MODBUS Protocol Specification

V1.0

MODBUS APPLICATION PROTOCOL SPECIFICATION

V1.0

CONTENTS

Part I

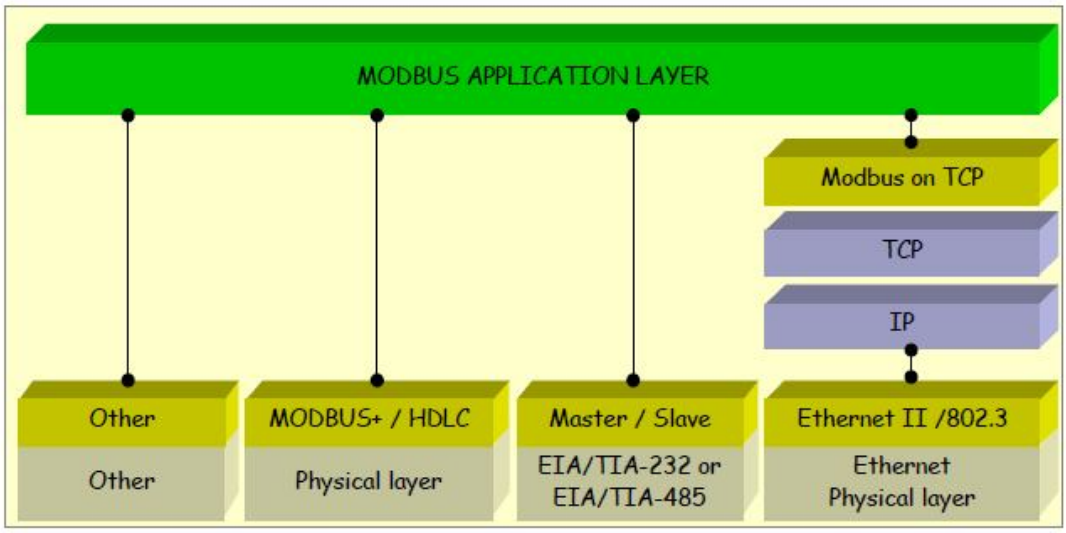
Introduction

1 Scope of this document

This document provides information for Fineco electric devices implementing the MODBUS RTU protocol.

MODBUS is an application layer messaging protocol, positioned at level 7 of the OSI model, that provides client/server communication between devices connected on different types of buses or networks. It is currently implemented using:

- TCP/IP over Ethernet. See MODBUS Messaging Implementation Guide V1.0a
- Asynchronous serial transmission over a variety of media (wire : EIA/TIA-232-E, EIA-422, EIA/TIA-485-A, fiber, radio, etc.)
- MODBUS PLUS, a high speed token passing network.



MODBUS Communication stack

The industry's serial de facto standard since 1979, MODBUS continue to enable millions of automation devices to communicate. Today support for the simple and elegant structure of MODBUS continues to grow. The Internet community can access MODBUS at a reserved system port 502 on the TCP/IP stack.

MODBUS is a request/reply protocol and offers services specified by function codes. MODBUS function codes are elements of MODBUS request/reply PDUs. The objectives of this document is to describe the function codes used within the framework of MODBUS

transactions.

2 Protocol overview

For a detailed description of the MODBUS protocol please view the web site www.modbus.org where the latest specs can be found.

3 Contacts

For further help and assistance please use the following contacts:

Tel:

Fax:

Email: antsmw@163.com

Part II

2 MODBUS transmission Modes

One serial transmission modes is defined: The RTU mode.

2.1 RTU Transmission mode

When devices communicate on a MODBUS serial line using the RTU (Remote Terminal Unit)mode, each 8-bit byte in a message contains two 4-bit hexadecimal characters. Each message must be transmitted in a continuous stream of characters.

The format (11bits) for each byte in RTU mode is:

Coding System: 8-bit binary

Bits per Byte: 1 start bit

8 data bits, least significant bit sent first

1 bit for parity completion

1 stop bit

Even parity is required.

2.2 Frame Checking Field:

Cyclical Redundancy Checking(CRC)

2.2.1 Frame description

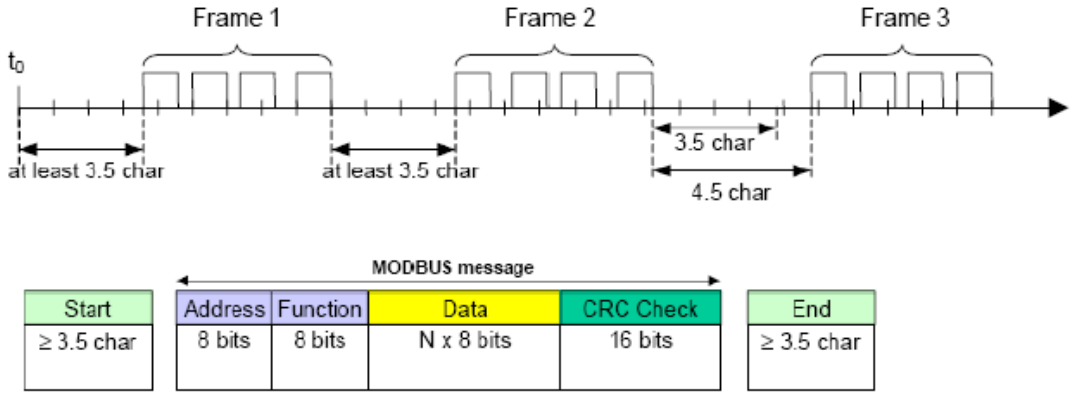
Slave Address	Function Code	Data	CRC
1byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low CRCHi

The maximum size of a MODBUS RTU frame is 256 bytes.

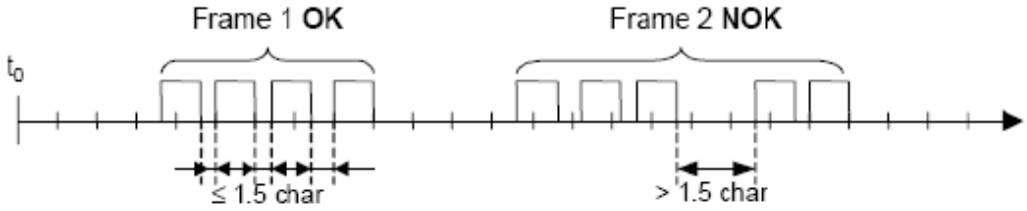
2.2.2 RTU Message Framing

A MODBUS message is placed by the transmitting device into a frame that has a known beginning and ending point. This allows devices that receive a new frame to begin at the start of the message, and to know when the message is completed. Partial message must be detected and errors must be set as a result.

In RTU mode, message frames are separated by a silent interval of at least 3.5 character times. In the following sections, this time interval is called t3.5.



The entire message frame must be transmitted as a continuous stream of characters. If a silent interval of more than 1.5 character times occurs between two characters, the message frame is declared incomplete and should be discarded by the receiver.



Note:

The implementation of RTU reception driver may imply the message of a lot of interruptions due to the t1.5 and t3.5 times.

2.2.3 RTU CRC Checking

The RTU mode includes an error-checking field that is based on a Cyclical Redundancy Checking(CRC) method performed on the message contents.

2.2.4 Data signal Rate

forlong's slave device supports the following baud rates

Baud Rate	Comments
1200	
2400	
4800	
9600	

2.2.5 Data Formats

2.2.5.1 unsigned 16-bit integer word Format

The Modbus applications support 16 bit integer information for several of the function codes.

A read or write to a modbus register comprise a 2×8 bit byte.

2.2.5.2 IEE 32-it Floating-point Register Format

The Modbus applicaion support IEE 32-bit floating point information for several of the function codes.

Part 

3 MODBUS Function Codes

Fineco electric Modbus RTU uses a subset of the standard Modbus function codes to provide access to measurement and information registers. These standard function codes provides basic support for IEE32-bit floating point number, 16 bit integer .

Function Code	Name	Usage
0x03	**Read Holding Register**	Read Multiple holding register
0x10	**Write multiple holding register**	Write multiple holding register

3.1 03(0x03)Read Holding Registers

This function code is used to read the contents of a contiguous block of holding registers in a remote device.The request PDU specifies the starting register address and the number of registers.In the PDU registers are address starting at zero.Therefore register numbered 1-16 are addressed as 0-15.

The register data in the response message are packed as two byte per register, with the binary contents right justified within each byte.For each register,the first byte contains the high order bits and the second contains the low order bits.

Request

Function code	1 Byte	0X03
Starting Address	2 Byte	0X0000 to 0XFFFF
Quantity of register	2 Byte	0x0001 to 0x007D

Response

Function code	1 Byte	0X03
Byte count	1 Byte	2 × N*
Register value	N* ×2 Bytes	

N* = Quantity of registers

Error

Function code	1 Byte	0x83
Exception code	1 Byte	0x01 or 0x02 or 0x03 or 0x04

An Example of a request to read register 0x006B-0x006D

Request	
Field Name	(Hex)
Slave Address	02
Function	03

Starting Address Hi	00
Starting Address Lo	6B
No. of Register Hi	00
No. of Register Lo	03
Check Sum	CRC
Check Sum	CRC

Response	
Field Name	(Hex)
Slave Address	02
Function Code	03
Byte Count	06
Register value Hi	02
Register value Lo	2B
Register value Hi	00
Register value Lo	00
Register value Hi	00
Register value Lo	64
Check Sum	11
Check Sum	8A

Register value Lo	90
Check Sum	CRC
Check Sum	CRC

3.2 16(0x10) Write Multiple register

This function code is used to write a block of contiguous registers in a remote device. The requested written values are specified in the request data field. Data is packed as two bytes per register.

The normal response returns the function code, starting address, and quantity of registers written.

Request

Function code	1 Byte	0X10
Starting Address	2 Byte	0X0000 to 0Xffff
Quantity of register	2 Byte	0X0000 to 0XFFFF
Byte Count	1 Byte	2 × N*
Register value	N*×2 Byte	Value

N* = Quantity of registers

Response

Function code	1 Byte	0X10
Starting Address	2 Byte	0X0000 to 0Xffff
Quantity of register	2 Bytes	1 to 123 (0x7B)

Error

Error code	1 Byte	0X90
Exception Code	1 Byte	0x01 or 0x02 or 0x03 or 0x04

Example

An example of a writing to register 40915 (Pulse value for power) the value 1.0,to slave address 5 in RTU mode

Request	
Field Name	(Hex)
Slave Address	05
Function code	10
Starting Address Hi	03
Starting Address Lo	92
No. of Register Hi	00
No. of Register Lo	02
Byte count	04
Register value Hi	3F
Value	80
value	00
Register value Lo	00
Check Sum	77
Check Sum	26

Response	
Field Name	(Hex)
Slave Address	05
Function	10
Starting Address Hi	03
Starting Address Lo	92
No. of Register Hi	00
No. of Register Lo	02

Check Sum	E1
Check Sum	E5

Part Ⅳ

4 MODBUS Register map

This appendix describes all parameters accessible by Function Codes 0x03, 0x10.Parameters are grouped together according to the measurement been made, to simplify and speed up the reading of the data.

The availability of parameters and functions is depended on the device been accessed.

4.1 register Map Overview

The following table describes the global register map for the function Codes for forlong producot.

EM735 series register map

Address (hex)	Length (bytes)	Parameter Name	Access (R/W)	Function code	Data Format	Units
0X000F	1	Modbus slave address number	R/W	03/10	Hex	
0x011E	2	Active energy	R	03	Hex	KWh
0x0120	2	Future use	R	03	Hex	
0x0122	1	Energy scale	R	03	Hex	FFFF = 10^{-1} 0000 = 10^{0}
0x0123	1	CT ratio	R	03	Hex	
0x0124	1	Meter mode	R	03	Hex	Type735.1.2: 0x0002 Type735.2.2: 0x0003
0x0125	1	Hardware version	R	03	Hex	
0x0126	1	Software version	R	03	Hex	Type735.1.2: V1.2 Type735.2.2: V2.2
0x0127	3	Serial number	R/W	03/10	BCD	000000000000-999999999999
0x011E—0x0127	12	Block read	R	03		
0xF800	1	Remote modify Modbus slave Baud rate with serial number	R/W	10	Hex	1200bps 0x0001 2400bps 0x0002 4800bps 0x0003 9600bps 0x0004

						Remote write with serial number
0xF810	1	Remote modify CT rate with serial number	W	10	Hex	1,10,15,20,25,30,40,50 60,80,100,120,150,160 200,240,250,300,400,480 500,600,800,1000,1200,1500
0xF820	1	Remote modify modbus id with serial number	W	10	Hex	
0xF840	3	Serial number confim	W	10	BCD	

Note:

Setting the Modbus ID

There are two methods of setting the Modbus ID. The first is by writing to address 0x000F while pressing the "SET" button. The second method is to write the Modbus ID to address 0xF820 together with the meter serial number in the same Modbus write command. For example, if the meter serial number is 2010 05 07 1234, the following command will set the Modbus ID to 8,

00 10 F8 20 00 04 **08** 00 08 20 10 05 07 12 34 <CRC>

Note that the broadcast Modbus ID 0 can be used. If the serial number does not match, the meter must not respond.

If register 0xF820 is read, the Modbus ID will be returned.

Serial Number Confirm

If the correct serial number is written to this address, the meter will acknowledge the write. If the incorrect serial number is written, the meter will not respond. For example, if the meter serial number is 2010 05 07 1234 and the Modbus ID is 5, the command,

00 10 F8 40 00 03 **06** 20 10 05 07 12 34 <CRC>

Will receive the acknowledgement

05 10 F8 40 00 03 <CRC>

If the incorrect serial number is written, there will be no response. This command can be broadcast to address 0 to find the Modbus ID of the meter with the serial number that is written. The serial number of the meter can also be read from this address.

Setting the Baud Rate

There are two methods of setting the baud rate. The first is by writing to address 0xF800 while pressing the PRG button. The second method is to write the baud rate to address 0xF800 together with the meter serial number in the same Modbus write command. For example, if the meter serial number is 2010 05 07 1234 and the Modbus ID is 5, the following command will set the baud rate to 2400,

05 10 F8 00 00 04 **08** 00 02 20 10 05 07 12 34 <CRC>

Table 4 shows the value that must be written for the various baud rates.

Baud Rate (0xF800)	
Value	**Scale**
1	1200
2	2400
3	4800
4	9600