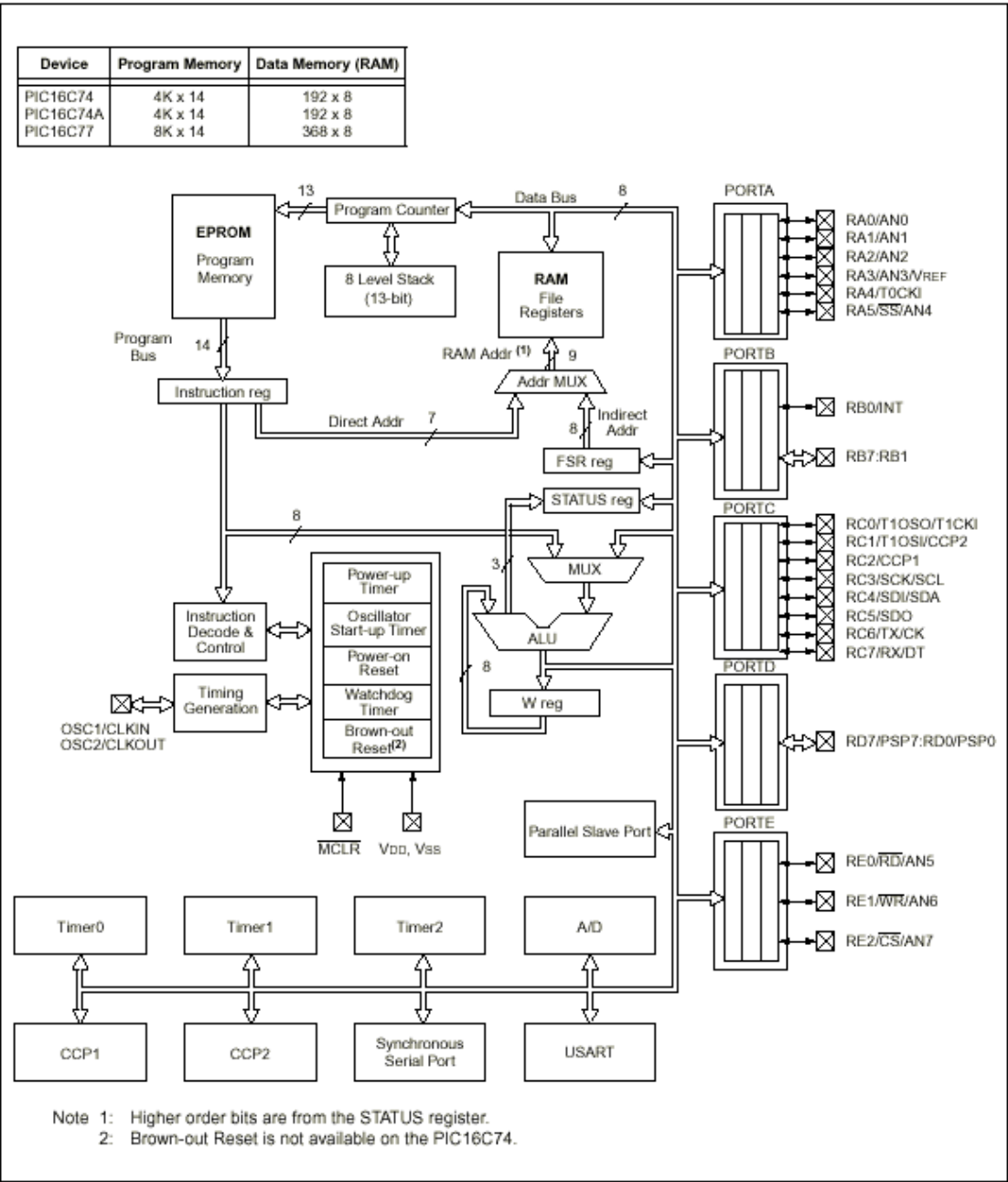
Note 1: Higher order bits are from the STATUS register.

FIGURE 3-2: PIC16C73/73A/76 BLOCK DIAGRAM



c. PIC16C73/73A/76

FIGURE 3-3: PIC16C74/74A/77 BLOCK DIAGRAM



d. PIC16C74/74A/77

图 2.2 PIC16C7X 内部结构

§ 2.5 指令时序


参阅 § 1.5, PIC16C7X 和 PIC16C6X 指令时序完全一样。

§ 2.6 程序存储器和堆栈

PIC16C7X 的程序计数器 (PC) 为 13 位宽, 最大可寻址 8K 字节。对于 PIC16C76/77, 用了 8K, 对于 PIC16C73A/73B/74A/74B, 仅使用头 4K 空间, PIC16C72 使用头 2K, PIC16C71/711 使用头 1K。超出这些空间的寻址将导致在物理空间上的循环回绕。

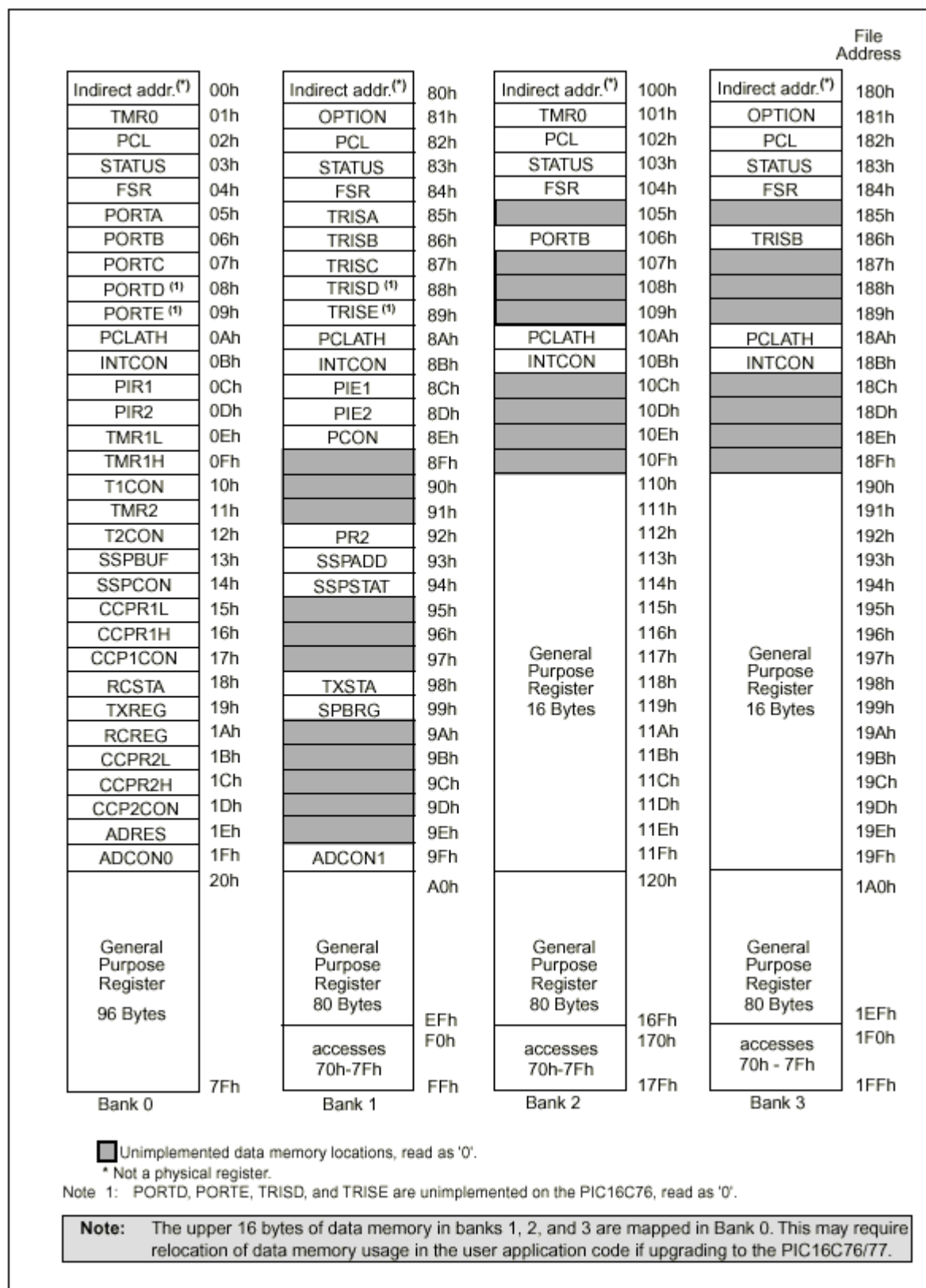
PIC16C7X 的堆栈有 13×8 的独立空间, 不占用程序存储器。

File Address			File Address
00h	INDF(1)	INDF(1)	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	PORTC	TRISC	87h
08h			88h
09h			89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	PIR1	PIE1	8Ch
0Dh			8Dh
0Eh	TMR1L	PCON	8Eh
0Fh	TMR1H		8Fh
10h	T1CON		90h
11h	TMR2		91h
12h	T2CON	PR2	92h
13h	SSPBUF	SSPADD	93h
14h	SSPCON	SSPSTAT	94h
15h	CCPR1L		95h
16h	CCPR1H		96h
17h	CCP1CON		97h
18h			98h
19h			99h
1Ah			9Ah
1Bh			9Bh
1Ch			9Ch
1Dh			9Dh
1Eh	ADRES		9Eh
1Fh	ADCON0	ADCON1	9Fh
20h	General Purpose Register	General Purpose Register	A0h
			BFh
			C0h
7Fh	Bank 0	Bank 1	FFh

 Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

[illegible]



PIC16C76/77

图 2.3 PIC16C7X 寄存器结构

从上图中可以看出，PIC16C7X 比 PIC16C6X 多出的寄存器是 ADRES（A/D 转换结果寄存器），ADCON0 及 ADCON1（A/D 控制寄存器）。

PIC16C7X 各特殊寄存器如下表所示：

地 址	名 称	功 能 说 明	上电复位值	其他复位值
Bank0（0 体）				

00h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000	0000	0000	0000
01h	TMR0	TIMER0 模块寄存器				xxxx	xxxx	uuuu	uuuu
02h	PCL	程序计数器 PC 的低 8 位				0000	0000	0000	0000
03h	STATUS	状态寄存器				0001	1xxx	000q	quuu
04h	FSR	间接寻址寄存器				xxxx	xxxx	uuuu	uuuu
05h	PORTA	—	—	—	PORTA 口寄存器	---x	xxxx	---u	uuuu
06h	PORTB	PORTB 寄存器				xxxx	xxxx	uuuu	uuuu
07h	—	——				—		—	
08h	ADCON0	A/D 控制寄存器 0				00-0	0000	00-0	0000
09h	ADRES	A/D 转换结果寄存器				xxxx	xxxx	uuuu	uuuu
0Ah	PCLATH	—	—	—	PC 高 5 位之写入器	---0	0000	---0	0000
0Bh	INTCON	中断控制寄存器				0000	000x	0000	000u
Bank1（1 体）									
80h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000	0000	0000	0000
81h	OPTION	系统功能定义寄存器				1111	1111	1111	1111
82h	PCL	程序计数器 PC 的低 8 位				0000	0000	0000	0000
83h	STATUS	状态寄存器				0001	1xxx	000q	quuu
84h	FSR	间接寻址寄存器				xxxx	xxxx	uuuu	uuuu
85h	TRISA	—	—	—	PORTA 方向寄存器	---1	1111	---1	1111
86h	TRISB	PORTB 方向寄存器				1111	1111	1111	1111
87h	PCON	上电复位及掉电复位标志位寄存器				----	--qq	----	--uu
88h	ADCON1	A/D 控制寄存器 1				----	--00	----	--00
89h	ADRES	A/D 转换结果寄存器				xxxx	xxxx	uuuu	uuuu
8Ah	PCLATH	—	—	—	PC 高 5 位之写入器	---0	0000	---0	0000
8Bh	INTCON	中断控制寄存器				0000	000X	0000	000u

注：X=不定， u=不变， q=取决于某条件， -=未用（读为 0）

a. 16C71/711 特殊功能寄存器

地 址	名 称	功 能 说 明				上电复位值	其他复位值
Bank0（0 体）							
00h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000 0000	0000 0000
01h	TMR0	TIMER0 模块寄存器				xxxx xxxx	uuuu uuuu
02h	PCL	程序计数器 PC 的低 8 位				0000 0000	0000 0000
03h	STATUS	状态寄存器				0001 1xxx	000q quuu
04h	FSR	间接寻址寄存器				xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	PORTA 口寄存器		--xx xxxx	--uu uuuu
06h	PORTB	PORTB 口寄存器				xxxx xxxx	uuuu uuuu
07h	PORTC	PORTC 口寄存器				xxxx xxxx	uuuu uuuu
08h	—	—				—	—
09h	—	—				—	—
0Ah	PCLATH	—	—	—	PC 高 5 位写入器	---0 0000	---0 0000
0Bh	INTCON	中断控制寄存器				0000 000x	0000 000u
0Ch	PIR1	某些中断标志位寄存器				-0-- 0000	-0-- 0000
0Dh	—	—				—	—
0Eh	TMR1L	TIMER1 模块寄存器低 8 位				xxxx xxxx	uuuu uuuu
0Fh	TMR1H	TIMER1 模块寄存器高 8 位				xxxx xxxx	uuuu uuuu
10h	T1CON	TIMER1 控制寄存器				--00 0000	--uu uuuu
11h	TMR2	TIMER2 模块寄存器				xxxx xxxx	uuuu uuuu
12h	T2CON	TIMER2 控制寄存器				-000 0000	-000 0000
13h	SSPBuF	同步串行口（SSP）发送/接收寄存器				xxxx xxxx	uuuu uuuu
14h	SSPCON	同步串行口（SSP）控制寄存器				0000 0000	0000 0000
15h	CCPR1L	CCP1 模块寄存器（低 8 位）				xxxx xxxx	uuuu uuuu
16h	CCPR1H	CCP1 模块寄存器（高 8 位）				xxxx xxxx	uuuu uuuu
17h	CCP1CON	CCP1 控制寄存器				--00 0000	--00 0000

18h~1Dh	—	—				—	—		
1Eh	ADRES	A/D 转换结果寄存器				xxxx	xxxx	uuuu	uuuu
1Fh	ADCON0	A/D 控制寄存器 0				0000	00-0	0000	00-0
Bank1（1 体）									
80h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000	0000	0000	0000
81h	OPTION	TIMER0 模块寄存器				xxxx	xxxx	uuuu	uuuu
82h	PCL	程序计数器 PC 的低 8 位				0000	0000	0000	0000
83h	STATUS	状态寄存器				0001	1xxx	000q	quuu
84h	FSR	间接寻址寄存器				xxxx	xxxx	uuuu	uuuu
85h	TRISA	PORTA 口方向寄存器				--11	1111	--11	1111
86h	TRISB	PORTB 口方向寄存器				1111	1111	1111	1111
87h	TRISC	PORTC 口方向寄存器				1111	1111	1111	1111
88h	—	—				—		—	
89h	—	—				—		—	
8Ah	PCLATH	—	—	—	PC 高 5 位写入器	---0	0000	---0	0000
8Bh	INTCON	中断控制寄存器				0000	000x	0000	000u
8Ch	PIE1	某些中断允许位寄存器				-0--	0000	-0--	0000
8Dh	—	—				—		—	
8Eh	PCON	上电复位（POR）和掉电复位（BOR）标志位				----	--qq	----	--uu
8Fh~91h	—	—				—		—	
92h	PR2	TIMER2 周期寄存器				1111	1111	1111	1111
93h	SSPAD	同步串行口（SSP）I ² C 模式下之地址寄存器				0000	0000	0000	0000
94h	SSPSTAT	同步串行口（SSP）之状态寄存器				0000	0000	0000	0000
95h~9Eh	—	—				—		—	
9Fh	ADCON1	A/D 控制寄存器 1				----	-000	----	-000

注: X=不定, u=不变, q=取决于某条件, -=未用(读为0)

c. 16C72/72A 特殊功能寄存器

地 址	名 称	功 能 说 明				上电复位值	其他复位值
Bank0（0 体）							
00h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000 0000	0000 0000
01h	TMR0	TIMER0 模块寄存器				xxxx xxxx	uuuu uuuu
02h	PCL	程序计数器 PC 的低 8 位				0000 0000	0000 0000
03h	STATUS	状态寄存器				0001 1xxx	000q quuu
04h	FSR	间接寻址寄存器				xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	PORTA 口寄存器		--xx xxxx	--uu uuuu
06h	PORTB	PORTB 口寄存器				xxxx xxxx	uuuu uuuu
07h	PORTC	PORTC 口寄存器				xxxx xxxx	uuuu uuuu
08h	—	—				—	—
09h	—	—				—	—
0Ah	PCLATH	—	—	—	PC 高 5 位写入器	---0 0000	---0 0000
0Bh	INTCON	中断控制寄存器				0000 000x	0000 000u
0Ch	PIR1	某些中断标志位寄存器				00-- 0000	00-- 0000
0Dh	PIR2	CCP2 中断标志位				---- ---0	---- ---0
0Eh	TMR1L	TIMER1 模块寄存器低 8 位				xxxx xxxx	uuuu uuuu
0Fh	TMR1H	TIMER1 模块寄存器高 8 位				xxxx xxxx	uuuu uuuu
10h	T1CON	TIMER1 控制寄存器				--00 0000	--uu uuuu
11h	TMR2	TIMER2 模块寄存器				xxxx xxxx	uuuu uuuu
12h	T2CON	TIMER2 控制寄存器				-000 0000	-000 0000
13h	SSPBuF	同步串行口（SSP）发送/接收寄存器				xxxx xxxx	uuuu uuuu
14h	SSPCON	同步串行口（SSP）控制寄存器				0000 0000	0000 0000
15h	CCPR1L	CCP1 模块寄存器（低 8 位）				xxxx xxxx	uuuu uuuu
16h	CCPR1H	CCP1 模块寄存器（高 8 位）				xxxx xxxx	uuuu uuuu
17h	CCP1CON	CCP1 控制寄存器				--00 0000	--00 0000

18h	RCSTA	SCI 接收状态和控制寄存器				0000	-00x	0000	-00x
19h	TXREG	USART 发送寄存器				0000	0000	0000	0000
1Ah	RCREG	USART 接收寄存器				0000	0000	0000	0000
1Bh	CCPR2L	CCP2 模块寄存器（低 8 位）				xxxx	xxxx	uuuu	uuuu
1Ch	CCPR2H	CCP2 模块寄存器（高 8 位）				xxxx	xxxx	uuuu	uuuu
1Dh	CCP2CON	CCP2 控制寄存器				--00	0000	--00	0000
1Eh	ADRES	A/D 转换结果寄存器				xxxx	xxxx	uuuu	uuuu
1Fh	ADCON0	A/D 控制寄存器 0				0000	00-0	0000	00-0
Bank1（1 体）									
80h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000	0000	0000	0000
81h	OPTION	TIMER0 模块寄存器				xxxx	xxxx	uuuu	uuuu
82h	PCL	程序计数器 PC 的低 8 位				0000	0000	0000	0000
83h	STATUS	状态寄存器				0001	1xxx	000q	quuu
84h	FSR	间接寻址寄存器				xxxx	xxxx	uuuu	uuuu
85h	TRISA	PORTA 口方向寄存器				--11	1111	--11	1111
86h	TRISB	PORTB 口方向寄存器				1111	1111	1111	1111
87h	TRISC	PORTC 口方向寄存器				1111	1111	1111	1111
88h	—	—				—		—	
89h	—	—				—		—	
8Ah	PCLATH	—	—	—	PC 高 5 位写入器	---0	0000	---0	0000
8Bh	INTCON	中断控制寄存器				0000	000x	0000	000u
8Ch	PIE1	某些中断允许位寄存器				0000	0000	0000	0000
8Dh	PIE2	CCP2 中断使能位寄存器				----	--0	----	--0
8Eh	PCON	上电复位（POR）和掉电复位（BOR）标志位				----	--qq	----	--uu
8Fh~91h	—	—				—		—	
92h	PR2	TIMER2 周期寄存器				1111	1111	1111	1111
93h	SSPADD	同步串行口（SSP）I ² C 模式下之地址寄存器				0000	0000	0000	0000
94h	SSPSTAT	同步串行口（SSP）之状态寄存器				0000	0000	0000	0000
95h~97h	—	—				—		—	
98h	TXSTA	SCI 发送状态及控制寄存器				0000	-010	0000	-010
99h	SPBRG	波特率发生器寄存器				0000	0000	0000	0000
9Ah~9Eh	—	—				—		—	
9Fh	ADCON1	A/D 控制寄存器 1				----	-000	----	-000

注：X=不定，u=不变，q=取决于某条件，-=未用（读为0）

c. 16C73A/73B特殊功能寄存器

地 址	名 称	功 能 说 明				上电复位值	其他复位值
Bank0（0 体）							
00h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000 0000	0000 0000
01h	TMR0	TIMER0 模块寄存器				xxxx xxxx	uuuu uuuu
02h	PCL	程序计数器 PC 的低 8 位				0000 0000	0000 0000
03h	STATUS	状态寄存器				0001 1xxx	000q quuu
04h	FSR	间接寻址寄存器				xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	PORTA 口寄存器		--xx xxxx	--uu uuuu
06h	PORTB	PORTB 口寄存器				xxxx xxxx	uuuu uuuu
07h	PORTC	PORTC 口寄存器				xxxx xxxx	uuuu uuuu
08h	PORTD	PORTD 口寄存器				xxxx xxxx	uuuu uuuu
09h	PORTE	PORTE 口寄存器				---- -xxx	---- -uuu
0Ah	PCLATH	—	—	—	PC 高 5 位写入器	---0 0000	---0 0000
0Bh	INTCON	中断控制寄存器				0000 000x	0000 000u
0Ch	PIR1	某些中断标志位寄存器				00-- 0000	00-- 0000
0Dh	PIR2	CCP2 中断标志位寄存器				---- --0	---- --0
0Eh	TMR1L	TIMER1 模块寄存器低 8 位				xxxx xxxx	uuuu uuuu
0Fh	TMR1H	TIMER1 模块寄存器高 8 位				xxxx xxxx	uuuu uuuu

10h	T1CON	TIMER1 控制寄存器				--00	0000	--uu	uuuu
11h	TMR2	TIMER2 模块寄存器				xxxx	xxxx	uuuu	uuuu
12h	T2CON	TIMER2 控制寄存器				-000	0000	-000	0000
13h	SSPBuF	同步串行口（SSP）发送/接收寄存器				xxxx	xxxx	uuuu	uuuu
14h	SSPCON	同步串行口（SSP）控制寄存器				0000	0000	0000	0000
15h	CCPR1L	CCP1 模块寄存器（低 8 位）				xxxx	xxxx	uuuu	uuuu
16h	CCPR1H	CCP1 模块寄存器（高 8 位）				xxxx	xxxx	uuuu	uuuu
17h	CCP1CON	CCP1 控制寄存器				--00	0000	--00	0000
18h	RCSTA	SCI 接收状态和控制寄存器				0000	-00x	0000	-00x
19h	TXREG	USART 发送寄存器				0000	0000	0000	0000
1Ah	RCREG	USART 接收寄存器				0000	0000	0000	0000
1Bh	CCPR2L	CCP2 模块寄存器（低 8 位）				xxxx	xxxx	uuuu	uuuu
1Ch	CCPR2H	CCP2 模块寄存器（高 8 位）				xxxx	xxxx	uuuu	uuuu
1Dh	CCP2CON	CCP2 控制寄存器				--00	0000	--00	0000
1Eh	ADRES	A/D 转换结果寄存器				xxxx	xxxx	uuuu	uuuu
1Fh	ADCON0	A/D 控制寄存器 0				0000	00-0	0000	00-0
Bank1（1 体）									
80h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000	0000	0000	0000
81h	OPTION	TIMER0 模块寄存器				xxxx	xxxx	uuuu	uuuu
82h	PCL	程序计数器 PC 的低 8 位				0000	0000	0000	0000
83h	STATUS	状态寄存器				0001	1xxx	000q	quuu
84h	FSR	间接寻址寄存器				xxxx	xxxx	uuuu	uuuu
85h	TRISA	PORTA 口方向寄存器				--11	1111	--11	1111
86h	TRISB	PORTB 口方向寄存器				1111	1111	1111	1111
87h	TRISC	PORTC 口方向寄存器				1111	1111	1111	1111
88h	TRISD	PORTD 口方向寄存器				1111	1111	1111	1111
89h	TRISE	PORTE 方向寄存器及 SPI 状态寄存器				0000	-111	0000	-111
8Ah	PCLATH	—	—	—	PC 高 5 位写入器	---0	0000	---0	0000
8Bh	INTCON	中断控制寄存器				0000	000x	0000	000u
8Ch	PIE1	某些中断允许位寄存器				0000	0000	0000	0000
8Dh	PIE2	CCP2 中断使能位寄存器				----	---0	----	---0
8Eh	PCON	上电复位（POR）和掉电复位（BOR）标志位				----	--qq	----	--uu
8Fh~91h	—	——				—		—	
92h	PR2	TIMER2 周期寄存器				1111	1111	1111	1111
93h	SSPADDD	同步串行口（SSP）I ² C 模式下之地址寄存器				0000	0000	0000	0000
94h	SSPSTAT	同步串行口（SSP）之状态寄存器				0000	0000	0000	0000
95h~97h	—	——				—		—	
98h	TXSTA	SCI 发送状态及控制寄存器				0000	-010	0000	-010
99h	SPBRG	波特率发生器寄存器				0000	0000	0000	0000
9Ah~9Eh	—	——				—		—	
9Fh	ADCON1	A/D 控制寄存器 1				----	-0000	----	-000

注： X=不定， u=不变， q=取决于某条件， -=未用（读为0）

c. 16C74A/74B特殊功能寄存器

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets (2)		
Bank 0													
00h ⁽⁴⁾	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	0000 0000		
01h	TMR0	Timer0 module's register								xxxx xxxx	uuuu uuuu		
02h ⁽⁴⁾	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	0000 0000		
03h ⁽⁴⁾	STATUS	IRP	RP1	RP0	\overline{TO}	\overline{PD}	Z	DC	C	0001 1xxx	000q quuu		
04h ⁽⁴⁾	FSR	Indirect data memory address pointer								xxxx xxxx	uuuu uuuu		
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read								--0x 0000	--0u 0000
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	uuuu uuuu		
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	uuuu uuuu		
08h ⁽⁵⁾	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	uuuu uuuu		
09h ⁽⁵⁾	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu		
0Ah ^(1,4)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	---0 0000		
0Bh ⁽⁴⁾	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u		
0Ch	PIR1	PSPIF ⁽³⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000		
0Dh	PIR2	—	—	—	—	—	—	—	CCP2IF	---- -000	---- -000		
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu		
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu		
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	TTSYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu		
11h	TMR2	Timer2 module's register								0000 0000	0000 0000		
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000		
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu		
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000		
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	uuuu uuuu		
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	uuuu uuuu		
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000		
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x		
19h	TXREG	USART Transmit Data Register								0000 0000	0000 0000		
1Ah	RCREG	USART Receive Data Register								0000 0000	0000 0000		
1Bh	CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	uuuu uuuu		
1Ch	CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	uuuu uuuu		
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000		
1Eh	ADRES	A/D Result Register								xxxx xxxx	uuuu uuuu		
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON	0000 00-0	0000 00-0		

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented read as '0'.

Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Other (non power-up) resets include external reset through MCLR and Watchdog Timer Reset.

3: Bits PSPIE and PSPIF are reserved on the PIC16C76, always maintain these bits clear.

4: These registers can be addressed from any bank.

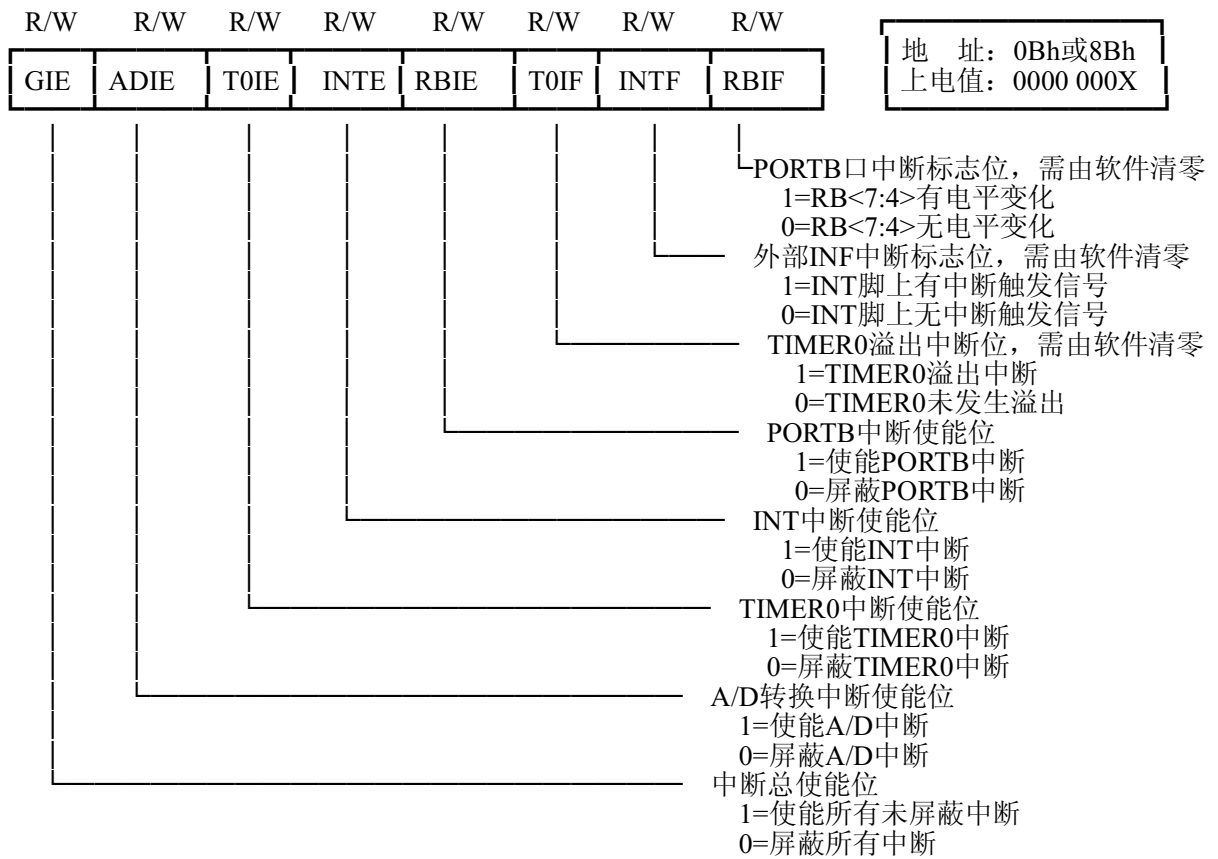
5: PORTD and PORTE are not physically implemented on the PIC16C76, read as '0'.

c. 16C74A/74B特殊功能寄存器表

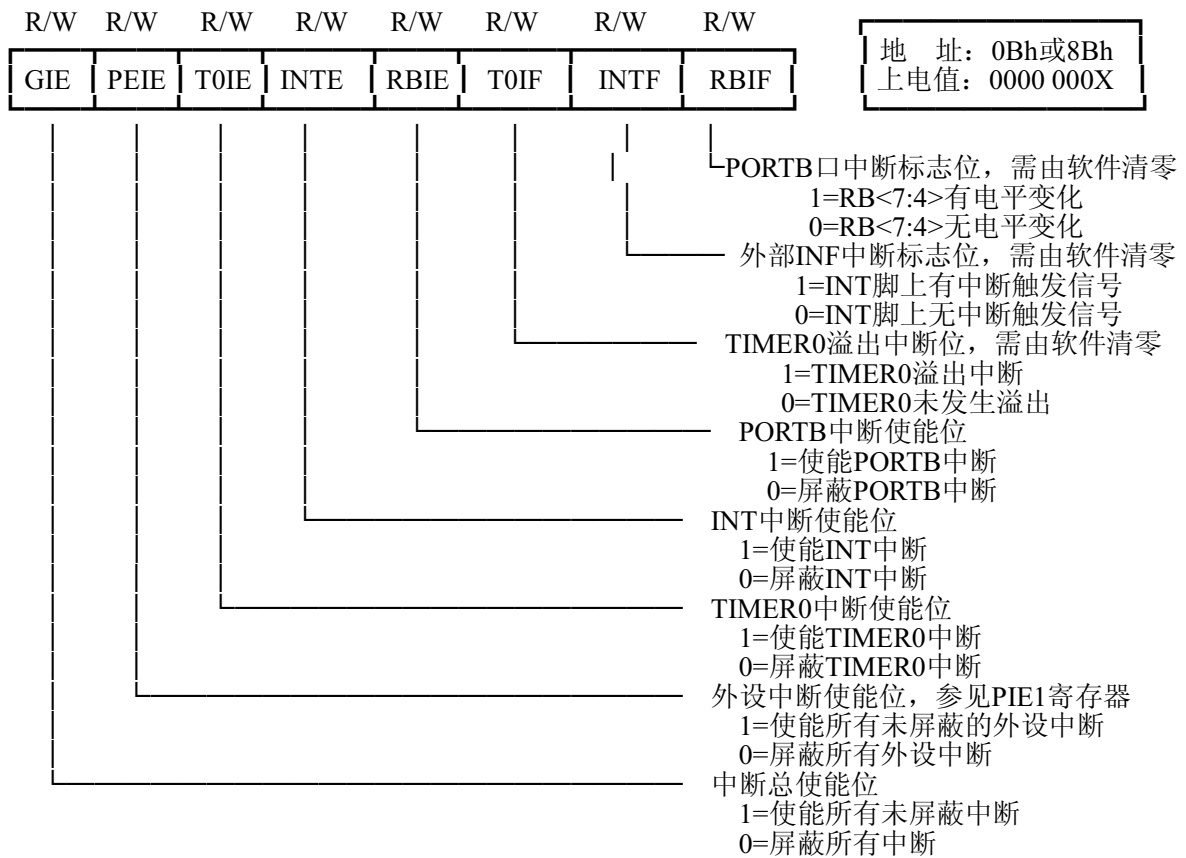
2.4 16C7X特殊功能寄存器

二、中断控制寄存器INTCON

PIC16C72/72A/73A/73B/74A/74B/76/77的INTCON寄存器和PIC16C67/66/65/64/63/62等的INTCON完全一样，而PIC16C71/711的INTCON和PIC16C61的INTCON相比多了一位A/D中断使能位，见下图：



a. 16C71/711



b. 16C72/72A/73A/73B/74A/74B

图2.4 PIC16C7X的INTCON寄存器

请参阅 § 1.7.2中对此有关内容。

三、寄存器PIE1

涉及型号							
72	72A	73A	73B	74A	74B	76	77

PIC16C7X中的PIE1寄存器比PIC16C6X的多了一位A/D中断使能位，见下图：

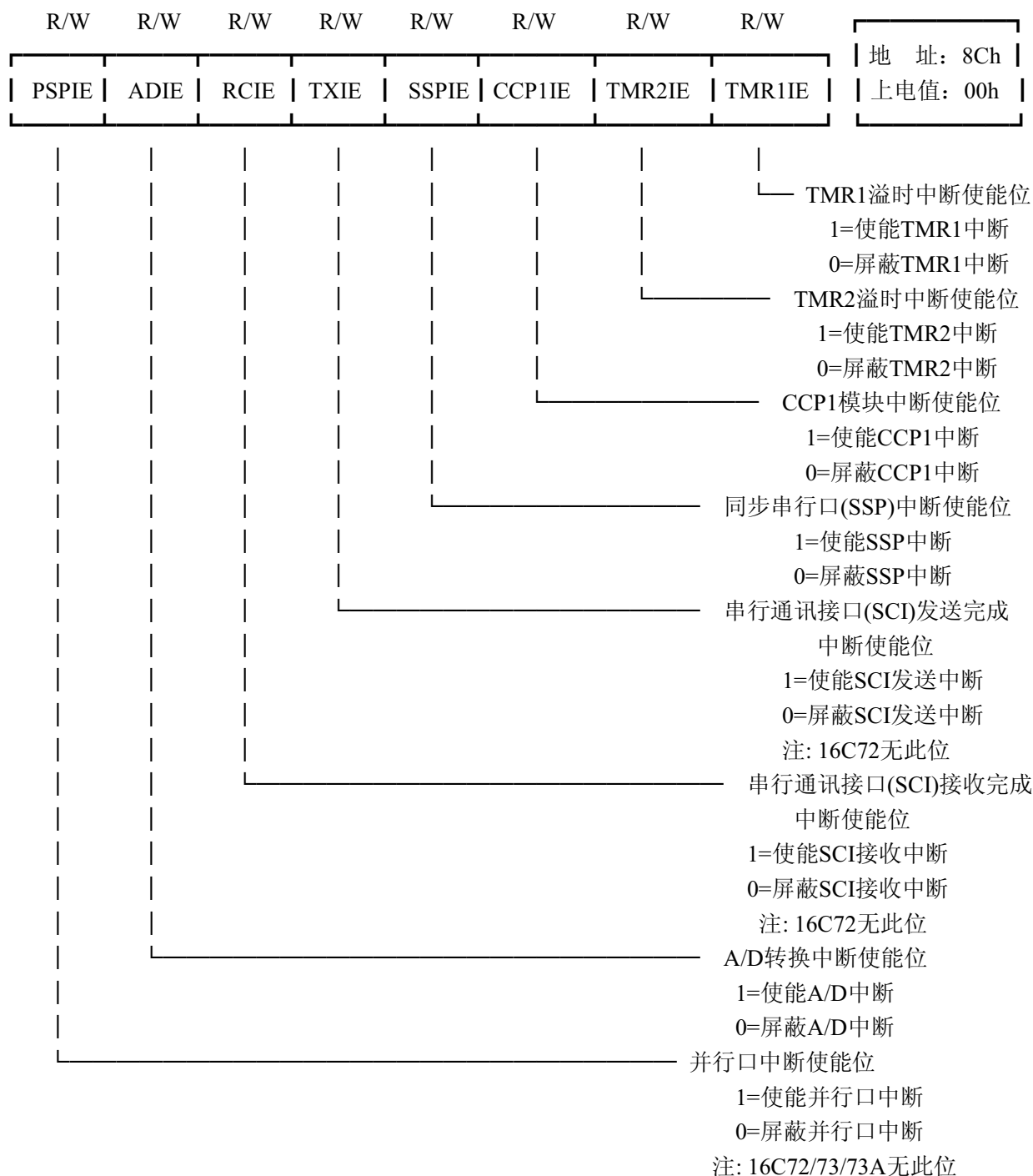


图2.5 PIE1寄存器

四、寄存器PIR1

涉及型号							
72	72A	73A	73B	74A	74B	76	77

PIC16C7X中的PIR1寄存器含有A/D转换完成中断标志位，见下图：

成的复位)。

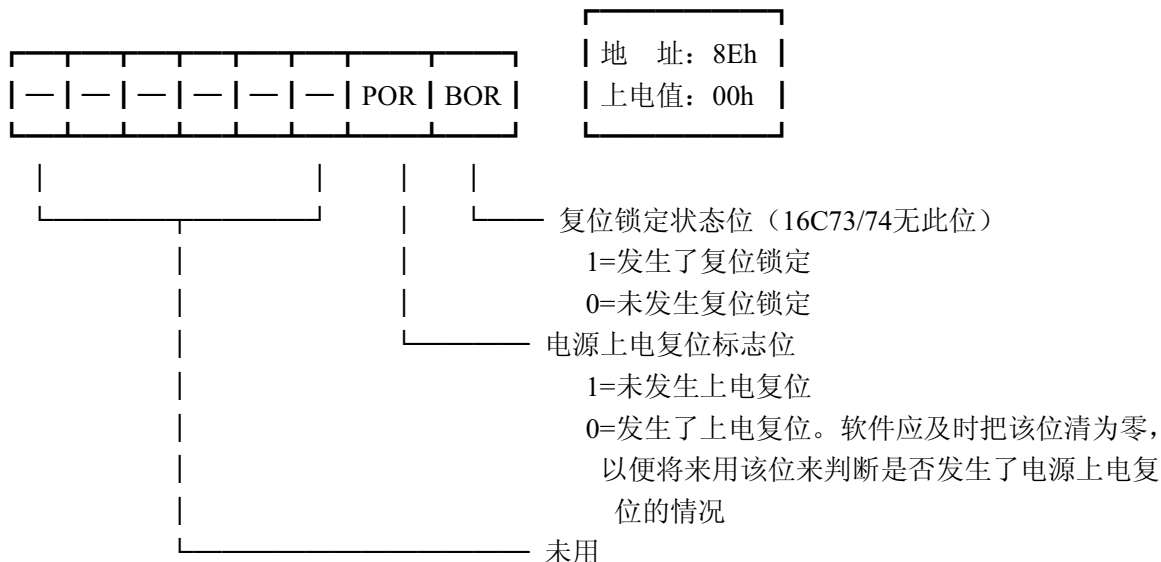


图2.7 PCON寄存器

§ 2.8 I/O □

在I/O口方面，PIC16C7X在PORTA口和PORTE口上增加了A/D功能，其余如PORTB、PORTC和PORTD则和PIC16C6X完全一样，所以我们下面即只对PORTA和PORTE有关A/D的部分做介绍，其余PORTB、PORTC及PORTD请读者参阅§1.8有关内容即可。

一、PORTA和TRISA寄存器

对于PIC16C71/711, RA<3:0>还可以编程作为A/D输入口线, 对于PIC16C72/72A/73A/73B/74A/74B/76/77, RA<3:0>以及RA<5>皆可编程作为A/D输入, 见下图:

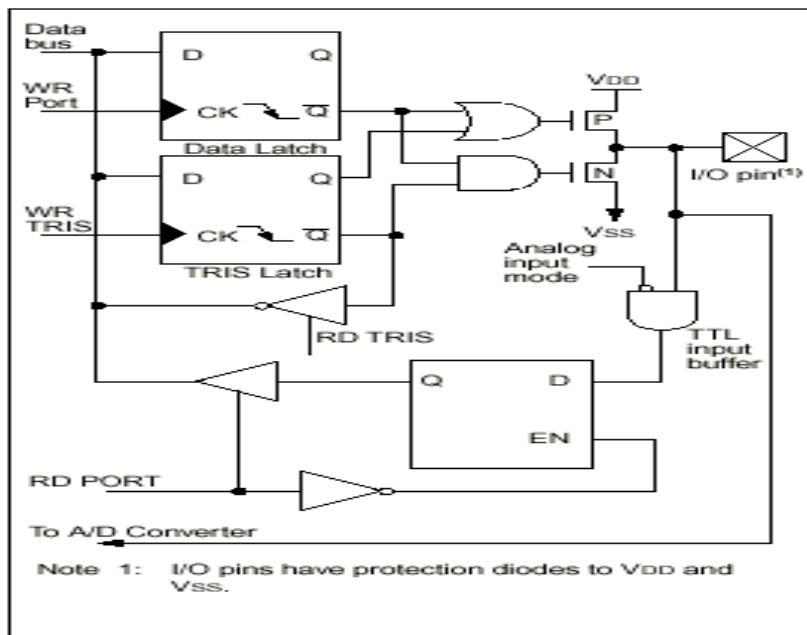


图2.8 RA<3:0>及RA<5>结构图

另外RA<3>还可编程作为模拟参考电压输入。

当RA口线编程作为A/D输入，需将这些口线置为输入态，则应置相应的TRISA位为“1”。当一口线作为A/D输入时，读该口线总是读入“0”。

例如要把RA<3:0>作为A/D输入，而RA<5:4>作为一般的数字I/O输出，则程序可编如下：

```
CLRF      PORTA           ; 清PORTA口
BSF       STATUS, RP0     ; 选Bank1，准备置数TRISA
MOVLW     0XCF             ; 11001111
MOVWF     TRISA           ; 置RA<3:0>为输入，RA<5:4>为输出
```

可通过设置ADCON1寄存器来选择这些口线是作为A/D输入或是作为一般的数字I/O口线使用，详见 § 2.13介绍。
其余有关PORTA的数字功能详请参阅 § 1.8.1。

二、PORTE和TRISE

涉及型号		
74A	74B	77

只有PIC16C74A/74B/77才有PORTE和TRISE。通过设置ADCON1寄存器的值，可以选择RE<3:0>作为一般数字I/O口使用或作为A/D输入，请参阅 § 2.13介绍。
有关PORTE和TRISE的其他功能介绍请参阅 § 1.8.5。
注意: 芯片上电后PORTA和PORTE中这些具A/D功能的口线是自动置为A/D输入的，如欲将它们作为一般数字I/O口线使用，则需先用软件把其设置为数字态：

A. PIC16C71/711

```
MOVLW     3               ; 00000011
MOVWF     ADCON1          ; RA<3:0>置为数字口
```

B. PIC16C72/72A/73A/73B/74A/74B

```
MOVLW     7               ; 00000111
MOVWF     ADCON1          ; RA<3:5>，RE<2:0>置为数字口
```

其他有关I/O的功能部件如并行口以及I/O口编程注意事项等PIC16C7X和PIC16C6X完全一样，请参阅 § 1.8.6和 § 1.8.7等章节介绍。

§ 2.9 定时器/计数器

关于这个部分，读者只要记住PIC16C72/72A/73A/73B/74A/74B/76/77和PIC16C67/66/65/63/62等一样，有TIMER0、TIMER1和TIMER2。而PIC16C71/711和PIC16C61一样只有TIMER0即可，读者可参阅 § 1.9。

§ 2.10 看门狗WDT

PIC16C7X的看门狗WDT和PIC16C6X的看门狗完全相同，请参阅 § 1.13.5。

§ 2.11 CCP模块

涉及型号									
71	711	72	72A	73A	73B	74A	74B	76	77

PIC16C73A/73B/74A/74B/76/77有2个CCP模块，PIC16C72/72A有一个CCP模块，而PIC16C71/711没有CCP模块。PIC16C7X的CCP模块功能和PIC16CXX之CCP功能基本上相同，只有一点，那就是当CCP设置成“比较输出触发事件”（CCPxM3:CCPxM0=1011）时，可以用来触发启动A/D转换，见下图：

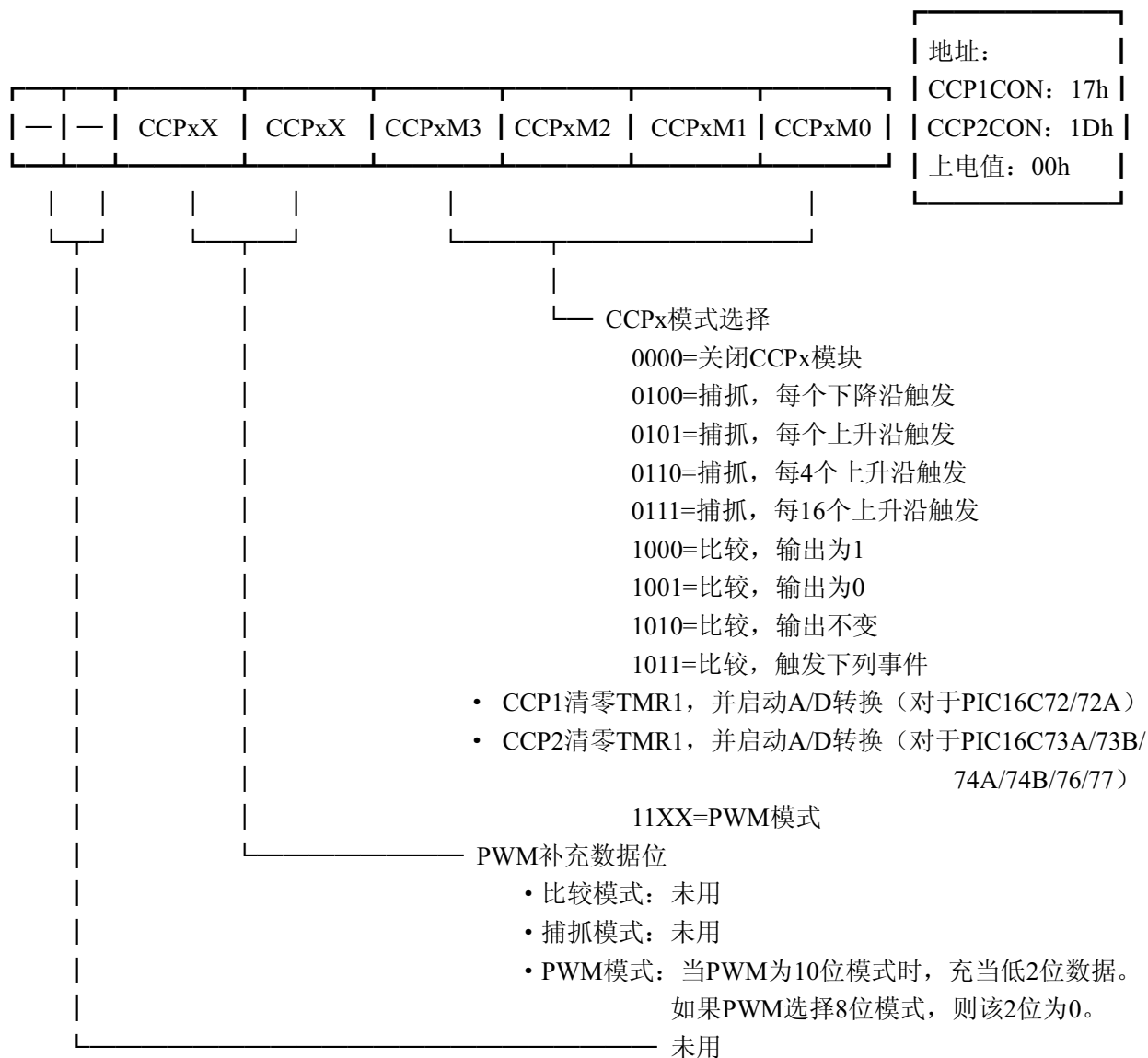


图2.9 CCP1CON/CCP2CON寄存器

§ 2.12 同步串行口模块 (SSP)

除 PIC16C71/711 外，其余 PIC16C72/72A/73A/73B/74A/74B/76/77 都有 SSP 模块，其功能和用法和 PIC16C62/63/65/66/67 的 SSP 完全一样，请参阅 § 1.11。

涉及型号					
73A	73B	74A	74B	76	77

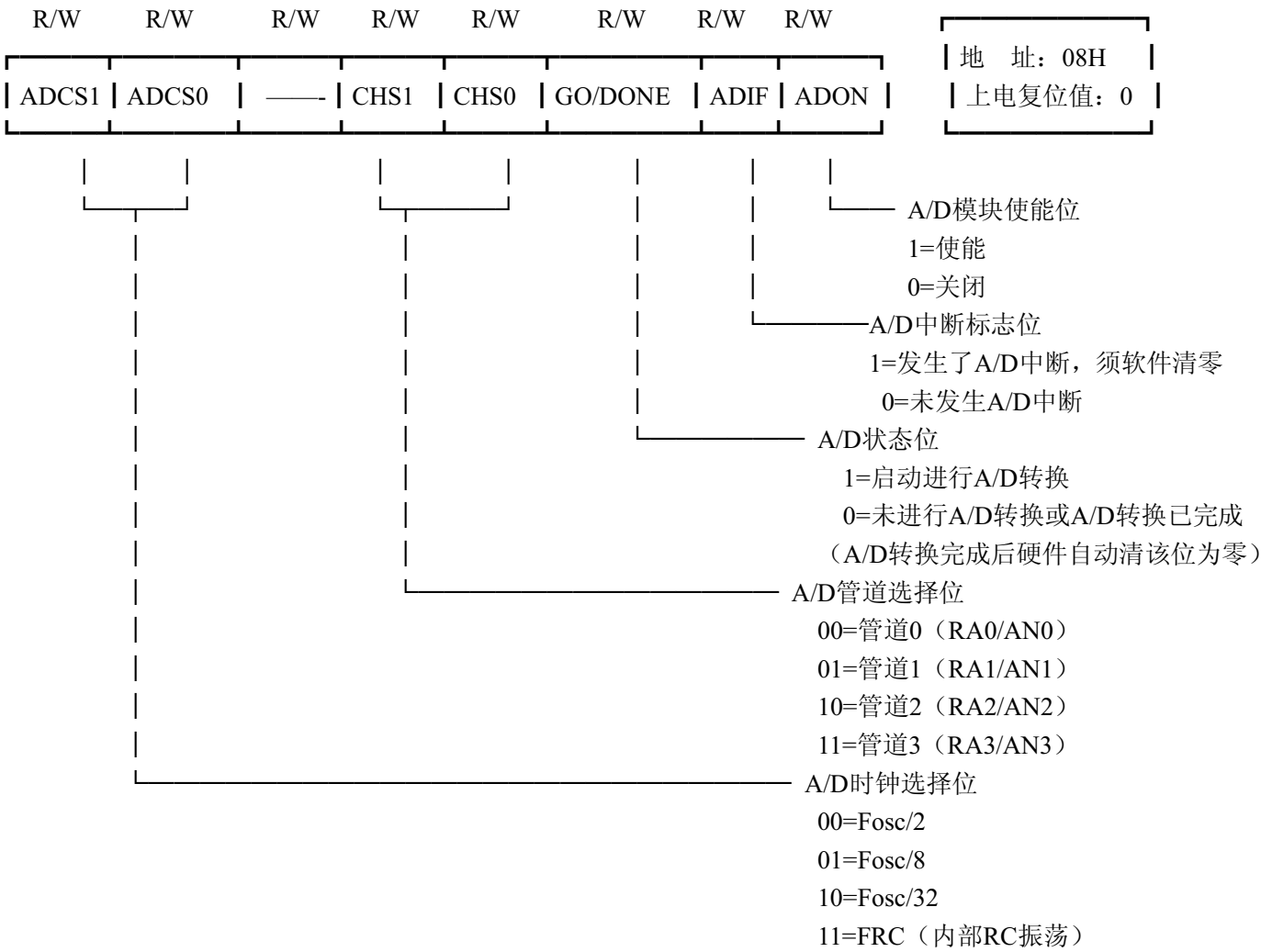
§ 2.14 A/D转换

PIC16C7X最显著的特点就是带8位A/D转换功能，其中PIC16C74A/74B/77有8路，PIC16C72/72A/73A/73B/76有5路，而PIC16C71/711有4路。这些芯片中间A/D输入都由一个多路开关连到相同的采样电路和保持电路上，采用逐次逼近算法进行模数转换。参考电压是软件可编程的，可以是芯片的电压值VDD，也可以选择RA3/AN3/VREF脚上的电压值。

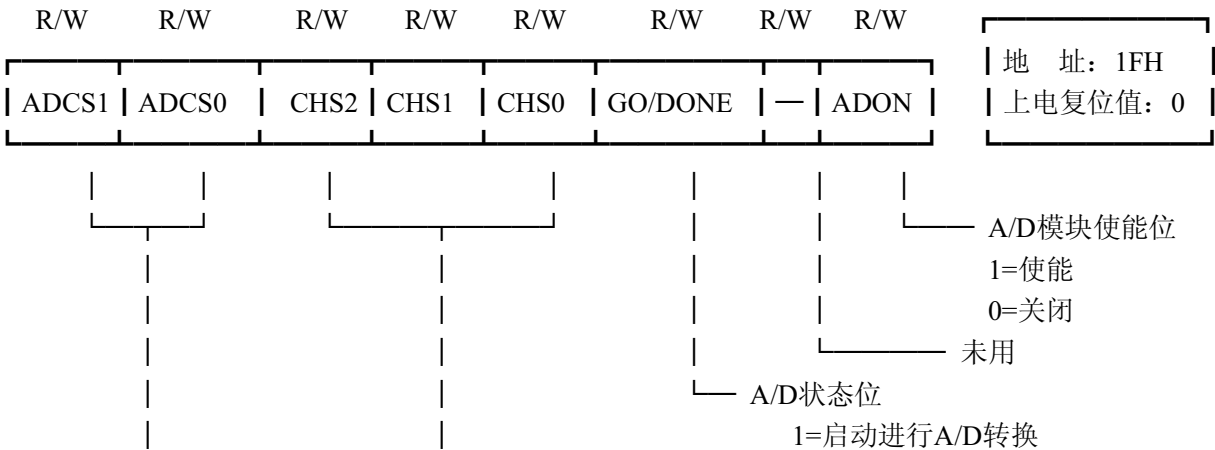
有关A/D模块有三个寄存器：

- 1. A/D结果寄存器 — ADRES，存放A/D转换的数值结果；
- 2. A/D控制寄存器0 — ADCON0，控制A/D转换操作；
- 3. A/D控制寄存器1 — ADCON1，控制选择A/D口线的功能。

下面是ADON0/ADCON1各个位的意义：



a. PIC16C70/71/711



注：A=模拟输入口 D=数字I/O口 VREF=参考电压 *=仅74/74A有

b. 16C72/72A/73A/73B/74A/74B

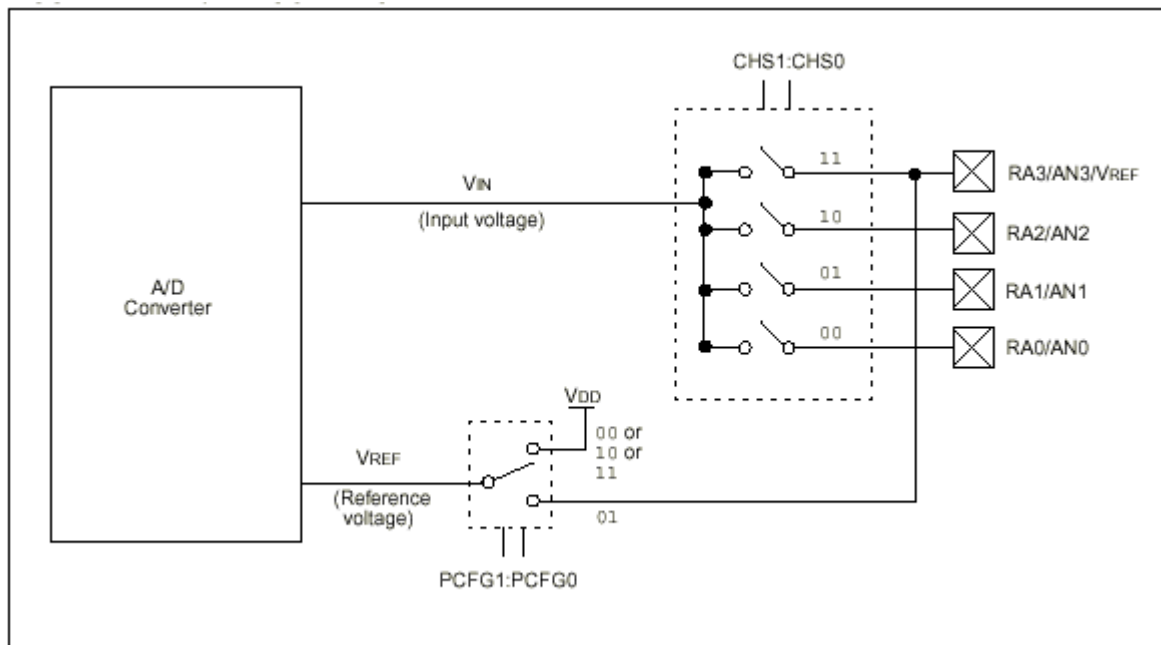
表2.5 A/D控制寄存器ADCON1

当将A/D口线选为模拟输入时，应把其相应的I/O方向控制位（TRIS）置为1。在A/D转换开始前，A/D输入口先采样一段时间，然后再进行A/D转换计算。当一次A/D转换完成后，其结果数值放入ADRES寄存器中，硬件会自动把GO/DONE（ADCON0<2>）清为零，并将A/D中断请求位ADIF（PIR1<6>）置为1发出中断请求。

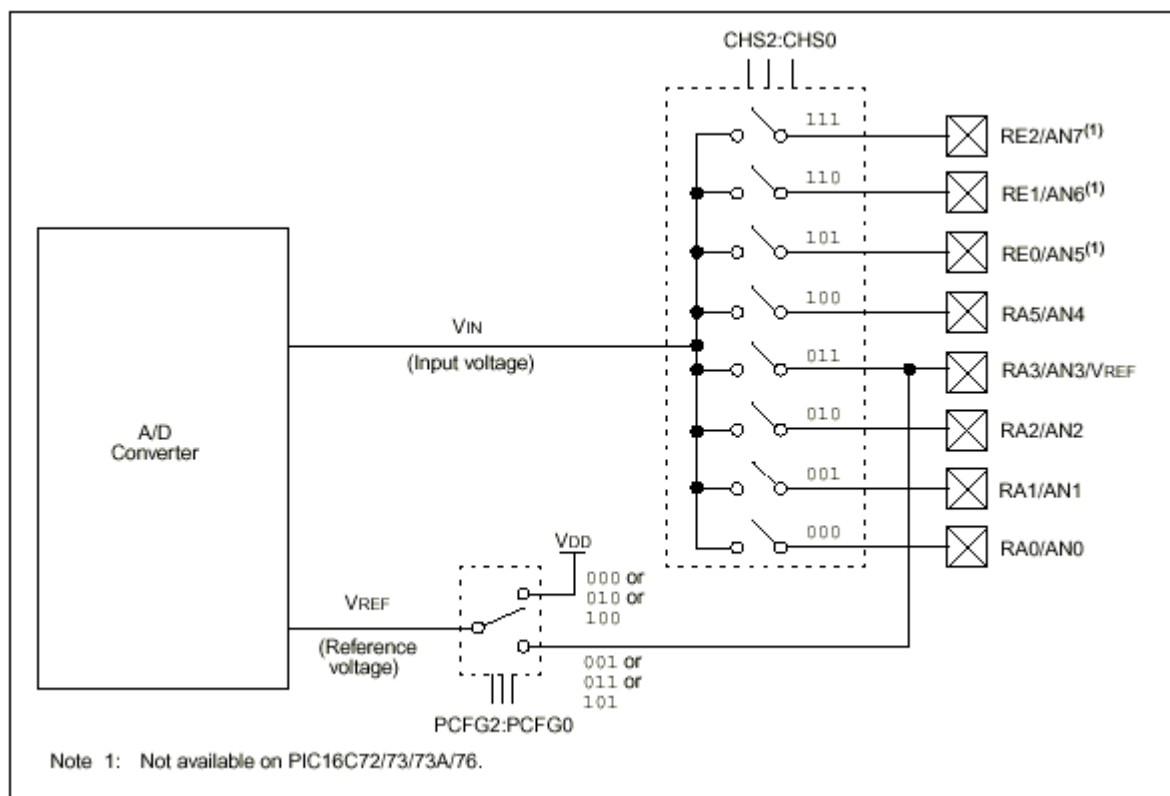
下面是进行A/D转换的程序编写步骤：

1. 设置A/D功能模块
 - 设置选择模拟输入口/参考电压源（ADCON1，ADCON0<5:3>）
 - 选择A/D转换时钟（ADCON0<7:6>）
 - 打开使能A/D模块（ADCON0<0>）
2. 设置A/D中断（如果需要A/D中断功能）
 - 清ADIF=0
 - 置ADIE=1
 - 置GIE=1
3. 等待A/D采样完成
4. 启动A/D转换
 - 置GO/DONE=1
5. 等待A/D转换完成，可以通过以下检测方法来判断
 - 或者软件查询GO/DONE的状态(看是否变为0)
 - 或者等待A/D中断产生
6. 读A/D结果寄存器ADRES
7. 如果需要再进行另一次A/D转换，须等待至少2TAD时间

下面是A/D模块方块图：



PIC71/711



PIC72/73/74/76
图2.11 A/D模块图

下面就A/D转换的各个环节进行详细叙述。

§ 2.14.1 A/D采样

大家知道，如果要使A/D转换达到希望的精度，就必须使采样电路中的保持电容（Chold）充分充电，让其达到A/D输入端的电压水平。

下图是PIC16C7X A/D输入的采样电路。

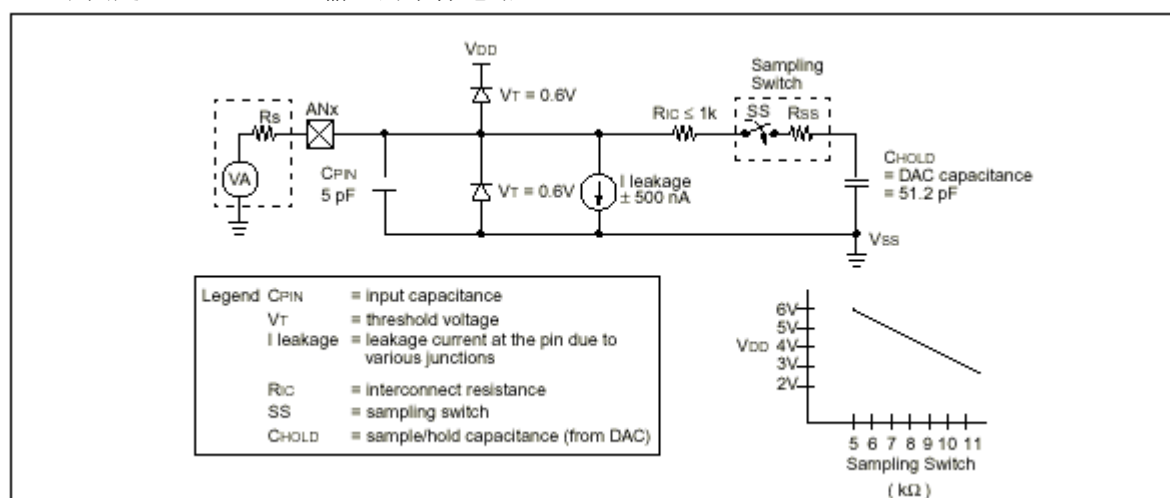


图 2.12 A/D 采样电路

从图中可看出，由于 A/D 输入脚亦可定义为数字脚，所以它们内部也接有反向偏压二极管，因此模拟输入必须落在 $VSS \sim VDD$ 范围内。如果模拟量输入超出 V_T (0.6V)，则会出现二极管偏置挂起。负载电阻 R_S 和采样开关电阻 R_{SS} 会直接影响 $CHOLD$ 的充电时间。如图中曲线所示， R_{SS} 的阻值亦随 VDD 而变化， VDD 越低，则 R_{SS} 会越大。对于负载电阻 R_S 而言，最大值不应超出 $10K\Omega$ ，这主要是为了满足管脚上的漏电流参数。在 $10K$ 的负

载阻抗下，当 VREF=VDD=5V 时，由于漏电流产生的最大可能误差为±5mv 或±1/4LSB。

下列公式可用来计算最短采样时间（即采样所需的最短时间）：

TSMP=放大器设置时间+Chold 充电时间+温度系数校正
=5 μ S+TC+[(Temp-25℃)(0.05 μ S/℃)]

其中充电时间 TC=512PF(RIC+RSS+RS)Ln(1/511)

该公式允许 1/2Lsb 的误差（512 步逐次逼近式 A/D）。

例：设 负载电阻 RS=10K Ω

电源电压 VDD=5V

温度=50℃

则 由 VDD=5V 可得出 RSS=7K Ω

而 RIC=1K Ω

得出 TC=512PF(1K Ω +7K Ω +10K Ω)Ln(0.0020)

=5.724 μ S

所以采样时间 TSMP=5 μ S+5.724 μ S+(50℃-25℃)(0.05 μ S/℃)
=11.974 μ S

大家可能注意到，参考电压 VREF 对采样时间并没有影响。

§ 2.14.2 A/D 转换时钟

我们把每位 A/D 转换所需时间定义为 TAD，则完成一次 8 位的 A/D 转换所需时间为 9.5TAD。A/D 转换时钟源可来自芯片内的 RC 振荡，也可来自芯片的 OSC1 振荡输入，这由软件设置选择如下：

- 1. 2TOSC
- 2. 8TOSC
- 3. 32TOSC
- 4. 内部 RC 振荡

为保证正确的 A/D 转换，TAD 必须满足一定的条件：

型 号	最短 TAD
PIC16C71	2.0 μ S
其他 PIC16C7X	1.6 μ S

表 2.6 最短 TAD 值

所以当我们设计选择芯片的型号、工作频率时，要考虑到最短 TAD 的因素，下表是 PIC16C7X 各种型号在各种振荡频率下的 TAD 值。

A/D 时钟源		芯 片 振 荡 频 率				
周 期	ADCS1:ADCS0	20MHz	16MHz	4MHz	1MHz	333.33KHz
2Tosc	00	100ns(2)	125ns(2)	500ns(2)	2.0 μ s	6 μ s
8Tosc	01	400ns(2)	500ns(2)	2.0 μ s	8.0 μ s	24 μ s(3)
32Tosc	10	1.6 μ s(2)	2.0 μ s	8.0 μ s	32.0 μ s(3)	96 μ s(3)
RC	11	2-6 μ s(1, 4)	2-6 μ s(1, 4)	2-6 μ s(1, 4)	2-6 μ s(1)	2-6 μ s(1)

- 注： 1. RC 时钟源典型 TAD=4 μ s。
2. 这些值超出最短 TAD。
3. 如果要取得更快的转换速度，建议选取别的时钟源。
4. 在 RC 模式下，如果芯片振荡频率高于 1MHz，则转换精度可能超出指标。

a. PIC16C71/711

A/D 时钟源		芯 片 振 荡 频 率			
周 期	ADCS1:ADCS0	20MHz	5MHz	1.25MHz	333.33KHz
2Tosc	00	100ns(2)	400ns(2)	1.6 μ s	6 μ s

8Tosc	01	400ns(2)	1.6 μ s	6.4 μ s	24 μ s(3)
32Tosc	10	1.6 μ s(2)	6.4 μ s	25.6 μ s(3)	96 μ s(3)
RC	11	2-6 μ s(1, 4)	2-6 μ s(1, 4)	2-6 μ s(1, 4)	2-6 μ s(1)

注：1. RC 时钟源典型 TAD=4 μ s。

2. 这些值超出最短 TAD。
3. 如果要取得更快的转换速度，建议选取别的时钟源。
4. 在 RC 模式下，如果芯片振荡频率高于 1MHz，则转换精度可能超出指标。

b. 其他 PIC16C7X

表 2.7 TAD 和芯片型号、频率关系表

另外一点，当一次 A/D 转换完成后，如果还要进行下一次的 A/D 转换，则它们之间至少要有 2TAD 的延迟（间隔）方可。

§ 2.14.3 设置 A/D 口

A/D 口由三个寄存器来设置控制：

ADCON1：数字/模拟选择及 VREF 选择。

TRISA：方向控制，当 PORTA 选为模拟口时应置为 1。

TRISE：方向控制，当 PORTE 选为模拟口时应置为 1，仅 PIC16C74 有。

请参阅这些寄存器的描述。

§ 2.14.4 A/D 转换例程

这里给出 2 个 A/D 转换的例程。

- 设：
- ①RA 作为 A/D 输入端
 - ②VREF=VDD
 - ③A/D 时钟采用内部 RC
 - ④使用 A/D 中断

例（a）：PIC16C71/711 例程

```

BSF      STATUS, RP0
CLRF     ADCON1           ;设置 A/D 口，参考电压源
BCF      STATUS, RP0
MOVLW    0XC1
MOVWF    ADCON0           ;A/D 时钟为 RC，RA0 作 A/D 输入等
BSF      INTCON, ADIE     ;开 A/D 中断
BSF      INTCON, GIE      ;开总中断
:
:                          ;一段延时以保证足够的采样时间
:
BSF      ADCON0, GO        ;启动 A/D 转换
:                          ;A/D 转换完成后会置 ADIF=1
:                          ;请求中断并清 GO/DONE=0

```

例（b）：PIC16C72/72A/73A/73B/74A/74B/76/77 例程

```

BSF      STATUS, RP0
CLRF     ADCON1           ;同例（a）
BSF      PIE1, ADIE       ;开 A/D 中断
BCF      STATUS, RP0
MOVLW    0XC1

```

MOVWF	ADCON0	
BCF	PIR1, ADIF	;清 A/D 中断标志位
BSF	INTCON, PEIE	;开“功能模块”中断
BSF	INTCON, GIE	;开总中断
...		
BSF	ADCON0, GO	;同例 (a)

当一个 A/D 转换过程正在进行当中，如果这时 GO/DONE 位被清零，则这次 A/D 转换就会中止，而 ADRES 寄存器保持原来的值不变。一次 A/D 转换完成或被中止后，如果想要进行下一次 A/D 转换，则需至少等待 2TAD 的延时时间，也即过 2TAD 时间后，采样电路恢复采样工作。

§ 2.14.5 睡眠中的 A/D 转换

在 PIC16C7X 中，A/D 模块在睡眠状态下仍可工作，但这首先需要用户选择内部 RC 作为 A/D 时钟 (ADCS1:ADCS0=I1)。当选择 RC 振荡为 A/D 时钟后，A/D 模块要等待一个指令周期后，才开始进行 A/D 转换，这样就允许让用户执行一条 Sleep (进入睡眠) 指令。当芯片进入睡眠后，将会消除 A/D 转换过程中产生的开关噪声，这样可使 A/D 转换更精确可靠。当 A/D 转换完成后，GO/DONE 位清零，转换结果被置入 ADRES 寄存器。这时如果 A/D 中断开启，则会产生中断请求并把芯片从睡眠中唤醒，但如果 A/D 中断关闭，则 A/D 模块也会关闭，尽管 ADON 位仍保持为 1。

注意，如果要在睡眠中进行 A/D 转换，还需在执行 Sleep 指令前置 GO/DONE=1。

如果 A/D 时钟不是 RC 振荡，那么 Sleep 指令将会中止本次 A/D 转换并关闭 A/D 模块，尽管 ADON 位仍会保持为 1。

当然，关闭 A/D 模块会使芯片的睡眠功耗处于最低。

§ 2.14.6 A/D 精度和误差

当 $VDD=5V \pm 10\%$ 且 $VREF=VDD$ 时，A/D 的精度是小于 $\pm 1\text{Lsb}$ ，如果 VDD 小于 5V 或者 VREF 小于 VDD，精度可能会降低。A/D 管脚上最大的漏电流是 $\pm 5\mu\text{A}$ 。

当芯片的振荡频率较低时，使用片内 RC 振荡作为 A/D 时钟会更好些。而当芯片振荡频率较高时，则选用 TOSC 作为 A/D 时钟较佳，且 TAD 最好不要短于所需的最短时间（见 § 2.13.2 描述），但也不要长于 $8\mu\text{S}$ 。

在睡眠中进行 A/D 转换则有利于转换的精度，见 § 2.14.5 的描述。

§ 2.14.7 复位对 A/D 的影响

因为复位会使所有的寄存器等于其复位值，所以复位将会关闭 A/D 模块并中止正在进行的 A/D 转换。

§ 2.14.8 CCP 模块触发 A/D 转换

在 PIC16C73A/73B/74A/74B/76/77 中，CCP2 模块可以用来触发 A/D 转换，而 PIC16C72/72A 则是用 CCP1 来触发 A/D 转换。

为了用 CCP 来触发 A/D 转换，必须使 CCP 模块设置成“比较输出触发特定事件”模式，即 $\text{CCPxCON}\langle 3:0 \rangle = 1011$ ，并打开 A/D 模块 ($\text{ADCON}=1$)。这样当触发发生时，GO/DONE=1 并启动 A/D 转换，TMR1 同时被清为零以便下一次计时触发。当然，在触发发生之前，用户应先完成 A/D 转换之前所需的设置（如 A/D 输入端、A/D 时钟选择等等）并等待足够的采样时间，参见 § 2.14.2 描述。

如果 A/D 模块没有开启 ($\text{ADON}=0$)，则 CCP 触发不会对 A/D 模块产生作用，但仍会使 TMR1 计数器清零。请参阅 § 2.11 CCP 模块的章节。

§ 2.14.9 A/D 电路连接

如果输入模拟电压超出 VSS 或 $VDD0.2V$ 以上，将会引起 A/D 转换精度低于期望值。对于 PIC16C70/71/71A

来说，由于 RA0/AN0 脚和 OSC1 紧靠在一起，所以会产生一定影响。如果不用到 4 路 A/D，建议用户慎用 AN0 端，而用其他的 A/D 输入端（AN1~AN3）。

有时为了清除输入量的偏差，可在 A/D 端上接 RC 滤波电路。但要注意电阻 R 的接入不能使负载电阻高于 10KΩ，任何外部元件的接入都要保证脚上的漏电流很小。

§ 2.14.10 A/D 传递函数

理想的 A/D 转换传递函数如下：

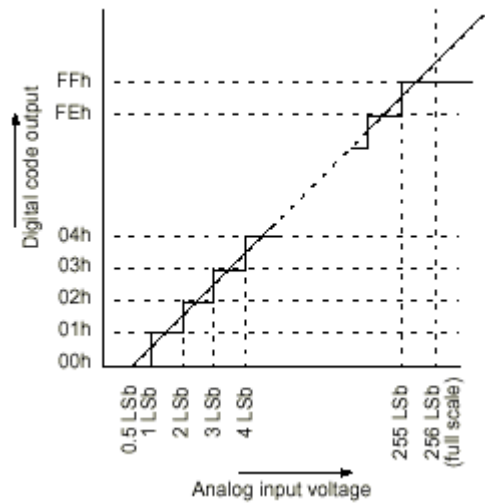
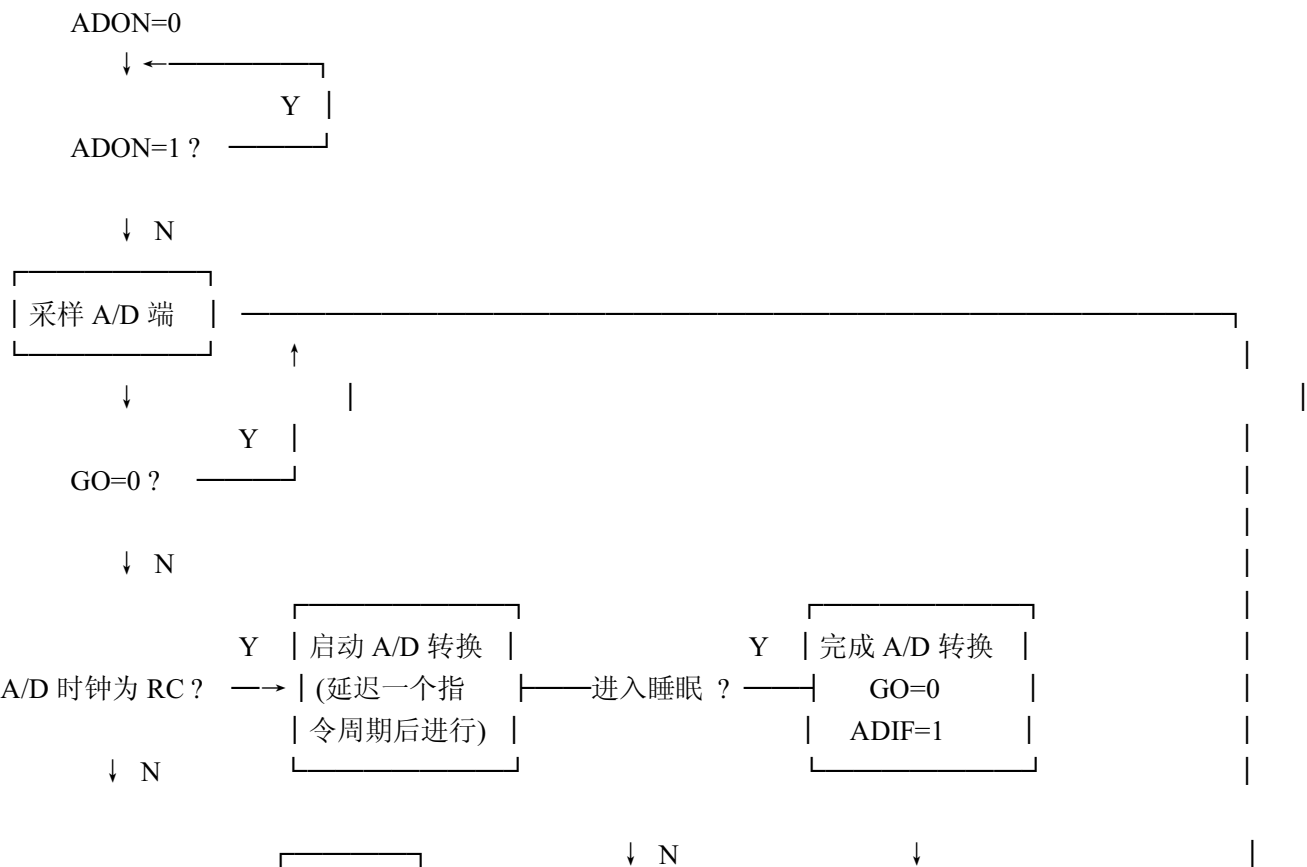


图 2.13 A/D 传递函数曲线

如图所示，当模拟输入电压为 1Lsb（或是 VREF 的 1/256）时发生第一个传递。

§ 2.14.11 A/D 转换流程

PIC16C7X A/D 转换的过程如下图所示：



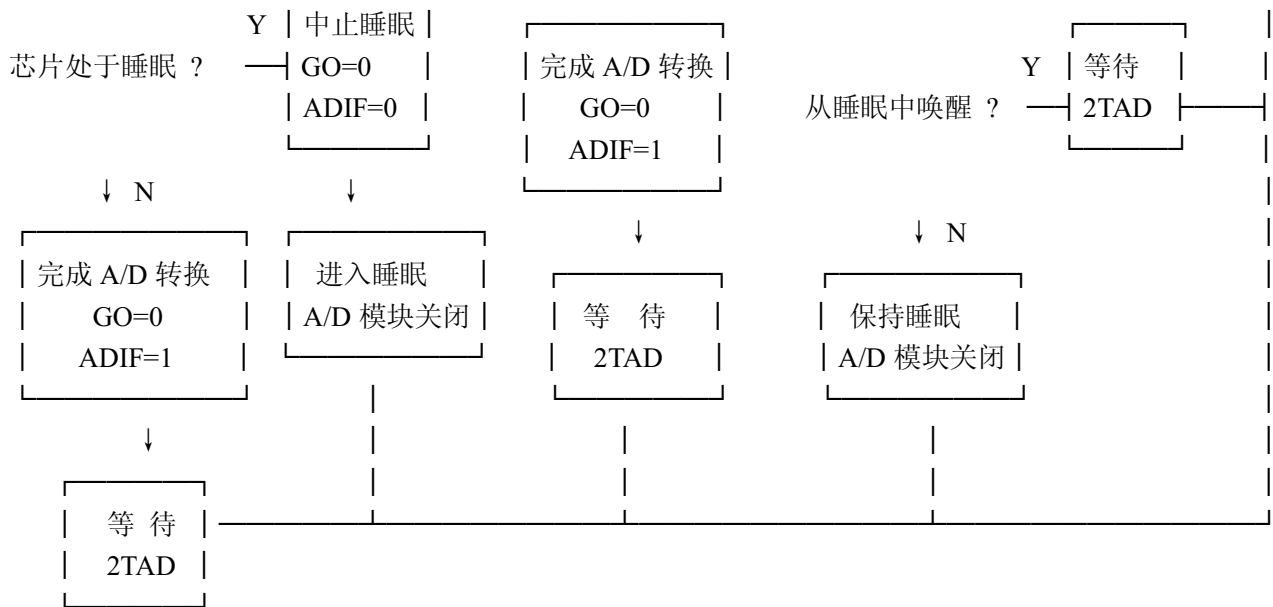
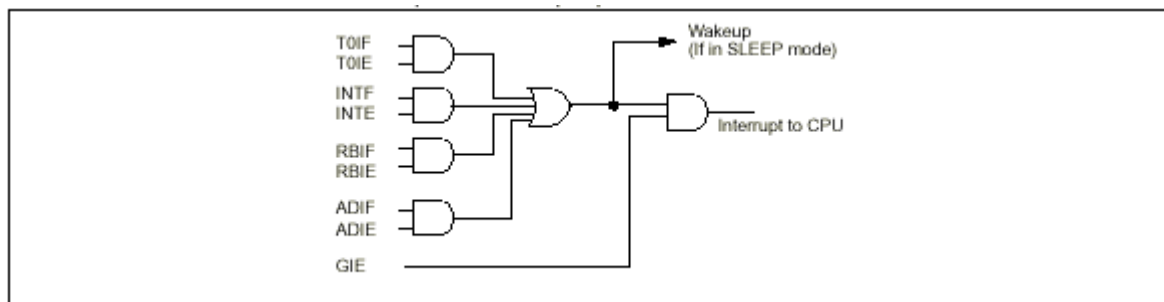


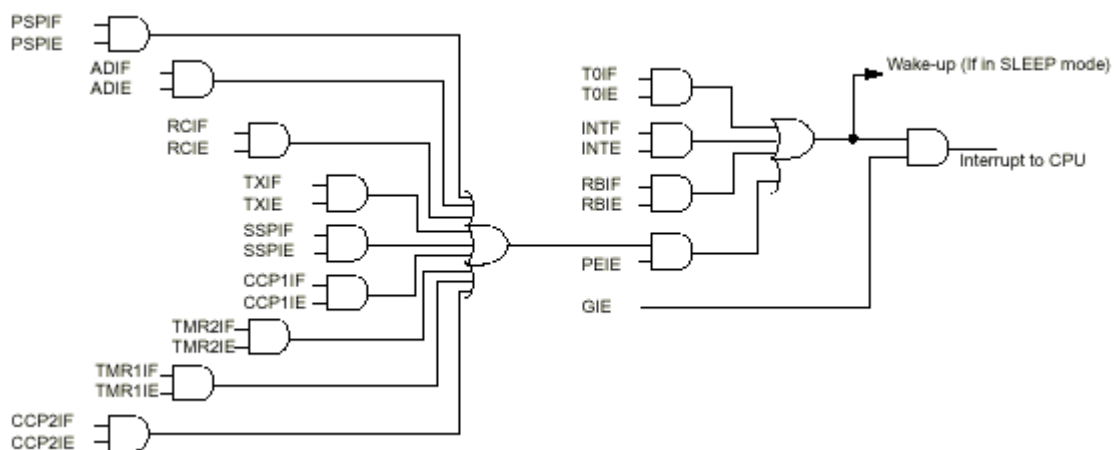
图 2.14 A/D 转换流程

§ 2.15 CPU 特性

PIC16C7X 和 PIC16C6X 一样具备有许多微处理器特性，象多种振荡方式选择、多种复位、程序保密位、掉电复位锁定、上电延时器 PWRTE、振荡起振定时器 OST、多种中断功能及 ID 码等等。在这些方面，除了在中断源中多出一种 A/D 中断方式外，PIC16C7X 和 PIC16C6X 是完全一样的。PIC16C7X 比 PIC16C6X 多出一种 A/D 中断方式，见下面图表。



PIC16C71/710/711



PIC16C7X

图 2.15 PIC16C7X 中断逻辑图

中 断 源	标志位	使能位	74A/74B	73A/73B	72/72A	71/711
外部触发中断 INT	INTF	INTE	√	√	√	√
TMR0 溢出中断	T0IF	T0IE	√	√	√	√
PORTB<7:4>中断	RBIF	RBIE	√	√	√	√
TMR1 中断	TMR1IF	TMR1IE	√	√	√	—
TMR2 中断	TMR2IF	TMR2IE	√	√	√	—
CCP1 中断	CCP1IF	CCP1IE	√	√	√	—
CCP2 中断	CCP2IF	CCP2IE	√	√	—	—
SCI 同步发送中断	TXIF	TXIE	√	√	—	—
SCI 同步接收中断	RCIF	RCIE	√	√	—	—
SSP 中断	SSPIF	SSPIE	√	√	√	—
并行口中断	PSPIF	PSPIE	√	—	—	—
A/D 中断	ADIF	ADIE	√	√	√	√

表 2.8 PIC16C7X 中断源

在中断现场保护方面，PIC16C71/711 和 PIC16C61 一样，而 PIC16C72/72A/73A/73B/74A/74B/76/77 则和其它 PIC16C6X 一样，请参阅 § 1.13.4 有关中断现场保护的例程。

关于 PIC16C7X 的其他 CPU 特性，由于和 PIC16C6X 完全一样，请读者参阅 § 1.13 章节的内容，不再赘述。具体对于其中 PIC16C7X 型号，读者可以先看它具备了哪些资源如程序存储器，寄存器是多少，I/O 有多少，功能部件（CCP、SPI、SCI、A/D 等）有哪些等等，然后再确认其中哪些内容是和 PIC16C6X 中的对应型号是一样的，则可以参阅有关 PIC16C6X 的章节，对属于 PIC16C7X 特有的，则仔细阅读本章节的描述。

第三章 PIC16C92X 功能原理

PIC16C92X 目前各型号主要功能配置如下：

型号	振荡	EPROM	RAM	定时器	CCP 模块	串行口	并行口	中断源	输入线	A/D	LCD 驱动	电压范围	I/O	封装
16C923	DC~8M	4K×14	174	3	1	SPI/I2C	1	8	27	—	4×32	3.0-6.0	25	64-DIP, TQFP 68-PLCC, DIE
16C924	DC~8M	4K×14	174	3	1	SPI/I2C	1	9	27	5 路	4×32	3.0-6.0	25	64-DIP, TQFP 68-PLCC, DIE

表 3.1 PIC16C92X 型号功能表

§ 3.1 主要功能特点

一、高性能 RISC 结构 CPU

- 精简指令集，仅 35 条单字节指令，易学易用
- 除地址分支跳转指令为双周期指令，其余皆为单周期指令
- 执行速度：DC~500ns
- 八级硬件堆栈
- 多种硬件中断功能

二、功能部件特性

- 25 根双向可独立编程 I/O 线
- 25~27 根单向输入/输出线
- 大驱动电流
 - 每个 I/O 引脚最大拉电流 10mA
 - 每个 I/O 引脚最大灌电流 10mA
- 8 位定时器/计数器 TMR0，带 8 位预分频器
- 16 位定时器/计数器 TMR1，睡眠中仍可计数
- 8 位定时器/计数器 TMR2，带有 8 位的周期寄存器及预分频器和后分频器
- 1 路 CCP 模块（捕获输入/比较输出/PWM 输出）
- 并行口操作
- 同步串行口 I2C/SPI 接口
- 同步通讯接口 SCI/USART 接口
- 可编程液晶（LCD）驱动模块
 - 双 LCD 时钟源
 - 睡眠中仍可驱动 LCD
 - 静态，1/2，1/3，1/4 四种 LCD 驱动模式
 - 16 字节 LCD 寄存器
 - 最多 4×29 LCD 像素驱动
- 多路 8 位 A/D 输入通道（仅 PIC16C924 有）

三、微控制器特性

- 内置上电复位电路
- 上电定时器和振荡起振定时器保障上电复位正常和建立稳定振荡
- 自振式看门狗，睡眠中仍能计时
- 程序保密位保障程序不被非法拷贝
- 低功耗模式
- 四种可选振荡方式
 - 低成本阻容：RC
 - 标准晶体/陶瓷：XT
 - 高速晶体/陶瓷：HS
 - 低频晶体：LP

四、CMOS 工艺特性

- 低功耗
 - < 2mA @5.5V, 4MHz
 - 22.5mA @4V, 32KHz
 - <1 μ A 低功耗模式下
- 全静态设计
- 宽工作电压：3.0V~6.0V
- 宽工作温度：
 - 商用级： 0℃~+70℃
 - 工业级： -40℃~+85℃
 - 汽车级： -40℃~+125℃

PIC16C92X 是片上带液晶显示驱动的单片机，所以它在带 LCD 的产品如家电、仪表、电信等领域有良好的前景。

§ 3.2 芯片类型

PIC16C92X 目前有以下三种类型的芯片可供选择：

一、可重擦写型（UV）

系陶瓷封装，中间开有窗口，可以用紫外光擦除重新烧写，是开发中理想的选用芯片。

二、一次性可编程型（OTP）

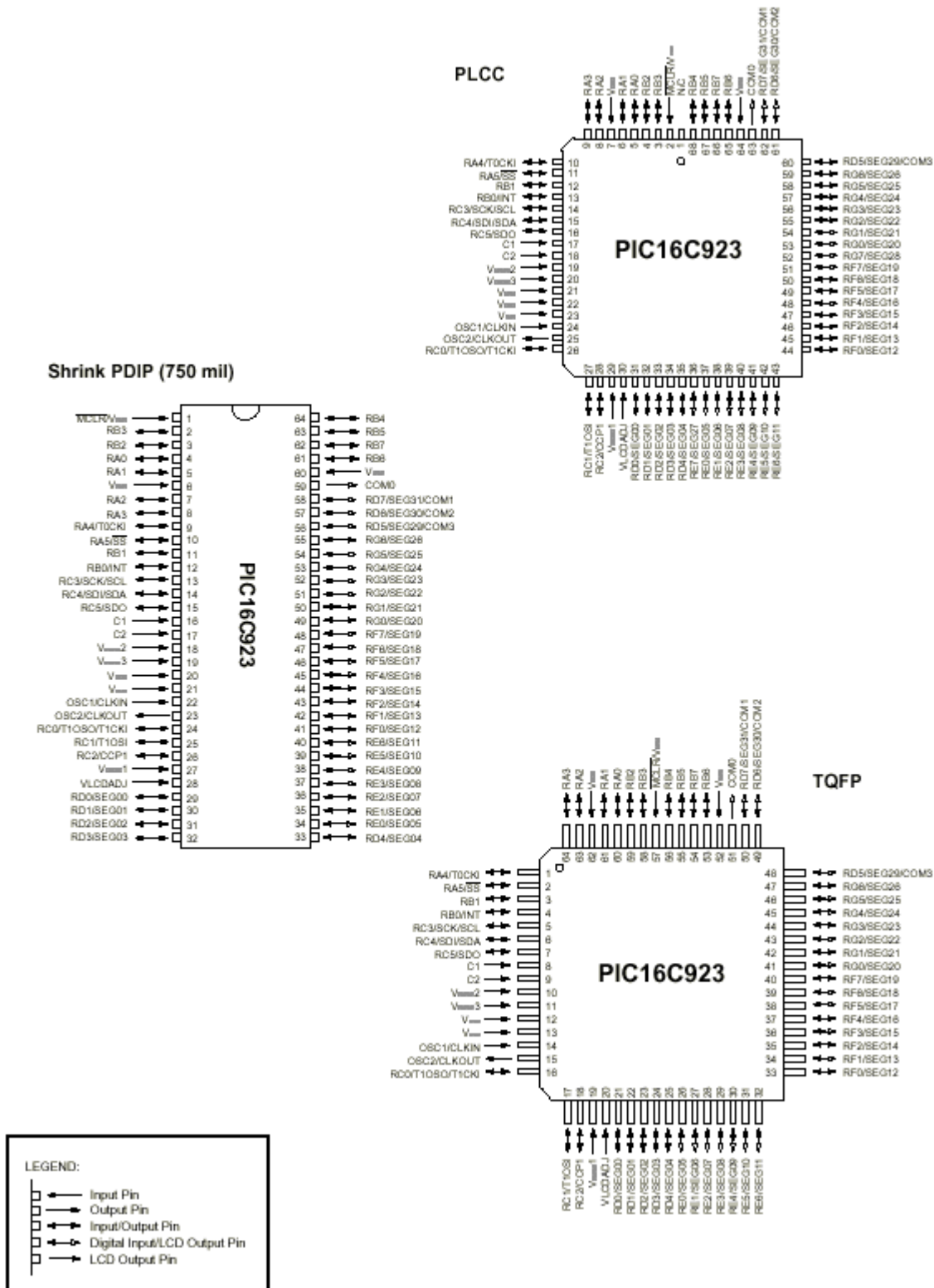
这是 PIC16C92X 最具代表性的芯片，系塑料封装，允许用户用编程器对其进行一次性编程。这个特点使它成为最具商用意义的型号。在多变竞争激烈的电子产品市场中，OTP 型芯片具有三大意义：1)产品快速上市；2)可随时修改程序，免除掩膜风险；3)减轻库存压力。

三、裸片（Dies）

未封装半导体芯，适合于大批量嵌入式应用。

§ 3.3 引脚介绍

PIC16C92X 芯片具有的引脚如下：



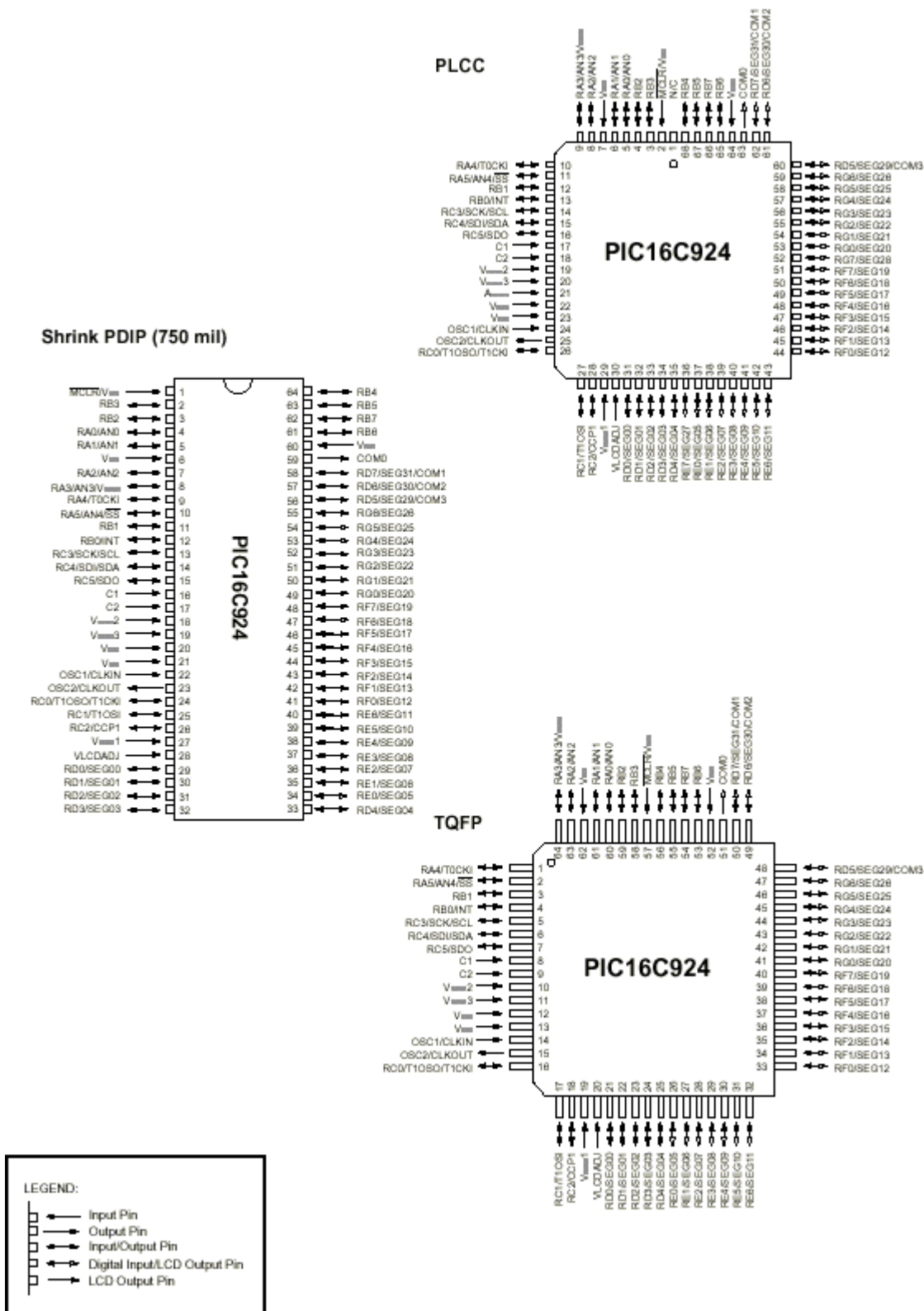


图 3.1 PIC16C92X 引脚

各型号的引脚意义如下：

引脚名	I/O 特性	电平	功能
OSC1/CLKIN	输入	CMOS	振荡输入脚
OSC2/CLKOUT	输出	—	振荡输出脚
MCLR	输入	斯密特输入	复位输入脚，低电平有效

RA0/AN0	I/O	TTL	PORTA I/O 口，双向可编程 在 16C924 中，亦可做为 A/D 输入 A/D 输入通道 0
RA1/AN1	I/O	TTL	A/D 输入通道 1
RA2/AN2	I/O	TTL	A/D 输入通道 2
RA3/AN3/VREF	I/O	TTL	A/D 输入通道 3
RA4/T0CKI	I/O	斯密特输入	也可作为 TMR0 计数器信号输入端
RA5/SS/AN4/SS	I/O	TTL	A/D 输入通道 4/同步串行口的从属器选择输入
RB0/INT	I/O	TTL/斯密特	PORTB I/O 口，双向可编程，并带可编程弱上拉 也可作为外部中断信号输入 具电平变化中断功能 具电平变化中断功能 具电平变化中断功能 具电平变化中断功能
RB1	I/O	TTL	
RB2	I/O	TTL	
RB3	I/O	TTL	
RB4	I/O	TTL	
RB5	I/O	TTL	
RB6	I/O	TTL	
RB7	I/O	TTL	
RC0/T1OS0/T1CKI	I/O	斯密特输入	PORTC I/O 口，双向可编程 亦可作 TMR1 振荡输出/TMR1 时钟输入
RC1/T1OS1/CCP2	I/O	斯密特输入	亦可作 TMR1 振荡输入/CCP2 输入输出
RC2/CCP1	I/O	斯密特输入	亦可作 CCP1 输入输出
RC3/SCK/SCL	I/O	斯密特输入	亦可作同步串行通讯时钟
RC4/SDI/SDA	I/O	斯密特输入	亦可作 SPI 通讯之数据线
RC5/SDO	I/O	斯密特输入	亦可作 SPI 通讯之数据输出线
C1	—	电源	LCD 电压产生端
C2	—	电源	LCD 电压产生端
COM0	—	L	COM0 端
RD0/SEG00	I/O/LCD	斯密特输入	PORTD 数字 I/O 口，双向可编程 亦可作为 LCD 驱动的 Seg 端或 COM 端 亦可作为 LCD 驱动 Seg00
RD1/SEG01	I/O/LCD	斯密特输入	
RD2/SEG02	I/O/LCD	斯密特输入	
RD3/SEG03	I/O/LCD	斯密特输入	
RD4/SEG04	I/O/LCD	斯密特输入	
RD5/SEG29/COM3	I/O/LCD	斯密特输入	
RD6/SEG30/COM2	I/O/LCD	斯密特输入	
RD7/SEG31/COM1	I/O/LCD	斯密特输入	
RE0/SEG05	I/O/LCD	斯密特输入	PORTE 数字 I/O 口，双向可编程 亦可作为 LCD 驱动的 Seg 端 亦可作为 LCD 驱动 Seg05
RE1/SEG06	I/O/LCD	斯密特输入	
RE2/SEG07	I/O/LCD	斯密特输入	
RE3/SEG08	I/O/LCD	斯密特输入	

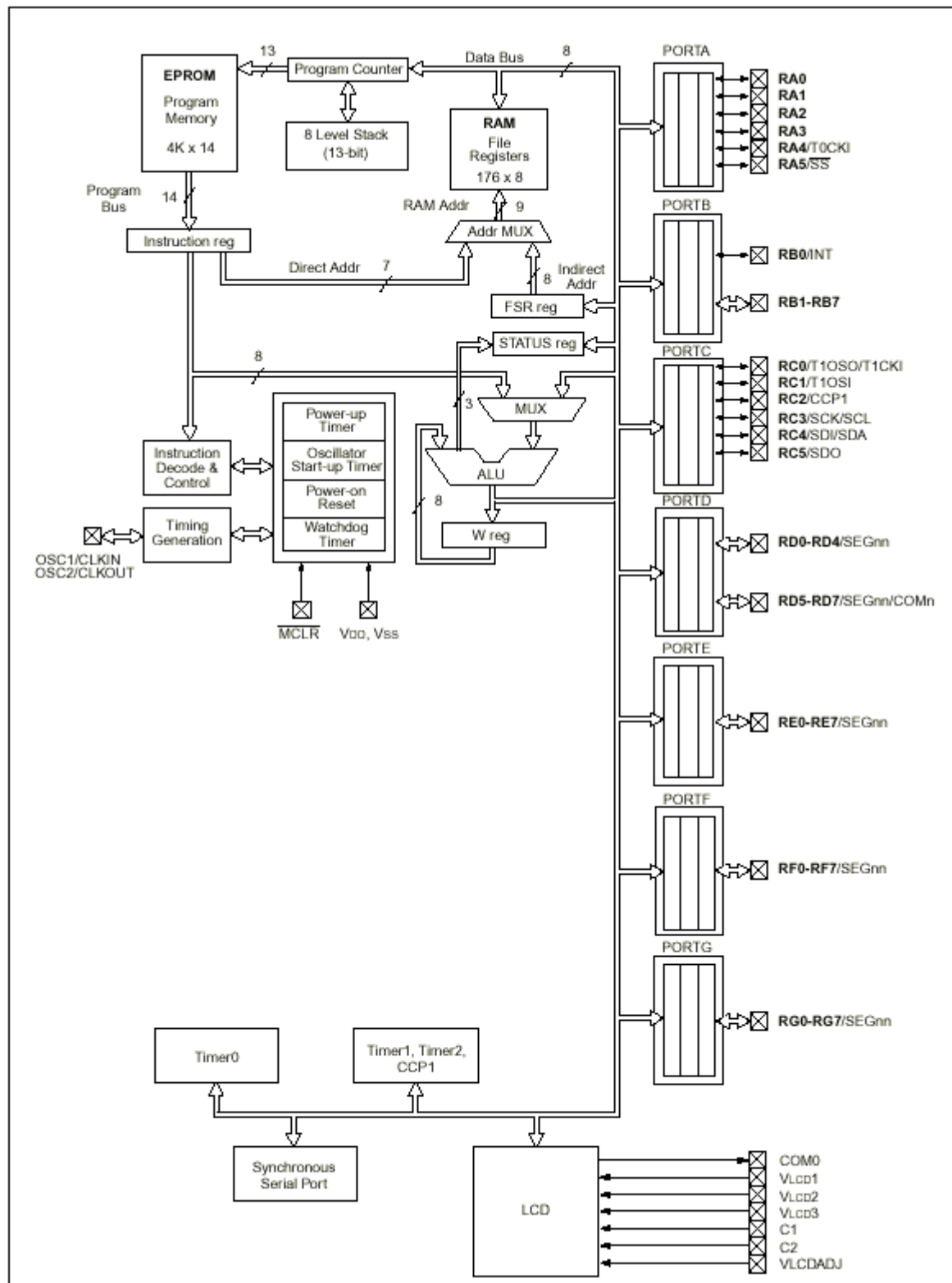
RE4/SEG09	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg09
RE5/SEG10	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg10
RE6/SEG11	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg11
RE7/SEG27	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg27
RF0/SEG12	I/O/LCD	斯密特输入	PORTF 数字输入口 亦可作为 LCD 驱动 Seg 端 亦可作为 LCD 驱动 Seg12
RF1/SEG13	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg13
RF2/SEG14	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg14
RF3/SEG15	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg15
RF4/SEG16	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg16
RF5/SEG17	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg17
RF6/SEG18	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg18
RF7/SEG19	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg19
RG0/SEG20	I/O/LCD	斯密特输入	PORTG 数字输入口 亦可作为 LCD 驱动之 Seg 端 亦可作为 LCD 驱动 Seg20
RG1/SEG21	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg21
RG2/SEG22	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg22
RG3/SEG23	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg23
RG4/SEG24	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg24
RG5/SEG25	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg25
RG6/SEG26	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg26
RG7/SEG28	I/O/LCD	斯密特输入	亦可作为 LCD 驱动 Seg28
VLCDADJ	—	电源	LCD 电平产生端
AVDD	—	电源	模拟电源
VLCD1	—	电源	LDD 电压
VLCD2	—	电源	LCD 电压
VLCD3	—	电源	LCD 电压
VDD	—	电源	数字电源
VSS	—	电源	电源地
NC	—	—	未用

表 3.2 PIC16C92X 引脚功能

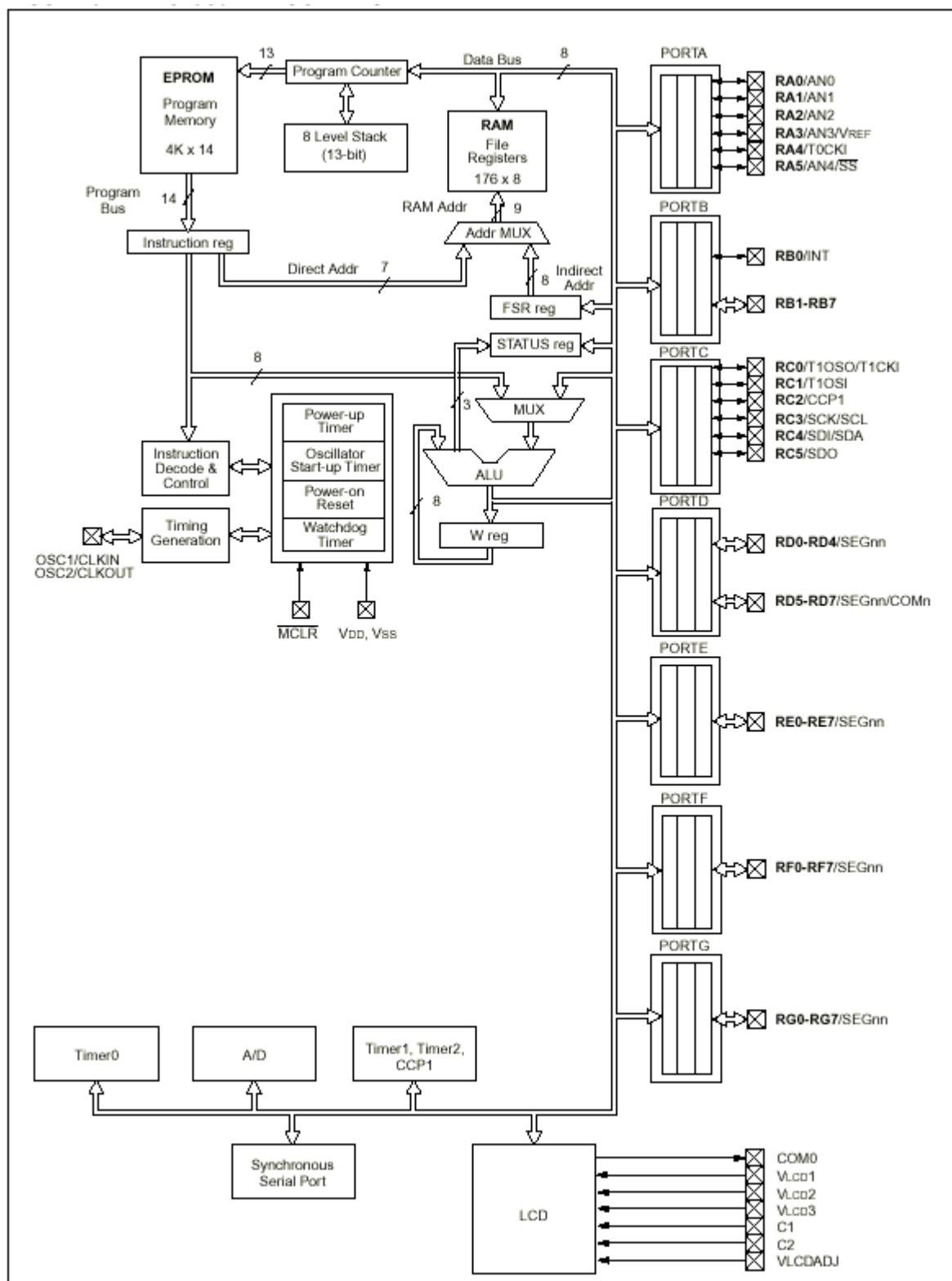
§ 3.4 内部结构

PIC16C92X 内部采用独立分离 8 位数据总线和 14 位指令总线的“哈佛”结构，它是一种“精简指令集”(RISC)的 CPU 设计，所以可以达到很高的运行速度。8 位的算术逻辑单元 ALU 可以完成加减、移位和各种布尔逻辑运算，另外它还集成了众多的功能模块如 I/O 口、定时器、A/D 模块 (16C924)、CCP 模块、LCD 驱动、SSP 串行口以及上电复位电路、看门狗电路、上电/起振延时器等等。

在 PIC16C92X 片内带有 4K 的 14 位宽程序存储器 (ROM)，176 个 8 位的数据寄存器 (RAM)。所有特殊寄存器包括程序计数器，I/O 寄存器等都直接映射到 RAM 单元中，所以程序编码非常简洁高效。



a. 16C923 内部结构



b. 16C924 内部结构

图 3.2 PIC16C92X 内部结构

§ 3.5 指令时序和流水作业

和其他 PIC16CXXX 一样，参阅 § 1.5。

§ 3.6 程序存储器和堆栈

PIC16C92X 有一个 13 位宽的计数器 PC，最大可寻址 8K 空间。目前 PIC16C92X 仅使用了头 4K，超出这 4K 的指令寻址将导致在物理空间上的回绕。

PIC16C92X 的复位地址在 0000h。

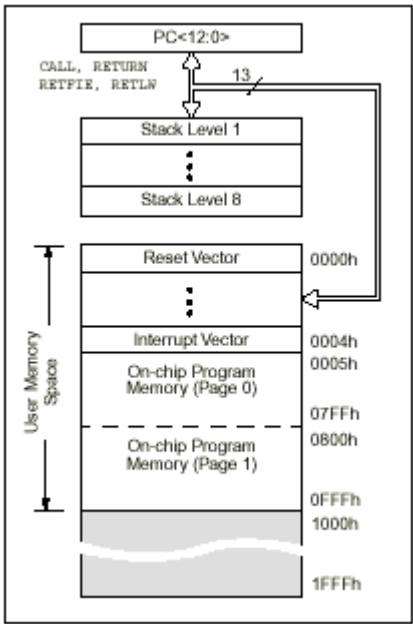


图 3.3 PIC16C92X 程序存储器和堆栈

PIC16C92X 的 4K 程序区被分为 2 个页面 (page)：页面 0 (0000h~07FFh)，页面 1 (0800h~1FFFh)，每个页面为 2K 空间，由 PCLATH<3>位控制，请参阅 § 1.7。

PIC16C92X 的堆栈有 13×8 的独立空间，不占有程序区。

§ 3.7 数据存储器

PIC16C92X 数据存储器被分为 4 个体，由状态寄存器 STATUS<6:5> (RP1:RP0) 来选择：

RP1	RP0	页面	寄存器地址
0	0	bank0	00h — 7Fh
0	1	bank1	80h — FFh
1	0	bank2	100h — 17Fh

表 3.3 寄存器体及其地址

这其中包括了通用寄存器和特殊寄存器。在每个体的上部，有一些特殊寄存器是相互映射的，即在物理上是相同的寄存器。另有一些位置上的寄存器物理上不存在，暂未使用。

[illegible]

图 3.4 PIC16C92X 数据存储器结构

§ 3.7.1 通用寄存器

通用寄存器即是可用于存储各种数据的寄存器，这些寄存器的内容在单片机上电复位后是随机不定的，在非上电复位后则保持复位前的内容不变。

20h~7Fh : Bank0

A0h~EFh : Bank1

§ 3.7.2 特殊寄存器

特殊功能寄存器被用以控制单片机 CPU 及功能部件的操作，所以一般把它们分成二类：有关 CPU 操作的在本节介绍，另外一类和特定功能部件有关的将在相应的章节介绍。下表列出了 PIC16C92X 各种型号的特殊功能寄存器以及它们在上电复位后和非上电复位后的值。

地 址	名 称	功 能 说 明	上电复位 值	其他复位 值
Bank0 (0 体)				
00h	INDF	间接寻址逻辑寄存器（物理上不存在）	0000 0000	0000 0000
01h	TMR0	TIMER0 模块寄存器	xxxx xxxx	uuuu uuuu
02h	PCL	程序计数器 PC 的低 8 位	0000 0000	0000 0000
03h	STATUS	状态寄存器	0001 1xxx	000q quuu
04h	FSR	间接寻址寄存器	xxxx xxxx	uuuu uuuu
05h	PORTA	— — — PORTA 口寄存器	(2)	(2)
06h	PORTB	PORTB 寄存器	xxxx xxxx	uuuu uuuu
07h	PORTC	PORTC 口寄存器	--xx xxxx	--uu uuuu
08h	PORTD	PORTD 口寄存器	0000 0000	0000 0000
09h	PORTE	— — — — — PORTE 寄存器	0000 0000	0000 0000
0Ah	PCLATH	— — — PC 高 5 位写入器	---0 0000	---0 0000
0Bh	INTCON	中断控制寄存器	0000 000X	0000 000u
0Ch	PIR1	某些中断标志位寄存器	00-- 0000	00-- 0000
0Dh	—	—	—	—
0Eh	TMR1L	TIMER1 模块寄存器低 8 位	xxxx xxxx	uuuu uuuu
0Fh	TMR1H	TIMER1 模块寄存器高 8 位	xxxx xxxx	uuuu uuuu
10h	T1CON	TIMER1 控制寄存器	--00 0000	--uu uuuu
11h	TMR2	TIMER2 模块寄存器	xxxx xxxx	uuuu uuuu
12h	T2CON	TIMER2 控制寄存器	-000 0000	-000 0000
13h	SSPBuF	同步串行口 (SSP) 发送/接收寄存器	xxxx xxxx	uuuu uuuu
14h	SSPCON	同步串行口 (SSP) 控制寄存器	0000 0000	0000 0000
15h	CCPR1L	CCP1 模块寄存器 (低 8 位)	xxxx xxxx	uuuu uuuu
16h	CCPR1H	CCP1 模块寄存器 (高 8 位)	xxxx xxxx	uuuu uuuu
17h	CCP1CON	CCP1 控制寄存器	--00 0000	--00 0000
18h~1Dh	—	—	—	—
1Eh (1)	ADRES	A/D 转换结果寄存器	xxxx xxxx	uuuu uuuu
1Fh	ADCON0	A/D 控制寄存器 0	0000 00-0	0000 00-0
Bank1 (1 体)				
80h	INDF	间接寻址逻辑寄存器（物理上不存在）	0000 0000	0000 0000
81h	OPTION	TIMER0 模块寄存器	xxxx xxxx	uuuu uuuu
82h	PCL	程序计数器 PC 的低 8 位	0000 0000	0000 0000
83h	STATUS	状态寄存器	0001 1xxx	000q quuu
84h	FSR	间接寻址寄存器	xxxx xxxx	uuuu uuuu
85h	TRISA	PORTA 口方向寄存器	--11 1111	--11 1111
86h	TRISB	PORTB 口方向寄存器	1111 1111	1111 1111
87h	TRISC	PORTC 口方向寄存器	--11 1111	--11 1111
88h	TRISD	PORTD 口方向寄存器	1111 1111	1111 1111
89h	TRISE	PORTE 方向寄存器及 SPI 状态寄存器	0000 -111	0000 -111
8Ah	PCLATH	— — — PC 高 5 位写入器	---0 0000	---0 0000
8Bh	INTCON	中断控制寄存器	0-00 000x	0-00 000u
8Ch	PIE1	某些中断允许位寄存器	00-- 0000	00-- 0000
8Dh	—	—	—	—
8Eh	PCON	上电复位 (POR) 标志位	---- --0-	---- --u-
8Fh~91h	—	—	—	—
92h	PR2	TIMER2 周期寄存器	1111 1111	1111 1111
93h	SSPADDD	同步串行口 (SSP) I2C 模式下之地址寄存器	0000 0000	0000 0000
94h	SSPSTAT	同步串行口 (SSP) 之状态寄存器	0000 0000	0000 0000
95h~9Eh	—	—	—	—
9Fh (1)	ADCON1	A/D 控制寄存器 1	---- -000	---- -000
Bank2				
100h	INDF	间接寻址逻辑寄存器（物理上不存在）	0000 0000	0000 0000
101h	TMR0	TIMER0 模块寄存器	xxxx xxxx	uuuu uuuu

102h	PCL	程序计数器 PC 的低 8 位								0000 0000	0000 0000
103h	STATUS	状态寄存器								0001 1xxx	000q uuuu
104h	FSR	间接寻址寄存器								xxxx xxxx	uuuu uuuu
105h	—	—								—	—
106h	PORTB	PORTB 数据寄存器								xxxx xxxx	uuuu uuuu
107h	PORTF	PORTC 数据寄存器								0000 0000	0000 0000
108h	PORTG	PORTD 数据寄存器								0000 0000	0000 0000
109h	—	—								—	—
10Ah	PCLATH	—	—	—	PC 高 5 位写入器					---0 0000	---0 0000
10Bh	INTCON	中断控制寄存器								0000 000x	0000 000u
10Ch	—	—								—	—
10Dh	LCDSE	SE29	SE27	SE20	SE16	SE12	SE9	SE5	SEG 0	1111 1111	1111 1111
10Eh	LCDPS	—	—	—	—	LP3	LP2	LP1	LP0	---- 0000	---- 0000
10Fh	LCDCON	LCD EN	SLP EN	—	VGE N	CS1	CS0	LMU X1	LMU X0	00-0 0000	00-0 0000
110h	LCDD00	SEG 07 COM 0	SEG 06 COM 0	SEG 05 COM 0	SEG 04 COM 0	SEG 03 COM 0	SEG 02 COM 0	SEG 01 COM 0	SEG 00 COM 0	xxxx xxxx	uuuu uuuu
111h	LCDD01	SEG 15 COM 0	SEG 14 COM 0	SEG 13 COM 0	SEG 12 COM 0	SEG 11 COM 0	SEG 10 COM 0	SEG 09 COM 0	SEG 08 COM 0	xxxx xxxx	uuuu uuuu
112h	LCDD02	SEG 23 COM 0	SEG 22 COM 0	SEG 21 COM 0	SEG 20 COM 0	SEG 19 COM 0	SEG 18 COM 0	SEG 17 COM 0	SEG 16 COM 0	xxxx xxxx	uuuu uuuu
113h	LCDD03	SEG 31 COM 0	SEG 30 COM 0	SEG 29 COM 0	SEG 28 COM 0	SEG 27 COM 0	SEG 26 COM 0	SEG 25 COM 0	SEG 24 COM 0	xxxx xxxx	uuuu uuuu
114h	LCDD04	SEG 07 COM 1	SEG 06 COM 1	SEG 05 COM 1	SEG 04 COM 1	SEG 03 COM 1	SEG 02 COM 1	SEG 01 COM 1	SEG 00 COM 1	xxxx xxxx	uuuu uuuu
115h	LCDD05	SEG 15 COM 1	SEG 14 COM 1	SEG 13 COM 1	SEG 12 COM 1	SEG 11 COM 1	SEG 10 COM 1	SEG 09 COM 1	SEG 08 COM 1	xxxx xxxx	uuuu uuuu
116h	LCDD06	SEG 23 COM 1	SEG 22 COM 1	SEG 21 COM 1	SEG 20 COM 1	SEG 19 COM 1	SEG 18 COM 1	SEG 17 COM 1	SEG 16 COM 1	xxxx xxxx	uuuu uuuu
117h	LCDD07	—	SEG 30 COM 1	SEG 29 COM 1	SEG 28 COM 1	SEG 27 COM 1	SEG 26 COM 1	SEG 25 COM 1	SEG 24 COM 1	xxxx xxxx	uuuu uuuu
118h	LCDD08	SEG 07 COM 2	SEG 06 COM 2	SEG 05 COM 2	SEG 04 COM 2	SEG 03 COM 2	SEG 02 COM 2	SEG 01 COM 2	SEG 00 COM 2	xxxx xxxx	uuuu uuuu
119h	LCDD09	SEG 15 COM 2	SEG 14 COM 2	SEG 13 COM 2	SEG 12 COM 2	SEG 11 COM 2	SEG 10 COM 2	SEG 09 COM 2	SEG 08 COM 2	xxxx xxxx	uuuu uuuu
11Ah	LCDD10	SEG 23 COM 2	SEG 22 COM 2	SEG 21 COM 2	SEG 20 COM 2	SEG 19 COM 2	SEG 18 COM 2	SEG 17 COM 2	SEG 16 COM 2	xxxx xxxx	uuuu uuuu
11Bh	LCDD11	—	—	SEG 29 COM 2	SEG 28 COM 2	SEG 27 COM 2	SEG 26 COM 2	SEG 25 COM 2	SEG 24 COM 2	xxxx xxxx	uuuu uuuu
11Ch	LCDD12	SEG 07 COM 3	SEG 06 COM 3	SEG 05 COM 3	SEG 04 COM 3	SEG 03 COM 3	SEG 02 COM 3	SEG 01 COM 3	SEG 00 COM 3	xxxx xxxx	uuuu uuuu
11Dh	LCDD13	SEG 15 COM 3	SEG 14 COM 3	SEG 13 COM 3	SEG 12 COM 3	SEG 11 COM 3	SEG 10 COM 3	SEG 09 COM 3	SEG 08 COM 3	xxxx xxxx	uuuu uuuu

11Eh	LCDD14	SEG 23 COM 3	SEG 22 COM 3	SEG 21 COM 3	SEG 20 COM 3	SEG 19 COM 3	SEG 18 COM 3	SEG 17 COM 3	SEG 16 COM 3	xxxx xxxx	uuuu uuuu
11Fh	LCDD15	—	—	—	SEG 28	SEG 27	SEG 26	SEG 25	SEG 24	xxxx xxxx	uuuu uuuu
Bank3											
180h	INDF	间接寻址逻辑寄存器（物理上不存在）								0000 0000	0000 0000
181h	OPTION	TIMER0 模块寄存器								1111 1111	1111 1111
182h	PCL	程序计数器 PC 的低 8 位								0000 0000	0000 0000
183h	STATUS	状态寄存器								0001 1xxx	000q quuu
184h	FSR	间接寻址寄存器								xxxx xxxx	uuuu uuuu
185h	—	——								—	—
186h	TRISB	PORTB 数据方向控制寄存器								1111 1111	1111 1111
187h	TRISF	PORTF 数据方向控制寄存器								1111 1111	1111 1111
188h	TRISG	PORTG 数据方向控制寄存器								1111 1111	1111 1111
189h	—	——								—	—
18Ah	PCLATH	—	—	—	PC 高 5 位写入器				---0 0000	---0 0000	
18Bh	INTCON	中断控制寄存器								0000 000X	0000 000u
18Ch~ 19Fh	—	——								—	—

注： X=不定， u=不变， q=取决于某条件， -=未用(读为 0)

(1) 这些寄存器仅 16C924 中有。

(2) 16C923：上电复位值为--XX XXXX，其他复位为--uu uuuu

16C924：上电复位值为--0X 0000，其他复位为--0u 0000

表 3.4 16C92X 特殊功能寄存器

下面分别叙述一些特殊寄存器的结构和功能。

一、状态寄存器 STATUS

和其他 PIC16CXXX 一样，参阅图 1.7，状态寄存器包含了 ALU 的算术状态，复位 RESET 状态及数据寄存器页面选择位。

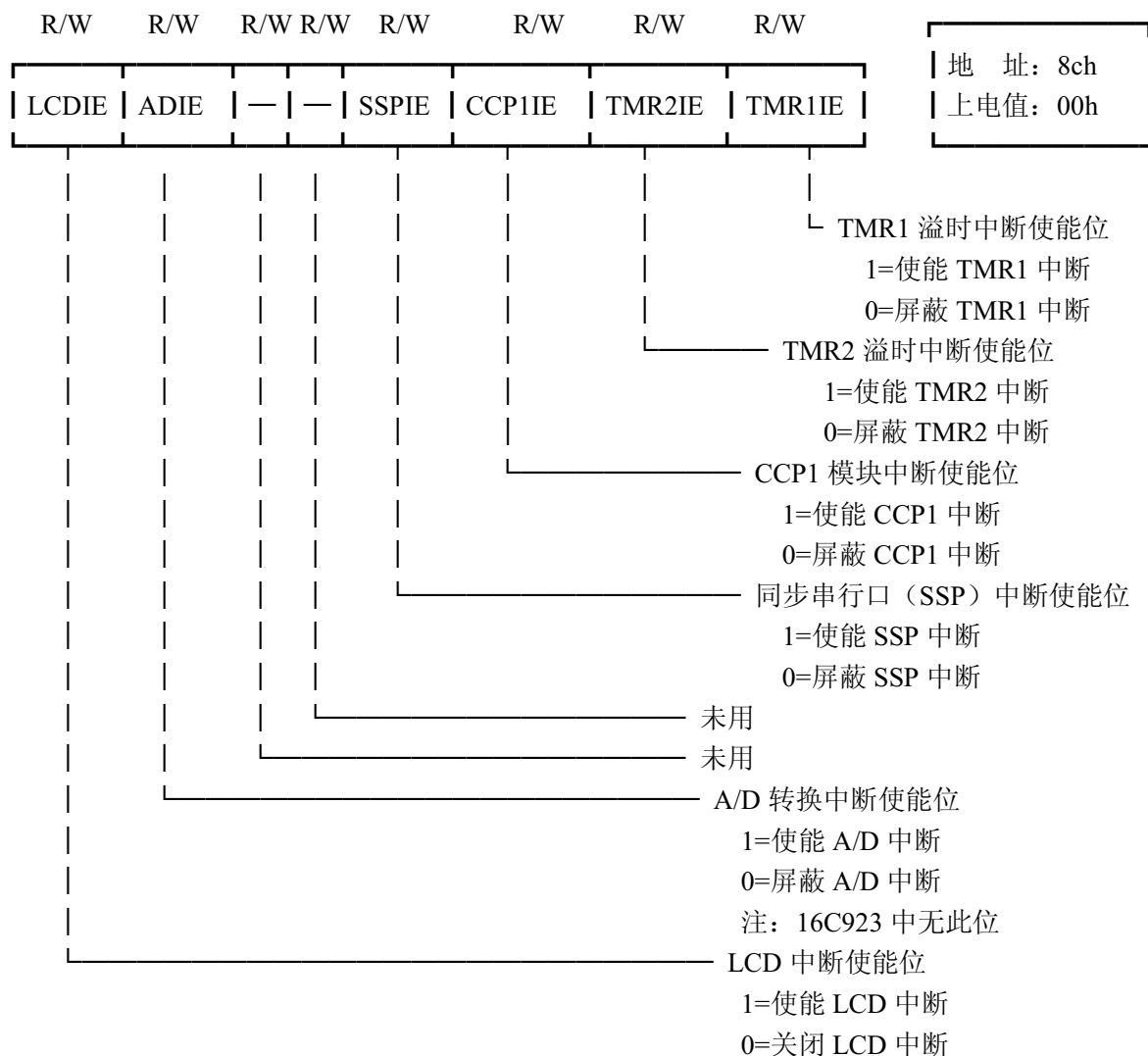
二、寄存器 OPTION

和其他 PIC16CXXX 一样，参阅图 1.8。

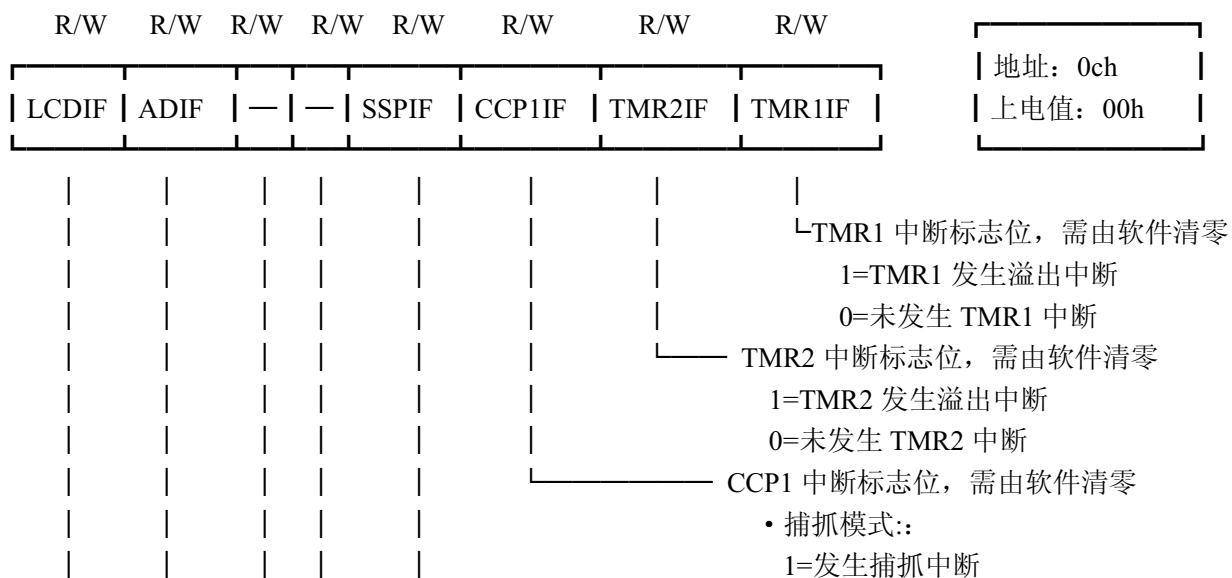
三、中断控制寄存器 INTCON

和 PIC16C62 等一样，参阅图 1.9。

四、寄存器 PIE1



五、寄存器 PIR1





地址：8Eh
上电值：00h

保留位
电源上电复位标志位
1=未发生上电复位
0=发生了上电复位。软件应及时把该位清为零以便将来用该位来判断是否发生了电源上电复位的情况。
未用

	MOVLW	0X20	
	MOVF	FSR	； 起始地址—FSR
NEXT	CLRF	INDF	； 清 FSR 内容所指的单元（20H-2FH）
	INCF	FSR	； FSR 内容增 1

```

    BTFSS      FSR, 4      ; 到 2FH; 否
    GOTO       NEXT       ; 循环
GOON:  ...              ; 完成清零

```

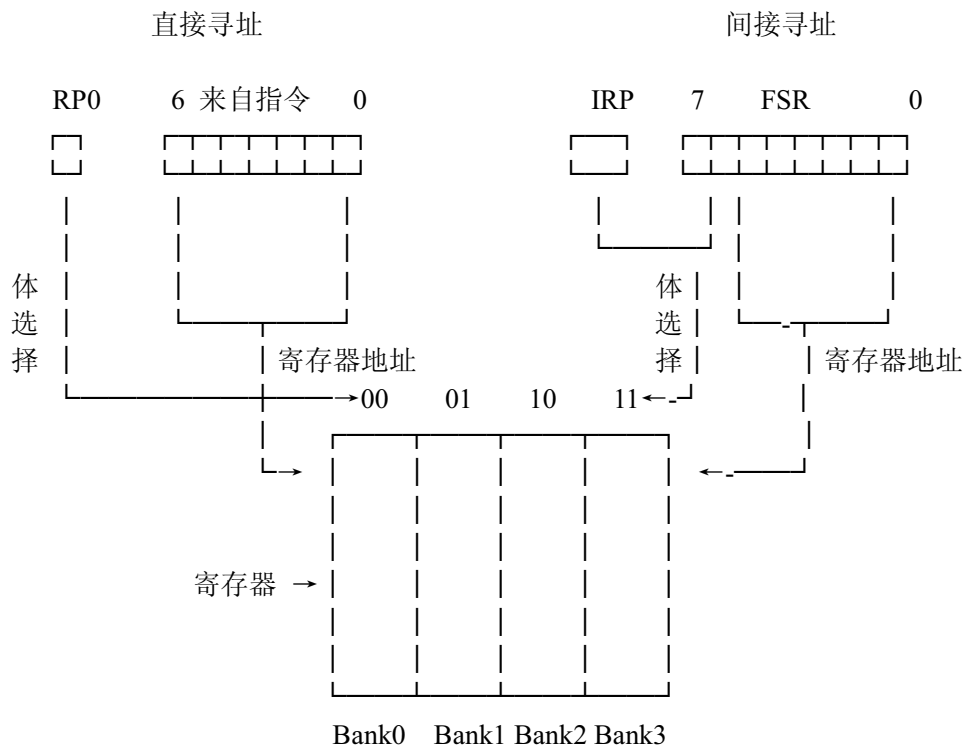


图 3.8 直接或间接寻址方式图

注意：关于寄存器体（Bank）选择，在直接寻址和间接寻址两种方式下，方法是不同的，见上图。

A.直接寻址

此时 Bank 由 STATUS 寄存器中的 RP0:RP1 来选择，见 STATUS 寄存器的描述。

B.间接寻址

此时 Bank 是由间接寻址寄存器 FSR<7>和状态寄存器 STATUS<7>两个位来选择。

IRP	FSR<7>	寄存器体
0	0	Bank0
0	1	Bank1
1	0	Bank2
1	1	Bank3

例程：用间接寻址方式往寄存器体 Bank1 中的 A0h-A5h 送数据 88h。

```

    MOVLW      0XA0
    MOVWF      FSR
    BCF        STATUS, 7
LOOP:  MOVLW      88h
    MOVWF      INDF
    MOVLW      0XA5
    XORWF      FSR, 0
    BTFSS      STATUS, Z
    GOTO       LOOP
    ...
    ...

```

PIC16C92X 有 7 个 I/O 口，这些 I/O 管脚有的和某些外部功能部件复用，即可以作为一般的 I/O 引脚，也可以作为某些功能的输入/输出。PIC16C92X 把 I/O 口都作为寄存器来处理，编程非常方便。

§ 3.8.1 PORTA 和 TRISA

PIC16C92X 的 PORTA 是 6 位宽的 I/O 口，和 PIC16C74 等完全一样，参阅 § 2.8。

§ 3.8.2 PORTB 和 TRISB

PORTB 是一个 8 位，双向可编程的 I/O 口，和其他 PIC16CXXX 完全一样，参阅 § 1.8.2。

§ 3.8.3 PORTC 和 TRISC

PORTC 是一个 6 位双向 I/O 口，和其他如 PIC16C64 等完全一样，参阅 § 1.8.3。
PORTC 的 I/O 脚还可以作为其他一些外部功能的引脚，这些将在相应的章节介绍。

§ 3.8.4 PORTD 和 TRISD

PORTD 是一个具有斯密特输入缓冲的 I/O 口，其中 RD0~RD4 可作为普通的 I/O 口线，也可以作 LCD 的 Seg 驱动；而 RD5~RD7 可作为普通的单向输入口线，也可以作为 LCD 的 Seg 驱动或 Com 驱动。当 PORTD 作为普通数字 I/O 口时，TRISD 控制其 RD0~RD4 的输入/输出方向。

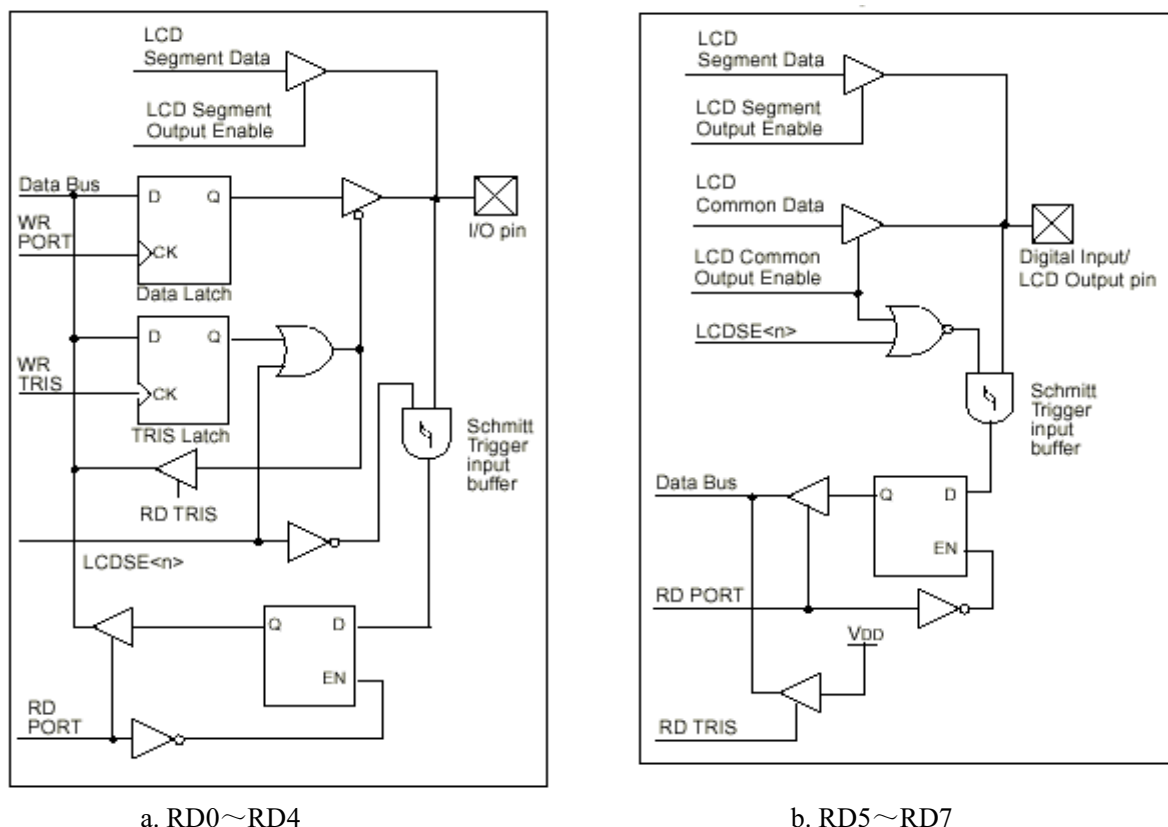


图 3.9 PORTD 结构

注意，芯片上电复位后，RD0~RD4 是自动初始化作为 LCD 的 Seg 驱动（Seg00~Seg04）。如果要定义其为数字口，必须把 LCDSE 寄存器中相应的位清零（参阅 § 1.13 有关 LCD 的叙述）。

寄存器	功 能	地 址	上电复位值
PORTD	I/O 口电平状态	08h	0000 0000
TRISD	RD0~RD5 方向控制	88h	1111 1111

LCDSE	LCD 驱动控制	10Dh	xxxx xxxx
-------	----------	------	-----------

表 3.5 PORTD 相关寄存器

§ 3.8.5 PORTE 和 TRISE

PORTE 是一个数字输入口，也可以作为 LCD 的 Seg 驱动。它们有斯密特触发缓冲。

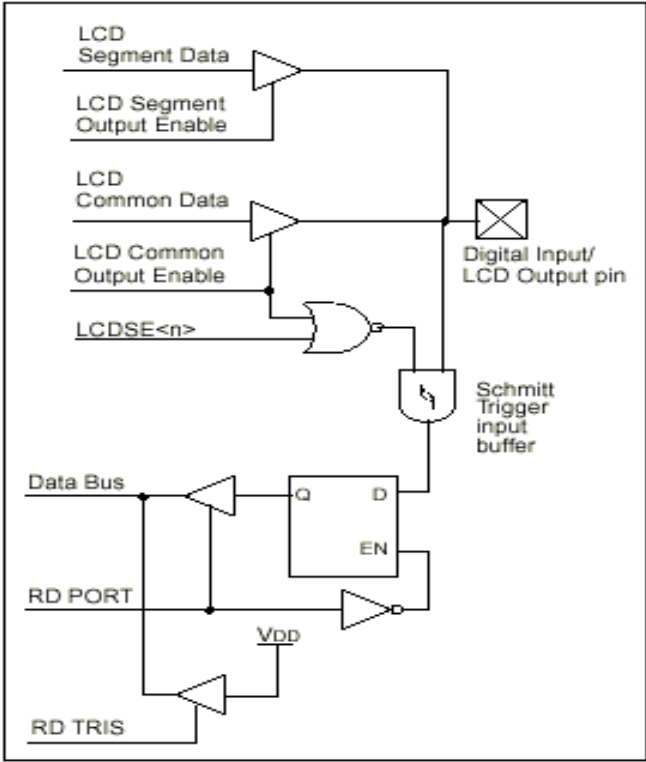


图 3.10 PORTE 结构图

注意，芯片上电复位后，PORTE 是被自动定义为 LCD 的 Seg 驱动。如果要使其成为数字输入口，须将 LCDSE 中相应的位清零。

例：

```
BCF    STATUS, RP0           ; 选 Bank2
BSF    STATUS, RP1
BCF    LCDSE, SE27           ; 使 RE<0:7>及 RG<7>
BCF    LCDSE, SE5            ; 为数字输入
BCF    LCDSE, SE9
```

寄存器	功 能	地 址	上电复位值
PORTE	I/O 口电平状态	09h	0000 0000
TRISE	—	89h	1111 1111
LCDSE	LCD 驱动控制	10Dh	1111 1111

表 3.6 PORTE 相关寄存器

§ 3.8.6 PORTF 和 TRISF

PORTF 是一个数字输入口，也可以作为 LCD 的 Seg 驱动。它们有斯密特触发缓冲。

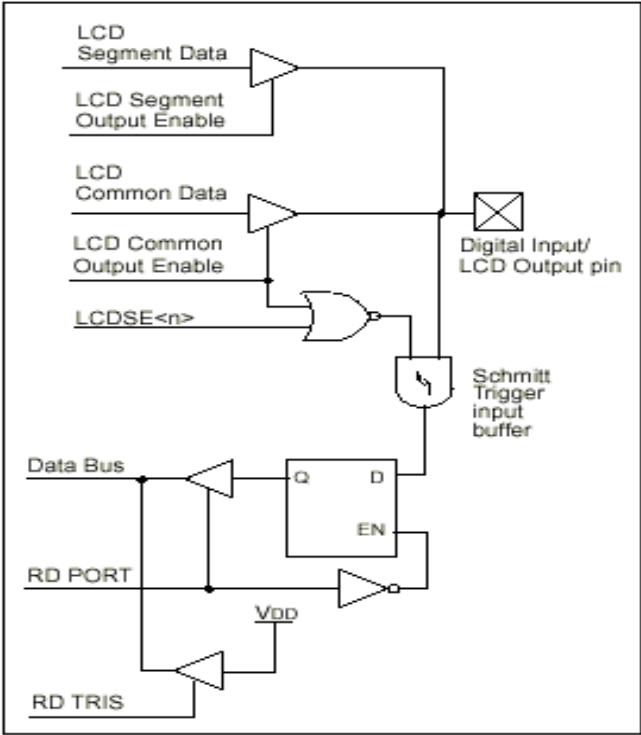


图 3.11 PORTF 结构图

注意，芯片上电复位后，PORTF 是被自动定义为 LCD 的 Seg 驱动。如果要使其成为数字输入口，须将 LCDSE 中相应的位清零。

例：

```
BCF STATUS, RP0           ; 选 Bank2
BSF STATUS, RP1
BCF LCDSE, SE16           ; 使 RF<0:7>
BCF LCDSE, SE12           ; 为数字输入
```

寄存器	功 能	地 址	上电复位值
PORTF	I/O 口电平状态	107h	0000 0000
TRISF	——	187h	1111 1111
LCDSE	LCD 驱动控制	10Dh	1111 1111

表 3.7 PORTF 相关寄存器

§ 3.8.7 PORTG 和 TRISG

PORTG 是一个数字输入/输出口，也可以作为 LCD 的 Seg 驱动。它们有施密特触发缓冲。

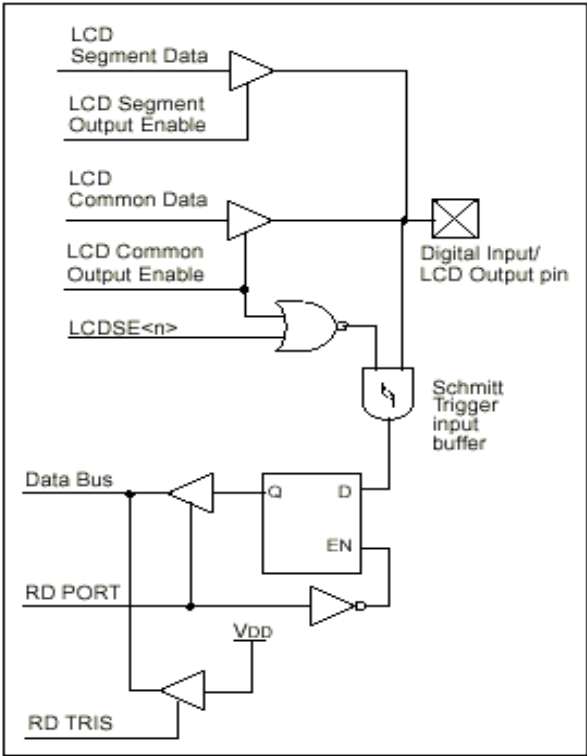


图 3.12 PORTG 结构图

注意，芯片上电复位后，PORTG 是被自动定义为 LCD 的 Seg 驱动。如果要使其成为数字输入/输出口，须将 LCDSE 中相应的位清零。

例：

```
BCF STATUS, RP0          ; 选 Bank2
BSF STATUS, RP1
BCF LCDSE, SE27           ; 使 RG<0:7>及 RE<7>
BCF LCDSE, SE20           ; 为数字输入
```

寄存器	功 能	地 址	上电复位值
PORTG	I/O 口电平状态	108h	0000 0000
TRISG	——	188h	1111 1111
LCDSE	LCD 驱动控制	10Dh	1111 1111

表 3.8 PORTG 相关寄存器

§ 3.9 定时器/计数器

PIC16C9XX 有三个定时器 TMR0、TMR1、TMR2，每个定时器可以产生中断请求，另外定时器也和一些别的模块配合来完成捕捉/PWM 等功能，这部分功能和 PIC16C74 等完全一样，请参阅 § 2.9。

§ 3.10 CCP 模块

PIC16C92X 的 CCP 模块和 PIC16C74 等完全一样，请参阅 § 2.11。

§ 3.11 同步串行口（SSP）模块

PIC16C92X 同步串行口模块（以下简称 SSP）和 PIC16C74 等完全一样，请参阅 § 2.12。

§ 3.12 A/D 转换

目前只有 PIC16C924 带 A/D 转换功能，它和 PIC16C73 的 A/D 转换完全一样，请参阅 § 2.14 中有关 PIC16C73 A/D 的内容。

§ 3.13 LCD 模块

PIC16C9XX 具有 LCD 驱动功能。芯片内的 LCD 模块用来产生时序波形以控制静态或动态复合 LCD 显示屏。它可以支持最多达 32 条 Seg 线和 4 条 Com 线，并且可以控制 LCD 像素。

芯片内由三个控制寄存器 (LCDCON、LCDSE 及 LCDPS) 控制时序输出，由 16 个 LCD 数据寄存器来控制 LCD 像素。在通常的操作中，控制寄存器用来控制 LCD 显示屏，初始化工作包括定义 LCD 之 Com 线的数量以及定义 LCD 相位时钟。一旦定义完成，LCD 的数据寄存器的数据即体现到 LCD 显示屏上像素的亮或灭，每个位对应一个像素，“1”为亮，“0”为灭。

一旦 LCD 模块被定义后，LCDEN 位 (LCDCON<7>) 即用来控制使能或关闭 LCD 模块。如果置 SLPEN (LCDCON<6>) = 0，则 LCD 模块在睡眠 (Sleep) 下依然可以工作。

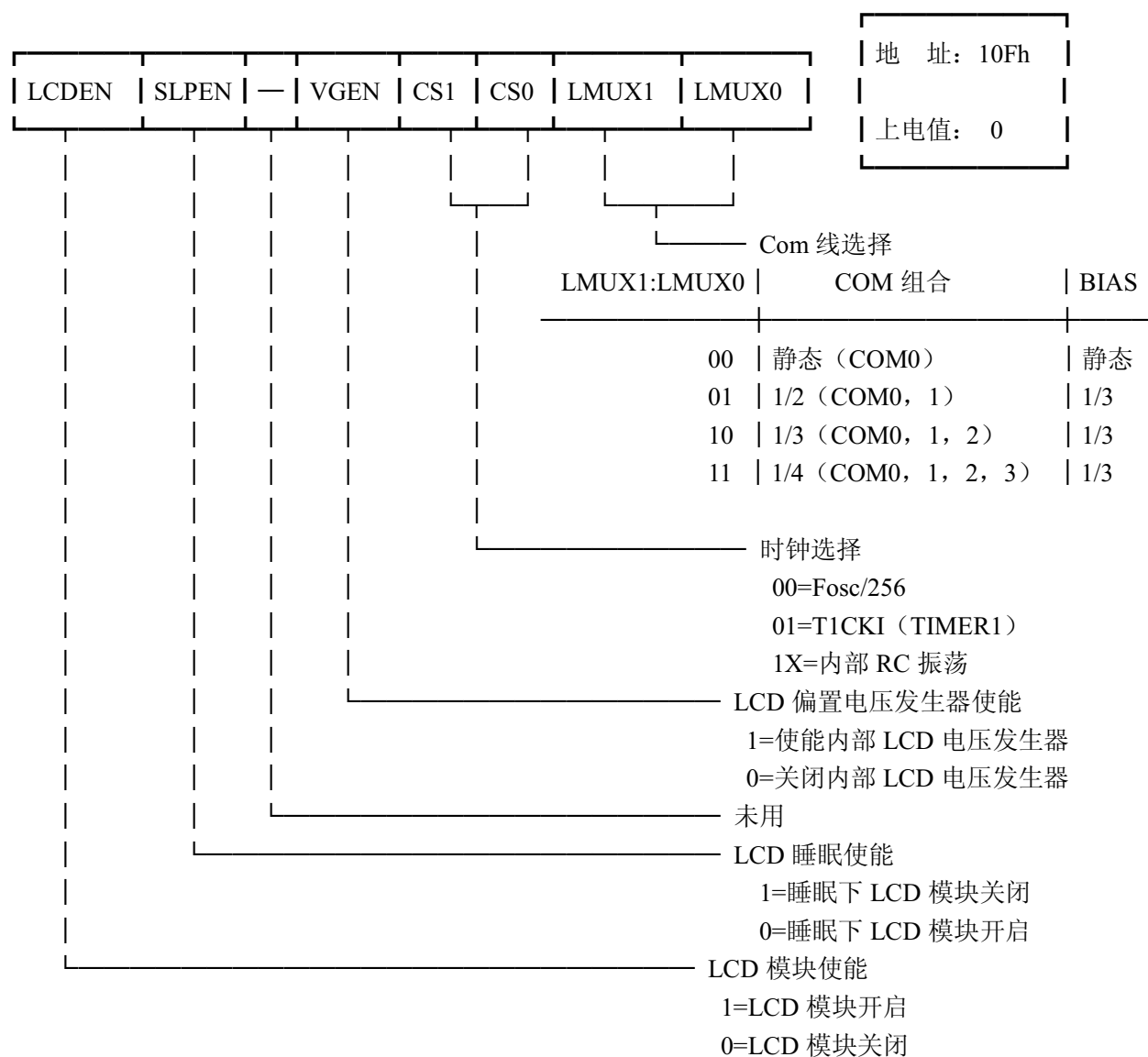


图 3.13 LCD COM 寄存器

LCD 模块如下图所示:

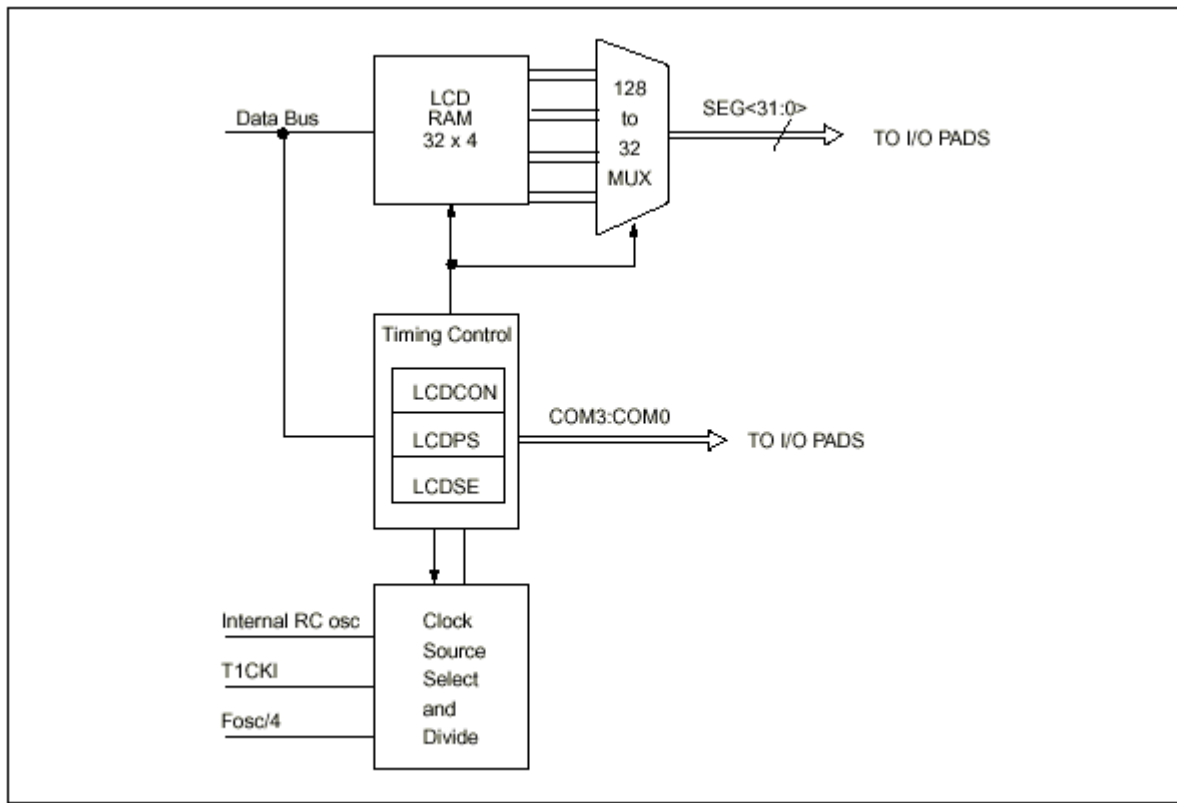


图 3.14 LCD 模块

§ 3.13.1 LCD 时序

PIC16C9XX 的 LCD 模块可以有 3 种时钟源，并且支持静态、1/2、1/3 和 1/4 的多路组合。

一、时钟源选择

三种时钟源：

1. 内部 RC 振荡 (14KHZ)
2. TIMER1
3. Fosc/256

第 1 种时钟频率较低，一般用于在睡眠中仍需 LCD 工作的场合，因为比较省电。如果不选用这个内部 RC 振荡或 LCD 模块不启用，它即自动关闭以节省功耗。

第 2 种是使用 TIMER1 的外部振荡，一般为 32KHZ，接在 T1OSO 和 T1OSI 之间。它也是较低频率的振荡，也是针对睡眠中的 LCD 而设计。

第 3 种是系统时钟 256 分频后的信号源，当系统 Fosc=8M 时，它约为 32KHZ。

时钟源由 LCDCON<3:2>选择，参见图 1.51。

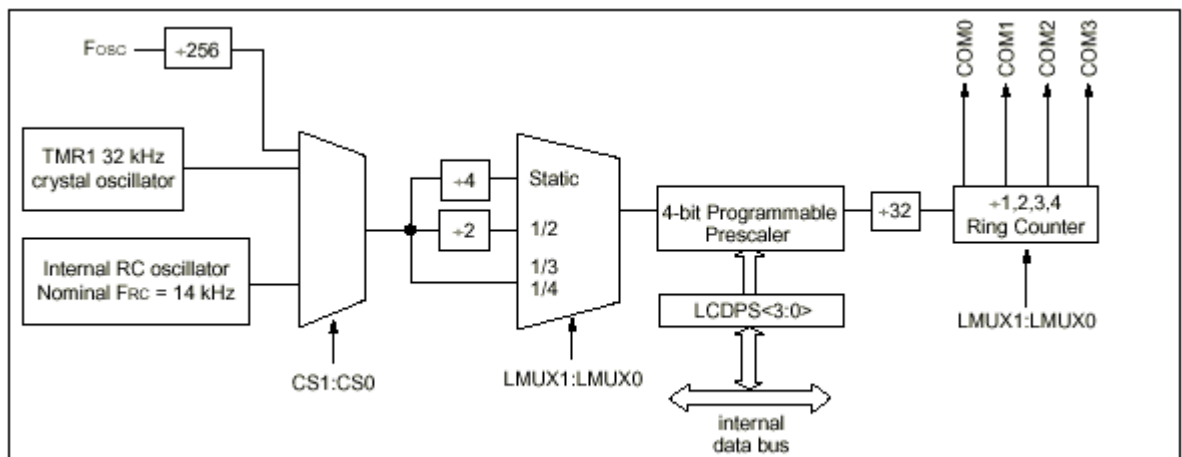


图 3.15 LCD 时钟电路

LCD 帧频率的计算方式如下表所示：

LMUX1:LMUX0	帧 频 率
00	源时钟频率/（128*（LP3:LP0+1））
01	源时钟频率/（128*（LP3:LP0+1））
10	源时钟频率/（96*（LP3:LP0+1））
11	源时钟频率/（96*（LP3:LP0+1））

表 3.9 LCD 帧频率计算式

其中 LP3:LP0 为预分频数，见下图：

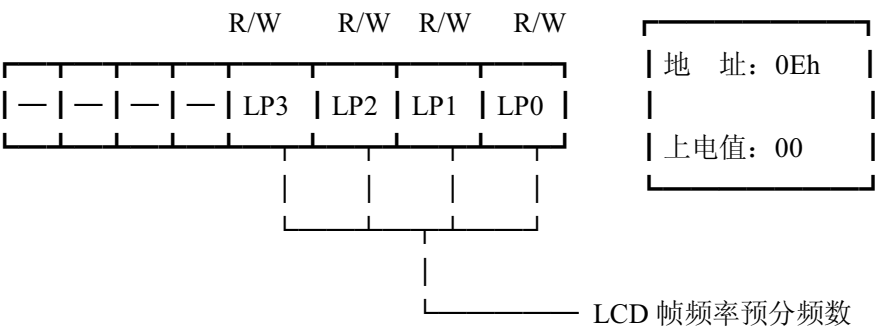


图 3.16 LCDPS 寄存器

二、多路时序产生电路

时序产生电路根据 LCD 显示方式共可产生 1~4 路 Com 多路时钟（COM0: COM3）信号，由 LMUX1:LMUX0（LCDCON<1:0>）定义，见图 3.13 所示。

下表列出各种情形下的 LCD 帧频率（计算式见表 1.28）。

LP3:LP0	静态	1/2	1/3	1/4
2	85	85	114	85
3	64	64	85	64
4	51	51	68	51
5	43	43	57	43
6	37	37	49	37
7	32	32	43	32

a. TIMER1 为时钟源
Fosc=8MHZ

LP3:LP0	静态	1/2	1/3	1/4
0	109	109	146	109
1	55	55	73	55
2	36	36	49	36
3	27	27	36	27

b. 内部 RC 为时钟源
约为 14KHZ

表 3.10 各种情形下的 LCD 帧频率（HZ）

下图是一个例子：

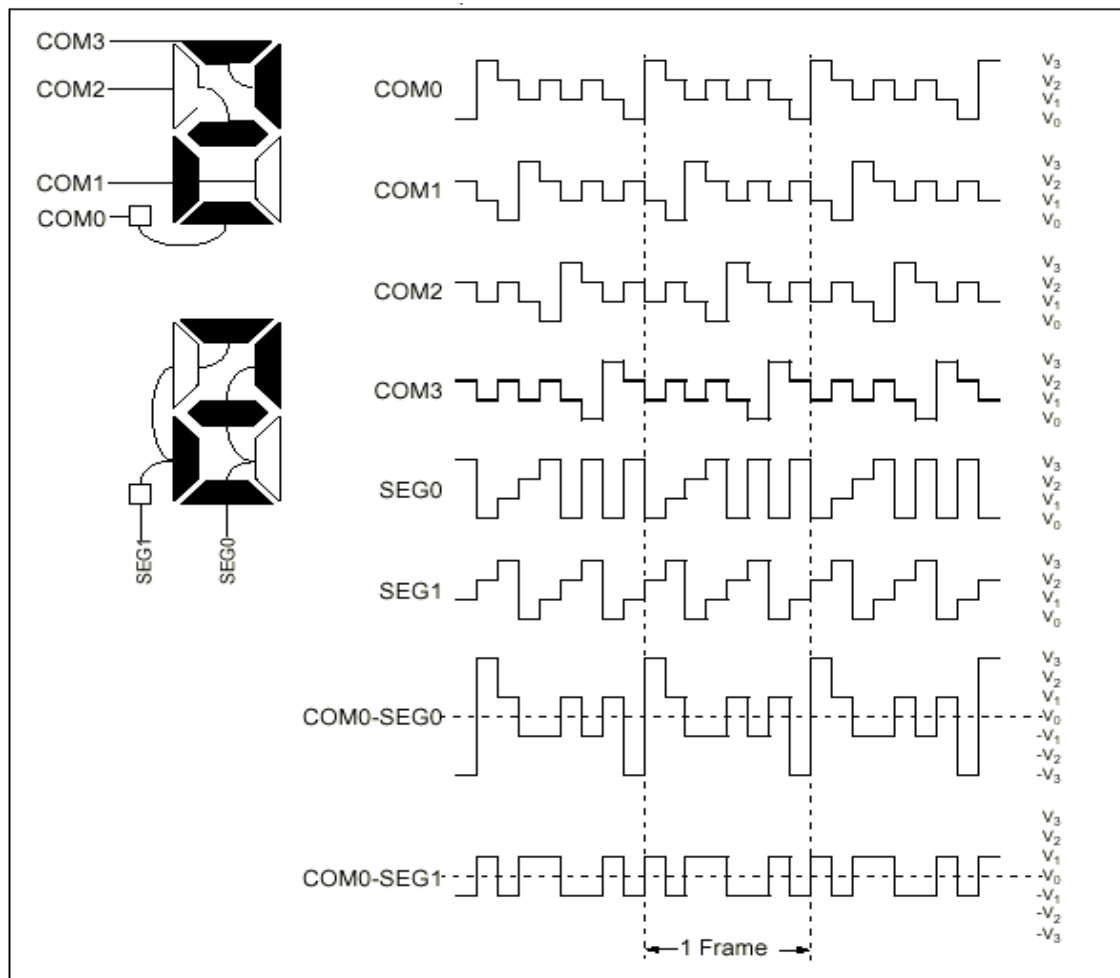


图 3.17 LCD 1/4 多路驱动显示的波形

§ 3.13.2 LCD 中断

LCD 时序发生电路提供一种中断，它发生在一帧时序波形产生过程中的某一个固定的时刻，主要是用来在一帧新的 LCD 波形产生之前，写入相应的像素数据，这样可以获得较好的 LCD 图像，见下图：

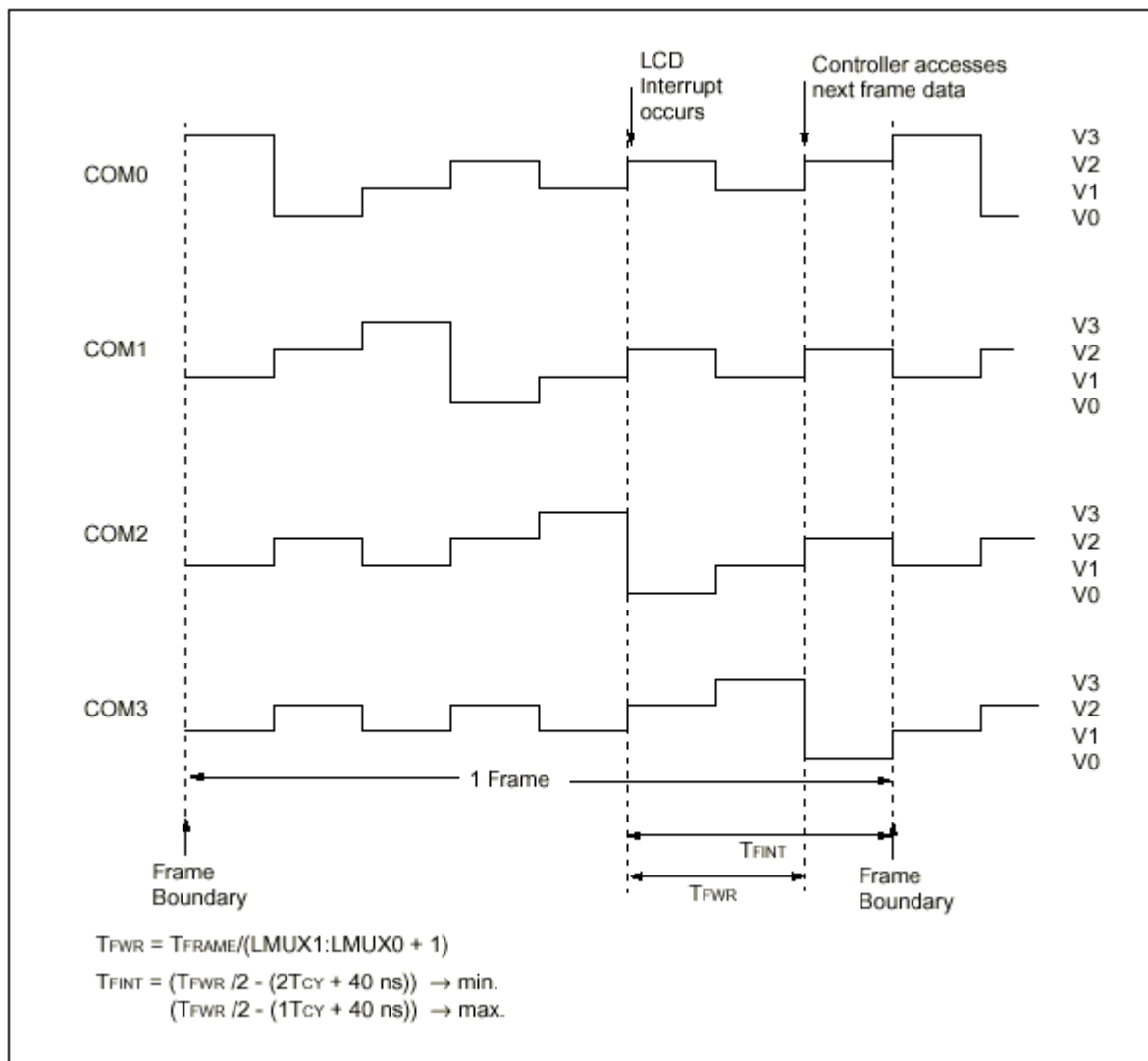


图 3.18 1/4 多路驱动

如上图所示，一旦 LCD 控制模块完成当前帧数据的读取，则发生中断请求，以便 CPU 在 TFWR 时间内为下一帧图像置入数据。

§ 3.13.3 像素控制

像素寄存器中的位控制每个像素的状态（亮或暗），每位控制一个相应的像素，见下图：

7	6	5	4	3	2	1	0
SEGS	—	—	—	—	—	SEGS	SEGS
COMC						COMC	COMC

SEGS:COMC=对应于 Seg(S):Com(C)的像素位

1= 亮

0= 暗

图 3.19 像素寄存器

§ 3.13.4 睡眠中的 LCD

PIC16C9XX 的 LCD 模块当芯片处于睡眠中时仍然可以继续工作，见图 3.13 LCDCON 寄存器的描述。

当 SLPEN (LCDCON<6>) =1, 则睡眠时 LCD 模块所有功能关闭, Seg 和 Com 上的电压达到最低, 见下图所示。

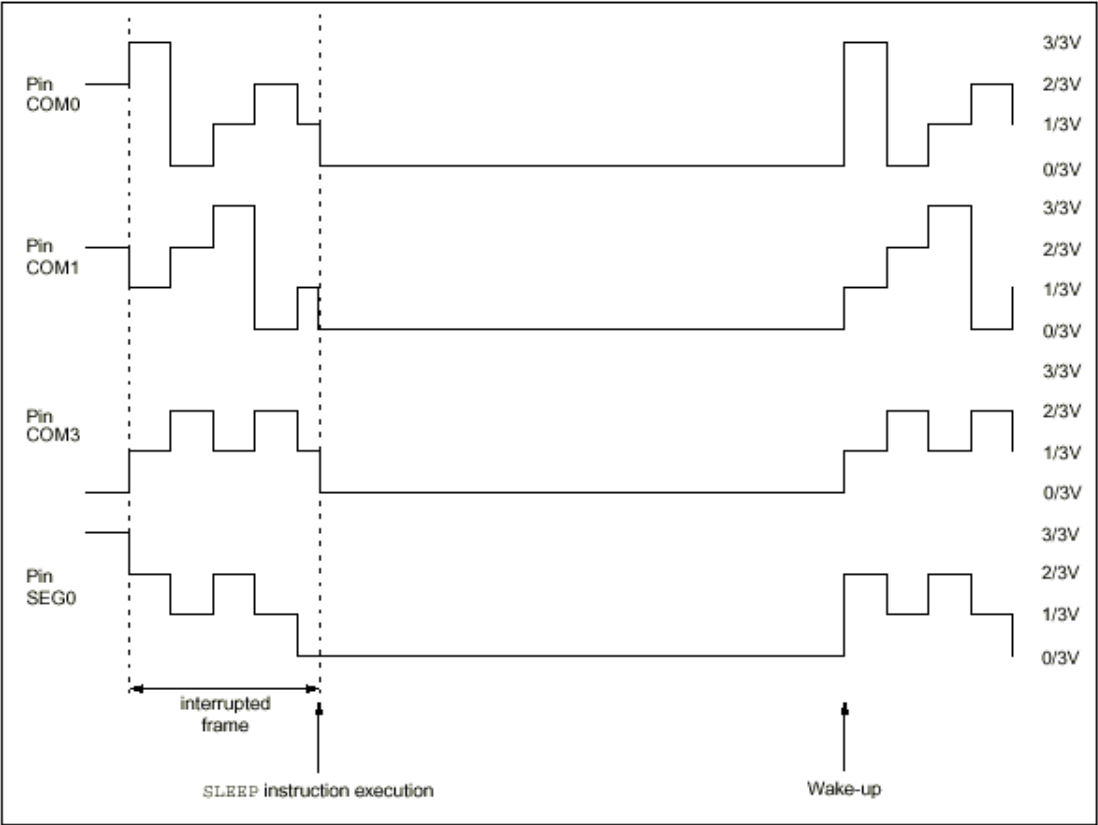


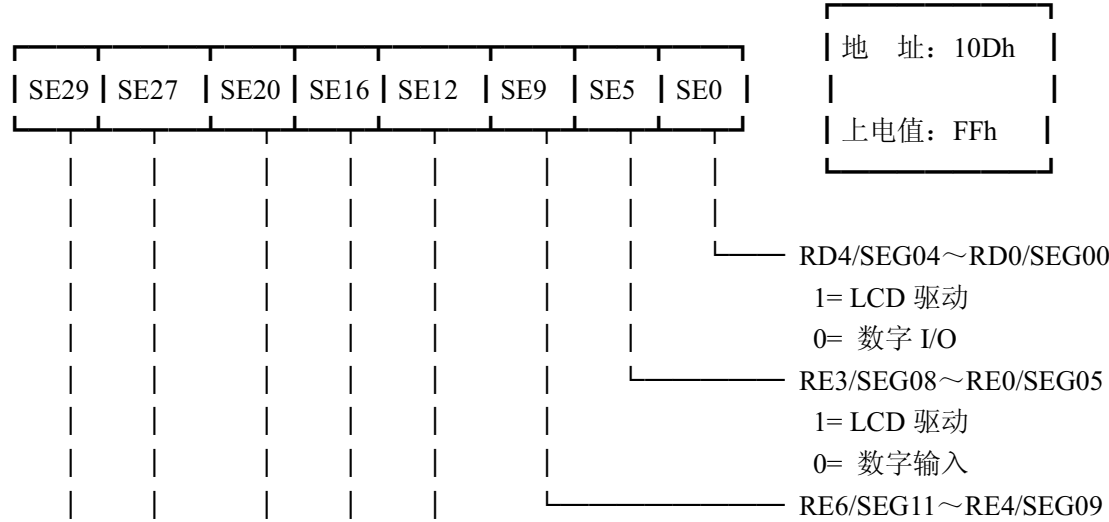
图 3.20 睡眠中的 LCD (SLPEN=1)

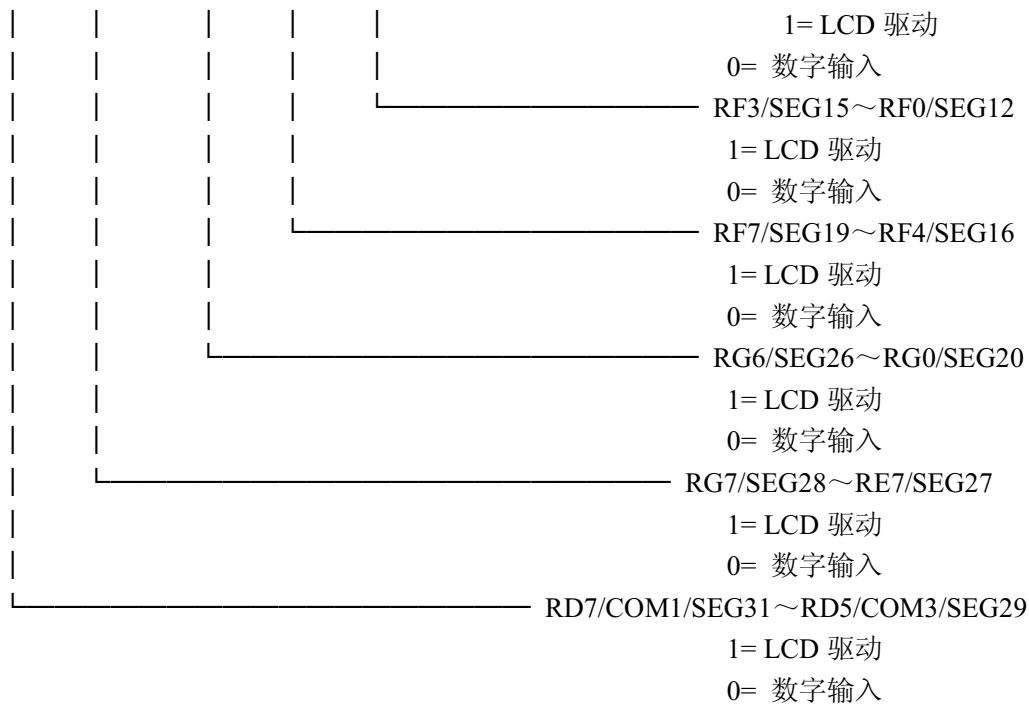
注意, 如果要保证 LCD 完成一帧后再进入睡眠, 必须在 LCD 帧波形的结束边沿执行“Sleep”指令, LCD 中断可以用来计算帧结束边沿, 详参见 § 1.13.2 及图 3.17。

当 SLPEN=0, 则睡眠中 LCD 模块显示当前 LCD 像素寄存器中的内容。为了达到这点, LCD 时钟源必须是内部 RC 振荡或 TIMER1 振荡。在睡眠中, LCD 图像不会变化 (CPU 已停止工作, 像素寄存器的值保持不变)。就 LCD 模块而言, 它会继续消耗电流, 但芯片别的部件已停止工作, 芯片总体功耗还是降低了。

§ 3.13.5 Seg 使能

PIC16C9XX 中有一些 I/O 口线即可作为通用数字口线, 也可以用作 LCD 的 Seg 驱动输出, 它们的选择定义由 LCDSE 寄存器来选择, 见下图:





注: LMUX1:LMUX0 优先开 LCDSE。

图 3.21 LCDSE 寄存器

如果为 I/O 口，则其方向由相应的 TRIS 寄存器控制，但 LCDSE 寄存器的控制优先级高于 TRIS 寄存器，即一旦定义为 LCD 驱动输出，那么它将不理睬 TRIS 中相应位的值是什么。

例：

```
a.      BCF      STATUS, RP0
          BSF      STATUS, RP1          ; 选 Bank2
          BCF      LCDCON, LMUX1       ; 选静态驱动
          BCF      LCDCON, LMUX0
          MOVLW    0XFF                 ; PORTD, E, F, G
          MOVWF    LCDSE                ; 为 LCD 驱动输出

b.      BCF      STATUS, RP0
          BSF      STATUS, RP1          ; 选 Bank2
          BSF      LCDCON, LMUX1       ; 选 1/3 多路驱动
          BCF      LCDCON, LMUX0
          MOVLW    0X87                 ; PORTD<7:0>, PORTE<6:0>
          MOVWF    LCDSE                ; 为 LCD 驱动输出
```

§ 3.13.6 偏置电压发生器

PIC16C92X 提供二种 LCD 偏置电压产生方式:内部的充电泵或外部电阻阶梯。

一、内部充电泵

内部充电泵电路如下图所示：

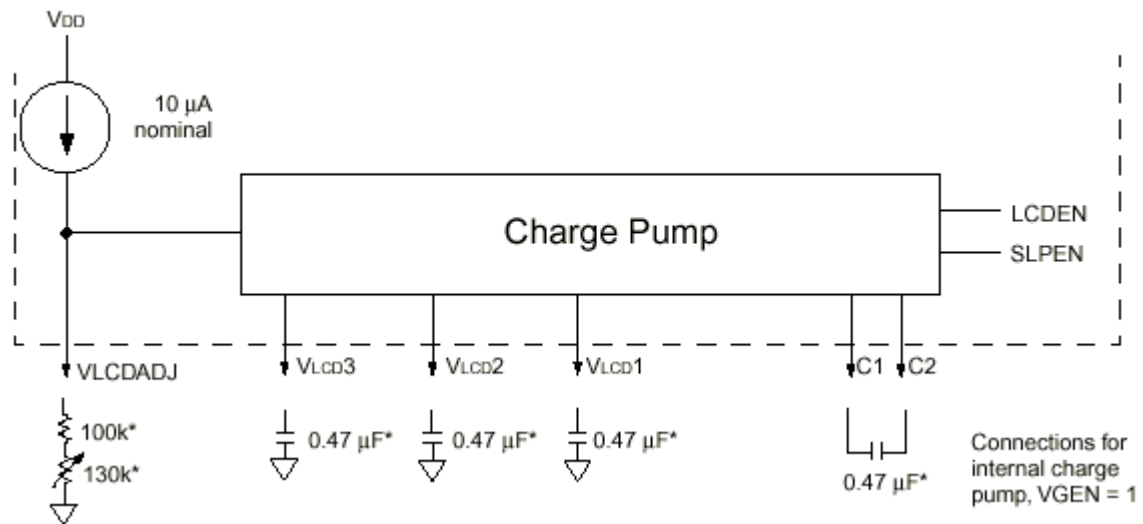


图 3.22 LCD 充电泵电路

其中 1.0V~2.3V 的调压器可以建立起稳定的基准电压源。调压器可通过接在 VLCDADJ 脚上的可调电阻来实现调节。基准电压加在充电泵的 VLCD1 上，充电泵使 VLCD2=2VLCD1，VLCD3=3VLCD1。

当充电泵不工作时，VLCD3 被直接联到 VDD。

二、外部电阻阶梯

LCD 模块也可以使用外部电阻阶梯电路来产生 LCD 偏置电压，如下图所示：

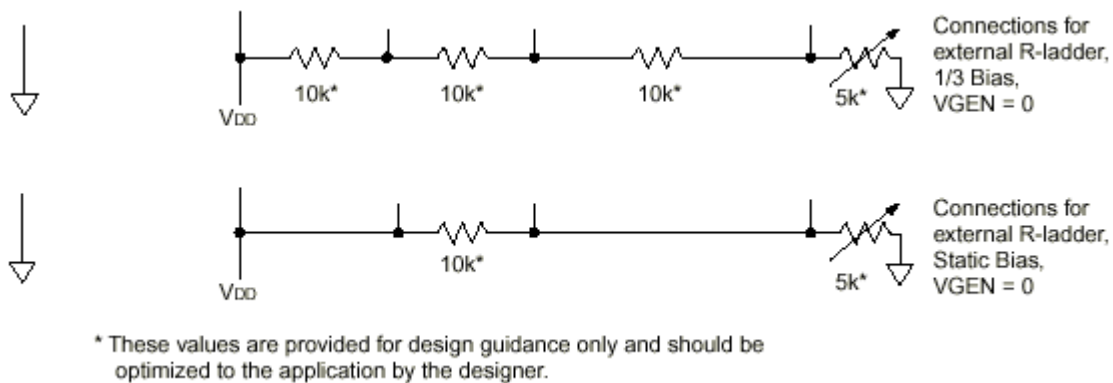


图 3.23 外部电阻阶梯

使用外部电阻阶梯时须置 VGEN (LCDCON<4>) = 0。

§ 3.13.7 LCD 模块的定义设置

用户一般可以遵循以下步骤来定义 LCD 模块的工作方式：

1. 定义 LCD 时钟预分频率：设置 LP3:LP0 (LCDPS<3:0>)。
2. 定义所需的 Seg 驱动：设置 LCDSE 寄存器。
3. 定义驱动模式和 Bias：设置 LMUX1:LMUX0 位。
4. 定义时钟源：设置 CS1:CS0 位。
5. 定义偏置电压源：设置 VGEN 位。
6. 定义睡眠下 LCD 模式：设置 SLPEN 位。
7. 置 LCD 像素数据：设置 LCD00~LCD15。
8. 清 LCD 中断标志位：LCDIF (PIR1<7>) = 0。
9. 如果需 LCD 中断：置 LCDIE (PIE1<7>) = 1。
10. 启动 LCD 模块：置 LCDEN (LCDCON<7>) = 1。

§ 3.14 CPU 特性

PIC16C9XX 单片机集成了一系列优秀微控制器的特性。

§ 3.14.1 系统定义字 (Configuration)

在 PIC16CXX 中有一个 13 位长的字节, 内含系统定义位, 用来定义单片机的一些系统性能, 和其他的 PIC16CXX 一样, 参阅 § 1.13.1。

§ 3.14.2 振 荡

一、振荡类型

PIC16C9XX 可以运行在 4 种类型的振荡方式:

1. LP — 低频晶体振荡
2. XT — 标准晶体/陶瓷振荡
3. HS — 高速晶体/陶瓷振荡
4. RC — 阻容振荡

和其他的 PIC16CXX 一样, 请参阅 § 1.13.2。

§ 3.14.3 复 位

PIC16C9XX 片内都集成有“上电复位”电路 (POR), 和其他的 PIC16CXX 一样, 请参阅 § 1.13.3。
PIC16C92X 各种复位后特殊功能寄存器的值或变化情况如下表所示:

复 位 / 唤 醒	程序计数器	状态寄存器	PCON 寄存器
上电复位	000h	0001 1xxx	---- --0-
正常运行时 MCLR 拉低复位	000h	000u uuuu	---- --u-
睡眠时 MCLR 拉低复位	000h	0001 0uuu	---- --u-
看门狗溢时复位	000h	0000 1uuu	---- --u-
睡眠时看门狗溢时唤醒	PC+1	uuu0 0uuu	---- --u-
睡眠时中断唤醒	PC+1(1)	uuu1 0uuu	---- --u-

注 1: 如果 GIE=1, 则 PC=0004h (复位向量)

表 3.11 复位后各特殊寄存器的值

§ 3.14.4 中 断

PIC16C92X 有多种中断源, 见下表:

中 断 源	标志位	使能位	923	924
外部触发中断 INT	INTF	INTE	√	√
TMR0 溢出中断	T0IF	T0IE	√	√
PORTB<7:4>中断	RBIF	RBIE	√	√
TMR1 中断	TMR1IF	TMR1IE	√	√
TMR2 中断	TMR2IF	TMR2IE	√	√
CCP1 中断	CCP1IF	CCP1IE	√	√
A/D 中断	ADIF	ADIE	无	√
SSP 中断	SSPIF	SSPIE	√	√
LCD 中断	LCDIF	LCDIE	√	√

表 3.12 PIC16C92X 中断源

除了 LCD 中断外，其余中断和 PIC16CXX 有关中断完全一样，请参阅 § 1.13.4。

§ 3.14.5 看门狗 (WDT)

和其他 PIC16CXX 完全一样，请参阅 § 1.13.5。

§ 3.14.6 睡眠 (SLEEP)

和其他 PIC16CXX 完全一样，请参阅 § 1.13.6。

§ 3.14.7 程序保密位 (Protect Fuse)

和其他 PIC16CXX 完全一样，请参阅 § 1.13.7。

§ 3.14.8 用户识别码 (ID Code)

和其他 PIC16CXX 完全一样，请参阅 § 1.13.8。

第四章 PIC16C8X 单片机

PIC16C8X 是带 E²PROM 部件的型号，目前有下列表中几种型号：

型 号	振荡	E2PROM 程序区	ROM 程序区	E2PRO M 数据 区	RAM	电压	中断 源	I/O	定时 器	复位 锁定	看门 狗	封装
16C83	DC-10M	0.5K×14	无	64×8	36	2.0-6.0	4	13	1	无	有	18
16C84	DC-10M	1K×14	无	64×8	36	2.0-6.0	4	13	1	无	有	18
16CR83	DC-10M	无	512×14	64×8	36	2.0-6.0	4	13	1	无	有	18
16CR84	DC-10M	无	1K×14	64×8	36	2.0-6.0	4	13	1	无	有	18

表 4.1 PIC16C8X 的型号功能表

PIC16C8X 没有 CCP、SPI、SCI 及并行口等功能模块，它是一种小型的易于嵌入应用的单片机。

注意 PIC16CR84 的程序区是掩膜 ROM，但数据区中仍带有 64 字节（8 位）的 E²PROM。这种型号是 PIC16C84 掩膜型，所以下面我们仅讲述 PIC16C83/84。

§ 4.1 主要功能特点

一、高性能 RISC 结构 CPU

- 精简指令集，仅 35 条单字节指令。
- 除地址分支指令外，其余全为单周期指令。
- 执行速度：DC-400ns。
- 14 位 E²PROM 型程序存储器，电可重擦写。
- 64 个 8 位 E²PROM 型数据寄存器。擦写次数达 100 万次，保存时间大于 40 年。
- 八级硬件堆栈。
- 多种硬件中断。
- 直接/间接/相对三种寻址方式。

二、功能部件

- 13 根可独立编程双向 I/O 口线。
- 高驱动电流 I/O 脚，可直接驱动 LED 显示。
 - 每根 I/O 口线最大拉电流 25mA
 - 每根 I/O 口线最大灌电流 20mA
- 一个 8 位定时器/计数器，可带 8 位预分频器。

三、微控制器特性

- 上电复位。
- 上电延时器保障 VDD 稳定建立。
- 振荡定时器保障振荡稳定建立。
- 自振式看门狗。
- 程序保密位。
- 低功耗睡眠状态。
- 四种可选择振荡方式：
 - 低成本阻容：RC
 - 标准晶体/陶瓷：XT
 - 高速晶体/陶瓷：HS

- 低频晶体: LP

四、CMOS 工艺特性

- 低功耗
 - $<2\text{mA}$ @5V, 4MHZ
 - $60\mu\text{A}$ @2V, 32KHZ
 - $26\mu\text{A}$ @2V, 睡眠模式下
- 全静态设计
- 宽工作电压
 - 商用级: 2.0V-6.0V
 - 工业级: 2.0V-6.0V
- 宽工作温度范围:
 - 商用级: $0^{\circ}\text{C} \sim +70^{\circ}\text{C}$
 - 工业级: $-40^{\circ}\text{C} \sim +85^{\circ}\text{C}$
 - 汽车级: $-40^{\circ}\text{C} \sim +125^{\circ}\text{C}$

PIC16C8X 由于具有 E²PROM 工艺特性（电可擦写），所以它极适合于那些可能会经常改动程序编码的应用，例如用户可以随时改动已经出厂的产品中的单片机程序以增加或调整产品的功能。最具特色的是它内部的 64 字节 E²PROM 型数据存储器不仅有掉电保护数据的功能，更重要的是它只能由单片机内部进行控制操作进行读写，即外部电路无法对其进行读写，所以有极高的数据保密性，这使得 PIC16C8X 在加密性产品如 IC 卡、密码锁、防盗系统等方面有很广泛的应用。

§ 4.2 芯片类型

PIC16C8X 都是 18 脚的塑封，有双列直插 DIP/表面封装 SOIC 二种方式，另还提供半导体芯 Dies。由于 PIC16C8X 是 E²PROM 工艺的程序存储器，所以它是电可擦写，没有窗口型（紫外光可擦）和 OTP（一次性可编程）的型号，这点和其他 PIC 单片机不一样。

§ 4.3 引脚介绍

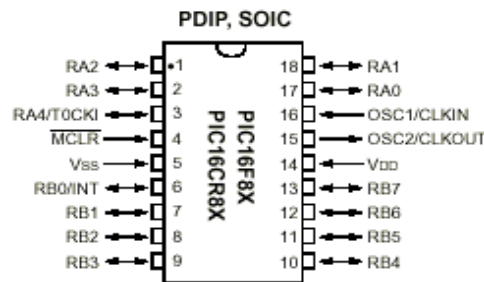


图 4.1 PIC16C8X 引脚图

细心的读者可能已经发现，PIC16C8X 的外形引脚和 PIC16C61 完全一样！的确，如果把 PIC16C61 的程序存储器换成 E²PROM 工艺，加上 64 字节的 E²PROM 型数据存储器，再增加几个有关 E²PROM 操作的寄存器和 E²PROM 中断方式，则就变成了 PIC16C84。所以我们下面对 PIC16C8X 的介绍，就侧重于 PIC16C8X 独有的部份，即有关 E²PROM 型数据存储器的部份，对于和 PIC16C61 完全相同的，请读者参阅第一章的有关内容。

关于 PIC16C83/84 各引脚意义请参阅 § 1.3 中 PIC16C61 的部分，两者完全一致。

§ 4.4 内部结构

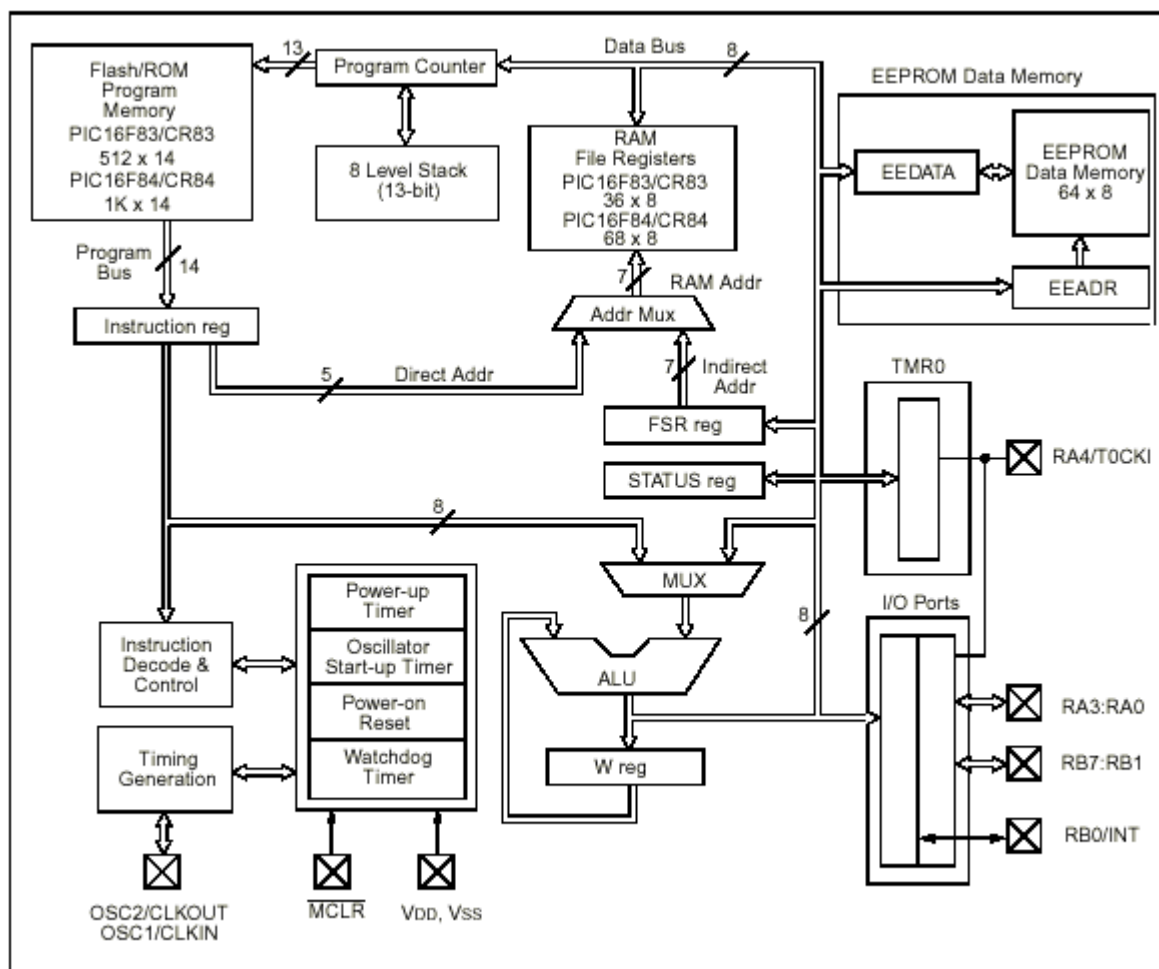


图 4.2 PIC16C8X 内部结构

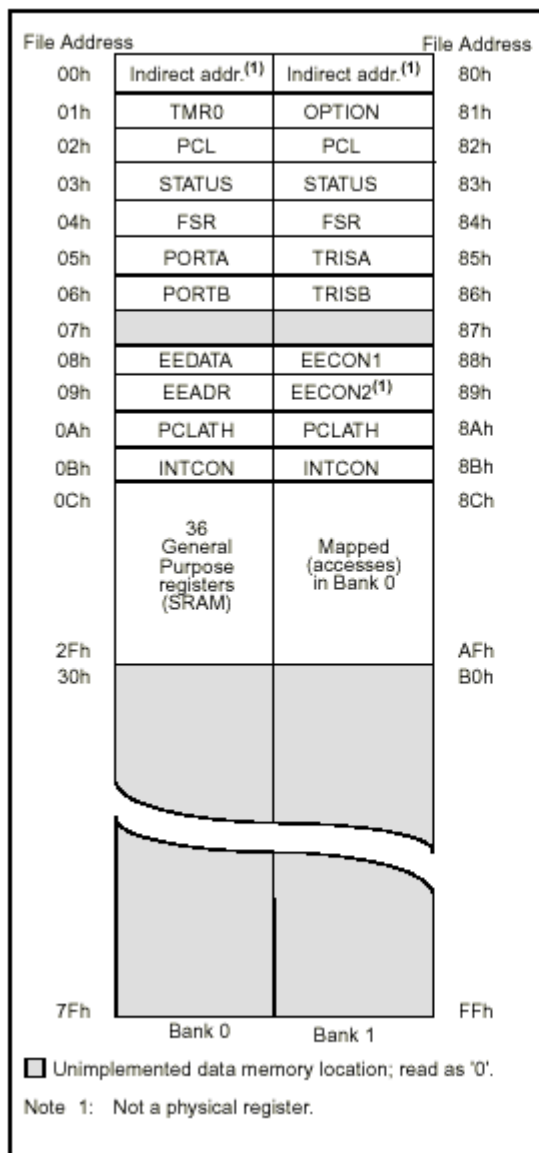
从图中可看出，内部的 E²PROM 数据存储器是完全由 CPU 控制的，外部无法对其进行操作。

§ 4.5 指令时序

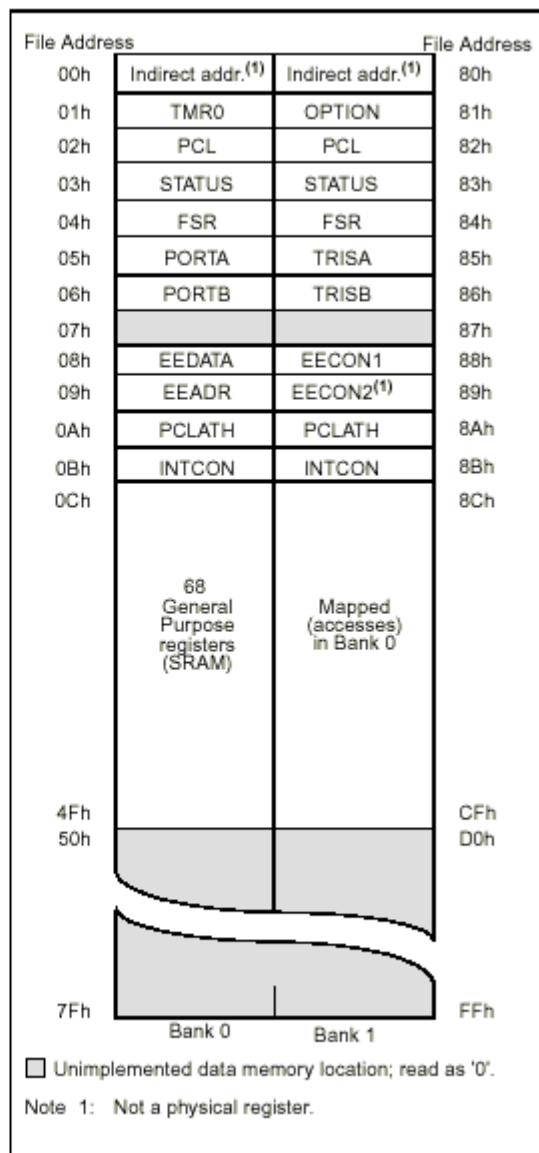
PIC16C8X 的指令时序和 PIC16C6X/7X 完全一样，请参阅 § 1.5。

§ 4.6 程序存储器和堆栈

PIC16C8X 的计数器 PC 也是 13 位长，最多可寻址 8K。但 PIC16C84 只使用前 1K，PIC16C83 只使用前 0.5K 空间，如下图：



a. PIC16F83/C83



b. PIC16F84/C84

图 4.4 PIC16C8X 寄存器结构

PIC16C8X 各特殊功能寄存器值如下表所示:

地 址	名 称	功 能 说 明	上电复位值	其他复位值
Bank0 (0 体)				
00h	INDF	间接寻址逻辑寄存器 (物理上不存在)	0000 0000	0000 0000
01h	TMR0	TIMER0 模块寄存器	xxxx xxxx	uuuu uuuu
02h	PCL	程序计数器 PC 的低 8 位	0000 0000	0000 0000
03h	STATUS	状态寄存器	0001 1xxx	000q quuu
04h	FSR	间接寻址寄存器	xxxx xxxx	uuuu uuuu
05h	PORTA	— — — PORTA 口寄存器	---x xxxx	---u uuuu
06h	PORTB	PORTB 寄存器	xxxx xxxx	uuuu uuuu
07h	—	—	—	—
08h	EEDATA	E2PROM 数据寄存器	xxxx xxxx	uuuu uuuu
09h	EEADR	E2PROM 地址寄存器	xxxx xxxx	uuuu uuuu
0Ah	PCLATH	— — — PC 高 5 位之写入器	---0 0000	---0 0000
0Bh	INTCON	中断控制寄存器	0000 000x	0000 000u
Bank1 (1 体)				
80h	INDF	间接寻址逻辑寄存器 (物理上不存在)	0000 0000	0000 0000
81h	OPTION	系统功能定义寄存器	1111 1111	1111 1111

82h	PCL	程序计数器 PC 的低 8 位				0000	0000	0000	0000
83h	STATUS	状态寄存器				0001	1xxx	000q	quuu
84h	FSR	间接寻址寄存器				xxxx	xxxx	uuuu	uuuu
85h	TRISA	—	—	—	PORTA 方向寄存器	---1	1111	---1	1111
86h	TRISB	PORTB 方向寄存器				1111	1111	1111	1111
87h	—	—				—	—	—	—
88h	EECON1	E2PROM 控制寄存器 1				---0	X000	---0	q000
89h	EECON2	E2PROM 控制寄存器 2（物理上不存在）				—	—	—	—
8Ah	PCLATH	—	—	—	PC 高 5 位之写入器	---0	0000	---0	0000
8Bh	INTCON	中断控制寄存器				0000	000X	0000	000u

注： X=不定， u=不变， q=取决于某条件， —=未用（读为 0）

表 4.2 16C8X 特殊功能寄存器

从图中可看出，PIC16C8X 的寄存器组中比 PIC16C61 多出四个：

1. EEDATA (08H) — E²PROM 数据寄存器
2. EEADR (09H) — E²PROM 地址寄存器
3. EECON1 (88H) — E²PROM 控制寄存器
4. EECON2 (89H) — E²PROM 控制寄存器

关于这四个新增寄存器我们将在下面一节中进行描述，其余的寄存器请读者参阅 PIC16C61 的寄存器部分，当然在中断控制寄存器 INTCON 中增加一位 E²PROM 操作中断位 EEIE (INTCON<6>)，这将在 § 4.9 中描述。

§ 4.8 E²PROM 型数据存储器

在 PIC16C8X 中，有 64×8 个 E²PROM 数据存储器，可由程序进行读写操作。它们不是映像普通的寄存器组中，不能由指令直接寻址，而需通过特殊寄存器来进行控制操作，下面即分别叙述。

一、数据寄存器 EEDATA

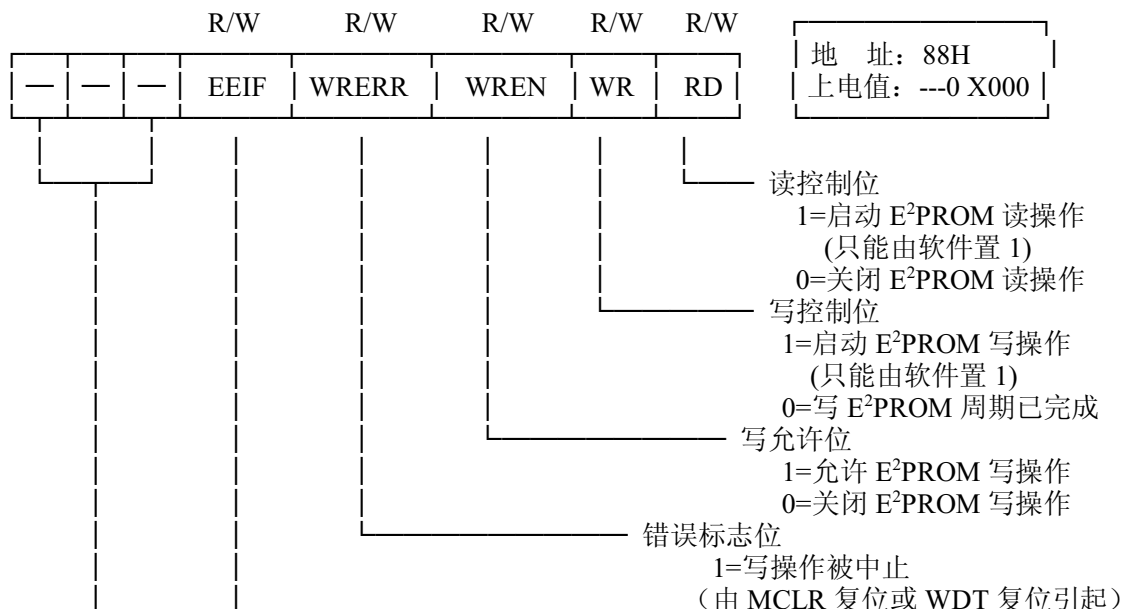
8 位的 EEDATA 寄存器中存放的是要读/写的数据。当要对 E²PROM 存储器写入时，先要把写入数据置入 EEDATA 中，而当读 E²PROM 存储器时，CPU 即是把读出数据放入 EEDATA 中再由用户程序去读取。

二、地址寄存器 EEADR

8 位的地址寄存器 EEADR 可以寻址 256 个字节的 E²PROM 存储器，但在 PIC16C8X 中目前只使用前 64 个字节地址 (0-3FH)，所以只有 EEADR<5:0>是真正用得上的，参见 § 4.8.3 内容。

三、控制寄存器 EECON1 和 EECON2

下图是 EECON1 中各位的意义。



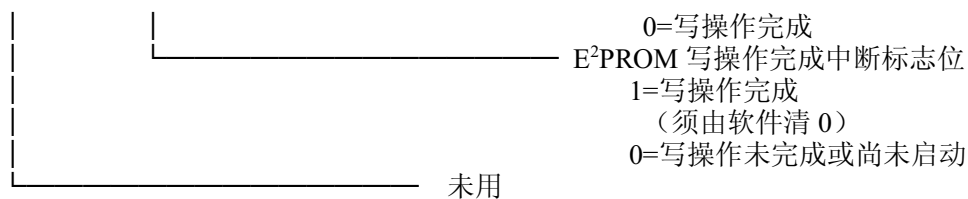


图 4.5 EECON1 寄存器

EECON2 不是一个物理上存在的寄存器，如果读它将是读到全 0，它只在写操作时起作用，见下面 § 4.8.2 的描述。

§ 4.8.1 E²PROM 读操作

为进行一次 E²PROM 读操作，须做如下步骤：

1. 将 E²PROM 的单元地址放入 EEADR；
2. 置 RD (EECON1<0>) = 1；
3. 读取 EEDATA 寄存器。

例程：读取 20h 处的 E²PROM 存储器数据。

```

...
BCF      STATUS, RP0      ;选 Bank0
MOVLW    20H
MOVWF    EEADR            ;地址 20H→EEADR
BSF      STATUS, RP0      ;选 Bank1
BSF      EECON1, RD       ;启动读操作
BCF      STATUS, RP0      ;选 Bank0
MOVWF    EEDATA, W        ;将 E²PROM 数据读入 W 寄存器
...

```

§ 4.8.2 E²PROM 写操作

为进行一次 E²PROM 写操作，用户程序须做如下步骤：

- 1、将 E²PROM 单元地址放入 EEADR，把写入数据放入 EEDATA；
- 2、将写入数据置入 EEDATA；
- 3、做一控制序列，见下面例程。

例程：将数据 88h 写入 E²PROM 之 20H 单元。

```

...
BCF      STATUS, RP0      ;选 Bank0
MOVLW    20H
MOVWF    EEADD            ;地址→EEADD
MOVLW    88H
MOVWF    EEDATA           ;写入数据→EEDATA
BSF      STATUS, RP0      ;选 Bank1
BSF      EECON1, WREN      ;写操作使能允许
① BCF      INTCON, GIE      ;关闭所有中断
② MOVLW    0X55
③ MOVWF    EECON2
④ MOVLW    0XAA
⑤ MOVWF    EECON2          ;操作 EECON2 序列
⑥ BSF      EECON1, WR      ;启动写操作
⑦ BSF      INTCON, GIE      ;恢复开中断

```

...

注意，上面例程中的序列②～⑥必须严格执行，否则将不能启动 E²PROM 写操作。而①和⑦则是我们特别建议用户这样做，即在 E²PROM 写操作序列步骤中要关闭所有中断以免这个序列被中断打断。

另外，WREN (EECON1<2>) 是用来保证 E²PROM 不会被意外写入 (象 24CXX 中的 WP 脚所起的作用)，所以在平时用户程序应保持 WREN=0 来禁止写操作，当需对 E²PROM 写入时才置 WREN=1，并在写入完成后将其恢复为 0。用户只有置 WREN=1 后才能置 WR (EECON1<1>)=1 启动写操作。等上电复位后 WREN 位是自动清为零。

E²PROM 写操作约需 10ms 的时间才能完成。用户程序可以通过查询 WR 位的状态 (当 WR=0 时表示写操作已完成) 或是利用 E²PROM 写入完成中断来判断一次 E²PROM 写操作是否完成。如要使用中断，应先置 EEIE (INTCON<6>)=1 开中断。

§ 4.8.3 E²PROM 操作功耗

E²PROM 操作有一定的功耗，为了使这个功耗最低，建议用户程序置 EEADR<7:6>=00，这样芯片的 IDD 约为 150 μA，而如果<7:6>=11，则 IDD 约为 400 μA。所以建议用户程序在上电复位后即将 EEADR<7:6>置为“00”。

例程：置 EEADR<7:6>=00

...

```
BCF      EEADR, 6
BCF      EEADR, 7
```

§ 4.9 I/O 口

PIC16C8X 具有 2 个双向可编程 I/O 口：5 位的 PORTA 和 8 位的 PORTB。它们和 PIC16C61 的 PORTA 和 PORTB 完全相同，请参阅 § 1.9。

§ 4.10 定时器/计数器

PIC16C84 有一个 8 位定时器/计数器 TIMER0，它和 PIC16C6X 的 TIMER0 完全一样，有 8 位预分频器，有溢出中断等，请参阅 § 1.10 的内容。

§ 4.11 中 断

PIC16C8X 有 4 种中断源：

- 1、外部 INT 触发中断；
- 2、TMR0 溢出中断；
- 3、PORTB<7:4>电平变化中断；
- 4、E²PROM 写操作完成中断。

其中 1～3 的中断方式和 PIC16C61 的 3 种中断方式完全一样，请参阅第一章中有关 PIC16C61 的这部分内容。而第 4 种中断则是 PIC16C8X 特有的，见下图：

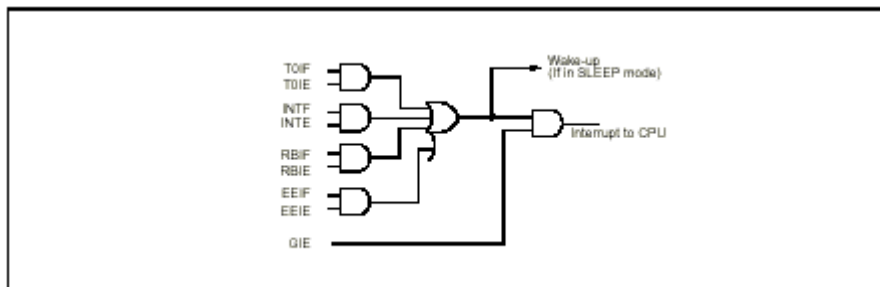


图 4.6 PIC16C8X 中断源

PIC16C8X 的中断控制寄存器 INTCON 如下图所示：

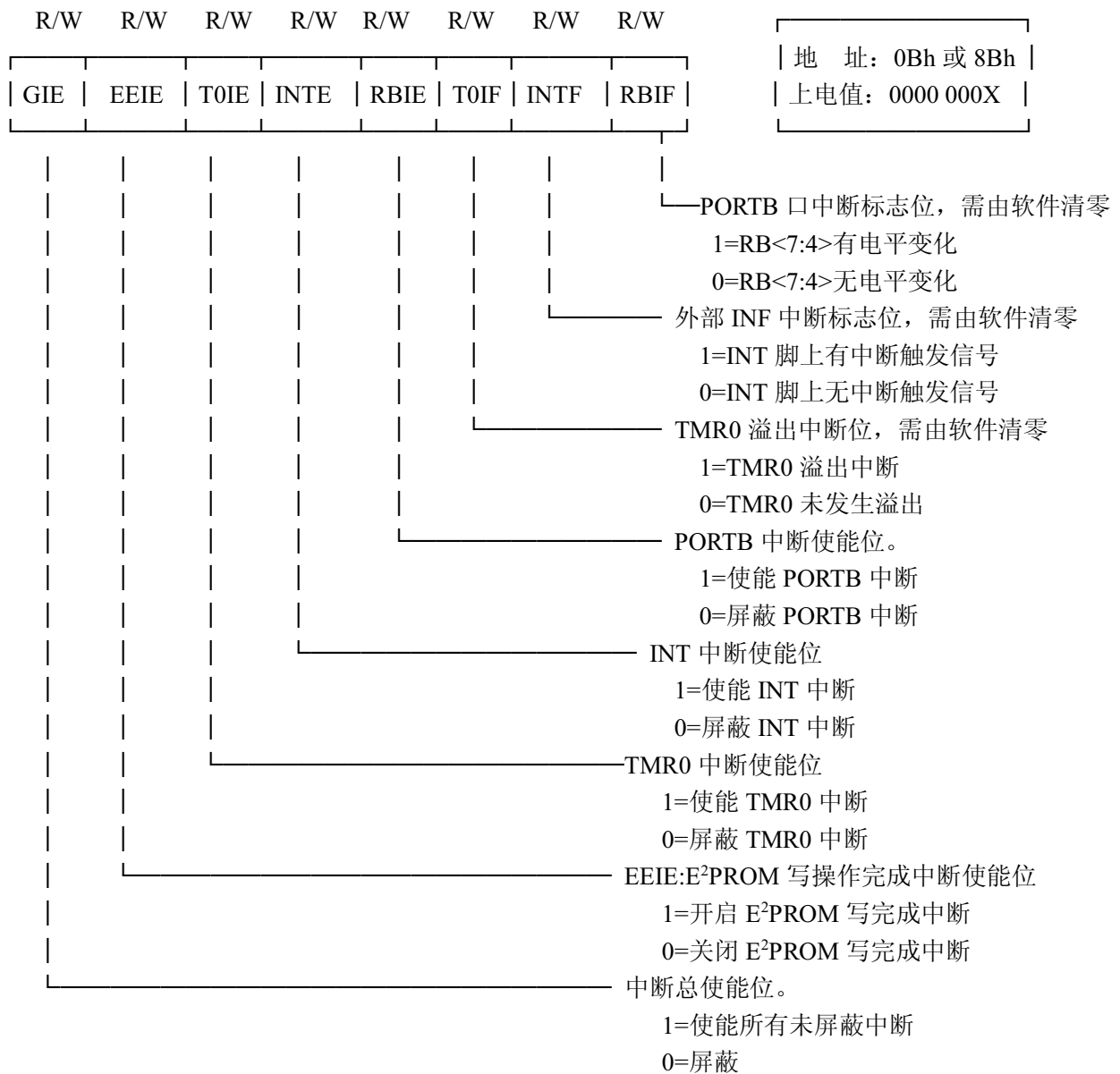


图 4.7 PIC16C8X 中断控制寄存器

E²PROM 中断标志位 EEIF 在 EECON1 寄存器中，请参阅图 4.5。

有关 PIC16C8X 的中断处理事项请参阅 § 1.13.4 有关 PIC16C6X 的中断处理部分，它们完全相同。

§ 4.12 CPU 特性

PIC16C8X 和其他 PIC 单片机一样具备许多微处理器特性，如看门狗 WDT、4 种振荡选择、多种复位方式、程序保密位以及上电延时定时器 PWRT 和振荡起振定时器 OST 等等，都和 PIC16C61 等完全一样，请参阅 § 1.13 有关描述。

PIC16C84 各种复位后各特殊寄存器的状态值如下表所示：

条 件	PCL (地址:02H)	状态寄存器 (地址:03H/83H)
上电复位	000h	0001 1xxx
正常运行时 MCLR 复位	000h	0001 1uuu
睡眠状态下 MCLR 复位	000h	0001 0uuu
正常运行时看门狗定时器超时复位	000h	0000 1uuu
睡眠状态下看门狗定时器超时复位	PC+1	uuu0 0uuu
睡眠状态下中断唤醒	PC+1	uuu1 0uuu

注：X=不定；u=不变

表 4.3 PIC16C84 复位或唤醒后 PCL 寄存器和状态寄存器的初值

寄存器	地 址	上电复位	• MCLR 复位： -正常操作中 -睡眠状态下 • 看门狗超时复位	• 在睡眠时唤醒 -中断唤醒 -看门狗超时唤醒
W	—	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF	00H	---- ----	---- ----	---- ----
TMR0	01H	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	02H	0000h	0000h	PC + 1
STATUS	03H	0001 1xxx	000? ?uuu	uuu? ?uuu3
FSR	04H	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	05H	---u uuuu	---u uuuu	---u uuuu
PORTB	06H	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEDATA	08H	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEADR	09H	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCLATH	0AH	---0 0000	---0 0000	---u uuuu
INTCON	0BH	0000 000x	0000 000u	uuuu uuuu1
INDF	80H	---- ----	---- ----	---- ----
OPTION	81H	1111 1111	1111 1111	uuuu uuuu
PCL	82H	0000h	0000h	PC + 1
STATUS	83H	0001 1xxx	000? ?uuu	uuu? ?uuu
FSR	84H	xxxx xxxx	uuuu uuuu	uuuu uuuu
TRISA	85H	---1 1111	---1 1111	---u uuuu
TRISB	86H	1111 1111	1111 1111	uuuu uuuu
EECON1	88H	---0 0000	---0 ?000	---0 ?uuu
EECON2	89H	---- ----	---- ----	---- ----
PCLATH	8AH	---0 0000	---0 0000	---u uuuu
INTCON	8BH	0000 000x	0000 000u	uuuu uuuu1

注：X=不定； u=不变

表 4.4 PIC16C84 复位或唤醒后各特殊寄存器的初值

总之，PIC16C8X 是易于学习和使用的单片机，读者如果已了解 PIC16C61 等芯片的知识，只要着重学习其相关 E²PROM 数据存储器方面的内容，即可掌握 PIC16C8X 的设计应用。

第五章 PIC16C62X 单片机

PIC16C62X 是内部带 2 路电压比较器的型号，目前有表中所列的几种型号：

型号	振荡	程序区	寄存器	电压	中断	I/O 脚	定时器	复位锁定	看门狗	电压比较器	封装
16C620	DC-20M	0.5K×14	80	3.0-6.0	4	13	1	有	有	2	18
16C621	DC-20M	1K×14	80	3.0-6.0	4	13	1	有	有	2	18
16C622	DC-20M	2K×14	128	3.0-6.0	4	13	1	有	有	2	18

表 5.1 PIC16C62X 型号功能表

大家注意到 PIC16C62X 没有 CCP、SCI、SSP 及并行口等功能模块，它是一种小型的易于嵌入式控制的单片机，比较接近 PIC16C61/71。下面我们会着重叙述 PIC16C62X 独有的“片内电压比较器”部份，而对于其他和 PIC16C61/71 相同的部分则请读者参阅第一章中有关的内容。

§ 5.1 主要功能特点

一、高性能 RISC 结构 CPU

- 精简指令集，仅 35 条单字节指令。
- 除地址分支指令外，其余全为单周期指令。
- 执行速度:DC-200ns。
- 八级硬件堆栈。
- 多种硬件中断。
- 直接/间接/相对三种寻址方式。

二、功能部件

- 13 根可独立编程双向 I/O 口线。
- 高驱动电流 I/O 脚，可直接驱动 LED 显示。
每根 I/O 口线最大拉电流 25mA
每根 I/O 口线最大灌电流 20mA
- 一个 8 位定时器/计数器，可带 8 位预分频器。
- 模拟比较器部件
 - 2 路电压比较器
 - 可编程片内参考电压 VREF
 - 可编程比较器输入
 - 比较器输出可作为输出信号

三、微控制器特性

- 上电复位。
- 上电延时器保障 VDD 稳定建立。
- 振荡定时器保障振荡稳定建立。
- 自振式看门狗。
- 程序保密位。
- 低功耗睡眠状态。
- 四种可选择振荡方式：
 - 低成本阻容：RC
 - 标准晶体/陶瓷：XT
 - 高速晶体/陶瓷：HS

- 低频晶体：LP

四、CMOS 工艺特性

- 低功耗
 - <2mA @5V, 4MHZ
 - 15 μ A @3V, 32KHZ
 - 1 μ A @3V, 睡眠模式下
- 全静态设计
- 宽工作电压
 - 商用级：3.0V-6.0V
 - 工业级：3.0V-6.0V
- 宽工作温度范围：
 - 商用级：0℃~+70℃
 - 工业级：-40℃~+85℃
 - 汽车级：-40℃~+125℃

PIC16C62X 内部的电压比较器提供了低价的模拟输入接口，使之能够在低成本模拟信号控制应用中得到青睐，大家可以在仪器仪表、家用电器、充电器、智能传感器等等产品中看到它的身影。

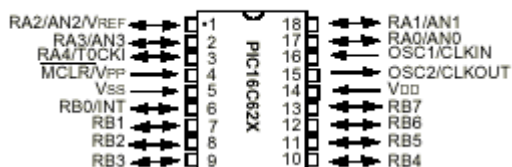
§ 5.2 芯片类型

和 PIC16C6X/7X 等完全一样，参阅 § 1.2 等章节。

§ 5.3 引脚介绍

PIC16C62X 的芯片引脚（PDIP）如下图所示：

PDIP, SOIC, Windowed CERDIP



SSOP

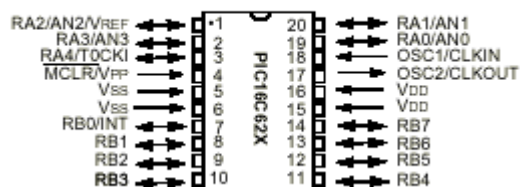


图 5.1 PIC16C62X 引脚图

从上图读者可以看到，PIC16C62X 外形引脚和 PIC16C61/71 非常接近，几乎可以完全兼容。的确，从 I/O 数字功能来看，三者完全兼容，PIC16C71 是加上 4 路 A/D 输入功能，而 PIC16C62X 是加上 4 路模拟比较输入功能。

引 脚 名	I/O 特性	电 平	功 能
OSC1/CLKIN	输入	CMOS	振荡输入脚
OSC2/CLKOUT	输出	—	振荡输出脚
MCLR	输入	ST	复位输入脚，低电平有效
RA0/AN0	I/O	TTL	PORTA 数字 I/O 口，双向可编程
RA1/AN1	I/O	TTL	比较器输入 0
RA2/AN2	I/O	TTL	比较器输入 1
RA3/AN3/VREF	I/O	TTL	比较器输入 2
RA4/T0CKI	I/O	ST	比较器输入 3/参考电压输入 VREF 也可作为 TMR0 外部时钟输入
RB0/INT	I/O	TTL/ST	PORTB 数字 I/O 口，双向可编程
	I/O		亦可作为外部中断信号输入脚（此时为 ST 输入）

RB1	I/O	TTL	具有电平变化中断功能 具电平变化中断功能 具电平变化中断功能 具电平变化中断功能
RB2	I/O	TTL	
RB3	I/O	TTL	
RB4	I/O	TTL	
RB5	I/O	TTL	
RB6	I/O	TTL	
RB7	I/O	TTL	
VSS	—	—	地
VDD	—	—	电源

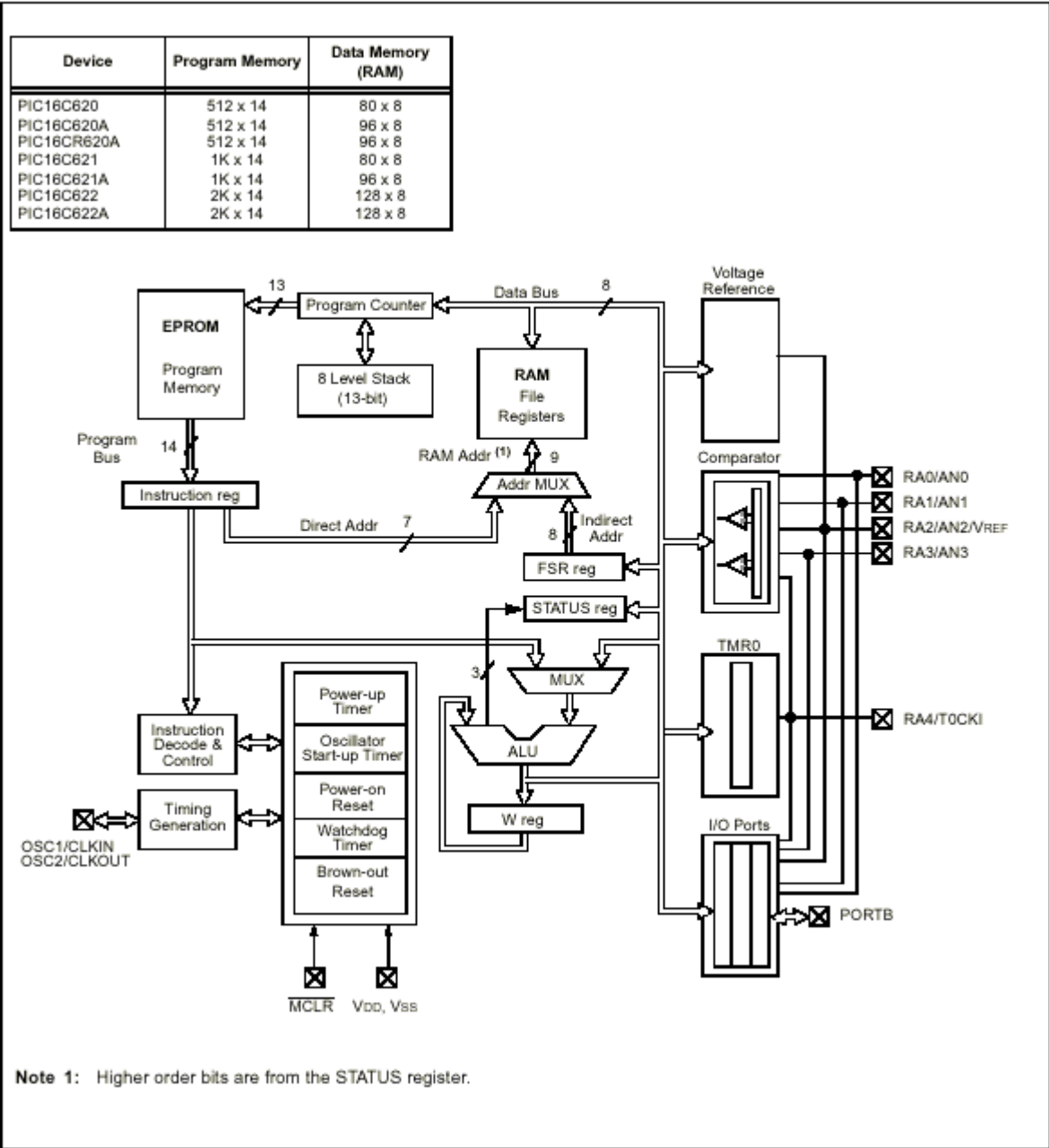
ST： 斯密特输入

表 5.2 PIC16C62X 引脚功能

§ 5.4 内部结构

PIC16C62X 的内部结构和其他的 PIC16CXX 基本上一样，只是增加了 2 个模拟比较器，如下图所示：

图 5.2 PIC16C62X 内部结构

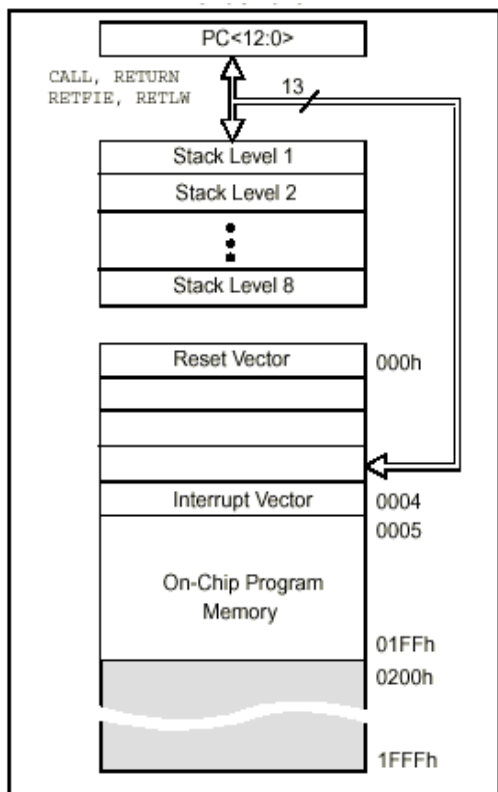


§ 5.5 指令时序和流水作业

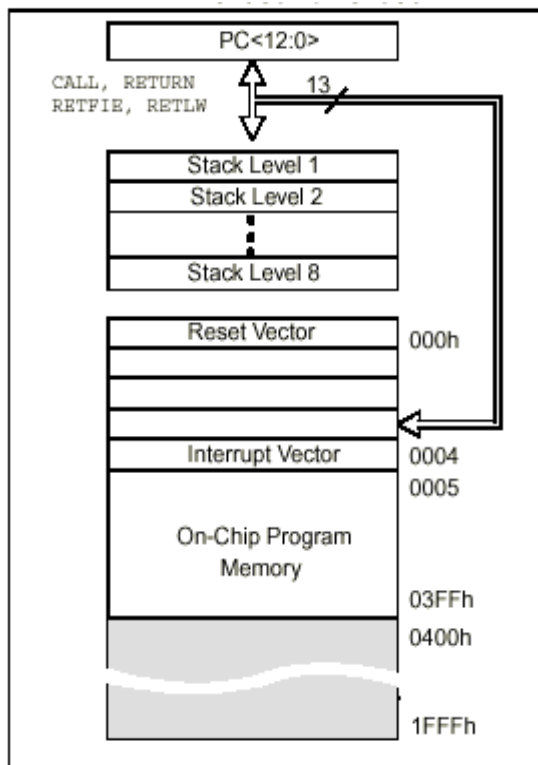
和其他 PIC16CXX 完全一样，请参阅 § 1.5。

§ 5.6 程序存储器和堆栈

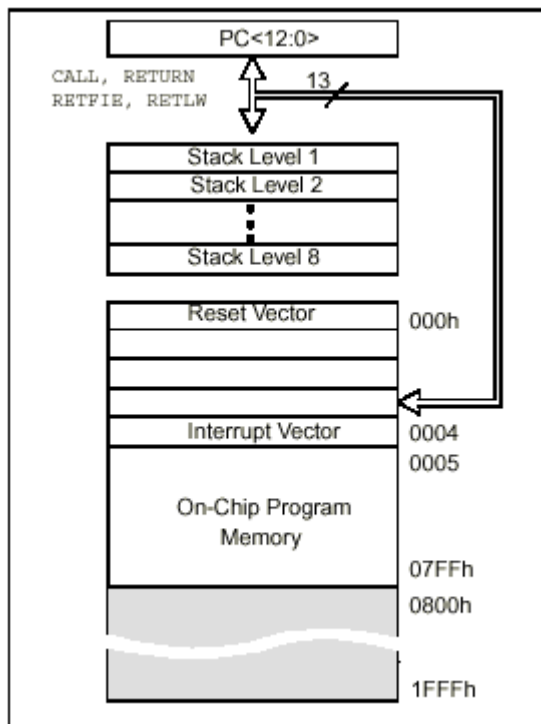
PIC16C62X 的程序计数器 PC 是 13 位长，最大可寻 8K 的空间，但目前只使用前 0.5K~2K 的空间，如下图所示：



a. PIC16C620



b. PIC16C621



c. PIC16C622

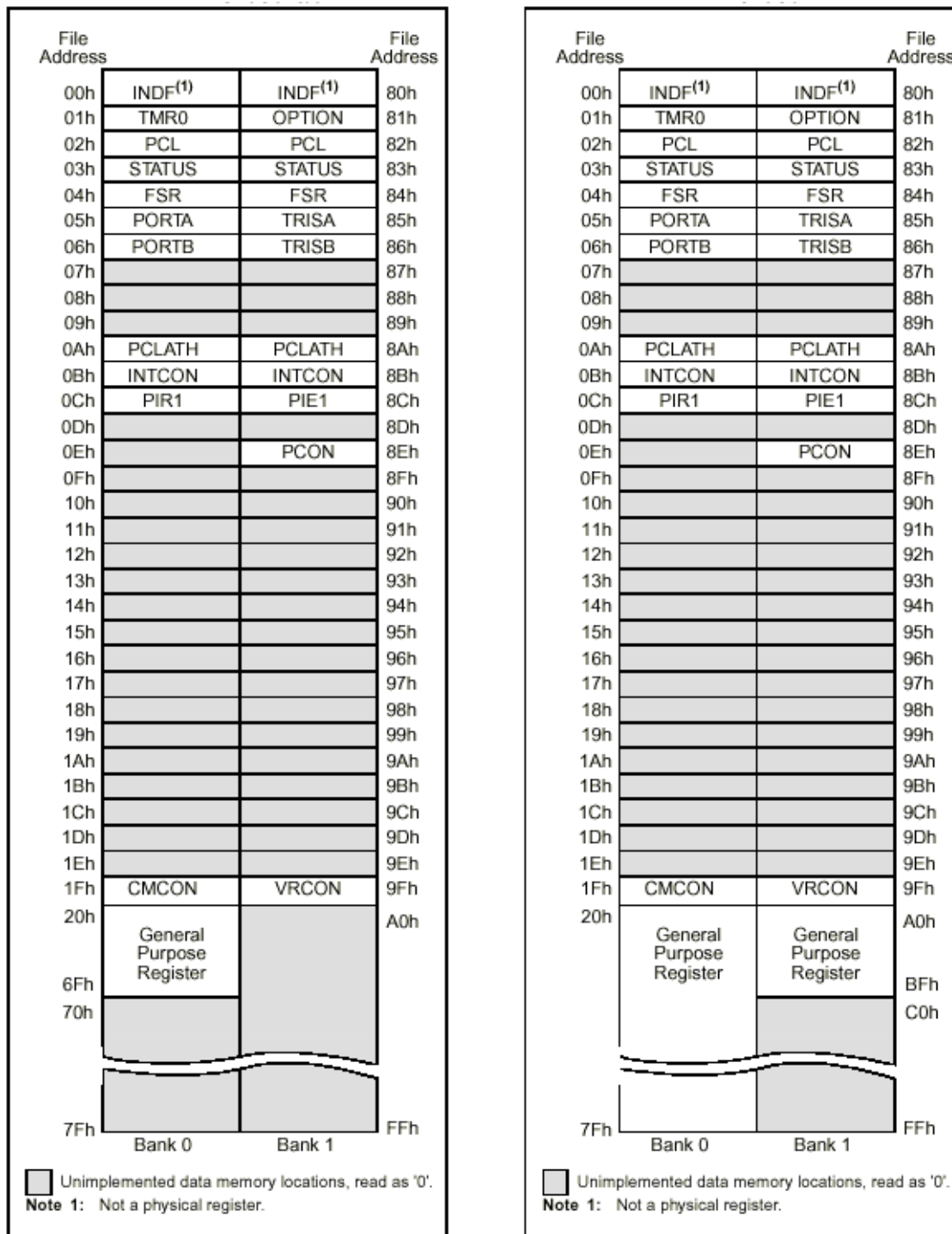
图 5.3 PIC16C62X 程序存储器结构

任何超出程序存储器实际空间的寻址都将是物理空间上的回绕而已。

PIC16C62X 也有独立 8 级硬件堆栈，不占用程序存储器空间。

§ 5.7 数据寄存器

PIC16C62X 的寄存器分为 2 个体:Bank0 和 Bank1, 如下图所示:



a. PIC16C620/621

b. PIC16C622

图 5.4 PIC16C62X 寄存器结构

体的选择由状态寄存器中的 RP0 和 RP1 两位来决定, 参阅 § 1.7.2 中有关 STATUS 寄存器的描述。从功能上分有特殊功能寄存器和通用寄存器两种, 和其他 PIC16CXX 完全一样。

地 址	名 称	功 能 说 明	上电复位值	其他复位值
Bank0 (0 体)				
00h	INDF	间接寻址逻辑寄存器 (物理上不存在)	0000 0000	0000 0000
01h	TMR0	TIMER0 模块寄存器	xxxx xxxx	uuuu uuuu
02h	PCL	程序计数器 PC 的低 8 位	0000 0000	0000 0000
03h	STATUS	状态寄存器	0001 1xxx	000q quuu

04h	FSR	间接寻址寄存器				XXXX XXXX	uuuu uuuu
05h	PORTA	—	—	—	PORTA 口寄存器	---X XXXX	---u uuuu
06h	PORTB	PORTB 寄存器				XXXX XXXX	uuuu uuuu
07h	—	——				—	—
08h	—	——				—	—
09h	—	——				—	—
0Ah	PCLATH	—	—	—	PC 高 5 位之写入器	---0 0000	---0 0000
0Bh	INTCON	中断控制寄存器				0-00 000x	0-00 000u
Bank1（1 体）							
80h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000 0000	0000 0000
81h	OPTION	系统功能定义寄存器				1111 1111	1111 1111
82h	PCL	程序计数器 PC 的低 8 位				0000 0000	0000 0000
83h	STATUS	状态寄存器				0001 1xxx	000q quuu
84h	FSR	间接寻址寄存器				XXXX XXXX	uuuu uuuu
85h	TRISA	—	—	—	PORTA 方向寄存器	---1 1111	---1 1111
86h	TRISB	PORTB 方向寄存器				1111 1111	1111 1111
87h	—	——				—	—
88h	—	——				—	—
89h	—	——				—	—
8Ah	PCLATH	—	—	—	PC 高 5 位之写入器	---0 0000	---0 0000
8Bh	INTCON	中断控制寄存器				0-00 000X	0-00 000u
8Ch	PIE1	比较器中中断使能位寄存器				-0-- ----	-0-- ----
8Dh	—	——				—	—
8Eh	PCON	上电复位和掉电复位标志位寄存器				---- --0X	---- --uq
8Fh~9Eh	—	——				—	—
9Fh	VRCON	比较器参考电压值寄存器				000- 0000	000- 0000
0Ch	PIR1	比较器中断标志位寄存器				-0-- ----	-0-- ----
0Dh~1Eh	—	——				—	—

注：X=不定， u=不变， q=取决于某条件， -=未用（读为 0）

表 5.3 PIC16C62X 特殊功能寄存器

其中有关模拟比较器的几个寄存器我们留待在有关章节中介绍，下面仅介绍寄存器 PCON。其他如状态寄存器 STATUS，OPTION 寄存器等和其他 PIC16CXX 完全一样，请参阅有关章节。

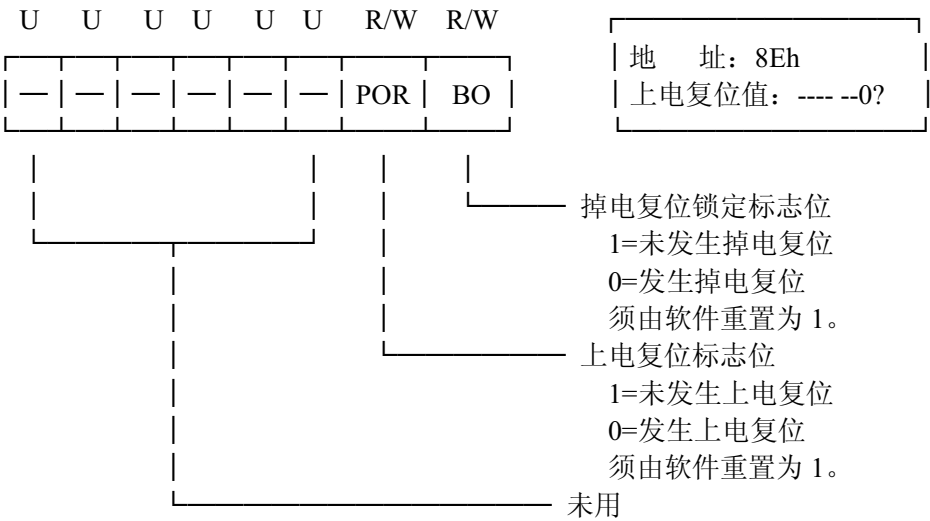
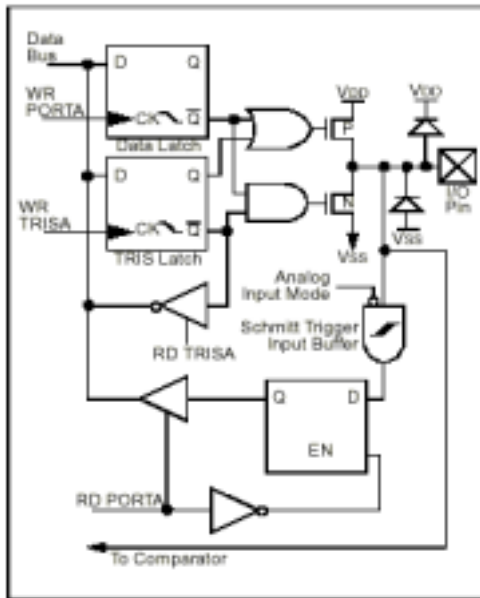


图 5.5 PCON 寄存器

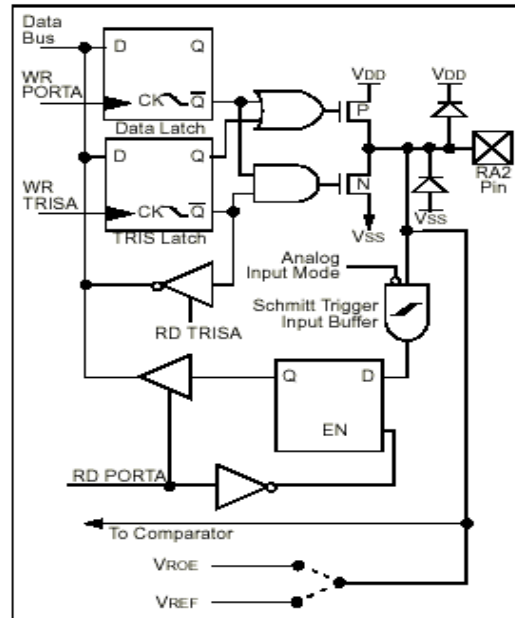
PIC16C62X 有二个 I/O 口：PORTA 和 PORTB，下面分别介绍。

§ 5.8.1 PORTA

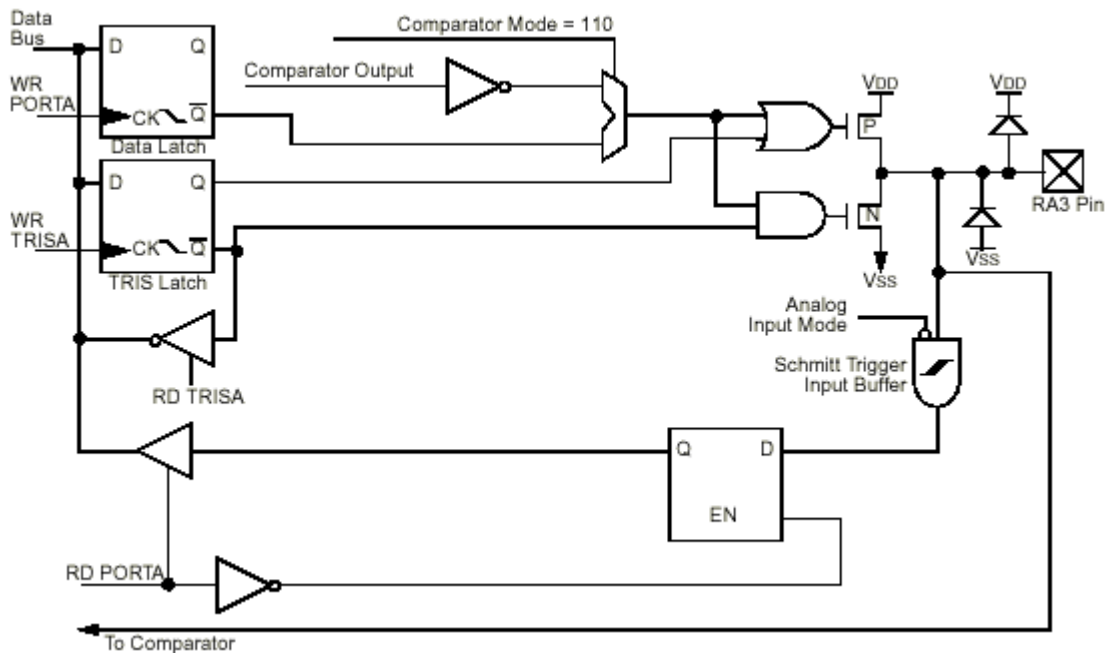
PORTA 是 5 位 (RA<0>~RA<4>) 的 I/O 口, 由 TRISA (85H) 控制 I/O 方向。作为数字 I/O 口, 它和其他的 PIC16CXX 完全一样。请参阅 § 1.8.1 的描述。在 PIC16C62X 中, PORTA 的 RA<0>~RA<3>还可以作为模拟比较输入/输出, 见下图:



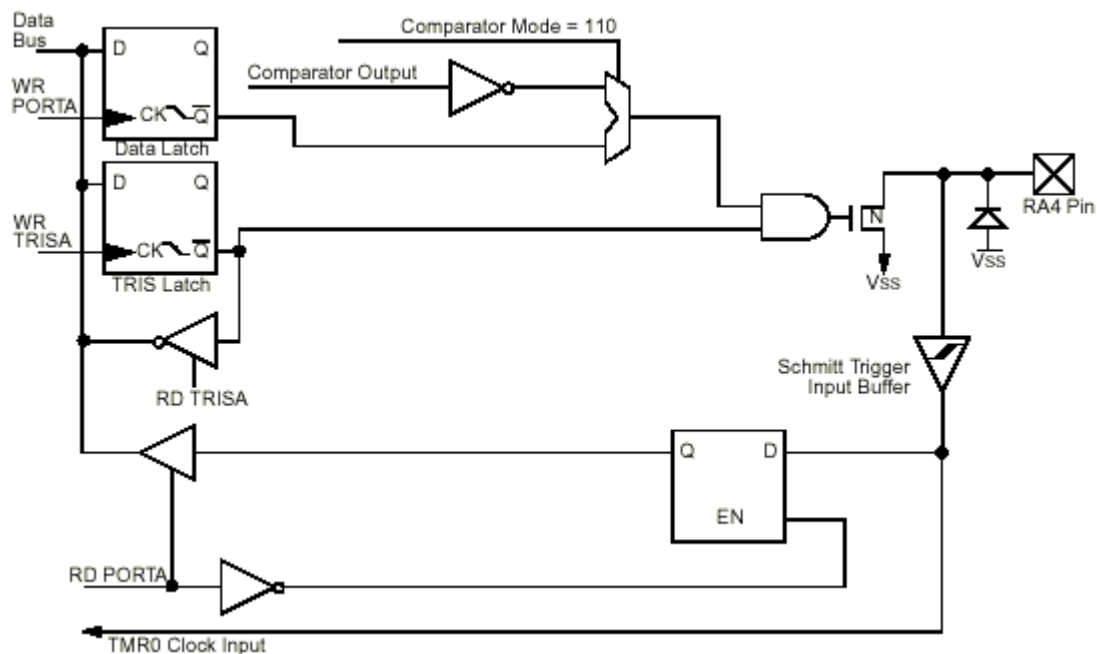
a. RA<1:0>



b. RA2



c.RA3



d. RA4

图 5.6 PORTA 结构

关于 PORTA 作为模拟比较器输入/输出的描述参阅 § 5.1。

§ 5.8.2 PORTB

PIC16C62X 的 8 位 PORTB 及其方向控制寄存器 TRISB (86H) 的功能和其他 PIC16CXX 完全一样，请参阅 § 1.8.2。

§ 5.9 计数器/定时器 TIMER0

和其他 PIC16CXX 完全一样，参阅 § 1.9。

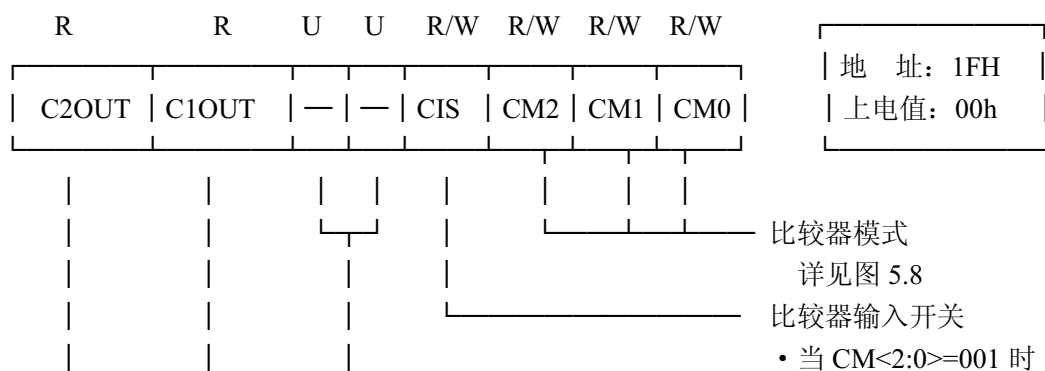
§ 5.10 看门狗

和其他 PIC16CXX 完全一样，参阅 § 1.13.5。

§ 5.11 比较器模块

PIC16C62X 独有的比较器模块含有 2 个模拟比较器。RA0~RA3 都可以作为比较器的输入，而片内的参考电压也可以作为比较器的输入。

比较器控制寄存器 CMCON 控制比较器的输入/输出组合，见下面二图：



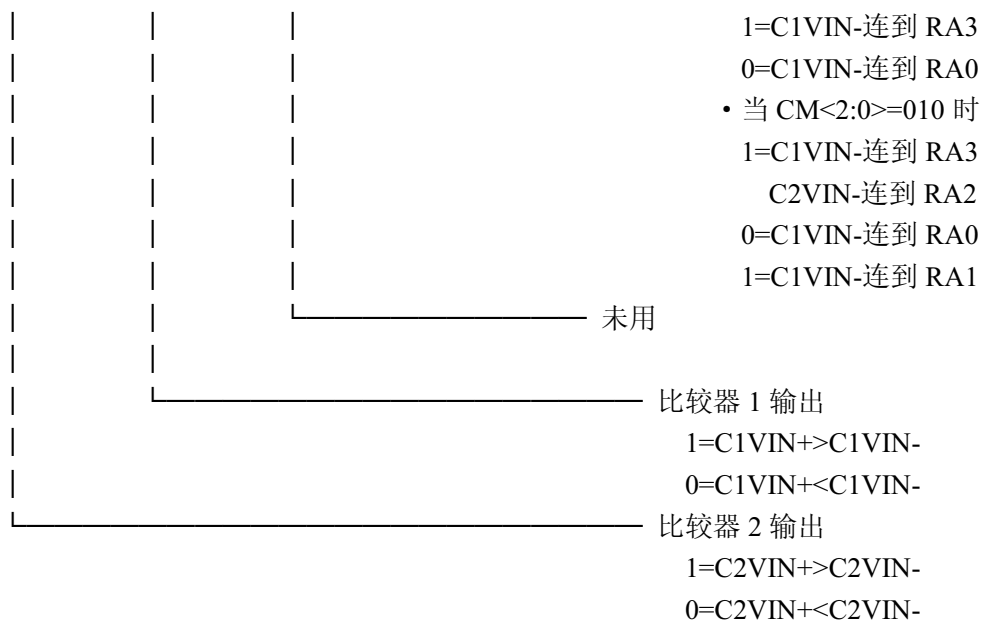


图 5.7 比较器控制器 CMCON

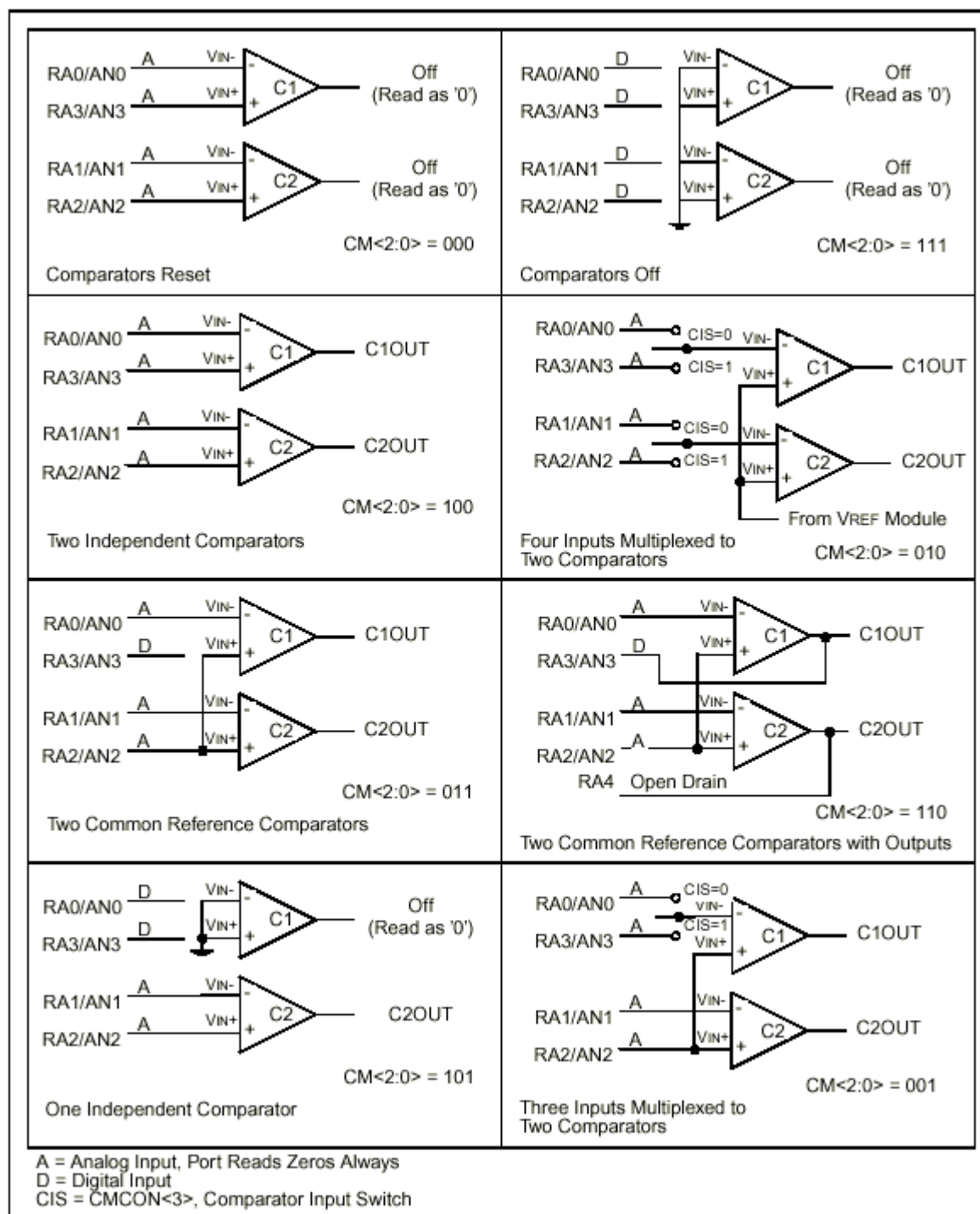


图 5.8 比较器工作模式

§ 5.11.1 比较器模式设置

从图 5.8 我们可以看到，PIC16C62X 的比较器有 8 种工作模式，它们由控制寄存器 CMCON<0:2>三位设置选择，寄存器 TRISA 控制每种模式下比较器的数据方向。如果要改变工作模式，则应先关闭比较器中断以免产生误中断。

下面这段程序是改变设置比较器工作模式的范例，假设 RA3 和 RA4 要设置为数据 I/O 口输出，而 RA0 和 RA1 要分别设置成两个比较器的 V-输入，RA2 则作为两个比较器 V+输入。

```

例：      FLAG_REG EQU      0X20
          CLRF      FLAG_REG      ;清标志寄存器
          CLRF      PORTA          ;清 PORTA
          MOVF      CMCON, W
          ANDLW     0XC0           ;
  
```

```

IORWF    FLAG_REG, 1    ;保存 C1OUT 和 C2OUT 到 FLAG_REG 中
MOVLW    0X03
MOVWF    CMCON           ;CM<2:0>=011
BSF       STATUS, RP0
MOVLW    0X07
MOVWF    TRISA           ;RA<2:0>为输入，RA<4:3>为输出
BCF       STATUS, RP0
CALL      DELAY-10       ;延迟 10 μ S
MOVF      CMCON, 1       ;读 CMCON，结束模式设置
BCF       PIR1, CMIF     ;清比较器中断标志位
BSF       STATUS, RP0
BSF       PIE1, CMIE     ;开比较器中断
BCF       STATUS, RP0
BSF       INTCON, PEIE   ;开部件中断
BSF       INTCON, GIE    ;开总中断

```

另外，当上电复位后，CM<0:2>=000，所以比较器工作在模式 0。

§ 5.11.2 比较器工作过程

下图是一个比较器的工作过程：

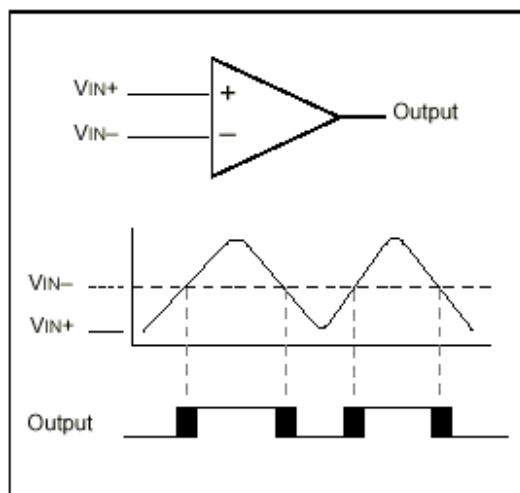


图 5.9 比较器工作过程

当 VIN+端的电压小于 VIN-的电平时，比较器输入为低电平，反之，输出则为高电平。图中阴影部分表示输出状态处于不定，这是因为转换响应延迟造成的。

§ 5.11.3 比较器参考源

比较器参考源有二种：外部参考源/内部参考源。

一、外部参考源

当使用外部参考电压时，两个比较器可以使用同一个参考源，也可以使用不同的参考源。参考电压必须介于 VSS 和 VDD 之间，可以加到比较器的任何一端输入端上。

二、内部参考源

比较器也可以使用芯片内部自己产生的参考电压。只要当比较器选择 CM<2:0>=010 工作模式时，才可以使用内部参考源，这时内部参考电压加到二个比较器的 VIN+输入端上，请参见图 5.8 所示。

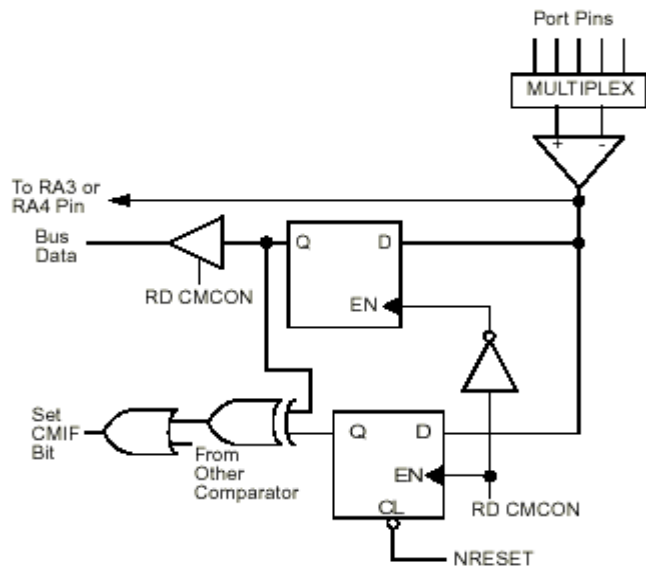
§ 5.11.4 比较器反应时间

比较器的反应时间指的是当选择新的参考电压或新的输入时，比较器的输出重新达到稳定状态所需的最短时间。

§ 5.11.5 比较器输出

两个比较器输出反映在 CMCON 寄存器的 C1OUT 和 C2OUT 两位上，它们可由程序读出，参阅图 5.7。另外比较器的输出也可通过 RA3 或 RA4 直接输出到 I/O 口线上。当 CM<2:0>=110 时，两个比较器的输出直接输出到 RA3 或 RA4，参阅图 5.8。在这种工作模式下，RA3 和 RA4 应通过 TRISA 置为输出态。下图是比较器的输出方块图：

图 5.10 比较器输出框图



如果把 I/O 线设置为模拟输入，则读该 I/O 口线将读回“0”，而如果数字输入口线加上模拟信号，则会根据斯密特触发器特性将其转换为数字电平，并有可能使输入缓冲器消耗额外的电流。

§ 5.11.6 比较器中断

先来看二个和比较器中断有关的寄存器:PIE1 和 PIR1。

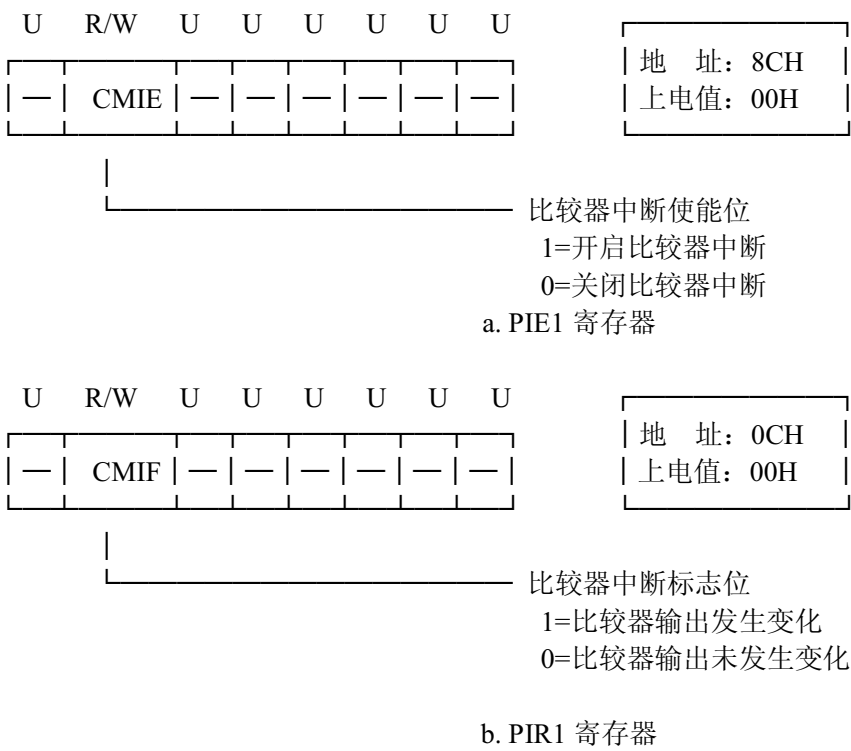


图 5.11 PIR1 和 PIE1 寄存器

当任一个比较器的输出发生电平变化后就会产生中断请求，并置 CMIF (PIR1<6>) =1。如果这时 CMIE (PIE1<6>) =PEIE (INTCON<6>) =GIE (INTCON<7>) =1，则会产生中断，关于中断控制寄存器请参阅 § 5.12。

为了判断是哪一个比较器输出发生了变化，用户程序必须保存 CMCON<7:6>两位的值以便和最新的变化做比较，下面是一段例程。

例： ①保存 COMCON<6:7>

```

MOVLW    0XC0
ANDWF    COMCON, 0
MOVWF    C_Buffer           ;保存 COMCON<6:7>到 C_Buffer

```

②判断变化源

```

MOVF     COMCON, 0
XORWF    C_Buffer, 1
BTFSC    C_Buffer, 6
GOTO     C1_Change          ;C1OUT 变化
BTFSC    C_Buffer, 7
GOTO     C2_Change          ;C2OUT 变化
...
...

```

§ 5.11.7 睡眠中的比较器

如果一个比较器是开启的，那么即使芯片进入睡眠状态后，该比较器仍然是处于工作状态的，其输出的变化仍会发出中断请求，如果比较器开中断，则该中断会把芯片从睡眠中唤醒过来。

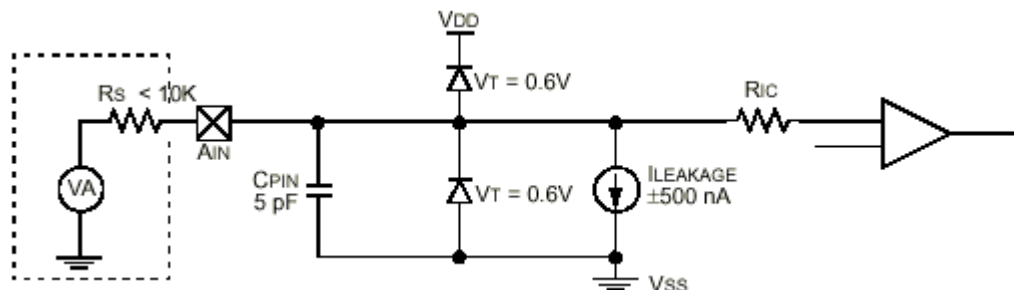
比较器在工作时要消耗一定的电流，所以如果在睡眠中比较器处于工作状态，则芯片的功耗会比一般的睡眠功耗高。如果要使芯片的功耗最低，可以在进入睡眠（执行 Sleep 指令）前关闭比较器模块（CM<2:0>=111）。芯片从睡眠中醒来（WDT 溢出唤醒或中断唤醒）时，寄存器 CMCON 的值不会变化，比较器模块的工作模式也就保持不变。

§ 5.11.8 复位的影响

芯片复位都会使 CMCON 清零，即 CM<2:0>=000。这就使得比较器模块处于复位状态，RA<0:3>皆作为比较器的模拟输入，参阅图 5.10。

§ 5.11.9 模拟输入的连接

下图是模拟输入的简图：



Cpin = 输入电容
VT = 门槛电压
Ilenkage = 漏电流
Ric = 连接电阻

图 5.12 比较器模拟输入图

§ 5.11.10 参考电压模块

图 5.13 比较器参考电压发生电路图

R/W	R/W	R/W	U	R/W	R/W	R/W	R/W
VREN	VROE	VRR	—	VR3	VR2	VR1	VR0

1=VREF 电路关闭, 不消耗电流

图 5.14 VRCON 寄存器

一、设置参考电压

对于高值和低值二个范围的 VREF，每个范围可输出 16 个不同的值：

VRR=1 则 $VREF=(VR<3:0>/24)*VDD$

VRR=0 则 $VREF=VDD/4+(VR<3:0>/32)*VDD$

下面给一个范例：

例：设 VDD=5.0V，需 VREF 输出 1.25V。

MOVLW	0X02	
MOVWF	CMCON	;设置比较模块工作在模式 2
MOVLW	0XA6	
MOVWF	VRCON	;设 VRR=1, VR<3:0>=6
CALL	DELAY10	;10 μ S 延时

二、参考电压的精度

由于参考电压模块的结构，参考电压输出值并不能达到 VSS~VDD 全量程的值，见图 5.13 所示，电阻网络中的三极管使得 VREF 的值不能达到 VSS 或 VDD。所示参考电压不能用来作为精密的参考源。

三、睡眠中的参考电压模块

当芯片从睡眠中被唤醒时寄存器 VRCON 的值不会改变，所以 VREF 仍然会保持不变。如果为了使睡眠中芯片功耗最低，事先应把参考电压模块关闭。

四、复位对参考电压的影响

芯片复位会清零 VCRCON 寄存器，所以 VREF=0、VROE=0，因此会使参考电压模块关闭并使参考电压输出和 RA2 脚断开。另外由于 VRR 位也被清 0，所以参考电压值定在了高值范围。

五、参考电压源的连接

参考电压模块的工作是独立于比较器工作模式的，如果 RA2 脚设置成输入（TRISA<2>=1）并且 VROE=1，那么参考电压的输出就有可能接到 RA2 脚上，至于是否真正连上则要看比较器模块的工作模式。

参考电压若输出到 RA2 或是 RA2 设置为输出且参考电压模块开启（VROE=1）都会增加芯片的功耗。

RA2 可以用做简单的 D/A 输出，当然其驱动能力较小。由于驱动能力小，所以必须加一级缓冲驱动，见下图：

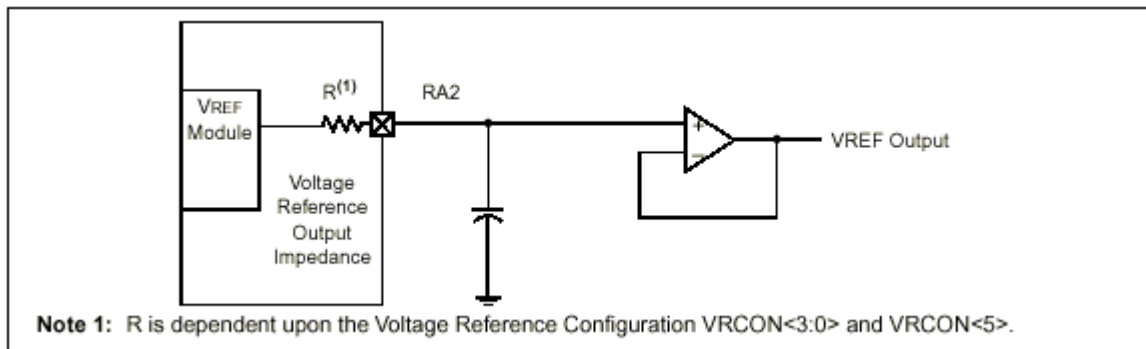


图 5.15 外部驱动参考电压

PIC16C62X 有 4 种中断源：

- 1.INT 外部触发中断；
- 2.TMR0 溢出中断；
- 3.PORTB 电平变化中断；
- 4.比较器中断。

其中断控制寄存器 INTCON 如下：

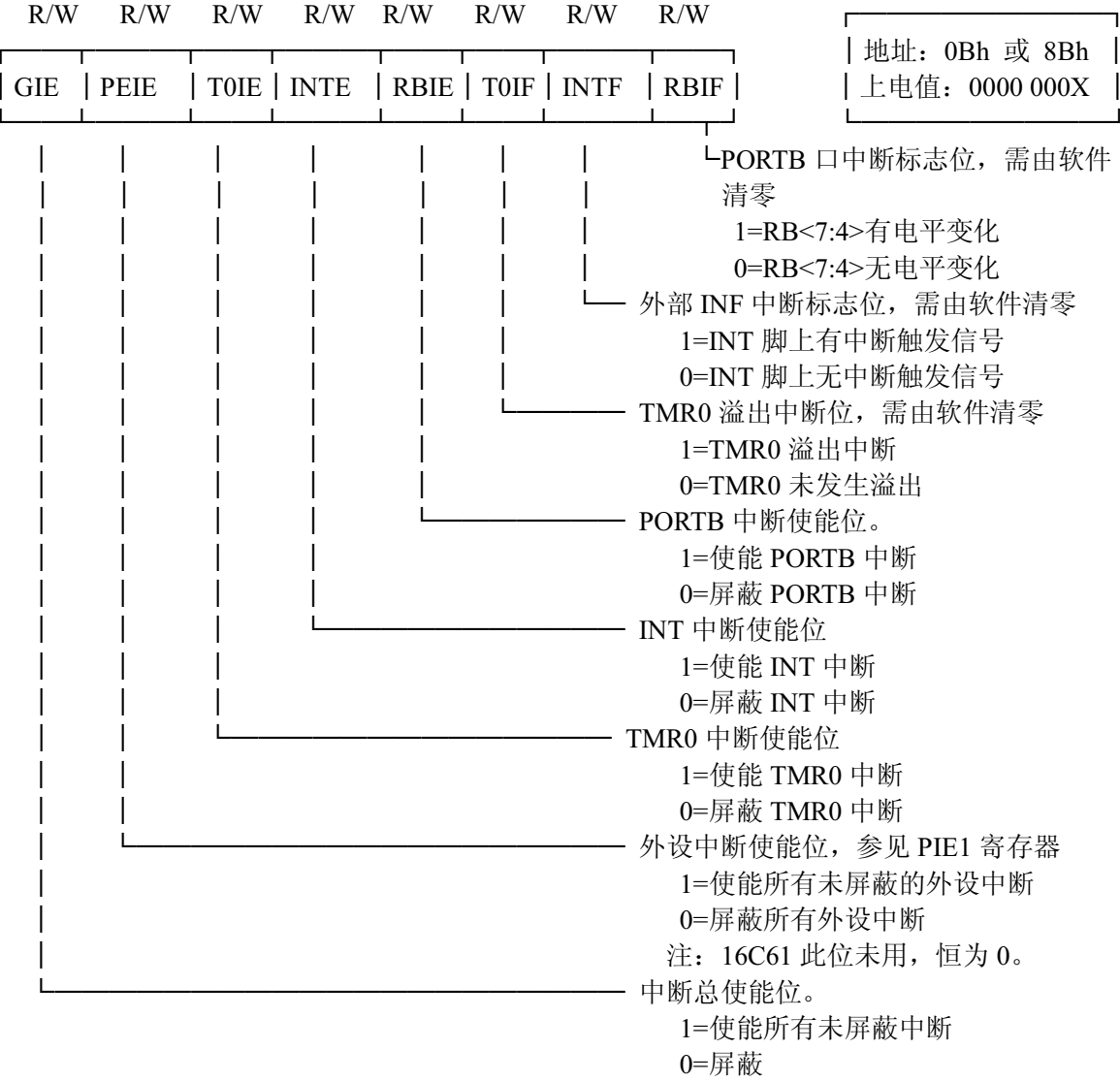


图 5.16 INTCON 寄存器

中断发生逻辑如下：

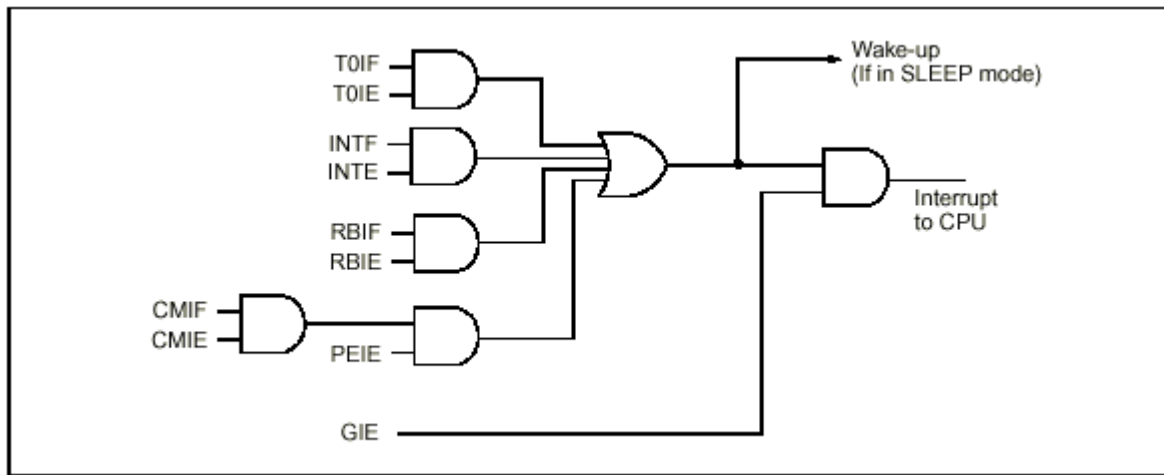


图 5.17 16C62X 中断逻辑

前 3 种中断和其他的 PIC16CXX 完全一样，第 4 种中断请参阅 § 5.11.6。

至于中断处理中的注意事项以及中断现场保护例程等和 PIC16C61 一样，请参阅 § 1.13.4。

§ 5.13 CPU 特性

PIC16C62X 和其他 PIC16CXX 一样具备多种微控制器特性如多种振荡方式选择、看门狗、掉电复位锁定、程序保密位及多种复位方式、上电定时器 PWRTE 和起振延时器 OST 等等，这些内容都和其他 PIC16CXX 一样，请参阅 § 1.13。另外 PIC16C62X 都有内部“掉电复位锁定”电路，可以由用户用烧写器选择使能或关闭，它的标志位则在 PCON 寄存器中，见图 5.5。

第六章 PIC16C55X 单片机

PIC16C55X 目前有如下几种型号：

型 号	振 荡	EPROM	RAM	定时器	中断源	电压范围	I/O	封 装
16C554	DC~20M	512×14	80×8	1	3	2.5-5.5	13	18 脚
16C558	DC~20M	2K×14	128×8	1	3	2.5-5.5	13	18 脚

表 6.1 PIC16C55X 型号功能表

读者可以发现，PIC16C55X 和 PIC16C16X 系列非常相似。的确，把 PIC16C62X 中的电压比较器去掉即是 PIC16C55X 系列。

§ 6.1 主要功能特点

一、高性能 RISC 结构 CPU

- 精简指令集，仅 35 条单字节指令，易学易用
- 除地址分支跳转指令（GOTO、CALL）为双周期指令，其余皆为单周期指令
- 执行速度：DC~200ns
- 3 种中断功能
- 八级硬件堆栈
- 直接、间接、相对三种寻址方式

二、功能部件特性

- 13 根双向可独立编程 I/O 口线
- 高驱动电流，I/O 脚可直接驱动数码管（LED）显示
 - 每个 I/O 引脚最大拉电流 25mA
 - 每个 I/O 引脚最大灌电流 20mA
- 8 位定时器/计数器，带 8 位可编程预分频器，具溢出中断功能

三、微控制器特性

- 内置上电复位电路（POR）
- 上电定时器，保障工作电压的稳定建立
- 振荡定时器，保障振荡的稳定建立
- 自振式看门狗
- 程序保密位，可防程序代码的非法拷贝
- 四种可选振荡方式
 - 低成本阻容：RC
 - 标准晶体/陶瓷：XT
 - 高速晶体：HS
 - 低频晶体：LP
- ID 码

四、CMOS 工艺特性

- 低功耗
 - <2mA @5V, 4MHz
 - <15 μ A @3V, 32KHz
 - <1 μ A @低功耗 Sleep 模式下
- 全静态设计
- 宽工作电压：2.5V~5.5V
- 宽工作温度：
 - 商用级： 0℃~+70℃

- 工业级：-40℃~+85℃
- 汽车级：-40℃~+125℃

PIC16C55X 单片机是 PIC16C54/56/58 的增强型，用户可以看到 PIC16C55X 比 PIC16C54/56/58 增加了很多资源如硬件中断、8 级硬件堆栈等，但外形引脚保持一致，指令也向下兼容，所以用户很容易从 PIC16C5X 升级到 PIC16C55X 上。PIC16C55X 在一个 18 脚的芯片内集成了众多的优秀微处理器的特性，是一种应用广泛的通用型单片机。

§ 6.2 引脚介绍

PIC16C55X 的芯片引脚如下图所示：

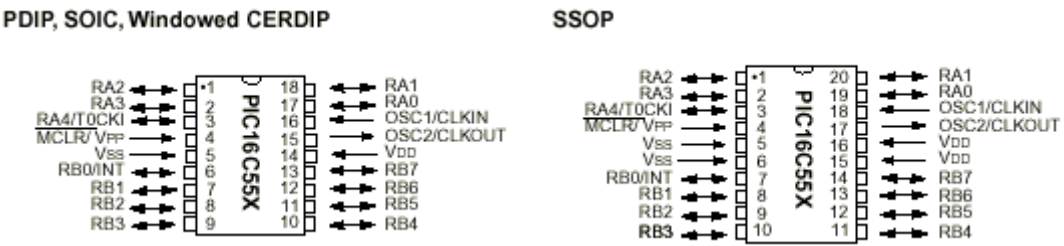


图 6.1 PIC16C55X 引脚

引 脚 名	I/O 特性	电 平	功 能
OSC1/CLKIN	输入	CMOS	振荡输入脚
OSC2/CLKOUT	输出	—	振荡输出脚
MCLR	输入	ST	复位输入脚，低电平有效
RA0 RA1 RA2 RA3 RA4/T0CKI	I/O I/O I/O I/O I/O	ST ST ST ST ST	PORTA 数字 I/O 口，双向可编程 亦可作为 TMR0 外部时钟输入
RB0/INT RB1 RB2 RB3 RB4 RB5 RB6 RB7	 	TTL/ST TTL TTL TTL TTL TTL TTL TTL	PORTB 数字 I/O 口，双向可编程 亦可作为外部中断信号输入脚（此时为 ST 输入） 具有电平变化中断功能 具电平变化中断功能 具电平变化中断功能 具电平变化中断功能
VSS	—	—	地
VDD	—	—	电源

ST：斯密特输入

表 6.2 PIC16C55X 引脚功能表

§ 6.3 内部结构

PIC16C55X 内部采用独立分离的 8 位数据总线和 14 位指令总线的“哈佛”结构，它是一种“精简指令集”(RISC) 的 CPU 设计，所以可以达到很高的运行速度。8 位的算术逻辑单元 ALU 可以完成加减、移位和各种布尔逻辑运算，另外它还集成了众多的功能模块如 I/O 口、定时器、上电复位电路、看门狗电路、上电/起振延时器等。

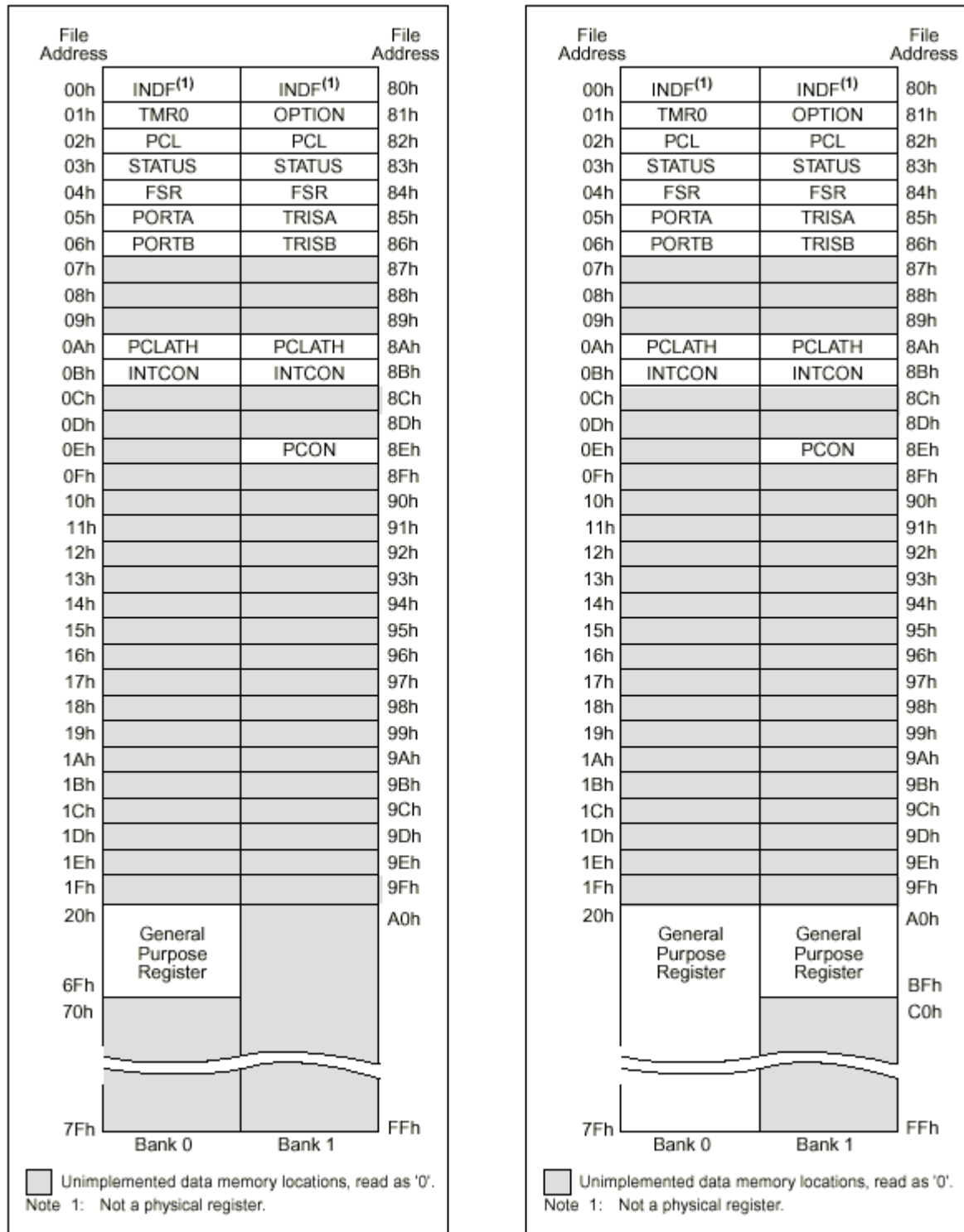
在 PIC16C55X 片内带有 0.5K~2K 的 14 位宽程序存储器（ROM）、80~128 个 8 位的数据寄存器（RAM），所有特殊寄存器包括程序计数器、I/O 寄存器等都直接映射到 RAM 单元中，所以程序编码非常简洁高效。

任何超出程序存储器实际空间的寻址都将是物理空间上的回绕而已。

PIC16C55X 也有独立 8 级硬件堆栈，不占用程序存储器空间。

§ 6.5 数据寄存器

PIC16C55X 的数据寄存器分为 2 个体: Bank0 和 Bank1，如下图:



a. PIC16C554

b. PIC16C558

图 6.4 PIC16C55X 寄存器结构

体（Bank）的选择由 STATUS 寄存器中的 RP0:RP1 两位来决定，参阅 § 1.7.2 有关 STATUS 寄存器的描述。寄存器从功能上分有特殊寄存器和通用寄存器两种，下表是 PIC16C55X 的特殊寄存器。

地 址	名 称	功 能 说 明				上电复位值	其他复位值
Bank0（0 体）							
00h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000 0000	0000 0000
01h	TMR0	TIMER0 模块寄存器				xxxx xxxx	uuuu uuuu
02h	PCL	程序计数器 PC 的低 8 位				0000 0000	0000 0000
03h	STATUS	状态寄存器				0001 1xxx	000q quuu
04h	FSR	间接寻址寄存器				xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	—	PORTA 口寄存器	---x xxxx	---u uuuu
06h	PORTB	PORTB 寄存器				xxxx xxxx	uuuu uuuu
07h	—	——				—	—
08h	—	——				—	—
09h	—	——				—	—
0Ah	PCLATH	—	—	—	PC 高 5 位之写入器	---0 0000	---0 0000
0Bh	INTCON	中断控制寄存器				0-00 000x	0-00 000u
0Ch~1Fh	—	——				—	—
Bank1（1 体）							
80h	INDF	间接寻址逻辑寄存器（物理上不存在）				0000 0000	0000 0000
81h	OPTION	系统功能定义寄存器				1111 1111	1111 1111
82h	PCL	程序计数器 PC 的低 8 位				0000 0000	0000 0000
83h	STATUS	状态寄存器				0001 1xxx	000q quuu
84h	FSR	间接寻址寄存器				xxxx xxxx	uuuu uuuu
85h	TRISA	—	—	—	PORTA 方向寄存器	---1 1111	---1 1111
86h	TRISB	PORTB 方向寄存器				1111 1111	1111 1111
87h	—	——				—	—
88h	—	——				—	—
89h	—	——				—	—
8Ah	PCLATH	—	—	—	PC 高 5 位之写入器	---0 0000	---0 0000
8Bh	INTCON	中断控制寄存器				0000 000X	0000 000u
8Ch~8Dh	—	——				—	—
8Eh	PCON	上电复位和掉电复位标志位寄存器				---- --0-	---- --0-
8Fh~9Fh	—	——				—	—

注：X=不定， u=不变， q=取决于某条件， -=未用（读为 0）

表 6.3 PIC16C55X 特殊功能寄存器

从本节读者可以看到，PIC16C55X 的寄存器结构上和 PIC16C62X 完全一样，只是比 PIC16C62X 少了几个有关模拟比较器的特殊寄存器而已。还有一点，即 PIC16C55X 芯片没有“掉电复位锁定”，所以 PIC16C55X 的 PCON 寄存器中没有 BO 控制位（参阅 § 5.7），见下图。

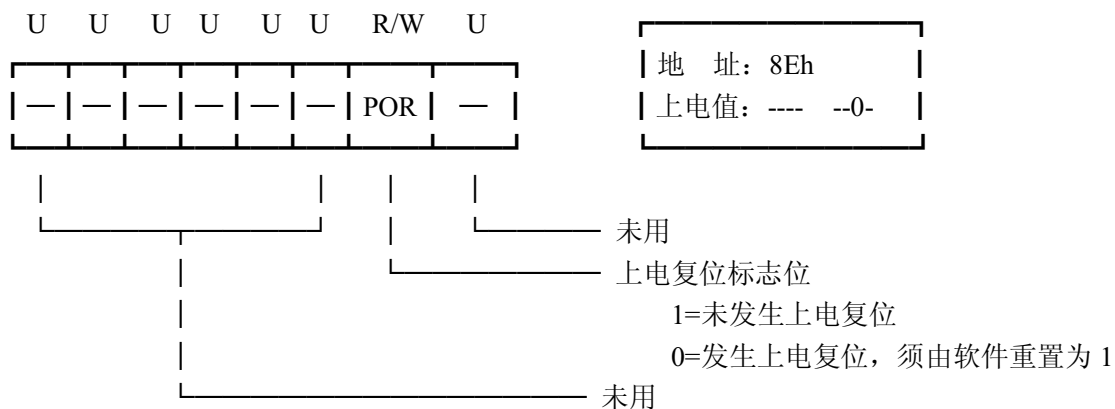


图 6.5 PCON 寄存器

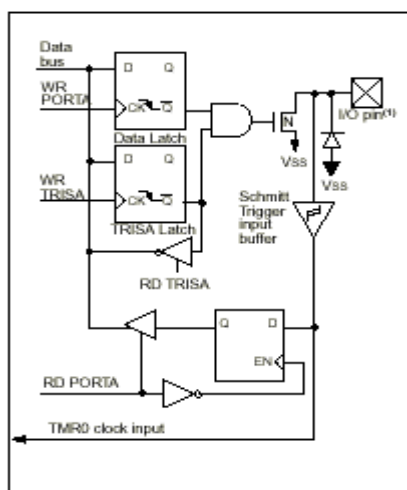
其他的特殊寄存器和 PIC16CXXX 相同, 不再赘述。

§ 6.6 I/O 口

PIC16C55X 有二个 I/O 口, PORTA 和 PORTB。

PORTA 是 5 位 (RA<4:0>) 的 I/O 口, 其中 RA4 具开极输出和斯密特触发输入, 并和 T0CKI 端复用, 见下图:

图 6.6 RA4 结构



RA<3:0>具斯密特输入和 CMOS 驱动输出, 见下图:

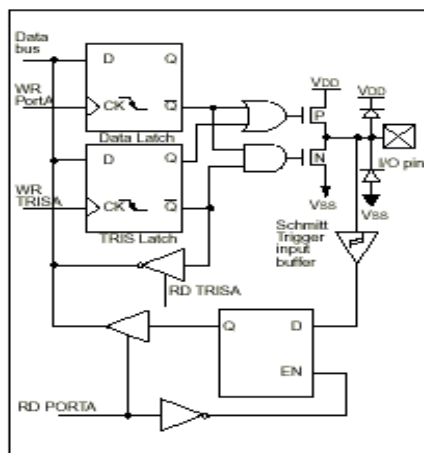


图 6.7 RA3:RA0 结构

至于 PORTB, 则和其他 PIC16CXXX 完全一样, 请参阅 § 1.8.2。

§ 6.7 计数器/定时器

PIC16C55X 有一个计数器/定时器 TIMER0，和其他 PIC16CXXX 完全一样，请参阅 § 1.9。

§ 6.8 看门狗

PIC16C55X 之看门狗 WDT 和其他 PIC16CXXX 完全一样，请参阅 § 1.13.5。

§ 6.9 中 断

PIC16C55X 的中断和 PIC16C61 完全一样，请参阅图 1.9 和 § 1.13.4 中有关 PIC16C61 中断的章节。

§ 6.10 CPU 特性

PIC16C55X 和其他 PIC 单片机一样具备许多微处理器特性，如看门狗 WDT、4 种振荡选择、多种复位方式、程序保密位以及上电延时定时器 PWRT 和振荡起振定时器 OST 等等，都和 PIC16C61 等完全一样，请参阅 § 1.13 有关描述。

第七章 PIC16CXX 指令系统

PIC16CXX 每条指令长 14 位，指令由操作码和操作数组成。PIC16CXX 共有 35 条指令，由操作对象可分为：

1. 面向字节操作类
2. 面向位操作类
3. 常数操作和控制类操作。

全部指令如图 7.1 所示。

面向字节操作类指令		(11-6)		(5)	(4-0)	
		OPCODE		d	f (FILE#)	
二进制代码	HEX	名称	助记符, 操作数	操作	状态影响	注
00 0000 0xx0 0000 0000		空操作	NOP		无	
00 0000 1fff ffff 008f		W 送到 f	MOVWF f	W→f	无	3
00 0001 0xxx xxxx 0100		W 清零	CLRW -	0→W	Z	
00 0001 1fff ffff 018f		f 清零	CLRF f	0→f	Z	3
00 0010 dfff ffff 02ff		f 减去 W	SUBWF f, d	f-W→d[f+W+1→d]	C, DC, Z	2, 3
00 0011 dfff ffff 03ff		f 递减	DECF f, d	f-1→d	Z	2, 3
00 0100 dfff ffff 04ff		W 和 f 做 或运算	IORWF f, d	W∨f→d	Z	2, 3
00 0101 dfff ffff 05ff		W 和 f 做 与运算	ANDWF f, d	W&f→d	Z	2, 3
00 0110 dfff ffff 06ff		W 和 f 做 异或运算	XORWF f, d	W⊙f→d	Z	2, 3
00 0111 dfff ffff 07ff		W 加 f	ADDWF f, d	W+f→d	C, DC, Z	2, 3
00 1000 dfff ffff 08ff		传送 f 到 d	MOVF f, d	f→d	Z	2, 3
00 1001 dfff ffff 09ff		f 取补	COMF f, d	f→d	Z	2, 3
00 1010 dfff ffff 0Aff		f 递增	INCF f, d	f+1→d	Z	2, 3
00 1011 dfff ffff 0Bff		f 递减, 为 0 则跳	DECFSZ f, d	f-1→d, skip if zero	无	2, 3
00 1100 dfff ffff 0Cff		f 循环右移	RRF f, d	f(n)→d(n-1), f(0)→C, C→d(7)	C	2, 3
00 1101 dfff ffff 0Dff		f 循环左移	RLF f, d	f(n)→d(n+1), f(7)→C, C→d(0)	C	2, 3
00 1110 dfff ffff 0Eff		f 半字节交 换	SWAPF f, d	f(0-3)↔f(4-7)→d	无	2, 3
00 1111 dfff ffff 0Fff		f 递增, 为 0 则跳	INCSZ f, d	f+1→d, skip if zero	无	2, 3
面向位操作类指令		(11-8)		(7-5)	(4-0)	
		OPCODE		b (BIT#)	f (FILE#)	
二进制代码	HEX	名称	助记符, 操作数	操作	状态影响	注
01 00bb bfff ffff 1bff		清除 f 的 位 b	BCF f, b	0→f(b)	无	2, 3
01 01bb bfff ffff 1bff		设置 f 的 位 b	BSF f, b	1→f(b)	无	2, 3
01 10bb bfff ffff 1bff		测试 f 的 位 b, 为 0 则跳	BTFSC f, b	Test bit(b) in file(f):Skip if clear	无	
01 11bb bfff ffff 1bff		测试 f 的 位 b, 为 1 则跳	BTFSS f, b	Test bit(b) in file(f):Skip if clear	无	
常数操作和控制类指令		(11-8)		(7-0)		
		OPCODE		k (LITERAL)		
二进制代码	HEX	名称	助记符, 操作数	操作	状态影响	注
00 0000 0110 0010 0062		写 OPTION 寄存器	OPTION -	W→OPTION register	无	1
00 0000 0110 0011 0063		进入睡眠 状态	SLEEP -	0→WDT, stop oscillator	TO, PD	

00 0000 0110 0100 0064	清除 WDT 计时器	CLRWDT -	0 → WDT(and prescaler , if assigned)	TO, PD	
00 0000 0110 0fff 006f	设置 I/O 状态	TRIS f	W→I/O control register f	无	1
11 01xx kkkk kkkk 34kk	子程序带参数返回	RETLW k	k→W, TOS→PC	无	
10 0kkk kkkk kkkk 2kkk	调用子程序	CALL k	PC+1→TOS, K→PC<10:0>, PCLATH<4:3>→PC<12:11>	无	
10 1kkk kkkk kkkk 2kkk	跳转(K 为 11 位)	GOTO k	k→PC<10:0>, PCLATH<4:3>→PC<12:11>	无	
11 00xx kkkk kkkk 30kk	常数置入 W	MOVLW k	k→W	无	
11 1000 kkkk kkkk 38kk	常数和 W 做或运算	IORLW k	k∨W→W	Z	
11 1001 kkkk kkkk 39kk	常数和 W 做与运算	ANDLW k	k&W→W	Z	
11 1010 kkkk kkkk 3Akk	常数和 W 做异或运算	XORLW k	k⊕W→W	Z	
11 110x kkkk kkkk 3Ckk		SUBLW k	K-W→W	无	
11 111x kkkk kkkk 3Ekk		ADDLW k	K+W→W	C, DC, Z	
00 0000 0000 1000 0008		RETURN -	TOS→PC	无	
00 0000 0000 1001 0009		RETFIE -	TOS→PC, '1'→GIE	无	

表 7.1 PIC16CXX 指令表

注：（1）PIC16CXX 有 35 条基本指令及二条附加指令，即 TRIS 和 OPTION。在 PIC16CXX 中 TRIS 和 OPTION 寄存器是直接可寻址的，所以不必用这二条指令，保留它们只是为了和 PIC16C5X 向上兼容，即便于 PIC16C5X 代码向 PIC16CXX 的移植，见下节指令详介。

（2）对 I/O 寄存器操作的指令，如“MOVF 6, 1”，使用的 F6（RB 口）的值是 RB 口脚上的状态值，而非输出锁存器的值。

（3）“f”代表寄存器，“d”代表目的寄存器，当 d=0，操作结果放入 W 寄存器，d=1，则结果放入 f 寄存器。“b”代表位（0~7），K 代表一个 8 位或 11 位的常数。

§ 7.1 PIC16CXX 指令寻址方式

PIC16XX 单片机寻址方式根据操作数的来源，要分为寄存器间接寻址、立即数寻址、直接寻址和位寻址四种。

一、寄存器间接寻址

这种寻址方式通过寄存器 F0、F4 来实现。实际的寄存器地址放在 F4 的低 5 位中，通过 F0 来进行间接寻址。

```
例：    MOVLW    05H        ;    W=5
        MOVWF    4          ;    W(=5)→F4
        MOVLW    55H        ;    W=55H
        MOVWF    0          ;    W(=55H)→F5
```

上面这段程序把 55H 送入 F5 寄存器。间址寻址方式主要用于编写查表、写表程序，非常方便。请参考程序设计技巧。

二、立即数寻址

这种方式就是操作数为立即数，可直接从指令中获取。

```
例：    MOVLW    16H        ;    16H →W
```

三、直接寻址

这种方式是对任何一寄存器直接寻址访问。

```
例：    MOVWF    8          ; W→F8 寄存器
```

MOVF 8, W ; F8→W

四、位寻址

这种寻址方式是对寄存器中的任一位 (bit) 进行操作。

例: BSF 11, 0 ; 把 F11 的第 0 位置为 “1”。

§ 7.2 PIC16CXX 指令详介

1. 立即数加法指令

格式: ADDLW K

代码:

11	111X	KKKK	KKKK
----	------	------	------

指令周期: 1

操作: W+K→W

影响状态位: C, DC, Z

说明: W 寄存器的内容与 8 位立即数 K 相加, 结果放入 W 寄存器。

例: ADDLW 60H ; W+60H→W

2. 寄存器加法指令

格式: ADDWF f, d

代码:

00	0111	dfff	ffff
----	------	------	------

指令周期: 1

操作: W+f→d

影响状态位: C, DC, Z

说明: W 寄存器内容和 f 寄存器内容相加, 结果存入 f (d=1) 或 W (d=0)

例: ADDWF 8, 0 ; F8+W→W

 ADDWF 8, 1 ; F8+W→F8

3. 立即数逻辑“与”指令

格式: ANDLW K

代码:

11	1001	KKKK	KKKK
----	------	------	------

指令周期: 1

操作: W∧K→K

影响状态位: Z

说明: W 和 8 位立即数 K 相 “与”, 结果存入 W。

例: ANDLW 55H, 0 ; W∧55H→W

4. 寄存器逻辑“与”指令

格式: ANDWF f, d

代码:

00	0101	dfff	ffff
----	------	------	------

指令周期: 1

操作: $W \wedge f \rightarrow d$

影响状态位: Z

说明: W 寄存器内容和 f 寄存器内容相“与”, 结果放入 W (d=0) 或 f (d=1)

例: ANDWF 8, 0 ; $W \wedge F8 \rightarrow W$
ANDWF 8, 1 ; $W \wedge F8 \rightarrow F8$

5. 位清零指令

格式: BCF f, b

代码:

01	00bb	bfff	ffff
----	------	------	------

指令周期: 1

操作: $0 \rightarrow f(b)$

影响状态位: 无

说明: 将 f 寄存器的 b 位清为 0。

例: BCF 8, 0 ; 将 F8 的 bit0 清为 0。
BCF 8, 2 ; 将 F8 的 bit2 清为 0。

6. 位置“1”指令

格式: BSF f, b

代码:

01	01bb	bfff	ffff
----	------	------	------

指令周期: 1

操作: $1 \rightarrow f(b)$

影响状态位: 无

说明: 将 f 寄存器的 b 位置为 1。

例: BSF 5, 1 ; 将 F5 (RA 口) 的 bit1 (RA1) 置为“1”。

7. 位测试, 为“0”则跳指令

格式: BTFSC f, b

代码:

01	10bb	bfff	ffff
----	------	------	------

指令周期: 1 或 2 (产生跳转为 2)

操作: 如果 $f(b) = 0$ 则跳 ($PC+1 \rightarrow PC$)

影响状态位: 无

说明: 测试 f 寄存器第 b 位, 如 $f(b) = 0$ 则跳过下一条指令 ($PC+1 \rightarrow PC$), 否则顺序执行下去。

例:

┌	BTFSC	8, 2	; 测试 F8 的 bit2
	跳	MOVF 5, 0	; bit2=1, 执行这条指令
└	INCF	9, 1	; bit2=0, 则跳到这条指令。

8. 位测试, 为“1”则跳指令

格式: BTFSS f, b

代码:

01	11bb	bfff	ffff
----	------	------	------

指令周期: 1 或 2 (产生跳转为 2)

操作： 如果 $f(b) = 1$ 则跳 ($PC+1 \rightarrow PC$)

影响状态位： 无

说明： 测试 f 寄存器的第 b 位，如位 $f(b) = 1$ 则跳过下一条指令，否则顺序执行下去。

例：

	└─ BTFSS	8, 2	;	测 F8 的 bit2	
bit2=1	跳	MOVF	5, 0	;	bit2=0, 执行这条指令
	└─ INCF	9, 1	;	bit2=1, 跳到这条指令。	

9. 子程序调用指令

格式： CALL K

代码：

01	0KKK	KKKK	KKKK
----	------	------	------

指令周期： 2

操作： $PC+1 \rightarrow$ 堆栈, $K \rightarrow PC(9:0, 0)$ PCLATH (4, 3) $\rightarrow PC(12, 11)$

影响状态位： 无

说明： 子程序调用。首先将 PC 加 1 推入堆栈，然后将常数 K (11 位) $\rightarrow PC(10, 0)$ ，同时 $PCLATH(4, 3) \rightarrow PC(12, 11)$ ，形成 PC =子程序入口地址。

例：

	CALL	DELAY	;	调用子程序	
	:				
	:				
DELAY	MOVLW	80H	└─	;	子程序
	:				
	:				
	RETLW	0	└─		

10. 寄存器清零指令

格式： CLRF f

代码：

00	0001	1fff	ffff
----	------	------	------

指令周期： 1

操作： $0 \rightarrow f$

影响状态位： Z

说明： f 寄存器被清为全零，状态位 $Z=1$ 。

例： CLRF 9 ; F9=00H

11. W 清零指令

格式： CLRW

代码：

00	0001	0XXX	XXXX
----	------	------	------

指令周期： 1

操作： $0 \rightarrow W$

影响状态位： Z

说明： W 寄存器清全零，状态位 $Z=1$ 。

例： CLRW ; W=00H

12. 看门狗计数器清零指令

格式： CLRWDW

代码：

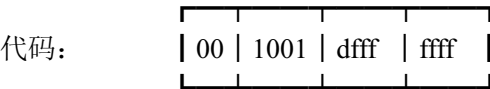
00	0000	0110	0100
----	------	------	------



指令周期： 1
 操作： 00→WDT， 0→WDT 分频器
 影响状态位： 1→TO， 1→PD
 说明： 清零 WDT 计数器，同时 WDT 分频器（如果欲分频倍数分配给 WDT）也清为零。指令执行后状态位 TO=1， PD=1。

13. 寄存器取反指令

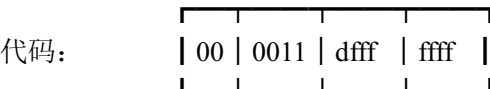
格式： COMF f, d



指令周期： 1
 操作： f→d
 影响状态位： Z
 说明： f 寄存器内容取反后送入 W（d=0）或 f 本身（d=1）。
 例： COMF 8, 0 ; F8→W
 COMF 8, 1 ; F8→F8

14. 寄存器减 1 指令

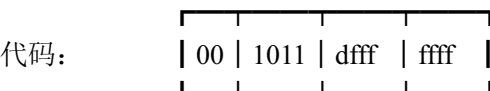
格式： DECF f, d



指令周期： 1
 操作： f-1→d
 影响状态位： Z
 说明： F 寄存器内容减 1 后送入 W（d=0）或 f（d=1）。

15. 寄存器减 1，结果为零则跳指令

格式： DECFSZ f, d



指令周期： 1 或 29 跳则为 2
 操作： f-1→d，结果为零则跳（PC+1→PC）
 影响状态位： 无
 说明： f 寄存器内容减 1 存入 W（d=0）或 f（d=1）。如结果为 0 则跳过下一条指令（PC+1→PC）。
 例： ┌── DECFSZ 10, 1 ; F10-1→F10
 F10-1=0 | MOVLW 55H
 └── MOVF 12, 0

16. 无条件跳转指令

格式： GOTO K



指令周期： 2
 操作： K→PC（10, 0），PCLATH（4, 3）→PC（12, 11）
 影响状态位： 无

说明： 无条件跳转，11 位常数 K→PC (10, 0) PCLATH (4, 3) →PC (12, 11)。

例： ┌→LOOP MOVLW 55H
 | :
 | :
 └──────── GOTO LOOP ; 无条件跳转

17. 寄存器加 1 指令

格式： INCF f, d

代码： ┌───┬───┬───┬───┐
 | 00 | 1010 | dfff | ffff |
 └───┬───┬───┬───┘

指令周期： 1

操作： f+1→d

影响状态位： Z

说明： f 寄存器内容加 1 后送 W (d=0) 或 f 本身 (d=1)。

18. 寄存器加 1，结果为 0 则跳

格式： INCFSZ f, d

代码： ┌───┬───┬───┬───┐
 | 00 | 1111 | dfff | ffff |
 └───┬───┬───┬───┘

指令周期： 1 或 2 (跳转为 2)

操作： f+1→d, 结果为 0 则跳 (PC+1→PC)

影响状态位： 无

说明： f 寄存器内容加 1，结果存入 W (d=0) 或 f (d=1)，如果结果为 0，则 PC+1→PC，跳过下一条指令。

例： LOOP INCFSZ 8, 1 ┌
 GOTO LOOP | F8=0
 MOVWF 9 └

19. 常数“或”指令

格式： IORLW K

代码： ┌───┬───┬───┬───┐
 | 11 | 1000 | KKKK | KKKK |
 └───┬───┬───┬───┘

指令周期： 1

操作： W∨K→W

影响状态位： Z

说明： W 寄存器内容和 8 位立即数 K 做逻辑或，结果放入 W。

例： IORLW 55H ; W∨55H→W

20. 寄存器“或”指令

格式： IORWF f, d

代码： ┌───┬───┬───┬───┐
 | 00 | 0100 | dfff | ffff |
 └───┬───┬───┬───┘

指令周期： 1

操作： W∨f→d

影响状态位： Z

说明: W 寄存器内容和 f 寄存器内容做逻辑或运算, 结果放入 W (d=0) 或 f (d=1)。

例: IORWF 10, 0 ; W∨F10→W
 IORWF 10, 1 ; W∨F10→F10

21. 常数传送指令

格式: MOVLW K

代码:

11	00XX	KKKK	KKKK
----	------	------	------

指令周期: 1

操作: K→W

影响状态位: 无

说明: 8 位立即数送入 W 寄存器。

22. f 寄存器传送指令

格式: MOVF f, d

代码:

00	1000	dfff	ffff
----	------	------	------

指令周期: 1

操作: f→d

影响状态位: Z

说明: 将 f 寄存器内容传至 W (d=0) 或 f 本身 (d=1)。这条指令会影响状态位 Z, 所以经常用来判断寄存器是否为 0。见下例。

例:

```

                                MOVF      10, 1      ; F10→F10
F10=0 ┌── BTFSS      3, 2      ; 判断 Z 状态位
      │   CLRW
      └─┐ ADDLW      55H
```

23. W 寄存器传送指令

格式: MOVWF f

代码:

00	0000	1fff	ffff
----	------	------	------

指令周期: 1

操作: W→f

影响状态位: 无

说明: W 寄存器内容传送至 f, W 保持不变。

例: MOVLW 55H ; 55H→W
 MOVWF F10 ; W(55H)→F10

24. 空操作指令

格式: NOP

代码:

00	0000	0XX0	0000
----	------	------	------

指令周期: 1

操作: 空操作

影响状态位: 无

说明： 不做任何操作，只使 PC 加 1 常用来起延时作用。

25. OPTION 寄存器赋值指令

格式： OPTION

代码：

00	0000	0110	0010
----	------	------	------

指令周期： 1

操作： W→OPTION 寄存器

影响状态位： 无

说明： 将 W 寄存器内容载入 OPTION 寄存器。详见 OPTION 寄存器详介。由于在 PIC16CXX 中 OPTION 寄存器是直接可读/写的，这点和 PIC16C5X 不同。所以在 PIC16CXX 中，用户不必使用 OPTION 指令。而可以直接读/写 OPTION 寄存器（81H）。保留它只是为了和 PIC16C5X 指令兼容，使得为 PIC16C5X 写的代码容易移植到 PIC16CXX 中。在 PIC16CXX 中，可以这样置 OPTION 寄存器。

例： BSF STATUS, RP0
 MOVLW OP_DATA
 MOVWF OPTION
 BCF STATUS, RP0

26. 中断返回指令

格式： RETFIE

代码：

00	0000	0000	1001
----	------	------	------

指令周期： 2

操作： 栈顶→PC, 1→GIE 位

影响状态位： 无

说明： 中断服务子程序返回指令。栈顶为返回地址，压入 PC。同时全体中断允许位 GIE（在 INTCON 中）置为“1”。

27. 子程序带参数返回指令

格式： RETLW K

代码：

11	01XX	KKKK	KKKK
----	------	------	------

指令周期： 2

操作： 栈顶→PC, K→W

影响状态位： 无

说明： 子程序返回，栈顶内容→PC 同时 8 位常数 K→W，返回到子程序调用处。

28. 子程序不带参数返回

格式： RETURN

代码：

00	0000	0000	1000
----	------	------	------

指令周期： 2

操作： 栈顶→PC

影响状态位： 无

说明： 子程序返回，栈顶内容→PC，返回到子程序调用处。注意返回不带参数，见上条指令。

29. 寄存器带 C 循环左移指令

格式: RLF f, d

代码:

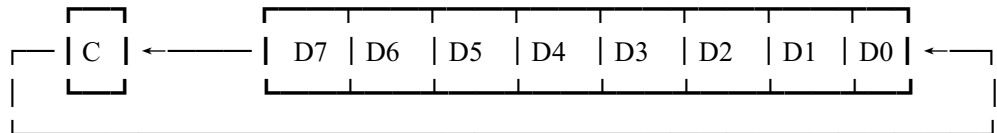
00	1101	dfff	ffff
----	------	------	------

指令周期: 1

操作: $f(n) \rightarrow d(n+1)$, $f(7) \rightarrow C$, $C \rightarrow d(0)$

影响状态位: C

说明: 将 f 寄存器带 C 循环左移, 结果存入 W (d=0) 或 f (d=1), 如下图所示。



30. 寄存器带 C 循环右移指令

格式: RRF f, d

代码:

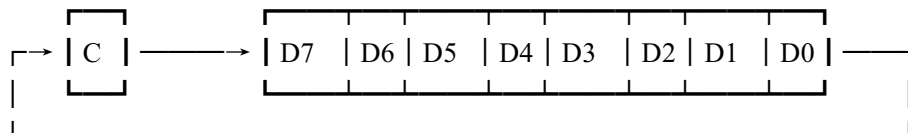
00	1100	dfff	ffff
----	------	------	------

指令周期: 1

操作: $f(n) \rightarrow d(n-1)$, $f(0) \rightarrow C$, $C \rightarrow d(7)$

影响状态位: C

说明: f 寄存器带 C 循环右移, 结果存入 W (d=0) 或 f (d=1), 如下图所示。



31. 进入低功耗睡眠指令

格式: SLEEP

代码:

00	0000	0110	0011
----	------	------	------

指令周期: 1

操作: 0→PD, 1→TO 00→WDT, 0→WDT 预分频器

影响状态位: TO, PD

说明: 执行本指令后芯片进入低功耗睡眠模式, 芯片 OSC1 振荡停止。

32. 常数减法指令

格式: SUBLW K

代码:

11	110X	KKKK	KKKK
----	------	------	------

指令周期: 1

操作: $K \rightarrow W$

影响状态位: C, DC, Z

说明: 8 位常数 K 减 W 寄存器内容, 结果放入 W。PIC 的减法运算通过做补码加法来实现。

例: MOVLW 01H ; 1→W

`SUBLW 02H` ; $2-W=2-1=1 \rightarrow W$
 ; $C=1$, 结果为正。
 例: `MOVLW 2` ; $2 \rightarrow W$
`SUBLW 1` ; $1-W=1-2=-1=FFH \rightarrow W$
 ; $C=0$, 结果为负

33. 寄存器减法指令

格式: `SUBWF f, d`

代码:

00	0010	dfff	ffff
----	------	------	------

指令周期: 1

操作: $f-W \rightarrow d$

影响状态位: C, DC, Z

说明: f 寄存器减 W 寄存器结果放入 W (d=0) 或 f (d=1)。

例: `CLRF 10` ; $F10=0$
`MOVLW 1` ; $1 \rightarrow W$
`SUBLW 10, 1` ; $F10-W=0-1=FFH \rightarrow F10$
 ; $C=0$, 结果为负。

例: `MOVLW 1`
`MOVWF 10` ; $F10=1$
`CLRWF` ; $W=0$
`SUBLW 10, 0` ; $F10-W=1-0=1 \rightarrow W$
 ; $C=1$, 结果为正

34. 寄存器半字节交换指令

格式: `SWAPF f, d`

代码:

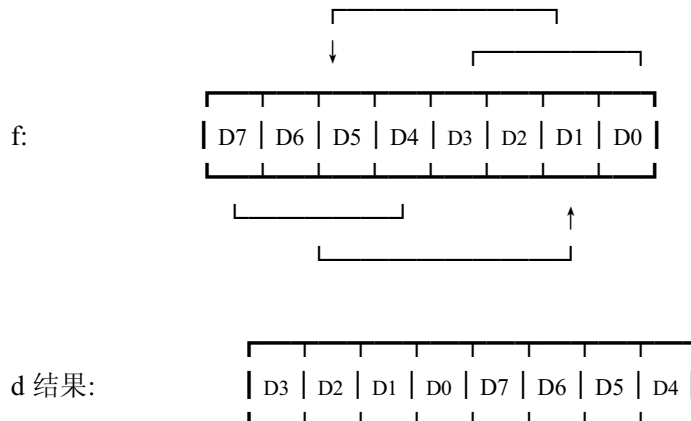
00	1100	dfff	ffff
----	------	------	------

指令周期: 1

操作: $f(0, 3) \rightarrow d(4, 7), f(4, 7) \rightarrow d(0, 3)$

影响状态位: 无

说明: f 寄存器高 4 位和低 4 位交换位置后结果存入 W(d=0)或 f(d=1), 见下图:



例: `MOVLW 56H`
`MOVWF 8` ; $56H \rightarrow F8$
`SWAPF 8, 1` ; $F8=65H$

35. 设置 I/O 方向控制寄存器指令

格式: TRIS f

代码:

00	0000	0110	0fff
----	------	------	------

指令周期: 1

操作: W→I/O 控制寄存器 TRISf (f=5, 6)

影响状态位: 无

说明: 由于在 PIC16CXX 中, TRIS (85H, 86H) 寄存器是直接可读/写的, 所以用户不必使用这条指令来设置 I/O 控制寄存器, 保留它只是为了和 PIC16C5X 向上兼容, 使为 PIC16C5X 写的代码易移植到 PIC16CXX 中。参考 OPTION 指令。在 PIC16CXX 中可以这样设 TRIS 寄存器。

例: BSF STATUS, RP0
 MOVLW TRISA_DA
 MOVWF TRISA
 BCF STATUS, RP0

36. 立即数“异或”指令

格式: XORLW K

代码:

11	1010	KKKK	KKKK
----	------	------	------

指令周期: 1

操作: W○K→W

影响状态位: Z

说明: W 寄存器和 8 位常数 K 做“异或”运算后存入 W。

37. 寄存器“异或”指令

格式: XORWF f, d

代码:

00	0110	dff	fff
----	------	-----	-----

指令周期: 1

操作: W○f→d

影响状态位: Z

说明: W 寄存器内容和 f 寄存器内容做“异或”运算, 结果存入 W (d=0) 或 f (d=1)。

例: MOVLW 55H
 MOVWF 10 ; F10=55H
 MOVLW AAH ; W=AAH
 XORWF f, 1 ; F10=FFH

§ 7.3 特殊指令助记符

PIC16CXX 的一些指令还可以用容易记忆的助记符来表示。汇编程序 MPASM 也可以认识这些助记符, 在汇编时会将其译成相应的 PIC16CXX 基本指令。

例如指令“BCF 3, 0”(清零 C)也可以写成 CLRC; “BSF 3, 0”(置 C=1)也可写成 SETC 等。

表 7.2 列出了这些助记符及其相对应的 PIC16CXX 指令。

二进制指令代码 (Hex)	名 称	助记符号	相对运算	状态影响
0100 0000 0011 (403)	清除 C 标号	CLRC	BCF 3, 0	—

0101 0000 0011 (503)	设置 C 标号	SETC	BSF 3, 0	—
0100 0010 0011 (423)	清除辅助进位标号	CLRDC	BCF 3, 1	—
0101 0010 0011 (523)	设置辅助进位标号	SETDC	BSF 3, 1	—
0100 0100 0011 (443)	清除 0 标号	CLRZ	BCF 3, 2	—
0101 0100 0011 (543)	设置 0 标号	SETZ	BSF 3, 2	—
0111 0000 0011 (703)	进位则跳	SKPC	BTFSS 3, 0	—
0110 0000 0011 (603)	无进位则跳	SKPNC	BTFSC 3, 0	—
0111 0010 0011 (723)	辅助进位为 1 则跳	SKPDC	BTFSS 3, 1	—
0110 0010 0011 (623)	辅助进位为 0 则跳	SKPNDC	BTFSC 3, 1	—
0111 0100 0011 (743)	不为 0 则跳	SKPZ	BTFSS 3, 2	—
0110 0100 0011 (643)	不为 0 则跳	SKPNZ	BTFSC 3, 2	—
0010 001f ffff (22f)	测试寄存器	TSTF f	MOVF f, 1	Z
0010 000f ffff (20f)	搬移寄存器到 W	MOVFW f	MOVF f, 0	Z
0010 011f ffff (26f)	寄存器取补码	NEGF f, d	COMF f, 1	Z
0010 10df ffff (28f)			INCF f, d	
0110 0000 0011 (603)	加进位到寄存器	ADDCF f, d	BTFSC 3, 0	Z
0010 10df ffff (28f)			INCF f, d	
0110 0000 0011 (603)	寄存器减进位	SUBCF f, d	BTFSC 3, 0	Z
0000 11df ffff (0cf)			DECF f, d	
0110 0010 0011 (623)	加辅助进位到寄存器	ADDDCF f, d	BTFSC 3, 1	Z
0010 10df ffff (28f)			INCF f, d	
0110 0010 0011 (623)	从寄存器减辅助进位	SUBDCF f, d	BTFSC 3, 1	Z
0000 11df ffff (0cf)			DECF f, d	
101k kkkk kkkk (akk)	分支	B k	GOTO k	—
0110 0000 0011 (603)	依进位分支	BC k	BTFSC 3, 0	—
101k kkkk kkkk (akk)			GOTO k	
0111 0000 0011 (703)	不进位分支	BNC k	BTFSS 3, 0	—
101k kkkk kkkk (akk)			GOTO k	
0111 0000 0011 (703)	辅助进位为 1 分支	BDC k	BTFSC 3, 1	—
101k kkkk kkkk (akk)			GOTO k	
0110 0100 0011 (643)	辅助进位为 0 分支	BNDC k	BTFSS 3, 1	—
101k kkkk kkkk (akk)			GOTO k	
0111 0100 0011 (743)	0 分支	BZ k	BTFSC 3, 2	—
101k kkkk kkkk (akk)			GOTO k	
0111 0100 0011 (743)	不为 0 分支	BNZ k	BTFSS 3, 2	—
101k kkkk kkkk (akk)			GOTO k	

表 7.2 特殊指令助记符表

在后面的例子里，你将看到程序中使用了很多的指令助记符。指令助记符容易记忆。使用它程序可读性也较好。但这取决于每个人的习惯，你可以只使用一部分你认为好记的助记符，甚至只用基本的指令符而不用助记符来编写程序。

第八章 PIC16CXX 程序设计基础

上面我们已详细介绍了 PIC16CXX 的每条指令。现在我们来总结一下它们的几个特点：

- 1、各寄存器的每一个位都可单独地被置位、清零或测试，无须通过间接比较，可节省执行时间和程序地址空间。
- 2、特殊功能寄存器的使用方法和通用寄存器的方法完全一样，即和通用寄存器一样看待。这样使程序执行和地址空间都简化很多。
- 3、对于跨页面的 CALL 和 GOTO 操作，要事先设置页面地址位 PCLATH<3>，对于 CALL 来说，子程序返回后还要将 PCLATH<3>恢复到本页面地址。

§ 8.1 程序的基本格式

先介绍二条伪指令：

1、EQU — 标号赋值伪指令

ORG — 地址定义伪指令

PIC16CXX 一旦 RESET 后指令计数器 PC 被置为 0，所以 PIC16CXX 所有型号芯片的复位地址为 0H。

一般说来，PIC 的源程序并没有要求统一的格式，大家可以根据自己的风格来编写。但这里我们推荐一种清晰明了的格式供参考。

```
TITLE          This      is.....      ;程序标题
;-----
;名称定义和变量定义
;-----
F0             EQU       0
RTCC           EQU       1
PC             EQU       2
STATUS        EQU       3
FSR            EQU       4
RA             EQU       5
RB             EQU       6
COUNTER        EQU      18H           ;寄存器变量及常数定义
A              EQU       46
B              EQU       42
;-----
INCLUDE      "16CXX.EQU"           ;引入定义文件
;-----
              ORG        0
              GOTO       MAIN
              ORG        4
              GOTO       INT_BODY   ;0004H 为中断程序入口
;-----
              ORG        5           ;从 0005H 开始放程序
              ...
              END                ;程序结束符
```

另一些指令书写注意事项请参阅宏汇编 MPASM 章节。

16CXX.EQU 是一个定义 PIC 各种寄存器变量的文件，用户可以在其中增加定义或删除定义。

```
;*****PIC16CXX 特殊功能寄存器定义*****
```

;特殊功能寄存器

INDF	EQU	0	
F0	EQU	0	;间址寄存器
TMR0	EQU	1	;TIMER0 实时时钟
OPTION	EQU	81H	
OPTION_R	EQU	81H	;TMR0 预分频寄存器
PCL	EQU	2	;程序计数器低 8 位
STATUS	EQU	3	;状态寄存器
FSR	EQU	4	;间址选择寄存器
PORT_A	EQU	5	;—
PORT_B	EQU	6	; 可编程
PORT_C	EQU	7	; 双向 I/O 口
PORT_D	EQU	8	;
PORT_E	EQU	9	;—
TRISA	EQU	85H	;—
TRISB	EQU	86H	; I/O 口方向
TRISC	EQU	87H	; 控制寄存器
TRISD	EQU	88H	;
TRISE	EQU	89H	;—
PCLATH	EQU	0AH	;程序计数器高 5 位
INTCON	EQU	0BH	;中断控制寄存器
PIR1	EQU	0CH	;外设中断标志寄存器 1
PIE1	EQU	8CH	;外设中断使能寄存器 1
PIR2	EQU	0DH	;外设中断标志寄存器 2
PIE2	EQU	8DH	;外设中断使能寄存器 2
TMR1L	EQU	0EH	;TIMER1 低位计数器
TMR1H	EQU	0FH	;TIMER1 高位计数器
PCON	EQU	8EH	;上电寄存器
T1CON	EQU	10H	;TIMER1 控制寄存器
TMR2	EQU	11H	;TIMER2
T2CON	EQU	12H	;TIMER2 控制寄存器
PR2	EQU	92H	;TIMER2 周期寄存器
SSPBUF	EQU	13H	;SSP 缓冲器
SSPCON	EQU	14H	;SSP 控制寄存器
SSPADDD	EQU	93H	;SSP 地址寄存器
SSPSTAT	EQU	94H	;SSP 标志寄存器
CCPR1L	EQU	15H	;CCP1 低位寄存器
CCPR1H	EQU	16H	;CCP1 高位寄存器
CCP1CON	EQU	17H	;CCP1 控制寄存器
RCSTA	EQU	18H	;SCI 接收标志控制寄存器
TXSTA	EQU	98H	;SCI 发送标志控制寄存器
TXREG	EQU	19H	;SCI 发送寄存器
SPBRG	EQU	99H	;波特率寄存器
RCREG	EQU	1AH	;SCI 接收寄存器
CCPR2L	EQU	1BH	;CCP2 低位寄存器
CCPR2H	EQU	1CH	;CCP2 高位寄存器
CCP2CON	EQU	1DH	;CCP2 控制寄存器
ADRES	EQU	1EH	;地址寄存器
ADCON0	EQU	1FH	;A/D 控制寄存器 0
ADCON1	EQU	9FH	;A/D 控制寄存器 1

;STATUS 位定义

CARRY	EQU	0	
C	EQU	0	;进/借位
DCARRY	EQU	1	
DC	EQU	1	;半进/借位
Z_BIT	EQU	2	
Z	EQU	2	;零标志位
P_DOWN	EQU	3	;低功耗位
PD	EQU	3	
T_OUT	EQU	4	;超时位
TO	EQU	4	
RP0	EQU	5	;┐
RP1	EQU	6	; 直接寻址时寄存器体选择
RP2	EQU	7	;┘
;FSR 位定义			
PS0	EQU	5	;┐ 间接寻址时
PS1	EQU	6	;┘ 寄存器体选择
;OPTION 位定义			
TS0	EQU	0	;┐
TS1	EQU	1	; 预分频比设置
TS2	EQU	2	;┘
PSA	EQU	3	; 对 TMR0 或 WDT 分频选择
T0SE	EQU	4	; T0CKI 上升/下降沿选择
T0CS	EQU	5	; 内部/外部时钟选择
INTEDG	EQU	6	; INT 上升/下降沿中断选择
RBPUP	EQU	7	; RB 口弱上拉设置
;T1CON 位定义			
TMR1ON	EQU	0	;开/关定时器 1
TMR1CS	EQU	1	;时钟选择
T1SYNC	EQU	2	;同步/异步时钟选择
T1OSCEN	EQU	3	;开/关 T1 外部振荡
T1CKPS0	EQU	4	;┐ 时钟分频
T1CKPS1	EQU	5	;┘ 设置
;T2CON 位定义			
T2CKPS0	EQU	0	;┐ 输入时钟
T2CKPS1	EQU	1	;┘ 分频比设置
TMR2ON	EQU	2	;开/关定时器 2
TOUTPS0	EQU	3	;┐
TOUTPS1	EQU	4	; 输出比例
TOUTPS2	EQU	5	; 设置
1TOUTPS3	EQU	6	;┘
;INTCON 位定义			
RBIF	EQU	0	;RB 口中断标志
INTF	EQU	1	;INT 中断标志
RTIF	EQU	2	;RTCC 溢出中断标志
RBIE	EQU	3	;RB 口中断允许
INTE	EQU	4	;INT 中断允许
RTIE	EQU	5	;RTCC 中断允许
ADIE	EQU	6	;A/D 中断允许 (只针对 16C71)
PEIE	EQU	6	;外设中断允许(除 16C71 以外)

GIE	EQU	7	;全体中断允许
-----	-----	---	---------

;ADCON0 位定义

ADON	EQU	0	;选通 A/D
ADIF	EQU	1	;A/D 中断标志
GODONE	EQU	2	;启动 A/D
CHS0	EQU	3	;¬
CHS1	EQU	4	; A/D 通道选择
CHS2	EQU	5	;¬
ADCS0	EQU	6	;¬ A/D 频率
ADCS1	EQU	7	;¬ 选择

;ADCON1 位定义

PCFG0	EQU	0	;¬
PCFG1	EQU	1	; A/D 口设置
PCFG2	EQU	2	;¬

;PIR1 位定义

TMR1IF	EQU	0	;TMR1 中断标志
TMR2IF	EQU	1	;TMR2 中断标志
CCP1IF	EQU	2	;CCP1 中断标志
SSPIF	EQU	3	;SSP 中断标志
TXIF	EQU	4	;SCI 发送中断标志
RCIF	EQU	5	;SCI 接收中断标志
ADIF	EQU	6	;A/D 中断标志
PSPIF	EQU	7	;并行口中断标志

;PIR2 位定义

CCP2IF	EQU	0	;CCP2 中断标志
--------	-----	---	------------

;SSPCON 位定义

SSPM0	EQU	0	;¬
SSPM1	EQU	1	; SPI 或 I2C 模式及
SSPM2	EQU	2	; 分频比设置
SSPM3	EQU	3	;¬
CKP	EQU	4	;时钟上升/下降沿选择
SSPEN	EQU	5	;串行口使能
SSPOV	EQU	6	;接收到数据标志位
WCOL	EQU	7	;准备好发送数据标志位

;CCP1CON 位定义

CCP1M0	EQU	0	;¬
CCP1M1	EQU	1	; CCP1 模式选择及
CCP1M2	EQU	2	; 分频比设置
CCP1M3	EQU	3	;¬
CCP1Y	EQU	4	;¬ PWM 模式中 10 位脉宽
CCP1X	EQU	5	;¬ 系数的低 2 位

;CCP2CON 位定义

CCP2M0	EQU	0	;¬
--------	-----	---	----

CCP2M1	EQU	1	; CCP2 模式选择
CCP2M2	EQU	2	; 及分频比设置
CCP2M3	EQU	3	;┐
CCP2Y	EQU	4	;┐ PWM 模式中 10 位
CCP2X	EQU	5	;┐ 脉宽系数的低 2 位
;TRISE 位定义			
PSPMODE	EQU	4	;并行口模式选择
IBOV	EQU	5	;读、写并行口冲突标志
OBF	EQU	6	;准备好发送并行口数据
IBF	EQU	7	;接收到并行口数据
;RCSTA 位定义			
RCD8	EQU	0	;接收到数据的第 9bit
OERR	EQU	1	;溢出错误标志
FERR	EQU	2	;传输间断错误标志
CREN	EQU	4	;连续接收位
SREN	EQU	5	;单个接收位
RC8/9	EQU	6	;8bit/9bit 选择
SPEN	EQU	7	;串行口使能
;PIE1 位定义			
TMR1IE	EQU	0	;T1 中断允许
TMR2IE	EQU	1	;T2 中断允许
CCP1IE	EQU	2	;CCP1 中断允许
SSPIE	EQU	3	;SSP 中断允许
TXIE	EQU	4	;T2 中断允许
RCIE	EQU	5	;RC 中断允许
ADIE	EQU	6	;A/D 中断允许 (16C72/73/74)
PSPIE	EQU	7	;PSP 中断允许
;PIE2 位定义			
CCP2IE	EQU	0	;CCP2 中断允许
;上电寄存器 PCON			
POR	EQU	1	;上电复位标志
;SSPSTAT 位定义			
BF	EQU	0	;接收完标志
UA	EQU	1	;要求更新地址标志
R/W	EQU	2	;读写控制
S	EQU	3	;I ² C 模式起始位
P	EQU	4	;I ² C 模式停止位
D/A	EQU	5	;数据/地址标志
;TXSTA 位定义			
TXD8	EQU	0	;发送数据的第 9bit
TRMT	EQU	1	;传输寄存器空标志
BRGH	EQU	2	;速率选择
SYNC	EQU	4	;同步/异步选择
TXEN	EQU	5	;发送选通
TX8/9	EQU	6	;8bit/9bit 格式选择

另外像 PIC16C8X、PIC16C62X 等一些功能寄存器及其位的定义用户也可以自己加进这个文件中。

§ 8.2 程序设计基础

一、设置 I/O 口的输入/输出方向

PIC16CXX 的 I/O 口一般为双向可编程, 即每一根 I/O 端线都可分别单独地由程序设置为输入或输出。这个过程由写 I/O 控制寄存器 TRISf 来实现, 写入值为“1”, 则为输入; 写入值为“0”, 则为输出。

MOVLW	0FH	;0000 1111(0FH)
BSF	STATUS, RP0	;W 中的 0FH 写入 B 口控制器，
MOVWF	TRISB	;B 口高 4 位为输出，低 4 位为输入。
BCF	STATUS, RP0	
MOVLW	0C0H	;11 000000 (C0H)
MOVWF	6	

二、检查寄存器是否为零

如果要判断一个寄存器内容是否为零，很简单：

	MOVF	10, 1	;F10→F10, 结果影响状态位 Z
Z=1	┌ SKPZ		;F10 为零则跳 (Z=1 否)
(F10=0)	GOTO		;F10 不为零
	...		
	└→...		
	...		
NZ	MOVLW	55H	
	...		
	...		

三、比较二个寄存器的大小

要比较二个寄存器的大小，可以将它们做减法运算，然后根据状态位 C 来判断。注意，相减的结果放入 W，则不会影响二寄存器原有的值。

例如 F8 和 F9 二个寄存器要比较大小:

MOVF	8, 0	;F8→W
SUBWF	9, 0	;F9-F8→W
SKPNZ		;判断 Z=1 否 (即 F9=F8 否)
GOTO	F8=F9	;F9=F8
SKPNC		;C=0 则跳
GOTO	F9>F8	;C=1, 相减 1 结果为正, F9>F8
GOTO	F8>F9	;C=0, 相减结果为负, F8>F9
...		
...		

四、循环 n 次的程序

如果要使某段程序循环执行 n 次，可以用一个寄存器作计数器。下例以 F10 做计数器，使程序循环 8 次。

```

COUNT    EQU                ;定义 F10 名称为 COUNT（计数器）
...
...

    MOVLW    8                ;循环次数→COUNT
    MOVWF    COUNT
LOOP:  CLRW                    ;循环体
    ...
    ...
    DECFSZ   COUNT, 1         ;COUNT 减 1，结果为零则跳
    GOTO     LOOP            ;结果不为零，继续循环
    ...                      ;结果为零，跳出循环
    ...

```

五、“IF.....THEN.....” 格式的程序

下面以 “IF X=Y THEN GOTO NEXT” 格式为例。

```

X    EQU    ××
Y    EQU    ××                ;X, Y 值由用户定义（变量）
...
...
    MOVLW    X
    MOVWF    10                ;X→F10
    MOVLW    Y
    MOVWF    W                 ;Y→W
    SUBWF    10, 0             ;X-Y→W
    SKPNZ    0                 ;X=Y 否
X≠Y | GOTO    NEXT            ;X=Y, 跳到 NEXT 去执行。
    | ...
    | ...

```

六、“FOR.....NEXT” 格式的程序

“FOR.....NEXT” 程序使循环在某个范围内进行。下例是 “FOR X=0 TO 5” 格式的程序。F10 放 X 的初值，F11 放 X 的终值。

```

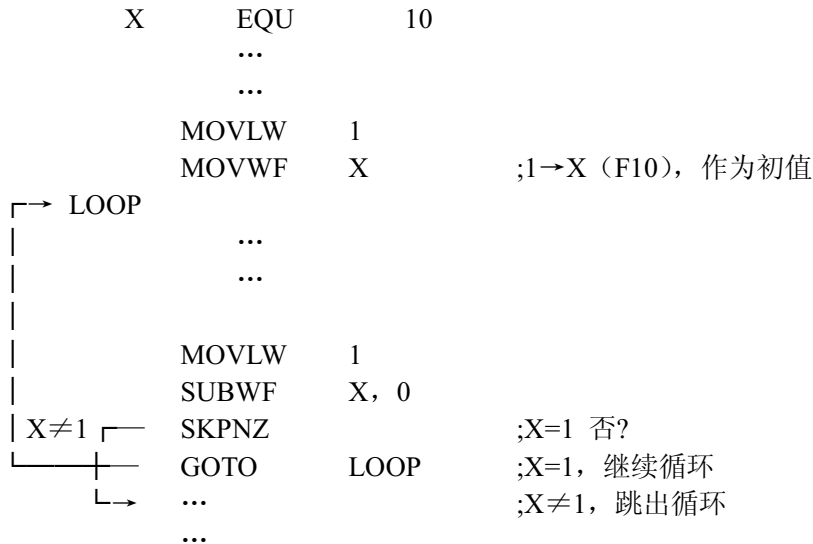
START    EQU    10
END       EQU    11
...
...
    MOVLW    0
    MOVWF    START        ;0→START (F10)
    MOVLW    5
    MOVWF    END'         ;5→END' (F11)
LOOP:    ...                ;循环体
    ...
    INCF     START, 1      ;START 值加 1
    MOVF     START, 0
    SUBWF    END, 0        ;START=END'? (X=5 否)

```



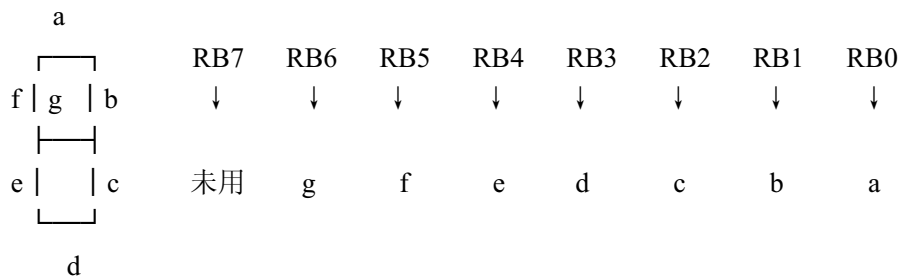
七、“DO WHILE.....END” 格式的程序

“DO WHILE.....END” 程序是在符合条件下执行循环。下例是“DO WHILE X=1” 格式的程序。F10 放 X 的值。



八、查表程序

查表是程序中经常用到的一种操作。下例是将十进制 0~9 转换成 7 段 LED 数字显示值。若以 B 口的 RB0~RB6 来驱动 LED 的 a~g 线段，则有如下关系：



设 LED 为共阳，则 0~9 数字对应的线段值如下表：

PIC 的查表程序可以利用子程序带值返回的特点来实现。具体是在主程序中先取表数据地址放入 W，接着调用子程序，子程序的第一条指令将 W 置入 PC，则程序跳到数据地址的地方，再由“RETLW”指令将数据放入 W 返回到主程序。

十进数	线段值	十进数	线段值
0	C0H	5	92H
1	C9H	6	82H
2	A4H	7	F8H
3	B0H	8	80H
4	99H	9	90H

表 8.1 0~9 LED 线段值

下面程序以 F10 放表头地址。

```

        MOVLW    TABLE      ;表头地址→F10
        MOVWF    10
        ...
        ...
        MOVLW    1            ;1→W，准备取“1”的线段值
        ADDWF    10, 1        ;F10+W=“1”的数据地址
        CALL     CONVERT
        MOVWF    6            ;线段值置到 B 口，点亮 LED。
        ...
        ...
CONVERT MOVWF    2            ;W→PC
TABLE   RETLW    C0H          ;“0”线段值
        RETLW    F9H          ;“1”线段值
        ...
        ...
        RETLW    90H          ;“9”线段值

```

九、“READ.....DATA，RESTORE”格式程序

“READ.....DATA”程序是每次读取数据表的一个数据，然后将数据指针加 1，准备下一次取下一个数据。下例程序中以 F10 被数据表起始地址，F11 做数据指针。

```

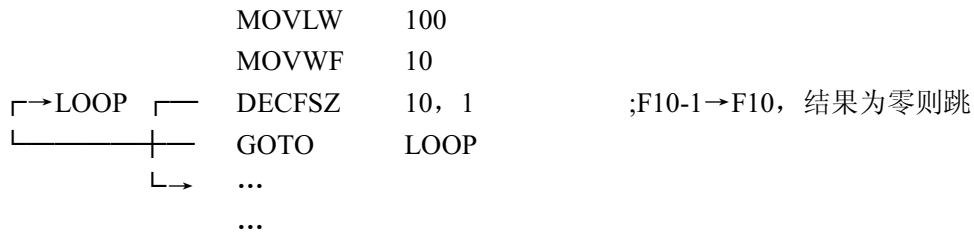
POINTER EQU    11            ;定义 F11 名称为 POINTER
        ...
        ...
        MOVLW    DATA
        MOVWF    10          ;数据表头地址→F10
        CLRF     POINTER     ;数据指针清零
        ...
        ...
        MOVF     POINTER, 0
        ADDWF    10, 0        ;W=F10+POINTER
        ...
        ...
        INCF     POINTER, 1   ;指针加 1
        CALL     CONVERT     ;调子程序，取表格数据
        ...
        ...
CONVERT MOVWF    2            ;数据地址→PC
DATA    RETLW    20H          ;数据
        ...
        RETLW    15H          ;数据

```

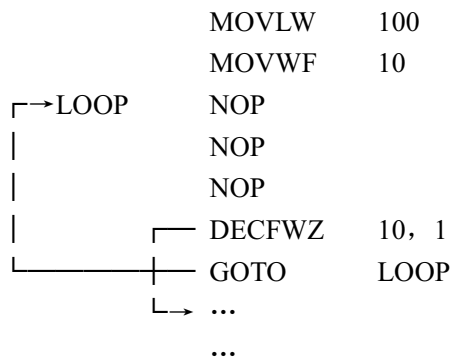
如果要执行“RESTORE”，只要执行一条“CLRF POINTER”即可。

十、延时程序

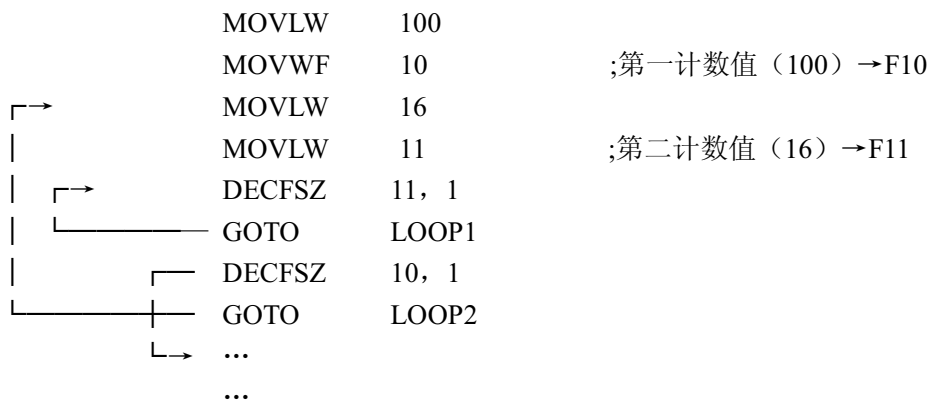
如果延时时间较短，可以让程序简单地连续执行几条空操作指令“NOP”。如果延时时间长，可以用循环来实现。下例以 F10 计算，使循环重复执行 100 次。



延时程序中计算指令执行的时间和即为延时时间。如果使用 4MHz 振荡, 则每个指令周期为 $1\mu\text{S}$ 。所以单周期指令执行时间为 $1\mu\text{S}$, 双周期指令为 $2\mu\text{S}$ 。在上例的 LOOP 循环延时时间即为: $(1+2) * 100 + 2 = 302 (\mu\text{S})$ 。在循环中插入空操作指令即可延长延时时间:


$$\text{延时时间} = (1+1+1+1+2) * 100 + 2 = 602 \text{ (}\mu\text{S)}。$$

用几个循环嵌套的方式可以大大延长延时时间。如下例用 2 个循环来做延时。


$$\text{延时时间} = [(1+2) \times 6 + 2] \times 1 + 2 + 1 + 1 \times 100 + 2 = 5502 (\mu\text{S})$$

十一、寄存器体 (Bank) 的寻址

在 PIC16CXX 中，寄存器有 2 个体：

Bank0: 00H~7FH

Bank1: 80H~FFH

而在 PIC16CXX 的指令代码中，只有 7 位是寄存器的地址位，所以指令中只能直接寻址一个体，因此在状态寄存器 STATUS 中增加 RP0 位 (STATUS<5>) 来选体。

以 PIC16C64 为例，假设用户要操作 Bank1 中的 A0 寄存器，则应：

BSF	STATUS, RP0	;选 Bank1
MOVLW	55H	
MOVWF	0XA0	;55H→A0H 寄存器
BCF	STATUS, RP0	;恢复到 Bank0

如果不作选体（假设目前是在 Bank0）:

MOVLW	55H
MOVWF	0XA0

那么 55H 实际上不是置入 Bank1 中的 A0H 寄存器，而是 Bank0 中的 20H 寄存器！

第九章 PIC16CXX 设计范例

前面几章已对 PIC16CXX 的硬件构成、指令系统、程序设计基础及系统扩充做了详细的叙述，相信你已经对 PIC16CXX 芯片有了相当的认识，下面几章的主要目的是帮助大家实际应用 PIC。

§ 9.1 开发步骤流程

当决定选择 PIC 来实现设计后，一般可以采取下面的开发步骤来完成：

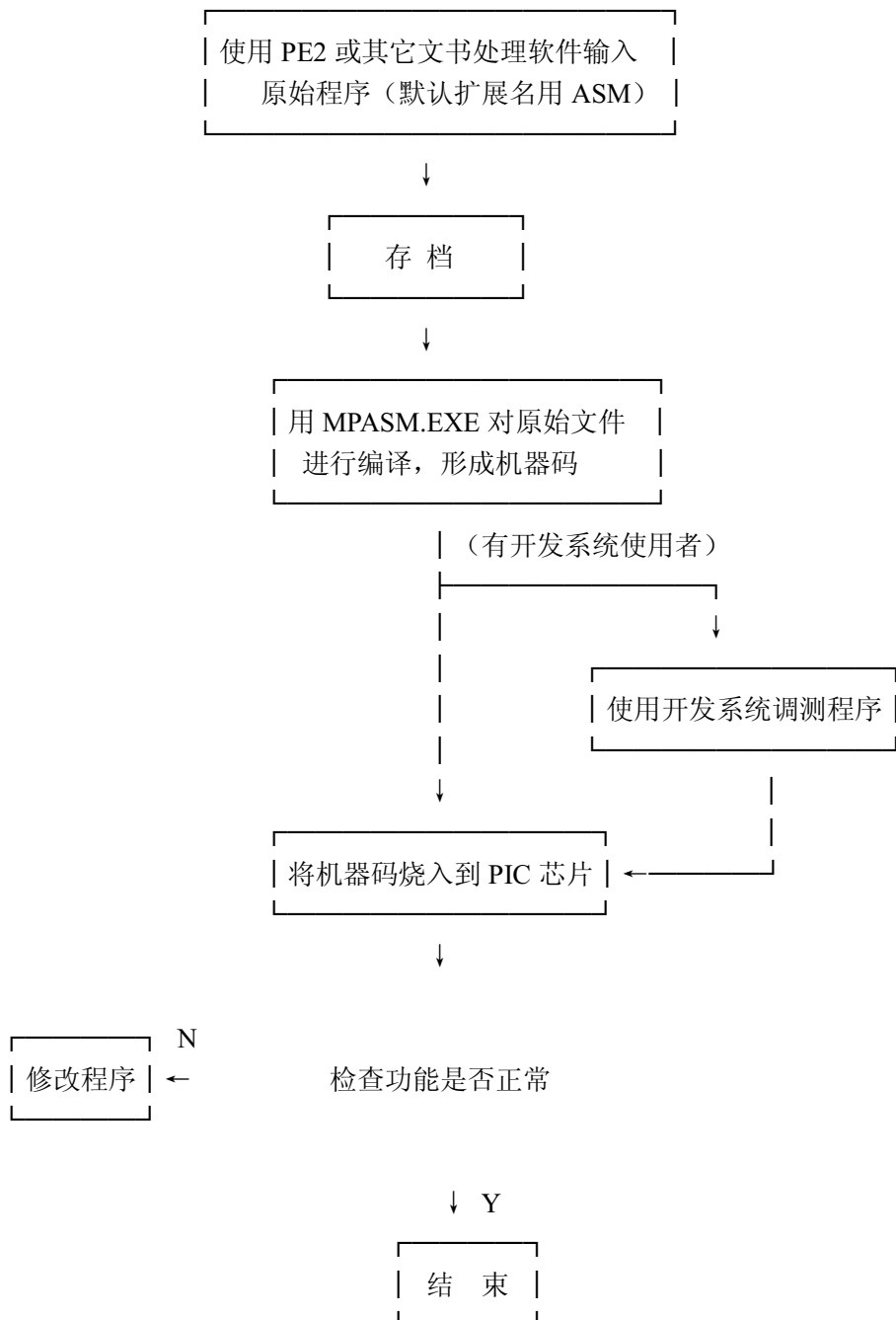


图 9.1 PIC 开发步骤

§ 9.2 设计实例

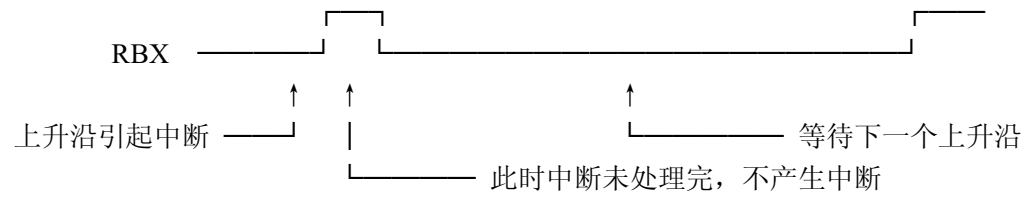
下面的这些实例仅向读者展示 PIC16CXX 的一些基本应用，用户可以做参考。

一、RB 口电平变化中断的几种情况

由于 RB 口中断是电平变化中断，无论有上升、下降沿变化都会产生中断。因此，若只有上升沿（或下降沿）中断是有效的，就必须设法剔除下降沿（或上升沿）的无效中断。

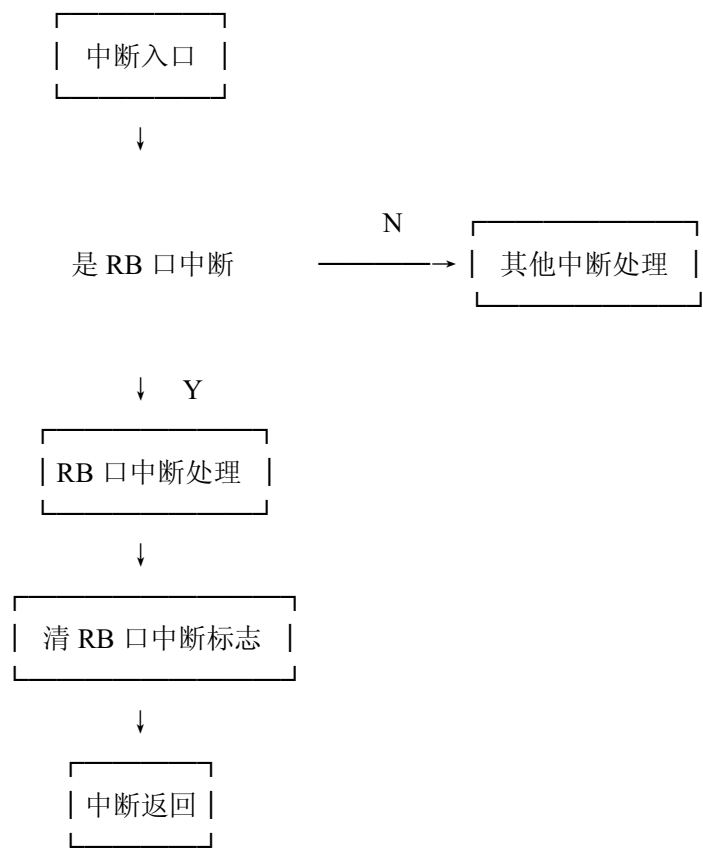
（一）窄脉冲的上升沿中断：

1、中断特点



窄脉冲时，上升沿产生中断响应后，关全体中断允许，因此下降沿时虽然也会把 RBIF 置 1，但此时不产生中断。在中断返回前把 RBIF 清零，就剔除了下降沿的无效中断：

2、流程图



3、中断程序

```
RER_INT      BTFSS      INTCON, RBIF      ;RB 口中断?
              GOTO       OTHER_INT        ;转到其他中断
              ...          ;RB 口中断处理

CLR_RBIMTF    MOVF       RB, 1             ;读 RB 口
              BCF         INTCON, RBIF     ;清 RB 口中断标志
              RETFIE

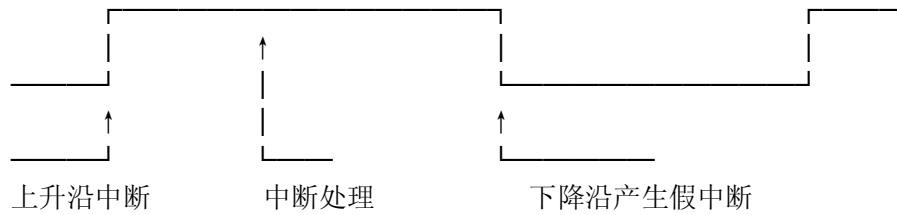
OTHER_INT
```

...
RETFIE

;其他中断处理

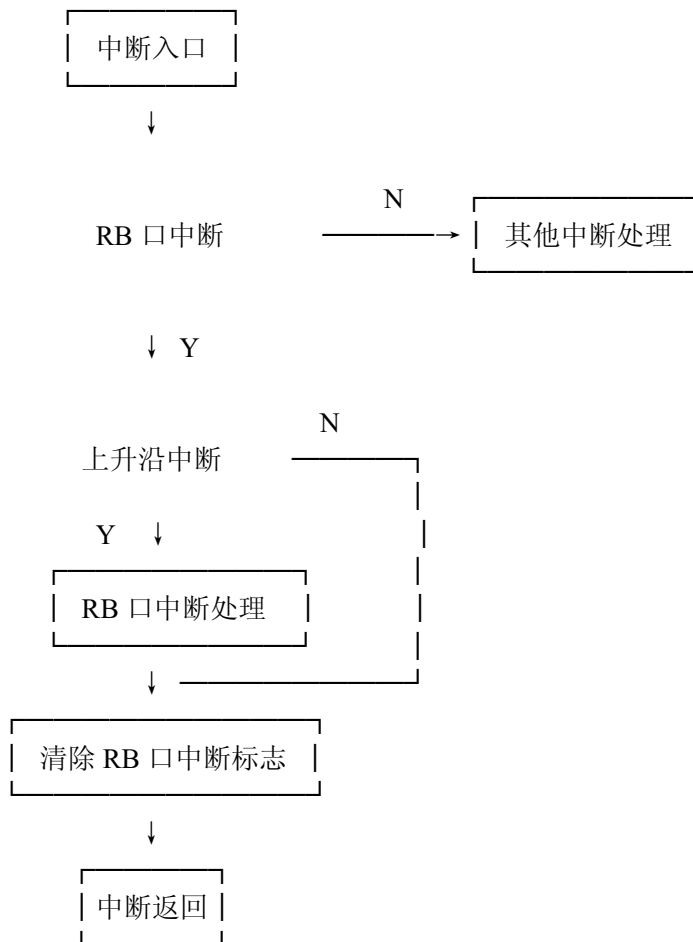
(二) 宽脉冲上升沿中断:

1、中断特性



当中断源脉冲很宽时，上升沿产生中断响应，下降沿将产生一个假中断，程序就该剔除这个假中断。

2、流程图



3、程序清单

PER_INT	BTFSS	INTCON, RBIF	;RB 口中断?
	GOTO	OTHER_INT	;其他中断
	BTFSS	RB, RBx	;上升沿中断?
	GOTO	CLR_RBINTF	;下降沿, 剔除
	...		;中断处理
CIR_RBINTF	MOVF	RB, 1	;读 RB 口
	BCF	INTCON, RBIF	;清 RB 口中断标志
	RETFIE		;中断返回

OTHER_INT ... ;其他中断处理

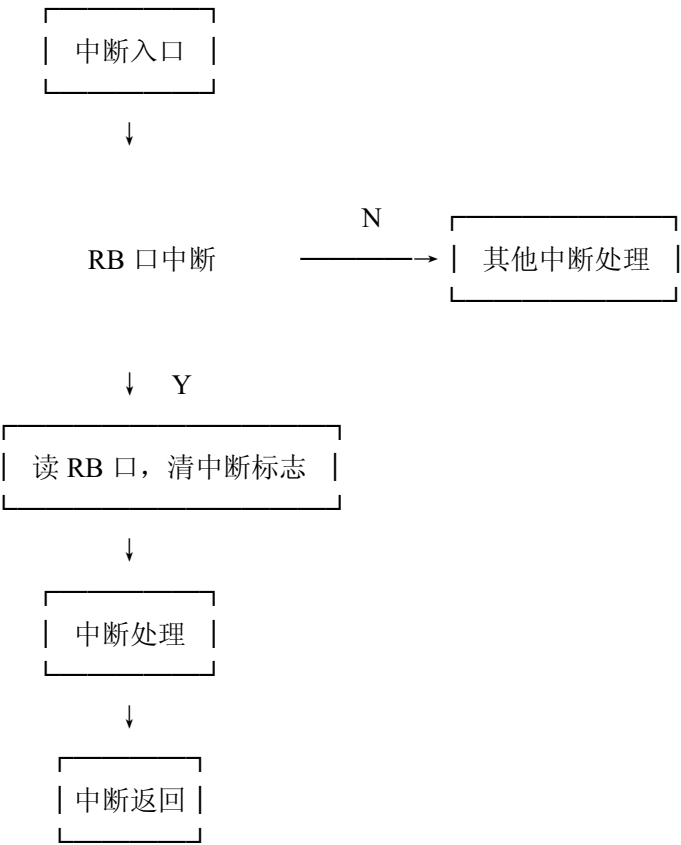
RETfie

(三) 电平跳变 RB 口中断

1、中断特点

若 RB 口的上跳和下跳都是产生有效中断，那么中断源脉宽必须足够宽以使能不漏掉中断，最小脉冲宽为中断沿开始到中断程序中读 RB 口和清 RBIF 之间的最大时间。

2、流程图



3、中断程序清单

PER_INT	BTFSS	INTCON, RBIF	;RB 口中断?
	GOTO	OTHER_INT	;其他中断
CLR_RBINTF	MOVF	RB, 1	;读 RB 口
	BCF	INTCON, RBIF	;清 RB 口中断标志
	...		;中断处理
	RETfie		;中断返回
OTHER_INT	...		;其他中断处理
	RETfie		

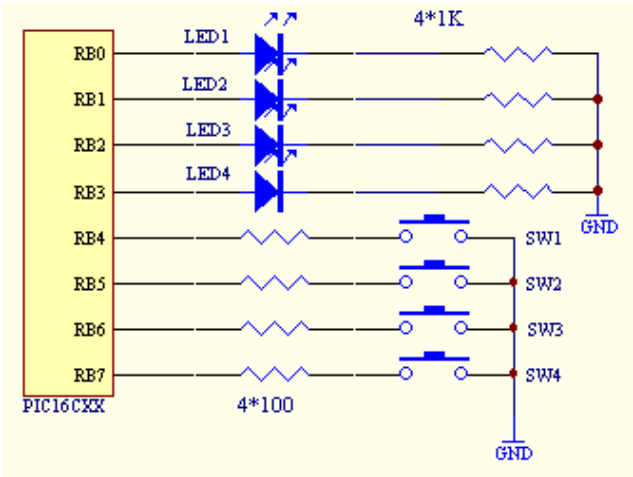
二、利用按键来唤醒 CPU

利用 PIC16CXX 系列 RB 口中断特性，可方便地由按键来唤醒 CPU，而大部分时间 CPU 处于省电状态。

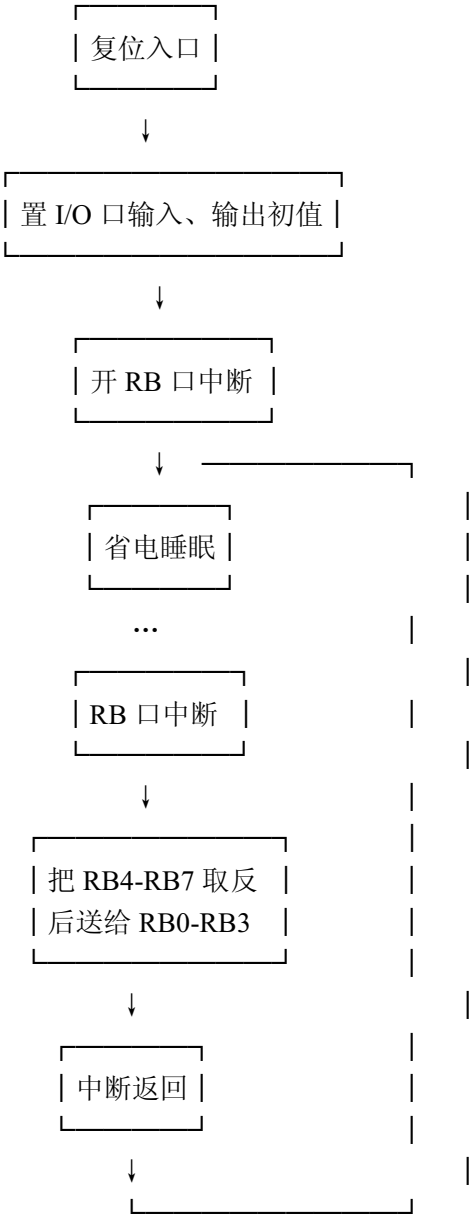
1、电路设计

选用 PIC16C61，此电路实现以下功能:按 SW1 键亮 LED1 灯，松开后灯灭，与此类推，SW4 对应 LED4。CPU 平时处于省电睡眠状态，在按键按下或松开时 RB 口电平变化引起中断，点亮或熄灭相应的 LED，然后又回到省电

睡眠。



2、程序流程图



3、程序清单


```

;
include    "PICREG.EQU"
TEMP      EQU      10H
OPTIONREG EQU      1H
RBPU      EQU      7H

                ORG      0                ;复位地址
                GOTO     START            ;
                ORG      4                ;中断向量
SERVICEINT    ;中断服务程序
                BTFSC    INTCON, RBIF     ;是否 RB 口中断?
                GOTO     SERVICEWAKUP     ;是, RB 口中断
                CLRF     INTCON
                MOVF     RB, W
                BSF      INTCON, RBIE
                RETFIE

SERVICEWAKUP
                BCF      INTCON, RBIF     ;清 RB 口中断标志
                COMF     RB, W            ;读 RB 口并取反
                MOVWF    TEMP            ;暂存在 TEMP
                SWAPF    TEMP, W         ;半字节交换后
                MOVWF    RB              ;送给 RB 口, 点亮截熄灭相应 LED
                MOVFW    RB              读 RB 口
                RETFIE                    ;中断返回

START
                BSF      STATUS, RP0     ;选择寄存器体 1
                MOVLW    0
                MOVWF    TRISA           ;置 RA 口为输出口
                MOVLW    11110000B      ;置 RB0-RB3 为输出口
                MOVWF    TRISB          ;置 RB4-RB7 为输入口
                BCF      OPTIONREG, RBPU ;使 RB 口弱上拉
                BCF      STATUS, RP0     ;选择寄存器体 0
                CLRF     RB
                CLRF     RA
                CLRF     INTCON          ;清中断控制寄存器
                MOVF     RB, W
                BSF      INTCON, RBIE     ;开 RB 口中断允许
                BSF      INTCON, GIE      ;开中断允许

LOOP
                SLEEP                    ;省电睡眠, 直至按键唤醒
                NOP
                GOTO     LOOP            ;

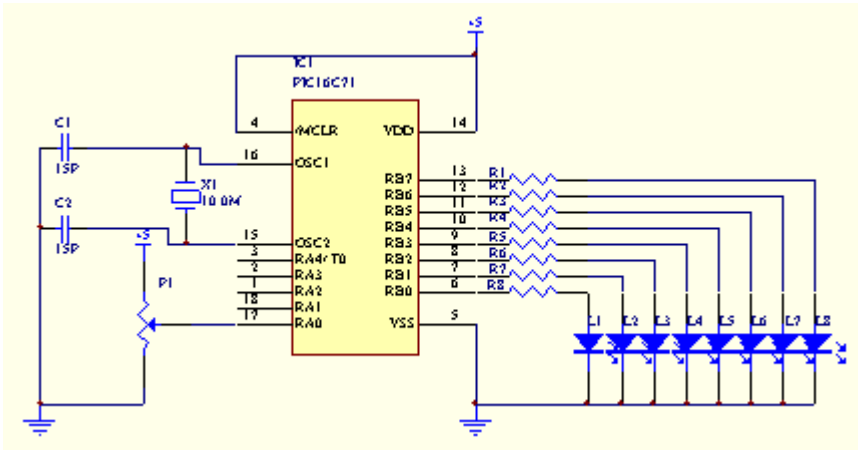
                END

```

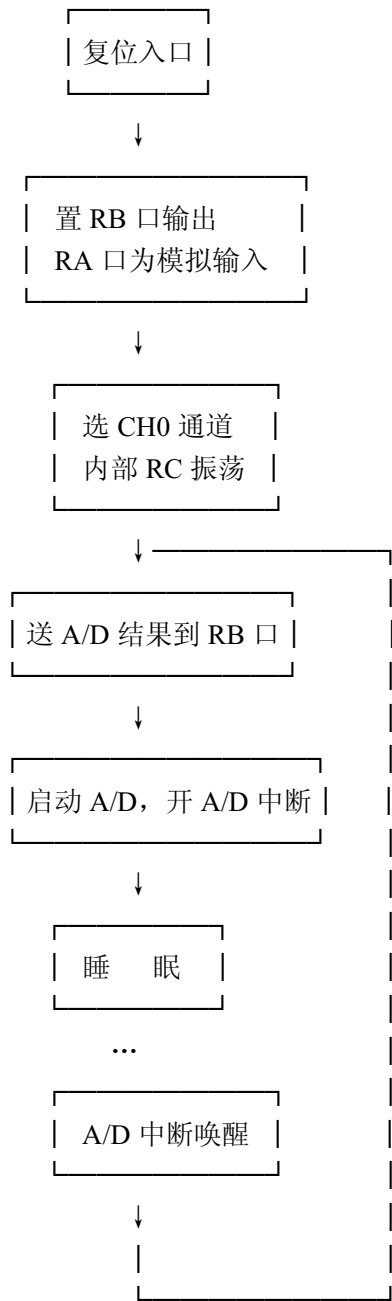
三、A/D 转换

1、电路设计

本例用 PIC16C71 实现单通道 8 位 A/D 转换，并将转换结果以二进制形式输出到 RB 口。由于 A/D 工作频率可选择为片内 RC 振荡，所以能在睡眠中实现 A/D 变化，而在 A/D 完成后唤醒 CPU。

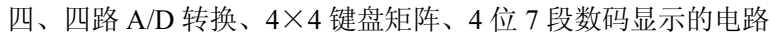


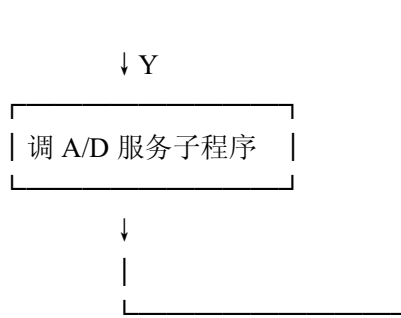
2、流程图



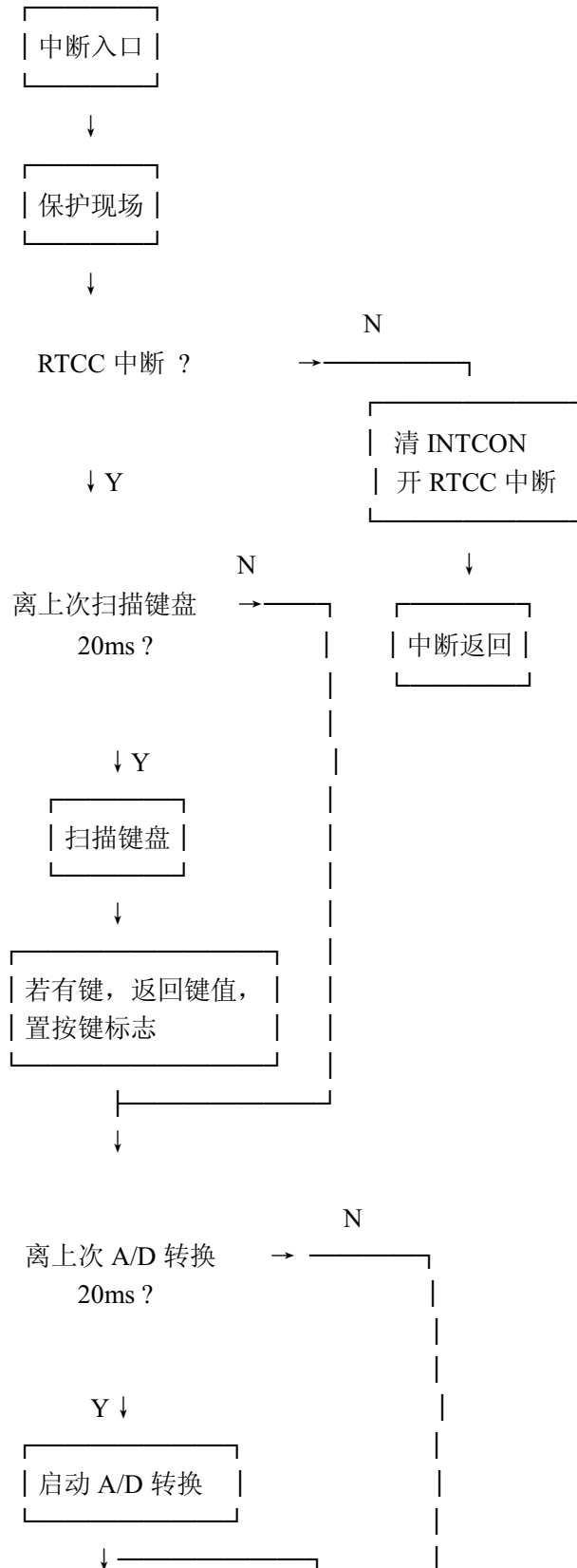
3、程序清单

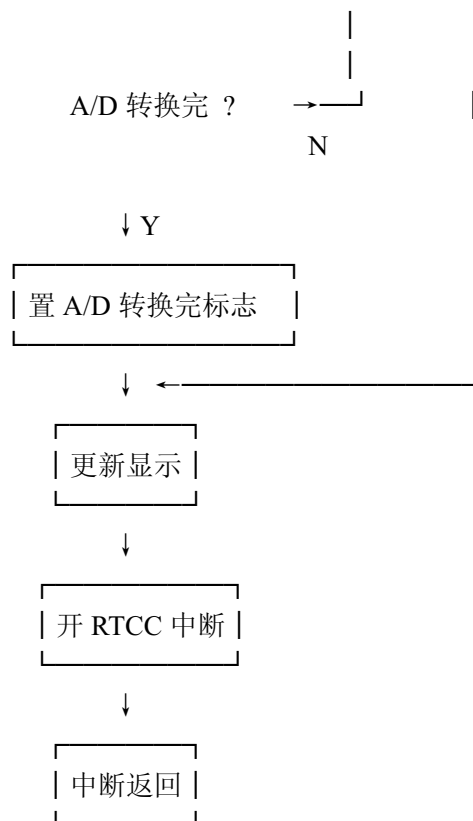
```
include "16cxx.equ"
;
TEMP EQU 10h
ADIF EQU 1
ADGO EQU 2
;
ORG 0x00 ;复位向量
GOTO START
;
ORG 0x04 ;中断向量
GOTO SERVICE_INT
;
ORG 0x10 ;主程序
START
    MOVLW B'00000000'
    MOVWF PORT_B ;清 RB 口
    BSF STATUS, 5 ;选寄存器体 1
    MOVWF TRISB ;设置 RB 口为输出
INITIALIZEAD
    MOVLW B'00000000'
    MOVWF ADCON1 ;CH0-CH3 为模拟输入通道
    BCF STATUS, 5 ;选寄存器体 0
    MOVLW B'11000001' ;选择 RC 振荡, CH0 通道
    MOVWF ADCON0 ;启动 A/D
    CLRF INTCON ;清所有中断标志
    BSF INTCON, ADIE ;允许 A/D 中断
    BSF INTCON, GIE ;开中断允许
UPDATE
    MOVF ADRES, w
    MOVWF PORT_B ;A/D 转换结果送到 RB 口
    CALL SETUPDELAY
    BSF ADCON0, ADGO ;启动下一个 A/D
;
SLEEP
GOTO UPDATE ;A/D 完成后唤醒并更新收据
;
SERVICE_INT
    BCF ADCON0, ADIF ;清 A/D 中断标志
    RETFIE
;
SETUPDELAY
    MOVLW .3
    MOVWF TEMP
SD
    DECFSZ TEMP
    GOTO SD
    RETURN
;
```



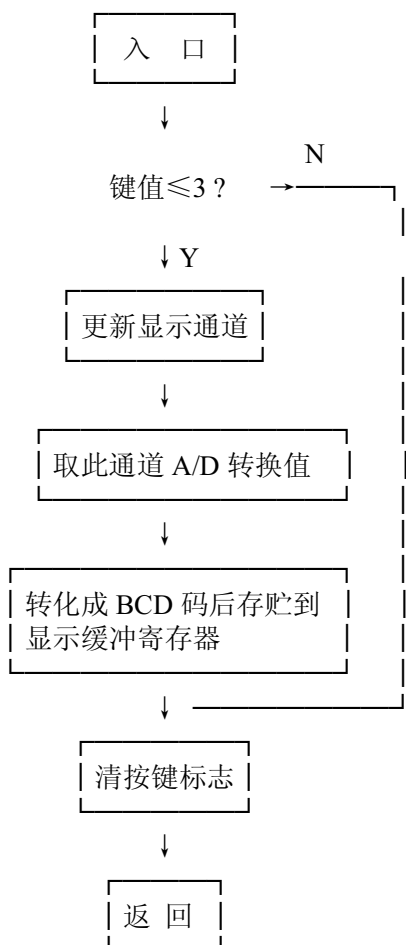


2、中断服务子程序

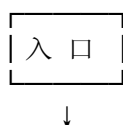


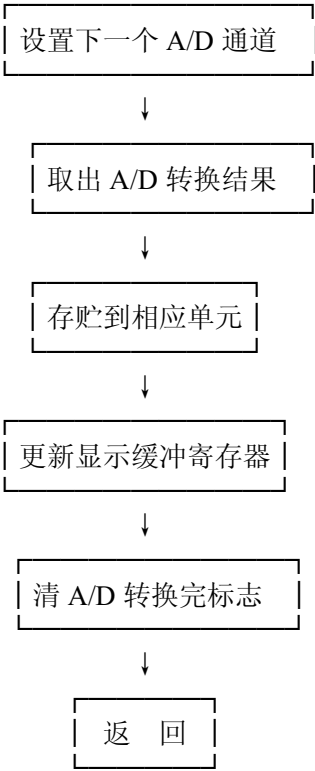


3、键盘服务子程序



4、A/D 服务子程序





(三) 程序清单

```
TempC      equ      0x0c
TempD      equ      0x0d
TempE      equ      0x0e
PABuf      equ      0x20
PBBuf      equ      0x21
Count      equ      0x0f
MsdTime    equ      0x10
LsdTime    equ      0x11
;
Flag       equ      0x12
#define keyhit      Flag, 0
#define Debnceon    Flag, 1
#define noentry     Flag, 2
#define servkey     Flag, 3
#define ADOver      Flag, 4
;
Debnce     equ      0x13
Newkey     equ      0x14
Displaych  equ      0x15
;
ADTABLE    equ      0x16
;
WBuffer    equ      0x2f
statBuffer equ      0x2e
optionReg  equ      1
PCL        equ      2
;
push      macro
```

;4 路 A/D 变换值分别存于 F16-F19

	movwf	WBuffer	;保存 W 值
	swapf	WBuffer	
	swapf	STATUS, w	
	movwf	StatBuffer	;保存状态寄存器值
	endm		
;			
pop	macro		
	swapf	StatBuffer, w	;恢复状态寄存器值
	movwf	STATUS	
	swapf	WBuffer, w	;恢复 W 值
	endm		
;			
	org	0	
	goto	start	
;			
	org	4	
	push		;保护现场
	call	serviceInterrupts	;中断服务子程序
	pop		;恢复现场
	retfie		
;			
start			
	call	Initports	;I/O 口初始化
	call	InitAd	;A/D 初始化
	call	InitTimers	;定时器初始化
loop			
	btfsc	servkey	;有按键?
	call	servicekey	;有, 按键处理
	btfsc	ADOver	;A/D 转换完?
	call	serviceAD	;是, A/D 转换结果处理
	goto	loop	
servicekey			
	bcf	servkey	;清按键标志
	movf	Newkey, w	;保存键值
	sublw	3	;键值>3?
	btfss	STATUS, C	
	return		;是, 无效按键, 忽略
	movf	Newkey, w	
	movwf	Displaych	;更新通道
LoadAD			
	movlw	ADTABLE	;A/D 存贮缓冲区首址加偏移量
	addwf	Displaych, w	
	movwf	FSR	
	movf	0, w	;取出 A/D 转换值
	movwf	L_byte	
	clrf	H_byte	
	call	B2_BCD	
	movf	R2, w	
	movwf	LsdTime	;存贮低字节
	movf	R1, w	

	movwf	MsdTime	;存储高字节
	return		
ServiceAD			
	movf	ADCON0, w	;取出 ADCON0 值
	movwf	TempC	;存在 TempC
	movlw	B'00001000'	;选择下一个通道
	addwf	ADCON0, w	
	btfsc	ADCON0, 5	;如果通道>ch3
	movlw	B'11000001'	;选择 ch0
	movwf	ADCON0	
	movlw	ADTABLE	
	movwf	FSR	;指出 A/D 存储缓冲区首地址
	rrf	TempC	
	rrf	TempC	
	rrf	TempC, w	;根据通道
	andlw	3	;得出偏移量
	addwf	FSR	;A/D 存储地址
	movf	ADRES, w	;取出 A/D 转换结果
	movwf	0	;存储到相应单元
	bcf	ADOver	;清 A/D 结束标志
	call	LoadAD	;更新显示寄存器
	return		
Initports			
	bst	STATUS, RP0	;选择寄存器体 1
	movlw	3	;RA0-3 为数字 I/O 口
	movwf	ADCON1	
	clrf	TRISA	;RA 口输出
	clrf	TRISB	;RB 口输出
	bcf	STATUS, RP0	;选择寄存器体 0
	clrf	PORT_A	
	clrf	PORT_B	
	bsf	PORT_A, 3	;点亮最高位
	return		
InitTimers			
	clrf	MsdTime	;清定时寄存器
	clrf	LsdTime	
	clrf	Displaych	;通道为 0
	clrf	Flag	;清标志
	bsf	STATUS, RP0	
	movlw	B'10000100'	
	movwf	OptionReg	;分频比为 32
	bcf	STATUS, RP0	
	movlw	B'00100000'	;开 RTCC 中断
	movwf	INTCON	
	movlw	.96	
	movwf	RTCC	;置 RTCC 初值
	retfie		
;中断服务子程序			
ServiceInterrupts			
	btfsc	INTCON, RTIF	;RTCC 中断?

	goto	ServiceRTCC	;是，转入 RTCC 中断处理
	clrf	INTCON	;否则，清除所有中断标志
	bsf	INTCON, RTIE	;允许 RTCC 中断
	return		
;RTCC 中断处理			
ServiceRTCC			
	movlw	.96	;置 RTCC 初值
	movwf	RTCC	
	bcf	INTCON, RTIF	;清除中断标志
	btfsc	PORT_A, 0	
	call	Scankeys	;用 20ms 扫描一次键盘
	btfsc	PORT_A, 3	
	call	SampleAd	;用 20ms 扫描一次 A/D
	call	UpdateDisplay	;更新显示
	return		
;扫描键盘，若有键，则返回从"0"-"F"的键值			
Scankeys			
	btfss	Debnceon	
	goto	scan1	
	decfsz	Debnce	
	return		
	bcf	Debnceon	
	return		
Scan1			
	call	Saveports	;保存 I/O 口值
	movlw	B'11101111'	
	movwf	TempD	;置 TempD 值为'11101111'
ScanNext			
	movf	PORT_B, w	
	bcf	INTCON, RBIF	;清 RB 口中断标志
	rrf	TempD	;设置扫描的键盘矩阵列
	btfss	STATUS, C	;若 C=0，则已扫描完
	goto	Nokey	
	movf	TempD, w	;无键
	movwf	PORT_B	;输出到 RB 口
	nop		
	btfss	INTCON, RBIF	;若 RB 口中断标志为 0
	goto	ScanNext	;则此列无键按入，继续下一列
	btfsc	keyhit	;有键按入，测试上次键是否松开?
	goto	SKreturn	;没有
	bsf	keyhit	;设按键标志
	swapf	PORT_B, w	
	movwf	TempE	;把 RB 口字节交换后存于 TempE
	call	GetkeyValue	;转换成键值"0"-"F"
	movwf	NewKey	;保持新键值
	bsf	Servkey	
	bsf	Debnceon	;设置标志
	movlw	4	
	movwf	Debnce	
SKreturn			

	call	RestorePorts	;恢复 I/O 口值
	return		
Nokey			
	bcf	keyhit	;清除标志
	goto	SKreturn	
GetKeyValue			
	clrf	TempC	
	btfss	TempD, 3	;第一列
	goto	RowValEnd	
	incf	TempC	
	btfss	TempD, 2	;第二列
	goto	RowValEnd	
	incf	TempC	
	btfss	TempD, 1	;第三列
	goto	RowValEnd	
	incf	TempC	;第四列
RowValEnd			
	btfss	TempE, 0	;第一行?
	goto	GetValCom	
	btfss	TempE, 1	;第二行?
	goto	Get4567	
	btfss	TempE, 2	;第三行?
	goto	Get89ab	
Getcdef			
	bsf	TempC, 2	
Get89ab			
	bsf	TempC, 3	
	goto	GetValCom	
Get4567			
	bsf	TempC, 2	
GetValCom			
	movf	TempC, w	
	addwf	PCL	
	retlw	0	
	retlw	1	
	retlw	2	
	retlw	3	
	retlw	4	
	retlw	5	
	retlw	6	
	retlw	7	
	retlw	8	
	retlw	9	
	retlw	0aH	
	retlw	0bH	
	retlw	0cH	
	retlw	0dH	
	retlw	0eH	
	retlw	0fH	

;在扫描键盘前先保存 RA 口、RB 口值

SavePorts

```
movf    PORT_A, w
movwf   PABuf           ;保存 RA 口值
clrf    PORT_A          ;关闭所有显示
movf    PORT_B, w
movwf   PBBuf           ;保存 RB 口值
movlw   0xff
movwf   PORT_B          ;RB 口输出全 1
bsf     STATUS, RP0
bcf     optionReg, 7     ;RB 口弱上拉
movlw   B'11110000'     ;RB0-RB3 为输出口
movwf   TRISB           ;RB4-RB7 为输入口
bcf     STATUS, RP0
return
```

;扫描键盘后, 恢复 RA 口、RB 口值

RestorePorts

```
movf    PBBuf, w
movwf   PORT_B          ;恢复 RB 口值
movf    PABuf, w
movwf   PORT_A          ;恢复 RA 口值
bsf     STATUS, RP0
bsf     OptionReg, 7     ;取消 RB 口弱上拉
clrf    TRISA
clrf    TRISB           ;设置 RA 口, RB 口为输出口
bcf     STATUS, RP0
return
```

UpdateDisplay

```
movf    PORT_A, w       ;保存 RA 口当前值
clrf    PORT_A          ;熄灭所有 LED 显示
andlw   0x0f
movwf   TempC           ;存贮在 TempC
bsf     TempC, 4
rrf     TempC           ;显示下一位
btfss   STATUS, CARRY   ;C=1?
bcf     TempC, 3        ;否, 清 TempC, 3
btfsc   TempC, 0        ;是最高位?
goto    UpdateMsd       ;是
btfsc   TempC, 1        ;是第 3 位?
goto    Update3rdLsd    ;是
btfsc   TempC, 2        ;是第 2 位?
goto    Update2ndLsd    ;是
```

UpdateLsd

```
;第一位
movf    LsdTime, w      ;取出第一位
andlw   0x0f
goto    Displayout
```

Update2ndLsd

```
;取出第二位
swapf   LsdTime, w
andlw   0x0f
goto    Displayout      ;显示
```

Update3rdLsd

UpdateMsd	movf	MsdTime, w	;取出第三位
	andlw	0x0f	
	goto	Displayout	;显示
Displayout	swapf	MsdTime, w	;取出最高位
	andlw	0x0f	;显示
	call	LedTable	;取出显示段码
	movwf	PORT_B	;送出 RB 口驱动
	movf	TempC, w	;取位驱动
	movwf	PORT_A	
LedTable	return		
			;显示段码表
	addwf	PCL	
	retlw	B'00111111'	; "0" 段码
	retlw	B'00000110'	; "1" 段码
	retlw	B'01011011'	; "2" 段码
	retlw	B'01001111'	; "3" 段码
	retlw	B'01100110'	; "4" 段码
	retlw	B'01101101'	; "5" 段码
	retlw	B'01111101'	; "6" 段码
	retlw	B'00000111'	; "7" 段码
	retlw	B'01111111'	; "8" 段码
	retlw	B'01100111'	; "9" 段码
	retlw	B'01110111'	; "A" 段码
	retlw	B'01111100'	; "b" 段码
	retlw	B'00111001'	; "c" 段码
	retlw	B'01011110'	; "d" 段码
	retlw	B'01111001'	; "E" 段码
	retlw	B'01110001'	; "F" 段码
InitAd			
	movlw	B'11000000'	;选择 RC 内部振荡
	movwf	ADCON0	
SampleAd	return		
	call	SavePorts	
	call	DoAd	;做 A/D 转换
AdDone			
	btfsc	ADCON0, GO	;A/D 完成?
	goto	AdDone	;否, 继续
	bsf	ADOver	;设置 A/D 完成标志
	call	RestorePorts	;恢复 I/O 口
	return		
DoAd			
	clrf	PORT_B	;关 LED 显示
	bsf	STATUS, RP0	
	movlw	0x0f	;设置 RA 为输入口
	movwf	TRISA	
	bcf	STATUS, RP0	
	bsf	ADCON0, ADON	;开 A/D

	movlw	.125	
	call	Wait	;延时
	bsf	ADCON0, GO	;启动 A/D 转换
	return		
Wait			
	movwf	TempC	;延时参数送给 TempC
Next			
	decfsz	TempC	
	goto	Next	
	return		
count	equ	26	
temp	equ	27	
H_byte	equ	20	
L_byte	equ	21	
R0	equ	22	
R1	equ	23	
R2	equ	24	
			;二进制转换成 BCD 码，二进制数放在 H_byte 和 L_byte
			;结果存贮在 R0、R1、R2 寄存器
B2_BCD	bcf	STATUS, 0	
	movlw	.16	
	movwf	count	
	clrf	R0	
	clrf	R1	
	clrf	R2	
loop16	rlf	L_byte	
	rlf	H_byte	
	rlf	R2	
	rlf	R1	
	rlf	R0	
	decfsz	count	
	goto	adjDEC	
	RETLW	0	
adjDEC	movlw	R2	
	movwf	FSR	
	call	adjBCD	
	movlw	R1	
	movwf	FSR	
	call	adjBCD	
	movlw	R0	
	movwf	FSR	
	call	adjBCD	
	goto	loop16	
adjBCD	movlw	3	
	addwf	0, w	
	movwf	temp	
	btfsc	temp, 3	

```

movwf    0
movlw    30
addwf    0, w
movwf    temp
btfsc    temp, 7
movwf    0
RETLW    0

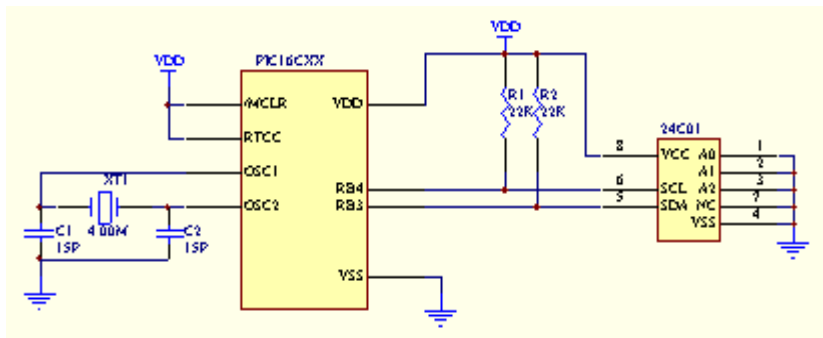
```

END

五、16CXX 和 24LC01 的联接

（一）电路设计

24LCXX 系列是以 I²C 串行总线为传输格式的 E²PROM 器件。有关 I²C 串行总线的知识请读者查阅有关资料。本例以 PIC16C54 和 24LC01 为例，给出了 PIC16CXX 和 24LCXX 之间的软件接口程序。下图为硬件连接图。



（二）程序清单

```

;
STATUS      EQU      3           ;标志寄存器
FSR         EQU      4
RA          EQU      5
RB          EQU      6
C           EQU      0
Z           EQU      2

;PORT OR RB
SDA         EQU      3           ;I2C 总线数据口
SCL         EQU      4           ;I2C 总线时钟口

SDA_IN      EQU      08H        ;置 SDA 为输入口
SDA_OUT     EQU      0          ;置 SDA 为输出口

FLAG        EQU      8H
BIT_COUNT   EQU      9H
BYTE_COUNT  EQU      0AH
CONTROL     EQU      0BH
REG3        EQU      12H
REG2        EQU      13H

```

REG1	EQU	14H	
REG0	EQU	15H	
READ_B	EQU	0	
BYTE	EQU	8	
DATA	EQU	5AH	
ADRESS	EQU	10H	
;-----			
	ORG	1FFH	
	GOTO	MAIN	
	ORG	0	
MAKE_S			
	BSF	RB, SDA	;把 SDA 强制拉高
	MOVLW	SDA_OUT	
	TRIS	RB	;置 SDA 输出
	NOP		
ACK_CHECK			;发起始位和 I ² C 器件地址
	MOVLW	SDA_IN	;置 SDA 输入
	TRIS	RB	
	NOP		
M_START			
	BSF	RB, SCL	
	NOP		
	BTFSS	RB, SDA	;若 SDA=0,
	GOTO	MAKE_S	;则 I ² C 总线忙
M_START0			
	BSF	RB, SCL	
	BCF	RB, SDA	;发起始位
	MOVLW	SDA_OUT	
	TRIS	RB	
	MOVLW	10100000B	;写命令及 I ² C 器件地址值
	BTFSC	FLAG, READ_B	
	MOVLW	10100001B	;读命令及 I ² C 器件地址值
	MOVWF	REG3	;送值给 REG3
	MOVLW	REG3	
	MOVWF	FSR	
W_BYTE			
	MOVLW	SDA_OUT	
	TRIS	RB	
	MOVLW	8H	
	MOVWF	BIT_COUNT	;再次送 8bit
BIT_LOOP			
	NOP		
	RLF	0H	
	BCF	RB, SCL	;时钟置低
	BTFSS	STATUS, C	
	GOTO	Y+3	
	BSF	RB, SDA	
	GOTO	Y+2	

	BCF	RB, SDA	
	NOP		
	BSF	RB, SCL	
	DECFSZ	BIT_COUNT	
	GOTO	BIT_LOOP	
	NOP		
	BCF	RB, SCL	
	MOVLW	SDA_IN	;SDA 置输入
	TRIS	RB	
	BSF	RB, SCL	
	NOP		
	BTFSC	RB, SDA	;检测是否有应答?
	GOTO	M_START0	;没有应答, 重新发送
	BCF	RB, SCL	
	RETLW	0	
W_STOP			
M_STOP			
	BCF	RB, SDA	
	MOVLW	SDA_OUT	;发终止位
	TRIS	RB	
	NOP		
	BSF	RB, SCL	;在 SCL 为高时置 SDA 为高
	NOP		
	BSF	RB, SDA	
	NOP		
	BCF	RB, SCL	
	RETLW	0H	
; 读 24LCXX 中数据			
RD_BYTES			
	MOVWF	BYTE_COUNT	;输入所要读的字节数
	MOVLW	REG1	
	MOVWF	FSR	;数据将保存在从 REG1 开始的寄存器单
			;元
RNXTB			
	BCF	RB, SCL	
	MOVLW	SDA_IN	
	TRIS	RB	
	MOVLW	8H	
	MOVWF	BIT_COUNT	
RNX			
	BCF	RB, SCL	
	NOP		
	BCF	STATUS, C	
	BTFSC	RB, SDA	;读 1bit
	BSF	STATUS, C	
	RLF	0H	
	BSF	RB, SCL	
	NOP		
	DECFSZ	BIT_COUNT	;读完 8bit?
	GOTO	RNX	

	INCF	FSR	
	MOVLW	SDA_OUT	
	BCF	RB, SCL	
	TRIS	RB	
	DECFSZ	BYTE_COUNT	;字节都读完?
	GOTO	T_ACKG	
R_STOP			
	GOTO	M_STOP	;发停止位
;			;读完一个字节, 必须发个应答信号
T_ACKG			
	BCF	RB, SDA	
	NOP		
	BSF	RB, SCL	
	NOP		
	GOTO	RNXTB	
;主程序			
MAIN			
	CLRF	FLAG	
	MOVLW	SDA_IN	;置 SDA 为输入, 其他为输出口
	TRIS	RB	
;	.		
;	:		
;页写操作			
PAGE_WRITE			
	MOVLW	ADRESS	
	MOVWF	REG2	;送地址给 REG2
	MOVLW	REG1	
	MOVWF	FSR	
	MOVLW	8	
	MOVWF	BYTE_COUNT	
SD1	MOVLW	DATA	;送数据给从 REG1 开始的寄存器单元
	MOVWF	F0	
	INCF	FSR	
	DECFSZ	BYTE_COUNT	
	GOTO	SD1	
	CALL	ACK_CHECK	;发起始位和 I ² C 器件地址
	INCF	FSR	
	CALL	W_BYTE	;发送地址
	MOVLW	8	
	MOVWF	BYTE_COUNT	;最多一页可写 8byte
SD2	INCF	FSR	
	CALL	W_BYTE	;发送 1byte 数据
	DECFSZ	BYTE_COUNT	;写完?
	GOTO	SD2	;写下一 byte
	CALL	W_STOP	;发停止位
OVER2			
;	.		
;	:		
;单字节写			
BYTE_WRITE			

```

MOV LW    ADDRESS          ;送地址给 REG2
MOV WF    REG2

MOV LW    DATA            ;送数据给 REG1
MOV WF    REG1

CALL      ACK_CHECK        ;发起始位和 I2C 器件地址
INCF      FSR
CALL      W_BYTE           ;发送地址
INCF      FSR
CALL      W_BYTE           ;发送数据
CALL      W_STOP           ;发停止位

OVER1
;
;
;读操作
READ

CALL      ACK_CHECK        ;发起始位和 I2C 器件地址
MOV LW    ADDRESS          ;送地址到 REG2
MOV WF    REG2
INCF      FSR
CALL      W_BYTE           ;发送地址
BSF       FLAG, READ_B
CALL      ACK_CHECK        ;发送读命令
BCF       FLAG, READ_B
MOV LW    BYTE             ;"BYTE" 字节要读
CALL      RD_BYTES         ;读数据

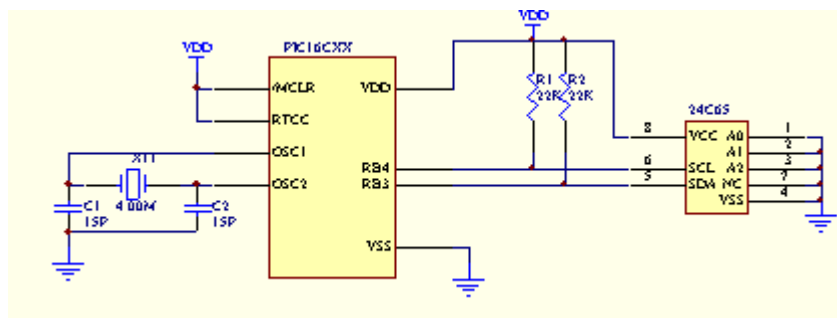
OVER
NOP
;
;

```

六、16CXX 和 24LC65 的联接

（一）电路设计

24LC65 和 24LC01 的读写时序基本相同，只是 24LC65 容量为 8K*8，在送完 I²C 器件地址后，还要发送 2 字节的数据地址。



（二）程序清单

```

;

```

STATUS	EQU	3	;标志寄存器
FSR	EQU	4	
RA	EQU	5	
RB	EQU	6	
C	EQU	0	
Z	EQU	2	
;			
SDA	EQU	3	;I ² C 总线数据口
SCL	EQU	4	;I ² C 总线时钟口
SDA_IN	EQU	08H	;置 SDA 为输入口
SDA_OUT	EQU	0	;置 SDA 为输出口
FLAG	EQU	8H	
BIT_COUNT	EQU	9H	
BYTE_COUNT	EQU	0AH	
CONTROL	EQU	0BH	
REG3	EQU	12H	
REG2	EQU	13H	
REG1	EQU	14H	
REG0	EQU	15H	
READ_B	EQU	0	
BYTE	EQU	8	
DATA	EQU	5AH	
ADRESSH	EQU	07H	
ADRESSL	EQU	10H	
ADRESSH	EQU	07H	
;-----			
	ORG	1FFH	
	GOTO	MAIN	
	ORG	0	
MAKE_S	BSF	RB, SDA	;把 SDA 强制拉高
	MOVLW	SDA_OUT	
	TRIS	RB	;置 SDA 输出
	NOP		
ACK_CHECK			;发起始位和 I ² C 器件地址
	MOVLW	SDA_IN	;置 SDA 输入
	TRIS	RB	
	NOP		
M_START	BSF	RB, SCL	
	NOP		
	BTFSS	RB, SDA	;若 SDA=0,
	GOTO	MAKE_S	;则 I ² C 总线忙
M_START0	BSF	RB, SCL	
	BCF	RB, SDA	;发起始位

	MOVLW	SDA_OUT	
	TRIS	RB	
	MOVLW	10100000B	;写命令及 I ² C 器件地址值
	BTFSC	FLAG, READ_B	
	MOVLW	10100001B	;读命令及 I ² C 器件地址值
	MOVWF	REG3	;送给 REG3
	MOVLW	REG3	
	MOVWF	FSR	
W_BYTE			
	MOVLW	SDA_OUT	
	TRIS	RB	
	MOVLW	8H	
	MOVWF	BIT_COUNT	;再次送 8bit
BIT_LOOP			
	NOP		
	RLF	0H	
	BCF	RB, SCL	;时钟置低
	BTFSS	STATUS, C	
	GOTO	^+3	
	BSF	RB, SDA	
	GOTO	^+2	
	BCF	RB, SDA	
	NOP		
	BSF	RB, SCL	
	DECFSZ	BIT_COUNT	
	GOTO	BIT_LOOP	
	NOP		
	BCF	RB, SCL	
	MOVLW	SDA_IN	;SDA 置输入
	TRIS	RB	
	BSF	RB, SCL	
	NOP		
	BTFSC	RB, SDA	;检测是否有应答?
	GOTO	M_START0	;没有应答, 重新发送
	BCF	RB, SCL	
	RETLW	0	
W_STOP			
M_STOP			
	BCF	RB, SDA	
	MOVLW	SDA_OUT	;发终止位
	TRIS	RB	
	NOP		
	BSF	RB, SCL	;在 SCL 为高时置 SDA 为高
	NOP		
	BSF	RB, SDA	
	NOP		
	BCF	RB, SCL	
	RETLW	0H	
; 读 24LCXX 中数据			
RD_BYTES			

	MOVWF	BYTE_COUNT	;输入所要读的字节数
	MOVLW	REG1	
	MOVWF	FSR	;数据将保存在从 REG1 开始的寄存器单元
RNXTB			
	BCF	RB, SCL	
	MOVLW	SDA_IN	
	TRIS	RB	
	MOVLW	8H	
	MOVWF	BIT_COUNT	
RNX			
	BCF	RB, SCL	
	NOP		
	BCF	STATUS, C	
	BTFSC	RB, SDA	;读 1bit
	BSF	STATUS, C	
	RLF	0H	
	BSF	RB, SCL	
	NOP		
	DECFSZ	BIT_COUNT	;读完 8bit?
	GOTO	RNX	
	INCF	FSR	
	MOVLW	SDA_OUT	
	BCF	RB, SCL	
	TRIS	RB	
	DECFSZ	BYTE_COUNT	;字节都读完?
	GOTO	T_ACKG	
R_STOP			
	GOTO	M_STOP	;发停止位
;			;读完一个字节，必须发个应答信号
T_ACKG			
	BCF	RB, SDA	
	NOP		
	BSF	RB, SCL	
	NOP		
	GOTO	RNXTB	
;主程序			
MAIN			
	CLRF	FLAG	
	MOVLW	SDA_IN	;置 SDA 为输入，其他为输出口
	TRIS	RB	
;	.		
;	:		
;页写操作			
PAGE_WRITE			
	MOVLW	ADRESSH	
	MOVWF	REG2	;送地址高位给 REG2
	MOVLW	ADRESSL	
	MOVWF	REG1	;送低位地址给 REG1
	MOVLW	REG0	
	MOVWF	FSR	

	MOVLW	8	
	MOVWF	BYTE_COUNT	
SD1	MOVLW	DATA	;送数据给从 REG0 开始的寄存器单元
	MOVWF	F0	
	INCF	FSR	
	DECFSZ	BYTE_COUNT	
	GOTO	SD1	
	CALL	ACK_CHECK	;发起始位和 I ² C 器件地址
	INCF	FSR	
	CALL	W_BYTE	;发送高位地址
	INCF	FSR	
	CALL	W_BYEE	;发送低位地址
	MOVLW	8	
	MOVWF	BYTE_COUNT	;最多一页可写 8byte
SD2	INCF	FSR	
	CALL	W_BYTE	;发送 1byte 数据
	DECFSZ	BYTE_COUNT	;写完?
	GOTO	SD2	;写下一 byte
	CALL	W_STOP	;发停止位
OVER2			
;	.		
;	:		
;单字节写			
BYTE_WRITE			
	MOVLW	ADRESSH	;送高位地址给 REG2
	MOVWF	REG2	
	MOVLW	ADRESSL	
	MOVWF	REG1	;送低位地址给 REG1
	MOVLW	DATA	;送数据给 REG0
	MOVWF	REG0	
	CALL	ACK_CHECK	;发起始位和 I ² C 器件地址
	INCF	FSR	
	CALL	W_BYTE	;发送高位地址
	INCF	FSR	
	CALL	W_BYTE	;发送低位地址
	INCF	FSR	
	CALL	W_BYTE	;发送数据
	CALL	W_STOP	;发停止位
OVER1			
;	.		
;	:		
;读操作			
READ			
	CALL	ACK_CHECK	;发起始位和 I ² C 器件地址
	MOVLW	ADRESSH	
	MOVWF	REG2	;送高位地址到 REG2
	MOVLW	ADRESSL	
	MOVWF	REG1	;送低位地址到 REG1
	INCF	FSR	
	CALL	W_BYTE	;发送高位地址

```

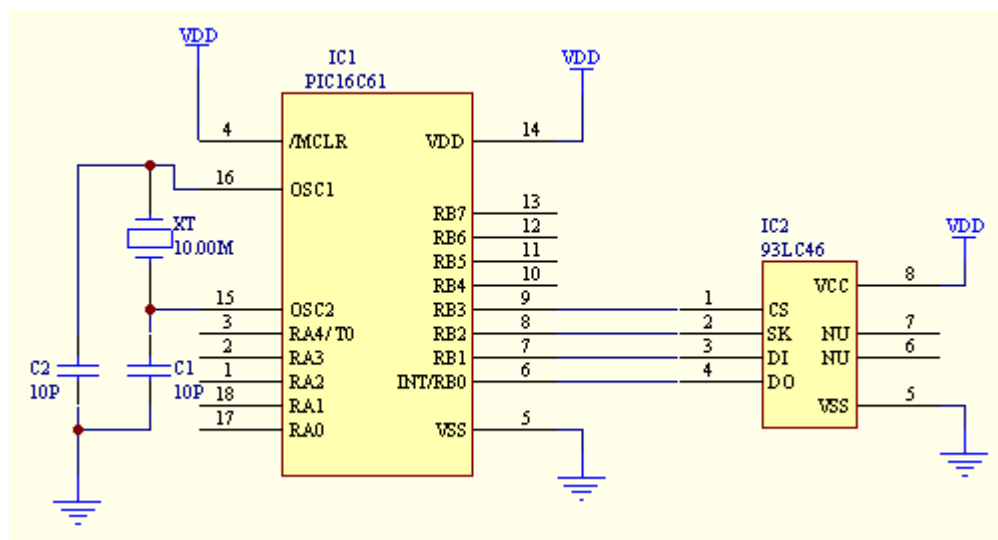
        INCF      FSR
        CALL      W_BYTE      ;发送低位地址
        BSF       FLAG, READ_B
        CALL      ACK_CHECK   ;发送读命令
        BCF       FLAG, READ_B
        MOVLW     BYTE        ;"BYTE" 字节要读
        CALL      RD_BYTES    ;读数据
OVER:    NOP
;
;

```

七、16CXX 和 93LC46 的联接

（一）电路设计

93LCXX 系列为三线式的 E²PROM 存储器件。本例以 PIC16C61 和 93LC46 为例，说明 PIC16CXX 和 93LCXX 的接口方法。



（二）程序清单

```

include    "16cxx.equ"
REGW      EQU      10H
REGST     EQU      11H
TEMR      EQU      12H
DATAH     EQU      13H
DATAL     EQU      14H
ADRESS    EQU      15H

CS46      EQU      3      ;RB3 选通 93C46
SK        EQU      2      ;RB2 发 93C46 时钟
DI        EQU      1      ;RB1 为写 93C46 数据口
DO        EQU      0      ;RB0 为读 93C46 数据口

;-----
                ORG      0      ;复位入口
                GOTO     MAIN

```



```

;-----
SERVICEINT
    ORG      4                                ;中断向量
    MOVWF    REGW                            ;中断服务程序
    SWAPF    STATUS, 0                      ;保存 W 值
    MOVWF    REGST                          ;保存标志寄存器值
    BCF      INTCON, INTF                    ;清 INT 中断标志
    BCF      INTCON, INTE                    ;关 INT 中断
    BCF      RB, CS46
    CALL     EWDS                            ;调禁止写 93C46 子程序
    SWAPF    REGST, 0
    MOVWF    STATUS                          ;恢复 STATUS 值
    SWAPF    REGW, 1                        ;恢复 W 值
    SWAPF    REGW, 0                        ;保护和恢复现场时不能用“MOVFW”指令
    RETURN                                   ;以免影响"Z"标志位

EWDS
    CLRF     TEMR                            ;禁止写 93C46 子程序
    GOTO     START_BYTE

ERAL
    MOVLW    2FH                            ;擦除子程序
    MOVWF    TEMR
    GOTO     START_BYTE

EWEN
    MOVLW    3FH                            ;写使能子程序
    MOVWF    TEMR

LOOP1
    BTFSC    INTCON, INTE                    ;若 INTE=1，则上次写未完
    GOTO     LOOP1                          ;等待

START_BYTE
START_BITS
    BSF      RB, CS46
    BCF      RB, SK
    NOP
    BSF      RB, DI
    NOP
    BSF      RB, SK
    NOP
    BCF      RB, SK

;-----
TRMIT
    MOVWLW   8                                ;送一个数给 93C46
    MOVWF    BIT_COUNT                      ;每次送 8bit 数据

TRM0
    BCF      3H, C
    BCF      RB, SK
    RLF      TEMR
    BTFSS    3H, C
    GOTO     KS1
    BSF      RB, DI
    GOTO     KS2

KS1
    BCF      RB, DI
KS2
    NOP

```

```

BSF      RB, SK
DECFSZ   BIT_COUNT
GOTO     TRM0
RETLW    0H
;-----
;写 93C46 子程序, 地址为 ADRESS, 数据为 DATAH, DATAL
WRITE

CALL     EWEN           ;写使能
BCF      RB, CS46
MOVLW    40H
IORWF    ADRESS, 0
MOVWF    TEMR           ;送写命令和地址给 93LC46
CALL     START_BYTE    ;
MOVFW    DATAH
MOVWF    TEMR           ;写高 8bit
CALL     TRMIT
MOVFW    DATAL
MOVWF    TEMR           ;写低 8bit
CALL     TRMIT
CALL     REDAY          ;检测是否写完成
RETLW    0
;-----
REDAY

BCF      RB, CS46
NOP
BSF      RB, CS46
BSF      OPTION, INTEDG ;设置 INT 上升沿中断
BSF      INTCON, INTE   ;开 INT 中断
BSF      INTCON, GIE
RETLW    0
;-----
;读 93C46 中数据, 地址为 ADRESS, 数据存贮到 DATAH, DATAL
READ

MOVLW    3FH
ANDWF    ADRESS, 0
IORLW    80H
MOVWF    TEMR
CALL     START_BYTE    ;送读命令和地址给 93C46
NOP
BCF      RB, SK
NOP
BTFSC    RB, DO         ;若 D0 口为 1
GOTO     R_ERROR        ;则 93C46 读失败

RECEIVE

MOVLW    10H            ;接收 16bit
MOVWF    BIT_COUNT

RV0

BSF      RB, SK
NOP
NOP

```

```

        BSF      3H, C
        BCF      RB, SK
        NOP
        BTFSS    RB, DO
        BCF      3H, C
        RLF      DATAL
        RLF      DATAH
        DECFSZ   BIT_COUNT      ;16bit 接收完
        GOTO     RV0             ;否, 继续
        RETLW    0               ;
R_ERROR      .
:
;-----
MAIN
        BSF      STATUS, RP0      ;选寄存器体 1
        MOVLW    0
        MOVWF    TRISA             ;置 RA 口存
        MOVLW    00000001B
        MOVWF    TRISB             ;置 RB0 输入, 其他输出
        BCF      STATUS, RP0      ;选寄存器体 0
        CLRF     RB
        CLRF     RA
        CLRF     INTCON            ;清中断控制寄存器
;
;
;-----
;芯片擦除
CLRALL
        CALL     EWEN              ;写使能
        CALL     ERAL              ;送擦除指令
        CALL     READY             ;检测是否擦除完
        .
        :
;-----
;写一次 93C16
WRITE_BYTE
        MOVLW    23H
        MOVWF    ADRESS            ;地址存贮在 ADRESS
        MOVLW    55H
        MOVWF    DATAH            ;数据存贮在 DATAL1
        MOVLW    0AAH
        MOVWF    DATAL             ;和 DATAL
        CALL     WRITE             ;写操作
        .
        :
;-----
;读 93C46
READ_BYTE
        MOVLW    23H
        MOVWF    ADRESS            ;地址存贮在 ADRESS

```

```

CALL      READ
.
:
;-----
END

```

八、16CXX SPI 接口和 93LCXX 的联接

（一）电路设计

PIC16C64/74 带有 SPI 接口。虽然 93 系列没有 SPI 接口，但经过软件编程，很容易通过 SPI 接口与 CPU 通讯。SPI 总线每次都传输 8bit（1BYTE），第一次传送起始位，读写命令和地址最高位 A8，第二次传送低 8 位地址，如下所示：

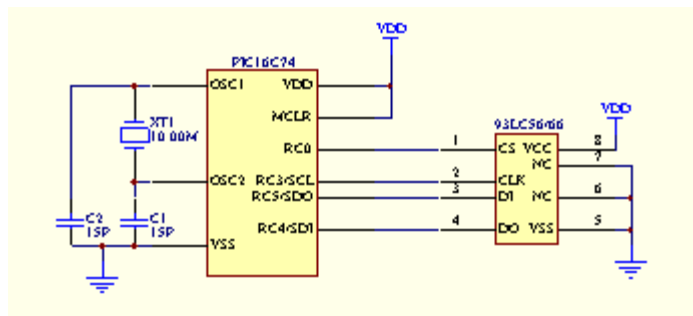
0	0	0	0	SB	OP1	OP0	A8
---	---	---	---	----	-----	-----	----

第一次传送

A7	A6	A5	A4	A3	A2	A1	A0
----	----	----	----	----	----	----	----

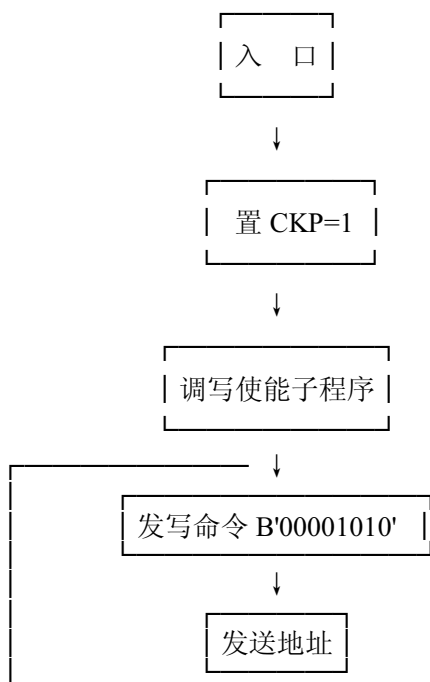
第二次传送

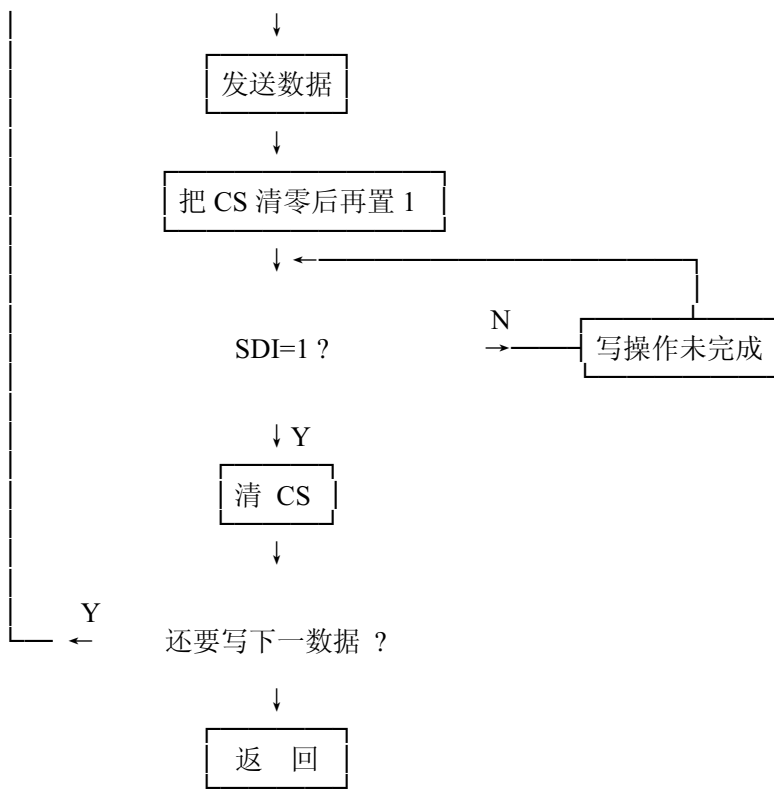
图一为 16C64/74 和 24LC56/66 的联接电路图。



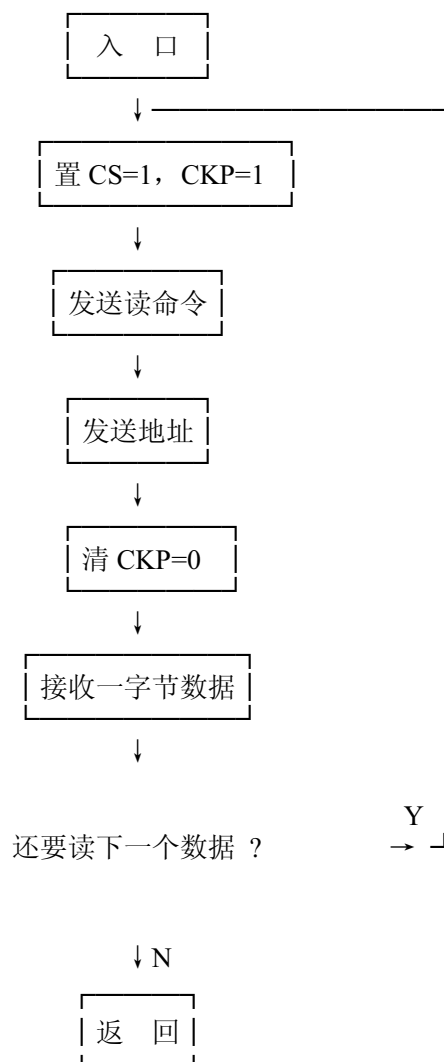
（二）程序流程图

1、写 93LCXX 流程图



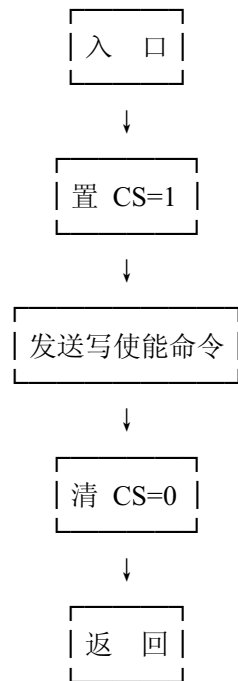


2、读 93LCXX 流程图

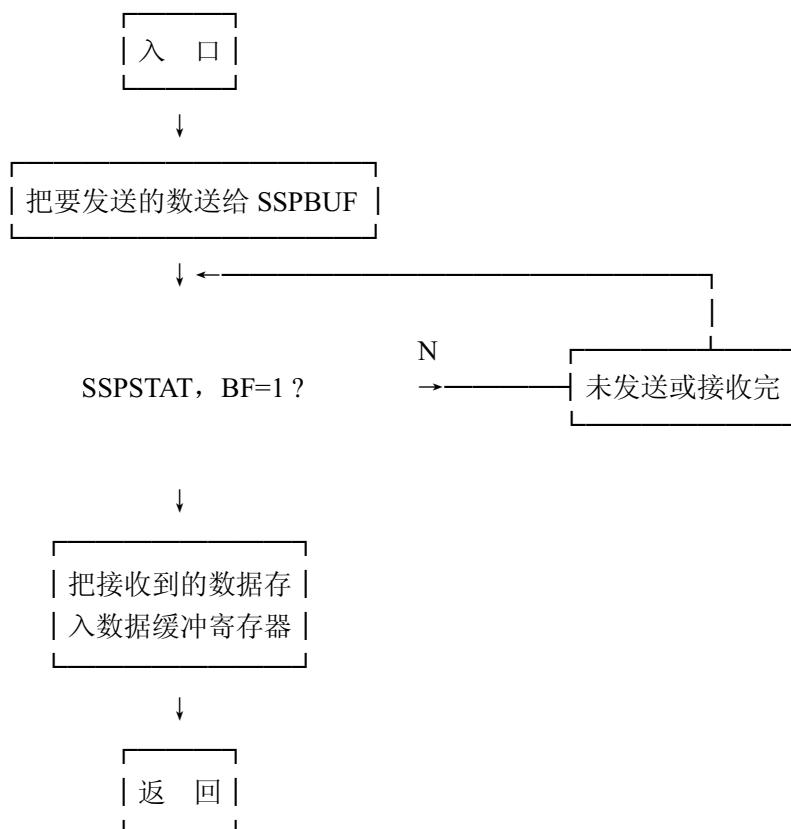


[注] 发送状态时 CKP 要置 1，接收状态时 SKP 要清 0。

3、写使能子程序流程图



4、发送接收一字节子程序流程图



(三) 程序清单

;本程序把数据写入 93LC56/66 地址为 10H-13H 空间，
;然后从 10H-13H 读回数据，存入 20H-23H 的寄存器

```
BXDATA      EQU      24H  
TXDATA      EQU      25H
```

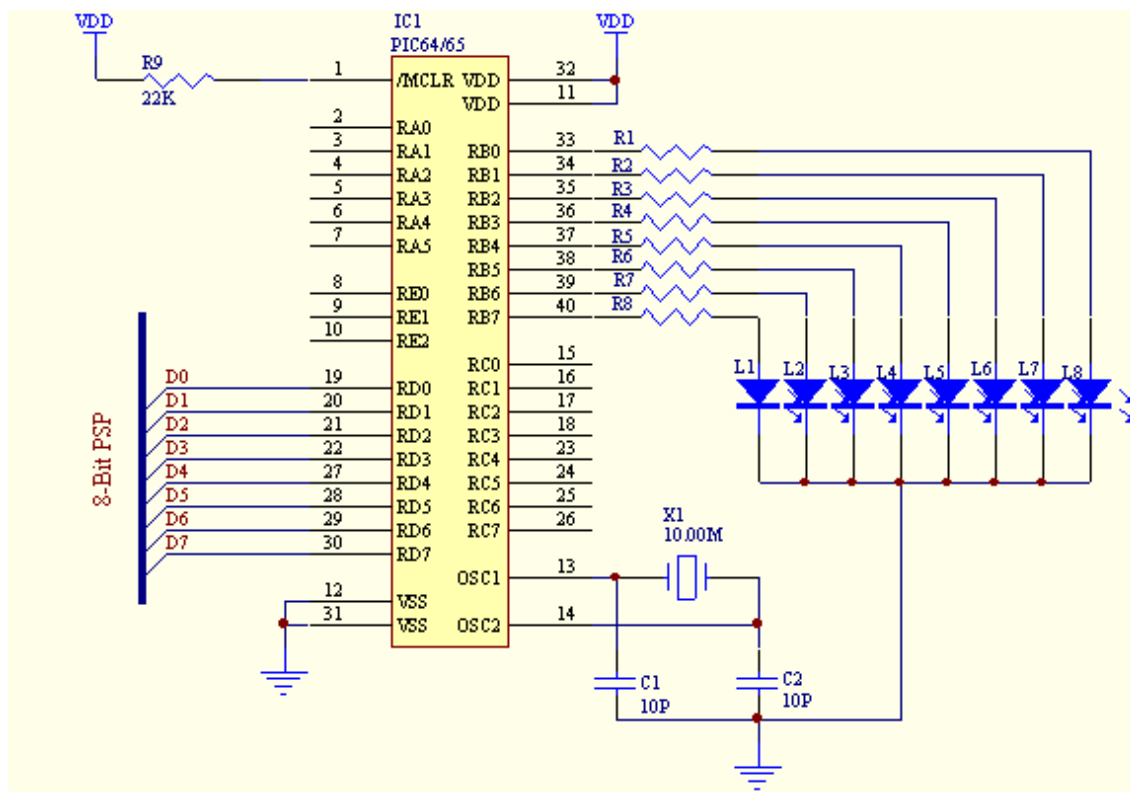
ADDR	EQU	26H	
LOOPS	EQU	27H	
LOOPS2	EQU	28H	
HIBYTE	EQU	29H	
LOBYTE	EQU	2AH	
DATBYT	EQU	2BH	
;			
CS	EQU	0	
SDI	EQU	4	
;			
INCLUDE	"16CXX.EQU"		
	ORG	0	;复位地址
	GOTO	START	
;发送/接收一字节			
OUTPUT	MOVWF	SSPBUF	;把数送给 SSPBUF 后开始发送
LOOP1	BSF	STATUS, RP0	
	BTFSS	SSPSTAT, BF	;发送/接收完?
	GOTO	LOOP1	;否, 继续
	BSF	STATUS, RP0	
	MOVFW	SSPBUF	;从 SSPBUF 中取出接到的数据
	MOVWF	RXDATA	;存贮到接收寄存器
	RETLW	0	
;写使能子程序			
EWEN	BCF	STATUS, RP0	
	BSF	RC, CS	;置 CS=1
	MOVLW	B'00001001'	;BIT3=1 为起始位
	CALL	OUTPUT	
	MOVLW	B'10000000'	;接下去'0011 为使能命令
	CALL	OUTPUT	
	BCF	RC, CS	;置 CS=0, 启动 E ² PROM
			;内部写时钟
	RETLW	0	;至少要拉低 CS 250ns
;写 1BYTE 字节到 93LCXX			
WRITE			
	BCF	STATUS, RP0	
	BSF	RC, CS	;置 CS=1
	MOVFW	HIBYTE	;先发送写命令
	CALL	OUTPUT	
	MOVFW	FSR	;接着发送地址
	CALL	OUTPUT	
	MOVFW	DATBYT	;然后发送数据
	CALL	OUTPUT	
	BCF	RC, CS	;清 CS=0, 启动 E ² PROM
			;内部写操作
	INCF	FSR	
	RETLW	0	
;从 93LCXX 读 1BYTE 字节			
READ	BCF	STATUS, RP0	
	BSF	RC, CS	;置 CS=1
	BSF	SSPCON, CKP	;置 CKP=1

	MOVFW	HIBYTE	;先发送读命令
	CALL	OUTPUT	
	MOVFW	LOBYTE	;接着发送地址
	CALL	OUTPUT	
	BCF	SSPCON, CKP	;清 CKP, 转到接收数据
	MOVLW	0	;此时送什么数无关紧要
	CALL	OUTPUT	
	BCF	RC, CS	;清 CS, 终止写命令
	MOVFW	RXDATA	;取出接收数据
	MOVWF	INDF	;存入数据缓冲区
	INCF	FSR	;下一次数据寄存器地址
	INCF	LOBYTE	;下一次 93LCXX 地址
	RETLW	0	
;主程序			
START	BCF	STATUS, RP0	
	CLRF	RC	
	BSF	STATUS, RP0	
	MOVLW	10H	;设置 RC 口状态
	MOVWF	TRISC	
	CLRF	PIE1	
	CLRF	INTCON	;禁止所有中断
	BCF	STATUS, RP0	
	MOVLW	31H	;SPI 主控方式, 设置 CLK/16
	MOVWF	SSPCON	;CKP=1
	CALL	EWEN	;写使能 93LCXX
	MOVLW	B'00001010'	;写命令码
	MOVWF	HIBYTE	
	MOVLW	10H	;欲写入的首地址为 10H
	MOVWF	FSR	
	MOVLW	5AH	;要写入的数据为 5AH
	MOVWF	DATBYT	
WRNEXT	CALL	WRITE	;写 1BYTE
	NOP		;清 CS 至少 250ns 后
	BSF	RC, CS	;置 CS=1
RBUSY	BTFSS	RC, SDI	;检测写数据是否完成
	GOTO	RBUSY	;未完, 总线非空, 继续等待
	BCF	RC, CS	;清 CS
	BTFSS	FSR, 2	;4BYTE 都写完?
	GOTO	WRNEXT	;否, 继续写下一个
;读回数据, 查看缓冲区内容就可知道读写			
;有无成功			
	MOVLW	20H	;数据缓冲区首址为 20H
	MOVWF	FSR	
	MOVLW	10H	;从 10H 地址开始读
	MOVWF	LOBYTE	
	MOVLW	B'00001100'	;写命令码
	MOVWF	HIBYTE	
RDNEXT	CALL	READ	
	BTFSS	FSR, 2	;读完 4BYTE?
	GOTO	RDNEXT	;否, 继续下一次
LIMBO	NOP		
	GOTO	LIMBO	
	END		

九、8 位并行口的使用

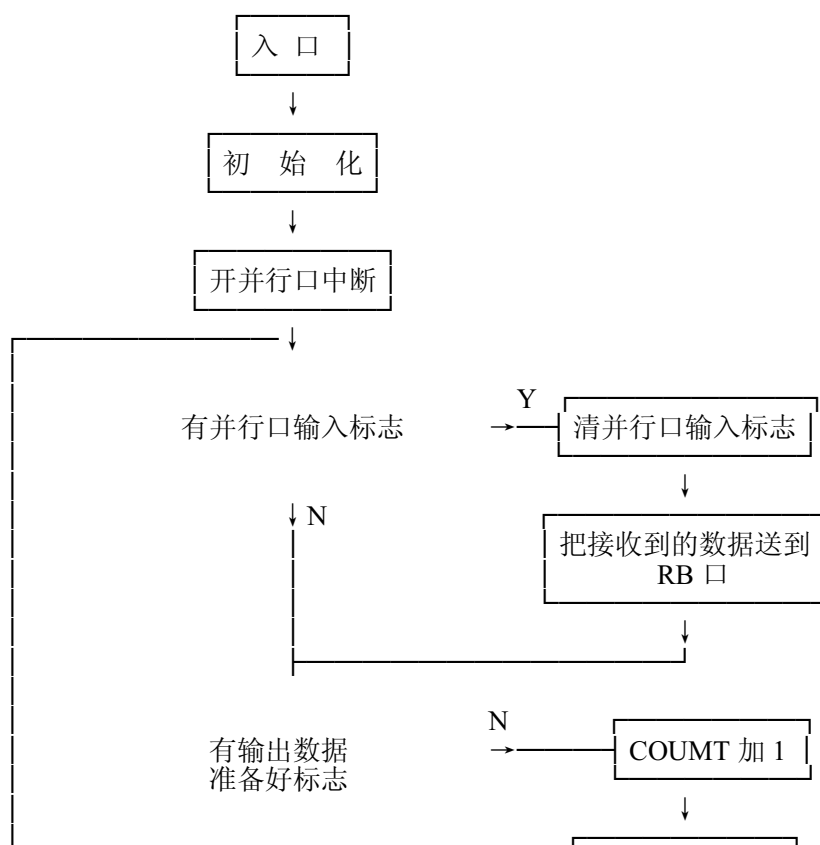
(一) 电路设计

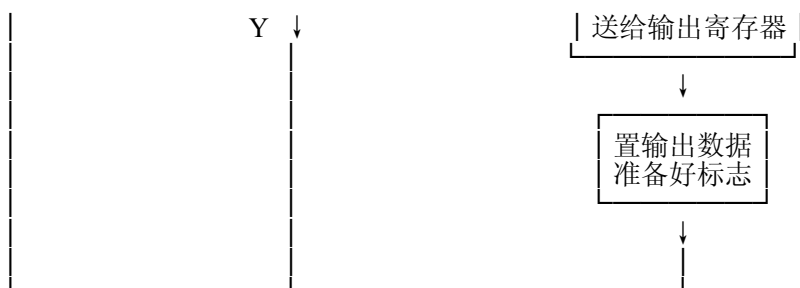
PIC16C64/74 带有一个 8 位并行口。本例由并行口读、写产生中断，读（对 CPU 是输出）中断时，把数据送出 PORT_D 口，计数寄存器加 1，并把新值送到输出队列，写（对 CPU 是输入）中断时，从 PORT_D 读取数据，并送给 RB 口。电路图如下：



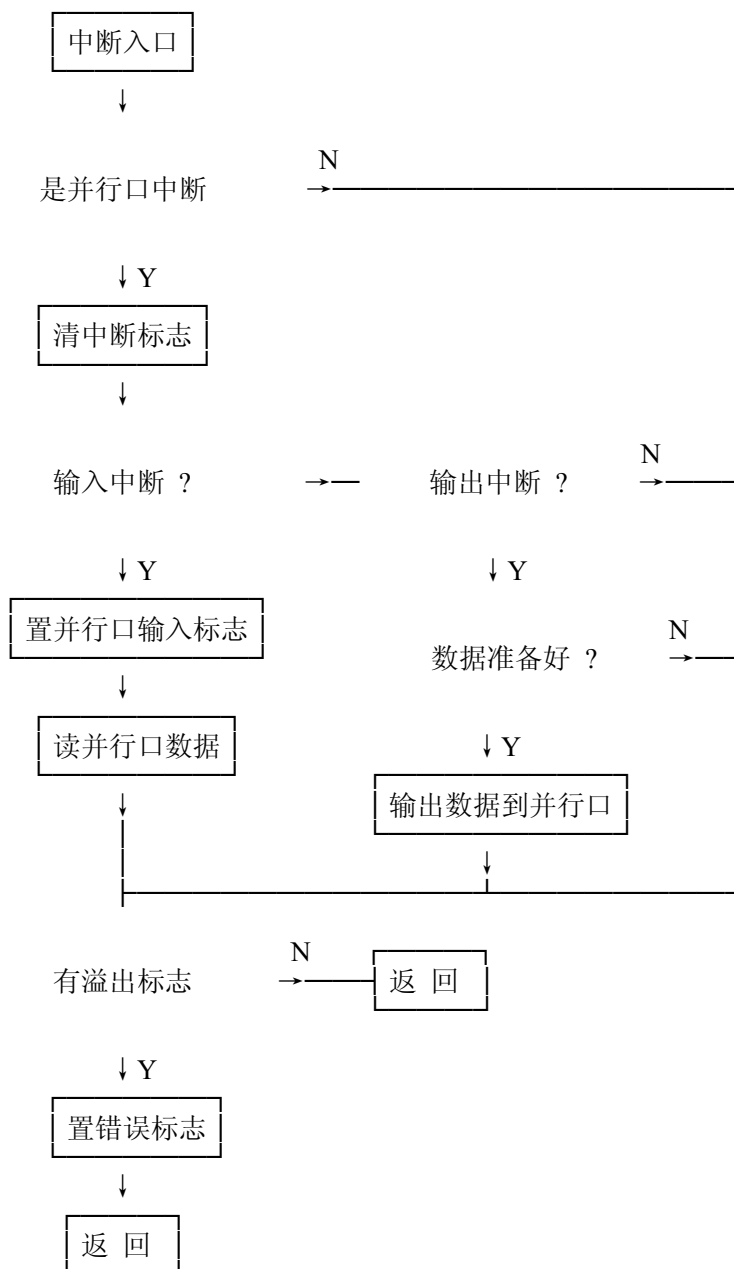
(二) 程序流程图

1、主程序





2、中断程序



(三) 程序清单

```

include    "16cxx.equ"
;寄存器定义:
FLAGREG    EQU        20H        ;标志位寄存器
OUTDATA    EQU        21H        ;输出寄存器
  
```

INDATA	EQU	22H	;输入寄存器
COUNT	EQU	23H	;计数寄存器
;标志位定义:			
ERROR	EQU	0	
OUTRDY	EQU	1	
INFULL	EQU	2	
;			
	ORG	00H	;复位地址
	GOTO	START	
;			
	ORG	04H	;中断向量
	GOTO	SERVICE_INT	
;			
START	CIRF	OUTDATA	
	CIRF	INDATA	
	BSF	STATUS, RP0	
	MOVLW	B'00010111'	
	MOVWF	TRISE	;并行口设置
	MOVLW	0	
	MOVWF	TRISB	;RB 口为输出口
	MOVLW	B'10000000'	
	MOVWF	PIE1	;开并行口中断
	BCF	STATUS, RP0	
	MOVFW	OUTDATA	
	MOVWF	PORT_D	
	MOVLW	B'11000000'	
	MOVWF	INTCON	;开中断允许
LOOP	BTFSS	FLAGREG, INFULL	;是否有输入数据
	GOTO	CHECKOUT	;没有, 转到检查输出
	BCF	FLAGREG, INFULL	;清输入标志
	MOVFW	INDATA	
	MOVWF	RB	;把数据输出到 RB 口
CHECKOUT			
	BTFSC	FLAGREG, OUTRDY	;检查输出数据是否准备好
	GOTO	LOOP	;已准备好
	INCF	COUNT	;没有, 则计数器加 1
	MOVFW	COUNT	
	MOVWF	OUTDATA	;把数据送给输入寄存器
	BSF	FLAGREG, OUTRDY	;置数据准备好标志
	GOTO	LOOP	
SERVICE_INT			
	BTFSS	PIR1, PSPIF	;是否并行口中断?
	GOTO	INTOUT	;不是, 结束中断
	BCF	PIR1, PSPIF	;清中断标志
	BSF	STATUS, RP0	
	BTFSS	TRISE, IBF	;输入数据准备好?
	GOTO	NOINPUT	;没有, 不是输入中断
	BCF	STATUS, RP0	;已准备好
	BSF	FLAGREG, INFULL	;置输入标志
	MOVFW	PORT_D	;读取并行口数据

	MOVWF	INDATA	;存入输入寄存器
NOINPUT	BTFSC	TRISE, OBF	;输出中断?
	GOTO	INOUT	;否, 结束中断
	BCF	STATUS, RP0	
	BTFSS	FLAGREG, OUTRDY	;输出数据准备好?
	GOTO	INOUT	;没有, 则结束中断
	MOVFW	OUTDATA, W	
	MOVWF	PORT_D	;数据送出并行口
	BCF	FLAGREG, OUTRDY	;清除输出数据准备好标志
INTOUT	BSF	STATUS, RP0	
	BTFSC	TRISE, IBOV	;是否溢出
	GOTO	INTERERROR	;溢出, 则出错
	BCF	STATUS, RP0	
	RETFIE		
INTERERROR			
	BCF	STATUS, RP0	
	BSF	FLAGREG, ERROR	;置错误标志
	RETFIE		
	END		

十、CPP 模式的应用例程

CPP 模式可工作在以下三种状态:

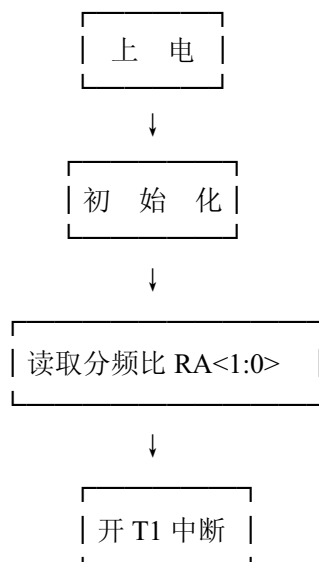
- 1、捕捉输入
- 2、比较输出
- 3、PWM (脉宽调制) 输出

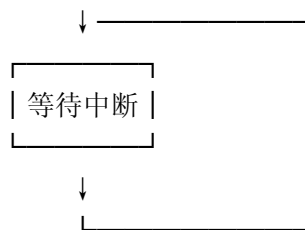
(一) PWM 输出

PWM 的周期由 PR2 决定, 脉宽比由 CCPR1L 值决定, 最高分辨率可达 10 位。本例由 RB 口输的值做为周期, RA<1:0>值作为 TMR2 的分频比, PORT_D 值为脉宽高 8 位, POR_E<1:0>为脉宽低 2 位。本程序上电时读取 RA<1:0>值做为 TMR2 分频比, 每次 TMR1 中断重新读取 RB、RD、RE 口值, 更新周期和脉宽。

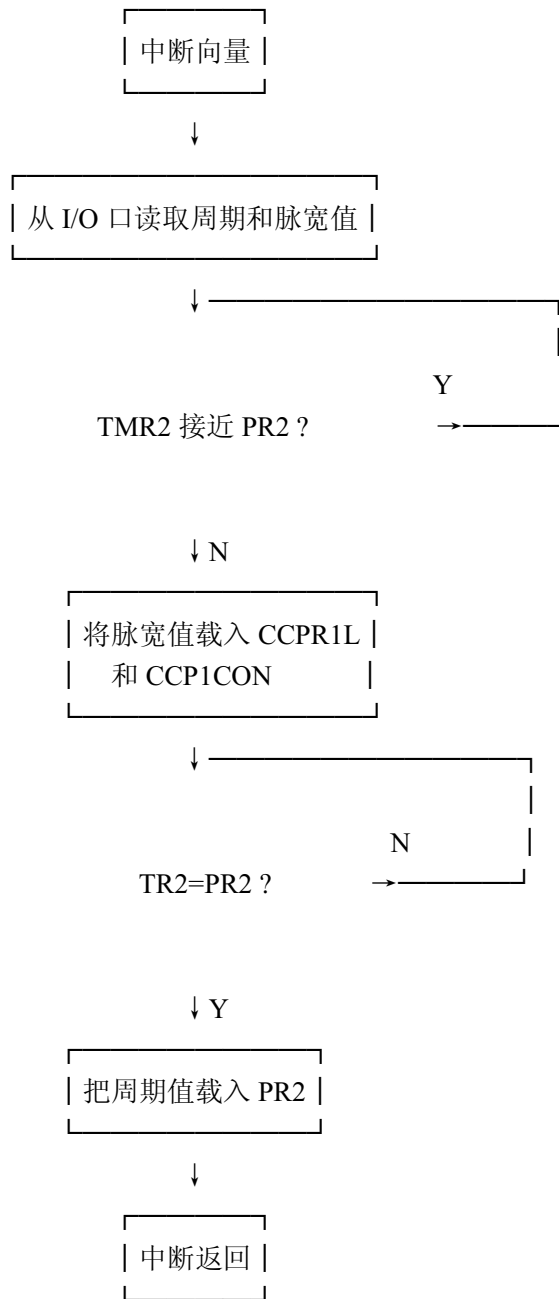
1、流程图

①主程序流程图





②中断服务程序



2、程序清单

```

COUNTER      EQU      21H
DC_HI        EQU      22H
DC_LO        EQU      23H
T2_PERIOD    EQU      0AH
include      '16CXX.EQU'

```

```

ORG          0

```

```

RESET

```

	GOTO	START	
	ORG	04H	;中断向量
PER_INT_V	BCF	STATUS, RP0	
	BTFSC	PIR1, TMR1IF	;T1 溢出中断
	GOTO	T1OVFL	;是, 中断处理
OTHER_INT	BSF	RA, 2	;其他中断
	BCF	RA, 2	;出错指示
	GOTO	OTHER_INT	
T1OVFL	BCF	PIR1, TMR1IF	;清 T1 中断标志
	MOVFW	PORT_D	;读取脉宽系数高 8 位
	MOVWF	DC_HI	
	MOVFW	PORT_E	;读取脉宽系数 (低 2 位)
	MOVWF	DC_LO	
	MOVFW	RB	
	BSF	STATUS, RP0	
	MOVWF	T2_PERIOD	;读取脉冲周期
	BCF	STATUS, RP0	
WAIT_DC	MOVFW	TR2	;读 T2 定时器值
	SUBWF	PR2, W	
	ANDLW	0FH	;检测 TMR2 是否接近 PR2
	BTFSC	STATUS, 2	
	GOTO	WAIT_DC	
	;要在 TMR2 未接近 PR2 时作以下操作		
	MOVFW	DC_HI	
	MOVWF	CCPR1L	;载入脉宽高 8 位
	MOVLW	0FH	
	ANDWF	CCP1CON	;设置脉宽低 2 位
	BTFSC	DC_LO, 1	
	BSF	CCP1CON, CCP1X	
	BTFSC	DC_LO, 0	
	BSF	CCP1CON, CCP1Y	
	BCF	PIR1, TMR2IF	;清 TR2=PR2 标志
WAIT_PR	BTFSS	PIR1, TMR2IF	;等待 TR2=PR2
	GOTO	WAIT_PR	;
	MOVFW	T2_PERIOD	
	MOVWF	PR2	;装入周期
	RETFIE		
;主程序			
START	BCF	STATUS, RP0	
	CLRF	TMR1H	
	CLRF	TMR1L	
	CLRF	INTCON	
	CLRF	PIR1	
	MOVLW	80H	
	MOVWF	OPTION	
	CLRF	PIE1	;关所有外部中断
	MOVLW	0FFH	;RA 口为数字口
	MOVWF	ADCON1	

```

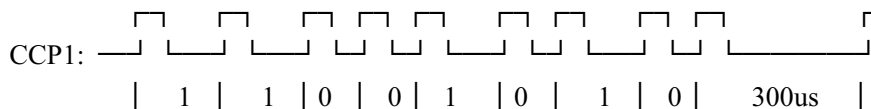
        MOVLW    0FFH
        MOVWF    TRISA
        MOVWF    TRISB
        CLRF     TRISC
        MOVWF    TRISD
        MOVWF    TRISE
        MOVWF    PR2                ;先设 PWM 周期最大
        BSF      PIE1, TMR1IE      ;开 TMR1 中断
        BCF      STATUS, RP0
        MOVLW    0CH                ;设置 CCP 为 PWM 输出
        MOVWF    CCP1CON
        CLRF     PIR1
        CLRF     T1CON
        CLRF     T2CON
        BTFSC    RA, 0              ;读取 T2 分频比
        BSF      T2CON, 0
        BTFSC    RA, 1
        BSF      T2CON, 1
        BSF      INTCON, PEIE      ;开外部中断
        BSF      INTCON, GIE       ;开全体中断允许
        BSF      T1CON, TMR1ON     ;开定时器 T1
        BSF      T2CON, TMR2ON     ;开定时器 T2
LZZ      NOP
        GOTO     LZZ                ;等待中断

        END

```

（二）比较输出

1、CCP1 工作在比较输出时，TMR1 将与 CCPR1H:CCPR1L 比较，相等时产生中断，并根据 CCP1CON，0 位的值来置或清 CCP1。本例从 RB 口读入要发送的数据，后 CCP1 发送出去，数据的格式如下：



每发 1bit 的开始是 18.8us 的高电平作同步头，若此位是“0”则发 18.8us 低电平，若为“1”则发 37.6us 低电平，8bit 发完有 300us 的间歇时间，然后再发送下一数据，如上图所示为发送“0CAH”。

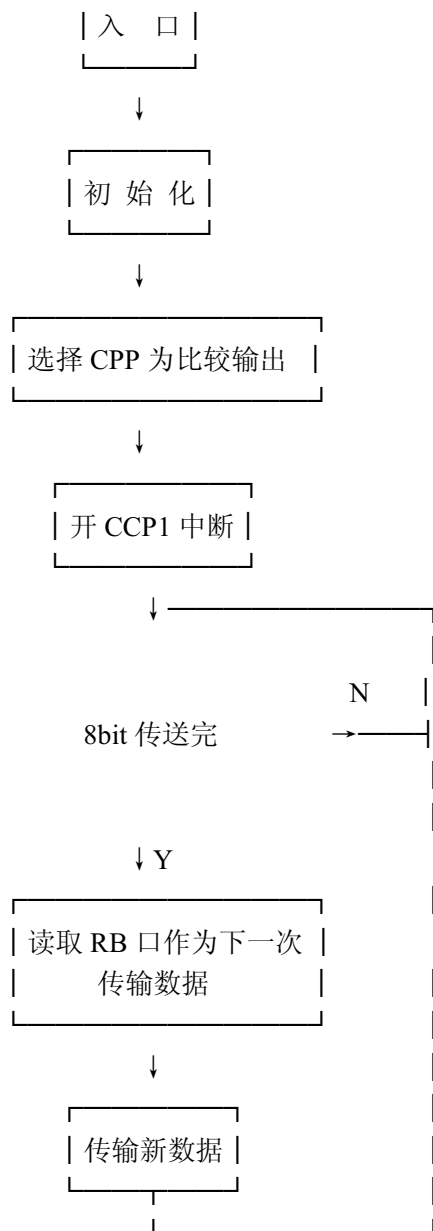
程序中发送脉冲宽度由 CCPR1H:CCPR1L 和 TMR1 的偏差来决定。每次 CCP1 中断后，不清除 TMR1 值，而是在当前的 CCPR1H:CCPR1L 值上加上相当的偏移量，以达到定时中断的效果。如果采用 10M 晶振，TMR1 设置为内部时钟且分频比为 1:1，则 TMR1 时钟为 2.5MHZ。

$$18.8\text{us 的偏移量} = \frac{18.8\text{us}}{0.4\text{us}} = 47$$

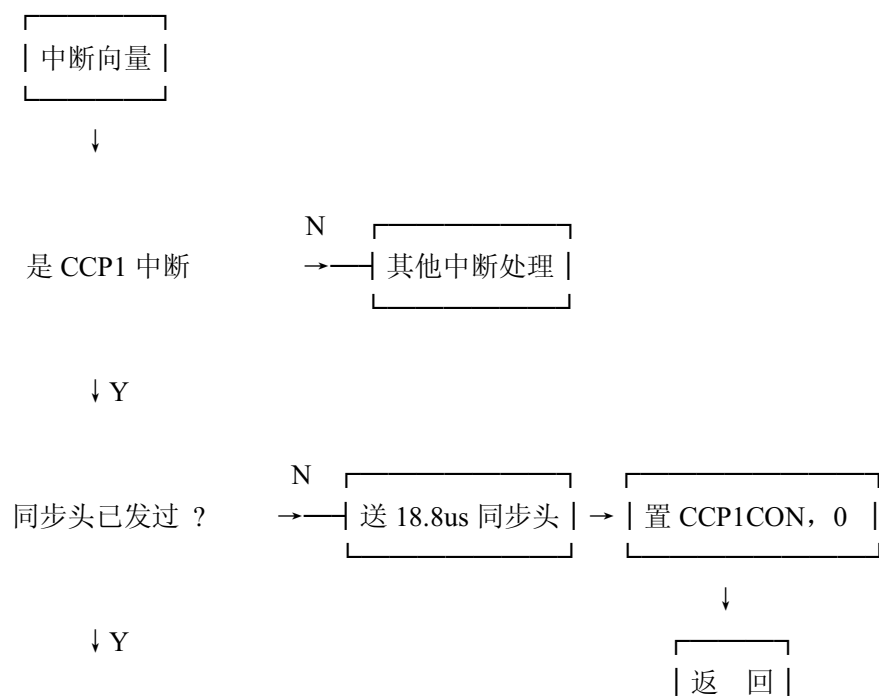
同样，37.6us 的偏移量为 94。

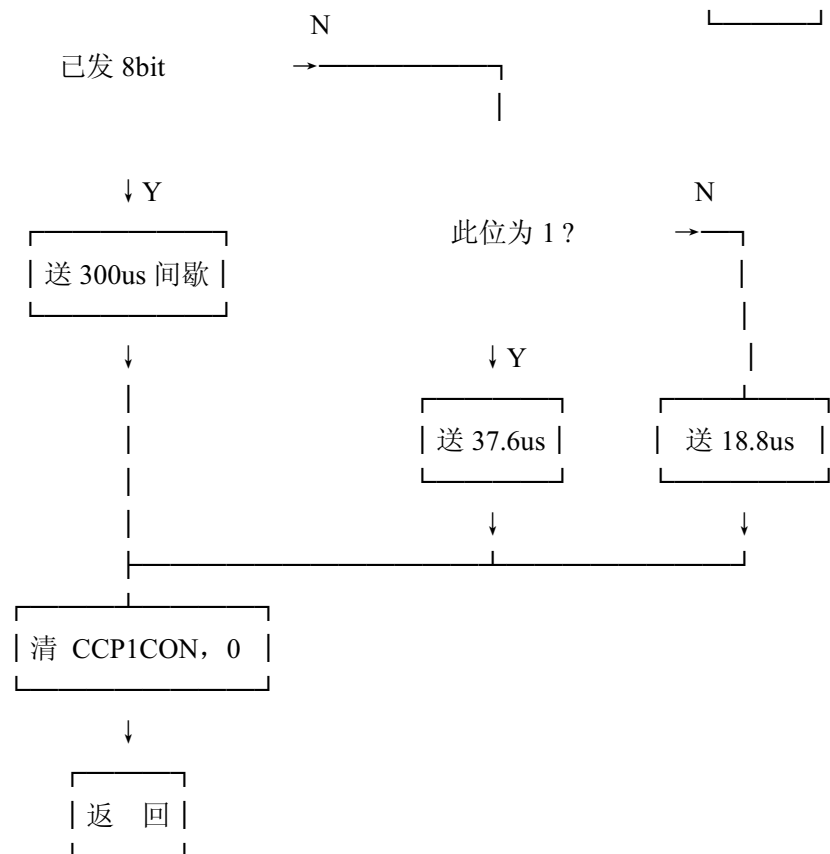
2、流程图

①主程序流程图



②中断服务子程序流程图





3、程序清单

```

XMIT_DATA    EQU    30H
DATA_CNT     EQU    31H
ONES_CNT     EQU    32H
CCP1_INT_CNT EQU    33H

```

```
include      '16CXX.EQU'
```

```

                ORG      0
                GOTO     START
                ORG      4
PER_INT_V      BCF      STATUS, RP0
                BTFSC    PIR1, CCP1IF      ;是比较中断?
                GOTO     CCP1_INT          ;是，中断处理
OTHER_INT      ;其他中断、出错
ERROR1         NOP
                GOTO     ERROR1

CCP1_INT       BCF      PIR1, CCP1IF      ;清 CCP1 中断标志
                INCF     CCP1_INT_CNT
                BTFSS    CCP1_INT_CNT, 0
                GOTO     SYNC_PULSE       ;送同步头
DATA_PULSE     DECF     DATA_CNT
                BTFSC    STATUS, Z        ;一个数据所有位都传完?
                GOTO     PERIOD_DELTA     ;是，延时 300us
                RLF      XMIT_DATA        ;取出下一位
                MOVLW    5EH              ;保持低电平 37.6us

```

	BTFSS	STATUS, C	;此位为"0"?
	MOVLW	2FH	;送低电平 18.8us
SEND_DATA			;
	ADDWF	CCPR1L	;更新比较寄存器值
	BTFSC	STATUS, C	
	INCF	CCPR1H	
	GOTO	RET_FIE	
SYNC_PULSE			;发同步头
	MOVLW	2FH	;保持高电平 18.8us
	ADDWF	CCPR1L	;更新比较寄存器
	BTFSC	STATUS, C	
	INCF	CCPR1H	
	BSF	CCP1CON, 0	;设置比较相等时 CCP1 清零
	RETFIE		
PERIOD_DELTA			;一个数传完后延时 300us
	MOVLW	0EH	
	ADDWF	CCPR1L	;更新比较寄存器
	BTFSC	STATUS, C	
	INCF	CCPR1H	
	MOVLW	2H	
	ADDWF	CCPR1H	
RET_FIE	BCF	CCP1CON, 0	;设置比较相等时 CCP1 置 1
	RETFIE		
;主程序			
START	BCF	STATUS, RP0	
	CLRF	TMR1H	
	CLRF	TMR1L	
	CLRF	INTCON	
	CLRF	PIR1	
	BSF	STATUS, RP0	
	MOVLW	80H	;RB 口取消弱上拉
	MOVWF	OPTION_R	
	CLRF	PIE1	;关闭所有外部中断
	CLRF	T1CON	;关 T1, 选择 T1 模式为内部时钟
	BSF	STATUS, RP0	
	MOVLW	0FFH	
	MOVWF	TRISB	;RB 口为输入口
	BSF	PIE1, CCP1IE	;开 CCP1 中断允许
	BCF	STATUS, RP0	
	BSF	INTCON, PEIE	;开外部中断允许
	BSF	INTCON, GIE	;开全体中断允许
	MOVLW	08H	;设置 CCP1 为比较输出
	MOVWF	CCP1CON	;比较相等时 CCP1 置 1
	MOVLW	09H	;每次传送 8bit
	MOVWF	DATA_CNT	
	MOVLW	0FFH	
	MOVWF	CCP1_INT_CNT	

	BSF	T1CON, TMR1ON	;开 T1 定时器
NEXT_BYTE			
WAIT	MOVFW	DATA_CNT	
	BTFSS	STATUS, Z	;8bit 传送完?
	GOTO	WAIT	;没有, 等待
	NOP		
	MOVFW	RB	
	MOVWF	XMIT_DATA	;读取新的传输数据
	MOVLW	0FFH	
	MOVWF	CCP1_INT_CNT	
	MOVLW	09H	;每次传送 8bit
	MOVWF	DATA_CNT	
	GOTO	NEXT_BYTE	
	END		

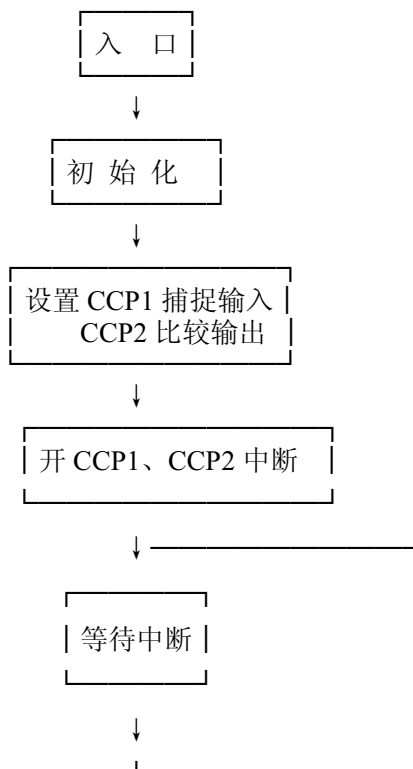
(三) 捕捉输入

1、程序设计

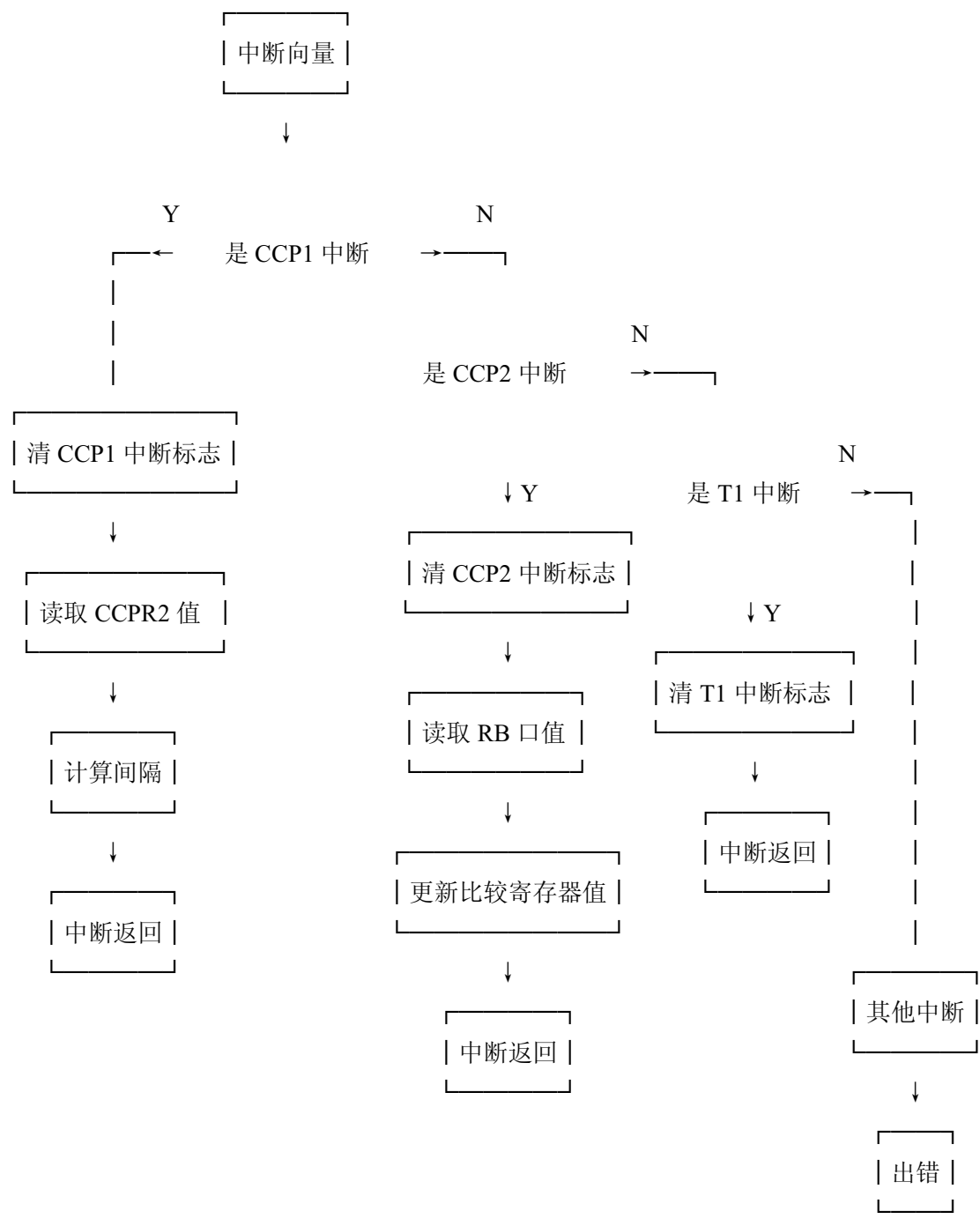
CCP 工作在捕捉输入时，CCPX 脚上有上升沿或下降沿将产生中断，并把此时 TMR1 的值存入 CCPRXH:CCPRXL 中。本例中，CCP1 作为捕捉输入，CCP2 作为比较输出，且 CCP2 的输出作为 CCP1 的输入。RB 口的输入值作为 CCP2 比较输出的间隔。CCP1 设置成每次上升沿捕捉中断，所以两次捕捉之间的间隔应该是 CCP2 比较输出间隔的 2 倍（即 RB 口值的 2 倍）。

2、程序流程图

①主程序



②中断程序流程图



3、程序清单

```
CCP2_INT_CNT    EQU    33H
CAPT_NEW_H      EQU    40H
CAPT_NEW_L      EQU    41H
CAPT_OLD_H      EQU    42H
CAPT_OLD_L      EQU    43H
```

```
include         'P16CXX.EQU'
```

```
                ORG     0
                GOTO    START
```

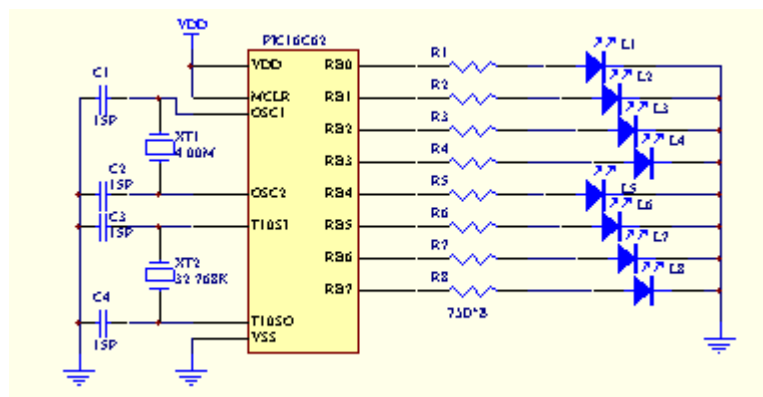
	ORG	4H	
PER_INT_V	BCF	STATUS, RP0	
	BTFSC	PIR1, CCP1IF	;是 CCP1 捕捉中断?
	GOTO	CAPTVRE	;是, 捕捉中断处理
	BTFSC	PIR2, CCP2IF	;是 CCP2 比较中断?
	GOTO	COMPARE	;是, 比较中断处理
	BTFSC	PIR1, TMR1IF	;是 T1 中断?
	GOTO	T1OVFL	;是, T1 溢出处理
OTHER_INT			;其他中断
	NOP		
	GOTO	OTHER_INT	;出错
COMPARE	BCF	PIR2, CCP2IF	;清 CCP2 中断标志
	MOVFW	RB	;从 RB 口读取比较间隔
	ADDWF	CCPR2L	;更新比较寄存器值
	BTFSC	STATUS, C	
	INCF	CCPR2H	
	MOVLW	1	;设置比较相等时,
	XORWF	CCP2CON	;CCP2 取反
	RETFIE		;中断返回
CAPTVRE	BCF	PIR1, CCP1IF	;清 CCP1 中断标志
	MOVFW	CCPR1L	;读取捕捉中断时 TMR1 值
	MOVWF	CAPT_NEW_L	
	MOVFW	CCPR1H	
	MOVWF	CAPT_NEW_H	
	MOVFW	CAPT_OLD_L	
	SUBWF	CAPT_NEW_L	
	BTFSS	STATUS, C	
	DECF	CAPT_NEW_H	
	MOVFW	CAPT_OLD_H	;计算最近两次捕捉的间隔,
	SUBWF	CAPT_NEW_H	;此间隔应该是 RB 口的两倍
LOAD_OLD	MOVFW	CCPR1L	
	MOVWF	CAPT_OLD_L	;存贮捕捉中断时
	MOVFW	CAPT_NEW_H	;TMR1 值以备用
	MOVWF	CAPT_OLD_H	
END_CAPTURE	RETFIE		
T1OVFL	BCF	PIR1, TMR1IF	;清 T1 溢出中断标志
	RETFIE		
START	BCF	STATUS, RP0	
	CLRF	TMR1H	
	CLRF	TMR1L	
	CLRF	INTCON	
	CLRF	PIR1	
	BSF	STATUS, RP0	
	CLRF	PIE1	
	CLRF	PIE2	;关所有外部中断允许
	MOVLW	0FFH	
	MOVWF	TRISB	;RB 口为输入口
	CLRF	TRISC	

	BSF	TRISC, 2	;CCP1 为输入
	BSF	PIE1, CCP1IE	;开 CCP1 中断允许
	BSF	PIE2, CCP2IE	;开 CCP2 中断允许
	BCF	STATUS, RP0	
	CLRF	PIR1	
	CLRF	PIR2	
	CLRF	T1CON	;选择 T1 模式
	BSF	INTCON, PEIE	;开外部中断允许
	BSF	INTCON, GIE	;开全体中断允许
	MOVFW	RB	
	ADDWF	CCPR2L	
	BTFSC	STATUS, C	
	INCF	CCPR2H	
	MOVLW	08H	;选择 CCP2 为比较输出
	MOVWF	CCP2CON	
	MOVLW	05H	;选择 CCP1 为上升沿捕捉输入
	MOVWF	CCP1CON	
	BSF	T1CON, TMR1ON	;开 T1 定时器
LZZ	NOP		
	GOTO	LZZ	;等待中断
	END		

十一、TMR1 异步时钟方式下的应用

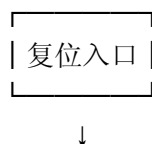
（一）电路设计

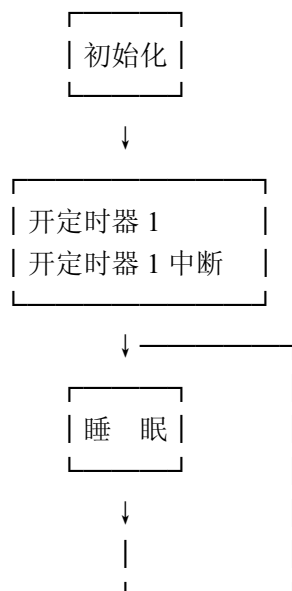
PIC16CXX 的定时器 TMR1，可接成异步时钟工作方式。在这种方式下，TMR1 有自己的振荡电路，即使在睡眠状态下，TMR1 也将继续工作。当 TMR1 溢出时将会产生中断，这种特性极适合于设计省电的实时时钟电路。本例 TMR1 时钟来自外接 32.768KHZ 晶振，TMR1 每 1S 产生一次中断，计数器加 1 后送出 RB 口，而平时 CPU 处于睡眠状态。电路图如下：



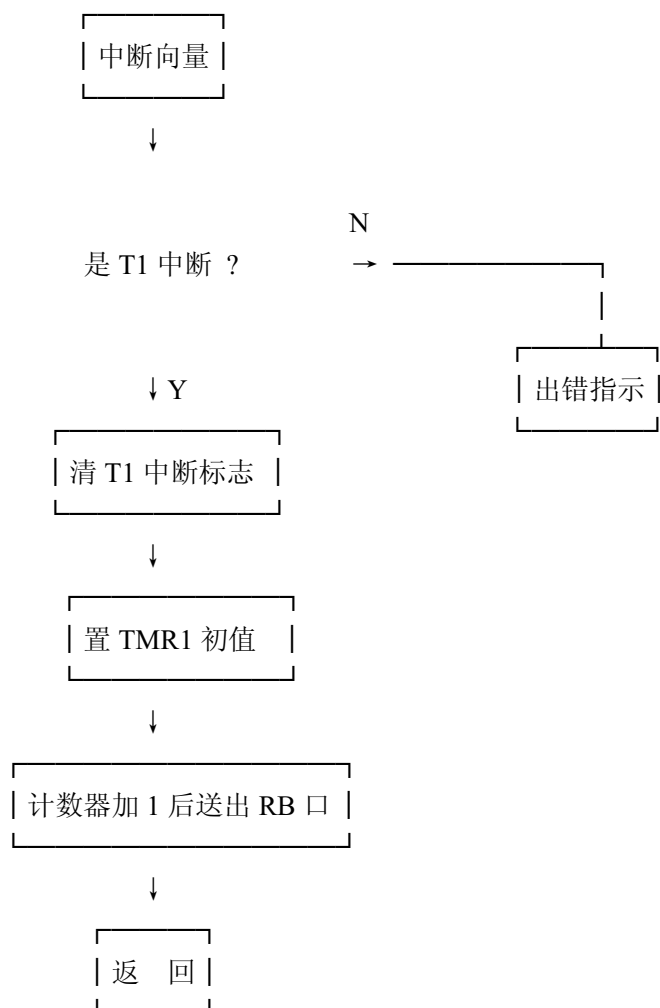
（二）程序流程图

1、主程序流程图





2、中断服务程序



（三）程序清单

```

COUNT    EQU    20H                ;计数寄存器
include    '16CXX.EQU'

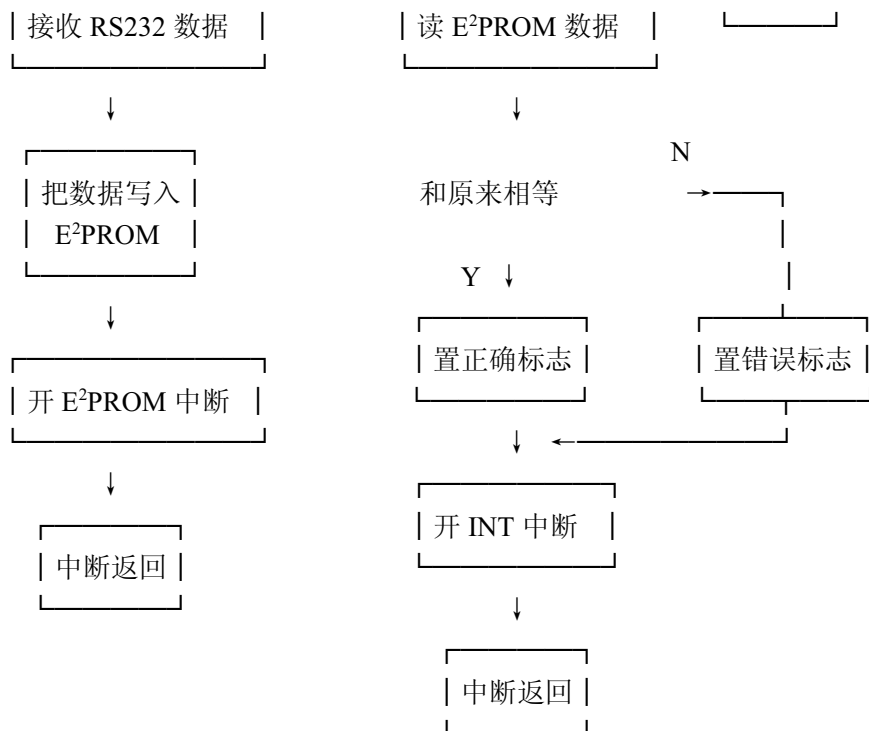
            ORG    0H
            GOTO    START
            ORG    4H
  
```

PER_INT_V			;中断向量
	BCF	STATUS, RP0	
	BTFSC	PIR1, TMR1IF	;是 T1 中断?
	GOTO	T1_OVRFL	;是, 转到中断服务
ERROR1:			;否则, 出错
	MOVLW	0FFH	
	MOVWF	RB	;RB 口全送 1
	GOTO	ERROR1	
T1_OVRFL			
	BCF	PIR1, TMR1IF	;清中断标志
	MOVLW	080H	
	MOVWF	TMR1H	;置 TMR1H 初值
	INCF	COUNT	;计数器加 1
	MOVFW	COUNT	
	MOVWF	RB	;送出 RB 口
	RETFIE		;中断返回
START	CLRF	STATUS	;选寄存器体 0
	BCF	T1CON, TMR1ON	;关 T1 定时器
	CLRF	COUNT	;清计数器
	CLRF	RB	
	MOVLW	80H	
	MOVWF	TMR1H	;置 TMR1 初值
	CLRF	TMR1L	
	CLRF	INTCON	
	CLRF	PIR1	
	BSF	STATUS, RP0	
	CLRF	TRISB	;RB 口置输出
	CLRF	PIE1	;关闭所有中断
	BSF	TRISC, T1OSO	;T1OSO 置成输入
	BSF	PIE1, TMR1IE	;开 TMER1 中断
	BCF	STATUS, RP0	
	CLRF	PIR1	
	BSF	INTCON, PEIE	;开外部中断允许
	BSF	INTCON, GIE	;开中断允许
	MOVLW	0EH	;选择 T1 振荡为外部振荡
	MOVWF	T1CON	;异步方式, 分频比为 1
	BSF	TMR10H	;开 T1 定时器
LOOP	SLEEP		;睡眠
	GOTO	LOOP	;直至 T1 溢出唤醒

十二、PIC16C84 内部数据 E2PROM 使用例程

(一) 电路设计

PIC16C84 内部有 64*8 E²PROM 数据存贮器, 可以由用户用程序方便地读写, 而外部不能对它读写, 所以不仅能掉电保持数据, 同时也具有很高的数据保密性。本例从 RS232 (波特率为 9600B/S) 的 TX 端接收到数据, 然后写入 E²PROM 某一数据单元。电路图如下:



(三) 程序清单

```

include "16cxx.equ"
EEDATA      EQU      8H          ;E²PROM 数据寄存器
EEADR       EQU      9H          ;E²PROM 地址寄存器
EECON1      EQU      88H         ;E²PROM 控制寄存器 1
EECON2      EQU      89H         ;E²PROM 控制寄存器 2
;E²PROM 控制寄存器 1 EECON1
RD          EQU      0           ;读控制位
WR          EQU      1           ;写控制位
WREN        EQU      2           ;写使能位
WRERR       EQU      3           ;写操作错误标志位
EEIF        EQU      4           ;写完成中断标志位

EEIE        EQU      6           ;INTCON<6>位为 E²PROM 中断允许位

DATARAM     EQU      20H         ;数据暂存寄存器
COUNT1     EQU      21H         ;
COUNT2     EQU      22H         ;
DATA        EQU      23H
ADDR        EQU      0001H       ;E²PROM 地址
;

                ORG      0H
                GOTO     MAIN
                ORG      4

INT_SERVICE   ;中断服务
                BTFSC    INTCON, INTF      ;是 INT 中断?
                GOTO     INT_RS232        ;是，转到接收 RS232 数据
                BSF      STATUS, RP0
                BTFSC    EECON1, EEIF      ;是 E²PROM 中断?
                GOTO     INT_EEPROM        ;是，转到 E²PROM 中断处理
  
```

ERROR_INT	BCF	STATUS, RP0	
	RETFIE		
INT_EEPROM			;E ² PROM 中断处理
	BSF	STATUS, RP0	
	BCF	EECON1, WREN	;禁止 E ² PROM 写
	BCF	STATUS, RP0	
	MOVLW	ADDR	;把地址送给
	MOVWF	EEADR	;EEADR
	BSF	STATUS, RP0	
	BSF	EECON1, RD	;读 E ² PROM
	BCF	STATUS, RP0	
	MOVFW	DATARAM	
	XORWF	EEDATA, 0	;和原数据比较
	BTFSS	STATUS, Z	;相等?
	GOTO	ERROR	;否, 出错
	BSF	RA, 0	;置正确标志
	BCF	R, 1	
RET_INT	BCF	INTCON, EEIE	;清 E ² PROM 中断允许
	BSF	INTCON, INTE	;开 INT 中断允许
	BSF	STATUS, RP0	
	BCF	EECON1, EEIF	;清 E ² PROM 中断标志
	BCF	STATUS, RP0	
	RETFIE		
ERROR	BCF	RA, 0	;置错误标志
	BSF	RA, 1	
	GOTO	RET_INT	
INT_RS232			
	BCF	INTCON, INTE	;清 INT 中断允许
	BCF	INTCON, INTF	;清 INT 中断标志
	MOVLW	8	
	MOVWF	COUNT1	;总共收 8Bit 数据
	CALL	DELAY_1/2B	;延时 1/2BYTE
LOOP			
	CALL	DELAY_1B	;延时 1BYTE
	SETC		
	BTFSS	RB, 0	;读 RS232 数据
	CLRC		
	RRF	DATA	;移位
	DECFSZ	COUNT1	;接收完 8Bit?
	GOTO	LOOP	;否, 继续
	BSF	STATUS, RP0	
	BSF	EECON1, WREN	;E ² PROM 写允许
	BCF	STATUS, RP0	
	MOVLW	ADDR	;地址送给 EEADR
	MOVWF	EEADR	
	MOVFW	DATA	;数据送给 EEDATA
	MOVWF	EEDATA	
	BSF	STATUS, RP0	
	MOVLW	55H	;┐

	MOVWF	EECON2	;
	MOVLW	0AAH	; 启动写操作
	MOVWF	EECON2	;
	BSF	EECON1, WR	; └─┘
	BCF	STATUS, RP0	
	BSF	INTCON, EEIE	;开 E ² PROM 中断允许
	RETFIE		
DELAY_1/2B			;延时 1/2BYTE
	MOVLW	.43	
	GOTO	SKIP1	
DELAY_1B			
	MOVLW	.86	;延时 1BYTE
SKIP1	MOVWF	COUNT2	
LOOPD	DECFSZ	COUNT2	
	GOTO	LOOPD	
	RETLW	0	
;主程序			
MAIN			
	BCF	STATUS, RP0	
	CLRF	RA	
	CLRF	RB	
	CLRF	INTCON	
	BSF	STATUS, RP0	
	CLRF	TRISA	;RA 口为输出口
	MOVLW	1	
	MOVWF	TRISB	;RB0 为输入口
	CLRF	EECON1	
	BCF	STATUS, RP0	
	BSF	INTCON, GIE	
	BSF	INTCON, INTE	;开 INT 中断
LZZ	NOP		;等待中断
	GOTO	LZZ	
	END		

宏汇编器 MPASM

MPASM 是 Microchip 公司推出的可适用于其 PIC16/17 全部单片机的宏汇编器，功能齐全，全屏幕操作。

§ 10.1 启动和操作

MPASM 的启动很简单，在 DOS 状态下：

> MPASM <Enter> （注意 MPASM 后面不要跟文件名）

屏幕上即显示：

MPASM 01.11 Released (c)1993, 94 Byte Craft Limited/Microchip Technology Inc.

Source File : SAMPLE.ASM

Processor Type : 12C509

Error File : Yes

Cross Refernece File : No

Listing File : Yes

Hex Dump Type : INHX8M .HEX

Assemble to Object File : No

↑ ↓ , Tab : Move Cursor	Esc : Quit	Press Enter to change value.
F1 : Help	F10 : Assemble	

图 4.1 MPASM 画面

Source File:	源程序文件名。可以带路径和通配符(*)。
Processor Type:	芯片型号。可通过 Enter 键来选择用户所需的型号。
Error File:	汇编后自动产生一个.ERR 文件，该文件记录了汇编中产生的错误语句和警告信息。
Cross Reference File:	产生一个参考文件.XRF。
Listing File:	产生一个列表文件.LST。该文件中包含了各种仿真环境中需要的参数，主要用于仿真调试。
Hex Dump Type:	产生的代码烧写文件，一般选择 INHX8M 格式，可适应众多的烧写器。
Assemble to Object File:	注意这里产生的.OBJ 文件不是通常认为的机器代码文件，而是预留给链接器(Linker)的可重定位文件。选择 NO 则汇编不产生任何.OBJ 文件。

§ 10.2 汇编语言格式

PIC 汇编语句的格式为：

〔标号〕〈指令助记符〉〔操作数〕 ；〔注释〕

指令助记符与标号间至少应有一个空格。若一行语句没有标号，则指令助记符前必须至少有一个空格，否则会当成是标号。一条语句最多字符个数为 255。

```

;
; Sample MPASM Source Code. It is for illustration only.
;

List      p=12C509, r=HEX

org      0h          ; 程序从 0h 处开始放
start
movlw    0x0a
movlw    0x0b        ;
goto     start       ; loop

end
```

图 4.2 汇编语言范例

一、标号

标号须由第一格起始写，最多可达 31 个字符，且第一个字符必须是字母。标号后可跟冒号 (:)、空格或行结束符。除非使用选择项/C，否则标号中的字母大小写是不一样的，如：

```

START
start
```

是二个不同的标号。

二、指令助记符

指 PIC 的指令或伪指令，宏定义符等。具体参阅有关各章节和资料。

三、操作数

操作数可以是常数，符号或表达式。两个操作数之间必须由逗号(,)分开。

(1) 符号——各种定义的符号、宏定义等。

例：MOVWF F10 ;F10 为操作数，是定义的代表寄存器 10 的符号。

(2) 常数——在 MPASM 中，常数可以是如下：

进制	书写格式	例子
十进制	D'<数字>'	D'255'
十六进制	H'<16 进制数字>' 或 0x<16 进制数字>	H'A8' 0xA8
二进制	B'<二进制数字>'	B'00111001'
八进制	O'<八进制数字>'	O'777'
字符 ASCII 码	'< 字 符 >'	'C'

注：MPASM 默认进制为 16 进制。

表 10.1

(3) 表达式——由常数、符号和各种算术运算符号按一定顺序组成。

MPASM 中的算术符号如表 4.2 所示。

运 算 符		例 子
(左括号	1+ (d*4)
)	右括号	同上
!	非	IF !(a-b)
+	加	a+b
-	减	a-b
*	乘	a*b
/	除	a/b

%	取模	a%2
<<	左移	<<a
>>	右移	>>a
>	大于	IF a>b
<	小于	IF a<b
<=	小于或等于	IF a<=b
==	等于	IF a==b
!=	不等于	IF a!=b
&	与	a & b
^	异或	a ^ b
	或	a b
~	取反	
& &	逻辑与	IF (a=2) && (b=3)
	逻辑或	IF (a=2) (b=3)
=	等于	a=b
+=	加，然后等于	a+=1
-=	减，然后等于	a-=1
=	乘，然后等于	a=5
/=	除，然后等于	a/=5
<<=	左移，然后等于	a<<=5
>>=	右移，然后等于	a>>=5
&=	与，然后等于	a&=5
=	或，然后等于	a =5
^=	异或，然后等于	a^=5
¥	返回当前 PC 值	GOTO ¥+3

表 10.2

四、注释

以分号 (;) 起始，用户可以注释程序。

CLRF F10 ; 清 F10 寄存器

§ 10.3 伪指令

所谓伪指令，是一些用来控制汇编器的命令。它们可放在源程序 (.ASM) 中，但不是被翻译成可执行的机器代码，而是用来控制汇编器的输入/输出以及数据的定位等。

在 MPASM 中，有四类伪指令：

- 1、数据伪指令：用于控制程序存储器的定位，定义数据的名称等。
- 2、列表伪指令：用于控制 MPASM 产生的列表文件 (.LST) 的格式等。
- 3、控制伪指令：用于控制汇编的路径，如条件汇编等。
- 4、宏汇编指令：用于控制宏定义体中的运行和数据定位。

一、数据伪指令

1. DATA——定义程序存储器的值。

格式：{<标号>} DATA <操作数>, {<操作数>...}

例：DATA 1, 2+AB, "Test"

2. DEFINE——定义字符串变量。

格式：#DEFINE<变量名> (<字符串>)

例：#DEFINE Length 20

#DEFINE control 0x19, 7

```
#DEFINE position (X, Y, Z) (Y-(2×Z+X))
...
...
test_Lable DATA position(1, Length, 52)
                bsf control ; 置 0X19 寄存器的 bit7
```

3. SET——对标号赋值。

格式： <标号> SET <表达式>

例： width SET 9
area SET 0x16
width SET area+8

用 SET 可对标号任意重新赋值，见上例 3。这和下面的另一条标号赋不同。

4. EQU——对标号赋值。

格式： <标号> EQU <表达式>

例： lable EQU 0x16

标号一旦由 EQU 赋值后，其值便不能再重新定义，参考上面 SET 命令。

5. RES——保留某段程序存储区。

格式： RES <单元个数>

例： RES 10

保留 10 个空白字节。

6. INCLUDE——调入外部文件，通常是定义文件，对一些标号和变量进行定义。

MPASM 提供一个名为 PICREG.EQU 的定义文件，读文件中定义了 PIC 寄存器的地址，复位向量及状态位址。

格式： INCLUDE “文件名”

例： INCLUDE “picreg.egu”

7. Radix——进制定义指令。

格式： RADIX <进制表达式>

例： RADIX dec ; 十进制
RADIX Hex ; 十六进制
RADIX oct ; 八进制

二、列表伪指令

1. LIST——列表选择指令，设置各种汇编参数。

格式： LIST〔<选择项>…<选择项>〕

例： LIST F=INHX8M, R=DEC, P=16C84

以下是 LIST 选项表：

选 项	默 认 值	作 用
C=nnn	80	行宽
N=nnn	59	每页的行数
T=ON/OFF	OFF	ON 截去超长行的超出部分
P=<type>	无	PIC12C/16C/17C
R=<radix>	hex	常数进制选择:hex, dec, oct
F=<format>	INHX8M	烧写文件格式: INHX16, INHX32 和 INHX8M

2. PAGE——分页命令。

格式: PAGE

在列表文件中（.lst）中产生分页效果，即下面的文件输出将从新页面开始。

3. TITLE——程序标头命令。

格式: TITLE '程序标头'

例: TITLE 'This is for PIC12C50X demo'

标头最长不超过 60 个字符。TITLE 令会造成分页，即标头总是在一页的第一行上。

三、控制伪指令

1. ORG——定义程序存放起始地址。

格式: <标号> ORG <地址表达式>

例: ORG 0h ; 起始程序存放地址

START: MOVWF OSCCAL

...

...

若 ORG 不带地址参数，则默认为 0。若 ORG 带标号，则地址参数也赋值给该标号。

2. END——程序结束命令。

格式: END

例: END

这条指令告诉 MPASM 这是源程序（.ASM）的结束行，后面若还有语句将被视为无效。

3. IF——条件汇编命令。

格式: IF <条件表达式>

<源程序行>

<ELSE>

<源程序行>

ENDIF

例: IF VER==100

MOVLW 5

WOVWF F11

ELSE

MOVLW 6

MOVWF F11

ENDIF

...

...

如果条件表达式为真，MPASM 将汇编 IF 和 ELSE 之间的语句，反之汇编 ELSE 和 ENDIF 之间的语句。ELSE 可以缺省，这样条件为真则汇编，反之不汇编。

4. WHILE——条件循环命令。

格式: WHILE <条件表达式>

...

...

ENSW

例: VARIABLE i

```

        WHILE      i<count
        MOVLW      i
        i=i+1
        ENDW

```

注：VARIABLE 也是一条定义变量的伪指令，和 EQU 及 SET 不同的是它不要求变量在定义时必须赋值给初值，如上例中的变量 i。关于这条伪指令不再赘述。

四、宏定义伪指令

1. MACRO——宏定义命令。宏是一段指令，可以插在源程序中。宏必须事先定义好，宏之间可以互相调用，也可以自己递归调用。宏本身不会产生代码，只是在调用它时把宏体插入源程序，这点和子程序调用有本质不同，即宏并不会节省程序空间，它主要的好处是令程序书写简洁明了。

格式： <标号> MACRO [<参数 1>...<参数 N>]
 (宏体)
 ENDM

例：GET MACRO X, Y, Z
 MOVWF X
 Y
 Z MOVLW 10
 GOTO Z
 ENDM

宏调用可以下为：

```

...
GET    F0, (INCF F17, W), ENTRY
...

```

则汇编后这句宏调用产生的源代码为：

```

      GET    F0, (INF17, W), ENTRY
+     MOVWF  F0
+     INCF   F17, W
+ ENTRY    MOVLW 10
+     GOTO   ENTRY

```

前面带+号表示是宏体中定义的程序。

§ 10.4 错误/警告信息

MPASM 汇编一个源程序后，可以产生一个.ERR 文件，该文件用来存放汇编后可能产生的错误或警告信息。必须强调的是错误信息（Error）是指出源程序中出现“致命”（fatal）的错误，用户必须修改直至汇编后 Errors= 0。而警告信息（Warnings）是指出源程序中可能有问题的地方，但并不一定是“致命”错误，只是提醒用户去注意这些被警告的地方。如果用户可以确认无误，便可以不理会产生产生的 Warnings。

一、错误信息

1.Address exceeds maximum limit avaiable

程序存储器地址溢出（超出）有效范围。

2.Attempt to redefine reserved word

MPASM 中的保留字如“END”、“ERROR”、“HIGH”、“LOW”和“PAGE”被重定义，用户必须避免再将其用做标号或变量。

3.Branch or jump out of range

程序跳转指令如“GOTO”、“CALL”等超出规定的范围。

4.Couldn't open...

TMPASM 不能打开 “.OBJ”、“.map”、“.Hex”、“.Err”、“.Lst” 或 “.ref” 文件。一般是电脑已没有足够的磁盘空间。

5.Couldn't open source file...

汇编的源程序文件不存在。

6.Duplicate lable or redefininy symbol that cannot be redefined

标号或变量名重复定义。

7.Error in parameter

参数错误。

8.Expected...

源程序行有错。

9.File not found

指定的文件找不到。

10.Illegal argument

非法参数。

11.Illegal condition

IF 语句中的条件符号出错。

12.Illegal condition, EOF encountered before END or conditional end directive

IF、WHILE 或 MACRO 语句中缺少相应的 ENDIF、ENDW 和 ENDM。

13.Illegal conditional compile

IF/ELSE/ENDIF 结构书写有错。

14.Illegal character...in label...

在标号字符中出现非法字符。合法的字符是 “-”、“.”、“A” ~ “Z”、“a” ~ “z”、“0” ~ “9”。

15.Illegal digit

非法数字。如在十进制数中出现十六进制符等。

16.Illegal opcode

非法操作数。

17.Include file not found

Include 指令中的文件找不到。

18.Include files nested too cleep

Include 文件嵌套太多。Include 文件嵌套最多的为 5 重。

19.Macro name missing

缺少宏定义名称。

20.Marco nested too deep

宏体嵌套太多。宏体中最多可嵌套 8 重。

21.Missing arguments

缺少参数，如指令中缺少操作数等。

22.Missing terminator

缺少配对符，如各种括号 “)”、“}” 或 “.”、空格等。

23.Nested forward reference not allowed.

使用未定义的标号、变量、宏定义等。或者是 MPASM 不能确认标号的类型，一般由标号重复定义引起。

24.Out of memory

程序空间溢出。

25.Overwriting previous address contents

程序空间重复使用。一般由地址定义指令 ORG 定义不当引起。

26.Processor type is undefined

单片机芯片型号未定义。要么在源程序中未定义，要么在使用 MPASM 汇编时未定义。

27.Processor type previously defined

单片机芯片型号重复定义。

28.Symbol table full

符号表溢出。

29.Temp file creation error

MPASM 在汇编过程中会使用一些临时性文件。这个错一般是电脑磁盘满或读写出错引起。

30.Too many arguments

参数太多，如指令操作符带过多的操作数。

31.Undefined argument

使用了未定义的参数。

32.Unknow error

MPASM 碰到未知的错，这种情况一般很少发生。

33.While failed to terminate within 256 iterationg

While 语句中没有结束的条件产生。

二、警告信息

1.Addresses above 32K not currently supported. Using MaxRom.

MPASM 目前只允许源程序使用 8000H（32K）以下的程序地址。将来可以增加到 64K。

2.Argument out of rauge, least significant bits used.

参数超出所允许的范围值。MPASM 一般会把超出的值自动截为认可的最大值。

3.Crossing page boundary--ensure page blts are set

MPASM 通知你程序跨页面了，建议你确认相应的页面位是否已经设置了。

4.···Is not currently supported

使用了 MPASM 尚未支持的指令。

5.···Not a single byte quantity

使用了超出 8 位的数值。

6.This number is being treated as a binary representation

MPASM 碰到不知是二进制或十六进制的值，如 b 0101。这时 MPASM 都会把它当做二进制处理。上例如果要表示 16 进制，应写成 ox b0101 或 H'b0101'。

§ 10.5 使用 MPASM 来汇编 PIC12C5XX 的问题

如果用户手中的 MPASM 版本是 97 年前的老版本，则在 Processor Type 中寻不到 PIC12C508/509，怎么办?不要紧，因为 PIC12C5XX 和 PIC16C5X 的指令是完全一样的。所以如果手中的 MPASM 是老版本，可以用如下方法来操作：

汇编对象	Processor Type
PIC12C508	16C54
PIC12C509	16C58