单片机——神奇的中断嵌套实验

原创

kevinup 于 2019-03-29 23:10:17 发布 7019 收藏 32
分类专栏： 单片机 文章标签： 单片机 中断嵌套

题目要求：

①INT0、INT1同时存在，INT1优先级高于INT0，均为电平触发；

②主程序8个led轮动12轮，再闪烁12次；INT0的中断服务程序控制8个led闪烁13次，INT1的中断服务程序控制8个led轮动10轮（所有计数通过数码管显示，且只能用数字显示，不能用字母（16进制）计数显示）。其中，led流动一下的时间为100ms,闪烁一次所用时间为50ms，所有的定时用中断处理，不允许使用软件定时。

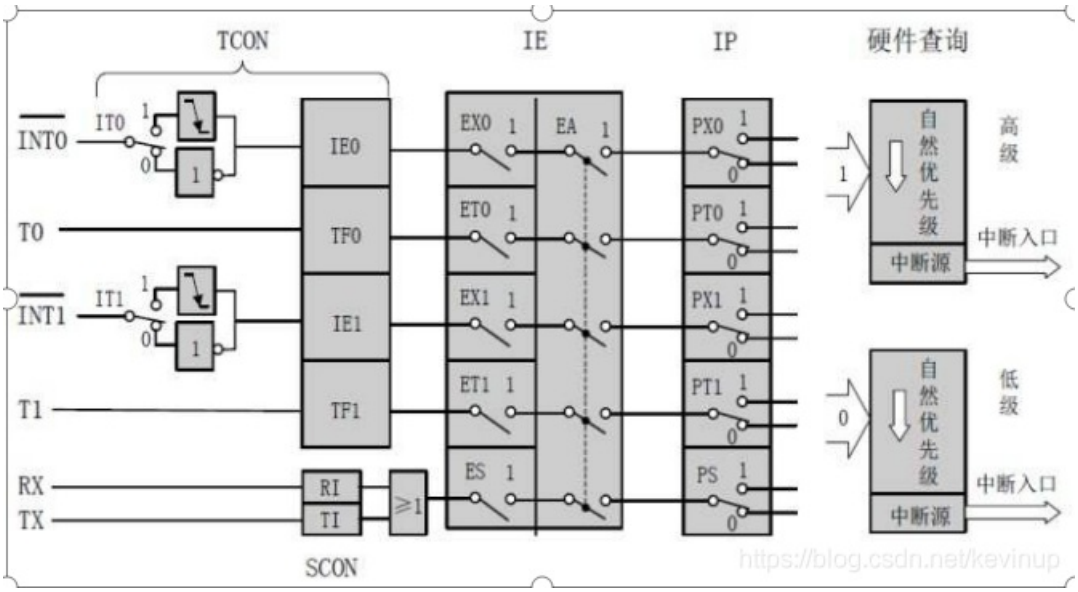
//==========

题目分析：这个题目纯粹没事搞事，这个题目纯粹没事搞事，这个题目纯粹没事搞事。

//==========

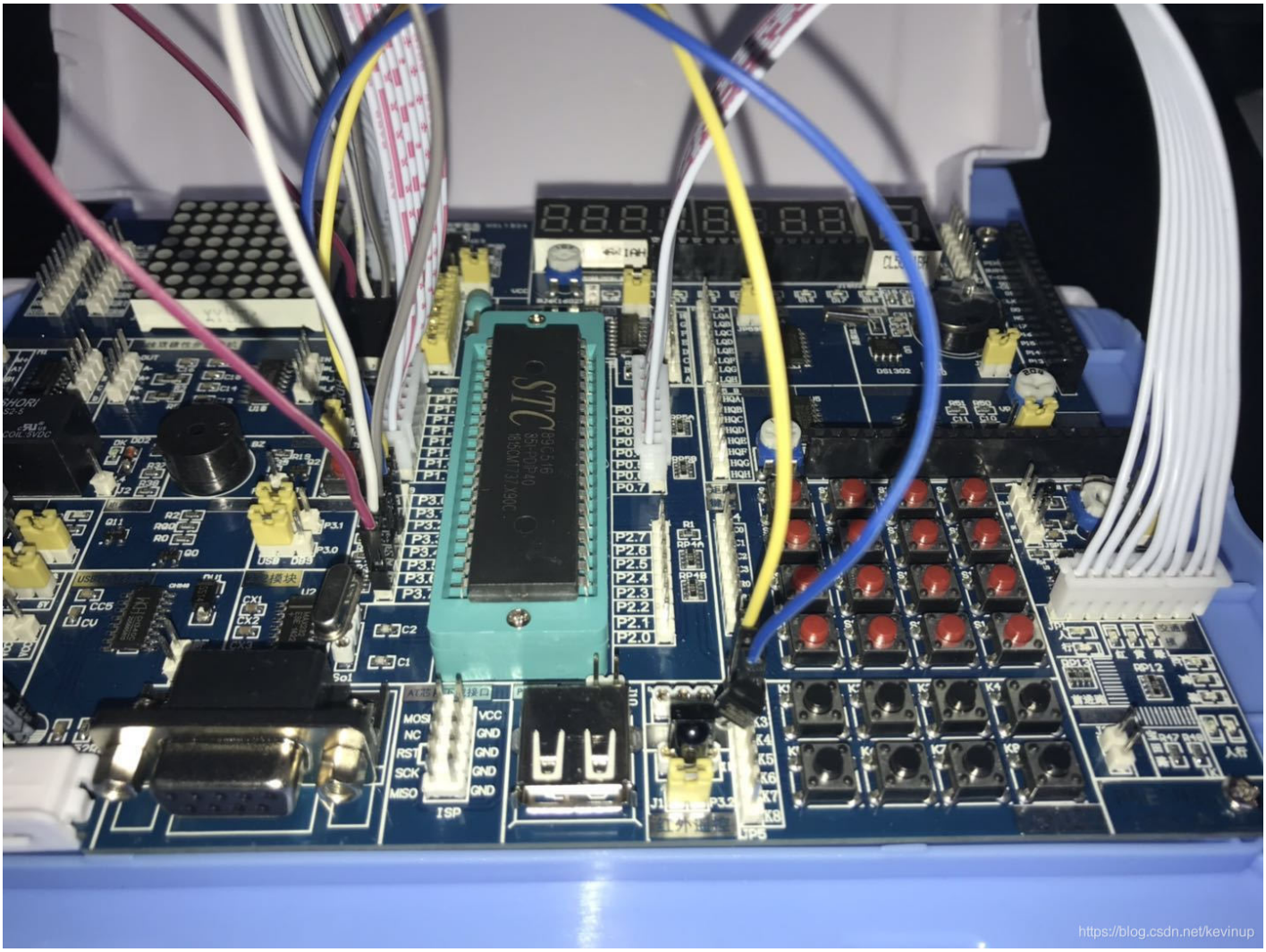
题目分析：因为不能用软件延时，因此我的考虑是开两个定时器，一个定时器走主程序，另外一个定时器就负责在外部中断来的时候走外部中断的对应功能。解决这个题目的关键在于处理好各个中断之间的优先级关系，要使INT1优先级高于INT0，就需要对相关的寄存器标志位做调整，即PX1=1，同时，因为中断优先级调整后其实最多是同级关系，比如外部中断1调高了优先级，而外部中断0本身就是最高优先级，这是物理电路制作时的默认关系，是客观存在的，是不以人的意志为转移的，所以外部中断0和外部中断1最多也就是平级，因此还需要通过PX0=0，把外部中断0的优先级调低，这样外部中断1才可能打断外部中断0。

同时，需要注意的是，定时器1的优先级在外部中断0、定时器0、外部中断1，在这几个老哥当中，他的优先级是最低的，因此我们外部中断需要实现的功能是不能直接写在外部中断里面服务函数里面的，不然led不会闪烁、流水灯不会流动，因为定时器1是进不去这两个中断的，优先级不够。所以我们的一种操作可以是，外部中断触发时，产生相应的标志位，定时器1定时检测标志位的状态，以此决定要不要执行相应的外部中断功能事件。



单片机开发平台：普中科技HC6800-EM3-V2.2

接线：JP8-J12、JP10-JP1、P32-K1、P33-K3、P35-A、P36-B、P37-C



代码实现：

把代码分成了不同的模块，公开如下：

//==========main.c=====================

```
#include <reg51.h>
#include <intrins.h>
#include "fun.h"
#include "timer0.h"
#include "int0.h"
#include "int1.h"
#include "timer1.h"

//所有功能由定时器和外部中断完成，四个中断相互配合，分别完成主程序，一号中断程序，二号中断程序
//主程序：8个发光二极管上下轮动12轮，再闪烁12次
//一号中断：8个led闪烁13次
//二号中断：led上下轮动10轮
//定时器0执行主程序，外部中断0执行1号中断，外部中断1执行2号中断，通过调整，PX1 = 1，将外部中断1优先级调整为最高级
//注意：当一个中断到来的时候，记得先关闭其他中断，执行完这个中断后，再打开别的中断继续执行

void main()
{
   T0_init();         //定时器0初始化
   INT0_init();   //外部中断0初始化
   INT1_init();   //外部中断1初始化
   Timer1Init();  //定时器1初始化
   while(1);
}

//==============================fun.c==================================

#include <reg51.h>
#include <intrins.h>
#include "fun.h"
#include "timer0.h"
#include "int0.h"
#include "int1.h"
#include "timer1.h"
/*********************************第4题*********************************************************************/
//用定时器0完成主程序，定时器1配合两个外部中断完成剩余功能
uchar num=0; //流水灯程序中使用的计数变量，num=8时表明完成1次正或者反的流水
uchar flag=0;  //flag=0时运行正向流水灯，flag=1反向，flag=2闪烁
uchar count2=0;  //count2作为流水次数计数的中间变量，每流水1次产生变化后赋值给seg_num，满12清0
uint  count3=0;       //用于主程序led闪烁次数的计数变量
uchar seg_num=0;  //根据seg_num的返回值对应到数组tab[]，从而显示出相应的数字
uchar temp1=0;    //定时器0为1ms定时器，temp1是用于选择数码管的变量
uchar temp2=0;    //定时器1为1ms定时器，temp2是用于选择数码管的变量
uint  count1=0; //count1是配合定时器0使用的计数变量,满50或者100清0
uint  count_T0=0; //count_T0是配合定时器1，外部中断0使用的计数变量
uint  count_T0_n=0;  //用于记录外部中断0中led闪烁次数的计数变量
uint  count_T1=0;  //count_T0是配合定时器1，外部中断1使用的计数变量
uchar flag_T0=1; //外部中断0对应标志位，flag_T0=0执行中断
uchar flag_T1=1; //外部中断1对应标志位，flag_T1=0执行中断
uchar code tab[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
//=============================================================================
//=============LED的work4,work5函数=========
```

```
void work4()      //work4为正向流水灯的程序
{
   if(flag==0)
   {
      P0=~(1<<num);
      num++;
      if(num==8) {num=0;flag=1;}
   }
}
void work5()          //work5为反向流水灯的程序
{
   if(flag==1)
   {
      P0=~(0x80>>num);
      num++;
      if(num==8) {num=0;flag=0;count2++;seg_num=count2;}
   }
}
//====================================
//====================seg=============
void seg(uchar dat)
{
   P1 = tab[dat];
}
//====================================
```

```
//====================int0.c============================

#include <reg51.h>
#include <intrins.h>
#include "fun.h"
#include "timer0.h"
#include "int0.h"
#include "int1.h"
#include "timer1.h"

extern uchar num;
extern uchar temp1;
extern uint  count1,count3;
extern uchar count2;
extern uint  count_T0,count_T0_n,count_T1;
extern uchar flag_T0,flag_T1;
extern uchar flag;
extern uchar seg_num;
extern uchar tab[];
```

```
//==================外部中断0=======================
void INT0_init()
{
   IT0 = 0;
   EX0 = 1;
   EA = 1;
}

void INT0_service() interrupt 0
{
   P0=0x00;     //外部中断0到来时，需要所有灯闪烁13次，在执行前先关闭所有led
   TR1 = 1;     //定时器1开始计时
   ET1 = 1;          //打开定时器1的中断
   flag_T0=0;
}
//================================================



//=========================int1.c=================================

#include <reg51.h>
#include <intrins.h>
#include "fun.h"
#include "timer0.h"
#include "int0.h"
#include "int1.h"
#include "timer1.h"

extern uchar num;
extern uchar temp1;
extern uint  count1,count3;
extern uchar count2;
extern uint  count_T0,count_T0_n,count_T1;
extern uchar flag_T0,flag_T1;
extern uchar flag;
extern uchar tab[];
extern uchar seg_num;

//================外部中断1=======================
void INT1_init()
{
   IT1 = 0;
   EX1 = 1;
   EA = 1;
   //在默认关系中，外部中断1的优先级为第3级，外部中断0的优先级为第1级，若要改变优先级，则通过改变寄存器IP的控制位PX1,令PX1=1
   PX1 = 1;  //将外部中断1优先级调整为最高级
}
```

```
void INT1_service() interrupt 2
{
   TR1=1;ET1=1;
   flag_T1=0;
   seg_num = 0;
}
//=================================================



//=======================timer0.c=========================

#include <reg51.h>
#include <intrins.h>
#include "fun.h"
#include "timer0.h"
#include "int0.h"
#include "int1.h"
#include "timer1.h"

extern uchar num;
extern uchar temp1;
extern uint  count1,count3;
extern uchar count2;
extern uint  count_T0,count_T0_n,count_T1;
extern uchar flag_T0,flag_T1;
extern uchar flag;
extern uchar seg_num;
extern uchar tab[];

//============定时器0=====================================================
void T0_init()          //1ms定时
{
   TMOD &= 0xF0;        //设置定时器模式
   TMOD |= 0x01;        //设置定时器模式
   TH0 = (65536-1000)/256;
   TL0 = (65536-1000)%256;
   TF0 = 0;    //清除TF0标志

   TR0 = 1;   //定时器0开始计时
   ET0 = 1;   //开定时器0中断
   EA  = 1;   //开总中断
}
```

```
void T0_service() interrupt 1
{
   TH0 = (65536-1000)/256;        //模式1，16位，需要手动重新装载计数值
   TL0 = (65536-1000)%256;

   count1++;
   if(count1==100)     //根据题意，灯是每100ms移动一次
   {
      if( (flag==0)|(flag==1) )
      {
         count1=0;    //进来之后记得count1重置为0，不然记完第一个100ms以后就无效了
         work4();
         work5();
         if(count2==12) {seg_num=0;flag=2;count2=0;P0=0x00;} //正反流水灯总次数达到12，count2清0，数码
管计数seg_num=0，flag置2，同时关闭所有led，以免程序错乱
      }
   }

   if( (count1==50)&&(flag==2) ) //根据题意，灯是每50ms闪烁一次
   {
      count1=0; //进来之后记得count1重置为0，不然记完第一个100ms以后就无效了
      P0=~P0;  //闪烁一下
      count3++;  //用于闪烁计数的变量，亮+灭 为一次闪烁
      seg_num=count3/2;
      if(count3==24) {count3=0;flag=0;seg_num=0;} //闪烁达到12次，count3清0，flag清0
   }

   //==================数码管显示====================
   C1 = (temp1>>2)&0x01;
   B1 = (temp1>>1)&0x01;
   A1 = (temp1&0x01);
   if(temp1==0)
   {
      seg(seg_num/10);  //十位
      temp1++;
   }
   else if(temp1==1)
   {
      seg(seg_num%10);  //个位
      temp1++;
      if(temp1==2) temp1=0;
   }
}
//====================================================================



//==============================timer1.c==============================
```

```
#include <reg51.h>
#include <intrins.h>
#include "fun.h"
#include "timer0.h"
#include "int0.h"
#include "int1.h"
#include "timer1.h"

extern uchar num;
extern uchar temp1;
extern uchar temp2;
extern uint  count1,count3;
extern uchar count2;
extern uint  count_T0,count_T0_n,count_T1;
extern uchar flag_T0,flag_T1;
extern uchar flag;
extern uchar tab[];
extern uchar seg_num;
uchar num1=0;
uchar count4=0;
//====================定时器1================================
void Timer1Init()       //1ms@11.0592MHz
{
   TMOD &= 0x0F;       //设置定时器模式
   TMOD |= 0x10;       //设置定时器模式
   TH1 = (65536-1000)/256;
   TL1 = (65536-1000)%256;

   TF1 = 0;       //清除TF1标志
   TR1 = 0;       //定时器1先不计时，等下用到再开始
   ET1 = 0;   //先关闭定时器1的中断，等下用到的时候再打开
   EA  = 1;
}

void T1_service() interrupt 3
{
   TH1 = (65536-1000)/256;
   TL1 = (65536-1000)%256;

   //=======================外部中断0对应程序
==============================================
   if(flag_T0==0)
   {
      //定时器1的开始执行，说明外部中断正在执行，此时应该先暂停执行主程序的定时器0，外部中断结束后再
继续执行定时器0
      TR0 = 0;   //关闭定时器0计时
      ET0 = 0;   //关闭定时器0中断
      count_T0++;
      if(count_T0==50)    //根据题意，50ms闪烁
      {
         P0=~P0;
```

```
        count_T0=0;
        count_T0_n++; //用于记录外部中断0中led闪烁次数的计数变量
        seg_num=count_T0_n/2;
        if(count_T0_n==26) {count_T0_n=0;flag_T0=1;TR1=0;ET1=0;TR0=1;ET0=1;} //外部中断0执行完毕，
关闭定时器1，打开定时器0
      }
   }


   //=========================外部中断1对应程序
=========================================
   count_T1++;
   if(count_T1==100)   //根据题意，灯是每100ms移动一次
   {
      count_T1=0;
      if(flag_T1==0)   //flag_T1为0，表明需要执行外部中断1
      {
         flag_T0=1; //确认要执行外部中断1，因此关闭外部中断0事件对应的标志位
         TR0 = 0;  //关闭定时器0计时
         ET0 = 0;  //关闭定时器0中断
         if( (flag==0)|(flag==1) )
         {
            work44();    //work44，work55功能与work4，work5一样，只是为了不要与主程序混用
            work55();
            if(count4==10) {seg_num=0;count4=0;P0=0x00;TR1=0;ET1=0;flag_T1=1;TR0 = 1;ET0 =
1;flag_T0=0;}//外部中断1执行完毕，同样需要关闭定时器1，打开定时器0，恢复外部中断0标志位
         }
      }
   }

   //=====================seg====================
   C1 = (temp2>>2)&0x01;
   B1 = (temp2>>1)&0x01;
   A1 = (temp2&0x01);
   if(temp2==0)
   {
      seg(seg_num/10);
      temp2++;
   }
   else if(temp2==1)
   {
      seg(seg_num%10);
      temp2++;
      if(temp2==2) temp2=0;
   }
   P1=0x00; //本身是1ms定时器可以不加消隐，加消隐降低seg亮度,以示和主程序的区别
}
//================================================
//=============LED的work44,work55函数=========
void work44()
{
```

```
{
   if(flag==0)
   {
      P0=~(1<<num1);
      num1++;
      if(num1==8) {num1=0;flag=1;}
   }
}
void work55()
{
   if(flag==1)
   {
      P0=~(0x80>>num1);
      num1++;
      if(num1==8) {num1=0;flag=0;count4++;seg_num=count4;}
   }
}
//=============================================================

//======================fun.h=================================

#ifndef _FUN_H
#define _FUN_H

#define uchar unsigned char
#define uint  unsigned int


sbit A1=P3^5;
sbit B1=P3^6;
sbit C1=P3^7;

//第4题函数声明
void work4();
void work5();
void seg(uchar dat);

#endif

//======================int0.h=================================

#ifndef _INT0_H
#define _INT0_H

#define uchar unsigned char
#define uint  unsigned int

void INT0_init();

#endif

//======================int1.h=================================

#ifndef _INT1_H
#define _INT1_H
```

```
#define uchar unsigned char
#define uint  unsigned int

void INT1_init();

#endif

//=========================timer0.h=================================

#ifndef _TIMER0_H
#define _TIMER0_H

#define uchar unsigned char
#define uint  unsigned int

void T0_init();

#endif

//=======================timer1.h=================================

#ifndef _TIMER1_H
#define _TIMER1_H

#define uchar unsigned char
#define uint  unsigned int

void Timer1Init();
void work44();
void work55();

#endif

//=================================================================
```

经过检验，功能是实现了，谈点感想吧，为什么取个名字叫神奇的中断嵌套呢，因为这个实验确实能帮助体验中断之间的优先级关系-_-不过挺花时间的，每个功能之间的逻辑关系需要反复调试

ps:这个实验是上周的内容，做完之后周末参加了蓝桥杯，这周回来之后再对着自己的代码打注释，能确实感觉到自己在提高