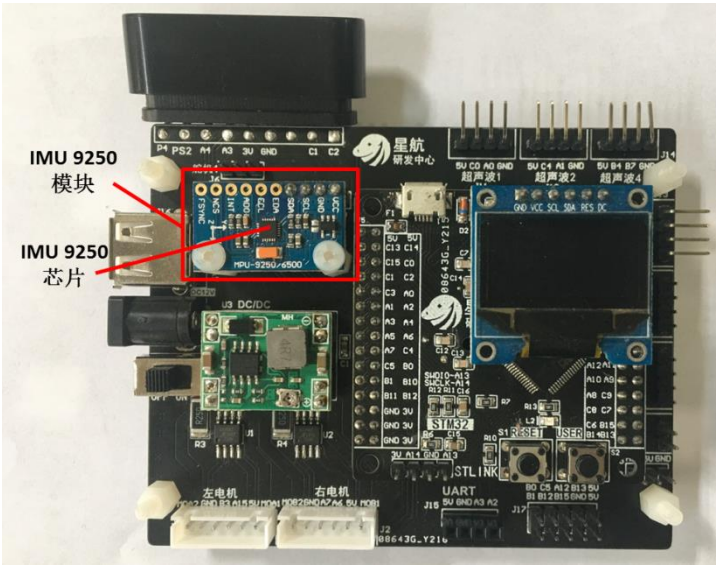
轻舟机器人惯导 IMU9250 数据采集

AI 航 团队

轻舟机器人采用 IMU9250 作为姿态传感器，与激光雷达进行数据融合，完成轻舟机器人的定位和导航功能，如下图所视为 IMU9250 模块。



1. 产品概述

9250 是一个 QFN 封装的复合芯片（MCM），它由 2 部分组成。一组是 3 轴加速度还有 3 轴陀螺仪，另一组则是 AKM 公司的 AK8963 3 轴磁力计。所以，9250 是一款 9 轴运动跟踪装置。通过 I2C 方案，可直接输出 9 轴的全部数据。一体化的设计，运动性的融合，时钟校准功能，让开发者避开了繁琐复杂的芯片选择和外设成本，保证最佳的性能。

三轴陀螺仪的特性：

可编量程（±250，±500，±1000 度/秒）三轴（x，y，z）16 位 ADC 角速度数字输出
可编程数字低通滤波
陀螺仪工作电流：3.2mA
休眠模式电流：8uA
出厂灵敏度校准
自我检测

MPU-9250 陀螺仪是由三个独立检测 X，Y，Z 轴的 MEMS 组成。利用科里奥利效应来检测每个轴的转动(当某个轴发生变化,相应的电容传感器会发生变化,产生的信号被放大,调解，滤波，最后产生个与角速率成正比的电压，然后将每一个轴的电压转换成 16 位的数据)。各种速率（±250, ±500, ±1000, or ±2000 °/s）都可以被编程。ADC 的采样速率也是可编程的，从每秒 3.9-8000 个，用户还可选择是否使用低通滤波器来滤掉多余的杂波。

三轴加速度计的特性：

用户可编量程（±2g， ±4g，±8g，±16g）三轴 16 位 ADC 加速度数字输出
加速度计正常工作电流：450uA
低功耗模式电流 0.98Hz---8.4uA 31.25Hz----19.8uA

休眠模式电流：8uA

用户可编程中断、运动中断唤醒功能

自我检测

MPU9250 的三轴加速度也是单独分开测量的。根据每个轴上的电容来测量轴的偏差度。结构上降低了各种因素造成的测量偏差。当被置于平面上的时候，它会测出在 X 和 Y 轴上为 0g，Z 轴上为 1g 的重力加速度。加速度计的校准是根据工厂的标准来设定的，电源电压也许和你用的不一样。每一个传感器都有专门的 ADC 来提供数字性的输出。输出的范围是通过编程可调的±2g, ±4g, ±8g, or ±16g。

磁场计的特性：

3 轴单片霍尔传感器

大量程低功耗高精度

14 位（0.6uT/LSB）和 16 位（15uT/LSB）的分辨率输出

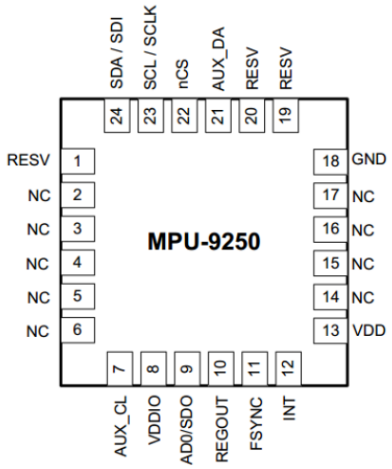
最大±4800uT 的测量范围

磁力计的正常工作电流：280uA—8Hz

内部自我检测功能

三轴磁力计采用高精度的霍尔效应传感器，通过驱动电路，信号放大和计算电路来处理信号来采集地磁场在 X，Y，Z 轴上的电磁强度。每个 ADC 均可满量程（±4800 μT）输出 16 位的数据。

2. 芯片说明

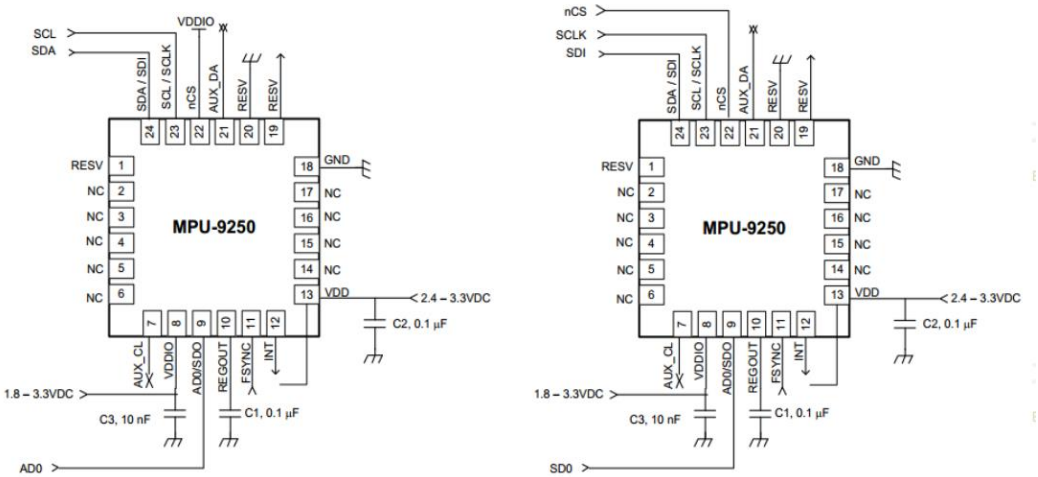


引脚功能说明

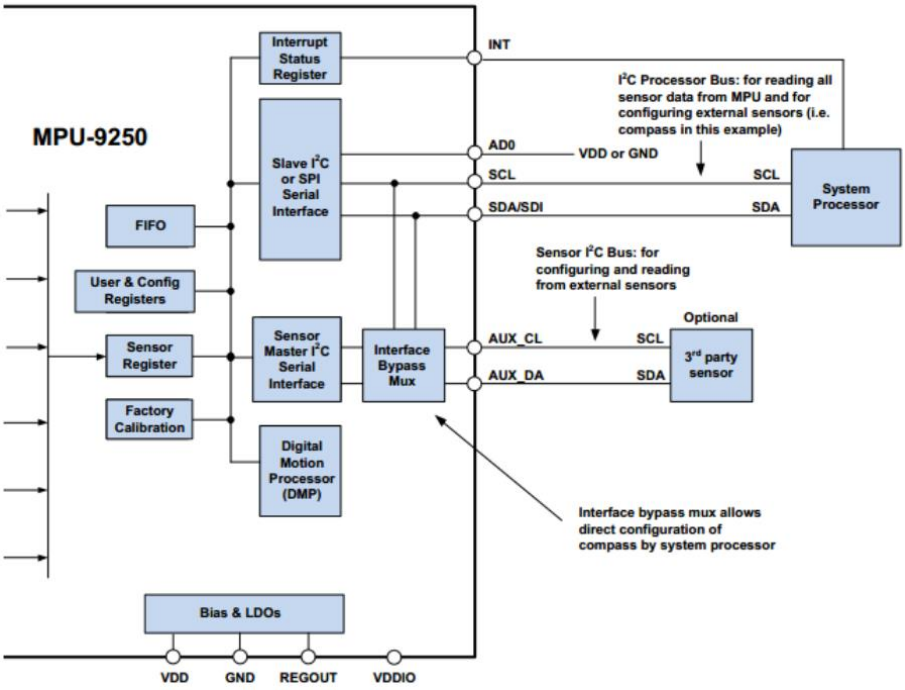
引脚号	引脚名	引脚功能
1	RESV	连接 VDDIO
7	AUX_CL	给从 I2C 设备提供主时钟
8	VDDIO	数字 I/O 口供压
9	AD0 / SDO	I2C 从机 LSB(AD0)地址;SPI 串口数据输出(SDO)
10	REGOUT	调节器引脚，连接滤波电容
11	FSYNC	数字同步输入帧，若不用请接地
12	INT	中断数字输出
13	VDD	电压供给端
18	GND	地

19	RESV	什么都不接
20	RESV	接地
21	AUX_DA	连接其他 I2C 设备的主机数据口
22	nCS	片选
23	SCL / SCLK	I2C 模式下的 SCL 、SPI 模式下的 SCLK
24	SDA / SDI	I2C 模式下的 SDA 、SPI 模式下的 SDI
2-6， 14-17	NC	什么都不接

两种典型电路如下图，左图为 I2C 通信方式，由图为 SPI 通信方式，轻舟机器人采用 I2C 方式进行数据读取。



下图的方案是采用 MPU9250 为主控芯片的 I2C 通信来控制从机。对第三方芯片来说 9250 是他的 I2C 主机。9250 作为 I2C 的主控有局限性，这要取决于系统对传感器的初始配置。I2C 的 SDA 和 SCL 是复用口，主控芯片可以通过它直接和辅助传感器通信(AUX_DA 和 AUX_CL)。一旦辅助传感器被主控芯片配置，复用功能不能再用，但是主控芯片可以通过辅助 I2C 口读取到第三方芯片的数据。中断脚必须连接系统的 GPIO 引脚，这样我们就可以从唤醒系统。



3. 数字接口

MPU—9250 的内部寄存器和储存器可以用 400KHz 的 I2C 或者 4 线模式在 1MHz 用 SPI 通讯。轻舟机器人采用 I2C 读取数据，使用到以下四个引脚：

引脚号	引脚名	引脚功能
8	VDDIO	数字 I/O 口提供电平
9	AD0/SD0	I2C 从机高位地址 LSB(AD0)； SPI 串行输出（SD0）
23	SCL/SCLK	I2C 时钟（SCL）； SPI 时钟（SCLK）
24	SDA/SDI	I2C 数据口(SDA);SPI 数据输入口（SDI）

I2C 是一个双线通信方案，它有 SDA 和 SCL 两根线分别传输数据和时钟信号。通常这 2 个接口是双向的开漏极接口。在连接设备的时候可以做主机或者从机。从机在通讯时，通过地址即可匹配。MPU-9250 通常和控制芯片连接时作为从机，SDA 和 SCL 通常需要上拉电阻到 VDD，最快通信速度达到 400KHz。

MPU9250 内部为 MPU6500 和 AK8963 的组合，实际上是两个不同的 I2C 地址。读取加速度和陀螺仪需要对 MPU6500 的 I2C 地址及进行操作，读取磁力计需要对 ak8963 地址进行操作。MPU-9250 作为从机时的地址为 7 位 110100X（B）。这个地址的 LSB 位由 AD0 引脚的电平确定，AD0 为低电平时 X 为 0，高电平 X 则为 1，在 mpu9250 中，电路已经确定，地址无法更改，磁力计地址为 0x0C。

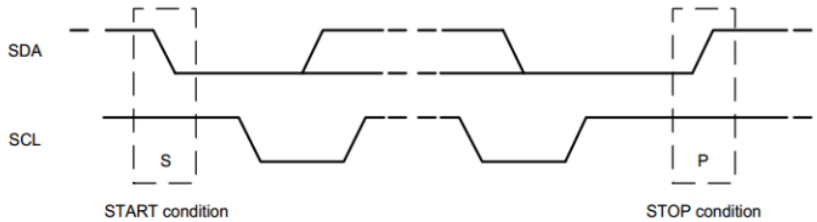
在读取时，注意需要人工将地址左移一位（I2C 读写为左对齐，第 8 位要存读写标志位），MPU6500 为 0xD0 或是 0xD2，磁力计为 0x18；

AD0=0 时，MPU6500 的 IIC 地址 1101000；AD0=1 时，IIC 地址 1101001，在实际应用中，这个字节的最低位应添加 0 或 1，表示写或者读；所以 MPU6500 写地址是：11010000 或 11010010(0xD0 或 0xD2)，磁力计写地址为 00011000（0x18）；读地址是:11010001 或 11010011(0xD1 或 0xD3)，磁力计读地址为 00011001(0x19)。

4. I2C 通信协议

（1）开始和停止条件：

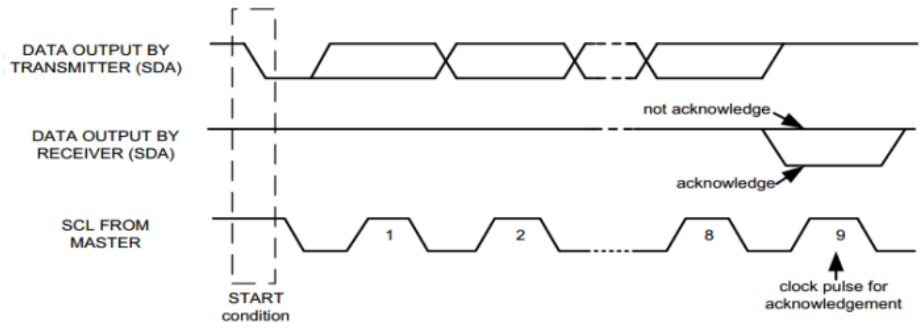
当主机将开始信号在 I2C 总线上初始的时候，表明准备开始通信。开始信号即当 SDA 处在下降沿时，SCL 置高。而当 SDA 产生上升时，SCL 置高，我们视作通讯停止信号。此外，除非再次出现开始信号或停止型号，否则总线一直通信。



开始和停止条件

（2）数据传输规则

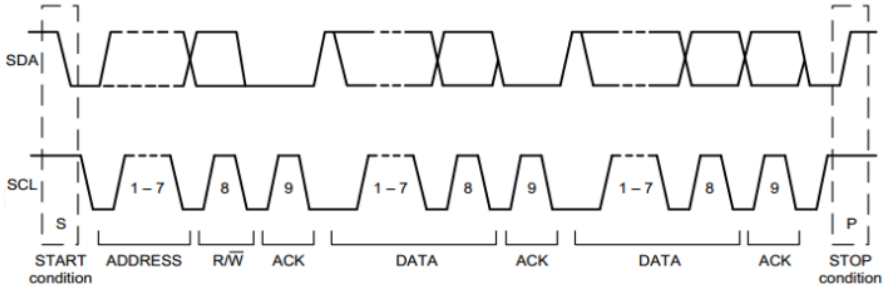
I2C 每帧为 8 位数据位和 1 位（ACK）数据接收方应答位。应答位 ACK 由从机负责拉低，从机在完整收到地址或数据后拉低 SDA 数据总线，表示正确接收。当从机忙碌无法传送其他数据的时候它会把 SCL 拉低，直到有数据输出，释放总线。



I2C 总线响应位 ACK

（3）通信

在开始信号发出后，主机开始发出 7 个地址位和 1 个读写位。读写位决定了主从机的读写状态。然后主机释放 SDA 线，等待从机的 ACK 应答信号。每次数据传输后必须跟一位读写位。从机应答即是拉低 SDA 到 SCL 高电平周期结束。当主机发出停止命令时，传输就会结束。然后主机重新发送开始信号继续和其他的 I2C 设备通信。当 SDA 出现上升沿并且 SCL 是高电平的时候，就表示停止信号。在通信时所有 SDA 信号的变化都是在 SCL 低电平的时候。



I2C 时序

（4）写 MPU250 的寄存器的方法：

主机发送开始信号和从机的 7 个地址位再加上 1 位的写入位。当在第 9 个时钟信号的时候，芯片产生应答。这时，主机输出寄存器地址，然后从机再次产生 ACK 应答，传输过程可以随时由停止信号停止。ACK 响应后，数据可以继续输入，除非没有产生停止位。芯片内部自带的递增寄存器可以自动将数据写入相应寄存。以下列出单字节和双字节的传输顺序。

单字节传输

Master	S	AD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

多字节传输

Master	S	AD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

（5）读 MPU9250 的寄存器的方法：

主机发送开始信号和从机的 7 个地址位再加上读位。此时，寄存器地址变成可读模式。此时会收到 MPU9250 的返回信号 ACK，然后主机再次发送开始信号和地址，9250 此时会发回应答信号 ACK 和数据。当主机发送 NACK 或停止位后通讯停止。NACK 信号就是第 9 个时钟脉冲 SDA 保持高电平。下图显示了单字节和双字节的读取时序。

单字节时序

Master	S	AD+W		RA		S	AD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

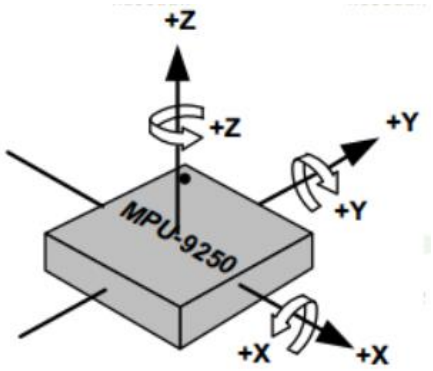
多字节读取时序

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

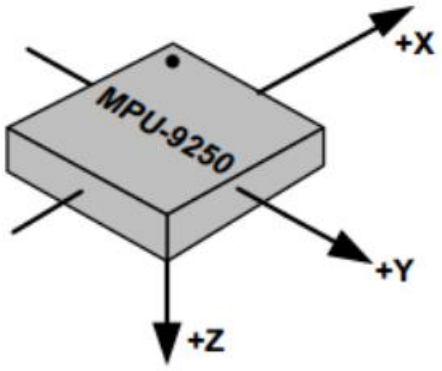
（6）I2C 符号说明

信号名称	作用
S	起使信号：当 SCL 处于高电平时 SDA 处于下降沿
AD	从机 I2C 地址
W	写位
R	读位
ACK	响应:SCL 高电平时 SDA 保持低电平
NACK	未响应：SDA 在第 9 个时钟周期保持高电平
RA	MPU9250 内部寄存器
DATA	发送或接受位
P	停止信号：SCL 置高时 SDA 产生上升沿

（7）定位轴示意图



加速度和陀螺仪的方向和极性



电子罗盘的方向

5. stm32 实现 9250 数据采集

（1）部分头文件 mpu9250.h 内容，包括定义 I2C 地址、9250 寄存器地址、相关函数声明等。

```
#ifndef __MPU9250_H
#define __MPU9250_H

#include "sys.h"
#include "delay.h"
#include "myiic.h"

/************** I2C Address *****************/
//AD0=0,IIC 地址 1101000，最低位添加 0 或 1,表示写或者读;
//所以写地址是:11010000 (0xD0);读地址是:11010001 (0xD1)
//磁力计地址为:11000(0x18)

#define ACCEL_ADDRESS     0xD0        //加速度 I2C 地址
#define GYRO_ADDRESS      0xD0        //角速度 I2C 地址
#define MAG_ADDRESS       0x18        //磁力计 I2C 地址

#define SMPLRT_DIV        0X19        //陀螺仪采样率  典型值为 0x07 1000/(1+7)=125 Hz
#define CONFIG            0X1A        //低通滤波器  典型值为 0x06 5Hz
#define GYRO_CONFIG       0X1B        //陀螺仪测量范围  0x18  正负 2000 度
#define ACCEL_CONFIG      0X1C        //加速度计测量范围  0x18  正负 16g
#define ACCEL_CONFIG2     0X1D       //加速度计低通滤波器  0x06 5Hz

//加速度输出数据寄存器地址
#define ACCEL_XOUT_H      0X3B
#define ACCEL_XOUT_L      0X3C
#define ACCEL_YOUT_H      0X3D
#define ACCEL_YOUT_L      0X3E
#define ACCEL_ZOUT_H      0X3F
#define ACCEL_ZOUT_L      0X40

//陀螺仪输出数据寄存器地址
#define GYRO_XOUT_H       0X43
#define GYRO_XOUT_L       0X44
#define GYRO_YOUT_H       0X45
#define GYRO_YOUT_L       0X46
#define GYRO_ZOUT_H       0X47
#define GYRO_ZOUT_L       0X48

//磁力计输出数据寄存器地址
#define MAG_XOUT_L        0x03
#define MAG_XOUT_H        0x04
#define MAG_YOUT_L        0x05
#define MAG_YOUT_H        0x06
#define MAG_ZOUT_L        0x07
#define MAG_ZOUT_H        0x08

void MPU9250_Write_Reg(u8 Slave_add,u8 reg_add,u8 reg_dat);        //写寄存器
u8 MPU9250_Read_Reg(u8 Slave_add,u8 reg_add);                      //读寄存器
void MPU6050_ReadData(u8 Slave_add,u8 reg_add,u8*Read,u8 num);   //读数据

u8 MPU9250_Init(void);                          //初始化
void MPU9250_READ_ACCEL(short *accData);   //读加速度
void MPU9250_READ_GYRO(short *gyroData);   //读角速度
void MPU9250_READ_MAG(short *magData);     //读磁力计

void readImu(void);                             //读 imu
#endif
```

（2）部分 mpu9250.c 文件内容，包括读、写函数、读取数据、初始化等函数

```
#include "mpu9250.h"

short Accel[3];                 // 加速度计
short Gyro[3];                  // 陀螺仪
short Mag[3];                   // 磁力计
short gyroX,gyroY,gyroZ;       //三个轴陀螺仪
short accelX,accelY,accelZ;    //三个轴加速度计
short magX,magY,magZ;          //三个轴磁力计

// I2C 写，根据第 4 节写寄存器的时序方法，实现写一个字节到从机
void MPU9250_Write_Reg(u8 Slave_add,u8 reg_add,u8 reg_dat)
{
        IIC_Start();                    //开始
        IIC_Send_Byte(Slave_add);       //发送 I2C 写地址
        IIC_Wait_Ack();                 //响应
        IIC_Send_Byte(reg_add);        //发送寄存器地址
        IIC_Wait_Ack();                 //响应
        IIC_Send_Byte(reg_dat);         //发送数据
        IIC_Wait_Ack();                 //响应
        IIC_Stop();                     //停止
}

//I2C 读，根据第 4 节度寄存器的时序方法，实现读一个字节
u8 MPU9250_Read_Reg(u8 Slave_add,u8 reg_add)
{
        u8 temp=0;

        IIC_Start();                    //开始
        IIC_Send_Byte(Slave_add);       //发送 I2C 写地址
        temp=IIC_Wait_Ack();            //响应
        IIC_Send_Byte(reg_add);         //发送寄存器地址
        temp=IIC_Wait_Ack();            //响应
        IIC_Start();                    //在此发送开始信号
        IIC_Send_Byte(Slave_add+1);     //发送 I2C 读地址
        temp=IIC_Wait_Ack();            //响应
        temp=IIC_Read_Byte(0);          //读取 1 字节
        IIC_Stop();

        return temp;
}

// 初始化
u8 MPU9250_Init(void)
{
        IIC_Init();
        if(MPU9250_Read_Reg(GYRO_ADDRESS,WHO_AM_I)==0x71)
        {
                MPU9250_Write_Reg(GYRO_ADDRESS,PWR_MGMT_1,0x00);   //解除休眠状态
                MPU9250_Write_Reg(GYRO_ADDRESS,SMPLRT_DIV,0x07);   //采样频率 125Hz
                MPU9250_Write_Reg(GYRO_ADDRESS,CONFIG,0X06);       //低通滤波器 5Hz
                MPU9250_Write_Reg(GYRO_ADDRESS,GYRO_CONFIG,0X18);  //陀螺仪量程,正负 2000 度
                MPU9250_Write_Reg(GYRO_ADDRESS,ACCEL_CONFIG,0X18);  //加速度量程,正负 16g
                return 0;
        }
        return 1;
}

//读取加速度数据的函数
void MPU9250_READ_ACCEL(short *accData)
{
        u8 BUF[6];
```

```
    BUF[0]=MPU9250_Read_Reg(ACCEL_ADDRESS,ACCEL_XOUT_L);    //读 X 加速度低字节
    BUF[1]=MPU9250_Read_Reg(ACCEL_ADDRESS,ACCEL_XOUT_H);    //读 X 加速度高字节
    accelX=(BUF[1]<<8)|BUF[0];

    BUF[2]=MPU9250_Read_Reg(ACCEL_ADDRESS,ACCEL_YOUT_L);    //读 Y 加速度低字节
    BUF[3]=MPU9250_Read_Reg(ACCEL_ADDRESS,ACCEL_YOUT_H);    //读 Y 加速度高字节
    accelY=(BUF[3]<<8)|BUF[2];

    BUF[4]=MPU9250_Read_Reg(ACCEL_ADDRESS,ACCEL_ZOUT_L);    //读 Z 加速度低字节
     BUF[5]=MPU9250_Read_Reg(ACCEL_ADDRESS,ACCEL_ZOUT_H);    //读 Z 加速度高字节
     accelZ=(BUF[5]<<8)|BUF[4];
}

//读取角速度数据的函数
void MPU9250_READ_GYRO(short *gyroData)
{
    u8 BUF[8];
    BUF[0]=MPU9250_Read_Reg(GYRO_ADDRESS,GYRO_XOUT_L);    //读 X 角速度低字节
    BUF[1]=MPU9250_Read_Reg(GYRO_ADDRESS,GYRO_XOUT_H);    //读 X 角速度高字节
    gyroX=(BUF[1]<<8)|BUF[0];

    BUF[2]=MPU9250_Read_Reg(GYRO_ADDRESS,GYRO_YOUT_L);    //读 Y 角速度低字节
    BUF[3]=MPU9250_Read_Reg(GYRO_ADDRESS,GYRO_YOUT_H);    //读 Y 角速度高字节
    gyroY=(BUF[3]<<8)|BUF[2];

    BUF[4]=MPU9250_Read_Reg(GYRO_ADDRESS,GYRO_ZOUT_L);    //读 Z 角速度低字节
    BUF[5]=MPU9250_Read_Reg(GYRO_ADDRESS,GYRO_ZOUT_H);    //读 Z 角速度高字节

    gyroZ=(BUF[5]<<8)|BUF[4];
}

//读取磁力计数据的函数
void MPU9250_READ_MAG(short *magData)
{
    u8 BUF[6];
    MPU9250_Write_Reg(GYRO_ADDRESS,INT_PIN_CFG,0x02); //turn on Bypass Mode
    delay_ms(10);
    MPU9250_Write_Reg(MAG_ADDRESS,0x0A,0x01);       //用来启动单次转换,否则磁力计输出的数据不变
    delay_ms(10);

    BUF[0]=MPU9250_Read_Reg(MAG_ADDRESS,MAG_XOUT_L);    //读 X 磁力计低字节
    BUF[1]=MPU9250_Read_Reg(MAG_ADDRESS,MAG_XOUT_H);    //读 X 磁力计高字节
    magX=(BUF[1]<<8)|BUF[0];

    BUF[2]=MPU9250_Read_Reg(MAG_ADDRESS,MAG_YOUT_L);    //读 Y 磁力计低字节
    BUF[3]=MPU9250_Read_Reg(MAG_ADDRESS,MAG_YOUT_H);    //读 Y 磁力计高字节
    magY=(BUF[3]<<8)|BUF[2];

    BUF[4]=MPU9250_Read_Reg(MAG_ADDRESS,MAG_ZOUT_L);    //读 Z 磁力计低字节
    BUF[5]=MPU9250_Read_Reg(MAG_ADDRESS,MAG_ZOUT_H);    //读 Z 磁力计高字节
    magZ=(BUF[5]<<8)|BUF[4];
}

void readImu()     //读取 IMU 数据函数
{
        MPU9250_READ_ACCEL(Accel);    //读取加速度
        MPU9250_READ_GYRO(Gyro);      //读取角速度
        MPU9250_READ_MAG(Mag);        //读取磁力计
}
```