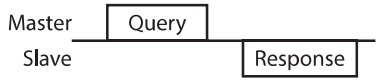
1 Modbus RTU specifications

The Modbus protocol is simple to use and its specification is open to the public, so this protocol is widely used in industrial applications.
Modbus communication is based on the single-master/multiple-slave method. Only the master can issue a query (command).
Each slave executes the process requested by query and returns a response message.
The driver supports the RTU mode only as the transmission mode. The ASC Ⅱ mode is not supported.
Under this protocol, messages are sent in one of three methods.

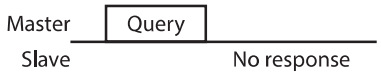
- **Unicast mode**

The master sends a query to only one slave. The slave executes the process and returns a response.



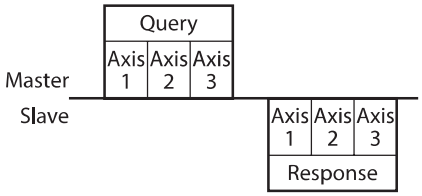
- **Broadcast mode**

If the slave address 0 is specified on the master, the master can send a command to all slaves. Each slave executes the process, but does not return a response.



- **ID share mode**

The master can send a query to multiple slaves at once by sharing a slave address (share ID) with multiple slaves. The slave executes the process and returns a response sequentially. In the ID share mode, synchronization between slaves is better than in the unicast mode since a query can be sent to multiple slaves at the same time. The ID share mode is our unique transmission method.

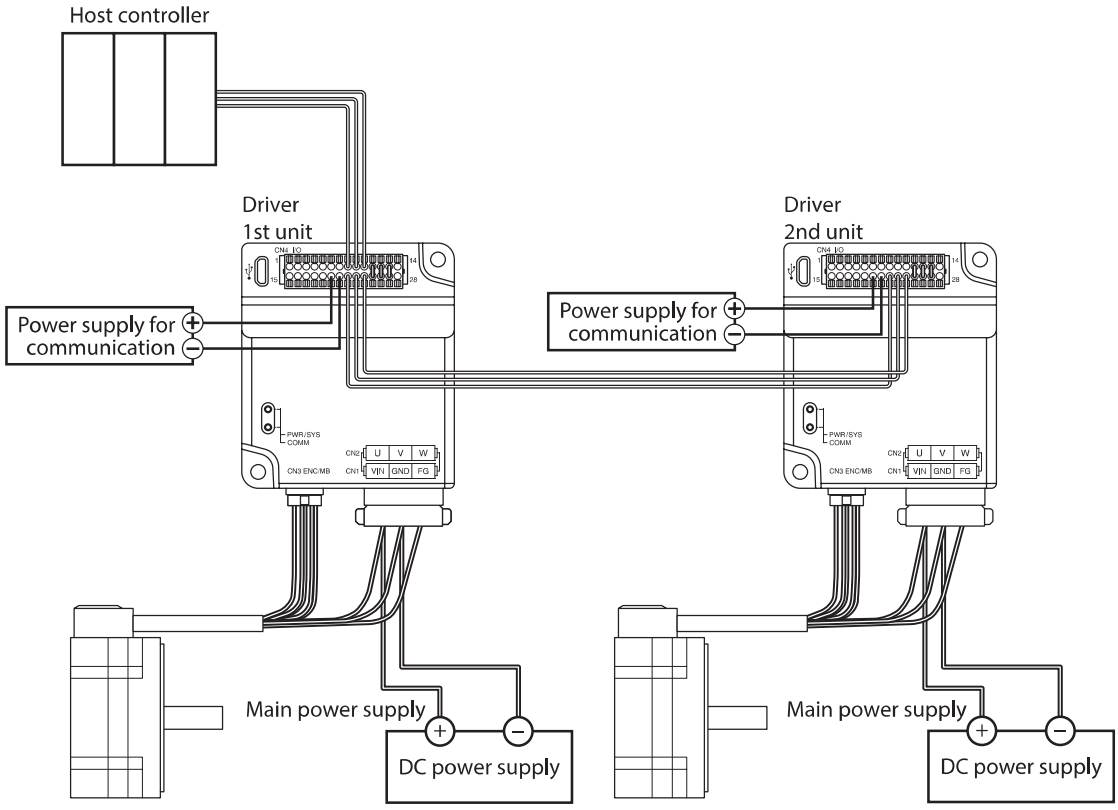


1-1 Communications specifications

Electrical characteristics	In conformance with EIA-485 Use twisted-pair wires and keep the total extension distance up to 10 m (32.8 ft.). *
Communication mode	Half duplex Asynchronous mode (data: 8 bits, stop bit: 1 bit/2 bits, parity: none/even number/odd number)
Transmission rate	Selectable from 9,600 bps, 19,200 bps, 38,400 bps, 57,600 bps, 115,200 bps, and 230,400 bps.
Protocol	Modbus RTU mode
Type of Connection	Up to 31 drivers can be connected to one host controller.

* If the motor cable or power supply cable generates an undesirable amount of noise depending on the wiring or configuration, shield the cable or install a ferrite core.

■ Connection example



memo This guidance is explained using **BLVD-KRD** as an example. For **BLVD-KBRD**, it is not necessary to connect a power supply for communication.

■ Termination resistor

Connect a termination resistor for a driver located the farthest away (positioned at the end) from the host controller. There are the following two methods for how to connect a termination resistor.

● When a termination resistor inside the driver is used

Using the support software, set the "RS-485 communication termination resistor" parameter to "Enable" or to the terminating slave address.

Name	Setting
RS-485 communication termination resistor	Enable

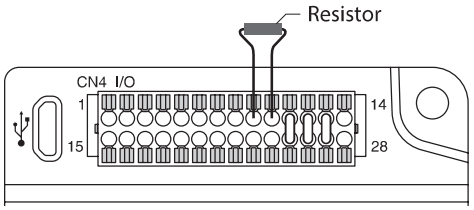
Note The termination resistor is turned ON only when the main power is supplied to the driver since it is turned ON or OFF inside the driver.

memo The termination resistor inside the driver is enabled when the slave address 4 is set (initial value). When the slave address 4 is used, check the connection of a termination resistor.

● When a resistor (120 Ω) is connected between the TR+ and TR− terminals of the CN4 connector

Connecting method

1. Connect lead wires to a resistor.
2. Connect the lead wires between the TR+ and TR− terminals of CN4.



Note
- Be sure to connect a resistor between the TR+ and TR− terminals. Incorrect connection may cause damage to the resistor.
- When connecting a resistor, set the "RS-485 communication termination resistor" parameter to "Disable."

memo For a resistor, use a metal film resistor of 120 Ω, 1/2 W or more.

Related parameter

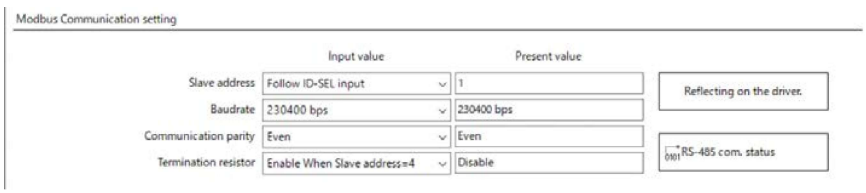
Register address		Parameter name	Description	Initial setting	
Upper	Lower			Initial value	Unit
990 (03DEh)	991 (03DFh)	RS-485 communication termination resistor	Selects the setting of the termination resistor for RS-485 communication built in the driver. **[Setting Range]** –1: Enable 0: Disable 1: Follow communication ID (Enable when the active communication ID is 1) 2: Follow communication ID (Enable when the active communication ID is 2) 3: Follow communication ID (Enable when the active communication ID is 3) 4: Follow communication ID (Enable when the active communication ID is 4) 5: Follow communication ID (Enable when the active communication ID is 5) 6: Follow communication ID (Enable when the active communication ID is 6) 7: Follow communication ID (Enable when the active communication ID is 7) 8: Follow communication ID (Enable when the active communication ID is 8) 9: Follow communication ID (Enable when the active communication ID is 9) 10: Follow communication ID (Enable when the active communication ID is 10) 11: Follow communication ID (Enable when the active communication ID is 11) 12: Follow communication ID (Enable when the active communication ID is 12) 13: Follow communication ID (Enable when the active communication ID is 13) 14: Follow communication ID (Enable when the active communication ID is 14) 15: Follow communication ID (Enable when the active communication ID is 15) 16: Follow communication ID (Enable when the active communication ID is 16) 17: Follow communication ID (Enable when the active communication ID is 17) 18: Follow communication ID (Enable when the active communication ID is 18) 19: Follow communication ID (Enable when the active communication ID is 19) 20: Follow communication ID (Enable when the active communication ID is 20) 21: Follow communication ID (Enable when the active communication ID is 21) 22: Follow communication ID (Enable when the active communication ID is 22) 23: Follow communication ID (Enable when the active communication ID is 23) 24: Follow communication ID (Enable when the active communication ID is 24) 25: Follow communication ID (Enable when the active communication ID is 25) 26: Follow communication ID (Enable when the active communication ID is 26) 27: Follow communication ID (Enable when the active communication ID is 27) 28: Follow communication ID (Enable when the active communication ID is 28) 29: Follow communication ID (Enable when the active communication ID is 29) 30: Follow communication ID (Enable when the active communication ID is 30) 31: Follow communication ID (Enable when the active communication ID is 31))	4	–

■ Address number setting (communication ID)

Set the address number (communication ID) of RS-485 communication.
There are the following two methods for how to set the address number.

- **When setting using the support software.**

  Set the address number with "Starts the simple setting." of the support software.

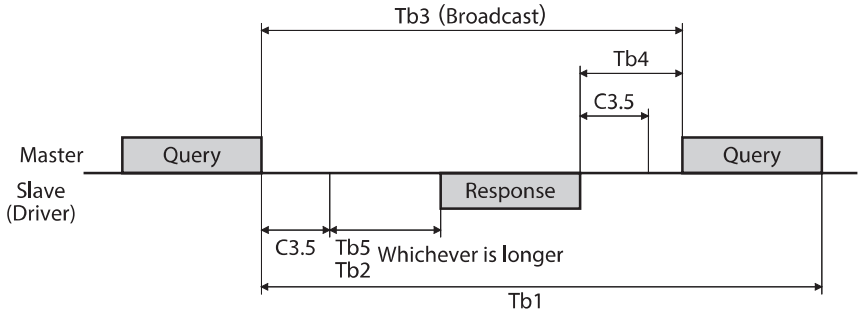


- **When setting using the ID-SEL0 to ID-SEL3 input signals.**

  Set the address number based on a combination of ON-OFF status of the ID-SEL0 to ID-SEL3 input signals.
  Refer to p.187 for ID-SEL input signals.

1-2 Communication timing

The communication time monitored by the driver and the communication timing of the master are as follows.



Code	Name	Description
Tb1	Communication timeout (driver)	The driver monitors an interval between received queries. If the driver cannot receive a query after the time set in the "Communication timeout (Modbus)" parameter (Initial value: Disable) has elapsed, an alarm of "Communication timeout" is generated. When normal messages including messages to other slaves were received, an alarm of "Communication timeout" is not generated.
Tb2	Transmission waiting time (driver)	This is the amount of time from when the driver receives a query from the master until when it starts sending a response. Set using the "Transmission waiting time (Modbus)" parameter.
Tb3	Broadcasting interval (master)	This is the amount of time until the master sends the next query in broadcasting. A time equivalent to or longer than the silent interval (C3.5) plus 5 ms is required.
Tb4	Transmission waiting time (master)	This is the amount of time from when the master receives the response until when it sends the next query (setting in the master side). Set so that it is equal to or longer than the time of the silent interval (C3.5). If the "Silent Interval (Modbus)" parameter is set to "0: Automatic," set the master side according to the "Estimate of transmission waiting time (master) (Tb4)" in the table below.
Tb5	Query processing time (driver)	This is the amount of time that the driver processes a received query. The query processing time varies depending on the message structure of the received query.
C3.5	Silent interval	This is the amount of time to determine the end of a query or response message. An interval equal to or longer than the time of the silent interval (C3.5) is required when the message ends. When the "Silent interval (Modbus)" parameter of the driver is set to "0: Automatic," the silent interval (C3.5) varies depending on the transmission rate. For details, refer to the "Silent interval (C3.5)" shown on the table below.

memo To communicate with the driver periodically, set the "Communication timeout (Modbus)" parameter. Setting the parameter can make an alarm of "RS-485 communication timeout" generate if communication between the master and the driver is disconnected.

■ When the "Silent interval (Modbus)" parameter is set to "Automatic"

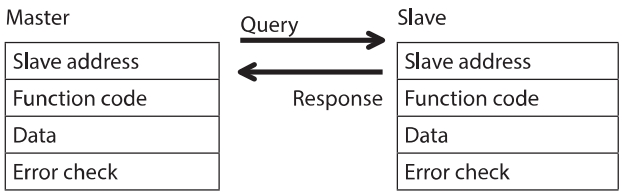
Transmission rate (bps)	Silent interval (C3.5)	Estimate of transmission waiting time (Master) (Tb4)
9,600	4.0 ms or more	5.0 ms or more
19,200 or more	2.5 ms or more	3.0 ms or more

Note

- If the transmission waiting time (Tb4) of the master is shorter than the silent interval, the slave discards the message and a communication error occurs. When a communication error occurs, check the silent interval of the slave and set the transmission waiting time (Tb4) of the master again.
- The silent interval (C3.5) may vary depending on the product series connected. When connecting multiple product series, set the driver parameters as follows.
  - "Silent interval (Modbus)" parameter: "0: Automatic"
  - "Transmission waiting time (Modbus)" parameter: 1.0 ms or more
- In a system where only products having the "Silent interval (Modbus)" parameter are connected, the communication cycle can be improved if the setting of the "Silent interval (Modbus)" parameter is common to the products connected. Use in a state of setting to "0: Automatic" normally.

2 Message structure

The message format is shown below.



2-1 Query

The query message structure is shown below.

Slave address	Function code	Data	Error check
8 bits	8 bits	Nx8 bits	16 bits

■ Slave address

Specify the slave address. (Unicast mode)
If the slave address is set to "0," the master can send a query to all slaves. (Broadcast mode)

■ Function code

The function codes and message lengths supported by the driver are as follows.

Function code	Function	Number of registers	Broadcast
03h	Reading from holding registers	1 to 125	Not possible
06h	Writing to a holding register	1	Possible
08h	Diagnosis	–	Not possible
10h	Writing to multiple holding registers	1 to 123	Possible
17h	Read/write of multiple holding registers	Read: 1 to 125 Write: 1 to 121	Not possible

■ Data

Set data related to the function code. The data length varies depending on the function code.

■ Error check

In the Modbus RTU mode, error checks are based on the CRC-16 method. The slave calculates a CRC-16 of each received message and compares the result against the error check value included in the message. If the calculated CRC-16 value matches the error check value, the slave determines that the message is normal.

- **CRC-16 calculation method**

1. Calculate an exclusive-OR (XOR) value of the default value of FFFFh and slave address (8 bits).
2. Shift the result of step 1 to the right by 1 bit. Repeat this shift until the overflow bit becomes "1."
3. Upon obtaining "1" as the overflow bit, calculate an XOR of the result of step 2 and A001h.
4. Repeat steps 2 and 3 until a shift is performed eight times.
5. Calculate an XOR of the result of step 4 and function code (8 bits).
   Repeat steps 2 to 4 for all bytes.
   The final result gives the result of CRC-16 calculation.

- **Calculation example of CRC-16**

The table shows a calculation example when setting the slave address of the first byte to 02h and the function code of the second byte to 07h.
The result of actual CRC-16 calculation is calculated including the data on and after the third byte.

Description	Result	Bit shifted out
CRC register initial value FFFFh	1111 1111 1111 1111	–
Lead byte 02h	0000 0000 0000 0010	–
Initial value FFFFh and XOR	1111 1111 1111 1101	–
First time of right shift	0111 1111 1111 1110	1
A001h and XOR	1010 0000 0000 0001 1101 1111 1111 1111	–
Second time of right shift	0110 1111 1111 1111	1
A001h and XOR	1010 0000 0000 0001 1100 1111 1111 1110	–
Third time of right shift	0110 0111 1111 1111	0
Fourth time of right shift	0011 0011 1111 1111	1
A001h and XOR	1010 0000 0000 0001 1001 0011 1111 1110	–
Fifth time of right shift	0100 1001 1111 1111	0
Sixth time of right shift	0010 0100 1111 1111	1
A001h and XOR	1010 0000 0000 0001 1000 0100 1111 1110	–
Seventh time of right shift	0100 0010 0111 1111	0
Eighth time of right shift	0010 0001 0011 1111	1
A001h and XOR	1010 0000 0000 0001 1000 0001 0011 1110	–
Next byte 07h and XOR	0000 0000 0000 0111 1000 0001 0011 1001	–
First time of right shift	0100 0000 1001 1100	1
A001h and XOR	1010 0000 0000 0001 1110 0000 1001 1101	–
Second time of right shift	0111 0000 0100 1110	1
A001h and XOR	1010 0000 0000 0001 1101 0000 0100 1111	–
Third time of right shift	0110 1000 0010 0111	1
A001h and XOR	1010 0000 0000 0001 1100 1000 0010 0110	–
Fourth time of right shift	0110 0100 0001 0011	0
Fifth time of right shift	0011 0010 0000 1001	1
A001h and XOR	1010 0000 0000 0001 1001 0010 0000 1000	–
Sixth time of right shift	0100 1001 0000 0100	0
Seventh time of right shift	0010 0100 1000 0010	0
Eighth time of right shift	0001 0010 0100 0001	0
Result of CRC-16	0001 0010 0100 0001	–

2-2 Response

Slave-returned responses are classified into three types: normal response, no response, and exception response. The response message structure is the same as the query message structure.

Slave address	Function code	Data	Error check
8 bits	8 bits	Nx8 bits	16 bits

■ Normal response

Upon receiving a query from the master, the slave executes the requested process and returns a response corresponding to the function code.

■ No response

The slave may not return a response to a query sent by the master. This condition is referred to as "No response." The causes of no response are explained below.

- **Transmission error**

The slave discards the query if any of the transmission errors in the next table is detected. No response is returned.

Cause of transmission error	Description
Framing error	Stop bit 0 was detected.
Parity error	A mismatch with the specified parity was detected.
Mismatched CRC	The calculated value of CRC-16 was found not matching the error check value.
Invalid message length	The message length exceeded 256 bytes.

- **Other than transmission error**

A response may not be returned without any transmission error being detected.

Cause	Description
Broadcast	If the query was broadcast, the slave executes the requested process but does not return a response.
Mismatched slave address	When the slave address in the query is not matched the slave address of the driver.

■ Exception response

An exception response is returned if the slave cannot execute the process requested by the query. Appended to this response is an exception code indicating why the process cannot be executed. The message structure of exception response is as follows.

Slave address	Function code	Exception code	Error check
8 bits	8 bits	8 bits	16 bits

- **Function code**

The function code in the exception response is a sum of the function code in the query and 80h.

Function code of query	Exception response
03h	83h
06h	86h
08h	88h
10h	90h
17h	97h

- **Example of exception response**

Slave address		01h	Query	Slave address		01h
Function code		10h	→	Function code		90h
Data	Register address (upper)	00h		Data	Exception code	04h
	Register address (lower)	5Eh	←	Error check (lower)		4Dh
	Number of registers (upper)	00h	Response	Error check (upper)		C3h
	Number of registers (lower)	02h				
	Number of data bytes	04h				
	Value write to register address (upper)	00h				
	Value write to register address (lower)	00h				
	Value write to register address +1 (upper)	01h				
	Value write to register address +1 (lower)	F4h				
Error check (lower)		77h				
Error check (upper)		08h				

- **Exception code**

This code indicates why the process cannot be executed.

Exception code	Communication error code	Cause	Description
01h	88h	Invalid function	The process could not be executed because the function code was invalid. • The function code is not supported. • The sub-function code for diagnosis (08h) is other than 00h.
02h	88h	Invalid data address	The process could not be executed because the data address was invalid. • The register address and the number of registers exceeded FFFFh in total.
03h	8Ch	Invalid data	The process could not be executed because the data was invalid. • The number of registers is 0. • The number of bytes is other than "the number of register ×2." • Invalid data length
04h	89h 8Ah 8Ch 8Dh	Slave error	The process could not be executed because an error occurred at the slave. • Communication with user I/F is in progress (89h). Execute the following with the support software - Data writing (under writing to the driver) - Initialization - Configuration - I/O test or remote operation • NV memory processing in progress (8Ah) - Internal processing is in progress (SYS-BSY is ON). - An alarm of "EEPROM error" is present. • Outside the parameter setting range (8Ch) The value write is outside the setting range. • Command execute disable (8Dh)

- **About slave error**

When the "Slave error response mode (Modbus)" parameter is set to "0: Normal response," even if the slave error occurs, a normal response is returned. Set it when no exception response is required, as in the case of a touch screen.

3 Function code

This chapter explains the function codes supported by the driver.
Note that function codes other than those described here cannot be executed even if they are sent.

3-1 Reading from a holding register(s) (03h)

This function code is used to read a register (16 bits). Up to 125 successive registers (125×16 bits) can be read.
Read the upper and lower data at the same time. If they are not read at the same time, the value may be invalid.
When multiple holding registers are read, they are read in order of register addresses.

■ Example of read

Read "Driver temperature" and "Motor temperature" of the slave address 1.

Description	Register address	Value read	Corresponding decimal
Driver temperature (upper)	248 (00F8h)	0000h	383
Driver temperature (lower)	249 (00F9h)	017Fh	
Motor temperature (upper)	250 (00FAh)	0000h	426
Motor temperature (lower)	251 (00FBh)	01AAh	

- **Query**

Field name		Data	Description
Slave address		01h	Slave address 1
Function code		03h	Reading from holding registers
Data	Register address (upper)	00h	Register address to start reading from
	Register address (lower)	F8h	
	Number of registers (upper)	00h	Number of registers to be read from the starting register address (4 registers=0004h)
	Number of registers (lower)	04h	
Error check (lower)		C5h	Calculation result of CRC-16
Error check (upper)		F8h	

- **Response**

Field name		Data	Description
Slave address		01h	Same as query
Function code		03h	Same as query
Data	Number of data bytes	08h	Twice the number of registers in the query
	Value read from register address (upper)	00h	Value read from register address 00F8h
	Value read from register address (lower)	00h	
	Value read from register address +1 (upper)	01h	Value read from register address 00F9h
	Value read from register address +1 (lower)	7Fh	
	Value read from register address +2 (upper)	00h	Value read from register address 00FAh
	Value read from register address +2 (lower)	00h	
	Value read from register address +3 (upper)	01h	Value read from register address 00FBh
	Value read from register address +3 (lower)	AAh	
Error check (lower)		00h	Calculation result of CRC-16
Error check (upper)		23h	

3-2 Writing to a holding register (06h)

This function code is used to write data to a specified register address. However, since the result combining the upper and lower may be outside the data range, write the upper and lower at the same time using the "Writing to multiple holding registers (10h)."

■ Example of write

Write 80 (50h) to the command filter time constant of the slave address 2.

Description	Register address	Value write	Corresponding decimal
Command filter time constant (lower)	597 (255h)	0050h	80

- **Query**

Field name		Data	Description
Slave address		02h	Slave address 2
Function code		06h	Writing to a holding register
Data	Register address (upper)	02h	Register address to be written
	Register address (lower)	55h	
	Value write (upper)	00h	Value written to the register address
	Value write (lower)	50h	
Error check (lower)		98h	Calculation result of CRC-16
Error check (upper)		6Dh	

- **Response**

Field name		Data	Description
Slave address		02h	Same as query
Function code		06h	Same as query
Data	Register address (upper)	02h	Same as query
	Register address (lower)	55h	
	Value write (upper)	00h	Same as query
	Value write (lower)	50h	
Error check (lower)		98h	Calculation result of CRC-16
Error check (upper)		6Dh	

3-3 Diagnosis (08h)

This function code is used to diagnose the communication between a master and a slave. Arbitrary data is sent and the result of returned data is used to determine whether the communication is normal. 00h (reply to query) is the only sub-function.

■ Example of diagnosis

Send arbitrary data (1234h) to the slave for diagnosis.

- **Query**

Field name		Data	Description
Slave address		03h	Slave address 3
Function code		08h	Diagnosis
Data	Sub-function code (upper)	00h	Return the query data
	Sub-function code (lower)	00h	
	Data value (upper)	12h	Arbitrary data (1234h)
	Data value (lower)	34h	
Error check (lower)		ECh	Calculation result of CRC-16
Error check (upper)		9Eh	

- **Response**

Field name		Data	Description
Slave address		03h	Same as query
Function code		08h	Same as query
Data	Sub-function code (upper)	00h	Same as query
	Sub-function code (lower)	00h	
	Data value (upper)	12h	Same as query
	Data value (lower)	34h	
Error check (lower)		ECh	Same as query
Error check (upper)		9Eh	

3-4 Writing to multiple holding registers (10h)

This function code is used to write data to multiple successive registers. Up to 123 registers can be written.
Write the data to the upper and lower at the same time. If not, an invalid value may be written.
Registers are written in order of register addresses. Note that even when an exception response is returned because some data is invalid as being outside the specified range, etc., other data may have been written properly.

■ Example of write

Set the following data to the "Operating velocity," "Acceleration rate," and "Deceleration rate" of direct data operation in the slave address 4.

Description	Register address	Value write	Corresponding decimal
Direct data operation operating velocity (upper)	94 (005Eh)	0000h	1,000
Direct data operation operating velocity (lower)	95 (005Fh)	03E8h	
Direct data operation acceleration rate (upper)	96 (0060h)	0000h	1,000
Direct data operation acceleration rate (lower)	97 (0061h)	03E8h	
Direct data operation deceleration rate (upper)	98 (0062h)	0000h	2,000
Direct data operation deceleration rate (lower)	99 (0063h)	07D0h	

- **Query**

Field name		Data	Description
Slave address		04h	Slave address 4
Function code		10h	Writing to multiple holding registers
Data	Register address (upper)	00h	Register address to start writing from
	Register address (lower)	5Eh	
	Number of registers (upper)	00h	Number of registers to be written from the starting register address (6 registers=0006h)
	Number of registers (lower)	06h	
	Number of data bytes	0Ch	Twice the number of registers in the query
	Value write to register address (upper)	00h	Value written to register address 005Eh
	Value write to register address (lower)	00h	
	Value write to register address +1 (upper)	03h	Value written to register address 005Fh
	Value write to register address +1 (lower)	E8h	
	Value write to register address +2 (upper)	00h	Value written to register address 0060h
	Value write to register address +2 (lower)	00h	
	Value write to register address +3 (upper)	03h	Value written to register address 0061h
	Value write to register address +3 (lower)	E8h	
	Value write to register address +4 (upper)	00h	Value written to register address 0062h
	Value write to register address +4 (lower)	00h	
	Value write to register address +5 (upper)	07h	Value written to register address 0063h
	Value write to register address +5 (lower)	D0h	
Error check (lower)		43h	Calculation result of CRC-16
Error check (upper)		C0h	

- **Response**

Field name		Data	Description
Slave address		04h	Same as query
Function code		10h	Same as query
Data	Register address (upper)	00h	Same as query
	Register address (lower)	5Eh	
	Number of registers (upper)	00h	Same as query
	Number of registers (lower)	06h	
Error check (lower)		21h	Calculation result of CRC-16
Error check (upper)		8Ch	

3-5 Read/write of multiple holding registers (17h)

With a single function code, reading data and writing data for multiple successive registers can be performed.
Data is written first, and then data is read from the specified registers.

■ Read

Data can be read from successive registers of up to 125.
Read the upper and lower data at the same time. If they are not read at the same time, the value may be invalid.
If multiple registers are read, they are read in order of register addresses.

■ Write

Data can be written to successive registers of up to 121.
Write the data to the upper and lower at the same time. If not, an invalid value may be written.
Registers are written in order of register addresses. Note that even when an exception response is returned because some data is invalid as being outside the specified range, etc., other data may have been written properly.

■ Example of read/write

Prepare the read address and write address in a single query.
In this example, after writing the data to the "Operating velocity," "Acceleration rate," and "Deceleration rate" of direct data operation in the slave address 1, read the present temperatures for the driver and the motor.

Description	Register address	Value write	Corresponding decimal
Direct data operation operating velocity (upper)	94 (005Eh)	0000h	1,000
Direct data operation operating velocity (lower)	95 (005Fh)	03E8h	
Direct data operation acceleration rate (upper)	96 (0060h)	0000h	1,000
Direct data operation acceleration rate (lower)	97 (0061h)	03E8h	
Direct data operation deceleration rate (upper)	98 (0062h)	0000h	2,000
Direct data operation deceleration rate (lower)	99 (0063h)	07D0h	

Description	Register address	Value read	Corresponding decimal
Driver temperature (upper)	248 (00F8h)	0000h	383
Driver temperature (lower)	249 (00F9h)	017Fh	
Motor temperature (upper)	250 (00FAh)	0000h	426
Motor temperature (lower)	251 (00FBh)	01AAh	

- **Query**

Field name		Data	Description
Slave address		01h	Slave address 1
Function code		17h	Read/write of multiple holding registers
Data	(Read) Register address (upper)	00h	Register address to start reading from
	(Read) Register address (lower)	F8h	
	(Read) Number of registers (upper)	00h	Number of registers to be read from the starting register address (4 registers=0004h)
	(Read) Number of registers (lower)	04h	
	(Write) Register address (upper)	00h	Register address to start writing from
	(Write) Register address (lower)	5Eh	
	(Write) Number of registers (upper)	00h	Number of registers to be written from the starting register address (6 registers=0006h)
	(Write) Number of registers (lower)	06h	
	(Write) Number of data bytes	0Ch	Twice the number of registers in the query
	(Write) Value write to register address (upper)	00h	Value written to register address 005Eh
	(Write) Value write to register address (lower)	00h	
	(Write) Value write to register address +1 (upper)	03h	Value written to register address 005Fh
	(Write) Value write to register address +1 (lower)	E8h	
	(Write) Value write to register address +2 (upper)	00h	Value written to register address 0060h
	(Write) Value write to register address +2 (lower)	00h	
	(Write) Value write to register address +3 (upper)	03h	Value written to register address 0061h
	(Write) Value write to register address +3 (lower)	E8h	
	(Write) Value write to register address +4 (upper)	00h	Value written to register address 0062h
	(Write) Value write to register address +4 (lower)	00h	
	(Write) Value write to register address +5 (upper)	07h	Value written to register address 0063h
	(Write) Value write to register address +5 (lower)	D0h	
Error check (lower)		C6h	Calculation result of CRC-16
Error check (upper)		00h	

- **Response**

Field name		Data	Description
Slave address		01h	Same as query
Function code		17h	Same as query
Data	(Read) Number of data bytes	08h	Twice the number of (Read) registers in the query
	(Read) Value read from register address (upper)	00h	Value read from register address 00F8h
	(Read) Value read from register address (lower)	00h	
	(Read) Value read from register address +1 (upper)	01h	Value read from register address 00F9h
	(Read) Value read from register address +1 (lower)	7Fh	
	(Read) Value read from register address +2 (upper)	00h	Value read from register address 00FAh
	(Read) Value read from register address +2 (lower)	00h	
	(Read) Value read from register address +3 (upper)	01h	Value read from register address 00FBh
	(Read) Value read from register address +3 (lower)	AAh	
Error check (lower)		40h	Calculation result of CRC-16
Error check (upper)		63h	