

---

# CHAPTER 3

## Modbus RTU PROTOCOL

This chapter describes the Modbus RTU protocol, as well as the host side procedure for using this protocol and error processing.

The Modbus RTU protocol was a set of specifications developed in the United States. For the FRENIC-Mini of which inverter ROM version is 0399 or earlier, the Modbus RTU functions are partially restricted. Contact us about details of restrictions. Check the inverter ROM version with menu "5\_14" described in "3.8 Reading Maintenance Information" under Chapter 3 of the FRENIC-Mini Instruction Manual.

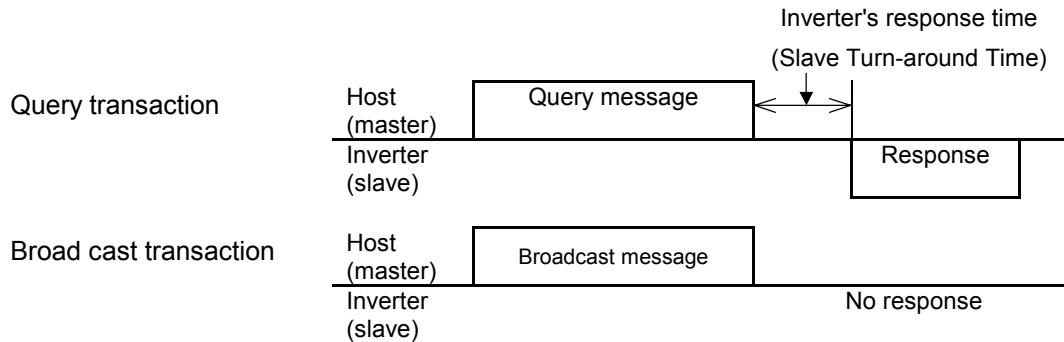
### Table of Contents

3.1	Messages .....	3-1
3.1.1	Message formats.....	3-1
3.1.2	Message types .....	3-1
3.1.3	Message frames.....	3-2
3.1.4	Message categories .....	3-3
3.1.5	Communications examples .....	3-7
3.2	Host Side Procedures.....	3-8
3.2.1	Inverter's response time.....	3-8
3.2.2	Timeout processing .....	3-8
3.2.3	Receiving preparation complete time and message timing from the host.....	3-9
3.2.4	Frame synchronization method.....	3-9
3.3	Communications Errors .....	3-11
3.3.1	Categories of communications errors .....	3-11
3.3.2	Operations in case of errors.....	3-12
3.4	CRC-16 .....	3-15
3.4.1	Overview of the CRC-16 .....	3-15
3.4.2	Algorithm .....	3-15
3.4.3	Calculation example.....	3-17
3.4.4	Frame length calculation .....	3-18

## 3.1 Messages

### 3.1.1 Message formats

The regular formats for transmitting RTU messages are shown below:



If the inverter receives from the host a message in the standby status and considers it properly received, it executes a transaction in response to the request and sends back normal response. If the inverter judges that the message has not been received properly, it returns error response. The inverter does not send back any response in the case of broadcast transactions.

### 3.1.2 Message types

Message types are classified into four types; query, normal response, error response, and broadcast.

#### Query

The host sends messages to a single inverter.

#### Normal response

After the inverter received a query from the host, the inverter executes a transaction in response to the request, and sends back corresponding normal response.

#### Error response

If the inverter receives a query but cannot execute the requested function because an invalid function code is specified or for other reasons, it sends back error response.

The error response is accompanied by a message describing the reason the request cannot be executed.

The inverter cannot send back any response in case of a CRC or physical transmission error (parity error, framing error, overrun error).

#### Broadcast

The master uses address 0 to send messages to all slaves. All slaves, which receive a broadcast message, execute the requested function. This transaction will be terminated upon timeout of the master.

### 3.1.3 Message frames

As shown below, a transmission frame consists of four blocks, which are called fields. Details depend on RTU function codes. To make a clear distinction between RTU function codes and the inverter's function codes, the former will be hereinafter referred to as FCs.

1 byte	1 byte	Up to 105 bytes	2 bytes
Station address	FC (RTU function code)	Information	Error check

#### **Station address**

The station address field is one byte long, in which a station address between 0 and 247 can be selected.

Selecting address 0 means the selection of all slave stations and a broadcast message.

#### **FC (RTU function code)**

The FC field is one byte long, in which a function code is defined with a number from 0 to 255. The FCs in hatching are available. Do not use any unavailable FC. Failure to observe this rule results in error response.

Table 3-1 List of FCs

FC	Description
0 to 2	Unused
3	Function reading (up to 50)
4 to 5	Unused
6	Single function writing
7	Unused
8	Maintenance code
9 to 15	Unused
16	Serial function writing (up to 50 data)
17 to 127	Unused
128 to 255	Reserved for exception response

#### **Information**

The information field contains all information (function code, byte count, number of data, data, etc.). For further information about the information field for each message type (broadcast, query, normal response, error response), see "3.1.4 Message categories."

#### **Error check**

The error check field is a CRC-16 check system and two bytes long. Since the length of the information field is variable, the frame length required for calculating the CRC-16 code is calculated based on the FC and the byte count data.

For further information about CRC-16 calculations and algorithm, see "3.4 CRC-16."

For byte counts, see "3.1.4 Message categories."

#### **Character format**

Each byte of a message is transmitted as a character. Character formats are described on the following page.

A character comprises a start bit (logical value 0), 8-bit data, an additional (optional) parity bit, and a stop bit (logical value 1).

A character always consists of eleven bits, and the number of stop bits varies depending on whether parity exists.

#### Without parity

LSB										MSB	
0	1	2	3	4	5	6	7	8	9	10	
Start	Data								Stop		

#### With parity

LSB									MSB		
0	1	2	3	4	5	6	7	8	9	10	
Start	Data								Parity (optional)		Stop

### 3.1.4 Message categories

There are five RTU message categories; function reading, single function writing, serial function writing, maintenance code, and error response.

Each category is described below:

#### [1] Reading function codes

##### Query

1 byte	1 byte	2 bytes	2 bytes	2 bytes
Station address	03 <sub>H</sub>	Function code	Number of data read	Error check
		Hi Lo	Hi Lo	

##### Normal response

1 byte	1 byte	1 byte	2 to 100 bytes	2 bytes
Station address	03 <sub>H</sub>	Byte count	Number of data read	Error check
			Hi, Lo (data 0); Hi, Lo (data 1); .....	

##### How to set a query

- This request is not available for broadcast transactions. Station address 0 will become invalid (no response).
- FC = 3 (03<sub>H</sub>)
- The function code is two bytes long. The Hi byte indicates the function code group (see Table 3.2), and the Lo byte represents a function code identification number (0 to 99).

(Example) When the function code is E15, the Hi byte is 01<sub>H</sub> and the Lo byte is 0F<sub>H</sub>.

Table 3.2 Function code group/code conversion table

Group	Code		Name	Group	Code		Name
F	0	00 <sub>H</sub>	Fundamental function	M	8	08 <sub>H</sub>	Monitor data
E	1	01 <sub>H</sub>	Extension terminal function	J	13	0D <sub>H</sub>	Application function
C	2	02 <sub>H</sub>	Control function of frequency	y	14	0E <sub>H</sub>	Link function
P	3	03 <sub>H</sub>	Motor parameter	W	15	0F <sub>H</sub>	Monitor 2
H	4	04 <sub>H</sub>	High performance function	X	16	10 <sub>H</sub>	Alarm 1
S	7	07 <sub>H</sub>	Command/Function data	Z	17	11 <sub>H</sub>	Alarm 2

- The length of the read data is up to 50 words (2 byte each).
- If the read data contains an unused function code, 0 will be read, which will not result in an error.
- Data does not extend over two or more function code groups. If, for example, reading of 40 words is specified from F40 but only function codes up to F40 are available, the data of F40 will be set at the first word, and the other 49 words will be 0.

#### **Interpretation of normal response**

- The data range of byte counts is between 2 and 100. A byte count is double the number of data read (1 - 50 data) of the response.
- The read data contains each word data in order of Hi byte and Lo byte, and each word data is sent back in order of the data of the function code (address) requested by the query, the data of that address number plus 1, the data of that number address number plus 2 ... If two or more function data are read and the second or any of the following data contains an unused function code (F09, etc.), the read data will become 0.

### **[2] Single function writing**

#### **Query**

1 byte	1 byte	2 bytes	2 bytes	2 bytes
Station address	06 <sub>H</sub>	Function code	Data written	Error check
		Hi    Lo	Hi    Lo	

#### **Normal response**

1 byte	1 byte	2 bytes	2 bytes	2 bytes
Station address	06 <sub>H</sub>	Function code	Data written	Error check

#### **How to set a query**

- When address 0 is selected, broadcast is available. In this case, all inverters do not respond even if a broadcast request is executed.
- FC = 6 (06<sub>H</sub>)
- The function code is two bytes long. The Hi byte indicates the function code group (see Table 3.2), and the Lo byte represents a function code identification number (0 to 99).
- The written data field is fixed two bytes long. Set the data on the function code to be written.

#### **Interpretation of normal response**

The frame is the same as the query.

### **[3] Serial function writing**

#### **Query**

1 byte	1 byte	2 bytes	2 bytes	1 byte	2 to 100 bytes	2 bytes
Station address	10 <sub>H</sub>	Function code	Number of data written	Byte count	Data written	Error check
		Hi    Lo	Hi    Lo		Hi, Lo; Hi, Lo...	

#### **Normal response**


1 byte	1 byte	2 bytes	2 bytes	2 bytes
Station address	10 <sub>H</sub>	Function code	Number of data written	Error check

**How to set a query**

- When the station address 0 is selected, broadcast is available. In this case, all inverters do not respond even if a broadcast request is executed.
- FC = 16 (10<sub>H</sub>)
- The function code is two bytes long. The Hi byte indicates the function code group (see Table 3.2), and the Lo byte represents a function code identification number (0 to 99).
- The number of data written is two bytes long, and the setting range is from 1 to 50. If 51 or a higher value is set, error response will result.
- The byte count field is one byte long, and the setting range is from 2 to 100. Set a value equivalent to the double of the number of data written.
- Set the lowest order code (the data on the function code requested by the query) at the first two bytes of the data written, and the higher order data (address plus 1, address plus 2 ...) at the following bytes.
- If the data written contains an unused function code, the writing will be ignored, which will not result in an error.

**Interpretation of normal response**

- With regard to the function code and the number of data written, the same values as those of the query will be sent back.

 **CAUTION** - When H03 (Data initialization) is used, use single function writing (FC = 6). H03 will be ignored even if it is written with serial function writing (FC = 16).

**[4] Maintenance code****Query**

1 byte	1 byte	2 bytes	2 bytes	2 bytes
Station address	08 <sub>H</sub>	Diagnosis code 0000 <sub>H</sub>	Data written Hi      Lo	Error check

**Normal response**

1 byte	1 byte	2 bytes	2 bytes	2 bytes
Station address	08 <sub>H</sub>	Diagnosis code 0000 <sub>H</sub>	Data written	Error check

**How to set a query**

- This request cannot use broadcast. Station address 0 will become invalid (no response).
- FC = 8 (08<sub>H</sub>)
- Set the diagnosis code field to be 2 bytes long fixed 0000<sub>H</sub>. Error response will result if data other than 0000<sub>H</sub> is set.
- The data written field is two bytes long, and any contents of data can be set.

**Interpretation of normal response**

- The frame is the same as the query.

## [5] Error response

If the inverter receives an improper query, it will not execute it, which will result in error response.

### Error response

1 byte	1 byte	1 byte	2 bytes
Station address	Exception function	Subcode	Error check

### Interpretation of error response

- The station address is the same as that of the query.
- The exception function is a value obtained by adding 80H to the FC of the query message (or the value of the FC if the FC is larger than 80H).  
For example, when the FC is 3, the exception function is  $3 + 128 = 131$  (83<sub>H</sub>).
- The subcode represents the code of the reason for the improper query.

Table 3.3 Subcodes

Subcode	Item		Description
1	Improper FC		An FC other than 3, 6, 8 or 16 was received.
2	Improper address	Improper function code	An unused function code or a function code out of range was received. When the data read/written (except the first one) containing an unused function code. <ul style="list-style-type: none"><li>- In function reading Zero (0) will be read, which will not result in an error.</li><li>- In serial function writing The writing will be ignored, which will not result in an error.</li></ul>
		Improper number of data	<ul style="list-style-type: none"><li>- When the number of data read/written is not between 1 and 50.</li><li>- No error will result when the value of the function code plus the number of data is beyond the setting range of the function code.</li></ul>
		Diagnosis code error (maintenance code)	A value other than 0 was received although the diagnosis code as the maintenance code was fixed to 0.
3	Improper data	Data range error	The data written is beyond the permissible write range.
7	NAK	No right of writing	This error does not occur in the FRENIC-Mini.
		Write disable	<ul style="list-style-type: none"><li>- Writing was attempted to the functions to which writing from RTU is prohibited or to which writing is disabled during operation.</li><li>- Writing was attempted to a function code (other than S01, S05, S06, S13, and S14) that could not be written when the voltage was insufficient.</li></ul>

- If response is sent back to an improper query, a subcode will be set in an error code (that can be referred to with M26).

### 3.1.5 Communications examples

Typical communications examples are shown below (the station address is 5 in all cases).

**(Example 1) M06: The actual frequency and speed values will be read.**

**Query (host ⇒ inverter)**

05	03	08	06	00	01	67	EF
----	----	----	----	----	----	----	----

**Normal response (inverter ⇒ host)**

05	03	01	27	10	A3	B8
----	----	----	----	----	----	----

The detected speed value is 2710<sub>H</sub>, or 10000<sub>d</sub>. The actual frequency is 30 Hz according to the expression shown below:

$$10000 \times \frac{\text{Maximum output frequency}}{20000} = 30 \text{ (Hz)} \quad (\text{Maximum output frequency: 60 Hz})$$

**(Example 2) S01: The value of 15Hz will be written to speed setting 1 (maximum output frequency: 60 Hz).**

According to the expression shown below, the value to be written is 1388<sub>H</sub>.

$$15\text{Hz} \times \frac{20000}{60 \text{ (Hz)}} = 5000_{\text{d}} = 1388_{\text{H}}$$

**Query (host ⇒ inverter)**

05	06	07	01	13	88	D5	AC
----	----	----	----	----	----	----	----

**Normal response (inverter ⇒ host)**

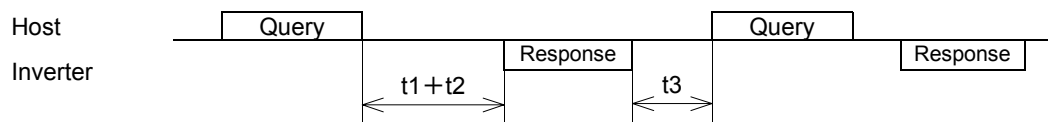
05	06	07	01	13	88	D5	AC
----	----	----	----	----	----	----	----



## 3.2 Host Side Procedures

### 3.2.1 Inverter's response time

Upon receipt of a query from the host, the inverter executes the queried transaction and sends back response after the response time shown below:



$t1 + t2$ : Inverter's response time

$t1$ : Response interval time (function code: y09)

The time until the inverter starts to send response to the query from the host controller, such as a personal computer, can be set. Setting the response interval time enables even a host controller with a slow transaction execution speed to adjust timing.

$t2$ : Inverter's transaction time

This is the time until the inverter executes the query and sends back response as shown in Table 3.4.

$t3$ : See "3.2.3 Receiving preparation complete time and message timing from the host."

Table 3.4 Inverter's transaction time

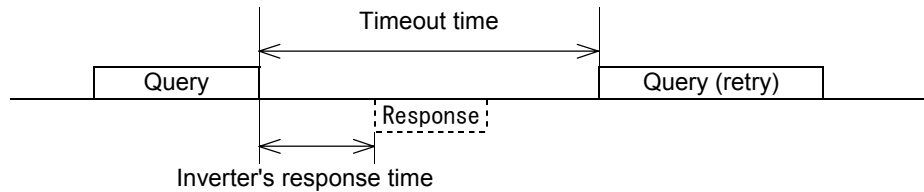
FC	Transaction	Description	t2	Timeout time (recommended)
3	Function code data reading	The number of data is three or less.	≤10ms	0.1s
		The number of data is four or more.	≤30ms	0.1s
6	Single function code data writing	S01, S05, S06, S13, S14: Command	≤10ms	0.1s
		Other than S01, S05, S06, S13, S14, and H03	≤100ms	0.5s
		H03 = 2: Motor parameter initialization	≤500ms	1.0s
		H03 = 1: Data initialization	≤5s	10.0s
8	Maintenance code		≤10ms	0.1s
16	Serial function code data writing	The function code data writing time is two seconds in the case of wiring 50 serial words.	≤2s	10.0s

### 3.2.2 Timeout processing

To read/write data from/to the host, transmit the next frame after confirming response. If response is not transmitted from the inverter for more than a specified period of time (timeout time), it is a timeout, and perform a retry. (If a retry begins before a timeout time elapses, the requested frame cannot be received properly.)

The timeout time must be set longer than the response time of the inverter. Table 3.4 above shows recommended timeout times when no response interval time is set.

In case of a timeout, retransmit the same frame or read details of the error (M26) to confirm whether the inverter sends back normal response. If normal response is returned, this indicates that some transient transmission error occurred due to noise or for other reasons, and subsequent communications is normal. (However, if this phenomenon frequently occurs even when normal response is sent back, some problem may exist. Perform a close investigation.) In case of no response, perform another retry. If the number of retries exceeds the set value (generally about three times), there may be a problem with the hardware and the software for the high-order appliance. Investigate and correct the cause.



### 3.2.3 Receiving preparation complete time and message timing from the host

The time from the return of response by the inverter until the completion of receiving preparation of the communications port (switching from transmission to receiving) is called a receiving preparation complete time.

Transmit the following messages after the receiving preparation complete time:

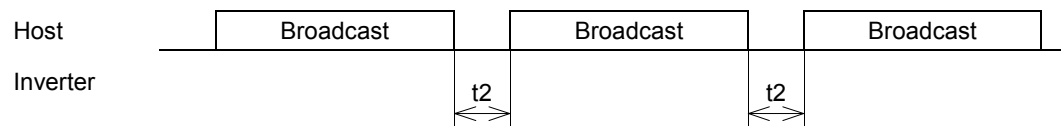
Receiving preparation complete time: 5 ms or less

Receiving waiting time from the host ( $t_3$ ):  $t_3 > 5$  ms

#### In the case of broadcast

Upon receipt of a query message from the host by broadcast, the inverter executes the query and enters the receiving enabled status.

Transmit the next message from the host after broadcast after the transaction time ( $t_2$ ) of the inverter.

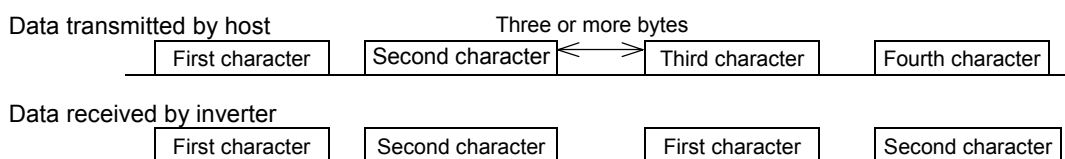


### 3.2.4 Frame synchronization method

Since the RTU transmits and receives binary data without using header characters for frame synchronization, a frame synchronization system is defined as a time without data to identify the head of the frame.

If data communications does not occur for a period equal to three bytes (33 bits including the start and stop bits) at the current transmission speed during receiving standby, initialize the frame information, and consider the first received data the first byte of the frame. Any frame being received before the time without data for three bytes will be discarded.

For this reason, the host must transmit data at a time interval of three or less bytes between two characters.



---

With regard to data to another station, messages from the host and response from that station will be received. In response transmission to identify the head of the frame, a waiting time of three bytes (33 bits including the start and stop bits) is required between the completion of data receipt by the station and the start of transmission. Any devices multi-dropped also requires such a waiting time.

## 3.3 Communications Errors

### 3.3.1 Categories of communications errors

The communications-related errors the inverter detects are listed below:

Table 3.5 Communications errors detected by inverter

Error category	Error name	Description	Error code (M26)
Logical error	Improper FC	See "Table 3.3 Subcodes" shown in 3.1.4 [5].	1(01 <sub>H</sub> )
	Improper address		2(02 <sub>H</sub> )
	Improper data		3(03 <sub>H</sub> )
	NAK		7(07 <sub>H</sub> )
Transmission error	CRC error	The frame to the local station is found unmatched in CRC collation.	71(47 <sub>H</sub> )
	Parity error	The parity is unmatched.	72(48 <sub>H</sub> )
	Other errors	Receiving errors other than the abovementioned (framing error, overrun error)	73(49 <sub>H</sub> )
Communications disconnection error	Communications disconnection error	The inverter did not receive a normal frame addressed to local or to other stations within the communications disconnection time set with the function code.	—

#### **Logical error (error codes 1 to 7)**

When a logical error is detected, an error response frame reports it. For further information, see "3.1.4 [5] Error response."

#### **Transmission error (error codes 71 to 73)**

When a transmission error occurs eight straight times, it is handled as a communications error. However, the inverter does not return response in order to avoid overlapping of response from multiple inverters. The count of eight straight times will be cleared upon normal receipt of a frame to another station or to the local inverter (station) itself.

#### **Communications disconnection error**

If the inverter in operation does not receive a normal frame to itself or to other stations when it has received a normal frame more than once and is operating via communications (frequency command or operation command), this status is considered disconnected.

When a disconnection status is set and remains over the setting of function code y08 (communications disconnection detection time), it is treated as a communications error.

1) Communications disconnection detection time (y08): 0 (without detection), 1 to 60 (seconds)

2) Condition to clear communications disconnection detection timer:

It will be cleared in a status other than disconnection.

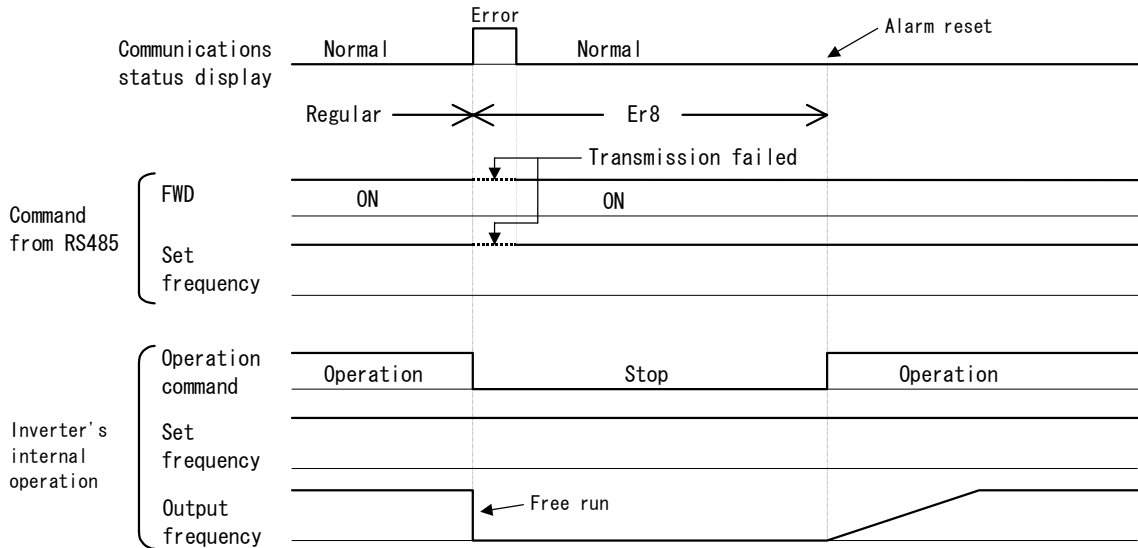
When it is necessary to take action against errors by factor, the factor can be identified by reading M26. (M26 stores the latest communications error codes.)

### 3.3.2 Operations in case of errors

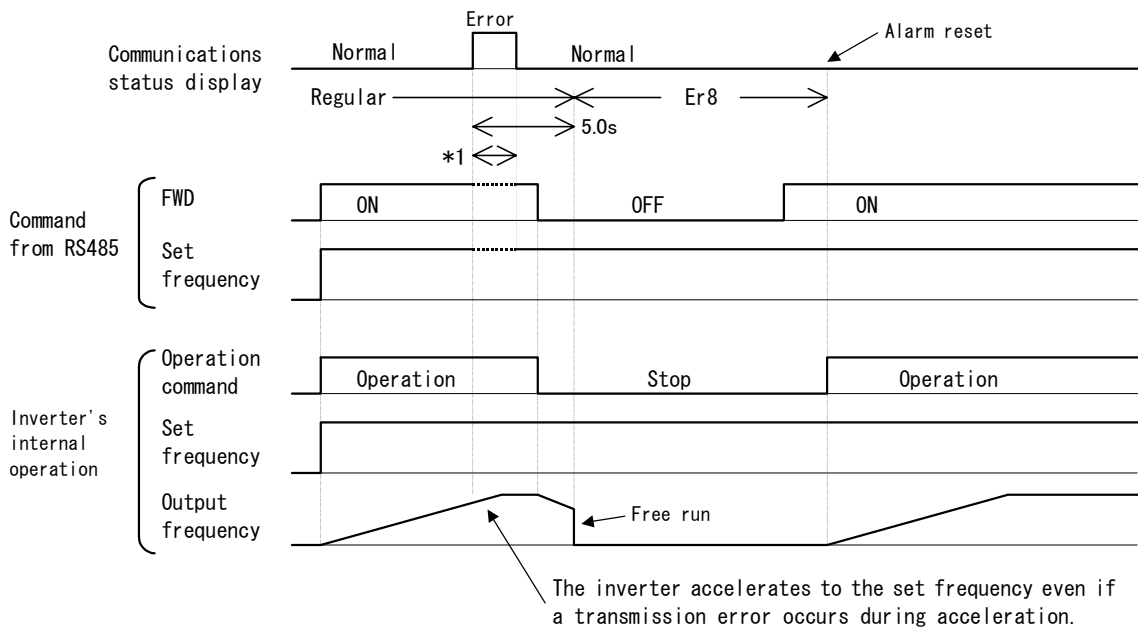
The action when a transmission or communications disconnection error occurs can be selected with function code y02. (For further information, see "2.4 Making RS485-related settings.")

This section shows specific examples of action by different settings of function code y02.

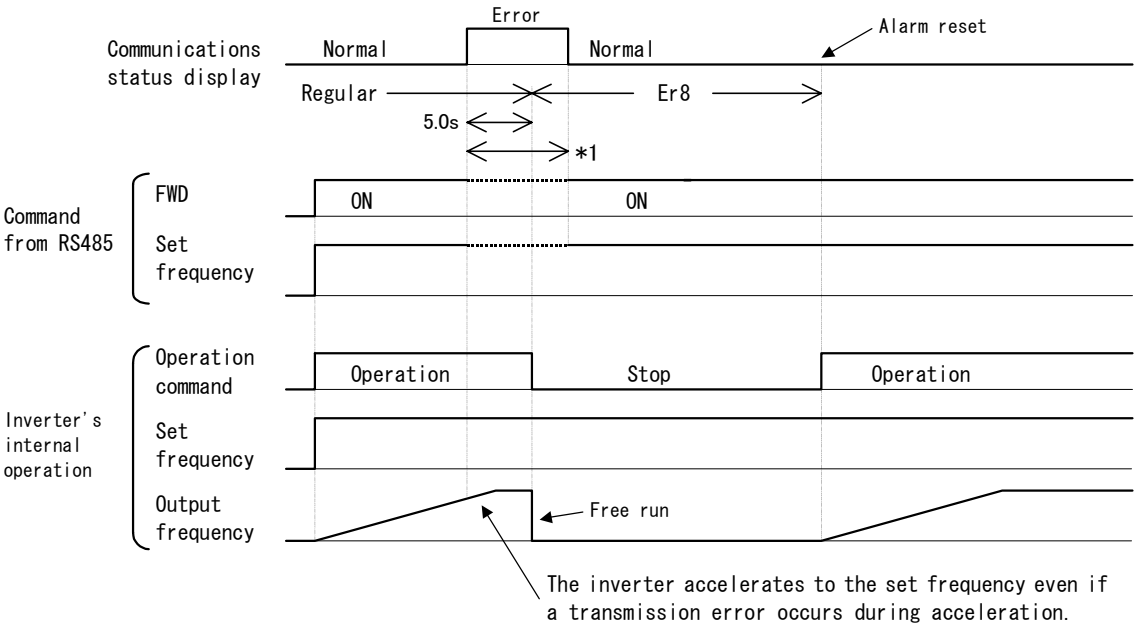
**When y02 = 0** (mode in which the inverter is forced to immediately stop in case of communications error)



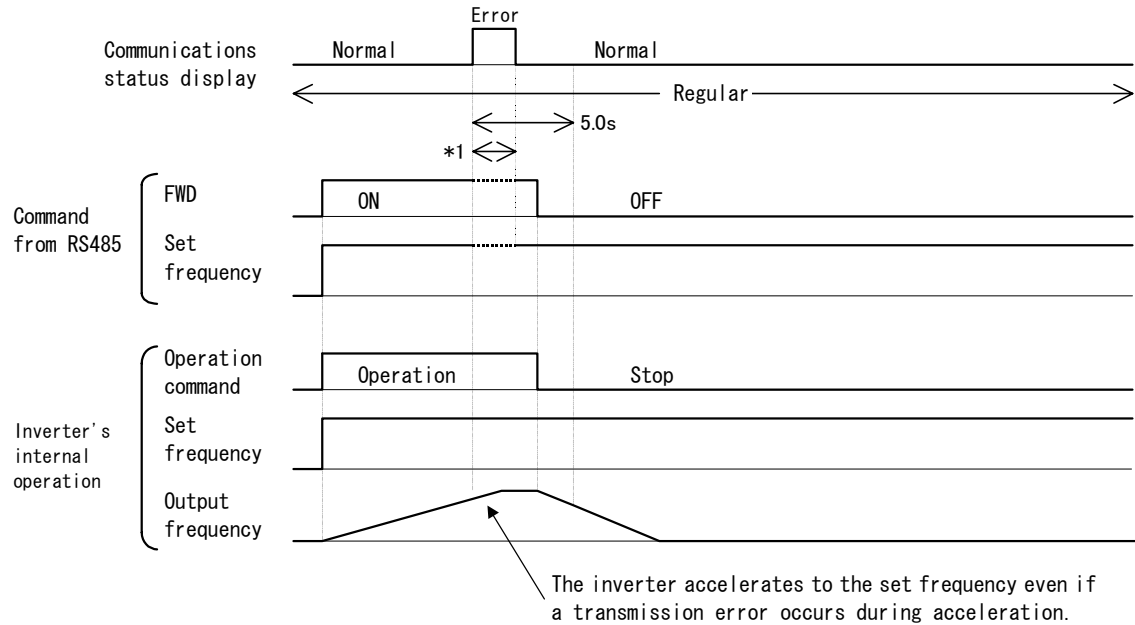
**When y02 = 1 and y03 = 5.0** (seconds) (mode in which the inverter is forced to stop five seconds after a communications error occurred)



When y02 = 2 and y03 = 5.0 (seconds)  
(when communications is not recovered although five seconds elapsed from the occurrence of a communications error, and an *Er8* trip occurs)

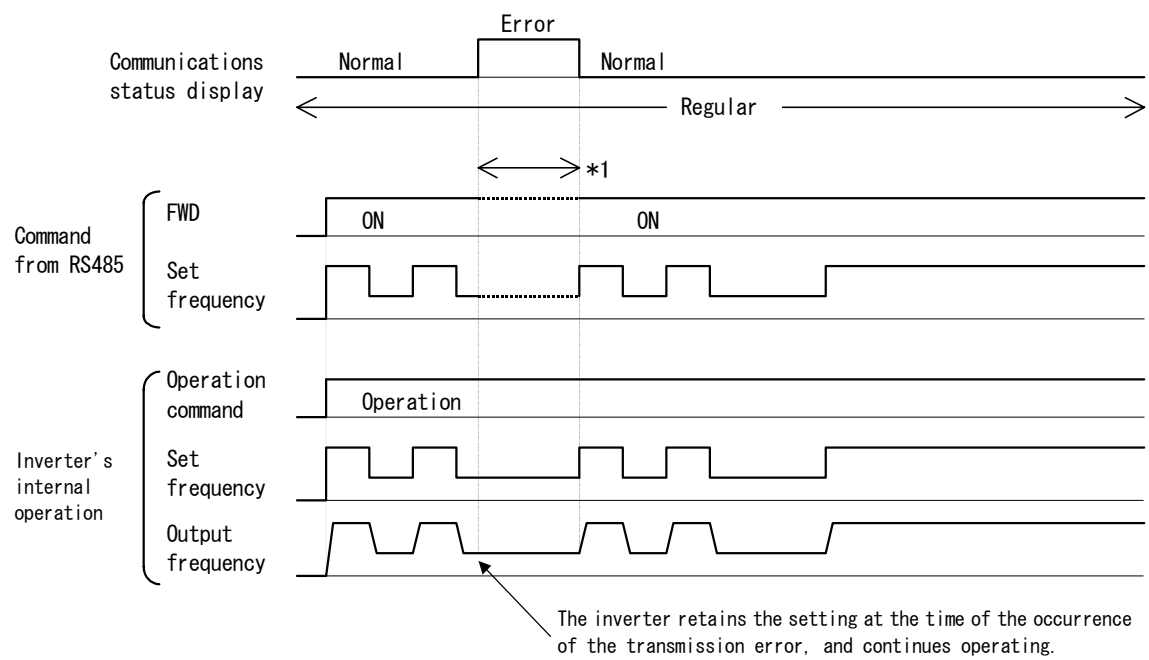


When y02 = 2 and y03 = 5.0 (seconds)  
(when a communications error occurred but communications was recovered within five seconds)



### When y02 = 3

(mode in which the inverter continues operating when a communications error occurs)



## 3.4 CRC-16

### 3.4.1 Overview of the CRC-16

The CRC (cyclic redundancy check) is a system to confirm whether there is any error in the communications frame during data transmission.

The CRC is among the most effective error check systems. The transmission station calculates and adds CRC data to the last block of the frame, and the receiving station also calculates CRC data against the data received, and compares them with each other.

#### Steps to calculate CRC data

- Polynomial data (for example, 1100 0000 0010 0001  $\rightarrow X^{15} + X^{14} + X^5 + 1$ ) is divided by a generative polynomial expression (17 bits:  $X^{16} + X^{15} + X^2 + 1$ ). CRC data is the remainder (16 bits) of this division.
- The quotient is ignored, and a message is transmitted with the remainder added to the last block of the data.
- The receiving station divides this message (with the CRC added) by the generative polynomial expression, and considers the transmitted message to have been received without any error if the "remainder" is 0.

#### CRC-16

The generative polynomial expression is expressed as a multiplier of  $X$ , such as  $X^3 + X^2 + 1$ , in place of the description of binary code 1101. Although any prime polynomial expression is acceptable as the generative polynomial expression, some standard generative polynomial expressions for optimizing error detection are defined and proposed. The RTU protocol uses the generative polynomial expression ( $X^{16} + X^{15} + X^2 + 1$ ) corresponding to binary code 1 (1000 0000 0000 0101). In this case, the CRC generated is well known as CRC-16.

### 3.4.2 Algorithm

Figure 3.1 on the following page shows the algorithm for calculating CRC-16. Consult it together with the calculation example that follows.

In this figure, the transmission station calculates CRC data and finally adds it to the transmission frame as a check code.

The receiving station uses the same algorithm to perform a transaction. However, it collates the CRC data it calculated with the transmitted CRC data.



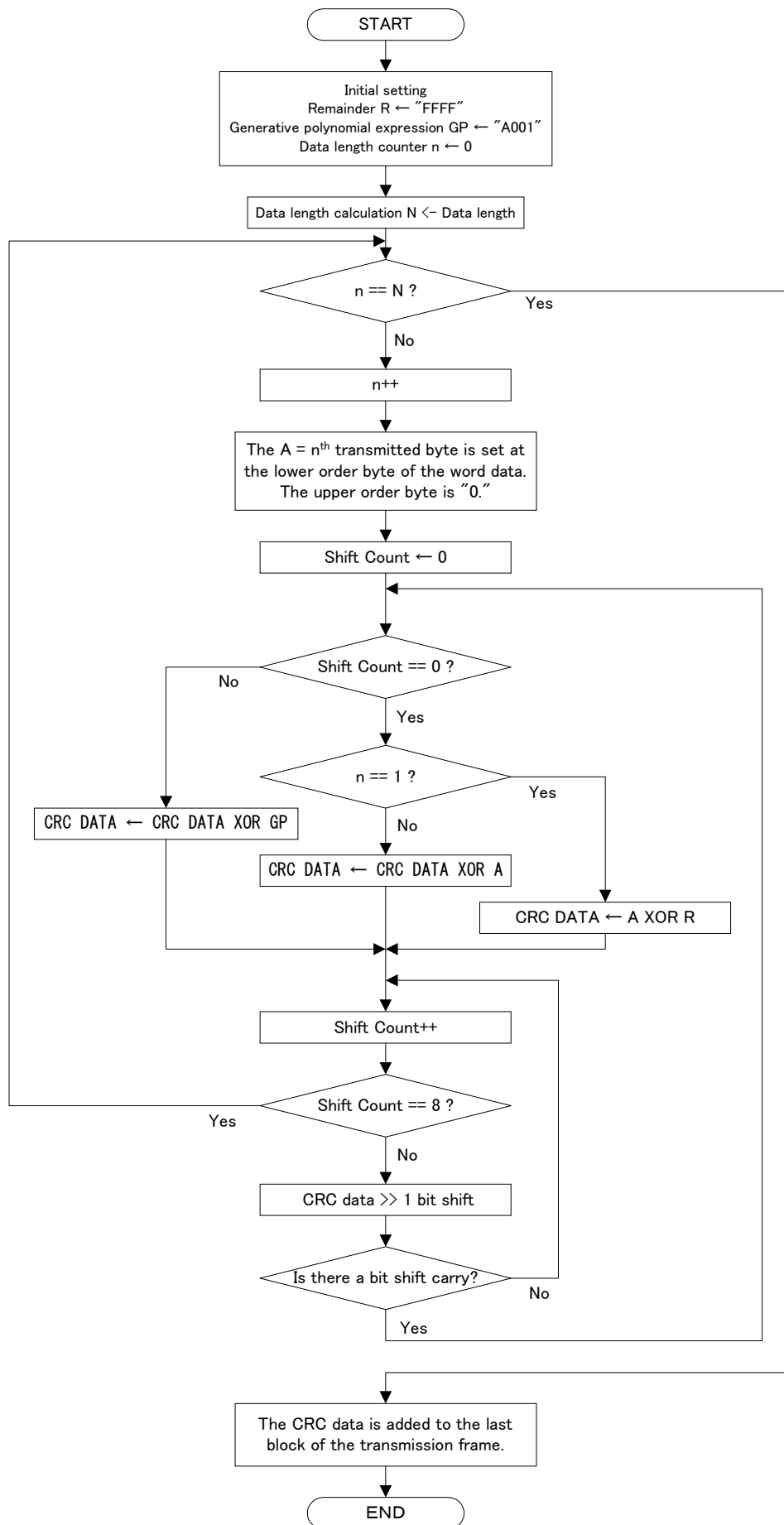


Figure 3.11 CRC algorithm

### 3.4.3 Calculation example

#### Example of transmitting data read

Station address = 1, FC = 3, function code = P02 (P = 03<sub>H</sub>, 02 = 02<sub>H</sub>), number of data read = 20, GP = generative polynomial expression(1010 0000 0000 0001)

Station address	FC	Function code		Number of data read	
01 <sub>H</sub>	03 <sub>H</sub>	03 <sub>H</sub>	02 <sub>H</sub>	00 <sub>H</sub>	14 <sub>H</sub>

Table 3.6 CRC data calculation table

N	PROCESS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Flag
1	Initial data R = "FFFF"	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2	1 <sup>st</sup> data byte	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
3	CRC = No.1 Xor No.2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	
4	Shift >> 2 (up to flag = 1)	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	CRC = No.4 Xor GP	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	
6	Shift >> 2	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1
7	CRC = No.6 Xor GP	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	
8	Shift >> 2	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
9	CRC = No.8 Xor GP	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	
10	Shift >> 2 (shift of No. 8 terminated)	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1
11	CRC = No.10 Xor GP	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	
12	2 <sup>nd</sup> data byte	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
13	CRC = No.11 Xor No.12	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	
14	Shift >> 1	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1
15	CRC = No.14 Xor GP	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	
16	Shift >> 1	0	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1
17	CRC = No.16 Xor GP	1	1	0	1	0	0	0	0	0	0	0	1	1	1	1	0	
18	Shift >> 2	0	0	1	1	0	1	0	0	0	0	0	0	0	1	1	1	1
19	CRC = No.18 Xor GP	1	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	
20	Shift >> 2	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	1
21	CRC = No.20 Xor GP	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	
22	Shift >> 2 (shift of No. 8 terminated)	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0
23	3 <sup>rd</sup> data byte	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
24	CRC = No.22 Xor No.23	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	1	
25	Shift >> 1	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	1
26	CRC = No.25 Xor GP	1	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0	
27	Shift >> 6	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	0	1
28	CRC = No.27 Xor GP	1	0	1	0	0	0	1	0	1	1	0	0	0	0	1	1	
29	Shift >> 1	0	1	0	1	0	0	0	1	0	1	1	0	0	0	0	1	1
30	CRC = No.29 Xor GP	1	1	1	1	0	0	0	1	0	1	1	0	0	0	0	0	
31	4 <sup>th</sup> data byte	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
32	CRC = No.30 Xor No.31	1	1	1	1	0	0	0	1	0	1	1	0	0	0	1	0	
33	Shift >> 2	0	0	1	1	1	1	0	0	0	1	0	1	1	0	0	0	1
34	CRC = No.33 Xor GP	1	0	0	1	1	1	0	0	0	1	0	1	1	0	0	1	
35	Shift >> 1	0	1	0	0	1	1	1	0	0	0	1	0	1	1	0	0	1
36	CRC = No.35 Xor GP	1	1	1	0	1	1	1	0	0	0	1	0	1	1	0	1	
37	Shift >> 1	0	1	1	1	0	1	1	1	0	0	0	1	0	1	1	0	1

(To be continued)

Table 3.6 CRC data calculation table (continued)

N	PROCESS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Flag
38	CRC = No.37 Xor GP	1	1	0	1	0	1	1	1	0	0	0	1	0	1	1	1	
39	Shift >> 1	0	1	1	0	1	0	1	1	1	0	0	0	1	0	1	1	1
40	CRC = No.39 Xor GP	1	1	0	0	1	0	1	1	1	0	0	0	1	0	1	0	
41	Shift >>2	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	0	1
42	CRC = No.41 Xor GP	1	0	0	1	0	0	1	0	1	1	1	0	0	0	1	1	
43	Shift >> 1	0	1	0	0	1	0	0	1	0	1	1	1	0	0	0	1	1
44	CRC = No.43 Xor GP	1	1	1	0	1	0	0	1	0	1	1	1	0	0	0	0	
45	5 <sup>th</sup> data byte	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
46	CRC = No.44 Xor No.45	1	1	1	0	1	0	0	1	0	1	1	1	0	0	0	0	
47	Shift >> 5	0	0	0	0	0	1	1	1	0	1	0	0	1	0	1	1	1
48	CRC = No.47 Xor GP	1	0	1	0	0	1	1	1	0	1	0	0	1	0	1	0	
49	Shift >> 2	0	0	1	0	1	0	0	1	1	1	0	1	0	0	1	0	1
50	CRC = No.49 Xor GP	1	0	0	0	1	0	0	1	1	1	0	1	0	0	1	1	
51	Shift >> 1	0	1	0	0	0	1	0	0	1	1	1	0	1	0	0	1	1
52	CRC = No.51 Xor GP	1	1	1	0	0	1	0	0	1	1	1	0	1	0	0	0	
53	6 <sup>th</sup> data byte	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
54	CRC = No.52 Xor No.53	1	1	1	0	0	1	0	0	1	1	1	1	1	1	0	0	
55	Shift >> 3	0	0	0	1	1	1	0	0	1	0	0	1	1	1	1	1	1
56	CRC = No.55 Xor GP	1	0	1	1	1	1	0	0	1	0	0	1	1	1	1	0	
57	Shift >> 2	0	0	1	0	1	1	1	1	0	0	1	0	0	1	1	1	1
58	CRC = No.57 Xor GP	1	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0	
59	Shift >> 2	0	0	1	0	0	0	1	1	1	1	0	0	1	0	0	1	1
60	CRC = No.59 Xor GP	1	0	0	0	0	0	1	1	1	1	0	0	1	0	0	0	
61	Shift >> 1 (shift of No. 8 terminated)	0	1	0	0	0	0	0	1	1	1	1	0	0	1	0	0	0
	<b>Transmitted CRC data</b>	<b>4</b>				<b>1</b>				<b>E</b>				<b>4</b>				

From the above calculation, the transmitted data is as shown below:

Station address	FC	Function code		Number of data read		CRC check	
01 <sub>H</sub>	03 <sub>H</sub>	03 <sub>H</sub>	02 <sub>H</sub>	00 <sub>H</sub>	14 <sub>H</sub>	41 <sub>H</sub>	E4 <sub>H</sub>

### 3.4.4 Frame length calculation

To calculate CRC-16, it is necessary to know the length of variable length messages. The length of all types of messages can be determined according to Table 3.7 Lengths of response messages.

Table 3.7 Length of response messages

FC	Description	Query/Broadcast message length (except CRC code)	Length of response message (except CRC code)
3	Function reading	6 bytes	3 + (3 <sup>rd</sup> ) bytes*
6	Single function writing	6 bytes	6 bytes
8	Maintenance code	6 bytes	6 bytes
16	Serial function writing	7 + (7 <sup>th</sup> ) bytes*	6 bytes
128 to 255	Exception function	Unused	3 bytes

\* 7<sup>th</sup>, 3<sup>rd</sup>: The 7<sup>th</sup> and 3<sup>rd</sup> byte count values stored in the frame.