51单片机汇编语言教程：第19课-单片机定时器、中断实验

（基于 HJ-1G、HJ-3G 实验板）

我们在学单片机时我们第一个例程就是灯的闪烁，那是用延时程序做的，现在回想起来，这样做不很恰当，为什么呢？我们的主程序做了灯的闪烁，就不能再干其它的事了，难道单片机只能这样工作吗？当然不是，我们能用定时器来实现灯的闪烁的功能。

例1：查询方式

```
ORG 0000H
AJMP START
ORG 30H
START:
MOV P1,#0FFH ;关所 灯
MOV TMOD,#00000001B ;定时/计数器0工作于方式1
MOV TH0,#15H
MOV TL0,#0A0H ;即数5536
SETB TR0 ;定时/计数器0开始运行
LOOP:JBC TF0,NEXT ;如果 TF0等于1，则清 TF0并转 NEXT 处
AJMP LOOP ;不然跳转到 LOOP 处运行
NEXT:CPL P1.0
MOV TH0,#15H
MOV TL0,#9FH;重置定时/计数器的初值
AJMP LOOP
END AJMP LOOP
END
```

键入程序，看到了什么？灯在闪烁了，这可是用定时器做的，不再是主程序的循环了。简单地分析一下程序，为什么用 JBC 呢？TF0是定时/计数器0的溢出标记位，当定时器产生溢出后，该位由0变1，所以查询该位就可知字时时间是否已到。该位为1后，要用软件将标记位清0，以便下一次定时是间到时该位由0变1，所以用了 JBC 指令，该指位在判1转移的同时，还将该位清0。

以上程序是能实现灯的闪烁了，可是主程序除了让灯闪烁外，还是不能做其他的事啊！不，不对，我们能在 LOOP：……和 AJMP LOOP 指令之间插入一些指令来做其他的事情，只

要保证执行这些指令的时间少于定时时间就行了。那我们在用软件延时程序的时候不是也能用一些指令来替代 DJNZ 吗？是的，但是那就要求你精确计算所用指令的时间，然后再减去对应的 DJNZ 循环次数，很不方便，而现在只要求所用指令的时间少于定时时间就行，显然要求低了。当然，这样的办法还是不好，所以我们常用以下的办法来实现。

程序2：用中断实现

```
ORG 0000H   ,http://www.51hei.com
AJMP START
ORG 000BH ;定时器0的中断向量地址
AJMP TIME0 ;跳转到真正的定时器程序处
ORG 30H
START:
MOV P1,#0FFH ;关所 灯
MOV TMOD,#00000001B ;定时/计数器0工作于方式1
MOV TH0,#15H
MOV TL0,#0A0H ;即数5536
SETB EA ;开总中断允许
SETB ET0 ;开定时/计数器0允许
SETB TR0 ;定时/计数器0开始运行
LOOP: AJMP LOOP ;真正工作时,这里可写任意程序
TIME0: ;定时器0的中断处理程序
PUSH ACC
PUSH PSW ;将 PSW 和 ACC 推入堆栈保护
CPL P1.0
MOV TH0,#15H
MOV TL0,#0A0H ;重置定时常数
POP PSW
POP ACC
RETI
END
```

上面的例程中，定时时间一到，TF0由0变1，就会引发中断，CPU 将自动转至000B 处寻

找程序并执行，由于留给定时器中断的空间只有8个字节，显然不足以写下所有有中断处理程序，所以在000B 处安排一条跳转指令，转到实际处理中断的程序处，这样，中断程序能写在任意地方，也能写任意长度了。进入定时中断后，首先要保存当前的一些状态，程序中只演示了保存存 ACC 和 PSW，实际工作中应该根据需要将可能会改变的单元的值都推入堆栈进行保护（本程序中实际不需保存护任何值，这里只作个演示）。

上面的两个单片机程序运行后，我们发现灯的闪烁非常快，根本分辨不出来，只是视觉上感到灯有些晃动而已，为什么呢？我们能计算一下，定时器中预置的数是5536，所以每计60000个脉冲就是定时时间到，这60000个脉冲的时间是多少呢？我们的晶体震荡器是12M，所以就是60000微秒，即60毫秒，因此速度是非常快的。如果我想实现一个1S 的定时，该怎么办呢？在该晶体震荡器濒率下，最长的定时也就是65。536个毫秒啊！上面给出一个例程。

```
ORG 0000H
AJMP START
ORG 000BH ;定时器0的中断向量地址
AJMP TIME0 ;跳转到真正的定时器程序处
ORG 30H
START:
MOV P1,#0FFH ;关所 灯
MOV 30H,#00H ;软件计数器预清0
MOV TMOD,#00000001B ;定时/计数器0工作于方式1
MOV TH0,#3CH
MOV TL0,#0B0H ;即数15536
SETB EA ;开总中断允许
SETB ET0 ;开定时/计数器0允许
SETB TR0 ;定时/计数器0开始运行
LOOP: AJMP LOOP ;真正工作时,这里可写任意程序
TIME0: ;定时器0的中断处理程序
PUSH ACC
PUSH PSW ;将 PSW 和 ACC 推入堆栈保护
INC 30H
MOV A,30H
```

```
CJNE A,#20,T_RET ;30H 单元中的值到了20了吗?
T_L1: CPL P1.0 ;到了,取反 P10
MOV 30H,#0 ;清软件计数器
T_RET:
MOV TH0,#15H
MOV TL0,#9FH ;重置定时常数
POP PSW
POP ACC
RETI
END
```

先自己分析一下，看看是怎么实现的？这里采用了软件计数器的概念，思路是这样的，先用定时/计数器0做一个50毫秒的定时器，定时是间到了以后并不是立即取反 P10，而是将软件计数器中的值加1，如果软件计数器计到了20，就取反 P10，并清掉软件计数器中的值，不然直接返回，这样，就变成了20次定时中断才取反一次 P10，因此定时时间就延长了成了20*50即1000毫秒了。

这个思路在工程中是非常有用的，有的时候我们需要若干个定时器，可51中总共才有2个，怎么办呢？其实，只要这几个定时的时间有一定的公约数，我们就能用软件定时器加以实现，如我要实现 P10口所接灯按1S 每次，而 P11口所接灯按2S 每次闪烁，怎么实现呢？对了我们用两个计数器，一个在它计到20时，取反 P10，并清零，就如上面所示，另一个计到40取反 P11，然后清0，不就行了吗？这部份的程序如下

```
ORG 0000H
AJMP START
ORG 000BH ;定时器0的中断向量地址
AJMP TIME0 ;跳转到真正的定时器程序处
ORG 30H
START:
MOV P1,#0FFH ;关所 灯
MOV 30H,#00H ;软件计数器预清0
MOV TMOD,#00000001B ;定时/计数器0工作于方式1
MOV TH0,#3CH
```

```
MOV TL0,#0B0H ;即数15536
SETB EA ;开总中断允许
SETB ET0 ;开定时/计数器0允许
SETB TR0 ;定时/计数器0开始运行
LOOP: AJMP LOOP ;真正工作时,这里可写任意程序
TIME0: ;定时器0的中断处理程序
PUSH ACC
PUSH PSW ;将 PSW 和 ACC 推入堆栈保护
INC 30H
INC 31H ;两个计数器都加1
MOV A,30H
CJNE A,#20,T_NEXT ;30H 单元中的值到了20了吗?
T_L1: CPL P1.0 ;到了,取反 P10
MOV 30H,#0 ;清软件计数器
T_NEXT:
MOV A,31H
CJNE A,#40,T_RET ;31h 单元中的值到40了吗?
T_L2:
CPL P1.1
MOV 31H,#0 ;到了,取反 P11,清计数器,返回
T_RET:
MOV TH0,#15H
MOV TL0,#9FH ;重置定时常数
POP PSW
POP ACC
RETI
END
```

您能用定时器的办法实现前面讲的流水灯吗？试试看。

51 单片机汇编语言教程-慧净电子会员收集整理 （全部 28 课）

51 实验板推荐(点击下面的图片可以进入下载资料链接)

HJ-1G	HJ-3G