ezManager Program Library for Visual C++

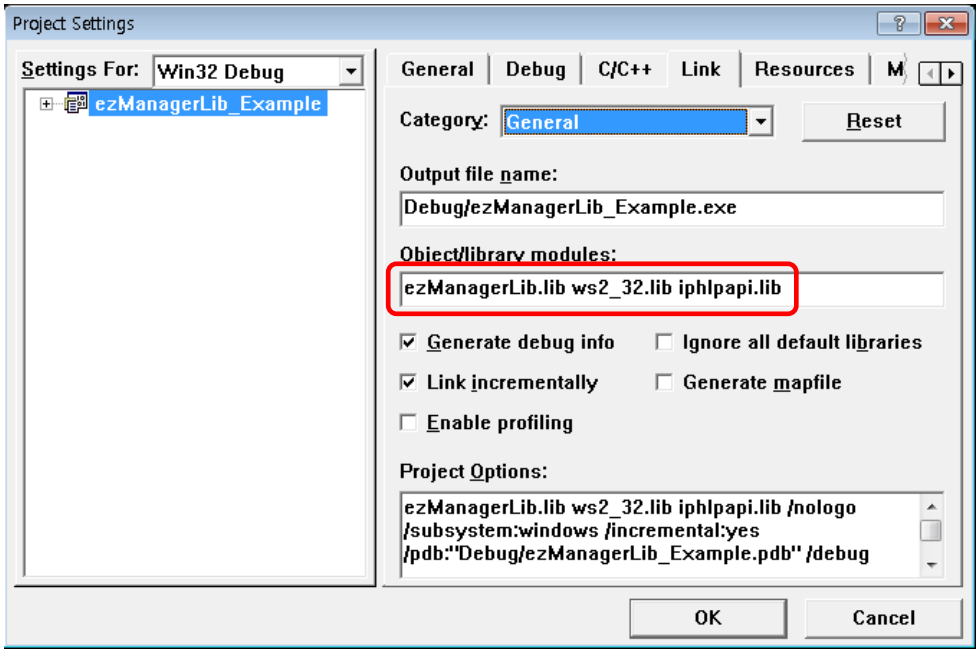
2012-11-16

Sollae Systems Co., Ltd.
http://www.sollae.co.kr

Agenda

1 Introduction

- This library offers main functionality of ezManager program.
- “ezManagerLib.h” is required to include in your source code.
- “ws2_32.lib”, “iphlpapi.lib” and “ezManagerLib.lib” are required.



WARNING:

- All "**RESERVED**" or "**NOT USED**" members of structure are NOT allowed use. Please don’t use “**RESERVED**” or “**NOT USED**” members.
- Please check ezTCP’s MAC Address or IP Address before using “write” function. If you write wrong information to ezTCP, then it may does not work correctly.
- We DO NOT guarantee any damage occurred by illegal use of this library.

2 Data Structures

2.1 lib_env

2.1.1 Overview

- Basic data structure for library functions.

```
#define MAX_COM_PORT        48

struct lib_env
{
        struct net_env          net_env;
        struct ip6_env          ip6_env;
        struct uart_env         uart_env  [MAX_COM_PORT];
        struct opt_env          opt_env;
        struct etc_env          etc_env;
        struct io_env           io_env;
        struct ip_trap_env      ip_trap_env;
        struct port_map_env     port_map_env;
        struct wlan_env         wlan_env;
        struct csc_hr2_env      csc_hr2_env;
};
```

2.1.2 Parameters

- net_env
  [in/out] Basic structure to store network variables.
- ip6_env
  [in/out] Basic structure to store network variables about IP6.
- uart_env
  [in/out] Basic structure to store UART variables.
- opt_env
  [in/out] Basic structure to store environment variables.
- etc_env
  [in/out] Basic structure to store environment variables.
- io_env
  [in/out] Basic structure to store environment variables for I/O product.
- ip_trap_env
  [in/out] Basic structure to store environment variables about IP address notification.
- port_map_env
  [in/out] Basic structure to store TCP port information.
- wlan_env
  [in/out] Basic structure to store WLAN variables.
- csc_hr2_env
  [in/out] Basic structure for CSC-HR2.

2.2 net_env

2.2.1 Overview

- Basic structure to store network variables.

```
struct net_env
{
        _u8     mac_addr        [6];    // READ ONLY. Never modify.
        _u8     secure          [6];    // READ ONLY. Never modify.
        _u16    major;                  // READ ONLY. Never modify.
        _u16    minor;                  // READ ONLY. Never modify.
        _u32    local_ip;
        _u32    net_mask;
        _u32    gate_ip;
        _u8     pppoe_id        [32];
        _u8     pppoe_pwd       [16];
        _u32    socket_ip;              // READ ONLY. Never modify.
        _u32    dns_ip;
        _u16    product_id_old;         // READ ONLY. Never modify.
        _u16    product_id_new;         // READ ONLY. Never modify.
        _u8     ddns_id         [32];
        _u8     ddns_pwd        [16];
        _u8     ddns_host_name  [64];
        _u8     ssh_id          [16];
        _u8     ssh_pwd         [16];
};
```

2.2.2 Parameters

- mac_addr
  [out] Ethernet address
  RESERVED, NOT allowed write (READ-ONLY). We DO NOT guarantee any damage occurred by using illegal directions and also we DO NOT support any individual use.
  e.g. The MAC address 00:30:f9:12:34:56 is stored in mac_addr array like this.
  mac_adr[0] = 0x00, mac_adr[1] = 0x30, mac_adr[2] = 0xf9,
  mac_adr[3] = 0x12, mac_adr[4] = 0x34, mac_adr[5] = 0x56
- major
  [out] Firmware major version
- minor
  [out] Firmware minor version and revision number

7(MSB)	6	5	4	3	2	1	0(LSB)
minor				revision			

- local_ip
  [in/out] Local IP address
  You can set IP A.B.C.D using this macro.
  #define MK_IP(__b1,__b2,__b3,__b4) \
  (((unsigned long)(__b4) << 24) | ((unsigned long)(__b3) << 16) | \
  ((unsigned long)(__b2) << 8) | ((unsigned long)(__b1)))
- net_mask
  [in/out] Subnet mask
  You can set subnet mask A.B.C.D using above macro.
- gate_ip;
  [in/out] Gateway IP address
  You can set gateway IP address A.B.C.D using above macro.
- poe_uid
  [in/out] PPPoE log-in ID.
- poe_pwd
  [in/out] PPPoE log-in password.
- socket_ip
  [out] The IP address of TCP/IP socket
  READ ONLY. This IP address is packet's IP address which is received from TCP/IP socket. It is useful when the ezTCP uses DHCP.
- dns_ip
  [in/out] DNS(Domain name system) server IP address
  You can set DNS server IP address A.B.C.D using above macro.
- product_id_old
  [out] The product id of ezTCP
  READ ONLY. Do not modify this variable.

Value	Product
0x10	CIE-H10
0x11	CSE-H20 / CSE-H21
0x12	CSE-M32
0x14	CSE-M73
0x15	CSW-H80
0x21	CIE-M10
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
0x3c	CSW-M84
0x3d	CSE-M53N
0x3e	CSE-H53N
0x3f	CSE-H55N

- product_id_new
  [out] The product id of ezTCP
  READ ONLY. Do not modify this variable. If this value is zero(0) then use product_id_old.

Value	Product
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
0x3c	CSW-M84
0x3d	CSE-M53N
0x3e	CSE-H53N
0x3f	CSE-H55N

- ddns_id
  [in/out] DDNS service log-in ID.
  The user id which is used to log-in DDNS service provider
- ddns_pwd
  [in/out] DDNS service log-in password.
  The user password which is used to log-in DDNS service provider
- ddns_host_name

[in/out] The host name for DDNS service.
※ Please refer to product user's manual for more detail information about DDNS.

- ssh_id, ssh_pwd
  [in/out] ID and password for SSH service.

2.2.3 Remarks

- Whole variables are stored by Little Endian(Host byte order), except below parameters.
- local_ip, net_mask, gate_ip, socket_ip and dns_ip are stored by Big Endian(Network Byte Order).

2.3 uart_env

2.3.1 Overview

- Basic structure to store UART variables.

```
struct uart_env
{
        struct uart_dev_env     uart_dev_env;
        struct uart_var_env     uart_var_env;
        struct uart_del_env     uart_del_env;           /* NOT USED. Never Modify. */
        _u8                     host_name       [64];
        _u8                     comment         [32];
        _u32                    use_flag;               // READ ONLY. Never modify.
};
```

2.3.2 Parameters

- uart_dev_env
  [in/out] Basic structure to store UART hardware related variables.
- uart_var_env
  [in/out] Basic structure to store UART operation related variables.
- host_name
  [in/out] DNS name of peer host.
  It is used when ezTCP runs as a TCP client. If you want to use DNS name instead of IP address then the “peer_ip” parameter in “uart_var_env” should be zero(0).
- comment
  [in/out] Short comment for UART.
  This option only using with CSE-T16 / T32 / T48.
- use_flag
  [out] The flag for indicating that it used or not
  READ ONLY. This parameter indicates whether the UART is valid or not. Only if this parameter is one(1) then it is valid UART.

2.3.3 Remarks

- The number of available UART is varying according to ezTCP. The “use_flag” represents existence of UART.

2.4 uart_dev_env

2.4.1 Overview

- Basic structure to store UART hardware related variables.

```
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.
        _u32            max_baud;               //READ ONLY. Never modify
        _u32            sio_baud;
};
```

2.4.2 Parameters

- 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

flowctrl	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

parity2	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 and 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. Wrong serial baud rate may cause a problem.

2.5 uart_var_env

2.5.1 Overview

- Basic structure to store UART operation related variables.

```
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;
};
```

2.5.2 Parameters

- 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 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 allow starting connection
  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 value: 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.

2.5.3 Remarks

- Whole variables are stored by Little Endian(Host byte order), except below parameters.
- peer_ip is stored by Big Endian(Network Byte Order).

2.6 opt_env

2.6.1 Overview

- Basic structure to store environment variables.

```
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    pad0            : 2;    // NOT USED. Never modify.
        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;
        unsigned int    pd              : 1;
        unsigned int    pad3            : 3;    // NOT USED. Never modify.
};
```

2.6.2 Parameters

- ezcfg_lock
  [in/out]
  If this parameter is one(1), the ezTCP replies to the host that has same MAC or IP address defined in the "allow_mac" or "allow_ip" in "etc_env" structure.
- 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_ip" in "net_env" structure will be used as the 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 value 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, an 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, an ezTCP sends its MAC address to peer when a TCP/IP connection is established.

2.7 etc_env

2.7.1 Overview

- Basic structure to store environment variables.

```
struct etc_env
{
        _u8     allow_mac       [6];
        _u8     pad0            [2];            // NOT USED. Never modify.
        _u32    allow_ip;
        _u32    allow_subnet;
        _u8     comment         [64];
        _u8     pwd_old         [PASSWD_SIZE];  // RESERVED. Never modify
        _u8     pwd             [PASSWD_SIZE];  // RESERVED. Never modify
        _u8     allow_ip6       [16];
        _u32    allow_ip6_prefix;
};
```

2.7.2 Parameters

- allow_mac
  [in/out]
  Host's MAC address that attempts to establish connection to ezTCP may be restricted. When this value is set, hosts with designated MAC address may only connect to ezTCP.
- allow_ip
  allow_subnet
  [in/out]
  If this value is set, host connections can be restricted based on designated "allow_ip" and "allow_subnet" setting value. When "allow_ip" value is selected, "allow_ip" and "allow_subnet" may be bit AND to identify allowed hosts. An example is as follow.

allow_ip	allow_subnet	Allowed Hosts IP address range
10.1.0.1	255.0.0.0	10.1.0.1 ~ 10.255.255.254
10.1.0.1	255.255.255.0	10.1.0.1 ~ 10.1.0.254
192.168.1.4	255.255.255.255	192.168.1.4

- comment
  [in/out]
  When use multiple ezTCP, the comments option help you distinguish each ezTCP.
- pwd_old, pwd
  [out] RESERVED
  This value is used for setting a password to ezTCP. But, it is not using directly.
  You can set or modify password to ezTCP through ezManager library functions.
  If pwd_old parameter has "********" then the ezTCP is protected by password.
- allow_ip6
  [in/out]
  An IP6 address that is only allowed to connect to the ezTCP.

- allow_ip6_prefix
  [in/out]
  Subnet prefix length of allow_ip6.

2.7.3 Remarks

- You should use Big endian(Network byte order) for allow_ip and allow_subnet parameters.

2.8 io_env

2.8.1 Overview

- Basic structure to store environment variables for I/O product.

```
#define MAX_DI                  8
#define MAX_DO                  8
#define IO_SCRIPT_LEN           32
#define IO_COMMENT_LEN          16

struct io_env
{
        _u8                     di_num;         // READ ONLY. Never modify
        _u8                     do_num;         // READ ONLY. Never modify
        _u16                    html_size;
        _u8                     script          [MAX_DO][IO_SCRIPT_LEN];
        _u8                     host_name       [64];
        _u8                     di_comment      [MAX_DI][IO_COMMENT_LEN];
        _u8                     do_comment      [MAX_DO][IO_COMMENT_LEN];
        struct  io_var_env      io_var_env;
        _u32                    use_flag;       // READ ONLY. Never modify
};
```

2.8.2 Parameters

- di_num
  [out] Number of available digital input ports
  READ ONLY.
- do_num
  [out] Number of available digital output ports
  READ ONLY.
- html_size
  [in/out] Memory size for HTML page(Unit : Kilobyte, KB).
  This value should be 80, 96 or 112. The 80 is default value.
- script
  [in/out] Macro for its digital output port
  Please see a product user's manual for more detail information.
- host_name
  [in/out] DNS name of peer host.
  It is used when ezTCP runs as a TCP client. If you want to use DNS name then you should set "peer_ip" parameter in "io_var_env" to zero(0).
- di_comment
  [in/out]
  Short comment for digital input port

- do_comment
  [in/out]
  Short comment for digital output port
- io_var_env
  [in/out] Basic structure to store environment variables for a digital input / output ports
- use_flag
  [out] The flag for indicating that it used or not
  READ ONLY. Only if this flag is one(1) then this is valid I/O.

2.9 io_var_env

2.9.1 Overview

- Basic structure to store environment variables for I/O ports.

```
struct io_var_env
{
        unsigned int    modbus          : 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];
};
```

2.9.2 Parameters

- modbus
  [in/out] Enable / Disable Modbus/TCP protocol.
- macro
  [in/out] Enable / Disable MACRO function.
- master
  [in/out]

master	Description
0	Modbus/TCP Slave
1	Modbus/TCP 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 products will send its input port values to the master when it's input port status changes.
- conns
  [in/out] Number of TCP/IP connections for Modbus/TCP
  If modbus 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.
  0^{th} bit(LSB) means 0^{th} 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)** 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)** 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 discrete)

- 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.
- 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]
  It has different meaning according to "master" variable.

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

- input_addr
  [in/out]
  It has different meaning according to "master" variable.

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

- output_addr
  [in/out]
  It has different meaning according to "master" variable.

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

- 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 (Unit: millisecond)
- 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.

2.9.3 Remarks

- You should use Big endian(Network byte order) for peer_ip parameter.

2.10 ip_trap_env

2.10.1 Overview

- Basic structure to store environment variables about IP address notification.

```
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;
};
```

2.10.2 Parameters

- level
  [in/out] If "ddns" parameter in "opt_env" is 2(TCP) or 3(UDP) then it decides the type of data.

level	Description
0	ASCII
1	Binary

- peer_ip
  [in/out] The hosts IP address to send a data.
- peer_port
  [in/out] The host port number to send a data
- interval
  [in/out] The interval for sending a data

2.10.3 Remarks

- You should use Big endian(Network byte order) for peer_ip parameter.

2.11 port_map_env

2.11.1 Overview

- Basic structure to store TCP port information.

```
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
};
```

2.11.2 Parameters

- 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	

2.12 wlan_env

2.12.1 Overview

- Basic structure to store WLAN variables.

```
struct wlan_env
{
        struct wlan_opt     wlan_opt_env;
        struct wlan_var     wlan_var_env;
        _u32                use_flag;           // READ ONLY. Never modify
};
```

2.12.2 Parameters

- wlan_opt_env
  [in/out] Basic structure to store WLAN variables
- wlan_var_env
  [in/out] Basic structure to store WLAN variables
- use_flag
  [out] The flag for indicating that it used or not
  READ ONLY. This parameter indicates this WLAN is valid or not. Only if this parameter is one(1) then it is valid WLAN.

2.13 wlan_opt_env

2.13.1 Overview

- Basic structure to store WLAN variables.

```
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
};
```

2.13.2 Parameters

- 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] WPA Encryption Strength

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

- 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.

2.14 wlan_var_env

2.14.1 Overview

- Basic structure to store WLAN variables.

```
struct wlan_var
{
        _u8     ssid                [32];
        _u8     key40               [4][5];
        _u8     key104              [4][13];
        _u8     wpa_passphrase      [32];
        _u8     wpa_psk             [32];       // READ ONLY. Never modify
        _u8     power_table         [16];       // READ ONLY. Never modify.
        _u8     pad0[3];                        // NOT USED. Never modify.
        _u8     key_flag;
        _u8     eap_id              [32];
        _u8     eap_pwd             [16];
};
```

2.14.2 Parameters

- ssid
  [in/out] SSID for wireless network
  ssid is ASCII string end with NULL(0x00). So, its maximum length is 31.
- key40
  [in/out] 4-sets of 40 bits key value
- key104
  [in/out] 4-sets of 104 bits key value
- wpa_passphrase
  [in/out] WPA(Wi-Fi Protected Access) passphrase.
  wpa_passphrase is ASCII string end with NULL(0x00). So, its maximum length is 63. It is used to make WPA-PSK. You should fill at least 8 bytes with A~Z, a~z or 0~9.

Product	The maximum length of WPA passphrase
CSW-H80	31-byte
CSW-M83 / M85	63-byte

- wpa_psk
  [in/out] WPA(Wi-Fi Protected Access) PSK(Pre-shared Key).
  The SSID and WPA passphrase are used for making WPA PSK.
  This value is automatically calculated when a program call the “EzManager_Write” library function.
  ※ READ ONLY. Never modify this variable.

- power_table
  [out] Signal Strength
  It is signal strength for each channel.
  ※ **READ ONLY. Never modify this variable.**

- key_flag
  [out] The type of WEP key

key_flag	Description
0	Hexadecimal code.
1	ASCII code.

- eap_id, eap_pwd
  [in/out]
  It is using when WPA authentication mode is EAP TLS, EAP TTLS or PEAP.
  Please refer to product user's manual for more detail information.

2.15 csc_hr2_env

2.15.1 Overview

- This basic structure for CSC-HR2

```
struct csc_hr2_env
{
        struct uart_dev_env         uart_dev_env;
        _u8                         mux_type;
        _u8                         pad                 [3];
        _u8                         csc_hr2_id          [16];
        _u8                         server_host_name1   [64];
        _u8                         server_host_name2   [64];
        struct redundancy_var_env   redundancy_var_env;
};
```

2.15.2 Parameters

- uart_dev_env
  [in/out]
  Basic structure to store UART hardware related variables. Please refer to 2.4 uart_dev_env for detail information.
- mux_type
  [in/out] Mode setting value

mux_type	Description
0	Automation mode : The CSC-HR2 is always a TCP client.
1	Firmware download mode for EZU-100 : It is using for downloading a firmware to EZU-100 through RS232 port.

- csc_hr2_id
  [in/out] ID of CSC-HR2
  This parameter is for a server which is using several CSC-HR2.
- server_host_name1
  [in/out] First server address
  If you want to use this parameter then the “peer_ip[0]” parameter in “redundancy_var_env” should be zero(0).
- server_host_name2
  [in/out] Second server address
  If you want to use this parameter then the “peer_ip[1]” parameter in “redundancy_var_env” should be zero(0).
- redundancy_var_env
  [in/out] Basic structure for redundancy function

2.16 redundancy_var_env

2.16.1 Overview

- Basic structure for redundancy function.

```
struct redundancy_var_env
{
        _u32    peer_ip         [2];
        _u16    peer_port       [2];
        _u16    timeout         [2];
        _u16    threshold       [2];
        _u16    check_port;
        _u16    pad0;
};
```

2.16.2 Parameters

- peer_ip[0]
  [in/out] First server IP address
- peer_ip[1]
  [in/out] Second server IP address
- peer_port[0]
  [in/out] First server port number
- peer_port[1]
  [in/out] Second server port number
- timeout[0]
  [in/out] Network transfer timeout [Unit: Second]
  This parameter is the reference time when switching from the wired LAN to 3G network.
- timeout[1]
  [in/out] Change server timeout [Unit: Second]
  This parameter is the reference time when changing the destination server.
- threshold[0]
  [in/out] Network transfer byte count [Unit: Byte]
  This parameter is the reference byte count when switching from the wired LAN to 3G network.
- threshold[1]
  [in/out] Change server byte count [Unit: Byte]
  This parameter is the reference byte count when changing the destination server.
- check_port
  [in/out] TCP port to investigate the quality of communication

2.17 tcp_status_env

2.17.1 Overview

- This data structure is to store status of TCP/IP(IPv4) session of ezTCP.

```
struct tcp_status_env
{
        _u8     stat;           // READ ONLY. Never modify
        _u8     pad1;           // NOT USED. Never modify
        _u16    winq_out;       // NOT USED. Never modify
        char    name[8];        // READ ONLY. Never modify
        _u32    local_ip;       // READ ONLY. Never modify
        _u32    peer_ip;        // READ ONLY. Never modify
        _u16    local_port;     // READ ONLY. Never modify
        _u16    peer_port;      // READ ONLY. Never modify
};
```

2.17.2 Parameters

- stat
  [out] TCP/IP(IPv4) connection status

stat	Description	
0	CLOSED	Connection is closed.
1	LISTEN	Wait a TCP/IP connection from peer.
2	SYN_SENT	SYN packet sent.
3	SYN_RCVD	SYN packet received.
4	ESTABLISHED	a TCP/IP Connection is established.
5	FIN_WAIT1	FIN packet sent.
6	FIN_WAIT2	ACK packet received when it is in FIN_WAIT1.
7	CLOSE_WAIT	FIN packet received when it is in ESTABLISHED.
8	CLOSING	FIN packet received when it is in FIN_WAIT1.
9	LAST_ACK	FIN packet sent when it is in CLOSE_WAIT.
10	TIME_WAIT	Wait for starting new TCP/IP session.
11	END	Connection is closed.

- name
  [out] The name of TCP/IP session
- local_ip
  [out] Local IPv4 address of ezTCP
- peer_ip

[out] The IPv4 address of target host

- local_port
  [out] Local port number of ezTCP
- peer_port
  [out] Port number of target host

2.18 tcp6_status_env

2.18.1 Overview

- This data structure is to store status of TCP/IP(IPv6) session of ezTCP.

```
struct tcp6_status_env
{
        _u8     stat;           // READ ONLY. Never modify
        _u8     pad1;           // NOT USED. Never modify
        _u16    winq_out;       // NOT USED. Never modify
        char    name    [8];    // READ ONLY. Never modify
        _u8     local_ip6 [16]; // READ ONLY. Never modify
        _u32    peer_ip6  [16]; // READ ONLY. Never modify
        _u16    local_port;     // READ ONLY. Never modify
        _u16    peer_port;      // READ ONLY. Never modify
};
```

2.18.2 Parameters

- stat
  [out] TCP/IP(IPv6) connection status

stat	Description	
0	CLOSED	Connection is closed.
1	LISTEN	Wait a TCP/IP connection from peer.
2	SYN_SENT	SYN packet sent.
3	SYN_RCVD	SYN packet received.
4	ESTABLISHED	a TCP/IP Connection is established.
5	FIN_WAIT1	FIN packet sent.
6	FIN_WAIT2	ACK packet received when it is in FIN_WAIT1.
7	CLOSE_WAIT	FIN packet received when it is in ESTABLISHED.
8	CLOSING	FIN packet received when it is in FIN_WAIT1.
9	LAST_ACK	FIN packet sent when it is in CLOSE_WAIT.
10	TIME_WAIT	Wait for starting new TCP/IP session.
11	END	Connection is closed.

- name
  [out] The name of TCP/IP session
- local_ip
  [out] Local IPv6 address of ezTCP
- peer_ip

[out] The IPv6 address of target host

- local_port
  [out] Local port number of ezTCP
- peer_port
  [out] Port number of target host

3 Functions

3.1 EzManager_Search

3.1.1 Overview

- Basic function to search ezTCPs on local or remote network.
- Each ezTCP is discriminated by MAC or IP address.

3.1.2 Prototype

```
int EzManager_Search(
        int             mode,
        _u32            ip,
        struct lib_env  *lib_env,
        int             *nResultCount,
        int             *nErrNum,
        int             port =0,
        _u32            bind_ip = 0
);
```

3.1.3 Parameters

- mode
  [in]

mode	Description
0	Search from local network
1	Search from remote network

- ip
  [in] The ezTCP's local ip address to read environment values
  This parameter has to use Big-Endian.
- lib_env
  [out] "lib_env" structure to store environment values.
- nResultCount
  [out] The number of founded ezTCP
- nErrNum
  [out] Error number
- port
  [in] The UDP port number for ezManager library
  Default UDP port numbers are 50005 and 50007

- bind_ip
  [in] The library associates “bind_ip” with a TCP/IP socket.
  If this value is zero(0) then the O/S associates one of its local addresses with a TCP/TCP socket.
  This parameter has to use Big-Endian.

3.1.4 Return Value

- If no error occurred, EzManager_Search returns 1.
- When any error is occurred, the return value is EZTCP_ERR(-1) and the error number is stored in eErrNum.

3.1.5 Remarks

- Before using this function, you have to reserve enough space for *lib_env* structure.
  e.g. lib_env = (struct lib_env *)malloc(sizeof(struct lib_env) * 256);
- **This function takes at least 2 seconds to complete running.**
- **You should call Exit_Library function to release dynamically allocated memory when your application is closed.**

3.2 EzManager_Read

3.2.1 Overview

- Basic function to read environment values from ezTCP.

3.2.2 Prototype

```
int EzManager_Read(
        int             mode,
        _u8             *mac_addr,
        _u32            ip,
        struct lib_env  *lib_env,
        int             *nErrNum,
        int             port = 0,
        _u32            bind_ip = 0
);
```

3.2.3 Parameters

- mode
  [in]

mode	Description
0	Read from local network
1	Read from remote network

- mac_addr
  [in] The MAC address of ezTCP.
- ip
  [in] The ezTCP's local ip address to read.
  This parameter has to use Big-Endian.
- lib_env
  [out] "lib_env" structure to store ezTCP environment values.
- nErrNum
  [out] Error number.
- port
  [in] The UDP port number for ezManager library.
  Default UDP port number is 50005 and 50007.
- bind_ip
  [in] The library associates "bind_ip" with a TCP/IP socket.
  If this value is zero(0) then the O/S associates one of its local addresses with a TCP/TCP socket.
  This parameter has to use Big-Endian.

3.2.4 Return Value

- If no error occurred, EzManager_Read returns 1.
- When any error has occurred, the return value is EZTCP_ERR(-1) and the error number is stored in eErrNum.

3.2.5 Remarks

- Before using this function, you have to reserve enough space for *lib_env* structure.
  eg. lib_env = (struct lib_env *)malloc(sizeof(struct lib_env) * 256);
- **This function takes at least 2 seconds to complete running.**
- **You should call Exit_Library function to release dynamically allocated memory when your application is closed.**

3.3 EzManager_Write

3.3.1 Overview

- Basic function to write environment values to ezTCP.

3.3.2 Prototype

```
int EzManager_Write(
        int             mode,
        struct lib_env  *lib_env,
        _u8             *cur_pwd,
        int             *nErrNum,
        int             port = 0,
        _u32            bind_ip = 0
);
```

3.3.3 Parameters

- mode
  [in]

mode	Description
0	Write to local network
1	Write to remote network

- lib_env
  [in] “lib_env” structure containing environment value to be written.
- cur_pwd
  [in] Current password of ezTCP
- nErrNum
  [out] Error number
- port
  [in] The UDP port number for ezManager library
  Default UDP port number is 50005 and 50007.
- bind_ip
  [in] The library associates “bind_ip” with a TCP/IP socket.
  If this value is zero(0) then the O/S associates one of its local addresses with a TCP/TCP socket.
  This parameter has to use Big-Endian.

3.3.4 Return Value

- If no error occurred, EzManager_Write returns 1.
- When any error has occurred, the return value is EZTCP_ERR(-1) and the error number is stored in nErrNum.

nErrNum	Description
EZTCP_ERR_PWD	Password mismatch error.
EZTCP_ERR_RES	No response from ezTCP
EZTCP_ERR_LOCAL_IP	Unavailable local IP address
EZTCP_ERR_CIE_H10_SCRIPT	Macro syntax error of I/O products.
EZTCP_ERR_CIE_H10_IO_ADDR	I/O addresses error of I/O products.
EZTCP_ERR_LOCAL_PORT_100	Local Port number is duplicated.
EZTCP_ERR_LOCAL_PORT_101	The port number 23 is not available for Local Port.
EZTCP_ERR_LOCAL_PORT_102	Local Port and Modbus/TCP numbers are duplicated.
EZTCP_ERR_LOCAL_PORT_103	The port number 80 is not available for Local Port.
EZTCP_ERR_LOCAL_PORT_104	The port number 50005 is not available for Local Port.
EZTCP_ERR_LOCAL_PORT_105	The port number 50006 is not available for Local Port.
EZTCP_ERR_PRODUCT_MISMATCH	Wrong product id.
EZTCP_ERR_UNKNOWN	Unknown error.
EZTCP_ERR_NO_INFO	If you try to write before using EzManager_Search function then this error will be caused.
EZTCP_ERR_NO_NETWORK	There is no proper network adapter.

3.3.5 Remarks

- Before using this function, you have to call EzManager_Search function for reliable data transmitting.
- This function waits a reply from ezTCP for 3 seconds.
- **You should call Exit_Library function to release dynamically allocated memory when your application is closed.**

3.4 EzManager_Status

3.4.1 Overview

- Basic function to read status information from ezTCP.
- The status information is composed by plain text.

3.4.2 Prototype

```
int EzManager_Status(
        int                     mode,
        struct lib_env          *lib_env,
        _u8                     *stat_buf,
        int                     stat_buf_len,
        struct tcp_status_env   *tcp_status_env,
        int                     *tcp_session_count,
        int                     *nErrNum,
        int                     port = 0,
        _u32                    bind_ip = 0
);
```

3.4.3 Parameters

- mode
  [in]

mode	Description
0	Send a query to local network
1	Send a query to remote network

- lib_env
  [in] “lib_env” structure which has target ezTCP’s environment values
- stat_buf
  [out] ezTCPs status information.
- stat_buf_len
  [in] The size of stat_buf
- tcp_status_env
  [in] “tcp_status_env” structure to store TCP/IP session status
- tcp_session_count
  [out] The number of TCP/IP sessions of ezTCP
- nErrNum
  [out] Error number

- port
  [in] The UDP port number for ezManager library
  Default UDP port numbers are 50005 and 50007
- bind_ip
  [in] The library associates “bind_ip” with a TCP/IP socket.
  If this value is zero(0) then the O/S associates one of its local addresses with a TCP/TCP socket.
  This parameter has to use Big-Endian.

3.4.4 Return Value

- If no error occurred, EzManager_Status returns 1.
- When any error has occurred, the return value is EZTCP_ERR(-1) and the error number is stored in eErrNum.

3.4.5 Remarks

- Before using this function, you have to call EaManager_Search function for reliable data transmitting.
- Before using this function, you have to reserve enough space for *tcp_status_env* structure. The maximum TCP/IP session is defined 16.
- This function waits a reply from ezTCP for 2 seconds.
- **You should call Exit_Library function to release dynamically allocated memory when your application is closed.**

3.5 EzManager_ChangePwd

3.5.1 Overview

- Basic function to change or erase password of ezTCP.

3.5.2 Prototype

```
int EzManager_ChangePwd(
        int             mode,
        struct lib_env  *lib_env,
        _u8             *cur_pwd,
        _u8             *new_pwd,
        int             *nErrNum,
        int             port = 0,
        _u32            bind_ip = 0
);
```

3.5.3 Parameters

- mode
  [in]

mode	Description
0	Send a query to local network
1	Send a query to remote network

- lib_env
  [in] "lib_env" structure which has target ezTCP's environment values
- cur_pwd
  [in] Current password
- new_pwd
  [in] New password
- nErrNum
  [out] Error number
- port
  [in] The UDP port number for ezManager library
  Default UDP port numbers are 50005 and 50007.
- bind_ip
  [in] The library associates "bind_ip" with a TCP/IP socket
  If this value is zero(0) then the O/S associates one of its local addresses with a TCP/TCP socket
  This parameter has to use Big-Endian.

3.5.4 Return Value

- If no error occurred, EzManager_ChangePwd returns 1.
- When any error has occurred, the return value is EZTCP_ERR(-1) and the error number is stored in nErrNum.

3.5.5 Remarks

- Before using this function, you have to call EzManager_Search function for reliable data transmitting.
- This function waits a reply from ezTCP for 3 seconds.
- **You should call Exit_Library function to release dynamically allocated memory when your application is closed.**

3.6 EzManager_CloseTCP

3.6.1 Overview

- Basic function to close TCP/IP session of ezTCP.

3.6.2 Prototype

```
int EzManager_CloseTCP(
        int                     mode,
        struct lib_env          *lib_env,
        struct tcp_status_env   *tcp_status_env,
        _u8                     *cur_pwd,
        int                     *nErrNum,
        int                     port = 0,
        _u32                    bind_ip = 0
);
```

3.6.3 Parameters

- mode
  [in]

mode	Description
0	Send a query to local network
1	Send a query to remote network

- lib_env
  [in] “lib_env” structure which has target ezTCP’s environment values
- tcp_status_env
  [in] “tcp_status_env” structure which has TCP/IP session information
- cur_pwd
  [in] Current password of ezTCP
- nErrNum
  [out] Error number
- port
  [in] The UDP port number for ezManager library
  Default UDP port numbers are 50005 and 50007
- bind_ip
  [in] The library associates “bind_ip” with a TCP/IP socket
  If this value is zero(0) then the O/S associates one of its local addresses with a TCP/TCP socket
  This parameter has to use Big-Endian.

3.6.4 Return Value

- If no error occurred, EzManager_CloseTCP returns 1.
- When any error has occurred, the return value is EZTCP_ERR(-1) and the error number is stored in nErrNum.

3.6.5 Remarks

- Before using this function, you have to call EzManager_Status function for getting TCP/IP session information of ezTCP
- This function waits a reply from ezTCP for 2 seconds.
- **You should call Exit_Library function to release dynamically allocated memory when your application is closed.**

3.7 EzManager_RdbOnOff

3.7.1 Overview

- Basic function to start or stop send DEBUG message.
- ezTCP broadcast DEBUG message to local network after completely booted up when its debug function is enabled.
- You can make a ezTCP that start or stop send DEBUG message to specific host by using “EzManager_RdbOnOff”
- This function is currently considered in below ezTCP products.

LAN type	Product
Wired LAN	CSE-M53
Wireless LAN	

3.7.2 Prototype

```
int EzManager_RdbOnOff(
        int         mode,
        _u8         *mac_addr,
        _u32        eztcp_ip,
        BOOL        onoff,
        _u32        rcvd_ip,
        int         *nErrNum,
        int         port = 0,
        _u32        bind_ip = 0
);
```

3.7.3 Parameters

- mode
  [in]

mode	Description
0	Send a query to local network
1	Send a query to remote network

- mac_addr
  [in] The MAC address of ezTCP.
- eztcp_ip
  [in] The IP address of ezTCP
  This parameter has to use Big-Endian.

- onoff
  [in]

onoff	Description
0	Stop sending DEBUG message.
1	Start sending DEBUG message.

- rcvd_ip
  [in] PCs IP address to receive DEBUG message
  This parameter has to use Big-Endian.
- nErrNum
  [out] Error number
- port
  [in] The UDP port number for ezManager library
  Default UDP port number is 50005 and 50007
- bind_ip
  [in] The library associates “bind_ip” with a TCP/IP socket
  If this value is zero(0) then the O/S associates one of its local addresses with a TCP/TCP socket
  This parameter has to use Big-Endian.

3.7.4 Return Value

- If no error occurred, EzManager_RdbOnOff returns 1.
- When any error has occurred, the return value is EZTCP_ERR(-1) and the error number is stored in nErrNum.

3.7.5 Remarks

- This function waits a reply from ezTCP for 2 seconds.
- **You should call Exit_Library function to release dynamically allocated memory when your application is closed.**

3.8 EzManager_ReBoot

3.8.1 Overview

- Basic function to reboot ezTCP.
- This function is currently considered in below ezTCP products.

LAN type	Product
Wired LAN	CSE-M53
Wireless LAN	

3.8.2 Prototype

```
int EzManager_ReBoot(
        int         mode,
        _u8         *mac_addr,
        _u32        eztcp_ip,
        int         *nErrNum,
        int         port = 0,
        _u32        bind_ip = 0
);
```

3.8.3 Parameters

- mode
  [in]

mode	Description
0	Send queries to local network
1	Send queries to remote network

- mac_addr
  [in] The MAC address of ezTCP
- eztcp_ip
  [in] The IP address of ezTCP
  This parameter has to use Big-Endian.
- nErrNum
  [out] Error number
- port
  [in] The UDP port number for ezManager library
  Default UDP port numbers are 50005 and 50007

- bind_ip
  [in] The library associates “bind_ip” with a TCP/IP socket.
  If this value is zero(0) then the O/S associates one of its local addresses with a TCP/TCP socket.
  This parameter has to use Big-Endian.

3.8.4 Return Value

- If no error occurred, EzManager_ReBoot returns 1.
- When any error has occurred, the return value is EZTCP_ERR(-1) and the error number is stored in nErrNum.

3.8.5 Remarks

- This function waits a reply from ezTCP for 2 seconds.
- **You should call Exit_Library function to release dynamically allocated memory when your application is closed.**

3.9 GetLibVer

3.9.1 Overview

- Basic function to get library version information

3.9.2 Prototype

```
char* GetLibVer(int* len);
```

3.9.3 Parameters

- len
  [out] The length of library version information string

3.9.4 Return Value

- The text of library version information.

3.10 GetProductName

3.10.1 Overview

- Basic function to get a product name.

3.10.2 Prototype

```
char* GetProductName(int product_id, int* len);
```

3.10.3 Parameters

- product_id
  [in] This value should be “product_id_old” or “product_id_new” in “net_env” structure. If “product_id_new” is zero(0) then “product_id_old” parameter should be used.
- len
  [out] The length of product name string

3.10.4 Return Value

- The text of a product name

3.11 Exit_Library

3.11.1 Overview

- This library dynamically allocates some memory block to store data. These memory blocks should be explicitly released by this function before terminating applications.

3.11.2 Prototype

```
void Exit_Library(void);
```