SENSECAP

CO2, Temperature and Humidity Sensor - Datasheet

S-CO2-03

Version: V 2.0

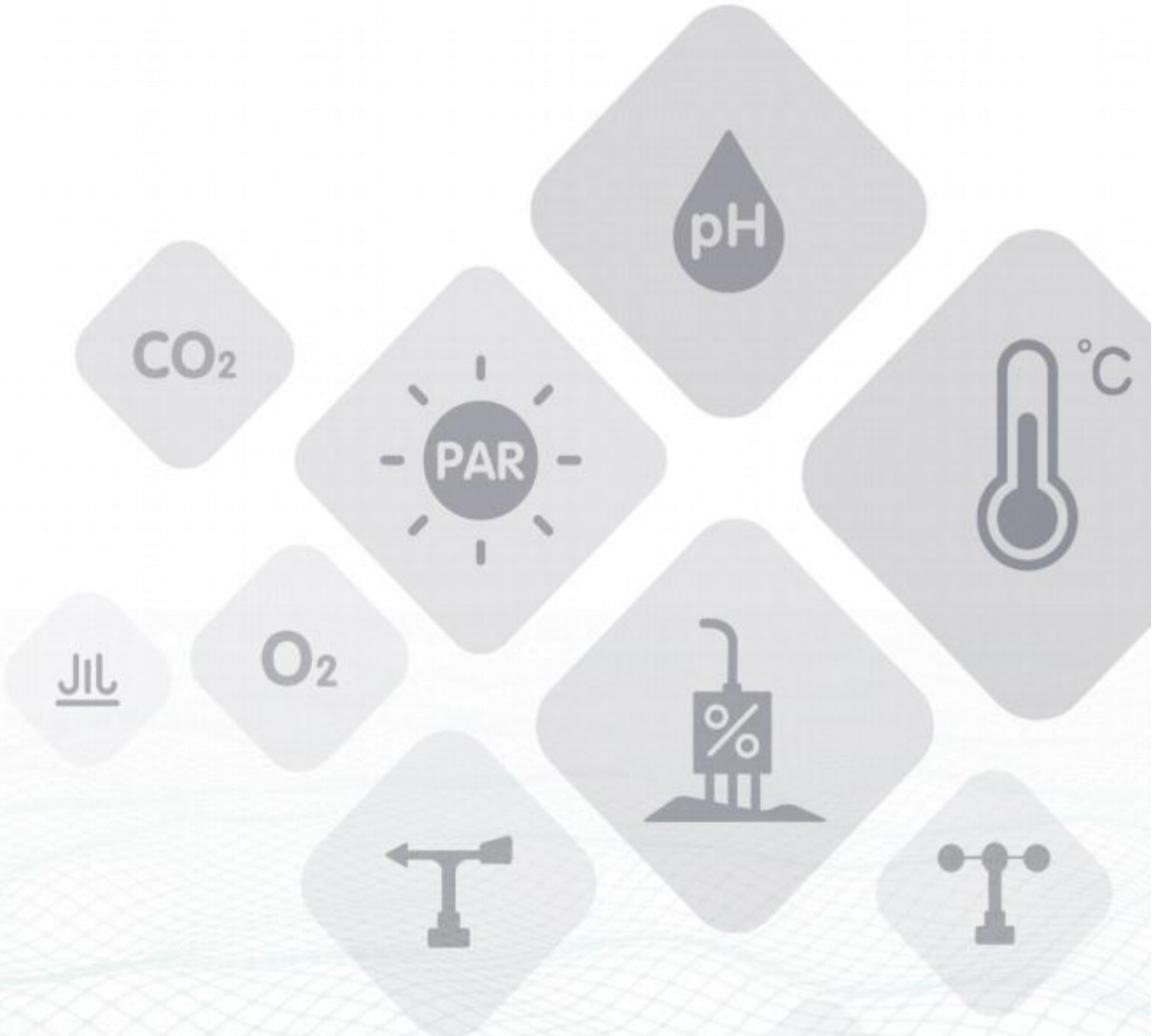


Table of contents

1. Product Introduction

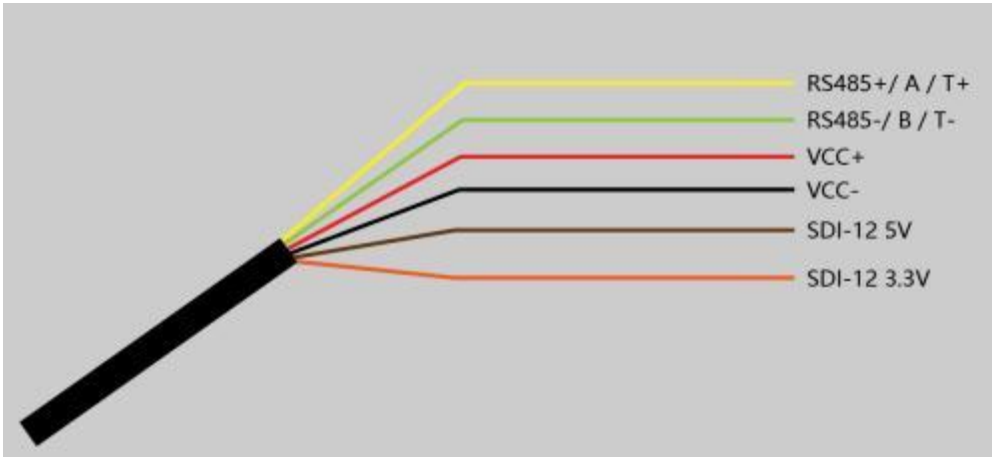
The sensor based on the non-dispersive infrared absorption principle - NDIR. It can continuously collect the carbon dioxide concentration, air temperature and air humidity in the air per unit volume and output it in the form of a common interface. It can monitor the environment in real time and provide users with reliable sensing data.

The device leads out 6 data lines, supports RS485 (Modbus -RTU / Modbus -ASCII) and SDI-12 two standard communication protocols, and is compatible with 5V ~ 24V external power supply voltage, so the device supports most data acquisition devices, users can obtain data and perform system integration out of the box without complex development. The sensor works in low power consumption mode, and the current can be as low as μA level(SDI-12).

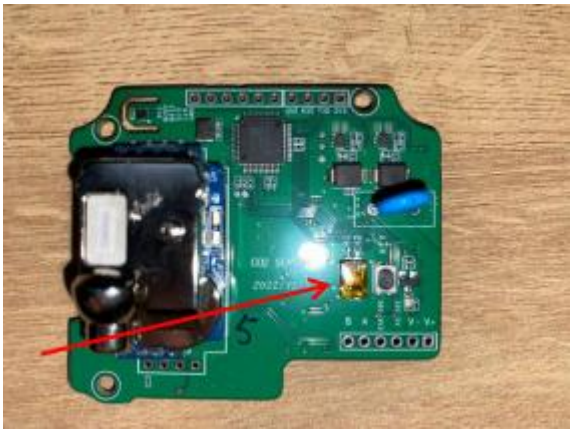
The working principle of the infrared non-dispersive absorption (NDIR) gas sensor is to calculate and determine the gas concentration by analyzing the relationship between gas concentration and absorption intensity (Lambert-Beer's law) according to the absorption characteristics of different gas molecules for the near-infrared spectrum. The sensor adopts the principle of non-spectral infrared absorption and adopts the design structure of single air chamber and double channel. At the same time, the device uses PTFE breathable membrane combined with a protective shell, which can improve the breathability based on protecting the sensor, thereby improving the accuracy of detection. The sensor uses high-quality chips, which work stably and are reliable in quality. The equipment is small and easy to install, and can be widely used in greenhouses, cities, and other scenarios.

2. Wiring

Sensor Wiring	
Yellow Line	RS485+/A/T+
Green Line	RS485-/B/T-
Red Line	VCC+
Black Line	VCC- (GND)
Brown Line	SDI-12 5V
Orange Line	SDI-12 3.3V



Mode selection (default RS-485 mode, if you switch to SDI-12, you need to open the cover and toggle the switch)	
RS-485 mode	Switch selection "ON" and " KE "
SDI-12 mode	Switch selection "1" and "2"



RS-485: “ON” and “KE”
SDI-12：“1” and “2”

3. List of parameters

Basic Parameters					
Product Model	S-CO2-03A / S-CO2-03B				
Supply Voltage	5 ~ 24 V				
Supporting Protocols	Modbus -RTU RS485/ Modbus -ASCII RS485/ SDI-12 (v1.4)				
IP Rating	IP65 (shading measures are required for outdoor installation, and the equipment cannot be exposed to direct sunlight and rainwater for a long time)				
Operating Temperature	0°C ~ +50°C (Due to CO2 sensor limitations)				
Working Humidity	0 ~ 85 %RH (no condensation)				
Cable Length	2 meters				
Measurement Parameters					
	Measuring Range	Measurement Accuracy			Resolution
Carbon Dioxide	0 ~ 10000ppm	±(40ppm+ 3 %*MV) ;			1ppm
Air Temperature	-40~85°C	±0.2℃			0.01°C
Air Humidity	0~100%RH	±1.8%RH			0.01%RH
Power Consumption Reference					
	Supply Voltage (V)	16	12	9	5
RS-485 mode	Working Current (mA)	11	15	19	33
SDI-12 mode	Working Current (mA)	10	14	18	31
	Sleep Current (μA)	288	284	528	755
performance reference					
	Parameter	Minimu	Typical	Maximum	Unit
RS-485 mode	Power warm-up time [1]	—	123 [2]	—	second(s)
	Measurement interval [3]	—	1000	—	millisecond (ms)
	Host Query Frequency [4]	—	1	—	Hz
	Read Response Time [5]	1	—	4	milliseconds (ms)
SDI-12 mode	Power warm-up time [6]	—	500	—	milliseconds (ms)
	Measurement time [7]	—	123 [8]	—	second(s)

[1] The preparation time required for the sensor to be powered on until the measured value of the sensor can be read through the Modbus command. This parameter should be paid attention to when the sensor is powered on again for each reading.

[2] Warm-up time = (register 0x0021 value + 3); read register 0x0000 before the warm-up time will get 0 and update the reading every second after the warm-up time; sensor T90 time is 300 seconds, T60 time is 120 seconds.

[3] The measurement data update interval, after the power-on warm-up time, if the power supply continues, the sensor will periodically update the readings at this interval.

[4] Modbus host query frequency.

[5] When the response delay register 0x0020 is set to 0, after the sensor receives the read command, it is the time to start sending the response.

[6] The preparation time required for the sensor to be powered on to receive SDI-12 commands, this parameter should be paid attention to when re-powering the sensor for each reading.

[7] The time required for the data collector to send the last bit of the aM! instruction until the sensor responds to the service request, when the data collector does not wait for the service request but directly sends the aD0! instruction to read the sensor after a period of delay Care should be taken with this parameter.

[8] Warm-up time = (register 0x0021 value + 3) ; read register 0x0000 before the warm-up time will get 0, and update the reading every second after the warm-up time; sensor T90 time is 300 seconds, T60 time is 120 seconds .

4.RS-485 Communication

4.1 Modbus-RTU RS-485 Protocol

Modbus protocol is a common protocol used in electronic equipment. Through this protocol, network communication is carried out between devices. It has become a common industry standard and is widely used in data collectors, sensor equipment, etc. Based on this protocol, devices produced by different manufacturers can communicate with each other for system integration.

Modbus protocol is a master-slave protocol. One node is the master, and other nodes participating in the communication using the Modbus protocol are slaves. Each slave device has a unique address. The sensor has an RS485 interface and supports Modbus-RTU protocol, and the sensing data and communication parameters can be obtained or modified by Modbus commands.

Note:

Default communication parameters: baud rate 9600bps, 1 start bit, 8 data bits, no parity, 1 stop bit.

4.2 Modbus register

Parameter	Register Address	Parameter Type	Function Number	Parameter Ranges and Descriptions	Defaults
Read-Only Register					
Carbon Dioxide	0x0000	uint16, read-only	3, 4	Unit: ppm	N/A
Air Temperature	0x0001	int16 (complement), read-only	3, 4	Unit: ℃	N/A
Air Humidity	0x0002	uint16, read-only	3, 4	Unit: %RH	N/A
Version	0x0007	uint16, read-only	3, 4	High byte: hardware version number Low byte: software version number	N/A
Communication Protocol Configuration Register					
Modbus Slave Address	0x0010	uint16, read and write	3, 6	1 ~ 247 (decimal)	The default is 45 (need to power on again after changing)
Serial Communication Baud Rate	0x0011	uint16, read and write	3, 6	0 ~ 7 0: 1200bps 1: 2400bps 2: 4800bps 3: 9600bps 4: 19200bps 5: 38400bps 6: 57600bps 7: 115200bps	Default 3 (requires power on after changing)

Serial Communication Parity	0x0012	uint16, read and write	3, 6	0 ~ 2 0: no parity 1: Odd parity 2: even parity	The default is 0 (you need to power on again after changing, **note**: when parity check is enabled,the host computer needs to set the data bit 7, and the safety communication protocol is ASCII)
Serial Communication Stop Bit	0x0013	uint16, read and write	3, 6	0 ~ 1 0: 1 stop bit 1: 2 stop bits	Default 0 (need to power on again after changing)
Modbus Protocol Type	0x0014	uint16, read and write	3, 6	0-1 0: Modbus -RTU 1: Modbus -ASCII	The default is 0 (need to power on again after changing)
SDI-12 Address	0x0015	uint16, read-only	3	ASCII code of a character	The ASCII code for '0' is 0x30
Function Configuration Register					
Delayed Response	0x0020	uint16, read and write	3, 6	0 ~ 65535 Unit: millisecond	10
CO2 Sensor Warm-Up Time	0x0021	uint16, read and write	3, 6	0 ~ 65535 Unit: second	120

- Error Code

According to the Modbus protocol, after a function code request is initiated, if an error occurs in the slave, an error response will be returned.

address field	error function code	error code	CRC check
1 byte	1 byte	1 byte	2 bytes
	Request function code +0x80	0x1: function code error 0x2: register address error 0x3: Writing or reading data exceeds the valid range 0x4: Slave internal error	

4.3 Modbus Register Parameter Description

Carbon Dioxide-CO2		
Parameter Range	0~ 10000 corresponds to 0 ~ 10000ppm	Register address: 0x0000
Example: If the returned value is 0x0AB2 (hexadecimal), then 0AB2 (HEX) = 2738 (DEC), then the measured value of carbon dioxide is 2738ppm		
Air Temperature		
Parameter Range	-4000-8000 corresponds to -40.00~80.00℃	Register address: 0x0001
Example: If the returned value is 0x 09C4 (hexadecimal), then 09C4 (HEX) = 2500 (DEC) =2500/100=25°C, then the air temperature measurement value is 25°C		
Air Humidity		
Parameter Range	0~ 9900 corresponds to 0~ 99%RH	Register address: 0x0002
Example: If the returned value is 0x 1388 (hexadecimal), then 1388 (HEX) = 5000 (DEC) =5000/100=50%RH, then the air humidity measurement value is 50%RH		
Version Number		
Parameter Range	High byte: hardware version number; low byte: software version number;	Register address: 0x0007
If the returned value is 0x0B0A, then hardware version = 0x0B / 10 = 1.1; software version = 0x0A / 10 = 1.0		

Modbus Slave Address		
Parameter Range	1 ~ 247 (default: 45)	Register address: 0x0010
After setting, you need to power off and restart to make this address take effect.		

Serial Communication Baud Rate		
Parameter Range	0 ~ 7 (default: 3)	Register address: 0x0011
0: 1200bps 1: 2400bps 2: 4800bps 3: 9600bps 4: 19200bps 5: 38400bps 6: 57600bps 7: 115200bps After setting, you need to power off and restart to make this address take effect.		

Serial Communication Parity		
Parameter Range	0 ~ 2 (default: 0)	Register address: 0x0012
0: no parity 1: Odd parity 2: even parity Note: When the parity check is enabled, the upper computer needs to set the data bit to 7, and the safety communication protocol is ASCII. After setting, you need to power off and restart to make this address take effect.		

Serial Communication Stop Bit		
Parameter Range	0, 1 (default: 0)	Register address: 0x0013
0: 1 stop bit 1: 2 stop bits After setting, you need to power off and restart to make this address take effect.		

Modbus Protocol Type		
Parameter Range	0, 1 (default: 0)	Register address: 0x0014
0: Modbus -RTU 1: Modbus -ASCII After setting, you need to power off and restart to make this address take effect.		

Delayed Response		
Parameter Range	0 ~ 65535 milliseconds (default: 10)	Register address: 0x0020
After receiving the host request command, the sensor is sampled, and after the sampling is completed, a delay is given for a period before a response is given. This command is mainly used in situations where the speed of the host is relatively slow when the host switches from the RS485 sending state to the receiving state. When set to 0, there is no additional delay. After setting, you need to restart the device to take effect.		

CO2 Sensor Warm-Up Time		
Parameter Range	0 ~ 65535 seconds (default: 120)	Register address: 0x0021
CO2 sensor warm-up time, reading register 0x0000 before the warm-up time will get 0, after the warm-up time, the reading is updated every second; T90 time is 300 seconds, T60 time is 120 seconds. (T90 is the time required to reach 90% accuracy) This configuration is still valid for SDI-12 mode. After setting, you need to restart the device to take effect.		

4.4 Modbus Protocol Communication

In the following description, the data starting with 0x or ending with H is hexadecimal data. The Modbus protocol has two commonly used register types:

(1) Holding registers, the stored data will not be lost when power is off, and it is readable and writable. Usually read with function number 3 (0x03) and write with function number 6 (0x06) or 16 (0x10).

(2) Input registers are used to store some read-only physical quantities, such as temperature values, which are read-only. Usually read with function number 4 (0x04).

4.4.1 Function Code 3 Communication Example

General request format: AA 03 RRRR NNNN CCCC

AAA	03	RRRR	NNNN	CCCC
1 byte	1 byte	2 bytes	2 bytes	2 bytes
Device address, range 0-247	Function number is 3	Start register address, first byte	Number of registers to read N, high byte first	CRC check

Common response format: AA 03 MM VV0 VV1 VV2 VV3... CCCC

AAA	03	MM	VV0	VV2	...	CCCC
1 byte	1 byte	2 bytes	2 bytes	2 bytes	...	2 bytes
Device address, range 1-247	Function number is 3	Returns the number of data bytes of the register value	return the first register value	returns the value of the second register	The returned Nth register value (N=MM/2)	CRC check

Example: read registers 0x0000 ~ 0x0003, that is, the measured values of carbon dioxide , air temperature, and air humidity

Request: 2D 03 0000 000 3 0267 (CRC check)

2D	03	0000	000 3	0267
1 byte	1 byte	2 bytes	2 bytes	2 bytes
Device Address 0x 2D	Function number is 3	Start register address 0x00	read 3 registers	CRC check

Response: 2D 03 06 00 00 0A C8 10 66 E278 (CRC check)

2D	03	0 6	0000	0AC8	1066	E278
1 byte	1 byte	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes

Device Address 0x 2D	Function number is 3	Return 6 bytes of register data	Returns the value of register 0x0000 (CO2)	Returns the value of register 0x0001 (air temperature)	Returns the value of register 0x0002 (air humidity)	CRC check

Note: The register data of CO2, air temperature and air humidity can be read separately.

4.4.2 Function Code 6 communication example

General request format: AA 06 RRRR NNNN CCCC

AAA	06	RRRR	NNNN	CCCC
1 byte	1 byte	2 bytes	2 bytes	2 bytes
Device address, range 0-247	Function number is 6	Register address to be written, high byte first	Write the value of the register, high byte first	CRC check

Generic response format: AA 06 RRRR VVVV CCCC

AAA	06	RRRR	VVVV	CCCC
1 byte	1 byte	2 bytes	2 bytes	2 bytes
Device address, range 0-247	Function number is 6	Register address to be written, high byte first	Write the value of the register, high byte first	CRC check

Example: Write register 0x0010, modify the slave address of the device to 0x 2D (decimal 45)

Request: 01 06 0010 00 2D 4812

01	06	0010	00 2D	4812
1 byte	1 byte	2 bytes	2 bytes	2 bytes
Device address, range 0-247	Function number is 6	Register address to be written, high byte first	Write the value of the register, high byte first	CRC check

Response: 01 06 0010 00 2D 4812

01	06	0010	00 2D	4812
1 byte	1 byte	2 bytes	2 bytes	2 bytes
Device address, range 0-247	Function number is 6	Register address to be written, high byte first	Write the value of the register, high byte first	CRC check

4.4.3 CRC check algorithm and examples

```
static const unsigned char aucCRCHi[] = {
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
};

static const unsigned char aucCRCLo[] = {
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
};

unsigned short usCRC16( unsigned char * pucFrame, unsigned short usLen )
{
unsigned char ucCRCHi = 0xFF;
unsigned char ucCRCLo = 0xFF;
int iIndex;

while( usLen-- )
{
iIndex = ucCRCLo ^ *( pucFrame++ );
ucCRCLo = ( UCHAR )( ucCRCHi ^ aucCRCHi[iIndex] );
ucCRCHi = aucCRCLo[iIndex];
}
return ( unsigned short )( ucCRCHi << 8 | ucCRCLo );
}
```

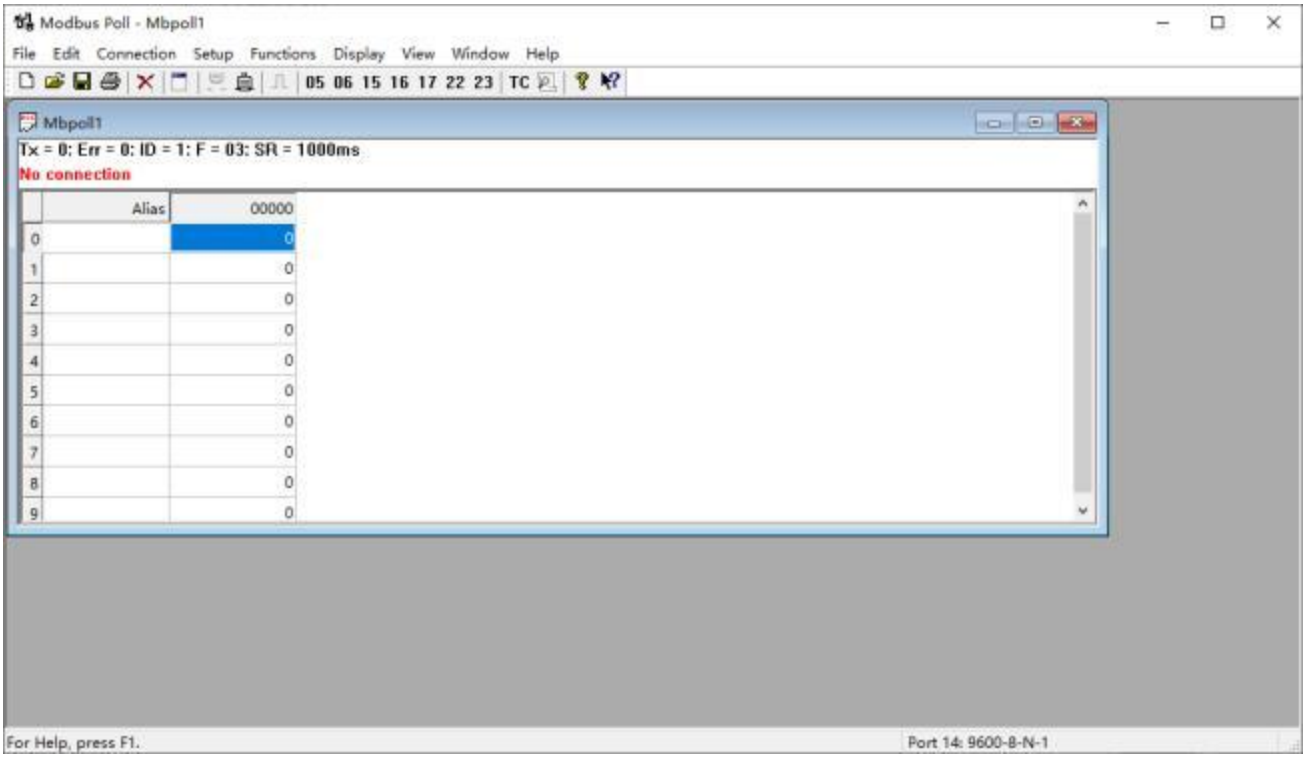
The CRC generated by this function has exchanged the high and low bytes, and can be directly put into the message for sending.

Example: Calculated by this function, the CRC16 of a certain frame is equal to 0x4112, and the inserted message is as follows:

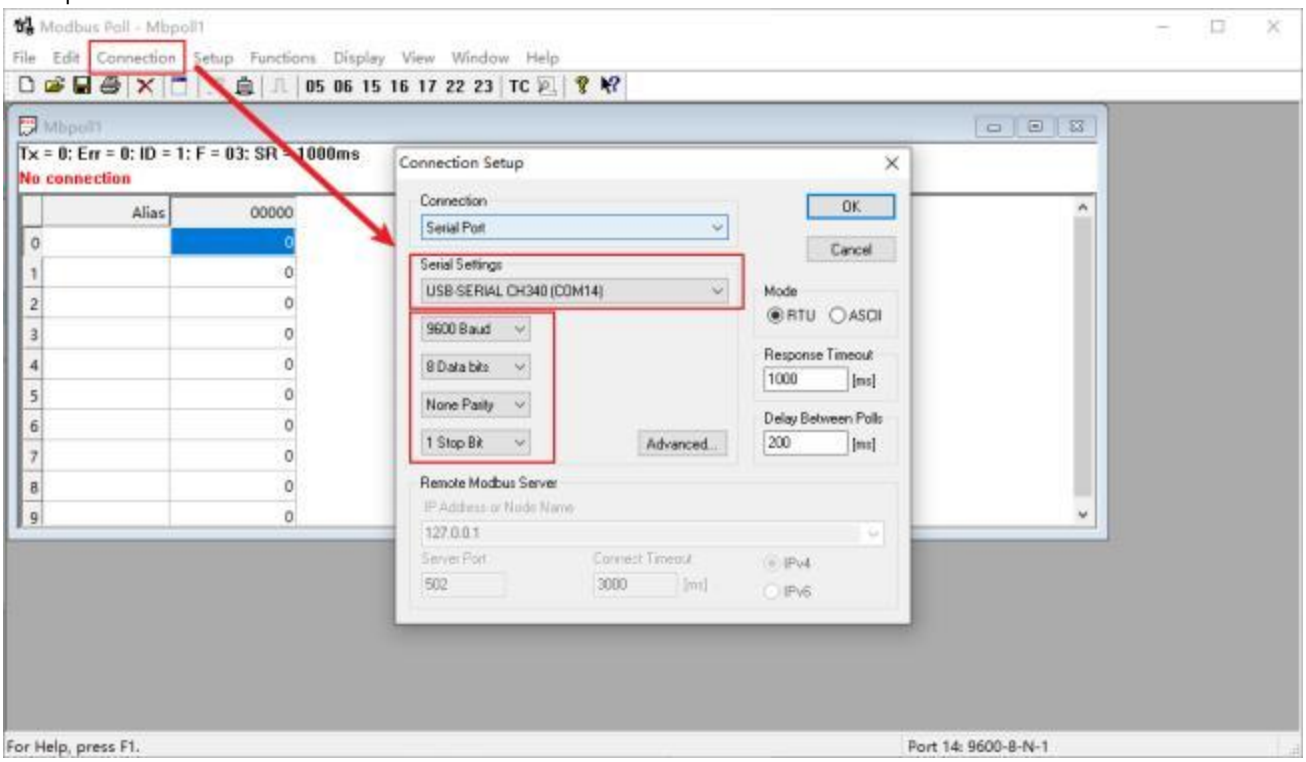
Address	Function	Data number	data	data	data	data	CRC high	CRC low
							0x41	0x12

4.5 Using Modbus Debugging Software

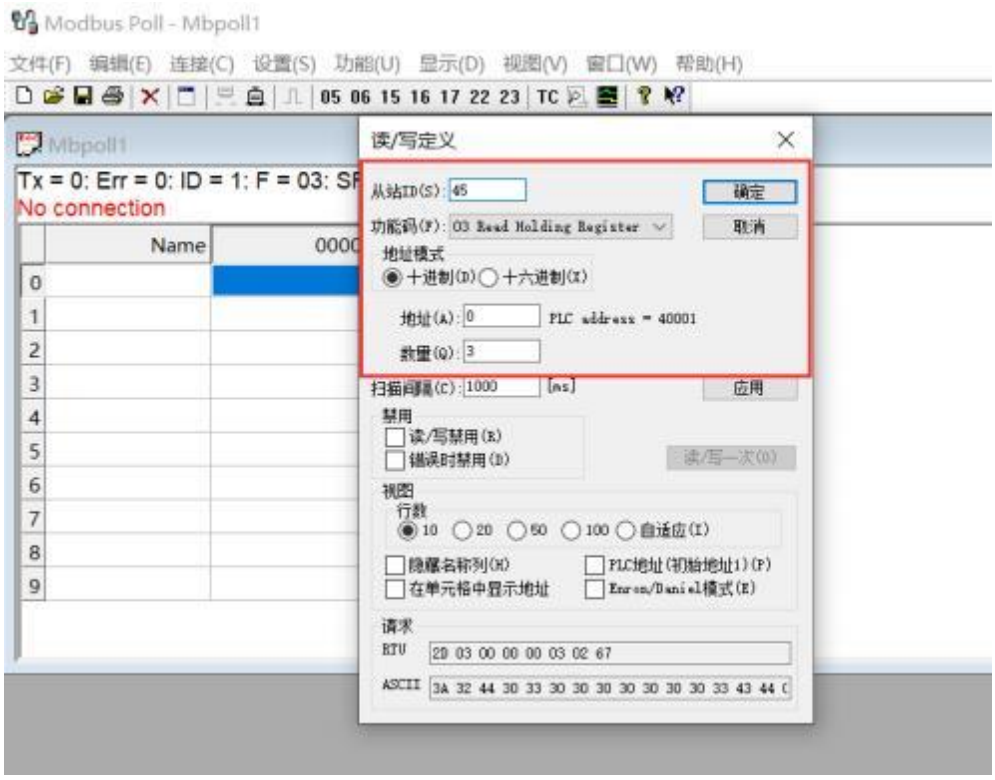
Here we take the Modbus Poll tool (download address:https://www. Modbus tools.com/download.html) as an example.



Configure connection parameters: baud rate 9600bps, 1 start bit, 8 data bits, none parity, and 1 stop bit.



Configure the parameters of the read register 0x0000 ~ 0x000 3: the default slave address is 45 , the function code is 03, the starting address is 0, and the quantity is 3



5.SDI-12 communication

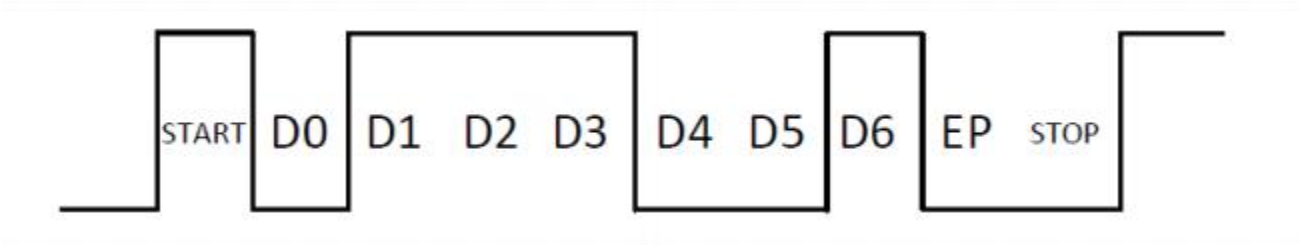
5.1 SDI interface overview

SDI-12 communication uses three wires, two of which are sensor power supply wires and the other one is SDI-12 signal wire.

Each sensor on the SDI-12 bus has a unique bus address, and the sensor address can be set as "0", "1"~"9", "A"~"Z", "a"~"z". The SDI-12 address of the CO2, Temperature and Humidity sensor is "0" by default. The commands supported by this sensor are shown in "SDI-12 command analysis", each of which complies with the SDI-12 v1.4 standard.

The power supply of the sensor adopts a DC power supply of 5 ~ 24 V. After the sensor is powered on, it immediately enters the sleep mode and waits for the data acquisition device to issue instructions. SDI-12 uses a serial baud rate of 1200bps, 1 start bit (high level), 7 data bits (high 0 low 1, negative logic), 1 even parity bit, and 1 stop bit.

The timing of each byte sent is shown in the figure below:



5.2 SDI-12 instruction analysis

Query Sensor Address		
Order	?!	? - address wildcard ! - end of command
Response	a<CR><LF>	a - address <CR><LF> - Response terminator
Example	Command: ?! Response: 0<CR><LF>	The sensor at address '0' responded to the query

Query The Online Status Of The Sensor		
Order	a!	a - address ! - end of command
Response	a<CR><LF>	a - address <CR><LF> - Response terminator
Example	Command: 0! Response: 0<CR><LF>	Sensor with address '0' is online

Query Sensor Information		
Order	aI!	a - address I – Sensor ID command ! - end of command
Response	allccccccccmmmmmmmvvvxxx . . . xxx<CR><LF>	a - address II - 2 characters, SDI-12 protocol version, for example, 14 represents version 1.4 protocol cccccccc - 8 characters, company name or product name mmmmmm - 6 characters, sensor model number vvv - 3 characters, sensor software version xxx . . . xx - optional, up to 13 characters, can be used to send serial number, or other information <CR><LF> - Response terminator
Example	Command: 0I! Response: 0I!014SENSECAPSCO2032.001234567CO2TH	0I!014SENSECAPSCO2032.001234567CO2TH 0I! - Instruction 0 - address 14 - Version 1.4 SENSECAP - product name SCO203 - Product model number, S-CO2- 03x 2.0 - Product software version

		01234567 - 8 characters, serial number CO2TH - output physical quantity

Modify Sensor Address		
Order	aAb!	a – current address A - Address Change b - the new address ! - end of command
Response	b<CR><LF>	b - new address <CR><LF> - Response terminator
Example	Command: 0A1! Response: 1<CR><LF>	Change the address 0 to 1, and the modification is successful after receiving the response

Turn On Sensor Measurement		
Order	aM! or aMC!	a - address M - Measure C - CRC ! - end of command
Response	atttn<CR><LF>	a - address ttt - 3 characters, measurement end time, unit: second n - 1 character, the number of physical quantities to be output <CR><LF> - Response terminator When the data collector issues the aMC! command, the sensor will return data with CRC.
Example	Command: 0M! Response: 01233 <CR><LF>	0M! - command 0 - address 00 123 - The measurement ends after a maximum of 28 seconds 3 - will output 1 physical quantity
Request for service service request	a<CR><LF>	a<CR><LF> a - address When the data logger sends the measurement command, the sensor responds to atttn<CR><LF> immediately, and then responds to the service request after the measurement is over, informing the

		data logger that data collection is possible. The data logger should not request other sensors after issuing the aM! command and before receiving the service request, unless a break is used to interrupt the measurement (the measured value will not be updated).

Read Sensor Data		
Order	aD0!	a - address D0 - Data (D1 . . . D9 and so on) ! - end of command
Response	a<values><CR><LF> or a<values><CRC><CR><LF>	a<values><CR><LF> or a<values><CRC><CR><LF> a - address values - see below <CRC> - 3 characters, CRC, carried when responding to aMC! command <CR><LF> - Response terminator values: Measurement values, consisting of the following parts: <symbol><integer>[.<decimal>]
Example	Command: 0D0! Response: +450+28.09+61.95	For example: 0D0!0+450+28.09+61.95 0D0! - command 0 - address +450 - CO2 concentration, ppm +28.09 - air temperature, degrees Celsius +61.95 - air humidity, percent

5.3 Precautions for Using SDI-12

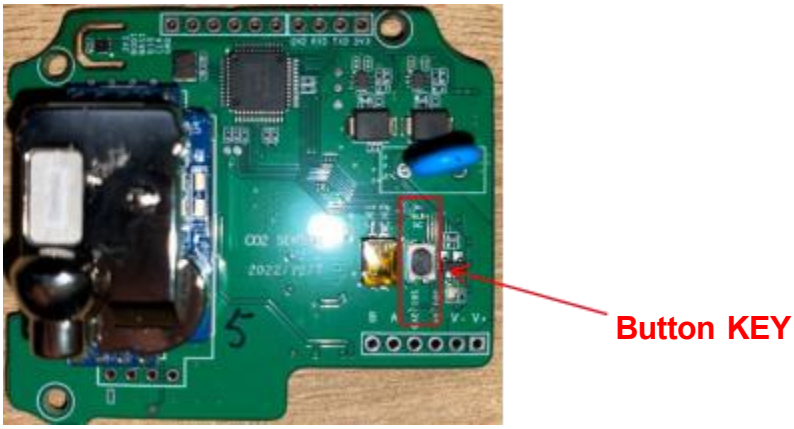
(1) Multiple sensors can be mounted on the SDI-12 bus, but attention should be paid to sensor status maintenance and timely detection of failed sensors, because a sensor failure may affect the normal operation of the entire bus, even if other sensors are normal.

(2) When the data collector operates the sensor, the logic should include retry, otherwise there will be a certain probability that the data cannot be read due to cable interference, baud rate deviation and other reasons.

6. CO2 Calibration Process

When it is found that the CO2 data of the sensor is very different from the actual environment, it is recommended to use this function. The sensor supports manual calibration with the click of a button.

The specific method is: open the upper cover of the sensor, place it in an outdoor ventilated environment, power on and press the button "KEY" for 5 seconds, the LED light is always on, and the preheating light on the CO2 module starts to flash, release it Press the button and wait for about 10 minutes . After the CO2 sensor is fully warmed up for 10 minutes, the MCU will write the 400ppm calibration information to the CO2 sensor, and then the sensor will automatically restart to complete the calibration.



Tips: During the calibration process, the equipment only needs to be powered on all the time without any communication instructions. During the calibration process, the sensor does not respond to any Modbus or SDI-12 requests.

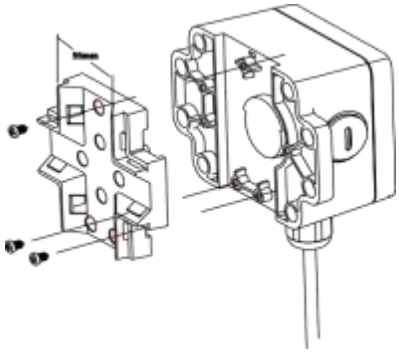
Note: During the calibration process, the surrounding environment of the sensor is guaranteed to be in a state of no one (human breathing action can increase the concentration of the surrounding environment, which may cause calibration failure)

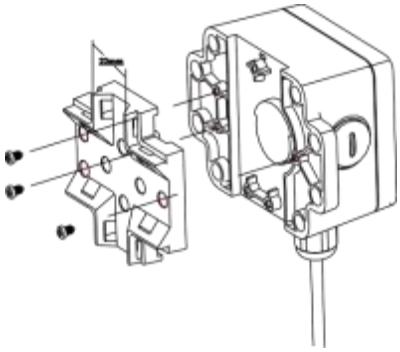
7.Installation Instructions

Two installation methods are introduced below: one is fixed on the pole, and the other is fixed on the wall.

A. Pole installation:

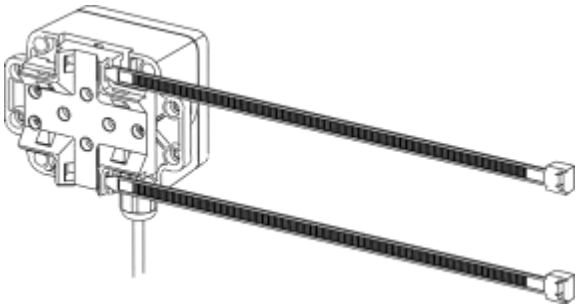
(1) The mounting rails are on the rear of the device:



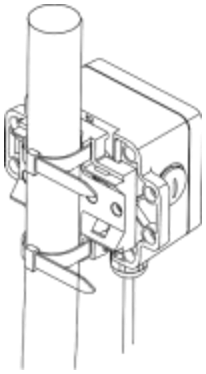


The first type (recommended rod diameter: 20-30mm) The second type (recommended rod diameter: 50- 60mm)

(2) Pass the cable tie through the hole of the guide rail, as shown in the figure:



(3) To lock the device to the pole:



B. Wall mount:

Fix the device with self-tapping screws according to the hole position, as shown in the figure: