第 5 章　51 系列单片机的中断系统

本章主要分析 51 系列单片机中断系统结构及工作原理，给出中断的相关概念、中断控制寄存器以及中断执行过程，并通过实例说明中断程序设计的思路。

【本章目标】

- 理解中断及中断嵌套的概念；
- 掌握 51 系列单片机中断模块结构、工作原理及中断控制寄存器；
- 能够运用汇编及 C 语言编写中断程序。

5.1　中断的概念

5.1.1　对中断的理解

中断是单片机或数字计算机为提高工作效率，并行实时处理事件的一种机制，关于中断的几种理解：

理解 1：如图 5.1 所示，CPU 正在处理某一事件 A 时，发生了事件 B(中断发生)，请求 CPU 迅速去处理 B；CPU 暂停(中断)当前的工作，转去处理事件 B(响应中断和执行中断服务)；待 CPU 处理完事件 B，再回到原事件 A 被中断的地方继续执行事件 A(中断返回)，这一过程称为中断。

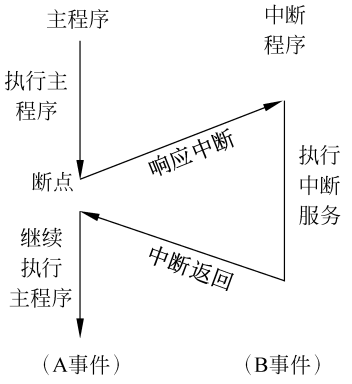

图 5.1　中断结构示意图

理解 2：处理器和外设交换信息时，存在着快速 CPU 和慢速外设间的矛盾，处理器内部有时也会出现突发事件，为此，处理器通常用中断技术解决上述问题。CPU 和外设并行工作，当外设数据准备好或有某种突发事件发生时，向 CPU 提出请求，CPU 暂停正在执行的主程序转而为外设服务(或处理突发事件)，处理完毕再回到主程序断点处继续执行原程序，这个过程称为中断。

5.1.2　中断系统

能实现中断任务处理并能对中断过程进行管理的硬件和软件称为中断系统，单片机的中断系统要解决如下问题：

(1) 中断检测：单片机在执行主程序过程中，如何知道有中断请求发生？

(2) 中断的开放和关闭：编程过程中，需要中断是可人为控制的，即如何打开或关闭中断？

(3) 中断的执行：中断请求是在执行程序的过程中随机发生的，那么中断发生时，如何实现正确的转移，以便为中断源服务？

(4) 中断的嵌套及优先级控制：中断源有多个，而 CPU 只有一个，当有多个中断源同时有中断请求时，如何控制 CPU 根据自己的需要安排响应顺序？

(5) 中断的返回：中断程序执行完毕，如何正确地返回到原程序断点处？

后面各节将围绕上述问题讨论 51 系列单片机的中断实现过程。

5.2　51 系列单片机的中断源

5.2.1　中断源

中断源：引起中断的原因和发出中断请求的来源。

8XX51 单片机有 5 个中断源，增强型 52 系列单片机增加了一个定时器/计数器 T_2，有 6 个中断源，其中有两个外部中断源，其余为内部中断源。这些中断源的符号、名称、产生条件及中断服务程序的入口地址见表 5.1。

表 5.1　8XX51/52 的中断源

中断源符号	名　　称	引起中断原因	中断入口地址	中断向量号
$\overline{INT_0}$	外部中断 0	P3.2 引脚的低电平或下降沿信号	0003H	0
T0	定时器 0 中断	定时器/计数器 0 计数回零溢出	000BH	1
$\overline{INT1}$	外部中断 1	P3.3 引脚的低电平或下降沿信号	0013H	2
T1	定时器 1 中断	定时器/计数器 1 计数回零溢出	001BH	3
TI/RI	串行口中断	串行通信完成一帧数据发送或接收引起中断	0023H	4

中断源可以是外设、紧急事件、定时器或人为设置的实时任务程序。外部中断，在单片机外部引脚上设置触发信号，满足一定条件就引起中断；内部中断是单片机内部中断源产生的中断请求，不需要外部引脚上的中断请求信号。8051 单片机各个中断源在程序存储器中均有各自固定的中断程序入口地址(见表 5.1)，当 CPU 响应中断时，硬件自动形成各自的入口地址，由此进入中断服务程序，从而实现正确的转移。这些中断源有两级中断优先级，可行使中断嵌套；三个特殊功能寄存器用于中断控制编程，后面将会具体分析。

5.2.2　中断优先级与中断嵌套

当有多个中断源同时向 CPU 申请中断时，CPU 优先响应最需紧急处理的中断请求，处理完毕再去响应优先级别较低的中断请求，这种可预先安排的中断先后响应次序，称为中断优先级。51 系列单片机和一般计算机一样，当几个中断源同时向 CPU 请求中断时，就存在 CPU 优先响应哪一个中断源的问题。一般 CPU 应优先响应最需紧急处理的中断请求，为此需要规定各个中断源的优先级，使 CPU 在多个中断源同时发出中断请求时能找到优先级最高的中断源，及时响应其请求。在优先级高的中断请求处理完后，再响应优先级低的中断请求。

当 CPU 正在处理一个优先级低的中断请求的时候，如果发生另一个优先级比它高的中断请求，CPU 暂停正在处理的中断源的处理程序，转而处理优先级高的中断请求，待处理完之后，再回到原来正在处理的低级中断程序，这种高级中断源中断低级中断源的中断处理过程称为中断嵌套。具有中断嵌套的系统称为多级中断系统，没有中断嵌套的系统称为单级中断系统。

8051 单片机中断源提供两个中断优先级，能实现两级中断嵌套。每一个中断源优先级

的高低都可以通过编程设定。两级中断嵌套的中断处理过程如图 5.2 所示。

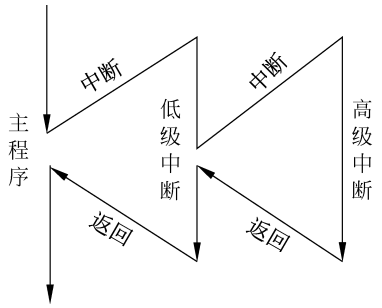

图 5.2　中断嵌套执行流程

当某几个中断源优先级设置相同时，由 CPU 内部查询确定优先级，优先响应先查询的中断请求，称该顺序为自然优先级。CPU 查询的顺序是：

INT0→T0→INT1→T1→TI/RI→T2

5.3　中断控制寄存器

单片机有多个中断源，根据前面提出的问题，如何知道是否有相应的中断请求？对于每个中断源，如何允许和关闭其中断请求？每个中断源有两个中断优先级，如何设置其优先级别，这些均通过下面的功能寄存器进行控制。

1. 中断标志寄存器 TCON

字节地址：88H	D7	D6	D5	D4	D3	D2	D1	D0	TCON
寻址位	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
	←——定时器使用位——→				←——外部中断使用位——→				

(1) IT_x($x=0$ 或 1)：外部中断 0 或 1 触发方式控制位。

当 $IT_x=0$ 时，为低电平触发；

当 $IT_x=1$ 时，为下降沿触发。

(2) IE_x($x=0$ 或 1)：外部中断 0 或 1 中断请求标志位。

若 $IE_X=1$，中断源有中断请求；$IE_X=0$，无中断请求。

(3) TR0 和 TR1 为定时器 T0 和 T1 启动和停止位。

(4) TF_x($x=0$ 或 1)：定时器/计数器 T0 或 T1 溢出中断请求标志位。

若 T0 或 T1 发出溢出中断请求，$TF_X=1$；无中断请求，$TF_X=0$。

2. 中断允许寄存器 IE

位	D7	D6	D5	D4	D3	D2	D1	D0	IE
字节地址：A8H	EA			ES	ET1	EX1	ET0	EX0	

0 禁止，1 允许

(1) EA：中断总控制位；

(2) ES：串行中断允许位；

(3) ET1：定时器/计数器 T1 中断允许位；

(4) EX1：外部中断 1 允许位；

(5) ET0：定时器/计数器 T0 中断允许位；

(6) EX0：外部中断 0 允许位。

3. 中断优先级寄存器 IP

位	D7	D6	D5	D4	D3	D2	D1	D0	IP
字节地址：**8BH**			PT2	PS	PT1	PX1	PT0	PX0	

各中断优先级控制位设为 0，低优先级；设为 1，高优先级。

当某几个中断源在 IP 寄存器相应位同为“1”或同为“0”时，由内部查询确定优先级。

5.4　中断执行过程

5.4.1　中断系统结构

如图 5.3 所示，外部中断有下降沿引起和低电平引起的选择；串行中断有发送(TI)和接收(RI)的区别；各个中断源打开与否，受中断自身的允许位和全局允许位的控制，并具有高优先级和低优先级的选择。

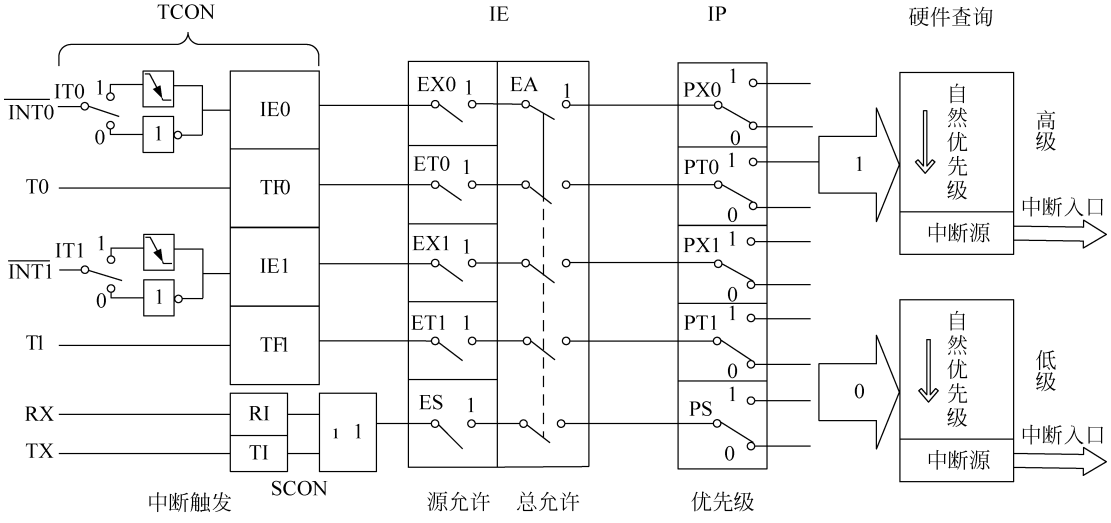

图 5.3　8XX51 单片机的中断系统

1. 中断触发

外部中断 INT0(P3.2)可由 IT0(TCON.0)选择其为低电平触发还是下降沿触发。当 CPU 检测到 P3.2 引脚上出现有效的中断信号时，中断标志 IE0(TCON.1)置 1，向 CPU 申

请中断。

外部中断 INT1(P3.3)可由 IT1(TCON.2)选择其为低电平触发还是下降沿触发。当 CPU 检测到 P3.3 引脚上出现有效的中断信号时,中断标志 IE1(TCON.3)置 1,向 CPU 申请中断。

片内定时器/计数器 T0 溢出中断请求标志 TF0(TCON.5)。当定时器/计数器 T0 发生溢出时,置位 TF0,并向 CPU 申请中断。

片内定时器/计数器 T1 溢出中断请求标志 TF1(TCON.7)。当定时器/计数器 T1 发生溢出时,置位 TF1,并向 CPU 申请中断。

串行接口中断请求标志 RI(SCON.0)或 TI(SCON.1)。当串行接口接收完一帧串行数据时置位 RI 或当串行接口发送完一帧串行数据时置位 TI,向 CPU 申请中断。

2. 中断允许

8051 单片机中断实行两级控制,允许某中断源的做法是,分别开放其中断控制位和总的中断控制位。例如,允许外部中断 0 响应中断。

```
SETB   EX0                                    ;INTO 中断控制位
SETB   EA                                     ;总的中断控制位
```

3. 中断优先级设置

8051 单片机每个中断源有两个中断优先级,通过寄存器 IP 各个位设置优先级,在多个中断源优先级相同时,硬件按自然优先级查询顺序。

例如:设置外部中断 0 位高优先级,设置定时器 T0 为低优先级。

```
SETB PX0                                      ;INTO 中断控制位
CLR  PT0                                      ;总的中断控制位
```

5.4.2 中断响应

1. 中断响应过程

CPU 响应中断时先把当前指令的下一条指令就是中断返回后将要执行的指令地址(断点地址)送入堆栈,然后根据中断标记,硬件执行跳转指令,转到相应的中断源入口处,执行中断服务程序,当遇到 RETI 返回指令时,返回到断点处继续执行程序,这些工作都是由硬件自动完成的。上述过程分为以下几个步骤:

(1) 停止主程序运行,当前指令执行完后立即终止现在执行的程序。

(2) 保护断点,即保存下一个将要执行的指令的地址,即把 PC 地址送入堆栈。

(3) 寻找中断入口,根据 5 个不同中断源所产生的中断,查找对应的入口地址。以上工作由硬件自动完成,与编程者无关。

(4) 执行中断处理程序。中断处理程序编写好存放于对应中断向量地址处,否则,中断程序就不能被执行。

(5) 中断返回:执行完中断服务程序,从断点返回主程序,继续执行主程序。

2. 中断响应条件

了解了上述中断响应过程,还需明确中断响应的条件。在下列 3 种情况之一时,CPU 将封锁对中断的响应:

（1）CPU 正在处理一个同级或更高级别的中断请求。

（2）现行的机器周期不是当前正执行指令的最后一个周期。单片机有单周期、双周期、三周期指令，需要等当前整条指令都执行完，才能响应中断，因为中断查询是在每个机器周期都可能查到的。

（3）当前正执行的指令是访问 IP、IE 寄存器的指令或返回指令 RETI，则 CPU 至少再执行一条指令才能响应中断。这些都是与中断有关的，如果正访问 IP、IE 则可能会开、关中断或改变中断的优先级，而执行中断返回指令则说明本次中断还没有处理完，所以都要等本指令处理结束，再执行一条指令才可以响应中断。在正常情况下，从中断请求信号有效开始，到中断得到响应，通常需要 3～8 个机器周期。

5.4.3　中断执行流程

51 系列单片机的中断过程流程如图 5.4 所示。中断处理过程主要分为 4 个阶段：中断请求、中断响应、中断服务和中断返回。

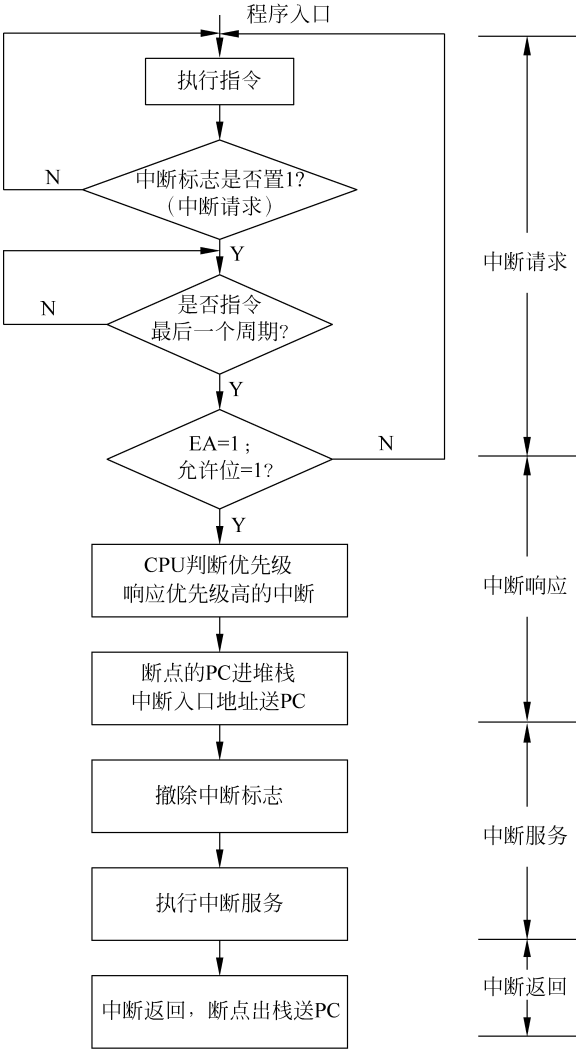

图 5.4　中断处理流程图

(1) 中断请求,CPU 执行程序时,在每一个指令周期的最后一个周期都检查是否有中断请求,如果有中断请求,寄存器 TCON 的相应位置"1",CPU 查到"1"标志后,如果允许中断,进入中断响应阶段,如果中断被禁止或没有中断请求,继续执行下一条指令。

(2) 中断响应阶段,如果有多个中断源,CPU 判断哪个的优先级高,优先响应优先级高的中断请求。阻断同级或低级的中断,硬件自动将断点 PC 压入堆栈,将所响应的中断源的入口地址送到 PC,转到中断服务程序执行。

中断服务是完成中断要处理的工作任务,程序员根据任务需要编写中断服务程序,让单片机借助中断实时处理特定事件。但要注意 51 单片机响应中断,不会自动保护标志寄存器 PSW,不会自动关中断,编写中断服务程序要注意将主程序中需要保护的寄存器内容进行自我保护,中断服务执行完毕再恢复寄存器内容,即保护现场,这些可以通过堆栈操作来完成。

CPU 响应中断后,应及时撤除中断请求,否则会引起再次中断。对定时器/计数器 T0、T1 的溢出中断,CPU 响应中断后,硬件清除中断请求标志 TF0 和 TF1,即自动撤除中断请求,除非 T0、T1 再次溢出,才产生中断。对边沿触发的外部中断 INT0 和 INT1,也是 CPU 响应中断后硬件自动清除 IE0 和 IE1。对于串口和定时器/计数器 T2 中断,CPU 响应中断后,没有用硬件清除中断请求标志 TI、RI、TF2 和 EXF2,即这些中断标志不会自动清除,必须用软件清除。对电平触发的外部中断,CPU 在响应中断时也不会自动清除中断标志,因此,在 CPU 响应中断后,应立即撤除 INT1 和 INT0 的低电平信号。

中断返回是通过执行一条 RETI 中断返回指令完成的,该指令使堆栈中被压入的断点地址弹到 PC,从而返回主程序的断点继续执行主程序。另外,RETI 还有恢复优先级状态触发器的作用,因此不能以 RET 指令代替 RETI 指令。

5.5 中断服务程序的编写

单片机为什么要有中断系统,使用中断编程的好处,具体来说有以下几点:

(1) 实行分时操作,提高 CPU 效率,只有当外设对象向 CPU 发出中断申请时,才去为它服务,这样,我们就可以利用中断功能同时为多个对象服务,从而大大提高 CPU 的工作效率。

(2) 实现实时处理。利用中断技术,各个服务对象可以根据需要随时向 CPU 发出中断申请,及时发现和处理中断请求并为之服务,以满足实时控制的要求。

(3) 进行故障处理。对难以预料的情况或故障,比如掉电等,可以向 CPU 发出请求中断,由 CPU 作出相应的处理。

5.5.1 汇编语言中断程序设计

对于一个独立的单片机应用系统,上电初始化时,PC 总指向 0000H 单元,从 0 单元开始执行程序,由于 8XX51/52 中断服务程序的入口地址分别为 0003H、000BH、0013H、001BH、0023H,为了让出中断源所占用的地址区,在程序存储器的 0 地址单元通常安排一条转移指令,以绕过中断向量地址空间。此外,我们发现中断向量地址之间相距很近,往往放不下一个中断服务程序,所以通常在中断向量地址单元中放一条转移指令,将中断服务程序安排在程序存储器后面的地址空间,当然,如果系统仅有一个中断任务且处理较少中断服

务程序可直接存放于中断向量地址处。一个完整的主程序如下：

```
        ORG    0000H
        LJMP   MAIN
        ORG    0003H
        LJMP   SER0                          ;转外部中断 0 服务程序
        ORG    000BH
        RETI                                 ;没有用定时器 0 中断,在此放一条 RETI
        ORG    0013H
        RETI                                 ;没有用外部中断 1 中断,在此放一条 RETI
        …
        ORG    0030H
MAIN:   …
        …
        SJMP   MAIN
        ORG    0100H
SER0:   …                                    ;外部中断 0 服务程序
        …
        RETI
        END
```

中断程序处理完成后，一定要执行一条 RETI 指令，执行这条指令后 CPU 将会把堆栈中保存着的断点地址取出，送回程序计数器 PC 中，那么程序就会根据 PC 中的值，从主程序的中断处继续往下执行了。

对中断模块的使用，实际就是对中断系统各特殊功能寄存器的设置和操作，即对中断的功能寄存器 TCON、IE、IP 的管理。编写中断服务程序，执行特定任务，必须根据需要先对这几个寄存器的有关位进行设置。中断程序编制基本思路：

(1) 初始化 IE、IP，开中断，设置中断优先级；

(2) 对外部中断，设置 ITx 位，选择中断触发方式，是低电平触发还是下降沿触发；

(3) 确定中断入口地址或中断号，编写中断服务程序。

5.5.2　C51 中断程序设计

C51 使用户能编写高效的中断服务程序，编译器在规定的中断源的矢量地址中放入无条件转移指令，使 CPU 响应中断后自动地从矢量地址跳转到中断服务程序的实际地址，而无须用户去安排。中断服务程序定义为函数，函数的完整定义如下。

```
返回值　函数名([参数])[再入]interrupt n[using m]
```

其中，interrupt　n 表示将函数声明为中断服务函数，n 为中断源编号，可以是 0～31 的整数，不允许是带运算符的表达式，n 通常取以下值：

0　外部中断 0；

1　定时器/计数器 T0 溢出中断；

2　外部中断 1；

3　定时器/计数器 T1 溢出中断；

4　串行口发送与接收中断；

5　定时器/计数器 T2 中断。

using m 定义函数使用的工作寄存器组，m 的取值范围为 0～3，可缺省，它对目标代码的影响是：函数入口处将当前寄存器保存，使用 m 指定的寄存器组，函数退出时原寄存器组恢复。选不同的工作寄存器组，可方便实现寄存器组的现场保护。

再入：属性关键字 reentrant 将函数定义为再入的，在 C51 中，普通函数（非再入的）不能递归调用，只有再入函数才可被递归调用。

以外部中断 0 为例，主程序中需要有以下代码：

```
EA = 1;                                    //打开总中断开关
EX0 = 1;                                   //开外部中断 0
IT0 = 1;                                   //设置外部中断的触发方式
```

中断服务函数：

```
void ser_int0( ) interrupt 0 using 0
{
    Do anything that you want
}
```

5.6 中断服务程序设计

【例 5-1】 如图 5.5 所示，P1 口接 8 个发光二极管，利用消抖电路产生中断请求信号，来回拨动一次开关 K，产生一次中断申请，实现下移一个灯亮。

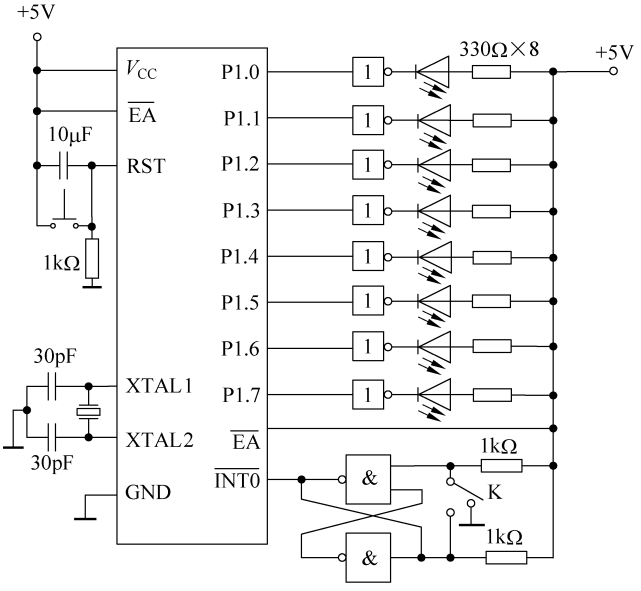

图 5.5 中断控制灯显示

解：

（1）C 程序。

```
#include <reg52.h>
```

```
typedef unsigned int u16;                    //对数据类型进行声明定义
typedef unsigned char u8;
u8 j;
sbit K = P3^2;                               //定义按键
void delay(u16 i)                            //定义延时函数
{
    while(i--);
}
void main( )
{
    IT0 = 1;                                 //跳变沿出发方式(下降沿)
    EX0 = 1;                                 //打开 INT0 的中断允许.
    EA = 1;                                  //打开总中断
    while(1);
}
void Ser_Int0( ) interrupt 0                 //INT0 的中断函数
{
    delay(1000);                             //延时消抖
    if(K == 0)
    {
     P2 = 0x01 << j;                         //将 1 左移 j 位,然后将结果取反赋值到 P2 口
     j++;
    }
     if(j > 7) j = 0;
}
```

(2) 汇编程序。

```
      ORG    0000H
      AJMP   MAIN
      ORG    0003H                           ;INT0 中断入口
      AJMP   SER0                            ;转中断服务程序
      ORG    0030H                           ;主程序
MAIN: MOV    P1,#01H                         ;灯初始状态设置
      MOV    A,P1
      SETB   IT0                             ;边沿触发中断
      SETB   EX0                             ;允许外部中断 0 中断
      SETB   EA                              ;开总中断
HERE: SJMP   HERE                            ;等中断
SER0: RL     A
      MOV    P1,A                            ;灯状态输出到 P1
      RETI
      END
```

以上通过中断方式分别用C语言和汇编语言控制灯的输出状态。如果不用中断,可否控制灯状态的切换？其实,与中断对应还有一种编程思路,即查询方式,上面例子也可以通过查询方式编写程序：

```
      …
      SETB   IT0
LOOP: JNB    IE0,$                           ;查询标志位
```

```
        RL      A
        MOV     P1,A
        CLR     IE0                          ;清零标志位
        ACALL   DELAY                        ;延时函数
        SJMP    LOOP
        …
```

采用查询方式编程，系统需反复循环查询事件是否发生，同时需注意对标志位的软件清零操作。

【例 5-2】 如图 5.6 所示的单片机 AT89S51，其 P1 口接一个共阴极的数码管，消抖开关接到外部中断 0 引脚，产生中断请求信号，每来回拨动一次开关 K，产生一次中断，数码管显示中断的次数，设不超过 15 次。

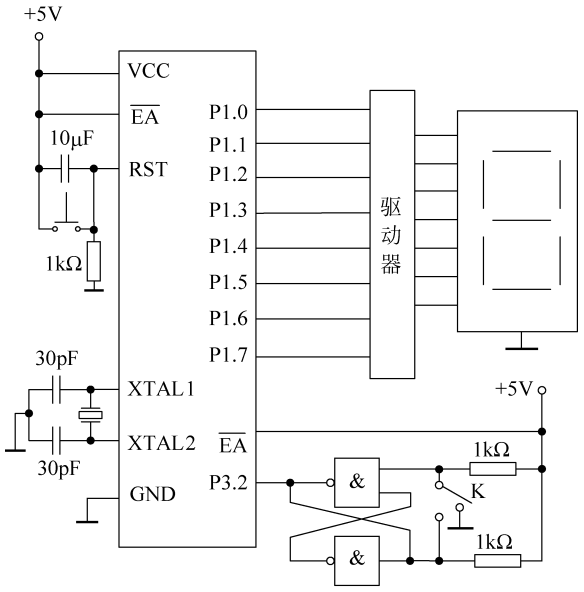

图 5.6　中断次数的显示

解：

(1) C 程序。

```
    #include <reg52.h>
    char i;
    code char tab[16] = {0x3f,0x06,0x5b,0x4F,0x66,0x6d,0x7d,0x07, 0x7f,0x6f,0x77,0x7c,0x39,
0x5e,0x79,0x71};
    main( )
   {
    EA = 1;
    EX0 = 1;
    IT0 = 1;
    P2 = 0x3f;
    while(1);                                //等待中断
   }
int0( ) interrupt 0                          //中断程序
```

```
  {   i++;
      if(i<16)
      {
      P1=tab[i];
      }
      else
      {
      i=0;
       P1=0x3f;
      }
}
```

（2）汇编程序。

```
      ORG    0000H
      AJMP   MAIN
      ORG    0003H                          ;INT0 中断入口
      AJMP   SER0                           ;转中断服务程序
      ORG    0030H
MAIN: SETB   IT0                            ;边沿触发中断
      SETB   EX0                            ;允许 INT0 中断
      SETB   EA                             ;开中断开关
      MOV    R0,   #0                       ;计数初值为 0
      MOV    A,    #3FH                     ;"0"的字形码送 A
      MOV    DPTR, #TAB                     ;指向字形码表
WAIT: SJMP   WAIT                           ;等待中断
SER0: INC    R0                             ;中断次数加 1
      MOV    A,   R0
      MOVC   A,   @A+DPTR                   ;查字形码表
      MOV    P1,   A                        ;显示
      CJNE   R0,    #0FH,   RE              ;15 次中断未到转 RE
      MOV    R0,   #0                       ;15 次到重新开始
 RE: RETI
TAB: DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,
    DB 7FH,6FH,77H,7CH,39H,5EH,79H,71H
END
```

本 章 小 结

（1）51 系列单片机有 5 个中断源，每个中断源有对应固定的中断入口地址，分别为 0003H、000BH、0013H、001BH、0023H，每个中断向量地址间隔 8B。

（2）在软件上通过特殊功能寄存器实现对中断模块的管理和使用，用于单片机中断管理的寄存器有中断标志寄存器 TCON、中断允许寄存器 IE、中断优先级寄存器 IP。

（3）本章主要介绍了两个外部中断 INT0 和 INT1，两个外部中断对应有两个外部引脚：P3.2 和 P3.3；两个外部中断标志位：IE0、IE1；6 个外部中断控制位：触发方式控制位 IT0、IT1，中断使能控制位 EX0、EX1，优先级设置位 PX0、PX1。

（4）掌握对中断过程的理解。可以通过 C 语言或汇编语言编写中断程序，采用 C 语言

注意对应的中断号，汇编语言注意中断服务程序要放到对应入口地址处；此外也可根据标志位采用查询方式编程，但中断编程效率更高。

本章习题

1. 什么是中断？采用中断方式编程的好处有哪些？

2. MCS-51 系列单片机有几个中断源？各中断入口地址分别是多少？

3. MCS-51 系列单片机外部中断有几个？对应标志位是什么？外部中断触发方式如何修改？

4. MCS-51 系列单片机的各中断源有几个优先级？如何设置优先级？如果两个以上中断优先级相同，单片机如何处理其查询顺序？

5. 参照图 5.5，让 8051 单片机的 P1 口接 8 个发光二极管，由外部中断引脚 P3.2 接一消抖开关，实现每中断一次，各发光管状态取反，分别采用汇编和 C 语言编制程序。

6. 在图 5.6 电路基础上，要求实现每中断一次，8 个发光二极管亮、灭变换 3 次，编写程序。