ЭНЕРГОМЕТРИКА
www.energometrika.ru

SPM90

MODBUS PROTOCOL
V1.3

ООО «Энергометрика», zakaz@energometrika.ru, тел.: +7(495) 276-0510, www.energometrika.ru

Content

Chapter1. Introduction

This document describes the input and output command, information and data of the SPM90 under MODBUS communication mode. So it is convenient for the 3rd part using and developing

1.1 Purpose of the Communication Protocol

The role of the SPM90 MODBUS communications protocol is to allow information and data to be efficiently transferred between MODBUS Master Station and SPM90. It includes:

1) Allowing MODBUS Master Station to set access and set all set-up parameters of SPM90.

2) Allowing reading all data measured and event records by SPM90.

1.2 Version of Communication Protocol

This document is proper for all versions of SPM90 meters. If there any change, it will be declared.

Chapter2. Detailed Description of PABT-GATE Modbus Protocol

2.1 Basic Rules of SPM90 Modbus Protocol

The following rules determine the communication rules for devices in RS485 (or RS232C or TCP Ethernet) loop controllers and other RS485 communication loops:

1） All RS485 loop communication should follow the master/slave mode. In this way, information and data are passed between a single master and up to 32 slaves (monitoring devices);

2） The MASTER will initiate and control all information transfer on the RS-485 communications loop;

3） No communication can start from one slave anyway;

4） All communication on the RS485 loop occurs in a "package" manner. A package is a simple string (8 bits per string), and a package can contain up to 255 bytes. The bytes that make up this package form the standard asynchronous data and are passed as 8-bit data bits, 1 stop bit, and no parity bit. The data stream is generated by a device similar to that used in RS232C;

5） The packages sent from MASTER are named request. The packages sent from SLAVE are named response;

6） Under any circumstance, Slave can just respond one request.

2.2 Modes of Transmission

The MODBUS protocol can transmit data in ASCII or RTU or TCP mode. SPM90 only supports MODBUS-TCP mode, 8-bit data bits, no parity bit, and 1 stop bit

2.3 Description of the Modbus Packet Structure

Every MODBUS package consists of four fields:

1) Address Field
2) Function Field
3) Data Field
4) Check Field

2.3.1 Address Field

The slave address field of MODBUS is one byte long and contains the slave address of the package transmission. Valid slave addresses range from 1 to 247. If the slave receives a parcel of slave address field information that matches its own address, the slave shall execute the commands contained in the package. The domain in the package that the slave responds to is its own address.

2.3.2 Function Field

The function field in the MODBUS package is one byte long to inform the slave what to do. The slave response package should contain the same functional field bytes requested by the master station. Refer to the table below for the function code of SPM90

Function Code	Meaning	Function
0x03	Read Holding Registers	Get one or more current register values inside the current SPM90
0x05	Relay control	Write 0xFF00 to close (ON) the relay; Write 0x0000 to open (OFF) the relay
0x10	Set Registers	Write the specified value to one or more registers inside SPM90

2.3.3 Data Field

The length of the MODBUS data field is variable depending on its specific function. The MODBUS data field uses the "BIG INDIAN" mode, where the high byte is first and the low byte is after. Examples are as follows:

Example 2.1

One 16 byte register value is 0x12AB; register is transmitted in below sequence:

High byte = 0x12

Low byte = 0x0AB

2.3.4 Check Field

The MODBUS-RTU mode uses a 16-bit CRC check. The transmitting device shall perform a CRC16 calculation on each of the data in the package, and the final result is stored into the check field. The receiving device should also perform CRC16 calculation on each data in the package (except the check field) and compare the result field and check field. Only the same package can be accepted. The specific CRC check algorithm refers to Chapter 4.

2.4 Abnormal Response

If the master sends an illegal package to SPM90 or the master requests an invalid data register, an abnormal data response will be generated. This abnormal data response consists of the slave address, function code, fault code, and check field. When the high bit position of the function code field is 1, it indicates that the data frame at this time is an abnormal response. The following table explains the meaning of the abnormal function code.

Function Code	Description
01 illegal function code	SPM90-MODBUS only supports 01H, 02H, 03H, 05H and 10H function codes, which means that the slave receives an illegal function code; or SPM90 receives an incorrect operation password.
02 illegal data address	Indicates that SPM90 received an invalid data address, or the request register is not within the valid register range
03 illegal data address	The requested register number is too long.

2.5 Broadcast command

The SPM90-MODBUS protocol supports broadcast commands (must be a write command (0x10)) for timing.

Chapter3. Communication Package

SPM90-MODBUS only supports reading function code. The standard MODBUS protocol only supports 16-bit data mode, which means that any measurement value is transmitted up to 65535.
Section 3.1 describes the format of the SPM90 read data package and response package. Section 3.2 describes the commands for relay control. Section 3.3 describe the format of the SPM90 write data package and response package.

3.1 Read Holding Registers (Function Code 03H)

The package sent by the master request responds to all valid registers of SPM90, and the reserved register contents are 0.

Standard Modbus-RTU protocol:

Read register package mode (Master→SPM90)			**Response mode (SPM90→Master)**	
Slave address	1 byte		Slave address	1 byte
03 H (Function Code)	1 byte		03 H (Function Code)	1 byte
Start address	2 bytes		Byte num. (2 * register num.)	1 byte
Registers num.	2 bytes		First register data	2 bytes
CRC check code	2 bytes		Second register data	2 bytes
			……	
			CRC check code	2 bytes

3.2 Control general purpose relay (Function Code 05H)

The general-purpose relay can be operated with the 05 command, and the relay address starts from 0. The data field is 0xFF00, the relay is closed; the data field is 0x0000, and the relay is disconnected.

Read register package mode (Master→SPM90)			**Response mode (SPM90→Master)**	
Slave address	1 byte		Slave address	1 byte
05 H (Function Code)	1 byte		05 H (Function Code)	1 byte
Start address	2 bytes		Start address	2 bytes
Data field	FF		Data field	FF
Data field	00		Data field	00
CRC check code	2 bytes		CRC check code	2 bytes

3.3 Write register (Function Code 10H)

This command allows the primary station to configure the SPM90 operating parameters. The following is the data format:

Read register package mode (Master→SPM90)			**Response mode (SPM90→Master)**	
Slave address	1 byte		Slave address	1 字节
10 H (Function Code)	1 byte		10 H (Function Code)	1 字节
Start address	2 bytes		Start address	2 字节
Registers num.	2 bytes		Registers num.	2 bytes
Byte num. (2 * register num.)	1 byte		CRC check code	2 bytes
First register data				
Second register data				
…….				
CRC check code	2 bytes			

Note:

SPM90 assumes that the written registers are contiguous from the first register;

Chapter4. Calculate CRC-16

This section describes the process of calculating CRC-16. The relevant byte in the frame is defined as a string of binary data (0, 1). The 16th checksum is obtained like this: The string data stream is multiplied by 216, then divide by the generator polynomial($X^{16}+X^{15}+X^{2}+1$), this formula is expressed in binary form as 1100000000000101. Quotient is ignored. The remainder of 16 bits is the value of CRC. When calculating the CRC-16 value, all arithmetic operations use the modulo two or exclusive OR (X0R) algorithm.

Follow the steps below to generate a checksum of CRC-16:

1) Omit the most significant bits of the generator and reverse the order of the bits. Form a new polynomial, the result is 1010000000000001 or A001 in hexadecimal.
2) Load all 1 or hexadecimal FFFFs into 16-bit registers.
3) XOR the first data byte with the low order byte in the 16-bit register and store the result in a 16-bit register.
4) Move the 16-bit register one bit to the right. If the overflow bit is 1, then go to step 5, otherwise go to step 6.
5) Perform a MOR operation on the 16-bit register with the new generator polynomial and store the result in 16 steps.
6) Repeat step 4 until the shift element is 8 times.
7) XOR the next data byte with the first byte of the 16-bit register and store the result in a 16-bit register.
8) Repeat steps 4-7 until all bytes of the packet have been XORed with a 16-bit register.

9) The contents of the 16-bit register are CRC-16.

The following example performs a CRC calculation on the byte of 6403 in hexadecimal.

step	byte	action	register	Bit#	Shift
2		Initial value	1111 1111 1111 1111		
	1	Load the first byte	0000 0000 0110 0100		
3		XOR	1111 1111 1001 1011		
4		Shift one bit to the right	0111 1111 1100 1101	1	1
5a		XOR polynomial	1101 1111 1100 1100		
4		Shift one bit to the right	0110 1111 1110 0110	2	0
4		Shift one bit to the right	0011 0111 1111 0011	3	0
4		Shift one bit to the right	0001 1011 1111 1001	4	1
5a		XOR polynomial	1011 1011 1111 1000		
4		Shift one bit to the right	0101 1101 1111 1100	5	0
4		Shift one bit to the right	0010 1110 1111 1110	6	0
4		Shift one bit to the right	0001 0111 0111 1111	7	0
4		Shift one bit to the right	0000 1011 1011 1111	8	1
5a		XOR polynomial	1010 1011 1011 1110		
	2	Load the second byte	0000 0000 0000 0011		
7		XOR	1010 1011 1011 1101		
4		Shift one bit to the right	0101 0101 1101 1110	1	1
5a		XOR polynomial	1111 0101 1101 1111		
4		Shift one bit to the right	0111 1010 1110 1111	2	1
5a		XOR polynomial	1101 1010 1110 1110		
4		Shift one bit to the right	0110 1101 0111 0111	3	0
4		Shift one bit to the right	0011 0110 1011 1011	4	1
5a		XOR polynomial	1001 0110 1011 1010		
4		Shift one bit to the right	0100 1011 0101 1101	5	0
4		Shift one bit to the right	0010 0101 1010 1110	6	1
5a		XOR polynomial	1000 0101 1010 1111		
4		Shift one bit to the right	0100 0010 1101 0111	7	1
5a		XOR polynomial	1110 0010 1101 0110		
4		Shift one bit to the right	0111 0001 0110 1011	8	0
		CRC-16	0111 0001 0110 1011		

Chapter5. SPM90 Register Description

All SPM90 registers have a base address of 4XXXX in the MODBUS communication protocol. According to the MODBUS protocol, when a register with address 4XXXX in SPM90 is requested, the primary station actually reads XXXX-1. For example, request the 40011 register in SPM90, the actual station register number is 10.

5.1 Real-time measurement data register

Register num	Attribute	Data type	Description	Remark
40001	RO	U16	Voltage	Calculation factor 0.1, unit V；
40002	RO	U16	Current	Calculation factor 0.01, unit A；
40003	RO	U16	Active power high	Active power, calculation factor 0.1, unit W
40004	RO	U16	Active power low	
40005	RO	U16	Total active energy high	Active energy, calculation factor 0.01, unit kW • H
40006	RO	U16	Total active energy low	
40007	RO	U16	Measuring voltage direction	0 means the measurement voltage is positive 1 means the measurement voltage is reversed Note: This version of the program does not have the function of judging the voltage reversal. The default value is consistent to the positive direction.
40008~40016	remain	remain	remain	remain

5.2 Device parameter data register

Register num	Attribute	Data type	Description	Remark
41001	RW	U16	Address	1--247
41002	RW	U16	Baud rate	0--3 0: 2400 1：4800 2：9600 3：19200
41003	RW	U16	Communication default	0：MODBUS 1：DLT645
41004	Remain	Remain	Remain	Remain
41005	RW	U16	Wiring	0：Standard wiring 1：Non-standard wiring
41006	Remain	Remain	Remain	Remain
41007	RW	U16	DLT645 communication address byte 1、0	0-9999

41008	RW	U16	DLT645's communication address byte 3, 2	0-9999
41009	RW	U16	DLT645's communication address byte 5, 4	0-9999
41010	RW	U16	Password byte 1, 0	0-9999
41011	RW	U16	Password byte 1, 0	0-9999
41012	Remain	Remain	Remain	Remain