产 品 规 格 书

产品名称： 数字式通用二氧化碳传感器

产品型号： C8

版本号： V1.0

编制	审核	标准化	批准
蔡晓峰 2019-2-17	张映君 2019-2-17	刘芳 2019-2-17	黄楚兴 2019-2-17

1. 主要特性

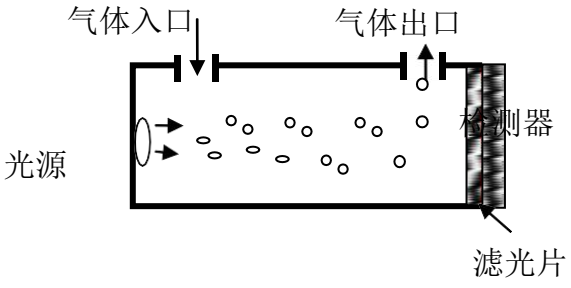
- ◆ 非分散红外吸收原理
- ◆ 内置自动校准算法
- ◆ 高灵敏度、低功耗
- ◆ 提供串口(UART)、PWM等输出方式

2. 概述

C8是一款采用NDIR 气体传感器技术测量室内空气中二氧化碳浓度的迷你型红外二氧化碳传感器模块，其设计小巧、性能优越、易于安装，可广泛适用于空气净化器、新风系统、带净化功能的空调、空气品质检测仪器、农业物联网、汽车以及消费类电子的配套。

3. 工作原理

红外非分散吸收（NDIR）气体传感器的工作原理是根据不同气体分子对于近红外光谱的吸收特性，通过气体浓度与吸收强度关系（朗伯-比尔定律)分析计算并确定气体的浓度。CO2、CO 等由异种原子构成的分子在红外线波长区域具有吸收光谱，其吸收强度遵循郎伯-比尔定律。当对应某一气体特征吸收波长的光波通过被测气体时，其强度将明显减弱，强度衰减程度与该气体浓度有关，两者之间的关系遵守朗伯-比尔定律。NDIR 传感器的基本原理结构如下图所示，



基本数学模型如下：大部分有机和无机多原子分子气体在红外区有特定吸收波长。当红外光通过时，这些气体分子对特定波长的透过光强可由朗伯-比尔定律表示：$I = I_0 e^{-kpl}$，吸收光强 i 可表示为：$i = I_0 - I = I_0 (1 - e^{-kpl})$ 。式中，I_0 为入射光强；I 为透过光强；l 为气体介质厚度，p 为气体浓度，k 为吸收系数。

4. 技术指标

如表 1 所示

表 1 传感器技术指标

参数	指标	单位
量程范围	400~5000	ppm
分辨率	1	ppm
精度	±（50ppm+5% *读数）	
响应时间 T90	<120	秒（s）
数据更新时间	<3（标准 1s）	秒（s）
预热时间	<25s（可操作） <2min（90%精度） <10min（最大精度）	
直流供电电压	Typ:5.0 Min:4.5 Max: 5.5	伏特（V）
工作电流	峰值80,平均30	毫安（mA）
数据接口电平	L <0.8 @3.3 H >2.7@3.3	伏特（V）
工作温度范围	-10~+50	摄氏度（℃）
工作湿度范围	0~85% 无凝结	
储存温度范围	-40~+75	摄氏度（℃）
平均无故障时间	10	年（Y）
最大尺寸*	33×21×11	毫米（mm）
针脚间距	2.54	毫米（mm）

注：最大尺寸不包括针脚长度。

5. 数字接口定义

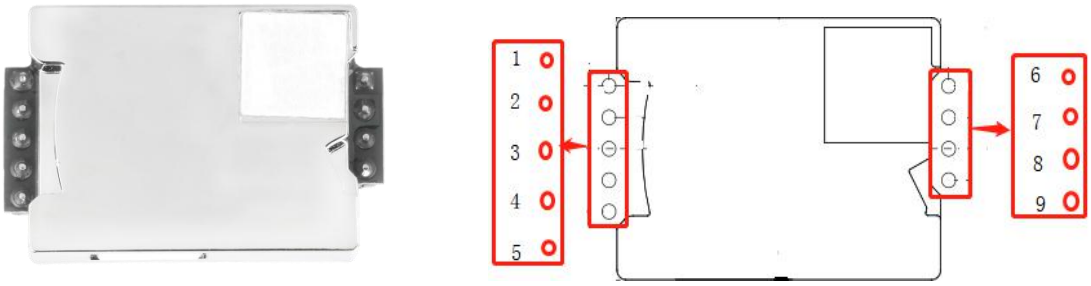
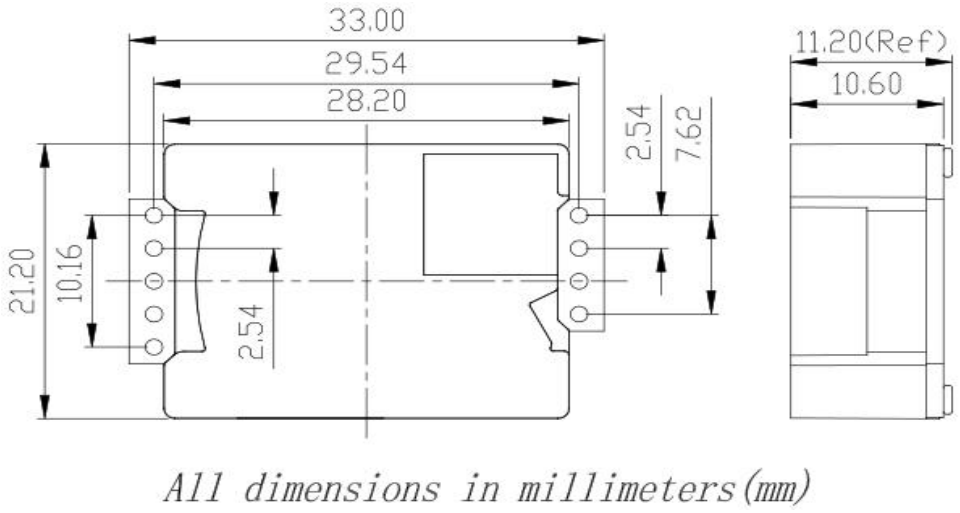


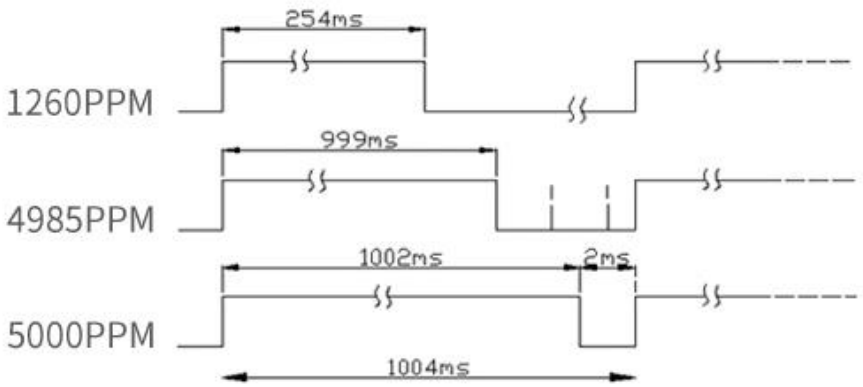
图 1 接口示意图

	序号	名称	描述
CON5	①	NC	预留
	②	RX	串口端（电路板串口接收端）
	③	TX	串口端（电路板串口发送端）
	④	NC	预留
	⑤	NC	预留
CON4	⑥	+5V	电源输入端（+5V 端）
	⑦	GND	电源输入端（接地端）
	⑧	NC	预留
	⑨	PWM	脉宽调制

6. 外形及结构尺寸



7. PWM输出

PWM 输出	
CO_2 浓度输出范围	0～5000ppm
周期	1004ms±5%
中部周期	1000ms±5%
通过PWM 获得当前 CO_2 浓度值的计算公式：$C_{ppm}=(T_H-2ms) * 5$	
Cppm 为通过计算得到的CO2 浓度值单位为 ppm， TH 为一个输出周期中输出为高电平的时间 TL 为一个输出周期中输出为低电平的时间	
	

8. 自校准说明

在产品使用过程中，由于运输、安装、焊接等操作可能会引发 NDIR 二氧化碳传感器的零点漂移和检测精度降低，故传感器通过内置的自校准算法对漂移进行修正。在传感器上电 24h 时，存储记录最低 CO2 浓度检测值，传感器自动执行校准算法，将基准值修正为室外的大气环境 CO2 浓度。

9. 串口通讯协议

串口配置：9600，8N1.

传感器有两种输出方式,主动输出及查询输出,查询一次后,主动输出停止.需重新上电,才可恢复.

上电后传感器主动输出数据：输出频率 1S 一次.

输出格式16BYTE。

数据头： BYTE0 = 0X42；BYTE1=4D

BYTE6 数据高位，BYTE7数据低位，表示CO2浓度。

BYTE15,数据校验和. BYTE15= BYTE0+BYTE1+…….+BYTE13;

例： 42 4D 0C 51 09 A2 07 2B 01 35 05 81 20 08 20 AD;

CO2浓度= BYTE6 X 256 + BYTE7 =07X256 + 2B = 1853;

查询读取：

读取CO2ppm值，返回522ppm

发送：64 69 03 5E 4E

返回：64 69 03 01 0A 02 00 00 00 00 00 00 9B F0

返回 14个BYTE.

BYTE4,BYTE5表示CO2浓度,转化为10进制既为浓度值,BYTE5 为高8位,BYTE4为低8位.

0x0A是16bit整形数的低字节，0x02是高字节，合在一起就是522

BYTE12,BYTE13为CRC检验数据

查询命令发送一次,传感器停止主动输出.每查询一次,输出一次.

CRC计算

```
uint16_t CO2ModbusComm::modbus_calcuCRC(uint8_t *dataarray, uint16_t datalen)
{
    uint8_t uchCRCHi = 0xFF ; /* CRC 的高字节初始化*/
    uint8_t uchCRCLo = 0xFF ; /* CRC 的低字节初始化*/
    uint16_t uIndex ; /* CRC 查询表索引*/
    uint16_t crc;

    const uint8_t auchCRCHi[] = {
        0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
        0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
        0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
        0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
        0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
        0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
        0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
        0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
        0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
        0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
        0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
        0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
        0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
        0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
```

```
        0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
        0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
        0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
        0x40
        };
    const uint8_t auchCRCLo[] = {
        0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
        0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9,
0x09,
        0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F,
0xDD,
        0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
        0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
        0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
        0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
        0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
        0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
        0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
        0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
        0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75,
0xB5,
        0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
        0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
        0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
        0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
        0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,
        0x40
        };

    while (datalen--) /*  完成整个报文缓冲区*/
    {
        uIndex = uchCRCLo ^ *dataarray++;/*  计算CRC */
        uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex];
        uchCRCHi = auchCRCLo[uIndex];
    }
    crc = (uint16_t)uchCRCHi *256;
    crc += (uint16_t)uchCRCLo;
    return crc;
}
```