SPECIFICATION

Differential Pressure Converter DPC250-M, DPC2500-M, DPC4000-M, DPC7000-M (MODBUS RTU)

5/20/2014

1. Introduction

Subject of this document is description of functionalities of the differential pressure converter that is based on the pressure sensor from Honeywell, HSC series, with RS-485 interface and integrated MODBUS RTU Protocol. The sensors measurement range is from ±2.5 to ±40 mbar.

NOTE: Before you start using the module, please read the content of this document.

1.1. Device Functions

- measurement of pressure difference (range depends on the sensor being used)
- configuration of output range
- configuration of measurement time
- constant sensor offset reset function
- device status indication by LED
- RS-485 serial interface (measurement value reading, operating parameters configuration)
  - MODBUS RTU Protocol
  - communication in HALF or FULL DUPLEX mode

1.2. Device Characteristics

Basing function of the DPCxxx-M converter is measurement of pressure difference values. Values measured by the integrated Honeywell sensor HSC, then computed and averaged in the microcontroller, are available in its memory (in HOLDING REGISTERS), in compliance with MODBUS standard. The registers are read using MODBUS Protocol sent through the RS-485 serial interface. The registers also keep information on set (configured) range of measurement, a time constant (also configured) and a percentage value referenced to the range. Indication of the sensor presence, exceeding the measurement range and converter busy status due to offset calibration, are also realized through the status registers.

1.3. Worth To Know

1 hPa = 100 Pa = 1 mbar
1 in H_2O = 249.089 Pa

NOTE: The converter displays pressure difference values in pascals [Pa].

2. Technical Data

2.1. Converter Parameters

Supply	
- direct current	DC 24V (20...30V)
- alternating current	AC 24V (16...26.5V)
Power input	
- minimal [1]	5.5 mA
- typical [2]	7.0 mA
- maximal[3]	11.0 mA
LED indication	0,2 Hz
Installation connector	screw in 3.81mm raster (0.75mm^2)
Dimensions	112 x 84 x 31 (L x H x W)
Weight	-
Installation [4]	
- **Protection**	IP**65**
Operating environment	free of dust, air, neutral gases
Operating temperature	-20 C ÷ 85 C
Storage conditions	
- temperature	-40 C ÷ 85°C
- relative humidity	20 ÷ 60 %RH

1) Average current input at conditions: no transmission; supply 24V DC;
2) Average current input at conditions: FULL DUPLEX transmission 10 queries per second; transmission rate 9600 b/s; simultaneous reading of 20 registers; bus terminating resistors 2 x 120Ω; supply 24V DC;
3) Maximal momentary current input at conditions as in 2): supply 24V DC;
4) The device must be installed by qualified personnel;

2.2. Pressure Difference Measurement Parameters

Sensor type	HSC
Range of measurement	±2,5 ÷ ±40 mbar
Resolution	12 bits
Accuracy:	
- in range 0 ÷ 50 C	±0.25 % of range
- in range -20 ÷ 85 C	not defined
Sampling frequency	100 Hz
Time of response [1]	0.8s / 4s [2]

1) Specified time of response is equal to one time constant that corresponds to 63% of determined value;
2) default value is shorter time of response;

2.3. Serial Interface Parameters

Physical layer	RS-485
Protocol	MODBUS RTU
Connection configurations [1]	HALF / FULL DUPLEX
Transmission rates	9600 / 19200 / 57600 / 115200 b/s

1) HALF DUPLEX – bidirectional communication with one pair of conductors; FULL DUPLEX – bidirectional communication with two pairs of conductors; configuration is changed using a jumper;

3. Installation

3.1. Safety

The device must be installed by qualified personnel!
All connections must be made in accordance with electric diagrams presented in this specification!
Before you turn the device on, check all electric connections!

3.2. Design

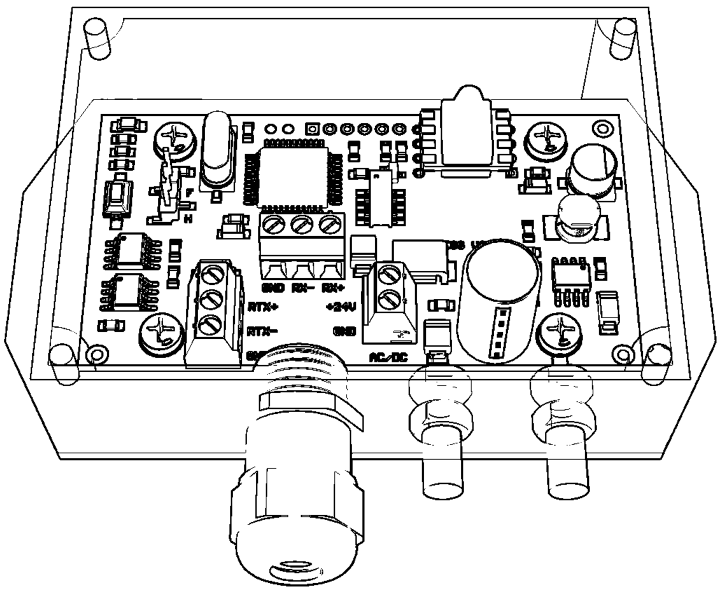
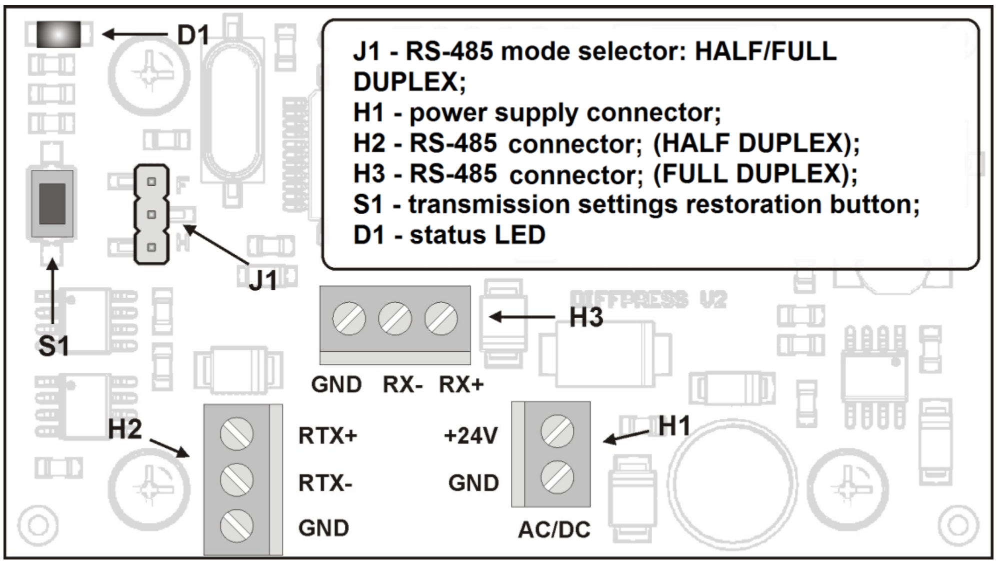


Figure 1. View of printed circuit.

3.3. Board Controls



J1 - RS-485 mode selector: HALF/FULL DUPLEX;
H1 - power supply connector;
H2 - RS-485 connector; (HALF DUPLEX);
H3 - RS-485 connector; (FULL DUPLEX);
S1 - transmission settings restoration button;
D1 - status LED

Figure 2. DPCxxx-M converter board controls

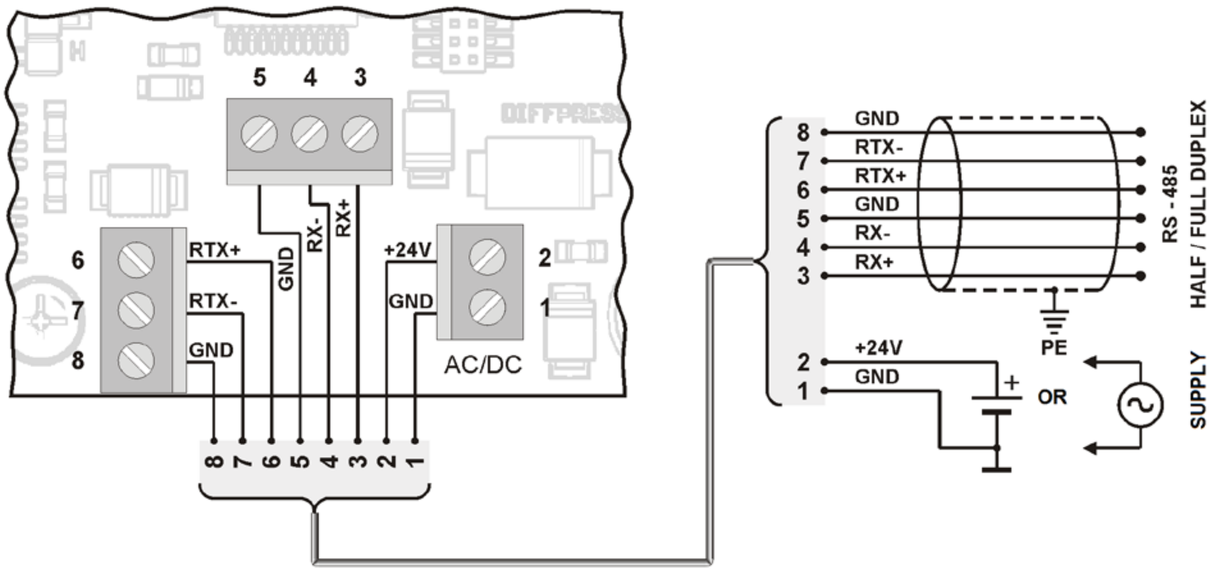


Figure 3. DPCxxx-M converter connections diagram

3.4. Hardware Configuration

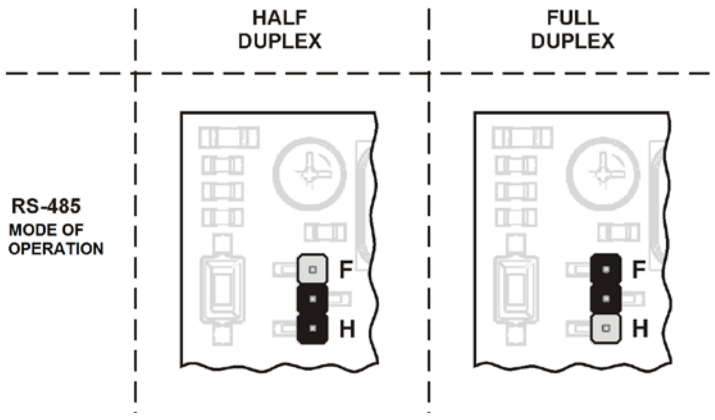


Figure 4. Available RS-485 configurations of DPCxxx-M converter

3.5. Offset Reset

Before you start offset calibration (reset), first set the output range, time constant and two sensor pipes locate in the same pressure (you may disconnect both hoses). Resetting process is started after sending the offset calibration command. Duration of the calibration depends on set time constant and it is respectively 0.8s (less accurate constant) or 4s (more accurate constant). After correct calibration, the device should indicate zero pressure.

3.6. Guidelines

In case of operating at large interferences, use shielded cables. Connect the cable shield to nearest PE point from the side of power unit.

Depending on how the converter is connected to RS-485 bus, select HALF or FULL DUPLEX mode, setting the jumper according to indications on the board.

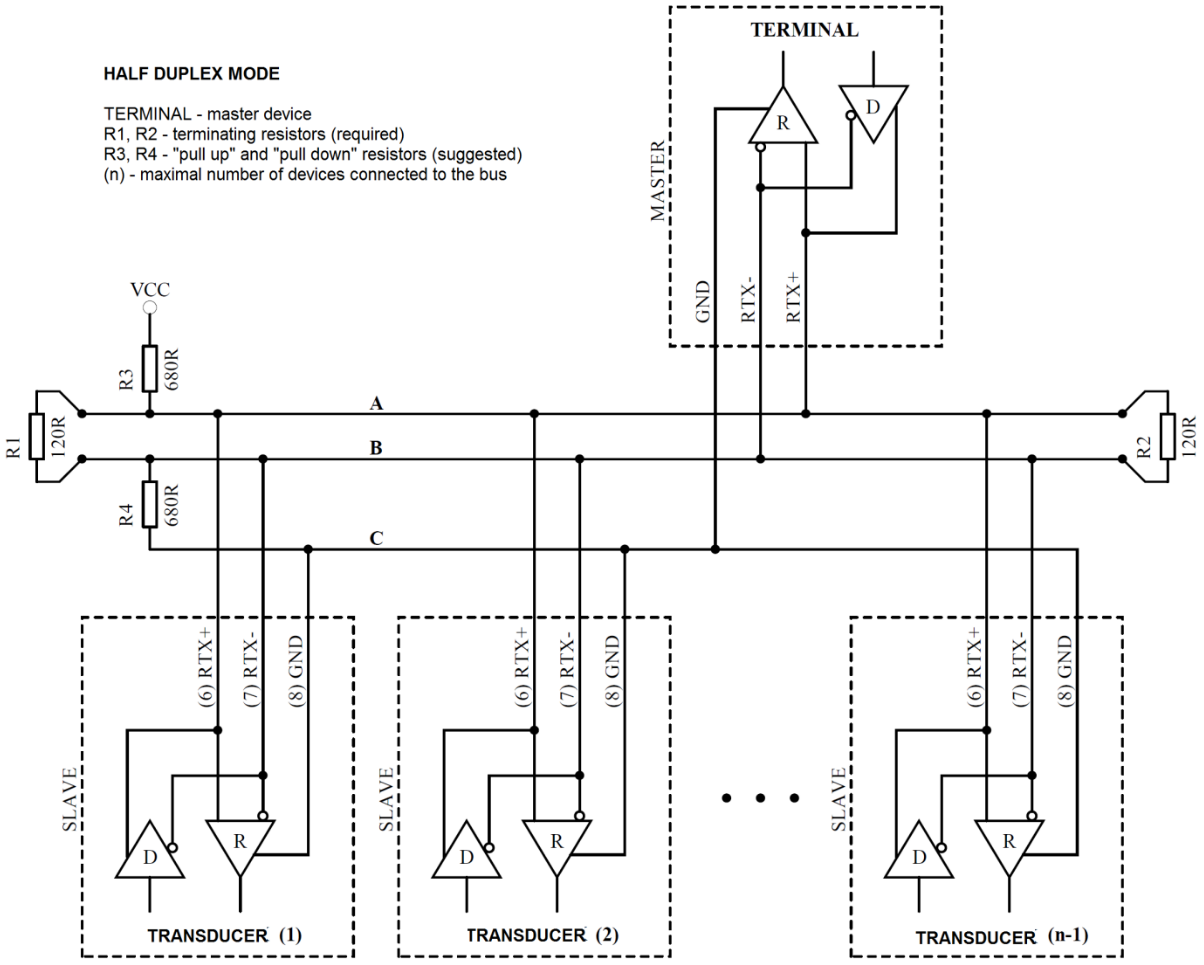


Figure 5. Converter connection to RS-485, operating in HALF DUPLEX mode.

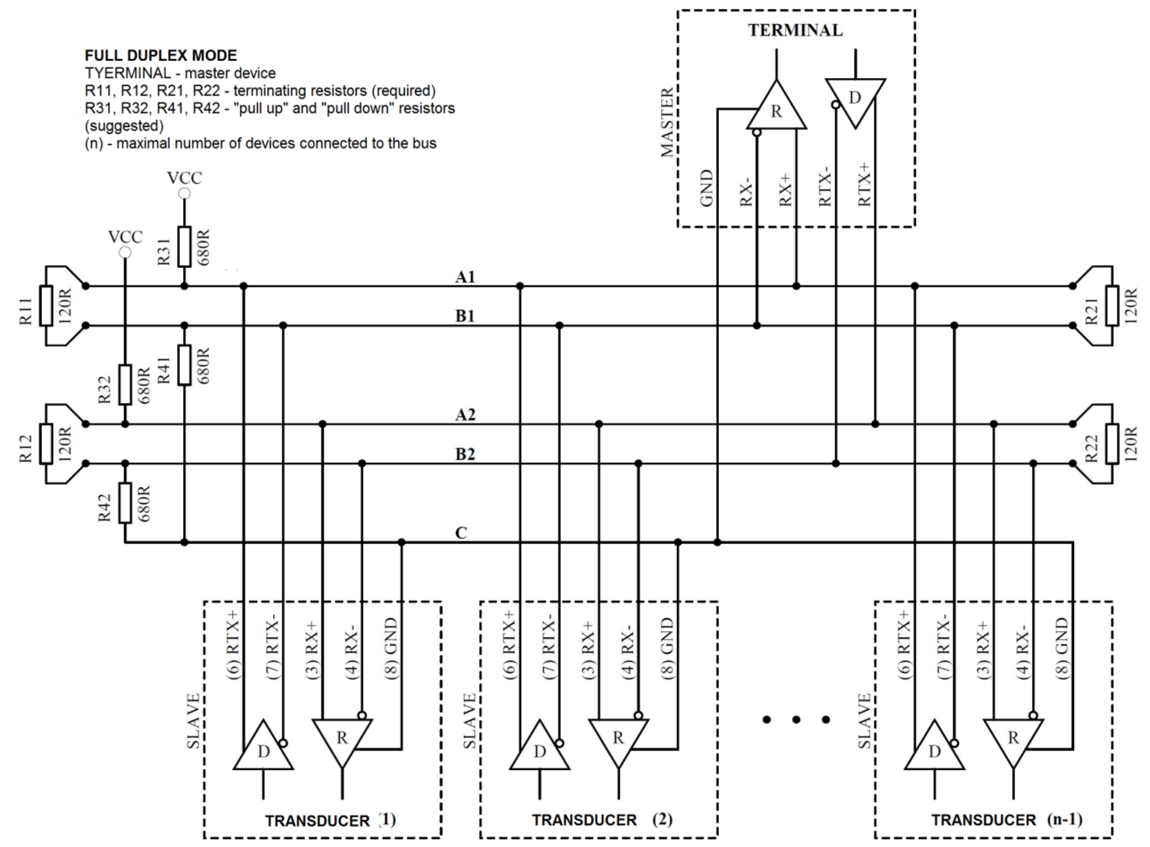


Figure 6. Converter connection to RS-485, operating in FULL DUPLEX mode.

4. MODBUS Protocol

4.1. Registers Map

Registers Table:

Register No.	Values	Description
1	-999 – 9999	Pressures difference (within range of measurement) [Pa] (1 = 1 Pa) with sign
2	0 – 1000	Pressures difference as referred to 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	per command table	Parameter register
7	0 / 1	Time constant TAU (0: 0.8s; 1: 4.0s)
8	0 / 1 / 2 / 3	Range of measurement (according to the table of measurement
9	-999 – 9999	ranges) Converter offset (informative) [Pa] (1 = 1 Pa) with sign
10	-999 – 9999	Lower value of measurement range (informative) [Pa] (1 = 1 Pa) with sign
11	-999 – 9999	Upper value of measurement range (informative) [Pa] (1 = 1 Pa) with sign
12	0 / 1	Offset calibration (reset) status (0: not active; 1: ongoing)
13	0-65535	Counter of correct frames
14	0-65535	Counter of exceptions
15	0-65535	Counter of incorrect CRCs
16	0-65535	Counter of incorrect bytes
17	0-65535	Counter of incorrect addresses
18	-999 – 9999	Pressures difference (not limited with range of measurement) only for service purposes) [Pa] (1 = 1 Pa) with sign

(*) “SENSOR OK” - correct sensor operation; “UNDERLOAD” - exceeded lower range limit; ”OVERLOAD” - exceeded upper range limit; ”NO SENSOR” – sensor is not present;

Table of commands:

Command No.	Function	Parameters
1	Set address of device	1 – 247 (1-default value)
2	Set rate of transmission	96 – 9600 b/s (default setting) 192 – 19200 b/s 576 – 57600 b/s 1152 – 115200 b/s
3	Set bits of parity	0 – NO PARITY; no parity bit 1 – EVEN PARITY; (default setting)
4	Set stop bits	1 – 1 x STOP; 1 stop bit (default setting)

		2 – 2 x STOP; 2 stop bits

5	Set time constant	0 – 0.8s;
		1 – 4.0s;

6	Set measurement range	ID according to the table of measurement ranges
7	Start calibration process	1- start offset calibration (reset)
8	Reset device	1 – device soft reset

Table of measurement ranges:

Range	**Pressure values for each model in Pa**			
ID	**DPC250-M**	**DPC4000-M**	**DPC7000-M**	**DPC2500-M**
0	-100 : 100	0 : 500	0 : 2500	-500 : 500
1	-50 : 50	0 : 1000	0 : 4000	-250 : 250
2	0 : 100	0 : 2500	0 : 5500	0 : 1000
3	0 : 250	0 : 4000	0 : 7000	0 : 2500

Notes:

If you select wrong parameter value or beyond range, the entry to register will be 0xEEEE.

For each call of command you must enter a password (1234 decimal). Call of command by single inputs to registers must be finished with entering a password.

4.2. Protocol Functions

In DPCxxx-M converter, the following functions of MODBUS standard were implemented:

CODE	***MEANING***
03 (0x03)	*Read N x 16-bit registers*
16 (0x10)	*Record N x 16-bit registers*

4.2.1. Reading of Output Registers Group Content (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*	*per computation*

Response format:

Description	*Size*	*Values*
Device address	*1 byte*	*1 – 247 (0xF7)*
Function code	*1 byte*	***0x03***
Byte counter	*1 byte*	*2 x N*
Values of registers	*2 bytes*	*per registers map*
CRC Checksum	*2 bytes*	*per computation*

Error format:

Description	*Size*	*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*	*per computation*

4.2.2. Records in Output Registers Group (0x10)

Request format:

Description	*Size*	*Values*
Device address	*1 byte*	*1 – 247 (0xF7)*
Function code	*1 byte*	***0x10***
Data block address	*2 bytes*	*0x0000 – 0xFFFF*
Number of registers (N)	*2 bytes*	*1 – 123 (0x7B)*
Byte counter	*1 byte*	*2 x N*
Values	*2 bytes*	*of user*
CRC Checksum	*2 bytes*	*per computation*

Response format:

Description	*Size*	*Values*
Device address	*1 byte*	*1 – 247 (0xF7)*
Function code	*1 byte*	***0x10***
Data block address	*2 bytes*	*0x0000 – 0xFFFF*
Number of registers (N)	*2 bytes*	*1 – 123 (0x7B)*
CRC Checksum	*2 bytes*	*per computation*

Error format:

Description	*Size*	*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*	*per computation*

4.3. Data Format

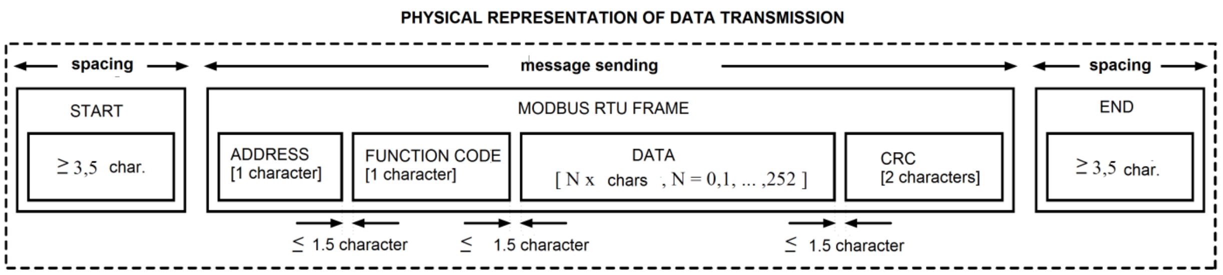


Figure 7. Data transfer in MODBUS RTU standard implemented in the converter.

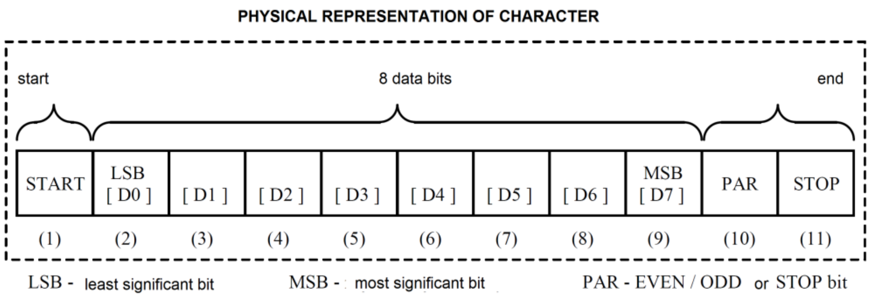


Figure 8. Character format in MODBUS RTU standard used in the converter.

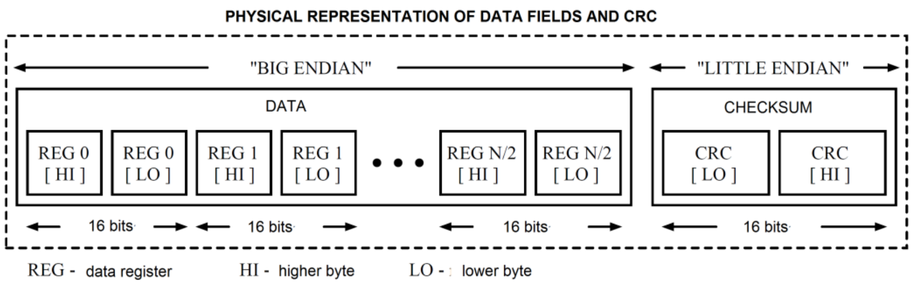


Figure 9. Data fields and CRC format in MODBUS RTU standard used in the converter.

4.4. CRC Checksum

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

4.4.1. Binary Algorithm for CRC Computing:

Determining CRC checksum using binary method:

a) loading 0xFFFF value to 16-bit CRC register;
b) taking first byte from data block and EX-OR operation with lower byte of CRC register, placing the result in register;
c) shifting CRC register content one bit in direction of the least significant bit;
(LSB), resetting the most significant bit (MSB);
d) checking the status of least significant bit (LSB) in CRC register, if it is 0, the process returns to letter c), if it is 1, EX-OR operation is performed for CRC register with 0xA001 constant;
e) repeating letters c) and d) up to eight times, what corresponds to whole byte processing;
f) repeating b), c), d), e) sequences for another byte of message, continuing the procedure until all bytes in the message are processed;
g) after the above mentioned operations are done, CRC register content will be the sought CRC checksum value;
h) adding CRC checksum to MODBUS RTU frame must be preceded by swapping the higher and lower byte in CRC register.

4.4.2. Tabular Algorithm for CRC Computing:

Example of implementing a procedure to determine CRC checksum using tabular 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
};
```