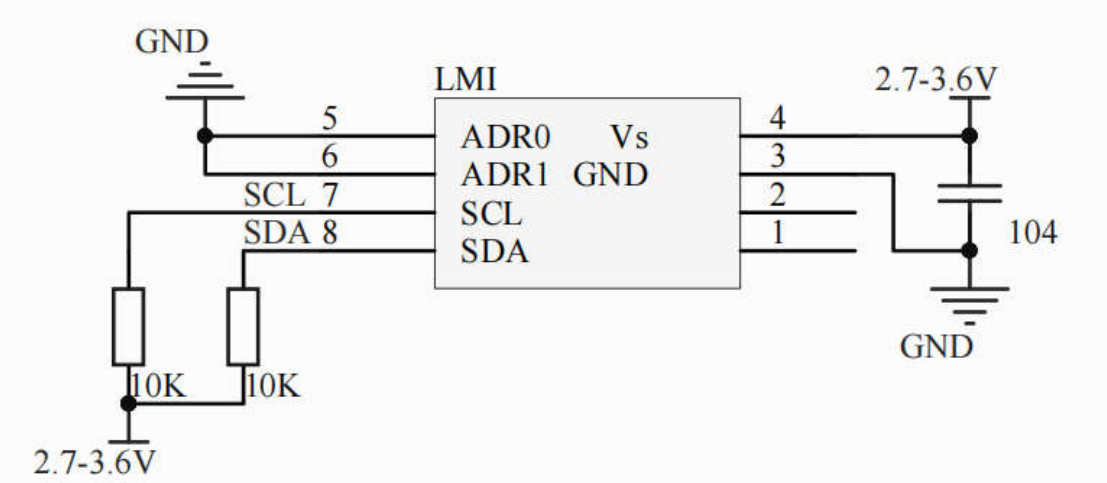
LMI系列中文应用笔记

1. 硬件连接

SDA, SCL 连接到单片机对应 IO 口，工作电压不能超过 3.6V。建议采用单片机 IO 口模拟 I2C 总线进行测试。



ADR0，ADR1 建议接地，这时传感器地址为 1011100(0x5C)， 在 7 位地址后面增加 1 位 R/W, 则变成 0xB8(W)、0xB9(R)。

MSB							LSB
1	0	1	1	1	ADR1	ADR0	R/W

2. 计算公式

读取的压力及温度原始值小于 0x8000 说明为正数，大于等于 0x8000 为负数。负数为补码存储，可以通过减去 0x10000 获得带符号整数的表示方法。计算的结果单位为 Pa。

```
if(temp >= 0x8000)   //正负判断
    temp -= 0x10000;
 if(press >= 0x8000) //负数判断
    press -= 0x10000;
 fTemp = (float)temp/72+20; //温度计算
 fPress= (float)press/Scale factor; //压力计算
```

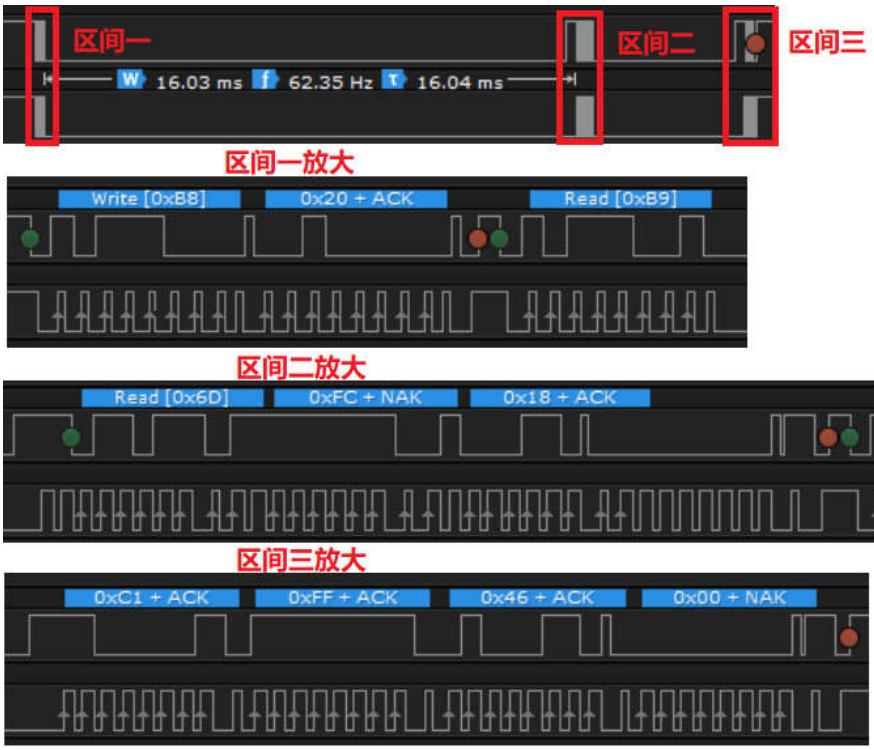
型号	LMIS025	LMIS050	LMIS100	LMIS250	LMIS500	LMIM012	LMIM025	LMIM050
量程	25 Pa	50 Pa	100 Pa	250 Pa	500 Pa	1250 Pa	2500 Pa	5000 Pa
Scale factor	1200	600	300	120	60	24	12	6

3. I2C 通信协议

3.1 协议格式

需要注意的是第一次读取数据的准备时间要求 16mS。如果数据准备好，SDA 线会被传感器拉高。从第二次开始不需要发送 0xB8，0x20 命令，只需要发送 0xB9 命令。可以通过配置 0xB9 命令与上一次压力数据之间时间间隔控制下一次数据准备时间，间隔在 0.7ms 以内数据准备时间为 5mS，间隔大于 0.7ms 数据准备时间 16ms。

如果改变时间间隔，需要从新发送 0xB8，0x20 命令，否则会引起 I2C 总线数据混乱。



3.2 数据存储格式

连续读取 4 字节传感器数据时，传感器发送格式如下，需要注意高低位的变换。

Byte 1	Byte 2	Byte 3	Byte 4
Pressure		Temperature	
Signed 15-bit value		Signed 16-bit value after offset subtraction and correction	
LSB	MSB	LSB	MSB

3.3 测试程序：

```
BSP_IIC_Start();
BSP_IIC_ByteWrite(0xB8);
BSP1_IIC_ByteWrite(0x20);
BSP_IIC_Stop();
BSP_IIC_Start();
BSP_IIC_ByteWrite(0xB9);
_Delay_us(14000);   //延时 16ms
BSP_IIC_ByteRead(2,1);          press= ((I2C_Byte>>8) | ((I2C_Byte&0xFF)<<8));
BSP_IIC_ByteRead(2,0);          temp = ((I2C_Byte>>8) | ((I2C_Byte&0xFF)<<8));
BSP_IIC_Stop();
_Delay_us(100); //间隔在 0.7ms 以内数据准备时间为 5mS，间隔大于 0.7ms 数据准备时间 16ms
BSP1_IIC_Start();
```

```
BSP_IIC_ByteWrite(0xB9);
_Delay_us(4000); //延时 5ms
BSP_IIC_ByteRead(2,1);          press= ((I2C_Byte>>8) | ((I2C_Byte&0xFF)<<8));
BSP_IIC_ByteRead(2,0);          temp = ((I2C_Byte>>8) | ((I2C_Byte&0xFF)<<8));
BSP1_IIC_Stop();
```

4. 底层驱动

4.1 I2C 启动命令，I2C 开始与设备通信

```
void _BSP_IIC_Start(void)
{
    SDA_SetOutput();
    SCL_SetVal();
    SDA_SetVal();  _Delay_us(2);
    SDA_ClrVal();  _Delay_us(2);
    SCL_ClrVal();  _Delay_us(2);
}
```

4.2 停止总线，结束与总线的通信

```
void _BSP_IIC_Stop(void)
{
    SDA_SetOutput();
    SCL_ClrVal();
    SDA_ClrVal();  _Delay_us(2);
    SCL_SetVal();  _Delay_us(2);
    SDA_SetVal();  _Delay_us(2);
}
```

4.3 写入一个 Byte 的数据

```
void _BSP_IIC_ByteWrite(uint8_t dat)
{
    uint8_t i;
    for(i=0; i<8; i++)
    {
        if(dat & 0x80)
        {           SDA_SetVal();       }
        else
        {           SDA_ClrVal();       }
        dat <<= 1;          _Delay_us(2);
        SCL_SetVal();       _Delay_us(2);
        SCL_ClrVal();       _Delay_us(2);
```

```
        }
        SDA_SetInput();
        SCL_SetVal();       _Delay_us(2);
        SCL_ClrVal();       _Delay_us(2);
        SDA_SetOutput();
    }
```

4.4 读取 lByte（小于等于 4）个字节的数据，以 ack 结尾。

```
void  BSP_IIC_ByteRead(char lByte,char ack)
{
    char i,j;
    I2C_Byte = 0;

    for(j=0; j<lByte; j++)
    {
        SDA_SetInput();
        for(i=0; i<8; i++)
        {
            SCL_SetVal();       _Delay_us(2);
            I2C_Byte <<= 1;
            if(SDA_GetVal())    I2C_Byte |= 1;
            SCL_ClrVal();       _Delay_us(2);
        }
        SDA_SetOutput();
        if((ack == 0)&&(j == lByte-1))
        {   SDA_SetVal();  }
        else
        {   SDA_ClrVal();}
        _Delay_us(2);
        SCL_SetVal();  _Delay_us(2);
        SCL_ClrVal();  _Delay_us(2);
    }
}
```

4.5 延时函数，需要根据不同单片机进行配置

```
    void _Delay_us(int32_t dly)
    {
        int32_t i;
        if(dly > 0)
        {       i = dly;        dly <<= 2;      dly += i;   }
```

```
        do
        {       dly--;      }
        while(dly > 0);
    }
```

5. 程序移植

5.1 I2C 接口配置，下面 IO 口部分需要自行根据不同单片机进行配置，本例程为 IO 口模拟 I2C 进行数据通信。

```
#define   SCL_SetOutput()
#define   SCL_SetVal()
#define   SCL_ClrVal()

#define   SDA_SetOutput()
#define   SDA_SetInput()
#define   SDA_SetVal()
#define   SDA_ClrVal()
#define   SDA_GetVal()
```

5.2 延时函数，需要保证通信协议部分的数据准备延迟时间。也可以通过监测 SDA 线的状态进行数据读取操作。

5.3 完整的 LMI_I2C.c, LMI_I2C.h 文件。