```
/*  Datei   :udpserver.c
    Version :0.1
    Datum   :16.08.2004 <D_Voelkel@web.de>

*/
////////////////////////////////////////////////////////////////////////////////
// Includes

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <asm/io.h>

#include "ssvhwa.h"
////////////////////////////////////////////////////////////////////////////////
//Defines

//taktregister
#define MCFBAR 0x40000000

//festlegen der daten und addressreister
#define PORTQS    (MCFBAR + 0x0010000D)  /* 8 bit */
#define DDRQS     (MCFBAR + 0x00100021)  /* 8 bit */
#define PORTQSP   (MCFBAR + 0x00100035)  /* 8 bit */
#define SETQS     (MCFBAR + 0x00100035)  /* 8 bit */
#define CLRQS     (MCFBAR + 0x00100049)  /* 8 bit */
#define PQSPAR    (MCFBAR + 0x00100059)  /* 8 bit */

// Zeitverzögerung
#define delay(Par)   usleep(Par*1000)       // Einfache Zeitverzögerung

#define UCHAR            unsigned char
#define USHORT           unsigned short
#define UINT             unsigned int

#define CTRL_BITS         8         // Bitanzahl MAX1270 Control Byte
#define RSLT_BITS     12        // Bitanzahl MAX1270 DigitalWert
#define MOD0        0x81        //1xxx 0001 - (0-5V) Betriebsmodus
#define MOD1        0x89        //1xxx 1001 - (0-10V) Betriebsmodus
#define MOD2        0x85        //1xxx 0101 - (-5V bis +5V) Betriebsmodus
#define MOD3        0x8D        //1xxx 1101 - (-10V bis +10V) Betriebsmodus
#define VOL_RANGE_5U      5    // MAX1270 - Eingangsspannungsbereich
#define VOL_RANGE_10U    10    // MAX1270 - Eingangsspannungsbereich
#define VOL_RANGE_5B     10    // MAX1270 - Eingangsspannungsbereich
#define VOL_RANGE_10B    20    // MAX1270 - Eingangsspannungsbereich
////////////////////////////////////////////////////////////////////////////////


// Function prototype

int  main      (int, char **);       // The one any only main function
void UdpServer (short);              // Wait for Client data...
USHORT ADCread (UCHAR mode, UCHAR channel);
  ////////////////////////////////////////////////////////////////////////////
// main

int main (int argc, char **argv)
{
//Deklaration der variablen
//=============================================================

   short   nPort;

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

   if (argc != 2) {
      printf ("\n  Server usage: udpserver PortNumber.");
      printf ("\n  Example.....: udpserver 5000<enter>\n\n");
      return (1);
```

```
    }
    nPort= atoi (argv[1]);

    // Do all the stuff a UDP server does...

    UdpServer (nPort);
    printf ("\n");
    return (1);
}


////////////////////////////////////////////////////////////////////////////
// UdpServer

void UdpServer (short nPort)
{
    int     nRet, theSocket,
            nLen= sizeof (struct sockaddr);
    char    szBuf[1024];

    struct sockaddr_in saServer, saClient;

    // Create a UDP/IP datagram socket.

    theSocket= socket (AF_INET, SOCK_DGRAM, 0);

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

    if (theSocket < 0)    {
       printf ("\n  Cant create Socket.\n");
       return;
    }

    // Fill in the address structure...

    saServer.sin_family= AF_INET;
    saServer.sin_addr.s_addr= INADDR_ANY;
    saServer.sin_port= htons (nPort);

    // Bind the name to the socket...

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

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

    if (nRet < 0) {
       printf ("\n  Error in bind function.\n");
       close (theSocket);
       return;
    }

    // Wait in a server loop ...

    while (1) {

     unsigned char szBuf1[1024];
     USHORT val1;
     float val2;
     USHORT val3;
     float val4;
     USHORT val5;
     float val6;
     USHORT val7;
     float val8;
     USHORT val9;
     float val10;


       // Wait for data from the client...

       printf ("\n  UDP Server (Linux) Vers. 1.00 is waiting on port \"%d\"", nPort);
       printf ("\n  ============================================================\n");
       memset (szBuf, 0, sizeof (szBuf));
       nRet= recvfrom (theSocket, szBuf, sizeof (szBuf), 0, &saClient, &nLen);
```

```
      // Show client addr and received data...

      printf ("\n  Data from addr %s received: %s", inet_ntoa (saClient.sin_addr), szBuf);


        // Aktivieren des SSVHWA treibers
//=============================================================================
    if (geteuid() != 0) {
    printf("No root access rights !\n");
    exit(1);
    }

    if (ssvhwa_open() < 0) {        // open ssvhwa driver
    perror("ssvhwa open");
    exit(-1);
    }


   ssvhwa_write8(PQSPAR, 0x00);



   //Starten der abfrage und der darstellung
        printf ("\n\n  MAX1270 per SPI am DIL/NetPC. Vers. 2.03");
        printf ("\n  ======================================");

    val1 = ADCread(MOD1,0);
    val2 = (float) val1 * VOL_RANGE_10U  / 0xfff;// berechnung Wert
        printf ("\n  Ch0= %2.3f V", val2);//ausgeben des wertes

    val3 = ADCread(MOD0,1);
    val4 = (float) val3 * VOL_RANGE_5U  / 0xfff;// berechnung Wert
        printf ("  Ch1= %2.3f V", val4);//ausgeben des wertes

    val5 = ADCread(MOD0,2);
    val6 = (float) val5 * VOL_RANGE_5U  / 0xfff;// berechnung Wert
        printf ("  Ch2= %2.3f V", val6);//ausgeben des wertes

    val7 = ADCread(MOD0,3);
    val8 = (float) val7 * VOL_RANGE_5U  / 0xfff;// berechnung Wert
        printf ("  Ch3= %2.3f V", val8);//ausgeben des wertes

    val9 = ADCread(MOD0,4);
    val10 = (float) val9 * VOL_RANGE_5U  / 0xfff;// berechnung Wert
        printf ("  Ch4= %2.3f V",val10);//ausgeben des wertes

    delay (1000 / 5);




    sprintf(szBuf1, "%i", val1);
      //strcpy (szBuf1, "hallo ich sehe dich");
      printf ("\n  Send some data back to %s : %s\n", inet_ntoa (saClient.sin_addr),
    szBuf1);
      sendto (theSocket, szBuf1, strlen (szBuf1), 0, &saClient, nLen);

      //Schliesen des treibers
      ssvhwa_close();
   }
}
//QSPI abfrage
//=============================================================================
    =========
USHORT ADCread (UCHAR mode, UCHAR channel)
{
   int i;
   USHORT result=0;
   UCHAR data;
```

```
    // Control Byte zusammenbauen

    channel <<= 4;
    mode |= channel;
    mode &= (channel | 0x8f);

    //CS auf High ziehen und  mit Low flanke komunikation starten
    ssvhwa_write8(DDRQS, 0x0E); //Ausgänge konfigurieren CS ausgang,
                //SCLK AUSGANG,DIN ausgang, DOUT eingang

    ssvhwa_write8(PORTQS, 0x08);//CS auf high setzen
    ssvhwa_write8(PORTQS, 0x00);//cs fallende flanke CS=aktive

    for (i = 1; i <= CTRL_BITS; i++)// BIT schleife zur übertragung
    {
       if (mode & 0x80)
       {
        ssvhwa_write8(PORTQS,0x02);//ausgangsregister auf HIGH
        ssvhwa_write8(PORTQS,0x06);//takt low SCLK
       }
       else
       {
       ssvhwa_write8(PORTQS,0x04);//takt high SCLK
       }
       ssvhwa_write8(PORTQS,0x00);//takt low SCLK
       mode <<= 1;
     }


       // 5 Taktimpulse ber SCLK fr den ADC erzeugen. Diese zeit dient als Ver-
       // z□erung, damit der MAX1270 einen Wert digitalisieren kann

       for (i= 1; i <= 5; i++)
       {
          ssvhwa_write8(PORTQS, 0x04); //takt HIGH sclk
          ssvhwa_write8(PORTQS,0x00); //takt low sclk
       }

       // Nun den digitalisierten Wert einlesen (12 Bits)
       //Bits werden in ein arry übertragen
       for (i=1;i <= RSLT_BITS; i++)
       {
       //BIT1
       //=============================================================================
          ssvhwa_write8(PORTQS, 0x04);// SCLK high takt, bit abfrage

      result<<=1;
          data = ssvhwa_read8(PORTQSP);// Datenbit einlesen
          ssvhwa_write8(PORTQS, 0x00);// SCLK LOW takt, bit übernahme
      if (data == 117) result++;// zusammenbauen des Berechnungsbit
       }

       return (result);

     //CS wieder auf HIG setzen und Komunikation beenden
     //=============================================================================
     ssvhwa_write8(PORTQS, 0x08);//Komunikation ende CS wieder auf High setzen


}
```