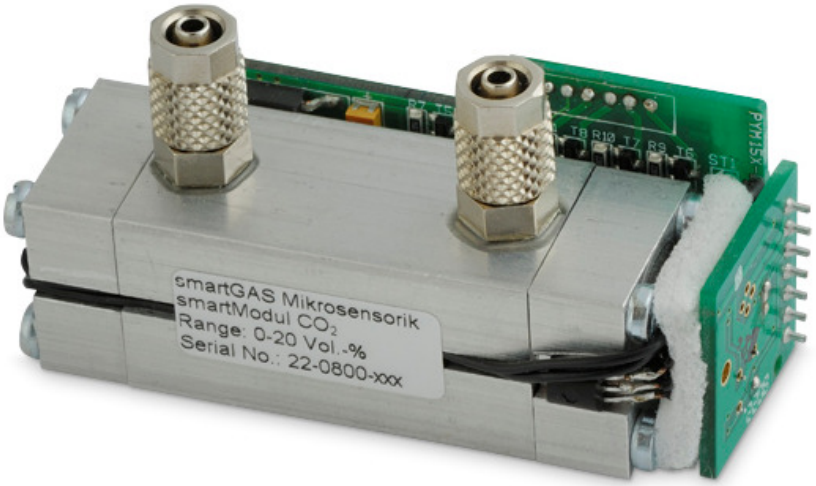
smart*MODUL*^{FLOW}

Module- and communication manual



smartGAS Mikrosensorik GmbH ▪ Kreuzenstrasse 98 ▪ 74076 Heilbronn ▪ Phone.: +49 (0)7131 / 797553-0
Fax.: +49 (0)7131 / 797553-10 ▪ mail@smartgas.eu ▪ www.smartgas.eu

Contents

1. General

The smart*MODUL*^{FLOW} is a self-contained gas measurement cell with a convenient data interface, making it easy to integrate into existing measurement and control systems.
Based on the physical measurement of infrared absorption, the device is not only highly selective but also provides high levels of accuracy and reliability when measuring gas concentration.
The compact construction and low maintenance requirements make it ideal for use even under very difficult conditions.

2. Measuring cell with gas line

The smart*MODUL*^{FLOW} housing is made of aluminium to protect the sensor from mechanical damage and is with fitted with gas line connectors (inlet and outlet) to allow perfusion of the smart*MODUL*^{FLOW} (see Figure 1).
Tubing with an internal diameter of 3 mm and external diameter of 5 mm is needed to connect up to the measurement cell. Ensure that tubing is securely attached to the inlet and outlet connectors.

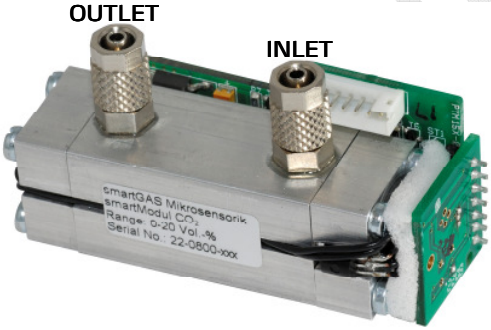


Figure 1: Measurement cell with gas inlet and outlet

Adhere to the directional designation of INLET and OUTLET; switching the direction of gas flow through the cell could lead to erroneous results.

NOTE:

Ensure the correct type of tubing is used. In some applications, corrosive gases occur and could cause problems with the tubing material.
Do not perfuse the smartMODUL^{FLOW} *at a gas flow rate of greater than 1l per minute! Any type of modification on the smartMODUL*^{FLOW} *is prohibited. In case of violation smartGAS does not take over any guarantee or liability.*

3. Assignment and properties of the connecting pins on smart*MODUL*^{FLOW}

There are two possible pin combinations for the smart*MODUL*^{FLOW}: a 4-pin version (V+, GND, COM, OUT 1) and a 7-pin version (V+, GND, OUT 1, OUT 2, OUT 3, OUT 4).
The following section describes the 7-pin version, as the first 4 pins are connected in exactly the same configuration as in the 4-pin version.

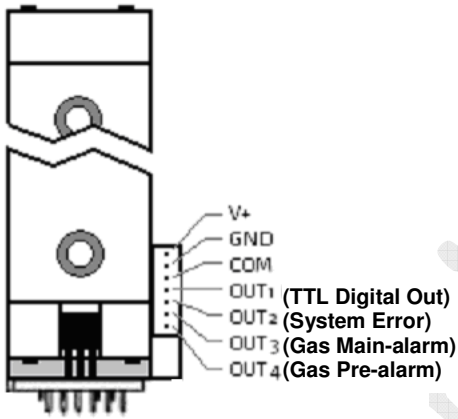


Figure 2: Pin connections in smart*MODUL*^{FLOW}

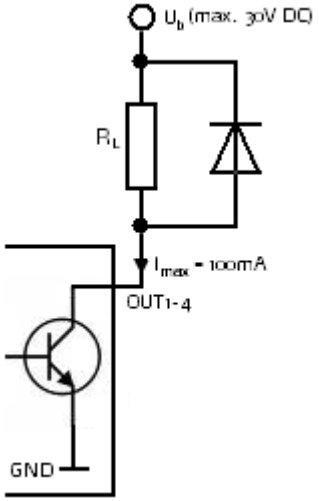


Figure 3: Loaded with a flyback diode

Pins:

V+ → Power supply 5.0V ±5% or 6.0V ±5% DC (see type label)

GND → Ground

COM → Modbus-Open-Protocol (ASCII). This bi-directional pin is used for the Modbus-Open-Protocol. Individual registers can be written and read via this pin. It also serves as trigger for a device test, in that the pin is switched relative to ground/GND for >200ms.

OUT 1 → TTL Digital Out. (active = 0) Open Collector output. This is used for communication, for example, with µ-Controller via TTL signals. The individual states are described in detail in a later section.

OUT 2 → System Error (active = 0). Open Collector output for displaying System Errors, e.g. radiation source failure.

OUT 3 → Gas Main-alarm (active = 0). Open Collector output for main gas alarm. The trigger threshold can be freely set using the software.

OUT 4 → Gas Pre-alarm (active = 0). Open Collector output for gas pre-alarm display. The trigger threshold can be freely set using the software.

Important: *Pins OUT 1 – 4 are configured as **open collector connections**. They switch relative to ground/earth and can be loaded to a maximum of 100 mA. A voltage of 30 V may not be exceeded. When connecting to an inductive load, it is important to use a flyback diode (Figure 3).*

4. Reading out the smart*MODUL*^{FLOW} using Pin OUT 1 (TTL Digital Out)

Displaying the smart*MODUL*^{FLOW} TTL signals

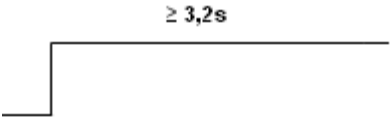
The smart*MODUL*^{FLOW} can display up to 7 different states via Pin OUT 1 (TTL Digital Out), as described below (active = 0, inactive = 1):

- All OK 3.2s inactive
- Gas Pre-Alarm 0.8s active / 2.4s inactive
- Gas Main-Alarm 1.2s active / 2.0s inactive
- Inc – Alarm: 1.6s active / 1.6s inactive (optionally set)
- Ex – Alarm: 2.0s active / 1.2s inactive (optionally set)
- Tox – Alarm 2.4s active / 0.8s inactive (optionally set)
- System Error 3.2s active

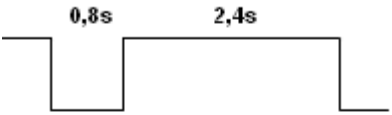
As shown above, the different alarm states can easily be displayed. Switch status is active = 0. The alarms Inc., Ex. and Tox. are optionally set alarm thresholds and not included in the standard smart*MODUL*^{FLOW} set up.

To display the TTL signals on an oscilloscope, a pull resistance of 2kΩ needs to be patched to V+.
The signals are then displayed as a voltage trace with amplitude V+:

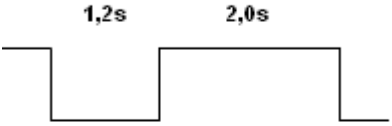
All OK



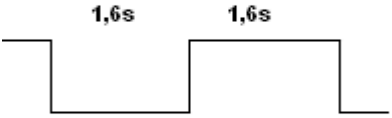
Gas Pre-Alarm



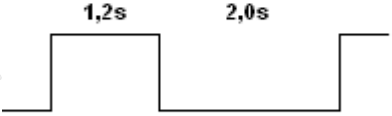
Main Gas Alarm



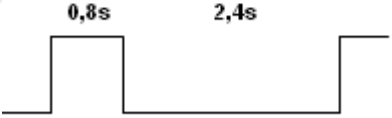
Inc. Alarm
(Optional)



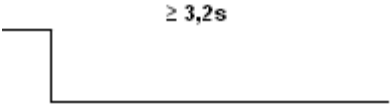
Ex. Alarm
(Optional)



Tox. Alarm
(Optional)



System Error



5. Hardware implementation of the smart*MODUL*^{FLOW}

This section describes connecting the smart*MODUL*^{FLOW} to a PC RS232 interface or µ-controller. The Modbus Protocol and individual registers of the smart*MODUL*^{FLOW} are explained in greater detail in later sections of the manual.

5.1 Connecting to a RS232 PC interface

There are two options for reading out data from the smart*MODUL*^{FLOW} using a PC RS232.
The first option uses a level converter and the second requires a little additional circuitry to enable bi-directional data transfer.

5.1.1 Connecting to a level converter

If a level converter is used the interface uses TTL levels for communication.
Since the send and receive paths in the smart*MODUL*^{FLOW} are combined, then the corresponding send and receive paths in the level converter also require combining. This is probably most easily achieved by using a 680 Ω resistance inserted in the send path of the level controller.
The RX line should then be used as common line to the module.

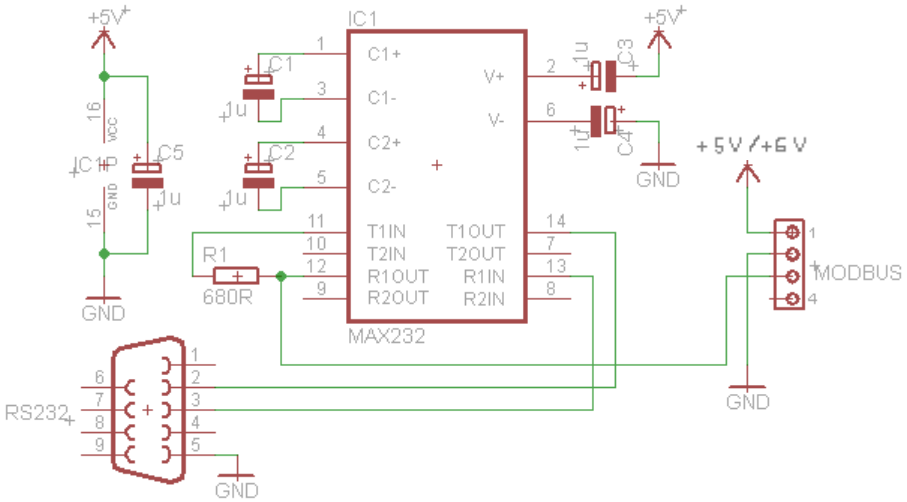


Figure 4. Connection to a MAX232 serial level converter

5.1.2 Connection to additional switching circuitry

The circuitry shown in Figure 5 is a type of RS232 interface that does not work with level converters but directly with the TTL signals. Nevertheless, it is possible to connect the signal output directly to the PC.
The send and receive paths are combined in a single path. This is possible since communication is exclusively initialized by the master.
If the smart*MODUL*^{FLOW} connected is not directly addressed, it remains on receive.
WARNING: The PC also receives signals it sends (echo) and this should be taken into consideration setting the system up.

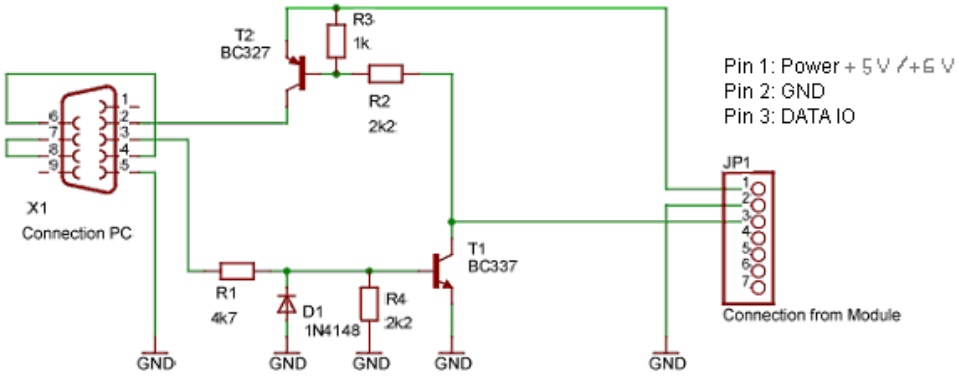


Figure 5. Hardware for communication via an RS232 interface.

WARNING: Pin 1 is connected to the power supply voltage!!! (see type label)

RS232 PC settings

Baud rate:	2400
Data bits:	7
Stop bits:	1
Parity:	Even
Timeout:	1000ms
Retries:	3

TIP: *In some cases it might be necessary to increase timeout time to 1.5s.*

5.2 Connection to a µ Controller

As an alternative to communication via a PC RS232 interface, it is also possible to connect the smart*MODUL*^{FLOW} directly to a µ controller.

5.2.1 Hardware UART (**U**niversal **A**synchronous **R**eceiver **T**ransmitter)

It is probably easiest to use available UART hardware and combine send and receive paths. As the receive path is usually high impedance, this should pose no serious problems. However, the send path must be switchable.
If the controller in circuit is programmed and the port pins adopt a non-defined function status then a protective resistor for restricting the current should be inserted in the send path.

The RS232 interface hardware has a buffer to which transmitted data it written. If the required data has been written, a flag is raised in the UART status register of the µ controller. In the example in section 9.1 a query is made of this flag so that reading of the data starts at the right time.

5.2.2 Software UART

Software RS232 has no buffer and needs to be directly read. A further problem is due to the kbhit() command, which checks to see if characters are on the received path. Software implementation of the interface requires several keystrokes for this command, which can lead to error.

To realize a software UART it is also possible to connect the smart*MODUL*^{FLOW} to a pin of the µ controller. This is a serial half-duplex connection, described in greater detail in Section 7.
The example in Section 9.1 describes how to detect this error.

RS232 settings for µ-Controller

Baud rate	2400
Data bits	7
Stop bits	1
Parity	Even
Timeout	1000ms
Retries	3

6. smart*MODUL*^{FLOW} register assignments

The sequence of registers in the current version is listed below; as they appear in the Host SW. Section 6.1 has examples of reading registers.

0x00C0 *(data type: number)*	Modbus_address	Current Modbus status of smart*MODUL*^{FLOW}. The addresses can be written and read. After an address has been changed, subsequent communication with the smart*MODUL*^{FLOW} is only possible via this address. Normally the Modbus address is equal to the last two digits from the serial number and has to be converted into hexidecimal (See page 16 "Address").
0x0080 – 0x0083 *(data type: string)*	DeviceType	The type of device connected. Read only.
0x0084 – 0x0085 *(data type: string)*	SoftwareVersion	Software version of device connected. Read only.
0x0086 – 0x0089 *(data type: serial number)*	SerialNr	Serial number of device connected. Read only.
0x0005 *(data type: signed number)*	MOD	Assumed value for internal concentration calculation. Read only. **Not equal to the concentration!**
0x000A *(data type: signed number)*	Concentration	Concentration is stored in this register as a numerical value. Depending on the smart*MODUL*^{FLOW} type a factor is still required for the calculation, found in the QS certificate supplied with every smart*MODUL*^{FLOW}.
0x0003 *(data type: signed number)*	T_module (0.1x°C)	Internal sensor temperature, as reference point for temperature correction. Read only.

0x0045 *(data type:*	Alarm_Level *signed number)*	Provides the threshold trigger value for the main gas alarm. Through the pressure admission with gas, this selector shaft can be adjusted. This modulation value is reduced by one, entered in this register and can be freely set by the user.
0x0044 *(data type:*	Warn_Level *signed number)*	Provides the threshold trigger value for the gas pre-alarm. Through the pressure admission with gas, this selector shaft can be adjusted. This modulation value is reduced by one, entered in this register and can be freely set by the user.
0x0047 *(data type:*	IR_4tagneu *number)*	Register to save the intensity of the measuring during zero point. (See page 21). Can be read and written.
0x0009 *(data type:*	Statusflags *number)*	Status flags indicate the states the module can adopt. Read only.

Individual flags, read from right to left, mean:

Flag 0:	Testflag,	value „1" with device test
Flag 1:	Warmup,	value „1" approx. 10s after start
Flag 2:	Syserr,	value „1" System Error
Flag 3:	Alarm,	value „1" with main gas alarm
Flag 4:	Warn,	value „1" with gas pre-alarm
Flag 5:	Startup,	value „1" in the start-up phase (less than 2 minutes)
Flag 6:	Korr,	value „1" when smart*MODUL*^{FLOW} is temperature-compensated
Flag 7:	mw_ok,	value „1" when the zero point was set

Flags 6 (Korr) and 7 (mw_ok) are internal flags, set for each process smart*MODUL*^{FLOW} during production.
They also have a quality control role and are set to "1" when the smart*MODUL*^{FLOW} is temperature–compensated and has been calibrated.

6.1 Examples of read offs from registers

All the following refer to a smart*MODUL* with address 160.

- **Read off "Device Type" register→ 0x0080 – 0x0083**

Send the following string:

Start	Adr.160	read	Start register	Register no.	checksum
:	A0	03	00 80	00 04	A0

Reply:

Start	Adr.160	read	no. of bytes	S	M	-	C	O	2			checksum
:	A0	03	08	53	4D	2D	43	4F	32	20	20	64

Data is transferred as characters and can be converted using an ASCII table.

- **Read off "Serial No." register→ 0x0086 – 0x0089:**

Send the following string:

Start	Adr.160	read	Start register	Register no.	checksum
:	A0	03	00 86	00 04	9A

Reply:

Start	Adr.160	read	no. of bytes	2	0	08	00	410		checksum
:	A0	03	08	32	30	08	00	019A	0000	99

The first two bytes are transferred as characters and can be converted using an ASCII table.
The third and fourth bytes are transferred as hexadecimal values and each yields a two-decimal place number.
The fifth and sixth bytes are summed as a hexadecimal value and produce a three-decimal place number.

- **Read off "Status flags" register→ 0x0009:**

Send the following string:

Start	Adr.160	read	Start register	register no.	checksum
:	A0	03	00 09	00 01	A2

Reply:

Start	Adr.160	read	no. of bytes	11000000	checksum
:	A0	03	02	00C0	F7

The two data bytes are summed and transferred as hexadecimal value. If this value is converted to binary number then the flags raised can be determined.

- **Read off "Software Version" register→ 0x0084 – 0x0086:**

Send the following string:

Start	Adr.16o	read	Start register	register no.	checksum
:	A0	03	00 84	00 02	9E

Reply:

Start	Adr.16o	read	no. of bytes	3	.	3	0	checksum
:	A0	03	04	33	2E	33	30	22

The data is transferred as characters and can be converted using an ASCII table.

- **Read off "Concentration" register→ 0x000A:**

Send the following string:

Start	Adr.16o	read	Start register	register no.	checksum
:	A0	03	00 0A	00 01	9A

Reply:

Start	Adr.16o	read	no.of bytes	456	checksum
:	A0	03	02	01C8	EE

The two data bytes are summed and transferred as a hexadecimal value. If this value is converted to a decimal number, the concentration can be determined (here - 456ppm).
If necessary the correction factor (to be found on the QM-document) has to be used.

7. Communication with the smart*MODUL*^{FLOW} via Modbus Open Protocol

Reading off TTL signals provides access to a small fraction of the information logged by the smart*MODUL*^{FLOW}.
Since smart*MODUL*^{FLOW} has a large amount of data potentially available it makes sense to use a BUS protocol.

The Modbus Protocol basically works on the master/slave principle. The master (PC or µ controller) sends a request to the slave (smart*MODUL*^{FLOW}), which in turn answers. The duration of this phase, until all data is received, depends on how many registers need to be read. As a rule, the smart*MODUL*^{FLOW} reacts to the request within **100ms**. The character string is sent directly, without reply pause. The slave does not send any data without a request.
The request is always first interpreted after dispatch by CRLF.

***WARNING**: Some programmes automatically send the CRLF; with most conventional programmes this needs to be tagged onto the string manually! The CRLF may only be sent once!*

The smart*MODUL*^{FLOW} sends no reply if it receives an incomplete request. This is also the case when one or more registers are absent from a register set (section request) An adapted form of the Modbus Open Protocol is used for smart*MODUL*^{FLOW}, differing from the standard version in that one path is used for send and receive. This ASCII protocol uses a serial half-duplex connection.

Datagram structure

The following section describes how a request data string to smart*MODUL*^{FLOW} is constructed. The example below shows the current modulation read off from a smart*MODUL*^{FLOW}, with address 160.

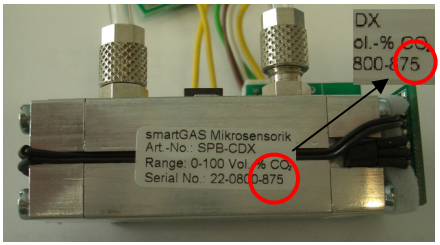
Example string looks as follows:
:A003000500001A6

Start	Address	Ctrl Com.	Data	checksum LRC
1 character	2 characters	2 characters	0-100 * 2 characters	2 characters
:	A0	03	00 05 00 01	A6

***NOTE:** Addresses, control commands and data are prefixed with "0x" and the actual address / commands as "nn". The "0x" merely indicates that the data is hexadecimal, but since the Modbus Protocol ASCII is defined as hexadecimal, this information is superfluous and only the address or command is transferred. The string contains no "0x" and "0x05" becomes "05".*

Start:
As a rule datastrings start with a colon":", irrespective of whether they are requests or replies.

Address
This defines to which device address the string is assigned. As standard, the device address is the last two digits of the *MODUL*^{FLOW} serial number as delivered. How to use and to convert it is shown in the example.



Device address = 75 decimal → 4B hexadecimal
Device address for Modbus ASCII communication is 4B.
In case that the serial number ends with „00" the address is 64 hexidecimal always.

To search for unknown Modbus addresses the sensors have to be connected at first. Now any register (e.g. concentration) can be requested from all module addresses (1-255) with a timeout of one second. A module with the correct address responds by sending a reply. This reply includes the module address so that at the end of the search cycle it is possible to see by processing the reply which module addresses are currently connected to the bus system.

Control commands
The control commands indicate what needs to be done with the aforesaid address.
Basically the *MODUL*^{FLOW} distinguishes between:

"**Read from register→0x0003**"

"**Write into register→0x0006**"

The command in the example shown here is "Read Register" (0x0003→03)

Data

The register number is sent in data as a parameter.
In the example here it is:

"**Start Address High (0x0000→00) / Low (0x0005→05)"**

and

"Number Register High (0x0000→00) / Low (0x0001→01)"

The Start Address High" and "Start Address Low" indicate to which register address the control command is directed; in this case, address 0005→0x0005 "MOD".
"Number Register High" and "Number Register Low" state how many registers beginning with the start address should be read. Should 10 Registers be read, then 0010 needs to be entered. In the example registers 05 to 14 would be read out and transferred.

NOTE: *Data is transferred in its hexadecimal form! The number of bytes doubles after conversion to from ASCII to Hex.*

7.1 Calculating the checksum in C

The checksum calculates according to a LRC method (Longitudinal Redundancy Check) from all the bytes sent without CR and LF characters. The bytes are added and the sum subtracted from oxFF. A "1" is added to the result, making the LRC complete.
In the example shown here the value is "A6"

(Command: Read register 5 only→ Modulation MOD):

Data[0]:=':'; Data[1]:='A'; Data[2]:='0'; Data[3]:='0'; Data[4]:='3';
Data[5]:='0'; Data[6]:='0'; Data[7]:='0'; Data[8]:='5'; Data[9]:='0';
Data[10]:='0'; Data[11]:='0'; Data[12]:='1'
Laenge=12;

Note: *LRC and CRLF do not belong to the data. CRLF is not included in the LRC calculation!*

1. Lrc=0; *//(checksum set to 0)*

   For(i=1;i<Length;i++)
2. Lrc=Lrc+daten[i]; *//(All bytes transferred binary summed with overlap (8 Bit). Example: 200+200=400. With 8Bit only 256→ 144 written = Lrc.*

(In the example above the rest sum is 90.

3. Lrc=oxFF-Lrc; *//(Total (90) is subtracted from 255.)*

4. Lrc=Lrc+1; *//(255-90+1=166=A6 in Hex (checksum reply)*

Example of how to calculate the checksum:

Conversion table for ASCII values:

hex.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
dec.	48	49	50	51	52	53	54	55	56	57	65	66	67	68	69	70

Example: (read out MOD value from device address 160)**:**
Data string → **A00300050001**

Data string conversion:

	address		command		start register				number of register				sum
String Hex.	A	0	0	3	0	0	0	5	0	0	0	1	---
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	
ASCII Dec.	65	48	48	51	48	48	48	53	48	48	48	49	**602**

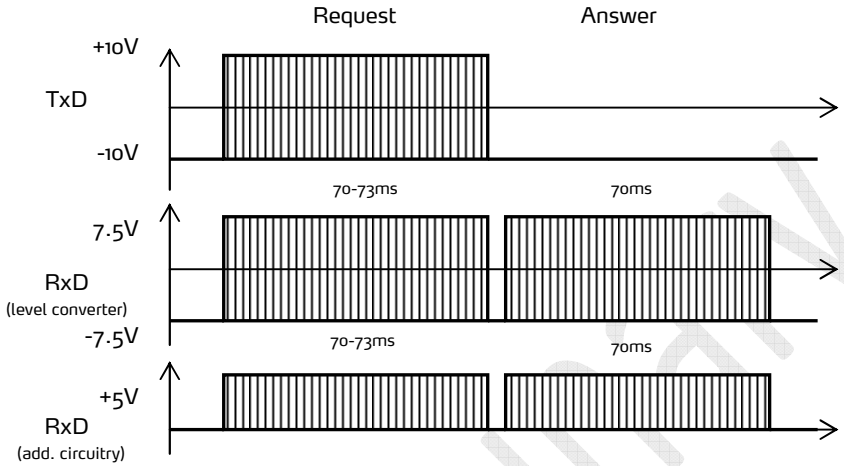
Sum as ASCII (dec): 602-256=346
346-256=**90** (rest)

Checksum: 255-90+1=**166 (ASCII dec.)** → **A6 (String hex.)**

After calculation of the checksum the complete data string would be:
Data string :A00300050001**A6**

The checksum is always transmitted with the data and recalculated by the recipient. Should a value in the data set become corrupt, then the checksum calculated by the recipient would be different from that sent. The data set in this case would be unusable.

Signal trace



A request string lasts between 70 – 73ms. This is transferred as an echo on the receive path. A short pause of max 100ms can occur. The module then replies. This depends on how many bytes need to be read off: if only one byte is read then the module reply lasts approx. 70ms. Reading off more bytes correspondingly increases the reply phase. The amplitude of the RxD line depends on the number of connection switches used. (see Section 5).

8. Zero point calibration of the smart*MODUL*^{FLOW} via Modbus Open Protocol

To calibrate the **zero point**, the smart*MODUL*^{FLOW} needs to be connected to the voltage supply and the Modbus ASCII Protocol. The sensor needs to be perfused with **zero gas**.
The register is then read as 0x0004 by the Modbus Protocol. The value in this register is analogous to the gas concentration. The following example describes reading off this register in a smart*MODUL*^{FLOW} with Modbus address 160:

Send the following command to the smart*MODUL*^{FLOW}:
:A003000400001A7

Start	Address 160	Read	Start register 0x0004	Register no.: 1	checksum
:	A0	03	00 04	00 01	A7

The following reply is dispatched (data content can vary):
:A003021F1AE1

Start	Address 160	Read	No. bytes: 2	Data: 7962	checksum
:	A0	03	02	1F 1A	E1

Now the register 0x0004 has content 0x**1F1A** (=7962).
If this value remains stable over period of time, then the measure gas concentration is stable. The zero point can now be set and register 0x0047 now needs to be written with the content of the previously read register (0x0004).

Send the following command to the smart*MODUL*^{FLOW}:
:A00600471**F1A**75

Start	Address 160	Write	Write to register: 0x0047	Data to write:0x1F1A	checksum
:	A0	06	00 47	1F 1A	75

If the command is correctly transferred then the same command is returned as reply.
Register 0x0047 can be checked to see if the values have been correctly written.

Send the following command to the smart*MODUL*^{FLOW}:
:A003004700001A0

Start	Address 160	Read	Start register 0x0047	Register no.	checksum
:	A0	03	00 47	00 01	A0

The following reply is sent:
:A003021F1AE1

Start	Address 160	Read	No. of bytes: 2	Data: 7962	checksum
:	A0	03	02	1F 1A	E1

Note: *Unless not else specified by the application, an inspection every 2 years is recommended.*

9. Examples of communication with the smart*MODUL*^{FLOW}

9.1 Communication via µ-controller software UART

In the example PIC 16F874 with 20 MHz is used and the checksum not computed.
The programme waits until the start character of the received string has been read at the interface. During this period the controller can undertake no other processes, an untenable situation for time-critical applications (USB connections).

```
//*** interface definition***
#use rs232(baud=2400,XMIT=PIN_D0,rcv=PIN_D1,STREAM=CO,bits=7,parity=E,errors)

void main(void) {                          // main function //
boolean d = 1;
boolean e = 1;
char c1;
int i;
while (TRUE) {
i = 0;
e = 1;
d = 1;
delay_ms(100);
fprintf(CO,":A0030005001A6\r\n");          // send request
//*** wait for start of string for send ***
while (e)
   {
   if(fgetc(CO) == ':') e=0;
   delay_us(2);
   }

while (d                                   // loop until LF appears
   {
   c1=fgetc(CO);                           // fetch character from string
   str[i++]=c1;                            //save character in string
   if(c1 == 10) d=0;
   }
auswerten();
}
}
```

9.2 Sending data via DELPHI

```
procedure modbus_send(str: string);
var
   lrci: byte;
   count: Integer;
   I, J, K: Integer;
begin
      if comactive=false then begin
            lrci:=0;
            count:=length(str);
            for i:=1 to count do begin
                  lrci:=lrci + ord(str[i]);
            end;
            lrci:=255-lrci;
            lrci:=lrci+1;
            logit('MOD-TX: '+str+IntToHex(lrci,2),1);
            str:=':'+str+IntToHex(lrci,2);
            sent:=str;
            form1.comport1.WriteStr(str+chr(13)+chr(10));
            form1.comtimer.enabled:=true;
            comactive:=true;
            end
      else logit('MODBUS Belegt',1);
      end;
modbus_send('A003000000oA');
/// received part
logit('RX:'+DateTimeToStr(Now)+'-->'+str,1);
if (str[1]<>':') then logit('No : at start',1);                    // Test : as Start
// LRC Testen
lrci:=0;
count:=length(str);
for i:=2 to count-2 do begin;
      lrci:=lrci + ord(str[i]);
end;
lrci:=255-lrci;
lrci:=lrci+1;
if (copy(str,count-1,2)<>IntToHex(lrci,2)) then logit('LRC FEHLER '+copy(str,count-1,2)+' <>
'+IntToHex(lrci,2),1);
```

Technical data

General features	
Measurement principle:	Non Dispersive Infra-Red (NDIR), dual wavelength
Measurement range:	dependent on model – see list
Gas supply:	by perfusion
Dimensions:	Length (model dependent) x 28 mm x 42 mm (L x W x H) [3]
Gas line connectors:	3 mm internal, 5mm outer diameter
Technical features @ 25°C, 1013 mbar gas pressure, 0.5 l/min constant gas flow	
Response time (t90):	Appr. 15 s (at 0.5 l/min) [3]
Resolution:	1 ppm to 0.01 Vol.% FS [1]
Accuracy:	≤ ±2 % FS [1]
Long term stability (zero):	≤ ±2 % FS [1] over 12 month period
Long term stability (span):	≤ ±2 % FS [1] over 12 month period
Repeatability:	≤ ±2 % FS [1]
Linearity error:	≤ ±1 % FS [1]
Lower detection limit:	≤ 1 % FS [1] (typically)
Operating temperature:	-10 °C to 40 °C
Storage temperature:	-20 °C to 60 °C
Humidity:	0 % to 95 % rel. humidity (not condensing)
Temp. dependence (zero):	≤ ±0.01 % FS [1] via °C
Temp. dependence (span):	≤ ±0.2 % FS [1] via °C
Air pressure:	950 to 1050 mbar
Pressure dependence (zero):	-
Pressure dependence (span):	0.1 % to 0.2 % via mbar [2]
Warm-up time:	< 2 minutes (start up time) < 30 minutes (full specification)
Flow rate:	0.2 - 1.5 l/min

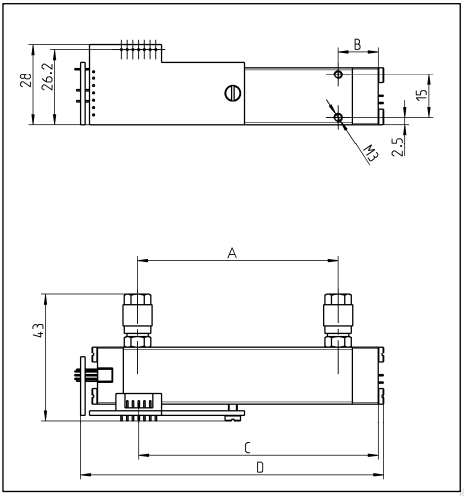
Communication	
Digital output signal:	Modbus ASCII via UART
Electrical data	
Supply voltage:	6 V DC ± 5 %
Supply current:	70 mA average, max. 140 mA
Power consumption:	< 1 Watt

[1]**FS=Full scale** | [2]**Dependent on the gas and the measurement range** | [3]**Dependent on model type**

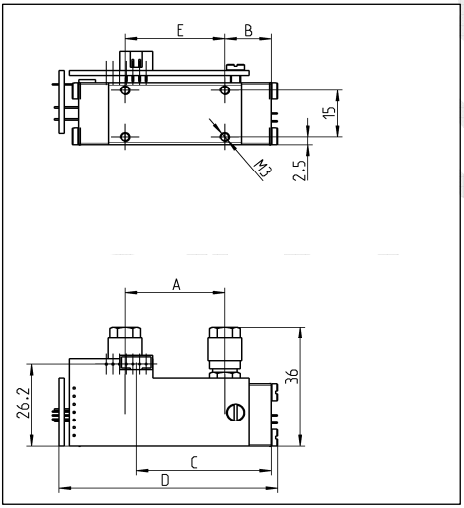
Please consult smartGAS Marketing for parts specified with other temperature and measurement ranges. At first initiation and depending on application and ambient conditions recalibration are recommended. Recurring cycles of recalibration recommended.

For more information, please visit www.smartGAS.eu or contact us at sales@smartgas.eu

Mechanical drawings



Item No.:	A	B	C	D
F1-010236-00000	70	14	84	106
F1-020146-00000				
F1-030246-00000				
F1-212505-00000				
F1-221206-00000				
F1-040446-00000				
F1-050176-00000				
F1-600503-00000				
F1-600105-00000				



Item No.:	A	B	C	D	E
F1-020108-00000	30	14	41	66	10
F1-212207-00000					
F1-212108-00000					
F1-221107-00000					
F1-221108-00000					
F1-040108-00000					
F1-050108-00000					

All dimensions in this manual are expressed in Millimetres (metric system).