/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// GY-9250 GY-9150 IIC测试程序

// 使用单片机STM32F103C8T6

// 晶振：8.00M

// 编译环境 Keil uVision4

// 时间：2012年5月1日

// 与模块连接 GPIOB6->SCL GPIOB7->SDA

// 使用：STM32F103C8T6串口1连接电脑

// 电脑串口助手显示，波特率：115200

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "stm32f10x\_lib.h"

#include <math.h> //Keil library

GPIO\_InitTypeDef GPIO\_InitStructure;

ErrorStatus HSEStartUpStatus;

#define uchar unsigned char

#define uint unsigned int

// 定义MPU9250内部地址

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define SMPLRT\_DIV 0x19 //陀螺仪采样率，典型值：0x07(125Hz)

#define CONFIG 0x1A //低通滤波频率，典型值：0x06(5Hz)

#define GYRO\_CONFIG 0x1B //陀螺仪自检及测量范围，典型值：0x18(不自检，2000deg/s)

#define ACCEL\_CONFIG 0x1C //加速计自检、测量范围及高通滤波频率，典型值：0x01(不自检，2G，5Hz)

#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

#define TEMP\_OUT\_H 0x41

#define TEMP\_OUT\_L 0x42

#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

#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

#define PWR\_MGMT\_1 0x6B //电源管理，典型值：0x00(正常启用)

#define WHO\_AM\_I 0x75 //IIC地址寄存器(默认数值0x68，只读)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define GYRO\_ADDRESS 0xD0 //陀螺地址

#define MAG\_ADDRESS 0x18 //磁场地址

#define ACCEL\_ADDRESS 0xD0

unsigned char TX\_DATA[4]; //显示据缓存区

unsigned char BUF[10]; //接收数据缓存区

char test=0; //IIC用到

short T\_X,T\_Y,T\_Z,T\_T; //X,Y,Z轴，温度

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*模拟IIC端口输出输入定义\*/

#define SCL\_H GPIOB->BSRR = GPIO\_Pin\_6

#define SCL\_L GPIOB->BRR = GPIO\_Pin\_6

#define SDA\_H GPIOB->BSRR = GPIO\_Pin\_7

#define SDA\_L GPIOB->BRR = GPIO\_Pin\_7

#define SCL\_read GPIOB->IDR & GPIO\_Pin\_6

#define SDA\_read GPIOB->IDR & GPIO\_Pin\_7

/\* 函数申明 -----------------------------------------------\*/

void RCC\_Configuration(void);

void GPIO\_Configuration(void);

void NVIC\_Configuration(void);

void USART1\_Configuration(void);

void WWDG\_Configuration(void);

void Delay(u32 nTime);

void Delayms(vu32 m);

/\* 变量定义 ----------------------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DATA\_printf(uchar \*s,short temp\_data)

{

if(temp\_data<0){

temp\_data=-temp\_data;

\*s='-';

}

else \*s=' ';

\*++s =temp\_data/100+0x30;

temp\_data=temp\_data%100; //取余运算

\*++s =temp\_data/10+0x30;

temp\_data=temp\_data%10; //取余运算

\*++s =temp\_data+0x30;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name : I2C\_GPIO\_Config

\* Description : Configration Simulation IIC GPIO

\* Input : None

\* Output : None

\* Return : None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void I2C\_GPIO\_Config(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_6;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_OD;

GPIO\_Init(GPIOB, &GPIO\_InitStructure);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_7;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_OD;

GPIO\_Init(GPIOB, &GPIO\_InitStructure);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name : I2C\_delay

\* Description : Simulation IIC Timing series delay

\* Input : None

\* Output : None

\* Return : None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void I2C\_delay(void)

{

u8 i=30; //这里可以优化速度 ，经测试最低到5还能写入

while(i)

{

i--;

}

}

void delay5ms(void)

{

int i=5000;

while(i)

{

i--;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name : I2C\_Start

\* Description : Master Start Simulation IIC Communication

\* Input : None

\* Output : None

\* Return : Wheather Start

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

bool I2C\_Start(void)

{

SDA\_H;

SCL\_H;

I2C\_delay();

if(!SDA\_read)return FALSE; //SDA线为低电平则总线忙,退出

SDA\_L;

I2C\_delay();

if(SDA\_read) return FALSE; //SDA线为高电平则总线出错,退出

SDA\_L;

I2C\_delay();

return TRUE;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name : I2C\_Stop

\* Description : Master Stop Simulation IIC Communication

\* Input : None

\* Output : None

\* Return : None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void I2C\_Stop(void)

{

SCL\_L;

I2C\_delay();

SDA\_L;

I2C\_delay();

SCL\_H;

I2C\_delay();

SDA\_H;

I2C\_delay();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name : I2C\_Ack

\* Description : Master Send Acknowledge Single

\* Input : None

\* Output : None

\* Return : None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void I2C\_Ack(void)

{

SCL\_L;

I2C\_delay();

SDA\_L;

I2C\_delay();

SCL\_H;

I2C\_delay();

SCL\_L;

I2C\_delay();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name : I2C\_NoAck

\* Description : Master Send No Acknowledge Single

\* Input : None

\* Output : None

\* Return : None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void I2C\_NoAck(void)

{

SCL\_L;

I2C\_delay();

SDA\_H;

I2C\_delay();

SCL\_H;

I2C\_delay();

SCL\_L;

I2C\_delay();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name : I2C\_WaitAck

\* Description : Master Reserive Slave Acknowledge Single

\* Input : None

\* Output : None

\* Return : Wheather Reserive Slave Acknowledge Single

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

bool I2C\_WaitAck(void) //返回为:=1有ACK,=0无ACK

{

SCL\_L;

I2C\_delay();

SDA\_H;

I2C\_delay();

SCL\_H;

I2C\_delay();

if(SDA\_read)

{

SCL\_L;

I2C\_delay();

return FALSE;

}

SCL\_L;

I2C\_delay();

return TRUE;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name : I2C\_SendByte

\* Description : Master Send a Byte to Slave

\* Input : Will Send Date

\* Output : None

\* Return : None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

void I2C\_SendByte(u8 SendByte) //数据从高位到低位//

{

u8 i=8;

while(i--)

{

SCL\_L;

I2C\_delay();

if(SendByte&0x80)

SDA\_H;

else

SDA\_L;

SendByte<<=1;

I2C\_delay();

SCL\_H;

I2C\_delay();

}

SCL\_L;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Function Name : I2C\_RadeByte

\* Description : Master Reserive a Byte From Slave

\* Input : None

\* Output : None

\* Return : Date From Slave

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

unsigned char I2C\_RadeByte(void) //数据从高位到低位//

{

u8 i=8;

u8 ReceiveByte=0;

SDA\_H;

while(i--)

{

ReceiveByte<<=1;

SCL\_L;

I2C\_delay();

SCL\_H;

I2C\_delay();

if(SDA\_read)

{

ReceiveByte|=0x01;

}

}

SCL\_L;

return ReceiveByte;

}

//ZRX

//单字节写入\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool Single\_Write(unsigned char SlaveAddress,unsigned char REG\_Address,unsigned char REG\_data) //void

{

if(!I2C\_Start())return FALSE;

I2C\_SendByte(SlaveAddress); //发送设备地址+写信号//I2C\_SendByte(((REG\_Address & 0x0700) >>7) | SlaveAddress & 0xFFFE);//设置高起始地址+器件地址

if(!I2C\_WaitAck()){I2C\_Stop(); return FALSE;}

I2C\_SendByte(REG\_Address ); //设置低起始地址

I2C\_WaitAck();

I2C\_SendByte(REG\_data);

I2C\_WaitAck();

I2C\_Stop();

delay5ms();

return TRUE;

}

//单字节读取\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char Single\_Read(unsigned char SlaveAddress,unsigned char REG\_Address)

{ unsigned char REG\_data;

if(!I2C\_Start())return FALSE;

I2C\_SendByte(SlaveAddress); //I2C\_SendByte(((REG\_Address & 0x0700) >>7) | REG\_Address & 0xFFFE);//设置高起始地址+器件地址

if(!I2C\_WaitAck()){I2C\_Stop();test=1; return FALSE;}

I2C\_SendByte((u8) REG\_Address); //设置低起始地址

I2C\_WaitAck();

I2C\_Start();

I2C\_SendByte(SlaveAddress+1);

I2C\_WaitAck();

REG\_data= I2C\_RadeByte();

I2C\_NoAck();

I2C\_Stop();

//return TRUE;

return REG\_data;

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 函数名称 ： RCC\_Configuration(void)

\*\* 函数功能 ： 时钟初始化

\*\* 输 入 ： 无

\*\* 输 出 ： 无

\*\* 返 回 ： 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void RCC\_Configuration(void)

{

/\* RCC system reset(for debug purpose) \*/

RCC\_DeInit();

/\* Enable HSE \*/

RCC\_HSEConfig(RCC\_HSE\_ON);

/\* Wait till HSE is ready \*/

HSEStartUpStatus = RCC\_WaitForHSEStartUp();

if(HSEStartUpStatus == SUCCESS)

{

/\* HCLK = SYSCLK \*/

RCC\_HCLKConfig(RCC\_SYSCLK\_Div1);

/\* PCLK2 = HCLK \*/

RCC\_PCLK2Config(RCC\_HCLK\_Div1);

/\* PCLK1 = HCLK/2 \*/

RCC\_PCLK1Config(RCC\_HCLK\_Div2);

/\* Flash 2 wait state \*/

FLASH\_SetLatency(FLASH\_Latency\_2);

/\* Enable Prefetch Buffer \*/

FLASH\_PrefetchBufferCmd(FLASH\_PrefetchBuffer\_Enable);

/\* PLLCLK = 8MHz \* 9 = 72 MHz \*/

RCC\_PLLConfig(RCC\_PLLSource\_HSE\_Div1, RCC\_PLLMul\_9);

/\* Enable PLL \*/

RCC\_PLLCmd(ENABLE);

/\* Wait till PLL is ready \*/

while(RCC\_GetFlagStatus(RCC\_FLAG\_PLLRDY) == RESET)

{

}

/\* Select PLL as system clock source \*/

RCC\_SYSCLKConfig(RCC\_SYSCLKSource\_PLLCLK);

/\* Wait till PLL is used as system clock source \*/

while(RCC\_GetSYSCLKSource() != 0x08)

{

}

}

/\* Enable GPIOA, GPIOB, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG and AFIO clocks \*/

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA | RCC\_APB2Periph\_GPIOB , ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOC | RCC\_APB2Periph\_GPIOD , ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOE | RCC\_APB2Periph\_GPIOF , ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOG | RCC\_APB2Periph\_AFIO , ENABLE);

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 函数名称 ： GPIO\_Configuration(void)

\*\* 函数功能 ： 端口初始化

\*\* 输 入 ： 无

\*\* 输 出 ： 无

\*\* 返 回 ： 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void GPIO\_Configuration(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd( RCC\_APB2Periph\_GPIOA | RCC\_APB2Periph\_GPIOB | RCC\_APB2Periph\_GPIOC | RCC\_APB2Periph\_GPIOD, ENABLE );

/\* Configure USART1 Tx (PA.09) as alternate function push-pull \*/

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9; // 选中管脚9

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP; // 复用推挽输出

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz; // 最高输出速率50MHz

GPIO\_Init(GPIOA, &GPIO\_InitStructure); // 选择A端口

/\* Configure USART1 Rx (PA.10) as input floating \*/

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_10; //选中管脚10

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING; //浮空输入

GPIO\_Init(GPIOA, &GPIO\_InitStructure); //选择A端口

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 函数名称 ： USART1\_Configuration(void)

\*\* 函数功能 ： 串口1初始化

\*\* 输 入 ： 无

\*\* 输 出 ： 无

\*\* 返 回 ： 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void USART1\_Configuration(void)

{

USART\_InitTypeDef USART\_InitStructure;

USART\_ClockInitTypeDef USART\_ClockInitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_TIM1 |RCC\_APB2Periph\_USART1, ENABLE );

USART\_ClockInitStructure.USART\_Clock = USART\_Clock\_Disable; // 时钟低电平活动

USART\_ClockInitStructure.USART\_CPOL = USART\_CPOL\_Low; // 时钟低电平

USART\_ClockInitStructure.USART\_CPHA = USART\_CPHA\_2Edge; // 时钟第二个边沿进行数据捕获

USART\_ClockInitStructure.USART\_LastBit = USART\_LastBit\_Disable; // 最后一位数据的时钟脉冲不从SCLK输出

/\* Configure the USART1 synchronous paramters \*/

USART\_ClockInit(USART1, &USART\_ClockInitStructure); // 时钟参数初始化设置

USART\_InitStructure.USART\_BaudRate = 115200; // 波特率为：115200

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b; // 8位数据

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1; // 在帧结尾传输1个停止位

USART\_InitStructure.USART\_Parity = USART\_Parity\_No ; // 奇偶失能

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None; // 硬件流控制失能

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx; // 发送使能+接收使能

/\* Configure USART1 basic and asynchronous paramters \*/

USART\_Init(USART1, &USART\_InitStructure);

/\* Enable USART1 \*/

USART\_ClearFlag(USART1, USART\_IT\_RXNE); //清中断，以免一启用中断后立即产生中断

USART\_ITConfig(USART1,USART\_IT\_RXNE, ENABLE); //使能USART1中断源

USART\_Cmd(USART1, ENABLE); //USART1总开关：开启

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 函数名称 ： NVIC\_Configuration(void)

\*\* 函数功能 ： 中断初始化

\*\* 输 入 ： 无

\*\* 输 出 ： 无

\*\* 返 回 ： 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void NVIC\_Configuration(void)

{

NVIC\_InitTypeDef NVIC\_InitStructure;

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_0);

NVIC\_InitStructure.NVIC\_IRQChannel = WWDG\_IRQChannel;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0;

NVIC\_Init(&NVIC\_InitStructure);

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 函数名称 ： WWDG\_Configuration(void)

\*\* 函数功能 ： 看门狗初始化

\*\* 输 入 ： 无

\*\* 输 出 ： 无

\*\* 返 回 ： 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void WWDG\_Configuration(void)

{

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_WWDG, ENABLE);

WWDG\_SetPrescaler(WWDG\_Prescaler\_8); // WWDG clock counter = (PCLK1/4096)/8 = 244 Hz (~4 ms)

WWDG\_SetWindowValue(0x41); // Set Window value to 0x41

WWDG\_Enable(0x50); // Enable WWDG and set counter value to 0x7F, WWDG timeout = ~4 ms \* 64 = 262 ms

WWDG\_ClearFlag(); // Clear EWI flag

WWDG\_EnableIT(); // Enable EW interrupt

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 函数名称 ： Delay(vu32 nCount)

\*\* 函数功能 ： 延时函数

\*\* 输 入 ： 无

\*\* 输 出 ： 无

\*\* 返 回 ： 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void Delay(vu32 nCount)

{

for(; nCount != 0; nCount--);

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 函数名称 ： void Delayms(vu32 m)

\*\* 函数功能 ： 长延时函数 m=1,延时1ms

\*\* 输 入 ： 无

\*\* 输 出 ： 无

\*\* 返 回 ： 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void Delayms(vu32 m)

{

u32 i;

for(; m != 0; m--)

for (i=0; i<50000; i++);

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 函数名称 ： WWDG\_IRQHandler(void)

\*\* 函数功能 ： 窗口提前唤醒中断

\*\* 输 入 ： 无

\*\* 输 出 ： 无

\*\* 返 回 ： 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void WWDG\_IRQHandler(void)

{

/\* Update WWDG counter \*/

WWDG\_SetCounter(0x50);

/\* Clear EWI flag \*/

WWDG\_ClearFlag();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void USART1\_SendData(uchar SendData)

{

USART\_SendData(USART1, SendData);

Delayms(1);

}

//初始化MPU9250，根据需要请参考pdf进行修改\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Init\_MPU9250(void)

{

/\*

Single\_Write(GYRO\_ADDRESS,PWR\_M, 0x80); //

Single\_Write(GYRO\_ADDRESS,SMPL, 0x07); //

Single\_Write(GYRO\_ADDRESS,DLPF, 0x1E); //±2000°

Single\_Write(GYRO\_ADDRESS,INT\_C, 0x00 ); //

Single\_Write(GYRO\_ADDRESS,PWR\_M, 0x00); //

\*/

Single\_Write(GYRO\_ADDRESS,PWR\_MGMT\_1, 0x00); //解除休眠状态

Single\_Write(GYRO\_ADDRESS,SMPLRT\_DIV, 0x07);

Single\_Write(GYRO\_ADDRESS,CONFIG, 0x06);

Single\_Write(GYRO\_ADDRESS,GYRO\_CONFIG, 0x18);

Single\_Write(GYRO\_ADDRESS,ACCEL\_CONFIG, 0x01);

//----------------

// Single\_Write(GYRO\_ADDRESS,0x6A,0x00);//close Master Mode

}

//\*\*\*\*\*\*读取MPU9250数据\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void READ\_MPU9250\_ACCEL(void)

{

BUF[0]=Single\_Read(ACCEL\_ADDRESS,ACCEL\_XOUT\_L);

BUF[1]=Single\_Read(ACCEL\_ADDRESS,ACCEL\_XOUT\_H);

T\_X= (BUF[1]<<8)|BUF[0];

T\_X/=164; //读取计算X轴数据

BUF[2]=Single\_Read(ACCEL\_ADDRESS,ACCEL\_YOUT\_L);

BUF[3]=Single\_Read(ACCEL\_ADDRESS,ACCEL\_YOUT\_H);

T\_Y= (BUF[3]<<8)|BUF[2];

T\_Y/=164; //读取计算Y轴数据

BUF[4]=Single\_Read(ACCEL\_ADDRESS,ACCEL\_ZOUT\_L);

BUF[5]=Single\_Read(ACCEL\_ADDRESS,ACCEL\_ZOUT\_H);

T\_Z= (BUF[5]<<8)|BUF[4];

T\_Z/=164; //读取计算Z轴数据

}

void READ\_MPU9250\_GYRO(void)

{

BUF[0]=Single\_Read(GYRO\_ADDRESS,GYRO\_XOUT\_L);

BUF[1]=Single\_Read(GYRO\_ADDRESS,GYRO\_XOUT\_H);

T\_X= (BUF[1]<<8)|BUF[0];

T\_X/=16.4; //读取计算X轴数据

BUF[2]=Single\_Read(GYRO\_ADDRESS,GYRO\_YOUT\_L);

BUF[3]=Single\_Read(GYRO\_ADDRESS,GYRO\_YOUT\_H);

T\_Y= (BUF[3]<<8)|BUF[2];

T\_Y/=16.4; //读取计算Y轴数据

BUF[4]=Single\_Read(GYRO\_ADDRESS,GYRO\_ZOUT\_L);

BUF[5]=Single\_Read(GYRO\_ADDRESS,GYRO\_ZOUT\_H);

T\_Z= (BUF[5]<<8)|BUF[4];

T\_Z/=16.4; //读取计算Z轴数据

// BUF[6]=Single\_Read(GYRO\_ADDRESS,TEMP\_OUT\_L);

// BUF[7]=Single\_Read(GYRO\_ADDRESS,TEMP\_OUT\_H);

// T\_T=(BUF[7]<<8)|BUF[6];

// T\_T = 35+ ((double) (T\_T + 13200)) / 280;// 读取计算出温度

}

void READ\_MPU9250\_MAG(void)

{

Single\_Write(GYRO\_ADDRESS,0x37,0x02);//turn on Bypass Mode

Delayms(10);

Single\_Write(MAG\_ADDRESS,0x0A,0x01);

Delayms(10);

BUF[0]=Single\_Read (MAG\_ADDRESS,MAG\_XOUT\_L);

BUF[1]=Single\_Read (MAG\_ADDRESS,MAG\_XOUT\_H);

T\_X=(BUF[1]<<8)|BUF[0];

BUF[2]=Single\_Read(MAG\_ADDRESS,MAG\_YOUT\_L);

BUF[3]=Single\_Read(MAG\_ADDRESS,MAG\_YOUT\_H);

T\_Y= (BUF[3]<<8)|BUF[2];

//读取计算Y轴数据

BUF[4]=Single\_Read(MAG\_ADDRESS,MAG\_ZOUT\_L);

BUF[5]=Single\_Read(MAG\_ADDRESS,MAG\_ZOUT\_H);

T\_Z= (BUF[5]<<8)|BUF[4];

//读取计算Z轴数据

}

//\*\*\*\*\*\*\*\*串口发送数据\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Send\_data(uchar MAG,uchar axis)

{uchar i;

USART1\_SendData(MAG);

USART1\_SendData(axis);

USART1\_SendData(':');

for(i=0;i<4;i++)USART1\_SendData(TX\_DATA[i]);

USART1\_SendData(' ');

USART1\_SendData(' ');

}

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 函数名称 ： main(void)

\*\* 函数功能 ： 主函数

\*\* 输 入 ： 无

\*\* 输 出 ： 无

\*\* 返 回 ： 无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

int main(void)

{

RCC\_Configuration(); //配置RCC

GPIO\_Configuration(); //配置GPIO

USART1\_Configuration(); //配置串口1

I2C\_GPIO\_Config(); //配置IIC使用端口

Delayms(10); //延时

Init\_MPU9250(); //初始化MPU9250

while(1)

{

READ\_MPU9250\_ACCEL(); //加速度

DATA\_printf(TX\_DATA,T\_X);//转换X轴数据到数组

Send\_data('A','X'); //发送X轴数

DATA\_printf(TX\_DATA,T\_Y);//转换Y轴数据到数组

Send\_data('A','Y'); //发送Y轴数

DATA\_printf(TX\_DATA,T\_Z);//转换Z轴数据到数组

Send\_data('A','Z'); //发送Z轴数

READ\_MPU9250\_GYRO(); //陀螺

DATA\_printf(TX\_DATA,T\_X);//转换X轴数据到数组

Send\_data('G','X'); //发送X轴数

DATA\_printf(TX\_DATA,T\_Y);//转换Y轴数据到数组

Send\_data('G','Y'); //发送Y轴数

DATA\_printf(TX\_DATA,T\_Z);//转换Z轴数据到数组

Send\_data('G','Z'); //发送Z轴数

READ\_MPU9250\_MAG(); //磁场

DATA\_printf(TX\_DATA,T\_X);//转换X轴数据到数组

Send\_data('M','X'); //发送X轴数

DATA\_printf(TX\_DATA,T\_Y);//转换Y轴数据到数组

Send\_data('M','Y'); //发送Y轴数

DATA\_printf(TX\_DATA,T\_Z);//转换Z轴数据到数组

Send\_data('M','Z'); //发送Z轴数

/\*

DATA\_printf(TX\_DATA,T\_T);//转换温度数据到数组

Send\_data('T'); //发送温度数据

\*/

USART1\_SendData(0X0D); //换行

USART1\_SendData(0X0A); //回车

Delayms(10); //延时

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/