# 7 SMART SENSOR MODULE

Description

This 7 Smart Sensor Module consists of a data collection and processing PCB assembly, a SemeaTech 7-Series electrochemical (EC) sensor, and a metal enclosure. The PCB assembly in the module collects the data from the gas sensor output and then processes it with amplification, sampling, filtration, and compensation through a built-in MCU to deliver stable and accurate digital output reflecting the actual target gas concentration. Any of SemeaTech 7-series EC sensors and 4-electrode EC sensors can be used to form a 7 Smart Sensor Module to detect the target gas with the digital output. This module provides the convenience and friendly user experience for users to quickly integrate gas sensors into their existing systems for a variety of gas detection applications.

- Unified electrical interface, mechanical dimensions and communication protocol;
- Working with all SemeaTech 7-series EC sensors and 4-electrode EC sensors;
- Built-in temperature and humidity sensors for better environmental independence;
- Multiple output interfaces, including USART, and I^2C;
- Equipped with metal case (optional), it can protect the internal circuit from dust and water;
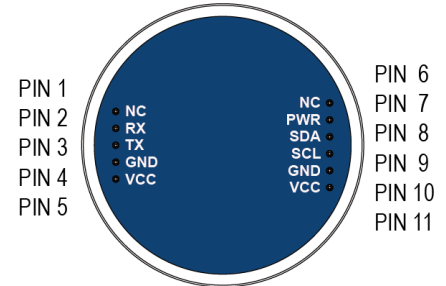- Programmable I^2C address for user friendly interfaces.

Product Dimensions

Top view　　　　　Side view　　　　　Bottom view
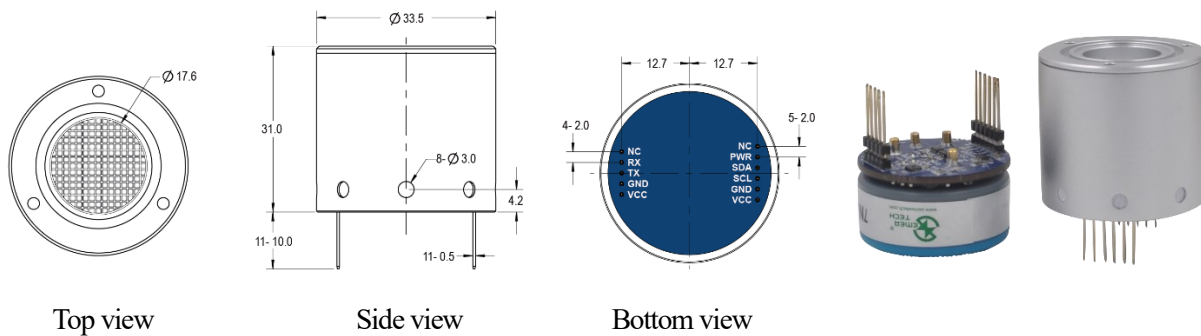
Specifications

Design for	SemeaTech 7-Series EC Sensors	Installation	11 pins, plug-in
Detection principle	Electrochemical	Operating temperature	-20°C ~ +55°C
Detection range	Refer to sensor data sheet	Operating humidity	0% ~90%RH Non-condensing
Detection resolution	Refer to sensor data sheet	Operating pressure	1 ± 0.1 atm
Measurement error	Refer to sensor data sheet	Enclosure material	Aluminum alloy
Operating voltage	3.2 ~ 5.5 VDC	Dimensions	Φ 33.5 x 31 mm
Operating current	≤ 5 mA @ 5V	Weight	30 g
Output mode	USART (3.3V TTL) I²C (3.3V TTL)	Expected operating life	PCBA: 5 years Sensor: Refer to sensor data sheet

Pin Definitions

Pin	Function	Description
1	NC	Reserved pins (suspended)
2	RX	Serial port receiving
3	TX	Serial port sending
4	GND	0V-Ground
5	VCC	3.2 ~ 5.5 VDC
6	NC	Reserved pins (suspended)
7	PWR	Module power enable, Default high level
8	SDA	I2C signal SDA, Default high level
9	SCL	I2C signal SCL, Default high level
10	GND	0V-Ground
11	VCC	3.2 ~ 5.5 VDC

PIN 1
PIN 2
PIN 3
PIN 4
PIN 5

NC
RX
TX
GND
VCC

NC
PWR
SDA
SCL
GND
VCC

PIN 6
PIN 7
PIN 8
PIN 9
PIN 10
PIN 11

*Note: Two VCC signals are connected internally.

USART Communication Protocol

1. Settings

Start bit – 1   Data bit – 8   Stop bit – 1   Check bit – None   Baud rate –115,200 bps
Without special instructions, the response time is less than 100ms (please refer to the specific instructions for special circumstances). System cannot respond to other commands until the current command is answered.

2. Frame Format (The format of each communication frame)

Header	Device Code	Function Code	Starting address	Data Length	Data	Check bit **(MSB first)**
H	ID	F	A	N	D	CRC16

H   – 1Byte, fixed as 0x3A
ID   – 1Byte, defaults as 0x10 and can be customized by users
F   – 1Byte, for example (0x03)
A   – 2Byte, for example (0x0001)
N   – 1Byte, in two bytes, for example (0x02: 4 bytes)
D   – N * 2Byte, Big Endian for example (MSB LSB) is defined as signed short
CRC16   – 2Byte, using MODBUS_CRC16 checking algorithm (see Appendix 1 for details)

3. Command Description

3.1 Sensor type reading

Request from host device

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x01	0x0000	0x01	0x0000	0x82B0

Module response for correct data receiving

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

Addendum: Sensor type code (Decimal)

0,1: Not define   2: CO   3: O2   4: H2   5: CH4   6: None   7: CO2   8: O3   9: H2S

10: SO2   11: NH3   12: CL2   13: ETO   14: HCL   15: PH3   16: None   17: HCN

18: None   19: HF   20: None   21: NO   22: NO2   23: NOX   24: CLO2   25: None

26: None   27: None   28: None   29: None   30: None   31: THT   32: C2H2   33: C2H4

34: CH2O   35: None   36: None   37: None   38: None   39: CH3SH   40: C2H3CL

For example: 3A 10 01 0F 4C AD (HEX 0F = DEC 15, so the sensor is PH3 sensor)

3.2 Sensor data reading (unit: $\mu g/m^3$)

Request from host device

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x03	0x0000	0x02	0x0000	0x7352

 Module response for correct data receiving

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x03	0x0000	0x02	D	CRC16

D: Received data, 4Byte, Big Endian

For example: 3A 10 03 00 00 02 00 00 00 5E 25 35

Sensor value ($\mu g/m^3$): 00 00 00 5E   i.e. 94 $\mu g/m^3$

3.3 Sensor data reading (ppb)

Request from host device

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x03	0x0002	0x02	0x0000	0x72EA

Module response for correct data receiving

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x03	0x0002	0x02	D	CRC16

D: Received data, 4Byte, Big Endian

For example: 3A 10 03 00 02 02 00 00 00 4C A4 DA

Sensor value (ppb): 00 00 00 4C   i.e. 76 ppb

Note: The display accuracy is 1 ppb, and the specific accuracy varies according to different sensors.

3.4 Sensor temperature data reading (°C)

Request from host device

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x03	0x0004	0x01	0x0000	0x8262

Module response for correct data receiving

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x03	0x0004	0x01	D	CRC16

D: Received data, 2Byte, Big Endian and divided by 100 to get the temperature value

For example: (D = 0x0A3D = 2621 to get the temperature of 26.21°C)

3.5 Sensor humidity data reading (%RH)

Request from host device

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x03	0x0005	0x01	0x0000	0x839E

Module response for correct data receiving

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x03	0x0005	0x01	D	CRC16

D: Received data, 2Byte, Big Endian and divided by 10000 to get the percentage humidity value

For example: (0x14 89 = 5257 to get the humidity of 52.57%)

3.6 Multiple parameters reading (address 0000-0005)

Request from host device

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x03	0x0000	0x06	0x0000	0x3293

Module response for correct data receiving

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x03	0x0000	0x06	D	CRC16

D: Received data, 12Byte

Followed by (MSB first): sensor reading $\mu g/m^3$ 4Bytes;

Sensor reading ppb 4Bytes; Temperature 2Bytes; Humidity 2Bytes.

For example: 3A 10 03 00 00 06 00 00 00 8F 00 00 00 50 0A 70 16 11 35 96

Sensor value ($\mu g/m^3$): 00 00 00 8F;

Sensor value (ppb): 00 00 00 50;

Temperature: 0A 70; Humidity: 16 11

3.7 Check error response

Header	Device Code	Function Code	Data	Check
0x3A	0x10	0x08	0x00	CRC16

For example: 3A1008000AF9

3.8 Zero Calibration

Request from host device

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x07	0x0000	0x01	0x0000	0x82D6

Module response for correct data receiving

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x07	0x0000	0x01	D	CRC16

D: 2Byte data

For example: 3A 10 07 00 00 01 04 7A 01 F5

Caution: Please place the module in the pure air environment stabilizing for at least 5 minutes then send the zero calibration command.

3.9 Gas Calibration

Request from host device

Module response for correct data receiving

Header	Device Code	Function Code	Starting address	Data Length	Data	Check
0x3A	0x10	0x09	0x0000	0x01	D	0x82D6

D: 2Byte gas concentration data, Big Endian, Unit: ppm, the value should be lower than 255 ppm

For example: 3A 10 09 00 00 01 00 0A 03 FF

i.e.: D=0x000A   10 ppm gas is used for calibration

Module response for correct data receiving

Header	Device Code	Function Code	Data	Check
0x3A	0x10	0x09	0x00 (Success) 0x01 (Processing) 0x02 (Fail)	CRC16

For example: 3A 10 09 01 CA A9 (Calibration processing)

Caution: The calibration process for environmental detection is about 300 seconds, and for industry usage is about 60 seconds, please wait for the module to respond before starting the command

I²C Communication Protocol

1. I²C Interface

Parameter	Definition	State	Min	Max	Unit
f_{ck}	I2C clock frequency	Slave Mode		100	kHz
t_{su}	Data input setup time	Slave Mode	6.5		ns
t_h	Data input Holding Time	Slave Mode	15.5		ns

2. Slave Device Address

Slave device addresses can be customized by software tools
Default setting:

CO:	0	0	0	0	0	0	1	R/W
O3:	0	0	0	0	1	0	0	R/W
SO2:	0	0	0	0	1	0	1	R/W
NO2:	0	0	0	1	0	1	1	R/W

The R/W bit is also used as a common data bit. When reading/writing, use 1 or 0 perform 'OR' operation with original data (read as 1, write as 0).
See the figure below for details.

3. I²C Communication Protocol
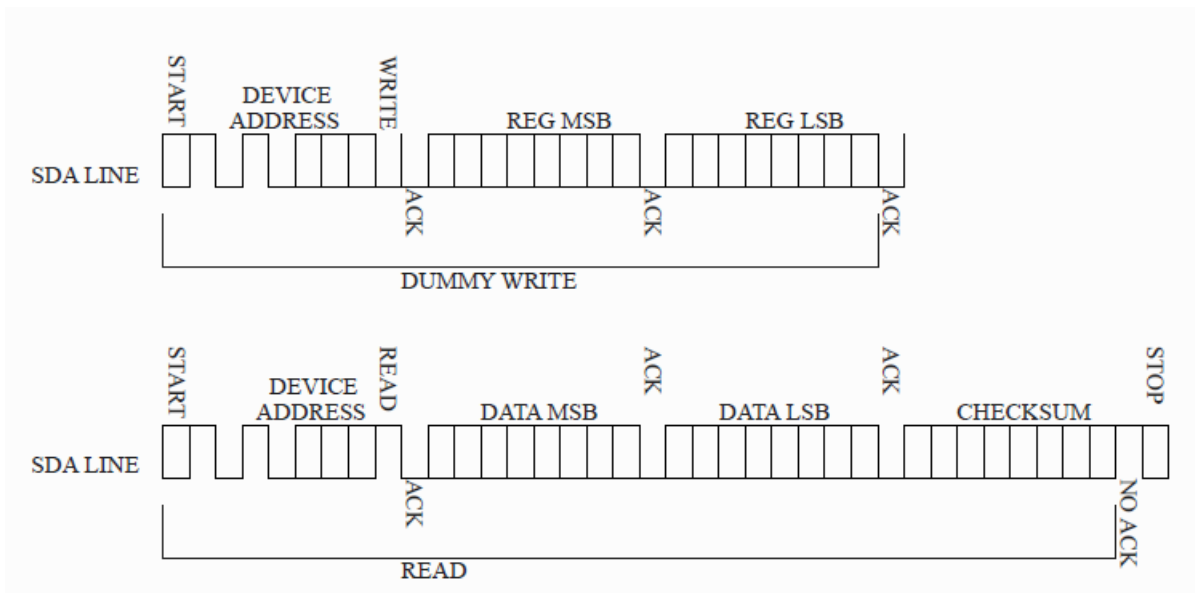
CHECKSUM is cumulative and retrieve check. See Appendix 2 for details.

4. Data Analysis

The REG parameters are shown below, which is the data address in the module.

REG	MSB	LSB
Sensor value ($\mu g/m^3$)	0x00	0x00
Sensor value (ppb)	0x00	0x01
Temperature	0x00	0x02
Humidity	0x00	0x03

Data examples:

DATA	MSB	LSB	CHECKSUM	Actual value	Remarks
Sensor value ($\mu g/m^3$)	0x0000	0x0005	0xFA	5 $\mu g/m^3$	Same with USART Data Conversion Method
Sensor value (ppb)	0x0000	0x0005	0xFA	5 ppb	
Temperature	0x0A	0x3D	0xB8	26.21°C	
Humidity	0x14	0x89	0x62	52.57%	

Caution: To receive 5Byte data when reading sensor value, and to receive 3Byte data when reading temperature and humidity

Warning!

1) This product does not have any intrinsic safety certification or explosion proof certification. Please do NOT use this product in any hazardous locations.

2) This product does not have reverse power protection and Electrostatic Discharge (ESD) protection. Please carefully verify the electrical polarity and make the ESD protection before each use or installation.

3) Please use a stable DC power supply for this gas sensor module. It is highly recommended to use a power supply with the output voltage fluctuation less than 1%.

Appendix 1: MODBUS CRC16 algorithm

```
unsigned short modbus_CRC16(unsigned char *ptr, unsigned char len)
{
unsigned short wcrc=0XFFFF; //
int i=0, j=0;
for (i=0; i<len; i++)
        {
                wcrc^=*prt++;
                for ( j=0; j<8; j++)
                    {
                            if (wcrc&0X0001)
                              {
                                 wcrc=wcrc>>1^0XA001;
                              }
                            else
                              {
                                 wcrc>>=1;
                              }
                    }
        }
    return wcrc<<8| wcrc>>8; //little endian (LSB fist)
}
```

In CRC calculation, only 8 data bits, start bit and stop bit are used. If there are parity bit, including this bit, they are not involved in CRC calculation.

The CRC calculation method:

1. Load a 16 bit register with the value of 0 xffff, which is CRC register.
2. The XOR of the first 8-bit binary data (the first byte of communication information frame) and the 16 bit CRC register is still stored in the CRC register.
3. Move the contents of CRC register one bit to the right, fill the highest bit with 0, and detect whether the moved out bit is 0 or 1.
4. If the move out bit is zero, repeat the third step (move one bit to the right again); If the shift out bit is 1, the CRC register XORs with 0xa001.
5. Repeat steps 3 and 4 until the right shift is 8 times, so that the entire 8-bit data is processed.
6. Repeat steps 2 and 5 to process the next byte of the communication information frame
7. After all the bytes of the communication information frame are calculated according to the above steps, the high and low bytes of the 16 bit CRC register are exchanged
8. Finally, the content of CRC register is CRC check code.
9. For example, a command 3A 10 09 00 00 01 00 32 get the wcrc value of 2D 02 through the above program. As the CRC code, 0x022D shall be filled in the CRC16 check code bit with the principle of high byte first. In this way, we get the calibration command: 3A 10 09 00 00 01 00 32 02 2D.

Appendix 2: CHECKSUM Accumulation and Verification

```c
unsigned char CheckSum(unsigned char *buf, unsigned char len) //return CheckSum value
{
    unit8_t i, ret = 0;
    for (i=0; i<len; i++)
    {
        ret +=*(buf++);
    }
    ret = ~ret;
    return ret;
}
```

The check sum method:

Sender: accumulate the data to get a sum, and reverse the sum to get our check value. Then send the data with the check value to the receiver.

For example:

Sender: to send 0xA8 and 0x50, we use the unsigned char (8 bits) to save the accumulated sum, i.e. 0xf8 (0b11111000), and get the check sum of 0x07 (0b00000111). Then send these three data out.