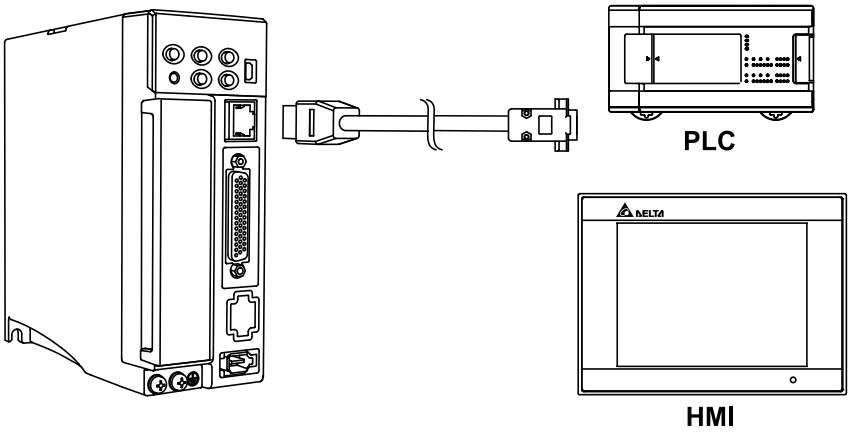
9 Modbus Communication

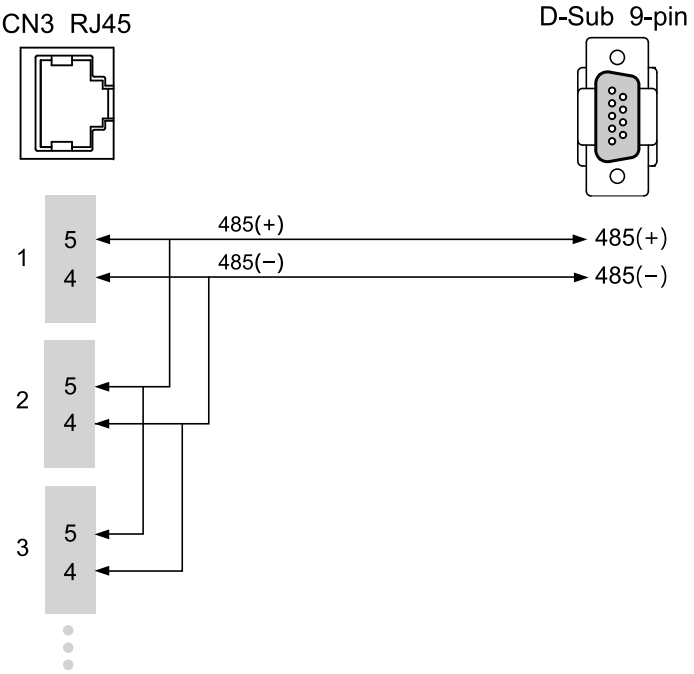
This chapter describes the Modbus communication which you use for reading and writing general parameters. For the motion control network, refer to the related DMCNET, CANopen, EtherCAT, and PROFINET documentation. The details of ASCII and RTU modes are also provided in this chapter.

9

9.1 RS-485 communication interface (hardware)

The servo drive supports RS-485 serial communication that you can use to access and change the parameters of the servo system. See the following description of the wiring:





Note:

1. The cable length can be up to 100 meters when the servo drive is installed in a quiet environment. If the required transmission speed is over 38,400 bps, a 15-meter cable is recommended to ensure data transmission accuracy.
2. The numbers 4 and 5 in the preceding figure represent the pin number of each connector.
3. Use 12 V_{DC} for the power supply.
4. When using RS-485 communication, you may connect up to 32 servo drives. Install a repeater to connect more servo drives (the maximum is 127 stations).
5. Refer to Wiring for the CN3 connector in Chapter 3.

9.2 RS-485 communication parameter settings

The required parameters for a single servo drive connection are: P3.000 (Address), P3.001 (Transmission speed), and P3.002 (Modbus communication protocol). P3.003 (Modbus communication error handling), P3.004 (Modbus communication timeout), P3.005 (Modbus communication), P3.006 (Digital input (DI) control switch), and P3.007 (Modbus communication response delay time) are optional settings. Refer to Chapter 8 for detailed descriptions of the relevant parameters.

9.3 Modbus communication protocol

There are two modes of Modbus network communication: ASCII (American Standard Code for Information Interchange) and RTU (Remote Terminal Unit). You can set the communication protocol (ASCII or RTU) with P3.002 according to your requirements. The servo drive also supports these functions: reading multiple words (03H), writing single word (06H), and writing multiple words (10H). Refer to the following descriptions.

Note: the servo drive does not support the broadcast mode.

Code description

ASCII mode:

In ASCII mode, data is transmitted in ASCII (American Standard Code for Information Interchange) format. For instance, to transmit “64H” between the master and slave, the ASCII codes “36H” and “34H” are sent to represent “6” and “4” respectively.

The corresponding ASCII codes for the numbers 0 to 9 and the characters A to F are as follows:

Symbol	‘0’	‘1’	‘2’	‘3’	‘4’	‘5’	‘6’	‘7’
ASCII code	30H	31H	32H	33H	34H	35H	36H	37H
Symbol	‘8’	‘9’	‘A’	‘B’	‘C’	‘D’	‘E’	‘F’
ASCII code	38H	39H	41H	42H	43H	44H	45H	46H

RTU mode:

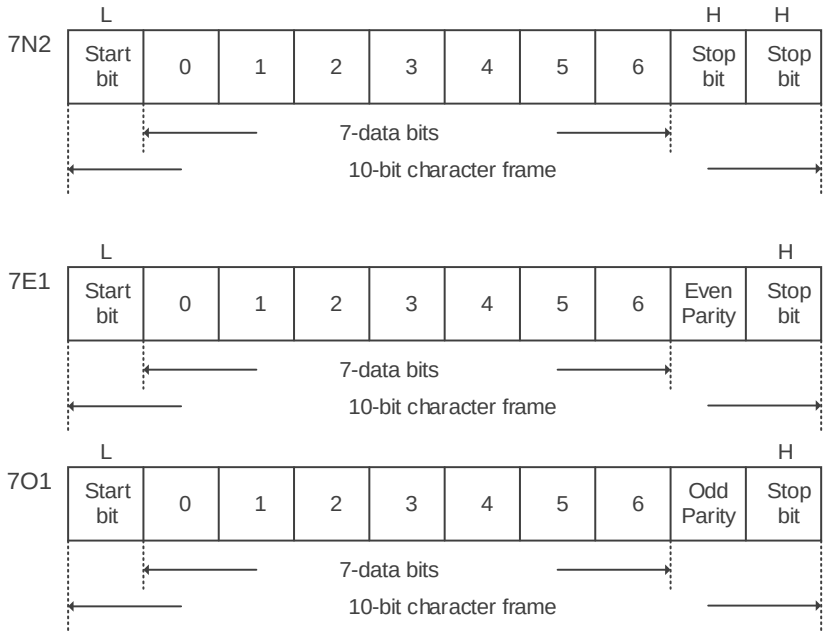
In RTU mode, each data frame consists of an 8-bit character (hexadecimal), which is more efficient than ASCII mode for data transmission because it can be done without code interchange. For instance, when transmitting “64H” between the master and slave, simply send “64H”.

Characters are encoded into the following frames and transmitted in series. The methods for checking each type of frame are as follows.

9

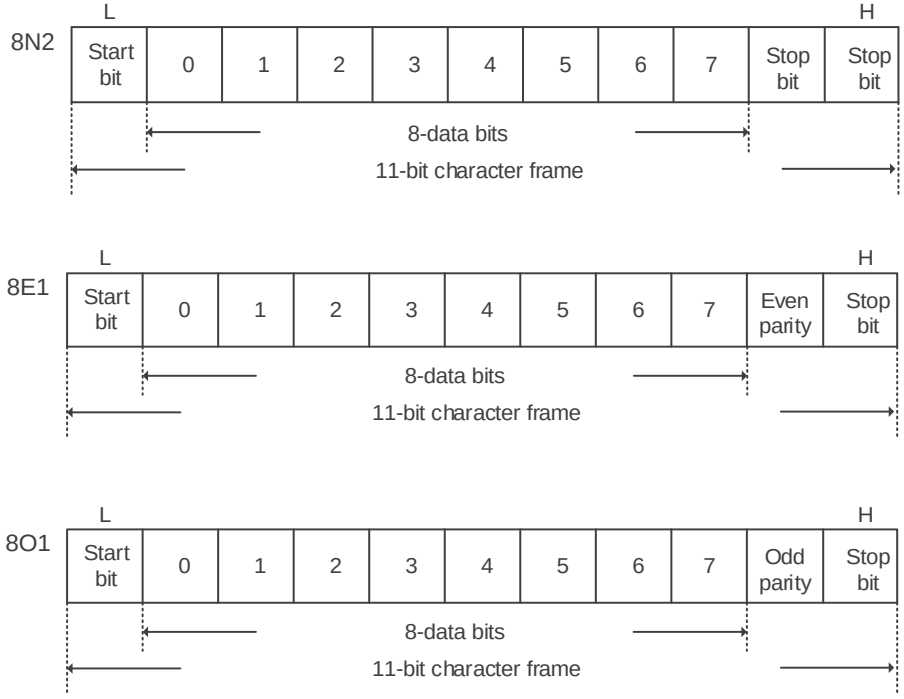
ASCII mode:

10-bit character frame (for 7-bit character)



RTU mode:

11-bit character frame (for 8-bit character)



Communication data structure

Definitions for the data frames in the two modes are as follows:

ASCII mode:

Start	Start character ':' (3AH)
Slave Address	Communication address: 1 byte, consists of 2 ASCII codes (ADR)
Function	Function code: 1 byte, consists of 2 ASCII codes (CMD)
Data (n-1)	Data content: n word(s) = 2n bytes (consists of 4n ASCII codes), n ≤ 10
…….	
Data (0)	
LRC	Error checking: 1 byte, consists of 2 ASCII codes
End 1	End code 1: (0DH) (CR)
End 0	End code 0: (0AH) (LF)

RTU mode:

Start	A silent interval of more than 10 ms
Slave Address	Communication address: 1 byte
Function	Function code: 1 byte
Data (n-1)	Data content: n word(s) = 2n bytes, n ≤ 10
…….	
Data (0)	
CRC	Error checking: 2 bytes
End 1	A silent interval of more than 10 ms

9

Example 1: function code 03H, reading multiple words

In the following example, the master issues a read command to the first slave.
The slave reads two continuous words starting from the start data address 0200H. In the response message from the slave, the content of the start data address 0200H is 00B1H and the content of the second data address 0201H is 1F40H. The maximum allowable number of data in one single access is 10 words.

ASCII mode:

Command Message (Master):

Start	':'
Slave Address	'0'
	'1'
Function	'0'
	'3'
Start Data Address	'0'
	'2'
	'0'
	'0'
Data Quantity (in words)	'0'
	'0'
	'0'
	'2'
LRC	'F'
	'8'
End 1	(0DH) (CR)
End 0	(0AH) (LF)

Response Message (Slave):

Start	':'
Slave Address	'0'
	'1'
Function	'0'
	'3'
Data Quantity (in bytes)	'0'
	'4'
Content of Start Data Address 0200H	'0'
	'0'
	'B'
	'1'
Content of the 2nd Data Address 0201H	'1'
	'F'
	'4'
	'0'
LRC	'E'
	'8'
End 1	(0DH) (CR)
End 0	(0AH) (LF)

RTU mode:

Command Message (Master):

Slave Address	01H
Function	03H
Start Data Address	02H (High)
	00H (Low)
Data Quantity (in words)	00H
	02H
CRC (Check Low)	C5H (Low)
CRC (Check High)	B3H (High)

Response Message (Slave):

Slave Address	01H
Function	03H
Data Quantity (in bytes)	04H
Content of Start Data Address 0200H	00H (High)
	B1H (Low)
Content of the 2nd Data Address 0201H	1FH (High)
	40H (Low)
CRC (Check Low)	A3H (Low)
CRC (Check High)	D4H (High)

Note: a silent interval of 10 ms is required before and after each transmission in RTU mode.

Example 2: function code 06H, writing single word

In the following example, the master issues a write command to the first slave.
The slave writes data 0064H to the start data address 0200H and sends a response message to the master after the writing is complete.

ASCII mode:

Command Message (Master):

Start	':'
Slave Address	'0'
	'1'
Function	'0'
	'6'
Start Data Address	'0'
	'2'
	'0'
	'0'
Data Content	'0'
	'0'
	'6'
	'4'
LRC	'9'
	'3'
End 1	(0DH) (CR)
End 0	(0AH) (LF)

Response Message (Slave):

Start	':'
Slave Address	'0'
	'1'
Function	'0'
	'6'
Start Data Address	'0'
	'2'
	'0'
	'0'
Data Content	'0'
	'0'
	'6'
	'4'
LRC	'9'
	'3'
End 1	(0DH) (CR)
End 0	(0AH) (LF)

RTU mode:

Command Message (Master):

Slave Address	01H
Function	06H
Start Data Address	02H (High)
	00H (Low)
Data Content	00H (High)
	64H (Low)
CRC (Check Low)	89H (Low)
CRC (Check High)	99H (High)

Response Message (Slave):

Slave Address	01H
Function	06H
Start Data Address	02H (High)
	00H (Low)
Data Content	00H (High)
	64H (Low)
CRC (Check Low)	89H (Low)
CRC (Check High)	99H (High)

Note: a silent interval of 10 ms is required before and after each transmission in RTU mode.

9

Example 3: function code 10H, writing multiple words

In the following example, the master issues a write command to the first slave.
The slave writes two words 0BB8H and 0000H starting from the start data address 0112H. In other words, 0BB8H is written into 0112H and 0000H is written into 0113H. The maximum allowable number of data in one single access is 8 words. The slave sends a response message to the master after the writing is complete.

ASCII mode:

Command Message (Master):

Start	':'
Slave Address	'0'
	'1'
Function	'1'
	'0'
Start Data Address	'0'
	'1'
	'1'
	'2'
Data Quantity (in words)	'0'
	'0'
	'0'
	'2'
Data Quantity (in bytes)	'0'
	'4'
Content of the 1st Data Frame	'0'
	'B'
	'B'
	'8'
Content of the 2nd Data Frame	'0'
	'0'
	'0'
	'0'
LRC	'1'
	'3'
End 1	(0DH) (CR)
End 0	(0AH) (LF)

Response Message (Slave):

Start	':'
Slave Address	'0'
	'1'
Function	'1'
	'0'
Start Data Address	'0'
	'1'
	'1'
	'2'
Data Quantity (in words)	'0'
	'0'
	'0'
	'2'
LRC	'D'
	'A'
End 1	(0DH) (CR)
End 0	(0AH) (LF)

RTU mode:

Command Message (Master):

Slave Address	01H
Function	10H
Start Data Address	01H (High)
	12H (Low)
Data Quantity (in words)	00H (High)
	02H (Low)
Data Quantity (in bytes)	04H
Content of the 1st Data Frame	0BH (High)
	B8H (Low)
Content of the 2nd Data Frame	00H (High)
	00H (Low)
CRC (Check Low)	FCH (Low)
CRC (Check High)	EBH (High)

Response Message (Slave):

Slave Address	01H
Function	10H
Start Data Address	01H (High)
	12H (Low)
Data Quantity (in words)	00H (High)
	02H (Low)
CRC (Check Low)	E0H (Low)
CRC (Check High)	31H (High)

Note: a silent interval of 10 ms is required before and after each transmission in RTU mode.

9

LRC and CRC transmission error checking

In ASCII mode, the error checking method is LRC (Longitudinal Redundancy Check). In RTU mode, the error checking method is CRC (Cyclic Redundancy Check). See the following details.

LRC (ASCII mode):

Start	':'
Slave Address	'7'
	'F'
Function	'0'
	'3'
Start Data Address	'0'
	'5'
	'C'
	'4'
Data Quantity (in words)	'0'
	'0'
	'0'
	'1'
LRC	'B'
	'4'
End 1	(0DH) (CR)
End 0	(0AH) (LF)

To calculate the LRC value: add all the bytes, round down the carry, and take the two's complement.

For example:

7FH + 03H + 05H + C4H + 00H + 01H = 14CH, round down the carry 1 and take 4CH.

The two's complement of 4CH is B4H.

CRC (RTU mode):

To calculate the CRC value:

Step 1: load a 16-bit register with the content of FFFFH, which is called the CRC register.

Step 2: perform (The low byte of the CRC register) XOR (The first byte of the command), and save the result in the CRC register.

Step 3: check the least significant bit (LSB) of the CRC register. If the bit is 0, shift the register one bit to the right. If the bit is 1, shift the register one bit to the right and perform (CRC register) XOR (A001H). Repeat this step 8 times.

Step 4: repeat Steps 2 and 3 until all bytes have been processed. The content of the CRC register is the CRC value.

After calculating the CRC value, fill in the low byte of the CRC value in the command message, and then the high byte. For example, if the result of CRC calculation is 3794H, put 94H in the message and then 37H as shown in the following table.

ADR	01H
CMD	03H
Start Data Address	01H (High)
	01H (Low)
Data Quantity (in words)	00H (High)
	02H (Low)
CRC (Check Low)	94H (Low)
CRC (Check High)	37H (High)

CRC program example:

This function calculates the CRC value in the C language. It needs two parameters:

```
unsigned char* data;
unsigned char length
//The function returns the CRC value in unsigned integer.
unsigned int crc_chk(unsigned char* data, unsigned char length) {
     int j;
     unsigned int reg_crc=0xFFFF;

     while( length-- ) {
          reg_crc^= *data++;
          for (j=0; j<8; j++ ) {
               if( reg_crc & 0x01 ) { /*LSB(bit 0 ) = 1 */
                    reg_crc = (reg_crc >> 1)^0xA001;
               } else {
                    reg_crc = (reg_crc>>1);
               }
          }
     }
     return reg_crc;
}
```

9

Example of a PC communication program:

```
#include<stdio.h>
#include<dos.h>
#include<conio.h>
#include<process.h>
#define PORT 0x03F8   /*  the address of COM 1  */
#define THR 0x0000
#define RDR 0x0000
#define BRDL 0x0000
#define IER 0x0001
#define BRDH 0x0001
#define LCR 0x0003
#define MCR 0x0004
#define LSR 0x0005
#define MSR 0x0006
unsigned char rdat[60];
/* read 2 data from address 0200H of ASD with address 1 */
unsigned char
tdat[60]={':','0','1','0','3','0','2','0','0','0','0','0','2','F','8','\r','\
n'};
void main() {
int I;
outportb(PORT+MCR,0x08);      /*  Interruption enable  */
outportb(PORT+IER,0x01);          /* Interruption as data in  */
outportb(PORT+LCR,( inportb(PORT+LCR) | 0x80 ) );
/*  the BRDL/BRDH can be access as LCR.b7 == 1  */
outportb(PORT+BRDL,12);
outportb(PORT+BRDH,0x00);
outportb(PORT+LCR,0x06);      /* set protocol
                                  <7,E,1> = 1AH,        <7,O,1> = 0AH
                                  <8,N,2> = 07H         <8,E,1> = 1BH
                                  <8,O,1> = 0BH                      */

for( I = 0; I<=16; I++ ) {
    while( !(inportb(PORT+LSR) & 0x20) );  /*  wait until THR empty  */
    outportb(PORT+THR,tdat[I]);            /*  send data to THR  */
}
I = 0;
while( !kbhit() ) {
    if( inportb(PORT+LSR)&0x01 ) {  /*  b0==1, data is read  */
        rdat[I++] = inportb(PORT+RDR); /*   read data from RDR  */
    }
}
}
```

9.4 Writing and reading communication parameters

Refer to Chapter 8 for the descriptions of the parameters that you can write or read through communication.

The servo drive parameters are divided into eight groups: Group 0 (Monitoring parameters), Group 1 (Basic parameters), Group 2 (Extension parameters), Group 3 (Communication parameters), Group 4 (Diagnosis parameters), Group 5 (Motion control parameters), and Group 6 and Group 7 (PR parameters). Except for the read-only parameters, all parameters can be set through communication.

Note the following additional details:

P3.001: when a new communication speed is set, the next data is written at the new transmission speed.

P3.002: when a new communication protocol is set, the next data is written with the new communication protocol.

P4.005: servo motor JOG control. Refer to Chapter 8 for detailed descriptions.

P4.006: force digital output (DO) contact control. You can use this parameter to test the DO contacts. Set P4.006 to 0x0001, 0x0002, 0x0004, 0x0008, 0x0010, and 0x0020 to test DO1, DO2, DO3, DO4, DO5, and DO6 respectively. Then, set P4.006 to 0x0000 to complete the test.

P4.010: hardware calibration options. First set P2.008 to 20 (14H in hexadecimal format) to enable this function.

P4.011 - P4.021: hardware offset calibration. The parameters were adjusted before delivery, so changing the parameter settings is not recommended. If you need to modify these parameters, first set P2.008 to 22 (16H in hexadecimal format) to enable this function.

Reading parameters through communication

You can read the values from all parameters of Group 0 to Group 7 through communication.

9.5 RS-485 communication specification

Compared with RS-232, the RS-485 communication can carry out one-to-many transmission and has better anti-interference ability. RS-485 uses a balanced transmission line for signal reception and transmission. The transmitter converts the TTL signal into a differential signal and then sends it to the receiver. The receiver receives the differential signal and then converts it back to the TTL signal. Since the transmission process uses the differential signal, it has better anti-interference ability. However, there are still restrictions on its use, so note the following when wiring.

- Number of stations
  CN3 can only support up to 32 servo drives. If your application requires more than 32 stations, install a repeater to connect more servo drives. The current maximum is 127 stations.

- Transmission distance
  The longer the transmission distance, the slower the transmission speed. The cable length can be up to 100 meters when the servo drive is installed in a quiet environment. If the required transmission speed is over 38,400 bps, a 15-meter cable is recommended to ensure data transmission accuracy.

- Transmission line
  The quality of the transmission line affects the signal transmission process. If there is interference during the transmission process, it may result in data loss. It is suggested that you use a shielded twisted-pair cable as it has metal shielded cover and a grounding wire, which ensures better anti-interference ability.

- Topology
  For topology, the closer to the master station, the more stable the transmitted signal. RS-485 supports bus topology. The transmission line must connect from the first station to the second station, and then from the second station to the third station, and so on until the last station. RS-485 does not support star and ring topologies.

- Terminal resistor
  In the communication transmission process, if the impedance is not continuous, it causes signal reflection and signal distortion. This usually happens to the device that is configured at the end of the transmission line. If the impedance is small or even 0 Ω, the signal will be reflected. To solve this problem, add a resistor of the same characteristic impedance as the cable at the end of the cable, which is called a terminal resistor. In general, the transmission line used in the RS-485 signal transmission circuit is a twisted-pair cable, and its characteristic impedance is about 120 Ω, so the impedance of the terminal resistor is also 120 Ω.

■ Anti-interference methods

In the signal transmission process, if there is interference, it may result in signal distortion. Therefore, it is important to eliminate interference. The elimination methods are as follows:

1. Add a terminal resistor.
2. Check if the servo drive is installed in a high magnetic field environment. If so, keep it as far away as possible.
3. Use a shielded twisted-pair cable for the transmission line.
4. When wiring, isolate the high voltage power cable from the signal line.
5. Use a ferrite ring at the power input. For its usage, refer to Section 2.6.
6. Add X capacitor and Y capacitor, which are IEC 60384-14 certified, at the power input.