**Guangzhou Haigu Electronics Technology Co., Ltd.**

**Product Specification**

**Product Name**: Digital Universal Carbon Dioxide Sensor  
**Model**: HC8  
**Version**: V1.0

Prepared by: Huang Wenqin  
Reviewed by: Liu Guohui  
Standardized by: Zeng Zhengqi  
Approved by: Xu Liang

Date: 2019-02-17

**Main Features**

* Non-dispersive infrared (NDIR) absorption principle
* Built-in automatic calibration algorithm
* High sensitivity and low power consumption
* Offers UART and PWM output modes

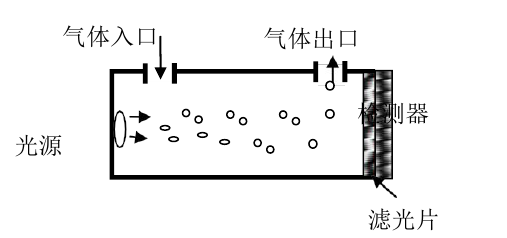


### Overview

The HC8 is a compact infrared CO2 sensor module using NDIR gas sensor technology to measure indoor CO2 concentrations. It is compact, high-performance, easy to install, and widely applicable in air purifiers, ventilation systems, air conditioners with purification functions, air quality monitoring instruments, agricultural IoT, automobiles, and consumer electronics.

**Working Principle**

The working principle of NDIR gas sensors is based on the absorption characteristics of specific gas molecules in the near-infrared spectrum. Using Lambert-Beer’s law, the gas concentration is analyzed by the relationship between absorption intensity and gas concentration. CO2 and other molecules composed of different atoms exhibit absorption spectra in the infrared wavelength region.



The basic structure of the NDIR sensor includes:

* **Gas Inlet and Outlet**: Channels for the target gas
* **Detector**: Measures light intensity
* **Light Source**: Emits infrared light
* **Filter**: Selects specific wavelengths

The mathematical model follows Lambert-Beer's law:

* I=I0e−kplI = I\_0 e^{-kpl}I=I0​e−kpl  
  Where:  
  I0I\_0I0​: Incident light intensity  
  III: Transmitted light intensity  
  kkk: Absorption coefficient  
  ppp: Gas concentration  
  lll: Thickness of the gas medium

**Technical Specifications**

| **Parameter** | **Value** | **Unit** |
| --- | --- | --- |
| Measurement Range | 400–5000 | ppm |
| Resolution | 1 | ppm |
| Accuracy | ±(50ppm + 5% of reading) |  |
| Response Time (T90) | < 120 | seconds |
| Data Update Interval | < 3 | seconds |
| Warm-up Time | < 25 (operable), < 2 min (90% accuracy), < 10 min (max accuracy) |  |
| Supply Voltage (DC) | 4.5–5.5 (Typ: 5.0) | volts |
| Operating Current | 30 (avg), 80 (peak) | mA |
| Operating Temperature | -10 to +50 | °C |
| Storage Temperature | -40 to +75 | °C |
| Operating Humidity | 0–85 (non-condensing) | % |
| MTBF | 10 | years |
| Dimensions (max) | 33 × 21 × 11 | mm |
| Pin Pitch | 2.54 | mm |

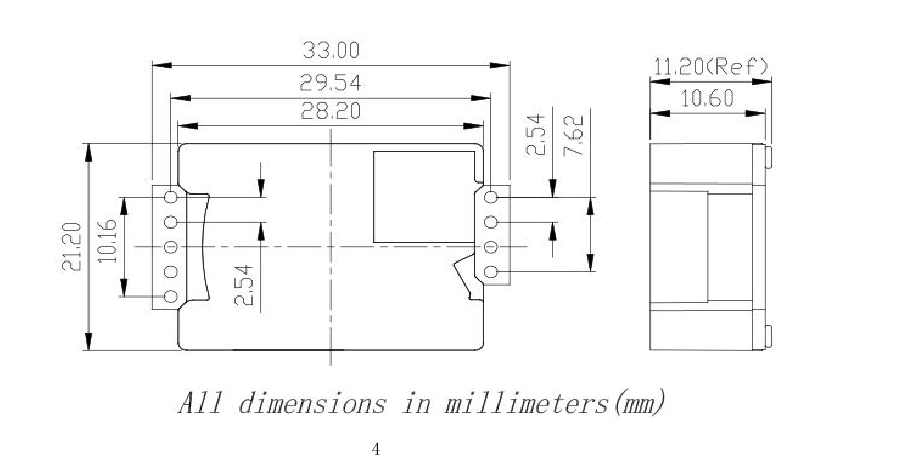
### Digital Interface Definition

### 

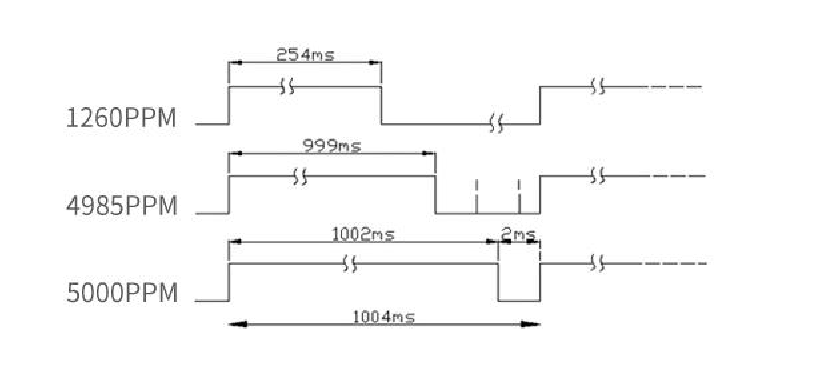
#### Interface Diagram:

| **Pin Number** | **Name** | **Description** |
| --- | --- | --- |
| 1 | NC | Reserved |
| 2 | RX | UART Receive (sensor input) |
| 3 | TX | UART Transmit (sensor output) |
| 4 | NC | Reserved |
| 5 | NC | Reserved |
| 6 | +5V | Power input |
| 7 | GND | Ground |
| 8 | NC | Reserved |
| 9 | PWM | Pulse Width Modulation output |

### Physical Dimensions

**PWM Output**

* **CO2 Measurement Range**: 0–5000 ppm
* **Period**: 1004 ms ± 5%
* **Formula for CO2 concentration**:
* Cppm​=(TH​−2ms)×5  
  Where THT\_HTH​ is the high-level duration in one cycle.



**UART Communication Protocol**

* **Configuration**: 9600 baud rate, 8 data bits, no parity, 1 stop bit.
* Two modes: Active output (1 Hz) and query mode.
* Data format: 16 bytes with checksum validation.
* Data header: BYTE0 = 0X42; BYTE1=4D
* BYTE6 data high bit, BYTE7 data low bit, indicating CO2 concentration.
* BYTE15, data checksum. BYTE15= BYTE0+BYTE1+…….+BYTE13;

Example:

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

CO2 concentration = BYTE6 X 256 + BYTE7 =07x256 + 2B = 1853;

**Query reading:**

Read CO2ppm value, return 522ppm

Send: 64 69 03 5E 4E

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

Return 14 BYTEs.

BYTE4, BYTE5 represent CO2 concentration, converted to decimal as concentration value, BYTE5 =High 8 bits, BYTE4

low 8 bits.

0x0A is the low byte of the 16-bit integer, 0x02 is the high byte, together they are 522

BYTE12, BYTE13 are CRC check data

After sending the query command once, the sensor stops actively outputting. Each query will output once.

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,

7

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;

}