# RAW-Connection User's Guide

## Introduction

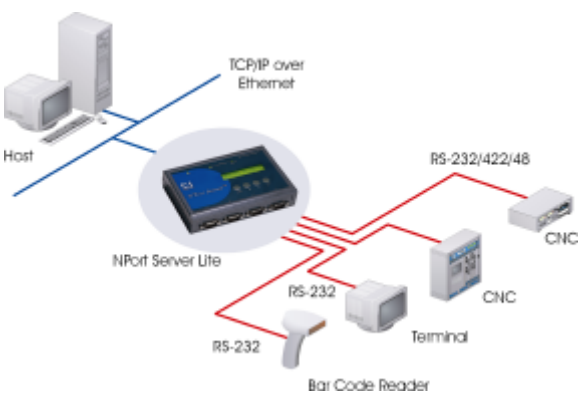
NPort Family products make an ideal Ethernet gateway for serial RS-232/422/485 data, and support raw data transmission, making it possible for serial data to travel over a LAN. When NPort server receives raw serial data, a TCP/IP header and trailer are added, and then the resulting packet is sent out over the Ethernet medium. Once the control host receives the TCP/IP frame, the NOS (Network Operating System) recovers the raw data by decoding the TCP/IP header and trailer. This allows the user to easily capture the raw serial data via the Ethernet, using either Telnet or a customized TCP/IP socket program, and provides an ideal solution for long-distance serial data transmission between host and serial device.

## The Client-Server Principle

In general, the "Client" is a program (e.g., Internet Explorer) which actively requests a specific service, with the service often located on a remote host. A "Server" on the other hand is a program that passively listens and responds to requests from Clients. It is often the case that Clients reside on an individual' s PC and Servers reside on larger and faster computers used specifically to run Server programs (in fact, this type of computer is itself often referred to as a Server).

The "raw connection" allows NPort Server to act as a passive server that listens for service requests from client hosts, with the request specified by a socket port (e.g., TCP port 4001 is the default for the NPort family). The host first connects to NPort Server via the TCP socket port, and then reads or writes the serial data after the connection has been established.

## The application Architecture



## Configuring NPort Server

**(1) NPort Server Lite (applies to firmware v. 2.00 or above for DE-301/302/304/331/332/334)**

1-1 Telnet NPort Server Lite' s IP address (default = 192.168.127.254) to access the Telnet console

✎ ***Note**: The LCD panel can be used to change the IP address & netmask to fit your network scheme. Refer to Appendix B of the User's Manual for more details.)*

1-2 Choose the VT-100 console terminal type by pressing 1.

1-3 Use the arrow key to move the cursor to "serverConfig" and then press Enter. *(Make sure "VT100 Arrows" is enabled. From the Telnet main menu, access Terminal→Preferences..)*

1-4 Move the cursor to Operating mode, press Enter, and then select "RAW Connection Mode".

1-5 Press ESC, move the cursor to Restart, and then press Enter to reboot the system.

**(2) NPort Server Pro (applies to firmware v. 1.10 or above for DE-303/308)**

2-1 Telnet NPort Server Pro' s IP address (default = 192.168.127.254) to access the Telnet console.

2-2 Choose the VT-100 console terminal type by pressing 1.

2-3 Move the cursor to "Server" and then press Enter to configure IP address, nestmask, DHCP, and gateway.

2-4 Press ESC, move the cursor to "OP_mode", press Enter, and then select RAW connection mode.

2-5 Press Enter to set the serial port(s) to RAW connection.

2-6 Move the cursor to "Pure raw data mode" and then press enter for more settings, including TCP port No., etc.

2-7 Press ESC, move the cursor to "Save" to save the configuration, and then move to Restart to reboot the system.

✎ ***Notes:***

*Destination IP: Keep blank to allow access for all hosts, or enter an IP address to allow access to only one host.*

*Inactivity time: (0-99 min.) the serial port will be reset if there is no transmission during this time.*

*TCP alive check time: (0-99 min.) the TCP connection will be reset if there is no activity for a length of time equal to tcp_time_out.*

## Introduction of socket interface

This interface provides easy-to-use commands for building up customized applications. When using the Socket Interface, connections are established with an IP address and port number. The IP address identifies a remote host and the port number identifies an application process (e.g., 80 for www browser service). The combination of <IP address : port number> allows the user to access the desired application process running on the remote host.

NPort Server will launch the bi-directional serial transmission service and then listen for requests from the Client via NPort server' s IP and TCP port 4001 (the manufacturer' s default value) after the user starts up the raw connection service. (The request is commonly made by application software built in or installed on the user interface PC or terminal–refer to the *Programming Example* given below for more details on how to use this operation mode.)

## Programming Example 1

The following example was developed under VC++ 4.2 for Windows 9x/NT/2000

```
////////////////////////////////////////////////////////////
//                                                        //
//  NPort RAW mode Tx/Rx example program                  //
//                                                        //
//  Date:      08-30-2001                                 //
//                                                        //
//  Version:   1.0                                        //
//                                                        //
//  Program description:                                  //
//     A dumb terminal emulation example program          //
//     via NPort RAW mode                                 //
//     1. hit <ESC> to stop the program                   //
//     2. program sends data from the keyboard to NPort   //
//     3. program prints to the screen any data           //
//        read from NPort                                 //
//     4. Syntax: as_raw NPort_IP TCP_PORT                //
//     5. Program developed under VC++ 4.2                //
//     6. May be used on Windows 9X/NT/2000 OS             //
//                                                        //
////////////////////////////////////////////////////////////

#include  <winsock2.h>
#include  <stdlib.h>
#include  <stdio.h>
#include  <fcntl.h>
#include  <string.h>
#include  <conio.h>

#define IP_ERROR    0xFFFEFFFFL // Invalid ip address
#define INTERVAL    100      // 10 msec
#define RETRY       50       // connect retry count

SOCKET              Fd;      // Socket descriptor used on data Tx/Rx

unsigned long     dot2ip(char *dot);
SOCKET            sioopen(unsigned long ipaddr,int p);

int  main(int argc, char *argv[])
{
     int       port,i;
     unsigned long  ip;
     WSADATA   wsaData;
     char      ch,len;
     char      buf[80];
     SOCKET         fd;

     if ( argc < 3 ) {
          printf("Syntax: %s NPortIP TCP_Port\n",argv[0]);
          return -1;
     }

     ip = dot2ip(argv[1]);
     if (ip == IP_ERROR) {
         printf("Invalid IP address %s!\n",argv[1]);
         return -2;
     }
     port = atoi(argv[2]);
```

```
//
// On windows we need to call WSAStartup before calling any SOCKET function
//
// If your project(VC++,VB,DELPHI) has include TCP/IP MODULE on it,
//    you do need not to call this function, because it is called automatically
//    when you select TCP/IP module.
//
     if (WSAStartup(0x202,&wsaData) == SOCKET_ERROR) {
         fprintf(stderr,"WSAStartup failed with error %d\n",WSAGetLastError());
        WSACleanup();
        return -5;
     }
//
// connect to remote
//
     printf("connecting to --> %s@%s....",argv[2],argv[1]);
     fd = sioopen(ip,port);
     if (fd != INVALID_SOCKET) {
        printf("ok\n");
         printf("<ESC> = stop program.\n");
         printf("Any key = send to remote.\n");
         printf("Dumb terminal begin ...\n");
        printf("\n");
         sprintf(buf,"Welcome to MOXA NPort RAW mode example prog.\r\n");
         send(fd,buf,strlen(buf),0);     // Send welcome string to remote
         for (;;) {
          if (kbhit()) {          // keyboard is hit
              ch = getch();
              if (ch == 27) {     // user hit <ESC> --> exit
               printf("\n");
               break;
              }
              send(fd,&ch,1,0);        // Send data to NPort
              if (ch == '\n') {
               send(fd,"\r",1,0);  // send LF as CR-LF
              }
              if (ch == '\r') {
               send(fd,"\n",1,0);  // send CR as CR-LF
              }
          }
          len=recv(fd, buf, sizeof(buf), 0);

          if (len <= 0)  // No data read
              Sleep(10); // Prevent from wasting too much of CPU time
          else {
              for (i=0;i<len;i++)
               printf("%c",buf[i]);
          }
        }
        closesocket(fd);// Close TCP connection
     } else {
        printf("fail!\n");
     }
//
// On windows we need to call WSACleanup to free SOCKET resource
//   before exiting the program
//
     WSACleanup();
     printf("hit any key to stop program...\n");
     getch();
     printf("program exit.\n");
```

```
     return 0;
}
//
//Convert dot notation to IP address
// ie: From "192.168.2.1" to 0x0102A8C0
//
unsigned long dot2ip(char *dot)
{
     unsigned long  ip;
     unsigned char  *c;
     int       i, d;

     c = (unsigned char *)&ip;
     for (i = 4; i-- > 0; ) {
          d = *dot++ - '0';
          if (d < 0 || d > 9)
               return IP_ERROR;
          while (*dot >= '0' && *dot <= '9') {
               d = d * 10 + *dot++ - '0';
               if (d > 255)
                    return IP_ERROR;
          }
          *c++ = d;
          if (*dot++ != '.')
               break;
     }
     if (*--dot || i)
          return IP_ERROR;
     return ip;
}
//
//Connect to remote TCP port
//
SOCKET sioopen(unsigned long ipaddr,int port)
{
     struct sockaddr_in  des;
     int            i,j,len;
     SOCKET              fd;
     BOOL           b = TRUE;
     ULONG               mode = 1;   /* set to non_delay mode */
     unsigned short      p;


     p = htons((unsigned short)port);
//
// open socket
//
     fd = socket(AF_INET, SOCK_STREAM, 0);
     if ( fd == INVALID_SOCKET ) {
          return(fd);
     }
//
// Set SOCKET to No Delay mode
//
     if (ioctlsocket(fd,FIONBIO,&mode)) {
        closesocket(fd);
        return(INVALID_SOCKET);
     }
//
// Set remote IP address and port no
//
```

```
     des.sin_family = AF_INET;
     des.sin_addr.s_addr = ipaddr;
     des.sin_port = p;
     len = sizeof(struct sockaddr_in);
//
// connect to remote
//
     i = 0;
     for (;;) {
          j = connect(fd,(struct sockaddr *)&des, len);
          if (j == 0)                  // connected
             break;
          if (WSAGetLastError() == WSAEISCONN) { // already connected
               j = 0;
               break;
          }
          if (i++ >= RETRY)   // Connected failed too many times --> give up
             break;

          Sleep(INTERVAL);// Sleep for a while before trying it again.
                    // Prevent from wasting too much of CPU time.
     }
     if( j != 0 ) {     // Can't connect to remote
        closesocket(fd);
        return(INVALID_SOCKET);
     }
     return(fd);
}
```

## Programming Example 2

The following program was developed under VB6.0 with serial settings 38400, n, 8, 1

---

```
Private Sub cmdConnect_Click()

    If txtIP.Text = "" Or txtPort.Text = "" Then Exit Sub

    Winsock1.Connect txtIP.Text, txtPort.Text

    txtStatus.Text = txtStatus.Text & "Connecting..." & vbCrLf

    Timer1.Enabled = True

End Sub


Private Sub cmdDisconnect_Click()

   Winsock1.Close

    txtStatus.Text = txtStatus.Text & "Connection Close." & vbCrLf

End Sub


Private Sub cmdClose_Click()
```

```
    If MsgBox("Are you sure to shutdown the Remote Server application?", vbQuestion
+ vbYesNo, "Shutdown") = vbNo Then Exit Sub
    SendData ("Close:")
    Winsock1.Close
End Sub

Private Sub cmdSendKey_Click()
    Dim strMsg As String
    strMsg = InputBox("Please enter any letters to send back to the Server.", "Send
Key", "")
    If strMsg <> "" Then
        If Not SendData("Keyboard:" & strMsg) Then
           Winsock1.Close
        End If
    End If
End Sub

Private Sub Timer1_Timer()
    MsgBox "Client could not find server.", vbCritical
    If Winsock1.State <> sckClosed Then
       Winsock1.Close
    End If
    Timer1.Enabled = False
    txtStatus.Text = txtStatus.Text & "Connection Fail." & vbCrLf
End Sub

Private Sub Winsock1_Connect()
    Timer1.Enabled = False
    txtStatus.Text = txtStatus.Text & "Connection Established." & vbCrLf
End Sub
```

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
    Dim strData As String
    Winsock1.GetData strData
    txtStatus.Text = txtStatus.Text & "Get data: " & strData & vbCrLf & vbCrLf
End Sub

Private Function SendData(sData As String) As Boolean
    On Error GoTo ErrorHandler
    Dim lngTime As Long
    blnReply = False
    Winsock1.SendData sData
    Do Until (Winsock1.State = 0) Or (lngTime < 10000)
       DoEvents
        lngTime = lngTime + 1
        If lngTime > 10000 Then Exit Do
    Loop
    SendData = True
    Exit Function

ErrorHandler:
    SendData = False
    MsgBox Err.Description, vbCritical
    Exit Function
End Function
```

---

## Reference Information

You may access Moxa's website at www.moxa.com for firmware downloads and upgrades.