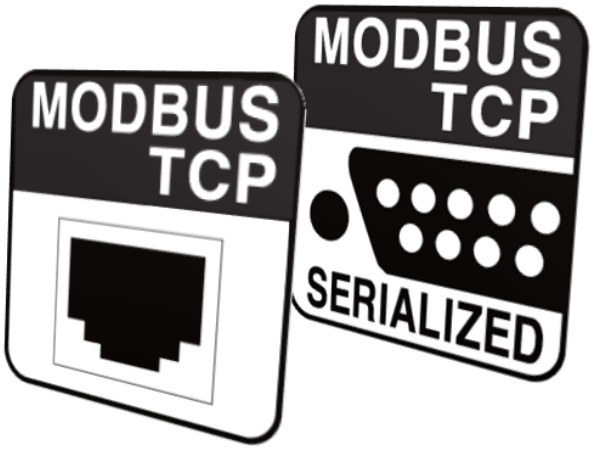
Application Note

Modbus/TCP of ezTCP

Version 1.7



Sollae Systems

https://www.ezTCP.com/

1 Contents

2 Introduction

2.1 Overview

MODBUS is a communication protocol widely used in the world for monitoring and control of various types of automation devices such as a PLC (Programmable Logic Controller). MODBUS is an application layer messaging protocol providing client/server communication between devices connected to different types of buses or networks. This document introduces the MODBUS messaging service over TCP/IP which is called Modbus/TCP and used in I/O gateway of Sollae Systems to manage its I/O ports.

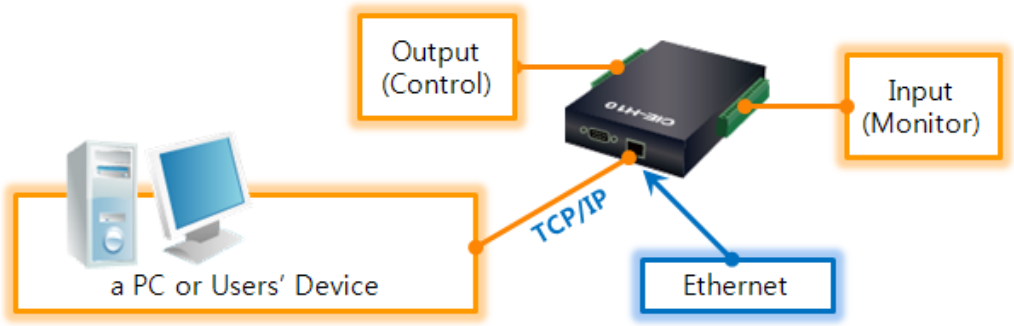


Figure 2-1 A diagram of Modbus/TCP system

There is a "Serialized Modbus/TCP" communication mode in case a user needs to communicate with I/O gateway through a serial port (related product: CIE series). Note that "Serialized Modbus/TCP" does NOT mean the standard MODBUS protocol over serial line. It just sends and receives data of the Modbus/TCP via a RS232 port.

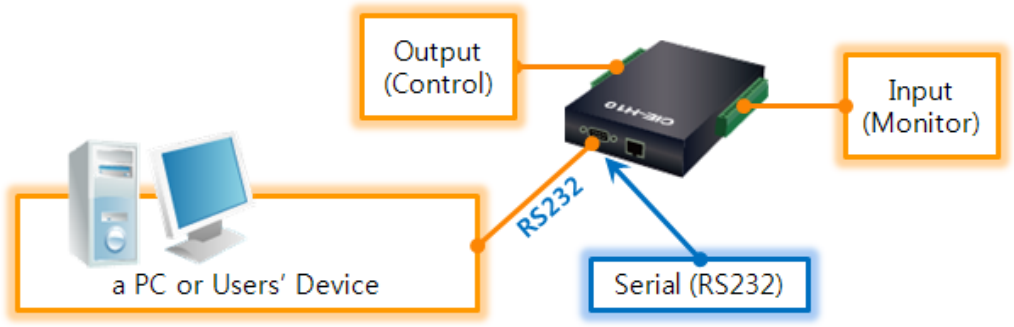


Figure 2-2 A diagram of Serialized Modbus/TCP system

2.2 Available products

- SIG series – SIG-5430, SIG-5440, SIG-5450, SIG-5600
- CIE series – CIE-H12A, CIE-H14A, CIE-H10A, CIE-M10A and etc
- EZI-10

2.3 Terms and Notations

2.3.1 Port Type

There are 3 kinds of ports.

- Digital Input Port
- Analog Input Port
- Digital Output Port

2.3.2 Port Number

The input/output ports are expressed as decimal numbers with '#' symbol. It is started from zero and increased by 1 in sequence. For example, CIE-H10A has 8 digital input / output ports, and they are represented as below.

- The 1st port – #0
- The 8th port – #7

2.3.3 MSB / LSB

- MSB (Most Significant Bit) – the bit position with the greatest value, the left-most bit
- LSB (Least Significant Bit) – the bit position with the least value, the right-most bit

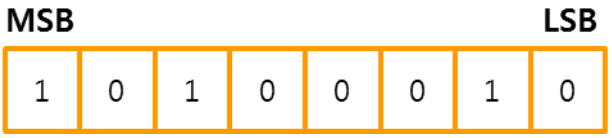


Figure 2-3 MSB and LSB

2.3.4 Abbreviations

- ADU Application Data Unit
- HMI Human Machine Interface
- MBAP MODBUS Application Protocol
- PDU Protocol Data Unit
- SCADA Supervisory Control And Data Acquisition

3 Protocol Overview

3.1 MODBUS

3.1.1 General Description

- An application layer messaging protocol (the OSI 7 layer)
- PDU (Protocol Data Unit)
  A simple data unit independent of the underlying communication layers

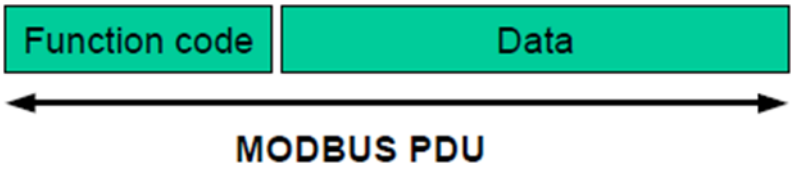


Figure 3-1 MODBUS PDU

- ADU (Application Data Unit)
  A data unit of MODBUS protocol on specific buses or network, some additional fields on PDU

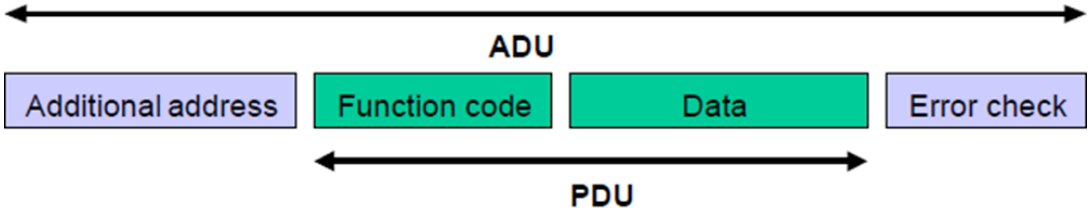


Figure 3-2 General MODBUS frame

☞ ***Note that the previous figure does NOT show a Modbus/TCP frame. Refer to 3.2.3 Modbus/TCP Frame for an ADU of Modbus/TCP.***

- Client/Server communication model

Figure 3-3 MODBUS Client/Server Model

3.1.2 Data Encoding

MODBUS uses the big-endian approach. Big-endian sends the most significant byte first. For example, the hexadecimal number 0x1234 would be sent in order of 0x12 and 0x34.

3.1.3 MODBUS Data model

MODBUS data items are categorized into four types based on the distinction between inputs and outputs, and between bit-addressable and word-addressable.

Data	Access	R/W	Descriptions
Discrete Inputs	bit	Read Only	Provided by a device
Coils	bit	Read/Write	Controlled by an HMI/SCADA
Input Registers	16-bit word	Read Only	Provided by a device
Holding Registers	16-bit word	Read/Write	Controlled by an HMI/SCADA

Table 3-1 MODBUS Data Model

3.1.4 Memory structure of I/O Gateway

MODBUS standard doesn't define the memory structure of the device, so it depends on the vendor of the device. Sollae System's I/O gateway has only 1 data block. All data model items are mapped into 1 data block, which means the same data can be reached via several MODBUS functions, either via a 16-bit access or via an access bit.

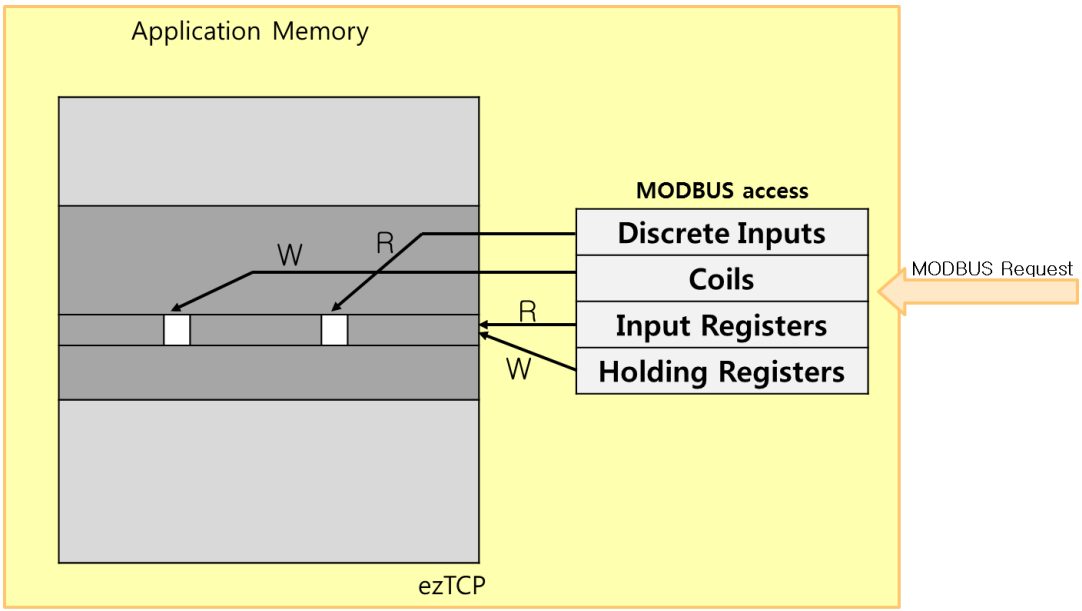


Figure 3-4 Memory structure of I/O Gateway

3.1.5 MODBUS Addressing by HMI/SCADA with ezTCP

In a MODBUS PDU, each data is addressed from 0 to 65535. On the other hand, in the MODBUS data model, each element within a data block is numbered from 1 to n.

Data	Reference on HMI/SCADA	Function Code	I/O Gateway
Coils	0X reference	01, 05, 15	Outputs (Digital)
Discrete Inputs	1X reference	02	Inputs (Digital)
Input Registers	3X reference	04	Inputs (Digital/Analog)
Holding Registers	4X reference	03, 06, 16	Inputs/Outputs (Digital/Analog)

Table 3-2 MODBUS Addressing with I/O Gateway

The examples below show MODBUS addressing in I/O Gateway.

- (Example 1) Digital Input Port Address – if the parameter value is '0'

Value	Data	FC	Address on HMI/SCADA
0	Coils	01, 05, 15	Not Available
	Discrete Inputs	02	10001
	Input Registers	04	30001
	Holding Registers	03, 06, 16	40001

Table 3-3 Example of Input Ports Address

- (Example 2) Digital Output Port Address – if the parameter value is '8'

Value	Data	FC	Address on HMI/SCADA
8	Coils	01, 05, 15	00009
	Discrete Inputs	02	Not Available
	Input Registers	04	Not Available
	Holding Registers	03, 06, 16	40009

Table 3-4 Example of Output Port Address

3.2 Modbus/TCP

3.2.1 General Description

- MODBUS protocol over TCP/IP network
- Connection oriented

  As you can guess from its name, Modbus/TCP uses a TCP which is a reliable transport layer protocol. Modbus/TCP uses TCP port 502 as a default.

3.2.2 Communication Model

In the standard, Modbus/TCP protocol defines client/server model. The communication is initiated by a client, who sends a request, while a server responds to the client's request by sending a response.

However, Modbus/TCP in I/O Gateway uses Master/Slave model. Both a master and a slave can be an initiator (TCP Client) by user configuration.

- Master

  A master sends a request to a slave periodically. After then, a master waits for a response from a slave.

- Slave

  When a request is arrived from a master, a slave sends a response to a master.

3.2.3 Modbus/TCP Frame

The application data unit (ADU) for Modbus/TCP is as below:

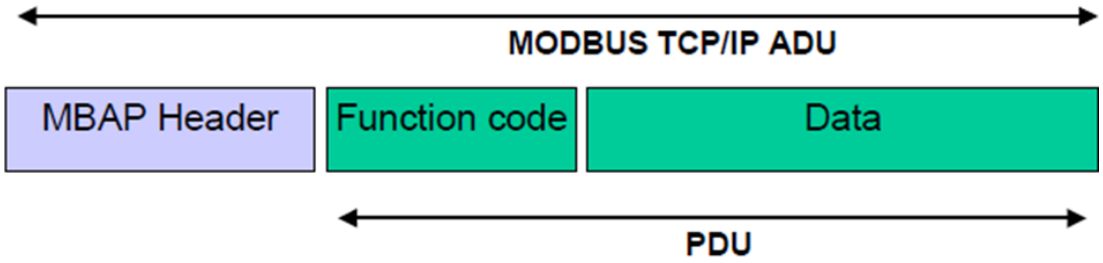


Figure 3-5 Modbus/TCP Frame

3.2.4 MBAP Header

The MBAP Header has 4 fields as below:

Fields	Length	Description
Transaction Identifier	2 bytes	Identification of a MODBUS request/response transaction
Protocol Identifier	2 bytes	0 = MODBUS protocol
Length	2 bytes	Number of following bytes
Unit Identifier	1 byte	Identification of a remote slave connected on a serial line or on other buses

Table 3-5 MBAP Header

Format of Modbus/TCP frame is described in the figure below.

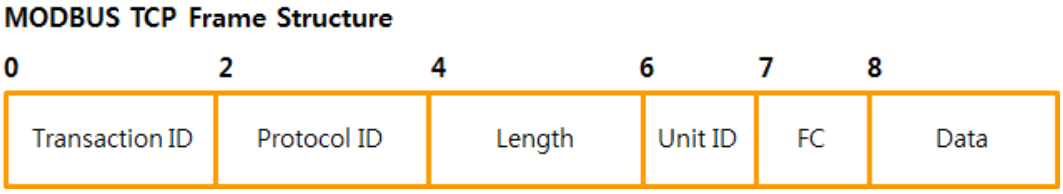


Figure 3-6 the format of Modbus/TCP frame

- byte 0 ~ 1: transaction ID (Transaction Identification)
  This means the sequence number of queries and responses. While I/O gateway operates as a master, it is incremented by one in every query (It does not matter if this field is set to 0x0000).
- byte 2 ~ 3: protocol ID (Protocol Identification)
  This means the protocol identification and the value is fixed as 0x0000 for Modbus/TCP
- byte 4 ~ 5: length
  The value of this means the number of bytes from next byte of length field to the end of the frame.
- byte 6: unit ID (Unit Identification)
- byte 7: FC (Function Code)
- byte 8~ : data depending on function code

☞ ***The maximum size of the Modbus/TCP ADU is 260 bytes.***

3.3 Function Codes

3.3.1 Categories

The function code defines the service provided in MODBUS protocol. Valid codes are in the range of 1 ~ 255 decimal, but the range 128 ~ 255 is reserved and used for exception responses. There are three categories of MODBUS function codes as below.

- Public Function Codes
  Publicly documented and validated by MODBUS.org community
  1 ~ 64, 73 ~ 99, 111 ~ 127
- User-Defined Function Codes
  Dependent on the vendor
  65 ~ 72, 100 ~ 110
- Reserved Function Codes
  Some of the Public Function Codes range, not available for public use but currently used by several companies for legacy products.
  8/19, 8/21~65535, 9, 10, 13, 14, 41, 42, 90, 91, 125, 126, 127

Number	Function Codes
111 ~ 127	Public
100 ~ 110	User-Defined
73 ~ 99	Public
65 ~ 72	User-Defined
1 ~ 64	Public

Figure 3-7 Function Code Categories

3.3.2 Public Function Codes

Type	Access	Ports	Name	Function Codes	
				Code	Sub-code
Data	Bit Access	Inputs (Digital)	Read Discrete Inputs	02 (0x02)	
		Outputs (Digital)	Read Coils	01 (0x01)	
			Write Single Coil	05 (0x05)	
			Write Multiple Coils	15 (0x0F)	
	16 bits Access	Inputs	Read Input Registers	04 (0x04)	
		Inputs Outputs	Read Holding Registers	03 (0x03)	
		Outputs	Write Single Register	06 (0x06)	
		Outputs	Write Multiple Registers	16 (0x10)	
Diagnostics			Read Exception Status	07 (0x07)	
			Read Device Identification	43 (0x2B)	14 (0x0E)
Other			Encapsulated Interface Transport	43 (0x2B)	

Table 3-6 Public Function Codes in I/O gateway

3.3.3 User-Defined Function Codes

Type	Access	Ports	Name	Function Codes	
				Code	Sub-code
Data	Bit Access	Outputs (Digital)	Write Pulse	105 (0x69)	
Data	16 bits Access	Inputs Outputs	Send Notify	108 (0x6C)	

Table 3-7 User-Defined Function Code in I/O gateway

☞ ***Supported function codes in I/O gateway can be different from the above tables depending on the product model and firmware version.***

4 Public Function Codes

4.1 Read Coils (FC 01)

This function code is used to read the status of digital output ports.

4.1.1 Request

Request of Read Coils

0	1	3
Function code (0x01)	Starting Address	Quantity of Outputs (1 ~ n)

Figure 4-1 Request of Read Coils

- byte 0: Function Code
  Function code of Read Coils is 0x01.
- byte 1~2: Starting Address
  This is the address of the first digital output port to read.
- byte 3~4: Quantity of Outputs
  Request PDU specifies the number of digital output ports ranging from 1 to n (1 ~ n).

☞ ***n: the number of digital output ports for each product***

4.1.2 Response

Response of Read Coils

0	1	2
Function code (0x01)	Byte Count (0x01)	Output Status

Figure 4-2 Response of Read Coils

- byte 0: Function Code (0x01)
- byte 1: Byte Count (0x01)
  (Quantity of Outputs + 7) / 8
- byte 2: Output Status
  This value shows the status of digital output ports. The data in the response message is packed as one byte per eight outputs. And this value represents output ports as one port per bit from the LSB to the MSB. Status is indicated as 1= ON and 0= OFF. If the requested quantity is not a multiple of eight, the remaining bits in the final data byte will be padded with zeros (toward the high order end of the byte).

Bit Order for Output Status	MSB							LSB
Digital Output Ports (DO)	#7	#6	#5	#4	#3	#2	#1	#0

Figure 4-3 Status of Output Ports

4.1.3 Exception

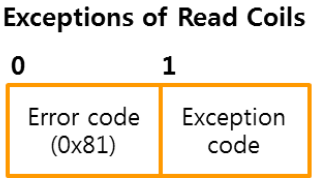


Figure 4-4 Exceptions of Read Coils

- byte 0: Error Code
  Function code + 0x80,
- byte 1: Exception Code
  Exception code can be 0x01, 0x02 or 0x03.

4.1.4 Examples

The example below shows reading digital output ports #0 ~ #7.

- Request

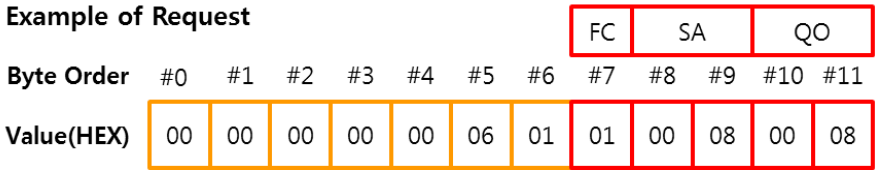


Figure 4-5 Request ADU

byte order	value	description
7	0x01	Function code
8~9	0x0008	Starting address to read is 8
10~11	0x0008	Read 8 digital output ports

Table 4-1 Request PDU

- Response

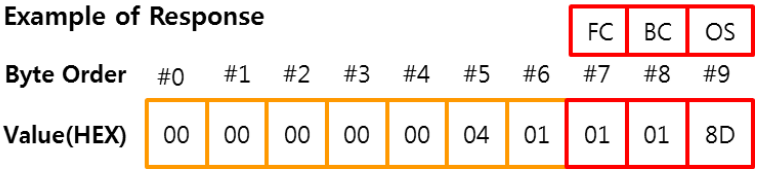


Figure 4-6 Response ADU

byte order	value	description
7	0x01	Function code
8	0x01	1 byte, 1 ~ 8 digital output ports
9	0x8D	(1000 1101) #0, 2, 3, 7 ON / #1, 4 ~ 6 OFF

Table 4-2 Response PDU

4.2 Read Discrete Inputs (FC 02)

This function code is used to read the status of digital input ports.

4.2.1 Request

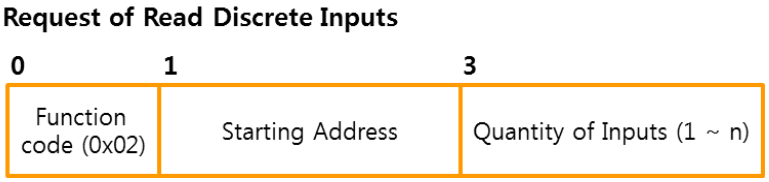


Figure 4-7 Request of Read Discrete Inputs

- byte 0: Function Code
  Function code of Read Discrete Input is 0x02.
- byte 1~2: Starting Address
  This is the address of the first digital input port to read.
- byte 3~4: Quantity of Inputs
  Request PDU specifies the number of digital input ports ranging from 1 to n (1 ~ n).

☞ ***n: the number of digital input ports for each product***

4.2.2 Response

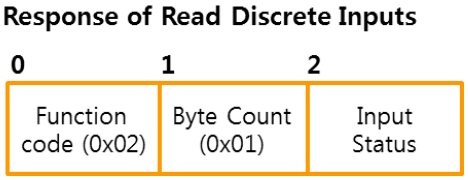


Figure 4-8 Response of Read Discrete Inputs

- byte 0: Function Code (0x02)
- byte 1: Byte Count (0x01)
  (Quantity of Inputs + 7) / 8
- byte 2: Input Status
  This value shows the status of digital input ports. The data in the response message is packed as one byte per eight inputs. And this value represents input ports as one port per bit from the LSB to the MSB. Status is indicated as 1= ON and 0= OFF. If the requested quantity is not a multiple of eight, the remaining bits in the final data byte will be padded with zeros (toward the high order end of the byte).

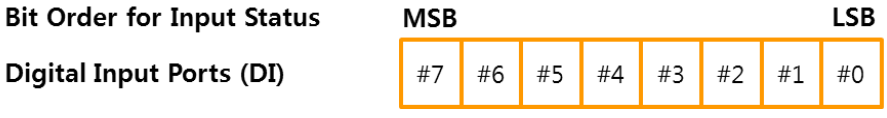


Figure 4-9 Status of Input Ports

4.2.3 Exception

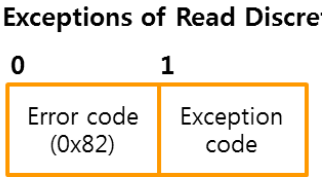


Figure 4-10 Exceptions of Read Discrete Inputs

- byte 0: Error Code
  Function code + 0x80

- byte 1: Exception Code
  Exception code can be 0x01, 0x02 or 0x03.

4.2.4 Examples

The example below shows reading digital input ports #0 ~ #7.

- Request

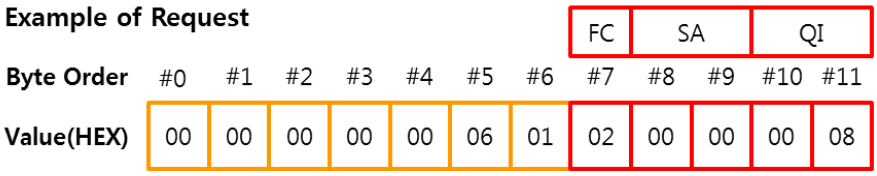


Figure 4-11 Request ADU

byte order	value	description
7	0x02	Function code
8~9	0x0000	Starting address to read is 0
10~11	0x0008	Read 8 digital input ports

Table 4-3 Request PDU

- an example of response

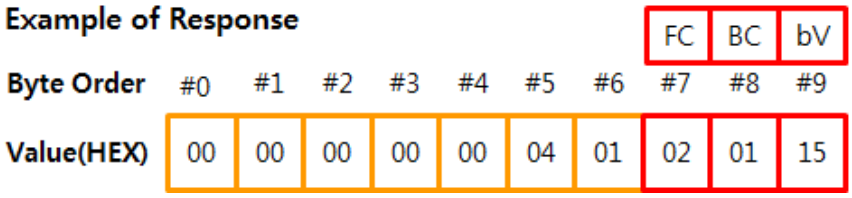


Figure 4-12 Response ADU

byte order	value	description
7	0x02	Function code
8	0x01	1 byte, 1 ~ 8 digital input ports
9	0x15	(0001 0101) #0, 2, 4 ON / #1, 3, 5 ~ 7 OFF

Table 4-4 Response PDU

4.3 Read Holding Registers (FC 03)

This function code is used to read the status of digital/analog inputs and digital outputs.

4.3.1 Request

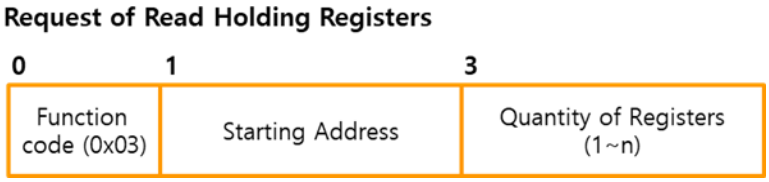


Figure 4-13 Request of Read Holding Registers

- byte 0: Function Code
  Function code of Read Holding Registers is 0x03.
- byte 1 ~ 2: Starting Address
  This is the address of the first register to read.
- byte 3 ~ 4: Quantity of Registers
  Request PDU specifies the number of registers. The available value for this field is 1 ~ n.

  ☞ ***n: 125 (SIG series), 8 (CIE series), 1 (EZI-10)***

4.3.2 Response

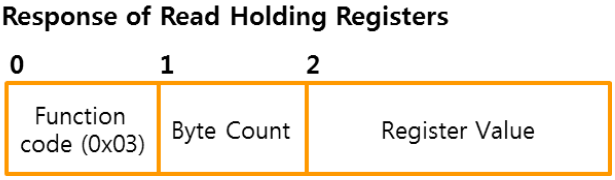


Figure 4-14 Response of Read Holding Registers

- byte 0: Function Code (0x03)
- byte 1 ~ 2: Byte Count
  "Quantity of Registers" × 2
- byte 2 ~ 3: Register Value
  This value shows the status of digital/analog input ports and digital output ports. The register data in the response message are packed as two bytes per register. And the register value represents digital ports as one port per bit from the LSB to the MSB. Status is indicated as 1= ON and 0= OFF. If there are bits mapped to nonexistent ports, they will be padded with zeros (toward the high order end of the register value).

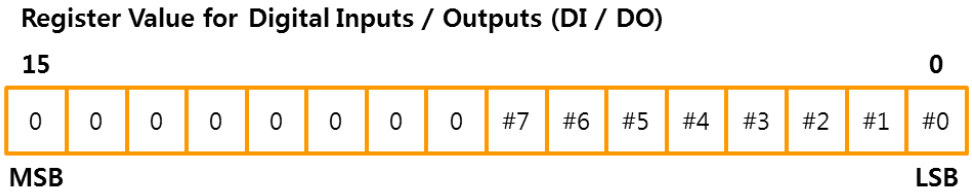


Figure 4-15 Register Value for Digital Ports

The register value also represents analog input ports ranging from 0 to n (0 ~ n).

☞ ***n: 4095 (SIG series, 12-bit Resolution), 1023 (CIE series, 10-bit Resolution)***

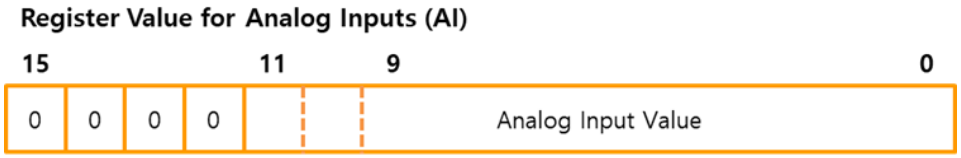


Figure 4-16 Register Value for Analog Ports

4.3.3 Exceptions

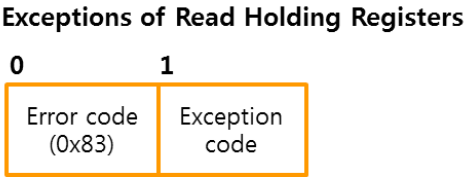


Figure 4-17 Exceptions of Read Holding Registers

- byte 0: Error Code
  Function code + 0x80
- byte 1: Exception Code
  Exception code can be 0x01, 0x02 or 0x03

4.3.4 Examples

The example below shows reading input ports.

● Request

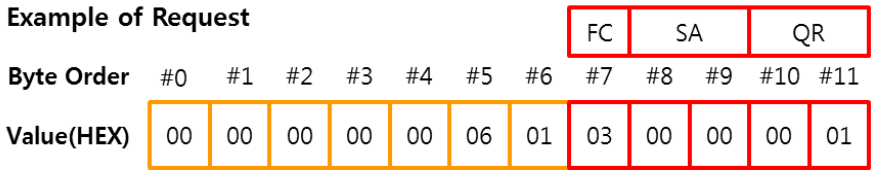


Figure 4-18 Request ADU

byte order	value	description
7	0x03	Function code
8~9	0x0000	Starting address to read is 0
10~11	0x0001	Read 1 register

Table 4-5 Request PDU

● Response

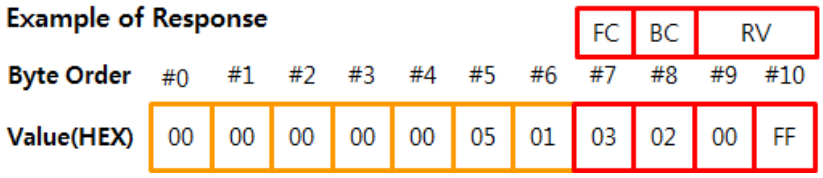


Figure 4-19 Response ADU

byte order	value	description
7	0x03	Function code
8	0x02	2 bytes, 1 register
9~10	0x00FF	(1111 1111) Digital Inputs #0 ~ 7 ON

Table 4-6 Response PDU

4.4 Read Input Registers (FC 04)

This function code is used to read the status of digital/analog inputs.

4.4.1 Request

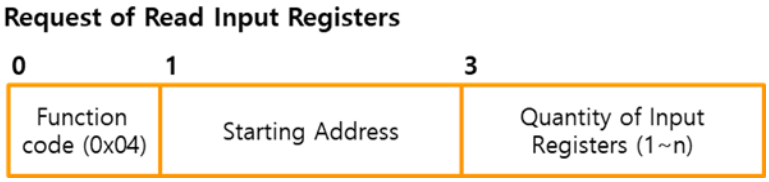


Figure 4-20 Request of Read Input Registers

- byte 0: Function Code
  Function code of Read Input Registers is 0x04.
- byte 1~2: Starting Address
  This is the address of the first register to read.
- byte 3~4: Quantity of Input Registers
  Request PDU specifies the number of registers. The available value for this field is 1 ~ n.

  ☞ ***n: 125 (SIG series), 8 (CIE series), 1 (EZI-10)***

4.4.2 Response

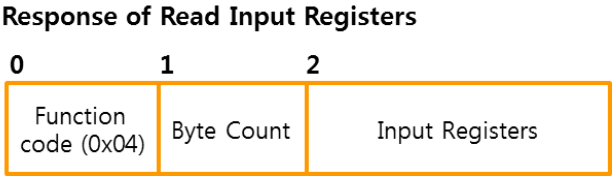


Figure 4-21 Response of Read Input Registers

- byte 0: Function Code (0x04)
- byte 1: Byte Count
  "Quantity of Registers" × 2
- byte 2~3: Input Registers
  This value shows the status of digital/analog input ports. The register data in the response message are packed as two bytes per register. And the register value represents digital ports as one port per bit from the LSB to the MSB. Status is indicated as 1= ON and 0= OFF. If there are bits mapped to nonexistent ports, they will be padded with zeros (toward the high order end of the register value).

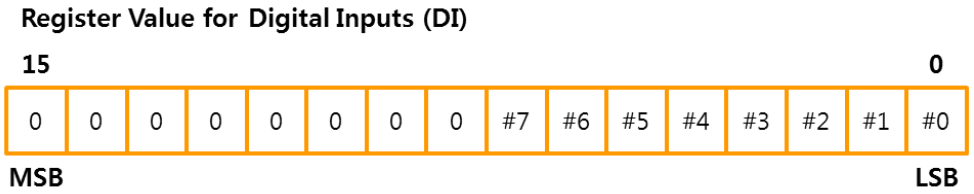


Figure 4-22 Register Value for Digital Input Ports

The register value also represents analog input ports ranging from 0 to n (0 ~ n).

☞ ***n: 4095 (SIG series, 12-bit Resolution), 1023 (CIE series, 10-bit Resolution)***

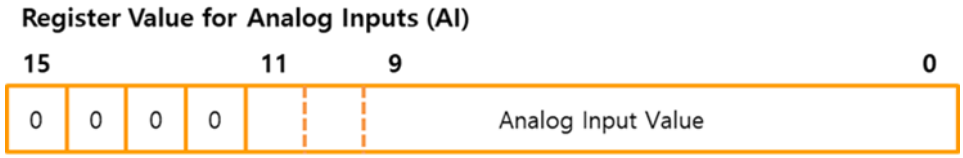


Figure 4-23 Register Value for analog ports

4.4.3 Exception

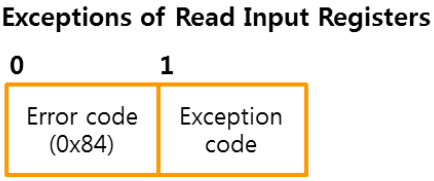


Figure 4-24 Exceptions of Read Input Registers

- byte 0: Error Code
  Function code + 0x80
- byte 1: Exception Code
  Exception code can be 0x01, 0x02 or 0x03.

4.4.4 Examples

The example below shows reading digital input ports.

● Request

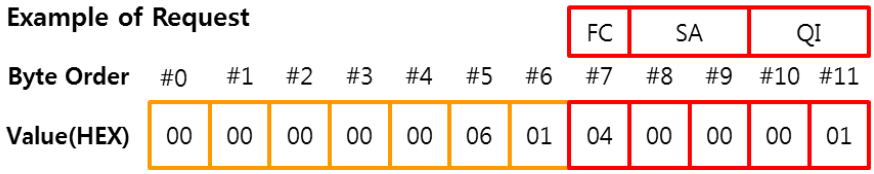


Figure 4-25 Request ADU

byte order	value	description
7	0x04	Function code
8~9	0x0000	Starting address to read is 0
10~11	0x0001	Read 1 register

Table 4-7 Request PDU

● Response

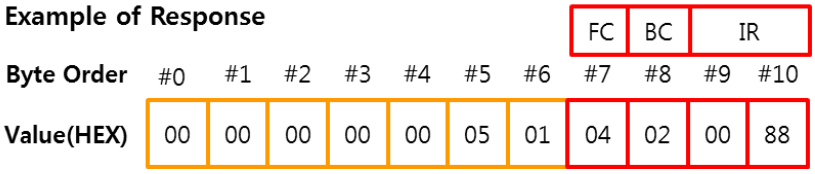


Figure 4-26 Response ADU

byte order	value	description
7	0x04	Function code
8	0x02	2 bytes, 1 register
9~10	0x0088	(1000 1000) #3, 7 ON / #0 ~ 2, 4 ~ 6 OFF

Table 4-8 Response PDU

4.5 Write Single Coil (FC 05)

This function code is used to control one digital output port.

4.5.1 Request / Response

Request / Response of Write Single Coil

Figure 4-27 Request / Response of Write Single Coil

- byte 0: Function Code
  Function code of Write Single Coil is 0x05.
- byte 1~2: Output Address
  This is the address of an output port to control.
- byte 3: Output Value
  A value of 0xFF00 requests the output port to be ON. On the contrary, a value of 0x0000 requests the output port to be OFF. All other values are illegal and will not affect the output port.

☞ ***The response ADU is identical to the request one.***

4.5.2 Exception

Exceptions of Write Single Coil

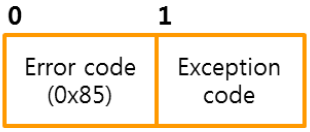


Figure 4-28 Exceptions of Write Single Coil

- byte 0: Error Code
  Function code + 0x80
- byte 1: Exception Code
  Exception code can be 0x01, 0x02, 0x03, 0x04, or 0x06.

4.5.3 Examples

The following is an example of a request to write Output Port #0 ON:

● Request / Response

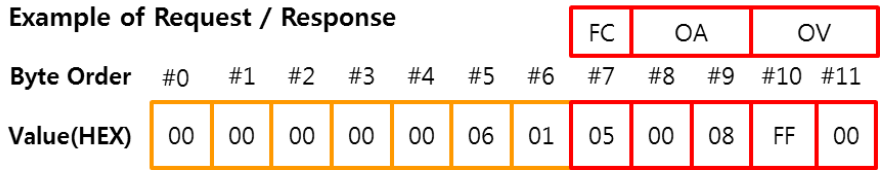


Figure 4-29 Request / Response ADU

byte order	value	description
7	0x05	Function code
8~9	0x0008	Output port address is 8
10~11	0xFF00	Output port ON

Table 4-9 Request / Response PDU

4.6 Write Single Register (FC 06)

This function code is used to control digital output ports ON/OFF.

4.6.1 Request / Response

Request / Response of Write Single Register

Figure 4-30 Request / Response of Write Single Register

- byte 0: Function Code
  Function code of Write Single Register is 0x06.
- byte 1~2: Register Address
  This is the output port address to control.
- byte 3~4: Register Value
  The register data are packed as two bytes per register. And they represent digital output ports as one port per bit from the LSB to the MSB. The value is indicated as 1= ON and 0= OFF. If there are bits mapped to nonexistent ports, they will be ignored except "SIG series" that reply with exception.

Register Value for Digital Outputs (DO)

Figure 4-31 Register Value for the digital output ports

☞ ***The response ADU is identical to the request one.***

4.6.2 Exception

Exceptions of Write Single Register

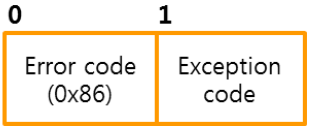


Figure 4-32 Exceptions of Write Single Register

- byte 0: Error Code
  Function code + 0x80
- byte 1: Exception Code
  Exception code can be 0x01, 0x02 or 0x04.

4.6.3 Examples

● Request / Response

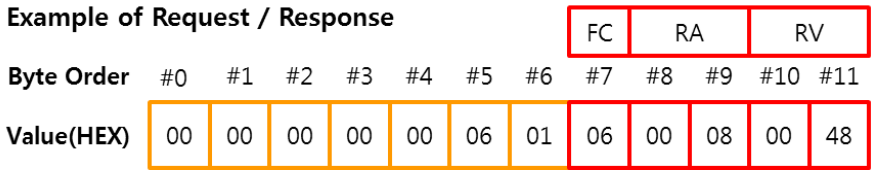


Figure 4-33 Request / Response ADU

byte order	value	description
7	0x06	Function code
8~9	0x0008	Resister address is 8
10~11	0x0048	(0100 1000) #3, 6 ON / #0 ~ 2, 4, 5, 7 OFF

Table 4-10 Request / Response PDU

4.7 Read Exception Status (FC 07)

Note that this function code is NOT relevant to 'Exception Response'. The 'Read Exception Status' is used to check if operation mode of digital output ports is Macro mode.

☞ ***"SIG series" does NOT support FC 07.***

4.7.1 Request

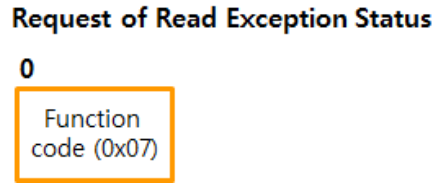


Figure 4-34 Request of Read Exception Status

- byte 0: Function Code
  Function code of Read Exception Status is 0x07.

4.7.2 Response

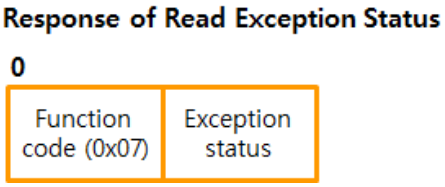


Figure 4-35 Response of Read Exception Status

- byte 0: Function Code (0x07)
- byte 1: Exception Status
  This value represents the status of output ports operating mode from the LSB to the MSB. '1' indicates the port is operated as Macro mode and '0' indicates the port is not. LSB is assigned to the first port. If there are bits mapped to nonexistent ports, they will be padded with zeros (toward the high order end of the value).

4.7.3 Exception

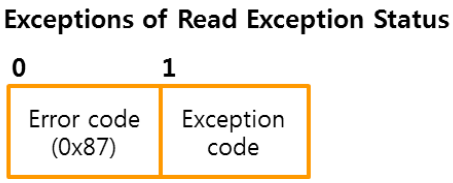


Figure 4-36 exceptions of read exception status

- byte 0: Error Code
  Function code + 0x80
- byte 1: Exception Code
  Exception code can be 0x01.

4.7.4 Examples

● Request

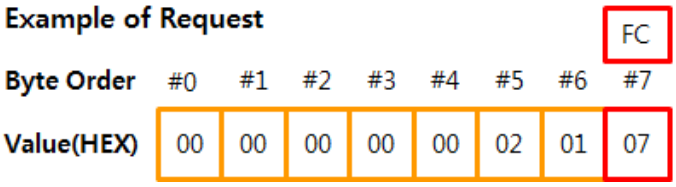


Figure 4-37 Request PDU

byte order	value	description
7	0x07	Function code is 7.

Table 4-11 Request PDU

● Response

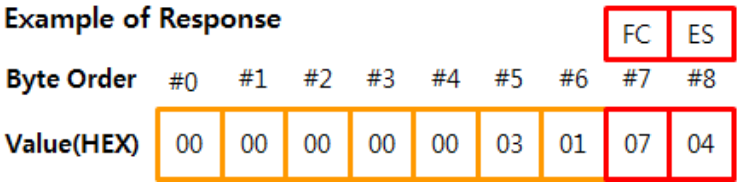


Figure 4-38 Response ADU

byte order	value	description
7	0x07	Function code
8	0x04	(0000 0100) output port #2 is Macro mode

Table 4-12 Response PDU

4.8 Write Multiple Coils (FC 15)

This function code is used to control multiple digital output ports at a single request.

4.8.1 Request

Request of Write Multiple Coils

0	1	3	5	6
Function code (0x0F)	Starting Address	Quantity of Outputs (1 ~ n)	Byte Count (0x01)	Output Value

Figure 4-39 Request of Write Multiple Coils

- byte 0: Function Code
  Function code of Write Multiple Coils is 0x0F.
- byte 1~2: Starting Address
  This is the address of the first output ports to control.
- byte 3~4: Quantity of Outputs
  Request PDU specifies the number of digital output ports ranging from 1 to n (1 ~ n).

☞ ***n: the number of digital output ports for each product***

- byte 5: Byte Count (0x01)
  (Quantity of Outputs + 7) / 8
- byte 6: Output Value
  This value is used to control output ports and packed as one byte per eight output ports from the LSB to the MSB. The bit value is indicated as 1= ON and 0= OFF. If there are bits mapped to nonexistent ports, they will be ignored.

Output Value for Digital Outputs (DO)

MSB							LSB
#7	#6	#5	#4	#3	#2	#1	#0

Figure 4-40 Output Value for the digital output ports

4.8.2 Response

Response of Write Multiple Coils

Figure 4-41 Response of Write Multiple Coils

- byte 0: Function Code (0x0F)
- byte 1~2: Starting Address
- byte 3~4: Quantity of Outputs

4.8.3 Exception

Exceptions of Write Multiple Coils

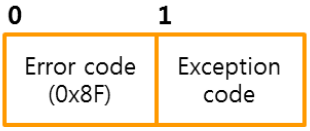


Figure 4-42 Exceptions of Write Multiple Coils

- byte 0: Error Code
  Function code + 0x80
- byte 1: Exception Code
  Exception code can be 0x01, 0x02, 0x03, 0x04 or 0x06.

4.8.4 Examples

The example below shows controlling 4 digital output ports #0 ~ #3.

● Request

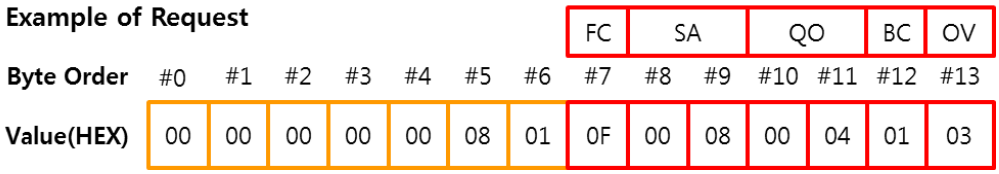


Figure 4-43 Request ADU

byte order	value	description
7	0x0F	Function code
8~9	0x0008	Starting address to control is 8
10~11	0x0004	Control 4 digital output ports
12	0x01	1 byte, 1 ~ 8 digital output ports
13	0x03	(0000 0011) #0, 1 ON / #2, 3 OFF

Table 4-13 Request PDU

● Response

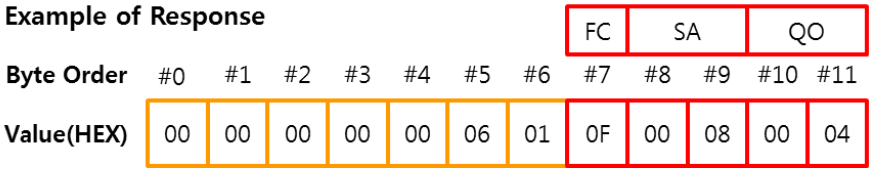


Figure 4-44 Response ADU

byte order	value	description
7	0x0F	Function code
8~9	0x0008	Starting address to control is 8
10~11	0x0004	Control 4 digital output ports

Table 4-14 Response PDU

4.9 Write Multiple Registers (FC 16)

4.9.1 Request

Request of Write Multiple Registers

Figure 4-45 Request of Write Multiple Registers

- byte 0: Function Code
  Function code of Write Multiple Registers is 0x10.
- byte 1~2: Starting Address
  This is the address of the first register to write.
- byte 3~4: Quantity of Registers (0x0001)
  Request PDU specifies the number of registers. The available value for this field is 1.
- byte 5: Byte Count (0x02)
  Quantity of Registers × 2
- byte 6~7: Register Value
  This value is used to control output ports and packed as two bytes per one register from the LSB to the MSB. The bit value is indicated as 1= ON and 0= OFF. If there are bits mapped to nonexistent ports, they will be ignored except “SIG series” that reply with exception.

Register Value for Digital Outputs (DO)

Figure 4-46 Register value for the digital output ports

4.9.2 Response

The response PDU is identical to the request one except the Byte Count and the Register Value.

Response of Write Multiple Registers

Figure 4-47 Response of Write Multiple Registers

- byte 0: Function Code (0x10)
- byte 1~2: Starting Address
- byte 3~4: Quantity of Registers (0x0001)

4.9.3 Exception

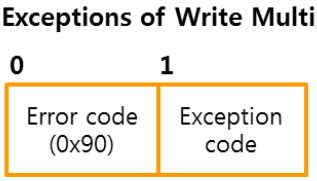


Figure 4-48 Exceptions of Write Multiple Registers

- byte 0: Error Code
  Function code + 0x80
- byte 1: Exception Code
  Exception code can be 0x01, 0x02, 0x03 or 0x04.

4.9.4 Examples

- an example of request

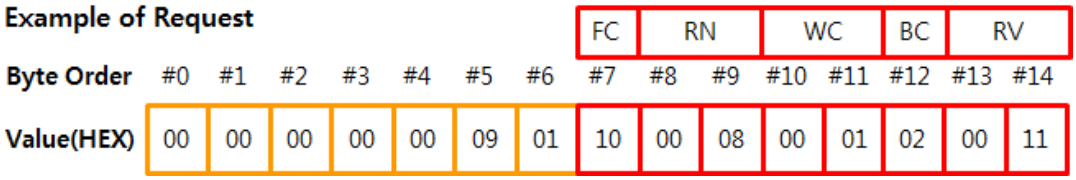


Figure 4-49 Request ADU

byte order	value	description
7	0x10	Function code
8~9	0x0008	Starting address to control is 8
10~11	0x0001	Write 1 register
12	0x02	2 bytes, 1 register
13~14	0x0011	(0001 0001) #0, 4 ON / #1 ~ 3, 5 ~ 7 OFF

Table 4-15 Request PDU

- an example of response

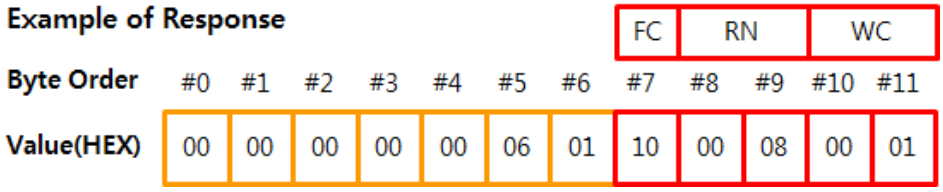


Figure 4-50 Response ADU

byte order	value	description
7	0x03	Function code
8~9	0x0008	Starting address to control is 8
10~11	0x0001	Write 1 register

Table 4-16 Response PDU

4.10 Encapsulated Interface Transport (FC 43)

This function code is designed for tunneling other protocols inside MODBUS. This mechanism is called MEI(MODBUS Encapsulated Interface) Transport. It is categorized to two types of MEI depending on capsulated protocols, one is 13 (0x0D)—CANopen General Reference and the other is 14 (0x0E)—Read Device Identification.

☞ ***EZI-10 does NOT support FC 43.***

4.10.1 Request

Request of Encapsulated Interface Transport

Figure 4-51 Request of Encapsulated Interface Transport

- byte 0: Function Code
  Function Code of Encapsulated Interface Transport is 0x2B.
- byte 1: MEI Type (13 or 14)
  ezTCP only supports 14 (0x0E)—Read Device Identification.

☞ ***MEI: Modbus Encapsulated Interface***

- byte 2~: MEI type specific data (n bytes)
  It is depending on the MEI Type.

4.10.2 Response

Response of Encapsulated Interface Transport

Figure 4-52 Response of Encapsulated Interface Transport

- byte 0: Function Code (0x2B)
- byte 1: MEI Type (0x0D or 0x0E)
- byte 2~: MEI type specific data (n bytes)

4.10.3 Exception

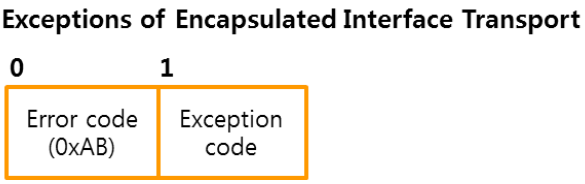


Figure 4-53 Exception of Encapsulated Interface Transport

- byte 0: Error Code
  Function code + 0x80
- byte 1: Exception Code
  Exception code can be 0x01, 0x02, 0x03 or 0x04.

4.10.4 Examples

Refer to 4.11 Read Device Identification (FC 43 / 14).

4.11 Read Device Identification (FC 43 / 14)

This function code is used to confirm the device information called an object. Objects are categorized as below.

- Basic Device Identification
- Regular Device Identification
- Extended Device Identification
  There are 8 objects are defined for the extended device identification.

Type	Object ID	Object Name	Data Type	M/O
Basic	0x00	VendorName	ASCII String	**Mandatory**
	0x01	ProductCode	ASCII String	**Mandatory**
	0x02	MajorMinorRevision	ASCII String	**Mandatory**
Regular	0x03	VendorUrl	ASCII String	Optional
	0x04	ProductName	ASCII String	Optional
	0x05	ModelName	ASCII String	Optional
	0x06	UserApplicationName	ASCII String	Optional
	0x07~0x7F	Reserved		Optional
Extended	0x80	Comment	Binary	Optional
	0x81	MAC Address	ASCII String	Optional
	0x82	Macro Mode	Binary	Optional
	0x83	TCP Session ID	Binary	Optional
	0x84	1-bit ADC Mode	Binary	Optional
	0x85	DO Pulse Mode	Binary	Optional
	0xA0 + n	Input Comments	Binary	Optional
	0xB0 + n	Output Comments	Binary	Optional

Table 4-17 Object ID

☞ ***n: the number of digital input and output ports for each product***

4.11.1 Request

Request of Read Device Identification

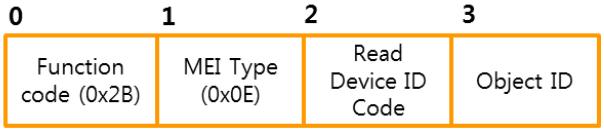


Figure 4-54 Request of Read Device Identification

- byte 0: Function Code (0x2B)
- byte 1: MEI Type (0x0E – Read Device Identification)
- byte 2: Read Device ID Code (0x01 / 0x02 / 0x03 / 0x04)
  Read Device ID Code allows to define four access types as the below:
  0x01: request to get the basic device identification (stream access)
  0x02: request to get the regular device identification (stream access)
  0x03: request to get the extended device identification (stream access)
  0x04: request to get the one specific identification object (individual access)

  In case of a response that does not fit into a single response, several transactions (request/ response) must be done. In case of the stream access, I/O gateway needs several transactions to Extended Device Identification while it needs a single transaction to Basic/Regular Device Identification.

- byte 3: Object ID
  The Object ID field gives the identification of the first object to obtain.
  For the first transaction in case of stream access – 0x00
  For the following transaction in case of stream access – the value in its previous response
  In case of the individual access – the identification of the object to obtain

  If the Object ID does not match any known object, the slave responds as if object 0 were requested. It means restart at the beginning.

4.11.2 Response

Response of Read Device Identification

0	1	2	3	4	5	6	7	8	9
Function code (0x2B)	MEI Type (0x0E)	Read Device ID Code	Conformity Level	More Follows	Next Object ID	Number of Objects	Object ID	Object Length	Object Value

Figure 4-55 Response of Read Device Identification

- byte 0: Function Code (0x2B)
- byte 1: MEI Type (0x0E – Read Device Identification
- byte 2: Read Device ID Code (0x01, 0x02, 0x03, 0x04) – identical to the request
- byte 3: Conformity Level
  Conformity Level represents supported access and object type.
  Conformity Level of I/O gateway is 0x83.
  0x01: Basic identification (stream access only)
  0x02: Regular identification (stream access only)
  0x03: Extended identification (stream access only)
  0x81: Basic identification (stream/individual access)
  0x82: Regular identification (stream/individual access)
  0x83: Extended identification (stream/individual access)
- byte 4: More Follows
  In case of the stream access,
  0x00: no more Objects are available
  0xFF: other Objects are available, and more MODBUS transactions are required.
  In case of the individual access, this value is fixed to 0x00.
- byte 5: Next Object ID
  When More Follows is 0xFF: the next Object to be asked by the following transaction
  When More Follows is 0x00: fixed to 0x00
- byte 6: Number of Objects
  Number of objects returned in the PDU (stream access)
  0x01 (individual access)
- byte 7: Object ID
  The first Object returned in the PDU (stream access)
  The requested Object (individual access)
- byte 8: Object Length
  Length of the first Object in byte
- byte 9~: Object Value
  Value of the first Object. If there are several objects in the PDU, Object ID/Object Length/Object Value are repeated in sequence.

4.11.3 Exception

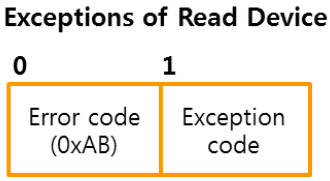


Figure 4-56 Exception of Read Device Identification

- byte 0: Error Code
  Function Code + 0x80
- byte 1: Exception Code
  Exception code can be 0x01, 0x02, 0x03 or 0x04.

4.11.4 Example – TCP Session ID

● Request

Example of Request

Byte Order	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Value(HEX)	00	00	00	00	00	05	01	2B	0E	04	83

Figure 4-57 Request ADU

byte order	Value	Meaning
7	0x2B	Function Code 43
8	0x0E	Read Device Identification
9	0x04	Individual access to get the one specific identification
10	0x83	Request “TCP Session ID”

Table 4-18 Request PDU

● Response

Example of Response

								FC	MT	DI	CL	MF	NI	OC	OI	OL	OD
Byte Order	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15	#16
Value(HEX)	00	00	00	00	00	09	01	2B	0E	04	83	00	00	01	83	01	02

Figure 4-58 Response ADU

byte order	Value	Meaning
7	0x2B	Function Code 43
8	0x0E	Read Device Identification
9	0x04	Individual access to get the one specific identification
10	0x83	Supports both stream and individual access
11	0x00	No more transaction
12	0x00	Last transaction
13	0x01	1 object in this PDU
14	0x83	TCP Session ID
15	0x01	1 byte
16	0x02	TCP session id is 0x02

Table 4-19 Response PDU

4.11.5 Example – Basic Device Identifcation

● Request

Name	Value	Meaning
Function Code	0x2B	Function Code 43
MEI Type	0x0E	Read Device Identification
Read Device ID Code	0x01	Stream access to Basic Identification
Object ID	0x00	VendorName (Beginning of the stream access)

Table 4-20 Request PDU

● Response

Name	Value	Meaning
Function Code	0x2B	Function Code 43
MEI Type	0x0E	Read Device Identification
Read Device ID Code	0x01	Stream access to Basic Identification
Conformity Level	0x83	Supports both stream and individual access
More Follows	0x00	No more transaction
Next Object ID	0x00	Last transaction
Number of Objects	0x03	Three objects in this PDU
Object ID	0x00	VendorName
Object Length	0x0E	14 bytes
Object Data	“Sollae Systems”	
Object ID	0x01	ProductCode
Object Length	0x02	2 bytes
Object Data	“20”	Product Code of CIE-H10A
Object ID	0x02	MajorMinorRevision
Object Length	0x05	5 bytes
Object Data	“V2.2B”	Firmware version 2.2B

Table 4-21 Response PDU

4.11.6 Example – Extended Device Identification

● Request 1

Name	Value	Meaning
Function Code	0x2B	Function Code 43
MEI Type	0x0E	Read Device Identification
Read Device ID Code	0x03	Stream access to Extended Identification
Object ID	0x00	Comment (Beginning of the stream access)

Table 4-22 Request PDU 1

● Response 1

Name	Value	Meaning
Function Code	0x2B	Function Code 43
MEI Type	0x0E	Read Device Identification
Read Device ID Code	0x03	Stream access to Extended Identification
Conformity Level	0x83	Supports both stream and individual access for all Identification
More Follows	0xFF	Further transactions
Next Object ID	0xA0	Input Comments
Number of Objects	0x03	Three objects in this PDU
Object ID	0x80	Comment
Object Length	0x00	0 byte
Object ID	0x81	MAC Address
Object Length	0x11	17 bytes
Object Data	“00:30:F9:00:00:01”	MAC address
Object ID	0x82	Macro Mode
Object Length	0x01	1 byte
Object Data	0x81	#0, 7 Macro mode ON

Table 4-23 Response PDU 1

● Request 2

Name	Value	Meaning
Function Code	0x2B	Function Code 43
MEI Type	0x0E	Read Device Identification
Read Device ID Code	0x03	Stream access to Extended Identification
Object ID	0xA0	Input Comments

Table 4-24 Request PDU 2

● Response 2

Name	Value	Meaning
Function Code	0x2B	Function Code 43
MEI Type	0x0E	Read Device Identification
Read Device ID Code	0x03	Stream access to Extended Identification
Conformity Level	0x83	Supports both stream and individual access for all Identification
More Follows	0xFF	Further transactions
Next Object ID	0xB0	Output Comments
Number of Objects	0x08	Eight objects in this PDU
Object ID	0xA0	Input port #0 Comment
Object Length	0x03	3 bytes
Object Data	“DI0"	Default value
Object ID	0xA1	Input port #1 Comment
Object Length	0x03	3 bytes
Object Data	“DI1"	Default value
Object ID	0xA2	Input port #2 Comment
Object Length	0x03	3 bytes
Object Data	“DI2"	Default value
Object ID	0xA3	Input port #3 Comment
Object Length	0x03	3 bytes
Object Data	“DI3"	Default value
Object ID	0xA4	Input port #4 Comment
Object Length	0x03	3 bytes
Object Data	“DI4"	Default value
Object ID	0xA5	Input port #5 Comment
Object Length	0x03	3 bytes
Object Data	“DI5"	Default value
Object ID	0xA6	Input port #6 Comment
Object Length	0x03	3 bytes
Object Data	“DI6"	Default value
Object ID	0xA7	Input port #7 Comment
Object Length	0x03	3 bytes
Object Data	“DI7"	Default value

Table 4-25 Response PDU 2

● Request 3

Name	Value	Meaning
Function Code	0x2B	Function Code 43
MEI Type	0x0E	Read Device Identification
Read Device ID Code	0x03	Stream access to Extended Identification
Object ID	0xB0	Output Comments

Table 4-26 Request 3

● Response 3

Name	Value	Meaning
Function Code	0x2B	Function Code 43
MEI Type	0x0E	Read Device Identification
Read Device ID Code	0x03	Stream access to Extended Identification
Conformity Level	0x83	Supports both stream and individual access for all Identification
More Follows	0x00	No more transaction
Next Object ID	0x00	Last transaction
Number of Objects	0x08	Eight objects in this PDU
Object ID	0xB0	Output port #0 Comment
Object Length	0x03	3 bytes
Object Data	“DO0"	Default value
Object ID	0xB1	Output port #1 Comment
Object Length	0x03	3 bytes
Object Data	“DO1"	Default value
Object ID	0xB2	Output port #2 Comment
Object Length	0x03	3 bytes
Object Data	“DO2"	Default value
Object ID	0xB3	Output port #3 Comment
Object Length	0x03	3 bytes
Object Data	“DO3"	Default value
Object ID	0xB4	Output port #4 Comment
Object Length	0x03	3 bytes
Object Data	“DO4"	Default value
Object ID	0xB5	Output port #5 Comment
Object Length	0x03	3 bytes
Object Data	“DO5"	Default value
Object ID	0xB6	Output port #6 Comment
Object Length	0x03	3 bytes
Object Data	“DO6"	Default value
Object ID	0xB7	Output port #7 Comment
Object Length	0x03	3 bytes
Object Data	“DO7"	Default value

Table 4-27 Response PDU 3

5 User-Defined Function Codes

5.1 Write Pulse (FC 105)

This function is used to make a pulse operation to the digital output port. Note that it is not specified in the standard of Modbus/TCP.

☞ ***"SIG series" does NOT support FC 105.***

5.1.1 Request / Response

Request / Response of Write Pulse

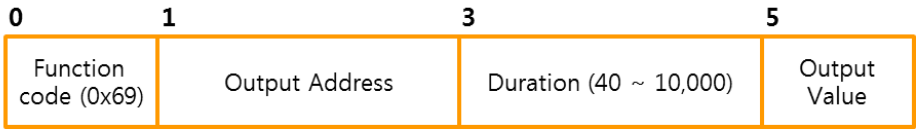


Figure 5-1 Request / Response of Write Pulse

- byte 0: Function Code
  Function code of Write Pulse is 0x69.
- byte 1~2: Output Address
  This is the address of an output port to control.
- byte 3~4: Duration (unit: millisecond)
  The unit is millisecond. The available range of this value is from 40 to 10000 (0x0028 ~ 0x2710).
- byte 5: Output Value
  A value of 0xFF requests the output port to make a high (ON) pulse operation. On the contrary, a value of 0x00 requests the output port to make a low (OFF) pulse. All other values are illegal and will not affect the output port. If this value is the same to the current output port status, the exception response will be returned with exception code 0x04.

☞ ***The response ADU is identical to the request one.***

5.1.2 Exception

Exceptions of Write Pulse

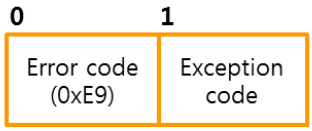


Figure 5-2 Exceptions of Write Pulse

- byte 0: Error Code (0xE9)
  Function code + 0x80
- byte 1: Exception Code
  Exception code can be 0x01, 0x02, 0x03 or 0x04.

5.1.3 Examples

● Request / Response

Example of Request / Response

								FC	OA		D		OV
Byte Order	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
Value(HEX)	00	00	00	00	00	07	01	69	00	08	03	E8	FF

Figure 5-3 Request / Response ADU

byte order	value	Description
7	0x69	Function code is 105.
8~9	0x0008	Output port address is 8
10~11	0x03E8	Duration is 1 second. (1000ms = 0x03E8)
12	0xFF	Data value is 0xFF (ON for the duration)

Table 5-1 Request / Response PDU

☞ ***If a port is either operating in the Macro mode or being controlled by the previous FC 105, the request is not accepted.***

5.2 Send Notify (FC 108)

It is used to notify the change of the port status. The response frame is only defined.

☞ ***"CIE series" and EZI-10 does NOT support FC 108.***

5.2.1 Response

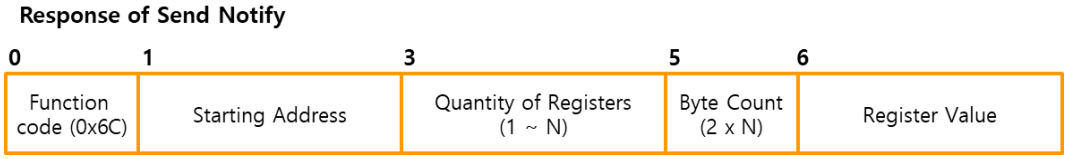


Figure 5-4 Response of Send Notify

- byte 0: Function Code
  Function code of Send Notify is 0x6C.

- byte 1~2: Starting Address
  This is the address of the first register to notify. The below show the address for each port.

Name	Address	Description
DI Counter	0x00A0	Counter Value of Digital Input Port
DI Value	0x00F0	Status of Digital Input Port
1-bit ADC	0x00FA	Bit conversion value for Analog Input Port
DO Value	0x0140	Status of Digital Output Port

Table 5-2 Start Address of each port

- byte 3~4: Quantity of Registers (1 ~ N)
  Request PDU specifies the number of registers. The available value for this field is 1 ~ N.

- byte 5: Byte Count (0x02)
  Quantity of Registers × 2

- byte 6~(2 x N + 5): Register Value

5.2.2 Examples

The below shows response frame to notify the change of the digital input port status.

● Response

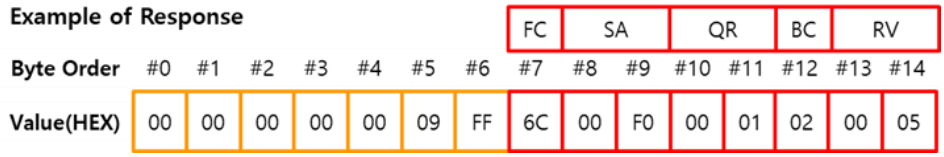


Figure 5-5 Response ADU

byte order	value	Description
7	0x6C	Function code
8~9	0x00F0	Starting address to control is 8
10~11	0x0001	Write 1 register
12	0x02	2 bytes, 1 register
13~14	0x0005	(0000 0101) #0, 2 ON / #1, 3 OFF

Table 5-3 Response PDU

6 Additional Instructions

6.1 Exception Codes

Exception Code	Name	Description
0x01	Illegal Function	Function code error
0x02	Illegal Data Address	Starting address error
0x03	Illegal Data Value	Data value error
0x04	Server Device Failure	Request action is failed (CIE series, EZI-10)
0x06	Slave Device Busy	Request action is failed (SIG series)

Table 6-1 Exception Codes

6.2 CIE-M10A ADC values

6.2.1 Request

The analog input (ADC) ports of CIE-M10A can be read by FC 04 (Read Input Register). This ADC port is mapped to [Input Port Base Address] + 4. For example, if the [Input Port Base Address] is set to zero, the register address for the ADC port is 4(0x0004). See the below example below.

- an example of request

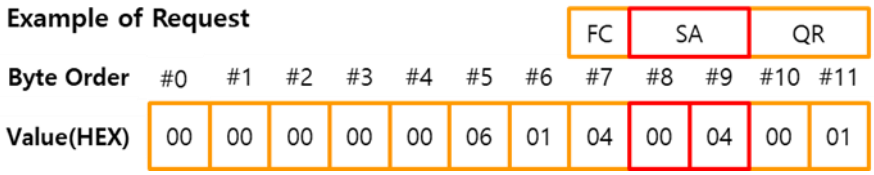


Figure 6-1 an example of request

6.2.2 Response

You can read the value of ADC port of CIE-M10A from the last two bytes of a command response of read input registers.

- an example of response

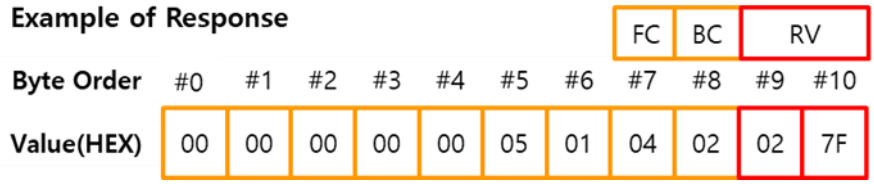


Figure 6-2 an example of response

In the above example, ADC value is 0x02 0x7f in hexadecimal. (639 in decimal)

6.3 Sample Codes

We have been offering sample codes for users who will make Modbus/TCP application program to use our I/O controllers. The codes can be downloaded from the [DOWNLOAD] >> [Download] page on our web site. (https://www.eztcp.com/en/download/pds.php)

☞ ***Click the [Search] button after select [Sample Code] from the [DOWNLOAD] category.***

6.3.1 Provided version

- C++ (Visual Studio 2008)

7 Serialized Modbus/TCP

Serialized Modbus/TCP mode is for only CIE-M10, CIE-H10 and CIE-H14. The other communication modes are all disabled when using this mode. The RS232 port is used for this.

7.1 Features

- Sending and Receiving the Modbus/TCP data through the RS232
- Controlling digital I/O ports via a serial port
- No connection processes
  In this mode, devices (or terminals) sends and receives data without any connection processes. Use the hardware flow control (RTS/CTS) to prevent data loss.

7.2 Using

7.2.1 Configuration

- Serialized Modbus/TCP configuration

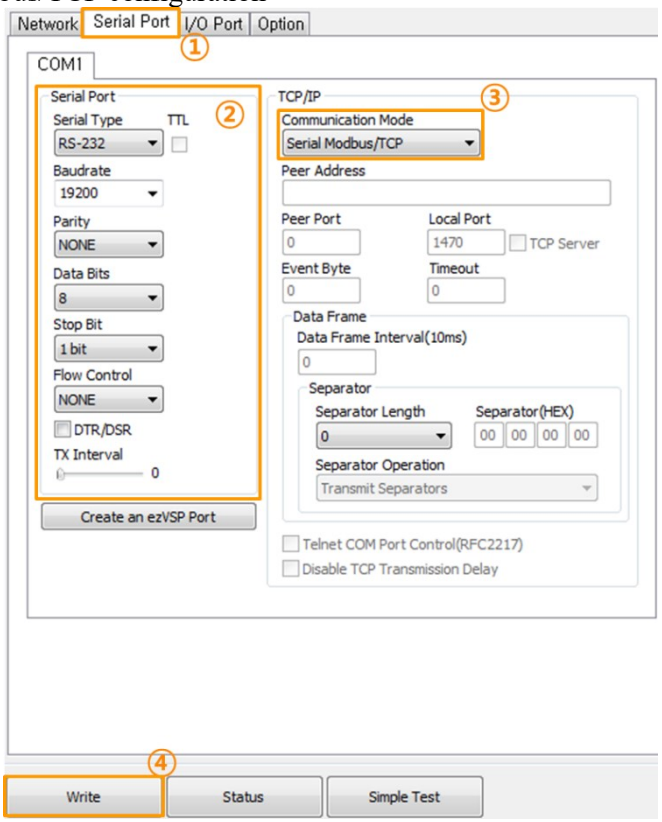


Figure 7-1 configuration of serialized Modbus/TCP

① Move to the [Serial Port] tab.

② Set and check parameters for the serial port.

③ Select [Serialized Modbus/TCP] on the [Communication Mode].

④ Press the [Write] button for saving.

7.3 Trial Run

7.3.1 Preperations for Communication

For testing the serialized Modbus/TCP mode, design the connection as follows:

☞ ***Connection via an Ethernet cable is not required.***

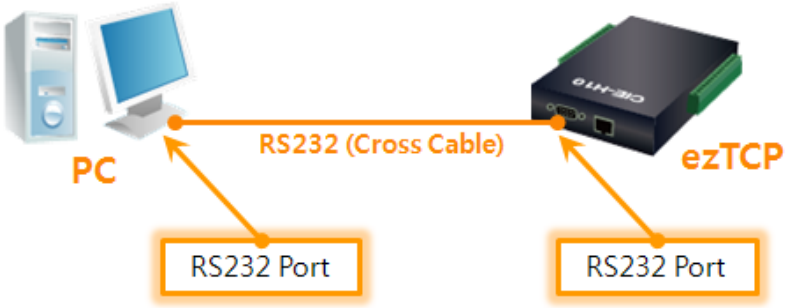


Figure 7-2 connection diagram

Then, for the test, the default value of Modbus/TCP setting has to be kept as follows:

Name	Default Value
Modbus/TCP	Checked
Notify Input Port Change	Unchecked
Master / Slave	Slave
Poll Interval	1,000
Unit ID	1
Input Port	0
Output Port	8

Table 7-1 default values for the parameters

7.3.2 Sending an Example data

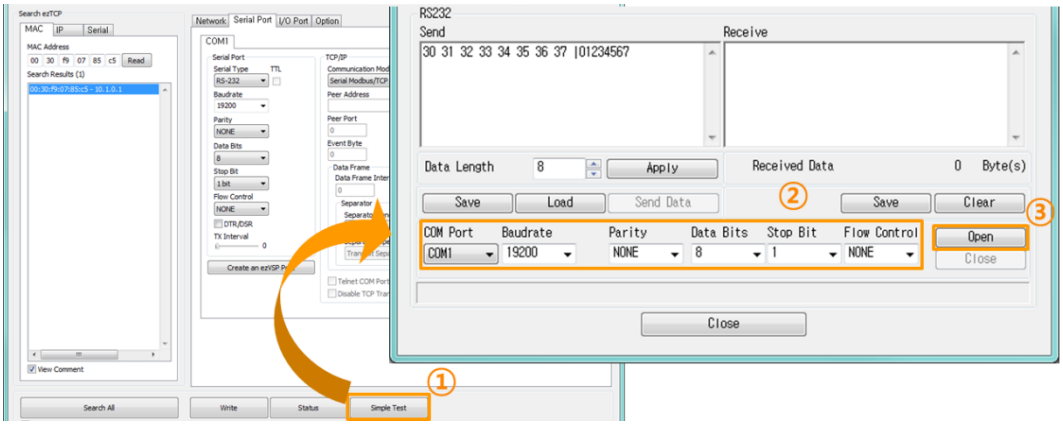


Figure 7-3 Run Simple Test Program

① Click the [Simple Test] button.

② Select and check all the parameters related with the serial port.

③ Press the [Open] button.

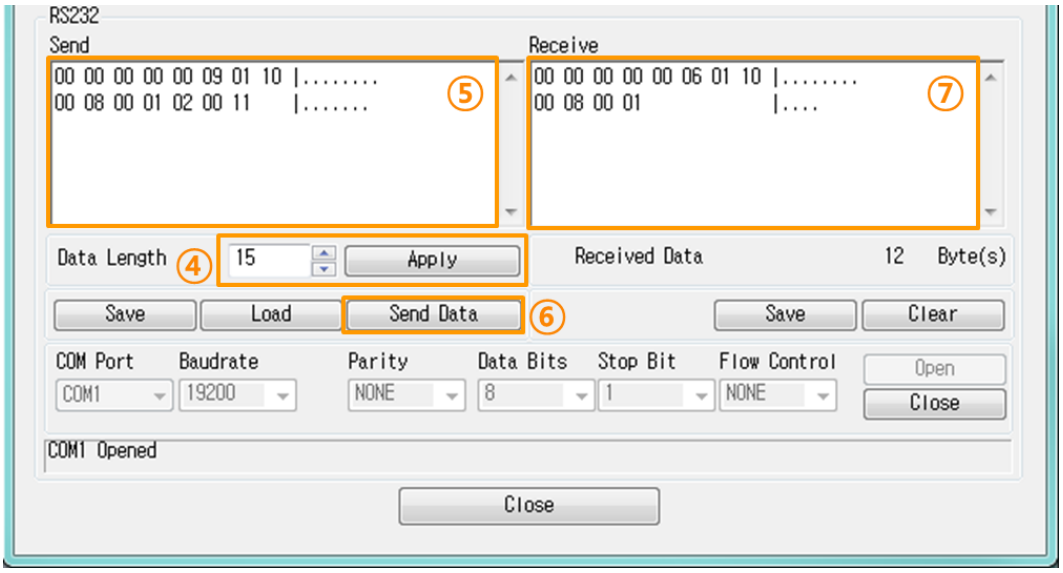


Figure 7-4 Send an example data

④ Click the [Apply] button after setting the [Data Length] to 15 bytes.

⑤ Input the data which is used in the example data of write multiple registers.

⑥ Press the [Send Data] button.

⑦ Check the real output ports of ezTCP if the response data in the [Receive] box is same with the above figure.

☞ ***The data sent on the step ⑤ means turning on the output port #0 and #4 (Write Multiple Registers). The slave should respond to the data appeared on the step ⑦.***

8 Precautions

- This document describes the Modbus/TCP protocol that I/O gateway supports based on the standard MODBUS protocol document – “MODBUS Application Protocol Specification (v1.1b3)” and “MODBUS Messaging Implementation Guide (v1.0b)”.
- The information presented in this document is subject to change without prior notice.
- It is assumed all contents in this document are accurate and reliable, but it does NOT mean we guarantee it.
- For the detailed information, refer to standard documents of MODBUS protocol.

9 Revision History

Date	Version	Comments	Author
2010.03.08.	1.0	○ Initial Release	Roy LEE
2010.07.20.	1.1	○ Name of the document has been changed ○ Contents of the Modbus/TCP document has been included ○ Contents about the EZI-10 has been added	Roy LEE
2010.11.23.	1.2	○ Descriptions about querying/response of ADC values have been added ○ Date item on the front page has been removed.	Roy LEE
2011.06.24.	1.3	○ Contents about new functions have been added. (FC 1, 2, 4, 5, 6, 7, 15, 105) ○ Most of figures have been updated. ○ Figures about configuration tools have been updated. ○ Document structure has been redesigned. ○ Title of document has been changed.	Roy LEE
2014.04.30.	1.4	○ Add description about MODBUS data and addressing ○ Contents about new functions have been added. (FC 43, FC 43 / 14) ○ Most of figures and description have been updated. ○ Figures about configuration tools have been updated. ○ Document structure has been redesigned. (Delete function code class categorization and etc.)	Andy Lee
2015.02.13.	1.5	○ Typo correction	Andy Lee
2017.08.04.	1.6	○ Fix MODBUS addressing description errors ○ Delete sample code descriptions and fix link errors	Andy Lee
2021.09.08.	1.7	○ Contents about the SIG series has been added ○ Add examples for FC 43 (TCP Session ID) ○ Contents about new functions have been added (FC 108) ○ Typo correction	Andy Lee