PIC 单片机开发的若干问题

PIC 单片机在国内日益流行，本文介绍 Microchip PIC 系列单片机开发过程中软、硬件设计的一些经验、技巧。

由美国 Microchip 公司生产的 PIC 系列单片机，由于其超小型、低功耗、低成本、多品种等特点，已广泛应用于工业控制、仪器、仪表、通信、家电、玩具等领域，本文总结了作者在 PIC 单片机开发过程中的一些经验、技巧，供同行参考。

1 怎样进一步降低功耗

功耗，在电池供电的仪器仪表中是一个重要的考虑因素。PIC16C××系列单片机本身的功耗较低（在 5V，4MHz 振荡频率时工作电流小于 2mA）。为进一步降低功耗，在保证满足工作要求的前提下，可采用降低工作频率的方法，工作频率的下降可大大降低功耗（如 PIC16C××在 3V，32kHz 下工作，其电流可减小到 15μA），但较低的工作频率可能导致部分子程序（如数学计算）需占用较多的时间。在这种情况下，当单片机的振荡方式采用 RC 电路形式时，可以采用中途提高工作频率的办法来解决。

具体做法是在闲置的一个 I/O 脚（如 RB1）和 OSC1 管脚之间跨接一电阻（R_1），如图 1 所示。低速状态置 RB1=0。需进行快速运算时先置 RB1=1，由于充电时，电容电压上升得快，工作频率增高，运算时间减少，运算结束又置 RB1=0，进入低速、低功耗状态。工作频率的变化量依 R_1 的阻值而定（注意 R_1 不能选得太小，以防振荡电路不起振，一般选取大于 5kΩ）。

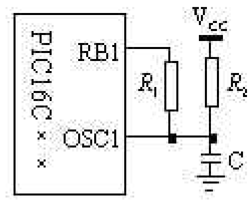

图 1 提高工作频率的方法

另外，进一步降低功耗可充分利用“sleep”指令。执行“sleep”指令，机器处于睡眠状态，功耗为几个微安。程序不仅可在待命状态使用“sleep”指令来等待事件，也可在延时程序里使用（见例 1、例 2）。在延时程序中使用“sleep”指令降低功耗是一个方面，同时，即使是关中断状态，Port B 端口电平的变化可唤醒“sleep”，提前结束延时程序。这一点在一些应用场合特别有用。同时注意在使用“sleep”时要处理好与 WDT、中断的关系。

例 1（用 Mplab-C 编写）

```
Delay()
  {
   /*此行可加开关中断指令*/
  for (i=0;i<=10;i++)
  SLEEP();
  }
```

例 2（用 Masm 编写）

```
Delay
;此行可加开关中断指令
movlw.10
movwf Counter
Loop1
Sleep
decfsz Counter
goto Loop1
return
```

2 注意 INTCON 中的 RBIF 位

INTCON 中的各中断允许位对中断状态位并无影响。当 PORTB 配置成输入方式时，RB<7:4>引脚输入在每个读操作周期被抽样并与旧的锁存值比较，一旦不同就产生一个高电平，置 RBIF=1。在开 RB 中断前，也许 RBIF 已置“1”，所以在开 RB 中断时应先清 RBIF 位，以免受 RBIF 原值的影响，同时在中断处理完成后最好是清 RBIF 位。

3 用 Mplab-C 高级语言写 PIC 单片机程序时要注意的问题

3.1 程序中嵌入汇编指令时注意书写格式 见例 3。

例 3

```
……
while(1){#asm
    ……
    #endasm
    }/*不能正确编译*/
……
```

```
……
while(1){
    #asm /*应另起一行*/
    ……
    #endasm
    }/*编译通过*/
……
```

当内嵌汇编指令时，从“#asm”到“endasm”每条指令都必须各占一行，否则编译时会出错。

3.2 加法、乘法的最安全的表示方法 见例 4。

例 4

```
#include<16c71.h>
#include<math.h>
unsigned int a,b;
unsigned long c;
void main()
{a=200;
 b=2;
 c=a∗b;
}/*得不到正确的结果 c=400*/
```

原因是 Mplab-C 以 8×8 乘法方式来编译 c=a∗b，返回单字节结果给 c，结果的溢出被忽略。改上例中的“c=a∗b；”表达式为“c=a；c=c∗b；”，最为安全（对加法的处理同上）。

3.3 了解乘除法函数对寄存器的占用

由于 PIC 片内 RAM 仅几十个字节，空间特别宝贵，而 Mplab-C 编译器对 RAM 地址具有不释放性，即一个变量使用的地址不能再分配给其它变量。如 RAM 空间不能满足太多变量的要求，一些变量只能由用户强制分配相同的 RAM 空间交替使用。而 Mplab-C 中的乘除法函数需借用 RAM 空间来存放中间结果，所以如果乘除法函数占用的 RAM 与用户变量的地址重叠时，就会导致出现不可预测的结果。如果 C 程序中用到乘除法运算，最好先通过程序机器码的反汇编代码（包含在生成的 LST 文件中）

查看乘除法占用地址是否与其它变量地址有冲突，以免程序跑飞。Mplab-C 手册并没有给出其乘除法函数对具体 RAM 地址的占用情况。例 5 是乘法函数对 0×13、0×14、0×19、0×1A 地址占用情况。

例 5

	部分反汇编代码		
#include <pic16c71>	01A7	081F	MOVF1F,W
#include<math.h>	01A8	0093	MOVWF13
	；借用		
unsigned long Value @0x1	01A9	0820	MOVF20,W
char Xm @0x2d;	01AA	0094	MOVWF14
	；借用		
void main()	01AB	082D	MOVF2D,W
{Value=20;	01AC	0099	MOVWF19
	；借用		
Xm=40;	01AD	019A	CLRF1A
	；借用		
Value=Value＊Xm	01AE	235F	CALL035Fh
	；调用乘法函数		
……	01AF	1283	BCF03，5
}	01B0	009F	MOVWF1F
	；返回结果低字节		
	01B1	0804	MOVF04，W
	01B2	00A0	MOVWF20
	；返回结果高字节		

4 对芯片重复编程

对无硬件仿真器的用户，总是选用带 EPROM 的芯片来调试程序。每更改一次程序，都是将原来的内容先擦除，再编程，其过程浪费了相当多的时间，又缩短了芯片的使用寿命。如果后一次编程的结果较前一次，仅是对应的机器码字节的相同位由“1”变成“0”，就可在前一次编程芯片上再次写入数据，而不必擦除原片内容。

在程序的调试过程中，经常遇到常数的调整，如常数的改变能保证对应位由“1”变“0”，都可在原片内容的基础继续编程。另外，由于指令“NOP”对应的机器码为“00”，调试过程中指令的删除，先用“NOP”指令替代，编译后也可在原片内容上继续编程。

另外,在对带 EPROM 的芯片编程时,特别注意程序保密状态位。厂家对新一代带 EPROM 芯片的保密状态位已由原来的 EPROM 可擦型改为了熔丝型，一旦程序代码保密熔丝编程为“0”，可重复编程的 EPROM 芯片就无法再次编程了。使用时应注意这点，以免造成不必要的浪费（Microchip 资料并未对此做出说明）。