DFF.PRESS.TRDC 6kPa/24VDC/010V/Mod

1 Introduction 3
1.1 Device functions 3
1.2 Characteristics of the device 3
2 Worth knowing 3
3 Technical data 4
3.1 General parameters of the transducer 4
3.2 Differential pressure measurement parameters 4
3.3 Parameters of the analog output 4
3.4 Parameters of the serial interface 5
4 Installation 5
4.1 Security 5
4.2 The construction of the device 5
4.3 Description of leads 6
4.4 Configuration of the analog output 7
4.5 Address configuration 7
4.6 Resetting the offset 8
4.7 Restoring factory settings 8
4.8 Guidelines 9
5 MODBUS protocol 10
5.1 Map of registers 10
5.2 Protocol functions 12
5.2.1 Reading the contents of output registers (0x03) 12
5.2.2 Writing to the group of output registers (0x10) 13
5.3 Data format 14
5.4 CRC checksum 14
5.4.1 Bitwise CRC calculation algorithm: 14
5.4.2 Table-based CRC calculation algorithm: 15

1 Introduction

The subject of this study is the characteristics of the differential pressure transmitter based on the Honeywell ABP series pressure sensor, with the RS-485 interface with the built-in MODBUS RTU protocol, and the analogue 0-10V output.

ATTENTION: Before starting the module, please read the text contained in this document.

1.1 Device functions

• differential pressure measurement (range depending on the sensor used)
• analog voltage output 0-10 [V] (proportional to the pressure difference value)
• configuration of the output range
• configuration of the time constant of the measurement
• sensor offset zeroing function
• LED signaling device operation
• serial RS-485 interface (readout of measured values, configuration of work parameters)
- MODBUS RTU protocol
-for communication in HALF DUPLEX mode
-for a hardware configurable address (1-31)

1.2 Characteristics of the device

The basic function of the transducer is to measure the value of the pressure difference. The values measured by the integrated Honeywell sensor of the ABP series, then converted and averaged in the microcontroller, are available in its memory (in the HOLDING REGISTERS) according to the MODBUS standard. The registers are read using the MODBUS protocol functions sent over the RS-485 serial interface. The registers also provide information on the currently set (configurable) measuring range, the time constant (also configurable) and the percentage of the pressure referenced to the range. The signaling of the lack of a sensor, the states of measuring range overrun, the occupancy of the transducer in the case of offset calibration is also carried out via status registers.

2 Worth knowing

1 hPa = 100 Pa = 1 mbar
1 inH2O = 249.089 Pa

3 Technical data

3.1 General parameters of the transducer

Power	
- constant voltage	DC 24V (20 ... 30V)
- alternating voltage	AC 24V (21.5 ... 26.5V)
Power consumption	
- minimum [1]	9.0 mA
- typical [2]	11.0 mA
- maximum [3]	22.0 mA
LED signaling	0.2 Hz
Installation connector	screw in 5.00mm pitch (≤ 2.5mm [2])
dimensions	112 x 84 x 31 (L x H x W)
Weight	approx. 100 g
Assembly [4]	wall
Level of security	IP65
Working environment	dust-free, air, neutral gases
Working temperature	-20 ° C ÷ 50 ° C
The storage conditions	
- temperature	-40 ° C ÷ 85 ° C
- relative humidity	20 ÷ 60% RH

1)Average device current consumption in the following conditions: no transmission; analogue output unloaded; 24V DC power supply;
2)Average power consumption of the device in conditions: transmission 10 queries per second; transmission speed 9600 b / s; simultaneous reading of 20 registers; bus terminating resistors 2 x 120 Ω; analog output set to 10V and loaded with a 10kΩ resistance; 24V DC power supply;
3)Maximum instantaneous current consumption in conditions: analog output loaded with 1kΩ resistance; signaling diode constantly on; other conditions as in point 2);
4)The device should be installed by qualified personnel;

3.2 Differential pressure measurement parameters

Sensor type	ABP
Measurement range	Up to 7,000 Pa
Resolution	12 bits
Accuracy:	
- in the range of 0 ÷ 50 ° C	± 0.25% of range
- in the range -20 ÷ 85 ° C	unspecified
Sampling frequency	100 Hz
Response time [1]	0.8s / 4s [2]

1) the given response time is equal to one time constant corresponding to 63% of the set value;
2) the default value is shorter response time;

3.3 Parameters of the analog output

Output type	voltage
Output range	0 - 10 V
Resolution	12 bits (5 mV)
Load capacity	R_L > 1 kΩ
Refresh rate	100 Hz

3.4 Parameters of the serial interface

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

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

4 Installation

4.1 Security

- The device should be installed by qualified personnel!
- All connections must be made in accordance with the wiring diagrams set out in this specification!
- Check all electrical connections before commissioning!

4.2 The construction of the device

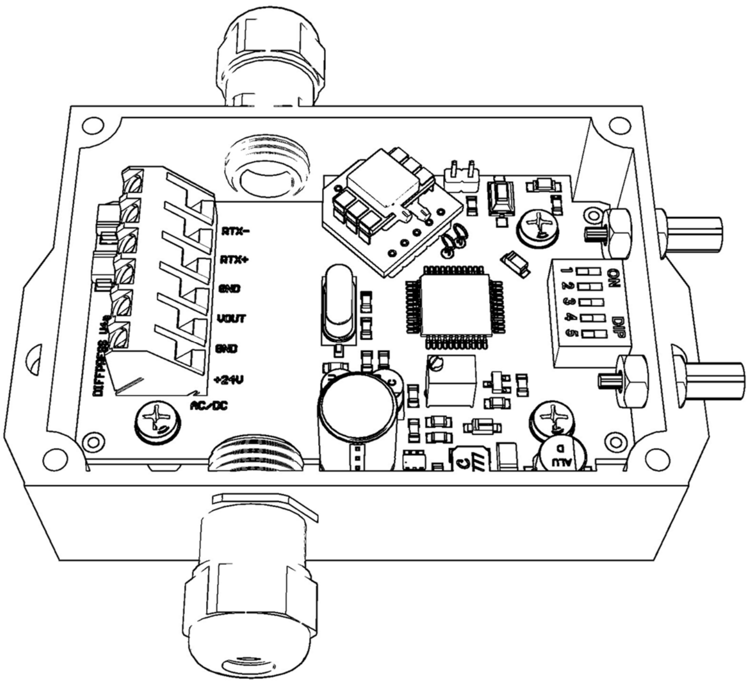

Figure 1. View of the printed circuit.

4.3 Description of leads

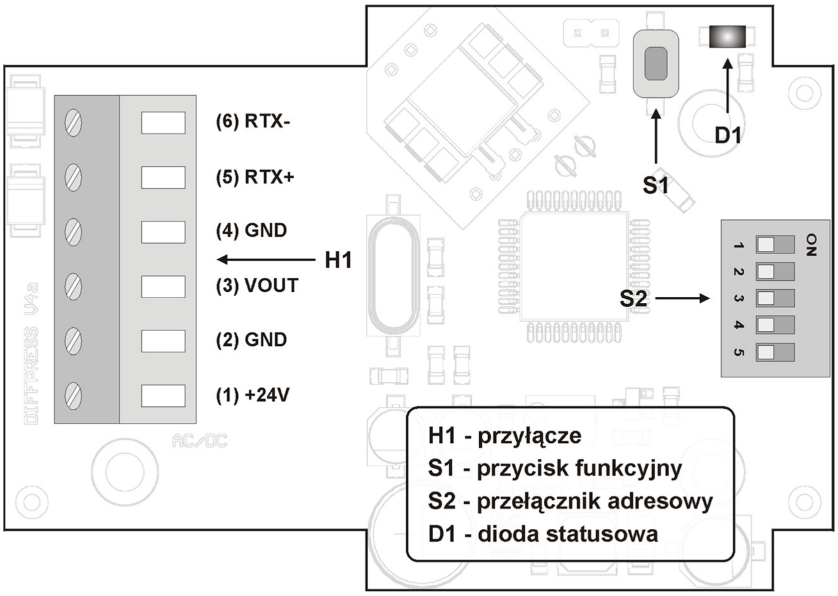

Figure 2. Description of the transmitter's leads.
H1 – main terminal
S1- function button
S2- address button
D1- status LED diode

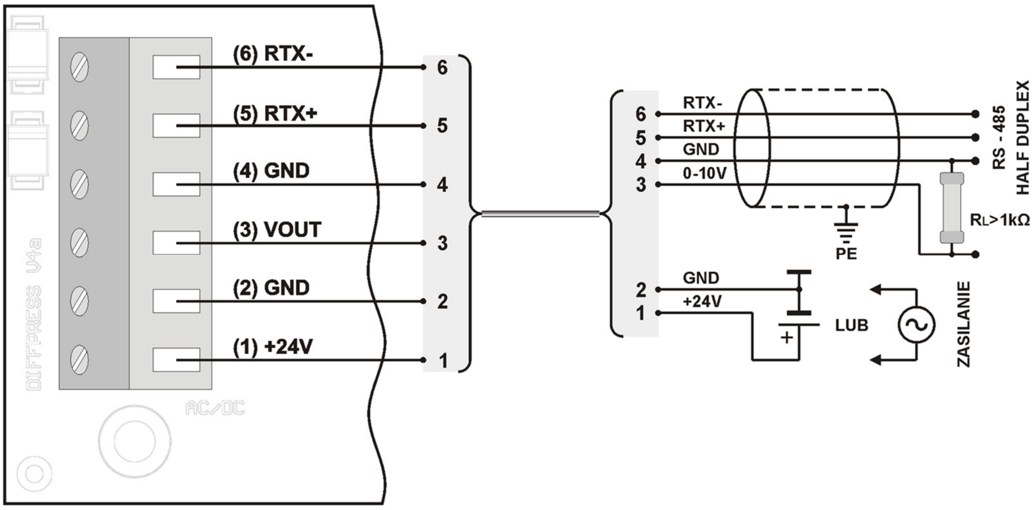

Figure 3. Connection diagram of the transducer.

4.4 Configuration of the analog output

The device is equipped with an analogue output: voltage range from 0 to 10V. Depending on the selected range, the device outputs a value proportional to the measured pressure difference. The measurement results are averaged and refreshed at the outputs according to the set time constant.

4.5 Address configuration

The device is equipped with a 5-position switch for hardware address setting (from "1" to "31"). Setting the address "0" on the switch will use the address stored in the device via the MODBUS protocol ("1" by default).

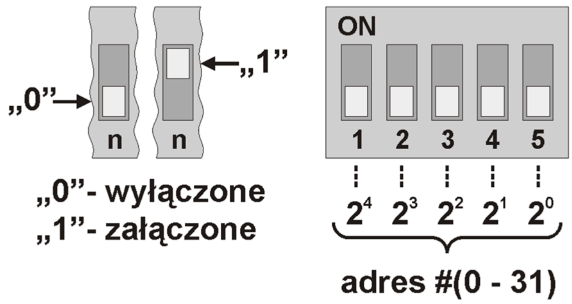

Figure 4. Transducer addressing.
"0"- off
"1"- on

4.6 Resetting the offset

Resetting with the command:
Before proceeding to reset (reset) the offset, you must first set the output range and place both connectors in the same pressure (you can disconnect both hoses). The zeroing process takes place after sending the offset calibration command. The calibration duration is about 7s. After correct calibration, the device should show zero pressure value.

Resetting with the button:
Before proceeding to reset (reset) the offset, both ports should be placed in the same pressure (both hoses can be disconnected). To trigger the zeroing process, press and hold button S1 for about 3 seconds until the diode D1 blinks. The duration of the calibration is counted from the moment the button is released and is about 7 seconds. The calibration process is signaled by the blinking of the diode D1. After correct calibration, the device should show zero pressure value.

4.7 Restoring factory settings

The function of restoring factory settings applies only to parameters of RS-485 interface transmission (including address). To restore the settings, press and hold the S1 button for about 10 seconds (the first 3 seconds after pressing the LED D1 is on, the next 7 seconds are blinking). When the D1 LED is permanently on again, release the button. The device will work with new settings automatically.

4.8 Guidelines

• In case of work in the vicinity of high interference, shielded cables should be used.
• Wire screen should be connected to the nearest PE point from the power supply side.

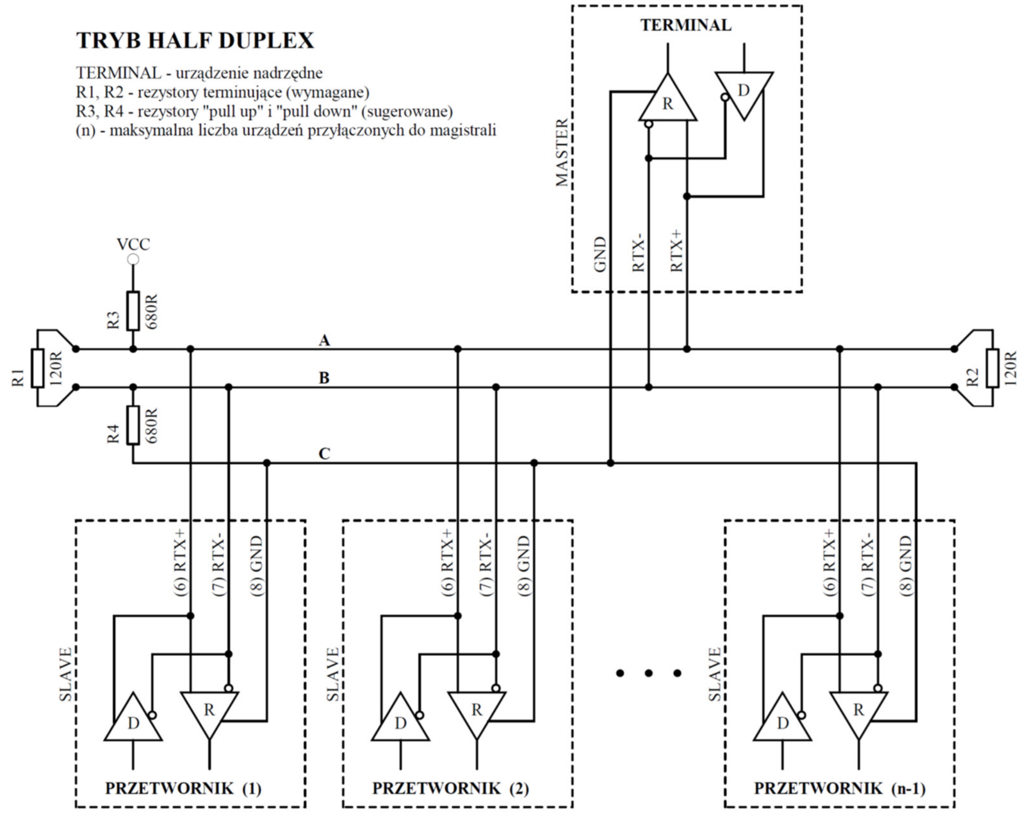

Figure 5. Connection of the transmitter to the RS-485 bus operating in HALF DUPLEX mode.

5 MODBUS protocol

5.1 Map of registers

Registry number	The values	Description
1	-999 - 9999	Pressure difference (limited by measuring range) [Pa] (1 = 1 Pa) with a sign
2	0 - 1000	Pressure difference related to the range (1 = 0.1%; 1000 = 100%)
3	0/1/2/3	Status register (0: "SENSOR OK", 1: "UNDERLOAD", 2: "OVERLOAD", 3: "NO SENSOR") (*)
4	1234	Password register
5	1/2/3	Command register
6	according to the command table	Parameter register
7	0/1	Time constant TAU (0: 0.8s; 1: 4.0s)
8	0/1/2/3/4/5/6	Measurement range (according to the measurement range table)
9	-999 - 9999	Transmitter offset (informative) [Pa] (1 = 1 Pa) with a sign
10	-999 - 9999	Lower value of the measuring range (informative) [Pa] (1 = 1 Pa) with a sign
11	-999 - 9999	Upper value of the measuring range (information) [Pa] (1 = 1 Pa) with a sign
12	0/1	The status of resetting (resetting) the offset (0: not active; 1: pending)
13	0-65535	Counter of valid frames
14	0-65535	Exception counter
15	0-65535	Counter of incorrect CRC
16	0-65535	Counter of erroneous bytes
17	0-65535	Counter of wrong addresses

(*) "SENSOR OK" - correct sensor operation; "UNDERLOAD" - exceeding the range from the bottom;

"OVERLOAD" - exceeding the range from the top; "NO SENSOR" - no sensor;

Table of commands:

Command no	Function	parameters
1	Set the device address	1 - 247 (1-default value)
2	Set the speed transmission	96 - 9600 bps (default) 192 - 19200 b / s 576 - 57600 bps 1152 - 115200 b / s
3	Set the parity bits	0 - NO PARITY; no parity bit (default value) 1 - EVEN PARITY; 2 - ODD PARITY,
4	Set the bits stop	1 - 1 x STOP; 1 stop bit (default value) 2 - 2 x STOP; 2 stop bits
5	Set a constant time	0 - 0.8s; 1 - 4.0s;
6	Set the range measuring	ID according to the measurement range table 0 - 0 ... 6000 Pa (default value)
7	Start calibration process	1 - start of resetting (resetting) the offset
8	reset devices	1 - software reset of the device

Table of measuring ranges:

Range		**Pressure values → voltage output**		
ID	**0 - 6000 [Pa]**	**0V**	**5V**	**10V**
0	0: 6000	0 Pa = 0V	3000 Pa = 5V	6000 Pa = 10V
1	0: 4000	0 Pa = 0V	2000 Pa = 5V	4000 Pa = 10V
2	0: 2500	0 Pa = 0V	1250 Pa = 5V	2500 Pa = 10V
3	0: 2000	0 Pa = 0V	1000 Pa = 5V	2000 Pa = 10V
4	0: 1500	0 Pa = 0V	750 Pa = 5V	1500 Pa = 10V
5	0: 1000	0 Pa = 0V	500 Pa = 5V	1000 Pa = 10V
6	0: 500	0 Pa = 0V	250 Pa = 5V	500 Pa = 10V

Comments:

- Specifying an incorrect or out of range value of the parameter results in entering the value of 0xEEEE in the register of commands.
- Each time a command is called, it must be accompanied by entering the password (1234 decimal).
- Calling a command by individual entries to registers must be completed with entering the password.

5.2 Protocol functions

The following functions of the MODBUS standard have been implemented in the DIFFPRESS v2 converter:

CODE	***IMPORTANCE***
03 (0x03)	*Reading N x 16-bit registers*
16 (0x10)	*Write of N x 16-bit registers*

5.2.1 Reading the contents of output registers (0x03)

The format of the request:

Description	***Size***	***The values***
Device address	*1 byte*	*1 - 247 (0xF7)*
Function code	*1 byte*	***0x03***
Address of the data block	*2 bytes*	*0x0000 - 0xFFFF*
Number of registers (N)	*2 bytes*	*1 - 125 (0x7D)*
CRC checksum	*2 bytes*	*according to calculations*

Response format:

Description	***Size***	***The values***
Device address	*1 byte*	*1 - 247 (0xF7)*
Function code	*1 byte*	***0x03***
Bytes counter	*1 bytes*	*2 x N*
Values of registers	*N x 2 bytes*	*according to the map of registers*
CRC checksum	*2 bytes*	*according to calculations*

The format of the error:

Description	***Size***	***The values***
Device address	*1 byte*	*1 - 247 (0xF7)*
Function code	*1 byte*	***0x83***
Error code	*1 byte*	*0x01 / 0x02 / 0x03 / 0x04*
CRC checksum	*2 bytes*	*according to calculations*

5.2.2 Writing to the group of output registers (0x10)

The format of the request:

Description	***Size***	***The values***
Device address	*1 byte*	*1 - 247 (0xF7)*
Function code	*1 byte*	***0x10***
Address of the data block	*2 bytes*	*0x0000 - 0xFFFF*
Number of registers (N)	*2 bytes*	*1 - 123 (0x7B)*
Bytes counter	*1 byte*	*2 x N*
The values	*N x 2 bytes*	*user*
CRC checksum	*2 bytes*	*according to calculations*

Response format:

Description	***Size***	***The values***
Device address	*1 byte*	*1 - 247 (0xF7)*
Function code	*1 byte*	***0x10***
Address of the data block	*2 bytes*	*0x0000 - 0xFFFF*
Number of registers (N)	*2 bytes*	*1 - 123 (0x7B)*
CRC checksum	*2 bytes*	*according to calculations*

The format of the error:

Description	***Size***	***The values***
Device address	*1 byte*	*1 - 247 (0xF7)*
Function code	*1 byte*	***0x90***
Error code	*1 byte*	*0x01 / 0x02 / 0x03 / 0x04*
CRC checksum	*2 bytes*	*according to calculations*

5.3 Data format

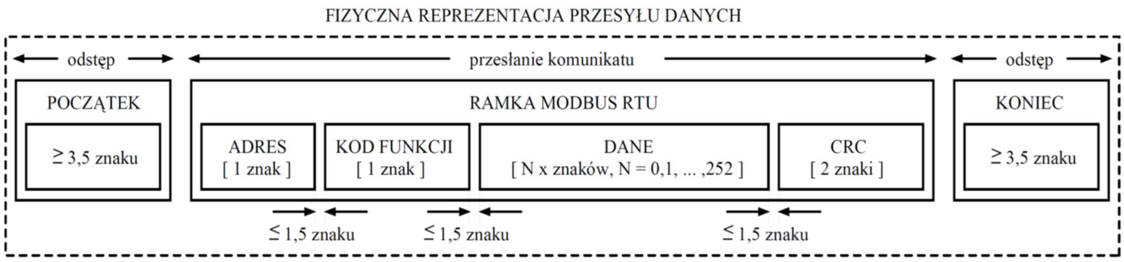

Figure 6. Data transfer in the MODBUS RTU standard implemented in the transducer.

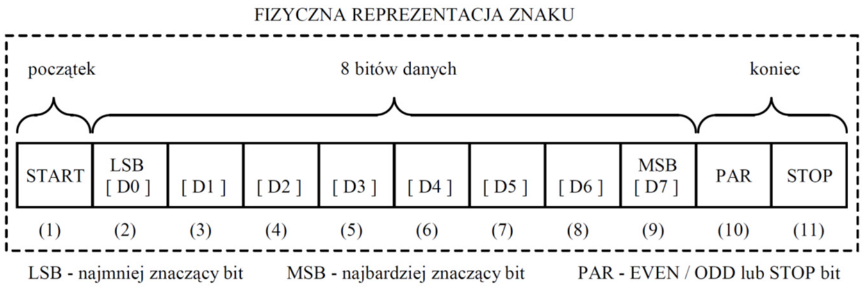

Figure 7. Character format in the MODBUS RTU standard used in the transducer.

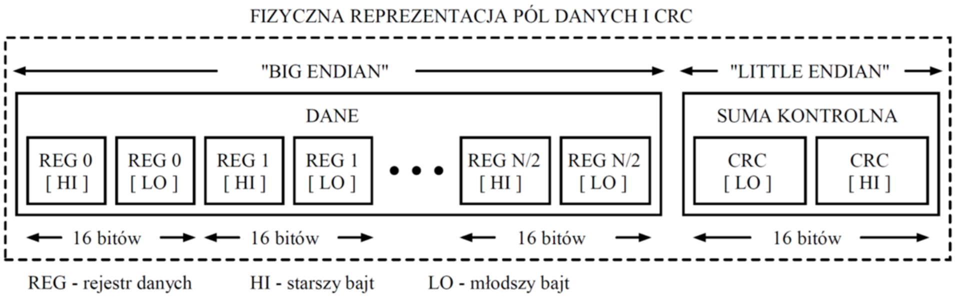

Figure 8. Format of data fields and CRC in the MODBUS RTU standard used in the transducer.

5.4 CRC checksum

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

5.4.1 Bitwise CRC calculation algorithm:

Procedure for determining the CRC checksum using the bit method:
a) loading the value of 0xFFFF into the 16-bit CRC register;
b) taking the first byte from the data block and performing the EX-OR operation with the younger byte of the CRC register, placing the result in the register;
c) shifting the CRC register content to the right one bit in the direction of least significant bit (LSB), resetting the most significant bit (MSB);

d) checking the status of the youngest bit (LSB) in the CRC register, if its status equals 0, then the return to point c takes place, if 1, then the EX-OR operation of the CRC register with the constant 0xA001 is performed;
e) repetition of points c and d to eight times, which corresponds to the processing of the entire byte;
f) repeating the sequence b, c, d, e for the next byte of the message, continue this process until all bytes of the message are processed;
g) the contents of the CRC register after the operations mentioned are the sought-after value of the CRC check sum;
h) adding a CRC checksum to the MODBUS RTU frame must be preceded by swapping of the older and younger bytes of the CRC register.

5.4.2 Table-based CRC calculation algorithm:

An example of the implementation of the procedure for determining the CRC checksum 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
};
```