参考程序

以下 MSP430 C 语言程序是在单片机上调试通过的程序中截取的部分代码，仅供大家参考，直接复制使用有可能出错，请根据自己的任务要求情况参考利用。

1．4×4 矩阵键盘扫描

```
unsigned char Key_scan (void)
{
   char scan_sig[4] = {0x0E,0x0D,0x0B,0x07};
   char MSB, LSB, key_code;
   P1OUT = 0x00;   // P1 口低 4 位作为行线输出低电平
   temp = (~P2IN) & 0x0F;     // P2 口低 4 位作为列线输入
   if(temp)
   {
delayms(10);//延时，再次读取输入，消除抖动
temp = (~P2IN) & 0x0F;
if(temp)//  确实有键按下
{
   //按行扫描
   for (char i = 0;i < 4; i++)
   {
P1OUT = scan_sig[i];
delayms(5);
temp = ~P2IN;
MSB = temp<<4;
if (MSB)
{
   LSB = (~scan_sig[i]) & 0x0F;
   key_code = MSB | LSB;
   break;
}
   }
}
else
   key_code = 0x00;     //  无键按下
   }
   else
key_code = 0x00; //  无键按下

   return(key_code);
}
```

2. DS18B20 读写及数值处理程序

```
#define delay_us(x) __delay_cycles((long)(CPU_F*(double)x/1000000.0))
#define delay_ms(x) __delay_cycles((long)(CPU_F*(double)x/1000.0))

#define    DQ_1P2OUT |= BIT0      //DQ 置位
#define    DQ_0P2OUT &= ~BIT0     //DQ 复位
#define    DQ_IN      P2DIR &= ~BIT0 //设置 DQ 为输入
#define    DQ_OUT    P2DIR |= BIT0 //设置 DQ 为输出
#define    READ_DQ P2IN & BIT0//读 DQ 电平

unsigned char DS18B20_Reset()
{
unsigned char flag;
DQ_OUT;//DQ 为输出
DQ_1;
delay_us(1);
DQ_0;//拉低总线
delay_us(750);//480~700us
DQ_1;//释放总线
delay_us(30);//15~60us
DQ_IN;//设置高阻态
if(READ_DQ)
flag=0;//等待从机 DS18B20 应答（低电平有效）
else
flag=1;
delay_us(240);//延时达 240us，让 DS18B20 释放总线
return(flag);
}


/*************************写数据到 DS18B20 函数名称: DS18B20_WriteData()
*/
void DS18B20_WriteData(unsigned char wData)
{
unsigned char i;
DQ_OUT;//DQ 为输出
for(i=0;i<8;i++)
{
if(wData&0x01)//发送 1 位
{
DQ_0;
delay_us(2);
```

```
DQ_1;
delay_us(60);
}
else
{
DQ_0;
delay_us(60);
DQ_1;
delay_us(2);
}
wData>>=1;//准备下一位数据的传送
}
  DQ_1;//释放总线
}

/***************从 DS18B20 中读出数据
*函数名称: DS18B20_ReadData()******************/

unsigned char DS18B20_ReadData()
{
unsigned char i,TmepData;
for(i=0;i<8;i++)
{
DQ_OUT;//DQ 为输出
DQ_0;
  TmepData>>=1;//数据右移  先读最低位
delay_us(2);//2us
DQ_1;
delay_us(15);//2us

DQ_IN;

if(READ_DQ)
{
TmepData|=0x80;
}

delay_us(50);
}
     // DQ_1;
return(TmepData);//返回读到的数据
}
```

```
/**********************DS18B20 转换温度
*函数名称:DS18B20_Conert()**********************/

unsigned int DS18B20_Conert()
{
unsigned int TempData1,TempData2;
DS18B20_Reset();
DS18B20_WriteData(0xCC);//跳过 ROM
DS18B20_WriteData(0x44);//开始转换
DS18B20_Reset();
DS18B20_WriteData(0xCC);//跳过 ROM
DS18B20_WriteData(0xBE);//读取 RAM
TempData1=DS18B20_ReadData();//读低八位，LSByte,RAM0
TempData2=DS18B20_ReadData();//读高八位，MSByte,RAM1
    return(((TempData2<<8)| TempData1)*0.625);
}
```

3. OLED 显示程序

```
/*********************************************************************
****************
*    Copyright (c), 2013, HelTec Automatic Technology co.,LTD.
*All rights reserved.
* Description:128*64
* heltec.taobao.com
* Others: none;
*
* Function List:
*1. void delay(unsigned int z) --  延时函数,毫秒
* 2. void IIC_Start() --  开启 I2C 总线
* 3. void IIC_Stop() --  关闭 I2C 总线
* 4. void Write_IIC_Byte(unsigned char IIC_Byte) --  通过 I2C 总线写一个 byte 的数
* 5. void OLED_WrDat(unsigned char dat) --  向 OLED 屏写数据
* 6. void OLED_WrCmd(unsigned char cmd) --  向 OLED 屏写命令
* 7. void OLED_Set_Pos(unsigned char x, unsigned char y) --  设置显示坐标
* 8. void OLED_Fill(unsigned char bmp_dat) --  全屏显示(显示 BMP 图片时才会用
到此功能)
* 9. void OLED_CLS(void) --  复位/清屏
* 10. void OLED_Init(void) -- OLED 屏初始化程序，此函数应在操作屏幕之前最
先调旷
* 11. void OLED_P6x8Str(unsigned char x, y,unsigned char ch[]) -- 6x8 点整，用于
显示 ASCII 码的最小阵列，不太清晰
* 12. void OLED_P8x16Str(unsigned char x, y,unsigned char ch[]) -- 8x16 点整，用
于显示 ASCII 码，非常清晰
* 13.void OLED_P16x16Ch(unsigned char x, y, N) -- 16x16 点整，用于显示汉字的
最小阵列，可设置各种字体、加粗、倾斜、下划线
* 14.void Draw_BMP(unsigned char x0, y0,x1, y1,unsigned char BMP[]) -- 28x64 像
素的 BMP 位图在取字软件中算出字表，然后复制到 codetab 中，此函数调用即
可
*15.void OLED_1(unsigned char x,unsigned char y,unsigned char key)显示数字 0~9
*16.void OLED_2(unsigned char x,unsigned char y)显示温度符号“℃”
*History:none;
*
*********************************************************************
***************/

//P1.7 为 OLED 数据信号，P1.6 为 OLED 时钟信号
#define SDA_H      P1OUT |=    BIT7    //SDA = 1
#define SDA_L      P1OUT &=~ BIT7    //SDA = 0
#define SCL_H      P1OUT |=    BIT6    //SCL = 1
```

```
#define SCL_L      P1OUT &=~ BIT6    //SCL = 0
#define  Brightness    0xCF
#define X_WIDTH    128
#define Y_WIDTH    64
/**********************OLED 延时函数 ************************/
void delay(unsigned int z)
{
      unsigned int x,y;
      for(x=z;x>0;x--)
            for(y=1330;y>0;y--);
}


void IIC_Start()
{
     SCL_H;
     SDA_H;
     SDA_L;
     SCL_L;
}

void IIC_Stop()
{
     SCL_L;
     SDA_L;
     SCL_H;
     SDA_H;
}


void Write_IIC_Byte(unsigned char IIC_Byte)
{
      unsigned char i;
      for(i=0;i<8;i++)
      {
            if(IIC_Byte & 0x80)
                  SDA_H;
            else
                  SDA_L;
            SCL_H;
            SCL_L;
            IIC_Byte<<=1;
      }
      SDA_H;
```

```
    SCL_H;
    SCL_L;
}


void OLED_WrDat(unsigned char IIC_Data)
{
    IIC_Start();
    Write_IIC_Byte(0x78);
    Write_IIC_Byte(0x40);            //write data
    Write_IIC_Byte(IIC_Data);
    IIC_Stop();
}

void OLED_WrCmd(unsigned char IIC_Command)
{
    IIC_Start();
    Write_IIC_Byte(0x78);//Slave address,SA0=0
    Write_IIC_Byte(0x00);            //write command
    Write_IIC_Byte(IIC_Command);
    IIC_Stop();
}

void OLED_Set_Pos(unsigned char x, unsigned char y)
{
    OLED_WrCmd(0xb0+y);
    OLED_WrCmd(((x&0xf0)>>4)|0x10);
    OLED_WrCmd((x&0x0f)|0x01);
}

void OLED_Fill(unsigned char bmp_dat)
{
    unsigned char y,x;
    for(y=0;y<8;y++)
    {
        OLED_WrCmd(0xb0+y);
        OLED_WrCmd(0x01);
        OLED_WrCmd(0x10);
        for(x=0;x<X_WIDTH;x++)
        OLED_WrDat(bmp_dat);
    }
}

void OLED_CLS(void)
```

```
{
    unsigned char y,x;
    for(y=0;y<8;y++)
    {
        OLED_WrCmd(0xb0+y);
        OLED_WrCmd(0x01);
        OLED_WrCmd(0x10);
        for(x=0;x<X_WIDTH;x++)
        OLED_WrDat(0);
    }
}

void OLED_Init(void)
{
    delay(500);//初始化之前的延时很重要!
    OLED_WrCmd(0xae);//--turn off oled panel
    OLED_WrCmd(0x00);//---set low column address
    OLED_WrCmd(0x10);//---set high column address
    OLED_WrCmd(0x40);//--set start line address  Set Mapping RAM Display Start Line (0x00~0x3F)
    OLED_WrCmd(0x81);//--set contrast control register
    OLED_WrCmd(Brightness); // Set SEG Output Current Brightness
    OLED_WrCmd(0xa1);//--Set SEG/Column Mapping 0xa0 左右反置 0xa1 正常
    OLED_WrCmd(0xc8);//Set COM/Row Scan Direction  0xc0 上下反置 0xc8 正常
    OLED_WrCmd(0xa6);//--set normal display
    OLED_WrCmd(0xa8);//--set multiplex ratio(1 to 64)
    OLED_WrCmd(0x3f);//--1/64 duty
    OLED_WrCmd(0xd3);//-set display offset Shift Mapping RAM Counter (0x00~0x3F)
    OLED_WrCmd(0x00);//-not offset
    OLED_WrCmd(0xd5);//--set display clock divide ratio/oscillator frequency
    OLED_WrCmd(0x80);//--set divide ratio, Set Clock as 100 Frames/Sec
    OLED_WrCmd(0xd9);//--set pre-charge period
    OLED_WrCmd(0xf1);//Set Pre-Charge as 15 Clocks & Discharge as 1 Clock
    OLED_WrCmd(0xda);//--set com pins hardware configuration
    OLED_WrCmd(0x12);
    OLED_WrCmd(0xdb);//--set vcomh
    OLED_WrCmd(0x40);//Set VCOM Deselect Level
    OLED_WrCmd(0x20);//-Set Page Addressing Mode (0x00/0x01/0x02)
    OLED_WrCmd(0x02);//
    OLED_WrCmd(0x8d);//--set Charge Pump enable/disable
    OLED_WrCmd(0x14);//--set(0x10) disable
    OLED_WrCmd(0xa4);// Disable Entire Display On (0xa4/0xa5)
```

```
    OLED_WrCmd(0xa6);// Disable Inverse Display On (0xa6/a7)
    OLED_WrCmd(0xaf);//--turn on oled panel
    OLED_Fill(0x00); //初始清屏
    OLED_Set_Pos(0,0);
}
/***************功能描述：显示一组标准 ASCII 字符，显示的坐标（x,y），y 为页范围*****************/
void OLED_P6x8Str(unsigned char x,unsigned char y,unsigned char ch[])
{
    unsigned char c=0,i=0,j=0;
    while (ch[j]!='\0')
    {
        c =ch[j]-32;
        if(x>126){x=0;y++;}
        OLED_Set_Pos(x,y);
        for(i=0;i<6;i++)
        OLED_WrDat(F6x8[c][i]);   // F6x8 为字符对应的编码矩阵，可通过字模软件得到
        x+=6;
        j++;
    }
}

void OLED_P8x16Str(unsigned char x, unsigned char y, unsigned char ch[])
{
    unsigned char c=0,i=0,j=0;
    while (ch[j]!='\0')
    {
        c =ch[j]-32;
        if(x>120){x=0;y++;}
        OLED_Set_Pos(x,y);
        for(i=0;i<8;i++)
        OLED_WrDat(F8X16[c*16+i]);
        OLED_Set_Pos(x,y+1);
        for(i=0;i<8;i++)
        OLED_WrDat(F8X16[c*16+i+8]);
        x+=8;
        j++;
    }
}

/*void OLED_P8x16Ch (unsigned char x, unsigned char y, unsigned int n)
{
    unsigned char wm=0;
    unsigned int adder=16*n;
```

```
    OLED_Set_Pos(x , y);
    for(wm = 0;wm < 8;wm++)
    {
        OLED_WrDat(F8X16[adder]);
        adder += 1;
    }
OLED_Set_Pos(x,y + 1);
    for(wm = 0;wm < 8;wm++)
    {
        OLED_WrDat(F8X16[adder]);
        adder += 1;
    }

}*/

void OLED_P16x16Ch(unsigned char x, unsigned char y, unsigned int n)
{
    unsigned char wm=0;
    unsigned int adder=32*n;
    OLED_Set_Pos(x , y);
    for(wm = 0;wm < 16;wm++)
    {
        OLED_WrDat(F16x16[adder]);
        adder += 1;
    }
    OLED_Set_Pos(x,y + 1);
    for(wm = 0;wm < 16;wm++)
    {
        OLED_WrDat(F16x16[adder]);
        adder += 1;
    }
}


void Draw_BMP(unsigned char x0, unsigned char y0,unsigned char x1,unsigned char
y1,unsigned char BMP[])
{
    unsigned int j=0;
    unsigned char x,y;

  if(y1%8==0) y=y1/8;
  else y=y1/8+1;
    for(y=y0;y<y1;y++)
    {
```

```
            OLED_Set_Pos(x0,y);
for(x=x0;x<x1;x++)
      {
            OLED_WrDat(BMP[j++]);
      }
      }
}

void OLED_1(unsigned char x,unsigned char y,unsigned char key){
  unsigned char i,j;
  for(i=0;i<2;i++){
  OLED_Set_Pos(x , y+i);
for(j=0;j<8;j++){
OLED_WrDat(F10X16[key][i*8+j]);
     }

  }

}

void OLED_2(unsigned char x,unsigned char y){
  unsigned char i,j;
  for(i=0;i<2;i++){
  OLED_Set_Pos(x , y+i);
for(j=0;j<16;j++){
OLED_WrDat(F10X16[12+i][j]);
     }

  }
}
```

4. A/D 采样程序

```
void ADC10_Init()
{
ADC10CTL0 &=~ENC;
ADC10CTL0|=ADC10SHT_2+ADC10ON+ADC10IE+REFON+REF2_5V+SREF_1
+MSC;    //开 ADC10 内核，取内部参考电压 2.5v，使采样时间为
16xADC10CLK(增大采样时间可以保证采样的准确性)

ADC10CTL1 |= INCH_2+CONSEQ_1;
ADC10DTC1|=0x03; //一共采样 3 次
ADC10AE0 |= BIT2+BIT1+BIT0;   //   ADC10CTL0 |= ENC;//开启使能,开始转换
}


//这是 main 函数中开始采样
  while(1)
{
   ADC10CTL0&=~ENC; //关闭采样使能
   while(ADC10CTL1&BUSY); //检测 AD 是否繁忙
   ADC10CTL0|=ENC+ADC10SC; //启动 ADC
   ADC10SA=(unsigned int)a; //获取 a[]的首地址。首先对 A2 A1、A0 采样，放入
a[0]和 a[1] A2 中。
   ADval = a[2];
   MidAdVal = a[1];
   delay = a[0]/20;
```

5. ADXL345 读写程序

```
//×××××××I2C 模拟通信协议××××××××//
#define SDA_1      P1OUT |=    BIT1    //SDA = 1
#define SDA_0      P1OUT &=~ BIT1    //SDA = 0
#define SCL_1      P1OUT |=    BIT0    //SCL = 1
#define SCL_0      P1OUT &=~ BIT0    //SCL = 0      P1.0 SCL      P1.1 SDA

#define SDA_IN    P1DIR &=~ BIT1    //I/O 口为输入,SDA_1
#define SDA_OUT P1DIR |=    BIT1    //I/0 口为输出

#define SCL_IN    P1DIR &=~ BIT0    //I/O 口为输入,SCL_1
#define SCL_OUT P1DIR |=    BIT0    //I/0 口为输出

void Init__IIC(void)
{
     P1SEL &= ~BIT0;
     P1SEL &= ~BIT1;    //选择 P1.1(SDA) P1.0(SCL)为 I/O 端口
     SCL_OUT; //P1.0(SCL)为输出
     SDA_IN;    //p1.1(SDA)为输入
     SCL_0;
}

//×××××开始××××××//
void Start(void) //SCL 为高电平期间，SDA 产生一个下降沿
{
   SDA_OUT;
   SDA_1;
   delay_us(5);
   SCL_1;
   delay_us(5);
   SDA_0;
   delay_us(5);
   SCL_0;
   delay_us(5);
}

//×××××停止×××××//
void Stop(void)     //SCL 高电平期间，SDA 产生一个上升沿
{
   SDA_OUT;
   SCL_0;
   delay_us(5);
```

```
    SDA_0;
    delay_us(5);
    SCL_1;
    delay_us(5);
    SDA_1;
    delay_us(5);
}

//×××××应答信号×××××//
void Ack(void) //IIC 总线应答//////SCL 为高电平时，SDA 为低电平
{
    SDA_OUT;
    SDA_0;
    delay_us(5);
    SCL_1;
    delay_us(5);
    SCL_0;
    delay_us(5);
    SDA_1;
}

void NoAck(void) //IIC 总线无应答//SDA 维持高电平，SCL 产生一个脉冲
{
    SDA_OUT;
    SDA_1;
    delay_us(5);
    SCL_1;
    delay_us(5);
    SCL_0;
    delay_us(5);
}
//IIC 总线检验应答（SCL 高电平期间，读 SDA 的值）
//返回值：IIC 应答的值  0：应答    1：无应答
//先转换端口为读，这样接收器拉底电平的应答才能确定拉底，否则
//电平被拉至高电平，从而造成无应答的情形

uchar TestACK(void)
{
    SDA_IN;     //SDA 设为输入
    delay_us(5);
    SCL_1;
    delay_us(5);
    ErrorBit=(P1IN & BIT1)>>1;
    delay_us(5);
```

```
    SCL_0;
    delay_us(5);
    //SDA_OUT;
    return(ErrorBit);
}

//×××××××写一个字节×××××××//
void WriteByte(uchar WriteData)
{
    uchar i;
    SDA_OUT;
    for (i=0; i<8; i++)
    {
SDA_OUT;
if (((WriteData >> 7) & 0x01) == 0x01)   //判断发送位，送数据到数据线上，从高位开始发送 bit
{
    SDA_1;
}
else
{
    SDA_0;
}
delay_us(10);
SCL_1;   //置时钟信号为高电平，使数据有效
delay_us(5);
SCL_0;
delay_us(10);
WriteData = WriteData << 1;
    }
    SDA_IN;
    delay_us(5);
}
//××××××××××读一个字节×××××××××××//
unsigned char ReadByte()
{
    SDA_IN; //置数据线为输入方向
    uchar i;
    uchar q0 ;
    uchar byte = 0;
    for (i=0; i<8; i++)
    {
SCL_1;     //置时钟为高电平，使数据线数据有效
delay_us(5);
```

```
byte=byte<<1;
SDA_IN;
q0=(P1IN & BIT1);
if (q0 == BIT1 ) byte=(byte|0x01);   //将数据存入 byte
delay_us(10);
SCL_0;
delay_us(10);
   }
   return(byte);
}

uchar Single_Write_ADXL345(uchar REG_Address,uchar REG_data)
{
   //uchar i=0;
   Start();
   WriteByte(0xA6);     //发送设备地址+写信号
   if(TestACK()!=0)//检验应答
   return 1; //若应答错误，则推出函数，返回错误
   WriteByte(REG_Address);     //内部寄存器地址
   if(TestACK()!=0)
   return 1;
   WriteByte(REG_data);   //内部寄存器数据
   if(TestACK()!=0)
   return 1;
   Stop();
   return 0;
}
//××××××××××连续读取××××××××××//
uchar ReadWords_ACC()
{
   uchar i;
   Start();
   WriteByte(0xA6);     //发送设备地址+写信号
   if(TestACK()!=0)
   return 1;
   WriteByte(0x32);
   if(TestACK()!=0)
   return 1;
   Start();//再次启动 IIC 总线
   WriteByte(0xA7); //读取
   if(TestACK()!=0)
   return 1;
   for (i=0; i<6; i++)
   {
```

```
acc[i]    = ReadByte();
  if(i==5)
  {
     NoAck();
  }
  else
  {
     Ack();
  }
   }
   Stop();
   return 0;
}
//***************加速度计ADXL345初始化××××××××××××××××××//
void Init_ADXL345()
{

    Single_Write_ADXL345(0x31,0x0B);    //测量范围,正负16g，13位模式
    Single_Write_ADXL345(0x2C,0x08);    //速率设定为12.5 参考pdf13页
    Single_Write_ADXL345(0x2D,0x08);    //选择电源模式    参考pdf24页
    Single_Write_ADXL345(0x2E,0x80);    //使能 DATA_READY 中断
    Single_Write_ADXL345(0x1E,0x00);    //X 偏移量 根据测试传感器的状态写入pdf29页
    Single_Write_ADXL345(0x1F,0x00);    //Y 偏移量 根据测试传感器的状态写入pdf29页
    Single_Write_ADXL345(0x20,0x05);    //Z 偏移量 根据测试传感器的状态写入pdf29页

}
```