

7 系列及环保系列传感器模块应用手册

--7 SMART Module 应用手册

产品概述

7 SMART Sensor Module 传感器模块由电化学传感器和数据采集处理板组成。传感器为标准 7 系结构（盛密科技电化学传感器），数据采集处理板对传感器信号进行放大、采样、滤波，经 MCU 数据处理后实现气体的检测。通过配置不同的传感器，模块可对环境中存在的各类有毒有害及可燃性气体进行实时浓度检测，方便用户在不同场合下以简洁的方式快速组成系统，适用于室内、室外的空气质量检测，以及工业领域等的气体检测。

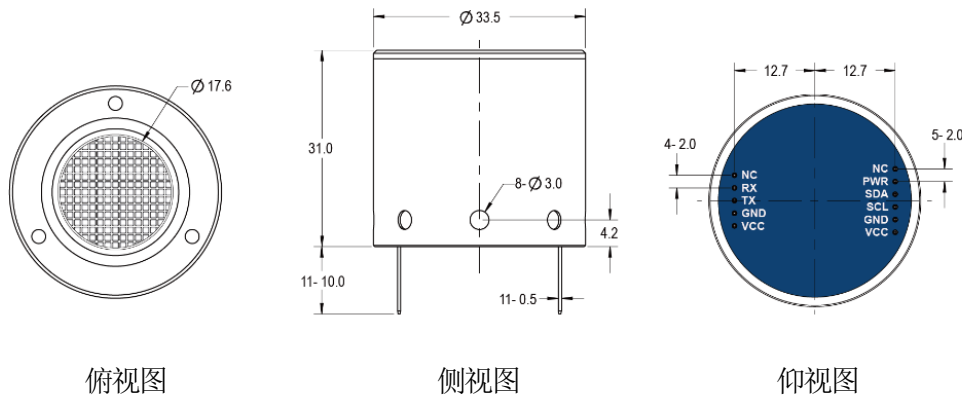
产品特点

- 统一的电气接口、机械尺寸和通讯协议；
- 可以根据需求选择不同的传感器，灵活应用于不同的场合；
- 模块内置温度、湿度传感器，并置入温度补偿算法，以减小环境温度变化对测量结果的影响；
- 多种输出接口，包括 USART、I²C；
- 配置金属外壳（可选），具有保护内部电路防尘防水功能；
- I²C 地址可编程，方便用户根据需求自定义。

技术参数

传感器配置	7 系列及环保系列传感器	安装方式	11 脚插针
检测原理	电化学	工作温度	-20°C ~ +55°C
测量范围	由所选传感器而定，详见数据手册	工作湿度	0% ~ 90%RH (无冷凝)
分辨率	由所选传感器而定，详见数据手册	工作压力	1 ± 0.1 标准大气压
测量误差	由所选传感器而定，详见数据手册	外壳材质	铝合金
工作电压	(3.2 ~ 5.5) VDC	外形尺寸	Φ 33.5 x 31 mm
工作电流	≤ 5 mA @ 5 V	重量	30 克
信号输出方式	USART (3.3V TTL 电平) I ² C (3.3V TTL 电平)	工作寿命	传感器寿命见数据手册 数据板寿命 5 年（无腐蚀环境下）

产品尺寸

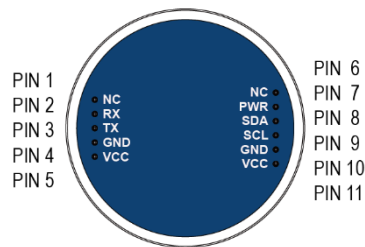


俯视图

侧视图

仰视图

引脚定义图



引脚定义

1	NC	预留引脚（悬空）
2	RX	串口输入 RX
3	TX	串口输出 TX
4	GND	电源地
5	VCC	电源（3.2~5.5 V）
6	NC	预留引脚（悬空）
7	PWR	模块电源使能（低电平关闭，高电平开启，内置上拉电阻）默认上拉高电平
8	SDA	I ² C 信号 SDA（内置 10kΩ 上拉电阻）默认上拉高电平
9	SCL	I ² C 信号 SCL（内置 10kΩ 上拉电阻）默认上拉高电平
10	GND	电源地
11	VCC	电源（3.2~5.5 V）

注：两个 VCC 信号内部连通

USART 通信协议

1. 串行通信参数

起始位：1； 数据位：8； 停止位：1； 校验位：无； 波特率：115200bps；

无特殊说明时，应答回复时间小于 100ms（特殊情况请参考具体指令说明），当前命令回复前无法响应其他指令

2. 帧格式：（每一通信帧的格式如下）

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
H	ID	F	A	N	D	CRC16

H: 数据头，1Byte 固定为 0x3A

ID: 设备码，1Byte 默认为 0x10，可由用户自定义

F: 功能码，1Byte，例如（0x03）

A: 起始地址，2Bytes，例如 0x0001

N: 数据长度，1Byte，以 2 个字节为单位，例如（0x02：4 字节）

D: 数据，N*2Bytes，高位在前，例如（MSB LSB）定义为有符号短整型（signed short）

CRC16: 数据校验，2Bytes，使用 MODBUS_CRC16 校验算法（算法详见附录 1）

3. 指令说明

3.1 读取传感器类型

上位机发送请求

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x01	0x0000	0x01	0x0000	0x82B0

例：**3A 10 01 00 00 01 00 00 82 B0**

模块接收正确数据应答

首字节	设备码	功能码	数据	校验位
0x3A	0x10	0x01	D (1byte 数据)	CRC16

传感器类型代码（十进制）：

0,1: 无定义 2: CO 3: O2 4: H2 5: CH4 6: 无 7: CO2 8: O3 9: H2S 10: SO2

11: NH3 12: CL2 13: ETO 14: HCL 15: PH3 16: 无 17: HCN 18: 无 19: HF 20: 无
21: NO 22: NO2 23: NOX 24: CLO2 25: 无 26: 无 27: 无 28: 无 29: 无 30: 无
31: THT 32: C2H2 33: C2H4 34: CH2O 35: 无 36: 无 37: 无 38: 无 39: CH3SH 40: C2H3CL
例: **3A 10 01 0F 4C AD** (十六进制 0F = 十进制 15, 即得到该传感器为 PH3 传感器)

3.2 读取传感器数据 (单位为 $\mu\text{g}/\text{m}^3$)

上位机发送请求

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x03	0x0000	0x02	0x0000	0x7352

例: **3A 10 03 00 00 02 00 00 73 52**

模块接收正确数据应答

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x03	0x0000	0x02	D	CRC16

D: 接收到的数据, 4Bytes 高位在前

例: **3A 10 03 00 00 02 00 00 00 5E 25 35** (D=0x0000005E=94, 得到传感器值为 $94\mu\text{g}/\text{m}^3$)

3.3 读取传感器数据 (单位为 ppb)

上位机发送请求

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x03	0x0002	0x02	0x0000	0x72EA

例: **3A 10 03 00 02 02 00 00 72 EA**

模块接收正确数据应答

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x03	0x0002	0x02	D	CRC16

D: 接收到的数据, 4Bytes 高位在前

例: **3A 10 03 00 02 02 00 00 00 4C A4 DA** (D=0x0000004C=76, 得到传感器值为 76ppb)

备注: 读取数据显示精度为 1ppb, 具体测试精度根据传感器不同而不同

3.4 读取温度传感器数据（单位为°C）

上位机发送请求

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x03	0x0004	0x01	0x0000	0x8262

例：**3A 10 03 00 04 01 00 00 82 62**

模块接收正确数据应答

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x03	0x0004	0x01	D	CRC16

D: 接收到的数据，2Bytes 高位在前，除以 100 后得到温度值。

例：**3A 10 03 00 04 01 0A 3D 45 13** (D=0x0A3D=2621, 除以 100 得到温度值为 26.21°C)

3.5 读取湿度传感器数据（单位为%RH）

上位机发送请求

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x03	0x0005	0x01	0x0000	0x839E

例：**3A 10 03 00 05 01 00 00 83 9E**

模块接收正确数据应答

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x03	0x0005	0x01	D	CRC16

D: 接收到的数据，2Bytes 高位在前，除以 10000 后得到百分比湿度。

例：**3A 10 03 00 05 01 14 89 4D 38** (D=0x1489=5257, 除以 10000 后得到湿度为 52.57%RH)

3.6 读取多个参数（地址 0000~0005）

上位机发送请求

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x03	0x0000	0x06	0x0000	0x3293

例：**3A 10 03 00 00 06 00 00 32 93**

模块接收正确数据应答

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x03	0x0000	0x06	D	CRC16

D: 接收到的数据, 12Bytes

依次为(高位在前): 传感器读数 $\mu\text{g}/\text{m}^3$ 4Bytes; 传感器读数 ppb 4Bytes; 温度值 2Bytes; 湿度值 2Bytes。

例: **3A 10 03 00 00 06 00 00 00 8F 00 00 00 50 0A 70 16 11 35 96**

其中: 传感器值 ($\mu\text{g}/\text{m}^3$): 00 00 00 8F 传感器值 (ppb): 00 00 00 50 温度: 0A 70 湿度: 16 11

3.7 校验错误应答

首字节	设备码	功能码	数据	校验位
0x3A	0x10	0x08	0x00	CRC16

例: **3A 10 08 00 0A F9**

3.8 零点标定

上位机发送请求

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x07	0x0000	0x01	0x0000	0x82D6

例: **3A 10 07 00 00 01 00 00 82 D6**

模块接收正确数据应答

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x07	0x0000	0x01	D	CRC16

D: 接收到的数据 2Bytes

例: **3A 10 07 00 00 01 04 7A 01 F5**

备注: 请将模块放置于零气环境中稳定至少 5 分钟后发送零点标定命令

3.9 灵敏度标定

上位机发送请求

首字节	设备码	功能码	起始地址	数据长度	数据	校验位
0x3A	0x10	0x09	0x0000	0x01	D	0x82D6

D: 所用标定气体的浓度 2Bytes, 高位在前, 单位 PPM。

例: **3A 10 09 00 00 01 00 0A 03 FF** 其中 D 为 00 0A 即: 使用 10PPM 浓度气体进行标定

模块接收正确数据应答

首字节	设备码	功能码	状态	校验位
0x3A	0x10	0x09	0x00 (标定成功) 0x01 (标定中) 0x02 (标定失败)	CRC16

例: **3A 10 09 01 CAA9** (标定中)

备注: 环保模块标定过程约 300 秒, 工业模块标定过程约 60 秒, 请将模块通入标准气体等待有响应后再发送标定指令。

I²C 通信协议

1. I²C 接口

参数	定义	状态	最小	最大	单位
f _{ck}	I ² C 时钟频率	从模式		100	kHz
t _{su}	数据输入建立时间	从模式	6.5		ns
t _h	数据输入保持时间	从模式	15.5		ns

2. 从设备地址

从设备地址可以通过软件工具自定义

默认设置如下:

CO:	0	0	0	0	0	0	1	R/W
O3:	0	0	0	0	1	0	0	R/W
SO2:	0	0	0	0	1	0	1	R/W
NO2:	0	0	0	1	0	1	1	R/W

R/W 位同时也作为普通的数据位, 当要进行读/写时, 用 1 或者 0 与原数据位进行 '或' 运算 (读为 1, 写为 0) 详细见下图

I²C 地址定义:

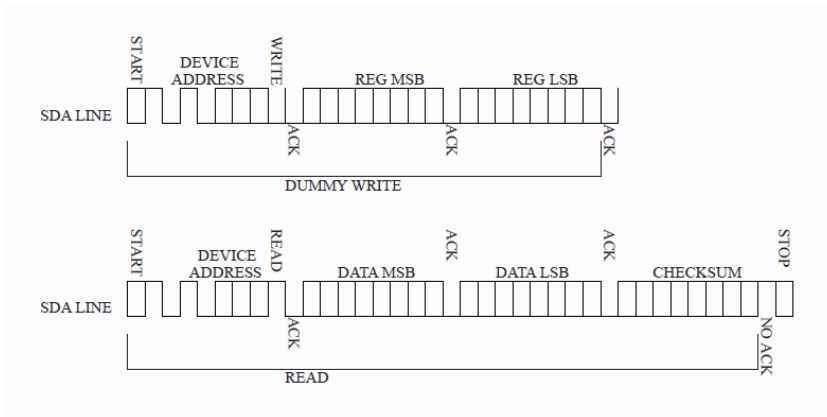
CO: 0x02,	CLO2: 0x18,	PH3: 0x14,
H2: 0x04,	C2H2: 0x20,	HCN: 0x1A,
O3: 0x08,	CH2O: 0x22,	HF: 0x1C,
SO2: 0x0A,	C2H3CL: 0x28,	NO: 0x1E,
CL2: 0x0C,	H2S: 0x06,	THT: 0x24,
HCL: 0x0E,	NH3: 0x10,	C2H4: 0x26,
NO2: 0x16,	ETO: 0x12,	CH3SH: 0x2A

注意事项:

在 2024 年 2 月 5 日之前收到模块的用户,请参照上述的 I²C 地址,使用 AT 191008 - 7 SMART Module Test 软件进行 I²C 地址修改后再进行 I²C 通讯。

在 2024 年 2 月 5 日之后收到模块的用户,请直接参照上述地址进行 I²C 通讯。

3. I²C 通信协议



CHECKSUM 为累加和取反校验, 算法详见附录 2

4. 数据解析

REG 参数如下表, 为模块中数据地址

REG	MSB	LSB
传感器值 ($\mu\text{g}/\text{m}^3$)	0x00	0x00
传感器值 (ppb)	0x00	0x02
温度 ($^{\circ}\text{C}$)	0x00	0x04

湿度 (%RH)	0x00	0x05
----------	------	------

DATA 举例:

DATA	MSB	LSB	CHECKSUM	实际值	备注
传感器值 ($\mu\text{g}/\text{m}^3$)	0x0000	0x0005	0xFA	$5\mu\text{g}/\text{m}^3$	与 USART 数据 转换方法相同
传感器值 (ppb)	0x0000	0x0005	0xFA	5ppb	
温度 ($^{\circ}\text{C}$)	0x0A	0x3D	0xB8	26.21 $^{\circ}\text{C}$	
湿度 (%RH)	0x14	0x89	0x62	52.57%RH	

备注: 读取传感器值接收 5Bytes 数据, 读取温度和湿度时接收 3Bytes 数据

注意事项

- 1) 本模块不具备电源反接保护及静电防护功能, 用户在使用时请正确连接模块电源, 并做好静电防护措施;
- 2) 请使用稳定的直流电源给模块供电, 电源电压波动应小于 1%。

附录 1: MODBUS CRC16 算法

```

unsigned short modbus_CRC16(unsigned char *ptr, unsigned char len)
{
    unsigned short wcr=0xFFFF; //
    int i=0, j=0;
    for (i=0; i<len; i++)
    {
        wcr^=*ptr++;
        for (j=0; j<8; j++)
        {
            if (wcr&0X0001)
            {
                wcr=wcr>>1^0XA001;
            }
            else
            {
                wcr>>=1;
            }
        }
    }
    return wcr<<8|wcr>>8; //低位在前, 高位在后
}

```

CRC-16/MODBUS 算法:

在 CRC 计算时只用 8 个数据位, 起始位及停止位, 如有奇偶校验位也包括奇偶校验位, 都不参与 CRC 计算。

CRC 计算方法是:

- 1、 加载一值为 0XFFFF 的 16 位寄存器, 此寄存器为 CRC 寄存器。
- 2、 把第一个 8 位二进制数据 (即通讯信息帧的第一个字节) 与 16 位的 CRC 寄存器的相异或, 异或的结果仍存放于该 CRC 寄存器中。
- 3、 把 CRC 寄存器的内容右移一位, 用 0 填补最高位, 并检测移出位是 0 还是 1。
- 4、 如果移出位为零, 则重复第三步 (再次右移一位); 如果移出位为 1, CRC 寄存器与 0XA001 进行异或。
- 5、 重复步骤 3 和 4, 直到右移 8 次, 这样整个 8 位数据全部进行了处理。
- 6、 重复步骤 2 和 5, 进行通讯信息帧下一个字节的处理。
- 7、 将该通讯信息帧所有字节按上述步骤计算完成后, 得到的 16 位 CRC 寄存器的高、低字节进行交换
- 8、 最后得到的 CRC 寄存器内容即为: CRC 校验码。
- 9、 举例: 一条命令 3A 10 09 00 00 01 00 32 通过上述程序, 得到 wrcrc 返回值为 02 2D, 作为校验码, 这样我们就得到了通气标定命令: 3A 10 09 00 00 01 00 32 02 2D。

附录 2: CHECKSUM 累加和校验

```
unsigned char CheckSum(unsigned char *buf, unsigned char len) //累加和校验值
{
    unit8_t i, ret = 0;
    for (i=0; i<len; i++)
    {
        ret +=*(buf++);
    }
    ret = ~ret;
    return ret;
}
```

累加和算法的实现:

发送方: 对要数据累加, 得到一个数据和, 对和求反, 即得到我们的校验值。然后把要发的数据和这个校验值一起发送给接收方。

举个例子:

发送方: 要发送 0xA8,0x50,我们使用 unsigned char (8 位) 来保存累加和, 即为 0xF8 (0b11111000),取反得到校验和为 0x07 (0b00000111)。然后将这三个数据发送出去。