SPECIFICATION

HTC-MODBUS-V-L

„CO2 Concentration transducer
► MODBUS RTU"

Developed by:

HOTCOLD s.c.

2017-06-08

Table of contents

1. Introduction

The subject of the present study is the functional characteristics of the transducer of CO2 concentration based on the MHZ14 sensor, with an interface RS-485 with a built-in MODBUS RTU protocol, with an analog output in the 0-10V standard and, depending on the version, retrofitted with a temperature measurement sensor - outputs available on the connector.

NOTICE: Read the text in this paper before running the module.

1.1. Device features

- concentration measurement of **CO2**
- analogue voltage output 0-10 [V] (proportional to CO2 concentration)
- Temperature sensor leads available on the connector
- LED signaling of device operation
- serial interface RS-485 (reading measurement values, configuring operating parameters)
    - protocol MODBUS RTU
    - communication in HALF DUPLEX mode
    - hardware configurable address (1-127)
    - hardware configurable speed (9600, 19200, 38400, 57600, 115200)

1.2. Device characteristics

The primary function of the CO2 v2 transmitter is to measure the CO2 concentration in the air. The CO2 concentration values measured via the integrated MHZ14 sensor are then recalculated and averaged in the microcontroller, and are available in its memory (in registers of type HOLDING REGISTERS) according to the MODBUS standard. The registers are read using MODBUS protocol functions transmitted over the serial RS-485 interface. Indication of sensor missing/error, measurement range exceeded status is realized by status registers. The values can also be presented in an analog form at the voltage output in the 0-10V standard.

As an option, it is also possible to use a thermoresistance sensor (RTD), with leads available on the connector.

2. Technical specifications

2.1. Transducer general specifications

Power supply	
- DC voltage	DC 24V (20...30V)
- AC voltage	AC 24V (20...27,6V)
Current consumption	
- usual [1]	*<35,0 mA*
LED signaling<70.0 mA - maximum [2]	description under "LED indication"
Installation coupling	screw in 5.00mm grid (≤ 2,5mm^2)
Dimensions	120 x 80 x 25 (L x H x W)
Weight	100 g
Montage [3]	wall-mounted
Operating environment	dust-free, air, neutral gases
Operating temperature	0°C ÷ 50°C

1) Average current consumption of the device in the following conditions: transmission of 10 requests per second; transmission speed 9600 b/s; simultaneous reading of 3 registers; bus terminating resistors 2 x 120Ω; power supply 24V DC, voltage output loaded with 10k resistance;
2) Maximum instantaneous current consumption of the device under conditions as in 1) + voltage output loaded with 1k resistance;
3) The unit should be installed by qualified personnel; Vertical orientation as marked
UP – up, DOWN – down;

2.2. CO2 measuring parameters

Sensor type	MHZ14
Measurement range	0 ÷ 2000 ppm
Accuracy:	
- in the range of 400 ÷ 1250 ppm	± 3 %
- in the range of 1250 ÷ 2000 ppm	± 5 % ± 50 ppm
Sampling rate	2Hz
Response time [1]	< 2 min

1) The response time shown is equal to one time constant corresponding to 90% of the set value;

2.3. Analog output parameters

Output type	voltage
Output range	10 V
Resolution	12 bit (5 mV)
Load capacity	R_L > 1 kΩ
Refresh rate	2 Hz

2.4. Temperature sensor parameters (RTD)

Sensor type	i.e. NTC1,8K, NTC5,02K, Pt100, Pt1000 and others
Measurement range	0°C ÷ 50°C
Accuracy	Depends on the sensor used, e.g. Pt100 cl.A +/- 0,15°C
Response time 1)	

1) The condition to obtain the given response times is an air flow > 1m/s; the given response time is equal to one time constant corresponding to 63% of the set value;

2.5. Serial interface parameters

Physical layer	RS-485
Communication protocol	MODBUS RTU
Connection configurations 1)	HALF DUPLEX
Transmission speeds	9600 / 19200 / 38400 / 57600 / 115200 b/s

1) HALF DUPLEX - two-way communication over one pair of wires;

3. Installation

3.1. Security

- The unit should be installed by qualified personnel!
- Make all connections in accordance with the wiring diagrams shown in this specification!
- Check all electrical connections before commissioning!

3.2. Device design

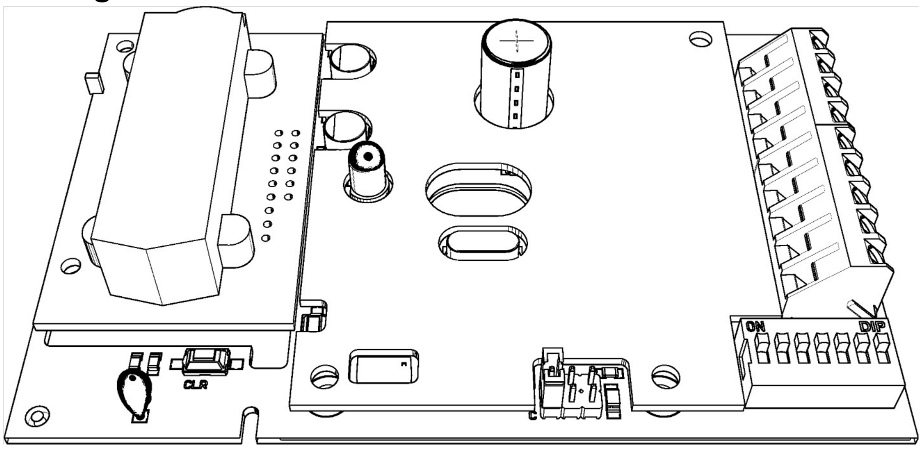


Figure 1. View of the transducer's printed circuit board.

3.3. Pinout description

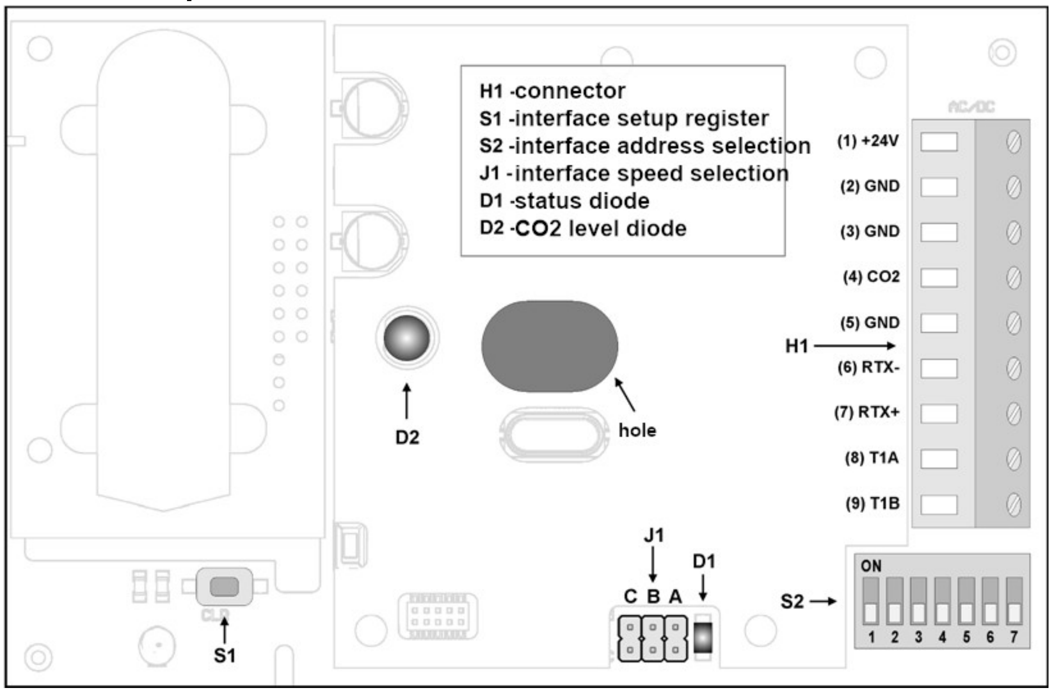


Figure 2. Description of CO2 transducer pinouts in the wall-mounted version.

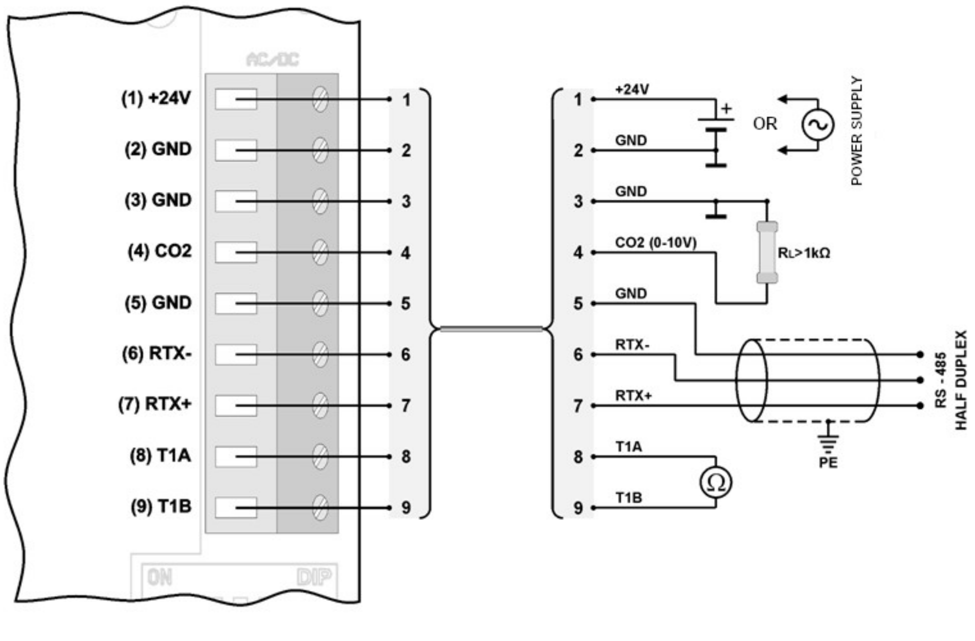


Figure 3. Connection diagram for wall-mounted CO2 transducer

3.4. Address configuration

The device is equipped with a DIP SWITCH with up to 7 positions, for hardware address setting (from "1" to max. "127"). Setting the address to "0" on the switch will use the address stored in the device via the MODBUS protocol (default „1").

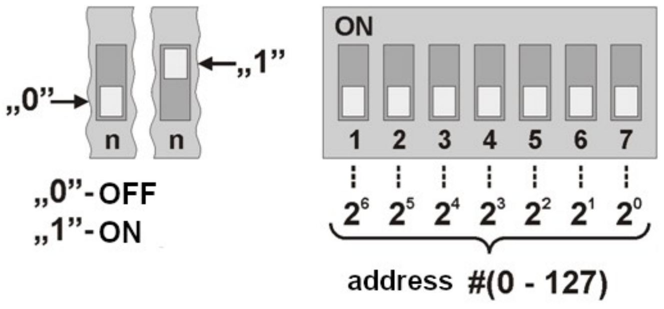


Figure 4. Transducer Addressing.

3.5. Speed configuration

The device is equipped with 3 jumpers for hardware RS-485 interface speed setting (according to the table below). No jumpers will use the speed value stored in the device via MODBUS protocol (default "9600 b/s").

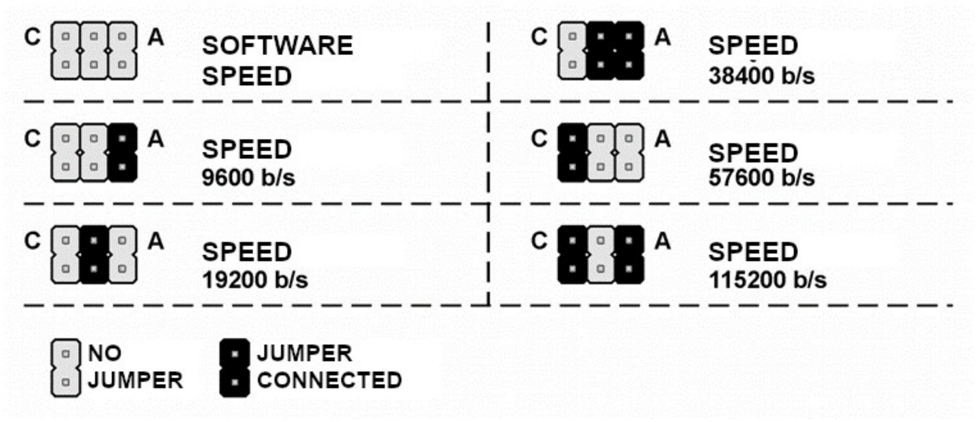


Figure 5. RS-485 interface speed configuration.

3.6. Restoring factory settings

The factory reset function applies only to RS-485 transmission parameters (including address and speed).

To restore the settings, press and hold the S1 button for about 2 seconds (protection against accidental pressing). When the D1 diode starts blinking release the button. The device will start working with the new settings automatically.

3.7. LED indication

Table of levels / statuses signaled on D2 diode:

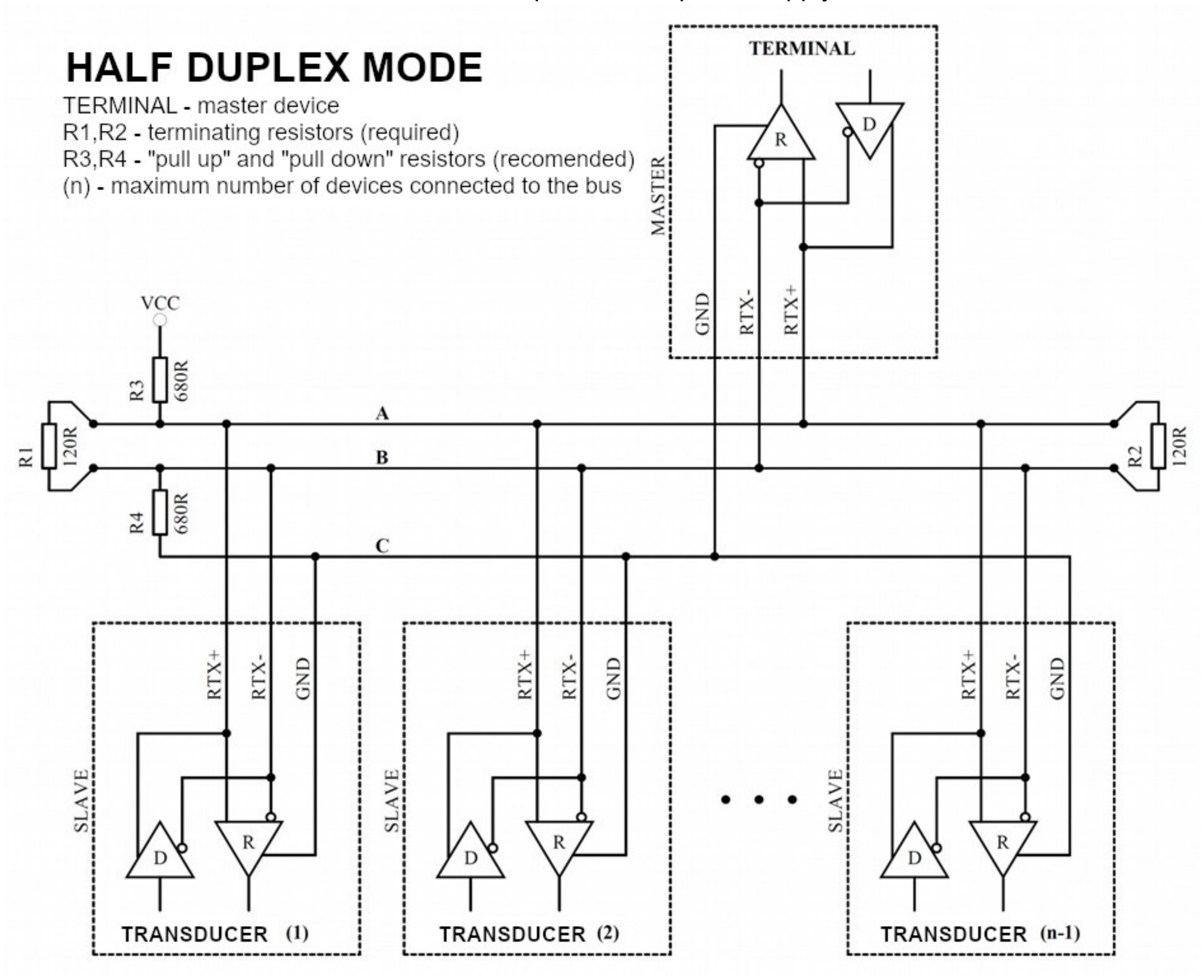
Status	Description	Diode color	Behavior
1	warming up the CO2 module	green	blinking (250ms / 250ms**)
2	0 – 800 [ppm] *	green	continuous light
3	800 – 1200 [ppm] *	yellow	continuous light
4	1200 – 2000 [ppm] *	red	continuous light
5	> 2000 [ppm] red	blinking (250ms / 250ms**)	
6	no CO2 sensor or other error	red	blinking (100ms / 600ms**)

(*) The switching hysteresis of the LED light state is ± 50 ppm.

(**) Blinking (XXX ms / YYY ms) means XXX - switching on time, YYY - switching off time

3.8. Guidelines

- When operating in a high noise environment, use shielded cables.
- Connect the cable shield to the nearest PE point on the power supply side.



HALF DUPLEX MODE
TERMINAL - master device
R1,R2 - terminating resistors (required)
R3,R4 - "pull up" and "pull down" resistors (recomended)
(n) - maximum number of devices connected to the bus
TERMINAL
MASTER
R
D
GND
RTX-
RTX+
VCC
R3
680R
R4
680R
R1
120R
R2
120R
A
B
C
SLAVE
TRANSDUCER (1)
TRANSDUCER (2)
TRANSDUCER (n-1)

Figure 6. The way of connecting the transducer to the RS-485 bus working in HALF DUPLEX mode.

4. MODBUS protocol

4.1. Map of registers

Registers table:

Registry No.	Values	Description
1	400 – 2000	C02 concentration [ppm] (1 = 1ppm)
2	*0 / 1 1 2 1 3 14 15 1 6*	Status register (O: "NO SENSOR", 1: "SENSOR OK", 2: "ERROR", 3: "WARM UP", 4:"CALIBRATION", 5: "IDLE", 6: "OVER RANGE (*)
3	1000 (Ox03E8)	Test value - to verify correct reading of registers
4	1234	Password register
5	1 / 2 / 3	Command register
6	By command table	Parameter register
7	0-65535	Count of valid frames
8	0-65535	Exceptions counter
9	0-65535	Count of erroneous CRCs
10	0-65535	Incorrect byte count
11	-	Not used
12	400 – 5000	for maintenance purposes only

(*)

"NO SENSOR" - no sensor; "SENSOR OK" - sensor OK; "ERROR" - sensor malfunction; "WARM UP" - sensor in process of warming up; "CALIBRATION" - sensor in calibration mode; "IDLE" - sensor in sleep mode; "OVER RANGE" - measurement range exceeded; Command table:

Command No.	Function	Parameters
1	Set device address	1 - 247 (1 - default value)
2	Set baud rate (transmission speed)	96 - 900 b/s (1 - default value) 192 – 19200 b/s 384 – 38400 b/s 576 – 57600 b/s 1152 – 115200 b/s
3	Set parity bits	0 - NO PARITY; no parity bit 1 - EVEN PARITY; (default value) 2 - ODD PARITY,
4	Set Stop Bits	1 - 1 x STOP; 1 stop bit (default value) 2 - 2 x STOP; 2 stop bits
5	Set elevation	0 – 2500 [m above sea level] (altitude)
105	Read the elevation	as above
6	Device reset	1 - software reset of the device

Notices:

- Specifying an incorrect or out-of-range parameter value results in 0xEEEE value being written to the command register.
- Each command invocation must be accompanied by the entry of a password (1234 decimal).
- When invoking the command through individual registry entries, it must be completed by entering a password.

4.2. Protocol functions

The following MODBUS standard functions are implemented in the transducer:

CODE	*MEANING*
03 (0x03)	*Reading N x 16-bit registers*
16 (0x10)	Writing N x 16-bit registers

4.2.1. Reading the contents of output register group (0x03)

Request format

Description	*Size*	*Values*
Device address	*1 byte*	*1 – 247 (0xF7)*
Function code	*1 byte*	***0x03***
Data block address	*2 bytes*	*0x0000 – 0xFFFF*
Number of registers (N)	*2 bytes*	*1 – 125 (0x7D)*
CRC Checksum	*2 bytes*	*as calculated*

Response Format:

Description	*Size*	*Values*
Device address	*1 byte*	*1 – 247 (0xF7)*
Function code	*1 byte*	***0x03***
Byte counter	*1 byte*	*2 x N*
Register values	*N x 2 bytes*	*by register map*
CRC checksum	*2 bytes*	*as calculated*

Error Format:

Description	*Size*	*Values*
Device address	*1 byte*	*1 – 247 (0xF7)*
Function code	*1 byte*	***0x03***
Error code	*1 byte*	*0x01 / 0x02 / 0x03 / 0x04*
CRC Checksum	*2 bytes*	*as calculated*

4.2.2. Writing to output register group (0x10)

Request Format:

Description	*Size*	*Values*
Device address	*1 byte*	*1 – 247 (0xF7)*
Function code	*1 byte*	***0x03***
Data block address	*2 bytes*	*0x0000 – 0xFFFF*
Number of registers (N)	*2 bytes*	*1 – 123 (0x7B)*
Byte counter	*1 byte*	*2 x N*
Values	*N x 2 bytes*	*User*
CRC Checksum	*2 bytes*	*as calculated*

Response Format:

Description	*Size*	*Values*
Device address	*1 byte*	*1 – 247 (0xF7)*
Function code	*1 byte*	***0x03***
Data block address	*2 bytes*	*0x0000 – 0xFFFF*
Number of registers (N)	*2 bytes*	*1 – 123 (0x7B)*
CRC Checksum	*2 bytes*	*as calculated*

Error Format:

Description	*Size*	*Values*
Device address	*1 byte*	*1 – 247 (0xF7)*
Function code	*1 byte*	***0x03***
Error code	*1 byte*	*0x01 / 0x02 / 0x03 / 0x04*
CRC Checksum	*2 bytes*	*as calculated*

4.3. Data format

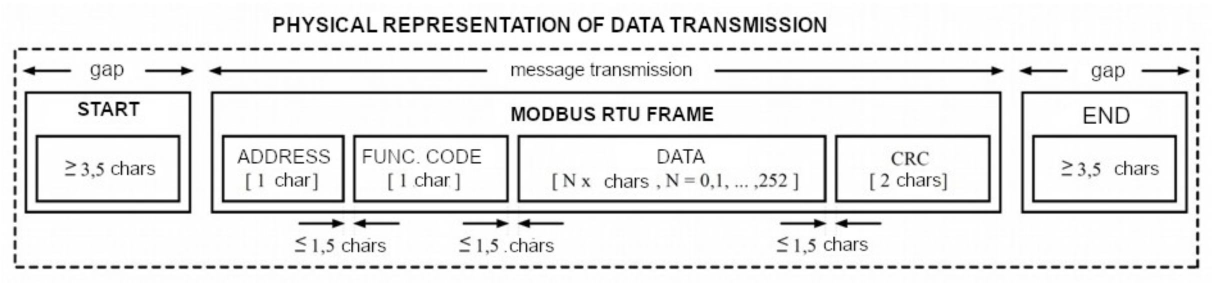


Figure 7. Data transmission in MODBUS RTU standard implemented in the transducer.

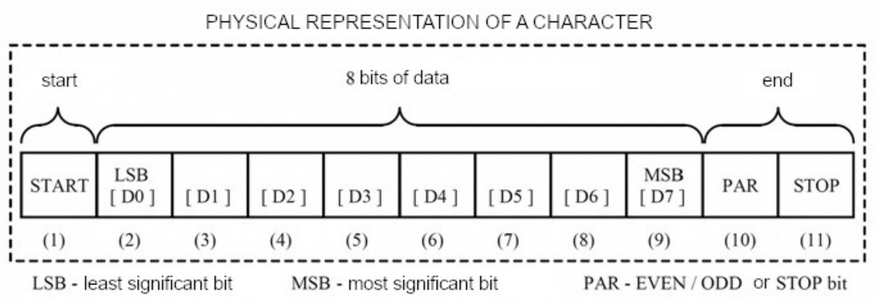


Figure 8. MODBUS RTU standard character format used in the transducer.

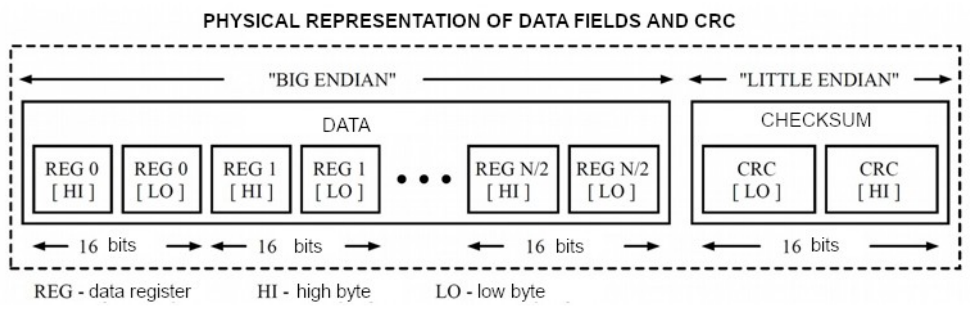


Figure 9. Format of data fields and CRC in MODBUS RTU standard used in the transducer.

4.4. CRC Checksum

According to the MODBUS standard, a polynomial was used to calculate the CRC checksum: X16 + X15 + X2 + 1.

4.4.1. Bitwise CRC counting algorithm:

Procedure for determining the CRC checksum using the bitwise method:

a) loading the value 0xFFFF into the 16-bit CRC register;

b) fetching the first byte from the data block and performing an EX-OR operation with the younger byte of the CRC register, placing the result in the register;

c) shifting the CRC register contents to the right by one bit towards the least significant bit (LSB), zeroing the most significant bit (MSB);

d) Checking the state of the youngest bit (LSB) in the CRC register, if its state equals 0, then return to the point c, if 1, then the EX-OR operation of the CRC register with the constant 0xA001 is performed;

e) repeating points c and d up to eight times, which corresponds to processing an entire byte;

f) repeating the sequence b, c, d, e for the next byte of the message, continuing this process until all bytes of the message have been processed;

g) the content of the CRC register after the listed operations are performed is the sought CRC checksum value;

h) adding a CRC checksum to the MODBUS RTU frame must be preceded by swapping places of the older and the younger byte of the CRC register.

4.4.2. Array CRC counting algorithm:

An example implementation of the CRC checksum determination procedure using the array method:

```
/* The function returns the CRC as a unsigned short type */
unsigned short CRC16 ( puchMsg, usDataLen )
/* message to calculate CRC upon */
unsigned char *puchMsg ;
/* quantity of bytes in message */
unsigned short usDataLen ;

{
        /* high byte of CRC initialized */
        unsigned char uchCRCHi =
        0xFF ; /* low byte of CRC
        initialized */ unsigned char
        uchCRCLo = 0xFF ;
        /* will index into CRC lookup table */
        unsigned uIndex ;

        /* pass through message buffer */
        while (usDataLen--)
        {
                /* calculate the CRC */ uIndex = uchCRCLo
                ^ *puchMsg++ ; uchCRCLo = uchCRCHi ^
                auchCRCHi[uIndex] ; uchCRCHi =
                auchCRCLo[uIndex] ;
        } return (uchCRCHi << 8 |
uchCRCLo) ;
}
/* Table of CRC values for high-order byte */
static unsigned char auchCRCHi[] = {
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
} ;

/* Table of CRC values for low-order byte */
static char auchCRCLo[] = {
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
};
```