口令触发模式

1. 学习目标

本次课程我们主要学习使用89C52RC型号的51单片机和语音识别模块V3版实现口令触发模式。

2. 课前准备

语音模块采用的是I2C通讯，将模块的SDA，SCL分别连接51板子的P2.0和P2.1引脚。VCC和GND分别连接51板子的5V和GND。

3. 模块协议表

	寄存器	备注
添加词组	0x01	数据长度+词组识别序号（可选0-254）+词组拼音+数据0。如设置识别词“ya bo”，序号为0，那么发送数据为:6|0|y|a| |b||o|0，6是后续的序号+词组拼音的长度，第一个0为识别序号，之后的拼音字符为识别词组，最后的0作为结束识别数据。拼音字符总长度不超过79，识别序号0-4默认用于口令，红灯，绿灯，蓝灯，关灯，255用于识别默认返回值表示无结果，所以不可将识别序号设置为0xff或者255。
模式	0x02	0 循环检测模式 1 口令模式 2 按键检测模式,口令模式下，词组识别序号为0的词作为唤醒口令，识别到唤醒口令蜂鸣器会响一声，按键检测模式下，当检测到按键按下，蜂鸣器会响一声。默认设置为循环检测模式
设置RGB灯颜色	0x03	需要发三位，1位R灯的值，2位G灯的值，3位B灯的值，值的范围均为0-255
麦克风灵敏度	0x04	灵敏度设置地址，灵敏度可设置为0x00-0x7f，值越高越容易检测但是越容易误判，供电为5V的时候建议设置值为0x40-0x55,默认值为0x40。
清除掉电缓存	0x05	清除掉电缓存操作地址，录入信息前均要清除下缓存区信息，0x00-0xff随意哪一个值都可以清除缓存。
按键模式启动	0x06	用于按键模式下，写1启动识别，其他值不识别。
识别提示音开关	0x07	口令识别到的声音常开无法关闭，所有模式识别到结果时蜂鸣器响两声，该提示可以通过该寄存器设置是否开启，设置为1开启，设置为0关闭。默认为1开启。
识别结果存放处	0x08	只读，用于存放识别结果，默认值为0xff表示无结果
蜂鸣器控制寄存器	0x09	蜂鸣器控制寄存器，写1开启，写0关闭
词组数目寄存器	0x0a	只读，返回缓存区存入的词组数
固件版本号寄存器	0x0b	只读，返回单片机固件版本号
模块忙闲寄存器	0x0c	只读，返回“添加词组”、“模式设置”、“清楚掉电缓存”的忙闲状态，该三个操作的寄存器操作完后读取忙闲寄存器以得知操作是否完成。空闲返回0，操作还在进行中返回对应寄存器的值即添加词组正在进行返回1、模式设置正在进行返回2、清除缓存区正在进行返回5。该3个操作后必须读取该寄存器待空闲后才可以进行其他操作。

由模块协议表可知，通过对寄存器0x02写入1可设置模块为口令触发模式。

4. 程序

定义模块的设备地址和寄存器地址便于后续操作

```
#define I2C_ADDR                0x1e   //语音识别模块地址，模块地址为0x0f由于最右端为读写位，所以需要左移以为为0x1e
#define ASR_ADD_WORD_ADDR       0x01   //词条添加地址
#define ASR_MODE_ADDR           0x02   //识别模式设置地址，值为0-2，0:循环识别模式 1:口令模式 ,2:按键模式，默认为循环检测
#define ASR_RGB_ADDR            0x03   //RGB灯设置地址,需要发两位，第一个直接为灯号1：蓝 2:红 3：绿 ，
                                       //第二个字节为亮度0-255，数值越大亮度越高
#define ASR_REC_GAIN            0x04   //识别灵敏度设置地址，灵敏度可设置为0x00-0x7f,值越高越容易检测但是越容易误判，
                                       //建议设置值为0x40-0x55,默认值为0x40
#define ASR_CLEAR_ADDR          0x05   //清除掉电缓存操作地址，录入信息前均要清除下缓存区信息

#define ASR_KEY_FLAG            0x06   //用于按键模式下，设置启动识别模式
#define ASR_VOICE_FLAG          0x07   //用于设置是否开启识别结果提示音
#define ASR_RESULT              0x08  //识别结果存放地址
#define ASR_BUZZER              0x09 //蜂鸣器控制写1开启，写0关闭
#define ASR_NUM_CLECK           0x0a //录入词条数目校验
#define FIRMWARE_VERSION        0x0b //读取固件版本
#define ASR_BUSY                0x0c//忙闲标志
```

I2C 单字节写函数，向模块写入一个字节数据。

```
bit I2C_ByteWrite_alone(unsigned char date)
{

    Start_I2c();                  //启动总线
    I2C_SendByte(I2C_ADDR);       //发送器件地址
    if(ack==0)return(0);
    I2C_SendByte(date);     //发送数据
    if(ack==0)return(0);

    Stop_I2c();                 //结束总线

    return(1);

}
```

I2C 字节写入函数，写一个字节到模块中指定的寄存器。

```
void I2C_ByteWrite(unsigned char reg_addr,unsigned char dat)
{
     I2C_ByteWrite_alone(reg_addr);
     I2C_ByteWrite_alone(dat);
}
```

I2C 字节读取函数，通过先对模块写入要读取的寄存器值，即这里要读取的是检测结果所以写入的是结果存放寄存器的地址值，然后再对模块进行读取操作获取识别到的值。

```
unsigned char I2C_BufferRead(unsigned char date)
{
    unsigned char dat;
    Start_I2c();                    //启动总线
    I2C_SendByte(I2C_ADDR);         //发送器件地址
    if(ack==0)return(0);
    I2C_SendByte(date);     //发送器件地址
    if(ack==0)return(0);
    Stop_I2c();
    delay(50);

    Start_I2c();                        //启动总线
    I2C_SendByte(I2C_ADDR+1);       //发送器件地址
    if(ack==0)return(0);
        delay(1);
    dat=I2C_RcvByte();              //读取数据

    Ack_I2c(1);             //发送非应答信号
    Stop_I2c();             //结束总线

    return(dat);
}
```

RGB 灯设置函数，通过先对模块写入操作的寄存器值，即 RGB 灯操作寄存器值，紧接着连续发送 3 个字节数据，分别为 R、G、B 的值。

```
void RGB_Set(unsigned char R,unsigned char G,unsigned char B)
{

     I2C_ByteWrite_alone(ASR_RGB_ADDR);
     I2C_ByteWrite_alone(R);
     I2C_ByteWrite_alone(G);
     I2C_ByteWrite_alone(B);
}
```

添加词条函数，添加词条的序号和添加词条的拼音，通过先写入要操作的词条寄存器地址，接着计算词组号加词条字符串数目，然后写入数据长度，紧接着发送词组号和词组字符串，最后发送 0 字节作为结束符。

```
void AsrAddWords(unsigned char idNum,unsigned char * words)
{
      int i = 0;
      unsigned char date_length = strlen(words)+2;
      unsigned char str_length = strlen(words);
      I2C_ByteWrite_alone(ASR_ADD_WORD_ADDR);        //发送寄存器地址
      I2C_ByteWrite_alone(date_length);             //发送数据长度
      I2C_ByteWrite_alone(idNum);
      for(i = 0;i<str_length;i++)
      {
         I2C_ByteWrite_alone(words[i]);
      }
       I2C_ByteWrite_alone(0);


}
```

忙闲等待函数，等待模块返回 0 表示设置完毕已空闲。“添加词组”、“模式设置”、“清空缓存区”等寄存器操作完后模块需要时间进行设置，需要等待设置完毕后才可以进行其他操作。

```
/*******************************
       忙闲等待取函数
*******************************/
void BusyWait(void)
{

  unsigned char busy_flag = 0xff;
  while(busy_flag != 0)
  {
      busy_flag = I2C_BufferRead(ASR_BUSY);
      delay(500);
    }

  }
```

实验前配置，对模块清除掉电缓存区后，设置模式，设置识别词语，并检查录入的词语数是否对应。模块具有对模式和录入词组的掉电保存机制，所以如果不需要对模块里面已经录入的词组和模式进行修改可以将#if 1 修

改为#if 0 不执行这部分代码已节约初始化时间。

```
#if 1
    I2C_ByteWrite(ASR_CLEAR_ADDR,0x40);//清除掉电保存区,录入前需要清除掉电保存区
    BusyWait();
    I2C_ByteWrite(ASR_MODE_ADDR,1);//设置模式为口令模式
    BusyWait();
    AsrAddWords(0,"xiao ya");
    BusyWait();
    AsrAddWords(1,"hong deng");
    BusyWait();
    AsrAddWords(2,"lv deng");
    BusyWait();
    AsrAddWords(3,"lan deng");
    BusyWait();
    AsrAddWords(4,"guan deng");
    BusyWait();
    AsrAddWords(5,"yi hao deng");
    BusyWait();
    AsrAddWords(6,"e hao deng");
    BusyWait();
    AsrAddWords(7,"san hao deng");
    BusyWait();
    AsrAddWords(8,"si hao deng");
    BusyWait();
    AsrAddWords(9,"wu hao deng");
    BusyWait();

     while(cleck != 10)
    {
      cleck = I2C_BufferRead(ASR_NUM_CLECK);
      delay(500);
      }
#endif
    I2C_ByteWrite(ASR_REC_GAIN,0x55);//设置灵敏度


    RGB_Set(255,255,255);
    I2C_ByteWrite(ASR_BUZZER,0x01);//开启蜂鸣器
    delay(1000);
    I2C_ByteWrite(ASR_BUZZER,0x00);//开启蜂鸣器
    RGB_Set(0,0,0);
```

循环读取模块识别到的数值，默认值为 0xff 表示没有识别到结果。

```
result = I2C_BufferRead(ASR_RESULT);
if(result == 5)
{
    led1=0;
}
else if(result == 6)
{
    led2 = 0;
}
else if(result == 7)
{
    led3 = 0;
}
else if(result == 8)
{
    led4 = 0;
}
else if(result == 9)
{
    led5 = 0;
}
else if(result == 4)
{
    led1 = 1;
    led2 = 1;
    led3 = 1;
    led4 = 1;
    led5 = 1;
}

delay(500);
```

5. 实验现象

程序下载后运行，如果设置为条件为 1，则需要等待清除缓存区和录入词组的时间，等待设置完毕后模块的 RGB 灯会亮起白色 1s 和蜂鸣器鸣笛 1s。之后通过口令小亚触发识别，当识别到录入词组后会读取到词组对应的值。如果之前已经录入过了且无需修改可以把条件改为 0，可以跳过清除缓存和录入词条的时间。