Communication Protocol of
FLDC series electromagnetic flowmeter

Chapter 1 General introduction of communication functions

1.1 Communication functions overview

The meter has a serial interface (RS485) as an additional specification.
The functions that each interface can use and the devices that use these functions (hardware and software) are as follows:

Serial interface (RS485)

Functions	Protocol	Connected equipment
Modbus subordinate	Modbus RTU	Modbus host device (measurement instrument, PC、PLC etc.)

1.2 Serial communication

The specifications of the serial interface (RS485) of this instrument are as follows:

RS485 interface specifications

Socket type	2 point terminal board
Connecting method	Multipoint, bus topology network
Communication method	Half duplex
Synchronously	Start-stop synchronization
Baud rate	1200，2400，4800，9600，19200，38400，57600[bps]
Starting bit	1bit (fixed)
Data bits	8 bits (fixed)
Check bit	Select Odd/Even/No Check
Stopping bit	1 bit (fixed)
Receive buffer size	256 bytes
Communication distance	Up to 1.2km
Terminal impedance*2	External: 120Ω, 1/2W resistor is recommended

Note:
***1** Please refer to the instrument manual for specific outlets.
***2** When using multipoint connections (including point-to-point connections), connect only one terminating resistor to the meter at the end of the link. Do not connect terminating resistors to the instrument in the middle of the link. If a converter is used, turn on its termination impedance. The recommended converter must have

external termination impedance and a built-in termination impedance converter.

1.3 Data frame format

1.3.1 Transmission mode

The MODBUS standard network has two transmission modes: ASCII and RTU. The company uses RTU mode and does not support ASCII mode. The following discussion is based on RTU mode. MODBUS frame structure as shown in the following table:

<----------PDU---------->

Additional address domain	Code domain	Data domain	Data check domain

<----------ADU---------->

PDU: Protocol Data Unit

ADU: Application Data Unit

Among them, the PDU part is necessary, and the difference between ADU and PDU varies according to the transmission network.

1.3.2 ADU format

For serial-based MODBUS, the additional address field uses a 1-byte slave address, and the data check field uses a 2-byte CRC check, so the serial MODBUS ADU frame format is as follows:

Subordinate address	Function code	Command data	CRC check
1Byte	1Byte	2Byte	2Byte

Subordinate address: The address range of a single device is 1~247. Address 0 is broadcast address so that all slave devices can recognize it.

Function Codes: Slave tells what actions the device needs to perform. The slave device performs the corresponding actions according to the instructions of the function code.

Command data: It consists of two sets of hexadecimal numbers, with the high byte first and the low byte second.

CRC check: Loop length detection of message content, low byte first and high byte second.

Chapter 2 instrument register address table

MODBUS communication address table of electromagnetic flowmeter consists of register addresses of instantaneous flow, heat and other data, see as follows:

Note: The unit of communication from instantaneous heat to cumulative heat only applies to FLRN series products

Parameters	Type	Address		Order	Instructions
Instantaneous flow	float	100		0x04	
Instantaneous flow rate	float	102		0x04	
Flow percentage	float	104		0x04	50 represents 50%
Conductivity	float	106		0x04	
Forward flow accumulates integer	ulong	108		0x04	
Forward flow accumulates decimal	ulong	110		0x04	The decimal part is enlarged 1000 times, 123 represents 0.123
Reverse flow accumulates integer	ulong	112		0x04	
Reverse flow accumulates dicimal	ulong	114		0x04	The decimal part is enlarged 1000 times, 123 represents 0.123
Flow unit	ushort	116		0x04	0x00:L 0x01:m3 0x02:kg 0x03:t
Time unit	ushort	117		0x04	0x00:s 0x01:min 0x02:h
Accumulative flow unit	ushort	118		0x04	The same as that of flow
Alarm status	ushort	119		0x04	
Systematic alarm	*bit*	*119.0*		*0x04*	*0: no alarm，1: with alarm*
Empty tube alarm	*bit*	*119.1*		*0x04*	*0: no alarm，1: with alarm*
Upper limit alarm	*bit*	*119.2*		*0x04*	*0: no alarm，1: with alarm*
Lower limit alarm	*bit*	*119.3*		*0x04*	*0: no alarm，1: with alarm*
Instantaneous heat	float	120		0x04	
Temperature input	float	122		0x04	
Temperature output	float	124		0x04	
Heat accumulative integer	ulong	126		0x04	
Heat accumulative decimal	ulong	128		0x04	The decimal part is enlarged 1000 times, 123 represents 0.123
Cold accumulative integer	ulong	130		0x04	
Cold accumulative decimal	ulong	132		0x04	The decimal part is enlarged 1000 times, 123 represents 0.123
Heat unit	ushort	134		0x04	0x00:kW 0x01:MW 0x02:kJ/h 0x03:MJ/h .0x04:GJ/h
Accumulative heat unit	ushort	135		0x04	0x00:kWh 0x01:MWh 0x02:kJ 0x03:MJ .0x04:GJ

Note: Following the Modbus standard, the offset of 1 register is actually subtracted when the command is

actually sent.

Chapter 3 Command instance and resolution

3.1 Function code 04 (0x04): Read one or more registers

Master station command frame:

Equipment address	Function code (0x04)	Register address	Register quantity	CRC check
1Byte	1Byte	2Byte	2Byte	2Byte

Subordinate response frame:

Equipment address	Function code (0x04)	The number of data bytes N	Return data	CRC check
1Byte	1Byte	1Byte	N 个 Byte	2Byte

Special instruction:

Master station command frame: The number of registers indicates the number of read registers.

Slave response frame: The upper byte of each register is first, followed by the lower byte. (floating point data is arranged in floating point format)

3.2 Examples

3.2.1instantaneous flow (float)

Master station command frame (Hex) :

Equipment address	Function code	Register address higher bit	Register address lower bit	Register quantity higher bit	Register quantity lower bit	CRC lower bit	CRC higher bit
08	04	00	63	00	02	81	4C

Subordinate response frame (Hex):

Equipment address	Function code	The number of data bytes	Instantaneous flow (4 bytes floating point)				CRC lower bit	CRC higher bit
08	04	04	31	D6	40	E2	3C	09

Data analysis:

Instantaneous flow: 31 D6 40 E2 -> 40 E2 31 D6 (high to low) = 7.068

The floating-point number uses the low byte to register before the high byte register, and each register has the highest bit before and the low bit behind it. The IEEE754 32-bit floating-point format is used. See Appendix 2 for a detailed analysis method.

Note: All float format data can be processed refer to instantaneous flow method.

3.2.2 Positive flow accumulation (ulong)

Master station command frame (Hex) :

Equipment address	Function code	Register address higher bit	Register address lower bit	Register quantity higher bit	Register quantity lower bit	CRC lower bit	CRC higher bit
08	04	00	6B	00	04	80	8C

Subordinate response frame (Hex) :

Equipment address	Function code	The number of data bytes	Positive cumulative integer (Unsigned long integer)				Positive cumulative decimal (Unsigned long integer)				CRC lower bit	CRC higher bit
08	04	08	27	25	00	00	01	06	00	00	3C	09

Data analysis:

Positive cumulative integer: 27 25 00 00 -> 00 00 27 25(from higher bit to lower bit) = 10021

Positive cumulative decimal: 01 06 00 00 -> 00 00 01 06(from higher bit to lower bit) = 262=>262/1000=0.262

Positive accumulation: 10021.262

Note: All accumulated data can be processed with reference to positive flow accumulation.

3.2.3 Flow unit (ushort)

Master station command frame (Hex):

Equipment address	Function code	Register address higher bit	Register address lower bit	Register quantity higher bit	Register quantity lower bit	CRC lower bit	CRC higher bit
08	04	00	73	00	01	C0	88

Subordinate response frame (Hex):

Equipment address	Function code	The number of data bytes	Flow unit (Unsigned short integer)		CRC lower bit	CRC higher bit
08	04	02	00	01	A4	F1

Data analysis:

Flow unit: 00 01 = 1, through the lookup table 3-1 shows that the flow unit: m3.

Note: All units can be handled with reference to flow units, flow units (Table 3-1), flow time units (Table 3-2), flow accumulation units (Table 3-3), heat units (Table 3-4), heat accumulation units (Table 3-5).

Table 3-1 flow unit table

Code	0	1	2	3
Flow unit	L	m3	kg	t

Table 3-2 flow time unit table

Code	0	1	2
Flow unit	s	Min	h

Table 3-3 flow accumulation unit table

Code	0	1	2	3
Flow unit	L	m3	kg	t

Table 3-4 heat unit table

Code	0	1	2	3	4
Heat unit	kW	MW	kJ/h	MJ/h	GJ/h

Table 3-5 heat accumulation unit table

Code	0	1	2	3	4
Heat accumulation unit	kWh	MWh	kJ	MJ	GJ

3.2.4 Alarm status (ushort)

Master station command frame (Hex):

Equipment address	Function code	Register address higher bit	Register address lower bit	Register quantity higher bit	Register quantity lower bit	CRC lower bit	CRC higher bit
08	04	00	76	00	01	D0	89

Subordinate response frame (Hex):

Equipment address	Function code	The number of data bytes	Flow unit (Unsigned short integer)		CRC lower bit	CRC higher bit
08	04	02	00	02	E4	F0

Data analysis:

Flow alarm: 00 02 => 0000 0000 0000 0010，that is Bit 1: 1, check table 3-6, ***empty tube alarm.***

Note: Two bytes indicate 16 alarm values, bit: 1 (resulting in corresponding alarm), Bit: 0 (no corresponding alarm), flow alarm (Table 3-6)

Table 3-6 flow alarm table

Bit	0	1	2	3
Alarm item	Systematic alarm	Empty tube alarm	Upper limit flow alarm	Lower limit flow alarm

3.3 Abnormal command return

Master station command frame (Hex) :

Equipment address	Function code	Register address higher bit	Register address lower bit	Register quantity higher bit	Register quantity lower bit	CRC lower bit	CRC higher bit
08	04	00	76	00	C8	10	DF

Subordinate response frame (Hex):

Equipment address	Function code	Abnormal code	CRC lower bit	CRC higher bit
08	84	03	E4	F0

Abnormal frame function code: 0x80 + master command frame function code.

Abnormal code: See Table 3-7 for details.

Table 3-7 Abnormal code table

Abnormal code	name	instruction
02H	Illegal data address	1. Data address slave does not recognize 2. Invalid address combined with data address and data quantity
03H	Illegal data value	1. The amount of data is out of range 2. Data length error 3. Illegal data value

Chapter 4 Common communication exception handling methods

4.1 Subordinate has no response

If the master station sends data and the subordinate station does not respond, consider the parameter settings and physical wiring issues.

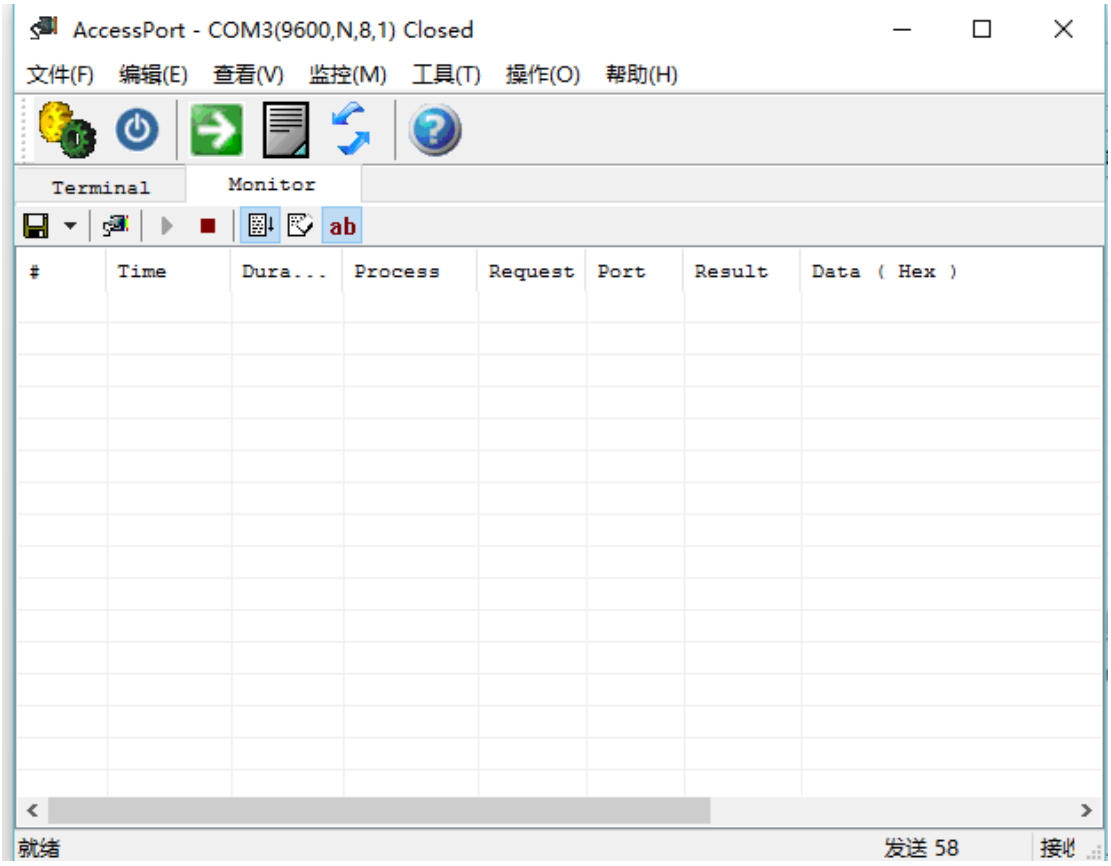
1. Determine whether the flowmeter has communication function: Query the flowmeter model and check whether it has communication function.
2. Check whether the communication line is connected correctly: Check if the A and B terminals of RS485 are connected incorrectly.
3. Determine whether the baud rate is correct: Check whether the baud rate of the upper computer and the baud rate set by the flowmeter are consistent.
4. Determine whether the communication address is correct: Check whether the communication address of the upper computer and the communication address of the flowmeter are consistent.
5. Judge whether the COM port is abnormal: In My Computer → Properties, check whether the serial port is available.

4.2 abnormal data analysis

Users can use the serial debugging assistant to detect the communication process. The specific steps are as follows:

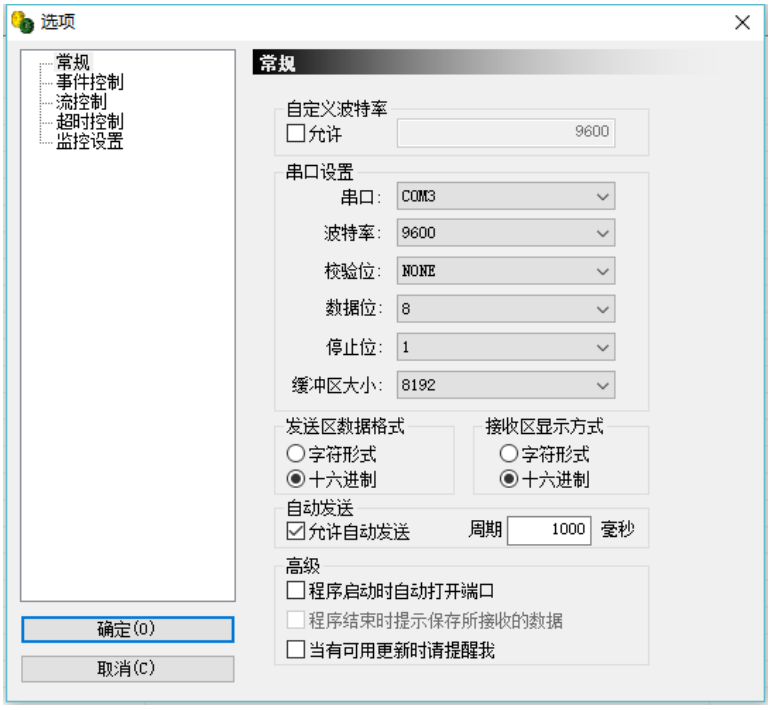
1. Connect the master station equipment and the flow meter correctly, and then use USB (or 232) to 485 ports in parallel to the system.

2. Open the serial debugging tool (AccesPort) and click "Monitor" to switch to the monitoring screen, click  , select monitoring serial port, and click  to start monitoring.



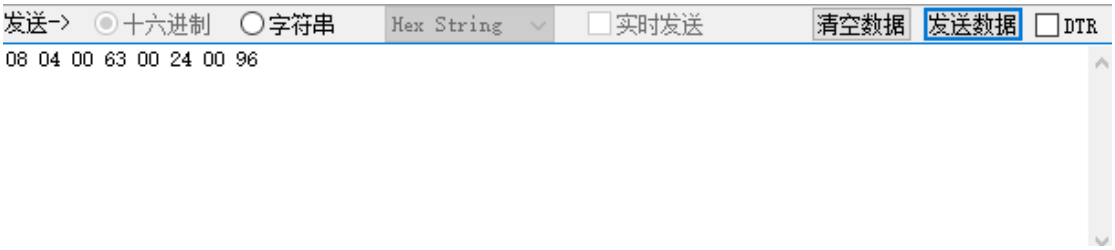
文件	File		操作	Operation
编辑	Edit		帮助	Help
查看	View		就绪	Ready
监控	Monitor		发送	Send
工具	tool		接收	Receive

3.return to “Terminal” mode, click parameter configuration icon  to set baud rate as9600, data bits 8, stop bit 1, no check, Send data format: Hexadecimal, receive data format: Hexadecimal.



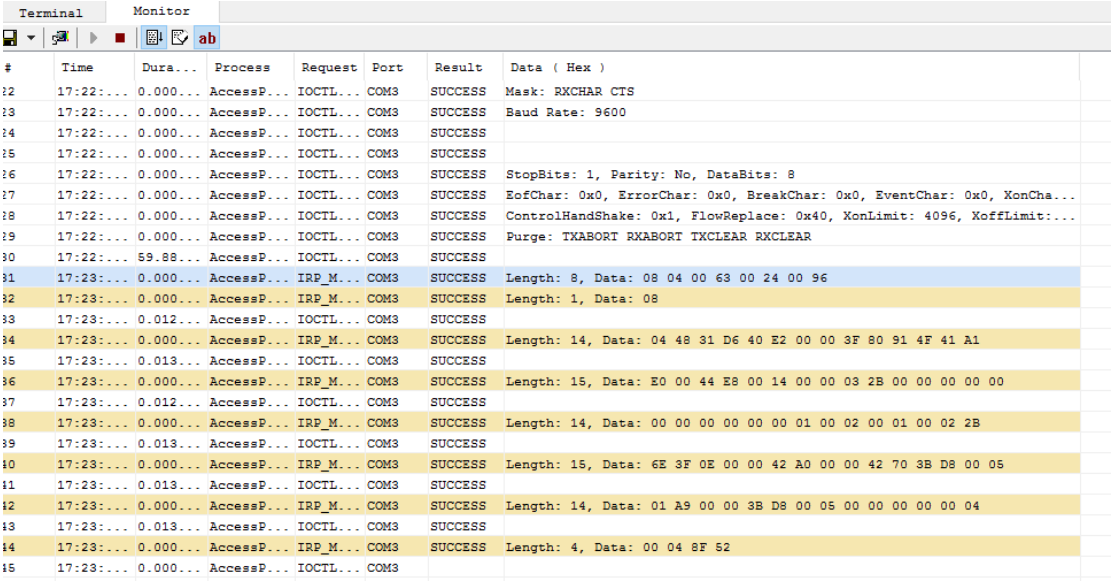
选项	Option		数据位	Data bit
常规	General		停止位	Stop bit
事件控制	Event Control		缓冲区大小	Buffer size
流控制	Flow control		发送区数据格式	Transmission area data format
超时控制	Over-time control		接收区显示方式	Receiving area display method
监控设置	Monitor setting		字符形式	Character form
确定	OK		十六进制	Hexadecimal
取消	Cancel		自动发送	Send automatically
自定义波特率	Customized baud rate		允许自动发送	Permit send automatically
允许	Permit		周期	circle
串口设置	Serial port settings		高级	Advanced
串口	Serial port		程序启动时自动打开端口	Automatically open interface when the program is started.
波特率	Baud rate		程序结束时提示保存所接收的数据	Prompt to save received data at the end of the program
校验位	Check bit		当有可用更新时请提醒我	Remind me when updates are available

4.Click  to open serial port, send a command to send a data frame and click Send Data.



发送	Send		实时发送	Real-time send
十六进制	Hexadecimal		清空数据	Clear data
字符串	Character serial		发送数据	Send data

5.Again return to “Monitor” to see monitoring data



#	Time	Dura...	Process	Request	Port	Result	Data (Hex)
22	17:22:...	0.000...	AccessP...	IOCTL...	COM3	SUCCESS	Mask: RXCHAR CTS
23	17:22:...	0.000...	AccessP...	IOCTL...	COM3	SUCCESS	Baud Rate: 9600
24	17:22:...	0.000...	AccessP...	IOCTL...	COM3	SUCCESS	
25	17:22:...	0.000...	AccessP...	IOCTL...	COM3	SUCCESS	
26	17:22:...	0.000...	AccessP...	IOCTL...	COM3	SUCCESS	StopBits: 1, Parity: No, DataBits: 8
27	17:22:...	0.000...	AccessP...	IOCTL...	COM3	SUCCESS	EofChar: 0x0, ErrorChar: 0x0, BreakChar: 0x0, EventChar: 0x0, XonCha...
28	17:22:...	0.000...	AccessP...	IOCTL...	COM3	SUCCESS	ControlHandShake: 0x1, FlowReplace: 0x40, XonLimit: 4096, XoffLimit:...
29	17:22:...	0.000...	AccessP...	IOCTL...	COM3	SUCCESS	Purge: TXABORT RXABORT TXCLEAR RXCLEAR
30	17:22:...	59.88...	AccessP...	IOCTL...	COM3	SUCCESS	
31	17:23:...	0.000...	AccessP...	IRP_M...	COM3	SUCCESS	Length: 8, Data: 08 04 00 63 00 24 00 96
32	17:23:...	0.000...	AccessP...	IRP_M...	COM3	SUCCESS	Length: 1, Data: 08
33	17:23:...	0.012...	AccessP...	IOCTL...	COM3	SUCCESS	
34	17:23:...	0.000...	AccessP...	IRP_M...	COM3	SUCCESS	Length: 14, Data: 04 48 31 D6 40 E2 00 00 3F 80 91 4F 41 A1
35	17:23:...	0.013...	AccessP...	IOCTL...	COM3	SUCCESS	
36	17:23:...	0.000...	AccessP...	IRP_M...	COM3	SUCCESS	Length: 15, Data: E0 00 44 E8 00 14 00 00 03 2B 00 00 00 00 00
37	17:23:...	0.012...	AccessP...	IOCTL...	COM3	SUCCESS	
38	17:23:...	0.000...	AccessP...	IRP_M...	COM3	SUCCESS	Length: 14, Data: 00 00 00 00 00 00 01 00 02 00 01 00 02 2B
39	17:23:...	0.013...	AccessP...	IOCTL...	COM3	SUCCESS	
40	17:23:...	0.000...	AccessP...	IRP_M...	COM3	SUCCESS	Length: 15, Data: 6E 3F 0E 00 00 42 A0 00 00 42 70 3B D8 00 05
41	17:23:...	0.013...	AccessP...	IOCTL...	COM3	SUCCESS	
42	17:23:...	0.000...	AccessP...	IRP_M...	COM3	SUCCESS	Length: 14, Data: 01 A9 00 00 3B D8 00 05 00 00 00 00 00 04
43	17:23:...	0.013...	AccessP...	IOCTL...	COM3	SUCCESS	
44	17:23:...	0.000...	AccessP...	IRP_M...	COM3	SUCCESS	Length: 4, Data: 00 04 8F 52
45	17:23:...	0.000...	AccessP...	IOCTL...	COM3		

The blue part is the data sent by the master station, and the yellow is the data from the subordinate station.

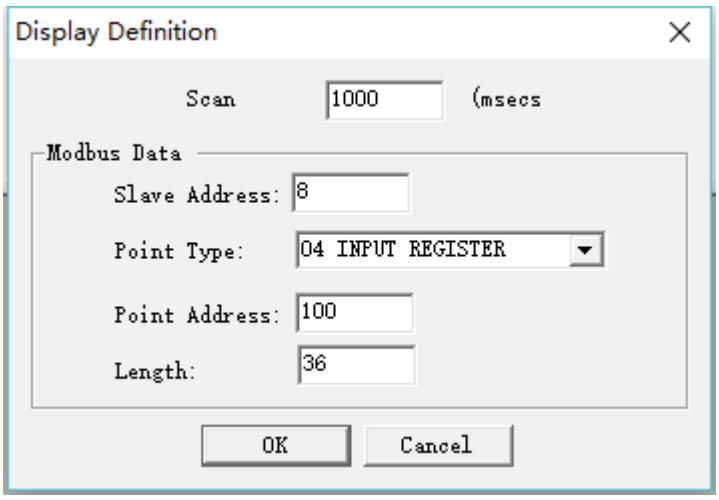
Analyze according to the intercepted data, analyze whether the communication address, function code, register address, register length, and CRC check code sent by the master station are correct.

Appendix

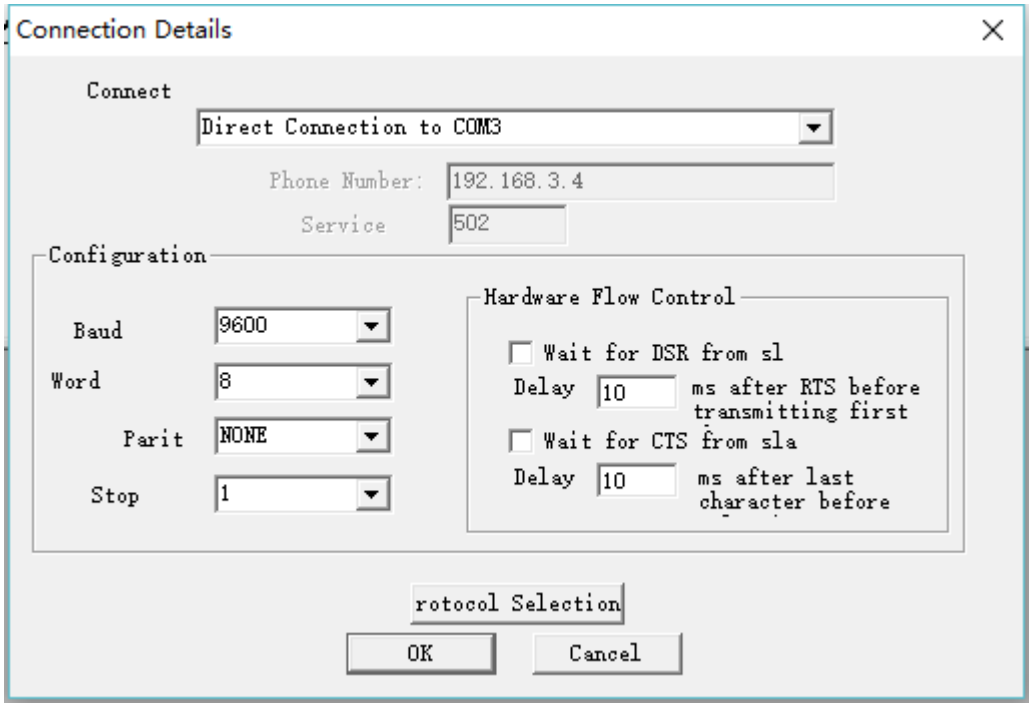
Appendix 1　modscan32 communication example

Flowmeter communication address was set as 8, baud rate was set as 9600.

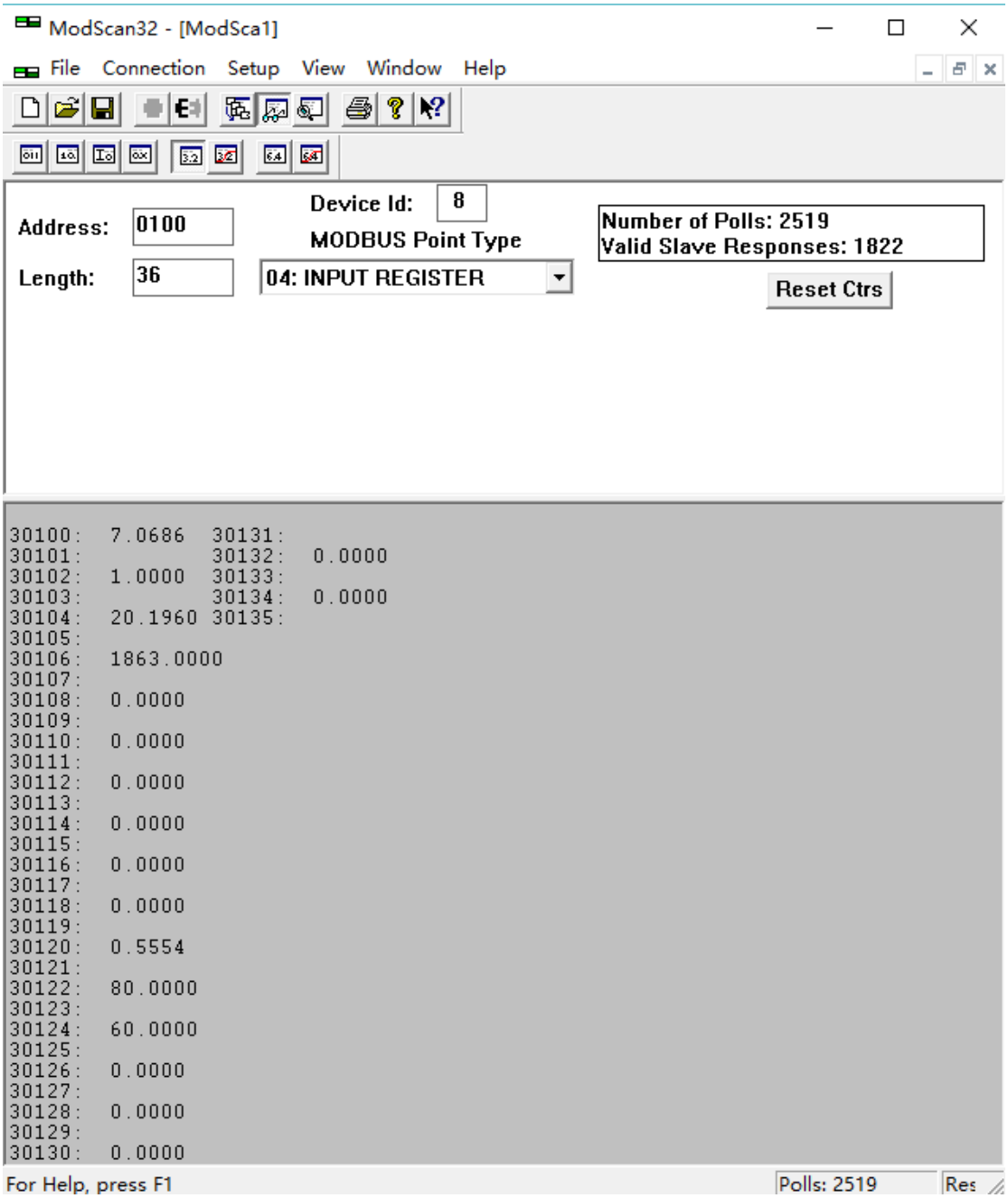
1. Click Setup→Poll Definition, Set acquisition commands include device address 8, MODBUS function code 04, register address 100, register length 36, and acquisition interval 1000.



2. Click Connection→Connection Detalls, set flowmeter serial port format: 1 start bit 8 bits data bit 1 stop bit, no check.



3. After you click OK, you can communicate.


ModScan32 - [ModSca1]
File Connection Setup View Window Help
Address: 0100
Length: 36
Device Id: 8
MODBUS Point Type
04: INPUT REGISTER
Number of Polls: 2519
Valid Slave Responses: 1822
Reset Ctrs
30100: 7.0686 30131:
30101: 30132: 0.0000
30102: 1.0000 30133:
30103: 30134: 0.0000
30104: 20.1960 30135:
30105:
30106: 1863.0000
30107:
30108: 0.0000
30109:
30110: 0.0000
30111:
30112: 0.0000
30113:
30114: 0.0000
30115:
30116: 0.0000
30117:
30118: 0.0000
30119:
30120: 0.5554
30121:
30122: 80.0000
30123:
30124: 60.0000
30125:
30126: 0.0000
30127:
30128: 0.0000
30129:
30130: 0.0000
For Help, press F1
Polls: 2519
Res

Appendix 2 CRC Cyclic redundancy check algorithm

1. CRC check overview

The basic idea of the CRC check code is to use linear coding theory. According to the k-bit binary code sequence to be transmitted at the sending end, a check code (CRC code) r bit is generated for checking and is attached to the information, a new number of binary code sequences (k+r) is formed and sent out. At the receiving end, a check is made based on the rules followed between the information code and the CRC code to determine if there was an error in the transmission.

2. CRC check algorithm

```
//CalCrc====================
//Funciton        calculation     CRC check
//Parameter       buf             checking buffer
//                length          checking length
//Return          CRC check result, short integer means HL
const uchar ucCRCHi[] =
{
    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
} ;
const uchar ucCRCLo[] =
```

```
{
    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
} ;
//CRC calculation
ushort CalCrc(uchar *pucData , ushort usDataLen)
{
    uchar ucCrcLo = 0xFF ;
    uchar ucCrcHi = 0xFF ;
    uchar ucIndex ;
    While (usDataLen--)
    {
        ucIndex = ucCrcLo ^ *pucData++ ;
        ucCrcLo = ucCrcHi ^ ucCRCHi[ucIndex] ;
        ucCrcHi = ucCRCLo[ucIndex] ;
    };
    Return (ucCrcHi * 0x100 + ucCrcLo) ;
}
```

Appendix 3　IEEE 4 byte floating point transfer and encoding

1.　IEEE4 byte floating point encoding

4 bytes total 32 bits

00-22 bit　　mantissa (1. mantissa)

23-30 bits　　mantissa (2^ (mantissa - 127))

31 bits　　symbol (0 positive; 1 negative)

Value = (symbol) [(1. mantissa) * (2^ (Order code - 127))]

2.　Examples

1、read data: 08 04 00 63 00 02 81 4C[offset 192, length 2 words]

2、receive data: 08 04 04 00 00 40 88 53 22[data 00004088, length 04]

3、analyze data: 00 00 40 88[FF1 FF2 FF3 FF4]

A、Before and after word exchange order 40 88 00 00[FF3 FF4 FF1 FF2]

Binary: 0100 0000 1000 1000 0000 0000 0000 0000

B、mantissa =　000 1000 0000 0000 0000 0000 B = 0.0625;

Order code = 10000001B - 127 = 129 - 127 = 2 ;

Symbol = (+);

C、calculate value:+ [1.0625 * 2^2] = 4.25

Appendix 4 ASCII code table

		Higher 4 bits															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
lower 4 bits	0			SP	0	@	P		p								
	1				1	A	Q	a	q								
	2				2	B	R	b	r								
	3			#	3	C	S	c	s								
	4				4	D	T	d	t								
	5			%	5	E	U	e	u								
	6			&	6	F	V	f	v								
	7				7	G	W	g	w								
	8			(	8	H	X	h	x								
	9			)	9	I	Y	i	y								
	A	LF		*	:	J	Z	j	z								
	B		ESC	+		K		k									
	C					L		l									
	D	CR		-		M		m									
	E			.		N		n									
	F			/		O		o									