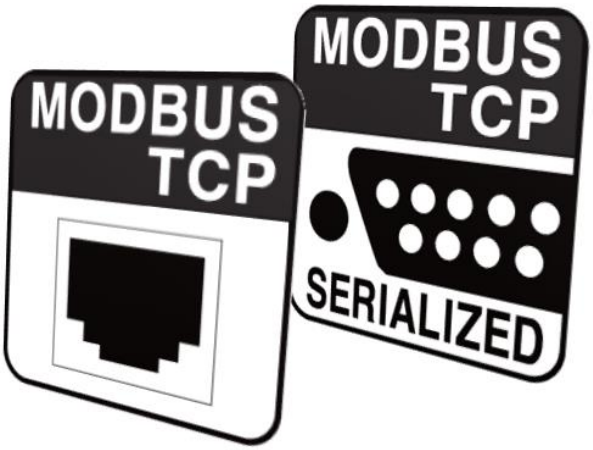
ezTCP 技术资料

Modbus/TCP of ezTCP

Version 1.4



☞ *注意: 此文件中的内容为了升级，没有预告的情况下也可进行变更。*

Sollae Systems Co., Ltd.

http://www.ezTCP.com

1 目录

2 概要

Modbus (Modbus)是在全世界PLC(Programmable Logic Controller)等广为使用的串行通信程序。Modbus的版本中**Modbus/TCP**在TCP/IP网络上执行，当前应用在ezTCP远程数据输入/输出控制产品中。

Modbus/TCP通过产品以太网端口进行通信。但是为了支持串口端口的设备监视及控制的用户，ezTCP支持"串口Modbus/TCP"模式。

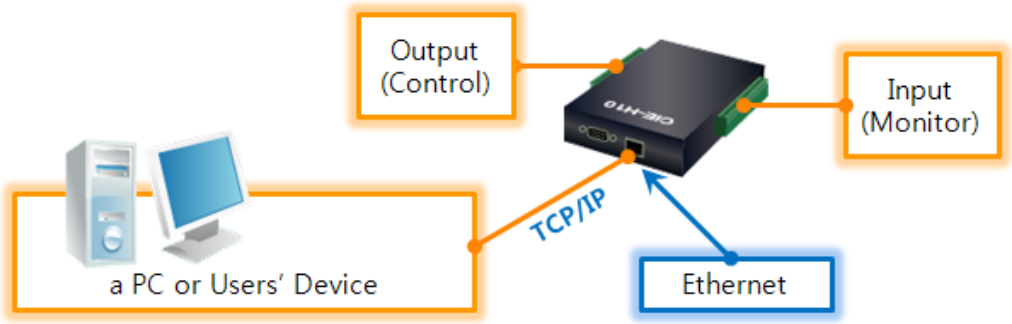


图 2-1 **Modbus/TCP** 使用构成 例

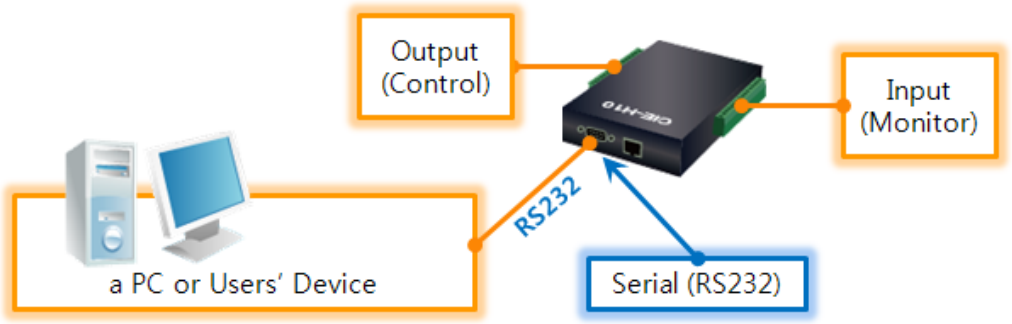


图 2-2 串行 **Modbus/TCP**使用构成 例

串口Modbus/TCP不是指标准的**Modbus**，是将之前提到的利用Modbus/TCP协议数据通过串行(RS232)数据按串行方式发送/接收的模式。

2.1 适用产品

- CIE-xxx 系列 – CIE-H12, CIE-H14, CIE-H10, CIE-M10
- EZI-10

2.2 用语

2.2.1 ezTCP 端口种类

ezTCP上有如下3种端口。

- 数字输入端口
- 模拟输入端口
- 数字输出端口

2.2.2 ezTCP 端口定义

产品输入/输出端口的第一个端口通过数字0开始，以后每增加1并通过'#'一起标示。如 CIE-H10分别有8个数字输入端口和数字输出端口，第一个端口标示为"#0"第二个端口为"#1"最后第八个端口标示为'#7'。

2.2.3 MSB / LSB

数字数据按字节单位显示时根据字节的位置区分的方法。

图 2-3 MSB与LSB

- MSB(Most Significant Bit) – 对应最大值的字节,最左端字节
- LSB(Least Significant Bit) – 对应最小值的字节,最右端字节

3 协议概要

3.1 MODBUS

3.1.1 特点

- 应用层协议 (OSI 7 阶层)
- PDU (Protocol Data Unit)
  不受下一阶层影响独立的数据结构

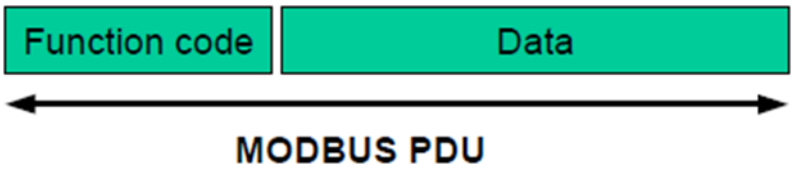


图 3-1 MODBUS PDU

- ADU (Application Data Unit)
  依据下一阶层最终被决定的数据结构

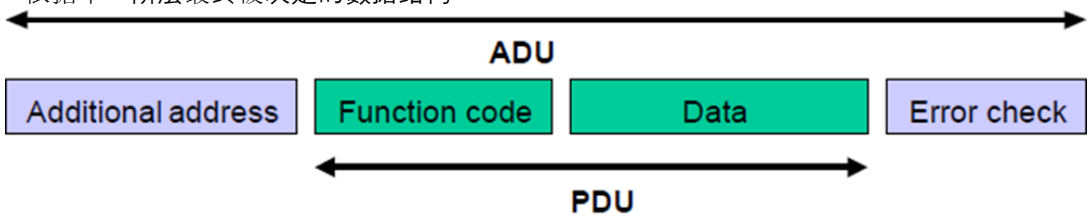


图 3-2 一般的 MODBUS框架结构

- 客户端/服务器 模型

图 3-3 MODBUS 客户端/服务器 模型

3.1.2 数据编码(Data Encoding)

MODBUS使用大端格式(Big-endian)。大端格式是指传送占一个字节以上内存值时，先传送最大值的方式。举例来讲16字节数据0x1234按0x12，0x34的顺序传送。

3.1.3 MODBUS 数据模块

MODBUS根据输入与输出和字节单位基准共分为4种形态。

种类	连接状态	读取/写	说明
Discrete Inputs	字节	读取	设备端口状态
Coils	字节	读取/写	可依据应用程序进行变更
Input Registers	16字节文字	读取	设备端口状态
Holding Registers	16字节文字	读取/写	可依据程序变更功能

表 3-1 Modbus数据模块

3.1.4 ezTCP 的 Modbus 数据模块结构

根据MODBUS数据模块闪存构造，根据构成结构的设备制造商各自不同。如根据数据块各自指定使用4个数据块，或将所有数据连接到一个数据块。

- ezTCP使用一个数据块

  ezTCP有一个数据块闪存构造，依据相同数据可使用多个形态的函数。这是指ezTCP的 MODBUS数据（输入/输出 端口信息）支持字节单位连接与16字节词连接。

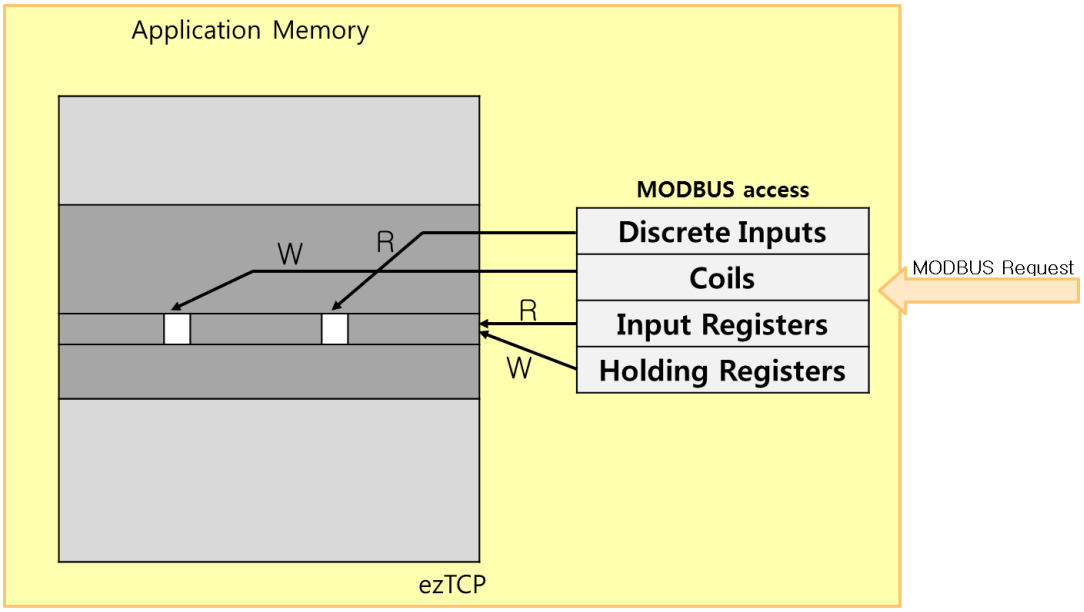


图 3-4 ezTCP的Modbus数据模块闪存构造

3.1.5 使用 MODBUS 地址

MODBUS标准定义可使用数据地址为0 ~ 65535(0xFFFF)，此地址在实际Modbus设备中根据数据种类各个对应1 ~ n(最大 65536)。大部分的Modbus/TCP主程序(以下 HMI – Human Machine Interface)为了分辨各个地址使用地址自首。而且根据4个Modbus数据形态各自申请后定义了可使用在被邀请提问的函数。

种类	区分字首	函数	ezTCP
Coils	0X reference	01, 05, 15	输出端口 (数字)
Discrete Inputs	1X reference	02	输入端口 (数字)
Input Registers	3X reference	04	输入端口 (数字/模拟)
Holding Registers	4X reference	03, 06, 16	输入/输出端口 (数字/模拟)

表 3-2 Modbus数据地址及ezTCP的构造

- (例) 数字输入端口地址 – 产品设定值为'0' (出厂基本值) 时

设定 值	种类	函数	在HMI的地址
0	Coils	01, 05, 15	不可使用
	Discrete Inputs	02	10001
	Input Registers	04	30001
	Holding Registers	03, 06, 16	40001

表 3-3 输入端口地址表示 例

- (例)数字输出端口地址 – 产品设定值为'8' (工厂基本参数)时

设定值	种类	函数	在HMI的地址
8	Coils	01, 05, 15	00009
	Discrete Inputs	02	不可使用
	Input Registers	04	不可使用
	Holding Registers	03, 06, 16	40009

表 3-4 标示输出端口地址 例

3.2 Modbus/TCP

3.2.1 特点

- MODBUS协议的TCP/IP用版本
- 先行TCP连接过程
  就像自名称可知晓的Modbus/TCP使用TCP。 基本端口号是TCP 502号。

3.2.2 在 ezTCP 通信模块

标准Modbus/TCP定义客户端/服务器模块。客户端发送邀请(Request)服务器对其发送应答(Response)的结构。Modbus/TCP客户端同时是TCP客户端，Modbus/TCP服务器是TCP服务器。

但是ezTCP不是客户端/服务器模块而是按Slave/Master工作。TCP连接的开始是根据是否为Slave/Master和独立的用户设定决定。

- Master
  除TCP连接的是否开始, 在标准指的是客户端。
- Slave
  除TCP连接的是否开始, 在标准指服务器。

☞ ***按Slave工作的ezTCP根据[输入端口变更提醒]设定，没有Master邀请(Request)时可发送(Response)。***

3.2.3 Modbus/TCP 框架

下面图显示的是Modbus/TCP框架构造。

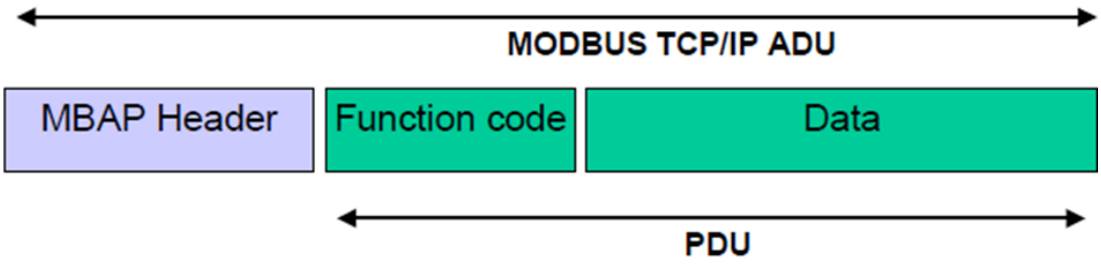


图 3-5 Modbus/TCP框架构造

3.2.4 MBAP 部首

MBAP是Modbus Application Protocol缩写其自首如下分为4个项目。

项目	长度	说明
Transaction Identifier	2 bytes	邀请/应答 区分为一双工作업
Protocol Identifier	2 bytes	0 = MODBUS protocol
Length	2 bytes	框架剩的长度

Unit Identifier	1 byte	通过不是TCP/IP的其他通信线路(例: 串口)连接的Slave区分

表 3-5 MBAP字首

全部Modbus/TCP框架按如下构造组成了。

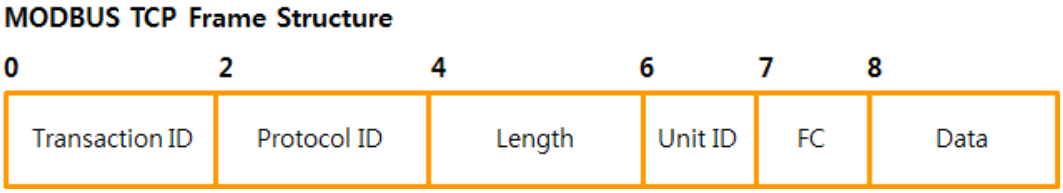


图 3-6 Modbus/TCP 框架构造

- byte 0 ~ 1: 事务标识器 (Transaction Identifier)
  是为了区别提问及回答而使用的号码，依据Master被设定。ezTCP的Master动作模式最初由0x0000开始每个命令增加1使用值，Slave直接复制Master的提问值后使用。

☞ ***HEX: 在此文件的HEX,或是标示0xABCD是指16进制。***

- byte 2 ~ 3: 协议帐号(Protocol Identifier)
  显示协议ID，固定值为 0x0000。
- byte 4 ~ 5: 长度 (Length)
  显示自Length字段到相应框架的长度。(单位: Byte)
- byte 6: 单位帐户 (Unit Identifier)
- byte 7: 函数代码 (Function Code)
- byte 8 ~: 根据函数的数据 (Data)

☞ ***为了与使用通信线路的串行Modbus兼容，Modbus/TCP一个框架的最大值限制为260字节。***

3.3 函数 (Function Codes)

3.3.1 函数种类

函数定义Modbus在协议中实际提供的服务。函数占Modbus框架中1字节的空间，可使用的领域是1 ~ 255。其中实际使用1 ~ 127之间的值，128 ~ 255间的值是在发生错误等例外应答时使用。函数代码0不可用。部分函数为了支持多个动作服务器函数代码可能追加使用。

函数种类根据目的大为分3个部分。

- Public Function Codes
  定义在标准文件的函数。
  1 ~ 64, 73 ~ 99, 111 ~ 127

- User-Defined Function Codes
  在标准文件没有定义，在设备制造商直接构成的相关功能与相关的函数。
  65 ~ 72, 100 ~ 110

- Reserved Function Codes
  Public领域中使用在部分制造商的旧型设备中使用的函数，无法公开使用的函数。
  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

表 3-6 函数代码种类

3.3.2 Public Function Codes

下面是ezTCP支持的Public函数代码。

种类	连接	端口	名称	函数代码	
				代码	服务器代码
数据	字节	输入端口(数字)	Read Discrete Inputs	02 (0x02)	
		输出端口(数字)	Read Coils	01 (0x01)	
			Write Single Coil	05 (0x05)	
			Write Multiple Coils	15 (0x0F)	
	16字节	输入端口	Read Input Registers	04 (0x04)	
		输入端口输出端口	Read Holding Registers	03 (0x03)	
		输出端口	Write Single Register	06 (0x06)	
		输入端口	Write Multiple Registers	16 (0x10)	
诊断			Read Exception Status	07 (0x07)	
			Read Device Identification	43 (0x2B)	14 (0x0E)
其他			Encapsulated Interface Transport	43 (0x2B)	

表 3-7 ezTCP支援的Public函数代码

☞ *根据产品种类与固件版本支援的函数代码会不同。*

3.3.3 User-Defined Function Codes

ezTCP支援一个用户定义的函数。

种类	连接	端口	名称	函数代码	
				代码	服务器代码
数据	字节	输出端口(数字)	Write Pulse	105 (0x69)	

表 3-8 ezTCP支援的用户定义函数

4 Public 函数

4.1 Read Coils (FC 01)

用在查看数字输出端口状态上。

4.1.1 邀请

Request of Read Coils

图 4-1 Request of Read Coils

- byte 0: 函数代码
  Read Coils的函数代码是0x01。
- byte 1~2: 开始地址
  读取值得第一个数字输出端口的地址。
- byte 3~4: 输出端口个数
  指定数字端口输出端口数。可用值得范围是1 ~ n。

☞ ***n: 各产品的数字输出端口个数***

4.1.2 应答

Response of Read Coils

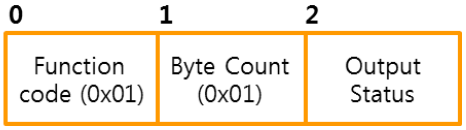


图 4-2 Response of Read Coils

- byte 0: 函数代码 (0x01)
- byte 1: 字节数 (0x01)
  (输出端口个数+7) / 8
- byte 2: 输出端口状态

  显示数字输出端口状态。根据端口个数追加字节，1个字节上8个端口按字节单位显示。由对应开始地址的端口，自LSB向MSB方向标示，字节值0指OFF，1指ON，没有被邀请或在产品没有的端口位的值由0填充。

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

图 4-3 输出端口状态

4.1.3 例外

Exceptions of Read Coils

0	1
Error code (0x81)	Exception code

图 4-4 Exception of Read Coils

- byte 0: 错误代码
  错误代码是 “函数代码 + 0x80”,即0x81。
- byte 1: 例外代码(Exception code)
  例外代码是0x01, 0x02或是0x03。

4.1.4 使用 例

下面是基本设定的读取ezTCP数字输出端口#0 ~ #7的例子。

- 邀请

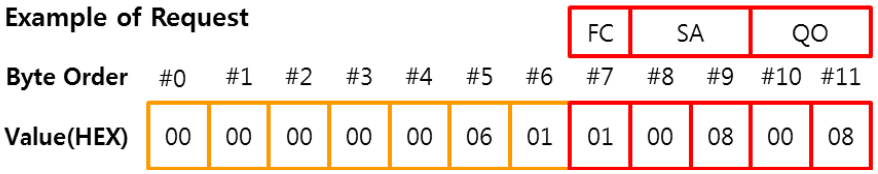


图 4-5 邀请例

字节顺序	值(HEX)	意义
7	0x01	函数代码01
8~9	0x0008	设定要读取的地址为8 (数字输出端口的基本值)
10~11	0x0008	自开始地址读取8个数字输出端口

表 4-1 邀请 例

- 应答

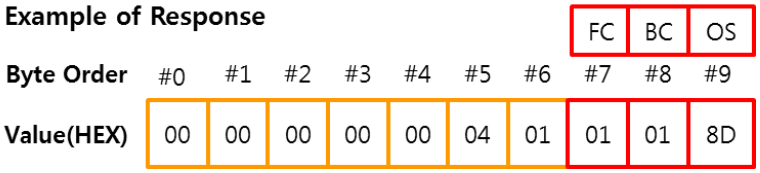


图 4-6 应答 例

字节顺序	值(HEX)	意义
7	0x01	函数代码01
8	0x01	1字节, 1 ~ 8个之间的输出端口
9	0x8D	(1000 1101) #0, 2, 3, 7 ON / #1, 4 ~ 6 OFF

表 4-2 应答 例

4.2 Read Discrete Inputs (FC 02)

确认数字输出端口状态时使用。

4.2.1 邀请

Request of Read Discrete Inputs

0	1	3
Function code (0x02)	Starting Address	Quantity of Inputs (1 ~ n)

图 4-7 Request of Read Discrete Inputs

- byte 0: 函数代码

  Read Discrete Inputs的函数代码是0x02。

- byte 1~2: 开始地址

  需要读取的第一个数字输入端口地址。

- byte 3~4: 输入端口个数

  指定需要读取输入端口的数。可用的值范围是1 ~ n。

☞ ***n: 各产品的数字输入端口个数***

4.2.2 应答

Response of Read Discrete Inputs

0	1	2
Function code (0x02)	Byte Count (0x01)	Input Status

图 4-8 Response of Read Discrete Inputs

- byte 0: 函数代码 (0x02)
- byte 1: 字节数 (0x01)

  (输入端口数+7)/8

- byte 2: 输入端口状态

  显示数字输入端口状态。根据端口个数按字节单位增加，1字节上通过字节单位显示8个端口。对应开始地址的端口由LSB到MSB方向标示，字节位0指OFF，1指ON，未被邀请或者在产品没有端口的相应字节值通过0填充。

图 4-9 输入端口状态

4.2.3 例外

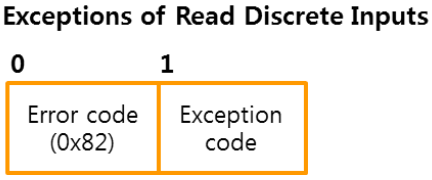


图 4-10 Exception of Read Discrete Inputs

- byte 0: 错误代码
  错误代码是"函数代码 + 0x80",即0x82。
- byte 1: 例外代码(Exception code)
  例外代码是0x01, 0x02或是0x03。

4.2.4 使用例

下面是读取基本设定的数字输入端口#0 ~ #7的例。

- 邀请

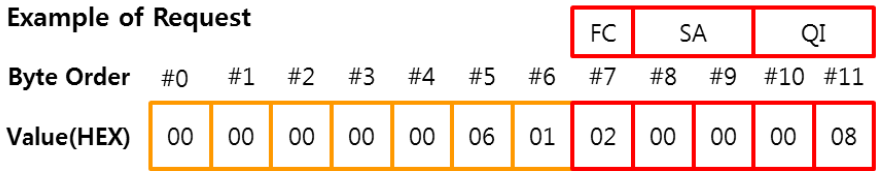


图 4-11 邀请例

字节顺序	值(HEX)	意义
7	0x02	函数代码 02
8~9	0x0000	设定预读取的地址0(数字输入端口基本值)
10~11	0x0008	自开始地址读取8个数字输入端口

表 4-3 邀请 例

- 应答

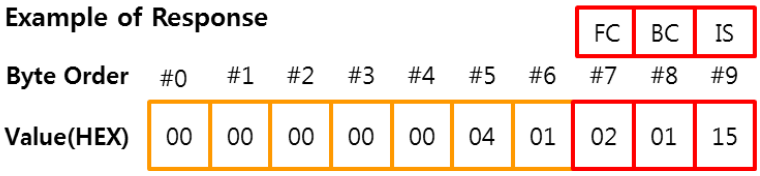


图 4-12 应答例

字节顺序	值(HEX)	意义
7	0x02	函数代码 02
8	0x01	1 字节, 1 ~ 8之间的输入端口
9	0x15	(0001 0101) #0, 2, 4 端口 / #1, 3, 5 ~ 7 端口OFF

表 4-4 应答 例

4.3 Read Holding Registers (FC 03)

数字/模拟输入端口，使用在数字输出端口状态确认。

4.3.1 邀请

Request of Read Holding Registers

图 4-13 Request of Read Holding Registers

- byte 0: 函数代码
  Read Holding Registers的函数代码是0x03。
- byte 1~2: 开始地址
  是读取状态值得第一个注册地址。

☞ ***可接近的ezTCP端口：输入的端口（数字/模拟），输出端口（数字）***

输入端口地址（数字）– 使用[输入端口地址]设定值

输入端口地址（模拟）– ***"[输入端口地址] + 4"***

输出端口地址（数字）– 使用[输出端口地址]设定值

- byte 3~4: 注册个数
  指定要读取的注册数。可使用值的范围是1至8。

4.3.2 应答

Response of Read Holding Registers

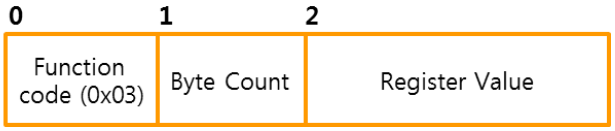


图 4-14 Response of Read Holding Registers

- byte 0: 函数代码 (0x03)
- byte 1: 字节 数
  注册数 × 2
- byte 2 ~: 注册 值
  显示数字/模拟端口状态。数字端口在一个注册上通过字节单位显示16个端口。对应开始地址的端口由LSB到MSB方向标示，字节位0指OFF，1指ON，未被邀请或者在产品没有端口的相应字节值通过0填充。

Register Value for Digital Inputs / Outputs (DI / DO)

图 4-15 有关数字端口的注册值

模拟端口将注册的下一阶层的10字节数据使用，上位6字节通过0填充。(数据范围: 0 ~ 1023).

Register Value for Analog Inputs (AI)

图 4-16 有关模拟端口的注册值

4.3.3 例外

Exceptions of Read Holding Registers

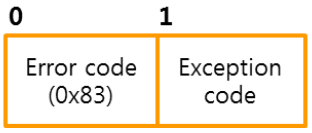


图 4-17 Exception of Read Holding Registers

- byte 0: 错误代码
  错误代码是"函数代码 + 0x80", 即0x83。
- byte 1: 例外代码(Exception code)
  例外代码是0x01, 0x02或是0x03。

4.3.4 使用例

下面是确认基本设定的ezTCP输入端口状态的内容。

- 邀请

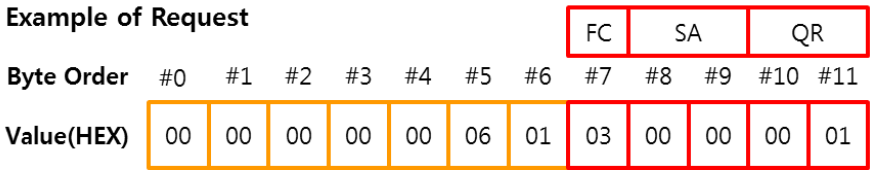


图 4-18 邀请 例

字节顺序	值(HEX)	意义
7	0x03	函数代码 03
8~9	0x0000	设定为要读取的地址0(数字 输入端口基本值)
10~11	0x0001	自开始地址读取一个注册

表 4-5 邀请 例

- 应答

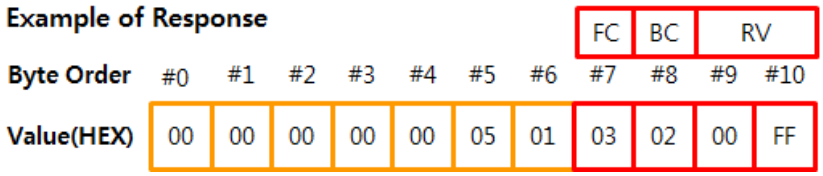


图 4-19 应答 例

字节顺序	值(HEX)	意义
7	0x03	函数代码 03
8	0x02	2 字节，即1个的注册
9~10	0x00FF	(1111 1111) 数字输入端口 #0 ~ 7 ON

表 4-6 应答 例

4.4 Read Input Registers (FC 04)

用在数字/模拟输入端口状态确认。

4.4.1 邀请

Request of Read Input Registers

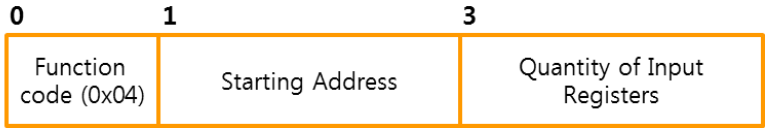


图 4-20 Request of Read Input Registers

- byte 0: 函数代码

  Read Input Registers的函数代码是0x04。

- byte 1~2: 开始地址

  需要读取值得第一个注册地址。

☞ ***可连接的ezTCP端口：输入端口(数字/模拟)***

输入端口地址（数字）– 使用[输入端口地址]设定值

输入端口地址（模拟）– "[输入端口 地址] + 4"

- byte 3~4: 注册个数

  指定可读取值得注册数。可使用值的范围是1至8。

4.4.2 应答

Response of Read Input Registers

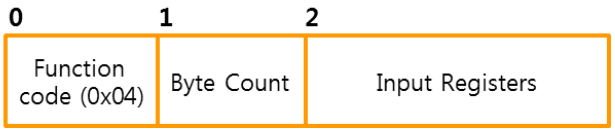


图 4-21 Response of Read Input Registers

- byte 0: 函数代码 (0x04)
- byte 1: 字节数

  注册个数 × 2

- byte 2 ~: 注册 值

  显示数字/模拟输入端口的状态。

  数字端口在一个注册上通过字节单位显示16个端口。对应开始地址的端口由LSB到MSB方向标示，字节位0指OFF，1指ON，未被邀请或者在产品没有端口的相应字节值通过0填充。

Register Value for Digital Inputs (DI)

15															0
0	0	0	0	0	0	0	0	#7	#6	#5	#4	#3	#2	#1	#0
MSB															LSB

图 4-22 对数字输入端口的注册 值

模拟端口使用注册的下一阶层10字节数据，上一阶层6字节填充为0。

(数据范围: 0 ~ 1023).

Register Value for Analog Inputs (AI)

15						9	0
0	0	0	0	0	0	Analog Input Value	

图 4-23 有关模拟端口的注册 值

4.4.3 例外

Exceptions of Read Input Registers

0	1
Error code (0x84)	Exception code

图 4-24 Exception of Read Input Registers

- byte 0: 错误代码

  错误代码是"函数代码 + 0x80",即0x84。

- byte 1: 例外代码(Exception code)

  例外代码는 0x01, 0x02或是0x03。

4.4.4 使用例

下面是确认基本设定ezTCP输入端口状态的使用例。

- 邀请

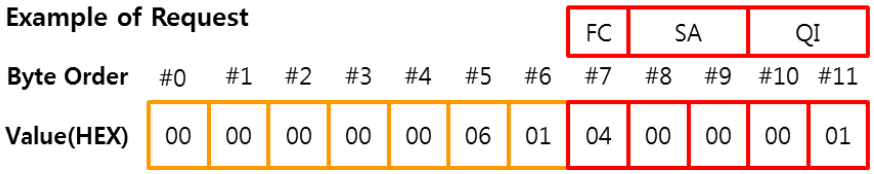


图 4-25 邀请 例

字节 顺序	值(HEX)	意义
7	0x04	函数代码 04
8~9	0x0000	设定读取地址为0(数组输入端口基本值)
10~11	0x0001	自开始地址读取1个注册

表 4-7 邀请 例

- 应答

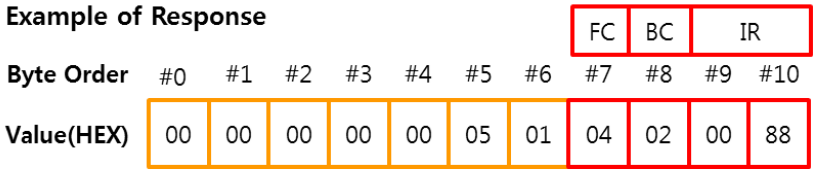


图 4-26 应答 例

字节顺序	值(HEX)	意义
7	0x04	函数代码 04
8	0x02	2字节, 1个注册
9~10	0x0088	(1000 1000) #3, 7 端口 ON / #0 ~ 2, 4 ~ 6 端口 OFF

表 4-8 应答 例

4.5 Write Single Coil (FC 05)

适用一个数字输出端口ON/OFF控制。

4.5.1 邀请 / 应答

Request / Response of Write Single Coil

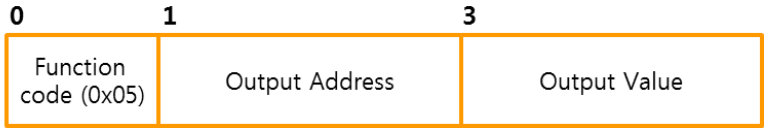


图 4-27 Request / Response of Write Single Coil

- byte 0: 函数代码
  Write Single Coil的函数代码是0x05。
- byte 1~2: 输出端口地址
  需要控制的数字输出端口的地址。
- byte 3~4: 数字 值
  0xFF00为了输出端口ON，0x0000为了OFF使用。

☞ ***Write Single Coil的邀请框架与应答框架结构相同。***

4.5.2 例外

Exceptions of Write Single Coil

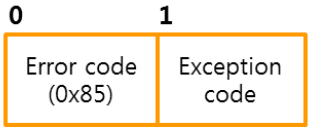


图 4-28 Exception of Write Single Coil

- byte 0: 错误代码
  错误代码的"函数代码 + 0x80",即0x85。
- byte 1: 例外代码(Exception code)
  例外代码是0x01, 0x02, 0x03或是0x04。

4.5.3 使用例

下面是基本设定的ezTCP数字输出端口#0进行ON的例。

- 邀请及应答

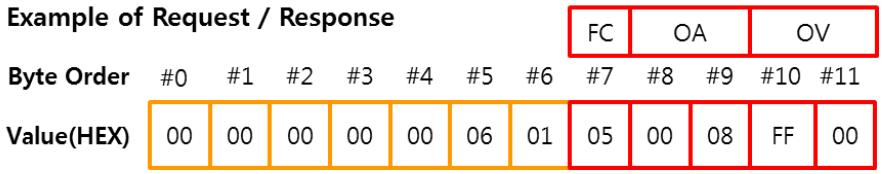


图 4-29 邀请 / 应答 例

字节顺序	值(HEX)	意义
7	0x05	函数代码 05
8~9	0x0008	设定需要控制的地址为8(数字输出端口基本值)
10~11	0xFF00	数据值 0xFF00 (输出端口 ON)

表 4-9 邀请 / 应答 例

4.6 Write Single Register (FC 06)

在控制输出端口ON/OFF时使用。

4.6.1 邀请 / 应答

Request / Response of Write Single Register

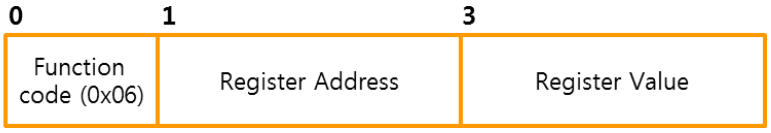


图 4-30 Request / Response of Write Single Register

- byte 0: 函数代码
  Write Single Register的函数代码是0x06。
- byte 1~2: 注册地址
  需要控制数字输出端口的地址。
- byte 3~4: 注册 值

  输出端口控制时利用的值，一个注册上将16个输出端口按字节单位显示。对应开始地址的端口由LSB到MSB方向标示，字节位0指OFF，1指ON，超过输出端口个数或对应产品上没有的端口的字节值将被无视。

  Register Value for Digital Outputs (DO)

图 4-31 对数字输出端口的注册值

☞ ***Write Single Register的邀请框架与应答框架构造相同。***

4.6.2 例外

Exceptions of Write Single Register

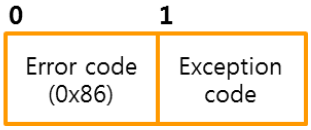


图 4-32 Exception of Write Single Register

- byte 0: 错误代码
  错误代码是"函数代码 + 0x80",即0x86。
- byte 1: 例外代码(Exception code)
  例外代码是0x01，0x02或是0x04。

4.6.3 使用例

下面是控制基本设定的ezTCP数字输出端口的使用例。

- 邀请及应答

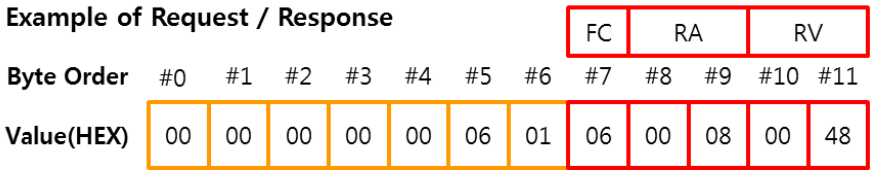


图 4-33 邀请 / 应答 例

字节顺序	值(HEX)	意义
7	0x06	函数代码 06
8~9	0x0008	设定要控制的地址为8(数字输出端口基本值)
10~11	0x0048	(0100 1000) #3, 6端口ON / #0 ~ 2, 4, 5, 7 端口OFF

表 4-10 邀请 / 应答 例

4.7 Read Exception Status (FC 07)

Read Exception Status与例外应答无关，确认ezTCP的输出端口中Macro设定的端口。

4.7.1 邀请

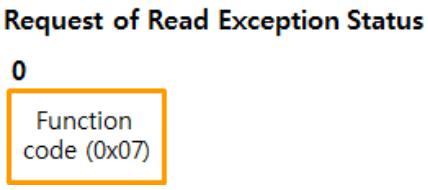


图 4-34 Request of Read Exception Status

- byte 0: 函数代码
  Read Exception Status的函数代码是0x07。

4.7.2 应答

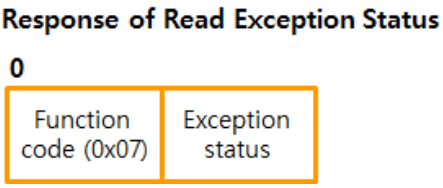


图 4-35 Response of Read Exception Status

- byte 0: 函数代码 (0x07)
- byte 1: 端口状态值 (Exception Status)
  设定Macro模式的输出端口为字节1，不是的端口为字节0。自第一个输出端口开始由LSB到MSB方向标示，对应在产品没有的端口字节值由0填充。

4.7.3 例外

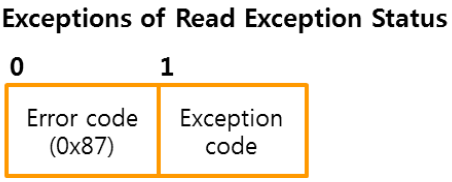


图 4-36 Exception of Read Exception Status

- byte 0: 错误代码
  错误代码是"函数代码 + 0x80"，即0x87。
- byte 1: 例外代码(Exception code)
  例外代码是0x01。

4.7.4 使用例

确认ezTCP的Macro模式被设定的数字输出端口的使用例。

- 邀请

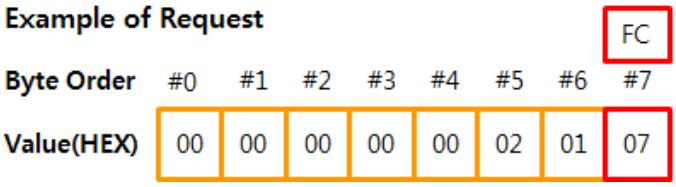


图 4-37 邀请 例

字节顺序	值(HEX)	意义
7	0x07	函数代码 07

表 4-11 邀请 例

- 应答

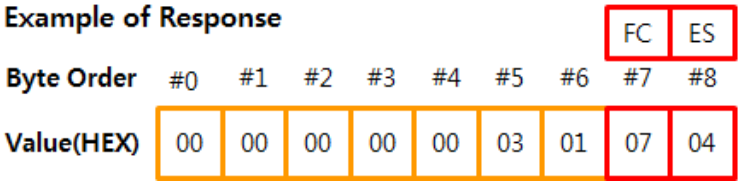


图 4-38 应答 例

字节顺序	值(HEX)	意义
7	0x07	函数代码 07
8	0x04	(0000 0100) #2 端口 Macro ON

表 4-12 应答 例

4.8 Write Multiple Coils (FC 15)

应用在连续有个几个数字输出端口ON/OFF控制上。

4.8.1 邀请

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

图 4-39 Request of Write Multiple Coils

- byte 0: 函数代码

  Write Multiple Coils的函数代码是0x0F。

- byte 1~2: 开始地址

  需要控制的第一个数字输出端口的地址。

- byte 3~4: 输出端口个数

  指定需要控制的数字输出端口数。可用值的范围是1 ~ n。

☞ ***n: 各产品的数字输出端口个数***

- byte 5: 字节数 (0x01)

  (输出端口个数7) / 8

- byte 6: 输出端口 值

  在数字输出端口控制中利用的值，根据端口个数按字节单位追加，在一个字节通过字节单位显示8个输出端口。对应开始地址端口自LSB到MSB方向适用，字节数0指OFF，1指ON。超过输出端口个数或是在产品中没有的产品端口其相应字节值被无视。

Output Value for Digital Outputs (DO)

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

图 4-40 对数字输出端口值

4.8.2 应答

除字节数与输出端口值部分与邀请数据包相同。

Response of Write Multiple Coils

图 4-41 Response of Write Multiple Coils

- byte 0: 函数代码 (0x0F)
- byte 1~2: 开始地址
- byte 3~4: 输出端口个数

4.8.3 例外

Exceptions of Write Multiple Coils

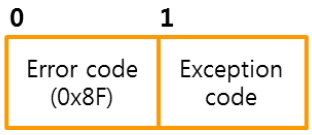


图 4-42 Exception of Write Multiple Coils

- byte 0: 错误代码
  错误代码是"函数代码 + 0x80",即0x8F。
- byte 1: 例外代码(Exception code)
  例外代码是0x01, 0x02, 0x03或是0x04。

4.8.4 使用例

下面是控制4个ezTCP输出端口的基本设定使用例。

- 邀请

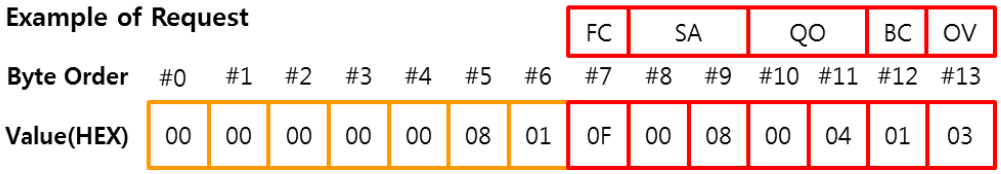


图 4-43 邀请 例

字节顺序	值(HEX)	意义
7	0x0F	函数代码 15
8~9	0x0008	设定要控制地址8(数字输出端口基本值)
10~11	0x0004	需要自开始地址控制4个数字输出端口
12	0x01	1 字节, 输出1 ~8个之间的数字输出端口
13	0x03	(0000 0011) #0, 1 端口ON / #2, 3端口OFF

表 4-13 邀请例

- 应答

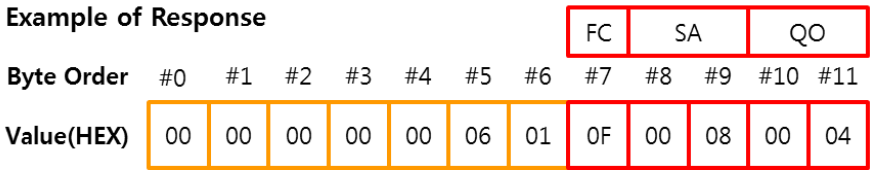


图 4-44 应答 例

字节顺序	值(HEX)	意义
7	0x0F	函数代码 15
8~9	0x0008	设定将控制的地址为8(数字输出端口基本值)
10~11	0x0004	自开始地址控制4个数字输出端口

表 4-14 应答 例

4.9 Write Multiple Registers (FC 16)

在控制输出端口ON/OFF使用。

4.9.1 邀请

Request of Write Multiple Registers

图 4-45 Request of Write Multiple Registers

- byte 0: 函数代码
  Write Multiple Registers的函数代码是0x10。
- byte 1~2: 开始地址
  要用值的第一个注册地址。
- byte 3~4: 注册个数 (0x0001)
  指定可用值注册数。可用值为1。
- byte 5: 字节数 (0x02)
  注册个数× 2
- byte 6~7: 注册 值

是控制输出端口的值，一个注册上以字节单位显示16个输出端口。对应开始地址端口自LSB到MSB方向适用，字节数0指OFF，1指ON。超过输出端口个数或是在产品中没有的产品端口其相应字节值被无视。

Register Value for Digital Outputs (DO)

图 4-46 对数字输出端口的注册值

4.9.2 应答

除字节数与注册值部分与邀请数据包相同。

Response of Write Multiple Registers

图 4-47 Response of Write Multiple Registers

- byte 0: 函数代码 (0x10)
- byte 1~2: 开始地址
- byte 3~4: 注册个数

4.9.3 例外

Exceptions of Write Multiple Registers

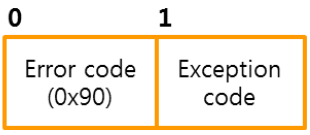


图 4-48 Exception of Write Multiple Registers

- byte 0: 错误代码

  错误代码是"函数代码 + 0x80",即0x90。

- byte 1: 例外代码

  例外代码是0x01, 0x02, 0x03或是0x04。

4.9.4 使用例

下面是控制基本设定的ezTCP输出端口的使用例。

- 邀请

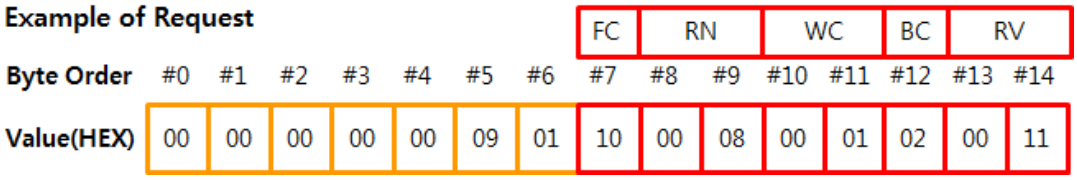


图 4-49 邀请 例

字节顺序	值(HEX)	意义
7	0x10	函数代码 16
8~9	0x0008	设定可控制的地址8(数字输出端口基本值)
10~11	0x0001	自开始地址使用一个注册值写。
12	0x02	2 字节, 1个注册
13~14	0x0011	(0001 0001) #0, 4端口 ON / #1 ~3, 5 ~ 7端口OFF

表 4-15 邀请 例

- 应答

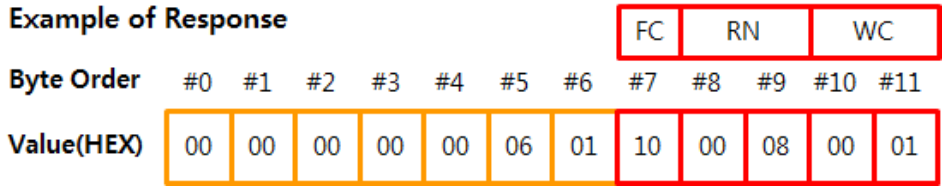


图 4-50 应答 例

字节顺序	值(HEX)	意义
7	0x03	函数代码 16
8~9	0x0008	设定需控制地址为8(数字输出端口基本值)
10~11	0x0001	开始地址起写一个注册值

表 4-16 应答 例

4.10 Encapsulated Interface Transport (FC 43)

将不是Modbus/TCP的，在其他协议使用的通信数据包，搭载在部分Modbus/TCP协议数据而进行通信，此种通信结构称为MEI(Modbus Encapsulated Interface)。根据封装协议的种类区分MEI，13(0x0D) – CANopen General Reference与14(0x0E) – Read Device Identification 共2个MEI种类。

4.10.1 邀请

Request of Encapsulated Interface Transport

TU 4-51 Request of Encapsulated Interface Transport

- byte 0: 函数代码

  Encapsulated Interface Transport的函数代码是0x2B。

- byte 1: MEI 种类 (0x0D或是0x0E)

  ezTCP只支持14(0x0E) – Read Device Identification。

☞ ***MEI: Modbus Encapsulated Interface的缩写***

- byte 2~: 实际数据(n bytes)

  根据MEI种类内容不同。

4.10.2 应答

Response of Encapsulated Interface Transport

图 4-52 Response of Encapsulated Interface Transport

- byte 0: 函数代码 (0x2B)
- byte 1: MEI种类(0x0D或是0x0E)
- byte 2~: 实际数据(n bytes)

4.10.3 例外

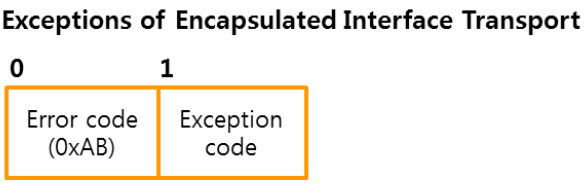


图 4-53 Exception of Encapsulated Interface Transport

- byte 0: 错误代码

  错误代码是"函数代码 + 0x80", 即0xAB。

- byte 1: 例外代码

  例外代码是0x01, 0x02, 0x03或是0x04。

4.10.4 使用例

请参考 "4.11 Read Device Identification (FC 43 / 14)" 。

4.11 Read Device Identification (FC 43 / 14)

为了确定Modbus服务器设备信息而使用，各自设备信息称为对象，对象根据其特征大为分3个种类。

- 基本信息 (Basic Device Identification) – 必选
  制造商,产品代码, 固件版本
- 常规设备标识(Regular Device Identification) – 选项
  制造商网页地址,产品名称,模块名称,用户应用程序
- 扩展设备标识(Extended Device Identification) – 选项
  ezTCP支持说明参数值、产品MAC地址、 确认Macro模式设定端口、输出端口说明、输出端口说明等共5个项目。

种类	帐户	对象名称及说明	数据状态	是否必选
Basic	0x00	VendorName	ASCII String	**必选**
	0x01	ProductCode	ASCII String	**必选**
	0x02	MajorMinorRevision	ASCII String	**必选**
Regular	0x03	VendorUrl	ASCII String	选项
	0x04	ProductName	ASCII String	选项
	0x05	ModelName	ASCII String	选项
	0x06	UserApplicationName	ASCII String	选项
	0x07~0x7F	Reserved		选项
Extended	0x80	Comment	Binary	选项
	0x81	MAC Address	ASCII String	选项
	0x82	Macro Mode	Binary	选项
	0xA0 + n	Input Comments	Binary	选项
	0xB0 + n	Output Comments	Binary	选项

标 4-17 对象帐户

☞ ***与其它对象相同，除以 ASCII型态传送的MAC Address， Extended对象项目只按产品设定值上设定的二进制型态传送。***

☞ ***n: 各产品的数字输入/输出端口个数***

4.11.1 邀请

Request of Read Device Identification

0	1	2	3
Function code (0x2B)	MEI Type (0x0E)	Read Device ID Code	Object ID

图 4-54 Request of Read Device Identification

- byte 0: 函数代码 (0x2B)
- byte 1: MEI 种类 (0x0E – Read Device Identification)
- byte 2: 设备帐户(0x01 / 0x02 / 0x03 / 0x04)
  0x01: Basic 全部 邀请
  0x02: Regular 全部 邀请
  0x03: Extended 全部 邀请
  0x04: Basic/Regular/Extended 没有区分特征只邀请对象帐户
  区分邀请的信息。全部邀请的情况， 通过一次交易无法发送全部信息时， 需要多个交易。ezTCP对Basic/Regular的全部邀请通过一次交易结束， 在Extended需要3遍交易。
- byte 3: 对象帐户
  指第一个接收的对象帐户。
  邀请全部对象: 第一个交易 – 0x00
  邀请全部对象: 第二个及之后交易 – 应答前接收的值
  邀请单个对象: 实际要接收的对象帐户值

  邀请全部对象时(设备帐户为 0x01, 0x02,或是0x03)对象帐户值不合适， ezTCP应答第一个交易， 交易将从头开始。

4.11.2 应答

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

图 4-55 Response of Read Device Identification

- byte 0: 函数代码 (0x2B)
- byte 1: MEI种类 (0x0E – Read Device Identification)
- byte 2: 设备帐户(0x01, 0x02, 0x03, 0x04) – 与邀请相同
- byte 3: Conformity Level
  区分支持的对象种类和邀请形态， ezTCP使用0x83。
  0x01: Basic (只支持全部邀)

0x02: Regular (只支持全部邀请)
0x03: Extended (只支持全部邀请)
0x81: Basic (全部/单一 邀请两个都支持)
0x82: Regular (全部/单一 邀请两个都支持)
0x83: Extended (全部/单一 邀请两个都支持)

- byte 4: More Follows
  邀请全部对象需要多个交易时使用。
  0x00: 没有再多的对象,指最后的交易。
  0xFF: 有更多的对象, 需要追加的交易
  邀请单个对象: 固定为0x00

- byte 5: 下个对象帐户
  More Follows为0xFF时:下个邀请需要使用的对象帐户
  More Follows为0x00时: 0x00

- byte 6: 对象个数
  邀请全部对象: 应答的对象个数
  邀请单个对象: 0x01

- byte 7: 对象帐户
  邀请全部对象: 第一个对象
  邀请单个对象: 邀请的对象

- byte 8: 对象长度
  单位是字节，显示第一个对象数据的长度。

- byte 9~: 对象数据
  第一个对象数据。如果应答对象的个数为多个, 第二个对象帐号/长度/数据项目反复被添加。

4.11.3 例外

Exceptions of Read Device Identification

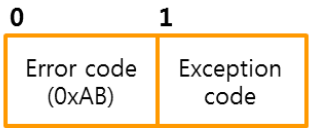


图 4-56 Exception of Read Device Identification

- byte 0: 错误代码
  错误代码是"函数代码 + 0x80", 即0xAB。

- byte 1: 例外代码
  例外代码是0x01, 0x02, 0x03或是0x04。

4.11.4 使用例 – Basic Device Identification

下面是读取CIE-H10的 Basic设备的信息使用例。

- 邀请

姓名	值(HEX)	意义
函数代码	0x2B	函数代码 43
MEI 种类	0x0E	Read Device Identification
设备帐户	0x01	邀请Basic全部对象
对象帐户	0x00	VendorName (开始全部邀请)

表 4-18 邀请 例

- 应答

名称	值	意义
函数代码	0x2B	函数代码 43
MEI 种类	0x0E	Read Device Identification
设备帐户	0x01	邀请Basic全部对象
Conformity Level	0x83	支持Extended 全部/单一 邀请
More Follows	0x00	没有追加交易
下面对象帐户	0x00	最后交易
对象个数	0x03	包括3个交易信息
对象帐户	0x00	VendorName
对象长度	0x18	24 字节
对象数据	"Sollae Systems Co., Ltd."	
对象帐户	0x01	ProductCode
对象长度	0x02	2 字节
对象数据	"20"	CIE-H10의 Product Code
对象帐户	0x02	MajorMinorRevision
对象长度	0x05	5 字节
对象数据	"V1.5B"	1.5B版本

表 4-19 应答 例

4.11.5 使用例 – Extended Device Identification

下面是读取CIE-H10的Extended设备信息的例。

- 邀请 1

名称	值(HEX)	意义
函数代码	0x2B	函数代码 43
MEI 种类	0x0E	Read Device Identification
设备帐户	0x03	邀请全部Extended对象
对象帐户	0x00	Comment (开始全部邀请)

表 4-20 邀请 1

- 应答 1

名称	值	意义
函数代码	0x2B	函数代码 43
MEI 种类	0x0E	Read Device Identification
设备帐户	0x03	邀请全部Extended对象
Conformity Level	0x83	支持Extended全部/单一邀请
More Follows	0xFF	有追加交易
下面对象帐户	0xA0	输入端口说明
对象个数	0x03	包括3个对象信息
对象帐户	0x80	Comment
对象长度	0x00	0 字节
对象帐户	0x81	MAC Address
对象长度	0x11	17 字节
对象数据	"00:30:F9:00:00:01"	产品MAC地址
对象帐户	0x82	Macro Mode
对象长度	0x01	1 字节
对象数据	0x81	#0, 7 端口Macro模式

表 4-21 应答 1

- 邀请 2

名称	值(HEX)	意义
函数代码	0x2B	函数代码 43
MEI 种类	0x0E	Read Device Identification
设备帐户	0x03	邀请全部Extended对象
对象帐户	0xA0	说明输入端口

表 4-22 邀请 2

- 应答 2

名称	值	意义
函数代码	0x2B	函数代码 43
MEI 种类	0x0E	Read Device Identification
设备帐户	0x03	Extended对象全部邀请
Conformity Level	0x83	支持Extended全部/单一邀请
More Follows	0xFF	有追加交易
下个对象帐户	0xB0	输出端口说明
对象个数	0x08	包括8个对象信息
对象帐户	0xA0	输入端口 #0 Comment
对象长度	0x03	3 字节
对象数据	"DI0"	基本设定值
对象帐户	0xA1	输入端口 #1 Comment
对象长度	0x03	3 字节
对象数据	"DI1"	基本设定值
对象帐户	0xA2	输入端口 #2 Comment
对象长度	0x03	3 字节
对象数据	"DI2"	基本设定值
对象帐户	0xA3	输入端口 #3 Comment
对象长度	0x03	3 字节
对象数据	"DI3"	基本设定值
对象帐户	0xA4	输入端口 #4 Comment
对象长度	0x03	3 字节
对象数据	"DI4"	基本设定值
对象帐户	0xA5	输入端口 #5 Comment
对象长度	0x03	3 字节
对象数据	"DI5"	基本设定值
对象帐户	0xA6	输入端口 #6 Comment
对象长度	0x03	3 字节
对象数据	"DI6"	基本设定值
对象帐户	0xA7	输入端口 #7 Comment
对象长度	0x03	3 字节
对象数据	"DI7"	基本设定值

表 4-23 应答 2

● 邀请 3

名称	值(HEX)	意义
函数代码	0x2B	函数代码 43
MEI 种类	0x0E	Read Device Identification
设备帐户	0x03	Extended对象全部邀请
对象帐户	0xB0	对象帐户

表 4-24 邀请 3

- 应答 3

名称	值	意义
函数代码	0x2B	函数代码 43
MEI 种类	0x0E	Read Device Identification
设备帐户	0x03	Extended对象全部邀请
Conformity Level	0x83	支持Extended 全部/单一邀请
More Follows	0x00	没有追加交易
下个对象帐户	0x00	最后交易
对象个数	0x08	包括8个对象信息
对象帐户	0xB0	输出端口 #0 Comment
对象长度	0x03	3 字节
对象数据	"DO0"	基本设定值
对象帐户	0xB1	输出端口 #1 Comment
对象长度	0x03	3 字节
对象数据	"DO1"	基本设定值
对象帐户	0xB2	输出端口 #2 Comment
对象长度	0x03	3 字节
对象数据	"DO2"	基本设定值
对象帐户	0xB3	输出端口 #3 Comment
对象长度	0x03	3 字节
对象数据	"DO3"	基本设定值
对象帐户	0xB4	输出端口 #4 Comment
对象长度	0x03	3 字节
对象数据	"DO4"	基本设定值
对象帐户	0xB5	输出端口 #5 Comment
对象长度	0x03	3 字节
对象数据	"DO5"	基本设定值
对象帐户	0xB6	输出端口 #6 Comment
对象长度	0x03	3 字节
对象数据	"DO6"	基本设定值
对象帐户	0xB7	输出端口 #7 Comment
对象长度	0x03	3 字节
对象数据	"DO7"	基本设定值

表 4-25 应答 3

5 用户定义函数

5.1 Write Pulse (FC 105)

为了控制输出端口只在一定时间期间维持ON或是OFF状态，再返回原来状态的脉冲状态而使用。

5.1.1 邀请 / 应答

Request / Response of Write Pulse

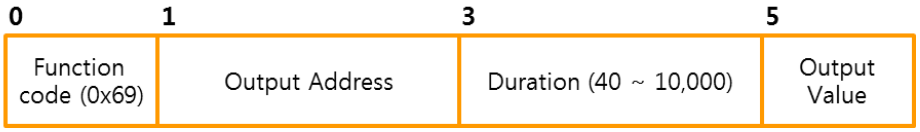


图 5-1 Request / Response of Write Pulse

- byte 0: 函数代码
  Write Pulse的函数代码是0x69。
- byte 1~2: 输出端口地址
  是控制数字输出端口的地址。
- byte 3~4: 维持时间
  单位是毫秒(ms)，可设定范围是40 ~ 10,000 (0x0028 ~ 0x2710)。
- byte 5: 输出端口值
  为了维持输出端口ON设定0xFF或是为了维持OFF，设定0x00。输出端口值为当前输出端口相同值时例外代码回应0x04。

☞ ***Write pulse的邀请框架与应答框架结构相同。***

5.1.2 例外

Exceptions of Write Pulse

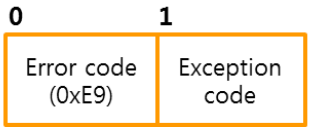


图 5-2 Exception of Write Pulse

- byte 0: 错误代码
  错误代码是 "函数代码 + 0x80"，即0xE9。
- byte 1: 例外代码(Exception code)
  例外代码是0x01, 0x02, 0x03或是0x04。

5.1.3 使用例

- 邀请 / 应答

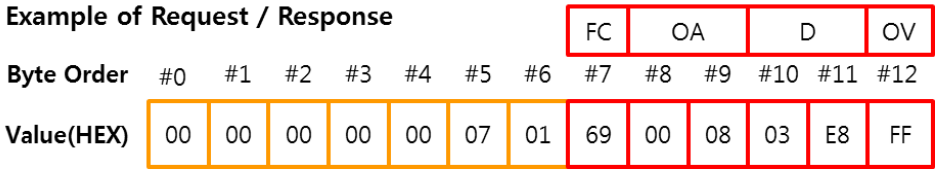


图 5-3 邀请 / 应答例

字节顺序	值(HEX)	意义
7	0x69	函数代码 105
8~9	0x0008	显示要控制的输出端口地址
10~11	0x03E8	维持1秒 (1000ms = 0x03E8)
12	0xFF	数据值0xFF (维持ON状态)

表 5-1 邀请 / 应答 例

☞ ***当前输出端口为之前接受的邀请 FC 105控制中或者MACRO模式时无法控制。***

6 其他需要知道的事项

6.1 例外代码及意义

例外代码	命令	意义
0x01	Illegal Function	函数代码错误
0x02	Illegal Data Address	开始地址错误
0x03	Illegal Data Value	数据值错误
0x04	Server Device Failure	实行邀请命令失败

表 6-1 例外代码

6.2 读取模拟端口值

6.2.1 邀请模拟端口值

ezTCP的模拟输入端口可利用 FC 03 (Read Holding Registers)或是 FC 04 (Read Input Registers)读取。为了此需要指定注册地址，ezTCP的模拟输入端口连接在了[输入端口开始地址] + 4上。如[输入端口开始地址]设定为0时，模拟输入端口的地址为4号。因此传送例如下（使用 FC 04时，除FC值之外部分相同。

- 邀请例

Example of Request												
								FC	SA		QR	
Byte Order	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11
Value(HEX)	00	00	00	00	00	06	01	03	00	04	00	01

图 6-1 ADC端口值邀请例

6.2.2 模拟值应答

模拟值得读取邀请的应答如下。

- 应答例

Example of Response											
								FC	BC	RV	
Byte Order	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Value(HEX)	00	00	00	00	00	05	01	03	02	02	7F

图 6-2 ADC端口值应答 例

在上表中模拟值通过0x027F利用10进制为639。

☞ ***模拟端口由2的10平方，即10字节(0 ~ 1,023)显示。***

6.3 使用

6.3.1 设定 Modbus/TCP

- CIE-M10/H12/H14/H10

  通过ezManager搜索产品后设定③上标示的部分。

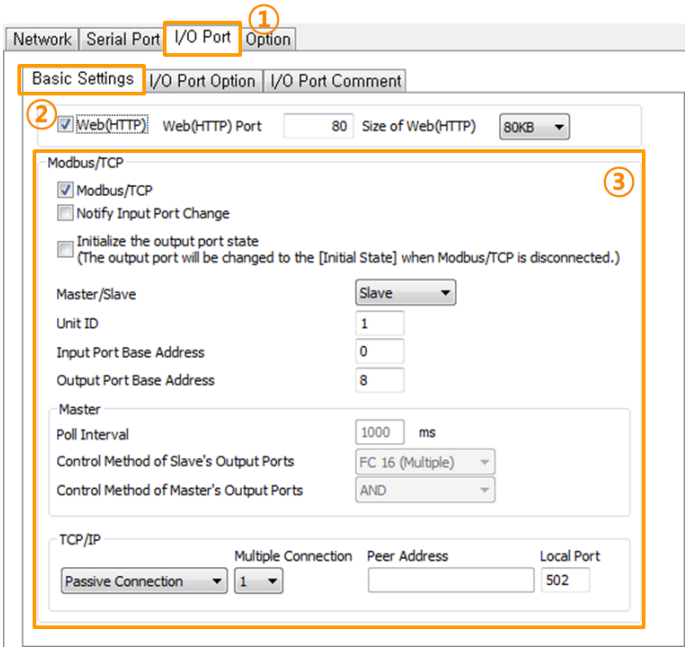


图 6-3 设定ezManager Modbus/TCP

- EZI-10

  通过ezConfigIO搜索产品后设定标示的部分。

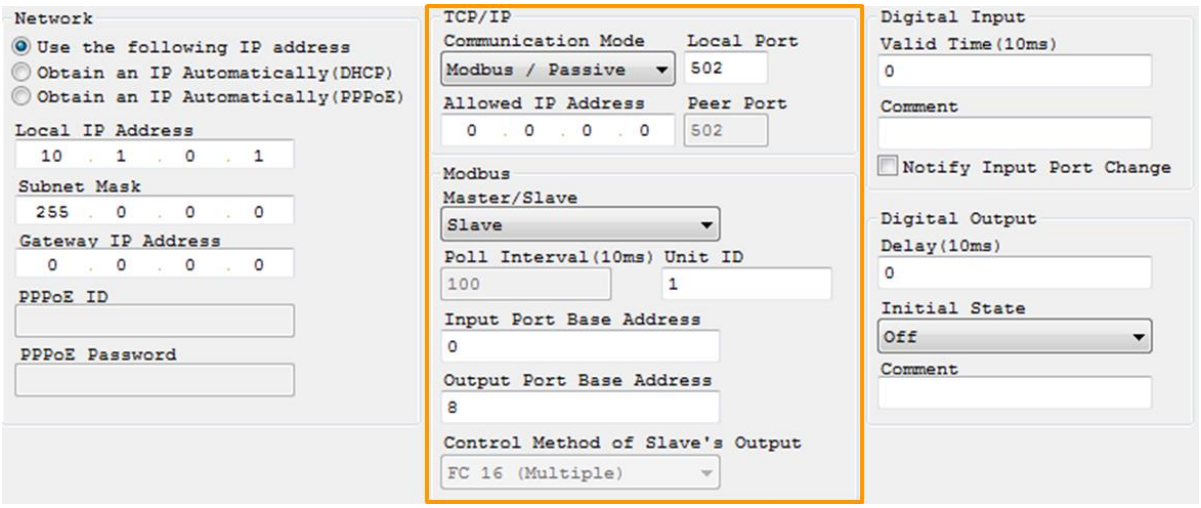


图 6-4 ezConfigIO Modbus/TCP 设定

● 设定项目

项目	说明
Modbus/TCP	是否使用协议
提醒输入端口变更	Slave在没有Master的提问，当输入端口变更时即时提醒的功能
初始化输出端口	Modbus/TCP连接终止时将输入端口状态可变更为[初始状态]
Master / Slave	Modbus/TCP 动作类型
通信周期기	Master发送提问的周期 (单位: 毫秒)
单位帐号	Master与Slave的帐户
输入端口地址	输入端口 开始地址
输出端口地址	输出端口 开始地址
Slave输出端口控制方式	Slave的输出端口控制方式(现有或是扩张)
Master输出端口控制方式	Master的输出端口控制方式(逻辑平方或是和)
手动/自动连接	自动: 试图连接, 手动: 等待连接
多重连接	使用多重TCP连接及对话数 (1 ~ 8)
通信地址	自动连接时地址
通信端口	通信端口号码

表 6-2 Modbus/TCP 设定项目

6.3.2 设定例

项目	ezTCP	其他ezTCP或是Modbus/TCP程序
本地IP地址	192.168.0.10	192.168.0.20
子网掩码	255.255.255.0	255.255.255.0
Modbus/TCP	确定或是选择	-
Master / Slave	Slave	Master
通信周期	-	1,000ms (1秒)
单位帐号	1	1
输入端口地址	0	0
输出端口地址仝	8	8
手动/自动连接	手动连接	自动连接
多重连接	3	-
通信地址	-	192.168.0.10
通信端口	-	502
产品本地端口	502	-

表 6-3 设定 例

6.4 样品代码

我公司为使用ezTCP 数字输入/输出功能功能的用户提供**Modbus/TCP**样品代码。请应用在程序实施计划中。

☞ ***样品代码可在我公司网站下载。(http://www.eztcp.com/ch/download/pds.php)***

6.4.1 提供了版本

- C++ (Visual Studio 6.0)
- C++ (Visual Studio 2008)
- Basic (Visual Basic)
- C (Linux)

7 串行 Modbus/TCP

串行**Modbus/TCP通过产品的串行端口**监视/控制I/O。 ezTCP I/O产品中有串行端口的产品支持串行Modbus/TCP。

7.1 特征

- 将现有**Modbus/TCP** 数据向串行端口发送/接收。
- 利用串行端口的数字输入/输出控制
- 没有连接过程，单纯数据发送/接收
  依据数据链的状态有可能丢失数据，为了防止此情况，请使用硬件流量控制(RTS/CTS)。

7.2 使用

7.2.1 设定方法

- 设定串行**Modbus/TCP**模式

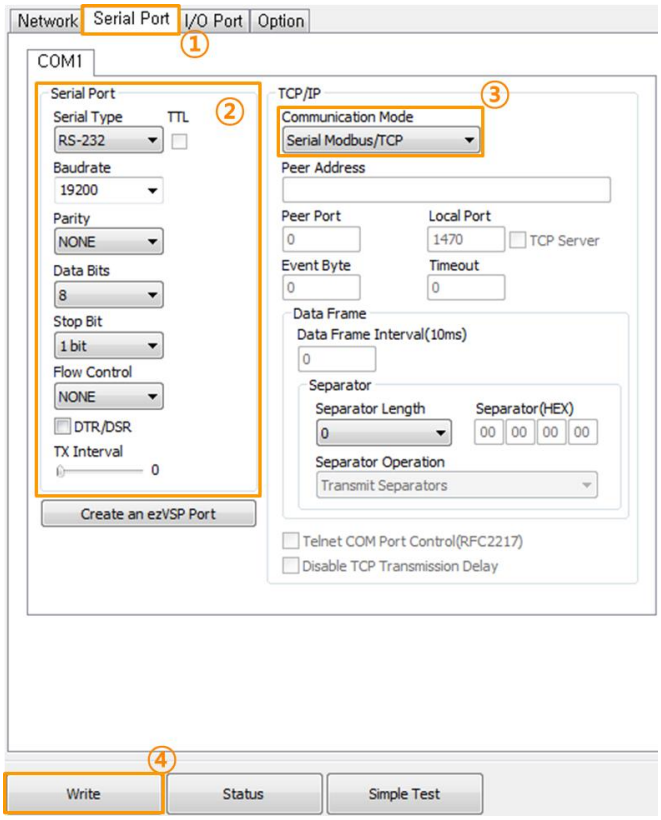


图 7-1 设定串行Modbus/TCP模式

① 移动到[串行端口]栏

② 设定串行端口项目

③ 在[TCP/IP 通信设定]将通信模式选择为[串行Modbus/TCP]

④ 通过[保存]按钮保存参数值

7.3 试启动

7.3.1 通信准备

为了测试串行**Modbus/TCP** 工作，请按如下构成。

☞ ***不连接网线也无妨。***

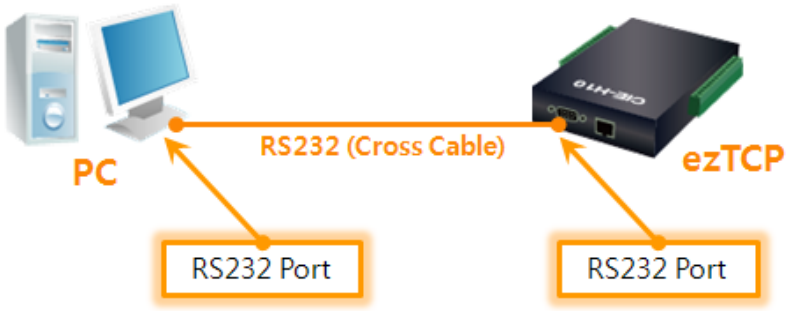


图 7-2 通信准备

为了测试Modbus/TCP设定请按如下基本值设定。

项目	基本值
Modbus/TCP	选定
提醒输入端口变更	不选
输出端口状态初始化	不选
Master /Slave	Slave
通信周期	1,000
单位帐号	1
输入端口 地址	0
输出端口 地址	8

表 7-1 Modbus/TCP 基本设定值

7.3.2 传送测试数据

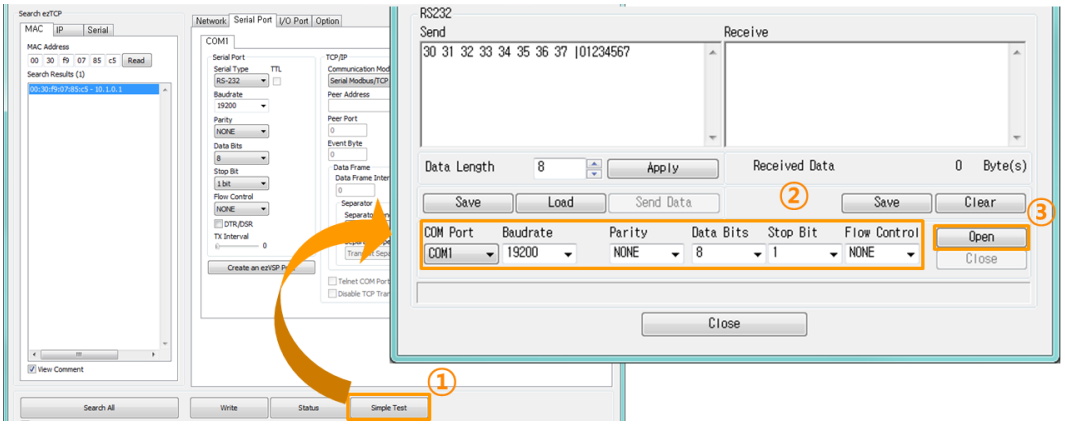


图 7-3 传送测试数据 1

① 点击ezManager的[通信测试]按钮

② 选择ezTCP与PC的COM端口后确认串行端口设定值

③ 点击[打开]按钮打开端口

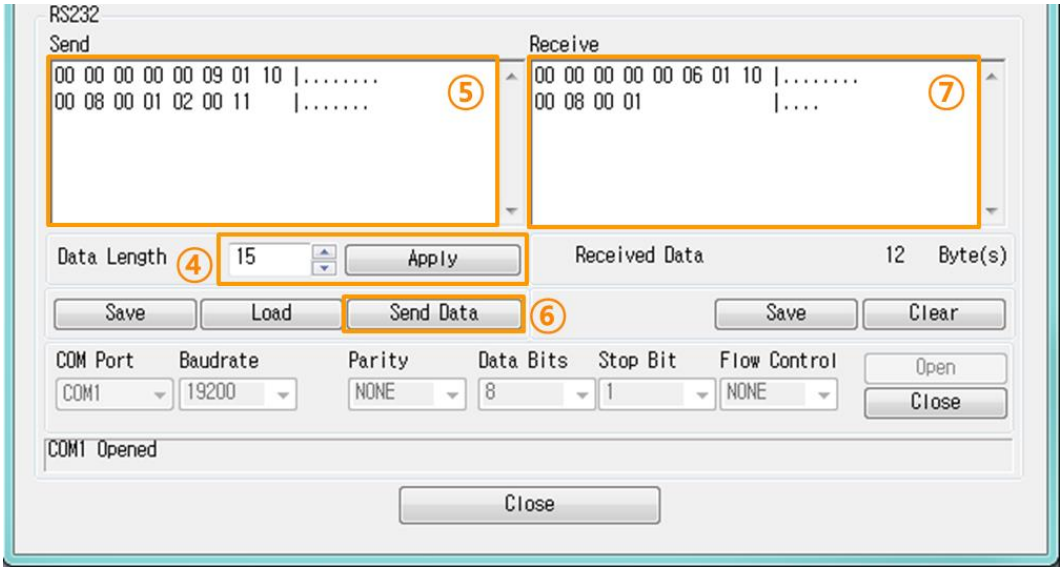


图 7-4 传送测试数据 2

④ 将数据长度设定为15(Bytes)后按[应用]按钮

⑤ 在[预发数据]上输入 write multiple register函数的数据

⑥ 点击[发送数据]按钮

⑦ 查看在[接收数据]上显示的ezTCP应答数据是否与上图一致

☞ ***⑤ 发送的数据是将Slave的0, 4 输出端口ON的Master的命令。(Write Multiple Registers). 因此Slave对其的应答需要发送在⑦显示的数据。***

8 注意事项

- 本资料是基于标准文献，说明了ezTCP支持的有关Modbus/TCP协议的说明。
- 我公司对该资料编写中虽然进行了充分的校验，但在文件内的说明不做任何承诺且会根据需要随时进行变更。
- 更详细的内容请参考 MODBUS Application Protocol Specification (v1.1b3)与 MODBUS Messaging Implementation Guide (v1.0b)。

9 变更履历

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
2015.02.13.	1.3	○ Change the document title ○ Redesign document style ○ Add descriptions for new function codes (FC 1, 2, 4, 5, 6, 7, 15, 43, 43 / 14, 105) ○ Add description about MODBUS data and addressing ○ Most of figures and description have been updated ○ Redesign description structure (delete function code class categorization and etc.)	Roy LEE Andy LEE
2017.08.04.	1.4	○ Fix MODBUS addressing description errors ○ Delete sample code descriptions and fix link errors	Andy LEE