SPECIFICATION

HCC-02Ka-Modbus

„Temperature transmitter PT100 ► MODBUS RTU”

Developed:

HOTCOLD s.c.

2023-11-13

1. Introduction

The subject of this study is the characterization of the functionality of a temperature transmitter with a PT100 thermoresistance sensor, with an RS-485 interface, with a built-in MODBUS RTU protocol.

NOTE: Before starting the module, please read the text contained in this study.

1.1. Device functions

- room temperature measurement
- measurement time constant configuration
- LED indication of device operation
- serial RS-485 interface (reading measurement values, configuration of operating parameters)
    - MODBUS RTU protocol
    - communication in HALF DUPLEX mode
    - hardware configured address (1-127)

1.2. Device characteristics

The basic function of the HCC-02Ka-Modbus transmitter is to measure temperature values. The values measured via the PT100 sensor, then converted and averaged in the microcontroller, are available in its memory (in HOLDING REGISTERS registers) in accordance with the MODBUS standard. The registers are read using the MODBUS protocol functions sent via the RS-485 serial interface. The registers also provide information about the measurement range, time constant (configurable) and the percentage of the temperature value related to the range. Signaling of sensor absence/short circuit and states of exceeding the measurement range is also carried out via status registers.

2. Technical data

2.1. General parameters of the HCC-02Ka-Modbus transducer

Power supply	
- constant voltage	DC 24V (20...30V)
- alternating voltage	AC 24V (21,5...26,5V)
Power consumption	
- minimum [1)]	10,0 mA
- typical [2)]	<10,0 mA
- maximum [3)]	10,0 mA
LED signaling	0,2 Hz
Installation connector	screws in a raster 5.00mm (≤ 2,5mm^2)
Dimensions	112 x 84 x 31 (L x H x W)
waight	80 g
Installation [4)]	wall-mounted
Level of protection	IP65
Working environment	dust-free, air, neutral gases
Operating temperature of the electronic system	-30°C ÷ 70°C

1) Average current consumption of the device in conditions: no transmission; 24V DC power supply;

2) Average current consumption of the device in conditions: transmission of 10 queries per second; transmission speed 9600 bps; simultaneous reading of 20 registers; bus terminating resistors 2 x 120Ω; 24V DC power supply;

3) Maximum instantaneous current consumption in the following conditions: signal diode constantly on; other conditions as in point 2);

4) The device should be installed by qualified personnel;

2.2. Temperature measurement parameters

Sensor type	PT100
Measuring range 1	-50°C ÷ 100°C
Measuring range 2	-50°C ÷ 400°C
Resolution 1	14 bitów (0,01 °C)
Resolution 2	12 bitów (0,1 °C)
Accuracy 1	±0,05 °C
Accuracy 12	±0,15 °C
Sampling frequency	15 Hz
Response time [1)]	0,75s / 2.50s [2)]

1) 1) The condition for obtaining 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) 2) The default is faster response time;

2.3. Serial interface parameters

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

A) HALF DUPLEX - bi-directional communication with one pair of wires;

3. Installation

3.1. Security

- The device should be installed by qualified personnel!
- All connections must be made in accordance with the electrical diagrams presented in this specification!
- Before starting, check all electrical connections!

3.2. Device design

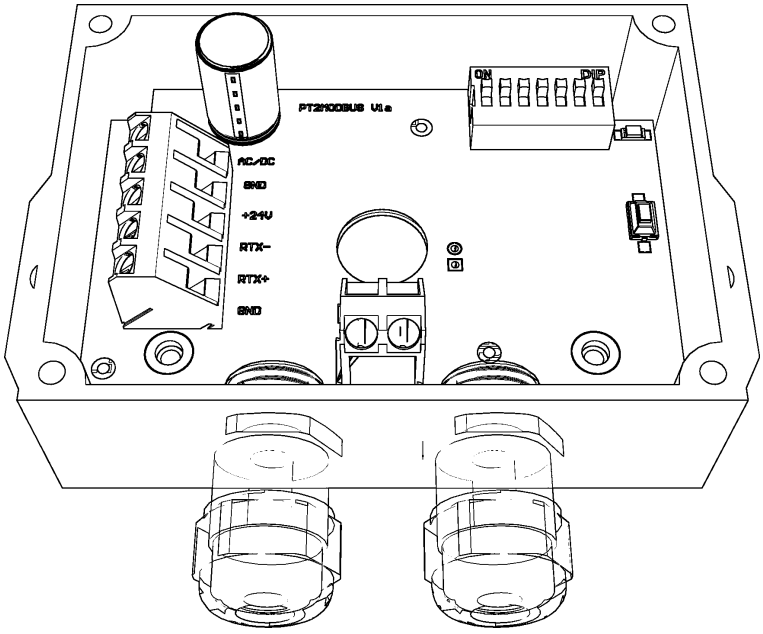

Figure 1. View of the printed circuit of the HCC-02Ka-Modbus transducer

3.3. Pinout description

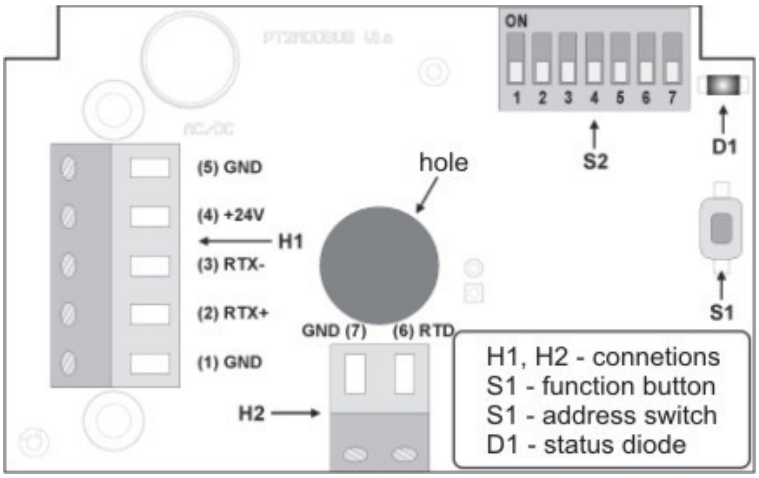

Figure 2. Description of the pinouts of the HCC-02Ka-Modbus transducer

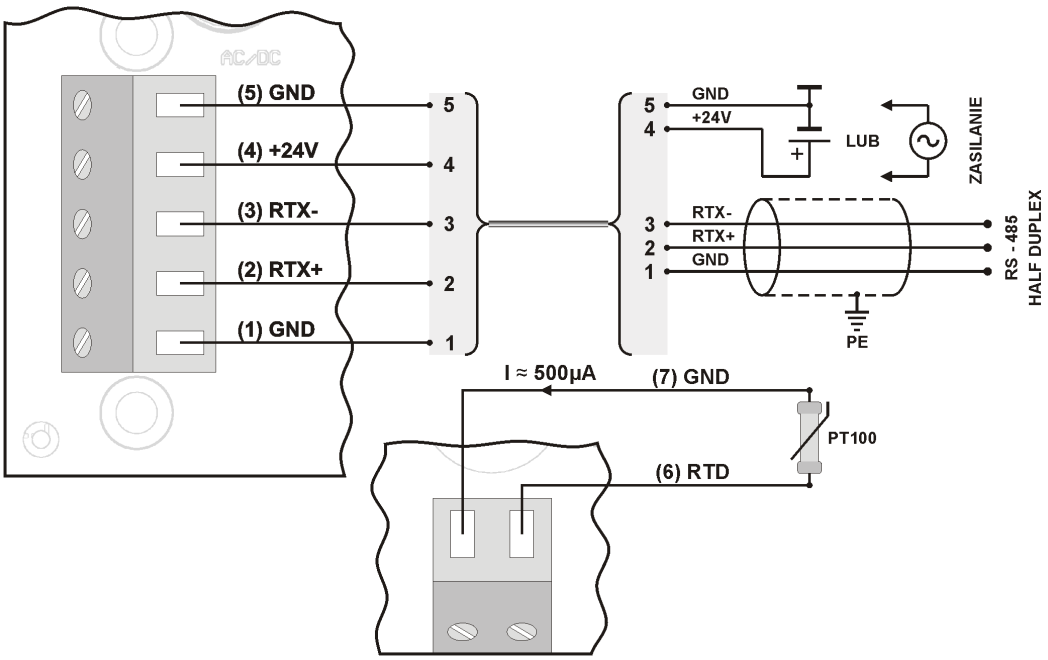

Figure 3. Connection diagram of the HCC-02Ka-Modbus transducer

3.4. Address configuration

The device is equipped with a 7-position switch for hardware address setting (from "1" to "127"). Setting the address "0" on the switch will use the address stored in the device via the MODBUS protocol (default "1").

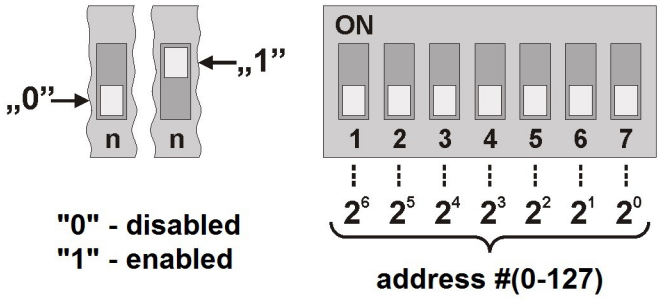

Figure 4. Transducer addressing.

3.5. Restore factory settings

The function of restoring factory settings applies only to the transmission parameters of the RS-485 interface (including the address). 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.

3.6. Guidelines

- When operating in an environment of high interference, shielded cables should be used.
- The cable screen should be connected to the nearest PE point from the power supply side.

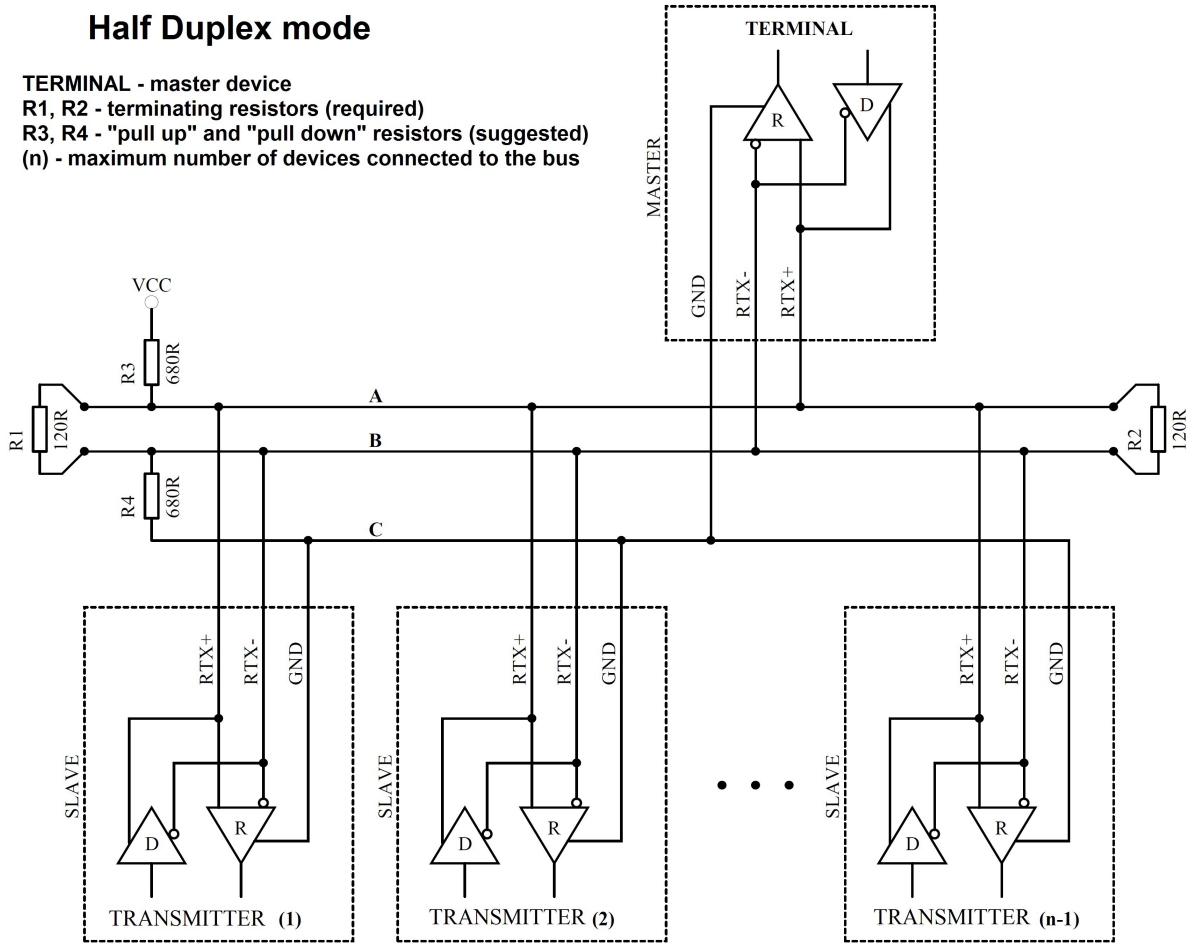

Figure 5. Method of connecting the transducer to the RS-485 bus operating in HALF DUPLEX mode.

4. MODBUS protocol

4.1. Register map

Register table:

Registration No	Values	Description
1 (-50 do 100°C)	-5000 – 10000	Temperature (limited by measurement range) [ºC] (1 = 0.01 ºC) with sign
1 (-50 do 400°C)	-500 – 4000	Temperature (limited by measurement range) [ºC] (1 = 0.1 ºC) with sign
2	0 – 10000	Temperature related to range (1 = 0,01%; 10000 = 100%)
3	0 / 1 / 2 / 3 / 4 / 5	Status register (0: “ADC ERROR”, 1: “OPEN”, 2: “OVERLOAD”, 3: “SENSOR OK”, 4: “UNDERLOAD”, 5: “SHORT”)
4	1234	Password register
5	1 / 2 / 3	Command register
6	wg tabeli poleceń	Parameter register
7	0 / 1	Time constant TAU (0: 0,75s; 1: 2,50s)
8	-	unused
9	-	unused
10	-500	Dolna wartość zakresu pomiarowego (informacyjnie) [ºC] (1 = 0,1 ºC) ze znakiem
11	4000	Upper value of the measurement range (information) [ºC] (1 = 0.1 ºC) with sign
12	0 / 1	Calibration mode (0: not active; 1: active) read-only
13	0-65535	Counter of valid frames
14	0-65535	Exception counter
15	0-65535	CRC invalid counter
16	0-65535	Bad byte counter
17	0-65535	Invalid address counter
18 (-50 do 100°C)	-5000 – 10000	Temperature (not limited by measurement range, for service purposes only) (**) [ºC] (1 = 0.01 ºC) with sign
18 (-50 do 400°C)	-500 – 4000	Temperature (not limited by measurement range, for service purposes only) (**) [ºC] (1 = 0.1 ºC) with sign
19	400 – 2560	Resistance (not limited by measurement range, for service purposes only) [Ω] (1 = 0.1Ω)
20	0 – 2048	Voltage on ADC (not limited by measurement range, for service purposes only) (**) [mV] (1 = 1 mV)

(*) “ADC ERROR” – transducer error; “OPEN” – no sensor; “OVERLOAD” - exceeding the range from above;

“SENSOR OK” - correct operation of the sensor; ”UNDERLOAD” - range exceeded from below;

"SHORT" – sensor short circuit;

(**) “ADC ERROR” -150ºC; “OPEN” 150ºC; "SHORT" -250ºC; (-50 to 100°C)

(**) “ADC ERROR” -150ºC; “OPEN” 500ºC; "SHORT" -250ºC; (-50 to 400°)

Command table:

Command No	Function	Parameters
1	Set the device address	1 – 247 (1-default value)
2	Set the speed transmission	96 – 9600 b/s (default value) 192 – 19200 b/s 384 – 38400 b/s 576 – 57600 b/s 1152 – 115200 b/s
3	Set the parity bits	0 – NO PARITY; no parity bit 1 – EVEN PARITY; (default value) 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 tim	0 – 0,75s; 1 – 2,50s;
6	-	unused
7	-	unused
8	Reset devices	1 – software reset of the device

Notes:

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 entering the password (1234 decimal).

Each command invocation through individual entries in registers must be followed by entering the password.

4.2. Protocol functions

The following MODBUS standard functions are implemented in the transmitter:

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

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

Request format:

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

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 map of registries*
CRC checksum	*2 byte*	*by calculation*

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 byte*	*by calculation*

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*	***0x10***
Data block address	*2 byte*	*0x0000 – 0xFFFF*
Number of registers (N)	*2 byte*	*1 – 123 (0x7B)*
Byte counter	*1 byte*	*2 x N*
Values	*N x 2 bytes*	*of the user*
CRC checksum	*2 byte*	*by calculation*

Response format:

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

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 byte*	*by calculation*

4.3. Data format

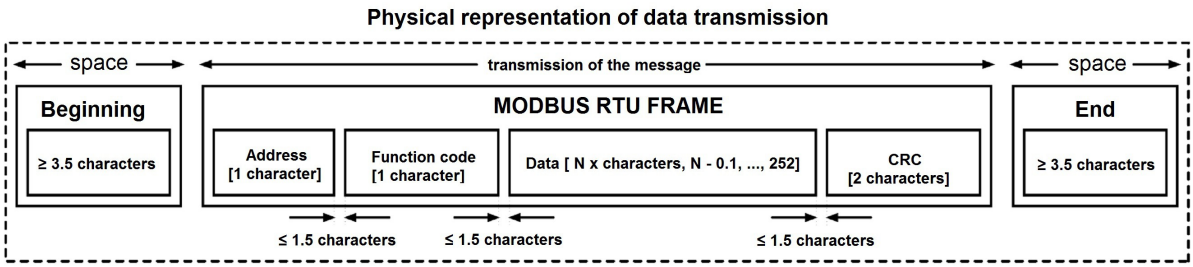

Figure 1. MODBUS RTU standard data transfer implemented in the transmitter.

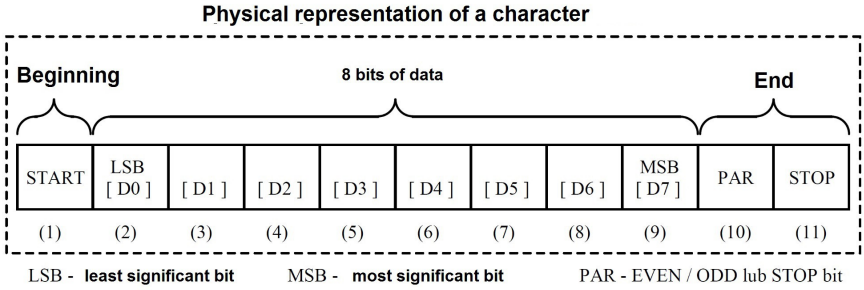

Figure 2. MODBUS RTU standard character format implemented in the transmitter.

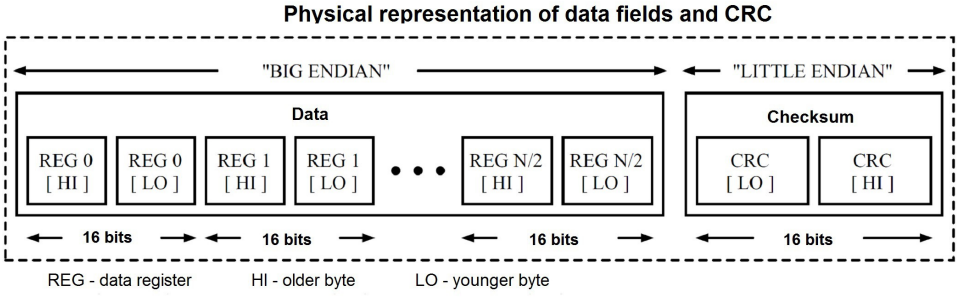

Figure 3. Format of data fields and CRC in MODBUS RTU standard as implemented in the transmitter.

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. Bitwise CRC calculation algorithm:

Procedure for determining the CRC checksum using the bitwise method:

1) loading the value 0xFFFF into the 16-bit CRC register;
2) taking 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;
3) shifting the CRC register contents to the right by one bit towards the least significant bit (LSB), clearing the most significant bit (MSB);
4) checking the state of the youngest bit (LSB) in the CRC register, if its state is 0, then return to the point c; if 1, then the EX-OR operation of the CRC register with the constant 0xA001 is performed;
5) repeat points c and d up to eight times, which corresponds to processing the entire byte;
6) repetition of the sequence b, c, d, e for the next byte of the message, continuation of this process until all bytes of the message have been processed;
7) the content of the CRC register after performing the mentioned operations is the wanted value of the CRC checksum;
8) adding 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. Tabular CRC calculation algorithm:

Example of an 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
};
```