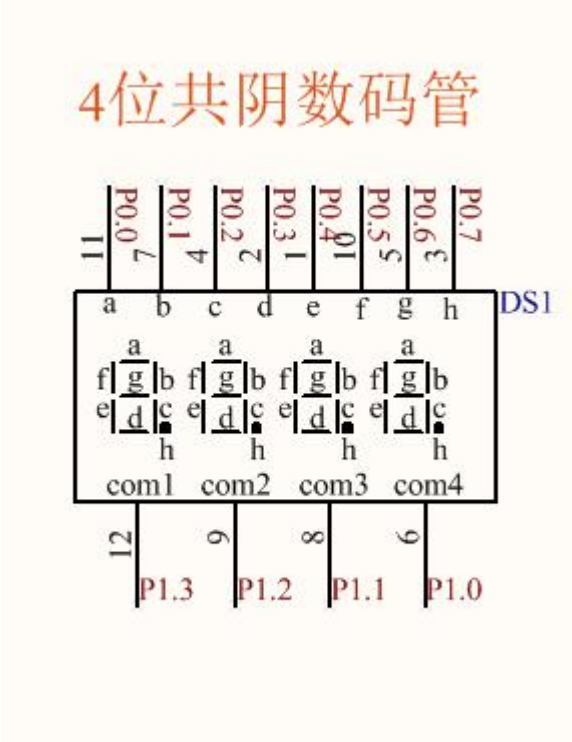
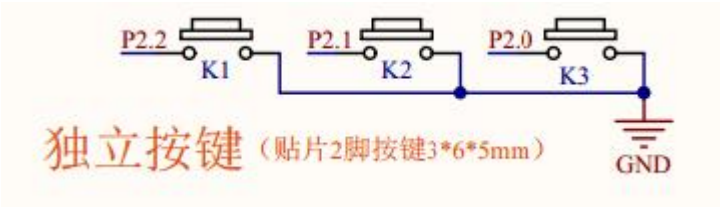
第一百一十七节： 按键切换数码管窗口来设置参数。

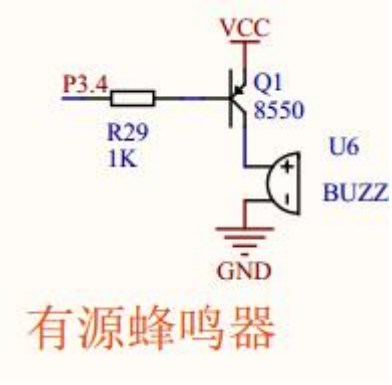
【117.1　按键切换数码管窗口来设置参数。】



上图 117.1.1　数码管



上图 117.1.2　独立按键



上图 117.1.3　有源蜂鸣器

单片机是“数据”驱动型的。按什么逻辑跑，以什么方式跑，都是“数据”决定的。人机交互的核心就是“人”以什么渠道去更改“机”内部的某些“数据”。在程序框架层面，按键更改或者编辑某些数据，我的核心思路都是“在某个窗口下去更改某个特定的数据”，如果某个窗口的数据很多，就需要在此窗口下进一步细分，细分为“某个窗口下的某个局部（子菜单、光标选择）”。可见，“窗口”是支点，“局部”是支点中再细分出来的支点。窗口对应一个名叫 “窗口选择”的全局变量 Gu8Wd，局部（子菜单、光标选择）对应一个名叫“局部选择”的全局变量 Gu8Part。数据发生变化的时候，才需要更新显示到数码管上，平时不用一直更新显示，因此，与“窗口选择”Gu8Wd 还对应一个名叫“整屏更新”的全局变量 Gu8WdUpdate，与“局部选择”Gu8Part 还对应一个名叫“局部更新”的全局变量 Gu8PartUpdate。本节的小项目程序只用到“窗口”，没有用到“局部”。

本节小项目的程序功能，利用按键与数码管的人机交互，可以对单片机内部三个参数 Gu8SetData_1，Gu8SetData_2，Gu8SetData_3 进行编辑。这三个参数分别在三个窗口下进行编辑，这三个窗口是数码管显示“1-XX”，“2-YY”，“3-ZZ”。其中，XX 代表 Gu8SetData_1 数据，YY 代表 Gu8SetData_2 数据，ZZ 代表 Gu8SetData_3 数据，这三个数据的范围是从 0 到 99。K1 是窗口切换按键，每按一次，窗口会在“1-XX”，“2-YY”，“3-ZZ”三个窗口之间进行切换。K2 是数字累加按键，每按一次，显示的数字会累加 1。K3 是数字递减按键，每按一次，显示的数字会递减 1。代码如下：

```
#include "REG52.H"

#define KEY_FILTER_TIME  25
#define SCAN_TIME  1
#define VOICE_TIME   50

void T0_time();
void SystemInitial(void) ;
void Delay(unsigned long u32DelayTime) ;
void PeripheralInitial(void) ;

void KeyScan(void);
void KeyTask(void);

void VoiceScan(void);
void DisplayScan(void);
void DisplayTask(void);  //上层显示的任务函数
void Wd1(void);   //窗口 1 显示函数
void Wd2(void);   //窗口 2 显示函数
void Wd3(void);   //窗口 3 显示函数

void BeepOpen(void);
void BeepClose(void);

sbit KEY_INPUT1=P2^2;
sbit KEY_INPUT2=P2^1;
sbit KEY_INPUT3=P2^0;
```

```
sbit P1_0=P1^0;
sbit P1_1=P1^1;
sbit P1_2=P1^2;
sbit P1_3=P1^3;

sbit P3_4=P3^4;

//数码管转换表
code unsigned char Cu8DigTable[]=
{
0x3f,  //0       序号 0
0x06,  //1       序号 1
0x5b,  //2       序号 2
0x4f,  //3       序号 3
0x66,  //4       序号 4
0x6d,  //5       序号 5
0x7d,  //6       序号 6
0x07,  //7       序号 7
0x7f,  //8       序号 8
0x6f,  //9       序号 9
0x00,  //不显示  序号 10
0x40,  //横杠-   序号 11
};

volatile unsigned char vGu8ScanTimerFlag=0;
volatile unsigned int vGu16ScanTimerCnt=0;

volatile unsigned char vGu8BeepTimerFlag=0;
volatile unsigned int vGu16BeepTimerCnt=0;

unsigned char Gu8SetData_1=0; //单片机内部第 1 个可编辑的参数
unsigned char Gu8SetData_2=0; //单片机内部第 2 个可编辑的参数
unsigned char Gu8SetData_3=0; //单片机内部第 3 个可编辑的参数

unsigned char Gu8Wd=1;   //窗口选择变量。人机交互程序框架的支点。初始化开机后显示第 1 个窗口。
unsigned char Gu8WdUpdate=1;  //整屏更新变量。初始化为 1 开机后整屏更新一次显示。

volatile unsigned char vGu8Display_Righ_4=1;  //显示窗口“1”
volatile unsigned char vGu8Display_Righ_3=11; //显示横杠“-”
volatile unsigned char vGu8Display_Righ_2=0;  //显示十位数值“0”
volatile unsigned char vGu8Display_Righ_1=0;  //显示个位数值“0”

volatile unsigned char vGu8Display_Righ_Dot_4=0;
volatile unsigned char vGu8Display_Righ_Dot_3=0;
```

```
volatile unsigned char vGu8Display_Righ_Dot_2=0;
volatile unsigned char vGu8Display_Righ_Dot_1=0;

volatile unsigned char vGu8KeySec=0;

void main()
{
    SystemInitial();
    Delay(10000);
    PeripheralInitial();
    while(1)
    {
        KeyTask();      //按键的任务函数
        DisplayTask();  //数码管显示的上层任务函数
    }
}

void KeyTask(void)    //按键的任务函数
{
    if(0==vGu8KeySec)
    {
        return;
    }

    switch(vGu8KeySec)
    {
       case 1:     //窗口切换的按键
            Gu8Wd++;  //窗口切换到下一个窗口
            if(Gu8Wd>3)  //一共 3 个窗口。切换第 3 个窗口之后，继续返回到第 1 个窗口
            {
                Gu8Wd=1;   //返回到第 1 个窗口
            }
            Gu8WdUpdate=1;  //整屏更新一次显示

            vGu8BeepTimerFlag=0;
            vGu16BeepTimerCnt=VOICE_TIME;  //蜂鸣器发出“滴”一声
            vGu8BeepTimerFlag=1;

            vGu8KeySec=0;
            break;

       case 2:     //累加的按键
            switch(Gu8Wd)  //以窗口选择 Gu8Wd 为支点，去编辑对应的数据。又一次用到 switch 语句
            {
```

```
            case 1:     //在第 1 个窗口下编辑 Gu8SetData_1 数据
                Gu8SetData_1++;
                if(Gu8SetData_1>99) //把最大范围限定在 99
                {
                    Gu8SetData_1=99;
                }
                Gu8WdUpdate=1;  //整屏更新一次显示
                break;

            case 2:     //在第 2 个窗口下编辑 Gu8SetData_2 数据
                Gu8SetData_2++;
                if(Gu8SetData_2>99) //把最大范围限定在 99
                {
                    Gu8SetData_2=99;
                }
                Gu8WdUpdate=1;  //整屏更新一次显示
                break;

            case 3:     //在第 3 个窗口下编辑 Gu8SetData_3 数据
                Gu8SetData_3++;
                if(Gu8SetData_3>99) //把最大范围限定在 99
                {
                    Gu8SetData_3=99;
                }
                Gu8WdUpdate=1;  //整屏更新一次显示
                break;
        }

        vGu8BeepTimerFlag=0;
        vGu16BeepTimerCnt=VOICE_TIME;  //蜂鸣器发出“滴”一声
        vGu8BeepTimerFlag=1;

        vGu8KeySec=0;
        break;

    case 3:     //递减的按键
        switch(Gu8Wd)  //以窗口选择 Gu8Wd 为支点，去编辑对应的数据。又一次用到 switch 语句
        {
            case 1:     //在第 1 个窗口下编辑 Gu8SetData_1 数据
                if(Gu8SetData_1>0) //把最小范围限定在 0
                {
                    Gu8SetData_1--;
                }
                Gu8WdUpdate=1;  //整屏更新一次显示
```

```
                    break;

                case 2:     //在第 2 个窗口下编辑 Gu8SetData_2 数据
                    if(Gu8SetData_2>0) //把最小范围限定在 0
                    {
                        Gu8SetData_2--;
                    }
                    Gu8WdUpdate=1;  //整屏更新一次显示
                    break;

                case 3:     //在第 3 个窗口下编辑 Gu8SetData_3 数据
                    if(Gu8SetData_3>0) //把最小范围限定在 0
                    {
                        Gu8SetData_3--;
                    }
                    Gu8WdUpdate=1;  //整屏更新一次显示
                    break;
            }

            vGu8BeepTimerFlag=0;
            vGu16BeepTimerCnt=VOICE_TIME;  //蜂鸣器发出“滴”一声
            vGu8BeepTimerFlag=1;

            vGu8KeySec=0;
            break;
    }
}

void DisplayTask(void) //数码管显示的上层任务函数
{
    switch(Gu8Wd)  //以窗口选择 Gu8Wd 为支点，去执行对应的窗口显示函数。又一次用到 switch 语句
    {
        case 1:
            Wd1();   //窗口 1 显示函数
            break;
        case 2:
            Wd2();   //窗口 2 显示函数
            break;
        case 3:
            Wd3();   //窗口 3 显示函数
            break;
    }

}
```

```
void Wd1(void)   //窗口 1 显示函数
{
    //需要借用的中间变量，用来拆分数据位。
    static unsigned char Su8Temp_4,Su8Temp_3,Su8Temp_2,Su8Temp_1; //需要借用的中间变量

    if(1==Gu8WdUpdate) //如果需要整屏更新
    {
          Gu8WdUpdate=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

          Su8Temp_4=1;  //窗口“1”
          Su8Temp_3=11; //横杠“-”
          Su8Temp_2=Gu8SetData_1/10%10; //十位数值
          Su8Temp_1=Gu8SetData_1/1%10;  //个位数值

          //上面先分解数据之后，再过渡需要显示的数据到底层驱动变量里，让过渡的时间越短越好
          vGu8Display_Righ_4=Su8Temp_4;  //过渡需要显示的数据到底层驱动变量
          vGu8Display_Righ_3=Su8Temp_3;
          vGu8Display_Righ_2=Su8Temp_2;
          vGu8Display_Righ_1=Su8Temp_1;

          //不显示任何一个小数点
          vGu8Display_Righ_Dot_4=0;
          vGu8Display_Righ_Dot_3=0;
          vGu8Display_Righ_Dot_2=0;
          vGu8Display_Righ_Dot_1=0;
    }
}

void Wd2(void)   //窗口 2 显示函数
{
    //需要借用的中间变量，用来拆分数据位。
    static unsigned char Su8Temp_4,Su8Temp_3,Su8Temp_2,Su8Temp_1; //需要借用的中间变量

    if(1==Gu8WdUpdate) //如果需要整屏更新
    {
          Gu8WdUpdate=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

          Su8Temp_4=2;  //窗口“2”
          Su8Temp_3=11; //横杠“-”
          Su8Temp_2=Gu8SetData_2/10%10; //十位数值
          Su8Temp_1=Gu8SetData_2/1%10;  //个位数值

          //上面先分解数据之后，再过渡需要显示的数据到底层驱动变量里，让过渡的时间越短越好
```

```
        vGu8Display_Righ_4=Su8Temp_4;  //过渡需要显示的数据到底层驱动变量
        vGu8Display_Righ_3=Su8Temp_3;
        vGu8Display_Righ_2=Su8Temp_2;
        vGu8Display_Righ_1=Su8Temp_1;

        //不显示任何一个小数点
        vGu8Display_Righ_Dot_4=0;
        vGu8Display_Righ_Dot_3=0;
        vGu8Display_Righ_Dot_2=0;
        vGu8Display_Righ_Dot_1=0;
    }
}

void Wd3(void)   //窗口 3 显示函数
{
    //需要借用的中间变量，用来拆分数据位。
    static unsigned char Su8Temp_4,Su8Temp_3,Su8Temp_2,Su8Temp_1; //需要借用的中间变量

    if(1==Gu8WdUpdate) //如果需要整屏更新
    {
        Gu8WdUpdate=0; //及时清零，只更新一次显示即可，避免一直进来更新显示

        Su8Temp_4=3;  //窗口“3”
        Su8Temp_3=11; //横杠“-”
        Su8Temp_2=Gu8SetData_3/10%10; //十位数值
        Su8Temp_1=Gu8SetData_3/1%10;  //个位数值

        //上面先分解数据之后，再过渡需要显示的数据到底层驱动变量里，让过渡的时间越短越好
        vGu8Display_Righ_4=Su8Temp_4;  //过渡需要显示的数据到底层驱动变量
        vGu8Display_Righ_3=Su8Temp_3;
        vGu8Display_Righ_2=Su8Temp_2;
        vGu8Display_Righ_1=Su8Temp_1;

        //不显示任何一个小数点
        vGu8Display_Righ_Dot_4=0;
        vGu8Display_Righ_Dot_3=0;
        vGu8Display_Righ_Dot_2=0;
        vGu8Display_Righ_Dot_1=0;
    }
}

void KeyScan(void)  //按键底层的驱动扫描函数，放在定时中断函数里
{
   static unsigned char Su8KeyLock1;
```

```
static unsigned int  Su16KeyCnt1;
static unsigned char Su8KeyLock2;
static unsigned int  Su16KeyCnt2;
 static unsigned char Su8KeyLock3;
static unsigned int  Su16KeyCnt3;

if(0!=KEY_INPUT1)
{
   Su8KeyLock1=0;
   Su16KeyCnt1=0;
}
else if(0==Su8KeyLock1)
{
   Su16KeyCnt1++;
   if(Su16KeyCnt1>=KEY_FILTER_TIME)
   {
      Su8KeyLock1=1;
      vGu8KeySec=1;
   }
}

if(0!=KEY_INPUT2)
{
   Su8KeyLock2=0;
   Su16KeyCnt2=0;
}
else if(0==Su8KeyLock2)
{
   Su16KeyCnt2++;
   if(Su16KeyCnt2>=KEY_FILTER_TIME)
   {
      Su8KeyLock2=1;
      vGu8KeySec=2;
   }
}

if(0!=KEY_INPUT3)
{
   Su8KeyLock3=0;
   Su16KeyCnt3=0;
}
else if(0==Su8KeyLock3)
{
   Su16KeyCnt3++;
```

```
      if(Su16KeyCnt3>=KEY_FILTER_TIME)
      {
         Su8KeyLock3=1;
         vGu8KeySec=3;
      }
   }

}

void DisplayScan(void)    //数码管底层的驱动扫描函数，放在定时中断函数里
{
    static unsigned char Su8GetCode;
    static unsigned char Su8ScanStep=1;

    if(0==vGu16ScanTimerCnt)
    {


        P0=0x00;
        P1_0=1;
        P1_1=1;
        P1_2=1;
        P1_3=1;

        switch(Su8ScanStep)
        {
           case 1:
                Su8GetCode=Cu8DigTable[vGu8Display_Righ_1];

                if(1==vGu8Display_Righ_Dot_1)
                {
                    Su8GetCode=Su8GetCode|0x80;
                }
                P0=Su8GetCode;
                P1_0=0;
                P1_1=1;
                P1_2=1;
                P1_3=1;
               break;

           case 2:
                Su8GetCode=Cu8DigTable[vGu8Display_Righ_2];
                if(1==vGu8Display_Righ_Dot_2)
                {
```

```
                Su8GetCode=Su8GetCode|0x80;
            }
            P0=Su8GetCode;
            P1_0=1;
            P1_1=0;
            P1_2=1;
            P1_3=1;
            break;

        case 3:
            Su8GetCode=Cu8DigTable[vGu8Display_Righ_3];
            if(1==vGu8Display_Righ_Dot_3)
            {
                Su8GetCode=Su8GetCode|0x80;
            }
            P0=Su8GetCode;
            P1_0=1;
            P1_1=1;
            P1_2=0;
            P1_3=1;
            break;

        case 4:
            Su8GetCode=Cu8DigTable[vGu8Display_Righ_4];
            if(1==vGu8Display_Righ_Dot_4)
            {
                Su8GetCode=Su8GetCode|0x80;
            }
            P0=Su8GetCode;
            P1_0=1;
            P1_1=1;
            P1_2=1;
            P1_3=0;
            break;

    }

    Su8ScanStep++;
    if(Su8ScanStep>4)
    {
        Su8ScanStep=1;
    }

    vGu8ScanTimerFlag=0;
```

```
        vGu16ScanTimerCnt=SCAN_TIME;
        vGu8ScanTimerFlag=1;
    }
}


void VoiceScan(void) //蜂鸣器的驱动函数
{

      static unsigned char Su8Lock=0;

      if(1==vGu8BeepTimerFlag&&vGu16BeepTimerCnt>0)
      {
          if(0==Su8Lock)
          {
               Su8Lock=1;
               BeepOpen();
          }
          else
          {

               vGu16BeepTimerCnt--;

               if(0==vGu16BeepTimerCnt)
               {
                   Su8Lock=0;
                   BeepClose();
               }

          }
      }
}

void BeepOpen(void)
{
    P3_4=0;
}

void BeepClose(void)
{
    P3_4=1;
}

void T0_time() interrupt 1
```

```
{
    VoiceScan();    //蜂鸣器的驱动函数
    KeyScan();      //按键底层的驱动扫描函数
    DisplayScan();  //数码管底层的驱动扫描函数

    if(1==vGu8ScanTimerFlag&&vGu16ScanTimerCnt>0)
    {
        vGu16ScanTimerCnt--;  //递减式的软件定时器
    }

    TH0=0xfc;
    TL0=0x66;
}


void SystemInitial(void)
{
    P0=0x00;
    P1_0=1;
    P1_1=1;
    P1_2=1;
    P1_3=1;

    TMOD=0x01;
    TH0=0xfc;
    TL0=0x66;
    EA=1;
    ET0=1;
    TR0=1;
}

void Delay(unsigned long u32DelayTime)
{
    for(;u32DelayTime>0;u32DelayTime--);
}

void PeripheralInitial(void)
{

}
```