#include <stdio.h>

#include "stm32f10x\_gpio.h"

#include "ADS1256.h"

void SPI2\_Init(void)

{

SPI\_InitTypeDef SPI\_InitStructure;

GPIO\_InitTypeDef GPIO\_InitStructure;

/\*\*\*\*Initial SPI2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Enable SPI2 and GPIOB clocks \*/

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

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_SPI2, ENABLE);

/\* Configure SPI2 pins: NSS, SCK, MISO and MOSI \*/

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_13 | GPIO\_Pin\_14 | GPIO\_Pin\_15;

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

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

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

//SPI2 NSS

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_12;

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

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

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

GPIO\_SetBits(GPIOB, GPIO\_Pin\_12);

/\* SPI2 configuration \*/

SPI\_InitStructure.SPI\_Direction = SPI\_Direction\_2Lines\_FullDuplex; //SPI1设置为两线全双工

SPI\_InitStructure.SPI\_Mode = SPI\_Mode\_Master; //设置SPI2为主模式

SPI\_InitStructure.SPI\_DataSize = SPI\_DataSize\_8b; //SPI发送接收8位帧结构

SPI\_InitStructure.SPI\_CPOL = SPI\_CPOL\_Low; //串行时钟在不操作时，时钟为低电平

SPI\_InitStructure.SPI\_CPHA = SPI\_CPHA\_2Edge; //第一个时钟沿开始采样数据

SPI\_InitStructure.SPI\_NSS = SPI\_NSS\_Soft; //NSS信号由软件（使用SSI位）管理

SPI\_InitStructure.SPI\_BaudRatePrescaler = SPI\_BaudRatePrescaler\_256; //定义波特率预分频的值:波特率预分频值为8

SPI\_InitStructure.SPI\_FirstBit = SPI\_FirstBit\_MSB; //数据传输从MSB位开始

SPI\_InitStructure.SPI\_CRCPolynomial = 7; //CRC值计算的多项式

SPI\_Init(SPI2, &SPI\_InitStructure);

/\* Enable SPI2 \*/

SPI\_Cmd(SPI2, ENABLE);

}

//初始化ADS1256 GPIO

void Init\_ADS1256\_GPIO(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_ADS1256Reset | RCC\_ADS1256DRDY, ENABLE);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_RCC\_ADS1256Reset; //GPIO\_Pin\_11

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP; //推挽输出

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

GPIO\_Init(GPIO\_RCC\_ADS1256Reset\_PORT, &GPIO\_InitStructure);

GPIO\_ResetBits(GPIO\_RCC\_ADS1256Reset\_PORT, GPIO\_RCC\_ADS1256Reset );

GPIO\_InitStructure.GPIO\_Pin = GPIO\_ADS1256DRDY; //GPIO\_Pin\_10

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU; //带上拉输入模式

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

GPIO\_Init(GPIO\_ADS1256DRDY\_PORT, &GPIO\_InitStructure);

SPI2\_Init();

}

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

// 功 能： 模拟SPI通信（与外设SPI通信时，如果单片机没有硬件SPI接口， 可以用 普通IO模拟SPI时序 作单片机的软SPI接口）

// 入口参数: / 发送的SPI数据

// 出口参数: / 接收的SPI数据

// 全局变量: /

// 备 注: 发送接收函数

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

unsigned char SPI\_WriteByte(unsigned char TxData)

{

unsigned char RxData=0;//无符号字符变量

while(SPI\_I2S\_GetFlagStatus(SPI2,SPI\_I2S\_FLAG\_TXE)==RESET); //检查发送是否完成，完成以后再发送数据

SPI\_I2S\_SendData(SPI2,TxData);//发送数据

while(SPI\_I2S\_GetFlagStatus(SPI2,SPI\_I2S\_FLAG\_RXNE)==RESET);//检查接收标志位，是否可以接收

RxData=SPI\_I2S\_ReceiveData(SPI2);//接收数据

return RxData;//返回通过SPIx最近接收的数据

}

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

// 功 能：ADS1256 写数据

// 入口参数: /

// 出口参数: /

//参数1：要写入的寄存器的地址

//参数2：要写入寄存器的内容

// 全局变量: /

// 备 注: 向ADS1256中地址为regaddr的寄存器写入一个字节databyte

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

//DRDY位为转换数据状态位，此位完全复制DRDY引脚的状态，DRDY低电平时，表明数据转换结束，结果可以读出;高电平时，表明没有数据转换完成或者正在转换数据，此时不能读数据

void ADS1256WREG(unsigned char regaddr,unsigned char databyte)//ADS1256WREG（）的实现，先写地址在写数据 ADS1256WREG(xx,xx)作用是向某寄存器写入数据

{

GPIO\_ResetBits(GPIOB, GPIO\_Pin\_12);//PB12连接ADS1256片选引脚，低电平有效

// while(GPIO\_ReadInputDataBit(GPIO\_ADS1256DRDY\_PORT,GPIO\_ADS1256DRDY));//当ADS1256\_DRDY为低时才能写寄存器，高电平空循环

SPI\_WriteByte(ADS1256\_CMD\_WREG | (regaddr & 0x0F));//向寄存器写入数据地址，例如0x51：前一个字节为5H是写寄存器指令，后一个字节为1H是要写的寄存器地址

SPI\_WriteByte(0x00);//写入数据的个数n-1

SPI\_WriteByte(databyte);//向regaddr地址指向的寄存器写入数据databyte

GPIO\_SetBits(GPIOB, GPIO\_Pin\_12);

}

//初始化ADS1256

void ADS1256\_Init(void)

{

ADS1256WREG(ADS1256\_STATUS,0x04); // 状态寄存器 高位在前、校准、使用缓冲0x06 不使用缓冲0x04 ADS1256WREG（）是向某寄存器写入数据

// ADS1256WREG(ADS1256\_MUX,0x08); // 状态寄存器 初始化端口A0为‘+’，AINCOM位‘-’

ADS1256WREG(ADS1256\_ADCON,0x00); // AD控制寄存器 放大倍数1

ADS1256WREG(ADS1256\_DRATE,ADS1256\_DRATE\_100SPS); // AD数据速率 速度100sps（每秒样本数）

ADS1256WREG(ADS1256\_IO,0x00); // 通用数字IO口

}

//读取AD值

unsigned int ADS1256ReadData()

{

unsigned char i=0;

unsigned int sum=0;

unsigned int r=0;

GPIO\_ResetBits(GPIOB, GPIO\_Pin\_12);

// while(GPIO\_ReadInputDataBit(GPIO\_ADS1256DRDY\_PORT,GPIO\_ADS1256DRDY)); //当ADS1256\_DRDY为低时才能写寄存器

SPI\_WriteByte(ADS1256\_CMD\_SYNC);//外同步AD信号，启动转换

SPI\_WriteByte(ADS1256\_CMD\_WAKEUP);//完成同步并退出待机模式

SPI\_WriteByte(ADS1256\_CMD\_RDATA);//读取数据

sum |= (SPI\_WriteByte(0xff) << 16);//一共传输24位数据，一次传输8位

sum |= (SPI\_WriteByte(0xff) << 8);

sum |= SPI\_WriteByte(0xff);

GPIO\_SetBits(GPIOB, GPIO\_Pin\_12); //拉高引脚输出电平

return sum;

}

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

// 功 能：读取ADS1256单路数据

// 入口参数: /

// 出口参数: /

// 全局变量: /

// 备 注: /

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

unsigned int ADS\_sum(unsigned char channel)//先选择通道再读取数据

{

ADS1256WREG(ADS1256\_MUX,channel); //设置通道，参数1:0x01寄存器地址，参数2:0x08写入的数据，要读取电压的通道

return ADS1256ReadData();//读取AD值，返回24位数据。

}