ezTCP Technical Document

# Serial Manager Protocol

Document Version 2.2

2012-11-16

Sollae Systems Co., Ltd.

http://www.sollae.co.kr

# Contents

# 1 Overview

- CSE, CIE, CSW products support to configure its environment variables through the serial port. We called this functionality “Serial Manager”.
- This document explains about “Serial Manager”
- Notations in this document are shown below.

| Types | Notation |
|---|---|
| Hexadecimal | The prefix “0x” is using in front of any number.<br>ex) 32 => 0x20 |
| Binary data | Hexadecimal is using. |
| Plain text | English or numeric characters are using. |
| 0x20 | <SP> |
| 0x0D | <CR> |
| 0x0A | <LF> |

- Some variables use Big-Endian with mentions and the rest of them use Little-Endian.

# 2 Serial Manager Mode

## 2.1 Serial Manager Mode

- You should change ezTCP's operation mode to set environment variables through serial port.
- A process to change operation mode is shown below.

| Product type | Process |
|---|---|
| Embedded | After complete boot-up, input low signal to the ISP- pin for 20ms ~ 1,000ms. |
| Stand-alone | After complete boot-up, press the function button on the side of the ezTCP for 20ms ~ 1,000ms. |

☞ *If you input low signal to the ISP- pin or press function button over 1,000ms then ezTCP's operation mode change for firmware downloadable mode.*

- ezTCP's serial port settings are changed when it works as "Serial Manager".

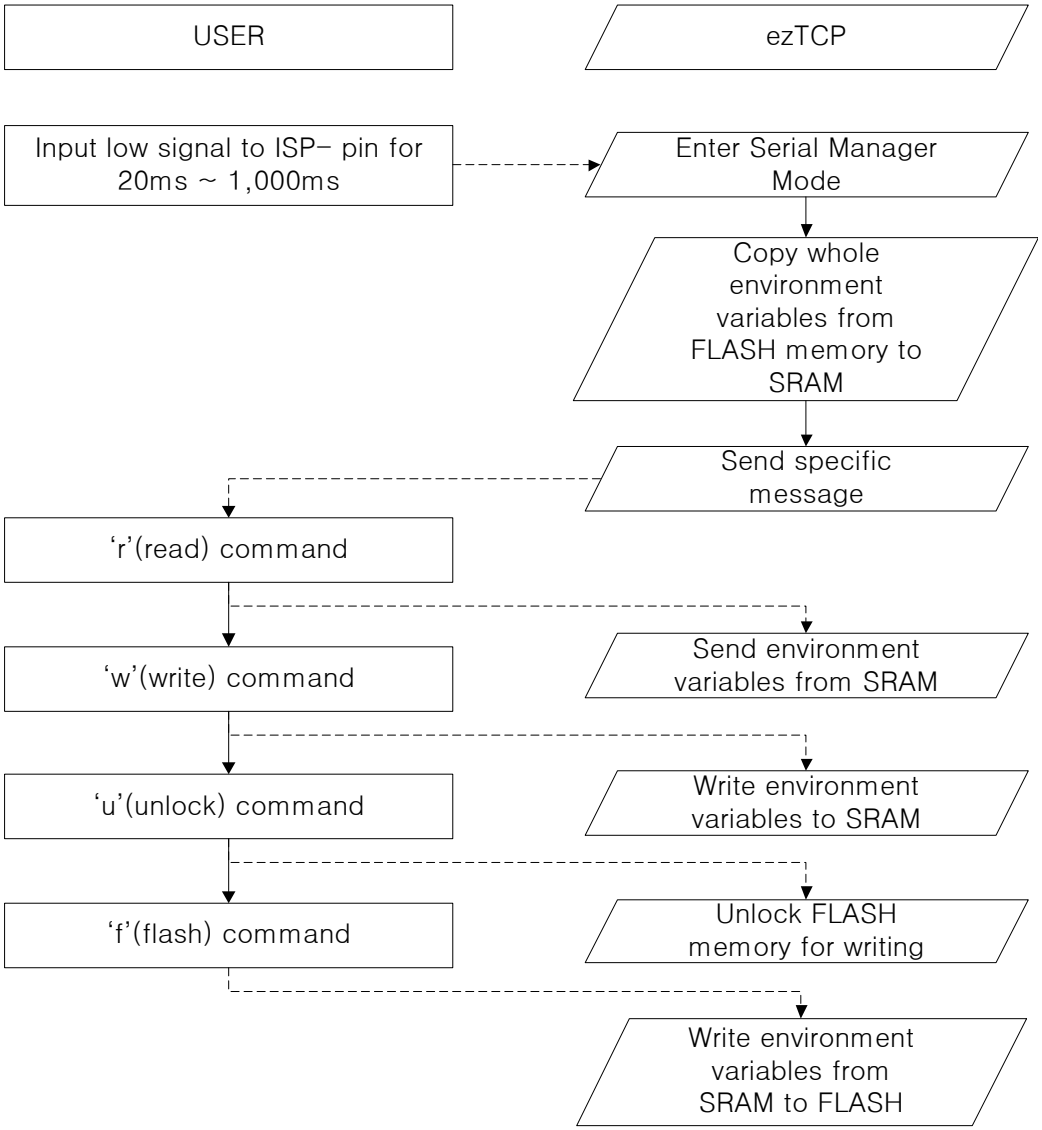
| | |
|---|---|
| Baud Rate | 115,200 bps |
| Data Bit | 8 |
| Stop Bit | 1 |
| Parity | None |
| Flow Control | None |

- ezTCP sends specific message when it starts as "Serial Manager".

| Data | Data Length(byte) | Binary | | | Text |
|---|---|---|---|---|---|
| Message code | 3 | 0x37 | 0x30 | 0x30 | 700 |
| <SP> | 1 | 0x20 | | | |
| Message | N | | | | |
| End code | 2 | 0x0D | 0x0A | | |

☞ *The "Message Code" is very useful when you have to identify entering "Serial Manager".*

## 2.2 Process of Serial Manager



# 3 Serial Manager Protocol

## 3.1 Basic Rule

- ezTCP converts its environment variables(binary data type) into ASCII data type and insert <SP> between each byte. And after last data, <CR> and <LF> are inserted instead of <SP>. <CR><LF> means the end of data stream.
- Here is an example of data conversion. It shows how to convert 2 bytes of binary data to ASCII data.

| Binary Data | | ASCII Data | | | | | Text |
|---|---|---|---|---|---|---|---|
| 0x61 | 0x7f | 0x36 | 0x31 | <SP> | 0x37 | 0x66 | 61<SP>7F |

- It's not case-sensitive.
- A function that converts binary data to ASCII and vice versa is required for supporting Serial Manager Protocol.

## 3.2 Environment Variables

### 3.2.1 Structure of Environment Variables

- The structure of each environment variable is shown below.

| Item | Length(byte) | Content |
|---|---|---|
| Code | 1 | Type of environment variable. |
| ID | 1 | ID of environment variable. |
| Length | 2 | Total length of environment variable from Code to CRC. (**Little-Endian)** |
| Data | N | Environment variable. |
| Pad | 0~N | It inserted in order to make 4-byte data alignment.<br>0x00 is inserted for Pad. |
| CRC | 2 | Code, ID, Length, Data and Pad are used to calculate CRC.<br>**(Little-Endian)** |

- The definition of Code and ID.

| Code | ID | Data Length (byte) | Content |
|---|---|---|---|
| 0x02 | 0x01 | 2 | Product ID. (**Little-Endian)**<br>**Refer to section 3.2.9.** |
| 0x03 | 0x00 | 4 | Local IP Address. (**Big-Endian)** |
| 0x03 | 0x01 | 4 | Subnet Mask. (**Big-Endian)** |
| 0x03 | 0x02 | 4 | Gateway IP Address. (**Big-Endian)** |
| 0x03 | 0x03 | 4 | DNS Server IP Address. (**Big-Endian)** |
| 0x04 | 0x00 | 18 | **IPv6 Address.**<br>Upper 16 bytes are IPv6 address and Lower 2 bytes are subnet prefix length. |
| 0x04 | 0x02 | 16 | Gateway IPv6 Address. |
| 0x04 | 0x03 | 16 | DNS Server IPv6 Address. |
| 0x04 | 0x10~ | 16 | Peer host's IPv6 address for UART. |
| 0x05 | 0x00 | 4 | ezTCP's Option.<br>**Refer to section 3.2.2.** |
| 0x06 | 0x00 | 10 | Current password. Currently, **8 bytes are used**. |
| 0x07 | 0x00 | 64 | Comment of product. |
| 0x07 | 0x01~0x08 | 16 | Comment of Digital Input Port.<br>Each digital input port has 16 bytes for its comment.<br>**This is only valid for I/O product.** |
| 0x07 | 0x09~0x10 | 16 | Comment of Digital Output Port.<br>Each digital output port has 16 bytes for its comment.<br>**This is only valid for I/O product.** |
| 0x08 | 0x00 | 48 | ID and password for PPPoE or EAPoL.<br>Preceded 32-byte is ID and next 16-byte is password. |
| 0x08 | 0x01 | 48 | ID and password for DDNS.<br>Preceded 32-byte is ID and next 16-byte is password. |
| 0x09 | 0x00 | 4 | Wireless LAN related variables. **Refer to section 3.2.3.** |
| 0x09 | 0x01 | 32 | SSID of Access Point.<br>**It is ASCII data end with 0x00**. So, maximum 31-byte is available. |
| 0x09 | 0x02 | 20 | 64bit WEP Key.<br>It is for 4-set of WEP key and each key's length is 5 bytes. |

| | | | |
|---|---|---|---|
| | | | Preceded 5-byte is WEP key for ID 0, next 5-byte is for ID 1, and so on. |
| 0x09 | 0x03 | 52 | 128bit WEP Key.<br>It is for 4-set of WEP key and each key's length is 13 bytes.<br>Preceded 13-byte is for WEP key for ID 0, next 5-byte is for ID 1, and so on. |
| 0x09 | 0x04 | 32 | WPA passphrase. **(for CSW-H80)**<br>**It is ASCII data end with 0x00**. So, maximum 31-byte is available. You should enter at least 8-byte.<br>**Refer to section 3.2.10.** |
| 0x09 | 0x08 | 64 | WPA passphrase. **(for CSW-M83 / M85)**<br>**It is ASCII data end with 0x00**. So, maximum 63-byte is available. You should enter at least 8-byte.<br>**Refer to section 3.2.10.** |
| 0x09 | 0x05 | 32 | WPA PSK. **(for CSW-H80)**<br>It is calculated by using WPA passphrase and SSID.<br>**Refer to section 3.2.10.** |
| 0x09 | 0x09 | 32 | WPA PSK. **(for CSW-M83 / M85)**<br>It is calculated by using WPA passphrase and SSID.<br>**Refer to section 3.2.10.** |
| 0x10,<br>0x14 | 0x00 ~ | 12 | Hardware configuration of UART.<br>If ezTCP has several UART then each UART is identified by ID. ID zero(0) indicate first UART.<br>**Refer to section 3.2.4.** |
| 0x20,<br>0x2c | 0x00 ~ | 20 | Operating configuration of UART.<br>If ezTCP has several UART then each UART is identified by ID. ID zero(0) indicate first UART.<br>**Refer to section 3.2.5.** |
| 0x21 | 0x00 | 6 | Access Control with MAC Address.<br>User can restrict hosts with MAC address.<br>Refer to products user manual for more detail information. |

| | | | |
|---|---|---|---|
| 0x22 | 0x00 | 8 | Access Control with IP Address.<br>User can restrict hosts with hosts IP address range.<br>Refer to products user manual for more detail information.<br>First 4 bytes are IP address and second 4 bytes are subnet mask.<br>**IP address and subnet mask are stored in by using Big-Endian**. |
| 0x23 | 0x00 | 64 | Peer host DNS name for UART ID zero(0)<br>**It is an ASCII data end with 0x00**. So, maximum 63 bytes are valid.<br>**Refer to section 3.2.5.** |
| 0x23 | 0x01 | 64 | Peer host DNS name for UART ID one(1)<br>**It is an ASCII data end with 0x00**. So, maximum 63 bytes are valid.<br>**Refer to section 3.2.5.** |
| 0x23 | 0x02 | 64 | Peer host DNS name for I/O product.<br>**It is an ASCII data end with 0x00**. So, maximum 63 bytes are valid.<br>**Refer to section 3.2.6.** |
| 0x23 | 0x80 | 64 | Host DNS name for DynDNS service.<br>**It is an ASCII data end with 0x00**. So, maximum 63 bytes are valid.<br>**Refer to section 3.2.7.** |
| 0x23 | 0x81 | 64 | Peer host DNS name for IP Address Notification Function.<br>**It is an ASCII data end with 0x00**. So, maximum 63 bytes are valid.<br>**Refer to section 3.2.7.** |
| 0x24 | 0x00 ~ 0x07 | 32 | Macro for Digital Output Port.<br>Maximum digital output port is 8 and each port has 32 bytes for macro. Each port is identified by ID.<br>**This is only valid for I/O product.** |
| 0x25 | 0x00 | 56 | Operating configuration for I/O product.<br>**Refer to section 3.2.6.** |
| 0x27 | 0x00 | 12 | Operating configuration for IP Address Notification Function.<br>**Refer to section 3.2.7.** |

| | | | |
|---|---|---|---|
| 0x28 | 0x00 | 10 | TCP port number.<br>Currently, you can only change HTTP port number for I/O product.<br>**Refer to section 3.2.8.** |
| 0x2a,<br>0x2d | 0x00 ~ | 6 | UART Separator.<br>ID zero(0) indicate first UART.<br>**Refer to section 3.2.11.** |

### 3.2.2 ezTCPs Option

| Code | ID |
|---|---|
| 0x05 | 0x00 |

- The total size is 4-byte and structure is shown below.

```
struct opt_env
{
        unsigned int ezcfg_lock      : 1;
        unsigned int rcfg            : 1;
        unsigned int arp             : 1;
        unsigned int dhcp            : 1;
        unsigned int pppoe           : 1;
        unsigned int auto_ns         : 1;
        unsigned int ip6             : 1;
        unsigned int ip6_eui         : 2;
        unsigned int ip6_gua         : 2;
        unsigned int pad1            : 3;       // NOT USED. Never modify.

        unsigned int debug           : 1;
        unsigned int telnet          : 1;
        unsigned int ssl             : 1;
        unsigned int ssh             : 1;
        unsigned int http            : 1;
        unsigned int ddns            : 3;
        unsigned int t2smc           : 1;
        unsigned int secure          : 1;       // READ ONLY. Never modify.
        unsigned int mac_id          : 1;
```

```
        unsigned int ps         : 1;    // NOT USED. Never modify.
        unsigned int pd         : 1;    // NOT USED. Never modify.
        unsigned int pad3       : 3;    // NOT USED. Never modify.
};
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| ezcfg_lock | rcfg | arp | dhcp | pppoe | auto_ns | pad0 | |

| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|
| ip6 | ip6_eui | | ip6_gua | | pad1 | | |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|
| debug | telnet | ssl | ssh | http | ddns | | |

| 24 | 25 | 26 | 27 | 28 | 29 | 31 | 31 |
|---|---|---|---|---|---|---|---|
| t2smc | secure | mac_id | ps | pd | pad3 | | |

- ezcfg_lock

  [in/out]

  If this parameter is one(1), the ezTCP only replies(with ezManager program) to the host that is defined in the [Code:0x21, ID: 0x00] or [Code:0x22, ID:0x00].

- rcfg

  [in/out]

  Enable / disable the remote configuration function.

- arp

  [in/out]

  If this parameter is set, you can temporary set ezTCP's IP address by using ARP packet.

- dhcp

  [in/out]

  If this parameter is non-zero, dhcp protocol is enabled.

- pppoe

  [in/out]

  If this parameter is non-zero, pppoe protocol is enabled.

- auto_ns

[in/out]

If ezTCP's IP address is set as a dynamic IP ( DHCP or PPPoE ), it will automatically receive DNS server address. If this parameter is not set, the IP address designated in the DNS Server IP Address[Code:0x03, ID:0x03] will be used as a DNS server address.

- ip6

  [in/out]

  Enable / Disable IP6 function.

- ip6_eui

  [in/out]

  The option to make Link-Local IP6 address.

| ip6_eui | Comment |
|---|---|
| 0 | ezTCP's MAC address is using to make Link-Local IP6 address. |
| 1 | Random number is using to make Link-Local IP6 address. |

- ip6_gua

  [in/out]

| ip6_gua | Comment |
|---|---|
| 0 | Obtain an IP Automatically. |
| 1 | Use static IP address. |

- debug

  [in/out]

  Enable/Disable the remote debugging function of ezTCP. If this value is set, it will send a debugging message by using UDP broadcast(port 50006).

- telnet

  [in/out] Enable/Disable Telnet Console.

  If this value is set, the ssh option is disabled.

- ssl

  [in/out]

  Enable / Disable SSL protocol.

- ssh

  [in/out] Enable / Disable SSH protocol

  If this values is set, SSH protocol is required to connect ezTCP's console and telnet option is disabled.

- http

  [in/out] Enable / Disable HTTP protocol for controlling or monitoring.

  This flag is currently considered in below ezTCP products.

| LAN Type | Product Name |
|---|---|
| Wired LAN | CIE-H10, CIE-M10, CIE-H12 |
| Wireless LAN | |

- ddns

  [in/out] Select DDNS service provider.

| ddns | Description |
|---|---|
| 0 | DDNS disabled |
| 1 | Use DynDNS service provider. |
| 2 | TCP |
| 3 | UDP |

☞ *Please see a product user's manual for more detail information.*

- t2smc

  [in/out]

  If this value is set, a ezTCP can accept multiple TCP/IP connection.

  This value is only considered in T2S mode and below ezTCP products.

| LAN Type | Product Name |
|---|---|
| Wired LAN | CSE-M73, CSE-H25 |
| Wireless LAN | |

- secure

  [out]

  **READ ONLY. This value is represented whether this product support SSL and SSH or not.**

- mac_id

  [in/out]

  If this value is set, a ezTCP sends its MAC address to peer when a TCP/IP connection is established.

### 3.2.3 Wireless LAN related variables.

| Code | ID |
|---|---|
| 0x09 | 0x00 |

- The total size is 4-byte and structure is shown below.

```
struct wlan_opt
{
        unsigned int cctype             : 4;
        unsigned int channel            : 4;
        unsigned int wep                : 2;
        unsigned int wep_id             : 2;
        unsigned int pad0               : 1;
        unsigned int bg_scan            : 1;
        unsigned int auth               : 2;
        unsigned int wpa                : 3;
        unsigned int cipher             : 2;
        unsigned int pad1               : 3;    //NOT USED. Never modify
        unsigned int antenna            : 1;
        unsigned int phy                : 3;
        unsigned int short_preamble     : 1;
        unsigned int short_slot         : 1;
        unsigned int cts_protection     : 1;
        unsigned int pad2               : 1;    //NOT USED. Never modify
};
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| cctyp | | | | channel | | | |

| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|
| wep | | wep_id | | pad0 | bg_scan | auth | |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|
| wpa | | | cipher | | pad1 | | |

| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|
| antenna | phy | | | SP | SS | CTS | pad2 |

- cctype

  [in/out] Connection Control Type

| cctype | Description |
|---|---|
| 0 | AD-Hoc |
| 1 | Infrastructure |

- channel

  [in/out] Wireless LAN channel

  It is valid only if cctype is AD-Hoc.

- wep

  [in/out] Encryption Method

| wep | Description |
|---|---|
| 0 | Disable |
| 1 | WEP – 64 bits |
| 2 | WEP – 128 bits |

☞ *WEP(Wired Equivalent Privacy)*

- wep_id

  [in/out] WEP Key index number (0, 1, 2, 3)

  This parameter is used for choosing a WEP Key when “wep” parameter is “1” or “2”.

- auth

  [in/out] Authentication mode for infrastructure network

| auth | Description |
|---|---|
| 0 | Disable |
| 1 | Open System |
| 2 | Shared Key |
| 3 | Both |

- wpa

  [in/out] WPA(Wi-Fi Protected Access) authentication mode

| wpa | Description |
|---|---|
| 0 | Disable |
| 1 | EAP TLS |
| 2 | WPA-PSK |
| 3 | EAP TTLS |
| 4 | WPA2-PSK |
| 5 | PEAP |

- cipher

  [in/out] Encryption Strength

| cipher | Description |
|---|---|
| 0 | Disable |
| 1 | TKIP (Temporal Key Integrity Protocol ) |
| 2 | AES ( Advanced Encryption Standard ) |
| 3 | TKIP / AED |

- antenna

  [in/out]

| antenna | Description |
|---|---|
| 0 | Internal Antenna |
| 1 | External Antenna |

***※ Currently, antenna is only valid for CSW-M85.***

- passive, bg_scan, phy, short_preamble, short_slot, cts_protection

  [in/out]

  These are advanced settings for WIRELESS LAN products.

*Please refer to product user's manual for more detail information.*

### 3.2.4 Hardware configuration of UART

| Code | ID |
|---|---|
| 0x10 | 0x00 ~ |

- The total size is 12-byte and structure is shown below.

```
struct uart_dev_env
{
unsigned int max_stype    : 2;      // RESERVED. Never Modify
unsigned int stype        : 2;
unsigned int databit      : 2;
unsigned int stopbit      : 2;
unsigned int parity       : 2;
unsigned int flowctrl     : 2;
unsigned int telcom       : 2;
unsigned int parity2      : 2;
unsigned int en_ttl       : 1;      // READ ONLY. Never modify.
unsigned int ttl          : 1;
```

```
unsigned int en_tx_delay   : 1;      // READ ONLY. Never modify.
unsigned int tx_delay      : 5;
unsigned int dtrdsr        : 1;
unsigned int pad0          : 1;
unsigned int pad3          : 6;      // NOT USED. Never modify.
DWORD max_baud;                      // READ ONLY. Never modify.
DWORD sio_baud;
};
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| max_stype | | stype | | databit | | stopbit | |

| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|
| parity | | flowctrl | | telcom | | parity2 | |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|
| en_ttl | ttl | en_tx_delay | tx_delay | | | | |

| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|
| dtrdsr | pad0 | pad3 | | | | | |

| $5^{th}$ byte |
|---|
| max_baud(Little-Endian) |

| $6^{th}$ byte |
|---|
| max_baud(Little-Endian) |

| $7^{th}$ byte |
|---|
| max_baud(Little-Endian) |

| $8^{th}$ byte |
|---|
| max_baud(Little-Endian) |

| $9^{th}$ byte |
|---|
| sio_baud(Little-Endian) |

| 10th byte |
| --- |
| sio _baud(Little-Endian) |

| 11th byte |
| --- |
| sio _baud(Little-Endian) |

| 12th byte |
| --- |
| sio _baud(Little-Endian) |

- stype

  [in/out] Serial port type

| stype | Description |
| --- | --- |
| 0 | RS-232 |
| 1 | RS-485 |
| 2 | RS-422 |

- databit

  [in/out] Serial Data Bit

| databit | Description |
| --- | --- |
| 0 | 5-Bit |
| 1 | 6-Bit |
| 2 | 7-Bit |
| 3 | 8-Bit |

- stopbit

  [in/out] Stop Bit

| stopbit | Description |
| --- | --- |
| 0 | 1 Stop Bit |
| 1 | 1.5 Stop Bit |
| 2 | 2 Stop Bit |

- parity

  [in/out] Serial Parity Bit

| parity | Description |
| --- | --- |
| 0 | None |
| 1 | Even |
| 2 | Odd |
| 3 | Use parity2 parameter |

- flowctrl

  [in/out] Serial Flow Control

| stype | Description |
|---|---|
| 0 | None |
| 1 | RTS / CTS |
| 2 | Xon / Xoff |

- telcom

  [in/out]

  Enable / Disable Telnet COM Port Control Option ( RFC2217 )

  Please see a product user's manual for more detail information.

- parity2

  [in/out] Serial Parity Bit

| parity | Description |
|---|---|
| 0 | Mark |
| 1 | Space |

- en_ttl

  [out]

  **READ ONLY. This value is represented that it supports TTL level output through its UART.**

  This value is only considered below ezTCP products.

| LAN Type | Product Name |
|---|---|
| Wired LAN | CSE-M73(H/W version 1.3, F/W version 1.4a or higher) |
| Wireless LAN | |

- ttl

  [in/out]

  Enable / disable TTL level output.

  This value is only considered below ezTCP products.

| LAN Type | Product Name |
|---|---|
| Wired LAN | CSE-M73(H/W version 1.3, F/W version 1.4a or higher) |
| Wireless LAN | |

- en_tx_delay

  [out]

  **READ ONLY. This value is represented that it supports delayed transmission through its UART.**

- tx_delay

  [in/out]

Please see a product user's manual for more detail information.

- dtrdsr

  [in/out] DTR/DSR flow control

- max_baud

  [out] Maximum Serial Baud Rate

  **READ ONLY. The max_baud is Maximum Serial Baud Rate of ezTCP.**

- sio_baud

  [in/out] Serial Baud Rate

  **Do not exceed the max_baud.**

### 3.2.5 Operating configuration of UART

| Code | ID |
|---|---|
| 0x20 | 0x00 ~ |

- The total size is 20-byte and structure is shown below.

```
struct uart_var_env
{
_u32 local_ip;              // NOT USED. Never modify.
_u32 peer_ip;
_u16 local_port;
_u16 peer_port;
_u8 mux_type;
_u8 no_delay    :1;
_u8 cod_listen  :1;         // NOT USED. Never modify.
_u8 secure      :2;
_u8 pad         :4;         // NOT USED. Never modify.
_u16 water_mark;
_u16 time_mark;
_u16 timeout;
};
```

| 1st byte |
|---|
| local_ip(Big-Endian) |

| 2nd byte |
| --- |
| local_ip(Big-Endian) |

| 3rd byte |
| --- |
| local_ip(Big-Endian) |

| 4th byte |
| --- |
| local_ip(Big-Endian) |

| 5th byte |
| --- |
| peer_ip(Big-Endian) |

| 6th byte |
| --- |
| peer_ip(Big-Endian) |

| 7th byte |
| --- |
| peer_ip(Big-Endian) |

| 8th byte |
| --- |
| peer_ip(Big-Endian) |

| 9th byte |
| --- |
| local_port(Little-Endian) |

| 10th byte |
| --- |
| local_port(Little-Endian) |

| 11th byte |
| --- |
| peer_port(Little-Endian) |

| 12th byte |
| --- |
| peer_port(Little-Endian) |

| 13th byte | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| mux_type | | | | | | | |

| 14th byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| no_delay | cod_listen | secure | | pad | | | |

| 15th byte |
|---|
| water_mark(Little-Endian) |

| 16th byte |
|---|
| water_mark(Little-Endian) |

| 17th byte |
|---|
| time_mark(Little-Endian) |

| 18th byte |
|---|
| time_mark(Little-Endian) |

| 19th byte |
|---|
| timeout(Little-Endian) |

| 20th byte |
|---|
| timeout(Little-Endian) |

- peer_ip

  [in/out] IP address of target host

  It is available when the “mux_type” is COD(2) or U2S(3).

  If you want to use DNS host name[Code:0x23, ID:0x00] then you should set it to zero(0).

- local_port

  [in/out] Local port number

  It is available when the “mux_type” is T2S(0) or U2S(3).

- peer_port

  [in/out] Port number of target host

  It is available when the “mux_type” is COD(2) or U2S(3).

- mux_type

  [in/out] Mode setting value

| mux_type | Mode | Description |
|---|---|---|
| 0 | T2S | TCP Server mode. ezTCP will wait for a TCP/IP connection. |

| | | |
|---|---|---|
| 1 | ATC | AT command mode. ezTCP can be a TCP sever or TCP client mode by using AT commands. |
| 2 | COD | TCP Client mode. ezTCP will connect to specified peer IP address and peer port, when amount of water mark data on serial port is arrived. |
| 3 | U2S | UDP mode. ezTCP will use UDP. |

- no_delay

  [in/out]

  Enable / Disable TCP transmission delay.

- water_mark

  [in/out] Amount of data that can be allowed to start connection (Unit: byte)

  This value is only considered in COD or U2S mode.

- time_mark

  [in/out]

  When ezTCP sends data from its serial port to the Ethernet, the "time_mark" is a unit between two packets. If there is no data from its serial port during the specified "time_mark", the ezTCP sends data to Ethernet.

  **※ Unit : 10ms, minimum is 4(40ms)**

- timeout

  [in/out] Time-out value

  In T2S, COD and ATC mode, after time-out value seconds without communication, ezTCP will disconnect the connection automatically. If this value is zero(0), ezTCP will not disconnect automatically. In the other mode, this value can be set for customization

  **※ Unit : Second**

### 3.2.6 Operating configuration for I/O product

| Code | ID |
|---|---|
| 0x25 | 0x00 |

- The total size is 56-byte and structure is shown below.

```
struct io_var_env
{
        unsigned int   odbus            : 1;
        unsigned int macro              : 1;
        unsigned int master             : 1;
```

```
        unsigned int active         : 1;
        unsigned int notify         : 1;
        unsigned int conns          : 3;
        unsigned int emacro         : 8;
        unsigned int query          : 1;
        unsigned int ctrl           : 1;
        unsigned int pad0           : 6;    // NOT USED. Never modify
        unsigned int pad1           : 8;    // NOT USED. Never modify
        _u32 peer_ip;
        _u16 peer_port;
        _u16 slave_id;
        _u16 input_addr;
        _u16 output_addr;
        _u32 init_output;
        _u32 poll_interval;
        _u16 input_valid_time[8];
        _u16 output_delay[8];
};
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| modbus | macro | master | active | notify | conns | | |

| 2nd byte |
|---|
| emacro |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|
| query | ctrl | pad0 | | | | | |

| 4th byte |
|---|
| pad1 |

| 5th byte |
|---|
| peer_ip(Big-Endian) |

| 6th byte |
|---|
| peer_ip(Big-Endian) |

| 7th byte |
| --- |
| peer_ip(Big-Endian) |

| 8th byte |
| --- |
| peer_ip(Big-Endian) |

| 9th byte |
| --- |
| peer_port(Little-Endian) |

| 10th byte |
| --- |
| peer_port(Little-Endian) |

| 11th byte |
| --- |
| slave_id(Little-Endian) |

| 12th byte |
| --- |
| slave_id(Little-Endian) |

| 13th byte |
| --- |
| input_addr(Little-Endian) |

| 14th byte |
| --- |
| input_addr(Little-Endian) |

| 15th byte |
| --- |
| output_addr(Little-Endian) |

| 16th byte |
| --- |
| output_addr(Little-Endian) |

| 17th byte | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| init_output(Little-Endian) | | | | | | | |
| port 0 | port 1 | port 2 | port 3 | port 4 | port 5 | port 6 | port 7 |

| 18th byte |
| --- |
| init_output(Little-Endian) |

| 19th byte |
| --- |
| init_output(Little-Endian) |

| 20th byte |
| --- |
| init_output(Little-Endian) |

| 21st byte |
| --- |
| poll_interval(Little-Endian) |

| 22nd byte |
| --- |
| poll_interval(Little-Endian) |

| 23rd byte |
| --- |
| poll_interval(Little-Endian) |

| 24th byte |
| --- |
| poll_interval(Little-Endian) |

| 25th byte |
| --- |
| input_valid_time (Little-Endian) : port 0 |

| 26th byte |
| --- |
| input_valid_time (Little-Endian) : port 0 |

| 27th byte |
| --- |
| input_valid_time (Little-Endian) : port 1 |

| 28th byte |
| --- |
| 21input_valid_time (Little-Endian) : port 1 |

| 29th byte |
| --- |
| input_valid_time (Little-Endian) : port 2 |

| 30th byte |
| --- |
| input_valid_time (Little-Endian) : port 2 |

| 31st byte |
| --- |
| input_valid_time (Little-Endian) : port 3 |

| 32nd byte |
| --- |
| input_valid_time (Little-Endian) : port 3 |

| 33rd byte |
| --- |
| input_valid_time (Little-Endian) : port 4 |

| 34th byte |
| --- |
| input_valid_time (Little-Endian) : port 4 |

| 35th byte |
| --- |
| input_valid_time (Little-Endian) : port 5 |

| 36th byte |
| --- |
| input_valid_time (Little-Endian) : port 5 |

| 37th byte |
| --- |
| input_valid_time (Little-Endian) : port 6 |

| 38th byte |
| --- |
| input_valid_time (Little-Endian) : port 6 |

| 39th byte |
| --- |
| input_valid_time (Little-Endian) : port 7 |

| 40th byte |
| --- |
| input_valid_time (Little-Endian) : port 7 |

| 41st byte |
| --- |
| output_delay (Little-Endian) : port 0 |

| 42nd byte |
| --- |
| output_delay (Little-Endian) : port 0 |

| 43rd byte |
| --- |
| output_delay (Little-Endian) : port 1 |

| 44th byte |
| --- |
| output_delay (Little-Endian) : port 1 |

| 45th byte |
| --- |
| output_delay (Little-Endian) : port 2 |

| 46th byte |
| --- |
| output_delay (Little-Endian) : port 2 |

| 47th byte |
| --- |
| output_delay (Little-Endian) : port 3 |

| 48th byte |
| --- |
| output_delay (Little-Endian) : port 3 |

| 49th byte |
| --- |
| output_delay (Little-Endian) : port 4 |

| 50th byte |
| --- |
| output_delay (Little-Endian) : port 4 |

| 51st byte |
| --- |
| output_delay (Little-Endian) : port 5 |

| 52nd byte |
| --- |
| output_delay (Little-Endian) : port 5 |

| 53rd byte |
| --- |
| output_delay (Little-Endian) : port 6 |

| 54th byte |
| --- |
| output_delay (Little-Endian) : port 6 |

| 55th byte |
| --- |
| output_delay (Little-Endian) : port 7 |

| 56th byte |
| --- |
| output_delay (Little-Endian) : port 7 |

- odbus

  [in/out] Enable / Disable Modbus/TCP protocol.

- macro

  [in/out] Enable / Disable MACRO function.

- master

  [in/out]

| master | Description |
| --- | --- |
| 0 | Slave |
| 1 | Master |

- active

  [in/out]

| active | Description |
| --- | --- |
| 0 | Passive ( TCP Server ) |
| 1 | Active ( TCP Client ) |

- notify

  [in/out]

  If this value is one(1) then I/O product will send “Read Multiple response” packet to a master when its input port state is changed without receiving queries from a master.

- conns

  [in/out] Number of TCP/IP connections for Modbus/TCP

  If odbus option is one(1) then this option represents the total number of TCP/IP connections for Modbus/TCP. The maximum value is 8.

  * **F/W version 1.3F or higher.**

- emacro

  [in/out] Enable / Disable MACRO function for each output port.

  0th bit(LSB) means 0th output port, and the value is one(1) then this function is enabled.

- query

  [in/out]

If master option value is one(1) (Modbus/TCP Master) then query is used for Modbus command type.

| query | Description |
|---|---|
| 0 | **FC 16(Multiple)**<br>The product controls the output ports and monitors the input ports of slaves with WORD unit by FC 16 ( write multiple register ) and FC 03 ( read multiple register ). |
| 1 | **FC 05(Single)**<br>The product controls the output ports and monitors the input ports of slaves with BIT unit by FC 05 ( write coil ) and FC 02 ( read input discretes ) |

- ctrl

  [in/out]

  If master option value is one(1) (Modbus/TCP Master) then ctrl is used for how to control its output ports.

| ctrl | Description |
|---|---|
| 0 | AND |
| 1 | OR |

- peer_ip

  [in/out] Target host IP address.

  If active option value is one(1) then peer_ip is used for a target host IP address.

  If you want to use DNS host name[Code:0x23, ID:0x02] then you should set it to zero(0).

- peer_port

  [in/out]

  If active option value is one(1) then peer_port is used for target host's port number. Otherwise, it is used for local port number.

- slave_id

  [in/out]

| master | Description |
|---|---|
| 0 – Slave Mode | Product's Unit ID. |
| 1 – Master Mode | Remote device's Unit ID. |

- input_addr

  [in/out]

| master | Description |
|---|---|
| 0 – Slave Mode | Product's input port address. |
| 1 – Master Mode | Slave device's input port address. |

☞ *Input port address and output port address must differ by a number lager than 8.*

- output_addr

  [in/out]

| master | Description |
|---|---|
| 0 – Slave Mode | Product's output port address. |
| 1 – Master Mode | Slave device's output port address. |

☞ *Input port address and Output port address must differ by a number lager than 8.*

- init_output

  [in/out]

  Output port value when product is booted

  "1" then output port is on, "0" then output port is off.

- poll_interval

  [in/out]

  Interval in millisecond between each query to a master

- input_valid_time

  [in/out]

  Refer to product user's manual for detail information.

- output_delay

  [in/out]

  Refer to product user's manual for detail information.

### 3.2.7 IP Address Notification Function

| Code | ID |
|---|---|
| 0x27 | 0x00 |

- The total size is 12-byte and structure is shown below.

```
struct ip_trap_env
{
        unsigned int level   : 3;
        unsigned int pad0    : 13;      // NOT USED. Never modify
        unsigned int pad1    : 16;      // NOT USED. Never modify
        _u32 peer_ip;
        _u16 peer_port;
```

```
        _u16 interval;
};
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| level | | | pad0 | | | | |

| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|
| pad0 | | | | | | | |
| 3rd byte | | | | | | | |
| pad1 | | | | | | | |

| 4th byte |
|---|
| pad1 |

| 5th byte |
|---|
| peer_ip(Big-Endian) |

| 6th byte |
|---|
| peer_ip(Big-Endian) |

| 7th byte |
|---|
| peer_ip(Big-Endian) |

| 8th byte |
|---|
| peer_ip(Big-Endian) |

| 9th byte |
|---|
| peer_port(Little-Endian) |

| 10th byte |
|---|
| peer_port(Little-Endian) |

| 11th byte |
|---|
| interval_port(Little-Endian) |

| 12th byte |
|---|

| interval_port(Little-Endian) |
|---|

- level

  [in/out]

  If "ddns" parameter in section 3.2.2. is 2(TCP) or 3(UDP) then it decides the type of data.

| level | Description |
|---|---|
| 0 | ASCII |
| 1 | Binary |

- peer_ip

  [in/out] The host IP address to send a data.

  If you want to use DNS host name then you should set it to zero(0).

  If "ddns" parameter in section 3.2.2. is 1(DDNS) then [Code:0x23, ID:0x80] will be used for DNS host name.

  If "ddns" parameter in section 3.2.2. is 2(TCP) or 3(UDP) then [Code:0x23, ID:0x81] will be used for DNS host name.

- peer_port

  [in/out] The host port number to send a data

- interval

  [in/out] The interval for sending a data

  It is valid when "ddns" parameter in section 3.2.2. is 2(TCP) or 3(UDP).

  ※ **Unit : minute**

### 3.2.8 TCP port number

| Code | ID |
|---|---|
| 0x28 | 0x00 |

- Currently, you can only change HTTP port number for I/O product.
- The total size is 10-byte and structure is shown below.

```
struct port_map_env
{
        _u16 http_port;
        _u16 reserved1;      // REVERVED. Never modify
```

```
        _u16 reserved2;     // REVERVED. Never modify
        _u16 reserved3;     // REVERVED. Never modify
        _u16 reserved4;     // REVERVED. Never modify
};
```

| 1st byte |
| --- |
| http_port(Little-Endian) |

| 2nd byte |
| --- |
| http_port(Little-Endian) |

| 3rd byte |
| --- |
| - |

| 4th byte |
| --- |
| - |

| 5th byte |
| --- |
| - |

| 6th byte |
| --- |
| - |

| 7th byte |
| --- |
| - |

| 8th byte |
| --- |
| - |

| 9th byte |
| --- |
| - |

| 10th byte |
| --- |
| - |

- http_port

  [in/out] The HTTP port number

  This value is only considered below ezTCP products.

| LAN Type | Product Name |
|---|---|
| Wired LAN | CIE-H10, CIE-M10, CIE-H12 |
| Wireless LAN | |

### 3.2.9 Product ID

| Code | ID |
|---|---|
| 0x02 | 0x01 |

- Product names are shown below.

| Product ID | Product Name |
|---|---|
| 0x20 | CIE-H10 |
| 0x21 | CIE-M10 |
| 0x23 | CSE-M32 |
| 0x24 | CSE-H20 |
| 0x25 | CSE-H21 |
| 0x26 | CSE-M73 |
| 0x27 | CSW-H80 |
| 0x29 | CSE-H25 |
| 0x2b | CSE-M53 |
| 0x2c | CSE-H53 |
| 0x2d | CSW-M83 |
| 0x2e | CSW-M85 |
| 0x2f | CSE-H55 |
| 0x30 | CSC-HR2 |
| 0x34 | CIE-H12 |
| 0x35 | CSW-H85 |
| 0x36 | CSE-T32 |
| 0x37 | CSE-M53A |
| 0x39 | CSE-T16 |
| 0x3a | CSE-T48 |
| 0x3b | CSE-H53A |

### 3.2.10 WPA

| Code | ID |
|---|---|
| 0x09 | 0x04 |

- WPA passphrase
  - **It's only for CSW-H80.**
  - WPA passphrase used to make WPA PSK. Its length is 32-byte.
  - WPA passphrase is ASCII data end with 0x00. So 31-byte is available.
  - A~Z, a~z or 0~9 are using for WPA passphrase.
  - You have to enter at least 8-byte

| Code | ID |
|---|---|
| 0x09 | 0x05 |

- WPA PSK(Wi-Fi Protected Access Pre-shared Key)
  - **It's only for CSW-H80**

    The SSID [ Code:0x09, ID:0x01 ] and WPA passphrase [ Code:0x09, ID:0x04 ] are used to make WPA PSK. You can make it by using PSK generate function.(refer to section 3.8.).
  - CSW-H80 needs at least 30 seconds to calculate WPA PSK with its CPU.
  - You can try to make WPA PSK with http://www.wireshark.org/tools/wpa-psk.html

| Code | ID |
|---|---|
| 0x09 | 0x08 |

- WPA passphrase
  - **It's only for CSW-M83 / M85.**
  - WPA passphrase used to make WPA PSK. Its length is 64-byte.
  - WPA passphrase is ASCII data end with 0x00. So 63-byte is available.
  - A~Z, a~z or 0~9 are using for WPA passphrase.
  - You have to enter at least 8-byte

| Code | ID |
|---|---|
| 0x09 | 0x09 |

- WPA PSK(Wi-Fi Protected Access Pre-shared Key)
  - **It's only for CSW-M83 / M85**
  - The SSID [ Code:0x09, ID:0x01 ] and WPA passphrase [ Code:0x09, ID:0x08 ] are used to make

WPA PSK. You can make it by using PSK generate function.(refer to section 3.8.)

- CSW-M83 or M85 needs at least 5 seconds to calculate WPA PSK with its CPU.
- You can try to make WPA PSK with http://www.wireshark.org/tools/wpa-psk.html

### 3.2.11 UART Separator

| Code | ID |
|---|---|
| 0x2a, 0x2d | 0x00 ~ |

- The total size is 10-byte and structure is shown below.

| 1st byte |
|---|
| Separator #0 |

| 2nd byte |
|---|
| Separator #1 |

| 3rd byte |
|---|
| Separator #2 |

| 4st byte |
|---|
| Separator #3 |

| 5st byte | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Length of separator | | | | Not used. | | | |

| 6st 바이트 | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Not used. | | | | Separator operation | | | |

- Separator operation

| Separator Operation | Comment |
|---|---|
| 0 | Transmit Seperators. |
| 1 | Transmit Seperators + 1 Byte. |

| | |
|---|---|
| 2 | Transmit Seperators + 2 Bytes. |

## 3.3 Read Command

### 3.3.1 Read Command

- The character ‘r’ or ‘R’ is using for read command. The code and id that is you want to read follow read command.
- The structure of read command is shown below

| Data | Data Length(byte) | Value |
|---|---|---|
| Read Command | 1 | r or R |
| Space | 1 | <SP> |
| Code | 2 | Code of environment variable |
| Space | 1 | <SP> |
| ID | 2 | ID of environment variable |
| End of command | 1 | <CR> |

- Ex) Read command for reading product id. [Code:0x02, ID:0x01]

| Read command | Space | Code | | Space | ID | | End of command |
|---|---|---|---|---|---|---|---|
| r or R | <SP> | 0 | 2 | <SP> | 0 | 1 | <CR> |
| 0x72 or 0x52 | 0x20 | 0x30 | 0x32 | 0x20 | 0x30 | 0x31 | 0x0D |

### 3.3.2 Response of Read Command

- There are three types of responses for read command. A response is composed response code and end code of response.

| Data | Data Length(byte) | Value | Comment |
|---|---|---|---|
| Response code | 3 | 701 | Read command has been successfully completed. |
| | 3 | 801 | Read command has failed. |
| | 3 | 802 | Read command has been successfully completed.<br>But, requested Code or ID is not available. |
| | 3 | 809 | Command syntax error. |
| End code | 2 | <CR><LF> | |

- Requested environment variable follows end code of response when the response code is 701.

- Ex) Response data for reading product id.[Code:0x02, ID:0x01]

```
37 30 31 0D 0A 30 32 20 30 31 20 30 38 20 30 30   701..02 01 08 00
20 32 45 20 30 30 20 30 35 20 42 30 0D 0A          2E 00 05 B0..
```

| Data | Data Length(byte) | Value | Comment |
|---|---|---|---|
| Response code | 3 | 701 | Read command has been successfully completed. |
| End code | 2 | <CR><LF> | End of response |
| Environment variable | 23 | 02<SP>01<SP>08<SP>00 <SP>2E<SP>00<SP>05<SP>B0 | Product ID |
| End code | 2 | <CR><LF> | End of environment variable |

## 3.4 Write Command

### 3.4.1 Write Command

- All environment variables are stored in FLASH memory. And temporally use SARM during Serial Manager Process.
- You have to use three types of write command to complete writing process.

| Command | Comment |
|---|---|
| w or W | Temporally save a received data to SRAM. |
| u or U | Unlock FLASH memory to write a data from SRAM. |
| f or F | Write a data from SRAM to FALSH. |

☞ ***You have to send 'u' or 'U' command to unlock FLASH memory before sending 'f' or 'F' command.***

☞ ***You have to check response code for 'f' or 'F' command. When you reset or cut the power before receiving response code, it leads crash on FLASH memory. If then, ezTCP won't work properly.***

- w or W command structure.

| Data | Data Length(byte) | Value |
|---|---|---|
| Command | 1 | w or W |
| Space | 1 | <SP> |
| Code | 2 | Code of environment variable |
| Space | 1 | <SP> |

| | | |
|---|---|---|
| ID | 2 | ID of environment variable |
| Space | 1 | <SP> |
| Contents | N | Environment variable<br>Refer to section 3.2. |
| End Code | 1 | <CR> |

- Ex) Write IP address 192.168.0.168 to Local IP Address. [Code:0x03, ID:0x00]

| Data | Data Length(byte) | Value |
|---|---|---|
| Command | 1 | w or W |
| Space | 1 | <SP> |
| Code | 2 | 03 |
| Space | 1 | <SP> |
| ID | 2 | 00 |
| Space | 1 | <SP> |
| Environment Variable | 35 | 03<SP>00<SP>0C<SP>00<SP><br>C0<SP>A8<SP>00<SP>A8<SP><br>00<SP>00<SP>C5<SP>A6 |
| End Code | 1 | <CR> |

The meaning of environment variable is shown below.

| Data | Comment |
|---|---|
| 03 | Code |
| 00 | ID |
| 0C 00 | Length (Little Endian) : 12 |
| C0 A8 00 A8 | IP Address (Big Endian) : 192.168.0.168 |
| 00 00 | Pad |
| C5 A6 | CRC |

- u or U command structure

| Data | Data Length(byte) | Value |
|---|---|---|
| Command | 1 | u or U |
| Space | 1 | <SP> |
| Unlock Code | 9 | aa55<SP>cc33 |
| End Code | 1 | <CR> |

☞ ***The Unlock code is always aa55<SP>cc33. If you use wrong unlock code then data should not be stored in FLASH memory.***

- f or F command structure

| Data | Data Length(byte) | Value |
|---|---|---|

| | | |
|---|---|---|
| Command | 1 | f or F |
| Space | 1 | <SP> |
| Code | 4 | a5c3 |
| End Code | 1 | <CR> |

☞ ***The Code is always a5c3. If you use wrong code then data should not be stored in FLASH memory***.

☞ ***You have to check response code for 'f' or 'F' command. When you reset or cut the power before receiving response code, it leads crash on FLASH memory. If then, ezTCP won't work properly.***

### 3.4.2 Response of Write Command

- Responses for 'w' or 'W' command are shown below. A response is composed response code and end code of response.

| Data | Data Length(byte) | Value | Comment |
|---|---|---|---|
| Response code | 3 | 701 | Command has been successfully completed. |
| | 3 | 801 | Command has failed. |
| | 3 | 802 | Command has been successfully completed.<br>But, requested Code or ID is not available. |
| | 3 | 803 | The content (environment variable) of 'w' or 'W' command is not available.<br>It will be caused by wrong CRC. |
| | 3 | 805 | Length of the content (environment variable) is too long. |
| | 3 | 806 | Malformed data in the content(environment variable). |
| | 3 | 807 | Invalid length of data in the content(environment variable). |
| | 3 | 808 | Requested Code or ID is not in the content(environment variable). |
| | 3 | 809 | Command syntax error. |
| End code | 2 | <CR><LF> | |

- Responses for ‘u’ or ‘U’ command are shown below. A response is composed response code and end code of response.

| Data | Data Length(byte) | Value | Comment |
|---|---|---|---|
| Response code | 3 | 701 | Command has been successfully completed. |
| | 3 | 801 | Command has failed. |
| | 3 | 809 | Command syntax error. |
| End code | 2 | <CR><LF> | |

☞ *ezTCP sends 701<CR><LF> even the unlock code is not aa55<SP>cc33. In this case, data wouldn’t be stored properly.*

- Responses for ‘f’ or ‘F’ command are shown below. A response is composed response code and end code of response.

| Data | Data Length(byte) | Value | Comment |
|---|---|---|---|
| Response code | 3 | 701 | Command has been successfully completed. |
| | 3 | 804 | Command has failed. |
| | 3 | 809 | Command syntax error. |
| End code | 2 | <CR><LF> | |

☞ *ezTCP sends 804<CR><LF> when its FLASH memory is locked.*

☞ *You have to check response code for ‘f’ or ‘F’ command. When you reset or cut the power before receiving response code, it leads crash on FLASH memory. If then, ezTCP won’t work properly.*

## 3.5 Echo Command

### 3.5.1 Echo Command

- The character ‘e’ or ‘E’ is using for echo command. A ezTCP sends a data that is received from user when user turn on the echo function.

- The structure of echo command is shown below

| Data | Data Length(byte) | Value |
|---|---|---|
| Command | 1 | e or E |

| | | |
|---|---|---|
| Space | 1 | <SP> |
| Flag | 1 | 0 : Turn Off echo function.<br>1 : Turn On echo function. |
| End code | 1 | <CR> |

- Ex) Turn On echo function.

| Echo Command | Space | Flag | End of command |
|---|---|---|---|
| e or E | <SP> | 1 | <CR> |
| 0x65 or 0x45 | 0x20 | 0x31 | 0x0D |

### 3.5.2 Response of Echo Command

- Responses for echo command are shown below. A response is composed response code and end code of response

| Data | Data Length(byte) | Value | Comment |
|---|---|---|---|
| Response code | 3 | 701 | Command has been successfully completed. |
| | 3 | 801 | Command has failed. |
| | 3 | 809 | Command syntax error. |
| End code | 2 | <CR><LF> | |

## 3.6 Reboot Command

### 3.6.1 Reboot Command

- The character ‘g’ or ‘G’ is using for echo command.
- The structure of reboot command is shown below.

| Data | Data Length(byte) | Value |
|---|---|---|
| Command | 1 | g or G |
| Space | 1 | <SP> |
| Code | 1 | 0 |
| End code | 1 | <CR> |

☞ *ezTCP automatically reboot when it receives reboot command.*

- Ex) Reboot command.

| Reboot Command | Space | Code | End of command |
|---|---|---|---|
| g or G | <SP> | 0 | <CR> |
| 0x67 or 0x47 | 0x20 | 0x30 | 0x0D |

### 3.6.2 Response of Reboot Command

- Responses for reboot command are shown below. A response is composed response code and end code of response.

| Data | Data Length(byte) | Value | Comment |
|---|---|---|---|
| Response code | 3 | 801 | Command has failed. |
| | 3 | 809 | Command syntax error. |
| End code | 2 | <CR><LF> | |

- ezTCP automatically reboot itself without a response when it takes reboot command properly.

## 3.7 Product Information Command

### 3.7.1 Product Information Command

- The character ‘c’ or ‘C’ is using for product information command. You can take product id, MAC address and firmware version information.

- The structure of product information command is shown below.

| Data | Data Length(byte) | Value |
|---|---|---|
| Product Information Command | 1 | c or C |
| Space | 1 | <SP> |
| Code | 1 | 0 |
| End of command | 1 | <CR> |

- Ex) Product Information Command

| Product Information Command | Space | Code | End of command |
|---|---|---|---|
| c or C | <SP> | 0 | <CR> |
| 0x63 or 0x43 | 0x20 | 0x30 | 0x0D |

### 3.7.2 Response of Product Information Command

- Responses for product information command are shown below. A response is composed

response code and end code of response.

| Data | Data Length(byte) | Value | Comment |
|---|---|---|---|
| Response code | 3 | 701 | Command has been successfully completed. |
| | 3 | 801 | Command has failed. |
| | 3 | 809 | Command syntax error. |
| End code | 2 | <CR><LF> | |

- Requested product information follows end code of response when the response code is 701 and end with <CR><LF>.
- The total size of product information is 16-byte.

| Data | Data Length(byte) |
|---|---|
| Product ID | 1 |
| Firmware Version(Major) | 1 |
| Firmware Version(Minor) | 1 |
| Firmware Version(Revision) | 1 |
| Pad | 4 |
| MAC Address | 6 |
| Pad | 2 |

☞ *Revision starts from 0, and 0 means A.*

- Ex) Product Information Command

```
Request: 2009-03-11 오전 11:44:45.534125064
 63 20 30 0D                                        c 0.
Answer: 2009-03-11 오전 11:44:46.081000064 (+0.000000000 seconds)
 37 30 31 0D 0A 31 35 20 30 31 20 30 30 20 30 31    701..15 01 00 01
 20 30 30 20 30 30 20 30 30 20 30 30 20 30 30 20     00 00 00 00 00
 33 30 20 46 39 20 30 41 20 38 30 20 32 45 20 34    30 F9 0A 80 2E 4
 34 20 38 31 0D 0A                                  4 81..
```

| Data | Data Length(byte) | Value | Comment |
|---|---|---|---|
| Response code | 3 | 701 | Command has been successfully completed. |
| End code | 2 | <CR><LF> | End of response. |
| Product ID | 2 | 15<SP> | CSW-H80 |
| Firmware Version | 6 | 01<SP>00<SP>01<SP> | 1.0B |

| | | | |
|---|---|---|---|
| Pad | 8 | 00<SP>00<SP>00<SP>00<SP> | Pad.(Don't care) |
| MAC Address | 12 | 00<SP>30<SP>F9<SP><br>0A<SP>80<SP>2E<SP> | 00:30:F9:0A:80:2E |
| Pad | 3 | 44<SP>81 | Pad.(Don't care) |
| End code | 2 | <CR><LF> | End of product information |

## 3.8 WPA PSK Generate Command

### 3.8.1 WPA PSK Generate Command for CSW-H80

- The character 'p' or 'P' is using for PSK generate command.
  - The SSID [ Code:0x09, ID:0x01 ] and WPA passphrase [ Code:0x09, ID:0x04 ] are used to calculate WPA PSK.
  - The command just returns calculated WPA PSK, it doesn't save it. So you have to write the key with "Write Command".
  - WPA PSK is 32-byte and it needs at least 30 seconds.
- The structure of WPA PSK generate command is shown below.

| Data | Data Length(byte) | Value |
|---|---|---|
| Command | 1 | p or P |
| Space | 1 | <SP> |
| Code | 1 | 0 |
| End of command | 1 | <CR> |

### 3.8.2 Response of WPA PSK Generate Command for CSW-H80

- Responses for WPA PSK generate command are shown below. A response is composed response code and end code.

| Data | Data Length(byte) | Value | Comment |
|---|---|---|---|
| Response code | 3 | 701 | Command has been successfully completed. |
| | 3 | 802 | Command has failed.<br>It is returned when SSID[Code:0x09, ID:0x01] or WPA passphrase[Code:0x09, ID:0x04] is not |

| | | | |
|---|---|---|---|
| | | | in environment variables. |
| | 3 | 809 | Command syntax error. |
| End code | 2 | <CR><LF> | |

☞ *WPA PSK follows end code of response when the response code is 701 and end with <CR><LF>.*

- Ex) WPA PSK Generate Command.

  This example generates WPA PSK with SSID "sollae" and WPA passphrase "12345678901234567890".

```
Request: 2009-03-27 오후 1:33:54.328125064 (+17.3125000000 seconds)
70 20 30 0D                                              p 0.
Answer: 2009-03-27 오후 1:33:55.515625064 (+0.0000000000 seconds)
37 30 31 0D 0A 45 46 20 38 34 20 33 43 20 30 46   701..EF 84 3C 0F
20 34 36 20 35 35 20 34 37 20 42 35 20 37 44 20    46 55 47 B5 7D
41 44 20 45 31 20 35 39 20 30 36 20 31 38 20 37   AD E1 59 06 18 7
38 20 41 37 20 43 42 20 39 32 20 37 45 20 33 42   8 A7 CB 92 7E 3B
20 33 31 20 42 46 20 32 34 20 45 33 20 41 35 20    31 BF 24 E3 A5
33 30 20 37 38 20 32 32 20 46 42 20 39 37 20 30   30 78 22 FB 97 0
38 20 35 41 0D 0A                                 8 5A..
```

| Data | Data Length(byte) | Value | Comment |
|---|---|---|---|
| Response code | 3 | 701 | Command has been successfully completed. |
| End code | 2 | <CR><LF> | End of response |
| WPA PSK | 95 | EF<SP>84<SP>3C<SP>0F<SP> 46<SP>55<SP>47<SP>B5<SP> 7D<SP>AD<SP>E1<SP>59<SP> 06<SP>18<SP>78<SP>A7<SP> CB<SP>92<SP>7E<SP>3B<SP> 31<SP>BF<SP>24<SP>E3<SP> A5<SP>30<SP>78<SP>22<SP> FB<SP>97<SP>08<SP>5A | WPA PSK |
| End code | 2 | <CR><LF> | End of WPA PSK |

### 3.8.3 WPA PSK Generate Command for CSW-M83 / M85

- The character 'p' or 'P' is using for PSK generate command.
  - The SSID [ Code:0x09, ID:0x01 ] and WPA passphrase [ Code:0x09, ID:0x08 ] are used to calculate WPA PSK.

- The command calculates WPA PSK and save it automatically. But it doesn't return the result.
- WPA PSK is 32-byte and it needs at least 5 seconds.

- The structure of WPA PSK generate command is shown below.

| Data | Data Length(byte) | Value |
|---|---|---|
| Command | 1 | p or P |
| Space | 1 | <SP> |
| Code | 1 | 0 |
| End of command | 1 | <CR> |

### 3.8.4 Response of WPA PSK Generate Command for CSW-M83 / M85

- Responses for WPA PSK generate command are shown below. A response is composed response code and end code.

| Data | Data Length(byte) | Value | Comment |
|---|---|---|---|
| Response code | 3 | 701 | Command has been successfully completed. |
| | 3 | 802 | Command has failed.<br>It is returned when SSID[Code:0x09, ID:0x01] or WPA passphrase[Code:0x09, ID:0x08] is not in environment variables. |
| | 3 | 809 | Command syntax error. |
| End code | 2 | <CR><LF> | |

# 4 Reference of Programming

## 4.1 Calculate CRC

- The function to calculate CRC.

```
unsigned short crc16_ccitt(unsigned char *buf, int len)
{
	unsigned long crc = 0xffff0000L;
	int bit, byte;

	for(byte = 0; byte < len + 2; byte++)
	{
		if(byte < len)
			crc |= (buf[byte] << 8);
		for(bit = 0; bit < 8; bit++)
			crc = (crc & 0x80000000L) ? (crc << 1) ^ 0x10210000L : (crc << 1);
	}
	return (unsigned short)(crc >> 16);
}
```

- Data type

| Data Type | Data Length(bit) |
|---|---|
| unsigned short | 16bit |
| unsigned long | 32bit |
| unsigned char | 8bit |

- Function parameters

| Parameter | Direction | Comment |
|---|---|---|
| unsigned char *buf | Input | A pointer to buffer that stores data. |
| int len | Input | The length of data. |

- Return Value

  This function returns CRC(16bit). It is Little-Endian.

## 4.2 Make Serial Manager Data

- Functions are making a data for serial manager protocol.

```
#define ENV_HEAD_SIZE		4

struct data_head
{
	_u8 code;
	_u8 id;
	_u16 len;
};

int get_pad_size(int data_size)
{
	int pad_size = 0;
	int reminder = data_size % 4;

	switch(reminder)
	{
	case 0:
		pad_size = 2;
		break;
	case 1:
		pad_size = 1;
		break;
	case 2:
		pad_size = 0;
		break;
	case 3:
		pad_size = 3;
		break;
	}
	return pad_size;
}
```

```
void make_data(_u8* buf, _u8 code, _u8 id, _u8 *data, int data_size, int *len)
{
//+----0------+-----1-----+----2----+----3----+----------n----------+---- 0~n ----+------2------+
// | CODE   |    ID    |    LENGTH    |        DATA        |     PAD    |    CRC   |
//+------------+-----------+------------------+--------------------+--------------+------------+
        struct data_head head;
        _u16 crc;
        _u8* ptr;

        int pad_size        = 0;
        int len1            = 0;

        ptr = buf;

        head.code           = code;
        head.id             = id;

        pad_size  = get_pad_size(data_size);
        head.len  = ENV_HEAD_SIZE + data_size + pad_size + 2;

        // head
        memcpy(ptr, &head, ENV_HEAD_SIZE);
        ptr         += ENV_HEAD_SIZE;
        len1        += ENV_HEAD_SIZE;

        // data
        if ( data_size != 0 )
        {
                memcpy(ptr, data, data_size);
                ptr         += data_size;
                len1        += data_size;
        }

        // pad
        if ( pad_size != 0 )
        {
                for ( int I = 0 ; I < pad_size ; i++ )
```

```
                {
                        *ptr = 0x00;
                        ptr++;
                        len1++;
                }
        }

        // crc
        crc     = crc16_ccitt(buf, len1);
        memcpy(ptr, &crc, 2);
        ptr     +=2;
        len1    +=2;

        // length
        *len = len1;
}
```

- Data type

| Data Type | Data Length(bit) |
|---|---|
| _u8 | 8bit |
| _u16 | 16bit |
| _u32 | 32bit |
| int | 32bit |

- Function parameters

| Parameter | Direction | Comment |
|---|---|---|
| _u8 *buf | Output | A pointer to buffer that will be stored converted data. |
| _u8 code | Input | Code of environment variable. |
| _u8 id | Input | ID of environment variable. |
| _u8 *data | Input | A pointer to buffer that stores environment data. |
| int data_size | Input | The length of environment data. |
| int *len | Output | A pointer to integer variable that will be stored length of converted data. |

# 5 Precaution

Please be careful when write an environment variable. Wrong information may cause improper behavior of ezTCP.

# 6 History

| Date | Version | Comments | Author |
|---|---|---|---|
| 2009.09.02 | 1.5 | ○ Initial release. | Jack KIM |
| 2009.12.15 | 1.6 | ○ Proofread whole pages. | Jack KIM |
| 2010.04.30 | 1.7 | ○ Entire style has been changed. | Roy LEE |
| 2011.02.07 | 1.8 | ○ Add product IDs. | Jack KIM |
| 2011.08.08 | 1.9 | ○ Add new parameters. | Jack KIM |
| 2012.01.16 | 2.0 | ○ Add response code.<br>○ Add warning message for ‘f’ or ‘F’ command.<br>○ Add new contents. | Jack KIM |
| 2012.08.24 | 2.1 | ○ Add new contents. | Jack KIM |
| 2012.11.16 | 2.2 | ○ passive option has deleted. | Jack KIM |