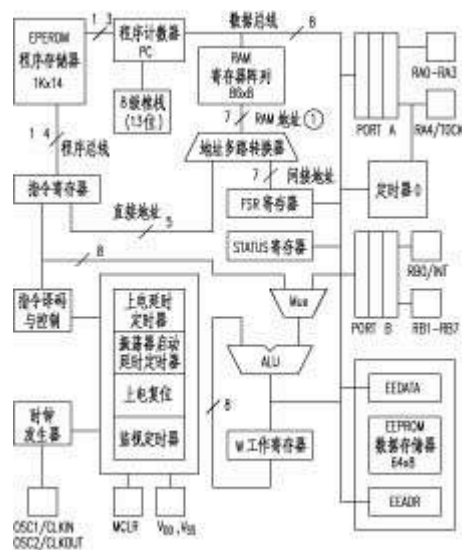


## PIC16F84 单片机的内部硬件资源

学些 PIC 单片机，在 Microchip 尚未推出其他 Flash 系列的情况下，很多菜鸟都是从 PIC16F84 开始的，我们把它整理了一份中文资料供大家学习。

首先介绍 PIC16F84 单片机的内部结构，如图 1 所示的框图。由图 1 看出，其基本组成可分为四个主要部分，即运算器 ALU 和工作寄存器 W；程序存储器；数据存储器 and 输入/输出(I/O)口；堆栈存储器和定时器等。现分别介绍如下。



### 1 运算器 ALU 及工作寄存器 W

运算器 ALU 是一个通用算术、逻辑运算单元，用它可以对工作寄存器 W 和任何通用寄存器中的两个数进行算术(如加、减、乘、除等)和逻辑运算(如与、或、异或等)。16F84 是八位单片机，ALU 的字长是八位。在有两个操作数的指令中，典型的情况是一个操作数在工作寄存器 W 中，而另一个操作数是在通用寄存器中，或者是一个立即数。在只有一个操作数的情况下，该数要么是在工作寄存器 W 中，要么是在通用寄存器中。W 寄存器是一个专用于 ALU 操作的寄存器，它是不可寻址的。

根据所执行的指令，ALU 还可能会影响框图中状态寄存器 STATUS 的进位标志 C、全零标志 Z 等。

### 2 程序存储器

单片机内存放程序指令的存储器称为程序存储器。PIC16F84 的所有指令字长为 14 位。所以程序存储器的各存储单元是 14 位宽。一个存储单元存放一条指令。16F84 的程序存储器有 1024(28)个存储单元(存储容量为 1k)。这些程序存储器都是由 EPEROM 构成的。

程序存储器是由程序计数器 PC 寻址的。16F84 的程序计数器为 13 位宽，可寻址 8K(8×1024)的程序存储器空间，但 16F84 实际上只使用了 1k 的空间(单元地址为 0~3FFH)。当访问超过这些地址空间的存储单元时，将导致循环回到有效的存储空间。

对于用过其它单片机的用户，可能会感到 16F84 的片内存储器容量太少了。实际上并非如此，因为 16F84 的指令系统都是由单字节指令构成的，相应于其它由二字节、三字节甚至四字节指令的单片机而言，PIC 单片机的程序存储器有效容量要比标称值扩大 2~5 倍到 3 倍。

### 3 数据存储器

在单片机 PIC16F84 中,除了存放程序的程序存储器外,还有数据存储器。单片机在执行程序过程中,往往需要随时向单片机输入一些数据,而且有些数据还可能随时改变。在这种情况下就需用数据存储器。由于数据存储器不但要能随时读取存放在其各个单元内的数据,而且还需随时写进新的数据,或改写原来的数据。因此,数据存储器需由随机存储器 RAM 构成。RAM 存储器在断电时,所存数据随即丢失,这在实际应用中有时会带来不便。但是,在 16F84 中有  $64 \times 8$  位 E2PROM 数据存储器。存放在 E2PROM 中的数据在断电时不会丢失。

16F84 中的 RAM 数据存储器如表 1 所示,该 RAM 分为两个存储体:即存储体 0(Bank0)和存储体 1(Bank1)。每个存储体均可以直接用内部总线传送信息,所以它们都是以寄存器方式工作和寻址。这些八位寄存器,又可分为通用寄存器和专用寄存器两个部分。通用寄存器存放数据,专用寄存器存放控制单片机运作的信息。每个存储体最大可扩展到 7FH(128 个字节)。在每个存储体中,专用寄存器被安排在低位地址空间,通用寄存器被安排在高位地址空间。

通用寄存器用法单一,但专用寄存器却各有各的用处,现将较基本的专用寄存器作一简单介绍。

(1)程序计数器(PCL、PCLATH)。程序计数器 PC 是对程序进行管理的计数器。PIC16F84 的程序计数器为 13 位宽,最大可寻址的存储空间为  $8k \times 14$  位。实际上 16F84 只使用前  $1k \times 14$  位(0000~03FFH)存储空间。因程序计数器有 13 位宽,而专用寄存器只有 8 位。因此 PC 由两个专用寄存器构成。其低八位 PCL 是一个可读/写寄存器(地址为 02H 或 82H),而高字节 PCH(有效位 5 位)不能直接进行读/写操作,它是通过一个 8 位的保持寄存器 PCLATH(地址为 0A 或 8AH)把高 5 位地址传送给程序计数器的高字节。当执行 CALL、GOTO 指令,或写 PCL 时,PC 值的高字节就从 PCLATH 寄存器中装入。

(2)状态寄存器 STATUS。状态寄存器 STATUS 含有算术逻辑单元 ALU 运算结果的状态(如有无进位等)、复位状态及数据存储体选择位。有关位位的设定如表 2 所示,功能如下:

1)第 0 位。进位/借位位 C。执行加、减运算指令

表 2

IRP RP1 RP0 TO PD Z DC C

后,若结果有进位或借位,则 C 被置 1,否则置 0。在执行移位指令时,也要用到这一位。

2)第 1 位。辅助进位/借位位 DC。执行加、减运算指令后,若结果的低四位向高四位有进位或借位,则 DC 置 1,否则置 0。

3)第 2 位。零标志位运算结果为零,Z 被置 1;运算结果不为零,Z 被清零。

4)第 3 位。低功耗标志位 PD。上电复位或执行 CLRWDT 指令后置 1,执行 SLEEP 指令后被清零。

5)第 4 位。定时时间到标志位 TO。上电复位或执行 CLRWDT、SLEEP 指令后被置 1,监视定时器的定时时间到被清零。

6)第 5 位和第 6 位(RP0、RP1)。这两位是用于直接寻址时的寄存器体选择位。即 00——选中 Bank0(00H~7FH);01——选中 Bank1(80H~FFH),16F84 只有两个存储体。故 10、11 不用。

7)第 7 位 IRP。这是间接寻址的寄存器体选择位。0——选中 Bank0、1(00H~FFH),1——选中 Bank2、3。16F84 只有 Bank0、1,所以此 IRP 位应被置为 0。

(3)间接寻址 INDF 和 FSR 寄存器

INDF 寄存器不是一个物理寄存器,而是一个逻辑功能的寄存器(地址为 00H 或 80H),当对 INDF 寄存器进行寻址时,实际上是访问 FSR 寄存器内容所指的单元,即把 FSR 寄存器作为间接寄存器使用。FSR 称为“寄存器选择”寄存器,地址为(04H 或 84H)。对 INDF 寄存器本身进行间接寻址访问,将读出 FSR 寄存器的内容,例如当 FSR=00H 时,间接寻址读出 INDF 的数据将为 00H。用间接寻址方

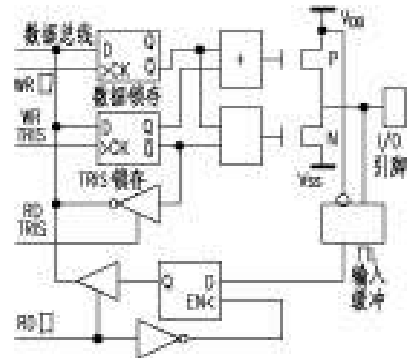
式写入 INDF 寄存器时, 虽然写入操作可能会影响 STATUS 中的状态字, 但写入的数据是无效的。

#### 4 I/O 口

单片机作为一个控制器件必定有数据输入和输出。输入量可能是温度、压力、转速等, 而输出量可能是开关量和数据, 以保证受控过程在规定的范围内运行。数据的输入和输出都需通过单片机内部有关电路, 再与引脚构成输入/输出 (I/O) 端口。PIC16F84 芯片有两个 I/O 端口 (PROTA 和 PORTB)。端口 A 为 5 位口, 端口 B 为 8 位口, 共占用 13 位引脚。每个端口由一个锁存器 (即数据存储器中的特殊功能寄存器 05H、06H 单元)、一个输出驱动器和输入缓冲器等组成。当把 I/O 口作输出时, 数据可以锁存; 作输入时, 数据可以缓冲。

16F84 PORTA 口中的 RA4 是斯密特触发输入、漏极开路输出。而其它的 RA 口引脚都是 TTL 电平输入和全 CMOS 驱动输出。端口 PORTB 是一个八位双向可编程 I/O 口。各端口虽然也由锁存器、驱动器、缓冲器等构成, 但因功能略有不同而导致电路亦存在差别。现以 PORTA 口的 RA0 ~RA3 的电路 (见左图) 为例, 说明其基本工作原理。

图中 RA 口的 I/O 引脚是由数据方向位 (寄存器 TRISA) 来定义数据流向。当 TRISA 寄存器的位置为 “1” 时, 其输出驱动器 (由 P 沟道和 N 沟道 MOS 管串接而成) 呈高阻态, 即两个 MOS 管均截止, I/O 口被定义为输入。此时, 数据由 I/O 端输入, 经 TTL 输入缓冲器到 D 触发器。当执行读指令时, 此 D 触发器使能, 数据经三态门进入数据总线。



当 TRISA 的位置为 “0” 时, I/O 口被定义为输出, 此时输出锁存器的输出电平就是 I/O 口的输出电平。

读 PORTA 寄存器的结果就是读取 I/O 引脚上的电平, 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 I/O 引脚电平, 然后由程序修改 (按要求给定一个值), 再置入 I/O 锁存器。

PIC16F84 的输出可提供 20mA 的电流, 所以它可直接驱动 LED。PORTA 和 PORTB 各个位均可分别定义为输入和输出。下面以 PORTA 口初始化程序的实例, 说明选择 I/O 口的方法。

CLRF PORTA; 端口 A 被清零

BSF STATUS; 状态寄存器 STATUS 的 RPO 位置为 1, 选 BANK1。

MOVLW 0xCF ; 将定向值

                  ; 11001111 置入 W 工作寄存器

MOVWF TRISA; 置 RA (3~0) 位为输入

                  ; RA 5~4 位为输出

                  ; TRISA 7~6 位未用

在使用 I/O 口时应注意:

(1) 当需要一个 I/O 口一会做输入、一会又做输出时, 输出值会不确定。

(2) I/O 引脚输出驱动电路为 CMOS 互补推挽输出。当其为输出状态时, 不能与其它输出脚接成 “线或” 或 “线与”, 否则, 会因电流过载烧坏单片机。

(3) 当对 I/O 口进行写操作后不宜直接进行读操作, 一般要求在两条连续的写、读指令间至少加入一条 NOP 指令。

例: MOVWF 6 ; 写 I/O

      NOP ; 稳定 I/O 电平

      MOVF 6, W; 读 I/O

#### 5 堆栈

单片机执行程序时，常常要执行调用子程序。这样就产生了一个问题：如何记忆是从何处调用的子程序，以便执行子程序之后正确返回。此外，在程序执行过程中，还可能会发生中断，转而执行中断子程序，这时，又如何记忆从何处中断，以便返回呢？

满足上述功能的方法就是“堆栈”技术。

“堆栈”是一个用来保存临时数据的栈区。当主程序调用子程序时，单片机执行到 CALL 指令或发生中断时，就自动将下一条指令的地址“压栈”保存到栈区。当子程序结束，单片机执行返回指令时，就自动地把栈区的内容“弹出”，作为下步指令执行的新地址。

PIC16F84 芯片内有一个 8 级 13 位宽(与 PC 同宽)的硬件堆栈，此堆栈既不占用程序存储空间，也不占用数据存储空间。当执行一条 CALL 指令或一个中断被响应后，程序计数器 PC 中的断点地址就自动被压栈(PUSH)保护，而当执行 RETURN、RETLW 或者 RETFIE 指令时，堆栈中的断点地址会弹出(POP)程序计数器 PC 中。无论是 PUSH 还是 POP 操作，都不影响 PCLATH 寄存器的内容。

## 6 定时器/计数器 TMRO

PIC 单片机 16F84 中有一个定时器，此定时器也可用于计数，因此称为定时器/计数器，符号为 TMRO。TMRO 可用于定时控制、延时、对外部事件计数和检测等场合。TMRO 是一个 8 位增量(加 1)计数器。它在数据存储寄存器中的地址为 01。定时器所用的时钟源可以是内部系统时钟(OSC/4，即四倍振荡周期)，也可以是外部时钟。若 TMRO 对内部系统时钟的标准脉冲系列进行计数时，就成为定时器；对外部脉冲进行计数时 TMRO 就成为计数器。

不管是定时还是计数方式，TMRO 在对内部时钟或对外部事件计数时，都不占用 CPU 时间，除非 TMRO 溢出，才可能中断 CPU 的当前操作。可见，定时器是单片机 16F84 中效率高且工作灵活的部件。

为了扩大定时或计数的范围，配合 TMRO 的使用，还有一个可编程预定标器。此定标器实际上是一个可编程分频器。

TMRO 的内部结构示意图如附图所示。其工作方式由数据存储寄存器中的项选寄存器 OPTION 控制。OPTION 是一个可读/写的寄存器，如附表所示。它含有配置 TMRO/WDT 预定标器、外部 INT 中断、TMRO 等的各种控制位。

TMRO 的定时、计数方式是由 OPTION 寄存器中的 D5(即 TOCS 位)确定。当 TOCS=0 时，工作于定时器方式；当 TOCS=1 时，工作于计数器方式。作定时器时，每个指令周期加 1(无预分频时)；而作计数器时，则在每个 RA4/TOCKI 引脚上电平变化时加 1。OPTION 寄存器的位 4(TOCS 位)决定外部脉冲的触发方式，当 TOSE=1，下降沿触发；TOSE=0，上升沿触发。当 TMRO 内部计数器发生计数溢出(从 FFh→00h)时，溢出位送入中断控制寄存器 INTCON。

由附图可知，预分频器也是一个 8 位计数器。其分频数是由 OPTION 寄存器中的 PS2~PS0 三位值来改变。分频数可以是以下 8 种之一：1:1、1:2、1:4、1:8、1:16、1:32、1:64 和 1:128。

当分频器用于 TMRO 时，所有写入 TMRO 的指令，如 CLRF 1、MOVWF 1、BSF 1、等都将对预分频器清零。需要注意的是，预分频器是不能读写的。此分频器可用于 TMRO，也可用于 WDT，其切换由软件控制。为了避免意外的芯片复位，当需要切换时，必须执行相应的一段程序，以下是从 WDT 切换到 TMRO 时所需执行的程序：

```
CLRWDI          ;
                ; 对 WDT 和预定标器清零
BSF          STATUS, RP0 ; 选中存储体 1
MOVLW  B' xxxx0xxx'      ; PSA=0, 选中 TMRO
MOVWF  OPTION          ; 送入 OPTION 寄存器
BCF          STATUS, RP0 ; 复位存储体 0
```

## 7 延时和定时

在设计单片机应用系统时，经常会遇到需要使某一过程（如加温、加压等）持续一段时间的情况，如连续加压 1 分钟，通电 2 分钟等。单片机如何正确确定这段时间呢？这里可通过两种方式，即延时和定时来实现。试看下列。

在应用系统中要求 PIC16F84 的 RA0 端控制一个发光二极管按一定频率闪亮，可通过右图的电路来实现。同时还必须为 16F84 编制一个程序。由电路图可知，要使发光二极管 LED 按一定的频率闪亮，只要使 RA0 端输出一个变化的高→低→高……电平即可。由此设计出如下的源程序（清单 1）：

```
list P=16F84, F=INHX8M
; .....
    ORG      0
    MOVLW 0      ; 主程序开始
    TRIS    5      ; 置 RA 口为输出
    BCF     5, 0    ; RA 口 0 位清零
LOOP: CALL   DELAY; 闪动延时
    COMF    5      ; RA 口求反，亮一灭交替
    GOTO   LOOP   ; 循环
; .....
DELAY      ; 以下为延时子程序
    MOVLW  D' 50
    MOVWF  8
LOOP1: MOVWF  9
LOOP2: DECFSZ 9, F
    GOTO   LOOP2
    DECFSZ  8, F
    GOTO   LOOP1
RETLW     0
```

由清单 1 可知，当主程序开始时，首先将工作寄存器 W 清零，然后将 W 寄存器的内容送 TRISA 寄存器，使其清零，以设置 RA 口为输出。接着又将 RA 口的第 5 位清零，使 LED 开始时处于熄灭状态。随之持续一段时间，即执行延时子程序，再将 RA 口取反，变为高电平输出，LED 发光，再延时，又使 RA 口取反，LED 熄灭……。这样，LED 就一暗一亮，持续交替进行。

在这里，使 LED 亮、暗持续一段时间是通过单片机执行延时子程序 DELAY 来实现的。此延时程序的核心就是让单片机的 CPU 反复执行使寄存器内容减 1 的指令 DECFSZ。即将十进制数 50 分别装入通用寄存器 F8、F9，以进行  $50 \times 50 = 2500$  次的减 1 操作。如果执行一次 DECFSZ 指令需 1 个指令周期（跳转时需 2 个周期），若设振荡频率为 100kHz，即指令周期为  $40 \mu s$ ，则延时时间为  $2500 \times 40 = 100000 \mu s = 100ms$ ，即 0.1 秒。实际上还略为大些。此延时时间已超过人眼的视觉保留时间。因而能看清 LED 的明、暗交替变化。

如果我们需要更长的延时时间，可仿照上例，装入更大的数或引入多重循环。因此，在原则上，延时时间可根据需要任意延长。

不过，采用延时程序来持续某一过程的方式有缺陷。延时就是使 CPU 在某几条指令上“转圈”，延时越长，“转圈”数越多，这时，CPU 不能再去执行其它操作，如监视温度、湿度等。这在某些实时控制系统中，不允许这样做。为此，在单片机 16F84 中，专门设置了一个“闹钟”——定时器 TMR0。

需要某过程延续多长时间, 可将其“拨入” TMR0, 到时它会发生“中断”, 告诉 CPU 定时时间到。要 CPU 暂停其它工作, 转过来执行“中断子程序”, 完成输出开、关信号之类的任务后, 再回去执行其中断的工作。这样, 就使 CPU 的工作效率提高。因而, 延时的使用有局限性, 采用定时器 TMR0 则可用于各种场合中。

## 8 中断

PIC 单片机 16F84 具有实时处理功能, 能对外界异常发生的事件由中断技术作及时处理。

当单片机的 CPU 正在处理某事件时, 若外部发生了某一事件(如定时器溢出、引脚上电平变化), 请求 CPU 迅速去处理, 于是 CPU 就暂时中止当前的工作, 转去处理所发生的事件。中断处理完该事件后, 再回到原来被中止的地方, 继续执行原来的工作, 如图 1 所示。实现这种功能的部件称为中断系统。产生中断的请求源称为中断源。中断源向 CPU 提出的处理请求, 称为中断请求或中断申请。CPU 暂时中断自身的事务, 转去处理事件的过程, 称为 CPU 的中断响应过程。对事件的整个处理过程, 称为中断服务(或中断处理)。处理完毕, 再回到原来被中止的地方, 称为中断返回。

PIC16F84 芯片有 4 种中断源, 其逻辑电路如图 2 所示。

## 9 中断控制

中断主要由中断控制寄存器 INTCON(图 3)来控制。INTCON 是一个可读/写寄存器, 含有定时器 TMR0 溢出、RB 口的变化和外部 INT 引脚中断等各种允许控制和标志位。



全局中断允许位 GIE(D7)置 1, 将开放所有未被屏蔽的中断, 如将该位清零, 将禁止所有的中断。在响应中断时, GIE 位将被清零, 以禁止其它中断, 返回的断点地址被压栈保护, 接着把中断入口地址 0004h 装入程序计数器 PC。在中断服务程序中, 通过对中断标志位进行查询, 确定中断标志位必须在重新开放中断之前用软件清零, 以避免不断地中断申请而反复进入中断。

(1) INT 中断。RBO/INT 引脚上的外部中断由边沿触发, 当 INTEDG 位(OPTION 寄存器第 6 位)被置 1 时, 选用上升沿触发, 如该位被清零, 则由下降沿触发。当检测到引脚上有规定的有效边沿时, 便把 INTE 位(INTCON 的 D4 位)置 1。在重新开放这个中断之前, 必须在中断服务程序中对 INTE 位清零。  
(2) TMR0 中断。当定时器 TMR0 的计数器计满溢出(即由 FFH 变成 00H)时, 硬件自动把 TOIF(INTCON 的 D2 位)置 1。其中断可以通过对 TOIE(INTCON 的 D5 位)置 1 或清零来控制该中断是否开放。

(3) PORTB 口引脚电平变化中断。在 PORTB 口的 D7~D0 引脚上一旦有电平变化, 就会把 RBIF(INTCON 的 D0 位)置 1。这个中断可以通过对 RBIE(INTCON 的 D3 位)置 1 或清零来控制该中断是否开放。

(4) 中断的现场保护。在发生中断时, 只有返回断点的地址被压栈保护。若用户还希望保护关键的寄存器(如 W 寄存器和 STATUS 寄存器)。这需要由软件来实现。有关中断的现场保护, 请参看本报第 15 期有关 PIC 单片机指令识读中的实例。

## 10 复位

复位是单片机的初始化操作。其主要功能是把程序计数器 PCL 初始化为 000H, 可使 16F84 单片机从 000H 单元开始执行程序。

PIC16F84 芯片有下列几种不同的复位方式。

- (1) 芯片上电复位 POR。
- (2) 正常工作状态下通过外部 MCLR 引脚加低电平复位。
- (3) 在省电休眠状态下通过外部 MCLR 引脚加低电平复位。
- (4) 监视定时器 WDT 超时溢出复位。

PIC16F84 片内集成有“上电复位”POR 电路，对于一般应用，只要把 MCLR 引脚接高电位即可。在正常工作或休眠状态下用 MCLR 复位，只需在 MCLR 引脚上加一按键瞬间接地即可。

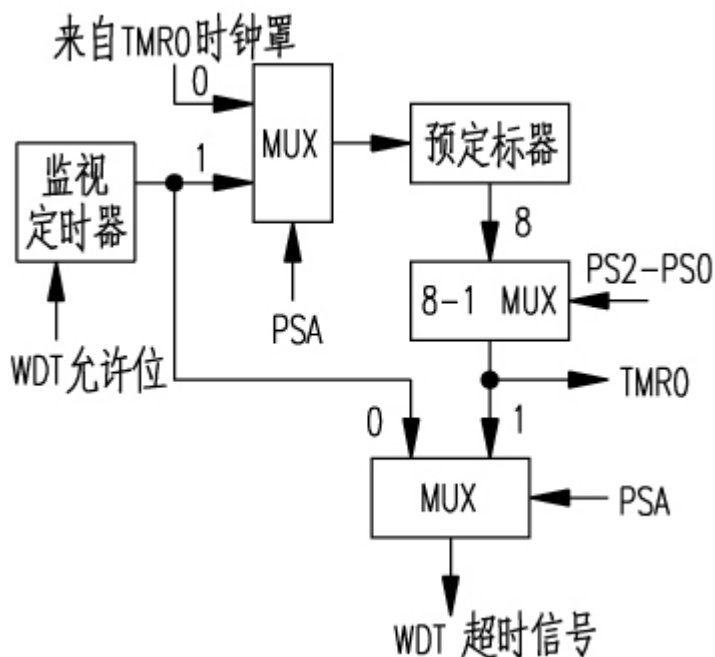
单片机 16F84 复位操作，对其它一些寄存器会有影响，如表 1 所示。

#### 11 监视定时器 WDT

单片机系统常用于工业控制，在操作现场通常会有各种干扰，可能会使执行程序弹飞到一种死循环，从而导致整个单片机控制系统瘫痪。如果操作者在场，就可进行人工复位，摆脱死循环。但操作者不能一直监视着系统，即使监视着系统，也往往是引起不良后果之后才进行人工复位。由于 PIC16F84 中具有程序运行自动监视系统，即监视定时器 WDT(Watch Dog Time)，直译为“看门狗”定时器。这好比是主人养了一条狗，主人在正常干活时总不忘每隔一段时间就给狗喂食，狗就保持安静，不影响主人干活。如果主人打瞌睡，不干活了，到一定时间，狗饿了，发现主人还没有给它吃东西，就会大叫起来，把主人唤醒。由此可见，WDT 有如下特性：

- (1) 本身能独立工作，基本上不依赖 CPU。
- (2) CPU 在一个固定的时间间隔中和 WDT 打一次交通(如使其清零，即喂一次狗)，以表明系统目前工作正常。
- (3) 当 CPU 落入死循环后，能被 WDT 及时发现(如 WDT 计数溢出)，并使系统复位。

PIC16F84 内的 WDT，其定时计数的脉冲序列由片内独立的 RC 振荡器产生，所以它不需要外接任何器件就可以工作。而且这个片内 RC 振荡器与 OSC1/CLKIN(引脚{16})上的振荡电路无关，即使 OSC1 和 OSC2 上的时钟不工作，WDT 照样可以监视定时。例如：当 PIC16F84 在执行 SLEEP 指令后，芯片进入休眠状态，CPU 不工作，主振荡器也停止工作，但是，WDT 照样可监视定时。当 WDT 超时溢出后，可激活(唤醒)芯片继续正常的操作。而在正常操作期间，WDT 超时溢出将产生一个复位信号。如果不需要这种监视定时功能，在固化编程时，可关闭这个功能。附图是监视定时器的结构框图。表 2 是与 WDT 有关的寄存器。



WDT 的定时周期在不加分频器的情况下，其基本定时时间是 18ms，这个定时时间还受温度、VDD 和不同元器件的工艺参数等的影响。如果需要更长的定时周期，还可以通过软件控制 OPT/ON 寄存器把预分频器配置给 WDT，这个预分频器的最大分频比可达到 1:128。这样就可把定时周期扩大 128 倍，即达到 2~3 秒。

如果把预分频器配置给 WDT，用 CLRWDT 和 SLEEP 指令可以同时给 WDT 和预分频器清零，从而防止计时溢出引起芯片复位。所以在正常情况下，必须在每次计时溢出之前执行一条 CLRWDT 指令（即喂一次“狗”），以避免引起芯片复位。当系统受到严重干扰处于失控状态时，就不可能在每次计时溢出之前执行一条 CLR WDT 指令，WDT 就产生计时溢出，从而引起芯片复位，从失控状态又重新进入正常运行状态。

当 WDT 计时溢出时，还会同时清除状态寄存器中的 D4 位 T0，检测 T0 位即可知道复位是否由于 WDT 计时溢出引起的。

## 12 E2PROM 的使用方法

在 PIC16F84 单片机中，除了可直接寻址的由 SRAM 构成的数据存储单元外，还另有可电擦、电写的 E2PROM 数据存储单元。该 E2PROM 共有 64 字节，其地址为 00~3FH 单元。由于 E2PROM 具有在线改写，并在掉电后仍能保持数据的特点，可为用户的特殊应用提供方便。16F84 的 E2PROM 在正常操作时的整个 VDD 工作电压范围内是可读写的，典型情况下可重写 100 万次，数据保存期大于 40 年。

PIC16F84 的 E2PROM 并未映象在寄存器组空间中，所以它们不能像 SRAM 通用寄存器那样用指令直接寻址访问，而需要通过专用寄存器进行间接寻址操作。因此，在 16F84 中增加了以下四个专用寄存器，即 EECON1、EECON2、EEDATA、EEADR，专门用于片内对 E2PROM 的操作。该专用寄存器中，EEDATA 存放 8 位读/写数据，EEADR 存放正在被访问的 E2PROM 存储单元的地址。

EECON1 是只有低五位的控制寄存器，其高三位不存在，读作“0”。具体见下表。

D7	D6	D5	D4	D3	D2	D1	D0
—	—	—	EEIF	WRERR	WREN	WR	RD

控制位 RD 和 WR 分别用于读写操作的启动，这两位可以由软件置 1，以启动读、写操作，但不能用软件清零，原因是防止不恰当的软件操作会使写入失败。当读写操作完成后由硬件自动清零，表

示此刻未对 E2PROM 进行读写操作。当 WREN 位被置 1 时, 允许进行写操作, 而在上电时该位被清零。在正常操作时, 一旦有 MCLR 或 WDT 复位, WRERR 位就置 1, 表示写操作被中止。当写操作完成时, EEIF 被置 1 (需由软件清零); 当写操作未完成或尚未启动时, EEIF 为“0”。

EECON2 仅是一个逻辑上的寄存器, 而不是一个物理上存在的寄存器, 读出时将总是为零。它只在写操作时起作用。

#### (1) E2PROM 的读操作

为进行一次 E2PROM 读操作, 需执行如下步骤:

1) 将 E2PROM 的单元地址放入 EEADR。2) 置 RD (EECON 的 D0 位)=1。3) 读取 EEDATA 寄存器。

程序段举例, 读取 25H 处的 E2PROM 存储器数据:

...

```
BCF    STATUS, RP0 ; 选 Bank0
MOVLW 25H
MOVWF EEADR      ; 地址 25H→EEADR
BSF    STATUS, RP0 ; 选 Bank1
BSF    EECON1, RD ; 启动读操作
BCF    STATUS, RP0 ; 选 Bank0
MOVF   EEDATA, W ; 将 E2PROM 数据
      ...      ; 读入 W 寄存器
```

#### (2) E2PROM 的写操作

要进行一次 E2PROM 写操作, 需执行如下步骤:

1) 将 E2PROM 单元地址放入 EEADR; 2) 将写入数据放入 EEDATA; 3) 执行一段控制程序段。

例如: 将数据 99H 写入 E2PROM 的 25H 单元, 需执行下列程序:

...

```
BCF    STATUS, RP0 ; 选 Bank0
MOVLW 25H
MOVWF EEADR      ; 地址→EEADR
MOVLW 99H
MOVWF EEDATA     ; 写入数据→EEDATA
BSF    STATUS, RP0 ; 选 Bank1
BSF    EECON1, WREN; 写操作功能允许
1 BCF   INTCON, GIE ; 关闭总中断
2 MOVLW 0x55
3 MOVWF EECON2
4 MOVLW 0xAA
5 MOVWF EECON2 ; 操作 EECON2
6 BSF   EECON1, WR; 启动写操作
7 BSF   INTCON, GIE ; 开放总中断
      ...
```

注意: 上列程序中的 2~6 条各语句必须严格执行, 否则不能启动 E2PROM 的写操作。而 1~7 条则是我们建议用户执行的操作, 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 的 D2 位) 是用来保证 E2PROM 不会被意外写入而设置的, 所以, 在平时, 用户程序应保持 WREN=0 以禁止写操作。只有当需对 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 WREN=1 启动写操作。上电复位后 WREN 位自动清零。

E2PROM 写操作约需 10ms 的时间才能完成。用户程序可通过查询 WR 位的状态(当 WR=0 时表示操作已完成), 或者用 E2PROM 写入完成中断来判断 E2PROM 写操作是否完成。如要使用中断, 应先置 EEIF(INTCON 的 D6)为 1, 以开中断。E2PROM 写完成要中断标志位 EEIF, 只能用软件清零。

### 13 编程指导

为了快速掌握 PIC 单片机源程序的基本结构, 这里给出一个典型的程序结构框架。建立源程序时首先用伪指令 TITLE 提供程序的标题, 接着给出整个程序的总说明, 并用列表伪指令 LIST 指定所用单片机型号和文件输出格式, 再利用 INCLUDE 伪指令读入 MPASM 中提供的定义文件如《P16F84 INC》, 然后对片内常用资源进行定义, 再给出一般程序的基本结构框架。现举例如下。

```
TITLE "This is....."; 程序标题
; 程序说明
LIST P=16F84, F=1NHX8M
;
include <p16F84.inc>
-config_RC_Qsc &_WDT_OFF...
; 资源定义和变量定义
STATUS EQU 03
FSR      EQU 04
PORTA   EQU 05
PORTB   EQU 06
J        EQU 01F
K        EQU 01E
; .....
      ORG 0000      ;
      goto MAIN    ; 跳过中断矢量
      ORG 0004
      goto INTSRV; 子程序入口地址
; .....
MAIN          ; 从 0005H 开始放主程序
      call Initports ; 端口初始化
      call InitTimers; 定时器初始化
      ...
INTSRV ...      ; 中断服务程序区
SVBRTH...      ; 子程序区
      END          ; 程序结束符
```

当然, 在编写程序时可根据实际情况加以调整。下面是一份实际程序清单, 要求将数据 88H 写入 PIC16F84 内部 EEPROM 的 20H 单元, 而后再从 20H 单元将其读出。

```
LIST P=16F84, F=INH8M
; .....
STATUS EQU 03      ; 定义寄存器
EEDATA EQU 08
EEADR   EQU 09
```

```

INTCON    EQU    0BH
EECON1    EQU    88H
EECON2    EQU    89H
; .....
RD        EQU    0        ; 定义位
WR        EQU    1
RPO       EQU    5
GIE       EQU    7
; .....
          ORG    0
          GOTO   WRSTART
; .....
          ORG    10H
WRSTART   ; 写入操作开始
          CLRW   ; 清 W, 使 W=0
          BCF   STATUS, RPO ; 选 BANK0
          MOVLW 20H
          MOVWF EEADR ; 地址→EEADR
          MOVLW 88H
          MOVWF EEDATA ; 写入数据→
                    ; EEDATA
          BSF   STATUS, RPO ; 选 BANK1
          BSF   EECON1, 2 ; 写操作使能允许
          BCF   INTCON, GIE ; 关闭所有的中断
          MOVLW 0X55
          MOVWF EECON2 ; 55H→EECON2
          MOVLW 0XAA
          MOVWF EECON2 ; AAH→EECON2
          BSF   EECON1, WR ; 启动写操作
          BSF   INTCON, GIE ; 恢复开中断
RDSTART   ; 读出操作开始
          BCF   STATUS, RPO
          MOVLW 20H
          MOVWF EEADR ; 地址→EEADR
          BSF   STATUS, RPO
          BSF   EECON1, RD ; 启动读操作
          BCF   STATUS, RPO
          MOVF  EEDATA, W ; 将 EEPROM
                    ; 数据读入 W
END

```

## 附录：练习题

1、简述 PIC 系列单片机助记符指令中操作数 f、d、b、k 取值的可能范围，并简要说明其依据。

- 2、指出 PIC16F84 框图(上文图 1)中寄存器的类型和不能访问的寄存器名称?
- 3、简述上文中 16F84 的数据存储器的一般特点。
- 4、用语言简述下工程清单中部分程序的功能, 并对主程序指令功能作进一步解释。
- 5、简述 PIC 单片机 I/O 口的功能。
- 6、PIC16C64A/RL64 与 PIC16C65 的管脚数相等, 管脚功能相近, 但 PIC16C64A/RL64 的 {16} 脚无 CCP2、{25} 脚无 TX/CK、{26} 脚无 RX/DT 等功能, 试绘出 PIC16C64A/RL64 的管脚功能图。
- 7、根据本版左下文介绍, 请以表的形式列出 PIC16C8X(有两个存储体和 A、B 两口)至少 11 个专用寄存器的名称和地址。
- 8、以本版上文电路为准, 在 A1 口上接一 LED, 要使该 LED 闪亮的延时间隔约为 0.4 秒, 请写出其源程序清单。
- 9、用简单的实例说明中断在 PIC 单片机中的用途。
- 10、简述 PIC 单片机中看门狗 WDT 的作用和功能。
- 11、简述 PIC 单片机带 E2PROM 的数据存储器的特点和用途。