" Concentration transducer CO 2 ► MODBUS RTU "

1. Introduction

The subject of this study is the functional characteristics of the CO 2 concentration transducer based on the TELAIRE 6613 sensor, with the RS - 485 interface with the built-in MODBUS RTU protocol and optionally with the analogue output in the 0 - 10V standard .

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

1.1. Device functions

- measurement of **CO 2** concentration
- analog voltage output 0-10 [V] (proportional to CO 2 concentration)
- LED signaling device operation
- serial RS-485 interface (readout of measurement values, configuration of work parameters)
  - MODBUS RTU protocol
  - communication in HALF DUPLEX mode
  - hardware configurable address (1-127)
  - hardware configurable speed (9600, 19200, 38400, 57600, 115200)

1.2. Characteristics of the device

The basic function of the CO2 2 v2 transducer is to measure the concentration of CO2 in the air. The CO 2 concentration values measured via the TELAIRE 6613 integrated sensor are then calculated and averaged in the microcontroller, they are available in its memory (in the HOLDING REGISTERS registers) according to the MODBUS standard. The registers are read using the MODBUS protocol functions sent over the RS-485 serial interface. The signaling of sensor absence / error, overmeasuring range is carried out via status registers. The values can also be presented in analog form on the voltage output in the 0-10V standard.

2. Specifications

2.1. General parameters of the transducer

Power	
- constant voltage	DC 24V (20 ... 30V)
- alternating voltage	AC 24V (20 ... 27,6V)
Power consumption	
- typical 1)	<35.0 mA
- maximum 2)	<70.0 mA
LED signaling	description in the section "LED signaling"
Installation connector	screw in 5.00 mm pitch (≤ 2.5mm ^{2})
dimensions	115 x 65 x 55 (L x H x W)
Weight	150g
Assembly 3)	
Working environment	dust-free , air, neutral gases
Working temperature	0 ° C ÷ 50 ° C

1) Average device current consumption in the following conditions: transmission of 10 queries per second; transmission speed 9600 b / s; simultaneous reading of 3 registers; bus terminating resistors 2 x 120 Ω; 24V DC power supply, voltage output with a 10k resistive load;

2) Maximum momentary current consumption of the device in conditions as in point 1) + voltage output with a 1k resistive load;

3) The device should be installed by qualified personnel;

2.2. Parameters of CO 2 measurement

Sensor type	TELAIRE 6613
Measurement range	0 ÷ 2000 ppm
Accuracy:	
- in the range of 400 ÷ 1250 ppm	± 3%
- in the range of 1250 ÷ 2000 ppm	± 5% ± 30 ppm
Sampling frequency	2 Hz
Response time [1]	<2 minutes

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

2.3. Parameters of the analog output

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

2.4. Parameters of the serial interface

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 with one pair of wires;

3. Installation

3.1. Security

- The device should be installed by qualified personnel!
- All connections should be made in accordance with the wiring diagrams set out in this specification!
- Before commencing the commissioning, check all electrical connections!

3.2. The construction of the device

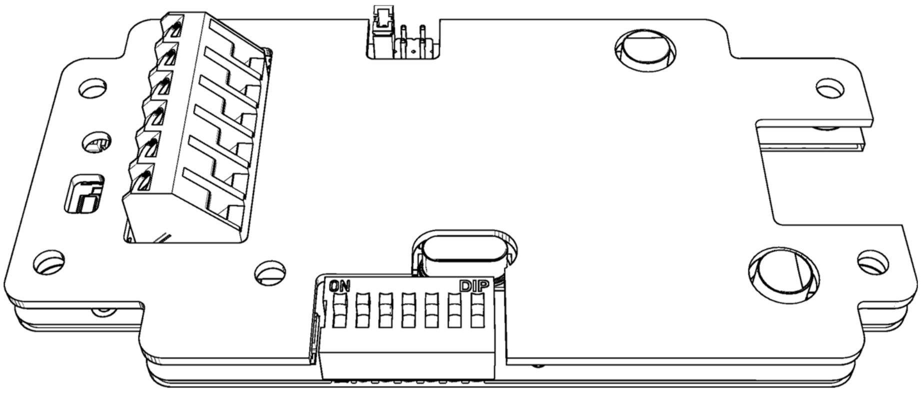

Figure 1. View of the printed circuit version of the **channel** transmitter.

3.3. Description of leads

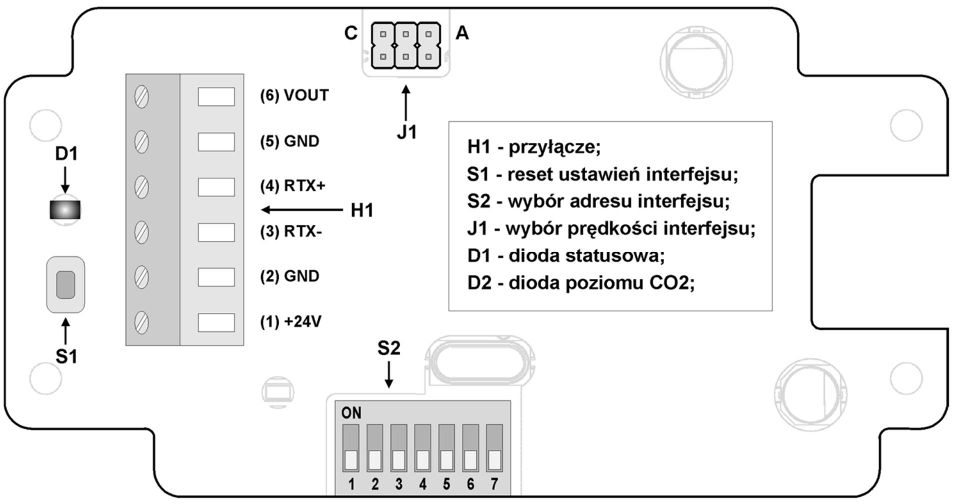

Figure 2. Description of CO 2 converter outputs in the channel version.
H1-connection pins
S1-reset of interface setting
S2- setting of interface address
J1- setting of interface speed
D1- status diode
D2- CO2 level diode

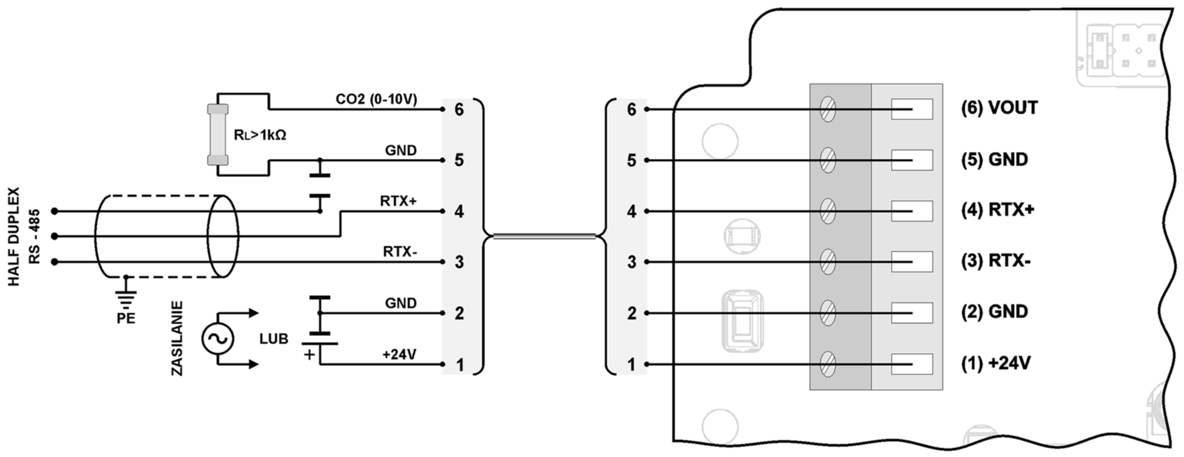

Figure 3. Connection diagram of the CO 2 transducer in the channel version.

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 MOD-BUS protocol ("1" by default).

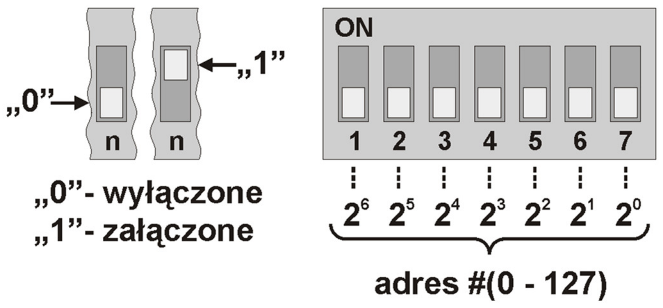

Figure 4. Transducer addressing.
“0”- off
“1” - on

3.5. Speed configuration

The device is equipped with a system of 3 jumpers for hardware determination of the RS - 485 interface speed (according to the table below). No jumpers will use the speed value stored in device via the MODBUS protocol (default "9600 bps").

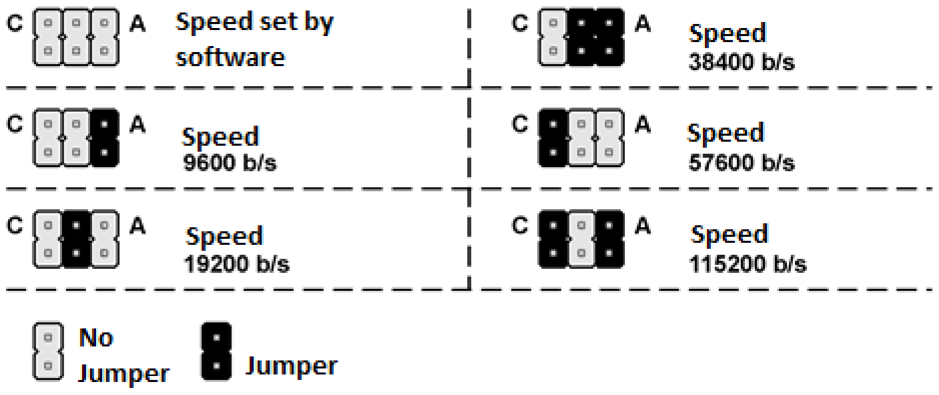

Figure 5. Configuration of the RS-485 interface speed.

3.6. Restoring factory settings

The function of restoring factory settings applies only to parameters of RS-485 interface transmission (including address and speed). To restore the settings, press and hold down the S1 button for about 2 seconds (protection against accidental operation). When the diode D1 blinks, release the button. The device will start working with new settings automatically.

4.

4.1. LED signaling

Table of levels / statuses indicated on diode D2:

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

(*) The hysteresis for switching the diode status is ± 50 ppm .

(**) Blinking (XXX ms / YYY ms) means XXX - ON time , YYY - OFF time

4.2. Guidelines

- In the case of work in the vicinity of large interferences, shielded cables should be used.
- The wire screen should be connected to the nearest PE point from the power supply side.

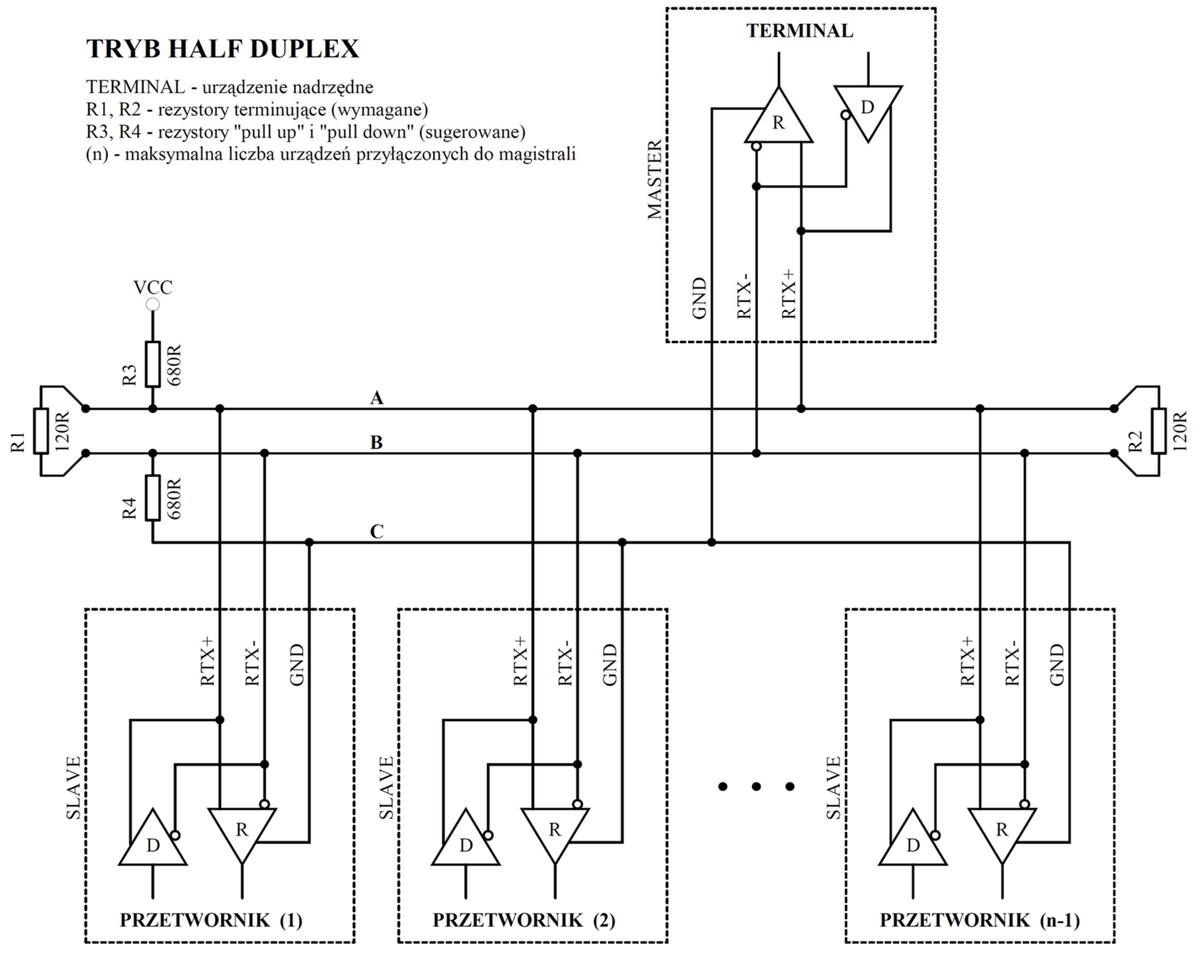
TRYB HALF DUPLEX
TERMINAL - urządzenie nadrzędne
R1, R2 - rezystory terminujące (wymagane)
R3, R4 - rezystory "pull up" i "pull down" (sugerowane)
(n) - maksymalna liczba urządzeń przyłączonych do magistrali
TERMINAL
MASTER
R
D
GND
RTX-
RTX+
VCC
R3
680R
R1
120R
A
B
C
R4
680R
R2
120R
SLAVE
RTX+
RTX-
GND
D
R
PRZETWORNIK (1)
PRZETWORNIK (2)
PRZETWORNIK (n-1)

Figure 6. Connection of the transmitter to the RS-485 bus operating in HALF DUPLEX mode.

5. MODBUS protocol

5.1. Map of registers

Table of registers:

Registry number	The values	Description
1	400 - 2000	CO2 concentration [ppm] (1 = **1** ppm)
2	0/1/2/3/4/5/6	Status register (0: "NO SENSOR", 1: "SENSOR OK", 2: "ERROR", 3: "WARM UP", 4: "CALIBRATION", 5: "IDLE", 6: "OVER RANGE **(*)**
3	1000 (0x03e8)	Test value - to verify the correctness of reading registers
4	1234	Password register
5	1/2/3	Command register
6	according to the command table	Parameter register
7	0-65535	Counter of valid frames
8	0-65535	Exception counter
9	0-65535	Counter of incorrect CRC
10	0-65535	Counter of erroneous bytes
11	-	not used
12	400 - 5000	only for service purposes

(*) "NO SENSOR" - no sensor; "SENSOR OK" - proper sensor operation; "ERROR" - sensor error; "WARM UP" - sensor during heating; "CALIBRATION" - sensor during calibration; "IDLE" - sensor in sleep mode; "OVER RANGE" - exceeding the measuring range;

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 384 - 38400 bps 576 - 57600 bps 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 the facade	0 - 2500 [m above sea level] (height above sea level)
105	Read the façade	JW.
6	reset devices	1 - software reset of the device

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 through individual entries to registers must be completed by entering the password.

5.2. Protocol functions

The transmitter has implemented the following functions of the MODBUS standard:

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

5.2.1. Reading the contents of the group 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*

Error format:

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*

Error format:

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 7. Data transfer in MODBUS RTU standard implemented in the transmitter.

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

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

5.4. 5.4. CRC checksum

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

5.4.1. 5.4.1. Bitwise CRC calculation algorithm :

Procedure for determining the CRC checksum using the bit method:
and) loading the value 0xFFFF into the 16-bit CRC register;
b) getting the first byte from the data block and performing the EX-OR operation from a younger byte of the CRC register, placing the result in the register;
c) shifting the CRC register content to the right by 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 follow the return to point c, if 1, then the EX-OR operation of the CRC register with constant 0xA001;
e) repeating c and d points to eight times, which corresponds to processing 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 content of the CRC register after performing the above mentioned operations is the wanted value of the CRC check sum;
h) adding a CRC checksum to the MODBUS RTU frame must be preceded by a replacement of the older and younger bytes of the CRC register.

5.4.2. 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 * /
        unsi gned 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, 0x 41, 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, 0 x03, 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
};
```