Telnet Server Application note for W5200

Version 1.0

Table of Contents

1 Introduction

This document will explain about Telnet, and implement the Telnet server by using W5200E01-M3. There are no extra file systems for Telnet, but LED 3 and 4, which are connected to the GPIO port of W5200E01-M3 can be controlled by using telnet. Section 2 of will go over Telnet, section 3 will demonstrate the functions of Telnet, and section 4 will cover the code analysis. All example codes in this document are based on IAR compiler.

2 Telnet

Telnet (teletype network) is a network protocol used in internet or local area networks to provide a bidirectional interactive communications facility. Telnet was developed in 1969 beginning with RFC15, and then later was standardized as internet standard. Most network equipment with TCP/IP protocol stack basically support Telnet service. Telnet is a widely used client/server application program.

There has been a tendency in decrease of importance of Telnet since more browsers with convenient functions have been introduced. But telnet is still an important tool for many PCs with multi-user accounts in order for remote connection; users can connect to the telnet server from home, work, or anywhere else with internet connection.

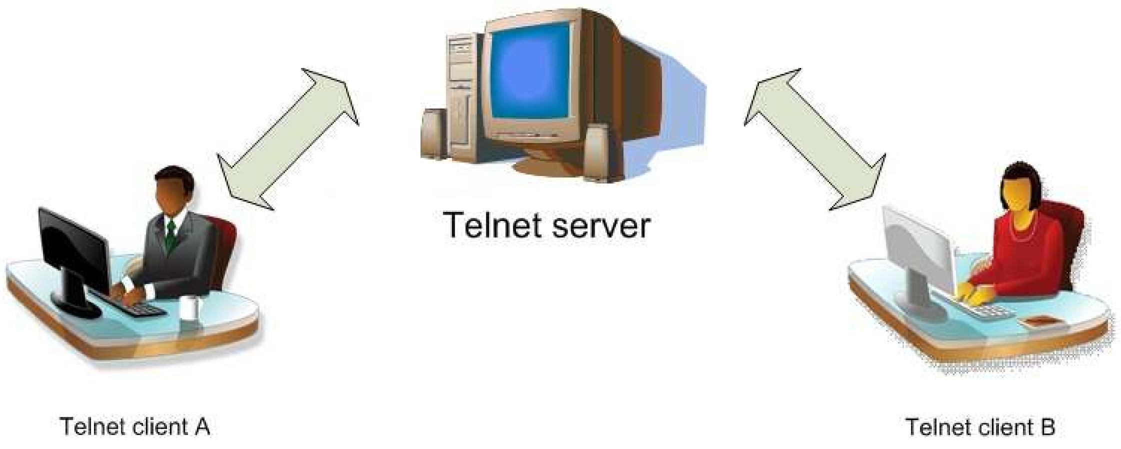


Figure 2.1 Telnet system

2.1 NVT (Network Virtual Terminal)

In general, the process for users to remotely connect to Telnet server is quite complicated because computers and OS(Operating System) accept combination of special characters as a token; this combination of special characters are different depending on the OS. Telnet defines an interface called NVT(Network Virtual Terminal) to solve this problem. By using this interface, telnet client translates the combination of characters, which are entered in NVT format from the local terminal, and sends it to the network. The Telnet server then translates the received NVT format characters into a format which a remote computer can accept and read. This process is shown in Figure 2.2 below.

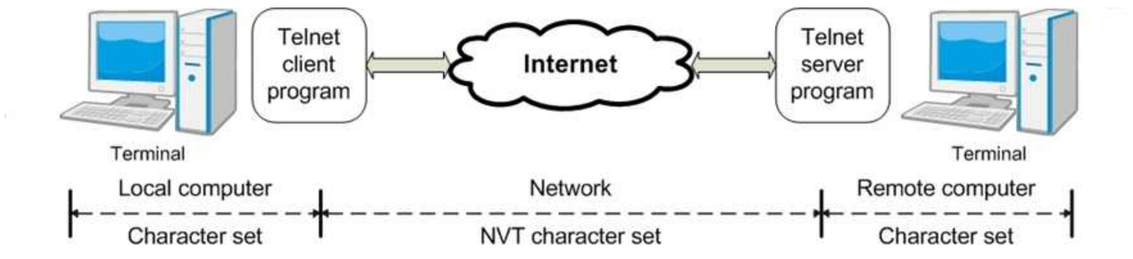


Figure 2.2 NVT of telnet system

NVT uses two kinds of character combinations; one for data-use and another for remote control-use.
NVT for data-use usually uses NVT ASCII. NVT ASCIII is an 8bit character combination, where the lower 7bits are the same as US ASCII and the highest bit is 0. The highest bit can be 1 or 0; in case it is 1, the option negotiation between the telnet client and server must be defined in advance.

NVT for remote control-use uses an 8bit control character which has its highest bit set to 1. Send the IAC (Interrupt As Command; has value of 0xFF) using TCP before sending the remote control-characters that are used for entering special keys, setting the connection, and changing the status. Therefore bytes that are received after the IAC character must be recognized as remote control-characters.

Table 2.1 NVT control character

Control character	Code	Meaning
EOF	236	End of file.
EOR	239	End of record.
SE	240	End of sub-option.
NOP	241	No operation.
Data Mark	242	The data stream portion of a Synch. This should always be accompanied by a TCP Urgent notification.
Break	243	NVT character BRK.

Interrupt Process	244	Suspend, interrupt, abort, or terminate the process to which the NVT is connected. Also, part of the out-of-band signal for other protocols which use TELNET.
Abort output	245	Allow the current process to (appear to) run to completion, but do not send its output to the user. Also, sends a Synch to the user.
Are You There	246	Send back to the NVT some visible (i.e., printable) evidence that the AYT was received.
Erase character	247	The recipient should delete the last preceding undeleted character or "print position" from the data stream.
Erase Line	248	The recipient should delete characters from the data stream except the "CR LF" sequence.
Go ahead	249	The GA signal.
SB	250	Indicates that what follows is the sub-negotiation of the indicated option.
WILL	251	Indicates the desire to begin performing, or confirmation that user is currently performing the indicated option.
WONT	252	Indicates the refusal to perform, or continue performing the indicated option.
DO	253	Indicates the request that the other party perform, or confirmation that you are expecting the other party to perform, the indicated option.
DON'T	254	Indicates the demand that the other party stop performing, or confirmation that you are no longer expecting the other party to perform, the indicated option.
IAC	255	Interpret as command

2.2 Options for Telnet Terminal

As mentioned earlier, options between the client and server can be negotiated before or after using the Telnet service. The table below shows some commonly used options. Telnet options were defined as part of the first version of telnet, and additions are specified in many RFC documents.
Please use the following link from IANA website to search updated telnet options: http://www.iana.org/assignments/telnet-options

Table. 2.2 Options for telnet

Option ID	Name	RFC
1	Echo	857
3	Suppress go ahead	858
5	Status	859
6	Timing mark	860
24	Terminal type	1091
31	Window size	1073
32	Terminal speed	1079
33	Remote flow control	1372
34	Line mode	1184
36	Environment variables	1408

The negotiation of options between the client and server is necessary in order to use various options of telnet. As shown in Table 2.1, 4 control characters (WILL, WONT, DO, and DON'T) are used for negotiation of options.

In order to activate options, the transmitter sends the WILL command, asking "may I activate this option?" Then, the receiver will send the DO command, meaning acceptance, or the DON'T command, meaning refusal. Another way to activate options is to send the DO command, meaning "activate this option," and the receiver will send back the WILL command or WONT command.

The process of deactivating options is as followed. The transmitter sends the WONT command, meaning "I won't use this option anymore." Then, the receiver will send back the DO command to accept or DONT command to reject.

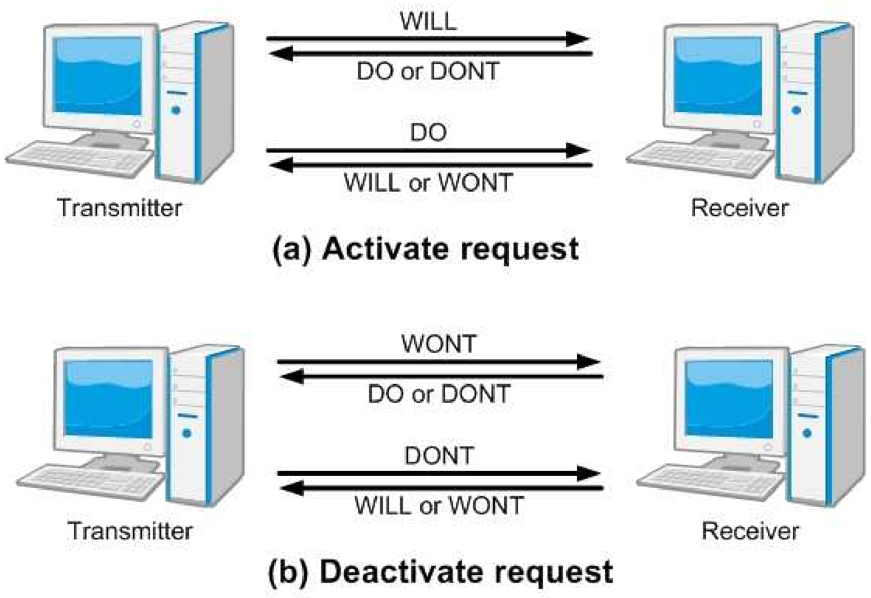


Figure 2.3 Negotiation of telnet options

2.3 Telnet Operation Mode

Most operations of Telnet are processed in 3 modes: General mode, Character mode, and Line mode.

-**General mode**: This mode is the basic mode when character or line mode is not selected from the option negotiation. In this mode, the client echoes all input characters and does not send it until a line is completed.

After all lines are sent to the server, the client waits for the GA-command until a new line is accepted. This mode can be ineffective when TCP connection is Full-duplex because the general Telnet operates in Half-duplex.

-**Character mode**: In this mode, each and every character is independently sent to the server when the client enters each character. The server usually allows the echo-character to appear on the client screen. In this mode, some delay can occur when an echo-character's transmission is slow; there can bea possibility of a minor slowdown of the network.

-**Line mode**: Line mode is the suggested mode to cover the flaws of general mode and character mode. In this mode, line editing (echo, delete character, delete line, etc) by client is possible, and they are sent to the server after editing. Line mode operates in full-duplex mode, and lines can be sent with no regards to the GA- command.

3 Telnet Demonstration

Section 3 will demonstrate Telnet. The W5200EVB_App.bin code that is used for demonstration is implemented to control LED3, LED4, which are connected to the GPIO port. The W5200E01-M3 will be the Telnet server and the user's PC will be the Telnet client. Telnet Client is included in Windows; when 'telnet' is entered in the command window (or command prompt), Microsoft Telnet Client will operate. The serial message from the serial terminal program tells the status of W5200E01-M3; user will use this message to check on the connection with the remote area.

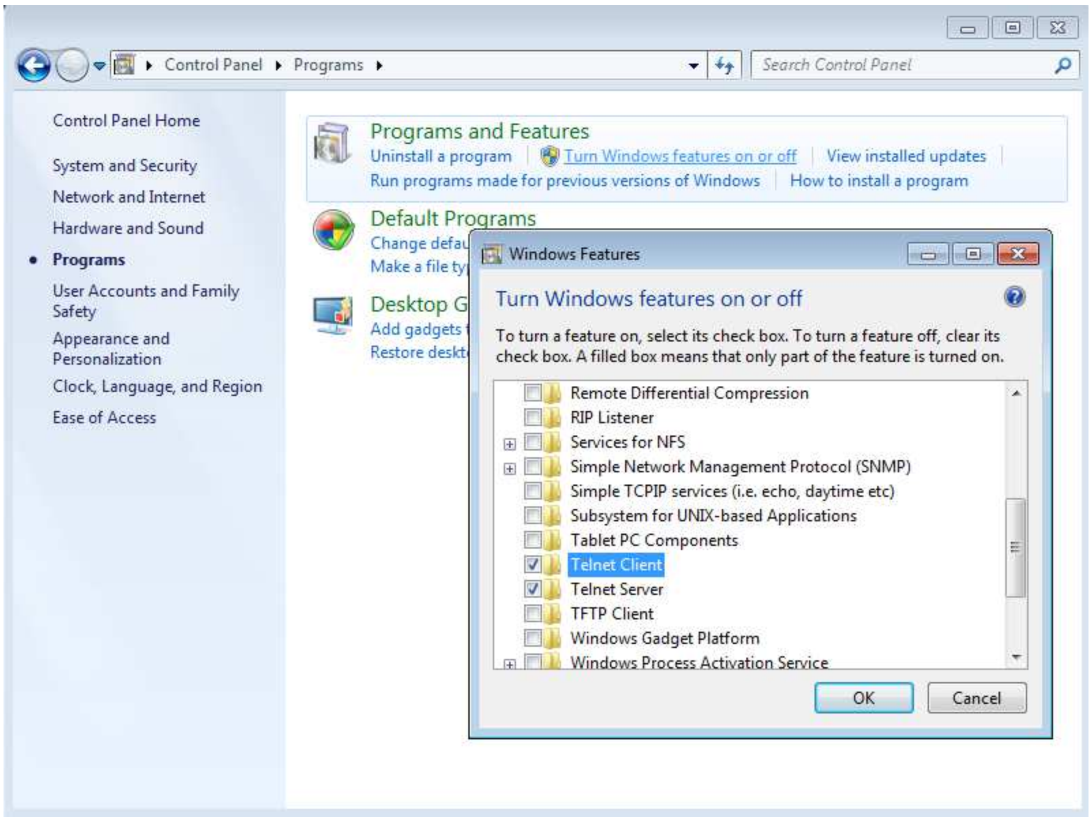
Since the use of Serial terminal program and Telnet Client depends on the user's OS, check the Windows version according to Table 3.1 & 3.2. If the user uses Windows Vista or 7, additional program settings are required.

Table 3.1 Windows setting for serial terminal

Serial terminal program	
Windows XP & older version	The Hyper terminal is included as Windows basic application program. Use the Hyper terminal to check serial message.
Windows Vista / Windows 7 version	The Hyper terminal program is not included as basic application program. Other serial terminal program must be used. TeraTerm can be used as open serial terminal program, and the port settings are identical as Hyper terminal. TeraTerm is an open serial termianl program with BSD license. TeraTerm can be downloaded at http://ttssh2.sourceforge.jp/

Table 3.2 Windows setting for Telnet client

Telnet Client	
Windows XP & older version	Telnet Client is enabled as Windows features and can use right away.
Windows Vista / Windows 7 version	Telnet client is disabled in Windows features list. Click Control Panel and go to [Programs and Features – Turn Windows Features on or off] and enable Telnet Client.



Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ☐ Remote Differential Compression
- ☐ RIP Listener
- ☐ Services for NFS
- ☐ Simple Network Management Protocol (SNMP)
- ☐ Simple TCPIP services (i.e. echo, daytime etc)
- ☐ Subsystem for UNIX-based Applications
- ☐ Tablet PC Components
- ☑ Telnet Client
- ☑ Telnet Server
- ☐ TFTP Client
- ☐ Windows Gadget Platform
- ☐ Windows Process Activation Service

Figure 3.1 Enable and Install Telnet Client

This document will use hyper terminal, which is the most used serial terminal program, to check the serial message of W5200E01-M3, which is the Telnet server in this case.

3.1 Setting for Running Telnet

Connect USB mini cable and LAN cable to W5200E01-M3 for Telnet demonstration. Use ‘Flash Loader Demonstrator’ from STMicroelectronics to load the telnet server binary image on W5200E01-M3 board. Please refer to the ‘W5200E01-M3 User’s Guide’ for more details on how to use W5200E01-M3 and Flash Loader Demonstrator.

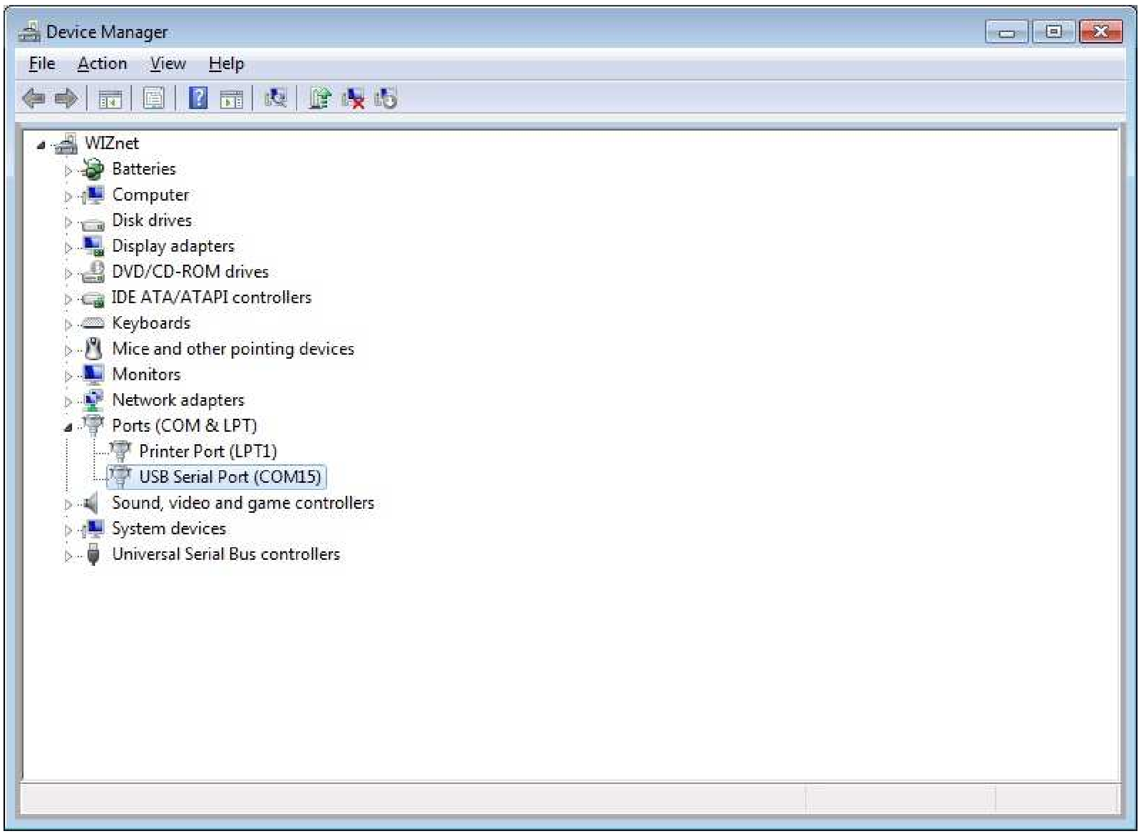
The process of loading image to W5200E01-M3 is as followed.

1. Connect the USB mini cable and LAN cable to W5200E01-M3.
2. Modify the provided source code from the IAR Compiler according to the user’s network.
3. Compile the modified source code and create an application image.
4. Select PROG, and use the Flash Loader Demonstrator to download the created image to the board.
5. After the download is complete, change the PROG S/W to RUN. Set the Serial terminal port and check the application.

3.1.1 Physical Connection

Once the user's PC and W5200E01-M3 is connected through USB mini cable, the COM port is assigned to W5200E01-M3. Check the assigned COM port in order to use Flash Loader Demonstrator to download image on the board. The document uses COM 15 as COM port, but this can change according to the user's setting.

User can check the COM port at [Control Panel – Device Manager].



- WIZnet
  - Batteries
  - Computer
  - Disk drives
  - Display adapters
  - DVD/CD-ROM drives
  - IDE ATA/ATAPI controllers
  - Keyboards
  - Mice and other pointing devices
  - Monitors
  - Network adapters
  - Ports (COM & LPT)
    - Printer Port (LPT1)
    - USB Serial Port (COM15)
  - Sound, video and game controllers
  - System devices
  - Universal Serial Bus controllers

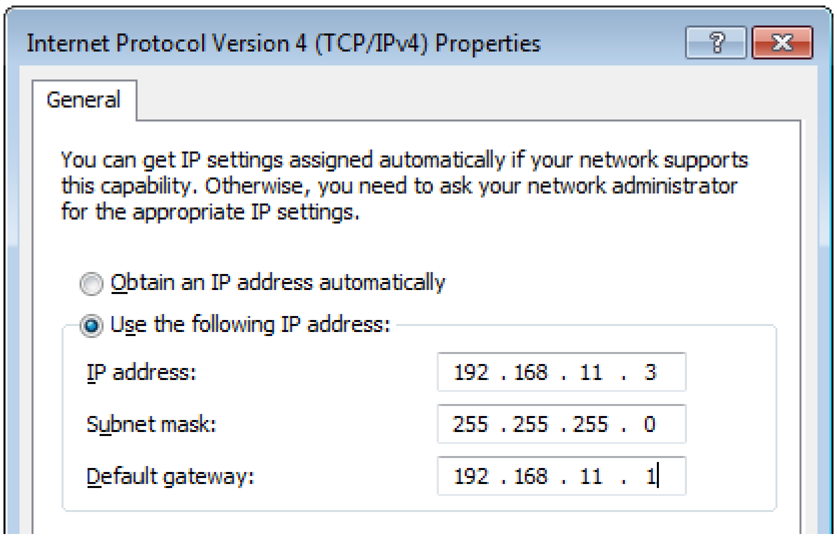
Figure 3.2 Check the COM port in Windows device manager

3.1.2 Network Configuration

Once the COM port is checked, prepare the binary image to program the board. The Telnet source code is implemented using the IAR Embedded Workbench IDE and is provided at WIZnet's homepage / [SUPPORT – Download]. The network information of the board is included in the main.c file of the source code, and the network information must be modified according to the user's settings.

```
/* main.c */
…
uint8 MAC[6] = {0x00, 0x08, 0xDC, 0x01, 0x02, 0x03};   // MAC Address
uint8 IP[4] = {192, 168, 11, 4};   // IP Address
uint8 GateWay[4] = {192, 168, 11, 1};   // Gateway Address
uint8 SubNet[4] = {255, 255, 255, 0};   // SubnetMask Address
…
```

Test MAC address and IP address have been used in this document for demonstration purpose. The IP address of the Test PC was set to 192.168.11.3; the IP address of the board and gateway must be set according to the Test PC. User can change IP address of PC: [Local Area Connection – Properties – Internet Protocol – Properties]



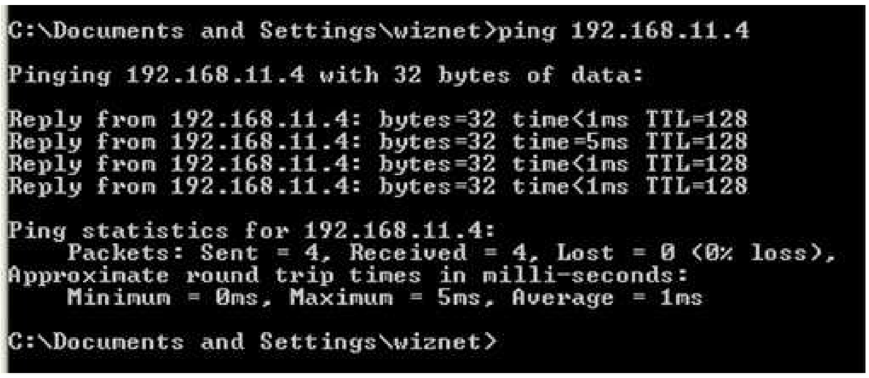
You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically
[✓ Use the following IP address:]

IP address: 192 . 168 . 11 . 3
Subnet mask: 255 . 255 . 255 . 0
Default gateway: 192 . 168 . 11 . 1

Figure 3.3 Internet Protocol Properties

The test MAC address is 00:08:DC:01:02:03 and test IP address is 192.168.11.4 for W5200e01-M3. When setting the network, enter the identical Gateway as the Test PC's; both user's PC and W5200E01-M3 must exist on the same sub-network in order for the Telnet server to remotely connect to W5200E01-M3. After setting the network, use ping test to check the communication between W5200E01-M3 and the network. Open the command window and enter [ping 192.168.11.4], and if W5200e01-M3 is connected, the screen will appear as shown in Figure 3.4.



```
C:\Documents and Settings\wiznet>ping 192.168.11.4

Pinging 192.168.11.4 with 32 bytes of data:

Reply from 192.168.11.4: bytes=32 time<1ms TTL=128
Reply from 192.168.11.4: bytes=32 time=5ms TTL=128
Reply from 192.168.11.4: bytes=32 time<1ms TTL=128
Reply from 192.168.11.4: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.11.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 5ms, Average = 1ms

C:\Documents and Settings\wiznet>
```

Figure 3.4 Ping Test at Command Prompt

If the ping test is successful, network settings are completed.

3.1.3 Compile

Once the network setting is completed, the source code must be compiled and linked in order to create the Telnet firmware image for programming W5200E01-M3. Use [Project – Make] from the menu of IAR Embedded Workbench IDE or press the [F7] key to operate the 'Make' process, and then the W5200EVB_App.bin file will be created in the project directory's [\Debug\Exe].

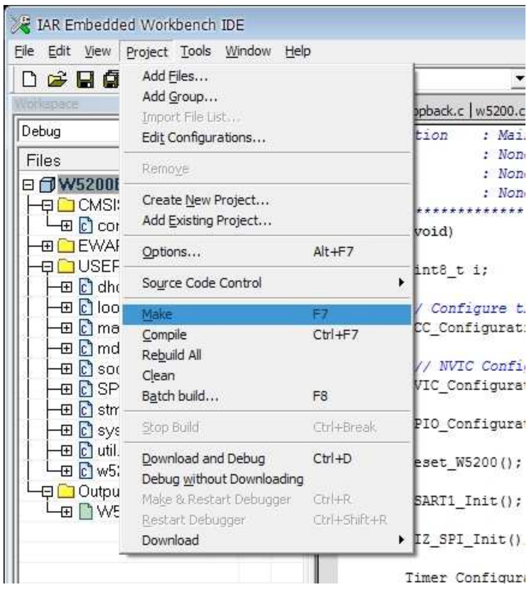


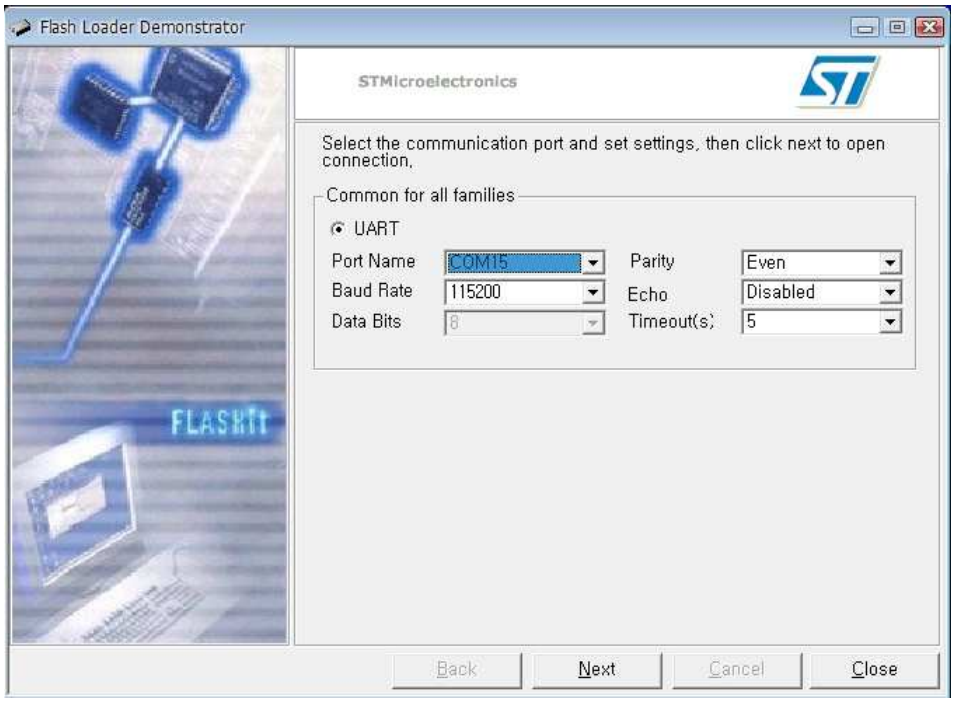
Figure 3.5 Compile on IAR Embedded Workbench IDE

3.1.4 Download

The process of downloading the created W5200EVB_App.bin file to the W5200E01-M3 board is as followed.

1. Create W5200EVB_App.bin in IAR complier
2. Select PROG for PROG S/W of W5200E01-M3 and reset the board
3. Run the Flash Loader Demonstrator and set serial port
4. Select Target device (STM32_Med-density_64K)
5. Select 'Download to device' and set the image file path

   (image file path : [\Work\App\Debug\Exe] / located in project directory)
6. Once download is completed, change the PROG S/W to RUN and reset the board



Select the communication port and set settings, then click next to open connection.

Common for all families

UART

Port Name	COM15	Parity	Even
Baud Rate	115200	Echo	Disabled
Data Bits	8	Timeout(s)	5

Figure 3.6 Flash Loader Demonstrator setting

Select the COM port that is connected with W5200E01-M3 for Port Name. The settings must be done according to user's PC's serial communication settings.

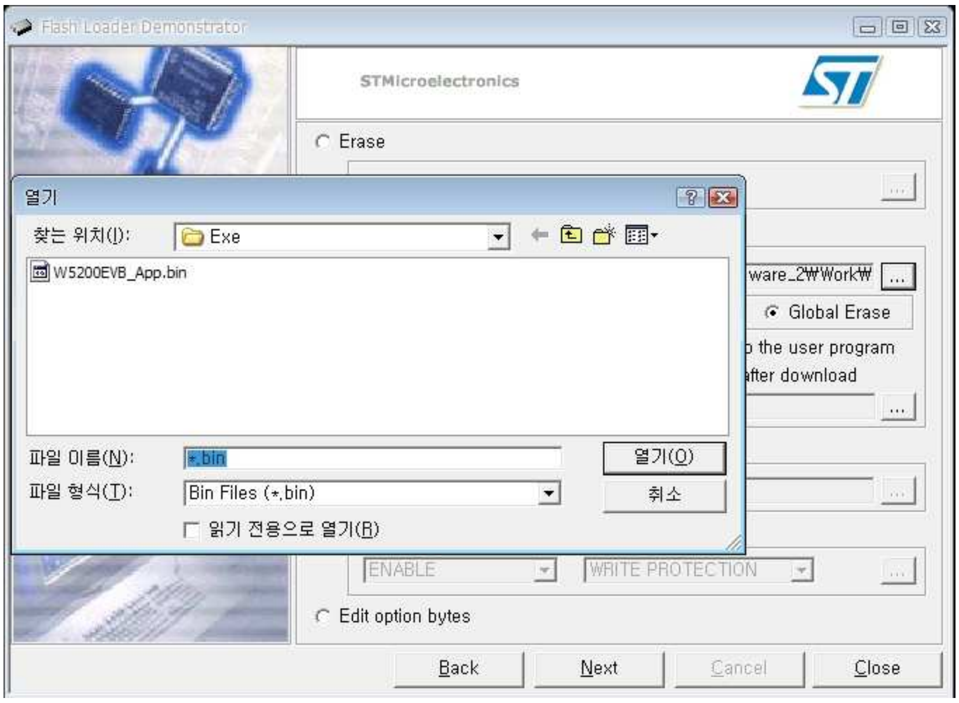


Figure 3.7 Select Binary image for Flash Loader Demonstrator

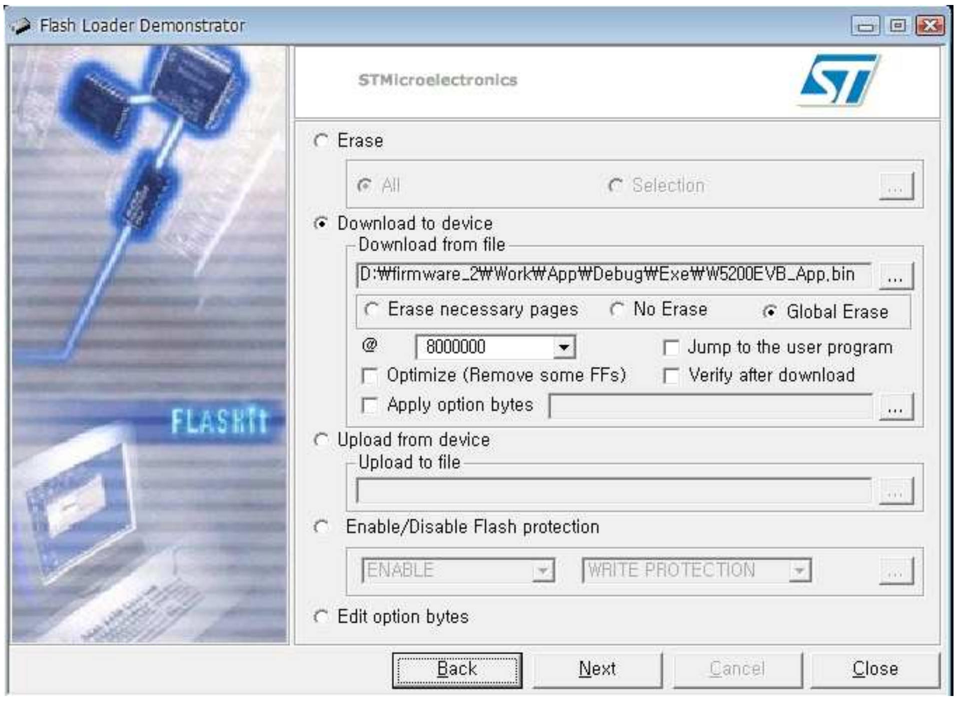
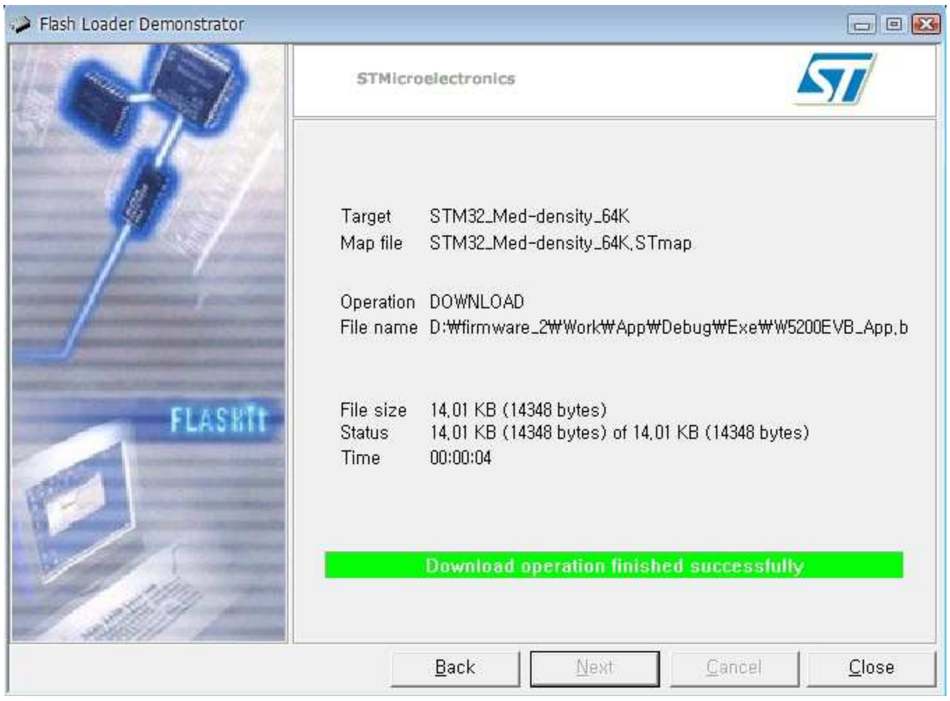


Fig 3.8 Download to device

Start the download from @ 0x08000000 of the memory address.

User may use a different starting address for download if needed.



Target STM32_Med-density_64K
Map file STM32_Med-density_64K.STmap

Operation DOWNLOAD
File name D:₩firmware_2₩Work₩App₩Debug₩Exe₩W5200EVB_App.b

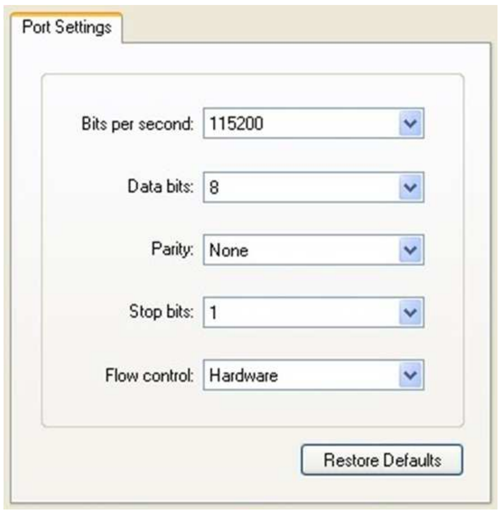
File size 14.01 KB (14348 bytes)
Status 14.01 KB (14348 bytes) of 14.01 KB (14348 bytes)
Time 00:00:04

Download operation finished successfully

Figure 3.9 Download Completetion

3.1.5 Setting for Serial Terminal

Once the image is successfully loaded to W5200E01-M3, set the serial terminal to test the board. As mentioned earlier, Hyper Terminal, which is the most used serial terminal, will be used to check the board's serial message. Run the Hyper terminal to set the Port information for communication with the board. Click the [Properties] in File menu and select COM port. And Press the [Configure] button. the screen will appear as shown in Figure 3.10.



Bits per second: 115200
Data bits: 8
Parity: None
Stop bits: 1
Flow control: Hardware

Figure 3.10 Hyper terminal setting

Table 3.3 Hyper terminal setting

Port Settings	Value
Bits/sec(Baud rate)	115200
Data bits	8
Parity	None
Stop bits	1
Flow control	Hardware

3.2 Demonstration

The implemented Telnet server test code operates as shown in Figure 3.11. Wait in LISTEN status for the Client's connection. If Client attempts to connect, wait for the user to enter after the ESTABLISHED process and telnet option negotiation. Once the user logs in to telnet, various commands that were implemented in the test code can be used. The initial ID is 'wiznet' and password is '0000.' The commands for controlling the board's LED are included in the source code. The commands, ID, and Password can be modified by the user.

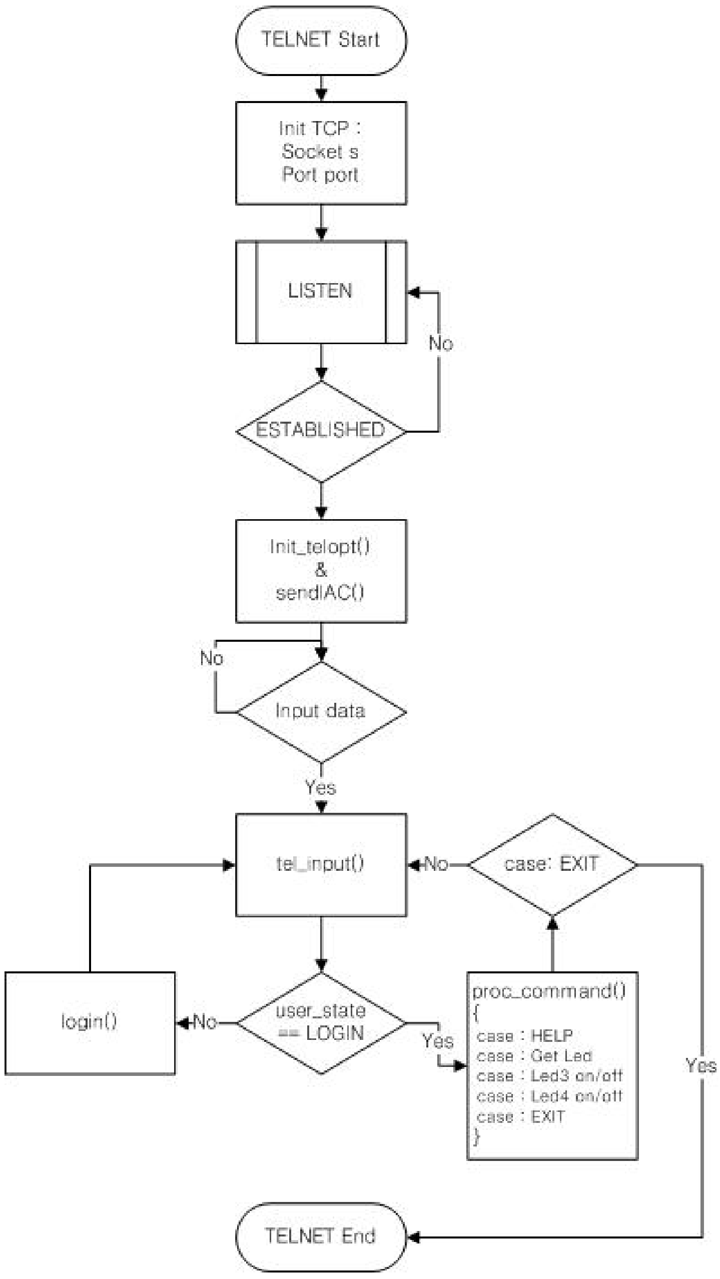
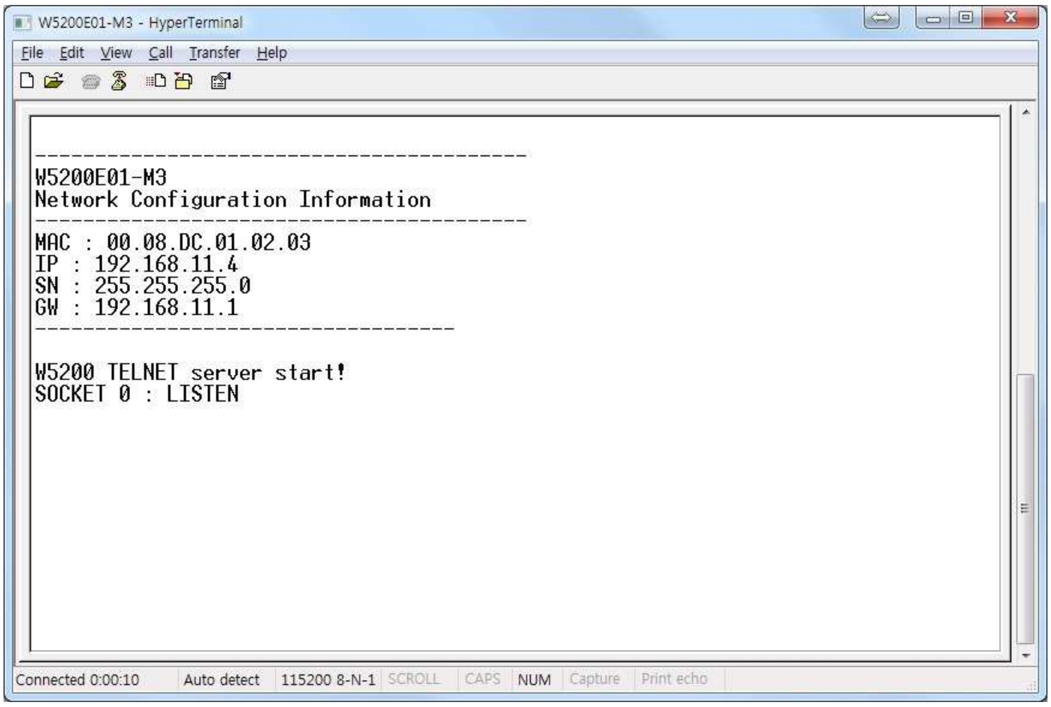

TELNET Start
Init TCP : Socket s Port port
LISTEN
No
ESTABLISHED
Init_telopt() & sendIAC()
No
Input data
Yes
tel_input()
No
case: EXIT
login()
No
user_state == LOGIN
Yes
proc_command() { case : HELP case : Get Led case : Led3 on/off case : Led4 on/off case : EXIT }
Yes
TELNET End

Figure 3.11 Telnet test code Flow chart

Set the Hyper terminal according to Table 3.3, click connect/call, and the message as shown in figure 3.12 will appear.

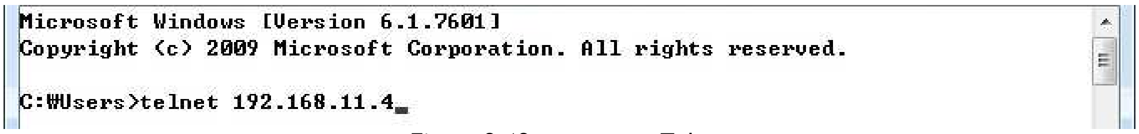

W5200E01-M3 - HyperTerminal
File Edit View Call Transfer Help
Connected 0:00:10 Auto detect 115200 8-N-1 SCROLL CAPS NUM Capture Print echo

```
-----------------------------------------
W5200E01-M3
Network Configuration Information
-----------------------------------------
MAC : 00.08.DC.01.02.03
IP  : 192.168.11.4
SN  : 255.255.255.0
GW  : 192.168.11.1
-----------------------------------

W5200 TELNET server start!
SOCKET 0 : LISTEN
```

Figure 3.12 Hyper terminal messages

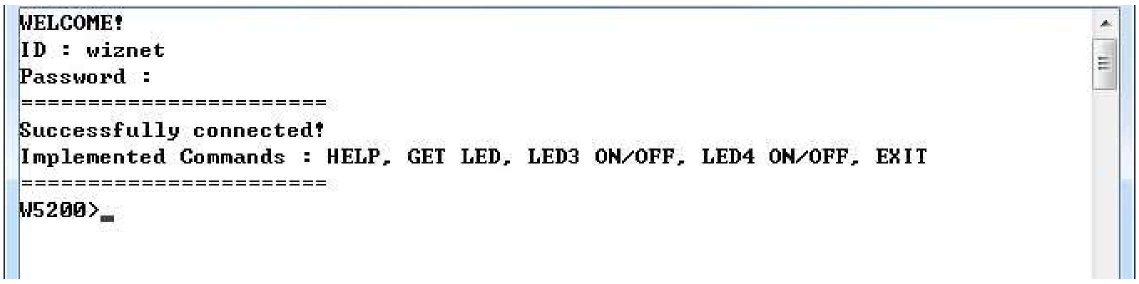
Use the Network Configuration Information of the hyper terminal to check the Telnet server's IP. Run the command window and enter 'telnet 192.168.11.4.' All information including the Telnet server's IP should be set according to user's network settings.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:WUsers>telnet 192.168.11.4_
```

Figure 3.13 connect to Telnet



```
WELCOME!
ID : wiznet
Password :
========================
Successfully connected!
Implemented Commands : HELP, GET LED, LED3 ON/OFF, LED4 ON/OFF, EXIT
========================
W5200>_
```

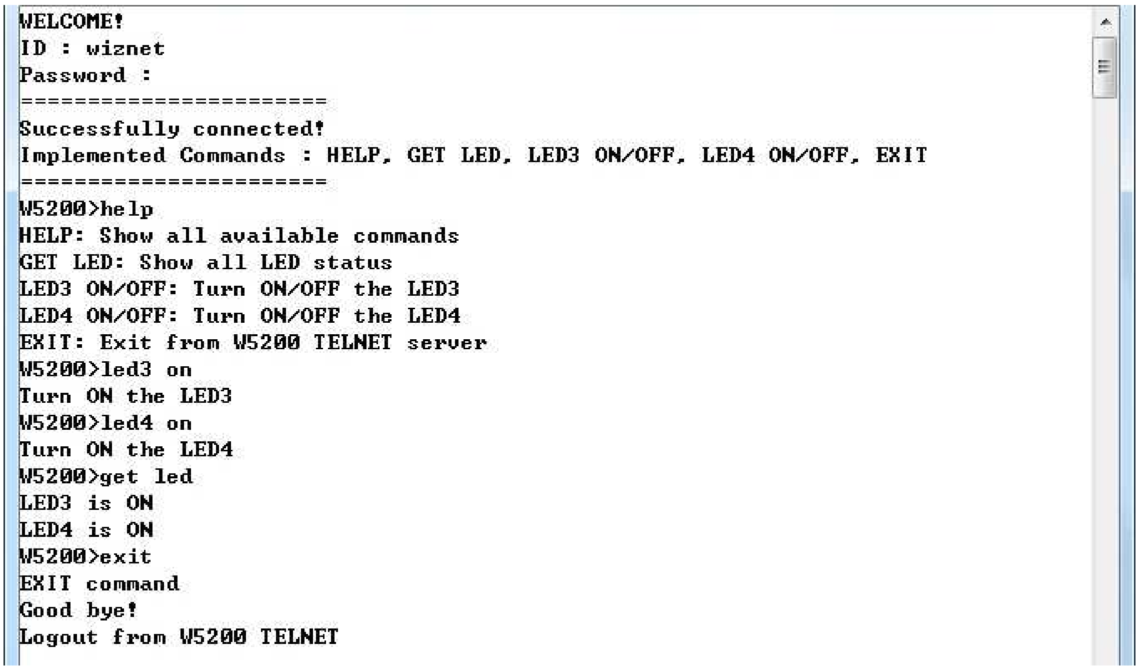
Figure 3.14 Login and connect success

Enter commands on the Telnet server of W5200E01-M3 to check its operations.

The implemented commands for the test code are as shown in Table 3.4.

Table 3.4 Commands for telnet

Command	Description
HELP	Show all available commands
GET LED	Show all LED's status
LED3 ON	Turn on the LED3
LED4 ON	Turn on the LED4
LED3 OFF	Turn off the LED3
LED4 OFF	Turn off the LED4
EXIT	Exit from telnet server



```
WELCOME!
ID : wiznet
Password :
========================
Successfully connected!
Implemented Commands : HELP, GET LED, LED3 ON/OFF, LED4 ON/OFF, EXIT
========================
W5200>help
HELP: Show all available commands
GET LED: Show all LED status
LED3 ON/OFF: Turn ON/OFF the LED3
LED4 ON/OFF: Turn ON/OFF the LED4
EXIT: Exit from W5200 TELNET server
W5200>led3 on
Turn ON the LED3
W5200>led4 on
Turn ON the LED4
W5200>get led
LED3 is ON
LED4 is ON
W5200>exit
EXIT command
Good bye!
Logout from W5200 TELNET
```

Figure 3.15 Telnet terminal running message

All available commands for HELP command can be checked. The 'LEDx ON/OFF' command is used to turn on/off the LED, and the 'GET LED' command is used to remotely check the status of LED. Figure 3.15 shows the screen for connecting and ending Telnet connection with LED3 and LED4 turned on. The status of LED can be checked directly from W5200E01-M3.

4 Implementation Code

This section will go over the Telnet server example codes loaded on W5200E01-M3. Please refer to document 'How to implement TCP for W7100' for details on TCP that is used by exchanging telnet messages. This section will further explain on TELNETS() function which is within the main() function; but will not explain on codes for MCU initialization or Network initialization.

4.1 TELNETS() Function

TELNETS() function is the basic function for implementing Telnet server, and is based on TCP to create a socket and wait for connection from the client. After connecting with the client, call the init_telopt() function to negotiate Telnet options. To enter commands, call tel_input() function. The next subsections will further explain on init_telopt() function and tel_input() function. Socket s for Telnet use can be 0 to 7. But the port number must be 23 because the Telnet exclusive port is 23 according to the Telnet standards.

```
void TELNETS(SOCKET s, uint16 port)
{
  switch (getSn_SR(s))        {  /* Get the state of socket s */
    case SOCK_ESTABLISHED :     /* If the socket is established */
      if(ch_status[s] == ready_state) {
        printf("\r\nTdkvdELNET server established via SOCKET %d\r\n", s);
        init_telopt(s);       /* Initialize and negotiate the options */
        ch_status[s] = connected_state;
      }

      if(getSn_RX_RSR(s) > 0) {
        tel_input(s);         /* If there is any received data, process it */
      }
      break;

    case SOCK_CLOSE_WAIT :
      printf("\r\nSOCKET %d : CLOSE_WAIT", s);
      disconnect(s);          /* Disconnect the socket s */
      break;
```

```
        case SOCK_CLOSED :
            if(!ch_status[s]) {
                printf("\r\n---------------------------------- \r\n");
                printf("\r\nW5200 TELNET server start!");
                ch_status[s] = ready_state;
            }
            if(socket(s,Sn_MR_TCP,port,0x00) == 0) {        /* reinitialize the socket */
                printf("\r\n%d : Fail to create socket.", s);
                ch_status[s] = close_state;
            } else {
                listen(s);        /* Listen sockets */
                printf("\r\nSOCKET %d : LISTEN", s);
                user_state = USERNAME;
            }
            break;
    }        /* End switch */
}        /* End TELNETS function */
```

Code 4.1 TELNETS() function

4.2 init_telopt() and sendIAC() Function

Init_telopt() function is used to negotiate options between the Telnet server and client. Since only the ECHO option is used for the test code, the user can use only the WILL command to negotiate the ECHO option. As explained in section 2, when sending the control character, the IAC (0xFF) character must be sent together. SendIAC() function is used to send the IAC character and control character.

```
void init_telopt(SOCKET s)
{
    sendIAC(s, WILL, TN_ECHO);          /* Negotiate ECHO option */
}       /* End init_telopt function */

void sendIAC(SOCKET s, uint8 r1, uint8 r2)
{
    switch(r1) {
        case WILL :
            printf("sent : will");          /* WILL command */
            break;
        case WONT :
          printf("sent : wont");          /* WONT command */
          break;

        case DO :
          printf("sent : do");          /*DO command */
          break;

        case DONT :
          printf("sent : dont");          /* DONT command */
          break;

        case IAC :
          break;
      }

      if(r2 <= NOPTIONS) {
        printf("%s\r\n", tel_options[r2]);
      } else {
        printf("%u\r\n", r2);
      }

      sprintf(buf, "%c%c%c", IAC, r1, r2);
      send(s, (uint8 const *)buf, strlen(buf), FALSE);       /* Send IAC, command and option to the client */
}     /* End sendIAC function */
```

Code 4.2 init_telopt() and sendIAC functions

4.3 tel_input() Function

Tel_input() function is the function for processing the entered commands in the Telnet terminal. Please refer to Table 3.1 for each commands and processing method.

```
  void tel_input(SOCKET s)
  {
    uint8 c;
    uint8 input_command[] ="W5200>";
```

```
while(1)
{
  if(getSn_RX_RSR(s) == 0 ) break;          /* If there is no received data, break */
  if(recv(s, &c, 1) == 0) break;          /* If there the received data is 0, break */
  if(user_state == LOGOUT) break;        /* If the user's state is LOGOUT, break */

  if(c != IAC) {          /* If the received data is not a control character */
    data_buf[buf_index++] = c;          /* Save the received data to data_buf */
    putchar(c);

    if(user_state != PASSWORD) {
      sprintf(buf, "%c", c);
      send(s, (uint8 const *)buf, strlen(buf), FALSE);
    }

    if(c == '\n') {        /* If receive an '\n' ASCII code */
      if(buf_index > 1) {

        if(data_buf[buf_index-2] == '\r') {
          data_buf[buf_index-2] = '\0';
        } else {
          data_buf[buf_index-1] = '\0';
        }

        if(user_state != LOGIN) login(s);        /* Call the login() to process login */
        else proc_command(s);        /* Process the received data */

        if(user_state == LOGIN) {
          send(s, input_command, 6, FALSE);
        }
      } else {
        send(s, input_command, 6, FALSE);
      }
      buf_index = 0;
    }
    continue;
  }
```

```
        if(recv(s, &c, 1) == 0) break;
        switch(c) {         /* If received an IAC character */
          case WILL :
            if(recv(s, &c, 1) == 0) break;
            willopt(s, c);          /* Call the willopt() to process WILL command */
            break;
          case WONT :
            if(recv(s, &c, 1) == 0) break;
            wontopt(s, c);          /* Call the wontopt() to process WONT command */
            break;
          case DO :
            if(recv(s, &c, 1) == 0) break;
            doopt(s, c);          /* Call the doopt() to process DO command */
            break;
          case DONT :
            if(recv(s, &c, 1) == 0) break;
            dontopt(c);         /* Call the dontopt() to process DONT command */
            break;
          case IAC :
            break;
        }
        break;
      }
}        /* End tel_input function */
```

Code 4.3 tel_input() function

4.4 login() Function

The login() function is for checking the ID and password when logging in to the Telnet server. Telnet connection will be successful if the entered information matches with the saved login information. But if the entered information does not match the saved login information, a message will appear to enter ID and password again.

```
uint8 telnet_ID[] = {
      "wiznet"
};
uint8 telnet_PW[] = {
      "0000"
};
void login(SOCKET s)
{
  if(user_state == USERNAME) {          /* input the client ID and Password */
      strcpy((char *)user_name, data_buf);
      sprintf(buf, "Password : ");
      send(s, (uint8 const *)buf, strlen(buf), FALSE);
      user_state = PASSWORD;
      return;
  } else if(user_state == PASSWORD) {
      strcpy((char *)user_password, data_buf);

      /*Check the client ID and Password*/
      if(!(strcmp((char const *)user_name, (char const *)telnet_ID)) && !(strcmp((char const
                                                   *)user_password, (char const *)telnet_PW))) {
         sprintf(buf, "\r\n======================");
         send(s, (uint8 const *)buf, strlen(buf), FALSE);
         sprintf(buf, "\r\nSuccessfully connected!\
           \r\nImplemented Commands : HELP, GET LED, LED3 ON/OFF, LED4 ON/OFF, EXIT\r\n");
         send(s, (uint8 const *)buf, strlen(buf), FALSE);
         sprintf(buf, "======================\r\n");
         send(s, (uint8 const *)buf, strlen(buf), FALSE);
         user_state = LOGIN;
         return;
      } else {
         /* If the ID or Password incorrect, print error msg */
         sprintf(buf, "\r\nID or Password incorrect!\r\n");
         send(s, (uint8 const *)buf, strlen(buf), FALSE);
         sprintf(buf, "ID : ");
         send(s, (uint8 const *)buf, strlen(buf), FALSE);
         user_state = USERNAME;
         return;
      }
  }
}      /* End login function */
```

Code 4.4 login() function

4.5 proc_command() Function

The proc_command() function processes the entered commands in the tel_input() function. It defines about the "HELP, GET LED, LED0 ON/OFF, LED1 ON/OFF and LED2 ON/OFF commands. For the undefined command it shows "BAD COMMAND" message.

```
void proc_command(SOCKET s)
{
   char **cmdp;
   char *cp;
   char *help = {"HELP: Show all available commands\
      \r\nGET LED: Show all LED status\
      \r\nLED3 ON/OFF: Turn ON/OFF the LED3\
      \r\nLED4 ON/OFF: Turn ON/OFF the LED4\
         \r\nEXIT: Exit from W5200 TELNET server\r\n"}; /* command HELP : Message */

   for(cp = data_buf; *cp != '\0';   cp++){
      *cp = tolower(*cp);        /* Translate big letter to small letter */
   }

   if(*data_buf != '\0') {
      /* Find the input command in table; if it isn't there, return Syntax Error */
      for(cmdp = commands; *cmdp != NULL; cmdp++) {
         if(strncmp(*cmdp, data_buf, strlen(*cmdp)) == 0) break;
      }

      if(*cmdp == NULL) {
         printf("NULL command\r\n");
         sprintf(buf, "%s : BAD command\r\n", data_buf);
         send(s, (uint8 const *)buf, strlen(buf), FALSE);
         return;
      }

      switch(cmdp - commands) {

         case HELP_CMD :        /* Process HELP command */
            printf("HELP_CMD\r\n");
            sprintf(buf, help);
```

```
        send(s, (uint8 const *)buf, strlen(buf), FALSE);
        break;

    case GET_LED_CMD :          /* Process GET LED command */
        printf("GET_LED_CMD\r\n");
        sprintf(buf, "LED%d is %s\r\n", 3, GPIO_ReadOutputDataBit(GPIOA, LED3) ? "OFF" : "ON");
        send(s, (uint8 const *)buf, strlen(buf), FALSE);
        sprintf(buf, "LED%d is %s\r\n", 4, GPIO_ReadOutputDataBit(GPIOA, LED4) ? "OFF" : "ON");
        send(s, (uint8 const *)buf, strlen(buf), FALSE);
        break;

    case LED3_ON_CMD :          /* Process LED3 ON command */
        printf("LED3_ON_CMD\r\n");
        sprintf(buf, "Turn ON the LED3\r\n");
        send(s, (uint8 const *)buf, strlen(buf), FALSE);
        GPIO_ResetBits(GPIOA, LED3); // led3 on
        break;

    case LED4_ON_CMD :          /* Process LED4 ON command */
        printf("LED4_ON_CMD\r\n");
        sprintf(buf, "Turn ON the LED4\r\n");
        send(s, (uint8 const *)buf, strlen(buf), FALSE);
        GPIO_ResetBits(GPIOA, LED4); // led4 on
        break;

    case LED3_OFF_CMD :         /* Process LED3 OFF command */
        printf("LED3_OFF_CMD\r\n");
        sprintf(buf, "Turn OFF the LED3\r\n");
        send(s, (uint8 const *)buf, strlen(buf), FALSE);
        GPIO_SetBits(GPIOA, LED3); // led3 off
        break;

    case LED4_OFF_CMD :         /* Process LED4 OFF command */
        printf("LED4_OFF_CMD\r\n");
        sprintf(buf, "Turn OFF the LED4\r\n");
        send(s, (uint8 const *)buf, strlen(buf), FALSE);
        GPIO_SetBits(GPIOA, LED4); // led4 off
        break;
        case EXIT_CMD :        /* Process EXIT command */
          printf("EXIT command\r\n");
          sprintf(buf, "EXIT command\r\nGood bye!\r\nLogout from W5200 TELNET\r\n");
          send(s, (uint8 const *)buf, strlen(buf), FALSE);
          close(s);
          user_state = LOGOUT;
          break;

        default :
          break;
      }
    }
}       /* End proc_command function */
```

Code 4.5 proc_command() function

4.6 willopt(), wontopt(), doopt() and dontopt() Functions

willopt(), wontopt(), doopt() and dontopt() functions are command for negotiation of Telnet options. They need the socket s and option as input parameters. For more detail about each command and options, please refer to Table 2.1 and 2.2.

```
void willopt(SOCKET s, uint16 opt)
{
  int ack;
  printf("recv: will");
  if(opt <= NOPTIONS) {
    printf("%s\r\n", tel_options[opt]);
  } else {
    printf("%u\r\n", opt);
  }

  switch(opt) {
  case TN_TRANSMIT_BINARY :
  case TN_ECHO :
  case TN_SUPPRESS_GA :
    ack = DO;        /* If receive 'WILL' and it has TN_SUPPRESS_GA option, transmit 'DO' */
```

```
        break;
    default :
        ack = DONT;          /* Refuse other commands which not defined */
    }
    sendIAC(s, ack, opt);
}        /* End willopt function */

void wontopt(SOCKET s, uint16 opt)
{
    printf("recv: wont");
    if(opt <= NOPTIONS) {
        printf("%s\r\n", tel_options[opt]);
    } else {
        printf("%u\r\n", opt);
    }

    switch(opt) {
    case TN_TRANSMIT_BINARY :
    case TN_ECHO :
    case TN_SUPPRESS_GA :         /* If receive WONT command with TN_SUPPRESS_GA option */
        if(remote[opt] == 0) {
            remote[opt] = 1;          /* Set the TN_SUPPRESS_GA option */
            sendIAC(s, DONT, opt);          /* Send DONT command with TN_SUPPRESS_GA */
        }
        break;
    }
}        /* End wontopt function */

void doopt(SOCKET s, uint16 opt)
{
    printf("recv: do ");
    if(opt <= NOPTIONS) {
        printf("%s\r\n", tel_options[opt]);
    } else {
        printf("%u\r\n", opt);
    }

    switch(opt) {
```

```
    case TN_SUPPRESS_GA :        /* If receive DO command with TN_SUPPRESS_GA option */
      sendIAC(s, WILL, opt);         /* Send WILL command with TN_SUPPRESS_GA */
      break;
    case TN_ECHO :          /* If receive DO command with TN_ECHO option */
      sprintf(buf, "WELCOME!\r\nID : ");
      send(s, (uint8 const *)buf, strlen(buf), FALSE);
      break;
    default :
      sendIAC(s, WONT, opt);
    }
  }      /* End doopt function */

  void dontopt(uint16 opt)
  {
    printf("recv: dont ");
    if(opt <= NOPTIONS) {
      printf("%s\r\n", tel_options[opt]);
    } else {
      printf("%u\r\n", opt);
    }

    switch(opt) {
    case TN_TRANSMIT_BINARY :
    case TN_ECHO :
    case TN_SUPPRESS_GA :        /* If receive DONT command with TN_SUPPRESS_GA option */
      if(remote[opt] == 0) {         /* Set the TN_SUPPRESS_GA option */
        remote[opt] = 1;
      }
      break;
    }
}      /* End dontopt function */
```

Code 4.6 willopt(), wontopt(), doopt() and dontopt() functions

Document History Information

Version	Date	Descriptions
Ver. 1.0	May, 2011	Release with W5200 launching