

# Ethernet IO Programming Manual

Design Gateway Co., Ltd.

*Rev 1.1*

*(PD0401-6-02-02E)*

\*\*\* Please read this manual carefully before using Ethernet IO \*\*\*



## Revision History

Revision	Date	Detail of change
1.0	5 August 2004	Initial Release
1.1	9 May 2005	Insert timing diagram and Update function of EtherIODll.dll for support Visual Basic.

## Table of Contents

1. How To Setup.....	1
1.1. Minimum Requirement.....	1
2. Contents in Software Directory.....	1
2.1. Description of Library Files .....	1
2.2. Communicate with Ethernet IO .....	2
3. Example of using Ethernet IO Library .....	2
3.1. Include Ethernet IO Library.....	2
3.2. Create New Class for Receiving Data .....	4
4. Running Example Application.....	18
4.1. Setting Up.....	18
4.2. Running Example Program.....	18
5. Prototype Functions .....	19
5.1. Communication Functions .....	19
5.2. I/O Port Functions .....	23
5.3. Parallel I/O Functions.....	25
5.4. Special Functions .....	29
6. Reference.....	31

# 1. How To Setup

## 1.1. Minimum Requirement

### 1 Personal Computer

Operating System	Windows XP, Windows 2000
Space requirement	10 Mbytes
CPU	300 MHz or higher Pentium PC or compatible PC
LAN interface card	10/100 Mbit.
Development program	Microsoft Visual C++ 6.0

### 2 Ethernet IO Board

Board version	1.0
Firmware Version	2.0
DLL version	1.0
Library version	1.0

## 2. Contents in Software Directory

- CFG\_APP Configuration application and source code
- Library Software library (DLL)

### 2.1. Description of Library Files

File	Detail
EtherIODll.dll	DLL file (use when executing)
EtherIODll.lib	Library file (use when compile)
EtherIOComm.h	Library header file

Ethernet IO development kit provides “Extension” DLL library. User can use this library with Microsoft Visual C++ and Visual Basic (version 6 recommended). For now it does not support Delphi etc...

## 2.2. Communicate with Ethernet IO

Ethernet IO's firmware is service network(LAN) communication and serial (RS232 or RS485). In network communication service, Ethernet IO uses TCP/IP port 4025 for serial communication and map uses 4026 for "Command service". User can communicate directly to serial port on Ethernet IO without this DLL. In case of using IO command, user must use this DLL.

## 3. Example of using Ethernet IO Library

This topic describes how to use DLL in user's application, just like source codes in configuration program do. However there is no deep details on coding. User should have some skill in Microsoft Visual C++ programming.

### 3.1. Include Ethernet IO Library

- 1). Open new MFC application, application name is "example" and choose dialog base program (another option is default).
- 2). After create MFC application, copy Library file "EtherIODll.dll" and "EtherIODll.lib" to "Debug" and "Release" directory.
- 3). Copy library header file "EtherIOComm.h" to project directory.
- 4). In Visual C++ GUI select menu project->Add To Project->File

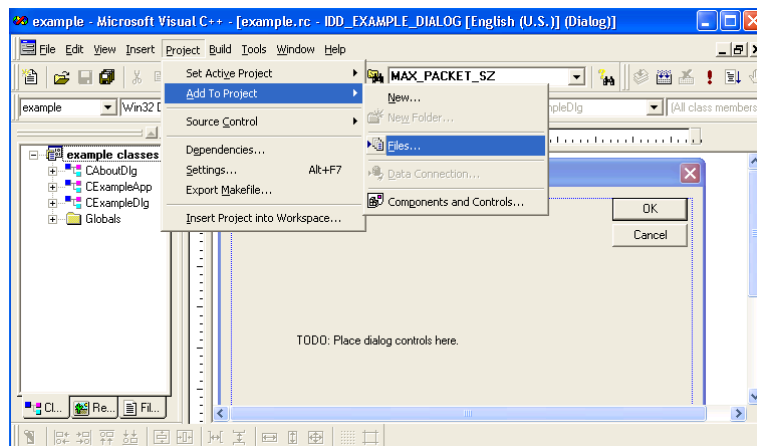


Figure 3-1 Adding EtherIOComm class

Add file “EtherIOComm.h” from project directory. This will automatically add EtherIOComm class to this project. EtherIOComm class contains many functions please refer to [Prototype Functions](#) for more detail in each class.

- 5). Add link to Ethernet IO’ s library file “EtherIODll.lib”, by selecting menu project->Setting then click “Link” tap.

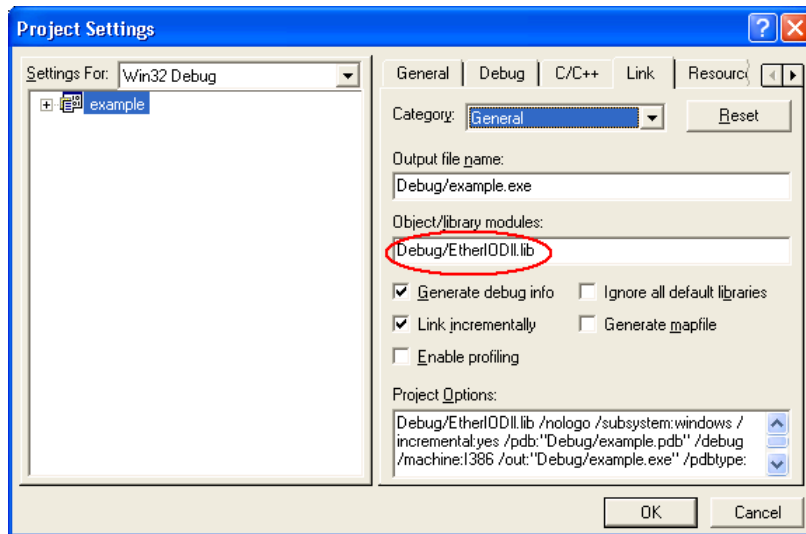


Figure 3-2. Adding link to Ethernet IO library file in debug version

- 6). Click at “Setting For” to change to release mode.

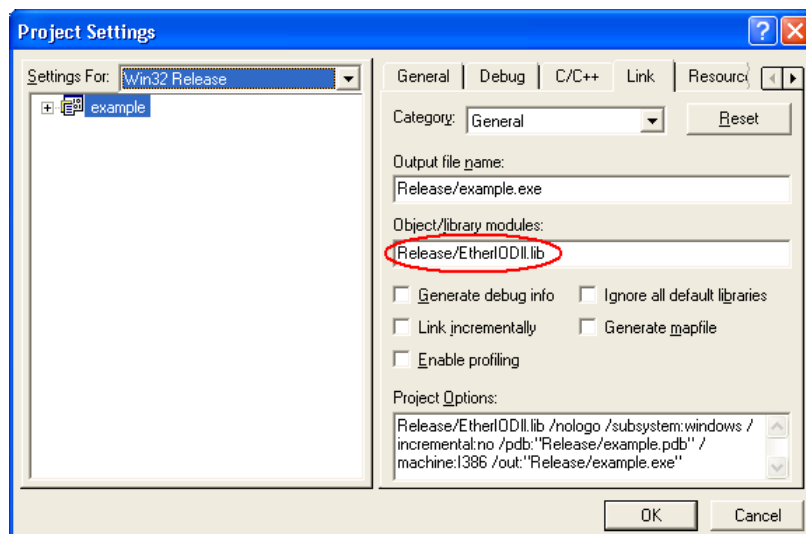


Figure 3-3. Adding link to Ethernet IO library file in Release version

### 3.2. Create New Class for Receiving Data

Now we are creating new class that inherits from “EtherIOComm” class. Why we need to do this? Because if user wants to receive incoming data from serial port on Ethernet IO kit, user must create inherit class or user can use “EtherIOComm” class directly, but cannot receive incoming data. Please note that all bold-source codes in this document are added codes and important codes.

- 1). Add a new class that inherits form “EtherIOComm” class, by selecting menu Insert>New Class

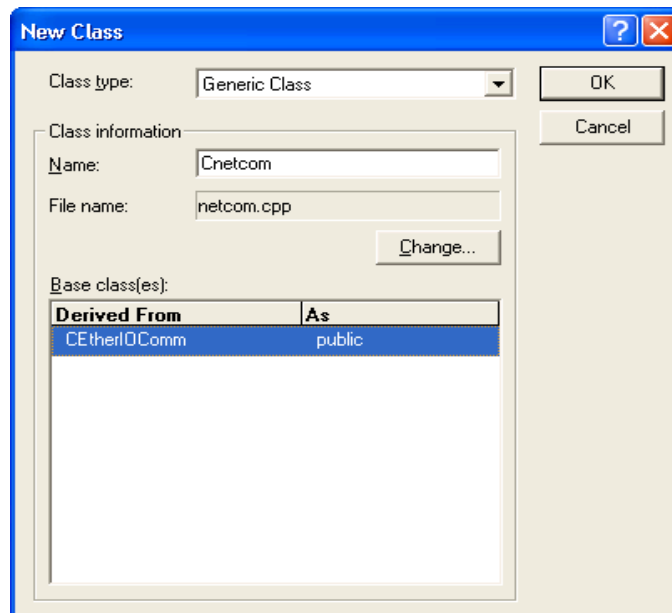


Figure 3-4 Adding inherit class

- 2). In header file of **Cnetcom** class please modify it as below

```

#define WM_UPDATE_CONNECTION      WM_USER+0x1234
#define WM_ON_DATA_RECEIVE       WM_USER+0x1235

class Cnetcom : public CEtherIOComm
{
public:
    Cnetcom();
    virtual ~Cnetcom();

    void SetMessageWindow(CEdit* pMsgCtrl);
    void SetStatusMsg(CString *pStatusMsg);
    void AppendMessage(LPCTSTR strText );
    virtual void OnDataReceived(const LPBYTE lpBuffer, DWORD dwCount);
    virtual void OnEvent(UINT uEvent);

protected:
    void DisplayData(const LPBYTE lpData, DWORD dwCount, const SockAddrIn& sfrom);

    CEdit* m_pMsgCtrl;
    CString *m_StatusMsg;
    CString m_ReceiveFileName;
    BOOL m_SaveReceiveToFile;
};

```

3). So this example program contains below classes

Class	Detail
CAboutDlg	Information dialog of this application
CEtherIOComm	<b>Ethernet IO DLL</b>
CExampleApp	Main application class
Cnetcom	Inherit from <b>Ethernet IO DLL</b> for communicate with Ethernet IO board

4). Then add the following function source codes to class “Cnetcom”

```

////////////////////////////////////
// Construction/Destruction
////////////////////////////////////
Cnetcom::Cnetcom()
{
    m_pMsgCtrl = NULL;
    m_StatusMsg = NULL;
}
Cnetcom::~Cnetcom()
{
}

#include <atlconv.h>
////////////////////////////////////
// Summary:   Show the input data to Receive edit box
// Return: -/
////////////////////////////////////
void Cnetcom::DisplayData(const LPBYTE lpData, DWORD dwCount, const SockAddrIn& sfrom)
{
    CString strData;
    memcpy(strData.GetBuffer(dwCount), A2CT((LPSTR)lpData), dwCount);

    strData.ReleaseBuffer();
    if (!sfrom.IsNull())
    {
        LONG uAddr = sfrom.GetIPAddr();
        BYTE* sAddr = (BYTE*) &uAddr;
        short nPort = ntohs( sfrom.GetPort() );    // show port in host format...
        CString strAddr;
    }
}

```

```

// Address is stored in network format...
strAddr.Format(_T("%u.%u.%u.%u (%d)>"),
              (UINT)(sAddr[0]), (UINT)(sAddr[1]),
              (UINT)(sAddr[2]), (UINT)(sAddr[3]), nPort);

    strData = strAddr + strData;
}

////////////////////////////////////

AppendMessage( strData );
}

////////////////////////////////////

// Summary:   Insert data to receive edit box
// Return: -
////////////////////////////////////

void Cnetcom::AppendMessage(LPCTSTR strText )
{
    if (NULL == m_pMsgCtrl)
        return;

    if (::IsWindow( m_pMsgCtrl->GetSafeHwnd() ))
    {
        int nLen = m_pMsgCtrl->GetWindowTextLength();
        if(nLen >= (m_pMsgCtrl->GetLimitText()-10)){
            m_pMsgCtrl->SetSel(0, -1);
            m_pMsgCtrl->ReplaceSel( " " );
            TRACE("Clear receive edit\n");
            return;
        }
        m_pMsgCtrl->SetSel(nLen, nLen);
        m_pMsgCtrl->ReplaceSel( strText );
    }
}

```

```

////////////////////////////////////
// Summary:   Set the receive edit box
// Return: -
////////////////////////////////////
void Cnetcom::SetMessageWindow(CEdit* pMsgCtrl)
{
    m_pMsgCtrl = pMsgCtrl;
}

////////////////////////////////////
// Summary:   Set status Message output
// Return: -
////////////////////////////////////
void Cnetcom::SetStatusMsg(CString *pStatusMsg)
{
    m_StatusMsg = pStatusMsg;
}

////////////////////////////////////
// Summary:   Data receive operation (using thread)
// Return: -
////////////////////////////////////
void Cnetcom::OnDataReceived(const LPBYTE lpBuffer, DWORD dwCount)
{
    static UINT byreceive= 0;
    byreceive += dwCount;
    TRACE("Data receive= %d\n!",byreceive);

    SockAddrIn saddr_in;
    LPBYTE lpData = lpBuffer;
    //////////////////////////////////////
    // Write it to file

```

```

if(m_SaveReceiveToFile==TRUE){

    CFile Rfile(m_ReceiveFileName ,
                CFile::modeCreate | CFile::modeWrite | CFile::modeNoTruncate);

    Rfile.SeekToEnd();
    Rfile.Write( lpBuffer,dwCount);
    Rfile.Close();
}
else{
    //////////////////////////////////////
    // Display data to message list
    DisplayData( lpData, dwCount, saddr_in );
}
return;
}

////////////////////////////////////
// Summary:   Send message to parent window to indicate connection status
// Return: -
////////////////////////////////////
void Cnetcom::OnEvent(UINT uEvent)
{
    TRACE("ONEVENT\n");

    if (NULL == m_pMsgCtrl)
        return;

    CWnd* pParent = m_pMsgCtrl->GetParent();
    if (!::IsWindow( pParent->GetSafeHwnd()))
        return;

    if(NULL == m_StatusMsg)
        return;
}

```

```

switch( uEvent )
{
    case EVT_CONSUCCESS:
        m_StatusMsg->Format("CONNECTION_ESTABLISHED");
        break;
    case EVT_CONFAILURE:
        m_StatusMsg->Format("CONNECTION_FAILED");
        break;
    case EVT_CONDROP:
        m_StatusMsg->Format("CONN_ABADON");
        break;
    case EVT_ZEROLENGTH:
        m_StatusMsg->Format("ZERO_LENGTH_MSG");
        break;
    default:
        m_StatusMsg->Format("UNKNOW_ERROR");
        break;
}
AfxGetMainWnd()->PostMessage( WM_UPDATE_CONNECTION, uEvent, (LPARAM) this);
}

```

Description of each function

Function	Detail
SetStatusMsg	Set output status message
SetMessageWindow	Set receiving edit box
AppendMessage	Append data to receiving edit box
OnDataReceived	Data receiving operation (by using thread)
OnEvent	Send message to parent window to indicate connection status
DisplayData	Show the input data to receiving edit box

- 5). Add Edit Box, static text and “Send” button, then change caption on “OK” button to “Connect”

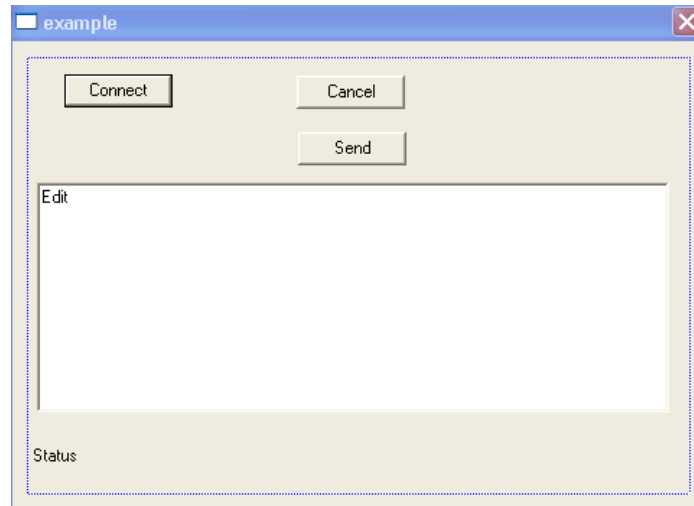


Figure 3-5 Dialog of this program

- 6). Double click on “Send” button to add “OnButton1” function
- 7). Add member variable to Edit box and static text in CExampleDlg.h (CExampleDlg class)

```
#include "netcom.h"
class CExampleDlg : public CDialog
{
// Construction
public:
    CExampleDlg(CWnd* pParent = NULL);    // standard constructor

    Cnetcom    m_EthernetIO;
    CString m_StatusStr;
    CEdit m_ReceiveEditCtrl;

// Dialog Data
   //{{AFX_DATA(CExampleDlg)
```

```

enum { IDD = IDD_EXAMPLE_DIALOG };

    // NOTE: the ClassWizard will add data members here
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CExampleDlg)
protected:
virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
//}}AFX_VIRTUAL

// Implementation
protected:
    HICON m_hIcon;

    // Generated message map functions
//{{AFX_MSG(CExampleDlg)
virtual BOOL OnInitDialog();
afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
afx_msg void OnPaint();
afx_msg HCURSOR OnQueryDragIcon();
virtual void OnOK();
virtual void OnCancel();
afx_msg void OnButton1();
//}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

- 8). Map member variable to Edit box and static text. Resource ID is “IDC\_EDIT1” and “IDC\_STATIC1”

```

void CExampleDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CExampleDlg)
        // NOTE: the ClassWizard will add DDX and DDV calls here
   //}}AFX_DATA_MAP
    DDX_Control(pDX, IDC_EDIT1, m_ReceiveEditCtrl);
    DDX_Text(pDX, IDC_STATIC1, m_StatusStr);
}

```

9). In dialog windows, double click on “Connect” button to add “OnOK” function.

Then add below source code

This function will make TCP/IP connection to Ethernet IO board in default IP.

```

void CExampleDlg::OnOK()
{
    CString IPServerStr;
    bool bSuccess;
    IPServerStr.Format("192.168.11.241");

    //connecting
    bSuccess = m_EthernetIO.ConnectTo( IPServerStr ); // TCP

    //if connecting sucesess Show message
    if (bSuccess && m_EthernetIO.WatchComm())
    {

        m_StatusStr = _T("Connection established with EtherIO: ") + IPServerStr;
        GetDlgItem(IDOK)->EnableWindow( FALSE );

        UpdateData(FALSE);
    }else{

```

```

        int error = WSAGetLastError();
        TRACE("socket error = 0x%X", error);
        m_StatusStr.Format("Connection fail");
        UpdateData(FALSE);
    }
    //CDialog::OnOK();
}

```

10). Add below source code to “OnInitDialog()” function

Adding code will set status message and receive control to **Edit box** control. Then Enable “**Connect**” button.

```

BOOL CExampleDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // Add "About..." menu item to system menu.

    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }
}

```

```

    }

}

// Set the icon for this dialog. The framework does this automatically
// when the application's main window is not a dialog
SetIcon(m_hIcon, TRUE);           // Set big icon
SetIcon(m_hIcon, FALSE);        // Set small icon

// TODO: Add extra initialization here

// Initialize socket manager
m_EthernetIO.SetMessageWindow( &m_ReceiveEditCtrl );
m_EthernetIO.SetStatusMsg( &m_StatusStr);
m_EthernetIO.SetServerState( false ); // run as client
m_EthernetIO.SetSmartAddressing( false ); // always send to server
GetDlgItem(IDOK)->EnableWindow( TRUE );

return TRUE; // return TRUE unless you set the focus to a control
}

```

- 11). Double click on “Cancel” button in dialog to go to “OnCancel” function. Then add below source code

```

void CExampleDlg::OnCancel()
{
    m_EthernetIO.StopComm();
    CDialog::OnCancel();
}

```

12). Double click on “Send” button in dialog to add “OnButton1” function, then add below source code

```
void CExampleDlg::OnButton1()
{
    // TODO: Add your control notification handler code here
    char message[]="hello world\n\r";

    if(m_EthernetIO.IsOpen()){
        if(m_EthernetIO.WriteComm((unsigned char*) &message[0], sizeof(message),
INFINITE) <= 0L){
            m_StatusStr.Format("Send fail");
            UpdateData(FALSE);
        }
    }
}
}
```

13).The last step is adding below source code to function “InitInstance()” in CexampleApp class. This will enable Winsock function on this program.

```
BOOL CExampleApp::InitInstance()
{
    AfxEnableControlContainer();

    //Initialize winsock
    #define WSA_VERSION MAKEWORD(2,0)

    WSADATA          WSAData = { 0 };
    if ( 0 != WSASStartup( WSA_VERSION, &WSAData ) )
    {
        // Tell the user that we could not find a usable
        // WinSock DLL.
    }
}
```

```

        if ( LOBYTE( WSADATA.wVersion ) != LOBYTE(WSA_VERSION) ||
            HIBYTE( WSADATA.wVersion ) != HIBYTE(WSA_VERSION) )
            ::MessageBox(NULL, _T("Incorrect version of Winsock.dll found"),
                _T("Error"), MB_OK);

        WSACleanup( );
        return FALSE;
    }

    // Standard initialization
    // If you are not using these features and wish to reduce the size
    // of your final executable, you should remove from the following
    // the specific initialization routines you do not need.

#ifdef _AFXDLL
    Enable3dControls();           // Call this when using MFC in a shared DLL
#else
    Enable3dControlsStatic();    // Call this when linking to MFC statically
#endif

    CExampleDlg dlg;
    m_pMainWnd = &dlg;
    int nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    {
        // TODO: Place code here to handle when the dialog is
        // dismissed with OK
    }
    else if (nResponse == IDCANCEL)
    {
        // TODO: Place code here to handle when the dialog is
        // dismissed with Cancel
    }

```

```
// Since the dialog has been closed, return FALSE so that we exit the
// application, rather than start the application's message pump.
return FALSE;
}
```

14). Then compile and run this program. If you find any error message during compiling please refer to example source code in development CD (X:\Software\example.zip : X is CD Rom Drive).

## 4. Running Example Application

After making example application finished, we need to test it with hardware board. This application is simple for sending and receiving data from serial port on Ethernet IO board. If user wants to use IO on Ethernet IO board, user can refer to [Prototype Functions](#).

### 4.1. Setting Up

- Connect “**Cross cable**” between Ethernet IO and host PC.
- Connect “**Serial cable**” of Ethernet IO to **COM1** port on PC and power on Ethernet IO board.
- Open **HyperTerminal** application and set serial port to **9600 bits per second, data bits = 8, Parity = None, Stop bits = 1, Flow Control = Hardware and COM1**

### 4.2. Running Example Program

- Open example application. Then click on “**Connect**” button. If program can connect to Ethernet IO board, the message “Connection established with EtherIO: 192.168.11.241” will appear at bottom of dialog.
- Click “**Send**” button. The message “hello world” will appear on **HyperTerminal** application.
- If user input any key on **HyperTerminal**, so that key will appear on example application too.

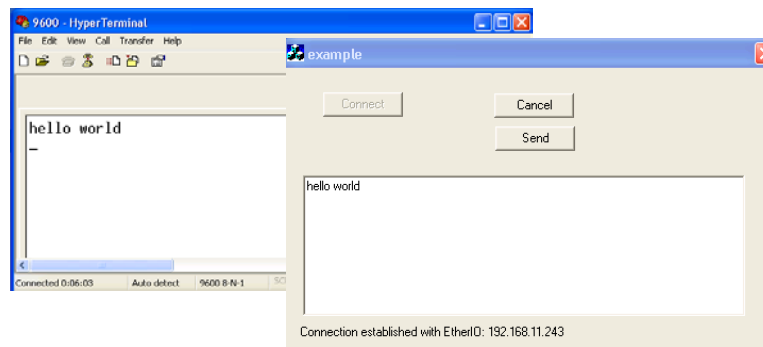


Figure 6 Running example application

- If user click on “Cancel” button, this program will disconnect from Ethernet IO and then close application.

## 5. Prototype Functions

This topic describes all functions that provided by Ethernet IO DLL. Ethernet IO development provides function library in DLL module (EtherIODll). It is easy for user to use it without any knowledge about Winsock. User can include this DLL module to user’s MFC project and develop application base on example program.

EtherIODll.DLL contains 2 classes as below

- **SockAddrIn** contains utility functions for Winsock programming (not necessary for communication with Ethernet IO board).
- **CEtherIOComm** is the main class for sending and receiving data from Ethernet IO. This class uses multithread working, one for sending packet data and another one thread will wait for incoming packet. User must override some functions in this class to handle incoming packet.

### 5.1. Communication Functions

Ethernet IO board is a network embedded board. Another application can control this board via TCP/IP protocol. Library also contains function for communication with Ethernet IO board.

### 1 ConnectTo

<b>Format</b>	bool ConnectTo(LPCTSTR strDestination);	
<b>Function</b>	Open communication with Ethernet IO board	
<b>Parameter</b>	LPCTSTR strDestination	IP address in string definition
<b>Return value</b>	(BOOL) TRUE	Open communication successful
	(BOOL) FALSE	Fail to open communication. Check LAN cable or power supply on Ethernet IO board

### 2 WatchComm

<b>Format</b>	bool WatchComm( void );	
<b>Function</b>	Create communication monitor thread for receiving data from Ethernet IO	
<b>Parameter</b>	-	
<b>Return value</b>	(BOOL) TRUE	Communication monitor thread created successfully
	(BOOL) FALSE	Communication monitor thread creation failed

### 3 WriteComm

<b>Format</b>	DWORD WriteComm( const LPBYTE lpBuffer, DWORD dwCount, DWORD dwTimeout );	
<b>Function</b>	Send data to UART port on Ethernet IO board.	
<b>Parameter</b>	const LPBYTE lpBuffer	Data to send in pointer format
	DWORD dwCount	Size of data to send
	DWORD dwTimeout	Specified time out for sending data. This parameter can be infinity time (INFINITE)
<b>Return value</b>	DWORD	Size of data that can send to Ethernet IO board





## 5.2. I/O Port Functions

In IO programming, user can control input/output of all ports. Ethernet IO supports this function. See below details for I/O port function.

### 1 SetBit

<b>Format</b>	BOOL SetBit ( unsigned char port, unsigned char pin );	
<b>Function</b>	Set status of specified pin to logic high.	
<b>Parameter</b>	unsigned char port	I/O port in Ethernet IO board. User must specified port in this definition  PORT_A      0x0020 PORT_B      0x0024 PORT_C      0x0028 PORT_D      0x002C PORT_E      0x0030 PORT_F      0x0034 PORT_G      0x0038
	unsigned char pin	Pin number to set. Can be 0 to 7
<b>Return value</b>	(BOOL) TRUE	Set port command success
	(BOOL) FALSE	Fail to set port on Ethernet IO board.

### 2 ClearBit

<b>Format</b>	BOOL ClearBit( unsigned char port, unsigned char pin );	
<b>Function</b>	Set status of specified pin to logic low.	
<b>Parameter</b>	unsigned char port	I/O port in Ethernet IO board
	unsigned char pin	Pin number to set. Can be 0 to 7
<b>Return value</b>	(BOOL) TRUE	Clear port command success
	(BOOL) FALSE	Fail to clear port on Ethernet IO board.





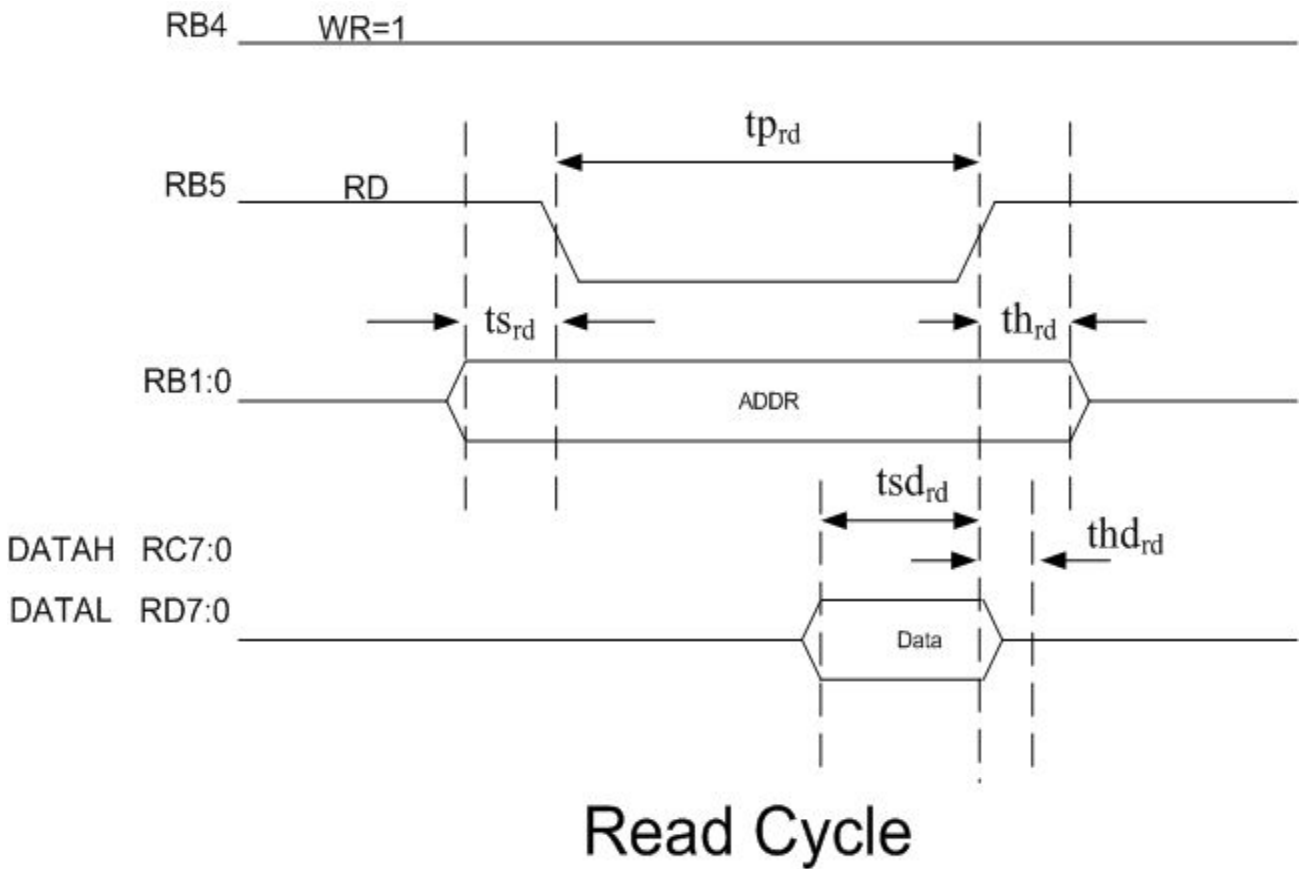


Figure 5-1 Read Timing Diagram

Table 5-1 Read Data timing diagram

Symbol	Parameter	Min	Max	Unit
$tp_{rd}$	Period of Read Data	120	-	nS.
$ts_{rd}$	Setup Time of Read Data	60	-	nS.
$th_{rd}$	Hold Time of Read Data	90	-	nS.
$tsd_{rd}$	Setup Time of Data Read	60	-	nS.
$thd_{rd}$	Hold Time of Data Read Data	60	-	nS.

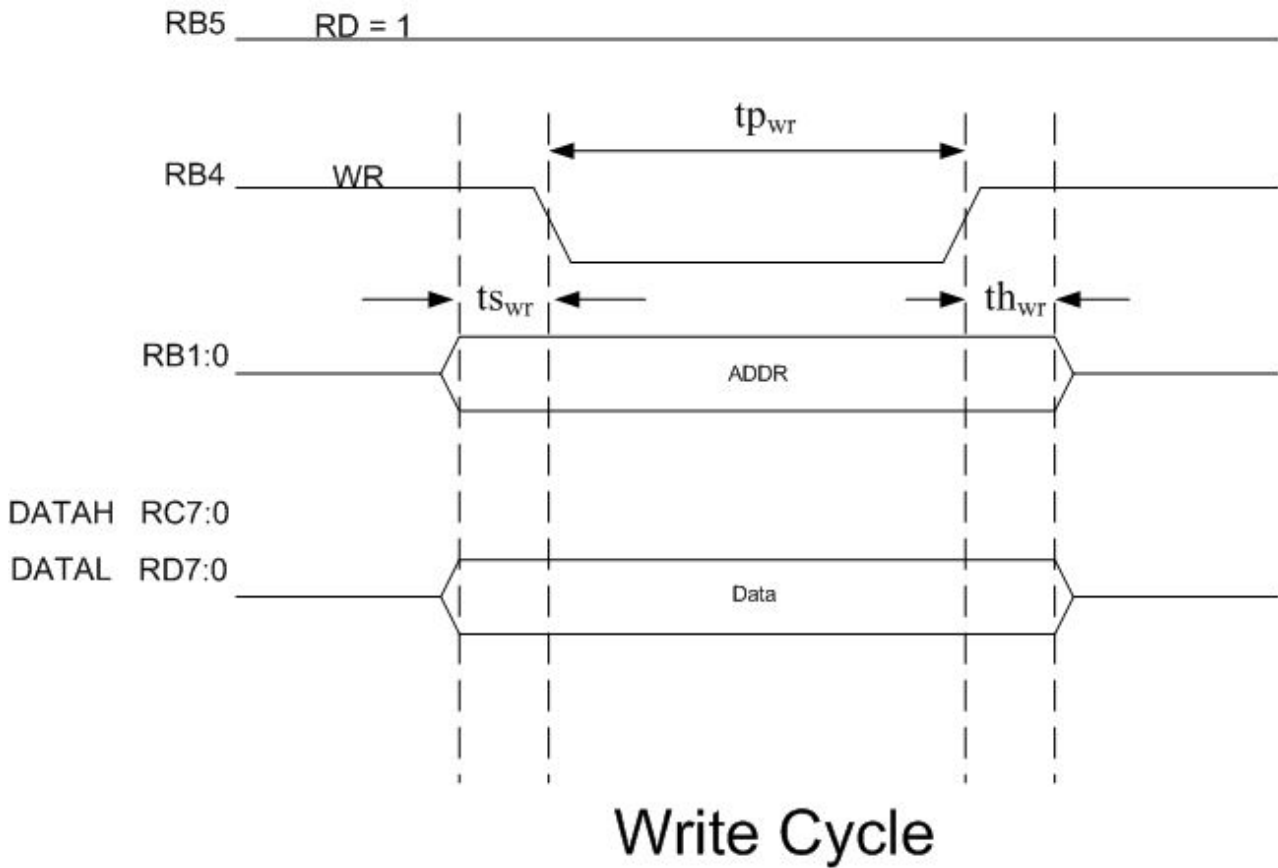


Figure 5-2 Write Timing Diagram

Table 5-2 Write Data Timing diagram

Symbol	Parameter	Min	Max	Unit
$tp_{wr}$	Period of Write Data	120	-	nS
$ts_{wr}$	Setup Time of Write Data	60	-	nS
$th_{wd}$	Hold Time of Write Data	80	-	nS

## 2 FifoRead

<b>Format</b>	<pre> BOOL FifoRead(     unsigned char addr,     const LPBYTE lpBuffer,     unsigned short dwCount );                     </pre>
<b>Function</b>	Reading from parallel I/O on Ethernet IO board. Size must not exceed 1024 bytes

Parameter	unsigned char addr	Address for parallel mode, can be 0 to 3
	const LPBYTE lpBuffer	Reading Buffer
	unsigned short dwCount	Size of data to read
Return value	(BOOL) TRUE	Reading completed
	(BOOL) FALSE	Reading failed

### 3 WRUserFlash

Format	BOOL WRUserFlash(unsigned short addr , const LPBYTE lpBuffer, unsigned short dwCount );	
Function	Writing data at Ethernet IO 's user area flash memory. Size of buffer must not exceed to 400 bytes.	
Parameter	unsigned short addr	Address of flash memory to write.
	const LPBYTE lpBuffer	Buffer that stored data to write
	unsigned short dwCount);	Size of data to write
Return value	(BOOL) TRUE	Flash memory writing complete.
	(BOOL) FALSE	Flash memory writing failed.

### 4 RDUserFlash

Format	BOOL RDUserFlash(unsigned short addr , const LPBYTE lpBuffer, unsigned short dwCount);	
Function	Read data from user area flash memory. Size of buffer must not exceed to 400 bytes.	
Parameter	unsigned short addr	Address of flash memory to read
	const LPBYTE lpBuffer	Reading Buffer
	unsigned short dwCount	Size of data to read
Return value	(BOOL) TRUE	Flash memory reading completed
	(BOOL) FALSE	Flash memory reading failed

## 5.4. Special Functions

### 1 SystemSoftReset

<b>Format</b>	BOOL SystemSoftReset(void);	
<b>Function</b>	Reset target Ethernet IO board. This function will cause the network processor on Ethernet IO to be "POWER ON RESET" state.	
<b>Parameter</b>	-	
<b>Return value</b>	(BOOL) TRUE	Ethernet IO board reset successfully
	(BOOL) FALSE	Ethernet IO board reset failed

### 2 SaveConfig

<b>Format</b>	BOOL SaveConfig ( unsigned char *DataBit, unsigned char *StopBit , unsigned char *ParityBit , BOOL *HwHandshak , DWORD *BaudRate , unsigned char *OldIP, unsigned char *SubnetIP, unsigned char *GatewayIP, unsigned char *MacAdd );	
<b>Function</b>	Save Ethernet IO configuration such as UART setting and network setting to flash memory in Ethernet IO. After call this function, it will reset EthernetIO board immediately. Please refer to example application for using this function	
<b>Parameter</b>	unsigned char DataBit	Number of data bit in serial protocol can be 7 or 8
	unsigned char StopBit	Number of stop bit in serial protocol can be 1 or 2
	unsigned char ParityBit	Number of parity bit in serial protocol can be 0 ( NO ) 1 ( ODD ) 2 ( EVEN )

	BOOL HwHandshak	TRUE For enable hardware handshake pin(CTS, RTS pin ) (unavailable in Ethernet IO board) FALSE Disable hardware handshake pin
	DWORD BaudRate	UART board rate in bit/seconds, can be set to 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200, 230400, 460800, 921600
	unsigned char *OldIP,	Old IP address of Ethernet IO board
	unsigned char *SubnetIP,	Subnet mask of Ethernet IO board
	Unsigned char *GatewayIP	Default Gateway IP address
	unsigned char *MacAdd	MAC address of Ethernet IO board
Return value	(BOOL) TRUE	Save configuration success
	(BOOL) FALSE	Fail to save configuration.

### 3 LoadConfig

Format	<pre> BOOL LoadConfig( unsigned char DataBit,                 unsigned char StopBit ,                 unsigned char ParityBit ,                 BOOL HwHandshak ,                 DWORD BaudRate ,                 unsigned char *OldIP,                 unsigned char *SubnetIP,                 unsigned char *GatewayIP,                 unsigned char *MacAdd ); </pre>	
Function	Load Ethernet IO configuration such as UART setting and network setting. Please refer to example application for using this function	
Parameter	unsigned char DataBit	Number of data bit in serial protocol can be 7 or 8
	unsigned char StopBit	Number of stop bit in serial protocol can be 1 or 2

	unsigned char ParityBit	Number of parity bit in serial protocol can be 0 ( NO ) 1 ( ODD ) 2 ( EVEN )
	BOOL HwHandshak	TRUE to enable hardware handshake pin(CTS, RTS pin ) ( no available in Ethernet IO board) FALSE to disable hardware handshake pin.
	DWORD BaudRate	UART Baud rate
	unsigned char *OldIP,	Old IP address of Ethernet IO board
	unsigned char *SubnetIP,	Subnet mask of Ethernet IO board
	unsigned char *GatewayIP	Default Gateway IP address
	unsigned char *MacAdd	MAC address of Ethernet IO board
Return value	(BOOL) TRUE	Load configuration success
	(BOOL) FALSE	Fail to load configuration

## 6. Reference

File Name	Version
<a href="http://www.design-gateway.com">http://www.design-gateway.com</a>	-
UbicomSystem.doc	-

## Note

## Note

## Note





54 BB Building, 13<sup>th</sup> Floor, Room No.1302, Sukhumvit 21 Rd. (Asoke)

Klongtoey-Nua, Wattana, Bangkok 10110 Thailand

Tel. (662)664-3069, Fax. (662)261-2290

[www.design-gateway.com](http://www.design-gateway.com)