

# PIC 系列单片机程序设计基础

## 1、程序的基本格式

先介绍二条伪指令：

EQU ——标号赋值伪指令

ORG ——地址定义伪指令

PIC16C5X 在 RESET 后指令计算器 PC 被置为全“1”，所以 PIC16C5X 几种型号芯片的复位地址为：

PIC16C54/55: 1FFH

PIC16C56: 3FFH

PIC16C57/58: 7FFH

一般来说，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
; -----
FO      EQU    0
RTCC    EQU    1
PC      EQU    2
STATUS  EQU    3
FSR     EQU    4
RA      EQU    5
RB      EQU    6
RC      EQU    7
... ..

PIC16C54 EQU 1FFH ; 芯片复位地址
PIC16C56 EQU 3FFH
PIC16C57 EQU 7FFH
; -----
ORG PIC16C54 GOTO MAIN ; 在复位地址处转入主程序
ORG 0 ; 在 0000H 开始存放程序
; -----
; 子程序区
; -----
DELAY MOVLW 255
... ..

RETLW 0
; -----
; 主程序区
; -----
MAIN
MOVLW B'00000000'
```

```

    TRIS RB          ; RB 已由伪指令定义为 6, 即 B 口
    ... ..
LOOP
    BSF RB, 7 CALL DELAY
    BCF RB, 7 CALL DELAY
    ... ..
GOTO LOOP
; -----
    END            ; 程序结束

```

注:MAIN 标号一定要处在 0 页面内。

## 2、程序设计基础

### 1) 设置 I/O 口的输入/输出方向

PIC16C5X 的 I/O 口皆为双向可编程, 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 TRIS f 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)
                    输入 输出
    TRIS 6         ; 将 W 中的 0FH 写入 B 口控制器,
                    ; B 口高 4 位为输出, 低 4 位为输入。
    MOVLW 0C0H    ; 11 000000 (0C0H)
                    RB4, RB5 输出 0 RB6, RB7 输出 1

```

### 2) 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, 现以寄存器 F10 为例:

```

    MOVF 10, 1     ; F10→F10, 结果影响零标记状态位 Z
    BTFSS STATUS, Z ; F10 为零则跳
    GOTO NZ        ; Z=0 即 F10 不为零转入标号 NZ 处程序
    ... ..        ; Z=1 即 F10=0 处理程序

```

### 3) 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—W (F8) →W
    BTFSC STATUS, Z ; 判断 F8=F9 否
    GOTO F8=F9
    BTFSC STATUS, C ; C=0 则跳
    GOTO F9>F8     ; C=1 相减结果为正, F9>F8
    GOTO F9<

```

F9 ; C=0 相减结果为负, F9<F8

↓

### 4) 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 10      ; 定义 F10 名称为 COUNT (计数器)
    |
    |
MOVW 8
MOVWF COUNT LOOP   ; 循环体
LOOP
    |
    |
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

```

#### 5) “IF……THEN……” 格式的程序

下面以 “IF X=Y THEN GOTO NEXT” 格式为例。

```

MOVF X, 0          ; X→W
SUBWF Y, 0         ; Y—W (X) →W
BTFSC STATUS, Z   ; X=Y 否
GOTO NEXT         ; X=Y, 跳到 NEXT 去执行。
    |
    |
                    ; X≠Y

```

#### 6) “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
DAEND EQU 11
    |
    |
MOVW 0
MOVWF START        ; 0→START (F10)
MOVW 5
MOVWF DAEND        ; 5→DAEND (F11)
LOOP
    |
    |
INCF START, 1      ; START 值加 1
MOVF START, 0
SUBWF DAEND, 0    ; START=DAEND ? (X=5 否)
BTFSS STATUS, Z
GOTO LOOP         ; X<5, 继续循环
    |
    |
                    ; X=5, 结束循环

```

#### 7) “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。

```

X EQU 10
    |
    |
MOVW 1
MOVWF X           ; 1→X (F10), 作为初值
LOOP
    |
    |
MOVW 1

```



```

    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。

#### 10) 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。

```

    MOVLW D '100'
    MOVWF 10
    LOOP DECFSZ 10, 1 ; F10—1→F10, 结果为零则跳
    GOTO LOOP
    ;

```

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 μS。所以单周期指令时间为 1 μS，双周期指令时间为 2 μS。在上例的 LOOP 循环延  
时时间即为： $(1+2) * 100 + 2 = 302$  (μS)。在循环中插入空操作指令即可延长延时时间：

```

    MOVLW D '100'
    MOVWF 10
    LOOP NOP
    NOP
    NOP
    DECFSZ 10, 1
    GOTO LOOP
    ;

```

延时时间 =  $(1+1+1+1+2) * 100 + 2 = 602$  (μS)。

用几个循环嵌套的方式可以大大延长延时时间。下例用 2 个循环来做延时：

```

    MOVLW D '100'
    MOVWF 10
    LOOP MOVLW D '16'
    MOVWF 11
    LOOP1 DECFSZ 11, 1
    GOTO LOOP1

```

```
DECFSZ    10, 1
```

```
GOTO LOOP
```

```
⋮
```

延时时间= $1+1+[1+1+(1+2)*16-1+1+2]*100-1=5201$  (μS)

#### 11) RTCC 计数器的使用

RTCC 是一个脉冲计数器，它的计数脉冲有二个来源，一个是从 RTCC 引脚输入的外部信号，一个是内部的指令时钟信号。可以用程序来选择其中一个信号源作为输入。RTCC 可被程序用作计时之用；程序读取 RTCC 寄存器值以计算时间。当 RTCC 作为内部计时器使用时需将 RTCC 管脚接 VDD 或 VSS，以减少干扰和耗电流。下例程序以 RTCC 做延时：

```
RTCC EQU 1
```

```
⋮
```

```
CLRF RTCC ; RTCC 清 0
```

```
MOVLW 07H
```

```
OPTION ; 选择预设倍数 1: 256→RTCC
```

```
LOOP MOVLW 255 ; RTCC 计数终值
```

```
SUBWF RTCC, 0
```

```
BTFSS STATUS, Z ; RTCC=255?
```

```
GOTO LOOP
```

```
⋮
```

这个延时程序中，每过 256 个指令周期 RTCC 寄存器增 1（分频比=1: 256），设芯片使用 4MHz 振荡，则：

延时时间= $256*256=65536$  (μS)

RTCC 是自振式的，在它计数时，程序可以去做别的事情，只要隔一段时间去读取它，检测它的计数值即可。

#### 12) 寄存器体 (BANK) 的寻址

对于 PIC16C54/55/56，寄存器有 32 个，只有一个体 (BANK)，故不存在体寻址问题，对于 PIC16C57/58 来说，寄存器则有 80 个，分为 4 个体 (BANK0-BANK3)。在对 F4 (FSR) 的说明中可知，F4 的 bit6 和 bit5 是寄存器体寻址位，其对应关系如下：

Bit6	Bit5	BANK	物理地址
0	0	BANK0	10H~1FH
0	1	BANK1	30H~3FH
1	0	BANK2	50H~5FH
1	1	BANK3	70H~7FH

当芯片上电 RESET 后，F4 的 bit6,bit5 是随机的，非上电的 RESET 则保持原先状态不变。

下面的例子对 BANK1 和 BANK2 的 30H 及 50H 寄存器写入数据。

例 1. (设目前体选为 BANK0)

```
BSF 4, 5 ; 置位 bit5=1,选择 BANK1
```

```
MOVLW DATA
```

```
MOVWF 10H ; DATA→30H
```

```
BCF 4, 5
```

```
BSF 4, 6 ; bit6=1,bit5=0 选择 BANK2
```

```
MOVWF 10H ; DATA→50H
```

从上例中我们看到，对某一体（BANK）中的寄存器进行读写，首先要先对 F4 中的体寻址位进行操作。实际应用中一般上电复位后先清 F4 的 bit6 和 bit5 为 0，使之指向 BANK0，以后再根据需要使其指向相应的体。

注意，在例子中对 30H 寄存器（BANK1）和 50H 寄存器（BANK2）写数时，用的指令“MOVWF 10H”中寄存器地址写的都是“10H”，而不是读者预期的“MOVWF 30H”和“MOVWF 50H”，为什么？

让我们回顾一下指令表。在 PIC16C5X 的所有有关寄存器的指令码中，寄存寻址位都只占 5 个位：ffff，只能寻址 32 个（00H—1FH）寄存器。所以要选 80 个寄存器，还要再用二位体选址位 PA1 和 PA0。当我们设置好体寻址位 PA1 和 PA0，使之指向一个 BANK，那么指令“MOVWF 10H”就是将 W 内容置入这个 BANK 中的相应寄存器内（10H, 30H, 50H, 或 70H）。

有些设计者第一次接触体选址的概念，难免理解上有出入，下面是一个例子：

例 2：（设目前体选为 BANK0）

```
MOVLW 55H
MOVWF 30H ; 欲把 55H→30H 寄存器
MOVLW 66H
MOVWF 50H ; 欲把 66H→50H 寄存器
```

以为“MOVWF 30H”一定能把 W 置入 30H，“MOVWF 50H”一定能把 W 置入 50H，这是错误的。因为这两条指令的实际效果是“MOVWF 10H”，原因上面已经说明过了。所以例 2 这段程序最后结果是 F10H=66H，而真正的 F30H 和 F50H 并没有被操作到。

建议：为使体选址的程序清晰明了，建议多用名称定义符来写程序，则不易混淆。 例

3：假设在程序中用到 BANK0，BANK1，BANK2 的几个寄存器如下：

BANK0	地址	BANK1	地址	BANK2	地址	BANK3	地址
A	10H	B	30H	C	50H	.	70H
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

```
A EQU 10H ; BANK0
B EQU 30H ; BANK1
C EQU 50H ; BANK2
;
FSR EQU 4
Bit6 EQU 6
Bit5 EQU 5
DATA EQU 55H
;
MOVLW DATA
MOVWF A
BSF FSR, Bit5
MOVWF B ; DATA→F30H
BCF FSR, Bit5
BSF FSR, Bit6
```



面后，马上设置相应的页面地址位（PA1，PA0）。页面处理是 PIC16C5X 编程中最麻烦的部分，不过并不难。只要做了一次实际的编程练习后，就能掌握了。

本文内容来自互联网，著作权归原作者所有。由电子零件城（<http://www.epcity.com/>）整理并制作成 PDF 文件，仅供个人学习之用，不得用于任何商业目的，否则后果自负。如果您认为本 PDF 文件侵犯了您的任何权利，请来信 [epcity@epcity.com](mailto:epcity@epcity.com) 通知，本站立即删除。

搜集整理：电子零件城-笨笨兔（QQ：154502842） 2004-04-10