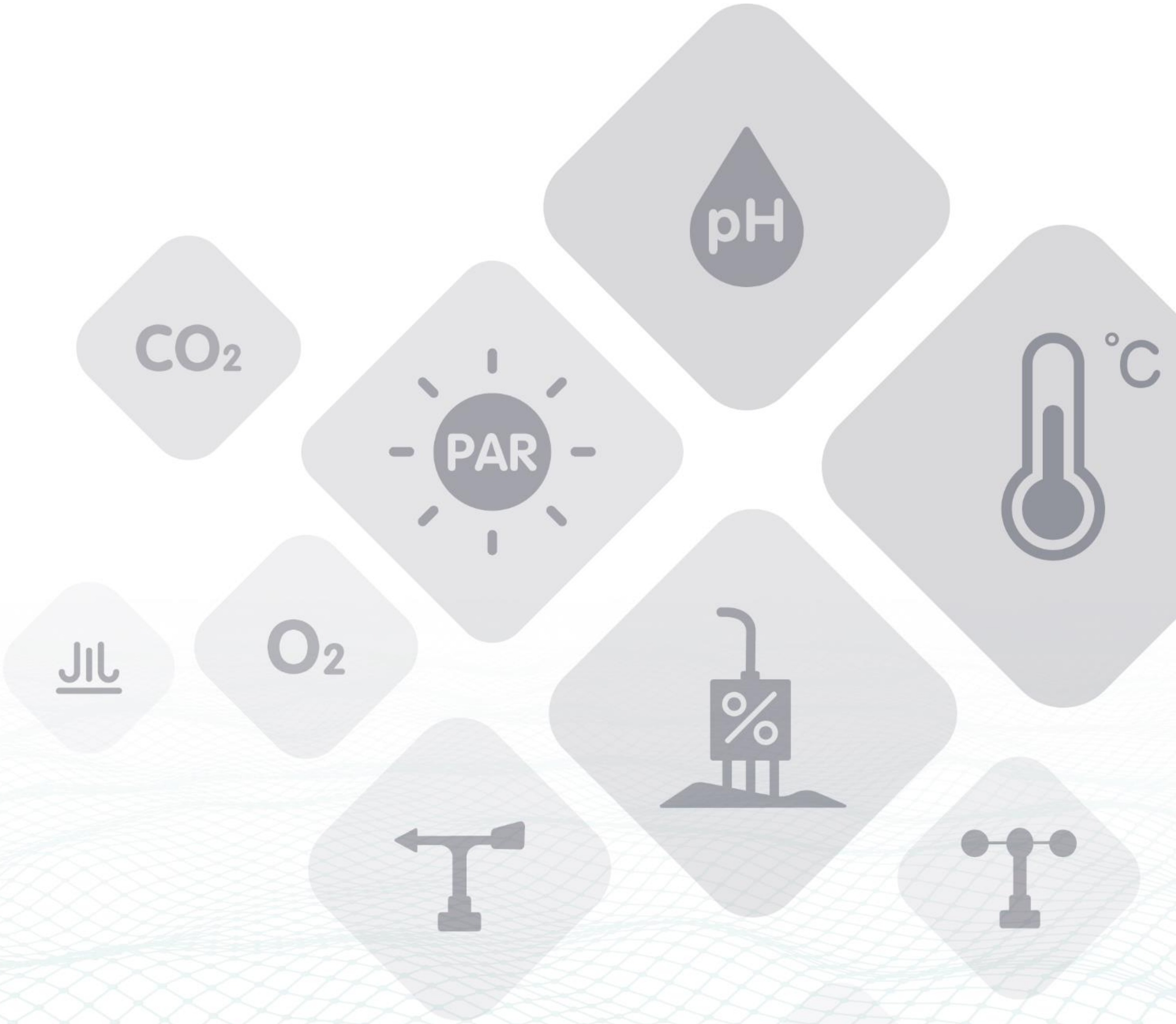
SENSECAP

SenseCAP SOLO CO2 5000 User Guide

Version: V1.0



1 Product Introduction

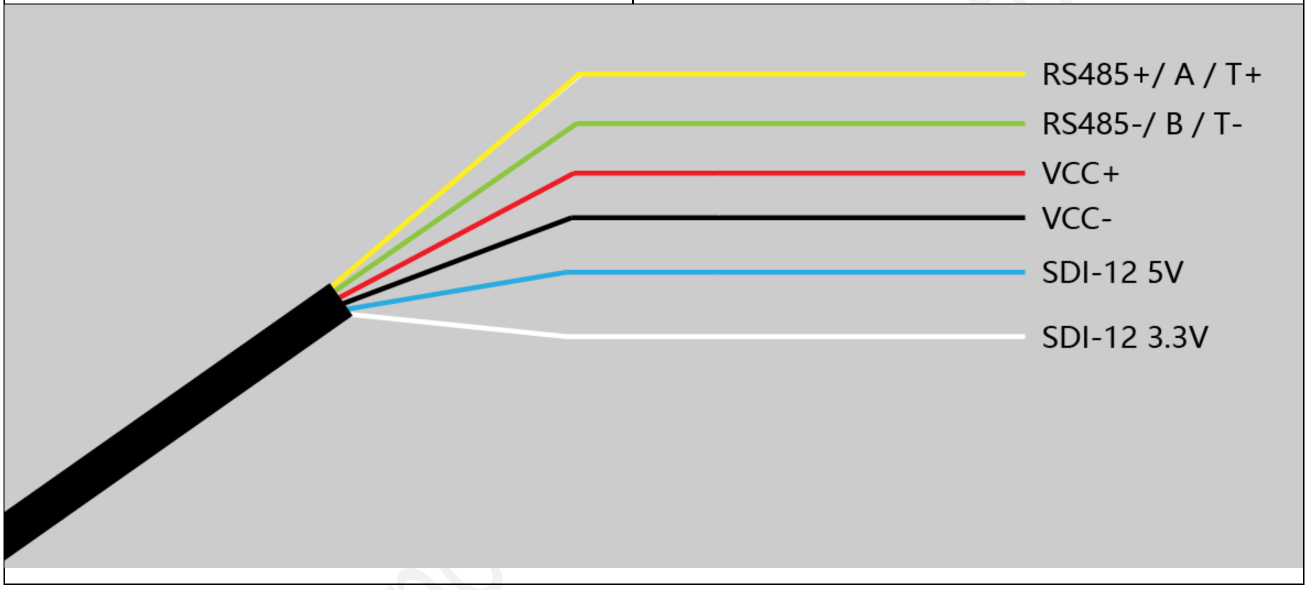
SenseCAP SOLO CO2 5000 is a digital CO2 concentration sensor based on NDIR, which can continuously collect and calculate the CO2 concentration in the air per unit volume and output it in the form of universal interface. It can monitor the environment in real time and provide users with reliable sensing data.

The device leads out 6 data lines, which support MODBUS (Modbus-RTU/Modbus-ASCII) RS485 and SDI-12 communication protocols, and is compatible with the wide power supply voltage of 5V ~ 16V. Therefore, the device supports the vast majority of data acquisition devices, and users do not need complicated development to obtain data and fast system integration. The sensor works in low power mode with current as low as uA.

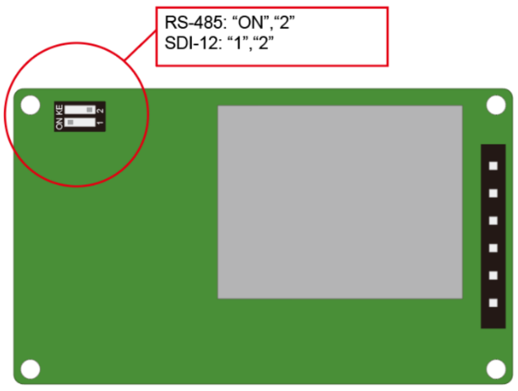
The working principle of NDIR gas sensor is to calculate and determine the concentration of gas according to the absorption characteristics of different gas molecules for near infrared spectrum through the analysis of the relationship between gas concentration and absorption intensity. The sensor adopts the principle of non-spectral infrared absorption and adopts the design structure of single air chamber and double channel. At the same time, the sensor adopts PTFE film combined with protective enclosure, which can improve the breathable performance, thus improving the accuracy of detection. The sensor is made of excellent chip with stable operation and reliable quality. The sensor is small in size and easy to install. It can be widely used in greenhouses, cities and other scenes.

2 Wiring

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



Mode Selection (Default: RS-485 Mode, open the cover to switch SDI-12)	
RS-485 Mode	Switch to choose "ON" and "2"
SDI-12 Mode	Switch to choose "1" and "2"



3 Specifications

General Parameters					
Product Model	SenseCAP SOLO CO2 5000				
Power Supply	5 ~ 16V				
Protocol	MODBUS-RTU RS485/ MODBUS-ASCII RS485/ SDI-12 (v1.4)				
IP Rating	IPx5, Indoor, the PTFE filter is not waterproof				
Operating Temperature	-10 ℃ ~ +50 ℃				
Operating Humidity	0 ~ 85 %RH (non-condensing)				
Measurement Parameters					
	Range	Accuracy			Resolution
CO2	400 ~ 5000 ppm	±(50ppm+5%*MV) Effective range: 400~5000ppm			1 ppm
Current Consumption					
	Power Supply (V)	16	12	9	5
RS-485	working current (mA)	22	28	34	57
SDI-12	working current (mA)	20	27	33	55
	Sleep current (μA)	28	29	30	35
Program Parameters					
	Parameter	Min	Typical	Max	Unit
RS-485	Warm-up Time[1]	—	123[2]	—	s
	Scan Interval [3]	—	1000	—	s
	Poll Rate[4]	—	1	—	Hz
	Response Time [5]	1	—	4	millisecond
SDI-12	Warm-up Time [6]	—	15	—	millisecond
	Scan Interval [7]	—	123[8]	—	s

[1] The time from when the sensor is powered on to when the data is read. Note the parameter when the sensor is powered on.

[2] Warm-up time = (register 0x0021 value +3). Reading register 0x0000 before the warm-up time will result in 0, and reading will be updated once per second after the warm-up time. The sensor T90 time is 300 seconds, and T60 time is 120 seconds.

[3] Measure data update time interval. The sensor periodically updates the readings after the power-on warm-up time, if the power supply continues.

[4] Modbus master poll rate.

[5] When the delay response register 0x0020 is set to 0, the time from the sensor receiving the read instruction to the start of sending data.

[6] The time from when the sensor is powered on to when the data is read. Note the parameter when the sensor is powered on.

[7] It refers to the time between the data logger sending the ‘aM!’ and the sensor responding to the service request. Note the parameter when the data logger does not wait for the service request but delays for a period of time and directly send ‘aD0!’.

[8] Scan interval time = (register 0x0021 value +3). Reading register 0x0000 before the warm-up time will result in 0, and reading will be updated once per second after the warm-up time. The sensor T90 time is 300 seconds, and T60 time is 120 seconds.

4 How to work on RS-485

4.1 Introduction to Modbus-RTU RS-485

Modbus protocol is a common protocol used in electronic devices. Through this protocol, the devices communicate with each other. It has become a common industry standard, widely used in data logger, sensor equipment and so on. Based on this protocol, devices from different vendors can communicate with each other for system integration.

Modbus protocol is a master-slave architecture protocol. One node is the Master and the other nodes that use Modbus protocol to participate in the communication are Slaves. Each slave device has a unique address. The SenseCAP ORCH S4 has RS485 interface and supports Modbus-RTU and Modbus-RTU ASCLL. Modbus commands can be used to obtain sensor data or modify communication parameters.

Note:

Default communication parameters: baud rate 9600bps, one start bit, eight data bits, no check, one stop bit.

4.2 Modbus Register

Parameter	Register Address	Type	Function Code	Range and Descript	Default
Read-only Register					
CO2	0x0000	uint16, read-only	3、4	Value=Register Value Unit: ppm	N/A
Version	0x0007	uint16, read-only	3、4	High Byte: Hardware Version Low Byte: Software Version	N/A
Protocol Configuration Register					
Modbus Slave Address	0x0010	uint16, read-only	3、6	1 ~ 247	1
Baud Rate	0x0011	uint16, read-only	3、6	0 ~ 7 0：1200bps 1：2400bps 2：4800bps 3：9600bps 4：19200bps 5：38400bps 6：57600bps 7：115200bps	3
Parity	0x0012	uint16, read-only	3、6	0 ~ 2 0：None 1：Odd 2：Even	0
Stop Bit	0x0013	uint16, read-write	3、6	0 ~ 1 0：1 Stop Bit 1：2 Stop Bit	0
Modbus protocol	0x0014	uint16, read-write	3、6	0-1 0：Modbus-RTU 1：Modbus-ASCII	0
Function Configuration Register					
Delay Response	0x0020	uint16, read-write	3、6	0 ~ 65535 Unit: millisecond	10
Warm-up Time	0x0021	uint16, read-write	3、6	0 ~ 65535 Unit: millisecond	120

- **Error Code**

If an error occurs from the slave, an error response is returned.

Address	Error Function Code	Error Code	CRC Check
1 byte	1 byte	1 byte	2 byte
	Request Code+0x80	0x1 : Function code error 0x2 : Register address error 0x3 : Write or read data out of range 0x4 : Slave internal error	

4.3 Modbus Register Detail Description

CO2		
Range	0 ~ 5000 corresponding 0 ~ 5000 ppm	Register Address: 0x0000
For example: If the returned value is 0x0AB2 (HEX), 0AB2 (HEX) = 2738 (DEC), then the temperature measurement is 2738ppm		

Version		
Range	High byte: hardware version; Low byte: software version	Register Address: 0x0007
If the return value is 0x0B0A, the hardware version = 0x0B / 10 =1.1; software version = 0x0A / 10 = 1.0		

Modbus Slave Address		
Range	1 ~ 247 (default: 1)	Register Address: 0x0010
The device needs to be restarted to take effect after setup.		

Baud Rate		
Range	0 ~ 7 (default: 3)	Register Address: 0x0011
0: 1200bps 1: 2400bps 2: 4800bps 3: 9600bps 4: 19200bps 5: 38400bps 6: 57600bps 7: 115200bps The device needs to be restarted to take effect after setup.		

Parity Bits		
Range	0 ~ 2 (default: 0)	Register Address: 0x0012
0: None 1: Odd 2: Even **Note:** when parity is enabled, the Master should set the data bit to 7 and the secure communication protocol to Modbus-ASCII. The device needs to be restarted to take effect after setup.		

Stop Bits		
Range	0 、1 (default: 0)	Register Address: 0x0013
0: 1 Stop bit 1: 2 Stop bits The device needs to be restarted to take effect after setup.		

Modbus Protocol		
Range	0 、1 (default: 0)	Register Address: 0x0014
0：Modbus-RTU 1：Modbus-ASCII The device needs to be restarted to take effect after setup.		

Delay Response		
Range	0 ~ 65535 millisecond (default: 10)	Register Address: 0x0020
After receiving the request from the Master, the sensor will be sampled, and the response will be given after the completion of the sampling. This command is mainly used in the situation where the speed of the Master is slow when switching from RS485 sending state to receiving state. When set to 0, there is no extra delay. The device needs to be restarted to take effect after setup.		

Warm-up Time		
Range	0 ~ 65535 s (default: 120)	Register Address: 0x0021
Warm-up time of CO2 sensor. Reading register 0x0000 before the preheating time will get 0. Reading will be updated once per second after the preheating time. The T90 time is 300 seconds and the T60 time is 120 seconds. (T90 is the time required to achieve 90% accuracy) This configuration is still valid for SDI-12 mode. The device needs to be restarted to take effect after setup.		

4.4 Example

In the following instructions, data beginning with 0x or ending with H is hexadecimal. The Modbus protocol has two common register types:

(1) keep the register, the storage data is not lost power, is readable and writable. Normally read with function number 3 (0x03) and write with function number 6 (0x06) or 16 (0x10).

(2) input register, used to store some read-only physical quantities, such as temperature value, is read-only. Normally read with function number 4 (0x04).

4.4.1 Example of Function Code 3

Request command: AA 03 RRRR NNNN CCCC

AA	03	RRRR	NNNN	CCCC
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Slave address, range 0-247	Function Code is 3	Start register address, high byte first	The number of registers to read, high byte first	CRC check

Response: AA 03 MM VV0 VV1 VV2 VV3… CCCC

AA	03	MM	VV0	VV2	…	CCCC
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes	…	2 Bytes
Slave address, range 0-247	Function Code is 3	Returns the number of data bytes of the register value	The first register value returned	The second register value returned	The N register value returned (N=MM/2)	CRC check

For example: Read registers 0x0000 to read the measured values of CO2.

Request: 01 03 0000 0001 840A

01	03	0000	0001	840A
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Slave address 0x01	Function Code is 3	Start register address: 0x00	Read 1 register	CRC check

Response: 01 03 02 02E6 38AE

01	03	02	02E6	38AE
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Slave address 0x01	Function Code is 3	Returns 2 bytes of register data	Returns the value of register 0x0000	CRC check

4.4.2 Example of Function Code 6

Request: AA 06 RRRR NNNN CCCC

AA	06	RRRR	NNNN	CCCC
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Slave address, range 0-247	Function Code is 6	Write to the register address, high byte first	Write the value of the register, high byte first	CRC check

Response: AA 06 RRRR VVVV CCCC

AA	06	RRRR	VVVV	CCCC
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Slave address, range 0-247	Function Code is 6	Write to the register address, high byte first	Write the value of the register, high byte first	CRC check

For example: Write register 0x0010 and modify the slave address of the device to 0x02.

Request: 01 06 0010 0002 09CE

01	06	0010	0002	09CE
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Slave address, range 0-247	Function Code is 6	Write to the register address, high byte first	Write the value of the register, high byte first	CRC check

Response: 01 06 0010 0002 09CE

01	06	0010	0002	09CE
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Slave address, range 0-247	Function Code is 6	Write to the register address, high byte first	Write the value of the register, high byte first	CRC check

4.4.3 CRC Check algorithm

```
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 );
}
```

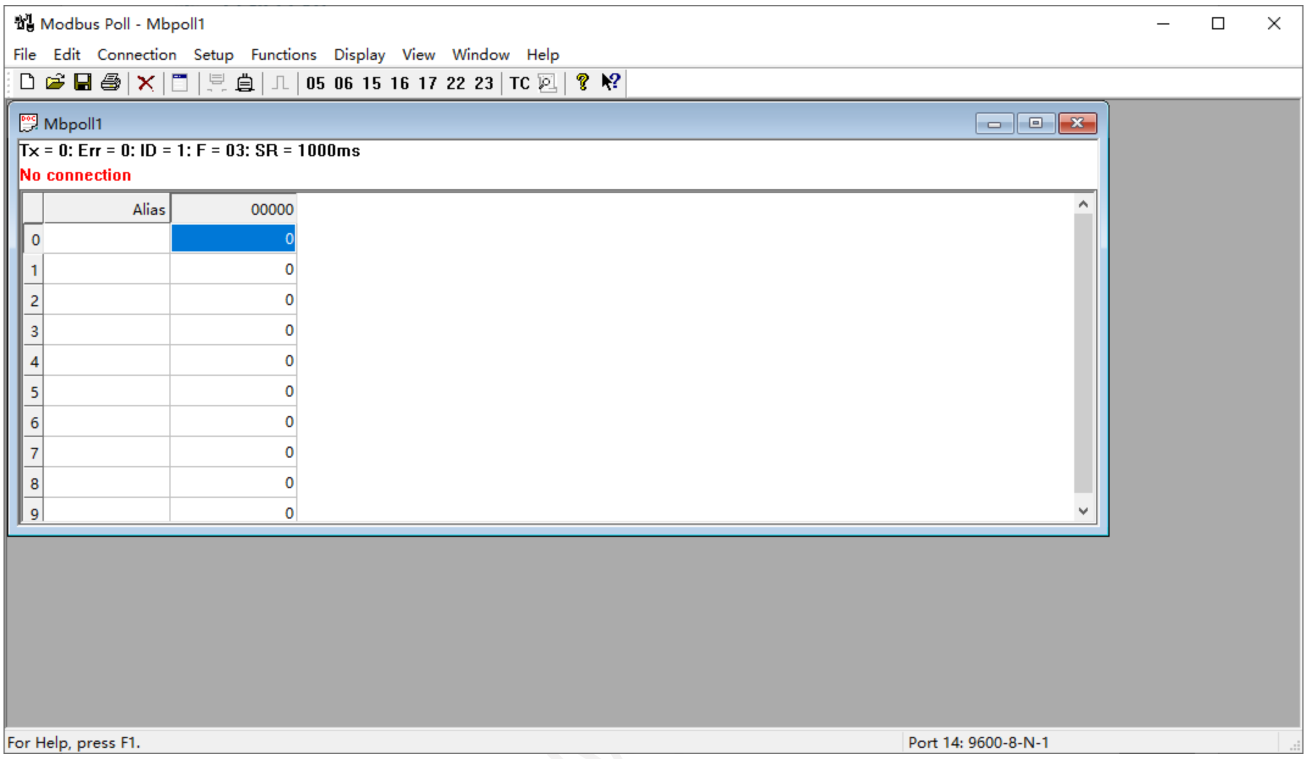
The CRC generated by this function has exchanged high and low bytes and can be sent directly into a message.

For example: The CRC16 of a certain frame is calculated by this function to be equal to 0x4112, then the message is placed as follows:

AA	03	RRRR	NNNN	CC	CC
1 Byte	1 Byte	2 Bytes	2 Bytes	1 Byte	1 Byte
Slave address, range 0-247	Function Code is 3	Start register address, high byte first	The number of registers to read, high byte first	CRC check, low byte 0x41	CRC check, high byte 0x12

4.5 Modbus Poll Tool

Take the Modbus Poll tool as an example.
Download: https://www.modbustools.com/download.html

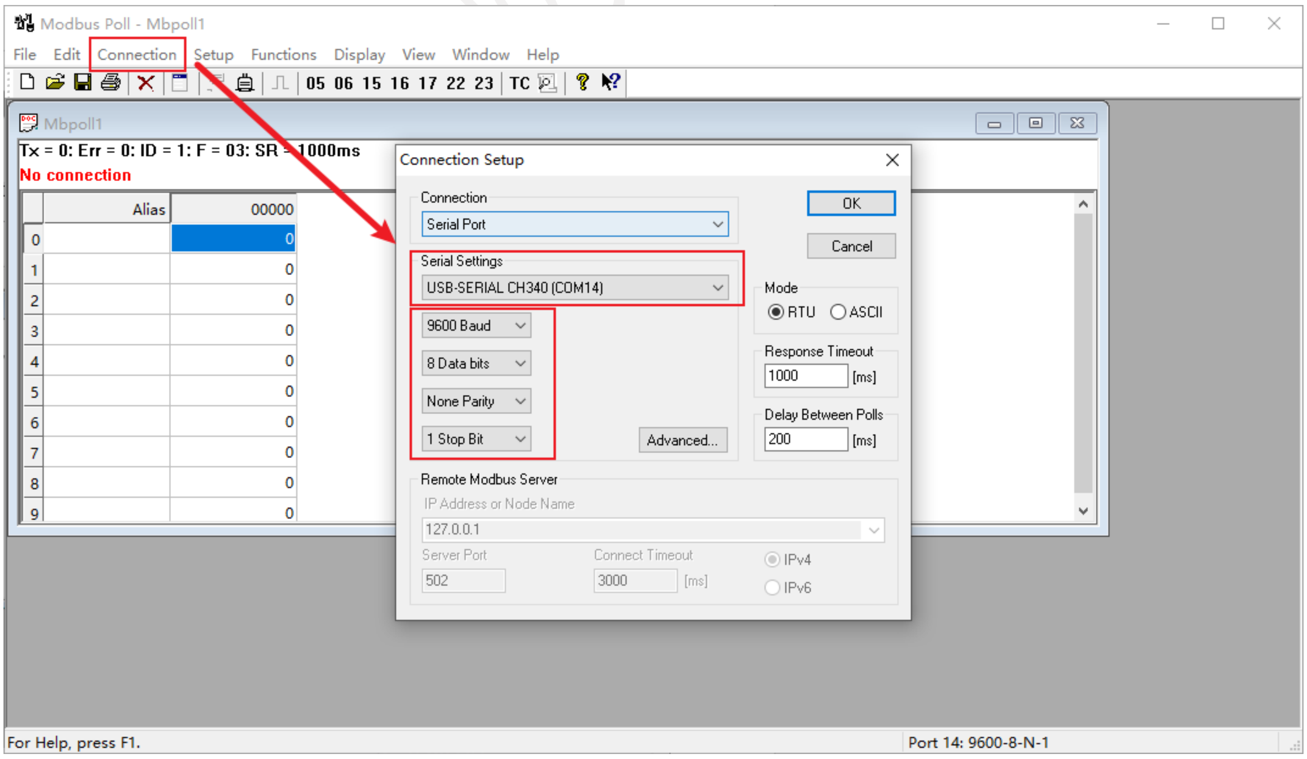

Modbus Poll - Mbpoll1
File Edit Connection Setup Functions Display View Window Help
05 06 15 16 17 22 23 TC
Mbpoll1

Tx = 0: Err = 0: ID = 1: F = 03: SR = 1000ms
No connection

	Alias	00000
0		0
1		0
2		0
3		0
4		0
5		0
6		0
7		0
8		0
9		0

For Help, press F1. Port 14: 9600-8-N-1

Communication parameters: baud rate 9600bps, 1 start bit, 8 data bits, no check, 1 stop bit.



Connection Setup

Connection: Serial Port

Serial Settings: USB-SERIAL CH340 (COM14)

9600 Baud
8 Data bits
None Parity
1 Stop Bit

Mode: [✓ RTU] ASCII

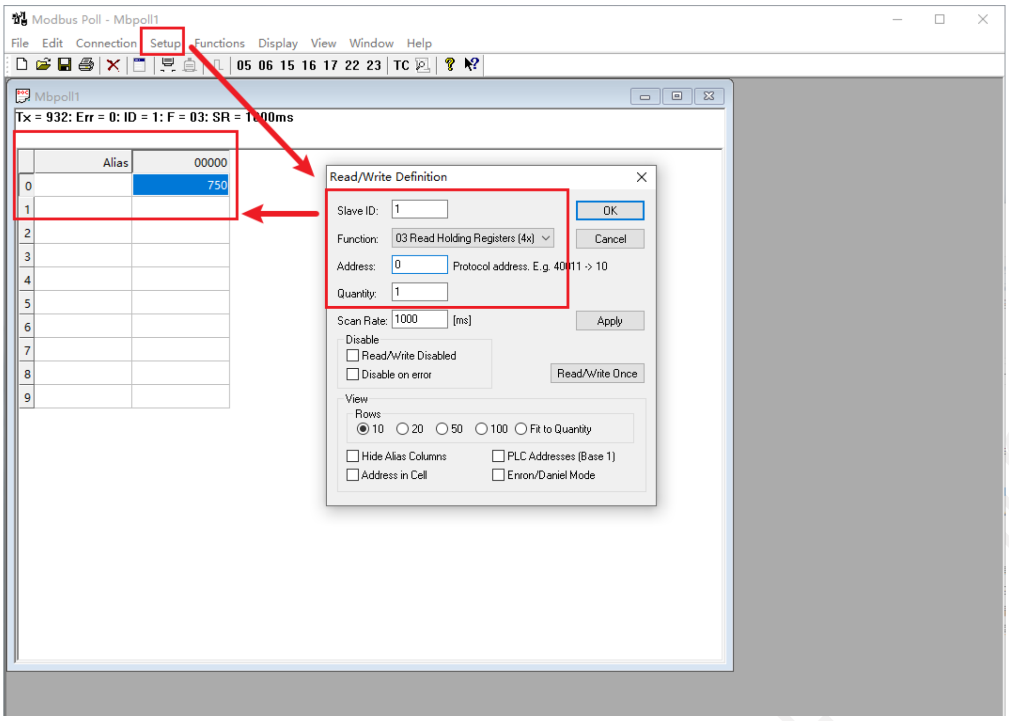
Response Timeout: 1000 [ms]

Delay Between Polls: 200 [ms]

Remote Modbus Server
IP Address or Node Name: 127.0.0.1
Server Port: 502
Connect Timeout: 3000 [ms]
IPv4
IPv6

For Help, press F1. Port 14: 9600-8-N-1

Configure the parameters of read registers 0x00: the slave address defaults to 1, function code 03, starting address 0, and number 1.



Tx = 932: Err = 0: ID = 1: F = 03: SR = 1000ms

	Alias	00000
0		750
1		
2		
3		
4		
5		
6		
7		
8		
9		

Slave ID: 1
Function: 03 Read Holding Registers (4x)
Address: 0 Protocol address. E.g. 40011 -> 10
Quantity: 1
Scan Rate: 1000 [ms]

Disable
☐ Read/Write Disabled
☐ Disable on error

View
Rows
[✓ 10] 20 50 100 Fit to Quantity
☐ Hide Alias Columns ☐ PLC Addresses (Base 1)
☐ Address in Cell ☐ Enron/Daniel Mode

5 SDI-12

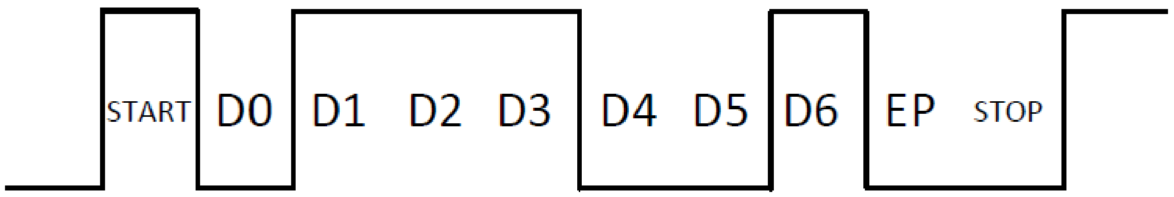
5.1 Introduction of SDI-12

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

Each sensor on the SDI-12 bus has a unique address, which can be set to ‘0’, ‘1’ ~ ‘9’, ‘A’ ~ ‘Z’, ‘A’ ~ ‘Z’. The SDI-12 address of the SenseCAP ORCH S4 defaults to ‘0’. The instructions supported by this sensor are shown in the next chapter, where each instruction conforms to the SDI-12 v1.4.

The sensor is powered by a DC power supply of 3.6~16V. After the sensor is powered on, it will go into sleep mode immediately and wait for the data acquisition equipment to give instructions. SDI-12 uses baud rate 9600bps, 1 start bit (high level), 7 data bits (high 0 and low 1, anti-logic), 1 even parity bit and 1 stop bit.

The sequence of each byte sent is shown in the following figure:



5.2 SDI-12 Commands

Query the device address		
Request	?!	? - Address wildcard ! - Instruction terminator
Response	a<CR><LF>	a - Address <CR><LF> - End of the response
Example	Request: ?! Response: 0<CR><LF>	The sensor at address '0' responded to the query

Query the device status		
Request	a!	a - Address ! - Instruction terminator
Response	a<CR><LF>	a - Address <CR><LF> - End of the response
Example	Request: 0! Response: 0<CR><LF>	Address '0' of device online

Query the device information		
Request	aI!	a - Address I – Identification ! - Instruction terminator
Response	allccccccccmmmmmmvvvxxx . . . xxx<CR><LF>	a - Address ll - 2 characters, SDI-12 protocol version, for example 14 represents v1.4 cccccccc - 8 characters, company name or product name mmmmmm - 6 characters, sensor type vvv - 3 characters, software version xxx . . . xx - Optional, up to 13 characters, can be used to send serial numbers, or other information <CR><LF> - End of the response
Example	Request: 0I! Response: 014SENSECAPSOLOCD1.0004A0040CO2	0I! - Command 0 - Address 14 – Hardware version 1.4 SENSECAP – Product brand SOLOCD -Product model 1.0 – Software version 1.0

		004A0040 - 8 characters, serial number CO2 - Output measurement, CO2

Modify device address		
Request	aAb!	a –The current address A - Address Change b – The new address ! - Instruction terminator
Response	b<CR><LF>	b –The new address <CR><LF> - End of the response
Example	Request：0A1! Response：1<CR><LF>	The address 0 was changed to 1. The response received was modified successfully

Start measuring		
Request	aM! 或 aMC!	a - Address M - Measure C - CRC ! - Instruction terminator
Response	atttn<CR><LF>	a - Address ttt - 3 characters, measurement end time, unit: second n - 1 character, the number of measurements to be output <CR><LF> - End of the response When the data logger sends ‘aMC!’ , the response returns data with a CRC.
Example	Request：0M! Response：00281<CR><LF>	0M! - Command 0 - Address 0028 - The measurement is completed after a maximum of 28 seconds 1 - One measurement will be output
Service request	a<CR><LF>	a - Address When the data logger sends a instruction, the sensor immediately responds to ' atttn<CR><LF>', and

		when the measurement is finished, the service request will be replied to inform the data logger that the data can be collected. The data collector should not request another sensor between the time the aM! is sent and the time it receives service request, unless the measurement is interrupted with a break (the measurement value will not be updated).

Read measured value		
Request	aD0!	a - Address D0 - Data（D1 . . . D9 and so on) ! - Instruction terminator
Response	a<values><CR><LF> or a<values><CRC><CR><LF>	a - Address values - as follows <CRC> - 3 characters, CRC, response to aMC! carry <CR><LF> - End of the response values： <Temperature><Humidity><Barometric Pressure><Light Intensity>， The order and number are shown in the 'aI!' , in which the measured value is composed of the following parts: <symbol><integer>[.<decimal>]
Example	Request：0D0! Response：0+450	0D0! - Command 0 - Address +450 – CO2, ppm

5.3 Precautions for the use of SDI-12

(1) Multiple sensors can be mounted on the sdi-12 bus, but the state maintenance of sensors should be paid attention to and the failed sensors should be detected in time, because the failure of one sensor may affect the normal work of the whole bus, even if other sensors are normal.

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