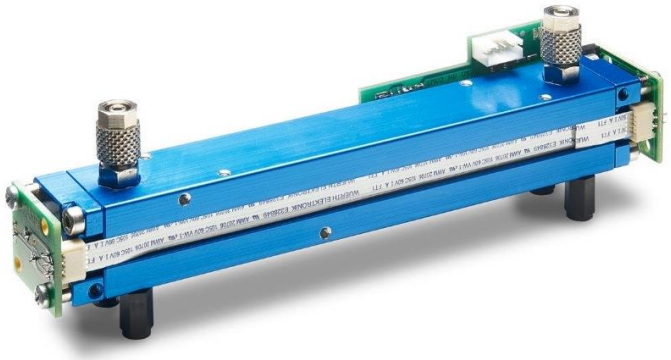
smartGAS.

FLOW^{EVO}

Module and Communication Description from Firmware Version 5.51



Contents

1 General ... 4

1.1 For your safety ... 4

1.2 Intended use ... 4

1.3 Implementation guidelines ... 5

1.4 Loss of warranty / liability / disclaimer ... 5

2 Measurement cell with hose connections ... 6

2.1 Hose connection / hose material ... 6

2.2 Gas flow ... 6

2.3 Mounting / installation site ... 6

3 Assignment and characteristics of the FLOW^{EVO} terminal pins ... 7

3.1 Current consumption ... 7

4 LED status display ... 8

5 Data interfaces ... 8

5.1 Function of the COM signal (communication) ... 8

5.2 Data exchange between master and FLOW^{EVO} (slave) ... 9

6 Modbus communication ... 9

6.1 Automatic detection of baud rate, framing format and Modbus dialect ... 10

6.2 Structure of Modbus data telegrams ... 10

6.3 Modbus communication device ... 11

6.4 Modbus slave ID ... 11

6.5 Modbus control commands ... 13

6.5.1 Control command 0x03 → Read Holding Register ... 13

6.5.2 Control command 0x06 → Write Single Register ... 14

6.6 Calculating the checksum ... 16

7 Register overview ... 17

7.1 Meaning of the individual bits in the status bit bar (SYS_Status): ... 18

7.2 Description of the unit code: ... 18

8 Information on start-up and operation ... 19

8.1 Self-test ... 19

8.2 Setting the zero point ... 19

8.3 Setting the end point ... 20

8.4 Calculating the correction value for the end point ... 20

8.5 Restoring the calibration parameters to factory settings ... 20

9 Annex 21
9.1 Mechanical dimensions [mm] 21
9.2 FLOW^{EVO} operation on a microcontroller 22
9.3 FLOW^{EVO} operation on a PC 23
10 Legal information 24

1 General

The FLOW^{EVO} is a further development the tried-and-tested smart*MODUL*^{FLOW}, which has been improved in many ways. Its convenient interfaces make it quick and easy to integrate into existing measuring and control systems.
The most important changes to the FLOW^{EVO} in comparison to the smart*MODUL*^{FLOW}

- Expanded operating voltage range of 3.3 V – 6.0 V DC (+/- 5%)
- Status display via 2 LEDs (red/green)
- Upgraded firmware, powerful processor
- Improved mechanical setup, more aesthetic design
- Modbus ASCII (standard and smartGAS-specific) and RTU protocols supported

The FLOW^{EVO} is based on the physical measurement method of infrared absorption and, in addition to its selectivity, it provides the best conditions for reliable and precise measurements.
Its compact design and the low maintenance effort make it ideal for use in difficult conditions.

1.1 For your safety

Meaning of warning signs

The following warning signs are used in this document to indicate the corresponding warning texts.

CAUTION!

Indicates a potential hazardous situation. If this is not avoided, injuries or damage to the product or environment may occur.
Also warns against improper use.

NOTE

Information on the use of the product

Before connecting and using the FLOW^{EVO}, ensure that you have fully ready and understood these instructions. Please contact our Service department if you have any questions or if anything is unclear. Warning signs indicate important information.
Keep these instructions in a safe place or give them to the device operator for safe keeping if necessary; if the device is sold, the instructions must be transferred to the purchaser. When installing and operating the device, you must follow the statutory requirements and guidelines that relate to this product!

1.2 Intended use

The FLOW^{EVO} is a gas measurement cell with independent measurement capabilities and is used to determine gas concentrations in accordance with its specifications. It is not suitable for any other measurement or testing purposes and must not be used in any other way.

CAUTION!

The sensor must not be operated in potentially explosive environments or under harsh conditions (e.g. high, condensing humidity, heavy air flow, in aggressive atmospheres or outdoors without a housing).

1.3 Implementation guidelines

- Dry sample gas with less than 5°C dew point – via electric cooler, silica cartridge or similar.
- Reliable particle filtration, e.g. coalescence filters or similar – needs to be checked and serviced frequently
- Stable gas sample flow rate between 0.1 … 1.0 l/min – pump without pressure fluctuation
- Frequent ZERO point check and adjustment – we recommend to use a small standard N2 bottle and a magnetic valve for SW-controlled automatic zero adjustment
- Frequent SPAN adjustment – requires appropriate test gas
- Before applying any form of adjustment, leave the sensor in operation for at least 30 minutes under stable environmental conditions
- Data communication via UART (F3 and B3 series) or RS485 (with connect interface) with Modbus RTU
- Stable sensor temperature insulated from housings or other materials - in the best case the environment or the sensor itself is heated to ~ 40°C

1.4 Loss of warranty / liability / disclaimer

CAUTION!

Opening the sensor as well as manipulating or damaging the device will invalidate the warranty!
The warranty may also be invalidated if aggressive chemicals are used, contamination occurs, liquids penetrate the device or the instructions in this module and communication description are not observed!
smartGAS Mikrosensorik GmbH assumes no liability for consequential loss, property damage or personal injury caused by failing to observe the the module and communication description.

2 Measurement cell with hose connections

The FLOW^{EVO} measuring cell is made of aluminium and gold-plated. It is equipped with hose connections which ensure that the measurement gas passes through the measurement process. The actual measurement cell is located between the gas inlet and gas outlet

2.1 Hose connection / hose material

Hoses must have an inside diameter of 3 mm and an outside diameter of 5 mm to connect to the measuring cell. Make sure that the hoses are firmly connected to the hose connections.
Please observe the direction of the gas flow, which is indicated by the labels “INLET” and “OUTLET”. Mixing up the gas flow would result in measurements that could significantly deviate from the factory calibration.

Ensure that hoses suitable for measurement are used. Certain applications can generate corrosive gases that could cause problems with the hose material.

2.2 Gas flow

The gas flow should be constant and between 0.1 l/min and 1.0 l/min. The gas must be dry and free from particles. Corresponding filters can be purchased from smartGAS.

2.3 Mounting / installation site

The FLOW^{EVO} is mounted using M3 screws with the four polyamide spacer bolts that are mounted on the underside of the cell. When screwing the sensor onto the mounting plate, make sure that no stress is applied during the mounting.
The smartGAS sensors allow for installation in various positions on the customer's devices. Since the calibration ex factory cannot cover every installation situation and ambient condition, the zero and end point need to be checked after installation and recalibrated if necessary. In any case we recommend a functional test of every device after final installation in the customer's application as part of commissioning.
When using different spacer bolts (than those mounted at the factory) or spacer sleeves, ensure a minimum clearance of 3 mm to the mounting plate.

NOTE

Do not use the other (free) threads in the sensor for mounting purposes.

3 Assignment and characteristics of the FLOW^{EVO} terminal pins

The grid dimension of the socket is 2 mm. The connector is **not** supplied as standard but can be ordered separately. (Designation: **4-pin JST connector, 2 mm contact spacing)**

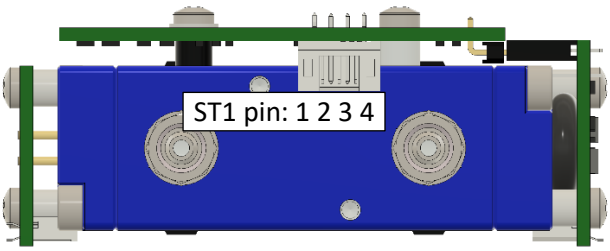


Figure 1: Position of the connectors

Connector ST1

Pin	Assignment
1	+Vcc 3.3 – 6.0 V DC (+/- 5%)
2	GND
3	COM
4	Do not connect

Table 1: ST1 pin assignment

3.1 Current consumption

The following table shows an overview of the current and power consumption. It is strongly recommended to only use adequately dimensioned and voltage-stabilized supply voltages in order to prevent malfunctions due to voltage dips.

Appropriate cable diameters must be used for long supply lines in order to avoid excessive voltage drops over the lines!

Supply voltage	Current consumption
3.3 V	400 mA
6.0 V	240 mA

Table 2: Voltage-dependent power consumption

NOTE

The power consumption can be briefly higher when the FLOW^{EVO} is switched on.

4 LED status display

Two LEDs (green/red) are located next to the connector strip. These show the current device status as per Table 3:

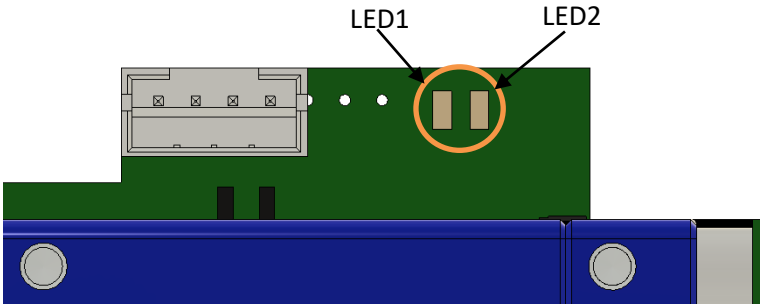


Figure 2: Position of the status LEDs

LD1 (green)	LD2 (red)	Device status
flashes	-	Start phase / self-test (approx. 40 seconds)
illuminates	-	Normal operation
-	flashes	Measurement overrange or underrange (OUT_OF_RANGE)
-	illuminates	Device error / contact service

Table 3: LED status display

NOTE

Measurement overrange or underrange (OUT_OF_RANGE) can be switched on or off.

5 Data interfaces

5.1 Function of the COM signal (communication)

The FLOW^{EVO} has a semi-duplex UART data interface that supplies and evaluates the non-inverted UART signals. The semi-duplex operation also means that only one communication signal (COM) is required. The level on the COM line is between 0 V and +3.3 V. It may therefore be necessary to include a level adjustment system depending on the communication partner (master) that is connected.

CAUTION!

The COM connection is designed as an open-collector connection with an internal pull-up resistor of 10 kOhm at 3.3 V. It is wired to GND and may be loaded with a maximum of 30 mA. Under no circumstances may the voltage exceed 7 V DC. The use of protective resistors, EMC filters, electrical isolation and other electrical or electronic measures may cause communication problems and must therefore be carefully considered by specialist personnel.

5.2 Data exchange between master and FLOW^{EVO} (slave)

Figure 3 shows a possible scenario between master and FLOW^{EVO} (= slave).
The following times refer to Modbus ASCII and a baud rate of 2400 Bd.

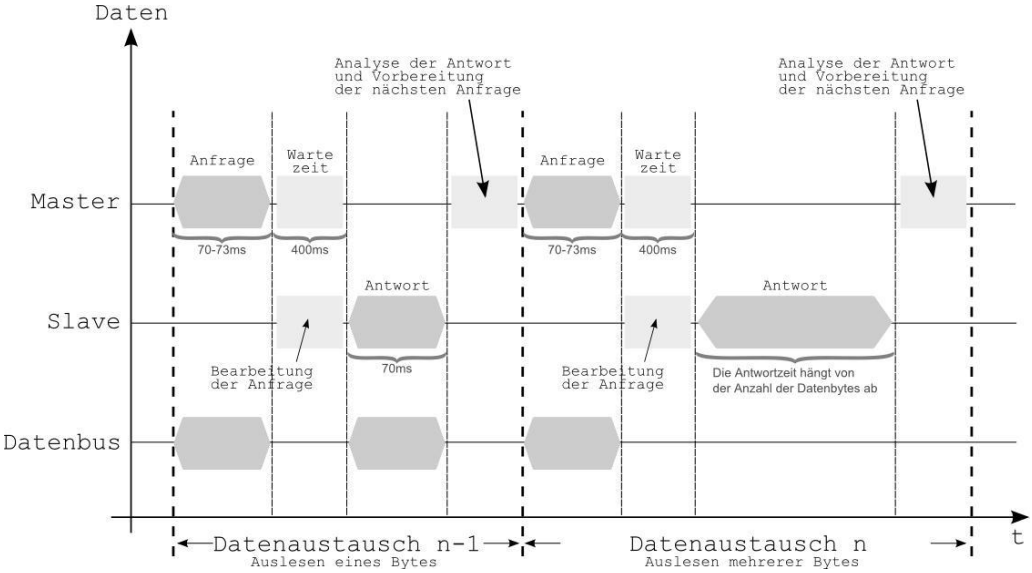


Figure 3: Time diagram – Data exchange between master and FLOW^{EVO} (slave)

The duration of a query string is 70 – 73 ms. A short pause may then follow (max. 400 ms). The module response then occurs. This depends on the number of bytes being read out. If only one byte is read out, the module response is around 70 ms. When multiple bytes are read out, the response phase is extended accordingly.

Basically, it can be said the FLOW^{EVO} sensor responds to a query within 400 ms. The character string is then sent immediately without a response pause.

CAUTION!

At higher baud rates (> 2400 Bd), significantly faster response times can be expected.

NOTE

The cable used for wiring should be twisted and shielded (twisted pair cable).

6 Modbus communication

The FLOW^{EVO} supports the Modbus protocol in ASCII and RTU mode thanks to its serial semi-duplex interface. In ASCII mode, in addition to the standard variant, there is a smartGAS-specific derivative that has a different checksum calculation.

In principle, Modbus communication functions on the basis of a query/response mechanism. The master sends the query to one of possibly several slaves (subscribers). Each connected subscriber therefore receives a subscriber address that is unique in the network. Only the subscriber that has found its address in the query from the master will respond.

The type of query is determined by a control command (function code). This can, for example, be about writing data or reading data to/from the subscriber. Depending on the control command, there is a data portion for both the query and the response.
Each query and each response must be clearly identified by its beginning and by its end. The use of a check field (=check word / CRC) is envisaged in the protocol to enable any possible communication errors to be detected. The Modbus derivatives implement this in different ways.

You can obtain detailed information about the Modbus protocol at www.modbus.org

6.1 Automatic detection of baud rate, framing format and Modbus dialect

The FLOW^{EVO} firmware is provided with automatic configuration detection. This means that the sensor automatically detects the baud rate, the framing as well as the Modbus dialect used when it is switched on for the first time and interacts on the bus line for the first time in the system.
The framing formats and Modbus baud rates listed in Table 4 harmonise with each another and can be freely combined among each other.

Framing formats and Modbus baud rates			
Data bits↓	**Parity↓**	**Stop bit↓**	**Baud rates ↓**
7	E	1	2400Bd
7	E	2	4800Bd
7	O	1	9600Bd
7	O	2	9200Bd
7	N	2	38400Bd
8	E	1	57600Bd
8	N	1	115200Bd
8	N	2	
8	O	1	

Table 4: Freely combinable framing formats / baud rates

NOTE

A framing format of 8 data bits must be used for the communication via Modbus RTU.

6.2 Structure of Modbus data telegrams

As previously mentioned elsewhere, smartGAS Mikrosensorik recommends creating a data telegram with Modbus RTU. It is also possible to create a data telegram with Modbus ASCII, but this is not explained here. The following tables show the basic structure of an RTU data telegram:

Dialect	Start	Slave ID	Function	Data	CRC	End
Modbus RTU		1 byte e.g.:0xA0	1 byte e.g.:0x03	0 to 1x252 bytes e.g.:0x00,0x05, 0x00,0x02	2 bytes e.g.:0xA4, 0xD3	
Communication:	Pause 3.5 character	0xA0	0x03	0x00,0x05,0x00,0x02	0xA4, 0xD3	Pause 3.5 characters

In RTU mode, each byte is transferred unchanged. This necessarily means that UART frames with 8 data bits need to be used in RTU mode. The advantage of RTU mode is the more effective utilisation of the interface: Only around half of the data volume needs to be transmitted compared to ASCII mode.

6.3 Modbus communication device

Figure Figure 4 shows the state diagram of the transmission and receiving devices in principle, regardless of whether master or slave:

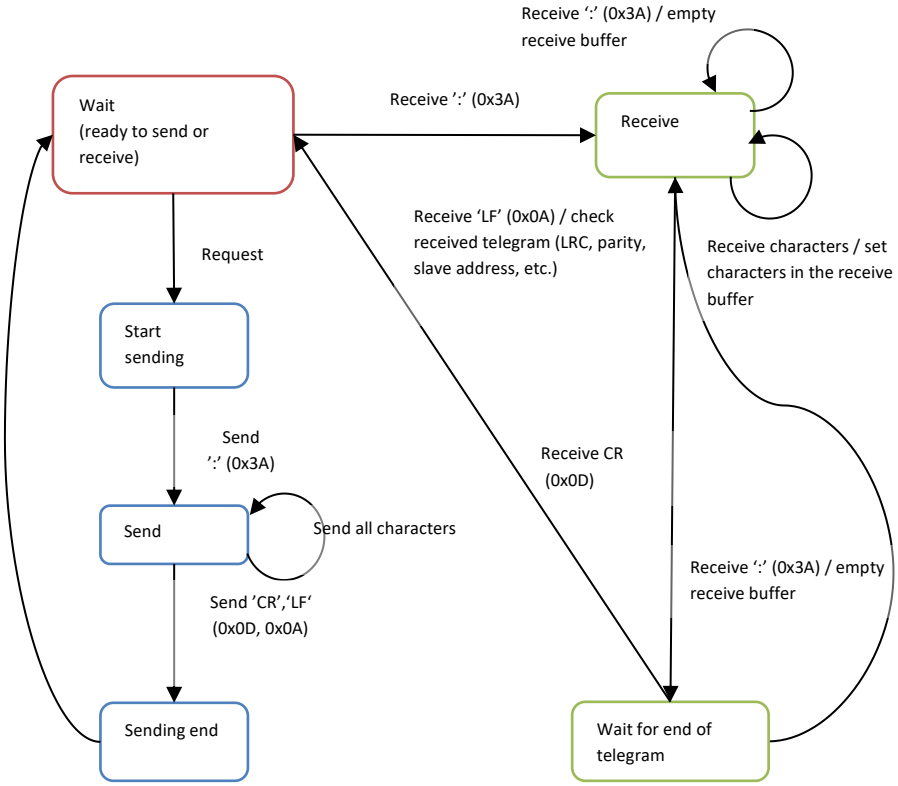


Figure 4: State diagram of a Modbus subscriber (ASCII operating mode)

If an incomplete query is sent to the FLOW^{EVO}, it does not return a response. The module behaves the same when at least one register in the register area being queried does not exist. Error-free telegrams are processed. Faulty telegrams are not answered.

6.4 Modbus slave ID

With the FLOW^{EVO} sensor, the as-delivered device address (slave ID) corresponds to the last two numbers of the serial number on the type plate.

In individual operation:

If only one device is connected with the Modbus master, the sensor module can be queried via the global slave ID 248.

In multiple operation:

If several devices are connected to the Modbus master, the sensor modules must be queried via their slave ID. The sensor modules cannot be queried via the global slave ID 248.

NOTE

Example for calculating the Modbus address:

Device address = **#35** decimal → 0x**23** hex

If the serial number ends with "00", the address is always #100 decimal =0x64 hexadecimal.

The address "0" must never be used!

Figure 5 is a flow diagram that shows how unknown Modbus module addresses can be determined. Any register (e.g. serial number) can be queried via all module addresses (1 – 247) with a timeout of one second. If a module is queried with the correct address, it reacts by sending a response. The module address is included in this response. Thus, at the end of the search cycle, module responses can be used to analyse which module addresses are presently connected to the bus system. When the serial numbers are queried, it is then possible to conclude which address is assigned to which module. The permitted address range for the FLOW^{EVO} is between 1 and 247. According to the Modbus specification, the addresses 248 – 255 are reserved. Address 0 stands for broadcast and must not be used!

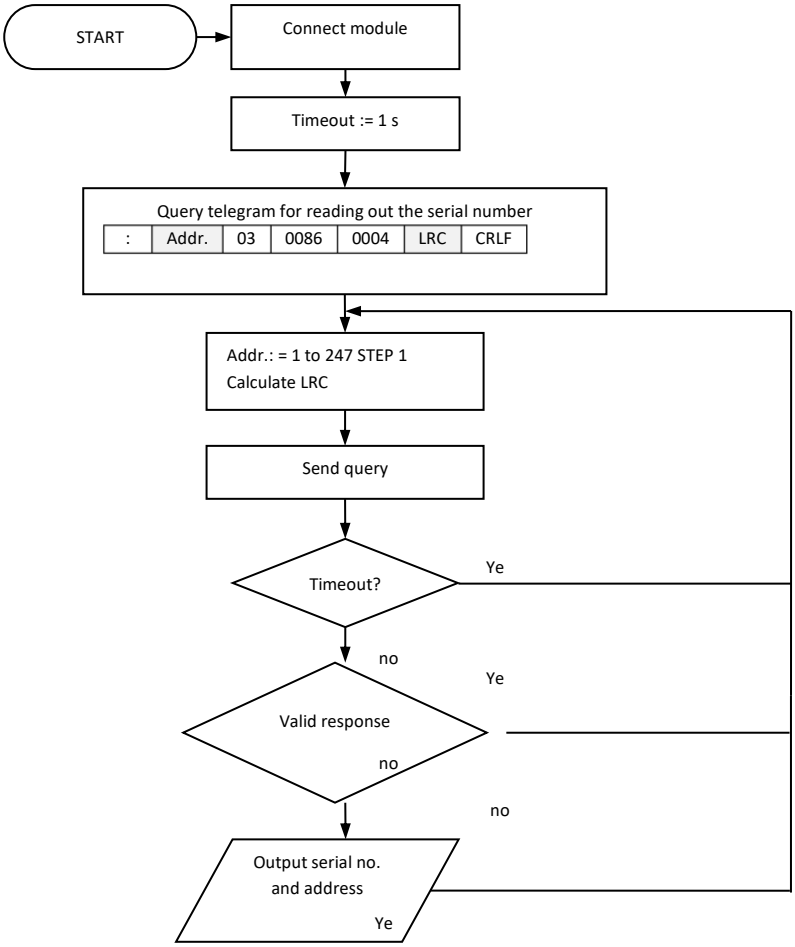


Figure 5: Flow diagram – Determination of module addresses

6.5 Modbus control commands

Communication with the FLOW^{EVO} sensor is supported by only two function codes:

- **0x03 – Read Holding Registers (multiple)**
- **0x06 – Write Single Register (single)**

One register is 16 bits wide and thus consists of 2 bytes:

Register	
15 …	0
High Order Byte - Hi	Low Order Byte - Lo

All the FLOW^{EVO} data that the user can access is shown on registers that are each 16 bits wide.

6.5.1 Control command 0x03 → Read Holding Register

This control command allows you to read values from the FLOW^{EVO} sensor. Note that only the registers defined in these instructions can be read. Therefore, this must be checked especially when multiple registers are queried.

Query		Response		Meaning of the data (according to ASCII table)
Field	(hex)	Field	(hex)	
Modbus address	0xXX	Modbus address	0xXX	
Function	0x03	Function	0x03	
Start Register Hi	0x00	Byte count	0x08	
Start Register Lo	0x80	Register value Hi (128)	0x53	'S'
Register count Hi	0x00	Register value Lo (128)	0x4D	'M'
Register count Lo	0x04	Register value Hi (129)	0x46	'F'
Checksum Lo	0xXX	Register value Lo (129)	0x43	'C'
Checksum Hi	0xXX	Register value Hi (130)	0x4F	'O'
		Register value Lo (130)	0x32	'2'
		Register value Hi (131)	0x20	' ' = Empty characters
		Register value Lo (131)	0x20	' ' = Empty characters
		Checksum Lo	0xXX	
		Checksum Hi	0xXX	

Example 1: Reading out the 4 registers for "Device Type"

In this example, four registers of the FLOW^{EVO} sensor were read starting from the register start address 0x0080 (decimal 128). The response consisted of a payload of 8 bytes that can be resolved with the aid of the ASCII table. Example: Response HEX 53 → to ASCII table → letter S

The response is now **"SMFCO2".** Thus, it is a FLOW^{EVO} sensor (**SMF**) for the measuring gas carbon dioxide (**CO2**).

Query		Response		Meaning of the data
Field	(hex)	Field	(hex)	
Modbus address	0xXX	Modbus address	0xXX	
Function	0x03	Function	0x03	
Start Register Hi	0x00	Byte count	0x02	
Start Register Lo	0x0A	Register value Hi (14)	0x01	456
Register count Hi	0x00	Register value Lo (14)	0xC8	
Register count Lo	0x01	Checksum Lo	0xXX	
Checksum Lo	0xXX	Checksum Hi	0xXX	
Checksum Hi	0xXX			

Example 2: Reading out the "Conc" register (for displaying the gas concentration)

In this example, one register was queried starting from the register start address 0x0A (decimal 10). The two data bytes were transmitted combined as a hexadecimal value. If this value (01C8) is converted to a decimal number, the result is a concentration value of 456.

Query		Response		Meaning of the data
Field	(hex)	Field	(hex)	
Modbus address	0xXX	Modbus address	0xXX	
Function	0x03	Function	0x03	
Start Register Hi	0x00	Byte count	0x02	
Start Register Lo	0x4F	Register value Hi (14)	0x00	3, means ppm x 1
Register count Hi	0x00	Register value Lo (14)	0x03	
Register count Lo	0x01	Checksum Lo	0xXX	
Checksum Lo	0xXX	Checksum Hi	0xXX	
Checksum Hi	0xXX			

Example 3: Reading out the "Unit" register

In this example, one register was queried starting from the register start address 0x004F (decimal 79). The two data bytes were transmitted combined as a hexadecimal value. If this value (0x0003) is converted to a decimal number, the result is "3". This stands for the unit ppm with the scaling x 1. Combined with the data from examples 1 and 2, the FLOW^{EVO} sensor that was read has therefore measured a gas concentration of 456 ppm CO_2.

6.5.2 Control command 0x06 → Write Single Register

This command enables a new value to be systematically written to an addressed register. However, it is only possible to write to those registers intended for this purpose.

Query		Response		Meaning of the data
Field	(hex)	Field	(hex)	
Modbus address	0xXX	Modbus address	0xXX	
Function	0x06	Function	0x06	
Start Register Hi	0x00	Start Register Hi	0x00	
Start Register Lo	0xC0	Start Register Lo	0xC0	
Register count Hi	0x00	Register count Hi	0x00	The new address of the module (160)
Register count Lo	0xA0	Register count Lo	0xA0	
Checksum Lo	0xXX	Checksum Lo	0xXX	
Checksum Hi	0xXX	Checksum Hi	0xXX	

Example 4: Writing to the "Modbus_address" register

In this example, a new Modbus address 0xA0 (hex) = 160 dec. was assigned to the FLOW^{EVO} sensor. Once this communication sequence is complete, the device is only responsive at this new address!

NOTE

The address 0 as well as addresses > 247 must not be assigned!

Query		Response		Meaning of the data
Field	(hex)	Field	(hex)	
Modbus address	0xXX	Modbus address	0xXX	
Function	0x06	Function	0x06	
Start Register Hi	0x00	Start Register Hi	0x00	
Start Register Lo	0x47	Start Register Lo	0x47	
Register count Hi	0x00	Register count Hi	0x00	The zero point has been reset
Register count Lo	0x01	Register count Lo	0x01	
Checksum Lo	0xXX	Checksum Lo	0xXX	
Checksum Hi	0xXX	Checksum Hi	0xXX	

Example 5: Writing to the IR_4tagneu register (setting the zero point)

In this example, the zero point has been reset for the FLOW^{EVO} sensor. This was done by writing the value 1 to register 0x0047 (decimal 71). The device subsequently internally calculated and saved the current correction value for the zero point. The value of the correction is then shown when the same register is read out.

NOTE

The zero point must only be set when zero gas has been applied and the concentration value subsequently remains stable.

Query		Response		Meaning of the data
Field	(hex)	Field	(hex)	
Modbus address	0xXX	Modbus address	0xXX	
Function	0x06	Function	0x06	
Start Register Hi	0x00	Start Register Hi	0x00	
Start Register Lo	0x54	Start Register Lo	0x54	
Register count Hi	0x27	Register count Hi	0x27	Correction value has been set to 10000
Register count Lo	0x10	Register count Lo	0x10	
Checksum Lo	0xXX	Checksum Lo	0xXX	
Checksum Hi	0xXX	Checksum Hi	0xXX	

Example 6: Writing to the SPAN register (end point correction comparison)

In this example, a new end point correction was set for the FLOW^{EVO} sensor. A value of 2710 (hex) = 10000 (decimal). This is also the delivery condition. A value of 11000 would mean, for example, that the concentration value displayed is 10% higher than internally measured. This register therefore makes it possible to correct deviations in the concentration display of the FLOW^{EVO} sensor.

NOTE

The end point must only be set in this way when a suitable test gas is applied and the concentration value subsequently remains stable!

Before the end point is set, the zero point must have previously been set correctly.

6.6 Calculating the checksum

The calculation of the checksum CRC specifically for the RTU operating mode will now be explained based on an example. How the calculation of the LRC checksums in ASCII standard works is described thoroughly in the documentation of the Modbus standard.

The checksum is calculated via the slave ID, the function and the associated data (start register and register count). As an example, we generate a query for reading out the Conc register from the FLOW^{EVO} sensor with the address 14 (decimal) = 0E (hex.)

Query	
Field	(hex)
Modbus address	0x0E
Function	0x03
Start Register Hi	0x00
Start Register Lo	0x0A
Register count Hi	0x00
Register count Lo	0x01
Checksum Lo	0xXX
Checksum Hi	0xXX

Therefore, in hexadecimal format, the resulting byte string is 0x0E, 0x03, 0x00, 0x0A, 0x00, 0x01. The checksum is now created; here is an example code for calculating the CRC checksum:

C# example to calculate modbus RTU checksum:

```
        /// <summary>
        /// Calculates the checksum of an modbus RTU message and adds it to the end
(last 2 bytes).
        /// </summary>
        /// <param name="Databytes"></param>
        /// <returns></returns>
        private void Calculate_CRC(ref byte[] Databytes)
        {
            UInt16 v_CRC = 0xFFFF;

            for (int x = 0; x < Databytes.Length - 2; x++)
            {
                v_CRC ^= (UInt16)Databytes[x];          // XOR byte into least sig.
byte of crc

                for (int y = 8; y != 0; y--)
                {    // Loop over each bit
                    if ((v_CRC & 0x0001) != 0)
                    {      // If the LSB is set
                        v_CRC >>= 1;                    // Shift right and XOR 0xA001
                        v_CRC ^= 0xA001;
                    }
                    else                                // Else LSB is not set
                        v_CRC >>= 1;                    // Just shift right
                }
            }
        }
```

Figure 6: Code example for creating CRC checksums

After the calculation of the checksum and the end code, the following data string would then be sent: **0xF7A4**

Query	
Field	(hex)
Modbus address	0x0E
Function	0x03
Start Register Hi	0x00
Start Register Lo	0x0A
Register count Hi	0x00
Register count Lo	0x01
Checksum Lo	0xA4
Checksum Hi	0xF7

The checksum is included each time data is sent and is then recalculated by the recipient again. If the data set is corrupted or adulterated, the checksum calculated by the recipient would deviate from the one that was sent. The data set would then be unusable.

7 Register overview

All of the registers listed below are holding registers.

Address	Name	R/W	Data type	Function / description
0x0003	T_m	R/---	signed number	Measured value for internal temperature (x0.1°C)
0x0009	Sys_status	R/---	number	Status bit bar, see page 18 for details
0x000A	Konz	R/---	signed number	Measured value for gas concentration in ppm, vol.% or %LEL (Please note unit code!)
0x0047	IR_4tagneu	R/W	number	Zero point reference value.
0x004F	Einheit	R/---	number	Unit code and scaling factor for the concentration. Details and calculation example later in this document.
0x0051	Konz_fs	R/---	number	Upper range value (full scale) of the concentration.
0x0054	Span	R/W	number	End point reference value. Value must be between 5000 and 15000; otherwise it is reset to 10000.
0x0059	fab_zero_value	R/---	signed number	Correction factor for zero point calibration ex works
0x005A	fab_span_value	R/---	signed number	Correction factor for span calibration ex works
0x0080-0x0083	DeviceType	R/---	string	Indicates the type of the connected device
0x0084-0x0085	SW-Version	R/---	string	Firmware version of the connected device
0x0086-0x0089	SerialNr	R/---	string	Serial number of the connected device
0x00C0	Modbus_ address	R/W	number	Modbus address of the FLOW^{EVO}. Once this address has been changed, communication with the FLOW^{EVO} can continue only via the new address.

Table 5: Modbus register table

- **R – Read Holding Registers (multiple)**

- **W – Write Single Register (single)**

NOTE

All other registers not described here must not be changed under any circumstances.

7.1 Meaning of the individual bits in the status bit bar (SYS_Status):

Faults and error messages can be identified with the aid of the SYS_Status register according to the following table.

Bit	Name	Value → Message
00	----	Without function (reserved)
01	WARMUP	1 → FLOW^{EVO} is in the warm-up phase (approx. 10 s)
02	SYS_ERR	1 → System fault
03	----	Without function (reserved)
04	----	Without function (reserved)
05	STARTUP	1 → FLOW^{EVO} is in the boot phase (approx. 40 s)
06	KORR	1 → Correction active (always)
07	MW_ok	1 → The zero point has been set
08	----	Without function (reserved)
09	----	Without function (reserved)
10	----	Without function (reserved)
11	MW_aktiv	1 → Averaging for drift correction is active
12	EEP_ERR	1 → EEPROM error
13	WDG_WRN	1 → After watchdog reset
14	POWER_ON	1 → Supply voltage switched on
15	OUT_OF_RANGE	1 → when “0x000A Conc” < -10% FS or “0x000A Conc” > 110% FS

Table 6: Allocation of the error messages in the status bit bar SYS_Status

FS= (Full scale) upper range value

NOTE

The value 0 always stands for the (error-free) normal state.

The two bits 6 (CORR) and 7 (MW_ok) are internal flags set during the manufacturing process of the individual FLOW^{EVO}. They are also used for quality control purposes and are set to the value “1” if the respective FLOW^{EVO} has been temperature-compensated and calibrated.

7.2 Description of the unit code:

The Conc register (0x000A) provides a numerical value with a varying scaling and unit depending on the FLOW^{EVO} version. The Unit register (0x004F) can be used to correctly calculate the concentration value. The meaning of the numerical value in the Unit register (0x004F) is therefore displayed as follows:

Register value	→	Unit / scaling
0	→	Unassigned, for special applications
1	→	ppm x 0.01
2	→	ppm x 0.1
3	→	ppm
4	→	Vol.% x 0.001
5	→	Vol.% x 0.01
6	→	Vol.% x 0.1
7	→	LEL x 0.01 %
8	→	LEL x 0.1 %

Table 7: Allocation of register value to the measuring unit and multiplier

NOTE

Partial quantities of < 1 vol.% are mostly specified as a ppm value. The following table shows the relationship of vol.% to ppm:

Vol.%	ppm
100	1,000,000
10	100,000
1	10,000
0.1	1000
0.01	100
0.001	10
0.0001	1

Table 8: Relationship of vol.% to ppm

8 Information on start-up and operation

NOTE

We recommend the smartGAS Calibration Tool for setting the zero point and the end value. This can be downloaded free of charge from the smartGAS homepage.

8.1 Self-test

After the FLOW^{EVO} sensor is switched on, an internal self-test is carried out and the green LED flashes. After that, the sensor supplies measured values, and system errors are evaluated.

NOTE

Correct measurement values are not output during the self-test.

8.2 Setting the zero point

It is advisable to set the zero point

- after reinstallation of the sensor or measuring system
- at regular intervals (must be adapted to the application)

- after repairs / maintenance work on the sensor or measuring system

NOTE

Before the zero point is adjusted, the sensor must be in operation for at least 30 minutes and a zero gas (e.g. N_2 – 100 vol.%) must flow through the sensor until the indicator for the gas concentration has reached a stable value.

If the aforementioned requirements are met, the value 1 is written in the register IR_4tagneu (0x0047) and the zero point is thus reset.

8.3 Setting the end point

Setting the end point (also called final value or span calibration) requires the use of a test gas, which should correspond as accurately as possible to the upper range value of the sensor to be calibrated.
The same preconditions apply here as when setting the zero point: the sensor must be in operation for at least 30 minutes and the test gas must flow through it until a stable value has been reached in the Conc register (0x000A).
If all requirements have been met, the correction value for the respective measurement channel is written in the Span register (0x0054).

8.4 Calculating the correction value for the end point

Let us assume that a sensor indicates a concentration of only 978 ppm (called “Conc_old” here) when a test gas is applied which has the value 1003 ppm (called “Conc_cal” here).
Reading the Span register yields the value 9985 (called “Span_old” here).

The new calculation of the correction value for the Span register then takes place as follows:

Span_new = Conc_cal x Span_old / Conc_old
Span_new = 1003 x 9985 / 978 = **10240**

The new value of **10240** is now written in the Span register (0x0054) and the process is complete!

NOTE

Always carry out a zero point adjustment with your smartGAS sensor first and then an end point adjustment.

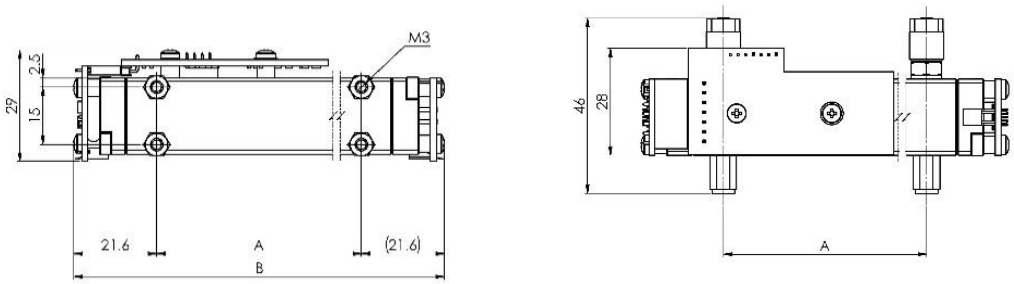
8.5 Restoring the calibration parameters to factory settings

To restore the calibration parameters to the factory settings, the registers “IR_4tagneu” and “Span” can be rewritten. For this purpose, the register value from “fab_zero_value” must be written to “IR_4tagneu” and “fab_span_value” to “Span”.

9 Annex

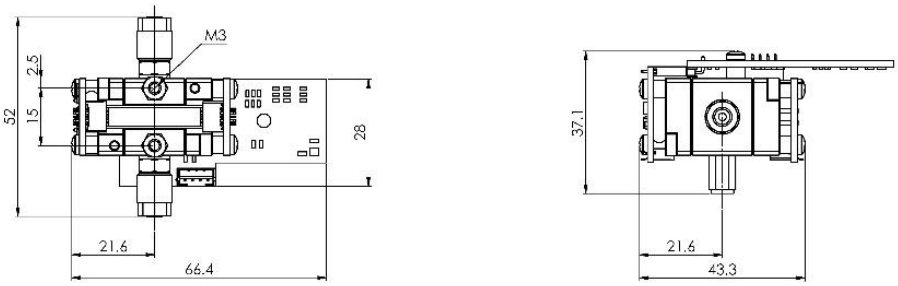
9.1 Mechanical dimensions [mm]

Design type I:

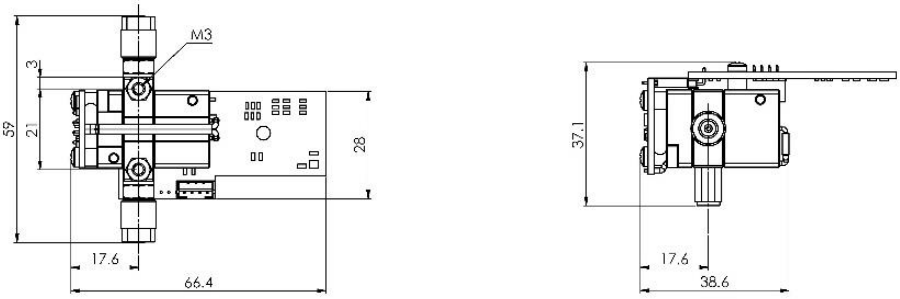


Cell length	A	B
45 mm	30	73.2
85 mm	70	113.2
105 mm	90	133.2
125 mm	110	153.2
205 mm	190	233.2
305 mm	290	333.2

Design type II:



Design type III:



9.2 FLOW^{EVO} operation on a microcontroller

For the FLOW^{EVO} to communicate with a microcontroller, the level of the signals at the module COM pin must be adapted to the microcontroller. The easiest method is via an RS485 interface module that must be selected according to the microcontroller voltage supply.

The UART signals TXD (transmit data), RXD (receive data) and a signal for activating the transmitter TXEN (transmitter enable) must be provided at the microcontroller. The following figure shows the circuit:

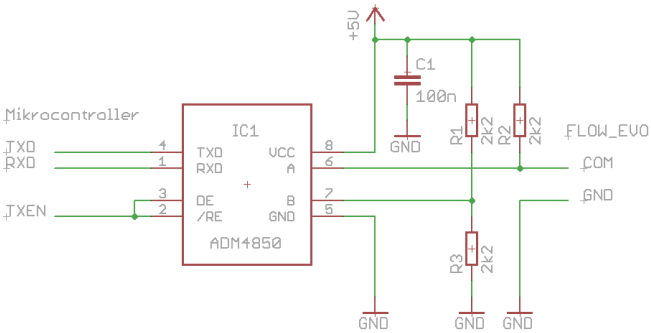


Figure 7: FLOW^{EVO} on a microcontroller

The circuit is designed for a microcontroller with 5 V operating voltage. When operating at 3.3 V, use an ADM3075 (or equivalent types) rather than the ADM4850 (or equivalent types). All other components are unaffected.

Up to 16 FLOW^{EVO} can be operated in parallel with this circuit. This requires the devices to have different Modbus addresses.

NOTE

It should be noted that FLOW^{EVO} must be connected to a power supply in addition to the communication connection shown above.

9.3 FLOW^{EVO} operation on a PC

Operating exactly **one** FLOW^{EVO} on a PC requires a special USB adapter incl. software; this can be purchased from smartGAS as an accessory (article no. Z6-000025). The FLOW^{EVO} is powered via the USB port; an additional power supply is not necessary. The following figure shows one such adapter:

Figure 8: USB adapter for operating a FLOW^{EVO} on the PC

10 Legal information

The figures and drawings used in this description may differ from the originals; they are provided solely for illustrative purposes.
All information – including technical specifications – is subject to change without notice.

smartGAS Mikrosensorik GmbH |Hünderstr. 1 | 74080 Heilbronn | Germany
Phone: +49 7131/797553-0 | fax: +49 7131/797553-10 | www.smartgas.eu|mail@smartgas.eu

Edition 04/11_22