气压传感器

1.数据采集

本实验选用的气压传感器型号为MPL3115A2。MPL3115A2具有三项主要功能。一是测量环境大气压，二是测量高度，三是测量温度。

气压传感器与主设备通信使用IIC协议，硬件设计主要是把气压传感器的数据线、时钟线与CC2531连接起来，其中数据线SDA与CC2531的P0\_1相连，时钟线SCL与CC2531的P0\_0相连。CC2531向气压传感器写入数据的地址为0xC0，读取数据的地址为0xC1。

在采集气压传感器数据之前，必须对其进行初始化，初始化主要是配置控制寄存器0x26，根据数据手册，把寄存器0x26的值配置为0x39。寄存器位分布如图所示：



图1-1控制寄存器0x26的位分布

其中，ALT位配置为0时，传感器工作于气压模式，ALT位配置为1时，传感器工作于高度模式。SBYB位配置为0时，传感器工作于standby模式，SBYB位配置为1时，传感器工作于active模式。因此，启动传感器工作，必须把SBYB位配置为1。初始化气压传感器之后，就可以采集数据了，使用I2C读数据函数从寄存器0x01、0x02、0x03中读取气压值，从寄存器0x04、0x05中读取温度值。

气压值表达为三个字节，例如，寄存器0x01的值为0x61, 寄存器0x02的值为0xF8, 寄存器0x03的值为0xE0。应用时将这三个字节组合成0x61F8E，即20位二进制位，丢弃最后四位二进制位。0x61F8E = 401294，此值401294 \* 0.25 = 100323.5（帕斯卡）。温度值表达为两个字节，例如，寄存器0x04的值为0x1C，寄存器0x05的值为0x50。应用时将这两个字节组合成0x1C5，即12位二进制位，丢弃最后四位二进制位。0x1C5 = 453，此值 453 \* 0.0625 = 28.3125℃。

2.数据上传

CC2531从气压传感器读取数据的地址为0xC1，之后通过CC2531与ARM网关之间进行的zigbee通信，将读到的数据上传至上位机。

3.非功能性需求

3.1精度

3.1.1数据采集精度

压力的采集精度为20-比特（Pascals），高度的采集精度为20-比特（m），温度的采集精度为12-比特（摄氏度），测量精度可达30cm。

3.1.2用户数据显示精度

压力显示精度为0.01Pa，灵敏度为0.25Pa。高度显示精度为0.0001m，灵敏度为0.0625m。温度显示精度为0.0001m，灵敏度为0.0625m。

3.2传感器采集频率

MPL3115A2具有灵活的采样频率，最高可达128Hz。

3.3灵活性

MPL3115A2可由用户设定阈值和窗口，其中，阈值是用户可编程限定采样数据的最大和最小极限值，窗口是设定采样数据在阈值上下浮动的范围。设定阈值和窗口后，用户可选择希望发生的中断提醒事件。同时用户也可通过编程设定水印，即FIFO采样次数，当采样次数达到水印值即可触发中断。

4.数据要求

各个寄存器存储包括整数和小数部分在内的气压、高度、温度数据，因此需要通过相应函数对数据进行格式转换，通过内部补偿输出直观的数值。

5.故障处理要求

5.1系统节点故障原因

芯片长时间使用会导致指针产生偏置，造成数据出现偏差、精度不够等问题。

5.2故障检测处理方案

MPL3115A2提供了OFF\_P、OFF\_T、OFF\_H三个寄存器，可对气压、温度、海拔的输出数据进行偏置修正。

6.源码阅读

#include "ioCC2530.h"

#include <string.h>

#include <stdio.h>

#define uint unsigned int

#define uchar unsigned char

#define TRUE 1

#define FALSE 0

#define SCL P0\_0 //时钟线

#define SDA P0\_1 //数据线

#define I2C\_WRITE 0x00 //\*T读

#define I2C\_READ 0x01 //\*R写

#define MPL3115A2\_I2C\_ADDRESS 0xC0 //CC2531向气压传感器写入数据的地址为0xC0

//定义控制灯的端口

#define YLED P2\_0 //定义LED1为P2\_0口控制

#define RLED P0\_5 //定义LED2为P0\_5口控制

char data\_result[6] = {0x55,0x55,0x55,0x55,0x55,0x55};

long int pressure\_temp = 0; //压力

char pressure\_result[15] = {0};

float pressure\_value = 0;

int temp\_temp = 0; //温度

char temp\_result[15] = {0};

float temp\_value = 0;

char INT\_FLAG[1] = {0x00};

int j = 0; //调试用

void Init\_IO\_INT(void);

void Delay\_1u(uint);

void WriteSDA1(void); //SDA为数据线

void WriteSDA0(void);

void WriteSCL1(void); //SCL为时钟线

void WriteSCL0(void);

void ReadSDA(void);

void I2C\_Start(void);

void I2C\_Stop(void);

void SEND\_0(void);

void SEND\_1(void);

char Check\_Acknowledge(void);

void WriteI2CByte(char);

char ReadI2CByte(void);

void Read\_Data(char, char \*);

void Write\_Data(char, char);

void initUARTtest(void);

void UartTX\_Send\_String(char \*Data,int len);

void initLed(void);

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

//主函数

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

void main(void)

{

initLed();

initUARTtest(); //初始化串口

Write\_Data(0x26, 0x39); //初始化传感器

while(1)

{

//分别读取传感器的5个寄存器，存储到data\_result数组中

Read\_Data(0x01, &data\_result[1]);

Read\_Data(0x02, &data\_result[2]);

Read\_Data(0x03, &data\_result[3]);

Read\_Data(0x04, &data\_result[4]);

Read\_Data(0x05, &data\_result[5]);

// 将压力的20位数据合并，并换算单位

pressure\_temp = data\_result[1];

pressure\_temp = pressure\_temp << 8;

pressure\_temp = pressure\_temp + data\_result[2];

pressure\_temp = pressure\_temp << 8;

pressure\_temp = pressure\_temp + data\_result[3];

pressure\_temp = pressure\_temp >> 4; // 第3个寄存器只有前4位是压力数据

pressure\_value = pressure\_temp \* 0.25;

// 格式化字符串并发送

sprintf(pressure\_result,"%s","pressure: ");

UartTX\_Send\_String(pressure\_result, 10);

sprintf(pressure\_result,"%9.2f",pressure\_value);

pressure\_result[9] = ':';

UartTX\_Send\_String(pressure\_result, 10);

// 从两个寄存器中读取温度数据并换算单位

temp\_temp = data\_result[4];

temp\_temp = temp\_temp << 8;

temp\_temp = temp\_temp + data\_result[5];

temp\_temp = temp\_temp >> 4;

temp\_value = temp\_temp \* 0.0625;

if (temp\_value > 22)

{

for(uchar i=0; i < 3; i++)

{

YLED = !YLED;

RLED = !RLED;

Delay\_1u(1000);

}

}

// 格式化温度数据并发送

sprintf(pressure\_result,"%s","temperature: ");

UartTX\_Send\_String(pressure\_result, 13);

sprintf(temp\_result, "%8.4f", temp\_value);

temp\_result[8] = '\n';

UartTX\_Send\_String(temp\_result, 9);

}

}

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

//io及中断初始化

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

void Init\_IO\_INT(void)

{

P1SEL &= 0xFD; //P11作为普通IO

P1DIR &= 0xFD; //P11作为输入

P1INP &= 0xFD;//P11有上拉、下拉

PICTL |= 0X02; //下降沿

EA = 1;

IEN2 |= 0X10; // P1IE = 1;

P1IEN |= 0X02; //使能P1\_1中断

P1IF = 0; //清中断标志

P1IFG = 0; //清中断标志

}

#pragma vector = P1INT\_VECTOR

\_\_interrupt void P1\_ISR(void)

{

if((P1IFG & 0x02) != 0) //P1\_1中断

{

P1IFG &= 0xFD;

Read\_Data(0x02, &INT\_FLAG[0]);

Write\_Data(0x02, INT\_FLAG[0] & 0x77);

// Read\_Data(0x08, &data\_result[1]);

// Read\_Data(0x09, &data\_result[2]);

// Read\_Data(0x0A, &data\_result[3]);

j++; //调试用

}

P1IFG = 0;//清中断标志

P1IF = 0; //清中断标志

}

void Delay\_1u(unsigned int microSecs) {

while(microSecs--)

{

/\* 32 NOPs == 1 usecs \*/

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop"); asm("nop"); asm("nop"); asm("nop");

asm("nop"); asm("nop");

}

}

void WriteSDA1(void)//SDA 输出1

{

P0DIR |= 0x02;

SDA = 1;

}

void WriteSDA0(void)//SDA 输出0

{

P0DIR |= 0x02;

SDA = 0;

}

void WriteSCL1(void)//SCL 输出1

{

P0DIR |= 0x01;

SCL = 1;

}

void WriteSCL0(void)//SCL 输出0

{

P0DIR |= 0x01;

SCL = 0;

}

void ReadSDA(void)//这里设置SDA对应IO口DIR可以接收数据

{

P0DIR &= 0xFD;

}

/\*启动I2C总线的函数，当SCL为高电平时使SDA产生一个负跳变\*/

void I2C\_Start(void)

{

WriteSDA1();

WriteSCL1();

Delay\_1u(50);

WriteSDA0();

Delay\_1u(50);

WriteSCL0();

Delay\_1u(50);

}

/\*终止I2C总线，当SCL为高电平时使SDA产生一个正跳变\*/

void I2C\_Stop(void)

{

WriteSDA0();

Delay\_1u(50);

WriteSCL1();

Delay\_1u(50);

WriteSDA1();

Delay\_1u(50);

WriteSCL0();

Delay\_1u(50);

}

/\*发送0，在SCL为高电平时使SDA信号为低\*/

void SEND\_0(void)

{

WriteSDA0();

WriteSCL1();

Delay\_1u(50);

WriteSCL0();

Delay\_1u(50);

}

/\*发送1，在SCL为高电平时使SDA信号为高\*/

void SEND\_1(void)

{

WriteSDA1();

WriteSCL1();

Delay\_1u(50);

WriteSCL0();

Delay\_1u(50);

}

/\*发送完一个字节后检验设备的应答信号\*/

char Check\_Acknowledge(void)

{

WriteSDA1();

WriteSCL1();

Delay\_1u(50);

F0=SDA;

Delay\_1u(50);

WriteSCL0();

Delay\_1u(50);

if(F0==1)

return FALSE;

return TRUE;

}

/\*向I2C总线写一个字节\*/

void WriteI2CByte(char b)

{

char i;

//逐位发送字节

for(i=0;i<8;i++)

{

//判断第i位是1还是0

if((b<<i)&0x80)

{

SEND\_1();

}

else

{

SEND\_0();

}

}

}

/\*从I2C总线读一个字节\*/

char ReadI2CByte(void)

{

char b=0,i;

//开始读取

WriteSDA1();

for(i=0;i<8;i++)

{

WriteSCL0();

Delay\_1u(50);

WriteSCL1();

Delay\_1u(50);

ReadSDA();

F0=SDA;//寄存器中的一位,用于存储SDA中的一位数据

//判断0和1，并将之前已读取的部分左移，若1，将末位置为1，若0，则左移过程中已自动置为0

if(F0==1)

{

b=b<<1;

b=b|0x01;

}

else

b=b<<1;

}

//结束读取

WriteSCL0();

return b;

}

void Read\_Data(char reg, char \*data)

{

// 启动I2C总线，当SCL为高电平时使SDA产生一个负跳变

I2C\_Start();

// 通知传感器要写数据

WriteI2CByte(MPL3115A2\_I2C\_ADDRESS + I2C\_WRITE);

// 等待传感器确认接收

while(Check\_Acknowledge() == FALSE);

// 发送要读取的寄存器

WriteI2CByte(reg);

// 等待传感器响应

while(Check\_Acknowledge() == FALSE);

// 再次开启I2C会话

I2C\_Start();

// 通知传感器要读数据

WriteI2CByte(MPL3115A2\_I2C\_ADDRESS + I2C\_READ);

// 等待传感器响应

while(Check\_Acknowledge() == FALSE);

// 读取数据并写进\*data

\*data = ReadI2CByte();

// 关闭I2C会话

I2C\_Stop();

}

void Write\_Data(char reg, char data)

{

I2C\_Start();

WriteI2CByte(MPL3115A2\_I2C\_ADDRESS + I2C\_WRITE);

while(Check\_Acknowledge() == FALSE);

WriteI2CByte(reg);

while(Check\_Acknowledge() == FALSE);

WriteI2CByte(data);

while(Check\_Acknowledge() == FALSE);

I2C\_Stop();

}

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

\*函数功能 ：初始化串口1

\*入口参数 ：无

\*返 回 值 ：无

\*说 明 ：57600-8-n-1

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

void initUARTtest(void)

{

CLKCONCMD &= ~0x40; //晶振

while(!(SLEEPSTA & 0x40)); //等待晶振稳定

CLKCONCMD &= ~0x47; //TICHSPD128分频，CLKSPD不分频

SLEEPSTA |= 0x04; //关闭不用的RC振荡器

PERCFG = 0x01; //位置1 串口0

P1SEL |= 0x30; //P1用作串口

U0CSR |= 0x80; //UART方式

U0GCR |= 8; //baud\_e

U0BAUD |= 59; //波特率设为9600

UTX0IF = 1;

U0CSR |= 0X40; //允许接收

IEN0 |= 0x84; //开总中断，接收中断

}

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

\*函数功能 ：串口发送字符串函数

\*入口参数 : data:数据

\* len :数据长度

\*返 回 值 ：无

\*说 明 ：

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

void UartTX\_Send\_String(char \*Data,int len)

{

int j;

for(j=0;j<len;j++)

{

U0DBUF = \*Data++;

while(UTX0IF == 0);

UTX0IF = 0;

}

}

void initLed(void)

{

P0DIR |= 0x20; //P0\_5定义为输出

P2DIR |= 0x01; //P2\_0定义为输出

RLED = 1;

YLED = 1; //LED灯灭

}