CRYSOUND

CRY2120 Noise Sensor

User Manual V1.0

Directory

1 Product Brief Introduction

CRY2120 noise sensor is a new type of industrial-grade noise sensor, comply with IEC 61672-1 standard.

CRY2120 noise sensor integrates measurement microphone, preamplifier, high-precision ADC, high-speed DSP processor and excellent hardware design which provides 110dB dynamic range and can measure lower to 25dBA environment noise. The sensor has a compact structure with small appearance, the stainless-steel shell gives a perfect corrosion resistance.

Fig 1.1 CRY2120 Noise Sensor

CRY2120 Modbus version uses RS-485 bus interface as the data communication port, with communication protocol based on Modbus RTU. Also, to facilitate industrial automation applications, it also provides 4-20mA current or 2-10V voltage output interface.

2 Features

[Perform Standard] IEC61672 Class 1

[Two Output Methods] 4-20mA and RS485 output, transmission distance exceeds 1km

[Easy for Integration] Connect to PC, PLC, DCS and other controllers, get noise data easily

[Perfect Performance] 110dB dynamic range, can measurement lower to 25dBA noise

[Small Appearance] Small size with 25mm diameter and 115mm length

[Wide Range Power Supply] From DC 5V to 24V

[Reliable and Durable] Stainless steel shell, robust in harsh environments

3 Typical Application

Fig 3.1 System Integrated Noise Monitoring

Fig 3.2 Construction Noise Monitoring

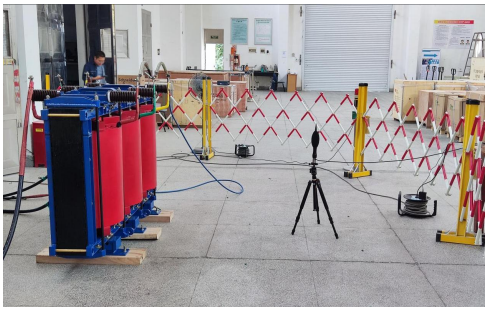


Fig 3.3 Product Noise Loudness Monitoring

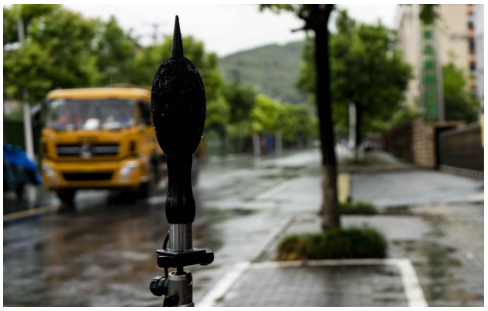


Fig 3.4 Urban Traffic Noise Monitoring

4 Technical Specifications

CRY2120 Modbus Technical Parameters	
Model	CRY2120
Applicable standards	IEC 61672-Class 1
Standard Microphone	CRY333
Optional Microphone	CRY311 (1/2” to 1” adaptor is required)
Measurement Range	25~130 dBA (Related to the microphone)
Dynamic Range	≥ 110 dB, without switch range
AD Sampling Rate	48k Hz
Detection Method	Fully digital
Background Noise	19 dB(A), 21 dB(C), 27 dB(Z)

Frequency Range	10 Hz ~ 20k Hz
Frequency Weighting	A / C / Z
Time Weighting	F / S
Measurement Function	Lxy (x=A/C/Z, y=F/S)
SPL Output	RS-485 & 4-20mA / 1-5V / 2-10V (Choose one of the three)
Power Supply	DC 5-24V (Use high voltage power supply when use analog outputs)
Dimensions	ϕ24.5mm × 111mm
Weight	115 g
Operating Conditions	-10 to +50 ℃; RH ≤ 90% (without condensation)

5 Modbus RTU Communication Protocol

The communication protocol of CRY2120 (Modbus) noise sensor is based on the protocol layer of Modbus RTU bus protocol.

5.1 Modbus Register Definition

There are 31 addresses data is stored in the holding registers, which can use function code 03 to read and 06 to write. To ensure the accuracy of the data and the reliability of the device, we disable registers outside the address range of form2. Reading or writing these registers will return the corresponding error code. For example, when sensor address is 01, probably it will return error code:"0x01 0x83 0x02 0xC0 0xF1".

CRY2120 noise sensor register distribution:

Holding Register(function code 0x03 to read, function code 0x06 to write)							
0x0001		0x0002	0x0003		0x0004		
0x00	address	Invalid write, read 0x0000	Invalid write, read 0x0000		Invalid write, read 0x0000		
0x0005			0x0007		0x0008		
Invalid write, read 0x0000			0x00	Frequency Weighting	0x00	Time Weighting	
SPL data register and device information register (read only)							
0x0101		0x0102	0x0103		0x0104		
LXY		1/3Oct-25Hz	1/3Oct-31.5Hz		1/3Oct-40Hz		
0x0105		0x0106	0x0107		0x0108		
1/3Oct-50Hz		1/3Oct-63Hz	1/3Oct-80Hz		1/3Oct-100Hz		
0x0109		0x010A	0x010B		0x010C		
1/3Oct-125Hz		1/3Oct-160Hz	1/3Oct-200Hz		1/3Oct-250Hz		
0x010D		0x010E	0x010F		0x0110		
1/3Oct-315Hz		1/3Oct-400Hz	1/3Oct-500Hz		1/3Oct-630Hz		
0x0111		0x0112	0x0113		0x0114		
1/3Oct-800Hz		1/3Oct-1000Hz	1/3Oct-1250Hz		1/3Oct-1600Hz		
0x0115		0x0116	0x0117		0x0118		
1/3Oct-2000Hz		1/3Oct-2500Hz	1/3Oct-3150Hz		1/3Oct-4000Hz		
0x0119		0x011A	0x011B		0x011C		
1/3Oct-5000Hz		1/3Oct-6300Hz	1/3Oct-8000Hz		1/3Oct-10000Hz		
0x011D		0x011E	0x011F		0x0120		
1/3Oct-12500Hz		1/3Oct-16000Hz	1/3Oct-20000Hz		Load State (Not add)		

Device information (read only)		
0x0201	……..	0x020C
Device information “CRY2120 Modbus V1.0”		
Coil Register (function code 0x01 to read, function code 0x05 to write)		
0x0002		
Calibration Trigger		

Note:

Frequency Weighting Flag: 0x00 A-weighting (Default), 0x01 C-weighting, 0x02 Z-weighting;

Time Weighting Flag: 0x00 F-weighting (Default), 0x01 S-weighting;

5.2 CRY2120 Modbus RTU Command Instructions and Responses

5.2.1 Function Code 03 Command

8-bit address + 8-bit function code + 16-bit address of read starting register + 16-bit read register number + 16-bit CRC check bit.

Response:

8-bit address + 8-bit function code + 8-bit bytes number +several 16-bit data groups + 16-bit CRC check bit.

5.2.2 Function Code 06 Command

8-bit address + 8-bit function code + 16-bit address of read register + 16-bit CRC check bit

Response:

8-bit address + 8-bit function code + 16-bit address of read register + 16-bit CRC check bit.

Note: In the segment, high-bit in front, low-bit in behind. CRC check value is low-bit in front, high-bit in behind.

5.3 CRY2120 Modbus RTU SPL Data Analyzing

Holding Register	Address 01XX HiByte	Address 01XX LowByte
Definition	SPL data HiByte	SPL data LowByte

SPL conversation formula:

SPL= (HiByte*256+ LowByte)/10

In above formula:

HiByte: Modbus register data high 8 expressed as a decimal value

LowByte: Modbus register data low 8 expressed as a decimal value

Address data in hexadecimal, sensor actually use only LowByte as device address, and HiByte data is stored as 0x00 by default.

Holding Register	Address 0001 HiByte	Address 0001 LowByte
Definition	Keep (read 0x00)	8-bit address

5.4 CRY2120 Modbus RTU Data Communication Example

E.g.1: Sensor in address 01 returned SPL(LXY)1:

Host send: 01 03 01 00 00 01 85 F6

Host send data	Hexadecimal	Note
Slave address	01	
Function code	03	
Register address HiByte	01	
Register address LowByte	00	
Register address HiByte	00	

Register address LowByte	01	
CRC check LowByte	85	
CRC check HiByte	F6	

Slave response: 01 03 02 00 CD 79 D1

Slave response data	Hexadecimal	Note
Slave address	01	
Function code	03	
Returned data number	02	Returned 2 groups of 8-bit data
Register address HiByte	00	
Register address LowByte	CD	
CRC check LowByte	79	
CRC check HiByte	D1	

E.g.2: Sensor in address 01 returned the target calibrated SPL, frequency weighting and time weighting

Host send: 01 03 00 05 00 03 15 CA

Host send data	Hexadecimal	Note
Slave address	01	
Function code	03	
Register address HiByte	00	
Register address LowByte	05	
Register address HiByte	00	

Register address LowByte	03	
CRC check LowByte	15	
CRC check HiByte	CA	

Slave response: 01 03 06 03 AB 00 00 00 00 04 9E

Slave response data	Hexadecimal	Note
Slave address	01	
Function code	03	
Returned data number	06	Returned 6 groups of 8-bit data
Register address HiByte	03	
Register address LowByte	AB	
Register address HiByte	00	
Register address LowByte	00	
Register address HiByte	00	
Register address LowByte	00	
CRC check LowByte	04	
CRC check HiByte	9E	

E.g.3: Set sensor in unknown address to address in 2

Host send: 00 06 00 00 00 02 09 DA

Host send data	Hexadecimal	Note
Slave address	00	

Function code	06	
Register address HiByte	00	
Register address LowByte	00	
Register address HiByte	00	
Register address LowByte	02	
CRC check LowByte	09	
CRC check HiByte	DA	

Slave no response:

E.g.4: Set sensor address in 1 to address in 2

Host send: 01 06 00 00 00 02 08 0B

Host send data	Hexadecimal	Note
Slave address	01	
Function code	06	
Register address HiByte	00	
Register address LowByte	00	
Register address HiByte	00	
Register address LowByte	02	
CRC check LowByte	08	
CRC check HiByte	0B	

Slave response: 01 06 00 00 00 02 08 0B

Slave response data	Hexadecimal	Note

Slave address	01	
Function code	06	
Register address HiByte	00	
Register address LowByte	00	
Register address HiByte	00	
Register address LowByte	02	
CRC check LowByte	08	
CRC check HiByte	0B	

6 Connection and Communication of Serial Port

6.1 Communication Parameters of RS-485 Interface

RS-485 serial port communication protocol:

Baud rate	38400	Data bit	8
Parity bit	None	Stop bit	1
Data format	Hexadecimal		

6.2 RS-485 to USB Connector (optional)

Use optional connector which has been supplied 5V power, standard cable constitutes the following connection mode.

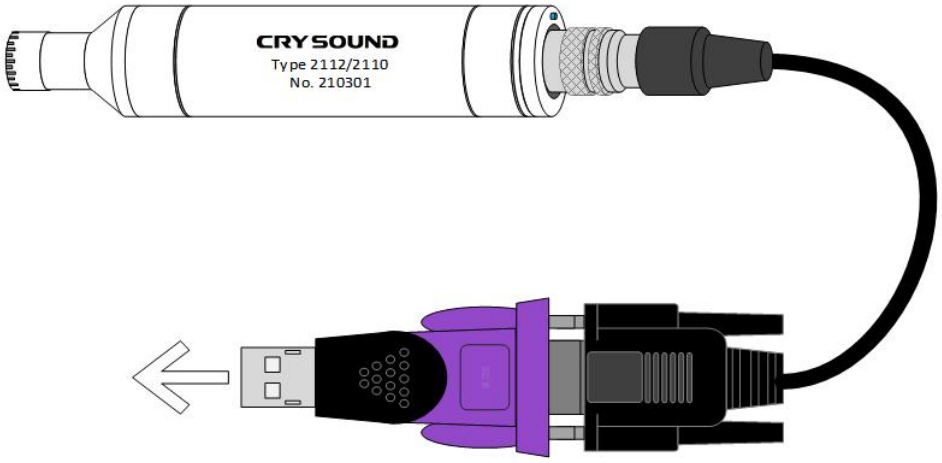


Fig 6.1 Connection diagram

After connection, use serial port send data for getting corresponding response.

6.3 CRY208 Channel Allocator (optional)

CRY208 can connect to PC, 8 slave interfaces can connect to noise sensor or host interface of other CRY208 to meet the cascade requirements. Computer controls all noise sensors by software to get data.

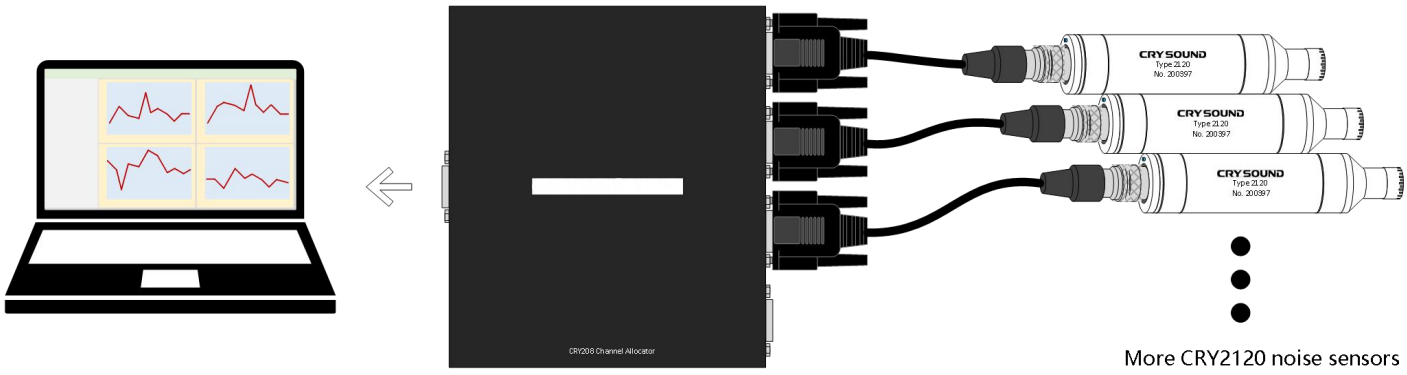


Fig 6.2 CRY208 connection diagram

7 Analog Quantity Connection and Signal Transmission

7.1 4-20mA Analog Quantity and SPL Calculation

4-20mA output has three pins which are power pin (5-24V), 4-20mA analog output pin and GND pin.

When using 4-20mA analog quantity output, 20-24V power supply voltage is recommended.

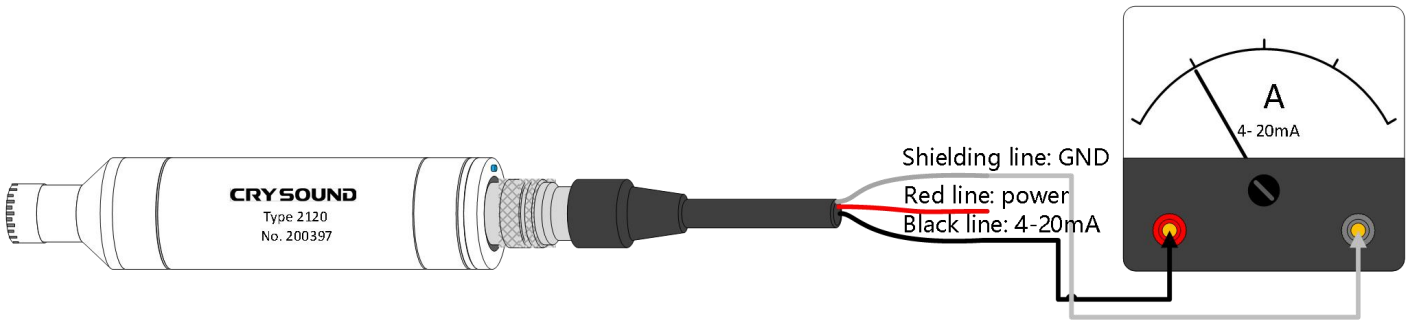


Fig 7.1 4-20mA analog quantity output connection diagram

SPL output range 20-140dB, corresponding current 4-20mA, formula of calculating SPL according to current:

In above formula:

—SPL: actual SPL

—I: current value of analog output

Current corresponding to SPL:

Current(mA)	SPL(dBSPL)	Current(mA)	SPL(dBSPL)
4	20	13	87.5
5	27.5	14	95
6	35	15	102.5
7	42.5	16	110
8	50	17	117.5
9	57.5	18	125
10	65	19	132.5
11	72.5	20	140

12	80		

7.2 Voltage Analog Quantity and SPL Calculation

Voltage output has three pins which are power pin (5-24V), 4-20mA analog output pin (4-20mA pin outputs voltage analog quantity) and GND pin. When using voltage analog quantity output, 20-24V power supply voltage is recommended.

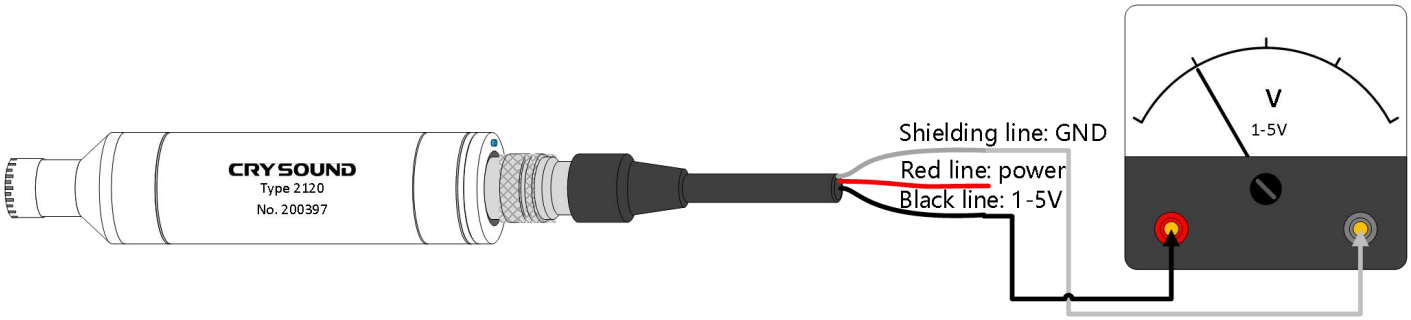


Fig 7.2 Voltage analog quantity output connection diagram

Sensor of voltage analog quantity output: 1-5V, 2-10V optional

Formula of calculating SPL and voltage:

(1-5V)

(2-10V)

In above formula:

— SPL: actual SPL

— U: voltage analog quantity output value

Voltage corresponding to SPL:

Voltage 1-5V	Voltage 2-10V	SPL (dBSPL)
1	2	20
1.5	3	35
2	4	50
2.5	5	65

3	6	80
3.5	7	95
4	8	110
4.5	9	125
5	10	140

8 Calibration

Calibration steps show below:

1. Use sound calibrator CRY5611(1kHz, 94dB);
2. Connect the noise sensor correctly;
3. Open sound calibrator after insert noise sensor into it.
4. Press the tail calibration button for one or two seconds, and the LED starts to blink, that means calibration starts.
5. If the rear LED blinks quicker than the start time, it means calibration failed. Please keep all the things steady and try again. If the rear LED does not blink quickly, it means calibration is successful.

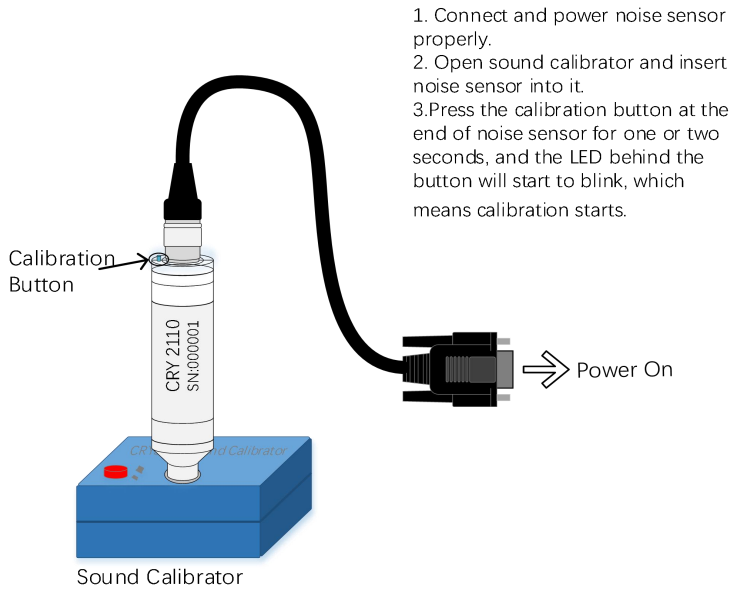


Fig 8.1 Sensor calibration steps

9 Interface Definition

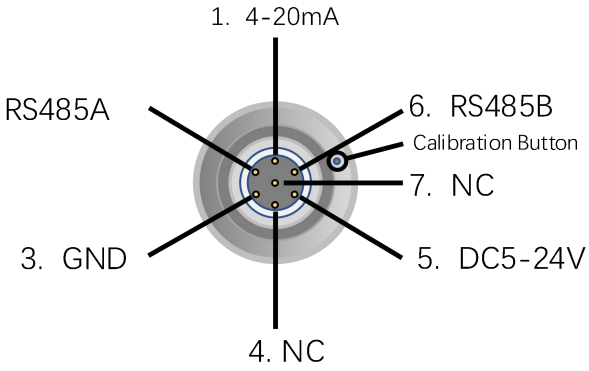


Fig 9.1 Noise sensor's rear ports definition

.

For ease of use, the other side of the cable has been connected to DB9 port. The DB9 port definition is as follows:

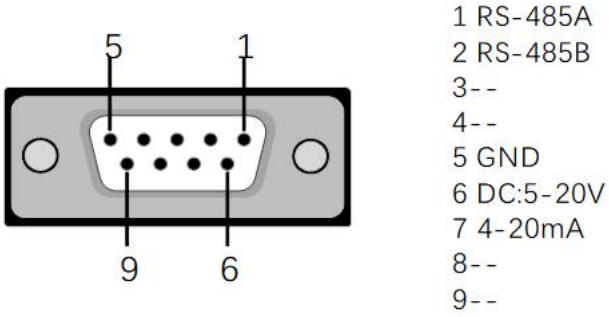


Fig 9. The DB9 interface definition

DB9DB9 interface pin definition and cable color definition

Pin	Definition	Color
1	RS485A	Green
2	RS485B	White
3	None	
4	None	
5	GND	Shield

6	DC 5-24V	Red
7	4-20mA	Black
8	None	
9	None	

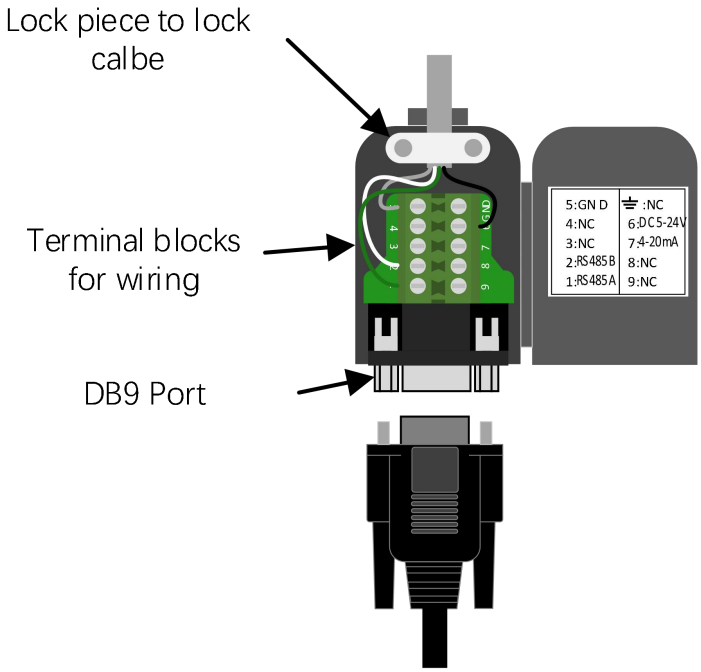
10 Configuration

Typical configuration:

NO	Name	Quantity
1	Noise sensor (CRY2120)	1
2	Windscreen	1
3	Adapter plate	1
4	Standard cable (2m)	1
5	User manual in paper	1
6	Delivery certificate and warranty card	1

Appendix1: How to Use the Adapter Plate

In order to facilitate users to self-wire, each sensor distribution an adapter plate, adapter plate can be directly connected to the standard DB9 female. After wiring the terminal of the adapter plate according to the pin definition of the data line, cable can be secured with white locking tabs and black screws to prevent disconnection when pulling the cable , fix the black screws from external to internal.



Noise sensor data cable

Appendix2: Modbus RTU Communication Protocol CRC Check Algorithm

```
static const unsigned char aucCRCHi[] = {
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
};
static const unsigned char aucCRCLo[] = {
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
    };
    unsigned short int usMBCRC16 (unsigned char * pucFrame, unsigned short int usLen )
    {
    unsigned char ucCRCHi = 0xFF;
    unsigned char ucCRCLo = 0xFF;
    int    iIndex;

    while( usLen-- ){
    iIndex = ucCRCLo ^ *( pucFrame++ );
    ucCRCLo = ( UCHAR )( ucCRCHi ^ aucCRCHi[iIndex] );
```

```
ucCRCHi = aucCRCLo[iIndex];

}

return (unsigned short int)( ucCRCHi << 8 | ucCRCLo );

}
```