YOSEMITECH

pH Sensor

MODBUS RTU
Programmer Manuel

Table of Contents

1 MODBUS RTU Overview

1.1 Scope

This document is about MODBUS of optical pH probes with hardware Rev1.1 and software Rev1.1 or later. This document is intended for software programmers with detailed information about MODBUS RTU protocols.

1.2 MODBUS Command Structure

Data format in this document:

----Binary number – shown with suffix B. For example: 10001B

----Decimal number – without nay suffix. For example: 256

----Hexadecimal number—shown with prefix 0x. For example: 0x2A

----ASCII character or string – shown with quotation marks. For example:" YL4314010022"

1.2.1 Command Structure

MODBUS defines a simple protocol data unit (PDU), which is transparent to communication layer.

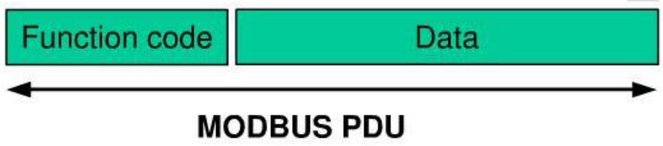

Figure 1: MODBUS Protocol Data Unit

The mapping of MODBUS protocol on a specific bus or network introduces some additional fields on the Protocol Data Unit. The client that initiates a MODBUS transaction builds the MODBUS PDU, and then adds fields in order to build the appropriate communication PDU.

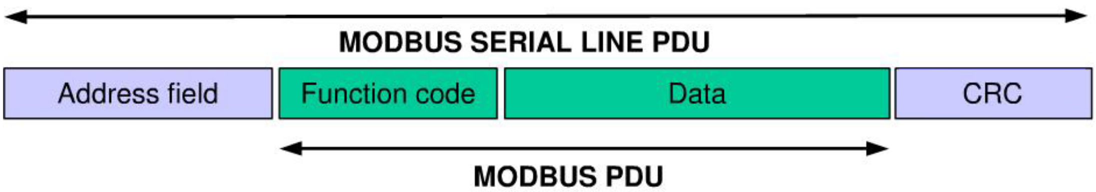

Figure 2: MODBUS Structure for Serial Communication

On a MODBUS serial bus, address field only includes addresses for slave devices.

Note:

- Slave address range for optical pH sensor is: 1…247
- Master device sends a "request frame" with a targeted slave address. When slave device responses, it has to put its own address in the "response frame", so that master device knows where the response comes from.
- Function code indicates type of operations
- CRC is the result of redundancy check.

1.2.2 MODBUS RTU Transmission Mode

When devices communicate on a MODBUS using RTU (remote terminal unit) mode, each 8-bit byte message contains two 4-bit hexadecimal characters. The main advantage of the RTU mode is that it has higher character density, which enables better throughput compare to ASCII mode at same baud rate. Each RTU message must be transmitted in a continuous string of characters.

RTU mode format for each byte (11 bits):

Encoding system 8 bit binary

Each 8-bit packet contains 4-bit hexadecimal characters (0-9, A-F)

Bit per byte: 1 start bit

8 data bits, least significant bit first

No parity check

2 stop bits

Baud rate: 9600bps

Serial transmission of characters:

Every character or byte is sent under this sequence (left to right):

Least Significant Bit (LSB)……Most Significant Bit(MSB)

Start	1	2	3	4	5	6	7	8	Stop

Figure 3: RTU Mode Bit Sequence

CRC Field Structure:

Redundancy check (CRC16)

Frame Structure:

Slave address	Function Code	Data	CRC	
1 byte	1 byte	0…252 bytes	2 bytes	
			CRC Low	CRC High

Figure 4: RTU Message Frame Structure

Maximum size of MODBUS frame is 256 bytes.

1.2.3 MODBUS RTU Message Frame

In RTU mode, message frames need to be separated by an idle interval of at least 3.5 character lengths. In rest of this document, this idle interval is called t3.5.

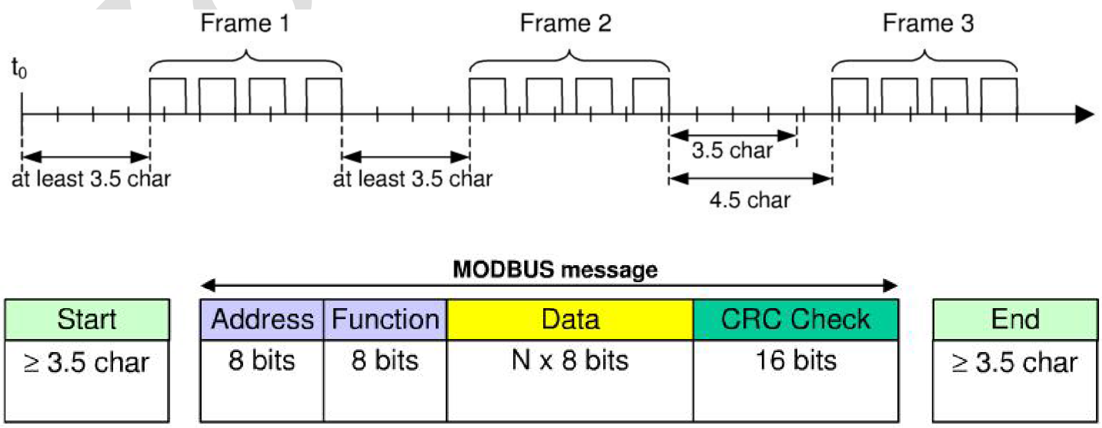

Figure 5: RTU Message Frame

Entire message frame must be sent as continuous stream of characters.

If idle time between two characters is longer than 1.5 characters, the message frame will be considered incomplete, and will be discarded by receiving side.

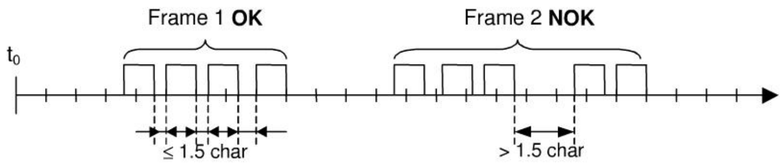

Figure 6: Frame transmission

1.2.4 MODBUS RTU CRC Check

In RTU mode, the error checking field is based on a cyclical redundant checking (CRC) method. The CRC field checks entire content of MODBUS message, regardless of the existence of parity check bit. CRC16 checking method is utilized. CRC result is a 16-bit value with two 8-bit bytes, low order 8-bit byte first followed by high order 8-bit byte.

1.3 MODBUS RTU for YOSEMITECH's Optical pH Sensor

Based on standard MODBUS definition, message frame starts with t3.5 idle interval, and similarly, ends with t3.5 idle interval. Device address and Function code are both 8-bit byte. Data character string has n*8 bits, it contains information about register start/end address and number of registers for read/write operation. CRC field is 16 bit in length.

	Start	Device address	Function code	Data	CRC		End
Value	Idle for 3.5 character length	1-247	Comply with MODBUS function code format	Comply with MODBUS data format	CRC Low	CRC High	Idle for 3.5 character length
Length (byte)	3.5	1	1	n	1	1	3.5

Figure 7: Message frame structure for Yosemitech's MODBUS

1.4 MODBUS RTU Function Code for YOSEMITECH's Optical pH Sensor

YOSEMITECH's optical pH sensor has two MODBUS function codes:

0x03: Read registers 0x10： Write registers

1.4.1 MODBUS Function Code 0x03: Read Registers

This function code is to read a block of continuous registers from a remote device. Request PDU defines start address and number of registers for the read operation. Register addressing starts from 0. Therefore, addresses for register 1-16 are 0-15. Data for each register in Response message have two bytes. For each register data, first byte is for high bits, and second byte for low bits.

Request Frame:

Function code	1 Byte	0x03
Start address	2 Bytes	0x0000….0xfffff
Number of registers	2 Bytes	1…125

Figure 8: Request frame for read registers

Response Frame:

Function code	1 byte	0x03
Number of byte	1 byte	N×2
Register data	N×2 bytes	

N = number of registers

Figure 9: Response frame for read registers

Below is an example of Request and Response frames (Read register 108-110. Register 108 is read only with 2-byte value of 0X022B. Registers 109-110 have values of 0X0000 and 0X0064).

Request Frame		Response Frame	
Data format	Hexadecimal	Data Format	Hexadecimal
Function code	0x03	Function code	0x03
Start address(high bits)	0x00	Number of bytes	0x06
Start address (low bits)	0x6B	Register value (high bits, 108)	0x02
Number of registers (high bits)	0x00	Register value (low bits, 108)	0x2B
Number of registers (low bits)	0x03	Register value (high bits, 109)	0x00
		Register value (low bits, 109)	0x00
		Register value (high bits, 110)	0x00
		Register value (low bits, 110)	0x64

Figure 10: Example of request frame and response frame for read operation

1.4.2 MODBUS Function Code 0x10: Write Registers

This function code is to write a block of continuous registers at a remote device. Request frame contains register data. Each register data have two character bytes. Response frame contains function code, start address, and number of registers that completed write operation.

Request Frame:

Function code	1 byte	0x10
Start address	2 bytes	0x0000….0xffff
Number of registers	2 bytes	0x0001….0x0078
Number of bytes	1 byte	N×2
Register data	N×2 bytes	value

N = number of registers

Figure 11: Request frame for write operation

Response Frame:

Function Code	1 byte	0x10
Start address	2 bytes	0x0000….0xffff
Number of registers	2 bytes	1…123(0x7B)

N = number of registers

Figure 12: Response frame for write operation

Below is an example of Request Frame and Response frame (write 0x000A and 0x0102 to two registers starting from address 2):

Request Frame		Response Frame	
Data Format	Hexadecimal	Data Format	Hexadecimal
Function code	0x10	Function code	0x10
Start address (high bits)	0x00	Start address (high bits)	0x00
Start address (low bits)	0x01	Start address (low bits)	0x01
Number of registers (high bits)	0x00	Number of registers (high bits)	0x00
Number of registers (low bits)	0x02	Number of registers (low bits)	0x02
Number of bytes	0x04		
Register value (high bits)	0x00		
Register value (low bits)	0x0A		
Register value (high bits)	0x01		
Register value (low bits)	0x02		

Figure 13: Example of Request frame and response frame for write operation

1.5 Data formats in optical pH sensor

1.5.1 Floating-point number

Definition: floating point number, comply with IEEE754 (single precision)

Note	Sign	Exponent	Fraction	Total
bit	31	30…23	22…0	32
Exponent deviation	127			

Figure 14: Single floating point number definition (4 bytes, 2 MODBUS registers)

Example: Convert decimal number 17.625 to binary number

Step 1: Convert decimal number 17.625 to a floating point number with binary format

First, convert integer to binary

$17_{decimal} = 16 + 1 = 1\times2^4 + 0\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0$

Thus, integer 17 in binary format is 10001B

Then convert decimal part to binary

$0.625_{decimal} = 0.5 + 0.125 = 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$

Thus, 0.625 in binary format is 0.101B

Combine above together, 17.625 in binary format is 10001.010B

Step 2: Calculate exponent

Left shift the binary number 10001.010B until only bit left before the decimal point --- 10001.101B = 1.0001101 B$\times 2^4$, so exponent value is 4. By adding 127, we have 131, which is 10000011B in binary format

Step 3: Get fraction

Fraction is simply the number after decimal point. Thus from 1.0001101B, fraction number is 0001101B. IMPORTANT NOTE about the 23 bit fraction number: the first bit which on the left side of decimal point is hidden bit and does not need to be compiled.

Step 4: Sign definition

Sign bit is 0 if the number is positive. Sign is 1 if the number is negative. For 17.625, sign 17.625, sign bit is 0.

Step 5: Convert to floating point number

1 Sign bit	+	8-bit exponent	+	23-bit fraction
0		10000011		00011010000000000000000B

(Corresponding hexadecimal number is 0x418D0000)

Sample code:

1. If your compiler has similar library functions, it can be called directly. For example if C language is used, we can directly call memcpy() function in C library to convert floating point number. Sample code:

```
float     floatdata;//floating point data to be converted
void*    outdata;
memcpy(outdata,&floatdata,4);
```

If floatdata=17.625,

In little-endian storage mode after the function is called:

Value at address of outdata is 0x00
Value at address of (outdata+1) is 0x00
Value at address of (outdata+2) is 0x8D
Value at address of (outdata+3) is 0x41

In big-endian storage mode after the function is called:

Value at address of outdata is 0x41
Value at address of (outdata+1) is 0x8D
Value at address of (outdata+2) is 0x00
Value at address of (outdata+3) is 0x00

2. If your complier doesn’t have the conversion function, then the following function can be used:

```
void memcpy(void *dest,void *src,int n)
{
    char *pd = (char *)dest;
    char *ps = (char *)src;
    for(int i=0;i<n;i++)    *pd++ = *ps++;
}
```

Then you can get same result by calling this function memcpy(outdata,&floatdata,4);

Example: Convert binary floating point number 0100 0010 0111 1011 0110 0110 0110 0110B to a decimal number

Step 1: Separate this binary number 0100 0010 0111 1011 0110 0110 0110 0110B and get values of Sign , exponent and fraction.

0	10000100	11110110110011001100110B
1 Sign bit	8-bit exponent	23-bit fraction

Sign bit(s): 0

Exponent(E):10000100B=$1\times2^7+0\times2^6+0\times2^5+0\times2^4+0\times2^3+1\times2^2+0\times2^1+0\times2^0$

$=128+0+0+0+0+4+0+0=132$

Fraction(M)： 11110110110011001100110B =8087142

Step 2: Calculate decimal value

$D = (-1)^S \times (1.0+M/2^{23}) \times 2^{E-127}$

$= (-1)^0 \times (1.0+8087142/2^{23}) \times 2^{132-127}$

$= 1 \times 1.964062452316284 \times 32$

$= 62.85$

Reference code:

```
float floatTOdecimal(long int byte0, long int byte1, long int byte2, long int byte3)
{
        long int realbyte0,realbyte1,realbyte2,realbyte3;
        char S;
          long int    E,M;
        float D;
        realbyte0 = byte3;
        realbyte1 = byte2;
        realbyte2 = byte1;
        realbyte3 = byte0;

        if((realbyte0&0x80)==0)
        {
            S = 0; //Positive
          }
          else
        {
            S = 1; //Negative
        }
        E = ((realbyte0<<1)|(realbyte1&0x80)>>7)-127;
        M = ((realbyte1&0x7f) << 16) | (realbyte2<< 8)| realbyte3;
        D = pow(-1,S)*(1.0 + M/pow(2,23))* pow(2,E);
        return D;
}
```

Note:

- Function parameters byte0, byte1, byte2 and byte3 represent the 4 sections of a binary floating number.
- Return value is value of decimal number after conversion

For example when a command is sent to a sensor to get pH value, response frame from the sensor will have measured pH. If the values are 4 byte floating point number 0x33, 0x33, 0xF3, 0x40, then the following function can be used to get pH in decimal value:

float pH = floatTOdecimal(0x33, 0x33, 0xF3, 0x40);

and pH = 7.6.

1.5.2 Characters

Definition: Character is shown by ASCII code.

Example: String "YL"could be shown by corresponding ASCII codes (refer to ASCII character chart)

"Y" is 0x59

"L" is 0x4C

2 MODBUS RTU Commands for Optical pH Sensor

2.1 Overview

In order to communicate with optical pH probe via MODBUS RTU, master terminal software will be needed. MODBUS RTU is an open standard. There are free commercial software tools available. For applications described in this document, MODBUS register address starts from 1. However, slave address in MODBUS protocol starts from 0, and usually master software compiles addresses. For example, register address 2090 will be compiled by master software as address 2089.

2.2 Command Description

2.2.1 Set Slave Device ID

Purpose: Set MODBUS slave address to a sensor probe. Range of address is 1~247.

Sensor probe slave address can be set via MODBUS register 0x3000:

Start address	Number of registers	Register 1	MODBUS Function code
0x3000	0x01	New Slave address	0x10

Figure 15: Register definition of Set Slave ID Command

Below is an example of request and response frames for setting slave device ID command. Old slave address is 0x01, new address is 0x14.

Definition	Address	Function code	Start address		Number of registers		Number of byte	Register value		CRC	
Byte	0	1	2	3	4	5	6	7	8	9	10
Value	0x01	0x10	0x30	0x00	0x00	0x01	0x02	0x14	0x00	0x99	0x53

Figure 16: Request frame to Set Slave ID

*Note: byte 8 is reserved

Definition	Address	Function code	Start address		Number of registers			CRC
Byte	0	1	2	3	4	5	6	7
Value	0x01	0x10	0x30	0x00	0x00	0x01	0x0E	0xC9

Figure 17: Response frame for Set Slave ID Command

2.2.2 Get SN

Purpose: Get sensor probe's serial number (SN). Each sensor probe has a unique SN.

Serial Number can be read from 7 continuous MODBUS registers starting from address 0x0900.

Start Address	Number of registers	Register 1-7	MODBUS Function code
0x0900	0x07	SN	0x03

Figure 18: Register definition of Get SN Command

Below is an example of request and response frames to get SN "YL4314010022" from a slave device (address 0x01) .

Definition	Address	Function code	Starting address		Number of registers		CRC	
Byte	0	1	2	3	4	5	6	7
Value	0x01	0x03	0x09	0x00	0x00	0x07	0x07	0x94

Figure 19: Request frame to Get SN Command

Definition	Address	Function code	Number of byte	Register value			CRC	
Byte	0	1	2	3	4-15	16	17	18
Value	0x01	0x03	0x0E	0x00	"YL4314010022"	0x00	0xAD	0x9C

Figure 20: Response frame for Get SN Command

Note: SN value is in ASCII code as below:

Byte	4	5	6	7	8	9	10	11	12	13	14	15
Value	0x59	0x4C	0x34	0x33	0x31	0x34	0x30	0x31	0x30	0x30	0x32	0x32

Figure 21: Sensor probe's SN

2.2.3 Get pH value

Purpose：Get pH measurement results. Temperature compensation is automatically applied to pH value.

pH data can be read from 2 continuous MODBUS registers starting from address 0x2800.

Start address	Number of registers	Register 1-2	MODBUS function code
0x2800	0x02	pH value	0x03

Figure 22: Request frame to Get pH Command

Below is an example of request and response frames for getting pH command, assuming slave device address is 0x01, returned pH value is 7.6.

Definition	Device address	Function code	Start address		Number of registers		CRC	
Byte	0	1	2	3	4	5	6	7
Value	0x01	0x03	0x28	0x00	0x00	0x02	0xCD	0xAB

Figure 23: Request frame to Get pH Command

Definition	Device address	Function code	Number of bytes	Register value	CRC	
Byte	0	1	2	3-6	11	12
Value	0x01	0x03	0x04	7.6	0x40	0x78

Figure 24: Response frame for Get pH Command

Note: pH values are floating point number in little-endian storage mode. See sample below:

pH value（3-6）			
0x33	0x33	0xF3	0x40

Figure 25: Registers definition for pH values.

2.2.4 Get Software and Hardware Rev

Purpose: Get current hardware and software Release Version

Hardware and software release version numbers of a sensor probe can be read from 2 continuous registers starting from address 0x0700.

Start address	Number registers	Register 1	Register 2	MODBUS function code
0x0700	0x02	HW Rev	SW Rev	0x03

Figure 26: Register definition for Get Software and Hardware Rev Command

Below is an example of request and response frames for getting hardware and software release version, assuming device slave address is 0x01, returned value for hardware Rev is 1.1 and software rev is 1.1.

Definition	Device address	Function code	Start address		Number of registers		CRC	
Byte	0	1	2	3	4	5	6	7
Value	0x01	0x03	0x07	0x00	0x00	0x02	0xC5	0x7F

Figure 27: Request frame to Get Hardware and Software Rev Command

Definition	Device address	Function code	Number of bytes	Register value				CRC	
Byte	0	1	2	3-4		5-6		7	8
Value	0x01	0x03	0x04	0x01	0x01	0x01	0x01	0x6A	0x5F

Figure 28: Response frame for Get Hardware and Software Rev Command

2.2.5 Get potential value

Purpose：Get potential measurement results, the unit is mV.

Potential data can be read from 2 continuous MODBUS registers starting from address 0x1200.

Start address	Number of registers	Register 1-2	MODBUS function code
0x1200	0x02	potential value	0x03

Figure 29: Request frame to Get potential Command

Below is an example of request and response frames for getting potential command, assuming slave device address is 0x01, returned potential value is -10.28.

Definition	Device address	Function code	Start address		Number of registers		CRC	
Byte	0	1	2	3	4	5	6	7
Value	0x01	0x03	0x12	0x00	0x00	0x02	0xC1	0x73

Figure30: Request frame to Get potential Command

Definition	Device address	Function code	Number of bytes	Register value	CRC	
Byte	0	1	2	3-6	11	12
Value	0x01	0x03	0x04	-10.28	0x37	0x46

Figure 31: Response frame for Get potential Command

potential value（3-6）			
0xE1	0x7A	0x24	0xC1

Figure 32: Registers definition for potential values.

2.2.6 Start pH calibration

Purpose：pH sensor can support three-point calibration. Standard solutions are:

4.00	6.86	9.18	（25℃）

The sequence for three-point calibration is in the order 4.00pH- 6.86pH - 9.18pH. Calibration procedure is following:

The first step: Put the sensor into the first calibration standard solution and wait for at least 1 minute.

The second step: Start calibration with inputting the pH value of the standard solution.

Calibration can be start from 2 continuous MODBUS registers starting from address 0x2300.

Start address	Number of registers	Register 1-2	MODBUS function code
0x2300	0x02	One pH value	0x10

Figure 33: Register definition of Start pH calibration Command

Below is an example of request and response frames for a device ,assuming slave address 0x01 ,pH value is 4.00

Definition	Device address	Function code	Start address		Number of registers		Number of bytes	Register value	CRC	
Byte	0	1	2	3	4	5	6	7-10	15	16
Value	0x01	0x10	0x23	0x00	0x00	0x02	0x04	4.00	0x1E	0xAE

Figure 34: Request frame to Start pH calibration Command

Definition	Device address	Function code	Start address		Number of registers		CRC	
Byte	0	1	2	3	4	5	6	7
Value	0x01	0x10	0x23	0x00	0x00	0x02	0X4A	0x4C

Figure 35: Response frame for Start pH calibration Command

The third step: Repeat above two steps to calibrate second and third point.

The fourth step: Read calibration status to determine whether the calibration is successful.

Calibration success criteria are:

pH value is stable for more than 1 minute;

The current measurement pH data must match one of the Standard solutions;

Calibration coefficients should be within a reasonable range.

2.2.7 Get Calibration Status

Purpose: If calibration is always failed (the calibration coefficients will recover the last automatically), user can read below register to analysis problem.

Calibration status can be read from 1 MODBUS registers starting from address 0x0E00.

Start address	Number of registers	Register 1	MODBUS function code
0x0E00	0x01	Status	0x03

Figure 36: Register definition of Get Calibration Status Command

Below is an example of request and response frames for getting calibration status from a device with slave address 0x01.

Definition	Device address	Function code	Start address		Number of registers		CRC	
Byte	0	1	2	3	4	5	6	7
Value	0x01	0x03	0x0E	0x00	0x00	0x01	0x86	0xE2

Figure 37: Request frame to Get Calibration Status Command

Definition	Device address	Function code	Number of bytes	Register value		CRC	
Byte	0	1	2	3	4	11	12
Value	0x01	0x03	0x02	Status	0		

Figure 38: Response frame for Get Calibration Status Command

Statue	00	01	02	04
Definition	success	Not match Calibration Standard	Less than three points	Calibration coefficients out of range

Figure 39: Registers for Calibration Status.

2.2.8 Get Temperature Value

Purpose: Get current temperature value of the pH solution to be measured .

The temperature value can be read from 2 continuous MODBUS registers starting from address 0x2400.

Start address	Number of registers	Register 1-2	MODBUS function code
0x2400	0x02	temperature	0x03

Figure 40: Request frame to Get temperature value Command

Below is an example of request and response frames for getting temperature value command, assuming slave device address is 0x01, returned temperature value is 15.8℃.

Definition	Device address	Function code	Start address		Number of registers		CRC	
Byte	0	1	2	3	4	5	6	7
Value	0x01	0x03	0x24	0x00	0x00	0x02	0xCE	0xFB

Figure 41: Request frame to Get temperature value Command

Definition	Device address	Function code	Number of bytes	Register value	CRC	
Byte	0	1	2	3-6	11	12
Value	0x01	0x03	0x04	15.8	0xE4	0x50

Figure 42: Response frame for Get temperature value Command

temperature value（3-6）			
0xCD	0xCC	0x7C	0x41

Figure 43: Registers definition for temperature value.

2.2.9 Set Customer Calibration Coefficients

Purpose**:** Recover pH factory calibration coefficients K1~K6.**K1=6.86, K2=-6.72, K3=0.04, K4=6.86, K5=-6.56, K6=-1.04** by default.

Customer coefficients can be set at 12 continuous MODBUS registers starting from address 0x2900.

Start address	Number of	Register 1-12	MODBUS function

	registers		code
0x2900	0x0C	K1~K6	0x10

Figure 44: Register definition of Set Customer Calibration Coefficients Command

Below is an example of request and response frames for setting customer calibration coefficients, assuming slave address is 0x01.

Definition	Device address	Function code	Start address		Number of registers		Number of bytes	Register value	CRC	
Byte	0	1	2	3	4	5	6	7-30	15	16
Value	0x01	0x10	0x29	0x00	0x00	0x0C	0x18	K1~K6	0x94	0x59

Figure 45: Request frame to Set Customer Calibration Coefficients Command

Note: Coefficients K1~K6, floating point numbers in little-endian storage mode

K1(7~10)				K2(11~14)				K3(15~18)				K4(19~22)			
0x1f	0x85	0xdb	0x40	0x3d	0x0a	0xd7	0xc0	0x0a	0xd7	0x23	0x3d	0x1f	0x85	0xdb	0x40
K5(23~26)				K6(27~30)											
0x85	0xeb	0xd1	0xc0	0xb8	0x1e	0x85	0xbf								

Figure 46: Registers for coefficients K1~K6

Definition	Device address	Function code	Start address		Number of registers		CRC	
Byte	0	1	2	3	4	5	6	7
Value	0x01	0x10	0x29	0x00	0x00	0x0C	0xC8	0x50

Figure 47: Response frame for Set Customer Calibration Coefficients Command

3 Procedure to get pH value

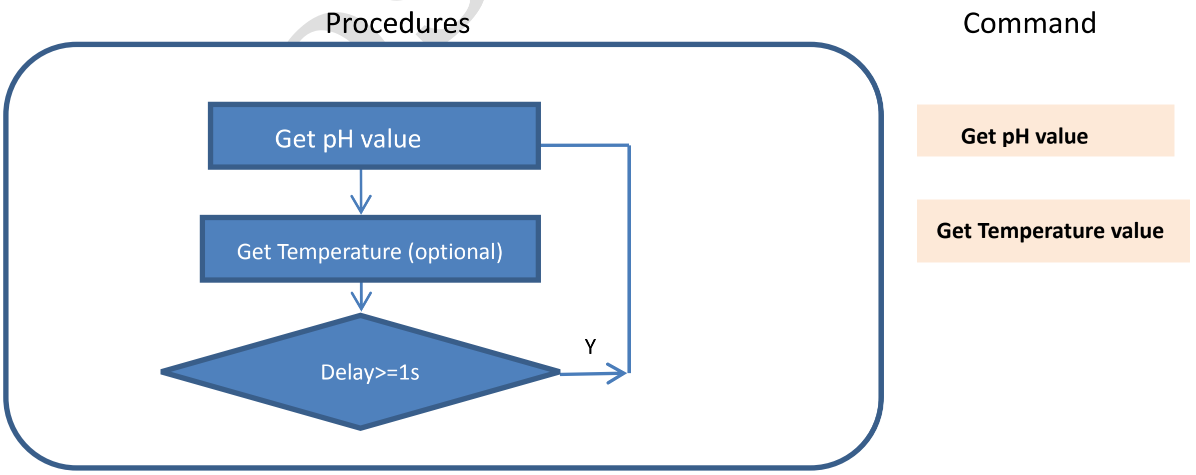

Figure 48: Flow chart to get pH measurement