模块 5　80C51 单片机中断系统分析及应用

技能目标

（1）掌握任务 22-1：用定时器 T1 中断方式控制 P3 口 8 位 LED 闪烁。

（2）掌握任务 23-1：用外中断$\overline{\text{INT1}}$控制 P2 口 8 个 LED 亮灭。

（3）掌握任务 23-2：外部中断$\overline{\text{INT0}}$控制 LED 灯左循环亮。

（4）掌握项目 24：用外中断$\overline{\text{INT1}}$测量负跳变信号累计数，并将结果送 P2 口显示。

（5）掌握项目 25：用外中断$\overline{\text{INT0}}$测量外部负脉冲宽度，并将结果送 P1 口显示。

（6）掌握项目 26：基于 AT89C51 单片机交通灯控制器的设计。

知识目标

学习目的：

（1）了解中断的概念和中断的功能。

（2）掌握 80C51 中断系统结构、处理过程和使用方法。

（3）掌握 80C51 中断各寄存器的设置。

（4）掌握 80C51 中断应用的程序设计及仿真。

（5）了解外部中断源的扩展方法。

学习重点和难点：

（1）中断系统结构、处理过程和使用方法。

（2）中断的综合应用。

（3）外部中断源的扩展方法。

概要资源

1-1 学习要求

1-2 重点与难点

1-3 学习指导

1-4 学习情境设计

1-5 教学设计

1-6 评价考核

1-7 PPT

项目22：认识80C51中断系统

● 技能目标

掌握任务22-1：用定时器T1中断方式控制P3口8位LED闪烁。

● 知识目标

学习目的：

(1) 了解中断的概念。

(2) 了解中断的特点及功能。

(3) 掌握80C51中断系统的结构。

(4) 掌握80C51中断各寄存器的设置。

(5) 掌握80C51单片机5个中断源。

学习重点和难点：

(1) 80C51中断系统的结构。

(2) 80C51单片机5个中断源。

任务22-1：用定时器T1中断方式控制P3口8位LED闪烁

1. 任务要求

(1) 了解中断概念。

(2) 掌握TCON寄存器中断的设置。

(3) 掌握定时器/计数器中断编程方法及仿真。

微课5-1

2. 任务描述

使用定时器/计数器T1工作于方式1，采用中断方式控制P3口8位LED的闪烁，其闪烁周期为100ms，即亮50ms，熄灭50ms，电路图如图5-1所示，设单片机晶振频率为12MHz。

3. 任务实现

1) 分析

用定时器T1、方式1，则TMOD=0001××××B=10H。由于是采用中断方式，T1作为中断源，根据80C51中断系统的结构图5-2，则EA=1，ET1=1。

由于$T_{机}$=12T时钟=12×1/f_{osc}=1μs，而方式1的最大定时时间为65.536ms，所以可选择50ms。所以，定时器初始值为：

```
TH1 = (65536 - 50000)/256;          //定时器 T1 的高 8 位赋初值
TL1 = (65536 - 50000) % 256;        //定时器 T1 的低 8 位赋初值
```

2) 程序设计

先建立文件夹XM22-1，然后建立XM22-1工程项目，最后建立源程序文件XM22-1.c，输入如下源程序：

```
#include<reg51.h>                    //包含 51 单片机寄存器定义的头文件
/*****************************************************************
函数功能: 主函数。
*****************************************************************/
void main(void)
{
  EA = 1;                           //开总中断
  ET1 = 1;                          //定时器 T1 中断允许
  TMOD = 0x10;                      //使用定时器 T1 的方式 1
 TH1 = (65536 - 50000)/256;         //定时器 T1 的高 8 位赋初值
  TL1 = (65536 - 50000) % 256;      //定时器 T1 的低 8 位赋初值
  TR1 = 1;                          //启动定时器 T1
   while(1)                         //无限循环,等待中断
      ;                             //空操作
  }
/*****************************************************************
函数功能: 定时器 T1 的中断服务程序。
*****************************************************************/
void Time1(void) interrupt3 using 0   //"interrupt"声明函数为中断服务函数
                      //其后的 3 为定时器 T1 的中断编号; 0 表示使用第 0 组工作寄存器
 {
  P3 = ~P3;                         //按位取反操作,将 P3 引脚输出电平取反
   TH1 = (65536 - 50000)/256;       //定时器 T1 的高 8 位重新赋初值
   TL1 = (65536 - 50000) % 256;     //定时器 T1 的低 8 位重新赋初值
 }
```

3) 用 Proteus 软件仿真

经过 Keil 软件编译通过后,在 Proteus ISIS 编辑环境中绘制仿真电路图,将编译好的 XM22-1.hex 文件加载到 AT89C51 里,然后启动仿真,就可以看到用定时器 T1 中断方式控制 P3 口 8 位 LED 闪烁,效果图如图 5-1 所示。

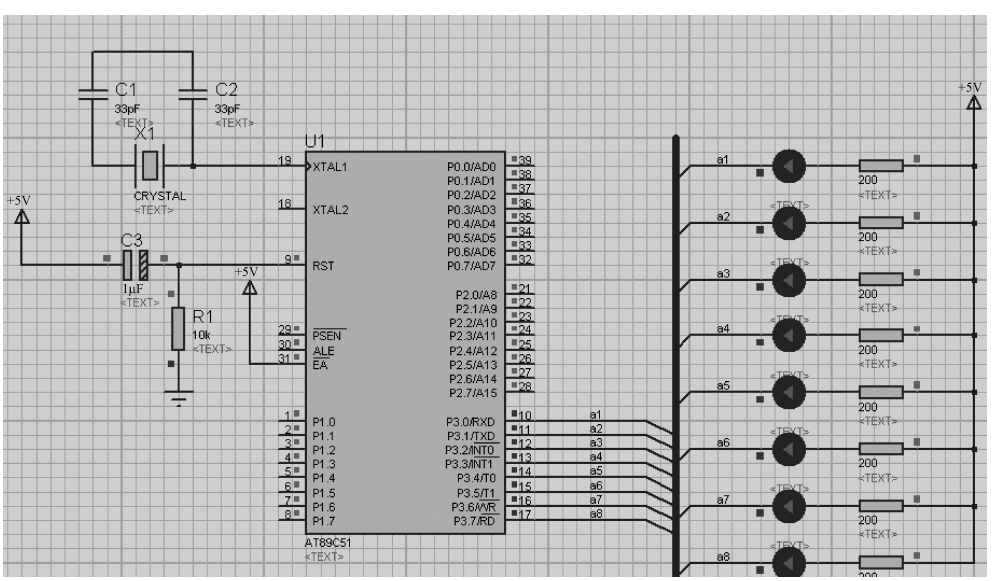

图 5-1 用定时器 T1 中断方式控制 P3 口 8 位 LED 闪烁效果图

任务22-2：相关知识

1. 中断的概念

中断是通过硬件来改变CPU的运行方向的。当CPU在执行程序时，由内部或外部的原因引起的随机事件要求CPU暂时停止正在执行的程序，而转向执行一个用于处理该随机事件的程序，处理完后又返回被中止的程序断点处继续执行，这一过程称为中断。

中断之后执行的相应的处理程序通常称为中断服务或中断处理子程序，原来正常运行的程序称为主程序。主程序被断开的位置(或地址)称为“断点”。引起中断的原因，或能发出中断申请的来源，称为“中断源”。中断源要求服务的请求称为“中断请求”(或中断申请)。

2. 中断的特点及功能

1) 中断的特点

(1) 提高CPU的效率。

中断可以解决高速的CPU与低速的外设之间的矛盾，使CPU和外设并行工作。CPU在启动外设工作后继续执行主程序，同时外设也在工作。每当外设做完一件事，就发出中断申请，请求CPU中断它正在执行的程序，转去执行中断服务程序，中断处理完之后，CPU恢复执行主程序，外设也继续工作。这样，CPU可启动多个外设同时工作，大大提高了CPU的效率。

(2) 实时处理。

在实时控制系统中，工业现场的各种参数和信息都会随时间而变化。这些外界变量可根据要求随时向CPU发出中断申请，请求CPU及时处理中断请求。如果满足中断条件，CPU就立刻响应，转入中断处理，从而实现实时处理。

(3) 故障处理。

控制系统的故障和紧急情况难以预料的，如掉电、设备运行出错等，可通过中断系统由故障源向CPU发出中断请求，再由CPU转到相应的故障处理程序进行处理。

2) 中断的功能

(1) 实现中断响应和中断返回。

当CPU收到中断请求后，CPU要根据相关条件(如中断优先级、是否允许中断)进行判断，决定是否响应这个中断请求。若响应，则在执行完当前指令后立刻响应这一中断请求。CPU中断过程为：第一步将断点处的PC值(即下一条应执行指令的地址)压入堆栈保留下来(这称为保护断点，由硬件自动执行)。第二步将有关的寄存器内容和标志PSW状态压入堆栈保留下来(这称为保护现场，由用户自己编程完成)。第三步执行中断服务程序。第四步中断返回，CPU继续执行原主程序。中断返回过程为：首先，恢复原保留寄存器的内容和标志位的状态，这称为恢复现场，由用户编程完成。然后，再加返回指令RETI，RETI指令的功能是恢复PC值，使CPU返回断点，这称为恢复断点。

(2) 实现优先权排队。

中断优先权也称为中断优先级。中断系统中存在着多个中断源，同一时刻可能会有不止一个中断源提出中断请求，因此需要给所有中断源安排不同的优先级别。CPU可通过中断优先级排队电路首先响应中断优先级高的中断请求，等到处理完优先级高的中断请求后，再来响应优先级低的中断请求。

(3) 实现中断嵌套。

当 CPU 响应某一中断时，若有中断优先级更高的中断源发出中断请求，CPU 会暂停正在执行的中断服务程序，并保留这个程序的断点，转向执行中断优先级更高的中断源的中断服务程序，等处理完这个高优先级的中断请求后，再返回继续执行被暂停的中断服务程序。这个过程称为中断嵌套。

80C51 中断可实现两级中断嵌套。高优先级中断源可中断正在执行的低优先级中断服务程序，除非执行了低优先级中断服务程序的 CPU 关中断指令。同级或低优先级的中断不能中断正在执行的中断服务程序。

3. 80C51 中断系统的结构及中断源

微课 5-2

1) 80C51 中断系统的结构

80C51 的中断系统有 5 个中断源，2 个优先级，可实现二级中断嵌套。80C51 中断系统的结构图如图 5-2 所示。由图 5-2 可知，与中断有关的有：①4 个寄存器，分别为中断源寄存器 TCON 和 SCON、中断允许控制寄存器 IE 和中断优先级控制寄存器 IP；②5 个中断源，分别为外部中断 0 请求($\overline{\text{INT0}}$)、外部中断 1 请求($\overline{\text{INT1}}$)、定时器 0 溢出中断请求 TF0、定时器 1 溢出中断请求 TF1 和串行中断请求 RI 或 TI；③中断优先级控制寄存器 IP 和顺序查询逻辑电路，共同决定 5 个中断源的排列顺序。5 个中断源分别对应 5 个固定的中断入口地址。

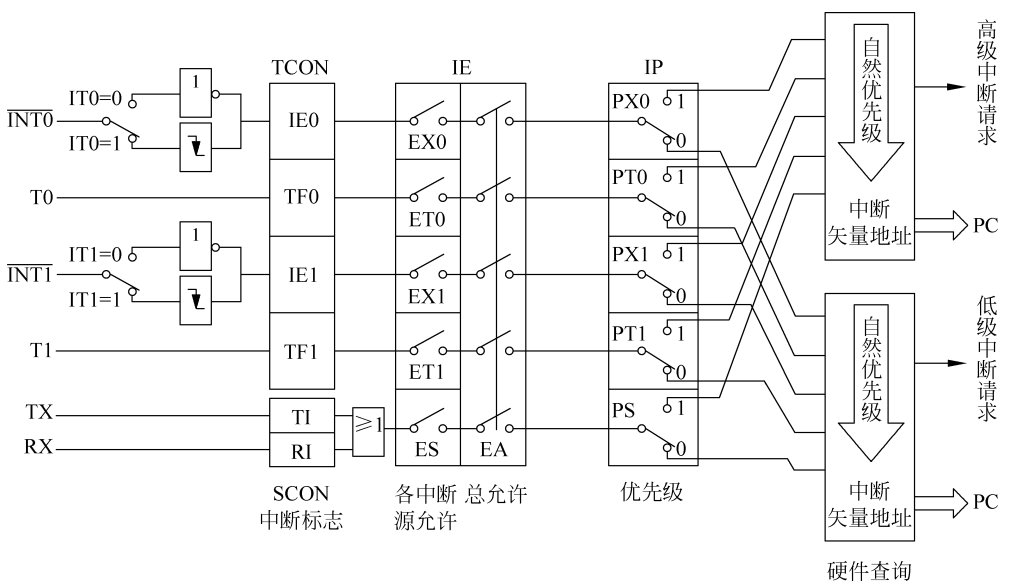

图 5-2　80C51 中断系统的结构图

2) 80C51 的中断源

80C51 有以下 5 个中断源。

(1) $\overline{\text{INT0}}$：外部中断 0 请求，由 80C51 的 P3.2 引脚输入。可由 IT0(TCON.0 位)选择其为低电平有效，还是下降沿有效。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 位)置 1，向 CPU 申请中断。中断服务程序的入口地址为 0003H。

(2) $\overline{\text{INT1}}$：外部中断1请求，由80C51的P3.3引脚输入。可由IT1(TCON.2位)选择其为低电平有效，还是下降沿有效。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位)置1，向CPU申请中断。中断服务程序的入口地址为0013H。

(3) TF0：定时器T0溢出中断请求。当定时器T0产生溢出时，定时器T0中断请求标志位(TCON.5位)置1(由硬件自动执行)，向CPU申请中断。中断服务程序的入口地址为000BH。

(4) TF1：定时器T1溢出中断请求。当定时器T1产生溢出时，定时器T1中断请求标志位(TCON.7位)置1(由硬件自动执行)，向CPU申请中断。中断服务程序的入口地址为001BH。

(5) RI或TI：串行中断请求。当串行口接收完一帧串行数据时置位RI(SCON.0位)(由硬件自动执行)，或当串行口发送完一帧串行数据时置位TI(SCON.1位)，向CPU申请中断。中断服务程序的入口地址为0023H。

项目23：认识80C51中断控制器

● 技能目标

(1) 掌握任务23-1：用外中断$\overline{\text{INT1}}$控制P2口8个LED亮灭。

(2) 掌握任务23-2：外部中断$\overline{\text{INT0}}$控制LED灯左循环亮。

● 知识目标

学习目的：

(1) 掌握定时器控制寄存器TCON设置。

(2) 掌握串行口控制寄存器SCON设置。

(3) 掌握中断允许寄存器IE设置。

(4) 掌握中断优先级寄存器IP设置。

(5) 了解中断优先级别。

(6) 了解80C51中断处理过程。

(7) 掌握80C51外部中断扩展。

(8) 掌握中断系统的应用。

学习重点和难点：

(1) 定时器控制寄存器TCON设置。

(2) 串行口控制寄存器SCON设置。

(3) 中断允许寄存器IE设置。

(4) 中断优先级寄存器IP设置。

(5) 80C51外部中断扩展。

(6) 中断系统的应用。

任务 23-1：用外中断$\overline{\text{INT1}}$控制 P2 口 8 个 LED 亮灭

微课 5-3

1. 任务要求

（1）了解外中断$\overline{\text{INT0}}$、$\overline{\text{INT1}}$应用。

（2）掌握 TCON 寄存器的设置。

（3）掌握 IE 寄存器的设置。

（4）掌握$\overline{\text{INT0}}$、$\overline{\text{INT1}}$中断编程方法。

2. 任务描述

在 P3.3 引脚($\overline{\text{INT1}}$)上接按键 S，使用外中断$\overline{\text{INT1}}$控制 P2 口 8 个 LED 亮灭。当第一次按下按键 S 时，P2 口 8 个 LED 亮，再次按下 S 按键，P2 口 8 个 LED 熄灭，如此循环，就可看见 LED 灯的亮、灭两种状态，电路图如图 5-3 所示。

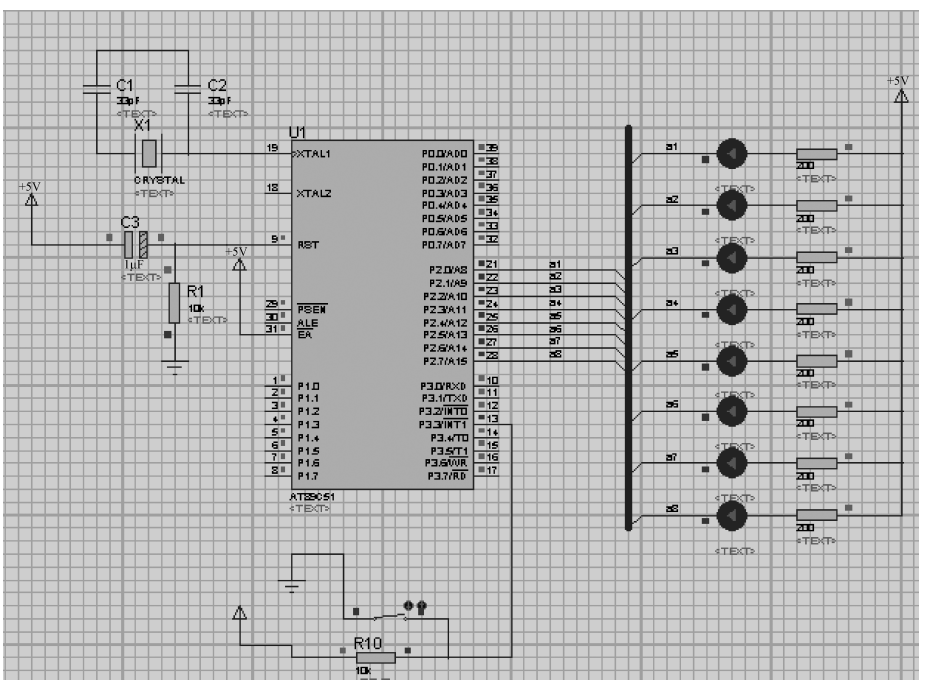

图 5-3　用外中断$\overline{\text{INT1}}$控制 P2 口 8 个 LED 亮、灭效果图

3. 任务实现

1）分析

设置寄存器 TCON 的 IT1＝1，选择负跳变来触发外中断，IE 设置即 EA＝1（开总中断）、EX1＝1（允许外中断 1 中断）。当按键 S 按下时，$\overline{\text{INT1}}$引脚为低电平，外中断$\overline{\text{INT1}}$产生中断请求，在执行外中断服务程序时，P2 口按位取反，就可达到控制 LED 亮、灭。

2）程序设计

先建立文件夹 XM23-1，然后建立 XM23-1 工程项目，最后建立源程序文件 XM23-1.c，输入如下源程序：

```
#include<reg51.h>                    //包含 51 单片机寄存器定义的头文件
```

```
void main(void)
{
  EA = 1;                              //开总中断
  EX1 = 1;                             //允许外中断 1 中断
  IT1 = 1;                             //负跳变来触发外中断
  P2 = 0xff;                           //使 8 个 LED 熄灭
     while(1)                          //无限循环,等待中断
     ;                                 //空操作
}
/ ************************************************************
函数功能: 外中断 INT1 的中断服务程序。
 ************************************************************ /
void int1(void) interrupt 2 using 0   //"interrupt"声明函数为中断服务函数
           //其后的 2 为外中断 INT1 的中断编号; 0 表示使用第 0 组工作寄存器
{
   P2 = ~P2;                           //每产生一次中断请求,P2 按位取反操作一次
}
```

3) 用 Proteus 软件仿真

经过 Keil 软件编译通过后,在 Proteus ISIS 编辑环境中绘制仿真电路图,将编译好的 XM23-1.hex 文件加载到 AT89C51 里,然后启动仿真,就可以看到用外中断$\overline{\text{INT1}}$控制 P2 口 8 个 LED 亮、灭,效果图如图 5-3 所示。

任务 23-2:外部中断$\overline{\text{INT0}}$控制 LED 灯左循环亮

微课 5-4

1. 任务要求

(1) 掌握 TCON 寄存器的设置。

(2) 掌握 IE 寄存器的设置。

(3) 掌握$\overline{\text{INT0}}$、$\overline{\text{INT1}}$中断编程方法。

2. 任务描述

在 P3.2 引脚($\overline{\text{INT0}}$)上接按键 S,使用外中断$\overline{\text{INT0}}$控制 P0 口 8 个 LED 灯左循环亮。

3. 任务实现

1) 分析

设置寄存器 TCON 的 IT0=1,选择负跳变来触发外中断,IE 设置即 EA=1(开总中断)、EX0=1(允许外中断 0 中断)。当按键 S 按下时,$\overline{\text{INT0}}$引脚为低电平,外中断$\overline{\text{INT0}}$产生中断请求,在执行外中断服务程序时,使 LED 灯左循环亮。

2) 程序设计

先建立文件夹 XM23-2,然后建立 XM23-2 工程项目,最后建立源程序文件 XM23-2.c,输入如下源程序:

```
#include<reg51.h>                      //包含 51 单片机寄存器定义的头文件
  sbit S = P3^2;                       //位操作,将 S 定义为 P3.2 引脚
unsigned char Count;                   //设置全局变量,存储中断次数
/ ************************************************************************
函数功能: 主函数。
 ************************************************************************ /
   void main(void)
   {
     EA = 1;                           //开放总中断
```

```
    EX0 = 1;                        //允许使用外中断 0
     IT0 = 1;                       //选择负跳变来触发外中断
     P0 = 0x01;                     //使 P0.0 输出高电平,经过反相器后使 LED0 亮
     Count = 0;                     //从 0 开始累计中断次数
     while(1)
      ;                             //无限循环,防止程序跑飞
    }
/ ************************************************************************
函数功能: 外部中断INT0的中断服务程序。
************************************************************************ /
    void int0(void) interrupt0 using 0 //外中断 0 的中断编号为 0
    {
       Count++;                     //中断次数自动加 1
       if(Count == 8)               //若中断次数累计满 8 次
         {
           P0 = 0x01;               //重新设置初值
           Count = 0;               //将 Count 清零,重新从 0 开始计数
         }
       else P0 = P0 << 1;           //每产生一次中断,P0 左移一次
    }
```

3) 用 Proteus 软件仿真

经过 Keil 软件编译通过后,在 Proteus ISIS 编辑环境中绘制仿真电路图,将编译好的 XM23-2.hex 文件加载到 AT89C51 里,然后启动仿真,就可以看到用外中断INT0控制 LED 灯左循环亮,效果图如图 5-4 所示。

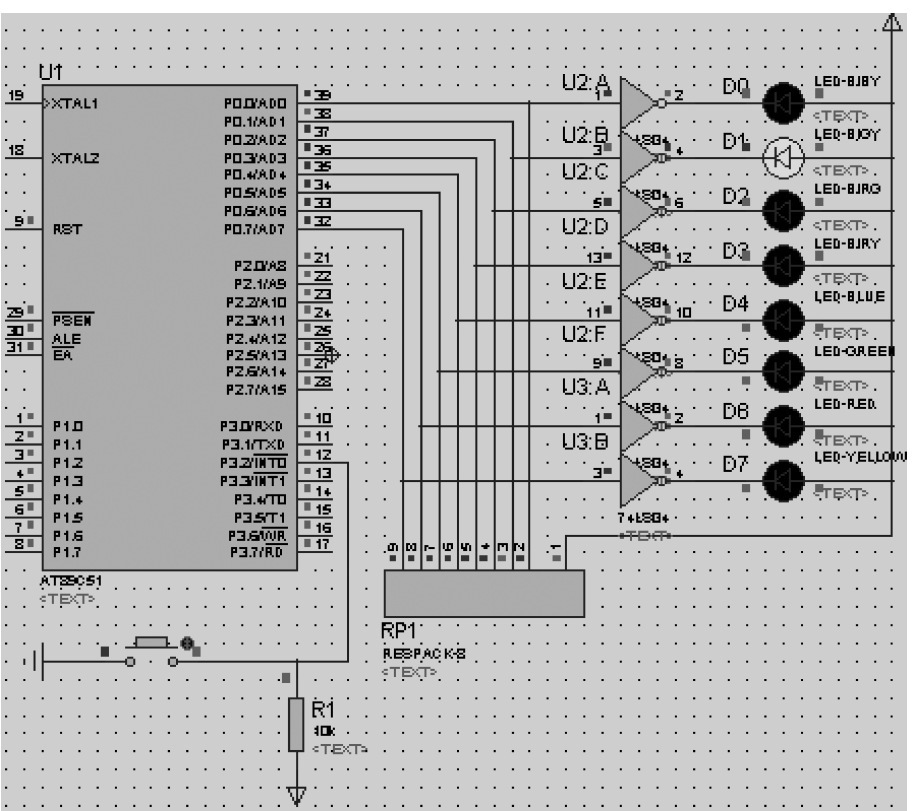

图 5-4　外部中断INT0控制 LED 灯左循环亮仿真效果图

任务 23-3：相关知识

微课 5-5

80C51 中断系统各寄存器的设置如下。

1. 定时器控制寄存器 TCON

定时器控制寄存器 TCON 的作用是控制定时器的启动与停止，并保存 T0、T1 的溢出中断标志和外部中断$\overline{\text{INT0}}$、$\overline{\text{INT1}}$的中断标志。

TCON 寄存器的格式和各位定义如下所示。

	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
TCON (88H)	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
	控制定时器/计数器的启停和中断请求				控制外部中断与定时器/计数器无关			

其中，TF1、TR1、TF0 和 TR0 这 4 位是控制定时器的启动与停止的，在模块 4 的任务 18-2：相关知识中已经详细讨论过，下面只作简单介绍。

(1) TF1(TCON.7 位)：定时器 1 溢出标志位。定时器 1 被启动计数后，从初值开始做加 1 计数，计满溢出后由硬件自动使 TF1 置 1，并申请中断。此标志一直保持到 CPU 响应中断后，才由硬件自动清 0。也可用软件查询该标志，并由软件清 0。

(2) TR1(TCON.6 位)：定时器 1 启停控制位。

(3) TF0(TCON.5 位)：定时器 0 溢出标志位。其功能同 TF1。

(4) TR0(TCON.4 位)：定时器 0 启、停控制位。其功能同 TR1。

以下 4 位直接与中断有关，需进行详细讨论。

(5) IT1(TCON.2 位)：外部中断 1($\overline{\text{INT1}}$)触发方式选择位。

当 IT1＝0 时，外部中断 1 为电平触发方式。在这种方式下，CPU 在每个机器周期的 S5P2 期间对$\overline{\text{INT1}}$(P3.3)引脚采样，若为低电平，则认为有中断申请，硬件自动使 IE1 标志置 1；若为高电平，则认为无中断申请或中断申请已撤除，硬件自动使 IE1 标志清 0。在电平触发方式中，CPU 响应中断后不能由硬件自动使 IE1 清 0，也不能由软件使 IE1 清 0，所以在中断返回前，必须撤销$\overline{\text{INT1}}$引脚上的低电平，否则将再次中断，导致出错。

当 IT1＝1 时，外部中断 1 为边沿触发方式。CPU 在每个机器周期的 S5P2 期间采样$\overline{\text{INT1}}$(P3.3)引脚。若在连续两个机器周期采样到先高电平后低电平，则认为有中断申请，硬件自动使 IE1 置 1，此标志一直保持到 CPU 响应中断时，才由硬件自动清 0。在边沿触发方式下，为保证 CPU 在两个机器周期内检测到先高后低的负跳变，输入高低电平的持续时间至少要保持 12 个时钟周期。

(6) IE1(TCON.3 位)：外部中断 1($\overline{\text{INT1}}$)请求标志位。IE1＝1 表示外部中断 1 向 CPU 申请中断。当 CPU 响应外部中断 1 的中断请求时，由硬件自动使 IE1 清 0(边沿触发方式)。

(7) IE0(TCON.1 位)：外部中断 0($\overline{\text{INT0}}$)请求标志位。其功能同 IE1。

(8) IT0(TCON.0 位)：外部中断 0 触发方式选择位。其功能同 IT1。

80C51 系统复位后，TCON 初值均清 0，应用时要注意各位的初始状态。

2. 串行口控制寄存器 SCON

串行口控制寄存器 SCON 的低 2 位 TI 和 RI 保存串行口的发送中断和接收中断标志。

SCON 寄存器的格式和各位定义如下所示。

							99H	98H
SCON (98H)							TI	RI

(1) TI (SCON.1 位)：串行发送中断请求标志。CPU 将一个字节数据写入发送缓冲器 SBUF 后，就启动发送，每发送完一个串行帧数据，硬件自动使 TI 置 1。但 CPU 响应中断后，硬件并不能自动使 TI 清 0，必须由软件使 TI 清 0。

(2) RI (SCON.0 位)：串行接收中断请求标志。在串行口允许接收时，每接收完一个串行帧数据，硬件自动使 RI 置 1。但 CPU 响应中断后，硬件并不能自动使 RI 清 0，必须由软件使 RI 清 0。

80C51 系统复位后，SCON 初值均清 0，应用时要注意各位的初始状态。

3. 中断允许寄存器 IE

80C51 单片机有 5 个中断源都是可屏蔽中断，其中断系统内部设有一个专用寄存器 IE，用于控制 CPU 对各中断源的开放或屏蔽。IE 寄存器的格式和各位定义如下所示。

	AFH			ACH	ABH	AAH	A9H	A8H
IE (A8H)	EA	—	—	ES	ET1	EX1	ET0	EX0

(1) EA(IE.7 位)：CPU 中断总允许控制位。EA＝1，CPU 开放所有中断。各中断源的允许还是禁止，分别由各中断源的中断允许位单独加以控制；EA＝0，CPU 禁止所有的中断，称为关中断。

(2) ES(IE.4 位)：串行口中断允许位。ES＝1，允许串行口中断；ES＝0，禁止串行口中断。

(3) ET1(IE.3 位)：定时器 1 中断允许位。ET1＝1，允许定时器 1 中断；ET1＝0，禁止定时器 1 中断。

(4) EX1(IE.2 位)：外部中断 1($\overline{\text{INT1}}$)中断允许位。EX1＝1，允许外部中断 1 中断；EX1＝0，禁止外部中断 1 中断。

(5) ET0(IE.1 位)：定时器 0 中断允许位。ET0＝1，允许定时器 0 中断；ET0＝0，禁止定时器 0 中断。

(6) EX0(IE.0 位)：外部中断 0($\overline{\text{INT0}}$)中断允许位。EX0＝1，允许外部中断 0 中断；EX0＝0，禁止外部中断 0 中断。

80C51 单片机系统复位后，IE 中各中断允许位均被清 0，即禁止所有中断。

开中断过程是：首先开总中断“SETB　EA”，然后，开 T1 中断“SETB　ET1”，这两条位操作指令也可合并为 1 条字节指令“MOV　IE，＃88H”。

4. 中断优先级寄存器 IP

80C51 单片机有两个中断优先级，每个中断源都可以通过编程确定为高优先级中断或低优先级中断，因此，可实现二级嵌套。同一优先级别中的中断源可能不止一个，也有中断优先权排队的问题。专用寄存器 IP 为中断优先级寄存器，锁存各中断源优先级控制位，IP 中的每一位均可由软件来置 1 或清 0，且 1 表示高优先级，0 表示低优先级。IP 寄存器的格式和各位定义如下所示。

				BCH	BBH	BAH	B9H	B8H
IP (B8H)	—	—	—	PS	PT1	PX1	PT0	PX0

(1) PS(IP.4位)：串行口中断优先级控制位。PS=1,串行口为高优先级中断；PS=0,串行口为低优先级中断。

(2) PT1(IP.3位)：定时器1中断优先级控制位。PT1=1,定时器1为高优先级中断；PT1=0,定时器1为低优先级中断。

(3) PX1(IP.2位)：外部中断1($\overline{INT1}$)中断优先级控制位。PX1=1,外部中断1为高优先级中断；PX1=0,外部中断1为低优先级中断。

(4) PT0(IP.1位)：定时器0中断优先级控制位。PT0=1,定时器0为高优先级中断；PT0=0,定时器0为低优先级中断。

(5) PX0(IP.0位)：外部中断0($\overline{INT0}$)中断优先级控制位。PX0=1,外部中断0为高优先级中断；PX0=0,外部中断0为低优先级中断。

5. 中断优先级别

当80C51系统复位后,IP低5位全部清0,所有中断源均设定为低优先级中断。

如果几个同一优先级的中断源同时向CPU申请中断,CPU通过内部硬件查询逻辑,按自然优先级顺序确定先响应哪个中断请求。自然优先级由硬件形成,排列见表5-1。

表5-1 80C51单片机中断源的自然优先级、入口地址及中断编号

中 断 源	自然优先级	中断入口地址	C51编译器对中断的编号
外部中断0	高	0003H	0
定时器T0溢出中断		000BH	1
外部中断1	↓	0013H	2
定时器T1溢出中断	低	001BH	3
串口通信中断R1或T1		0023H	4

80C51有5个中断源,应有相应的中断服务程序,这些中断服务程序有规定的存放位置,如表5-1中的中断入口地址,当有中断请求后,CPU可以根据入口地址找到中断服务程序并执行,大大提高了执行效率。

为了方便用C语言编写中断程序,C51编译器也支持80C51单片机的中断服务程序,而且用C语言编写中断服务程序,比用汇编语言方便很多。C语言编写中断服务函数的格式为：

```
函数类型  函数名(形式参数列表)[interrupt n][using m]
```

其中：interrupt后面的n是中断编号,取值为0～4；using中的m表示使用的工作寄存器组号(如不声明,则默认用第0组)。

6. 80C51中断处理过程

中断处理过程可分为3个阶段,即中断响应、中断处理和中断返回。不同的计算机因其中断系统的硬件结构不同,因此,中断响应的方式也有所不同。这里仅以80C51单片机为例进行介绍。

1) 中断响应

中断响应是CPU对中断源中断请求的响应,包括保护断点和将程序转向中断服务程

序的入口地址。CPU 并不是任何时刻都响应中断请求，而是在中断响应条件满足之后才会响应。CPU 响应中断必须首先满足以下 3 个基本条件。

(1) 中断源要有中断请求。

(2) 中断总允许位 EA＝1。

(3) 中断源的中断允许位为 1。

在满足以上条件的基础上，CPU 一般会响应中断，但若有下列任何一种情况存在，中断响应都会受到阻碍。

(1) CPU 正在响应同级或高优先级的中断服务程序。

(2) 当前执行的指令尚未执行完。

(3) 正在执行指令 RET、RETI 或任何对专用寄存器 IE、IP 进行读/写的指令。CPU 在执行完上述指令之后，要再执行一条指令，才能响应中断请求。

若由于上述条件的阻碍，中断未能得到响应，当条件消失时，该中断标志却已不再有效，那么该中断将不被响应。

2) 中断响应时间

在控制系统中，为了满足控制精度和时间要求，需要弄清 CPU 响应中断所需的时间。响应中断的时间分为最短时间(需要 3 个机器周期)和最长时间(需要 8 个机器周期)。

3) 中断处理

中断处理就是执行中断服务程序。中断服务程序从中断入口地址开始执行，到返回指令 RETI 为止。此过程一般包括三部分内容：一是保护现场；二是处理中断源的请求；三是恢复现场。通常，主程序和中断服务程序都会用到累加器 A、状态寄存器 PSW 及其他一些寄存器。执行中断服务程序时，CPU 若用到上述寄存器，就会破坏原先存在这些寄存器中的内容，一旦中断返回，将会造成主程序混乱。因此，进入中断服务程序后，一般要先保护现场，然后再执行中断处理程序，在中断返回主程序之前，再恢复现场。

7. 80C51 外部中断扩展

80C51 单片机只有两个外部中断请求输入端$\overline{\text{INT0}}$和$\overline{\text{INT1}}$，实际应用中，若外部中断源超过两个，就不够用了，因此需扩充外部中断源，这里介绍两种简单的方法，即定时器扩展法和中断加查询扩展法。定时器扩展法用于外部中断源个数不太多并且定时器有空余的场合。中断加查询扩展法用于外部中断源个数较多的场合，但因查询时间较长，在实时控制中要注意能否满足实时控制要求。

8. 中断系统的应用

中断系统的初始化实质上是针对 4 个与中断有关的特殊功能寄存器 TCON、SCON、IE 和 IP 进行控制和管理的，具体步骤如下所示。

(1) 开 CPU 中断总开关(EA)。

(2) 设置中断允许寄存器 IE 中相应的位，确定各个中断源是否允许中断。

(3) 对多级中断设置中断优先级寄存器 IP 中相应的位，确定各中断源的优先级别。

(4) 设置定时器控制寄存器 TCON 中相应的位，确定外部中断的触发方式是边沿触发，还是电平触发。

● 技能目标

(1) 掌握项目 24：用外中断$\overline{\text{INT1}}$测量负跳变信号累计数，并将结果送 P2 口显示。

(2) 掌握项目 25：用外中断$\overline{\text{INT0}}$测量外部负脉冲宽度，并将结果送 P1 口显示。

(3) 掌握项目 26：基于 AT89C51 单片机交通灯控制器的设计。

* 项目 24：用外中断$\overline{\text{INT1}}$测量负跳变信号累计数，并将结果送 P2 口显示

微课 5-6

1. 项目要求

(1) 掌握 TCON 寄存器的设置。

(2) 掌握 IE 寄存器的设置。

(3) 掌握$\overline{\text{INT0}}$、$\overline{\text{INT1}}$中断编程方法。

2. 项目描述

使用外中断$\overline{\text{INT1}}$(P3.3 引脚)测量从 P3.7 引脚输出负跳变信号累计数，并将结果送 P2 口显示，电路图如图 5-5 所示。

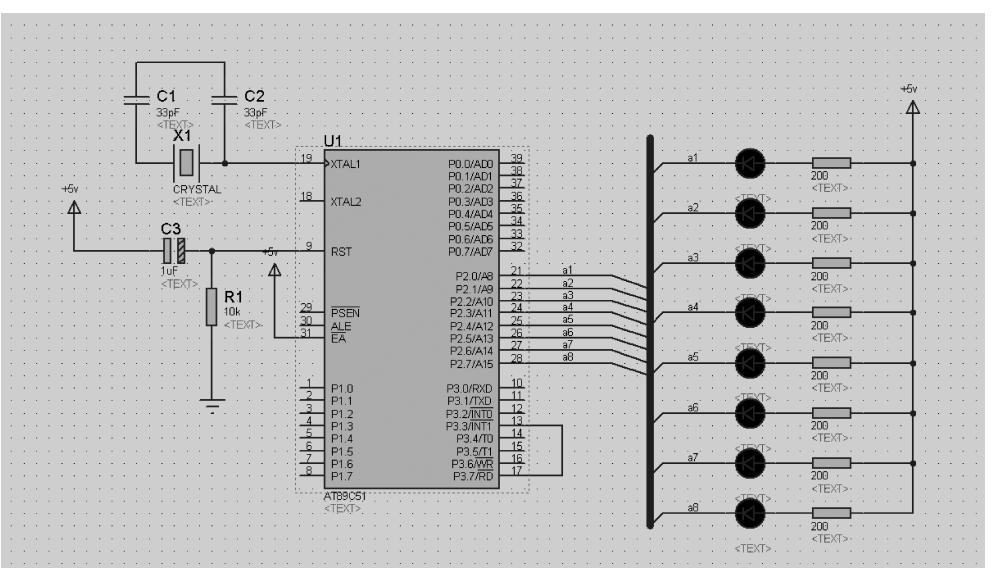

图 5-5　用外中断$\overline{\text{INT1}}$测量负跳变信号累计数，并将结果送 P2 口显示效果图

3. 项目实现

1) 分析

设置寄存器 TCON 的 IT1＝1，选择负跳变触发外中断，IE 设置即 EA＝1(开总中断)、EX1＝1(允许外中断 1 中断)。负跳变由软件控制 P3.7 引脚输出电平产生。负跳变信号累计由外中断$\overline{\text{INT1}}$的中断服务程序完成，每产生一次中断，计数变量加 1，将结果送 P2 口显示。

2）程序设计

先建立文件夹 XM24，然后建立 XM24 工程项目，最后建立源程序文件 XM24.c，输入如下源程序：

```
#include<reg51.h>                    //包含 51 单片机寄存器定义的头文件
sbit M=P3^7;                         //变量 M 定义为 P3.7，从该引脚输出负跳变脉冲
unsigned char Countor;               //设置全局变量，负跳变信号累计数
/*****************************************************************
函数功能：延时约 60ms(3×200×100=60000μs=60ms)。
*****************************************************************/
void delay60ms(void)
{
  unsigned char m,n;
  for(m=0;m<200;m++)
  for(n=0;n<100;n++)
       ;
}
/*****************************************************************
函数功能：延时约 30ms(3×100×100=30000μs=30ms)。
*****************************************************************/
void delay30ms(void)
{
  unsigned char m,n;
  for(m=0;m<100;m++)
  for(n=0;n<100;n++)
       ;
}
/*****************************************************************
函数功能：主函数。
*****************************************************************/
void main(void)
{
unsigned char i;
  EA=1;                              //开总中断
  EX1=1;                             //允许外中断 1 中断
  IT1=1;                             //负跳变来触发外中断
  Countor=0;                         //计数变量清 0
  for(i=0;i<100;i++);                //输出 100 个负跳变
     {
       M=1;                          //P3.7 引脚输出高电平
       delay60ms();
       M=0;                          //P3.7 引脚输出低电平
       delay30ms();
     }
  While(1)
   ;                                 //无限循环
}
/*****************************************************************
```

```
函数功能: 外中断INT1的中断服务程序。
*************************************************************/
void int1(void) interrupt2 using 0    //"interrupt"声明函数为中断服务函数
          //其后的 2 为外中断INT1的中断编号; 0 表示使用第 0 组工作寄存器
{
Countor++;                            //每产生一次外中断,计数变量加 1
P2 = Countor;                         //计数结果送 P2 口显示
}
```

3) 用 Proteus 软件仿真

经过 Keil 软件编译通过后,在 Proteus ISIS 编辑环境中绘制仿真电路图,将编译好的 XM24.hex 文件加载到 AT89C51 里,然后启动仿真,就可以看到用外中断INT1测量负跳变信号累计数,并将结果送 P2 口显示,效果图如图 5-5 所示。

课堂练习:用定时器 T0 中断方式控制 P1.0 的蜂鸣器发出 1kHz 音频,要求仿真实现。

* 项目 25:用外中断INT0测量外部负脉冲宽度,并将结果送 P1 口显示

微课 5-7

1. 项目要求

(1) 掌握 TCON 寄存器的设置。

(2) 掌握 IE 寄存器的设置。

(3) 掌握INT0、INT1中断编程方法。

2. 项目描述

使用外中断INT0(P3.2 引脚)和定时器 T1 测量外部输入的负脉冲宽度。由单片机 U1(P1.0 引脚)输出方波,再由单片机 U2(P3.2 引脚)接收并检测负脉冲宽度,结果送 P1 口 8 个 LED 显示,电路图如图 5-6 所示。

3. 项目实现

1) 分析

设置寄存器 TCON 的 IT0=1,选择负跳变来触发外中断,IE 设置即 EA=1(开总中断)、EX0=1(允许外中断 0 中断)。负跳变由定时器 T1 控制 P1.0 引脚输出电平产生。

2) 程序设计

(1) 产生负脉冲宽度为 250μs 方波的程序。

先建立一个文件夹 XM25,然后再建立子文件夹 XM25-1 工程项目,最后建立源程序文件 XM25-1.c,输入如下源程序:

```
#include<reg51.h>                     //包含 51 单片机寄存器定义的头文件
sbit M = P1^0;                        //变量 M 定义为 P1.0,从该引脚输出负跳变脉冲
/*************************************************************
函数功能: 主函数。
*************************************************************/
void main(void)
{
  EA = 1;                             //开总中断
```

```
  ET1 = 1;                          //定时器 T1 中断允许
  TMOD = 0x20;                      //使用定时器 T1 的方式 2
  TH1 = 256 - 250;                  //定时器 T1 的高 8 位赋初值
  TL1 = 256 - 250;                  //定时器 T1 的低 8 位赋初值
  TR1 = 1;                          //启动定时器 T1
  while(1)                          //无限循环,等待中断
     ;                              //空操作
}
/************************************************************
函数功能: 定时器 T1 的中断服务程序。
************************************************************/
void Time1(void) interrupt3 using 0   //"interrupt"声明函数为中断服务函数
                        //其后的 3 为定时器 T1 的中断编号; 0 表示使用第 0 组工作寄存器
    {
     M = ~M;                        //将 P1.0 引脚输出电平取反,产生负脉冲宽度为 500μs 的方波
}
```

(2) 测量负脉冲宽度的程序。

在文件夹 XM25 下,建立子文件夹 XM25-2 工程项目,最后建立源程序文件 XM25-2.c,输入如下源程序:

```
#include <reg51.h>                  //包含 51 单片机寄存器定义的头文件
sbit W = P3^2;                      //变量 W 定义为 P3.2
/************************************************************
函数功能: 主函数。
************************************************************/
void main(void)
{
  EA = 1;                           //开总中断
  EX0 = 1;                          //允许外中断 0 中断
  TMOD = 0x20;                      //使用定时器 T1 的方式 2
  IT0 = 1;                          //负跳变来触发外中断
  ET1 = 1;                          //允许定时器 T1 中断
  TH1 = 0;                          //定时器 T1 的高 8 位赋初值设置为 0
  TL1 = 0;                          //定时器 T1 的低 8 位赋初值设置为 0
  TR1 = 0;                          //先关闭定时器 T1
  while(1)                          //无限循环,等待中断
     ;                              //空操作
}

/************************************************************
函数功能: 外中断INT0的中断服务程序。
************************************************************/
  void int0(void) interrupt0 using 0  //"interrupt"声明函数为中断服务函数
                        //其后的 0 为外中断INT0的中断编号; 0 表示使用第 0 组工作寄存器
   {
   TR1 = 1;                         //外中断到来即启动定时器 T1
    TL1 = 0;                        //从 0 开始计时
   while(W == 0)                    //低电平时,等待 T1 计时
    ;                               //空操作
```

```
P1 = TL1;                          //计数结果送 P1 口显示
TR1 = 0;                           //关闭定时器 T1
}
```


3) 用 Proteus 软件仿真

经过 Keil 软件编译通过后,在 Proteus ISIS 编辑环境中绘制仿真电路图,将编译好的 XM25-1.hex、XM25-2.hex 文件分别加载到 U1、U2 单片机 AT89C51 里,然后启动仿真,就可以看到用外中断$\overline{\text{INT0}}$测量外部负脉冲宽度,并将结果送 P1 口显示,效果图如图 5-6 所示。

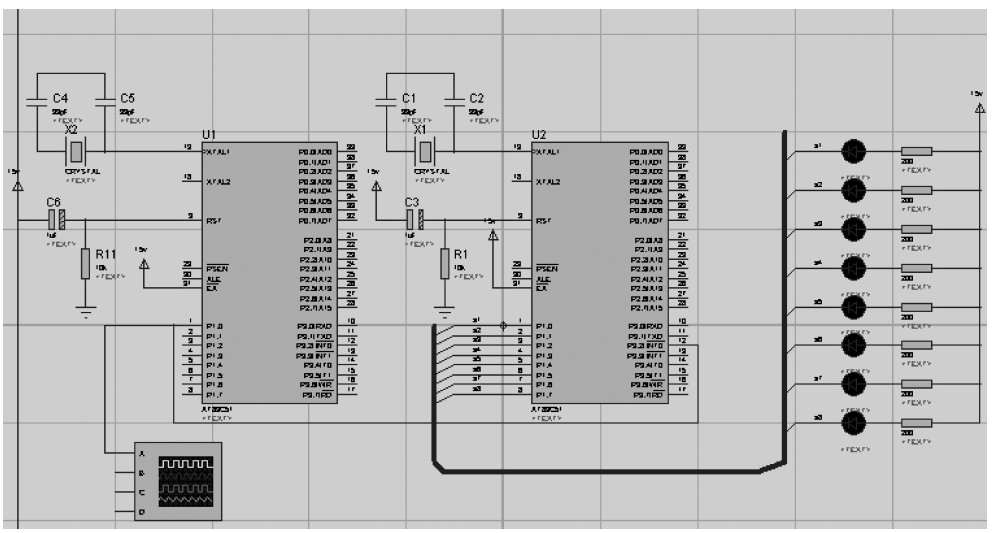

图 5-6　用外中断$\overline{\text{INT0}}$测量外部负脉冲宽度,并将结果送 P1 口显示效果图

课堂练习:(1) 用中断方式实现将 T1 计数的结果送 P0 口显示,要求仿真。

(2) 用中断方式实现单片机控制 LED 灯左循环亮,要求仿真。

* 项目 26:基于 AT89S52 单片机交通灯控制器的设计

1. 项目要求

微课 5-8

用 AT89S52 单片机控制一个交通信号灯系统,晶振频率采用 12MHz。设 A 车道与 B 车道交叉组成十字路口,A 是主道,B 是支道。设计要求如下所示。

(1) 用发光二极管模拟交通信号灯,用按键开关模拟车辆检测信号。

(2) 正常情况下,A、B 两车道轮流放行,A 车道放行 50s,其中 5s 用于警告;B 车道放行 30s,其中 5s 用于警告。

(3) 在交通繁忙时,交通信号灯控制系统应有手控开关,可人为地改变信号灯的状态,以缓解交通拥挤状况。在 B 车道放行期间,若 A 车道有车,而 B 车道无车,按下开关 K1 使 A 车道放行 15s;在 A 车道放行期间,若 B 车道有车,而 A 车道无车,按下开关 K2 使 B 车道放行 15s。

(4) 有紧急车辆通过时,按下 K3 开关使 A、B 车道均为红灯,禁行 20s。

2. 项目描述

交通控制系统主要控制 A 车道、B 车道的交通，以 AT89S52 单片机为核心芯片，通过控制三色 LED 的亮灭来控制各车道的通行；另外，通过 3 个按键来模拟各车道有没车辆的情况和紧急车辆情况。根据设计要求，制定总体设计思想如下所示。

- 正常情况下运行主程序，采用 0.5s 延时子程序的反复调用来实现各种定时时间。
- 一道有车而另一道无车时，采用外部中断 1 执行中断服务程序，并设置该中断为低优先级中断。
- 有紧急车辆通过时，采用外部中断 0 执行中断服务程序，并设置该中断为高优先级中断，实现二级中断嵌套。

交通信号灯模拟控制电路图如图 5-7 所示。

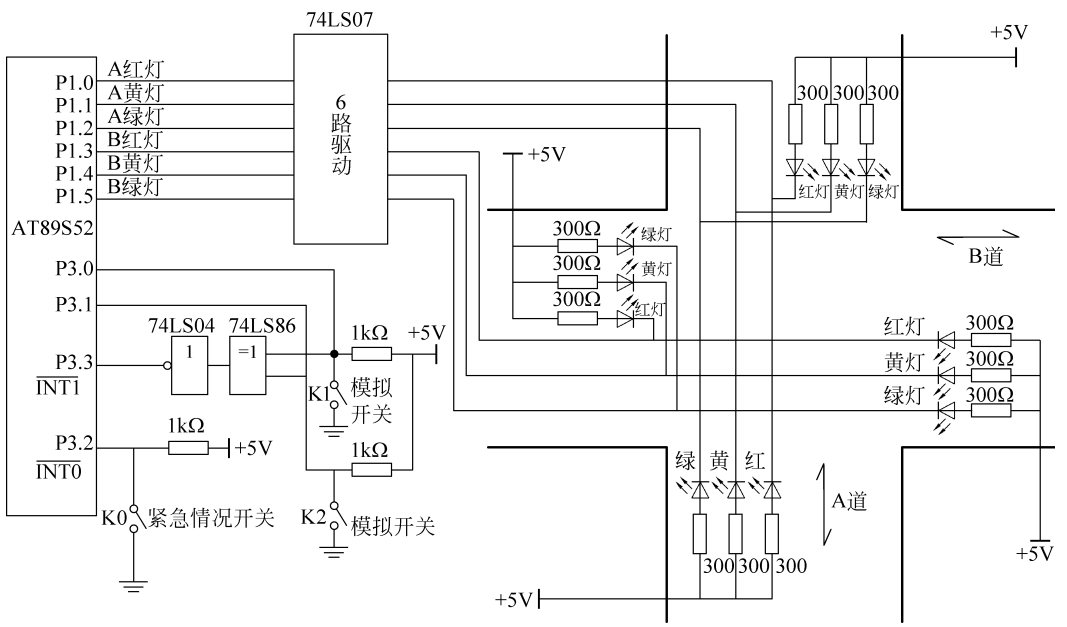

图 5-7　交通信号灯模拟控制电路图

3. 项目实现

1）分析

（1）硬件设计。

用 12 只发光二极管模拟交通信号灯，以 AT89S52 单片机的 P1 口控制这 12 只发光二极管，由于单片机带负载能力有限，因此，在 P1 口与发光二极管之间用 74LS07 作驱动电路，P1 口输出低电平时，信号灯亮；输出高电平时，信号灯灭。在正常情况和交通繁忙时，A、B 两车道的 6 只信号灯的控制状态有 5 种形式，即 P1 口控制功能及相应控制码，见表 5-2。分别以按键 K1、K2 模拟 A、B 道的车辆检测信号，开关 K1 按下时，A 车道放行；开关 K2 按下时，B 车道放行；开关 K1 和 K2 的控制信号经异或取反后，产生中断请求信号（低电平有效），通过外部中断 1 向 CPU 发出中断请求；因此产生外部中断 1 中断的条件应是：$\overline{INT1}=\overline{K1 \oplus K2}$，可用集成块 74LS266（如无 74LS266，可用 74LS86 与 74LS04 组合）来实现。采用中断加查询扩展法，可以判断出要求放行的是 A 车道（按下开关 K1），还是 B 车道（按下开

关K2)。

以按键K0模拟紧急车辆通过开关,当K0为高电平时,属正常情况,当K0为低电平时,属紧急车辆通过的情况,直接将K0信号接至$\overline{\text{INT0}}$(P3.2)脚即可实现外部中断0中断。

表5-2　交通信号灯与控制状态对应关系

控制状态	P1口控制码	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
		未用	未用	B道绿灯	B道黄灯	B道红灯	A道绿灯	A道黄灯	A道红灯
A道放行,B道禁止	F3H	1	1	1	1	0	0	1	1
A道警告,B道禁止	F5H	1	1	1	1	0	1	0	1
A道禁止,B道放行	DEH	1	1	0	1	1	1	1	0
A道禁止,B道警告	EEH	1	1	1	0	1	1	1	0
A道禁止,B道禁止	F6H	1	1	1	1	0	1	1	0

(2) 软件设计。

交通信号灯模拟控制系统程序流程图如图5-8所示。

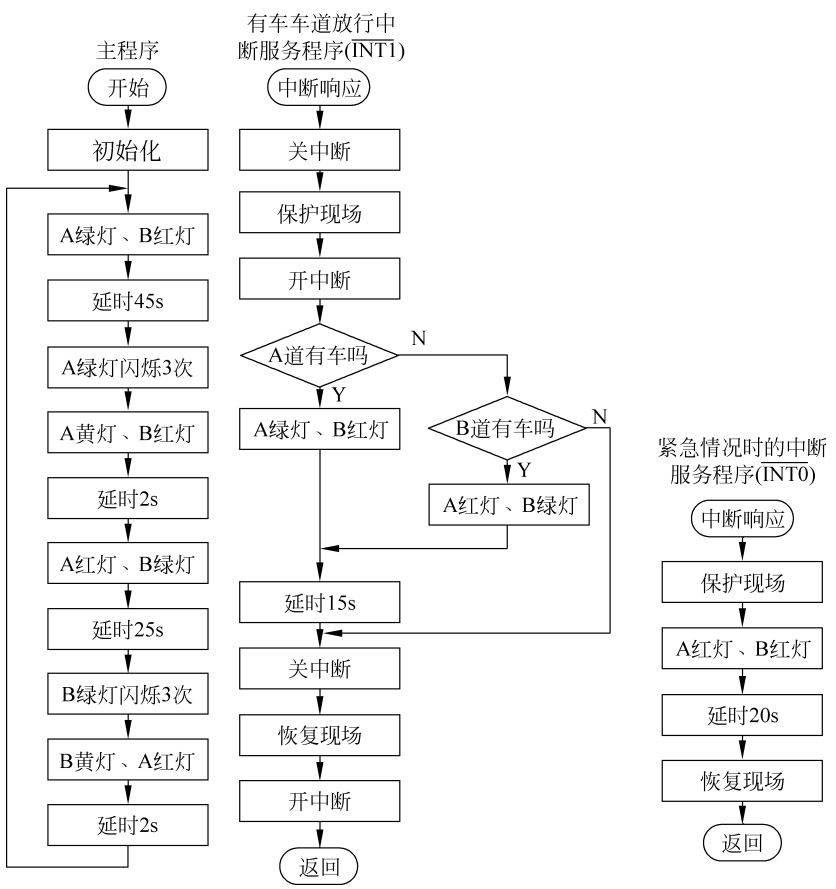

图5-8　交通信号灯模拟控制系统程序流程图

2) 程序设计

先建立一个文件夹XM26,然后再建立子文件夹XM26工程项目,最后建立源程序文件

XM26.c,输入如下源程序：

```
#include<reg51.h>
  sbit P1_2 = P1^2;                    //A道绿灯
  sbit P1_5 = P1^5;                    //B道绿灯
  sbit P3_0 = P3^0;                    //A车道放行
  sbit P3_1 = P3^1;                    //B车道放行
void DELAY()                           //0.5s定时
 {  int n1 = 0;
    TMOD = 0x01;
    TR0 = 1;
    while(1)
    {
    TH0 = (65536 - 50000)/256;
    TL0 = (65536 - 50000) % 256;
    n1++;
    do{
    TF0 = 0;
    }
    while(!TF0);
    if(n1 == 10)
    {n1 = 0;
     break;}
    }
}
void timer0(void) interrupt 0 using 0 //外部中断0入口
{
    int n2;
    for(n2 = 0;n2<40;n2++)             //20s定时
    {
    DELAY();
    P1 = 0xF6;
     }
}
void timer1(void) interrupt 2 using 2 //外部中断1入口
{  int n2,n3;
    if(P3_1 == 0)                      //B车道开关闭合放行
    {
    P3_0 = 1;
     for(n2 = 0;n2<30;n2++)            //15s定时
     {
        DELAY();
        P1 = 0xde;
     }
     }
     if(P3_0 == 0)                     //A车道开关闭合放行15s
   {
    P3_1 = 1;
     for(n3 = 0;n3<30;n3++)
     {
        DELAY();
        P1 = 0xf3;
     }
  }
```

```
}
void main()
{
    int n = 0;
    int i;
    IP = 0x01;                          //外部中断 1 最高优先级
    IE = 0x85;                          //开中断
    TCON = 0x00;                        //置外部中断为电平触发
     while(1)
     {
        P1 = 0xF3;                      //A 道放行,B 道禁止
        DELAY();                        //调延时函数
        n++;
        if(n == 90)                     //45s 计时,90 × 0.5s = 45s
        {
            for(i = 0;i < 6;i++)        //A 道绿灯闪烁 3s,每 0.5s 灭一次,0.5 × 6 = 3s
            {
                DELAY();
                P1_2 = !P1_2;
            }
            for(i = 0;i < 4;i++)        //A 道黄灯亮 2s,0.5 × 4 = 2s
            {
                DELAY();
                P1 = 0xF5;
            }
            n = 0;
            break;                      //A 道结束
        }
     }
     while(1)
     {

        DELAY();
        n++;
        P1 = 0xDE;                      //B 道放行,A 道禁止
        if(n == 50)                     //25s 计时,50 × 0.5s = 25s
        {
            for(i = 0;i < 6;i++)        //B 道绿灯闪烁 3s,每 0.5s 灭一次,0.5 × 6 = 3s
            {
                DELAY();
            P1_5 = !P1_5;

            }
            for(i = 0;i < 4;i++)        //B 道黄灯亮 2s,0.5 × 4 = 2s
            {
                DELAY();
                P1 = 0xEE;

            }
            n = 0;
            break;                      //B 道结束
        }
     }

}
```

3）用 Proteus 软件仿真

经过 Keil 软件编译通过后，在 Proteus ISIS 编辑环境中绘制仿真电路图，将编译好的 XM26.hex 文件分别加载到 AT89C51 里，然后启动仿真，就可以看到交通灯，效果图如图 5-9 所示。

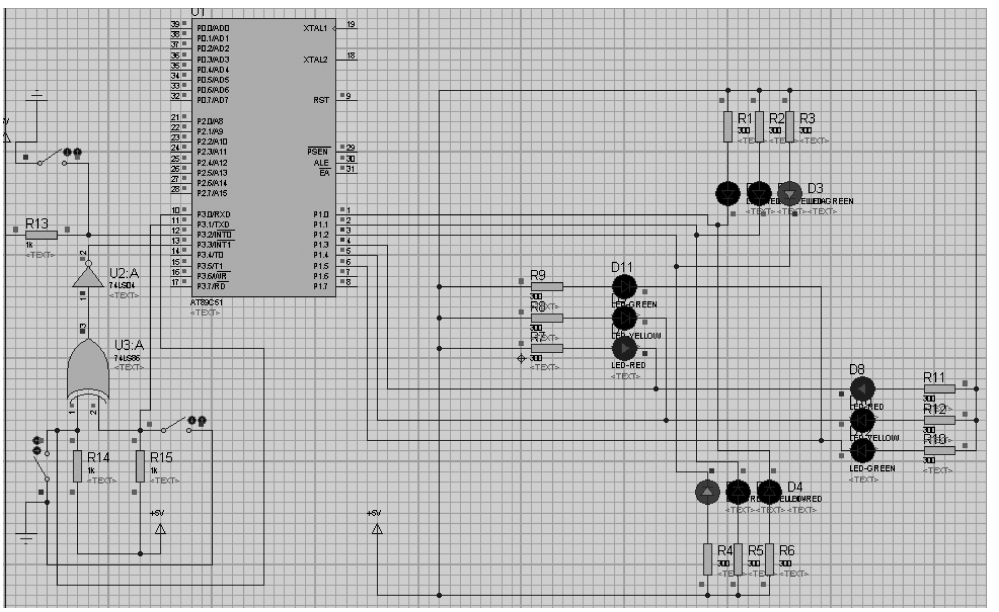

图 5-9　交通灯仿真效果图

模 块 小 结

（1）80C51 中断系统主要由定时器控制寄存器 TCON、串行口控制寄存器 SCON、中断允许寄存器 IE、中断优先级寄存器 IP 和硬件查询电路等组成。

（2）中断处理过程包括中断响应、中断处理和中断返回 3 个阶段。

（3）中断系统初始化的内容包括开放中断允许、确定中断源的优先级别和外部中断的触发方式。

（4）扩展外部中断源的方法有定时器扩展法和中断加查询扩展法两种。

课后练习题

（1）什么是中断？中断系统的功能和特点有哪些？

（2）8051 单片机的中断源有几个？自然优先级是如何排列的？

（3）外部中断触发方式有几种？它们的特点是什么？

（4）中断处理过程包括几个阶段？

（5）请简述中断响应的过程。

（6）外部中断请求撤销时要注意哪些事项？

(7) 中断系统的初始化一般包括哪些内容?

(8) 扩展外部中断源的方法有几种?

(9) 采用中断方式,利用定时器/计数器 T0 从 P1.0 输出周期为 1s,脉宽为 20ms 的正脉冲信号,晶振频率为 12MHz,用 Proteus 软件进行仿真。

(10) 采用中断方式,用方式 0 设计两个不同频率的方波,P1.0 输出频率为 200Hz,P1.1 输出频率为 100Hz,晶振频率为 12MHz,用 Proteus 软件进行仿真。

(11) 采用中断方式,P1.0 输出脉冲宽度调制(PWM)信号,即脉冲频率为 2kHz、占空比为 7∶10 的矩形波,晶振频率为 12MHz,用 Proteus 软件进行仿真。

(12) 采用中断方式,两只开关分别接入 P3.0、P3.1,在开关信号 4 种不同的组合逻辑状态,使 P1.0 分别输出频率 0.5kHz、1kHz、2kHz、4kHz 的方波,晶振频率为 12MHz,用 Proteus 软件进行仿真。

(13) 采用中断方式,有一组高电平脉冲的宽度在 50～100ms 之间,利用定时器 0 测量脉冲的宽度,结果存放到片内 RAM 区以 50H 单元为首地址的单元中,晶振频率为 12MHz,用 Proteus 软件进行仿真。

(14) 用外中断$\overline{\text{INT0}}$测量负跳变信号累计数,并将结果送 P1 口显示,用 Proteus 软件进行仿真。

(15) 用外中断$\overline{\text{INT1}}$测量外部负脉冲宽度,并将结果送 P2 口显示,用 Proteus 软件进行仿真。

(16) 基于 AT89C51 单片机交通灯控制器,要求硬件、软件设计并仿真,具体技术指标如下:

用 AT89S52 单片机控制一个交通信号灯系统,晶振频率采用 12MHz。设 A 车道与 B 车道交叉组成十字路口,A 是主道,B 是支道。设计要求如下所示。

① 用发光二极管模拟交通信号灯,用按键开关模拟车辆检测信号。

② 正常情况下,A、B 两车道轮流放行,A 车道放行 60s,其中 5s 用于警告;B 车道放行 40s,其中 5s 用于警告。

③ 有紧急车辆通过时,按下 K3 开关使 A、B 车道均为红灯,禁行 15s。

参 考 文 献

[1] 杨居义.单片机原理及应用项目教程(基于C语言)[M].北京:清华大学出版社,2014.

[2] 王东锋,王会良,董冠强.单片机C语言应用100例[M].北京:电子工业出版社,2009.

[3] 杨居义.单片机案例教程[M].北京:清华大学出版社,2015.

[4] 杨居义,等.单片机原理与工程应用[M].北京:清华大学出版社,2009.

[5] 杨居义.单片机课程实例教程[M].北京:清华大学出版社,2010.

[6] 楼然苗.8051系列单片机C程序设计[M].北京:北京航空航天大学出版社,2007.

[7] 求是科技.单片机应用技术[M].2版.北京:人民邮电出版社,2008.

[8] 马忠梅.单片机的C语言程序设计[M].4版.北京:北京航空航天出版社,2007.

[9] 张道德.单片机接口技术(C51版)[M].北京:中国水利水电出版社,2007.

[10] 吕凤翥.C语言程序设计[M].北京:清华大学出版社,2006.

[11] 徐爱钧.Keil Cx51 V7.0单片机高级语言编程与μVision2应用实践[M].北京:电子工业出版社,2004.