Embedded system experiment report #1: Clock

Hong Chenhui[a]

[a] *DrRedTheRed*

Professor/You Wang

Abstract—A rudimentary digital clock with timer and alarm functionalities, adjustable via a mere three buttons. Powered by AT89S52.

keywords—*LaTeX No.*

Contents

1. Task description

Develop a real-time clock capable of displaying on a digital display, adjustable via a minimal number of buttons, not exceeding three. Please establish a logical button configuration. The system should feature programmable timer and alarm functions, indicated by a buzzer. The experiment will be conducted using a laboratory kit.

2. Code structure

Nothing particularily interesting on code structure. Primitive implementation as follows.

1. **Main loop:** contains the main program, where
    - The keys undergo a scanning process to procure the input slated for processing.
    - The alarm or timer is engaged to initiate the activation of the buzzer.
    - The digital displays are configured.
2. **Interrupt:** contains the interrupt service routine, with
    - The timer 0 and 1.
    - The countdown.
    - The display refreshes.

3. Implementation

3.1. Hardware

3.1.1. Items

1. STC89C51 chip
2. Buzzer
3. Digital Display
4. 74HC138 line decoder
5. Key module

3.1.2. Wiring

- **STC89C51** P2.2-P2.4 → **74HC138** C-A
- **STC89C51** P0 → **Digital Display**
- **STC89C51** P1.5 → **Buzzer** J7
- **STC89C51** P3.0-P3.2 → **Key module** K1-K3

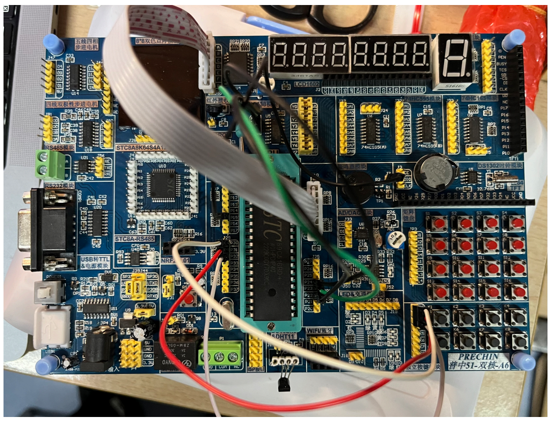

Figure 1. Board layout.

3.2. Manual

- **K1:** Mode selection.

Table 1. Mode division

Clock(1-3)	Alarm(4-6)	Countdown(7-9)
Hour(1)	Alarm.hour(4)	Countdown.hour(7)
Minute(2)	Alarm.minute(5)	Countdown.minute(8)
Second(3)	Alarm.second(6)	Countdown.second(9)

Note: Default mode is set to hours, each press of the K1 key increments sequentially

- **K2:** Each press increases the mode number incrementally.
- **K3:** Activate/Deactivate the alarm/countdown in corresponding mode.

3.3. Code

The code could roughly be divided into two parts: the main loop and the two interrupt. The main loop is responsible for the main program, while both interrupts is responsible for the interrupt service routine.

```c
#include<reg52.h>

#define KeyPort P3
#define u8 unsigned char //but actually nobody use it so
#define u16 unsigned int //same tbh

/*------------------------------------------------
                   Proclaims
------------------------------------------------*/

sbit LSA=P2^2;
sbit LSB=P2^3;
sbit LSC=P2^4;

sbit ALA=P1^5;//ports

u8 dis=0;//used to change the display number in digital display

unsigned char hour,minute,second;

bit UpdateTimeFlag;

unsigned char code dofly_DuanMa[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
unsigned char code dofly_WeiMa[]={0xfe,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f};

```

```c
u8 code smgduan[17]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,
          0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};

unsigned char TempData[8],Key_Num;

/*My proclaims*/
enum MODE{HOUR,MINUTE,SECOND,ALARM_H,ALARM_M,ALARM_S,COUNT_H,COUNT_M,COUNT_S};

enum MODE mode = HOUR;

struct TIMESTRUCT{
  unsigned char hour;
  unsigned char minute;
  unsigned char second;
};

struct TIMESTRUCT alarm = {0,0,0};

struct TIMESTRUCT count = {0,0,0};

enum CONFIRM{NO,YES};

enum CONFIRM confirm = NO;

enum ACTIVATE{NO_1,YES_1};

enum ACTIVATE alarm_activate = NO_1;
enum ACTIVATE count_activate = NO_1;
enum ACTIVATE buzzer_activate = NO_1;
/*My proclaims ended*/

void countdown();
void playChineseTea();

void DelayUsNo(unsigned int t);//same as the next one
void DelayUs2x(unsigned char t);//Delay on microsecond
void DelayMs(unsigned char t); //Delay on millisecond
void DigDisplay();//Digital display
unsigned char KeyScan(void);//Keyboard scan
void    Init_Timer0(void);//Timer initialization
/*------------------------------------------------
                    Main functions
------------------------------------------------*/
void delay(u16 i)
{
  while(i--);
}
/*------------------------------------------------*/
void main (void)
{
Init_Timer0();

while (1)
  {
          int i = 0;

          Key_Num=KeyScan();
          switch(Key_Num)
            {
                  case 1:
                      mode++;
                      mode = (mode%9);
                    break;
                  case 2:
                    if(mode == HOUR)
                    {
                      hour++;if(hour==24)hour=0;
                    }else if(mode == MINUTE)
                    {
                      minute++;if(minute==60)minute=0;
                    }else if(mode == SECOND)
                    {
                      second++;if(second==60)second=0;
                    }else if(mode == ALARM_H)
                    {
                      alarm.hour++;if(alarm.hour==24)alarm.hour=0;
                    }else if(mode == ALARM_M)
                    {
                      alarm.minute++;if(alarm.minute==60)alarm.minute=0;
                    }else if(mode == ALARM_S)
                    {
                      alarm.second++;if(alarm.second==60)alarm.second=0;
                    }else if(mode == COUNT_H)
                    {
                      count.hour++;if(count.hour==24)count.hour=0;
                    }else if(mode == COUNT_M)
                    {
                      count.minute++;if(count.minute==60)count.minute=0;
                    }else if(mode == COUNT_S)
                    {
                      count.second++;if(count.second==60)count.second=0;
                    }
                    break;
                  case 3:confirm = YES;
                    break;
                  default:break;
            }

          if(mode == HOUR || mode == MINUTE || mode == SECOND)
          {
             TempData[0]=dofly_DuanMa[hour/10];
             TempData[1]=dofly_DuanMa[hour%10];
             TempData[2]=0x40;
             TempData[3]=dofly_DuanMa[minute/10];
             TempData[4]=dofly_DuanMa[minute%10];
             TempData[5]=0x40;
             TempData[6]=dofly_DuanMa[second/10];//Store the time
             TempData[7]=dofly_DuanMa[second%10];
          }
          else if (mode == ALARM_H || mode == ALARM_M || mode == ALARM_S )
          {
             TempData[0]=dofly_DuanMa[alarm.hour/10];
             TempData[1]=dofly_DuanMa[alarm.hour%10];
             TempData[2]=0x40;
             TempData[3]=dofly_DuanMa[alarm.minute/10];
             TempData[4]=dofly_DuanMa[alarm.minute%10];
             TempData[5]=0x40;
             TempData[6]=dofly_DuanMa[alarm.second/10];
             TempData[7]=dofly_DuanMa[alarm.second%10];
          }
          else if (mode == COUNT_H || mode == COUNT_M || mode == COUNT_S )
          {
             TempData[0]=dofly_DuanMa[count.hour/10];
             TempData[1]=dofly_DuanMa[count.hour%10];
             TempData[2]=0x40;
             TempData[3]=dofly_DuanMa[count.minute/10];
             TempData[4]=dofly_DuanMa[count.minute%10];
             TempData[5]=0x40;
             TempData[6]=dofly_DuanMa[count.second/10];
             TempData[7]=dofly_DuanMa[count.second%10];
          }

          if(confirm == YES)
          {
            confirm = NO;
            if(mode == ALARM_H || mode == ALARM_M || mode == ALARM_S)
            {
              alarm_activate = !alarm_activate;
            }else if(mode == COUNT_H || mode == COUNT_M || mode == COUNT_S)
            {
              count_activate = !count_activate;
            }
            mode = HOUR;
          }

          if( alarm_activate == YES_1 && alarm.hour == hour && alarm.minute == minute && alarm.second == second)
          {
            buzzer_activate = YES_1;
          }

          if(buzzer_activate == YES_1)
          {
            buzzer_activate = !buzzer_activate;

            playChineseTea();

          }

```

```c
              //DigDisplay();

 }
}

/*------------------------------------------------
         Countdown is called in Interrupt
------------------------------------------------*/

void countdown()
{
  static unsigned int num;

  if( count_activate == YES_1 && count.hour == 0 &&
     count.minute == 0 && count.second == 0)
  {
    count_activate = !count_activate;

    buzzer_activate = YES_1;

    mode = HOUR;
  }

  if(count_activate == YES_1)
  {
    num++;
     if(num==500)        //Roughly 1 second
   {
    num=0;
  count.second--;
  if(count.second==255)
   {
     count.second = 59;
    count.minute--;
    if(count.minute==255)
      {
     count.minute=59;
     count.hour--;
      }
    }
  }
  }
}

/*------------------------------------------------
Yes I know this could be done better but idc.
This is a song in Touhou Project 6: Embodiment of
 Scarlet Devil. After writing this the program
 reached its memory limit.
------------------------------------------------*/

void playChineseTea()
{
    int i;
  //standard 1 note: 33800
    for( i = 0; i< 150 ; i++ )
    {
      ALA = 1;
      DelayUsNo(675/2);
      ALA = 0;
      DelayUsNo(675/2);
    }
    DelayMs(20);

    for( i = 0; i< 100 ; i++ )
    {
      ALA = 1;
      DelayUsNo(338/2);
      ALA = 0;
      DelayUsNo(338/2);
    }
    DelayMs(20);

    for( i = 0; i< 134 ; i++ )
    {
      ALA = 1;
      DelayUsNo(379/2);
      ALA = 0;
      DelayUsNo(379/2);
    }
    DelayMs(20);

    for( i = 0; i< 179 ; i++ )
    {
      ALA = 1;
      DelayUsNo(284/2);
      ALA = 0;
      DelayUsNo(284/2);
    }
    DelayMs(20);

    for( i = 0; i< 112 ; i++ )
    {
      ALA = 1;
      DelayUsNo(301/2);
      ALA = 0;
      DelayUsNo(301/2);
    }
    DelayMs(20);

    for( i = 0; i< 88 ; i++ )
    {
      ALA = 1;
      DelayUsNo(379/2);
      ALA = 0;
      DelayUsNo(379/2);
    }
    DelayMs(20);

    for( i = 0; i< 74 ; i++ )
    {
      ALA = 1;
      DelayUsNo(451/2);
      ALA = 0;
      DelayUsNo(451/2);
    }
    DelayMs(20);

    for( i = 0; i< 88 ; i++ )
    {
      ALA = 1;
      DelayUsNo(379/2);
      ALA = 0;
      DelayUsNo(379/2);
    }
    DelayMs(20);

    for( i = 0; i< 44 ; i++ )
    {
      ALA = 1;
      DelayUsNo(379/2);
      ALA = 0;
      DelayUsNo(379/2);
    }
    DelayMs(20);

    for( i = 0; i< 550 ; i++ )
    {
      ALA = 1;
      DelayUsNo(338/2);
      ALA = 0;
      DelayUsNo(338/2);
    }
    DelayMs(20);
    //1st sentence
    for( i = 0; i< 150 ; i++ )
    {
      ALA = 1;
      DelayUsNo(675/2);
      ALA = 0;
      DelayUsNo(675/2);
    }
    DelayMs(20);

    for( i = 0; i< 100 ; i++ )
    {
      ALA = 1;
      DelayUsNo(338/2);
      ALA = 0;
      DelayUsNo(338/2);
    }
    DelayMs(20);

    for( i = 0; i< 142 ; i++ )
    {
      ALA = 1;
      DelayUsNo(358/2);
      ALA = 0;
      DelayUsNo(358/2);
    }
    DelayMs(20);

    for( i = 0; i< 150 ; i++ )
    {
      ALA = 1;
      DelayUsNo(338/2);
      ALA = 0;
      DelayUsNo(338/2);
    }
    DelayMs(20);

    for( i = 0; i< 112 ; i++ )
    {
      ALA = 1;
      DelayUsNo(301/2);
      ALA = 0;
      DelayUsNo(301/2);
    }
    DelayMs(20);

    for( i = 0; i< 600 ; i++ )
    {
      ALA = 1;
      DelayUsNo(451/2);
      ALA = 0;
      DelayUsNo(451/2);
    }
    DelayMs(20);
    //second sentence

}
/*-------------------------------------------------
for the us delay function, incorporating an input
parameter unsigned char t and no return value,
where unsigned char represents the definition of
an unsigned character variable with a value range
of 0-255, given a 12M crystal oscillator,
for precise delay, assembly language is recommended.
-------------------------------------------------*/
void DelayUsNo(unsigned int t)
{
  while(--t);
}

void DelayUs2x(unsigned char t)
{
 while(--t);
}
/*-------------------------------------------------
same as us buuuut changed to ms
-------------------------------------------------*/
void DelayMs(unsigned char t)
{

 while(t--)
 {
     //roughly 1ms
     DelayUs2x(245);
     DelayUs2x(245);
 }
}
/*-------------------------------------------------
 Digital display
-------------------------------------------------*/
void DigDisplay()
{
    switch(dis)
    {
      case(0):
        LSA=0;LSB=0;LSC=0; break;//0th digit
      case(1):
        LSA=1;LSB=0;LSC=0; break;//1st
      case(2):
        LSA=0;LSB=1;LSC=0; break;//2nd
      case(3):
        LSA=1;LSB=1;LSC=0; break;//3rd
      case(4):
        LSA=0;LSB=0;LSC=1; break;//4th
      case(5):
        LSA=1;LSB=0;LSC=1; break;//5th
      case(6):
        LSA=0;LSB=1;LSC=1; break;//6th
      case(7):
        LSA=1;LSB=1;LSC=1; break;//7th
    }
    P0=TempData[dis];//segmentcode output
    //delay(100); //scan once a while
    //P0=0x00;//Eliminate residual
  }
/*-------------------------------------------------
               Timer Initialization
-------------------------------------------------*/
void Init_Timer0(void)
{
 TMOD = 0x11;   //mode 1 16bit timer
 //TH0=0x00;        //nope, not this
 //TL0=0x00;
 EA=1;            //main Interrupt on
 ET0=1;           //Timer Interrupt on
 TR0=1;           //Timer on
 ET1=1;
 TR1=1;
}
/*-------------------------------------------------
            Interrupt sub program
-------------------------------------------------*/
void Timer0_isr(void) interrupt 1
{
 static unsigned int num;
 TH0=(65536-2000)/256;//give 2ms value
 TL0=(65536-2000)%256;

 num++;

 if(num==500)        //roughly 1s
   {
    num=0;
  second++;
  if(second==60)     //yup
   {
    second=0;
    minute++;
    if(minute==60)  //same
      {
     minute=0;
     hour++;
     if(hour==24)   //same
       hour=0;
    }
   }

  }

    DigDisplay();

  countdown();
}

void Timer0_isr1(void) interrupt 3
{

  TH1=(65536-2000)/256;//2ms is enough for the display to be stable
  TL1=(65536-2000)%256;


  dis++;
  if(dis == 8)
  {
    dis = 0;
  }
}

/*-------------------------------------------------
Scan the key and return the key value
-------------------------------------------------*/
unsigned char KeyScan(void)
{
 unsigned char keyvalue;

 if(KeyPort!=0xff)
   {
    DelayMs(10);
    if(KeyPort!=0xff)
     {
      keyvalue=KeyPort;
      while(KeyPort!=0xff)
      {
        DigDisplay();
      };
    switch(keyvalue)
    {
     case 0xfe:return 1;break;
     case 0xfd:return 2;break;
     case 0xfb:return 3;break;
     case 0xf7:return 4;break;
     case 0xef:return 5;break;
     case 0xdf:return 6;break;
     case 0xbf:return 7;break;
     case 0x7f:return 8;break;
     default:return   0;break;
    }
    }
   }
   return 0;
}
```

Code 1. Full Implementation