Application Note 5484
FreeWave Technologies

Understanding Modbus Commands, Responses and Failures

Applies To

FreeWave Technologies:

- IOE-X-4422PC
- IOE-X-4422P
- FGR2-IO-IOE
- FGR2-IOS

Summary

Modbus is a communication interface used for communication between devices connected on different types of buses and networks. The Modbus protocol uses a request/reply architecture where a network master issues requests to specific slaves and the specific slave responds. By understanding the information contained in communication packets, IO users will have a clear picture of the state of remote equipment.

This application note describes Modbus function codes supported by FreeWave devices, their respective responses and failure codes that may occur.

Modbus

All Modbus commands share a general Modbus frame. The Modbus frame combines the Modbus ID, Modbus function code, data and error checking in a specific order. Figure 1 shows the general format of a Modbus frame. The address is Modbus ID of the target equipment. The function code is one of the supported Modbus function codes. The data is the value to write to the target equipment. The error check field is a cyclical redundancy check that is used to confirm that the data received is correct.

Figure 1. Data organization of a Modbus frame [1]

Address	Function Code	Data	Error Check
1 byte	1 byte	N bytes	2 bytes

The IO product family supports the following Modbus commands:

- 1: Read Coils
- 2: Read Discrete Inputs
- 3: Read Holding Registers
- 4: Read Input Registers
- 5: Write Single Coil
- 6: Write Single Register
- 15: Write Multiple Coils
- 16: Write Multiple Registers

For all function codes, if a processing error occurs at the slave device, the returned function code will be the original request function code plus 0x80 (128 in decimal). For example a failed request for command 1 will return a response where the function code is 0x81 (129 in decimal).

Understanding Modbus Commands, Responses and Failures

Extended Modbus Addressing

As shown in figure 1, the Modbus specification uses 1 byte (8 bits) for the address field. This limits the number of addressable slave to 246. To increase the number of Modbus slaves addressable by a single Modbus master, FreeWave Technologies developed extended Modbus addressing. Extended Modbus addressing uses an address field that is 2 bytes (16 bits) long. Extended Modbus addressing allows Modbus slave addresses to range from 1 to 65,535.

Figure 2. Data organization with extended Modbus addressing

Address	Function Code	Data	Error Check
2 bytes	1 byte	N bytes	2 bytes

Extended Modbus addressing does not change any data field other than the address field.

Function Code 1: Read Coil

This function is used to read between 1 and 2000 contiguous status of coils in a slave. The coils in the response message are packed as one coil per bit of the data field. Status is indicated as 1 = ON and 0 = OFF. The least significant bit of the first data byte contains the output addressed in the query. The other coils follow toward the high order end of this byte and from low order to high order in subsequent bytes. [1]

Table 1.1 shows the structure of the request. The example asks the equipment at Modbus ID 1 for the contents of 1 coil starting at address 1. Table 1.2 shows the structure of the response.

Table 1.1. Request structure for function 1 (in hexadecimal)

Field	Address	Function	Starting address	Number of coils	Checksum
Size	1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Sample	01	01	00 01	00 01	AC 0A

Figure 1.2. Response structure for function 1 (in hexadecimal)

Field	Address	Function	Byte count	Coil status	Checksum
Size	1 Byte	1 Byte	1 Bytes	n Bytes	2 Bytes
Sample	01	01	01	01	90 48

Table 1.3 shows the error code for a failed command. Consult the list of exception codes in table 9 on page 10 to determine the meaning of the failure.

Figure 1.3. Error structure for function 1 (in hexadecimal)

Field	Address	Function	Exception code	Checksum
Size	1 Byte	1 Byte	1 Byte	2 Bytes
Sample	01	81	02	C1 91

Function Code 2: Read Discrete Inputs

This function code is used to read from 1 to 2000 contiguous status of discrete inputs in a slave. The discrete inputs in the response message are packed as one input per bit of the data field. Status is indicated as 1 = ON and 0 = OFF. The least significant bit of the first data byte contains the input addressed in the query. The other inputs follow toward the high order end of this byte, and from low order to high order in subsequent bytes. [1]

Table 2.1 shows the structure of the request. The example asks the slave at Modbus ID 1 for the contents of one (1) discrete input starting at address 1. Table 2.2 shows the structure of the response to the request in table 2.1.

Table 2.1. Request structure (in hexadecimal) for function 2

Field	Address	Function	Starting address	Number of inputs	Checksum
Size	1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Sample	01	02	00 01	00 01	E8 0A

Figure 2.2. Response structure (in hexadecimal) for function 2 in table 2.1

Field	Address	Function	Byte count	Discrete input	Checksum
Size	1 Byte	1 Byte	1 Bytes	n Bytes	2 Bytes
Sample	01	02	01	01	90 48

Table 2.3 shows the error code for a failed command. Consult the list of exception codes in table 9 on page 10 to determine the meaning of the failure.

Figure 2.3. Error structure for function 2 (in hexadecimal)

Field	Address	Function	Exception code	Checksum
Size	1 Byte	1 Byte	1 Byte	2 Bytes

Sample	01	82	02	C1 61

Understanding Modbus Commands, Responses and Failures

Function Code 3: Read Holding Registers

This function code is used to read the contents of a contiguous block of holding registers in a slave. The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the second contains the low order bits. [1]

Table 3.1 shows the structure of the request. Table 3.2 shows the structure of the response to the request in table 3.1. The example asks the slave at Modbus ID 1 for the contents of one (1) holding register starting at address 1.

Table 3.1. Request structure (in hexadecimal) for function 3

Field	Address	Function	Starting address	Number of registers	Checksum
Size	1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Sample	01	03	00 01	00 01	D5 CA

Figure 3.2. Response structure (in hexadecimal) for function 3 in table 3.1

Field	Address	Function	Byte count	Holding register	Checksum
Size	1 Byte	1 Byte	1 Bytes	2 * n Bytes	2 Bytes
Sample	01	03	02	FF FF	B9 F4

Table 3.3 shows the error code for a failed command. Consult the list of exception codes in table 9 on page 10 to determine the meaning of the failure.

Figure 3.3. Error structure for function 3 (in hexadecimal)

Field	Address	Function	Exception code	Checksum
Size	1 Byte	1 Byte	1 Byte	2 Bytes
Sample	01	83	02	C0 F1

Understanding Modbus Commands, Responses and Failures

Function Code 4: Read Input Registers

This function code is used to read from 1 to 125 contiguous input registers in a slave. The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the second contains the low order bits. [1]

Table 4.1 shows the structure of the request. Table 4.2 shows the structure of the response to the request in table 4.1. The example asks the slave at Modbus ID 1 for the contents of one (1) input register starting at address 1.

Table 4.1. Request structure (in hexadecimal) for function 4

Field	**Address**	**Function**	**Starting address**	**Number of registers**	**Checksum**
Size	1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Sample	01	04	00 01	00 01	60 0A

Figure 4.2. Response structure (in hexadecimal) for function 4 in table 4.1

Field	**Address**	**Function**	**Byte count**	**Input Register**	**Checksum**
Size	1 Byte	1 Byte	1 Bytes	2 * n Bytes	2 Bytes
Sample	01	04	02	00 00	B9 30

Table 4.3 shows the error code for a failed command. Consult the list of exception codes in table 9 on page 10 to determine the meaning of the failure.

Figure 4.3. Error structure for function 4 (in hexadecimal)

Field	**Address**	**Function**	**Exception code**	**Checksum**
Size	1 Byte	1 Byte	1 Byte	2 Bytes
Sample	01	84	02	C2 C1

Function Code 5: Write Single Coil

This function code is used to write a single output to either ON or OFF in a slave. The requested state is specified by a constant in the request data field. A value of 0xFF 0x00 requests the output to be ON. A value of 0x00 0x00 requests it to be OFF. All other values are illegal and will not affect the output. [1]

Table 5.1 shows the structure of the request. Table 5.2 shows the structure of the response to the request in table 5.1. The example writes a value of 0 to the coil at address 1 for the slave at Modbus ID 1.

Table 5.1. Request structure (in hexadecimal) for function 5

Field	Address	Function	Output address	Output Value	Checksum
Size	1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Sample	01	05	00 01	00 00	9C 0A

Figure 5.2. Response structure (in hexadecimal) for function 5 in table 5.1

Field	Address	Function	Output address	Output Value	Checksum
Size	1 Byte	1 Byte	1 Bytes	2 Bytes	2 Bytes
Sample	01	05	00 01	00 00	9C 0A

Table 5.3 shows the error code for a failed command. Consult the list of exception codes in table 9 on page 10 to determine the meaning of the failure.

Figure 5.3. Error structure for function 5 (in hexadecimal)

Field	Address	Function	Exception code	Checksum
Size	1 Byte	1 Byte	1 Byte	2 Bytes
Sample	01	85	02	C3 51

Function Code 6: Write Single Register

This function is used to write a single holding register in a slave. The normal response is an echo of the request, returned after the register constants have been written. [1]

Table 6.1 shows the structure of the request. Table 6.2 shows the structure of the response to the request in table 6.1. The example writes a value of 0 to the register at address 1 for the slave at Modbus ID 1.

Table 6.1. Request structure (in hexadecimal) for function 6

Field	Address	Function	Output register address	Output register value	Checksum
Size	1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Sample	01	06	00 01	00 00	D8 0A

Figure 6.2. Response structure (in hexadecimal) for function 6 in table 6.1

Field	Address	Function	Output register address	Output register value	Checksum
Size	1 Byte	1 Byte	1 Bytes	2 Bytes	2 Bytes
Sample	01	06	00 01	00 00	D8 0A

Table 6.3 shows the error code for a failed command. Consult the list of exception codes in table 9 on page 10 to determine the meaning of the failure.

Figure 6.3. Error structure for function 6 (in hexadecimal)

Field	Address	Function	Exception code	Checksum
Size	1 Byte	1 Byte	1 Byte	2 Bytes
Sample	01	86	02	C3 A1

Function Code 15: Write Multiple Coils

This function is used to force each coil in a sequence of coils to either ON or OFF in a slave. The requested ON or OFF states are specified by contents of the request data field. A logical '1' in a bit position of the field requests the corresponding output to be ON. A logical '0' requests it to be OFF. [1]

Table 7.1 shows the structure of the request. Table 7.2 shows the structure of the response to the request in table 7.1. The example writes a value of 1 to two registers starting at address 1 for the slave at Modbus ID 1.

Table 7.1. Request structure (in hexadecimal) for function 7

Field	Address	Function	Start address	Quantity of Outputs	Byte Count	Outputs Value	Checksum
Size	1 Byte	1 Byte	2 Bytes	2 Bytes	1 Bytes	N Bytes	2 Bytes
Sample	01	0F	00 01	00 02	01	03	A3 56

Figure 7.2. Response structure (in hexadecimal) for function 15 in table 7.1

Field	Address	Function	Starting Address	Quantity of Outputs	Checksum
Size	1 Byte	1 Byte	1 Bytes	2 Bytes	2 Bytes
Sample	01	0F	00 01	00 02	85 CA

Table 7.3 shows the error code for a failed command. Consult the list of exception codes in table 9 on page 10 to determine the meaning of the failure.

Figure 7.3. Error structure for function 15 (in hexadecimal)

Field	Address	Function	Exception code	Checksum
Size	1 Byte	1 Byte	1 Byte	2 Bytes
Sample string	01	8F	02	C5 F1

Function Code 16: Write Multiple Registers

This function code is used to write a block of contiguous (1 to 123 registers) in a slave. The requested written values are specified in the request data field. Data is packed as two bytes per register. The normal response returns the function code, starting address, and quantity of registers written. [1]

Table 8.1 shows the structure of the request. Table 8.2 shows the structure of the response to the request in table 8.1. The example writes a value of 1 to the register at address 1 for the slave at Modbus ID 1.

Table 8.1. Request structure (in hexadecimal) for function 16

Field	**Address**	**Function**	**Start address**	**Quantity of Registers**	**Byte Count**	**Registers Value**	**Checksum**
Size	1 Byte	1 Byte	2 Bytes	2 Bytes	1 Bytes	N Bytes	2 Bytes
Sample	01	10	00 01	00 01	02	00 01	66 41

Figure 8.2. Response structure (in hexadecimal) for function 15 in table 8.1

Field	**Address**	**Function**	**Starting Address**	**Quantity of Outputs**	**Checksum**
Size	1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
Sample	01	10	00 01	00 01	50 09

Table 8.3 shows the error code for a failed command. Consult the list of exception codes in table 9 on page 10 to determine the meaning of the failure.

Figure 8.3. Error structure for function 16 (in hexadecimal)

Field	**Address**	**Function**	**Exception code**	**Checksum**
Size	1 Byte	1 Byte	1 Byte	2 Bytes
Sample string	01	90	02	CD C1

Understanding Modbus Commands, Responses and Failures

Modbus Exception Codes

All failed Modbus requests will return an exception code as part of the response. The following table shows the meaning of the different exception codes.

Table 9. Modbus exception codes [1].

Code	Name	Meaning
01	Illegal function	The function code received is not supported by the slave. It can indicate that the slave is an older model that does not support the function or it could indicate that the slave is in the wrong state to process a request of this type (e.g. in configuration).
02	Illegal data address	The combination of data address and data length is not valid for the slave.
03	Illegal data value	A value in the query data field is not an allowable value for the slave. It indicates a fault in the structure of the remainder of a complex request, such as an incorrect length. It does not indicate a data item submitted for storage is outside the expected range, since the Modbus protocol is unaware of any particular value for any particular register.
04	Slave device failure	An unrecoverable error occurred while the slave was attempting to perform the requested action.
05	Acknowledge	This is used in special cases in conjunction with programming commands. The slave has accepted the request and is processing it, but a long duration of time will be required to do so.
06	Slave device busy	This is used in special cases in conjunction with programming commands. The slave is engaged in processing a long-duration command. The master should retransmit the message later when the slave is free.

References

[1] Modbus Organization Inc., "MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b," December 2006. [Online]. Available: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_ 1b.pdf [Accessed: January 29, 2010].