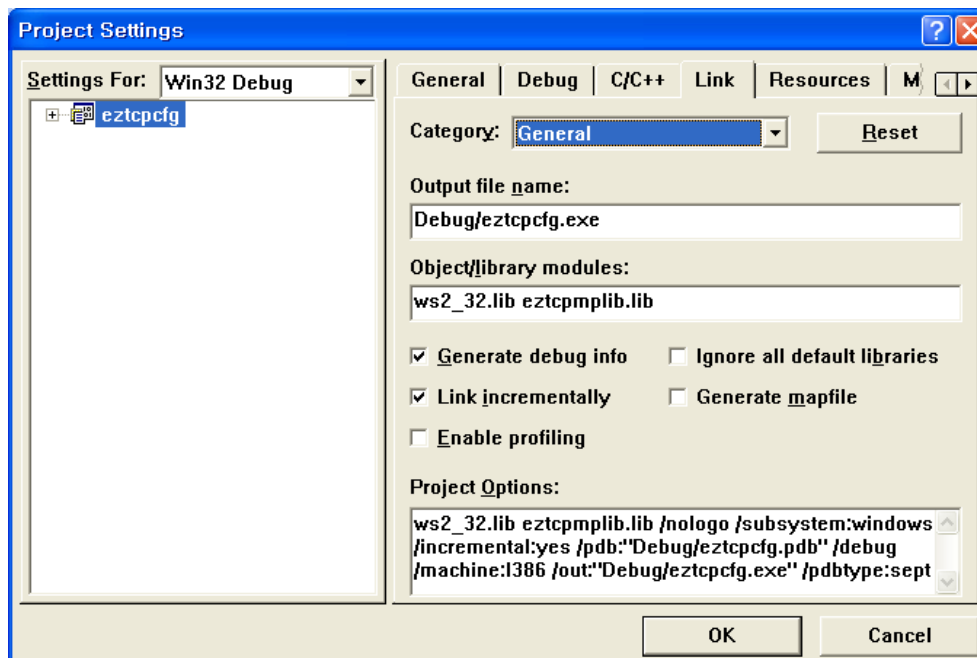


WARNING:

1. All "reserved members" or "not used members" in data structure are NOT allowed use. Please don't use "reserved" or "not used" member.
2. Please check ezTCP's MAC address or IP address before using "write" function.
If you write wrong information to ezTCP, then it does not correctly work .
3. We DO NOT guarantee any damage occurred by illegal use.

■ Introduction

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



port_env

Overview

Basic structure for function probe, write and status that are used in ezCfgr library.

It stored COM port configuration in EZL-400 series.

```
struct port_env
{
    unsigned int databit;      /* 0 : 5bits, 1 : 6bits, 2 : 7bits, 3 : 8bits */
    unsigned int stopbit;      /* 0 : 1bit, 1: 2bit */
    unsigned int parity;        /* none(0), odd(1), even(3) */
    unsigned int flowctrl;      /* none(0), rtscts(1) */
    unsigned int pad0;          // NOT USED. Never modify.
    unsigned int mux_type;      /* T2S(0), ATC(1), COD(2), U2S(3) */
    unsigned int pad1;          // NOT USED. Never modify.
    unsigned int pad2;          // NOT USED. Never modify.

    _u32 local_ip;
    _u32 peer_ip;
    _u16 local_port;
    _u16 peer_port;
    _u32 baud_rate;
    _u16 timeout;
    _u16 water_mark;
};
```

Parameters

databit

[in/out] Serial data bit.

stopbit

[in/out] Serial stop bit.

parity

[in/out] Serial parity bit.

flowctrl

[in/out] Serial flow-control bit.

flowctrl	Description
0	none
1	RTC / CTS

mux_type

[in/out] Mode setting value.

mux_type	Mode	Description
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.

local_ip

[in/out] Local IP address.

This value is only considered when EZL-400 use multi-ip.

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)) )
```

peer_ip;

[in/out] Target IP address.

This value is only considered in COD or U2S mode.

local_port

[in/out] Local IP address.

This value is only considered in T2S or U2S mode.

peer_port

[in/out] Target port number.

This value is only considered in COD or U2S mode.

baud_rate

[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.

water_mark

[in/out] Amount of data that can allow to start connection.

This value is only considered in COD or U2S mode.

ezl_env

Overview

Basic structure for function probe, write and status that are used in ezCfg library.

```
struct ezl_env
{
    u8 mac_addr[6];                // Ethernet address. Never modify.
    u8 secure[18];                // Reserved. Never modify.
    u8 major, minor;              // Reserved. Never modify.
    u8 pad0[2];                   // Reserved. Never modify.
    unsigned int upgrate;          // Reserved. Never modify.
    unsigned int multi_ip;
    unsigned int ip4_type;
    unsigned int ip6_type;        // Reserved. Never modify.
    unsigned int arp;
    unsigned int telnet;
    unsigned int http;            // Reserved. Never modify.
    unsigned int eapol;           // Reserved. Never modify.
    unsigned int ssl;             // Reserved. Never modify.
    unsigned int telcom;          // Reserved. Never modify.
    unsigned int en_comment;      // Reserved. Never modify.
    unsigned int en_rcfg;         // Reserved. Never modify.
    unsigned int rcfg;

    unsigned int pad2;            // NOT USED. Never modify.
    unsigned int pad3;            // NOT USED. Never modify.

    u32 uniq_ip;
    u32 net_mask;
    u32 gate_ip;
    u8 poe_uid[32];
    u8 poe_pwd[12];
    u8 ezl_pwd[12];
    u8 comment[32];
    u8 ext_opt[12];               // Reserved. Never modify.
    u8 pad4[4];                  // NOT USED. Never modify.
    u8 pad5[160];                 // NOT USED. Never modify.

    struct port_env port[8];
    u8 crc[4];                    // Reserved. Never modify.
};
```

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.

eg. The MAC address 0030f9123456 is stored in eth_addr array like this.

eth_addr[0] = 0x00, eth_addr[1] = 0x30, eth_addr[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.

multi_ip

[in/out] If this value is non zero, the COM ports in ezTCP can have own ip address.

In this case "uniq_ip" parameter is ignored.

ip4_type

[in/out]

ip4_type	Mode	Description
0	Static IP	ezTCP has a static IP address.
1	DHCP	ezTCP try to receive a IP address from a DHCP server.
2	PPPoE	ezTCP use a PPPoE protocol. it is need a "poe_uid" and "poe_pwd" parameter.

arp

[in/out] If this value is non zero, arp protocol is enabled.

telnet

[in/out] If this value non zero, telnet connection is enabled for device setting.

rcfg

[in/out] If this value is non zero, remote configuration is enabled.

uniq_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.

ezi_pwd

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

comment

[in/out] You can enter a description to a ezTCP.

port

[in/out] The data structure array of "port_env".

It stored COM port configuration in EZL-400 series.

Remarks

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

■ mp_stat_env

Overview

Basic status structure has status value from function status that are used in EzCfg library.

```
struct mp_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    sio1_rx;
    long unsigned int    sio1_tx;
    long unsigned int    sio2_rx;
    long unsigned int    sio2_tx;
    long unsigned int    sio3_rx;
    long unsigned int    sio3_tx;
    long unsigned int    sio4_rx;
    long unsigned int    sio4_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];
};
```

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 function will make EzTCP reboot.

uptime_hour

[out] ezTCP's alive hours since the last boot up of ezCFG.

WriteEzTCP function will make EzTCP reboot.

uptime_minute

[out] ezTCP's alive minutes since the last boot up of ezCFG.

WriteEzTCP function will make EzTCP reboot.

uptime_second

[out] ezTCP's alive seconds since the last boot up of ezCFG.

WriteEzTCP function will make EzTCP reboot.

sio1_rx

[out] Received bytes from serial connection via COM1.

sio1_tx

[out] Transmitted bytes to serial connection via COM1.

sio2_rx

[out] Received bytes from serial connection via COM2.

sio2_tx

[out] Transmitted bytes to serial connection via COM2

sio3_rx

[out] Received bytes from serial connection via COM3.

sio3_tx

[out] Transmitted bytes to serial connection via COM3

sio4_rx

[out] Received bytes from serial connection via COM4.

sio4_tx

[out] Transmitted bytes to serial connection via COM4

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.

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.

■ ProbeMPEzTCP

Overview

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

Prototype

```
int ProbeMPEzTCP(struct ezl_env * eztcpenv,  
                 int * nResultCount,  
                 int * nErrNum);
```

Parameters

eztcpenv

[out] Pointer to ezl_env structure which ezTCP environment 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.

Return value

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

Remarks

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

eg. *env_base = (struct ezl_env *)malloc(sizeof(struct ezl_env) * 256);*

Above example shows that env_base can store 256 ezTCP environment value. It means the maximum count of probe result is 256.

You have to include *eztcpmplib.h* in your source code and add library *ws2_32.lib* and *eztcpmplib.lib*.

This function take at least 2 seconds to complete running and may take longer if the local network has a lot of ezTCP.

WriteMPEzTCP

Overview

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

Prototype

```
int WriteMPEzTCP(struct ezl_env * eztcpenv,
                 int *nErrNum);
```

Parameters

eztcpenv

[in] Pointer to ezl_env structure containing environment value to be written.

nErrNum

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

Return value

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

nErrNum	Description
EZTCP_ERR_PWD	If you seted a password to ezTCP, then you should set a password in spe_env.passwd.
EZTCP_ERR_RES	When error occurred during wirte environment values, EZTCP_ERR_RES is stored int nErrNum.

Remarks

Before using this function, you have to call *ProbeMPEzTCP* function for reliable data transmitting.

You have to include *eztcpmplib.h* in your source code and add library *ws2_32.lib* and *eztcpmplib.lib*.

This function take at least 3 seconds to complete running and may take longer if the

local network has a lot of ezTCP.

■ StatusMPEzTCP

Overview

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

Prototype

```
int StatusMPEzTCP(struct ezl_env * eztcpenv,
                  struct mp_stat_env * eztcpstat,
                  int *nErrNum);
```

Parameters

eztcpenv

[in] Pointer to ezl_env structure which has target MAC address.

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.

Return value

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

Remarks

Before using this function, you have to call *ProbeMPEzTCP* function for reliable data transmitting.

You have to include *eztcpmplib.h* in your source code and add library *ws2_32.lib* and *eztcpmplib.lib*.

This function take at least 2 seconds to complete running and may take longer if the local network has a lot of ezTCP.

■ ChangePwMPEzTCP

Overview

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

Prototype

```
int ChangePwMPEzTCP(    struct ezl_env * eztcpenv,
                        const char * szChangePw,
                        int *nErrNum);
```

Parameters

eztcpenv

[in] Pointer to ezl_env structure which has target MAC address.

szChangePw

[in] Pointer to new passwords string.

nErrNum

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

Return value

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

Remarks

Before using this function, you have to call *ProbeMPEzTCP* function for reliable data transmitting.

You have to include *eztcpmplib.h* in your source code and add library *ws2_32.lib* and *eztcpmplib.lib*.

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

eg.

```
#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 = ChangePwdMPEzTCP((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);
}
```


RemoteReadMPEzTCP

Overview

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

Prototype

```
int RemoteReadMPEzTCP(DWORD ip,
                      struct ezl_env * eztcpenv,
                      int * nResultCount,
                      int * nErrNum);
```

Parameters

ip

[in] The ezTCP's local ip address to read environment values.

Do not use MK_IP macro, because "htonl" function is used before use "ip" parameter.

eztcpenv

[out] Pointer to ezl_env structure which ezTCP environment 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.

Return value

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

Remarks

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

eg. *env_base = (struct ezl_env *)malloc(sizeof(struct ezl_env) * 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.

You have to include *eztcpmplib.h* in your source code and add library *ws2_32.lib* and *eztcpmplib.lib*.

This function take at least 2 seconds to complete running and may take longer if the local network has a lot of ezTCP.

RemoteWriteMPEzTCP

Overview

Write environment value to ezTCP specified IP address.

Prototype

```
int RemoteWriteMPEzTCP(DWORD ip,
                        struct ezl_env * eztcpenv,
                        int *nErrNum);
```

Parameters

ip

[in] The ezTCP's local ip address to write environment values.

Do not use MK_IP macro, because "htonl" function is used before use "ip" parameter.

eztcpenv

[in] Pointer to ezl_env structure containing environment value to be written.

nErrNum

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

Return value

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

nErrNum	Description
EZTCP_ERR_PWD	If you seted a password to ezTCP, then you should set a password in spe_env.passwd.
EZTCP_ERR_RES	When error occurred during wirte environment values, EZTCP_ERR_RES is stored int nErrNum.

Remarks

Before using this function, you have to call *RemoteReadMPEzTCP* function for reliable data transmitting.

You have to include *eztcpmplib.h* in your source code and add library *ws2_32.lib* and *eztcpmplib.lib*.

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

■ RemoteStatusMPEzTCP

Overview

Read status value from ezTCP using specified IP address.

Prototype

```
int RemoteStatusMPEzTCP(DWORD ip,
                        struct ezl_env * eztcpenv,
                        struct mp_stat_env * eztcpstat,
                        int *nErrNum);
```

Parameters

ip

[in] The ezTCP's local ip address to write environment values.

Do not use MK_IP macro, because "htonl" function is used before use "ip" parameter.

eztcpenv

[in] Pointer to ezl_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.

Return value

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

Remarks

Before using this function, you have to call *RemoteReadMPEzTCP* function for reliable data transmitting.

You have to include *eztcpmplib.h* in your source code and add library *ws2_32.lib* and

eztcpmplib.lib.

This function take at least 2 seconds to complete running and may take longer if the local network has a lot of ezTCP.

■ RemoteChangePwdMPEzTCP

Overview

Change or erase password of ezTCP using specified IP address.

Prototype

```
int RemoteChangePwdMPEzTCP(DWORD ip,
                             struct ezl_env * eztcpenv,
                             const char * szChangePwd,
                             int *nErrNum);
```

Parameters

ip

[in] The ezTCP's local ip address to change password.

Do not use MK_IP macro, because "htonl" function is used before use "ip" parameter.

eztcpenv

[in] Pointer to ezl_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.

Return value

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

Remarks

Before using this function, you have to call *RemoteChangePwdMPEzTCP* function for reliable data transmitting.

You have to include *eztcpmplib.h* in your source code and add library *ws2_32.lib* and

eztcpmplib.lib.

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

```
eg.

#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 = RemoteChangePwdMPEzTCP(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);
}
```


■ GetLibVerMPEzTCP

Overview

Get library version information.

Prototype

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

Parameters

szVersion

[out] Pointer to version information string.

len

[in] The length of *szVersion* pointer. It is at least 125.

Return value

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

Remarks

You have to reserve at least 125bytes space for version information string.