

零死角玩转STM32

与野火同行 乐意惬意无边



原创教程，完全开源。



由浅入深，结合实操。



通俗易懂，详尽解读。



配套板子，全面玩转。



强强联合，不断更新。



野火团队 Wild Fire Team

0、友情提示

《零死角玩转 STM32》系列教程由初级篇、中级篇、高级篇、系统篇、四个部分组成，根据野火 STM32 开发板旧版教程升级而来，且经过重新深入编写，重新排版，更适合初学者，步步为营，从入门到精通，从裸奔到系统，让您零死角玩转 STM32。M3 的世界，与野火同行，乐意惬意无边。

另外，野火团队历时一年精心打造的《STM32 库开发实战指南》将于今年 10 月份由机械工业出版社出版，该书的排版更适于纸质书本阅读以及更有利于查阅资料。内容上会给你带来更多的惊喜。是一本学习 STM32 必备的工具书。敬请期待！



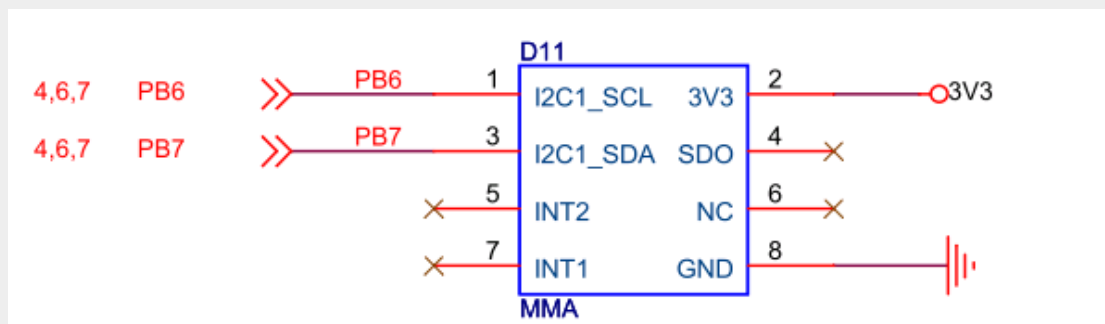
10、重力感应/三轴加速（MMA7455）

10.1 实验描述及工程文件清单

| | |
|---------|---|
| 实验描述 | 控制倾角传感器（MMA7455）测量加速度和倾角，通过串口打印测量得的数据。 |
| 硬件连接 | PB6-I2C1_SCL PB7-I2C1_SDA |
| 用到的库文件 | startup/start_stm32f10x_hd.c CMSIS/core_cm3.c CMSIS/system_stm32f10x.c FWlib/stm32f10x_gpio.c FWlib/stm32f10x_rcc.c FWlib/stm32f10x_usart.c FWlib/stm32f10x_i2c.c |
| 用户编写的文件 | USER/main.c USER/stm32f10x_it.c USER/usart1.c USER/I2C_MMA.c |

在这个加速度传感器例程中野火采用使用了集成模块，只需要把模块的电源线和 SDA，SCL 端口连接到开发板上的 I2C 总线即可。EEPROM 的也使用了 I2C 总线，所以可以参照 EEPROM 的件原理图。

野火 STM32 开发板 2.4G 无线模块接口图:



10.2 MMA7455 简介

MMA7455 是一款数字三轴加速度传感器，可以利用测量出来的重力加速度在某方向上的分量来计算器件与水平面间的夹角，测量倾角是很多有趣应用的基础。

本实验采用 I2C 方式与传感器通讯，控制传感器用 2g 的量程。

在测量前对传感器进行校准。

10.3 代码分析

首先要添加用的库文件，在工程文件夹下 **Fwlib** 下我们需添加以下库文件：

```
1. stm32f10x_gpio.c
2. stm32f10x_rcc.c
3. stm32f10x_usart.c
4. stm32f10x_i2c.c
```

还要在 `stm32f10x_conf.h` 中把相应的头文件添加进来：

```
1. #include "stm32f10x_gpio.h"
2. #include "stm32f10x_i2c.h"
3. #include "stm32f10x_rcc.h"
4. #include "stm32f10x_usart.h"
```

配置好所需的库文件之后，我们就从 `main` 函数开始分析：

```
1. *
2. * 函数名: main
3. * 描述   : 主函数
4. * 输入   : 无
```

```
5.  * 输出 : 无
6.  * 返回 : 无
7.  */
8.  int main(void)
9.  {
10.     /* 配置系统时钟为 72M */
11.     SystemInit();
12.
13.     /* 串口1 初始化 */
14.     USART1_Config();
15.
16.     /*重力传感器初始化*/
17.     I2C_MMA_Init();
18.
19.     /*重力传感器校准*/
20.     I2C_MMA_Cal();
21.
22.     printf("\r\n-----这是一个重力传感器测试程序-----\r\n");
23.
24.
25.     /* 检测倾角*/
26.     I2C_MMA_Test(&X_Value);
27.     I2C_MMA_Test(&Y_Value);
28.     I2C_MMA_Test(&Z_Value);
29.
30.     printf("\r\n-----X 方向的数据-----\r\n");
31.     I2C_MMA_Printf(&X_Value);
32.
33.     printf("\r\n-----Y 方向的数据-----\r\n");
34.     I2C_MMA_Printf(&Y_Value);
35.
36.     printf("\r\n-----Z 方向的数据-----\r\n");
37.     I2C_MMA_Printf(&Z_Value);
38.
39.     /*进入省电模式*/
40.     if(I2C_MMA_Standby() == SUCCESS )
41.
42.         printf("\r\n Acceleration enter standby mode! \r\n");
43.     else
44.         printf("\r\n Standby mode ERROR! \r\n");
45.
46. }
```

系统库函数 `SystemInit()`；将系统时钟设置为 72M，`USART1_Config()`；配置串口，关于这两个函数的具体讲解可以参考前面的教程，这里不再详述。

```
1.  /*
2.  * 函数名: I2C_MMA_Init
3.  * 描述 : I2C 外设(MMA7455)初始化
4.  * 输入 : 无
5.  * 输出 : 无
6.  * 调用 : 外部调用
7.  */
8.  void I2C_MMA_Init(void)
9.  {
10.     I2C_GPIO_Config();
11.     I2C_Mode_Config();
12. }
```

`I2C_MMA_Init()`; 是用户编写的函数, 其中调用了 `I2C_GPIO_Config()`; 配置好 I2C 所用的 I/O 端口, 调用 `I2C_Mode_Configu()`; 设置 I2C 的工作模式, 并使能相关外设的时钟。

`I2C_MMA_Cal()` 函数是用来校正传感器的, 校正需要先测量原始数据, 我们先看

`I2C_MMA_Test()` 函数

```
1.  /*
2.  * 函数名: I2C_MMA_Test
3.  * 描述   : 测量倾角 和 加速度 (量程 0-2g)
4.  * 输入   : 数据结构体的指针
5.  * 输出   : 无
6.  * 调用   : 外部调用
7.  */
8. void I2C_MMA_Test(MMA_Dat* MMA_Value)
9. {
10.     u8 temp;
11.
12.     /*MMA 进入 2g 量程测试模式*/
13.     I2C_MMA_ByteWrite(0x05,MMA_MCTL_Addr);
14.
15.     /*DRDY 标置位,等待测试完毕*/
16.     while(!(I2C_MMA_ByteRead(MMA_STATUS_Addr)&0x01));
17.
18.     /*读取测得的数据*/
19.     MMA_Value->Out = I2C_MMA_ByteRead(MMA_Value->Addr);
20.
21.     if((MMA_Value->Out&0x80) ==0x00) /*读出的原始值为正数 */
22.     {
23.         temp = MMA_Value->Out;
24.
25.         /*将原始值转换为加速度, 乘以 -1 为方向处理*/
26.         MMA_Value->Acc = (float) (-1)*temp *ACC_Gravity/64;
27.
28.         /*将原始值转换为角度*/
29.         if(temp >=64)
30.             /*加速度值大于 1g */
31.             MMA_Value->Angle = 90.0;
32.
33.         else
34.             /*加速度小于 1 g, Angle = asin(Acc/9.8)*57.32; 弧度制转换
35.             57.32 = 180/3.14*/
36.             MMA_Value->Angle = asin((float) temp/64)*57.32;
37.
38.         /*读出的原始值为负数 */
39.     else
40.     {
41.         temp = MMA_Value->Out;
42.
43.         /*二补码转换*/
44.         temp -= 1;
45.         temp = ~temp;
46.
47.         /*将原始值转换为加速度*/
48.         MMA_Value->Acc = (float) temp *ACC_Gravity/64;
49.
50.         /*将原始值转换为角度, 乘以 -1 为方向处理*/
51.         if(temp>=64)
52.             MMA_Value->Angle = -90.0;
53.         else
54.             /* Angle = asin(Acc/9.8)*57.32 */
55.             MMA_Value->Angle = (-1)*asin((float) temp/64)*57.32;
```



```
56.     }  
57. }
```

这个函数调用了 `I2C_MMA_ByteWrite()`，以下是函数原型：

```
1.  /*  
2.   * 函数名: I2C_MMA_ByteWrite  
3.   * 描述  : 写一个字节到 I2C MMA 寄存器中  
4.   * 输入  : -pBuffer 缓冲区指针  
5.   *          -WriteAddr 接收数据的 MMA 寄存器的地址  
6.   * 输出  : 无  
7.   * 返回  : 无  
8.   * 调用  : 内部调用  
9.   */  
10. static void I2C_MMA_ByteWrite(u8 pBuffer, u8 WriteAddr)  
11. {  
12.     /*wait until I2C bus is not busy*/  
13.     while(I2C_GetFlagStatus(I2C1, I2C_FLAG_BUSY));  
14.  
15.     /* Send START condition */  
16.     I2C_GenerateSTART(I2C1, ENABLE);  
17.  
18.     /* Test on EV5 and clear it */  
19.     while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT));  
20.  
21.     /* Send MMA address for write */  
22.     I2C_Send7bitAddress(I2C1, MMA_ADRESS, I2C_Direction_Transmitter);  
23.  
24.     /* Test on EV6 and clear it */  
25.     while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECT  
26. ED));  
27.  
28.     /* Send the MMA's Register address to write to */  
29.     I2C_SendData(I2C1, WriteAddr);  
30.  
31.     /* Test on EV8 and clear it */  
32.     while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED));  
33.  
34.     /* Send the byte to be written */  
35.     I2C_SendData(I2C1, pBuffer);  
36.  
37.     /* Test on EV8 and clear it */  
38.     while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED));  
39.  
40.     /* Send STOP condition */  
41.     I2C_GenerateSTOP(I2C1, ENABLE);  
42. }
```

这与 EEPROM 的 `I2C_EE_PageWrite()` 函数很类似，第一个参数为要写入的数值，第二个参数为待写入寄存器的地址。这些寄存器的地址在 `I2C_MMA.h` 文件中都有宏定义。

其中要强调的地方是 MMA7455 的器件地址（分清器件地址与寄存器地址的区别哦！），在 MMA7455 的 DataSheet 中查到的器件地址为 `0X1D`，但这个地址是不正确的，与野火使用的这块芯片有区别，市面上的传感器一般这个器件地址也为 `0x3A`。



在用 Jlink 调试程序的时候，程序循环运行在第 25 行就可以知道是器件地址出了问题，其它通讯的错误也可以用这样的方式查找出来，省了示波器 ^_^。

I2C_MMA_ByteRead() 类似，功能是读取寄存器的值，就不分析了。

回到 I2C_MMA_Test() 函数中，I2C_MMA_Test() 的第 13 行代码是向 MMA7455 写入命令，开启 2g 转换模式，开启了检测模式后，通过查询 MMA7455 的 DRDY 位，等待 MMA7455 的加速度转化完成。以下为 DataSheet 的说明，附上野火的注释：

截图来自《MMA7455L》

\$16: Mode Control Register (Read/Write)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Bit |
|----|------|-------|------|---------|---------|---------|---------|----------|
| -- | DRPD | SPI3W | STON | GLVL[1] | GLVL[0] | MODE[1] | MODE[0] | Function |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Default |

Table 5. Configuring the g-Select for 8-bit output using Register \$16 with GLVL[1:0] bits

| GLVL [1:0] | g-Range | Sensitivity |
|------------|---------|-------------|
| 00 | 8g | 16 LSB/g |
| 01 | 2g | 64 LSB/g |
| 10 | 4g | 32 LSB/g |

2g模式最小量化单位

Standby Mode

This digital output 3-axis accelerometer provides a standby mode that is ideal for battery operated products. When standby mode is active, the device outputs are turned off, providing significant reduction of operating current. When the device is in standby mode the current will be reduced to 2.5 μ A typical. In standby mode the device can read and write to the registers with the I²C/SPI available, but no new measurements can be taken in this mode as all current consuming parts are off. The mode of the device is controlled through the mode control register by accessing the two mode bits as shown in Table 6.

Table 6. Configuring the Mode using Register \$16 with MODE[1:0] bits

| MODE [1:0] | Function |
|------------|----------------------|
| 00 | Standby Mode |
| 01 | Measurement Mode |
| 10 | Level Detection Mode |
| 11 | Pulse Detection Mode |

使用检测模式

Measurement Mode

The device can read XYZ measurements in this mode. The pulse and threshold interrupt mode, continuous measurements on all three axes enabled. The g-range for 2g, 4g, or 8g-range of 8g is selectable with 10-bit data. The sample rate during measurement mode is 250 Hz with the 125 Hz filter selected. Therefore, when a conversion is complete (signaled by the DRDY flag), the next measurement will be ready.

读取数据前检查
DRDY寄存器

When measurements on all three axes are completed, a logic high level is output to the DRDY pin, indicating "measurement data is ready." The DRDY status can be monitored by the DRDY bit in Status Register (Address: \$09). The DRDY pin is kept high until one of the three Output Value Registers are read. If the next measurement data is written before the previous data is read, the DOVR bit in the Status Register will be set. Also note that in measurement mode, level detection mode and pulse detection mode are not available.

转化完成后根据传入的参数，把相应的 MMA7455 数据寄存器中的数据读取出来，各个方向的原始数据储存在相应的结构体的 .Out 变量中：

```
1. typedef struct
2. {
3.     uc8 Addr;           //寄存器的地址
4.     uc8 Name;           //数据的方向，X，Y 或 Z
5.     s8 Out;             //寄存器的值
6.     float Acc;          //加速度值
7.     float Angle;        //角度值
8. }MMA_Dat;
```

在结构体初始化的时候把寄存器的地址和名称定义好了：




```
1.  /*****/
2.  MMA_Dat X_Value={MMA_XOUT8_Addr, 'X'};
3.  MMA_Dat Y_Value={MMA_YOUT8_Addr, 'Y'};
4.  MMA_Dat Z_Value={MMA_ZOUT8_Addr, 'Z'};
```

要注意的是 MMA7455 中的数据以**二补数**的形式来储存：

截图来自《MMA7455L》

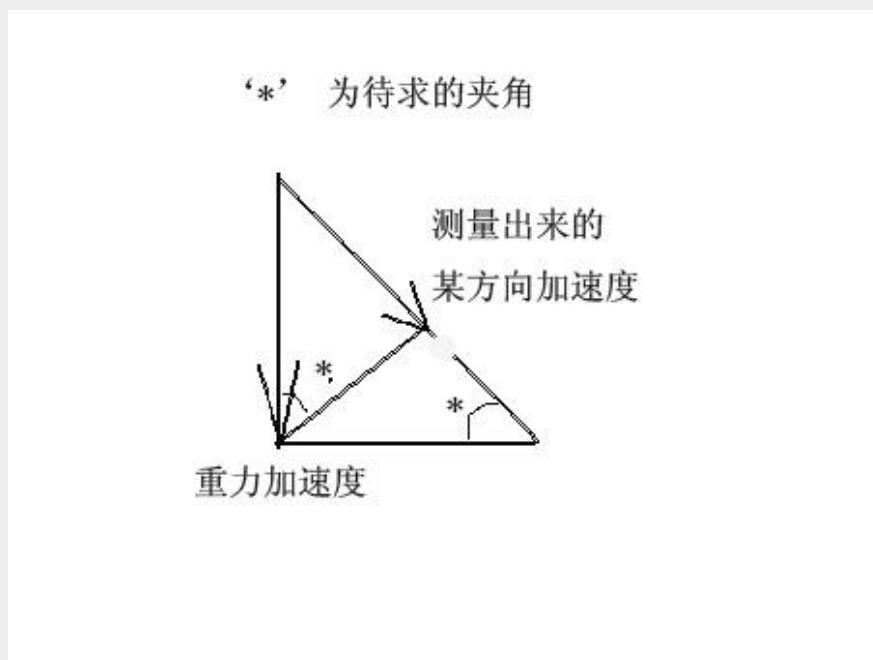
\$00: 10bits Output Value X LSB (Read only)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Bit |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| XOUT [7] | XOUT [6] | XOUT [5] | XOUT [4] | XOUT [3] | XOUT [2] | XOUT [1] | XOUT [0] | Function |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Default |

Signed byte data (2's complement): $0g = 10'h000$

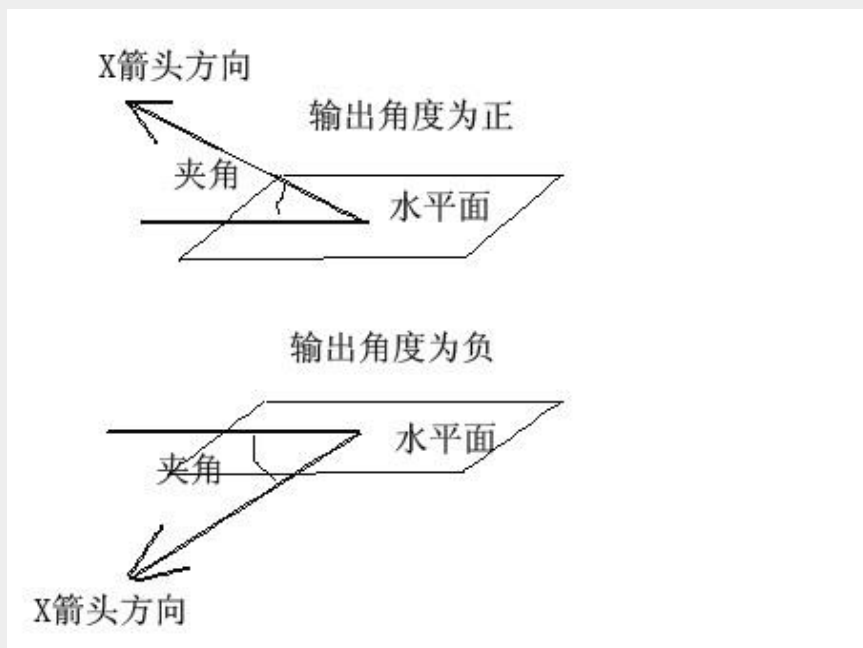
把二补数转换为原码后就可以像处理 ADC 的数据一样了，我们选择的是 **2g** 量程，敏感度为：**64LSB/g**。

至于角度的计算公式，把重力加速度按各方向的分解，可以推导出来。



计算出来的 **加速度或角度** 乘以 **-1** 是野火根据传感器上标注的**箭头的方向**，并以此为**正方向**进行调整的，不同的厂家生产的传感器稍有区别。

下图是以 X 方向输出的角度为例：



输出的角度为各箭头方向的直线与水平面间的夹角，在水平面上方为正，下方为负。

接下来分析一下 校准函数：

```
1. /*
2.  * 函数名: I2C_MMA_Cal
3.  * 描述 : MMA7455 校准
4.  * 输入 : 无
5.  * 输出 : 无
6.  * 调用 : 外部调用，在初始化后调用
7.  */
8. void I2C_MMA_Cal(void)
9. {
10.     I2C_MMA_Write(0x00, MMA_YOFFL_Addr); /*校正 x 值 00 */
11.     I2C_MMA_Write(0x30, MMA_YOFFL_Addr); /*校正 y 值 48*/
12.     I2C_MMA_Write(0xE2, MMA_ZOFFL_Addr); /*校正 z 值 -30 的补码 */
13.     I2C_MMA_Write(0xFF, MMA_ZOFFH_Addr); /*校正 z 值,校正值为负数,要把
        高位写 1;*/
14. }
```

`I2C_MMA_Cal()` 函数是用来校正传感器的，称为 **0g 校准**。这个函数在第一次测量前必须调用，而且每个传感器的校正值都有不同，其中的校正参数就要大家亲手去调试出来啦。

参照 DataSheet《AN3745》按以下步骤校准：

1. 把传感器按水平方式放置，读取各方向寄存器输出值。

这个情况下，Z 轴方向标准输出应为 **1g**，X 轴和 Y 轴均为 **0**。对应到各个寄存器的原始数据就应是 **ZOUT8 = 64**，**XOUT8 = 0**，**YOUT8 = 0**。但是未校准前，各个寄存器的输出会有一定的**偏差**。

以下为传感器水平放置，野火的例程中未校准得出的数据

```
-----这是一个重力传感器测试程序-----  
-----X方向的数据-----  
寄存器的原始数据是: XOUT8 = 0  
以 箭头标注 方向为 正 方向, X方向加速度是: 0.00 m/s^2  
以 箭头标注 方向为 正 方向, X方向与水平面的夹角是: 0.00 度  
-----Y方向的数据-----  
寄存器的原始数据是: YOUT8 = -19  
以 箭头标注 方向为 正 方向, Y方向加速度是: 2.91 m/s^2  
以 箭头标注 方向为 正 方向, Y方向与水平面的夹角是: -17.28 度  
-----Z方向的数据-----  
寄存器的原始数据是: ZOUT8 = 75  
以 箭头标注 方向为 正 方向, Z方向加速度是: -11.48 m/s^2  
以 箭头标注 方向为 正 方向, Z方向与水平面的夹角是: 90.00 度  
Acceleration enter standby mode!
```

2.向相应的 OffSet 寄存器写入校准值:

要注意两个问题。

一是校准寄存器中的值为 $1/2 \text{ LSB}$ ，所以我们写入的值要相应地乘以 2 倍:

XOUT8 很标准，不用写入校准值，或向 XOFFL 写入 0;

YOUT8 输出为-19，所以应 YOFFL 写入 $38 = 2 \times 19$;

ZOUT8 输出为 75，所以应写入 $-22 = 2 \times (64 - 75)$;

截图来自《MMA7455L》

\$10: Offset Drift X LSB (Read/Write)

The following Offset Drift Registers are used for setting and storing the offset calibrations to eliminate the 0g offset. Please refer to Freescale application note AN3745 for detailed instructions on the process to set and store the calibration values.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Bit |
|---------|---------|---------|---------|---------|---------|---------|---------|----------|
| XOFF[7] | XOFF[6] | XOFF[5] | XOFF[4] | XOFF[3] | XOFF[2] | XOFF[1] | XOFF[0] | Function |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Default |

Signed byte data (2's complement): User level offset trim value for X-axis

| Bit | XOFF[7] | XOFF[6] | XOFF[5] | XOFF[4] | XOFF[3] | XOFF[2] | XOFF[1] | XOFF[0] |
|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| Weight* | 64 LSB | 32 LSB | 16 LSB | 8 LSB | 4 LSB | 2 LSB | 1 LSB | 0.5 LSB |

*Bit weight is for 8g 10-bit data output. Typical value for reference only. Variation is specified in "Electrical Characteristics" section.

\$11: Offset Drift X MSB (Read/Write)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Bit |
|----|----|----|----|----|----------|---------|---------|----------|
| -- | -- | -- | -- | -- | XOFF[10] | XOFF[9] | XOFF[8] | Function |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Default |

Signed byte data (2's complement): User level offset trim value for X-axis

当校准值要写入负数时，要注意第二个问题：

就是在 MMA7455 寄存器中数值是以补码的形式储存的，所以实际写入的数据要经过转化：-22 的补码为 0xEA。但是，向 ZOFFL 写入 0xEA 还是未能校准，因为校准寄存器的还有高 8 位，高 8 位必须全写入 1 才是 -22 的补码，所以还要向 ZOFFH 写入 0xff。

选取恰当的校准值往往要经过多次的检测，为了提高准确度，就辛苦一点吧。

最后讲解一下 Standby 模式，只要向 MCTL 寄存器写入 0x04 命令就可进入 Standby 模式，这时传感器不工作，功耗大大降低：

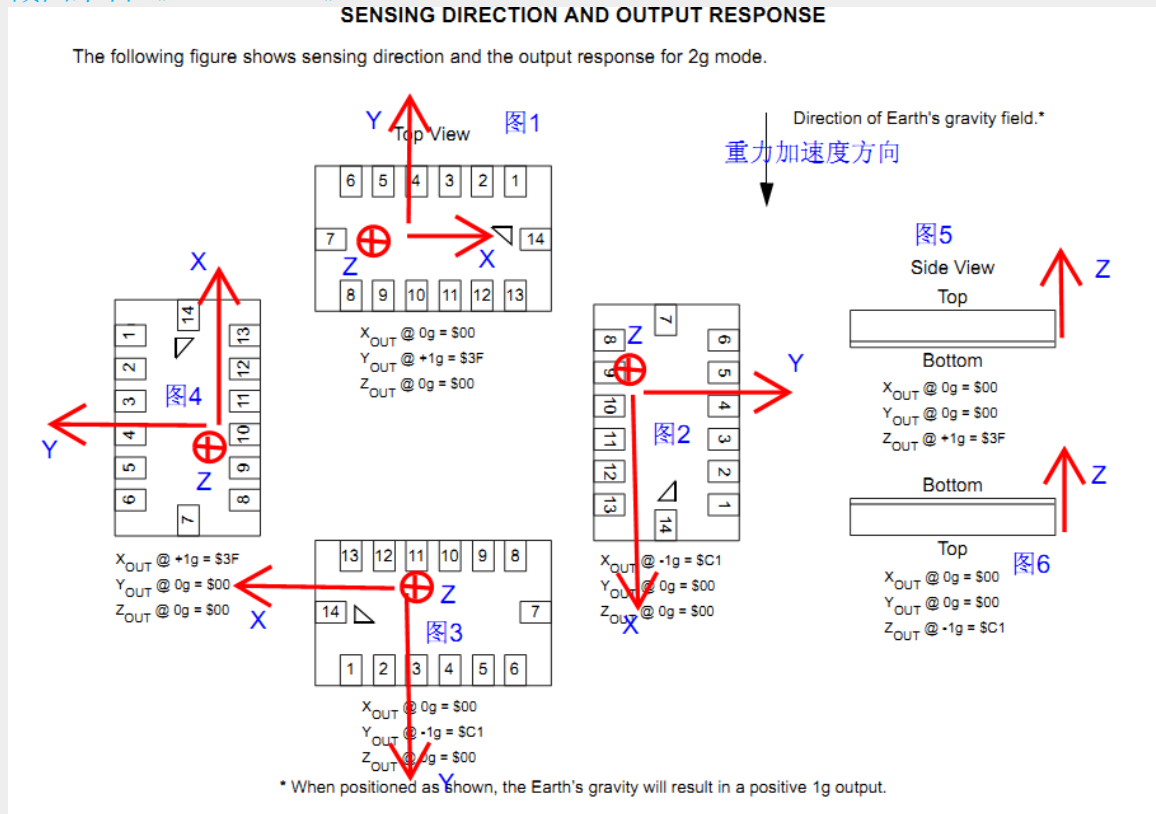
```
1. *
2. * 函数名: I2C_MMA_Standby
3. * 描述   : 重力传感器进入 Standby 模式，省电
4. * 输入   : 无
5. * 输出   : 无
6. * 返回   : 是否成功进入 Standby 模式
7. * 调用   : 外部调用
8. */
9. u8 I2C_MMA_Standby(void)
10. {
11.     u8 MMA_Test;
12.
13.     /*MMA Standby Mode*/
14.     I2C_MMA_ByteWrite(0x04, MMA_MCTL_Addr);
15.
16.     /*MMA_Test*/
17.     MMA_Test = I2C_MMA_ByteRead(MMA_MCTL_Addr);
18.
19.     if(MMA_Test == 0x04)
20.         return SUCCESS;
21.     else
22.         return ERROR;
23. }
```

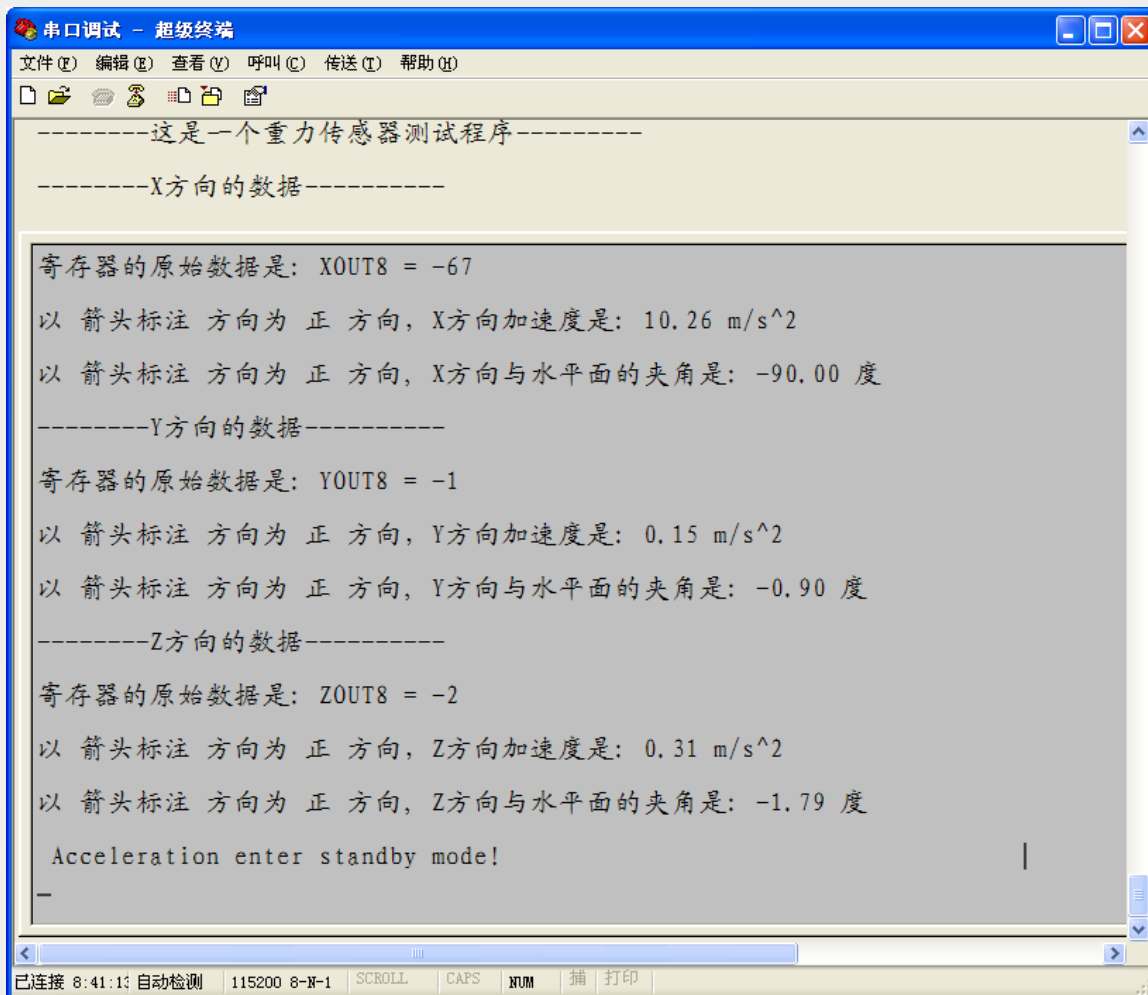
10.4 实验现象

很多第一次使用加速度传感器的人都会被它的方向弄糊涂，以下为传感器分别按 DataSheet 的这图 3，图 4，图 5，三个个方向来放置时没得的数据。

注释中的各个 XYZ 轴的方向为野火使用的传感器上标注的箭头方向，设为正方向。将野火 STM32 开发板供电(DC5V)，插上 JLINK。

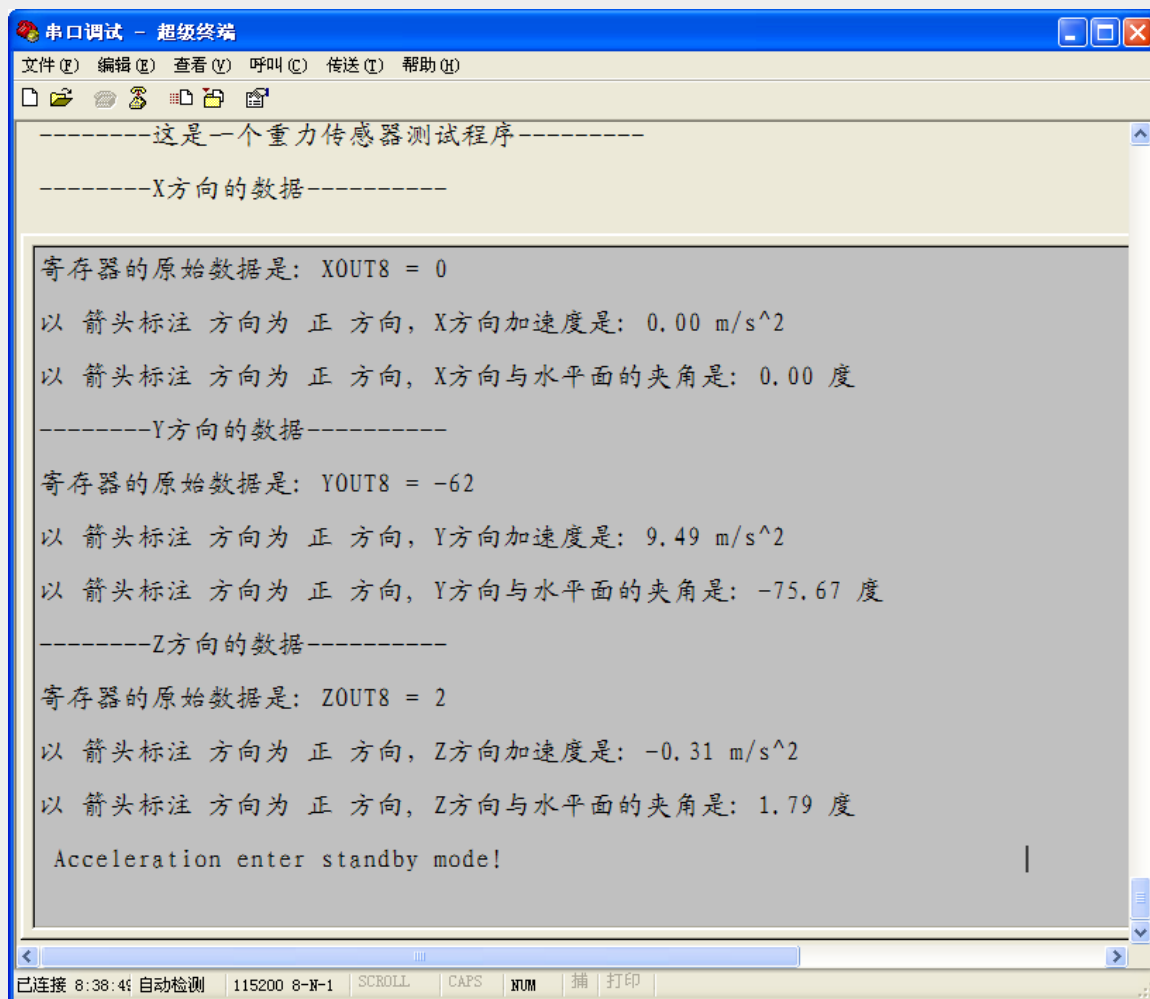
截图来自《MMA7455L》





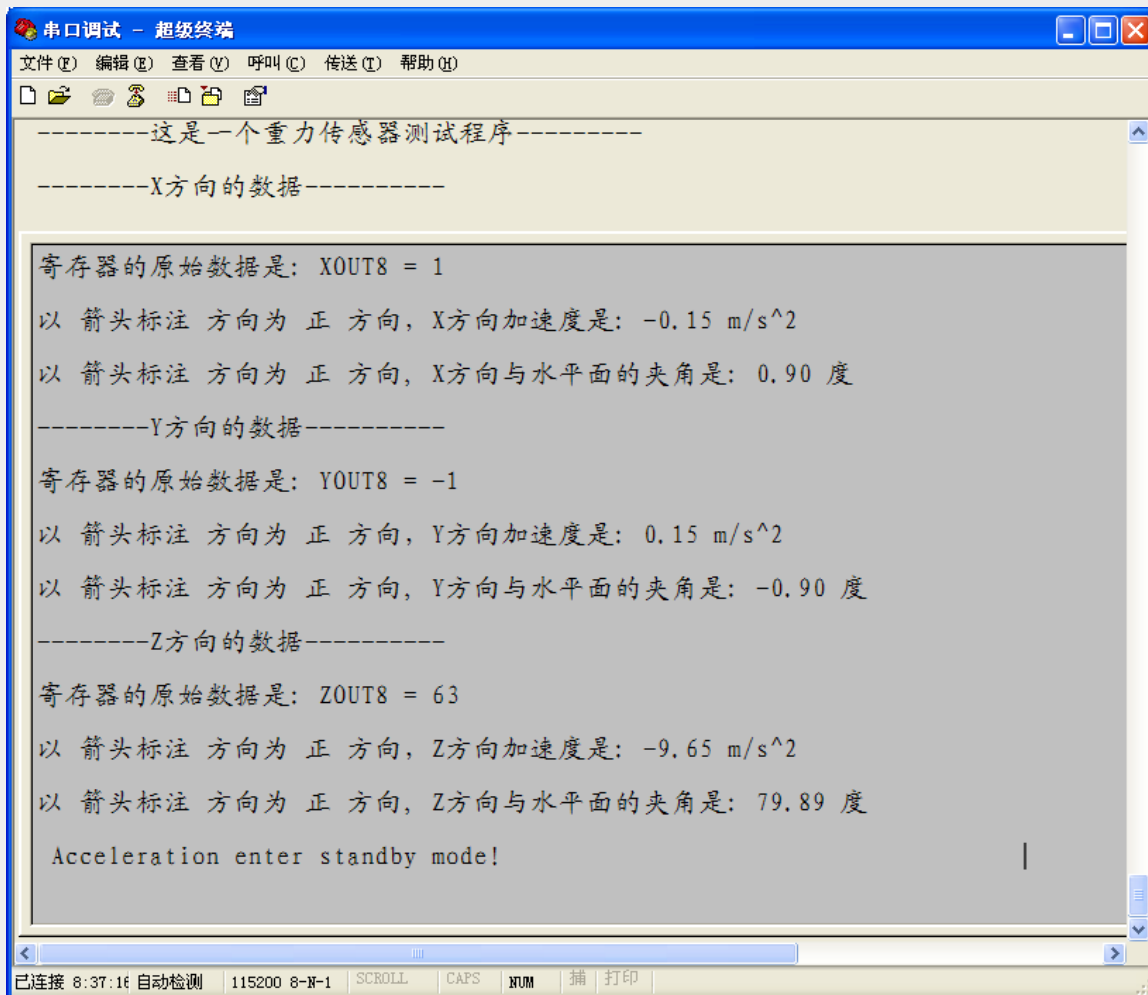
```
串口调试 - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
-----这是一个重力传感器测试程序-----
-----X方向的数据-----
寄存器的原始数据是: XOUT8 = -67
以 箭头标注 方向为 正 方向, X方向加速度是: 10.26 m/s^2
以 箭头标注 方向为 正 方向, X方向与水平面的夹角是: -90.00 度
-----Y方向的数据-----
寄存器的原始数据是: YOUT8 = -1
以 箭头标注 方向为 正 方向, Y方向加速度是: 0.15 m/s^2
以 箭头标注 方向为 正 方向, Y方向与水平面的夹角是: -0.90 度
-----Z方向的数据-----
寄存器的原始数据是: ZOUT8 = -2
以 箭头标注 方向为 正 方向, Z方向加速度是: 0.31 m/s^2
以 箭头标注 方向为 正 方向, Z方向与水平面的夹角是: -1.79 度
Acceleration enter standby mode!
-
已连接 8:41:13 自动检测 115200 8-N-1 SCROLL CAPS NUM 捕 打印
```

按图 2 方向放置



```
串口调试 - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
-----这是一个重力传感器测试程序-----
-----X方向的数据-----
寄存器的原始数据是: XOUT8 = 0
以 箭头标注 方向为 正 方向, X方向加速度是: 0.00 m/s^2
以 箭头标注 方向为 正 方向, X方向与水平面的夹角是: 0.00 度
-----Y方向的数据-----
寄存器的原始数据是: YOUT8 = -62
以 箭头标注 方向为 正 方向, Y方向加速度是: 9.49 m/s^2
以 箭头标注 方向为 正 方向, Y方向与水平面的夹角是: -75.67 度
-----Z方向的数据-----
寄存器的原始数据是: ZOUT8 = 2
以 箭头标注 方向为 正 方向, Z方向加速度是: -0.31 m/s^2
以 箭头标注 方向为 正 方向, Z方向与水平面的夹角是: 1.79 度
Acceleration enter standby mode!
已连接 8:38:46 自动检测 115200 8-N-1 SCROLL CAPS NUM 捕 打印
```

按图 3 方向放置



```
串口调试 - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
-----这是一个重力传感器测试程序-----
-----X方向的数据-----
寄存器的原始数据是: XOUT8 = 1
以 箭头标注 方向为 正 方向, X方向加速度是: -0.15 m/s^2
以 箭头标注 方向为 正 方向, X方向与水平面的夹角是: 0.90 度
-----Y方向的数据-----
寄存器的原始数据是: YOUT8 = -1
以 箭头标注 方向为 正 方向, Y方向加速度是: 0.15 m/s^2
以 箭头标注 方向为 正 方向, Y方向与水平面的夹角是: -0.90 度
-----Z方向的数据-----
寄存器的原始数据是: ZOUT8 = 63
以 箭头标注 方向为 正 方向, Z方向加速度是: -9.65 m/s^2
以 箭头标注 方向为 正 方向, Z方向与水平面的夹角是: 79.89 度
Acceleration enter standby mode!
```

已连接 8:37:16 自动检测 115200 8-N-1 SCROLL CAPS NUM 捕 打印