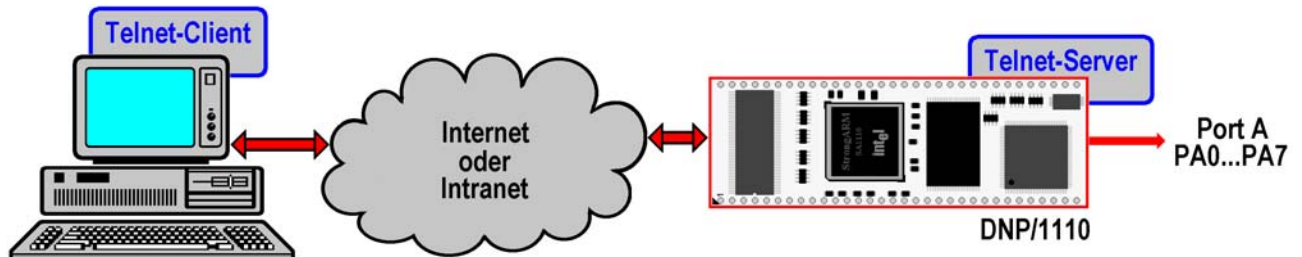


How to write a Measurement Telnet Server

A measurement Telnet server allows you to access remote I/Os with a standard Telnet client program. The following samples shows a way to set the LEDs of a DNP/EVA2 evaluation board with a Telnet client program.



The C sample code in this document forms a measurement (MSR) Telnet server for the DNP/1110. The server allows you to enter values between 0 and 255 within a Telnet client window. This value is converted to a 8-bit binary number. The measurement Telnet server writes the binary number to the DNP/1110 parallel I/O port A. This port drives the DNP/EVA2 LED array.

- **1. Step:** Download the sample source code `msrtsvr6.c` from www.dilnetpc.com (i.e. enter the URL www.dilnetpc.com/msrtsvr6.c to your browser). The following listing shows this sample code. Please note: We have remove some text within some `printf`-statements for cutting the total length of some lines.

```
// msrtsvr6.c: Simple MSR Telnet server program. Use cross-gcc ...
// Vers. 2.00 - 24.July 2002
// k.d.walter@t-online.de
```

```
// Includes
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

```
// Typedefs
```

```
typedef unsigned char  u8;
typedef unsigned short u16;
typedef unsigned int   u32;
```

```
// Defines
```

```

#define GPLR (*(pGPIO_BASE + 0x00 / 4)) // SA-1110: GPLR @ 0x90040000
#define GPDR (*(pGPIO_BASE + 0x04 / 4)) // SA-1110: GPDR @ 0x90040004
#define GPSR (*(pGPIO_BASE + 0x08 / 4)) // SA-1110: GPSR @ 0x90040008
#define GPCR (*(pGPIO_BASE + 0x0C / 4)) // SA-1110: GPCR @ 0x9004000C
#define MAP_PAGESIZE 4096UL // For pMapmemory

//=====
// Functions: 1. pMapmemory

void *pMapmemory (off_t phy_addr, size_t phy_lenght)
{
    int iFd;
    void *pMem;

    if ((phy_addr % MAP_PAGESIZE) != 0) {
        fprintf (stderr, "\n Physical address ...");
        return (NULL);
    }

    if ((phy_lenght % MAP_PAGESIZE) != 0) {
        fprintf (stderr, "\n Physical lenght ...");
        return (NULL);
    }

    // Open mem device for read/write...

    iFd= open("/dev/mem", O_RDWR | O_SYNC);
    if (iFd < 0) {
        fprintf (stderr, "\n Open of /dev/mem fail !\n");
        return (NULL);
    }

    // Get pointer to DNP1110 memory...

    pMem= mmap (NULL, phy_lenght, (PROT_READ | PROT_WRITE),
                MAP_SHARED, iFd, phy_addr);

    if ((pMem == MAP_FAILED) || (pMem == NULL)) {
        fprintf (stderr, "\n Mmap of /dev/mem fail !\n");
        return (NULL);
    }

    // Close mem device...

    if (close(iFd) != 0)
        fprintf(stderr, "\n Close of /dev/mem fail !\n");
}

```

```

    return (pMem);
}

//=====
// Functions: 2. main

int main (int argc, char* argv[])
{
    int nLen= sizeof (struct sockaddr), nRet, s1, s2,
        noLine, link, cnt, i, err;
    char szBuf[256], szLine[256];
    struct sockaddr_in saServer, saClient;
    volatile u32 *pGPIO_BASE;

    // Check user access rights...

    if (geteuid () != 0) {
        printf ("\n No root access rights!\n");
        return (-1);
    }

    // Map SA-1110 GPIO register space at 0x90040000 / size 4096 bytes...

    pGPIO_BASE= pMapmemory (0x90040000, 0x00001000);
    if (pGPIO_BASE == NULL) {
        printf ("\n Can not map memory at 0x90040000!\n");
        return (-1);
    }

    // Check for port argument. Print usage if no port number...

    if (argc != 2) {
        printf ("\n Usage.: msrtsvr6 PortNumber");
        printf ("\n Sample: msrtsvr6 5000<enter>\n");
        return (-1);
    }

    // Set DNP/1110 PIO port A = output. Reset port to 0...

    GPDR |= 0xFF00;                // GPDR = 0xFFFFFFF
    GPCR = (0xff << 8);           // GPCR = 0x0000XX00
    GPSR = (0 << 8);              // GPSR = 0x0000XX00

    // Create a TCP/IP stream socket...

    s1= socket (AF_INET, SOCK_STREAM, 0);

    // ... and check for error...

```

```
if (s1 < 0) {
    printf ("\n Can not create socket.\n");
    return (-1);
}

// Fill in the server internet address structure...

saServer.sin_family= AF_INET;           // Address family
saServer.sin_addr.s_addr= INADDR_ANY;   // Any IP address
saServer.sin_port= htons (atoi (argv[1])); // Portnumber
memset (&(saServer.sin_zero), 0x00, 8); // Zero the rest...

// Bind IP address/portnumber to socket...

nRet= bind (s1, (struct sockaddr *) &saServer,
            sizeof (struct sockaddr));

// ... and check for any errors...

if (nRet < 0) {
    printf ("\n Can not bind socket to address structure.\n");
    close (s1);
    return (-1);
}

// Execute a listen (queue client connect requests)...

nRet= listen (s1, 5);

// ... and check for any errors...

if (nRet < 0) {
    printf ("\n Can not use listen function.\n");
    close (s1);
    return (-1);
}

// Server loop. Wait for client request without end...

while (1) {

    // Waiting for connect from client...

    printf ("\n Waiting for connect...");
    fflush (stdout);
    if ((s2= accept (s1, (struct sockaddr *) &saClient, &nLen)) < 0) {
        printf ("\n Can not use accept function.\n");
    }
}
```

```
    close (s1);
    close (s2);
    return (-1);
}
printf ("\n Connect to %s\n", inet_ntoa (saClient.sin_addr));
fflush (stdout);

// Send usage message to client...

sprintf (szBuf, "\033[2J\033[H"); // Clear screen, cursor home
send (s2, szBuf, strlen (szBuf), 0);
sprintf (szBuf, "MSR-Telnet-Server. Vers. 1.00 ...");
send (s2, szBuf, strlen (szBuf), 0);
sprintf (szBuf, "Enter number 0 - 255 and press enter or ...");
send (s2, szBuf, strlen (szBuf), 0);

// We have a link...

link= 1;
while (link) {

    // Get next line with number or CTRL-C...

    noLine= 1;
    err= 0;
    cnt= 0;
    sprintf (szBuf, ">");
    send (s2, szBuf, strlen (szBuf), 0);
    memset (szLine, 0, sizeof (szBuf));
    while (noLine) {

        // Wait for client data...

        memset (szBuf, 0, sizeof (szBuf));
        nRet= recv (s2, szBuf, sizeof (szBuf), 0);

        // Watch the link state...

        if (nRet <= 0) {
            link= 0; // Disconnect by client
            noLine=0;
        }
        else {

            // Remove all char´ s from szBuf...

            i= 0;
            while (nRet != 0 && noLine != 0 && link != 0) {
```

```
// If valid char: save data...

if (szBuf[i] >= 0x30 && szBuf[i] <= 0x39) {
    szLine[cnt]= szBuf[i];
    cnt++;
}
else {

    // Count invalid char' s...

    if (szBuf[i] != '\r' && szBuf[i] != 0x04)
        err++;
}

// Check for line end ('\r') or max char count...

if (szBuf[i] == '\r' || cnt > 16)
    noLine= 0;

// Check for session end (CTRL-D)...

if (szBuf[i] == 0x04) {
    link= 0;
    noLine=0;
}

// Next char...

i++;
nRet--;
}
}

// We have a new line...

if (link == 1) {

    // Check for valid number (0 - 255)...

    if (strlen (szLine) == 0 || atoi (szLine) > 255 || err != 0)
        sprintf (szBuf, "ERR\n\r");
    else {
        sprintf (szBuf, "OK\n\r");

        // Output operation: write value to port A
```

```

        GPCR = ((atoi (szLine)^0xff) << 8); // GPCR = 0x0000XX00
        GPSR = (atoi (szLine) << 8);      // GPSR = 0x0000XX00
    }
    send (s2, szBuf, strlen (szBuf), 0);
}
}

// Close data socket...

close (s2);
}
}

```

- **2. Step:** Build an executable from the source code. Use the following listing for a makefile. With a makefile simply enter **make msrtsvr6** for build an executable.

```

CROSS    =    /usr/local/arm/2.95.2/bin/arm-linux-
CC       =    $(CROSS)gcc
CFLAGS   =    -Wall -O2 -march=armv4 -mtune=strongarm
PROJ     =    telnet

all:     $(PROJ).o
         $(CC) $(CFLAGS) -o $(PROJ) $(PROJ).o
         $(CROSS)strip $(PROJ)

clean:
        rm -f $(PROJ) $(PROJ).o

```

- **3. Step:** Transfer the executable for the measurement Telnet server from your development system to the DNP/1110. You can use a FTP session for this file transfer.
- **4. Step:** Run the measurement Telnet server on your DNP/1110. Make sure to have superuser rights. Then enter

```
./msrtsvr6 5000
```

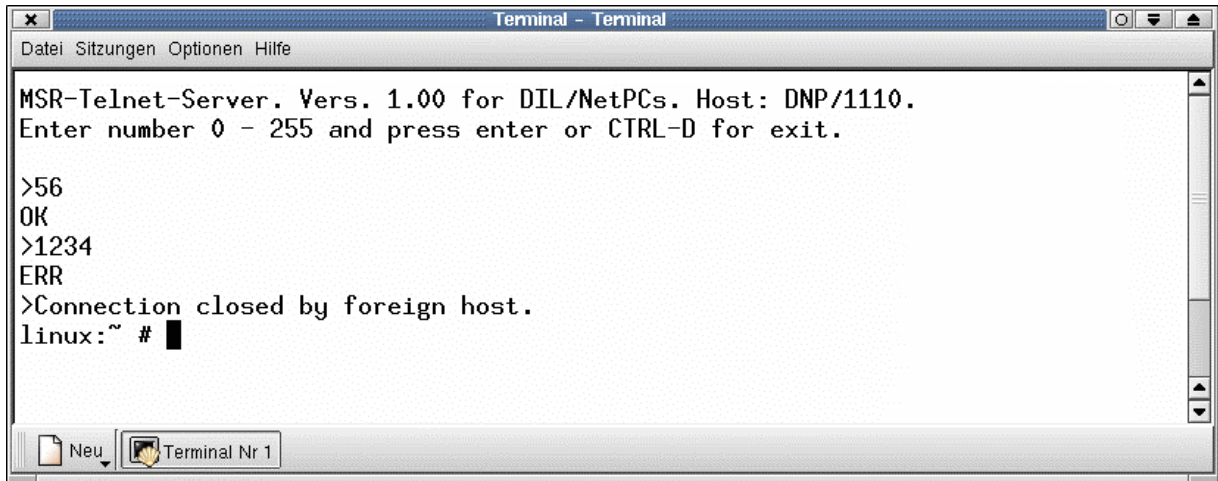
“5000” is a TCP port number. You can use any other unused port number, but not “23”. This is the standard port number for Telnet servers. Please note, that you have a other Telnet server currently running. This server is using the TCP port number 23. A good choice are port numbers over 1024.

- **5. Step:** Now you can test the measurement Telnet server. Run a new Telnet client program on your development system. Then enter

```
telnet 192.168.0.126 5000
```

within a text console window. After that a Telnet client for TCP port number 5000 starts and

builds a TCP connection. Within the new Telnet client window you see the sign-in message of the measurement Telnet server. Now you can enter valid numbers. Watch the LEDs of your DNP/EVA2 evaluation board.



```
Terminal - Terminal
Datei Sitzungen Optionen Hilfe

MSR-Telnet-Server. Vers. 1.00 for DIL/NetPCs. Host: DNP/1110.
Enter number 0 - 255 and press enter or CTRL-D for exit.

>56
OK
>1234
ERR
>Connection closed by foreign host.
linux:~ #
```

For disconnect the TCP connection between the Telnet server and the Telnet client, please use the key combination CTRL-D.