ezConfig Program Library for Visual C++ (DLL)

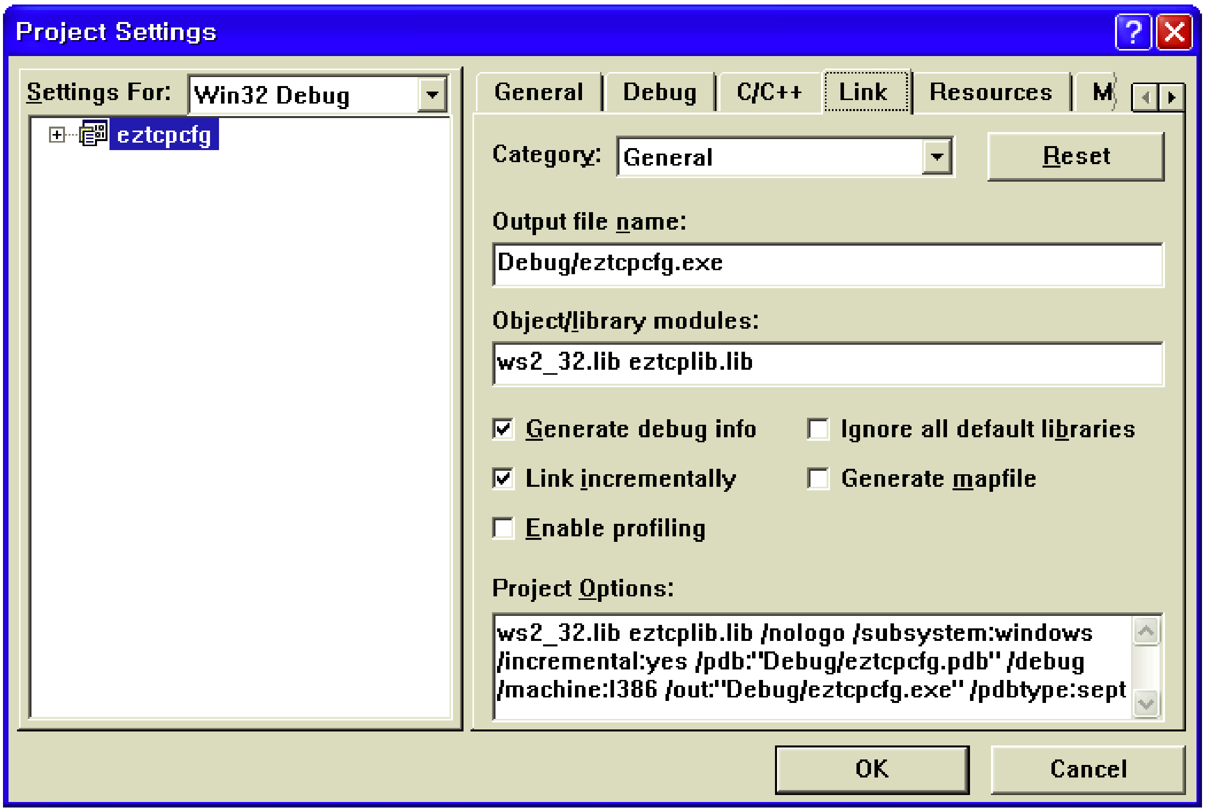
2009-10-08

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

Agenda

1 Introduction

- You have to include eztcplib.h in your source code and add library ws2_32.lib and eztcplib.lib



Settings For: Win32 Debug

Category: General

Output file name: Debug/eztcpcfg.exe

Object/library modules: ws2_32.lib eztcplib.lib

☑ Generate debug info ☐ Ignore all default libraries

☑ Link incrementally ☐ Generate mapfile

☐ Enable profiling

Project Options:

```
ws2_32.lib eztcplib.lib /nologo /subsystem:windows
/incremental:yes /pdb:"Debug/eztcpcfg.pdb" /debug
/machine:I386 /out:"Debug/eztcpcfg.exe" /pdbtype:sept
```

- And add below source code for using library functions.

```
#define IMPORT extern "C" __declspec(dllimport)

IMPORT int ProbeEzTCP(
            struct spe_env * eztcpenv,
            struct wlan_env * eztcpwenv,
            struct etc_opt * eztcpopt,
            int * nResultCount,
            int *nErrNum
);

IMPORT int ReadEzTCP(
            _u8* mac_addr,
            struct spe_env * eztcpenv,
            struct wlan_env * eztcpwenv,
            struct etc_opt * eztcpopt,
            int *nErrNum
);

IMPORT int WriteEzTCP(
        struct spe_env * eztcpenv,
        struct wlan_env * eztcpwenv,
        struct etc_opt * eztcpopt,
        int *nErrNum
);

IMPORT int StatusEzTCP(
        struct spe_env * eztcpenv,
        struct stat_env * eztcpstat,
        int *nErrNum
);

IMPORT int ChangePwdEzTCP(
        struct spe_env * eztcpenv,
        const char * ChangePwd,
        int *nErrNum
);

IMPORT int GetLibVerEzTCP(char * szVersion, int len);

IMPORT int RemoteChangePwdEzTCP(
        DWORD ip,
        struct spe_env * eztcpenv,
        const char * ChangePwd,
        int *nErrNum
);

IMPORT int RemoteReadEzTCP(
        DWORD ip,
        struct spe_env * eztcpenv,
        struct wlan_env * eztcpwenv,
        struct etc_opt * eztcpopt,
        int * nResultCount,
        int *nErrNum
);

IMPORT int RemoteStatusEzTCP(
        DWORD ip,
        struct spe_env * eztcpenv,
        struct stat_env * eztcpstat,
        int *nErrNum
);

IMPORT int RemoteWriteEzTCP(
        DWORD ip,
        struct spe_env * eztcpenv,
        struct wlan_env * eztcpwenv,
        struct etc_opt * eztcpopt,
        int *err
);

IMPORT int CloseTCP(
        struct spe_env * eztcpenv,
        const char * Pwd,
        int *nErrNum
);
```

```
IMPORT int ResetEzTCP(
        struct spe_env * eztcpenv,
        const char * Pwd,
        int *nErrNum
);

IMPORT int RemoteCloseTCP(
        DWORD ip, struct spe_env * eztcpenv,
        const char * Pwd,
        int *nErrNum
);

IMPORT int RemoteResetEzTCP(
        DWORD ip, struct spe_env * eztcpenv,
        const char * Pwd,
        int *nErrNum
);
```

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 spe_env

2.1.1 Overview

Basic data structure for ezConfig Program Library Functions.

```
struct spe_env
{
        /* RESERVED SECTION. NEVER MODIFY */
        _u8 config[4];                  // Reserved. Never modify.
        _u8 eth_addr[6];                // Ethernet address. Never modify.
        _u8 secure[18];                 // Reserved. Never modify.
        unsigned int level;             // Reserved. Never modify.
        unsigned int eap_lock;          // Reserved. Never modify.
        unsigned int en_xonoff;         // Reserved. Never modify.
        unsigned int ezl50;             // Reserved. Never modify.
        unsigned int dhcp_lock;         // Reserved. Never modify.
        unsigned int pppoe_lock;        // Reserved. Never modify.
        unsigned int to_lock;           // Reserved. Never modify.
        unsigned int lp_lock;           // Reserved. Never modify.
        unsigned int rp_lock;           // Reserved. Never modify.
        unsigned int wm_lock;           // Reserved. Never modify.
        unsigned int rip_lock;          // Reserved. Never modify.
        unsigned int en_databit;        // Reserved. Never modify.
        unsigned int en_parity;         // Reserved. Never modify.
        unsigned int major;             // Reserved. Never modify.
        unsigned int minor;             // Reserved. Never modify.

        /* network section */
        unsigned int dhcp;
        unsigned int bootp;             // NOT USED. Never modify.
        unsigned int arp;
        unsigned int ssl;               // Reserved. Never modify.
        unsigned int ezcfg;
        unsigned int nat;               // NOT USED. Never modify.
        unsigned int pppoe;
        unsigned int stat;              // Reserved. Never modify.
        unsigned int arp_probe;         // NOT USED. Never modify.
        unsigned int max_mux;           // Reserved. Never modify.
        unsigned int eapol;             // Reserved. Never modify.
        unsigned int t2s_peer;          // Reserved. Never modify.
        unsigned int pwr_down;
        unsigned int en_rcfg;           // Read only. Never modify.
        unsigned int rcfg;
        unsigned int comment;           // Read only. Never modify.
        unsigned int pad2a;             // NOT USED. Never modify.
        unsigned int mac_id;
        unsigned int pad2b;             // NOT USED. Never modify.
        unsigned int sio_no_idle;
        unsigned int sio_rx_drop;
        unsigned int pad2c;             // NOT USED. Never modify.
        _u32 local_ip;
        _u32 net_mask;
        _u32 gate_ip;
        _u8 poe_uid3[16];            // NOT USED. Never modify.
        _u8 poe_uid2[8];             // NOT USED. Never modify.
        _u8 passwd[12];

        /* mux/sio section */
        unsigned int parity;
        unsigned int databit;
        unsigned int rtscts;
        unsigned int xonoff;
        unsigned int telnet;
        unsigned int http;           // NOT USED. Never modify.
        unsigned int fcs;            // NOT USED. Never modify.
        unsigned int seq_ack;        // NOT USED. Never modify.
        unsigned int hdlc_emu;       // NOT USED. Never modify.
        unsigned int logo;           // NOT USED. Never modify.
        unsigned int oem;            // NOT USED. Never modify.
        unsigned int wlancfg;        // Read only. Never modify.
        unsigned int stype;
        unsigned int stopbit;
        unsigned int t2mc;
        unsigned int parity2;
        unsigned int slow_tx;
        unsigned int pad3b;          // NOT USED. Never modify.
        _u32 sio_baud;
        _u16 timeout;
        _u8 mux_type;
        _u8 mux_pad;                 // NOT USED. Never modify.
        _u16 water_mark;
        _u16 pad4;                   // NOT USED. Never modify.
        _u32 remote_ip;
        _u16 remote_port;
        _u16 local_port;
        _u8 poe_uid[8];
        _u8 poe_pwd[8];
        _u32 id;                     // Reserved. Never modify.
};
```

2.1.2 Parameters

- eth_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.
  eg. The MAC address 00:30:f9:12:34:56 is stored in eth_addr array like this.
  eth_adr[0] = 0x00, eth_adr[1] = 0x30, eth_adr[2] = 0xf9,
  eth_adr[3] = 0x12, eth_adr[4] = 0x34, eth_adr[5] = 0x56

- major
  [out] Major version number.
  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.

- minor
  [out] Minor version number.
  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.
- dhcp
  [in/out] If this value is non zero, dhcp protocol is enabled.
- arp
  [in/out] If this value is non zero, arp protocol is enabled.
- ezcfg
  [in/out] If this value is non zero, ezCFG probe is enabled.
  When this flag is not set, you can probe only in ISP mode.
- pppoe
  [in/out] If this value is non zero, pppoe protocol is enabled.
- pwr_down
  [in/out] If this value is non zero, power down mode is enabled.
  - This parameter is for Wireless LAN product only.
- en_rcfg
  [out] If this value is non zero, this product supports remote configuration.
  *Please see ezConfig program's manual for more information about remote configuration.
- rcfg
  [in/out] If this value is non zero, remote configuration is enabled.

  This flag is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-70, EZL-200L, EZL-220, EZL-200F
Wireless LAN	EZL-300L, EZL-300S, EZL-80 / 80c / 90 / 300W Lite (Firmware v1.3i or higher)

- comment
  [out] If this value is non zero, this product supports comment function.

  This flag is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-70, EZL-200L, EZL-220, EZL-200F

Wireless LAN	EZL-300L, EZL-300S, EZL-80 / 80c / 90 / 300W Lite (Firmware v1.3i or higher)

- mac_id

  [out] Send MAC Address

  If this value is non zero, a ezTCP sends its MAC address to the remote host right after the connection is established.

  This flag is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-70, EZL-200L (Firmware v1.2c or higher)
Wireless LAN	

- sio_no_idle

  [out] Disable TCP Transmission Delay

  If this value is non zero, a ezTCP sends data from serial port to the network immediately. Because of this, it may cause inefficiency with each TCP header when the data comes frequently.

  This flag is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-70, EZL-200L (Firmware v1.2c or higher)
Wireless LAN	

- sio_rx_drop

  [out] Drop SIO RX Data

  If this value is non zero, a ezTCP drops the data which are received from SIO before TCP/IP connection is established. This flag is only considered when a ezTCP running as a TCP client.

  This flag is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-70, EZL-200L (Firmware v1.2c or higher)
Wireless LAN	

- 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.
- password
  [in/out] If you set a password to a ezTCP then "*password*" parameter need for "write" function.
- parity
  [in/out] Serial parity bit.

Parity	Description
0	None
1	Even
2	Odd
3	Use parity2 parameter.

The value 3 is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-220, EZL-200F
Wireless LAN	EZL-300S

- databit
  [in/out] Serial data bit.
- rtscts
  [in/out] If this value non zero, flow control is enabled.
- xonoff
  [in/out] If this value non zero, xonxoff is enabled.
- telnet
  [in/out] If this value non zero, telnet connection is enabled for device setting.
  This flag is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-100, EZL-110, EZL-200A, EZL-220, EZL-200F
Wireless LAN	EZL-300S

- wlancfg
  [out] If this value non zero, this product is wireless LAN product.
- stype
  [in/out] Serial port type.

Stype	Description
0	RS-232
1	RS-422
2	RS-485

This flag is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-220, EZL-200F
Wireless LAN	EZL-300S

- stopbit
  [in/out] Stop bit.

Stop Bit	Description
0	1 Stop Bit
1	2 Stop Bit

This flag is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-220, EZL-200F
Wireless LAN	EZL-300S

- t2mc
  [in/out] If telnet flag is non zero and this flag is non zero then telnet multi connection is enabled for device setting.
  This flag is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-200F
Wireless LAN	EZL-300S

- parity2
  [in/out] Serial Parity Bit

Parity	Description
0	Mark
1	Space

  This parameter is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-220, EZL-200F
Wireless LAN	EZL-300S

- tx_delay
  [in/out] UART Slow Transmission
  Please refer to EZL-50L user’s manual for more information.

  This flag is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-200L, EZL-50M, EZL-70 (Firmware v1.1k or higher)
Wireless LAN	

- sio_baud
  [in/out] Serial baud rate.
- 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, ezTCP will

not disconnect automatically. In the other mode, this value can be set for customization.

- mux_type
  [in/out] Mode setting value.

mux_type	Mode	Product Name
0	T2S	Server mode. ezTCP will wait for connection.
1	ATC	AT command mode. ezTCP can sever or client mode by using AT commands.
2	COD	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.

  In case of EZL-50L, EZL-50M, EZL-200, EZL-200L, EZL-220, EZL-200F, EZL-300S, EZL-300L, EZL-80, EZL-80c and EZL-90 assigning this value is allowed. However EZL-50 and EZL-60 are set this value by using a HOTFLASH.

- water_mark
  [in/out] Amount of data that can allow to start connection.
  This value is only considered in COD or U2S mode.

- remote_ip;
  [in/out] Target IP address.
  This value is only considered in COD or U2S mode.

- remote_port
  [in/out] Target port number.
  This value is only considered in COD or U2S mode.

- local_port
  [in/out] Local IP address.
  This value is only considered in T2S mode.

- poe_uid
  [in/out] PPPoE log-in ID.

- poe_pwd
  [in/out] PPPoE log-in password.

2.1.3 Remarks

- Not mentioned value is reserved. DO NOT modify the type, name and order of not mentioned parameters.

- All parameters are stored by using Little Endian(Host Byte Order) except below **parameters. The local_ip, net_mask, gate_ip and remote_ip are stored by using**

Big Endian(Network Byte Order). You should use Big Endian to set above parameters up.

2.2 wlan_env

2.2.1 Overview

wlan_env structure is for wireless LAN products.

LAN type	Product Name
Wireless LAN	EZL-80 / 80c / 90 / 300W Lite / 300L

Basic data structure for ezConfig Program Library Functions.

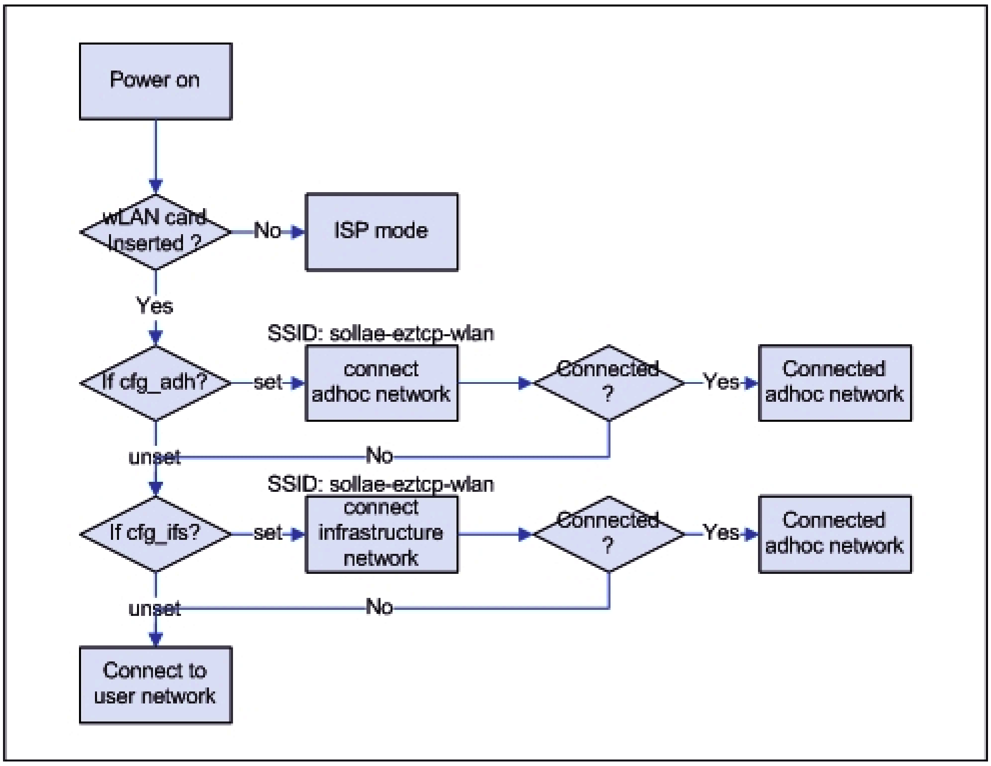
```
struct wlan_env
{
        unsigned int cctype;
        unsigned int channel;
        unsigned int wep;
        unsigned int wep_id;
        unsigned int cfg_ahc;
        unsigned int cfg_ifs;
        unsigned int auth;
        unsigned int pad1;          // NOT USED. Never modify.
        unsigned int pad2;          // NOT USED. Never modify.
        _u8 pad3[12];               // NOT USED. Never modify.
        _u8 target_ssid[32];
        _u8 new_ssid[32];
        _u8 key64[4][6];
        _u8 key128[4][14];
};
```

2.2.2 Parameters

- cctype
  [in/out] Connection Control Type (0 - adhoc, 1 - infrastructure)
- channel
  [in/out] Wireless LAN channel
- wep
  [in/out] WEP(Wired Equivalent Privacy) setting (0-no WEP, 1 - 64bit, 2 - 128 bit)
- wep_id
  [in/out] WEP index number (0, 1, 2, 3)
- cfg_ahc

[in/out] if this flag is set, ezTCP connects to default adhoc network during several seconds after booting. See below figure.

- cfg_ifs
  [in/out] if this flag is set, ezTCP connects to default infrastructure network during several seconds after booting. See below figure.
- auth
  [in/out] Authentication mode for infrastructure network. (0 - Open system, 1 - Shared key)
- target_ssid
  [in/out] SSID for infrastructure network
- new_ssid
  [in/out] SSID for adhoc network
- key64
  [in/out] 4-sets of 64 bits key value
- key128
  [in/out] 4-sets of 128 bits key value



2.3 etc_opt

2.3.1 Overview

etc_opt structure is for ezTCP products, which are supports etcetera options.

This structure is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-70, EZL-200L, EZL-220, EZL-200F
Wireless LAN	EZL-300L, EZL-300S, EZL-80 / 80c / 90 / 300W Lite (Firmware v1.3i or higher)

```
struct etc_opt
{
        _u8 comment[32];
        _u8 ext_opt[32];     // NOT USED. Never modify.
        _u8 opt[20];         // NOT USED. Never modify.
};
```

2.3.2 Parameters

- comment
  [in/out] When use multiple ezTCP, the comments option help you distinguish ezTCP.

2.4 stat_env

2.4.1 Overview

Basic data structure for ezConfig Program Library Functions.

```
struct stat_env
{
        _u8 ver[8];
        _u8 freq[6];
        _u8 boot_ver[8];
        _u8 mac_addr[6];
        _u8 ip_addr[4];
        _u8 sub_mask[4];
        _u8 gate_addr[4];
        unsigned int uptime_day;
        unsigned int uptime_hour;
        unsigned int uptime_minute;
        unsigned int uptime_second;
        long unsigned int sio_rx;
        long unsigned int sio_tx;
        long unsigned int eth_rx;
        long unsigned int eth_tx;
        unsigned int eth_crc;
        unsigned int eth_align;
        unsigned int eth_lost;
        char text[256];
};
```

2.4.2 Parameters

- ver[8]
  [out] Major version.
- freq[6]
  [out] Frequency of ezTCP.
- boot_ver[8]
  [out] Boot version. Only available in EZL-50 and EZL-60.
- mac_addr[6]
  [out] MAC address.
- ip_addr[4]
  [out] Local IP address.
- sub_mask[4]
  [out] Subnet mask.
- gate_addr[4]
  [out] Gateway IP address.
- uptime_day
  [out] ezTCP's alive days since the last boot up of ezCFG.
  WriteEzTCP or RemoteWriteEzTCP function will make EzTCP reboot.
- uptime_hour
  [out] ezTCP's alive hours since the last boot up of ezCFG.
  WriteEzTCP or RemoteWriteEzTCP function will make EzTCP reboot.
- uptime_minute
  [out] ezTCP's alive minutes since the last boot up of ezCFG.
  WriteEzTCP or RemoteWriteEzTCP function will make EzTCP reboot.
- uptime_second
  [out] ezTCP's alive seconds since the last boot up of ezCFG.
  WriteEzTCP or RemoteWriteEzTCP function will make EzTCP reboot.
- sio_rx
  [out] Received bytes from serial connection.
- sio_tx
  [out] Transmitted bytes to serial connection.

- eth_rx
  [out] Received packets from ethernet connection.
- eth_tx
  [out] Transmitted packets to ethernet connection.
- eth_crc
  [out] The number of CRC error occurred packets.
- eth_align
  [out] The number of align error occurred packets.
- eth_lost
  [out] The number of lost packets.
- text[256]
  [out] Reserved.

2.4.3 Remarks

- All members of this structure are NOT allowed write (READ-ONLY).
  We DO NOT **guarantee any damage occurred by using illegal directions.**
  We DO NOT support any individual use.

3 Functions

3.1 ProbeEzTCP

3.1.1 Overview

The ProbeEzTCP function can find ezTCP in local network. Each ezTCP can discriminate by MAC address.

3.1.2 Prototype

```
int ProbeEzTCP(
        struct spe_env * eztcpenv,
        struct wlan_env * eztcpwenv,
        struct etc_opt * eztcpopt,
        int * nResultCount,
        int *nErrNum
);
```

3.1.3 Parameters

- eztcpenv
  [out] Pointer to spe_env structure which ezTCP environment values will be stored in.
- eztcpwenv
  [out] Pointer to wlan_env structure which ezTCP wireless environment values will be stored in
- .eztcpopt
  [out] Pointer to etc_opt structure which ezTCP etcetera option values will be stored in.
- nResultCount
  [out] Pointer to integer which the number of founded ezTCP will be stored in.
- nErrNum
  [out] Pointer to integer which the error number, when error is occurred.

3.1.4 Return Value

- If no error occurred, ProbeEzTCP returns 1. When 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 *eztcpenv*, *eztcpwenv*, *eztcpopt* structure.

```
eg.   env_base = (struct spe_env *)malloc(sizeof(struct spe_env) * 256);
      wenv_base = (struct wlan_env *)malloc(sizeof(struct wlan_env) * 256);
      etc_base = (struct etc_opt *)malloc(sizeof(struct etc_opt) * 256);
```

- Above example shows that env_base, wenv_base and etc_base can store 256 ezTCP environment. It means the maximum count of probe result is 256.
- This function take at least 2 seconds to complete running and may take longer if the **local network has a lot of ezTCP.**

3.2 ReadEzTCP

3.2.1 Overview

Read environment value from ezTCP in local network.

3.2.2 Prototype

```
int ReadEzTCP(
        _u8* mac_addr,
        struct spe_env * eztcpenv,
        struct wlan_env * eztcpwenv,
        struct etc_opt * eztcpopt,
        int *nErrNum
);
```

3.2.3 Parameters

- mac_addr
  [in] The ezTCP's MAC address to read environment values.
- eztcpenv
  [out] Pointer to spe_env structure which ezTCP environment values will be stored in.
- eztcpwenv
  [out] Pointer to wlan_env structure which ezTCP wireless environment values will be stored in
- eztcpopt
  [out] Pointer to etc_opt structure which ezTCP etcetera option values will be stored in.
- nErrNum
  [out] Pointer to integer which the error number, when error is occurred.

3.2.4 Return Value

- If no error occurred, *ReadEzTCP* returns 1. When error is 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 *eztcpenv*, *eztcpwenv*, *eztcpopt* structure.

```
eg.  env_base = (struct spe_env *)malloc(sizeof(struct spe_env) * 256);
     wenv_base = (struct wlan_env *)malloc(sizeof(struct wlan_env) * 256);
     etc_base = (struct etc_opt *)malloc(sizeof(struct etc_opt) * 256);
```

- Above example shows that env_base, wenv_base and etc_base can store 256 ezTCP environment. It means the maximum count of probe result is 256.
- This function take at least 2 seconds to complete running.

3.3 WriteEzTCP

3.3.1 Overview

Write environment value to ezTCP specified MAC address in *eztcpenv*.

3.3.2 Prototype

```
int WriteEzTCP(
        struct spe_env * eztcpenv,
        struct wlan_env * eztcpwenv,
        struct etc_opt * eztcpopt,
        int *nErrNum
);
```

3.3.3 Parameters

- eztcpenv
  [in] Pointer to spe_env structure containing environment value to be written.
- eztcpwenv
  [in] Pointer to wlan_env structure containing wireless environment values to be written.
- eztcpopt
  [out] Pointer to etc_opt structure containing etcetera option values to be written.
- nErrNum
  [out] Pointer to integer which the error number, when error is occurred.

3.3.4 Return Value

- If no error occurred, *WriteEzTCP* returns 1. When error is occurred, the return value is EZTCP_ERR(-1) and the error number is stored in n*ErrNum*.

nErrNum	Description
EZTCP_ERR_PWD	If you configured a password, then you should set a password in spe_env.passwd.
EZTCP_ERR_RES	When error occurred during wirte operation, EZTCP_ERR_RES is stored int nErrNum.

3.3.5 Remarks

- Before using this function, you have to call *ProbeEzTCP* function for reliable data transmitting.
- This function takes at least 3 seconds to complete running.

3.4 StatusEzTCP

3.4.1 Overview

Read status value from ezTCP specified MAC address in *eztcpenv*.

3.4.2 Prototype

```
int StatusEzTCP(
        struct spe_env * eztcpenv,
        struct stat_env * eztcpstat,
        int *nErrNum
);
```

3.4.3 Parameters

- eztcpenv
  [in] Pointer to spe_env structure which has target MAC address.
- eztcpstat
  [out] Pointer to stat_env structure which stores ezTCP status information.
- nErrNum
  [out] Pointer to integer which the error number, when error is occurred.

3.4.4 Return Value

- If no error occurred, *StatusEzTCP* returns 1. When error is 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 *ProbeEzTCP* function for reliable data transmitting.
- This function takes at least 2 seconds to complete running.

3.5 ChangePwdEzTCP

3.5.1 Overview

Change or erase password of ezTCP specified MAC address in *eztcpenv*.

3.5.2 Prototype

```
int ChangePwdEzTCP(
        struct spe_env * eztcpenv,
        const char * szChangePwd,
        int *nErrNum
);
```

3.5.3 Parameters

- eztcpenv
  [in] Pointer to spe_env structure which has target MAC address.
- szChangePwd
  [in] Pointer to new password string.
- nErrNum
  [out] Pointer to integer which the error number, when error is occurred.

3.5.4 Return Value

- If no error occurred, *ChangePwdEzTCP* returns 1. When error is 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 *ProbeEzTCP* function for reliable data transmitting.

- This function take at least 3 seconds to complete running and may take longer if the **local network has a lot of ezTCP.**

```
#define CURRENT_PWD "current_password"
#define CHANGE_PWD "new_password"

// eg. Change or erase password
memset(env_base)->passwd, 0, 12);
memcpy(env_base)->passwd, CURRENT_PWD, sizeof(CURRENT_PWD));
memset(szPWD, 0, 12);

// Remark this line to erase password.
memcpy(szPWD, CHANGE_PWD, sizeof(CHANGE_PWD));
nResult = ChangePwdEzTCP((env_base), szPWD, &nErr);

if (nResult == EZTCP_ERR)
{
    if (nErr == EZTCP_ERR_PWD)
        MessageBox(hWnd, "The password is mismatch!",
                         "Change password fail!", MB_OK);
    if (nErr == EZTCP_ERR_RES)
        MessageBox(hWnd, "There is no response of ezTCP!",
                         "Change password fail!", MB_OK);
}
else
{
    MessageBox(hWnd, "The password is changed!",
                     "ezTCP notice!", MB_OK);
}
```

3.6 ResetEzTCP

3.6.1 Overview

The ResetEzTCP function can reset specific ezTCP in network.
Before using this function you have to set a password to ezTCP.

This function is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-200L, EZL-70 (Firmware v1.1k or higher)
Wireless LAN	

3.6.2 Prototype

```
int ResetEzTCP(
        struct spe_env * eztcpenv,
        const char * pwd,
        int *nErrNum
);
```

3.6.3 Parameters

- eztcpenv
  [in] Pointer to spe_env sturuction which has target MAC address.
- pwd
  [in] Pointer to password of ezTCP.
- nErrNum
  [out] Pointer to integer which the error number, when error is occurred.

3.6.4 Return Value

- If no error occurred, *ResetEzTCP* returns 1. When error is occurred, the return value is EZTCP_ERR(-1) and the error number is stored in n*ErrNum*.

nErrNum	Description
EZTCP_ERR_PWD	If you configured a password, then you should set a password in spe_env.passwd.
EZTCP_ERR_RES	When error occurred during wirte operation, EZTCP_ERR_RES is stored int nErrNum.

3.7 CloseTCP

3.7.1 Overview

The CloseTCp function can terminate a TCP/IP connection of specific ezTCP in network. Before using this function you have to set a password to ezTCP.

This function is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-200L, EZL-70 (Firmware v1.1k or higher)
Wireless LAN	

3.7.2 Prototype

```
int CloseTCP(
        struct spe_env * eztcpenv,
        const char * Pwd,
        int *nErrNum
);
```

3.7.3 Parameters

- eztcpenv
  [in] Pointer to spe_env sturuction which has target MAC address.
- pwd
  [in] Pointer to password of ezTCP.
- nErrNum
  [out] Pointer to integer which the error number, when error is occurred.

3.7.4 Return Value

- If no error occurred, *CloseTCP* returns 1. When error is occurred, the return value is EZTCP_ERR(-1) and the error number is stored in n*ErrNum*.

nErrNum	Description
EZTCP_ERR_PWD	If you configured a password, then you should set a password in spe_env.passwd.
EZTCP_ERR_RES	When error occurred during wirte operation, EZTCP_ERR_RES is stored int nErrNum.

3.8 RemoteReadEzTCP

3.8.1 Overview

The RemoteReadEzTCP function can find ezTCP in local or remote network.
Each ezTCP can discriminate by IP address.

This function is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-70, EZL-200L, EZL-220, EZL-200F
Wireless LAN	EZL-300L, EZL-300S, EZL-80 / 80c / 90 / 300W Lite (Firmware v1.3i or higher)

3.8.2 Prototype

```
int RemoteReadEzTCP(
        DWORD ip,
        struct spe_env * eztcpenv,
        struct wlan_env * eztcpwenv,
        struct etc_opt * eztcpopt,
        int * nResultCount,
        int *nErrNum
);
```

3.8.3 Parameters

- ip
  [in] The ezTCP's local ip address to read environment values.
  This parameter should be passed by using Little Endian.
  It changes to Big Endian(Network Byte Order) before use it inside of this function.
- eztcpenv
  [out] Pointer to spe_env structure which ezTCP environment values will be stored in.
- eztcpwenv
  [out] Pointer to wlan_env structure which ezTCP wireless environment values will be stored in
- eztcpopt
  [out] Pointer to etc_opt structure which ezTCP etcetera option values will be stored in.
- nResultCount
  [out] Pointer to integer which the number of founded ezTCP will be stored in.
- nErrNum
  [out] Pointer to integer which the error number, when error is occurred.

3.8.4 Return Value

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

3.8.5 Remarks

- Before using this function, you have to reserve enough space for *eztcpenv*, *eztcpwenv*, *eztcpopt* structure.

```
eg.   env_base = (struct spe_env *)malloc(sizeof(struct spe_env) * 256);
      wenv_base = (struct wlan_env *)malloc(sizeof(struct wlan_env) * 256);
      etc_base = (struct etc_opt *)malloc(sizeof(struct etc_opt) * 256);
```

- Above example shows that env_base, wenv_base and etc_base can store 256 ezTCP environment. It means the maximum count of probe result is 256.

- This function take at least 2 seconds to complete running.

3.9 RemoteWriteEzTCP

3.9.1 Overview

Write environment value to ezTCP specified IP address.

This function is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-70, EZL-200L, EZL-220, EZL-200F
Wireless LAN	EZL-300L, EZL-300S, EZL-80 / 80c / 90 / 300W Lite (Firmware v1.3i or higher)

3.9.2 Prototype

```
int RemoteWriteEzTCP(
        DWORD ip,
        struct spe_env * eztcpenv,
        struct wlan_env * eztcpwenv,
        struct etc_opt * eztcpopt,
        int *nErrNum
);
```

3.9.3 Parameters

- ip
  [in] The ezTCP's local ip address to write environment values.
  This parameter should be passed by using Little Endian.
  It changes to Big Endian(Network Byte Order) before use it inside of this function.

- eztcpenv
  [in] Pointer to spe_env structure containing environment value to be written.

- eztcpwenv
  [in] Pointer to wlan_env structure containing wireless environment values to be written.

- eztcpopt
  [out] Pointer to etc_opt structure containing etcetera option values to be written.

- nErrNum
  [out] Pointer to integer which the error number, when error is occurred.

3.9.4 Return Value

- If no error occurred, *RemoteWriteEzTCP* returns 1. When error is occurred, the return value is EZTCP_ERR(-1) and the error number is stored in n*ErrNum*.

nErrNum	Description
EZTCP_ERR_PWD	If you configured a password, then you should set a password in spe_env.passwd.
EZTCP_ERR_RES	When error occurred during wirte operation, EZTCP_ERR_RES is stored int nErrNum.

3.9.5 Remarks

- Before using this function, you have to call *RemoteReadEzTCP* function for reliable data transmitting.
- This function takes at least 3 seconds to complete running.

3.10 RemoteStatusEzTCP

3.10.1 Overview

Read status value from ezTCP using specified IP address.

This function is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-70, EZL-200L, EZL-220, EZL-200F
Wireless LAN	EZL-300L, EZL-300S, EZL-80 / 80c / 90 / 300W Lite (Firmware v1.3i or higher)

3.10.2 Prototype

```
int StatusEzTCP(
        DWORD ip,
        struct spe_env * eztcpenv,
        struct stat_env * eztcpstat,
        int *nErrNum
);
```

3.10.3 Parameters

- ip
  [in] The ezTCP's local ip address to write environment values.
  This parameter should be passed by using Little Endian.
  It changes to Big Endian(Network Byte Order) before use it inside of this function.

- eztcpenv
  [in] Pointer to spe_env structure which has target ezTCP's environment values.

- eztcpstat
  [out] Pointer to spe_env structure which ezTCP status values will be stored in.

- nErrNum
  [out] Pointer to integer which the error number, when error is occurred.

3.10.4 Return Value

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

3.10.5 Remarks

- Before using this function, you have to call *RemoteReadEzTCP* function for reliable data transmitting.
- This function take at least 2 seconds to complete running.

3.11 RemoteChangePwdEzTCP

3.11.1 Overview

Change or erase password of ezTCP using specified IP address. This function is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-70, EZL-200L, EZL-220, EZL-200F
Wireless LAN	EZL-300L, EZL-300S, EZL-80 / 80c / 90 / 300W Lite (Firmware v1.3i or higher)

3.11.2 Prototype

```
int RemoteChangePwdEzTCP(
        DWORD ip,
        struct spe_env * eztcpenv,
        const char * szChangePwd,
        int *nErrNum
);
```

3.11.3 Parameters

- ip
  [in] The ezTCP's local ip address to change password.
  This parameter should be passed by using Little Endian.
  It changes to Big Endian(Network Byte Order) before use it inside of this function.
- eztcpenv
  [in] Pointer to spe_env structure which has target ezTCP's environment values.
- szChangePwd
  [in] Pointer to new passwords string.
- nErrNum
  [out] Pointer to integer which the error number, when error is occurred.

3.11.4 Remarks

- Before using this function, you have to call *RemoteReadEzTCP* function for reliable data transmitting.
- You have to include *eztcplib.h* in your source code and add library *ws2_32.lib* and ***eztcplib.lib***.
- This function take at least 3 seconds to complete running.

```
#define CURRENT_PWD "current_password"
#define CHANGE_PWD "new_password"

// eg. Change or erase password
DWORD ip;
memset(env_base)->passwd, 0, 12);
memcpy(env_base)->passwd, CURRENT_PWD, sizeof(CURRENT_PWD));
memset(szPWD, 0, 12);

// Remark this line to erase password.
memcpy(szPWD, CHANGE_PWD, sizeof(CHANGE_PWD));

IpAddressControl.GetAddress(ip);

nResult = RemoteChangePwdEzTCP(ip, (env_base), szPWD, &nErr);

if (nResult == EZTCP_ERR)
{
          if (nErr == EZTCP_ERR_PWD)
```

```
                MessageBox(hWnd, "The password is mismatch!",
                                    "Change password fail!", MB_OK);
        if (nErr == EZTCP_ERR_RES)
                MessageBox(hWnd, "There is no response of ezTCP!",
                                    "Change password fail!", MB_OK);
}
else
{
        MessageBox(hWnd, "The password is changed!",
                            "ezTCP notice!", MB_OK);
}
```

3.12 RemoteResetEzTCP

3.12.1 Overview

The RemoteResetEzTCP function can reset specific ezTCP in network.
Before using this function you have to set a password to ezTCP.

This function is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-200L, EZL-70 (Firmware v1.1k or higher)
Wireless LAN	

3.12.2 Prototype

```
int RemoteResetEzTCP(
        DWORD ip,
        struct spe_env * eztcpenv,
        const char * pwd,
        int *nErrNum
);
```

3.12.3 Parameters

- ip
  [in] The ezTCP's local ip address.
  This parameter should be passed by using Little Endian.
  It changes to Big Endian(Network Byte Order) before use it inside of this function.
- eztcpenv
  [in] Pointer to spe_env sturuction which has target MAC address.

- pwd
  [in] Pointer to password of ezTCP.
- nErrNum
  [out] Pointer to integer which the error number, when error is occurred.

3.12.4 Return Value

- If no error occurred, *RemoteResetEzTCP* returns 1. When error is occurred, the return value is EZTCP_ERR(-1) and the error number is stored in n*ErrNum*.

nErrNum	Description
EZTCP_ERR_PWD	If you configured a password, then you should set a password in spe_env.passwd.
EZTCP_ERR_RES	When error occurred during wirte operation, EZTCP_ERR_RES is stored int nErrNum.

3.13 RemoteCloseTCP

3.13.1 Overview

The CloseTCp function can terminate a TCP/IP connection of specific ezTCP in network. Before using this function you have to set a password to ezTCP.

This function is currently considered in below ezTCP products.

LAN type	Product Name
Wired LAN	EZL-50L, EZL-50M, EZL-200L, EZL-70 (Firmware v1.1k or higher)
Wireless LAN	

3.13.2 Prototype

```
int RemoteCloseTCP(
        DWORD ip,
        struct spe_env * eztcpenv,
        const char * Pwd,
        int *nErrNum
);
```

3.13.3 Parameters

- ip
  [in] The ezTCP's local ip address.
  This parameter should be passed by using Little Endian.
  It changes to Big Endian(Network Byte Order) before use it inside of this function.
- eztcpenv
  [in] Pointer to spe_env sturuction which has target MAC address.
- pwd
  [in] Pointer to password of ezTCP.
- nErrNum
  [out] Pointer to integer which the error number, when error is occurred.

3.13.4 Return Value

- If no error occurred, *RemoteCloseTCP* returns 1. When error is occurred, the return value is EZTCP_ERR(-1) and the error number is stored in n*ErrNum*.

nErrNum	Description
EZTCP_ERR_PWD	If you configured a password, then you should set a password in spe_env.passwd.
EZTCP_ERR_RES	When error occurred during wirte operation, EZTCP_ERR_RES is stored int nErrNum.

3.14 GetLibVerEzTCP

3.14.1 Overview

Get library version information.

3.14.2 Prototype

```
int GetLibVerEzTCP( char * szVersion, int len );
```

3.14.3 Parameters

- szVersion
  [out] Pointer to version information string.
- len

[in] The size of szVersion pointer.

3.14.4 Return Value

- If no error occurred, *GetLibVerEzTCP* returns 1. When error is occurred, the return value is EZTCP_ERR(-1).

3.14.5 Remarks

- You have to reserve at least 70bytes space for version information string.