1. **主要特性**
   1. 发行版包括三轴陀螺仪ITG3205、三轴加速度计LIS3LV02DQ、6轴传感器LSM303DLH；加速度计信息采用的是LIS3LV02DQ；
   2. 陀螺量程：±2000°/s；
   3. 加速度计量程：±2g；
   4. 电源电压5~12V，推荐5V；
   5. 与上位机的通信接口采用8P FFC接口，用FFC软排线连接。FFC管脚间距1mm，下接。管脚定义如下

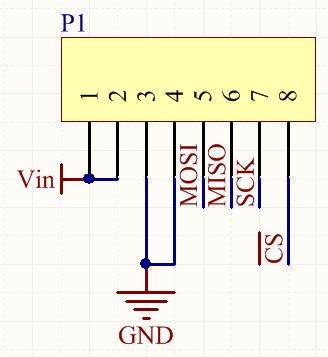


图 1 FFC8P接口定义

1. **启动过程**

上电之后，LED每秒闪烁三次（下称三闪）。程序首先配置STM32 I2C、SPI接口，延时3秒等待稳定，通过串行总线对各个传感器进行配置。这个过程结束前，检测模块是否倒置，若倒置，则进入加速度计标定模式，见第3部分内容，若未倒置，则开始测量陀螺仪零偏，这时LED每秒闪烁两次（下称双闪）。零偏测量结束后，进入正常的工作循环，从此以后，LED每秒闪烁1次（下称单闪）。

在进入正常工作循环之前，上位机无法通过SPI读取任何信息。

1. **使用之前需要进行加速度计和磁罗盘的标定**
   1. 加速度计标定

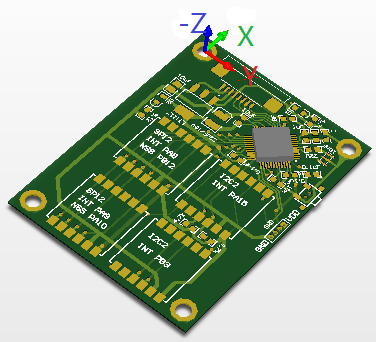


图 2 北东地坐标系

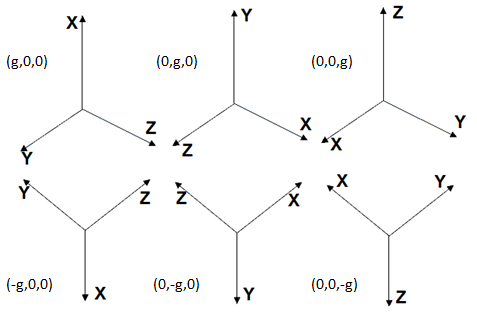


图 3 标定加速度计的6个位置

若上电时模块处于倒置状态，LED三闪，5秒钟后进入加速度计标定模式。此时LED双闪。接下来的2秒钟程序读取静置状态下的陀螺数据，并计算其方差，作为之后判断静止的条件。之后进入正式标定过程。在标定过程中，LED三闪表示模块当前放置位置不正确，当模块置于正确的位置时，LED双闪表示正在采集数据，这时请务必保证模块处于静置状态。模块放置的顺序为：x轴向上—>x轴向下—>y轴向上—>y轴向下—>z轴向上—>z轴向下。加速度计标定过程结束后，LED双闪，校正陀螺仪，直到LED单闪，进入正常工作循环。

* 1. 磁罗盘标定

磁罗盘的标定采用椭球标定法，较加速度计稍微复杂一些。具体原理参见相关文献。磁罗盘的标定需要连接上位机，如图 4所示。因为只有通过上位机才能将采集到的数据通过串口发送给电脑。



图 4 AHRS与上位机（图中红色板子）连接

1. AHRS与上位机连接，上位机读取到数据，通过串口发送给电脑，电脑上打开超级终端（如securityCRT）接收数据并保存为txt。格式如图 5所示，为多行3列的形式。

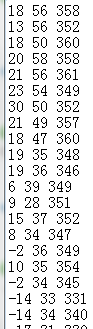


图 5 磁罗盘原始数据

1. 利用matlab的椭球拟合函数拟合椭球参数。

d*ata=load('data.txt');*

*mag=data;*

*[ center, radii, evecs, v ] = ellipsoid\_fit(mag,1);*

*>> center*

*center =*

*132.7803*

*82.9096*

*30.7253*

*>> radii*

*radii =*

*386.8008*

*381.2543*

*341.8893*

*>> frac=1./radii\*500*

*frac =*

*1.2927*

*1.3115*

*1.4625*

3. 得到椭球参数后，对磁罗盘的零偏和标度因数进行补偿。就上述例子而言，补偿方法如下：

Mag[0] = frac[0]\*(mag\_raw[0]-center[0]);

Mag[1] = frac[1]\*(mag\_raw[1]-center[1]);

Mag[2] = frac[2]\*(mag\_raw[2]-center[2]);

其中，mag\_raw表示补偿前的磁罗盘测量值，mag表示补偿后的输出。将这段代码放在AHRS源码中函数void LSM303DLH\_Raw2Mag(u8 \*raw, s16 \*mag)的最后。以上三式的含义是不言自明的。

1. **数据读取方法**
   1. 读取传感器原始数据

传感器原始数据包括角速率(0.00025rad/s/LSB)、加速度(0.001m/s^2/LSB)和磁场强度。这些数据已经经过了补偿校正。用于通信的原始数据结构体为

typedef struct{

s16 data[9];//gyr[3],acc[3],mag[3]

s32 Check;//Check=sum(data)

}ComType;

该结构体共包括22个字节，前18个字节为传感器数据，后四个字节为校验和。上位机先发送一个0x51，然后连续读22个字节，将前18个字节拼成9个int16\_t，后4个字节拼成一个int32\_t，再将9个int16\_t相加求和与之比较，判断数据有效性。

推荐的读取方法：

u8 ReadAHRSRaw(SensorDataType\* sd)

{

u8 i;

static u32 miss=0;

static u32 hit=0;

ComType cmt;

u8 byteBuffer[50];

u8 byteToRead=sizeof(ComType);

s32 CheckSum=0;

AHRS\_SPI\_CS\_LOW();

Delay\_us(80);

AHRS\_SPI\_SendByte(0x51); //send 0x51 to init raw data read

Delay\_us(200);

for(i=0;i<byteToRead;i++) //read 22 bytes

{

byteBuffer[i]=AHRS\_SPI\_ReadByte();

Delay\_us(4);

}

AHRS\_SPI\_CS\_HIGH();

cmt=\*(ComType \*)byteBuffer; //强制类型转换

for(i=0;i<9;i++) CheckSum+=cmt.data[i];

if(CheckSum == cmt.Check && CheckSum!=0)

{

for(i=0;i<3;i++) sd->gyr[i]=cmt.data[i]\*0.00025;

for(i=0;i<3;i++) sd->acc[i]=cmt.data[i+3]\*0.001;

for(i=0;i<3;i++) sd->mag[i]=cmt.data[i+6];

return 1;

}

return 0;

}

* 1. 读取姿态数据

姿态数据结构体如下所示：

typedef struct

{

s16 data[7];

s32 check;

}AttComType;

姿态数据共包括18个字节。上位机先发送一个0xa9，之后再连续读28个字节。方法与原始数据的读取类似