SPECIFICATIONS

Differential pressure transmitter DPC6000_MV
(MODBUS, 0-10V)

Prepared by:
HOTCOLD s.c.

2020-05-13

1. Introduction....3
1.1. Functions....3
1.2. Characteristics....3
1.3. Worth knowing....3
2. Technical data....4
2.1. General parameters....4
2.2. Differential pressure measurement parameters....4
2.3. Analogue output parameters....4
2.4. Serial interface parameters....4
3. Installation....5
3.1. Safety....5
3.2. Unit design....5
3.3. Description of Connections....6
3.4. Analogue output configuration....6
3.5. Address setup....6
3.6. Offset zeroing....7
3.7. Restoring factory settings....7
3.8. Guidelines....7
4. MODBUS protocol....9
4.1. Map of registers....9
4.2. Protocol functions....10
4.2.1. Reading the contents of the output register group (0x03)....11
4.2.2. Entry to the output register group (0x10)....11
4.3. Data format....12
4.4. CRC checksum....13
4.4.1. Bit CRC counting algorithm:....13
4.4.2. CRC counting array algorithm:....13

1. Introduction

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

CAUTION: Before starting the unit, please read the content of this document.

1.1. Functions

- differential pressure measurement (range depending on the sensor used)
- analogue voltage output 0-10 [V] (proportional to pressure difference)
- output range configuration
- measurement time constant configuration
- sensor offset zeroing function
- LED unit status indication
- serial interface RS-485 (reading of measurement values, setup of operating parameters)
    - MODBUS RTU protocol
    - communication in HALF DUPLEX mode
    - hardware configurable address (1-31)

1.2. Characteristics

The basic function of the DPC6000_RSV is to measure differential pressure values. The values measured by the Honeywell ABP series integrated sensor, then recalculated and averaged in the microcontroller, are available in its memory (in HOLDING REGISTERS) according to the MODBUS standard. Reading of registers is performed by means of MODBUS protocol function sent by serial interface RS-485. The registers also contain information about the currently set (configurable) measuring range, the time constant (also configurable) and the percentage of pressure related to the range. Signals like no sensor, exceeded measuring range, transmitter busy during offset calibrations, are also realized by means of status registers.

1.3. Worth knowing

1 hPa = 100 Pa = 1 mbar
1 inH_2O = 249.089 Pa

2. Technical data

2.1. General parameters

Power	
- DC	DC 24V (20...30V)
- AC	AC 24V (21.5...26.5V)
Current consumption	
- minimum [1]	9.0 mA
- typical [2]	11.0 mA
- maximum [3]	22.0 mA
LED signaling	0.2 Hz
Installation connector	screwed in raster 5.00mm (≤ 2.5mm^2)
Dimensions	112 x 84 x 31 (L x H x W)
Weight	approx. 100 g.
Mount [4]	wall mounted
Degree of protection	IP65
Operating environment	dust-free, air, neutral gases
Operating temperature	-20°C ÷ 50°C
Storage conditions	
- temperature	-40°C ÷ 85°C
- relative humidity	20 ÷ 60 %RH

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

2.2. Differential pressure measurement parameters

Sensor type	ABP
Measuring range	up to 7000 Pa
Resolution	12 bits
Accuracy:	
- within 0 ÷ 50°C	±0,25% of range
- within -20 ÷ 85°C	unspecified
Sampling rate	100 Hz
Response time [1]	0.8s/ 4s)

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

2.3. Analogue output parameters

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

2.4. Serial interface parameters

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

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

3. Installation

3.1. Safety

- The unit must be installed by qualified personnel!
- All connections must be made according to the wiring diagrams shown in this specification!
- Check all electrical connections before starting up!

3.2. Unit design

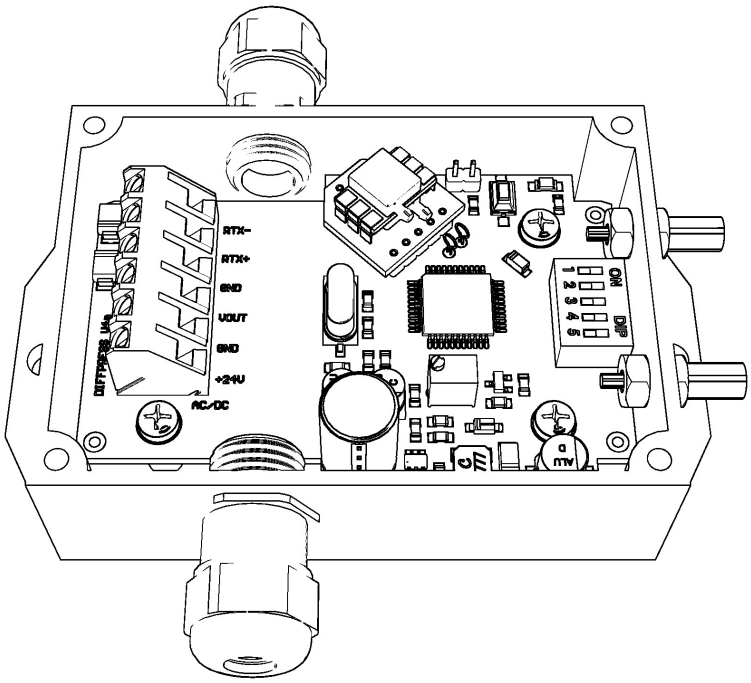

Figure 1. View of PCB.

3.3. Description of Connections

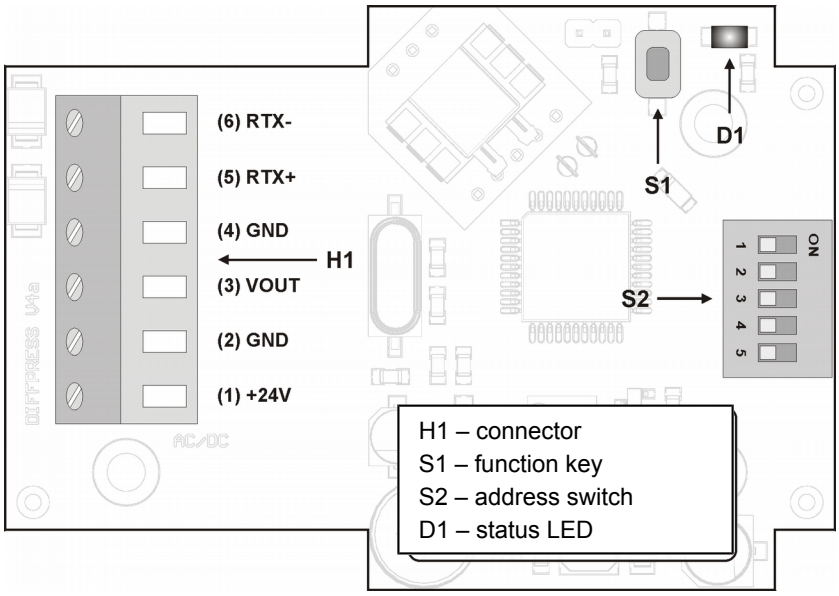

Figure 2. Transmitter leads description.

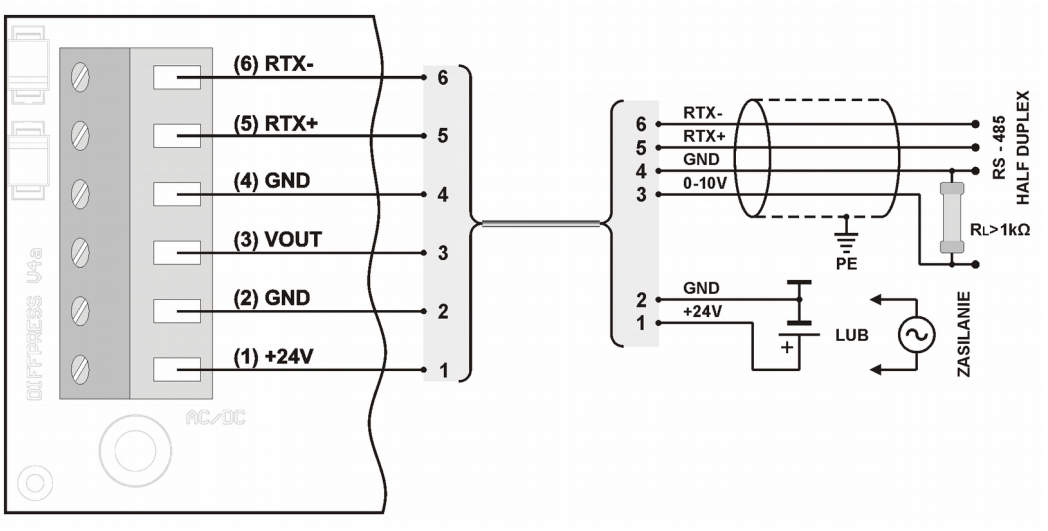

Figure 3. Wiring diagram.

3.4. Analogue output configuration

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

3.5. Address setup

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

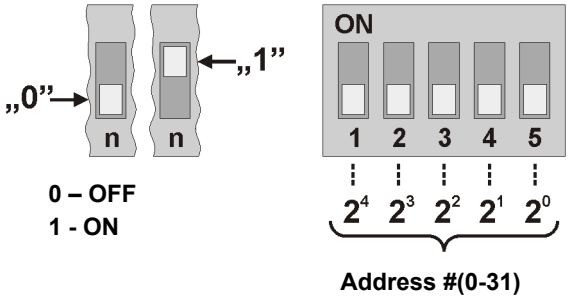

Figure 4. Addressing the transmitter.

3.6. Offset zeroing

Zero command:

Before calibrating (zeroing) the offset, the output range must be set beforehand and both connectors must have the same pressure (you may disconnect both hoses). The zeroing process takes place after sending the offset calibration command. The calibration time is about 7s. After correct calibration the device should indicate zero pressure.

Zero button:

Before calibrating (zeroing) the offset, the output range must be set beforehand and both connectors must have the same pressure (you may disconnect both hoses). To trigger the zeroing process, press and hold S1 button for approximately 3 seconds until D1 starts flashing. The calibration time is counted from the moment the button is released and it takes about 7 seconds. Calibration process is indicated by the blinking of D1 LED. After correct calibration the device should indicate zero pressure.

3.7. Restoring factory settings

The factory settings reset function only applies to the transmission parameters of the RS-485 interface (including address). To restore the settings, press and hold S1 button for about 10 seconds (D1 LED is lit for first 3 seconds, then it blinks for 7 seconds). Release the button when D1 LED is steady lit again. The unit will start working with the new settings automatically.

3.8. Guidelines

- Use shielded cables when working in a highly noised environment.
- Connect the cable screed to the nearest PE point on the power supply side.

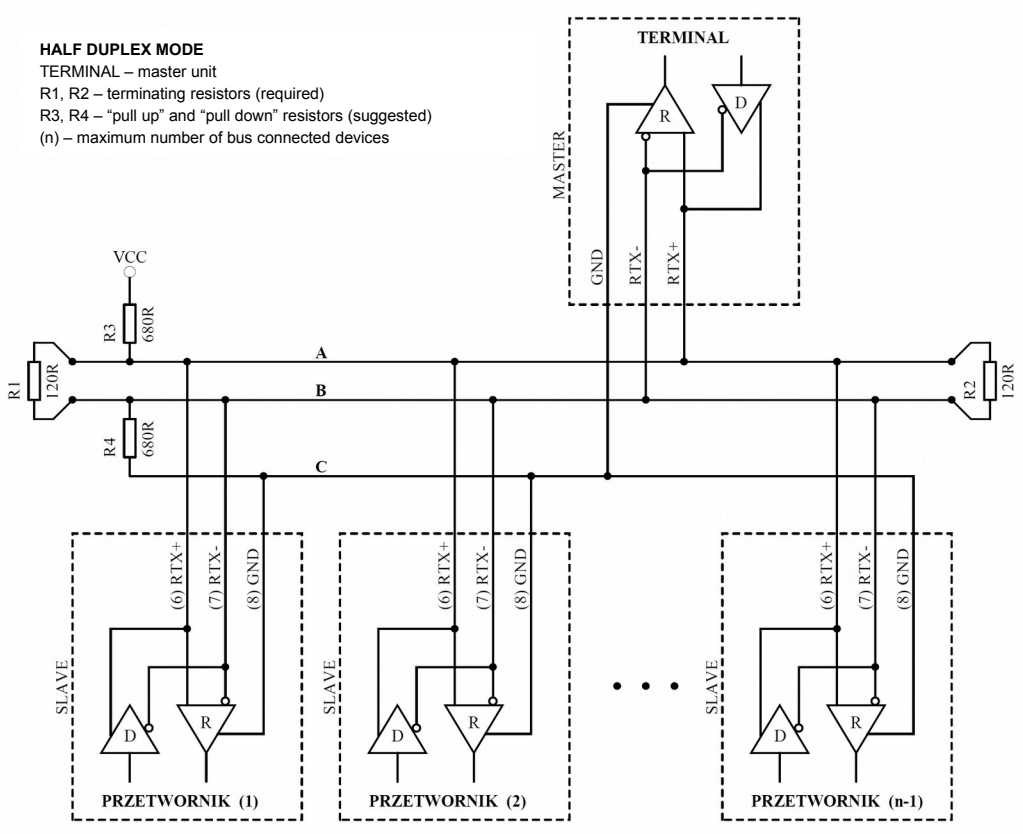

Figure 5. Connection of the transmitter to the RS-485 bus operating in HALF DUPLEX mode.

4. MODBUS protocol

4.1. Map of registers

Table of registers:

Register no.	Values	Description
1	-999 - 9999	Differential pressure (limited by measuring range) [Pa] (1 = 1 Pa) with sign
2	0 - 1000	Differential pressure related to range (1 = 0.1%; 1000 = 100%)
3	0 / 1 / 2 / 3	Status log (0: “SENSOR OK”, 1: “UNDERLOAD”, 2: “OVERLOAD”, 3: “NO SENSOR”) (*)
4	1234	Password log
5	1 / 2 / 3	Command log
6	as per command table	Parameter log
7	0 / 1	TAU time constant (0: 0.8s; 1: 4.0s)
8	0 / 1 / 2 / 3 / 4 / 5 / 6	Measuring range (according to table of measuring ranges)
9	-999 - 9999	Transmitter offset (informative) [Pa] (1 = 1 Pa) with sign
10	-999 - 9999	Lower value of the measuring range (informative) [Pa] (1 = 1 Pa) with sign
11	-999 - 9999	Upper value of the measuring range (informative) [Pa] (1 = 1 Pa) with sign
12	0 / 1	Offset calibration (zeroing) status (0: not active; 1: in progress)
13	0-65535	Counter of correct frames
14	0-65535	Exception counter
15	0-65535	CRC error counter
16	0-65535	Byte error counter
17	0-65535	Address error counter

(*) "SENSOR OK" - correct functioning of the sensor; "UNDERLOAD" - exceeded lower limit;

"OVERLOAD" - exceeded upper limit; 'NO SENSOR - sensor is missing;

Command table:

Command no.	Function	Parameter
1	Set device address	1 - 247 (1-default value)
2	Set transmission rate	96 - 9600 b/s (default value) 192 – 19200 b/s 576 – 57600 b/s 1152 – 115200 b/s
3	Set parity	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 time constant	0 – 0.8s; 1 – 4.0s;

6	Set measuring range	ID according to table of measuring ranges)
7	Start calibration process	1- start offset calibration (zeroing)
8	Device reset	1 - software device reset

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

Notes:

- If you input wrong or out of range parameter value, the value of 0xEEEE shall be entered in the command register.
- Each time the command is called, it must be accompanied by a password (1234 decimal).
- Calling the command through individual registry entries, must be completed by entering the password.

4.2. Protocol functions

The following MODBUS functions are implemented in DIFFPRESS v2 tranmitter:

CODE	*DESCRIPTION*
03 (0x03)	*N reading x 16-bit registers*
16 (0x10)	*N writing x 16-bit registers*

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

Request format:

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

Error format:

Description	Size	Values
Unit address	*1 byte*	*1 – 247 (0xF7)*
Function code	*1 byte*	***0x83***
Error code	*1 byte*	*0x01 / 0x02 / 0x03 / 0x04*
CRC checksum	*2 bytes*	*as calculated*

4.2.2. Entry to the output register group (0x10)

Request format:

Description	Size	Values
Unit 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	*N x 2 user bytes*	
CRC checksum	*2 bytes*	*as calculated*

Response format:

Description	Size	Values
Unit 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*	*as calculated*

Error format:

Description	Size	Values
Unit address	*1 byte*	*1 – 247 (0xF7)*
Function code	*1 byte*	***0x90***
Error code	*1 byte*	*0x01 / 0x02 / 0x03 / 0x04*
CRC checksum	*2 bytes*	*as calculated*

4.3. Data format

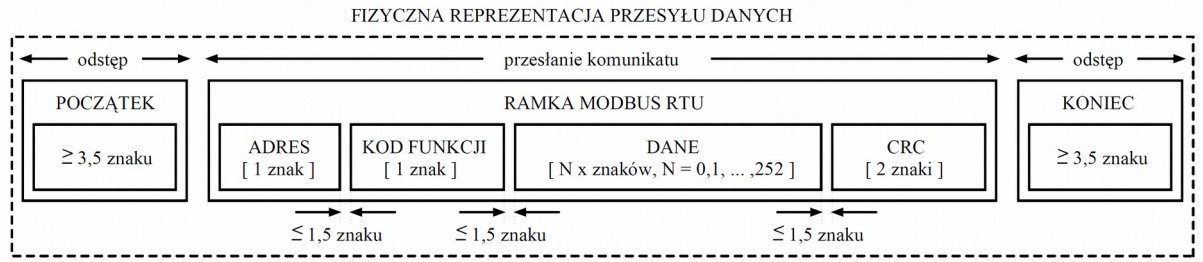

Figure 6. MODBUS RTU standard data transmission implemented in the transmitter.

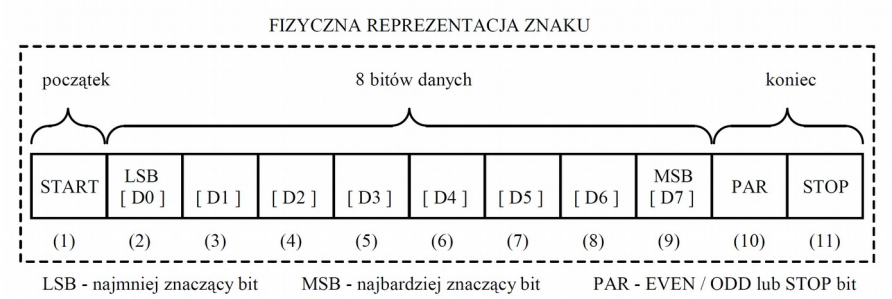

Figure 7. MODBUS RTU standard character format used in the transmitter.

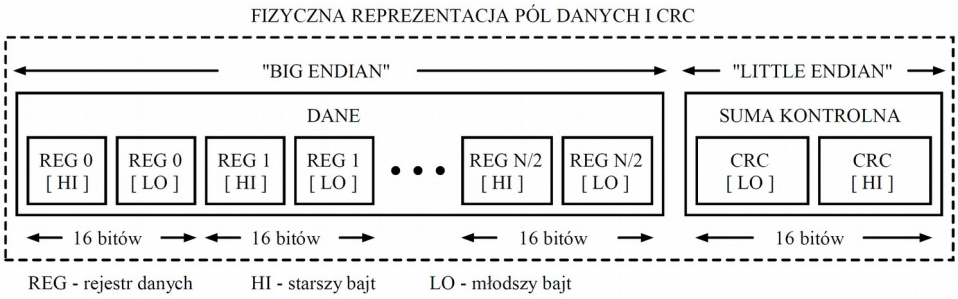

Figure 8. Data field format and CRC in MODBUS RTU standard used in the transmitter.

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. Bit CRC counting algorithm:

Procedure for determining the CRC checksum by bit:

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

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

c) shift contents of the CRC registry to the right by one bit in the direction of the least significant bit (LSB), zeroing of the most significant bit (MSB);

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

e) repeat steps c and d up to eight times, which corresponds to processing the whole byte;

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

g) content of the CRC register after the execution of the listed operations is the value of the CRC check sum sought;

h) addition of the CRC checksum to the MODBUS RTU frame must be preceded by replacing the older and younger byte of the CRC register.

4.4.2. CRC counting array algorithm:

Example procedure for determining the CRC with 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
};
```