

## Product Data Sheet

NDIR CO2 SENSOR MODULE (4CO2-5000) (PN: 096-0100-000)

### • Description

This infrared sensor functions based on the NDIR principle for monitoring carbon dioxide (CO<sub>2</sub>) presence. It features a highly accurate optical system equipped with a durable light source and a temperature-compensated infrared detector with dual channels. The electronic circuit includes a microprocessor with embedded intelligent software, which handles signal acquisition, processing, and output. Furthermore, it simultaneously performs environmental temperature compensation, rectifies response non-linearity, and offers multiple output options.

### • Performance Characteristics

Output Mode:	UART
	Analog (0.4 ~ 2.0 V)
Number of Channel:	Dual
Dimension:	Industrial standard 4-series

### • Environmental

Storage Temperature:	-20°C ~ 50°C
Working Temperature:	-40°C ~ 70°C
Working Humidity:	0% ~ 95% RH non-condensing

### • Pinout

Pin 1 - VCC	Pin 4 - RX
Pin 2 - GND	Pin 5 - DA
Pin 3 - TX	

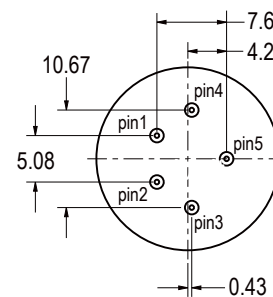
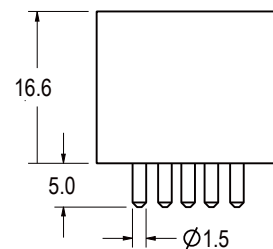
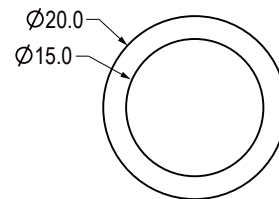
### • Life Time

Expected Operating Life:	6 years in clean air
Warranty:	24 months

### • Technical Data

Detection Range	0 ~ 5,000 (customizable) ppm
Detection Accuracy	±(50 ppm + 5% of reading)
Response Time (T90)	< 30s @ 20°C ambient
Warm-Up Time	Start to work < 30s @ 20°C ambient
	Precision reached < 5min @ 20°C ambient
Working Voltage	3.0 ~ 5.5 VDC
Working Current	I <sub>avg</sub> : 45 / I <sub>peak</sub> : 74 mA
Resolution	1% of measuring range
Zero Repeatability	±500 ppm @ 20°C ambient
Span Repeatability	±500 ppm @ 20°C ambient
Long Term Zero Drift	±500 ppm / month @ 20°C ambient

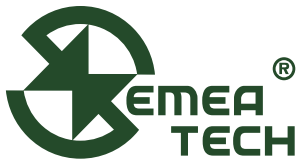
### • Product Dimensions



All dimensions in mm  
All tolerances ±0.20mm unless otherwise stated

### • Note

All the above performance parameters are measured in a standard test environment. Please contact us if you need more details.



### • UART Protocol

Baud rate: 9600bps (settable), 8-bit data, 1-bit stop bit, no check bit.

#### 1. Active upload mode

The sensor actively uploads the concentration value and outputs the data in ASCII format as follows:

32	32	x	x	x	x	x	32	p	p	m	\r	\n
----	----	---	---	---	---	---	----	---	---	---	----	----

Where 32 is the ASCII code of the space, and the output ends with a new line character.

For example : output of 12345 ppm :

		1	2	3	4	5		p	p	m
0x20,	0x20,	0x31,	0x32,	0x33,	0x34,	0x35,	0x20,	0x70,	0x70,	0x6d,

#### 2. Passive upload mode

The sensor will automatically switch to passive mode after receiving any of the passive commands.

##### 2.1 Passive Command Format :

(The format of each communication frame is as follows)

Header	Device Code	Function Code	Starting Address	Data Length	Data	Check Bit
H	ID	F	A	N	D	CRC16

**H:** 1Byte, fixed as 0x3A

**ID:** 1Byte, fixed as 0x10

**F:** 1Byte, e.g. 0x03

**A:** 2Bytes, e.g. 0x0001

**N:** 1Byte, e.g. 0x01

**D:** N\*2Bytes, high byte first, e.g.(MSB LSB)defined as a signed short integer.

**CRC16:** Check data from first byte to data bit, 2Bytes, using MODBUS\_CRC16 checking algorithm.

(see Appendix 1 for details)

##### 2.2 Sensor Type Reading

###### Request from host device

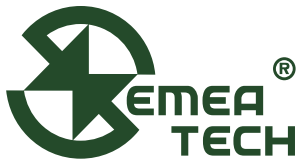
Header	Device Code	Function Code	Starting Address	Check Bit
0x3A	0x10	0x01	0x00	0x0C A9

###### Module receiving the correct data reply

Header	Device Code	Function Code	Data	Check Bit
0x3A	0x10	0x01	D (1byte)	CRC16

D data definition: 0A: CO2 0B: CH4

e.g., 3A 10 01 0A 8C AE Read data D as 0x0A and module as CO2 module.



### 2.3 Gas Concentration Reading

#### Request from host device

Header	Device Code	Function Code	Starting Address	Data Length	Data	Check Bit
0x3A	0x10	0x03	0x0000	0x01	0x00	0xC6 03

#### Module receiving the correct data reply

Header	Device Code	Function Code	Starting Address	Data Length	Data	Check Bit
0x3A	0x10	0x03	0x0000	0x01	D	CRC16

D: Received data, 4Bytes, high byte first

e.g., 3A 10 03 00 00 01 00 00 03 09 20 3B

Concentration(ppm): 00 00 03 09 i.e., 777ppm

### 2.4 Calibration Instruction

#### Request from host device

Header	Device Code	Function Code	Starting Address	Data	Check Bit
0x3A	0x10	0x07	0x00	D	CRC16

e.g., 2Bytes, high byte first

send 3A 10 07 00 01 90 C4 0A D is 0x01\*256+0x90, for calibration of the 400 ppm concentration

send 3A 10 07 00 00 00 C5 F6 D is 0x00\*256+0x00, for calibration of the 400 ppm concentration

#### Module response (data bit 0x00 indicates successful calibration, 0x01 indicates failed calibration)

Header	Device Code	Function Code	Data	Check Bit
0x3A	0x10	0x07	0x00/0x01	0x0F 09

Note: It is necessary to place the sensor module in a stable environment for at least 10 minutes before calibration.

Calibration is complete when the calibration receives a response.

### 2.5 Factory Default Setting

#### Request from host device

Header	Device Code	Function Code	Data	Check Bit
0x3A	0x10	0x09	0x00	0x0B 69

#### Module receiving the correct data reply

Header	Device Code	Function Code	Data	Check Bit
0x3A	0x10	0x09	0x00	0x0B 69

Note: The Restore Factory Settings command restores the calibrated settings in 2.4 to the manufacturer default settings.

### 2.6 Modification of baud rate

#### Request from host device

Header	Device Code	Function Code	Starting Address	Data	Check Bit
0x3A	0x10	0x05	0x01 1C	D	CRC16

D: 1Byte data, 0x00 i.e., baud rate =9600bps

0x01 i.e., baud rate =19200bps

0x02 i.e., baud rate =38400bps

e.g., 3A 10 05 01 1C 00 9D 4E (baud rate =9600bps)

3A 10 05 01 1C 01 5C 8E (baud rate =19200bps)

3A 10 05 01 1C 02 1C 8F (baud rate =8400bps)

#### Module response (data bit 0x00 indicates successful calibration, 0x01 indicates failed calibration)

Header	Device Code	Function Code	Starting Address	Data	Check Bit
0x3A	0x10	0x05	0x01 1C	D	CRC16

#### Appendix1: MODBUS CRC16 algorithm

```

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; // low byte first, then high byte
}

```