# How to implement Telnet server using W7100A

version 1.1

For more information, visit our website at [http://www.wiznet.co.kr](http://www.wiznet.co.kr)

# Table of Contents

# 1 Introduction

This document will explain about Telnet, and implement the Telnet server by using iMCU7100EVB. There is no extra file system for Telnet but the LED0, 1 and 2 which connected to the GPIO port of iMCU7100EVB. **(Note : the W7100A QFN 64pin package cannot control the LEDs because it has limited number of GPIO pins.)** They can be controlled by using telnet. Section 2 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 Keil uVision3 compiler.

# 2 Telnet

Telnet (teletype network) is a network protocol used on the 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. Network equipments with TCP/IP protocol stack basically supports Telnet service. Telnet is a widely used client/server application program.

Since more browsers with convenient functions came out, the importance of Telnet has decreased. 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 with internet connection.

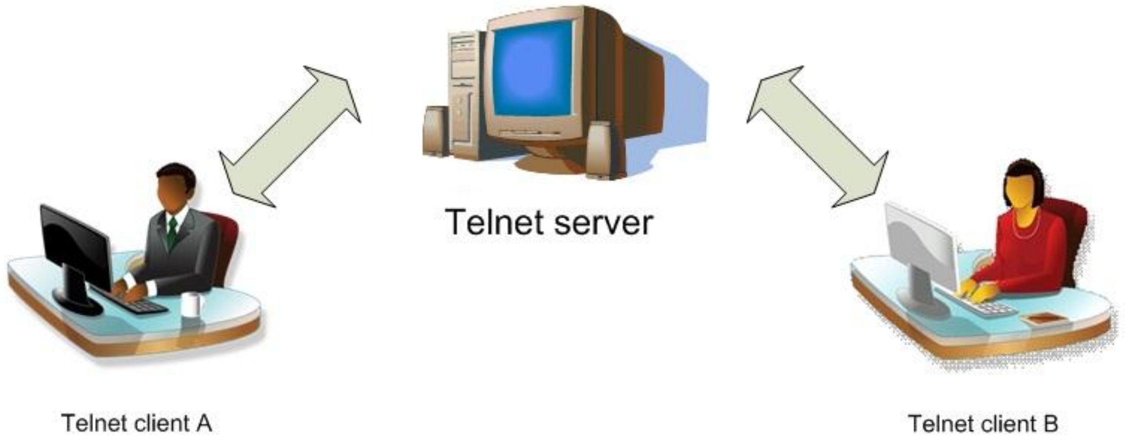


Fig. 2.1 Telnet system

## 2.1 NVT (Network Virtual Terminal)

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

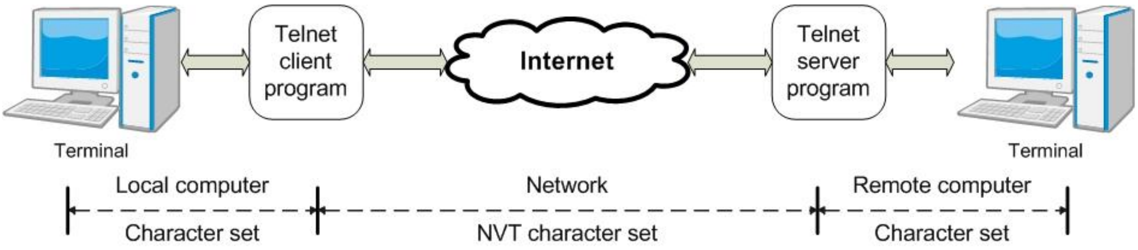


Fig. 2.2 NVT of telnet system

NVT uses two kinds of character combinations; one is for data-use and another is for remote control.

NVT for data use usually uses NVT ASCII. NVT ASCIII is an 8bit character combination; the lower 7bits are the same as US ASCII and the highest bit is 0. The highest bit can be 1 instead of 0; in this case, the option negotiation between the telnet client and server should be defined in advance.

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

Tab. 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, send 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 back to, but not including, the last "CR LF" sequence sent over the TELNET connection. |
| Go ahead | 249 | The GA signal. |
| SB | 250 | Indicates that what follows is subnegotiation of the indicated option. |
| WILL | 251 | Indicates the desire to begin performing, or confirmation that you are now 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.

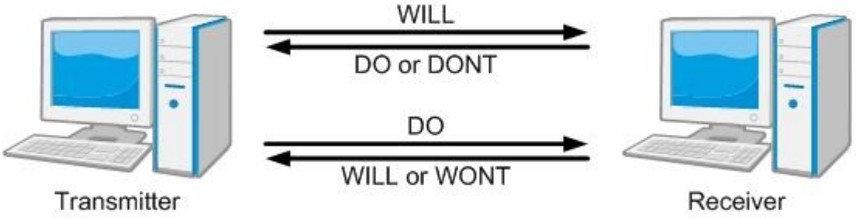
Tab. 2.2 Options for telnet

| 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 |

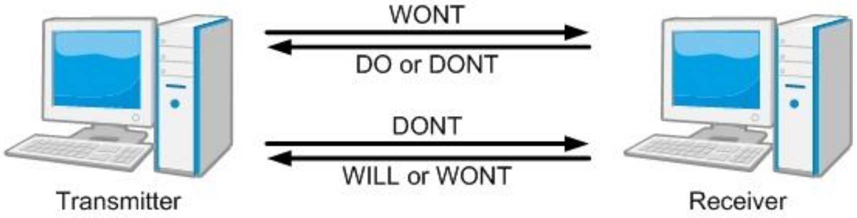
The negotiation of options between the client and server is necessary in order to use various options of telnet. As shown in Table 2.1, 4 control characters (WILL, WONT, DO, and DON’T) are used for negotiation of options.

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.



**(a) Activate request**



**(b) Deactivate request**

Fig. 2.3 Negotiation of telnet options

## 2.3 Telnet Operation Mode

Most operation 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, echoes all input characters and do not send it until a line is completed.

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

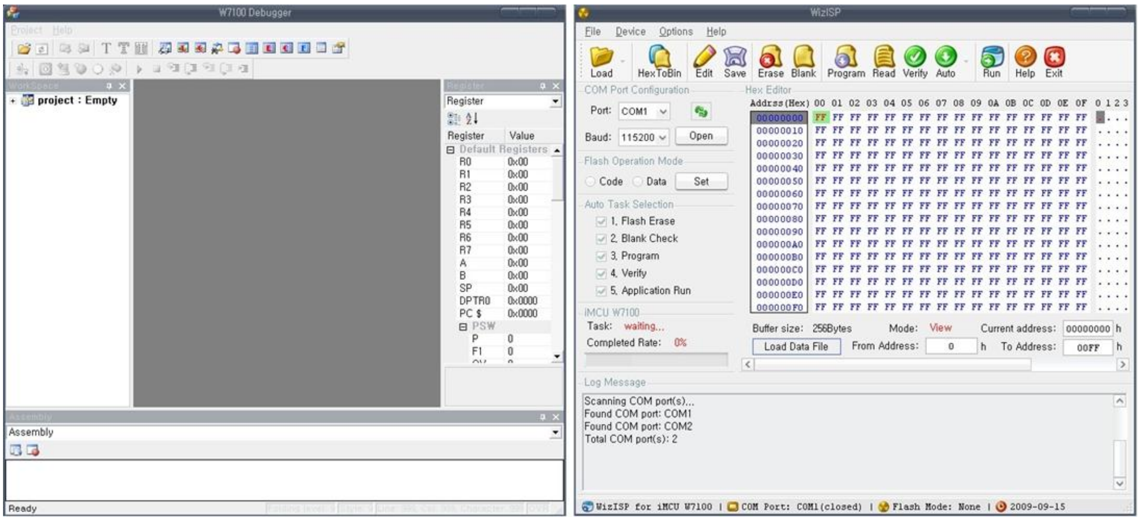
-**Character mode**: When the client enters characters, each character is sent to the server. The server allows the echo-character to appear on the client screen. In this mode, some delay can be happened when an echo-character’s transmission is too slow; in this case, it gives some load to the network.

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

# 3 Telnet Demonstration

Section 3 will demonstrate Telnet as explained in section 2. The telnets.hex code that will be used for demonstration is implemented to control LED0, LED1, LED2, which are connected to the GPIO port. **(Note : the W7100A QFN 64pin package cannot control the LEDs because it has limited number of GPIO pins.)** The iMCU7100EVB will be the Telnet server and the user's PC will be the Telnet client. Since the status of iMCU7100EVB will output serial message, it can be checked using the Hyper terminal or a common serial terminal. Especially the hyper terminal is the basic application program from MS windows and can be used easily.

Connect the power cable for iMCU7100EVB, LAN cable, and serial cable for Telnet demonstration. For more details on iMCU7100EVB board, please refer to the 'iMCU7100EVB User's Guide.' The programs for loading the 'telnet server HEX code' on the iMCU7100EVB board are the W7100 debugger program or the WizISP program. When using the WizISP program, connect the serial cable to the PC and board. When using the W7100 debugger program, connect the debugger cable to the PC and board. For more details on iMCU7100 debugger program and WizISP program, please refer to 'W7100 Debugger Guide' and 'WizISP Program User Guide for W7100.' These programs are included in the bundle CD that are provided with the EVB, or can be downloaded from the WIZnet home page.



| Register | Value |
|---|---|
| Default Registers | |
| R0 | 0x00 |
| R1 | 0x00 |
| R2 | 0x00 |
| R3 | 0x00 |
| R4 | 0x00 |
| R5 | 0x00 |
| R6 | 0x00 |
| R7 | 0x00 |
| A | 0x00 |
| B | 0x00 |
| SP | 0x00 |
| DPTR0 | 0x0000 |
| PC $ | 0x0000 |
| PSW | |
| P | 0 |
| F1 | 0 |

Port: COM1

Baud: 115200

○ Code ○ Data

☑ 1. Flash Erase
☑ 2. Blank Check
☑ 3. Program
☑ 4. Verify
☑ 5. Application Run

Task: waiting...

Completed Rate: 0%

```
Addrss(Hex) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000010    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000020    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000030    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000040    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000050    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000060    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000070    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000080    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000090    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000A0    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000B0    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000C0    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000D0    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000E0    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000F0    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Buffer size: 256Bytes Mode: View Current address: 00000000 h

From Address: 0 h To Address: 00FF h

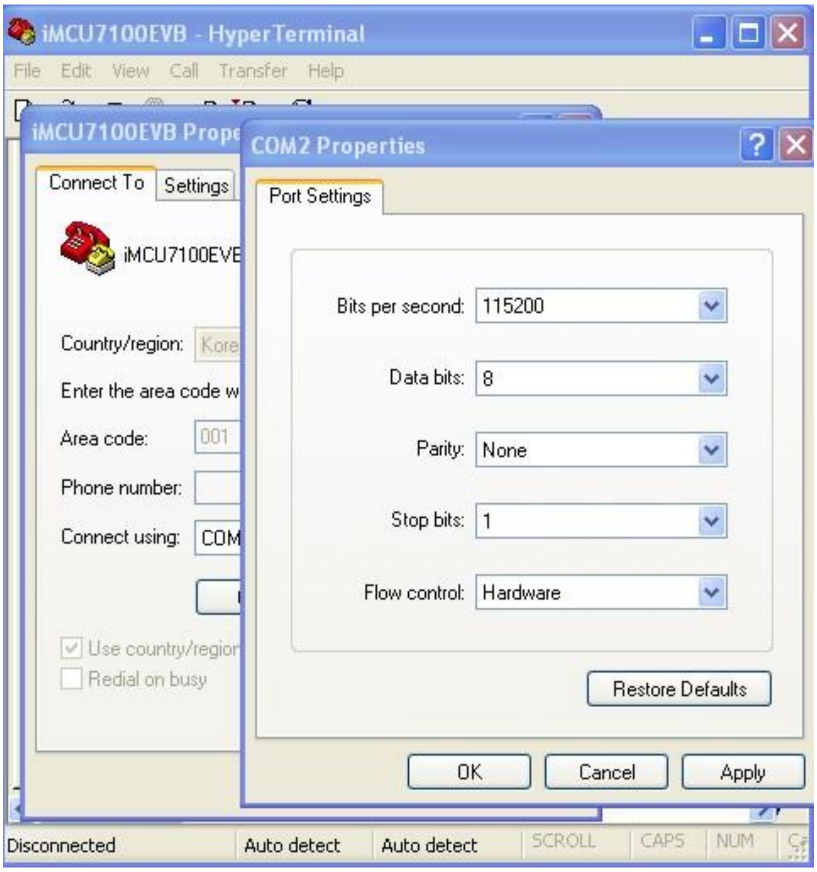
```
Scanning COM port(s)...
Found COM port: COM1
Found COM port: COM2
Total COM port(s): 2
```

WizISP for iMCU W7100 | COM Port: COM1(closed) | Flash Mode: None | 2009-09-15

(a) W7100 Debugger (b) WizISP

Fig. 3.1 W7100 Debugger & WizISP program interface

After loading the telnets.hex file on the iMCU7100EVB board, turn on the Hyper terminal program. If use the WizISP program, turn off the BOOTSEL pin on the EVB board and reset the EVB board. If you are going to use the WizISP program, click on 'run' or end the debugger and reset the EVB board. The COM2 port from the settings shown in fig 3.2, is the serial port number that is being used in the test PC. Therefore this number should be set according to the user's PC settings. Settings for Baud-rate and etc should be modified according to telnet.hex code also. If the Hyper terminal settings are correctly set, some message will show like shown in fig 3.3.



Bits per second: 115200

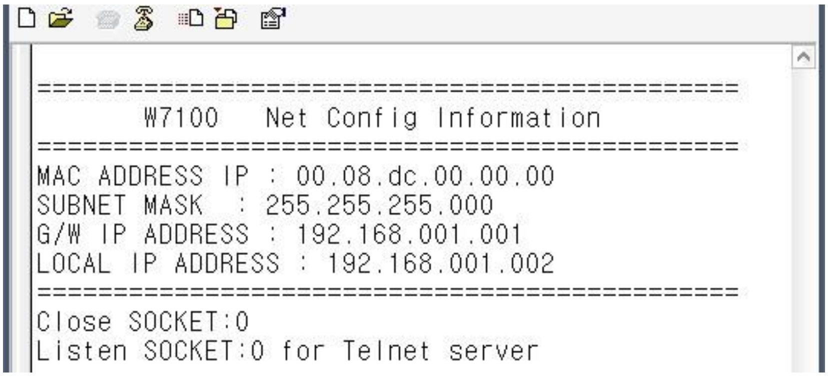
Data bits: 8

Parity: None

Stop bits: 1

Flow control: Hardware

Fig. 3.2 Hyper terminal setting



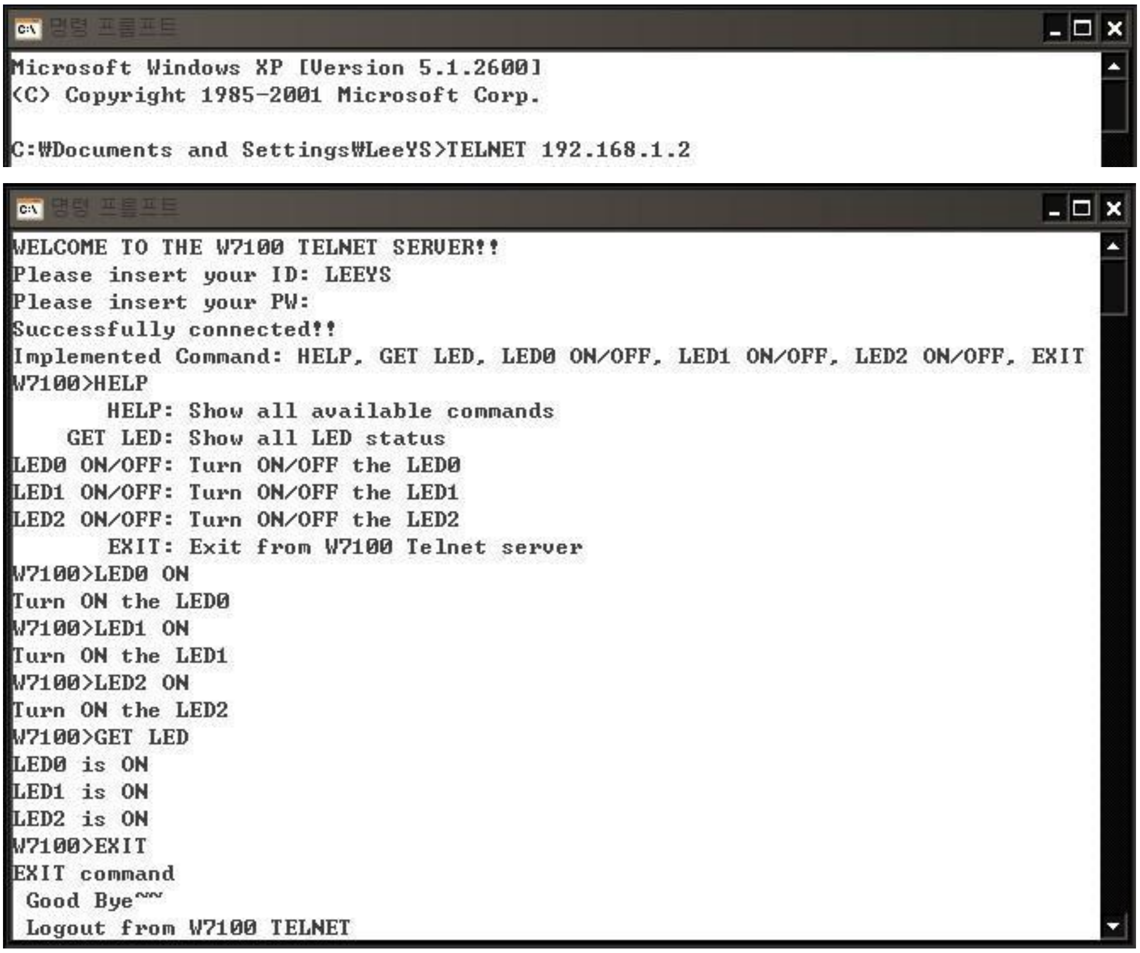
```
==============================================
       W7100   Net Config Information
==============================================
MAC ADDRESS IP : 00.08.dc.00.00.00
SUBNET MASK   : 255.255.255.000
G/W IP ADDRESS : 192.168.001.001
LOCAL IP ADDRESS : 192.168.001.002
==============================================
Close SOCKET:0
Listen SOCKET:0 for Telnet server
```

Fig. 3.3 Hyper terminal message

Check the Telnet server IP using the Hyper terminal, and open the command window to enter “telnet 192.168.1.2.” The Telnet server IP should be set according to the user’s network. A virtual IP 192.168.1.2 was used for the test code. After the connection with Telnet server is successful, enter the ID and PW. User can use any ID and PW to login. Many commands from the test code can be used after logging in. These commands are listed in Table 3.1, and the process for using the commands are shown in fig 3.4. User can confirm the results of these commands using Telnet terminal and Hyper terminal.

Tab. 3.1 Commands for telnet

| Command | Description |
|---|---|
| HELP | Show all available commands |
| GET LED | Show all LED's status |
| LED0 ON | Turn on the LED0 |
| LED1 ON | Turn on the LED1 |
| LED2 ON | Turn on the LED2 |
| LED0 OFF | Turn off the LED0 |
| LED1 OFF | Turn off the LED1 |
| LED2 OFF | Turn off the LED2 |
| EXIT | Exit from telnet server |



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:₩Documents and Settings₩LeeYS>TELNET 192.168.1.2
```

```
WELCOME TO THE W7100 TELNET SERVER!!
Please insert your ID: LEEYS
Please insert your PW:
Successfully connected!!
Implemented Command: HELP, GET LED, LED0 ON/OFF, LED1 ON/OFF, LED2 ON/OFF, EXIT
W7100>HELP
       HELP: Show all available commands
    GET LED: Show all LED status
LED0 ON/OFF: Turn ON/OFF the LED0
LED1 ON/OFF: Turn ON/OFF the LED1
LED2 ON/OFF: Turn ON/OFF the LED2
       EXIT: Exit from W7100 Telnet server
W7100>LED0 ON
Turn ON the LED0
W7100>LED1 ON
Turn ON the LED1
W7100>LED2 ON
Turn ON the LED2
W7100>GET LED
LED0 is ON
LED1 is ON
LED2 is ON
W7100>EXIT
EXIT command
 Good Bye~^^
 Logout from W7100 TELNET
```

Fig. 3.4 Telnet terminal running message

User can turn on/off LED using the 'LEDx ON/OFF' command, and check the status of LED remotely by using the 'GET LED' command. In figure 3.4, LED0, LED1, and LED2 are turned on. The status of LED can be checked by just watching the lights on iMCU7100EVB as shown in Figure 3.5.

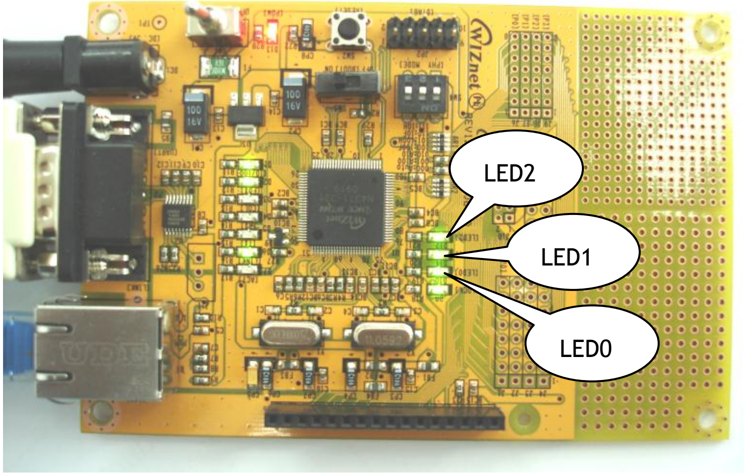


Fig. 3.5 iMCU7100EVB running with Telnet server

# 4 Implementation Code

This section will go over the Telnet server example codes loaded on iMCU7100EVB. 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)
{
  uint8 first;
    switch(getSn_SR(s))      /* Get the state of socket s */
    {
      case SOCK_ESTABLISHED:      /* If the socket is established */
         if(first == 1)       /* If it is first connection with client */
         {
           printf("W7100 TELENT server started via SOCKET%bu\r\n", s);
           init_telopt(s);        /* Initialize and negotiate the options */
           first = 0;
         }
         if((getSn_RX_RSR(s)) > 0)   tel_input(s);   /* If there is any received data, process it*/
         break;
      case SOCK_CLOSE_WAIT:
         disconnect(s);       /* Disconnect the socket s */
         break;
      case SOCK_CLOSED:
         printf("Close SOCKET:%bu\r\n", s);
         close(s);       /* Close the socket s */
         socket(s, Sn_MR_TCP, port, 0);       /* Open the socket s for TCP */
         break;
      case SOCK_INIT:
         listen(s);     /* Listen socket s */
         printf("Listen SOCKET:%bu for Telnet server\r\n", s);
         user_state = USERNAME;
         first = 1;
         break;
     }   /* END switch */
}   /* END TELNETS function */
```

Code 4.1 TELNETS() fucntion

## 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. 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 */
}

void sendIAC(SOCKET s, char r1, char r2)
{
  switch(r1)
  {
    case WILL:     /* WILL command */
      printf("sent: will");
      break;
    case WONT:     /* WONT command */
      printf("sent: wont");
      break;
    case DO:     /*DO command */
      printf("sent: do");
      break;
    case DONT:     /* DONT command */
      printf("sent: dont");
      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, buf, strlen(buf));        /* Send IAC, command and option to the client */
}
```

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 xdata c;
   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 == LOGOUT)   break;
         if(user_state != PASSWORD){
             sprintf(buf, "%c", c);
             send(s, buf, strlen(buf));
         }
         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';
                proc_command(s);          /* Process the received data */
                if(user_state == LOGIN) {
                   sprintf(buf, "W7100>");
                   send(s, buf, strlen(buf));
                }
             }
             else{
                sprintf(buf, "W7100>");
                send(s, buf, strlen(buf));
             }
             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;
   }
   return;
}
```

Code 4.3 parseMSG() function

## 4.4 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)
{
   uint8 i;
   char **cmdp, *cp;
   char *help = {"HELP: Show all available commands\r\n\GET LED: Show all LED status\
                 \r\nLED0 ON/OFF: Turn ON/OFF the LED0\r\nLED1 ON/OFF: Turn ON/OFF the LED1\
                 \r\nLED2 ON/OFF: Turn ON/OFF the LED2\r\nEXIT: Exit from W7100 Telnet server\r\n"};
              /*Translate the first word to lower case*/
   for(cp = data_buf; *cp !='\0'; cp++){
      *cp = tolower(*cp);        /* Translate big letter to small letter */
   }
   if(user_state == USERNAME){
      strcpy(user_name, data_buf);
      sprintf(buf, "Please insert your PW: ");
      send(s, buf, strlen(buf));
      user_state = PASSWORD;
      return;
   }
   else if(user_state == PASSWORD){
      strcpy(user_password, data_buf);
      sprintf(buf, "\r\nSuccessfully connected!!\r\nImplemented Command: \
                HELP, GET LED, LED0 ON/OFF, LED1 ON/OFF, LED2 ON/OFF, EXIT\r\n");
      send(s, buf, strlen(buf));
      user_state = LOGIN;
      return;
   }        /*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, "BAD command\r\n");
      send(s, buf, strlen(buf));
      return;
   }
   switch(cmdp - commands){
      case HELP_CMD:              /* Process HELP command */
         printf("HELP_CMD\r\n");
         sprintf(buf, help);
         send(s, buf, strlen(buf));
         break;
      case GET_LED_CMD:           /* Process GET LED command */
```

```
    printf("GET_LED_CMD\r\n");
    for(i = 0 ; i < 3 ; i++){
        sprintf(buf, "LED%bd is %s\r\n", i, ((P0 >> (i+3)) & 0x01) ? "OFF" : "ON");
        send(s, buf, strlen(buf));
    }
    break;
  case LED0_ON_CMD:            /* Process LED0 ON command */
    printf("LED0_ON_CMD\r\n");
    sprintf(buf, "Turn ON the LED0\r\n");
    send(s, buf, strlen(buf));
    P0_3 = 0;          /* Set 0 the GPIO 0_3, connected with active low LED */
    break;
  case LED1_ON_CMD: /* Process LED1 ON command */
    printf("LED1_ON_CMD\r\n");
    sprintf(buf, "Turn ON the LED1\r\n");
    send(s, buf, strlen(buf));
    P0_4 = 0;          /* Set 0 the GPIO 0_4, connected with active low LED */
    break;
  case LED2_ON_CMD:            /* Process LED2 ON command */
    printf("LED2_ON_CMD\r\n");
    sprintf(buf, "Turn ON the LED2\r\n");
    send(s, buf, strlen(buf));
    P0_5 = 0;           /* Set 0 the GPIO 0_5, connected with active low LED */
    break;
  case LED0_OFF_CMD:           /* Process LED0 OFF command */
    printf("LED0_OFF_CMD\r\n");
    sprintf(buf, "Turn OFF the LED0\r\n");
    send(s, buf, strlen(buf));
    P0_3 = 1;           /* Set 1 the GPIO 0_3, connected with active low LED */
    break;
  case LED1_OFF_CMD:           /* Process LED1 OFF command */
    printf("LED1_OFF_CMD\r\n");
    sprintf(buf, "Turn OFF the LED1\r\n");
    send(s, buf, strlen(buf));
    P0_4 = 1;           /* Set 1 the GPIO 0_4, connected with active low LED */
    break;
  case LED2_OFF_CMD:           /* Process LED2 OFF command */
    printf("LED2_OFF_CMD\r\n");
```

```
            sprintf(buf, "Turn OFF the LED2\r\n");
            send(s, buf, strlen(buf));
            P0_5 = 1;               /* Set 1 the GPIO 0_5, connected with active low LED */
            break;
        case EXIT_CMD:              /* Process EXIT command */
            printf("EXIT command\r\n");
            sprintf(buf, "EXIT command\r\n Good Bye~~\r\n Logout from W7100 TELNET");
            send(s, buf, strlen(buf));
            close(s);
            user_state = LOGOUT;
            break;
        default:
            break;
    }
}
```

Code 4.4 proc_command() function

## 4.5 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, int 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);
}
```

```
void wontopt(SOCKET s, int 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;
  }
}

void doopt(SOCKET s, int 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 TO THE W7100 TELNET SERVER!!\r\nPlease insert your ID: ");
      send(s, buf, strlen(buf));
      break;
    default:
      sendIAC(s, WONT, opt);
  }
}

void dontopt(int 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) remote[opt] = 1;    /* Set the TN_SUPPRESS_GA option */
            break;
    }
}
```

Code 4.5 willopt(), wontopt(), doopt() and dontopt() functions

# Document History Information

| Version | Date | Descriptions |
|---|---|---|
| Ver. 0.9 Beta | Oct. 2009 | Release with W7100 launching |
| Ver. 1.0 | Mar. 2011 | Modify for W7100A 64pin package |

## Copyright Notice

Technical Support: support@wiznet.co.kr
Sales & Distribution: sales@wiznet.co.kr

For more information, visit our website at http://www.wiznet.co.kr