VEDA MC

MCD2 series
communication protocol Modbus RTU

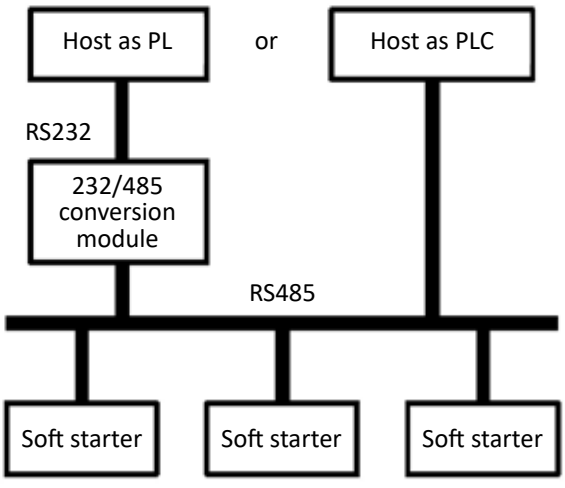


1. Preface

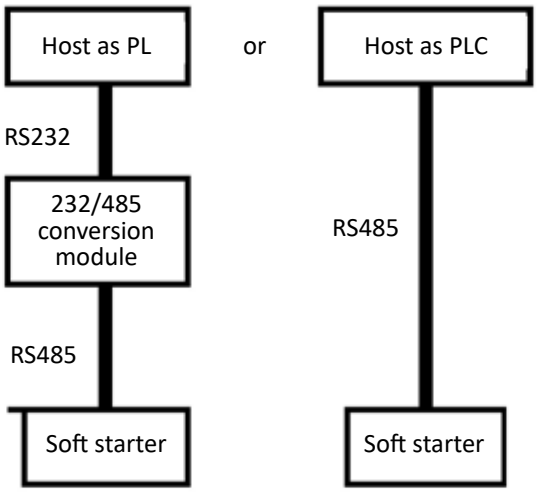
This communication protocol manual mainly describes the networking mode of the soft starter and the related aspects of the communication protocol, which is convenient for users to realize the configuration. This soft starter can provide the user with RS485/RS232 communication interface. The interface of the host computer equipment (such as PC, PLC, DCS controller, etc.) to communicate, realize the centralized monitoring and remote operation of the soft start (such as setting various parameters, control start, stop, read various parameters and soft start working status, etc.).

This communication protocol is an interface specification document designed to realize the above functions, please read carefully and follow the programming to realize remote and network control of the soft starter.

2. Networking method



Single master multi-slave network diagram



Single master single slave network diagram

3. Modbus RTU Overview

General knowledge

For the exchange of information between automation systems, between automation systems and connected decentralized field devices, serial fieldbuses are today mainly used as communication systems. Thousands of applications have strongly demonstrated that up to 40% of the costs of wiring, commissioning and maintenance can be saved by using fieldbus technology.

All relevant information of field devices, such as input and output data, parameters, diagnostic data, can be transmitted using only two wires. Fieldbuses used in the past tended to be manufacturer-specific fieldbuses and line is not compatible. The fieldbuses in use today are almost completely open and standardized. This means that users can choose the best product at the most reasonable price without relying on each individual manufacturer.

Modbus RTU is an international, open field bus standard. As an easy-to-implement fieldbus protocol, Modbus has been successfully applied all over the world. Application areas include automation in production processes, process control and building automation.

Basic Features

Modbus RTU determines the technical and functional characteristics of a serial field bus system that connects distributed digital automation equipment together. It is designed to deliver data at extremely high speeds at the field level. The central control equipment here, such as PLC or PC, communicates the

input and output data of switch quantity and/or analog quantity with the peripheral field station through the relevant high-speed serial interface.

Master device - controls data exchange on the bus. The master is allowed to send information without an external request. Slave Devices - Peripherals such as AKS series soft starters. They do not have the ability to access the bus, so that is, they can only acknowledge messages that have been received, or, at the request of the master, send messages to the master. Slaves are also often referred to as passive stations.

The following rules generally apply:

- On a bus, only one master station is allowed to work (single master bus system).
- Up to 247 slave stations can be connected to a bus.
- The communication is always initiated by the master station, and the slave station can only respond to the request of the master station.
- Protection features: “sum check” and parity bits.
- There are two ways for the communication between the master station and the field device (slave station):
- Individual communication method - direct request to a specific field device. (Slave address cannot be “0”):
- “Normal” operation: The master sends a request to a field device, which then responds.
- Broadcast mode - the total request for all field devices. (Slave address is “0”):

The master sends a request (telegram) to all bus stations. For example, sending an “urgent” command without any slave response.

4. Protocol Basic Structure

- **message structure**

Based on serial data transmission, it is transmitted bit by bit. RS485 interface: asynchronous, half-dual. The default data format is 8-N-1, 19200bps.

- **Interface method**

As shown in the figure below, pin 15 is the B(-) line, and pin 16 is the A(+) line.



The RS485 communication interface is placed in a high-impedance state in addition to sending data in general. In order to prevent the output from being unstable and causing malfunction at this time, the RS485 communication interface circuit of the computer sometimes has a built-in safety circuit that boosts and attenuates the output signal to maintain a low-impedance state. Please install a terminating resistor when communicating. As shown in Figure 3:

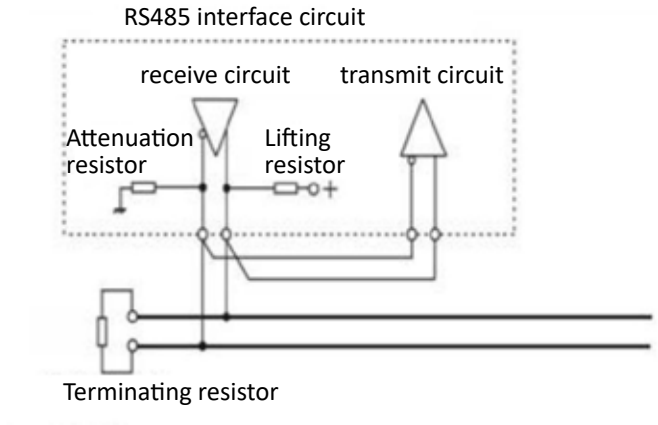


Figure 3. Install terminating resistor.

Note: Please do not operate the function codes related to communication during the communication process. During the communication process, the normality of the communication cannot be guaranteed, which will lead to the unstable operation of the soft starter.

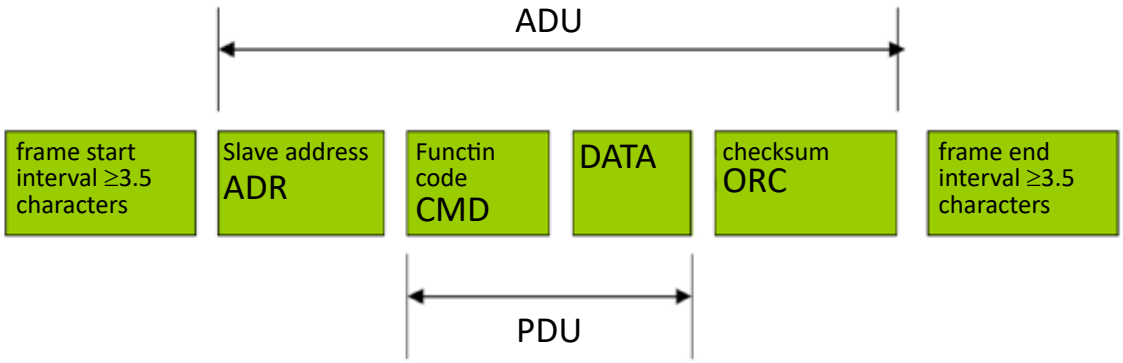
For the function code settings related to computer and serial communication, please refer to Table 2:

Table 2 Serial communication function code setting table

Function code	Setting value	Illustrations
F1.00 control mode	2, 4, 5, 6	The communication controller is valid, otherwise only data can be read
F8.00 Communication enabled	1	When set to 1, the communication is valid
F8.01 local address	1~247	Local address, 0 is the broadcast address

- **Protocol format**

Modbus ADU (Application Data Unit) consists of address, function code, actual data and CRC. Modbus PDU (Protocol Data Unit) consists of two parts: function code and actual data



The Modbus ADU (Application Data Unit) check is obtained by exchanging the high and low bytes of the CRC16 checksum of the first three parts of the ADU. If the operation request fails, the PDU (Protocol Data Unit) response is an error code and an abnormal code.

The three PDU types of the communication process use three different communication types for communication.

- Request – a request from the DCS/controller (master).
- Response - error-free processing by the field device (slave).
- Abnormal response - the field device returns after modifying the original function code requested (the highest bit is set to logic 1); the abnormal code contains error type information.

Slave Address (ADR):

The legal address is between 0 and 247, the address is 0, which means to broadcast to all soft-starts, in the case of broadcast, the slave will not respond any information.

For example: to communicate with the soft starter whose address is 16 (decimal), ADR=0x10 function code (CMD):

- **CMD=0x03,** read N registers, the maximum N is 8,example: The starting address of the soft start with the slave address 01 is 2001H, and 2 registers are read continuously.

The master sends	
ADR	01
CMD	03
Start address	20
	01
Number of registers	00
	02
CRC Low	DE
CRC High	0A

Slave responds correctly	
ADR	01
CMD	03
number of bytes read	04
2000 data	00
	00
2001 data	00
	00
CRC Low	72
CRC High	73

Slave error response (if the address is incorrect)	
ADR	01
CMD	83
Abnormal code	02
CRC Low	C0
CRC High	F1

- **CMD=0x06,** write a single register example: set F0.00 (control mode) as 1.

The master sends	
ADR	01
CMD	06
Data address	00
	00
Data content	00
	01
CRC Low	48
CRC High	0A

Slave responds correctly	
ADR	01
CMD	06
Data address	00
	00
Data content	00
	01
CRC Low	48
CRC High	0A

Example: Start motor 1.

The master sends	
ADR	01
CMD	06
Data address	20
	00
Data content	00
	21
CRC Low	42
CRC High	12

Slave responds correctly	
ADR	01
CMD	06
Data address	20
	00
Data content	00
	21
CRC Low	42
CRC High	12

- **CRC Cyclic Redundancy Check**

The Cyclic Redundancy Check CRC area is 2 bytes, containing a 16-bit binary data. The CRC value is calculated by the sending device, and the calculated value is attached to the information. When the receiving device receives the information, it recalculates the CRC value and compares the calculated value with the received actual value in the CRC area. If the two are different, An error is generated.

At the beginning of CRC, first set all 16 bits of the register to “1”, and then put the data of two adjacent 8-bit bytes into the current register, only the 8-bit data of each character is used to generate CRC, the start bit , stop bits and parity bits are not added to the CRC. During CRC generation, each 8-bit data is XORed with the value in the register, the result is shifted one bit to the right (towards the LSB direction), and the MSB is filled with “0”, and the LSB is detected. The fixed value set is XORed. If the LSB is “0”, the XOR operation will not be performed.

Repeat the above process until shifting 8 times. After completing the 8th shift, the next 8-bit data is XORed with the current value of the register. After all the information is processed, the final value in the register is the CRC value.

The process of generating CRC:

Set the 16-bit CRC register to FFFFH.

1. The first 8-bit data is XORed with the lower 8 bits of the CRC register, and the result is placed in the CRC register.
2. The CRC register is shifted one bit to the right, the MSB is zero-filled, and the LSB is checked.
3. (If LSB is 0): Repeat 3, then shift right by one. (if LSB is 1): XOR the CRC register with Aool H
4. Repeat 3 and 4 until 8 shifts are completed, completing the processing of the 8-bit byte.
5. Repeat steps 2 to 5 to process the next 8 bits of data until all bytes have been processed.
6. The final value of the CRC register is the CRC value.
7. When putting the CRC value into the message, the upper 8 bits and the lower 8 bits should be placed separately. Put the CRC value into the message, when sending the 16-bit CRC value in the message, send the lower 8 bits first, and then send the higher 8 bits.

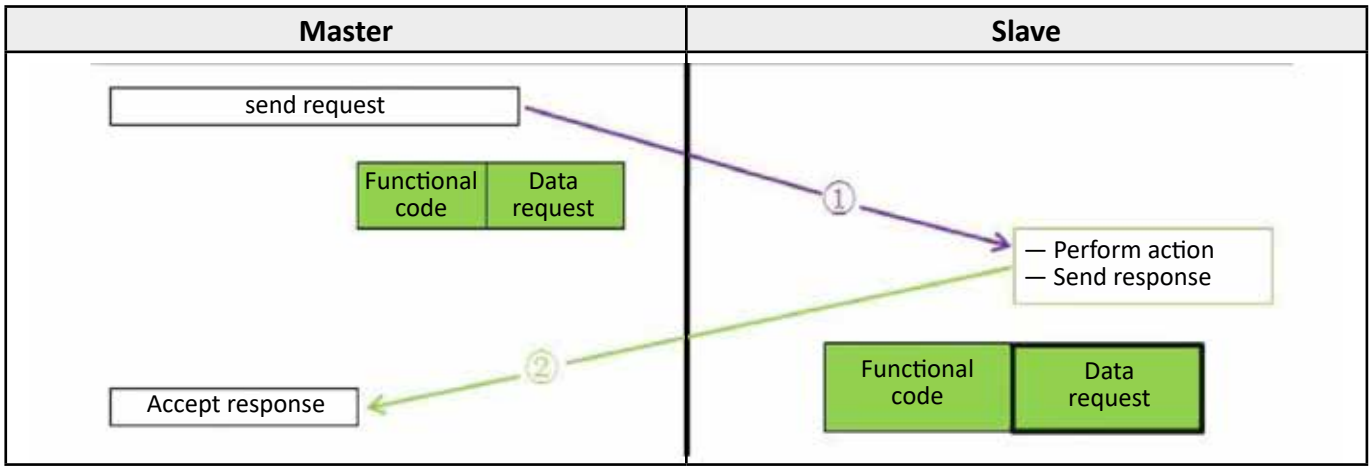
Example: Generate CRC value in C language, this function requires two parameters:

```
unsigned int crc_16(unsigned char* data, unsigned char length) {
int i,crc_res=0xffff;
while (length--)
{
crc_res^=*data++; for (i=0;i<8;i++)
{
if (crc_res&0x01)
crc_res=(crc_res>>1)^0xa001; else
                                        crc_res=crc_res>>1;

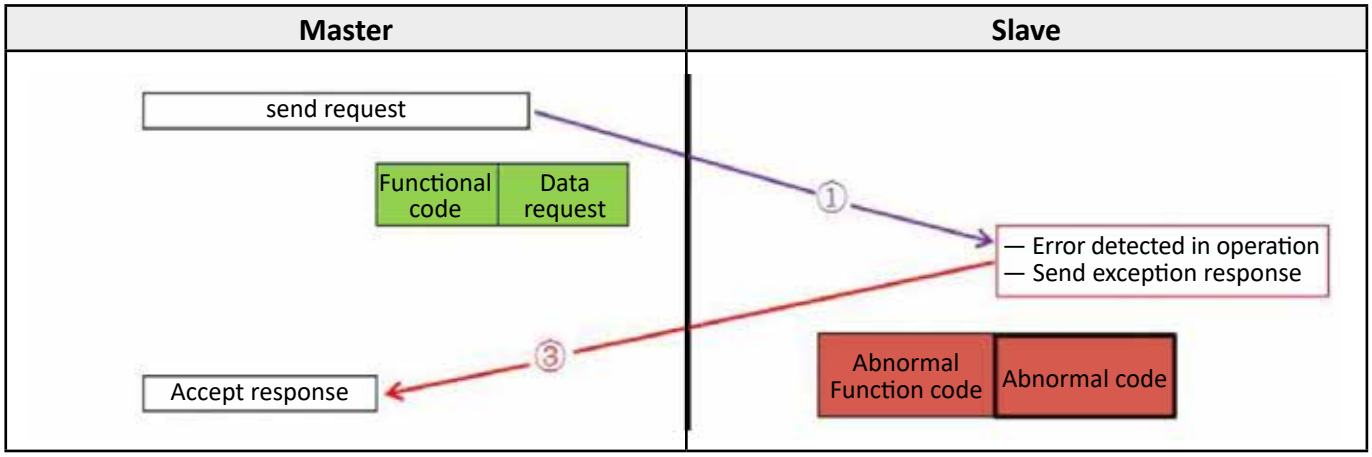
}
}
return crc_res;
}
```

- **Communication process:**

  - ***error free***



  - ***abnormal response***



The abnormal code is as follows: **function code = function code + 0x80**

Abnormal code	Meaning shown
0x01	Illegal function code
0x02	illegal data address
0x03	Illegal data, the data exceeds the upper and lower limits
0x04	The parameter is read-only and cannot be modified

Address space and Commands

Definition	Address space	Function illustration	
Soft starter **inner parameters (03&06)**	0x0000~ 0x0d00	The corresponding relationship between the soft start function code and the Modbus RTU protocol register address, the high byte is the function code group number, F0~FC, the corresponding high byte is 0x00~0x0C, and the low byte is the internal number of this group, for example: 0x0a04 corresponds to the function code	
Soft Start **Control Command (06)**	0x2000	0x0021	Start command
		0x0022	Stop command
		0x0023	Reset command
System status (03)	0x2001	0x0000	System stop
		0x0001	Start delay
		0x0002	Soft starting
		0x0003	Bypass operation
		0x0004	Run online
		0x0005	Soft stopping
		0xNN06	System fault, NN is the fault code
		0x0007	Debugging
Running data (03)	0x2002	Read slave average current	
	0x2003	Read slave A phase current	
	0x2004	Read slave B phase current	
	0x2005	Read slave C phase current	
	0x2006	Read slave input voltage	
	0x2007	Read slave radiator temperature -20 degrees	

0x2100 Fault code as below:

Fault list	Fault description	Fault list	Fault description
01	System runs over current	02	System undercurrent protection
03	System motor overload	04	Electronic Insurance Protection
05	System phase sequence error	06	Over-start protection
07	System three-phase unbalance	08	System main power phase loss
09	System output phase loss	0A	System mains overvoltage
0B	System main power undervoltage	0c	System radiator overheating
0D	Start overtime	0E	System communication error
0F	External control terminal error	10	System parameters are missing
11	System CPU error	00	No system alarm