Baoding Shenchen Precision Pump Co., Ltd.
Add: No.103, Building 2, ZhiDian Industrial Park, FuXing East Road 999, Baoding, China.
Tel: 0086-312-5958380 Fax: 0086-312-6780636
Website: www.good-pump.com Email: amy@good-pump.com

Standard MODBUS Communication Protocol

Note: The hexadecimal numbers are expressed by 'XXXXH' or 'XXH' in the below description.

1. MODBUS-RTU Standard Communication Format

This communication uses MODBUS RTU mode, message frame as below:

Slave address	Function code	Data area	CRC Check (Cyclic Redundancy Check)	
1 Byte	1 Byte	0 or up to 252 bytes	2 Bytes	
			CRC low	CRC high

(1) **Slave address**: Host control peristaltic pump address No.. The pump address No. should not be same when they are in the same 485 line. The address No. range is 1~32, 0 means broadcast.

(2) **Function code**: The protocol uses 2 common function codes which defined by MODBUS protocol.

03H: Read the contents of holding register

06H: Write a word to the holding register

10H: Write a long type to holding register

(3) Data zone: Specific information instructions that the peristaltic pump needs to execute, such as start/stop, direction, accelerate/decelerate etc.

(4) **CRC check**: CRC code is 2 bytes, 16 check codes. Use CRC-16(which used in American binary synchronous system).

Polynomial: G(X)=X16+X15+X2+1**.**

CRC check C language code please refer to Appendix 1.

2. Communication Setting

(1) Communication baud rate: 1200, 2400, 4800, 9600 optional

(2) **Byte structure:** 1 start bit + 8 data bits +1 even parity bit + 1 stop bit

(3) Bit sequence sending order: The least significant big(**LSB**)….. The most significant bit (**MSB**)

Start	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Check**	**Stop**

(4) Data transferring format:

Integer (2 bytes):

Data: (High bit) The second byte The first byte (Low bit)

Send: The second byte The first byte

For example: 1234H send 12H 34H

Floating-point type (4 bytes):

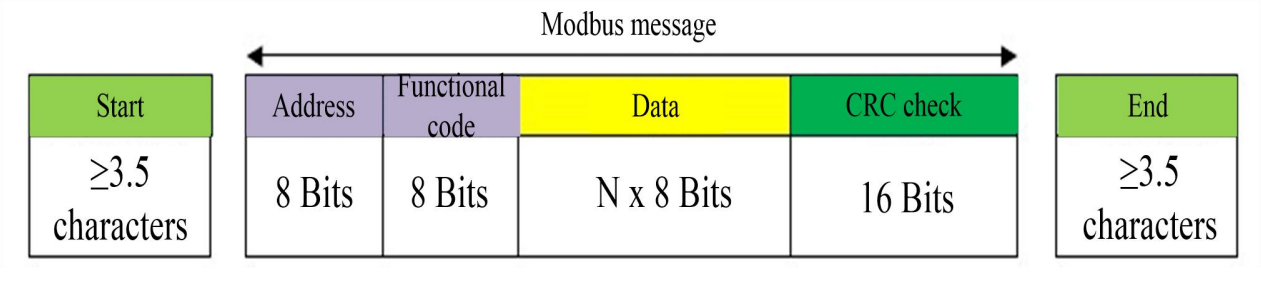
Data: (High bit) The fourth byte The third byte The second byte The first byte(Low bit)

Send: The fourth byte The third byte The second byte The first byte

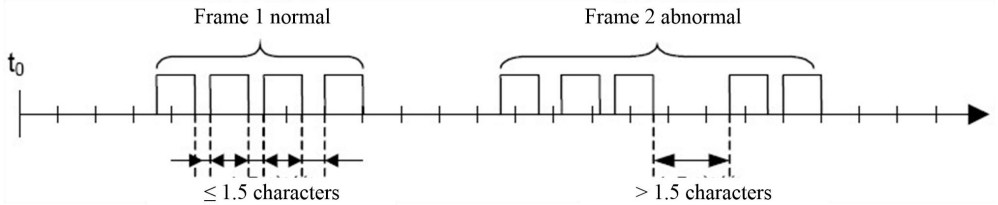
For example: 8.9 send 41H 0EH 66H 66H

3. MODBUS Message RTU Frame Format

In RTU mode, the message frames are distinguished by idle intervals with a duration of at least 3.5 characters. As the follow picture:



The entire message frame must be sent in a continuous stream of characters. If the idle space between two characters is greater than 1.5 character times, the message frame is considered as incomplete, should be discarded by receiving node. As below:



4. Abnormal Response

When host sending request data, the slave receives abnormal data, an abnormal response is required. If the address code sent from host is wrong, there is no this address code between slaves or the data received by slave is wrong when CRC check, no abnormal code return, the host should have super response

mechanism.

Function code domain: Abnormal response function code is normal response function code +80H.

Data domain: Return to abnormal code, define as below:

Chart 1: Abnormal code definition

Code	Name	Meaning
01H	Illegal function code	The function code received by peristaltic pump except 03H/06H/10H.
02H	Illegal data address	This abnormal code means the register address is not allowed data which received by peristaltic pump.
03H	Illegal data value	Written data does not meet the operating range.
06H	Slave (peristaltic pump) busy	The current state of the peristaltic pump conflict with the command received, unable to complete the command.

Peristaltic Pump only receive MODBUS command with the Main Interface, other interface do not receive message.

5. Holding Register Address And Content

Basic Parameters Setting

Address (Decimal)	Name	Range	Data Type
1000	Start/Stop Control	1: Start 0: Stop	unsigned short int (2 Bytes)
1001	Direction Control	1: clockwise 0: counterclockwise	unsigned short int (2 Bytes)
1002	Run speed	0.1-600rpm	Float (4 Bytes)

6. Send Data Format

Unsigned short int format

Peristaltic pump address	Function code	Register address		Data (unsigned short int)		CRC check	
	06H	Address H	Address L	Data H	Data L	L	H

Float format

Peristaltic pump address	Function code	Register address		Register quantity		The number of bytes	Data (Float)				CRC check	
	10H	H	L	00H	02H	04H	L1	L2	H1	H2	L	H

(1) Start/stop control

The peristaltic pump address is 1, 0001H to control start, 0000H to control stop.

Send start: 01 06 03 E8 00 01 C8 7A

Back: 01 06 03 E8 00 01 C8 7A

Send stop: 01 06 03 E8 00 00 09 BA

Back: 01 06 03 E8 00 00 09 BA

(2) Direction control

The peristaltic pump address is 1, clockwise is 0001H, counterclockwise is 0000H.

Send: 01 06 03 E9 00 01 99 BA (Clockwise)

Back: 01 06 03 E9 00 01 99 BA

(3) Setting the motor speed

The peristaltic pump address is 1, set the motor speed to 58.8 rpm.

Send: 01 10 03 EA 00 02 04 42 6B 33 33 58 29

Back: 01 10 03 EA 00 02 60 78

7. Appendix 1——CRC Check C Language Code

CRC generation process:

1) Put one 16 bits register into hexadecimal FFFF(all 1), we call it CRC register.

2) Make the first 8 bytes of the message with low bytes XOR of 16 CRC register , the result put in CRC register.

3) Detect LSB of CRC register

If LSB is 0, move CRC register 1 bit to right (towards to direction of LSB) , MSB zeroing.

If LSB is 1, move CRC register 1 bit to right (towards to direction of LSB), MSB zeroing, then XOR the polynomial value of the CRC register 0xA001 (1010 0000 0000 0001).

4) Repeat Step 3, until finish 8 shifts. After finish this operation, will finish the complete operation for 8 Bytes.

5) Repeat Step 2 to Step 5 for the next Bytes in message. Continue this operation till all the message be deal with finished.

6) The final content in CRC register is CRC value.

7) When put CRC value in message, high and low Bytes must be exchanged, described as below:

C language code:

```
void CRCVerify(char *rec,char CRClen,char CRCdata[2])
{
    char i1,j;
    unsigned int crc_data=0xffff;
    for(i1=0; i1<CRClen; i1++)
    {
        crc_data=crc_data^rec[i1];
        for(j=0; j<8; j++)
        {
            if(crc_data&0x0001)
            {
                crc_data>>=1;
                crc_data^=0xA001;
            }
            else
```

```
                {
                    crc_data>>=1;
                }
            }
        }
        CRCdata[0]=(char)(crc_data);
        CRCdata[1]=(char)(crc_data>>8);
    }
```