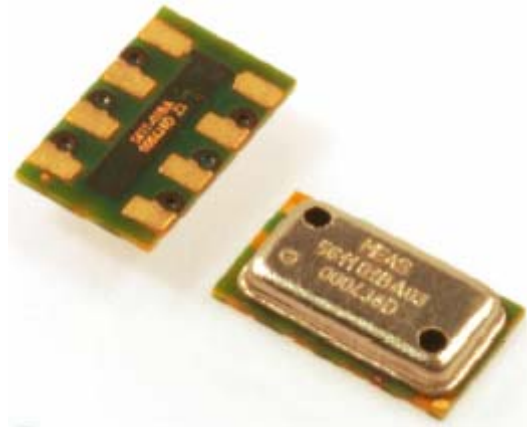


MS5611-01BA03 金属封装气压计组件



性能描述

高度分辨率组件，10cm

转换时间低于1ms

低功耗，工作电流1uA（待机状态<0.15uA）

QFN 封装尺寸：5.0×3.0×1.0 mm³

供电电压1.8V~3.6V

集成数字气压传感器（24位 ADC）

测量/工作范围：10~1200mbar（毫巴=百帕），-40~+85℃

I2C 和 SPI 接口，传输速率可达20MHz

无外接元件（内置振荡器）

长期稳定性好

描述

MS5611-01BA气压传感器是由MEAS（瑞士）推出的一款SPI和I²C总线接口的新一代高分辨率气压传感器，分辨率可达到10cm。该传感器模块包括一个高线性度的压力传感器和一个超低功耗的24位Σ模数转换器（工厂校准系数）。MS5611提供了一个精确的24位数字压力值和温度值以及不同的操作模式，可以提高转换速度并优化电流消耗。高分辨率的温度输出无须额外传感器可实现高度计/温度计功能。可以与几乎任何微控制器连接。通信协议简单，无需在设备内部寄存器编程。MS5611压力传感器只有5.0毫米×3.0毫米×1.0毫米的小尺寸可以集成在移动设备中。这款传感器采用领先的MEMS技术并得益于MEAS（瑞士）十余年的成熟设计以及大批量制造经验，保证产品具有高稳定性以及非常低的压力信号滞后。

内部结构及技术数据

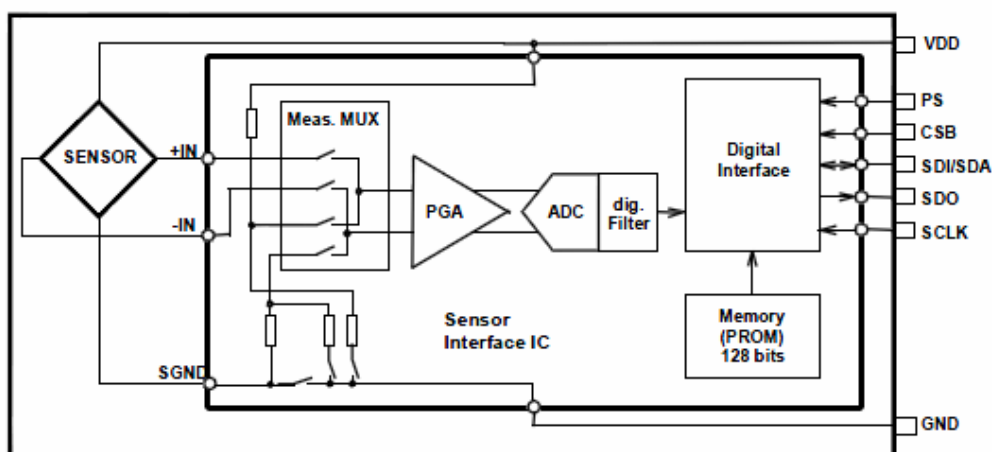
移动高度计/气压计系统

自行车电脑

气压表

医疗警报高度计

室内导航



原理框图

Sensor Performances ($V_{DD} = 3\text{ V}$)				
Pressure	Min	Typ	Max	Unit
Range	10		1200	mbar
ADC	24			bit
Resolution (1)	0.065 / 0.042 / 0.027 / 0.018 / 0.012			mbar
Accuracy 25°C, 750 mbar	-1.5		+1.5	mbar
Error band, -20°C to +85°C 450 to 1100 mbar (2)	-2.5		+2.5	mbar
Response time (1)	0.5 / 1.1 / 2.1 / 4.1 / 8.22			ms
Long term stability		-1		mbar/yr
Temperature	Min	Typ	Max	Unit
Range	-40		+85	°C
Resolution		<0.01		°C
Accuracy	-0.8		+0.8	°C
Notes: (1) Oversampling Ratio: 256 / 512 / 1024 / 2048 / 4096				
(2) With autozero at one pressure point				

技术数据

性能参数

最大工作范围

参数	符号	条件	最小	典型	最大
电源电压	V_{DD}		-0.3V		+4.0V
最大压力值	P_{max}				6bar
最大焊接温度	T_{max}	最长 40 秒			250°C

电气特性

参数	符号	条件	最小	典型	最大
工作电压	V_{DD}		1.8V	3.0V	3.6V
工作温度	T		-40°C	+25°C	+85°C
工作电流 (1 sample per sec.)	I_{DD}	OSR 4096 2048 1024 512		12.5uA 6.3uA 3.2uA 1.7uA	

		256		0.9uA	
VDD 对地电容		VDD to GND	100nF		

模数转换（ADC）

参数	符号	条件	最小	典型	最大
输出字长（bit）				24	
转换时间（ms）	t_c	OSR 4096	7.40	8.22	9.04
		2048	3.72	4.13	4.54
		1024	1.88	2.08	2.28
		512	0.95	1.06	1.17
		256	0.48	0.54	0.60

气压值输出特征

温度值输出特征

数字输入（CSB, I2C, DIN, SCLK）

气压输出（I2C, DOUT）

功能描述

综合描述

MS5611-01BA 是由压阻传感器和传感器接口组成的的集成电路，主要功能是把测得未得补偿模拟气压值经 ADC 转换成 24 位的数字值输出，同时也可以输出一个 24 位的数字温度值。

出厂校验

每个模块都在两种温度和两种压力下有其单独的出厂校验，6 系数必要补偿为过程变化和温度变化计算和存储到一个内部的 128-bit 存储器（PROM）中，这些值（划分成 6 个数）用软件来读取并要通过程序将 D1 和 D2 中的值转换成标准气压、温度值。

串行接口

MS5611-01BA 有两种类型的串行接口:SPI 和 I2C。通过调节 PS 引脚的电压来选择使用 I2C 或 SPI 通信接口：

Pin PS	Mode	Pins used
High	I ² C	SDA
Low	SPI	SDI, SDO, CSB

SPI 模式

外部微控制器通过输入SCLK(串行时钟)和SDI(串行数据)来传输数据。在SPI模式下时钟极性和相位允许同时模式0和模式3。SDO(串行数据)引脚为传感器的响应输出。CSB(芯片选择)引脚用来控制芯片使能/禁用，所以,其他设备可以共用同一组SPI总线。在命令发送完毕或命令执行结束(例如结束的转换)时CSB引脚将被拉高。在SPI总线空闲模式下模块有较好的噪声性能和在ADC转换时与其他设备链接。

I2C 模式

外部微控制器通过输入SCLK(串行时钟)和SDA(串行数据)来传输数据。传感器的响应在一根双向的I2C总线接口SDA线上。所以这个接口类型只使用2信号线路而不需要片选信号，这可以减少板空间。在I2C模式下补充引脚CSB（芯片选择）代表了LSB的I2C地址。在I2C总线上可以使用两个传感器和两个不同的地址。CSB引脚应当连接到VDD或GND（不能悬空）。

指令

MS5611-01BA03 只有 5 个基本命令：

- 1.复位（Reset）
- 2.读取存储器（128-bit PROM）
- 3.D1 转换
- 4.D2 转换
- 5.读取 ADC 结果（24-bit 气压/温度值）

气压和温度计算（详见原文）

1.开始

计算结果最大值: $P_{MIN} = 10\text{mbar}$, $P_{MAX} = 1200\text{mbar}$, $T_{MIN} = -40^{\circ}\text{C}$, $T_{MAX} = 85^{\circ}\text{C}$,
 $T_{REF} = 20^{\circ}\text{C}$

2.从 PROM 读取出厂校准数据

变量	描述 方程	推荐变量类型	Size	值		例子/典型
			bit	min	Max	
C1	压力灵敏度 SENS _{T1}	uint 16	16	0	65535	40127
C2	压力抵消 OFF _{T1}	uint 16	16	0	65535	36924
C3	温度压力灵敏度系数 TCS	uint 16	16	0	65535	23317
C4	温度系数的压力抵消 TCO	uint 16	16	0	65535	23282
C5	参考温度 T _{REF}	uint 16	16	0	65535	33464
C6	温度系数的温度 TEMPSENS	uint 16	16	0	65535	28312

3.读取数字气压和温度值

D1	数字压力值	uint 32	24	0	16777216	9085466
D2	数字温度值	uint 32	24	0	16777216	8569150

4.计算温度

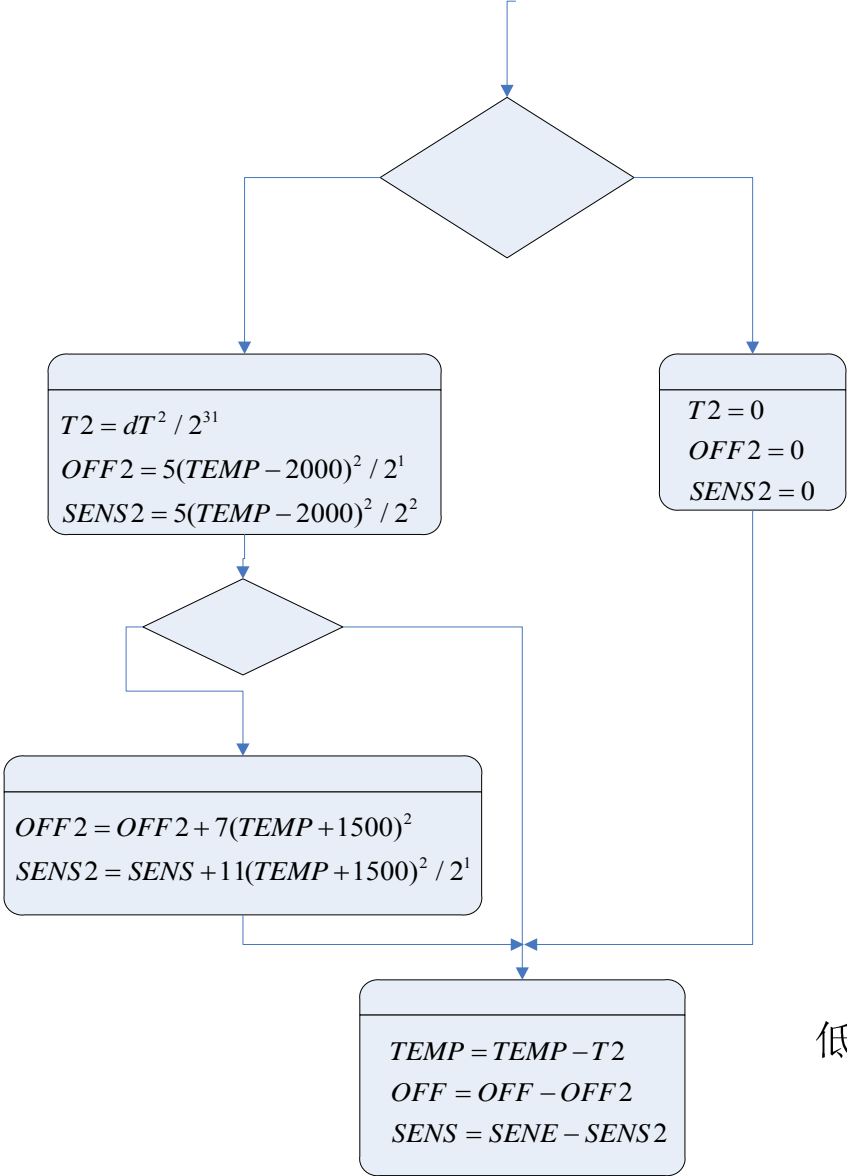
dT	实际和参考温度之间的差异 $dT = D2 - T_{REF} = D2 - C5 * 2^8$	int 32	25	-16776960	16777216	2366
TEMP	实际温度(-40...85° C 0.01° C 的分辨率) $TEMP = 20^{\circ}\text{C} + dT * TEMPSENS$ $= 2000 + dT * C6 / 2^{23}$	int 32	41	-4000	8500	2007=20.07°C

5.计算温度补偿下的气压值

OFF	实际温度抵消 $OFF = OFF_{T1} + TCO * dT$ $= C2 * 2^{16} + (C4 * dT) / 2^7$	int 64	41	-8589672450	12884705280	2420281617
SENS	实际温度灵敏度 $SENS = SENS_{T1} + TCS * dT$ $= c1 * 2^{15} + (C3 * dT) / 2^8$	int 64	41	-4294836225	6442352640	1315097036
P	温度补偿压力(10...1200mbar 与 0.01mbar 分辨率) $P = D1 * SENS - OFF$ $= (D1 * SENS / 2^{21} - OFF) / 2^{15}$	int 32	58	1000		100009=1000.09 mbar

6.得到气压和温度值。

二阶温度补偿



是

低温

对压力和温度达到最佳精度的流程图

SPI 接口

SPI 命令

下面的表格描述中每个命令的大小是1字节(8位)。执行ADC read指令后将会返回一个24-bit的结果，执行PROM read指令后返回16-bit的结果。存储器（PROM）的地址在PROM read命令中的a2，a1和a0位。

是

否

TEMP<-15C

非常低的温度

	Command byte								hex value
Bit number	0	1	2	3	4	5	6	7	
Bit name	PR M	COV	-	Typ	Ad2/ Os2	Ad1/ Os1	Ad0/ Os0	Stop	
Command									
Reset	0	0	0	1	1	1	1	0	0x1E
Convert D1 (OSR=256)	0	1	0	0	0	0	0	0	0x40
Convert D1 (OSR=512)	0	1	0	0	0	0	1	0	0x42
Convert D1 (OSR=1024)	0	1	0	0	0	1	0	0	0x44
Convert D1 (OSR=2048)	0	1	0	0	0	1	1	0	0x46
Convert D1 (OSR=4096)	0	1	0	0	1	0	0	0	0x48
Convert D2 (OSR=256)	0	1	0	1	0	0	0	0	0x50
Convert D2 (OSR=512)	0	1	0	1	0	0	1	0	0x52
Convert D2 (OSR=1024)	0	1	0	1	0	1	0	0	0x54
Convert D2 (OSR=2048)	0	1	0	1	0	1	1	0	0x56
Convert D2 (OSR=4096)	0	1	0	1	1	0	0	0	0x58
ADC Read	0	0	0	0	0	0	0	0	0x00
PROM Read	1	0	1	0	Ad2	Ad1	Ad0	0	0xA0 to 0xAE

命令结构

SPI 复位时序

转换时序

存储器读取时序图参见原文。

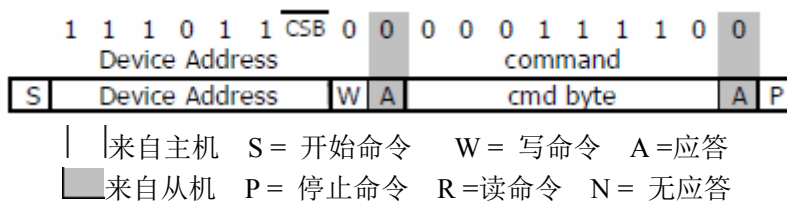
I2C 接口

I2C 命令

每个 I2C 通信消息都有开始和停止状态。MS5611-01BA 的 I2C 地址为 111011Cx，其中 C 为 CSB 引脚的补码值（取反）。因为传感器内并没有微控制器，所有 I2C 的命令和 SPI 是相同的。

I2C 复位时序

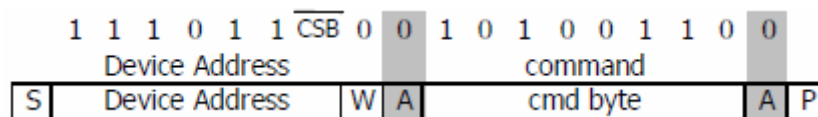
复位指令可以在任何时间发送。如果没有成功的上电复位，这可能是被屏蔽的 SDA 模块在应答状态。MS5611-01BA 唯一的复位方式是发送几个 SCLKs 后跟一个复位指令或上电复位。



I2C 复位指令

存储器读取时序

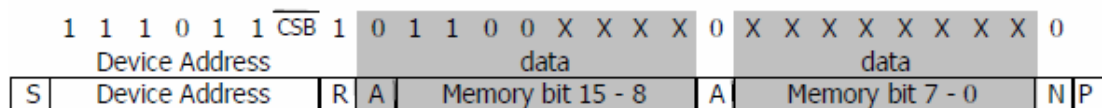
PROM 读指令由两部分构成，第一部分使系统处于 PROM 读模式，第二部分从系统中读取数据。



□ 来自主机 S = 开始命令 W = 写命令 A = 应答

■ 来自从机 P = 停止命令 R = 读命令 N = 无应答

I2C 读存储器指令，地址=011（系数：3）



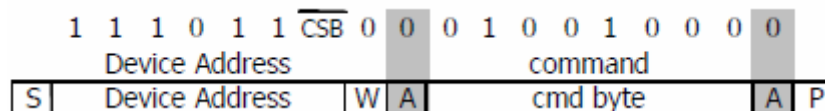
□ 来自主机 S = 开始命令 W = 写命令 A = 应答

■ 来自从机 P = 停止命令 R = 读命令 N = 无应答

I2C 从芯片中应答

转换时序

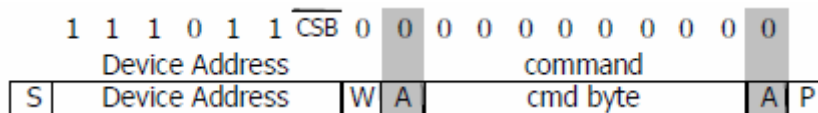
通过向MS5611-01BA发送指令可以进入转换模式。当命令写入到系统中，系统处于忙碌状态，直到转换完成。当转换完成后可以发送一个读指令，此时MS5611-01BA发回一个应答，24个SCLK时钟将所有bit位传送出来。每隔8bit就会等待一个应答信号



□ 来自主机 S = 开始命令 W = 写命令 A = 应答

■ 来自从机 P = 停止命令 R = 读命令 N = 无应答

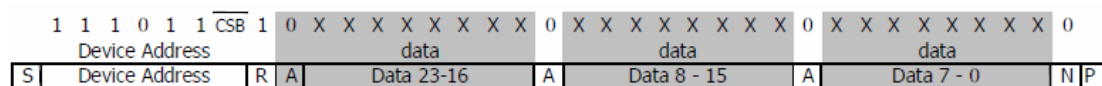
I2C 启动压力转换指令(OSR=4096, typ=D1)



□ 来自主机 S = 开始命令 W = 写命令 A = 应答

■ 来自从机 P = 停止命令 R = 读命令 N = 无应答

ADC 读时序



□ 来自主机 S = 开始命令 W = 写命令 A = 应答

■ 来自从机 P = 停止命令 R = 读命令 N = 无应答

I2C 从 MS5611-01BA 读取数据

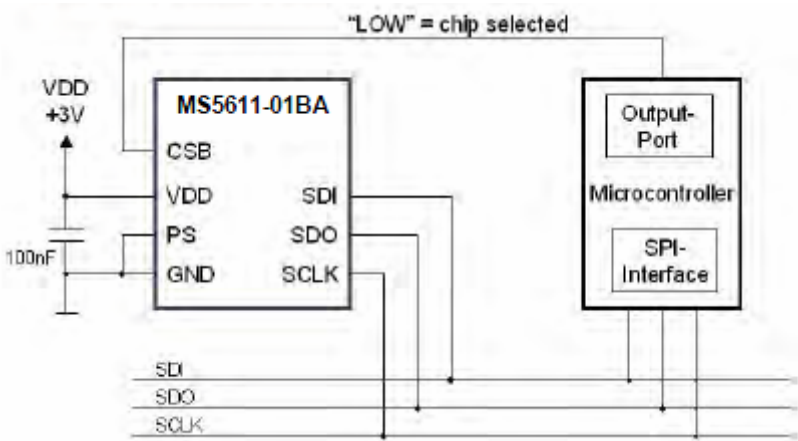
循环冗余检查（CRC）

MS5611-01BA包含128-Bit的PROM存储器。存储器中有一个4bit的CRC数据检测位。下面详细描述了CRC-4代码的使用。

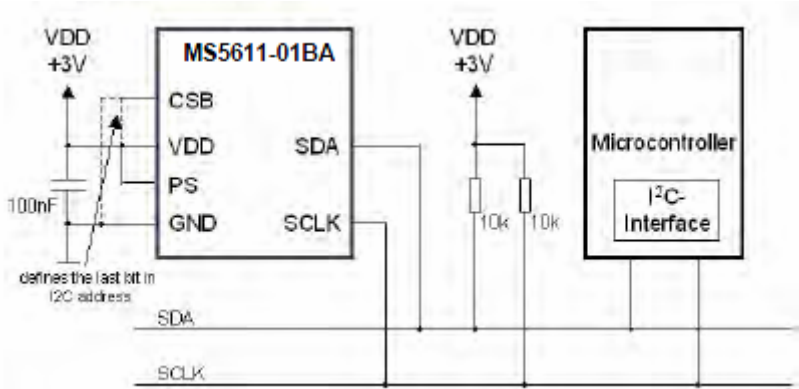
A d d	D B 1 5	D B 1 4	D B 1 3	D B 1 2	D B 1 1	D B 1 0	D B 9	D B 8	D B 7	D B 6	D B 5	D B 4	D B 3	D B 2	D B 1	D B 0
0	16 bit reserved for manufacturer															
1	Coefficient 1 (16 bit unsigned)															
2	Coefficient 2 (16 bit unsigned)															
3	Coefficient 3 (16 bit unsigned)															
4	Coefficient 4 (16 bit unsigned)															
5	Coefficient 5 (16 bit unsigned)															
6	Coefficient 6 (16 bit unsigned)															
7																CRC

典型应用电路

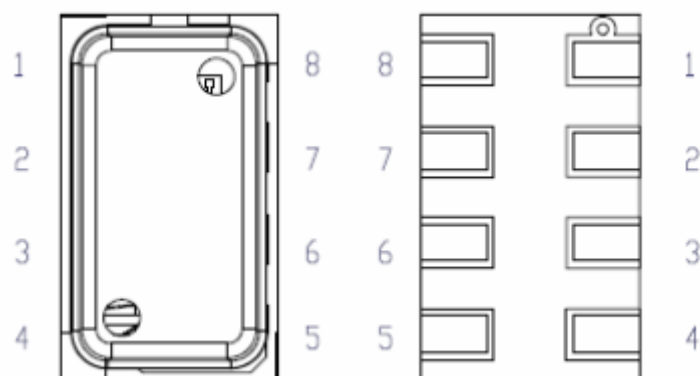
SPI 协议



I2C 协议



引脚定义



引脚	名称	类型	描述
1	VDD	P	电源电压
2	PS	I	通讯协议选择 PS high (VDD) → I2C PS low (GND) → SPI
3	GND	G	接地
4	CSB	I	片选 (低电平有效), 内部连接
5			
6	SDO	O	串口数据输出
7	SDI/SDA	I/O	串口数据输入/I2C 数据
8	SCLK	I	串口时钟

***注：原文附录有一些典型测量值的特征曲线可参考。其中的一些关键点：**ADC** 的测量值 **D1** 和 **D2** 是分别用来测气压和温度的，且基本成线性关系；在常温（20℃）或大于常温时，测量误差很小，而温度低于 20℃时，气压和温度测量误差会明显增大；电源电压为 3V 时，气压和温度测量误差很小，而其它供电电压下误差会大幅度增加。

51 单片机测试代码

```

/*****MS5611 模块*****/
/**用途：MS5611 模块 IIC 测试程序****/
/*****串口波特率设置为 9600*****/
#include <REG52.H>
#include <math.h> //Keil library
#include <stdlib.h> //Keil library
#include <stdio.h> //Keil library
#include <INTRINS.H> //Keil library

#define uchar unsigned char
#define uint unsigned int
#define ulong unsigned long

```

```

sbit SCL=P0^1;      //IIC 时钟引脚定义
sbit SDA=P0^2;      //IIC 数据引脚定义

#define MS561101BA_SlaveAddress 0xee //定义器件在 IIC 总线中的从地址

#define MS561101BA_D1      0x40
#define MS561101BA_D2      0x50
#define MS561101BA_RST     0x1E

// #define MS561101BA_D1_OSR_256    0x40
// #define MS561101BA_D1_OSR_512    0x42
// #define MS561101BA_D1_OSR_1024   0x44
// #define MS561101BA_D1_OSR_2048   0x46
#define MS561101BA_D1_OSR_4096    0x48

// #define MS561101BA_D2_OSR_256    0x50
// #define MS561101BA_D2_OSR_512    0x52
// #define MS561101BA_D2_OSR_1024   0x54
// #define MS561101BA_D2_OSR_2048   0x56
#define MS561101BA_D2_OSR_4096    0x58

#define MS561101BA_ADC_RD      0x00
#define MS561101BA_PROM_RD    0xA0
#define MS561101BA_PROM_CRC   0xAE

////////////////////////////////////

unsigned int Cal_C[7];          //用于存放 PROM 中的 8 组数据
unsigned long D1_Pres,D2_Temp; // 存放压力和温度
float dT,TEMP;
double OFF_,SENS;
float Pressure;                //大气压
float TEMP2,Aux,OFF2,SENS2; //温度校验值

ulong ex_Pressure;             //串口读数转换值
uchar exchange_num[8];

//-----子函数声明-----
void delay(unsigned int k);
void Delay5us();
void Delay5ms();
void I2C_Start();
void I2C_Stop();
void I2C_SendACK(bit ack);

```

```

bit   I2C_RecvACK();
void  I2C_SendByte(uchar dat);
uchar I2C_RecvByte();

void  MS561101BA_RESET();
void  MS561101BA_PROM_READ();
ulong MS561101BA_DO_CONVERSION(uchar command);
void  MS561101BA_getTemperature(uchar OSR_Temp);
void  MS561101BA_getPressure(uchar OSR_Pres);
void  MS561101BA_Init();

void  init_uart();
void  SeriPushSend(uchar send_data);
void  Exchange_Number();
//-----

/*****

/*****
//延时
/*****

void delay(unsigned int k)
{
    unsigned int i,j;
    for(i=0;i<k;i++)
    {
        for(j=0;j<121;j++);
    }
}
/*****

延时 5 微秒(STC90C52RC@12M)
不同的工作环境,需要调整此函数, 注意时钟过快时需要修改
当改用 1T 的 MCU 时,请调整此延时函数
*****/

void Delay5us()
{
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
    _nop_();_nop_();_nop_();_nop_();
}

/*****

延时 5 毫秒(STC90C52RC@12M)

```

不同的工作环境,需要调整此函数
当改用 1T 的 MCU 时,请调整此延时函数

```
*****/

void Delay5ms()
{
    uint n = 560;

    while (n--);
}

//*****
//I2C 起始信号
//*****
void I2C_Start()
{
    SDA = 1;           //拉高数据线
    SCL = 1;           //拉高时钟线
    Delay5us();         //延时
    SDA = 0;           //产生下降沿
    Delay5us();         //延时
    SCL = 0;           //拉低时钟线
}
//*****
//I2C 停止信号
//*****
void I2C_Stop()
{
    SDA = 0;           //拉低数据线
    SCL = 1;           //拉高时钟线
    Delay5us();         //延时
    SDA = 1;           //产生上升沿
    Delay5us();         //延时
}
//*****
//I2C 发送应答信号
//入口参数:ack (0:ACK 1:NAK)
//*****
void I2C_SendACK(bit ack)
{
    SDA = ack;         //写应答信号
    SCL = 1;           //拉高时钟线
    Delay5us();         //延时
    SCL = 0;           //拉低时钟线
    Delay5us();         //延时
}
```

```

}
//*****
//I2C 接收应答信号
//*****
bit I2C_RecvACK()
{
    SCL = 1;                //拉高时钟线
    Delay5us();             //延时
    CY = SDA;               //读应答信号
    SCL = 0;                //拉低时钟线
    Delay5us();             //延时
    return CY;
}
//*****
//向 I2C 总线发送一个字节数据
//*****
void I2C_SendByte(uchar dat)
{
    uchar i;
    for (i=0; i<8; i++)      //8 位计数器
    {
        dat <<= 1;          //移出数据的最高位
        SDA = CY;            //送数据口
        SCL = 1;            //拉高时钟线
        Delay5us();          //延时
        SCL = 0;            //拉低时钟线
        Delay5us();          //延时
    }
    I2C_RecvACK();
}
//*****
//从 I2C 总线接收一个字节数据
//*****
uchar I2C_RecvByte()
{
    uchar i;
    uchar dat = 0;
    SDA = 1;                //使能内部上拉,准备读取数据,
    for (i=0; i<8; i++)      //8 位计数器
    {
        dat <<= 1;
        SCL = 1;            //拉高时钟线
        Delay5us();          //延时
        dat |= SDA;          //读数据
    }
}

```

```

        SCL = 0;                //拉低时钟线
        Delay5us();             //延时
    }
    return dat;
}

//=====
//*****MS561101BA 程序*****
//=====

void MS561101BA_RESET()
{
    I2C_Start();
    I2C_SendByte(MS561101BA_SlaveAddress);
    // I2C_RecvACK();
    I2C_SendByte(MS561101BA_RST);
    // I2C_RecvACK();
    I2C_Stop();
}

void MS561101BA_PROM_READ()
{
    uchar d1,d2,i;
    for(i=0;i<=6;i++)
    {
        I2C_Start();
        I2C_SendByte(MS561101BA_SlaveAddress);
        I2C_SendByte((MS561101BA_PROM_RD+i*2));

        I2C_Start();
        I2C_SendByte(MS561101BA_SlaveAddress+1);
        d1=I2C_RecvByte();
        I2C_SendACK(0);
        d2=I2C_RecvByte();
        I2C_SendACK(1);
        I2C_Stop();

        Delay5ms();

        Cal_C[i]=((uint)d1<<8)|d2;
    }
}

ulong MS561101BA_DO_CONVERSION(uchar command)

```

```

{
    ulong conversion=0;
    ulong conv1,conv2,conv3;
    I2C_Start();
    I2C_SendByte(MS561101BA_SlaveAddress);
    I2C_SendByte(command);
    I2C_Stop();

    delay(100);

    I2C_Start();
    I2C_SendByte(MS561101BA_SlaveAddress);
    I2C_SendByte(0);

    I2C_Start();
    I2C_SendByte(MS561101BA_SlaveAddress+1);
    conv1=I2C_RecvByte();
    I2C_SendACK(0);
    conv2=I2C_RecvByte();
    I2C_SendACK(0);
    conv3=I2C_RecvByte();

    I2C_SendACK(1);
    I2C_Stop();

    conversion=conv1*65535+conv2*256+conv3;

    return conversion;
}

```

```

void MS561101BA_getTemperature(uchar OSR_Temp)

```

```

{

    D2_Temp= MS561101BA_DO_CONVERSION(OSR_Temp);
    delay(10);
    dT=D2_Temp - (((ulong)Cal_C[5])<<8);
    TEMP=2000+dT*((ulong)Cal_C[6])/8388608;

}

```

```

void MS561101BA_getPressure(uchar OSR_Pres)

```

```

{

```

```

D1_Pres= MS561101BA_DO_CONVERSION(OSR_Pres);
delay(10);
OFF_=(ulong)Cal_C[2]*65536+((ulong)Cal_C[4]*dT)/128;
SENS=(ulong)Cal_C[1]*32768+((ulong)Cal_C[3]*dT)/256;

if(TEMP<2000)
{
    // second order temperature compensation when under 20 degrees C
    T2 = (dT*dT) / 0x80000000;
    Aux = TEMP*TEMP;
    OFF2 = 2.5*Aux;
    SENS2 = 1.25*Aux;
    TEMP = TEMP - TEMP2;
    OFF_ = OFF_ - OFF2;
    SENS = SENS - SENS2;
}

Pressure=(D1_Pres*SENS/2097152-OFF_)/32768;
}

void MS561101BA_Init()
{
    MS561101BA_RESET();
    delay(1000);
    MS561101BA_PROM_READ();
    delay(1000);
}

/*****
*****串口程序*****
*****/

void init_uart()
{
    TMOD=0x21;
    TH1=0xfd;
    TL1=0xfd;

    SCON=0x50;
    PS=1;        //串口中断设为高优先级
    TR0=1;       //启动定时器
    TR1=1;
    ET0=1;       //打开定时器 0 中断
    ES=1;
    EA=1;

```



```

}

//*****
void SeriPushSend(uchar send_data)
{
    if(send_data==0)send_data=0x20;
    SBUF=send_data;
    while(!TI);TI=0;
}
//*****数据转换*****
void Exchange_Number()
{
    ex_Pressure=(long)(Pressure);

    if(ex_Pressure<0)
    {
        ex_Pressure=-ex_Pressure;
        exchange_num[0]='-';
    }
    else exchange_num[0]=' ';

    exchange_num[1]=ex_Pressure/100000;//+0x30;
    ex_Pressure=ex_Pressure%100000;

    exchange_num[2]=ex_Pressure/10000+0x30;
    ex_Pressure=ex_Pressure%10000;

    exchange_num[3]=ex_Pressure/1000+0x30;
    ex_Pressure=ex_Pressure%1000;

    exchange_num[4]=ex_Pressure/100+0x30;
    ex_Pressure=ex_Pressure%100;

    exchange_num[5]='.';

    exchange_num[6]=ex_Pressure/10+0x30;
    ex_Pressure=ex_Pressure%10;

    exchange_num[7]=ex_Pressure+0x30;

}
//*****
//*****主程序*****

```

```

//*****
void main()
{
    uchar i;
    delay(50);                //上电延时
    init_uart();              //串口初始化，波特率 9600
    MS561101BA_Init();        //MS561101BA 初始化
    while(1)                  //循环
    {
        MS561101BA_getTemperature(MS561101BA_D2_OSR_4096);
        MS561101BA_getPressure(MS561101BA_D1_OSR_4096);

        Exchange_Number();
        for(i=0;i<=7;i++)
        {
            SeriPushSend(exchange_num[i]);
        }
        SeriPushSend('m');
        SeriPushSend('b');
        SeriPushSend('a');
        SeriPushSend('r');
        SeriPushSend(0x0d);
        SeriPushSend(0x0a);//换行，回车
    }
}

```