/*

--- CAEN SpA - Computing Systems Division ---

Name : CAENVMEDemoVme.c

Project : CaenVmeDemo

Description : Example program for V1718 & V2718 control.

Date : November 2004

Release : 1.0

Author : C.Landi

Date : August 2006

Release : 1.1

Author : NDA

Description : 64 bit porting (CAENVMElib rev >= 2.5)

---

*/

```
#include <stdlib.h>
#include <stdio.h>
#ifdef LINUX
	#include <inttypes.h>
	#include <stdint.h>
#else
	// From /usr/include/inttypes.h  kernel 2.6.15
	#define PRIX32		"X"
	#define PRIu32		"u"
	#define SCNx32		"x"
	#define SCNu32		"u"
#endif

#include <math.h>
#include <string.h>
#include <ctype.h>

#include <stdarg.h>
#include "CAENVMElib.h"
#include "console.h"



// ----------------------------------------------------------------------------

// ---------------------------------------
//   X,Y Position for Field display
// ---------------------------------------

#define X_ADDR			32
```

```
#define Y_ADDR                  9
#define X_BADDR                 32
#define Y_BADDR                 10
#define X_DSIZE                 32
#define Y_DSIZE         11
#define X_AM            32
#define Y_AM            12
#define X_BLTSIZE       32
#define Y_BLTSIZE       13
#define X_AUTOINC       32
#define Y_AUTOINC       14
#define X_NCYCLES       32
#define Y_NCYCLES       15
#define X_COMM                  1
#define Y_COMM                  21

// ---------------------------------------


typedef struct man_par
{
uint32_t        addr ;          // Address
uint32_t        data ;      // Data
ushort          level ;         // Interrupt level
uchar           irqstat; // IRQ status
ushort          am ;            // Addressing Mode
CVDataWidth dtsize ;            // Data Format
uint32_t        basaddr ;       // Base Address
uint32_t        blts ;          // Block size for blt (bytes)
```

```
ushort          ncyc ;          // Number of cycles
ushort          autoinc ;       // Auto increment address
uint32_t        *buff ;     // Mmemory buffer for blt
} man_par_t ;


// --------------------------------------
//   Prototype Functions Declaration
// --------------------------------------

void CaenVmeIrqCheck(int32_t, man_par_t *) ;
void CaenVmeWriteBlt(int32_t, man_par_t *) ;
void CaenVmeReadBlt(int32_t, man_par_t *) ;
void CaenVmeWrite(int32_t, man_par_t *) ;
void CaenVmeRead(int32_t, man_par_t *) ;
void ViewReadBltData(man_par_t *)  ;
void CaenVmeManual(int32_t, short) ;
void MarcoVmeWrite(int32_t, man_par_t *, uint32_t) ;
void MarcoVmeRead(int32_t, man_par_t *, uint32_t);

// ------------------------------------------------------------------------


/*
        ------------------------------------------------------------------------

        Name            :       CaenVmeIrqCheck
        Description :   Interrupt Check
```

```
        Input           :       BHandle             ->      Board Id (V1718,V2718)
                                man                 ->      pointer to 'man' structure
        Output          :       None
        Release         :       1.0

        ------------------------------------------------------------------------
*/

void CaenVmeIrqCheck(int32_t BHandle, man_par_t *man)
{
CVErrorCodes    ret ;

con_printf(" CHECK IRQ\n");

CAENVME_IRQCheck(BHandle,&man->irqstat);            // Check IRQ Status

con_printf(" IRQ status: %02X\n",man->irqstat);
con_printf(" Interrupt Level to Acknowledge (0 to exit) : ");  // Get the int level
con_scanf("%x",&man->level);

if (man->level == 0)
        {
   clear_line(Y_COMM);
   clear_line(Y_COMM+1);
   clear_line(Y_COMM+2);
        return ;
   }
```

```
con_printf_xy(X_COMM,Y_COMM+2," Waiting for interrupt ... Press any key to stop.");

while (!(man->irqstat & (1<<(man->level-1))) && !con_kbhit())   // Check Int loop
          CAENVME_IRQCheck(BHandle,&man->irqstat);

gotoxy(X_COMM,Y_COMM) ;

if(man->irqstat & (1<<(man->level-1)))
 {
   CVIRQLevels level;
   switch (man->level) {
        case 1:
             level = cvIRQ1;
             break;
        case 2:
             level = cvIRQ2;
             break;
        case 3:
             level = cvIRQ3;
             break;
        case 4:
             level = cvIRQ4;
             break;
        case 5:
             level = cvIRQ5;
             break;
        case 6:
             level = cvIRQ6;
             break;
```

```
        case 7:
            level = cvIRQ7;
            break;
        default:
            level = cvIRQ1;
            break;
    }
    ret = CAENVME_IACKCycle(BHandle,level,&man->data,man->dtsize);

        switch (ret)
                {
                case cvSuccess   : con_printf(" Cycle completed normally\n");

                                                con_printf(" Int. Ack. on IRQ%d: StatusID = %0X",man-
>level,man->data);
                                                break ;
                case cvBusError  : con_printf(" Bus Error !!!");
                                                break ;
                case cvCommError : con_printf(" Communication Error !!!");
                                                        break ;
                default         : con_printf(" Unknown Error !!!");
                                                break ;


      }
        }
else
        con_printf(" Interrupt request IRQ%d not active",man->level);
```

```
}



/*
        ---------------------------------------------------------------------------

        Name            :       CaenVmeWriteBlt
        Description :   Vme Block Transfer Write

        Input           :       BHandle                 ->      Board Id (V1718,V2718)
                                man                     ->      pointer to 'man' structure
        Output          :       None
        Release         :       1.0

        ---------------------------------------------------------------------------
*/

void CaenVmeWriteBlt(int32_t BHandle, man_par_t *man)
{
int                             nb;
uint32_t                        i,incr ;
CVAddressModifier       am;
CVErrorCodes            ret,old_ret=cvSuccess;



if(man->am == cvA16_U)
        {
   con_printf(" Can't execute a A16 BLT Write Cycle");
```

```
        return ;
        }
if(man->am == cvCR_CSR)
        {
   con_printf(" Can't execute a CR/CSR BLT Write Cycle");
        return ;
        }

con_printf_xy(X_COMM,Y_COMM+2," First Data [hex] : ") ;
con_scanf("%x",&man->data);                                    // Get data to write
clear_line(Y_COMM+2) ;
con_printf_xy(X_COMM,Y_COMM+2," Data Increment [hex] : ") ; // Get increment for data
con_scanf("%x",&incr);
for(i=0; i<(man->blts/4); i++)                                 // Fill the data buffer
        man->buff[i] = man->data+incr*i;

if (man->dtsize == cvD64)
        {
        if (man->am == cvA24_U_DATA)
                am = cvA24_U_MBLT;
        else
                am = cvA32_U_MBLT;
        }
else
        {
        if (man->am == cvA24_U_DATA)
                am = cvA24_U_BLT;
        else
                am = cvA32_U_BLT;
```

```
        }

if(man->ncyc == 0)                                  // Infinite Loop
        con_printf_xy(X_COMM,Y_COMM+2," Running ...    Press any key to stop.");

for (i=0; ((man->ncyc==0) || (i<man->ncyc)) && !con_kbhit(); i++)
        {

        ret = CAENVME_BLTWriteCycle(BHandle,man->addr,(char *)man->buff,man->blts,am,man-
>dtsize,&nb);

        if((i==0) || (ret != old_ret))
                {
                gotoxy(X_COMM,Y_COMM) ;

                switch (ret)
                        {
                        case cvSuccess   : con_printf(" Cycle(s) completed normally\n");
                                                con_printf(" Written %u bytes",nb);
                                                break ;
                        case cvBusError  : con_printf(" Bus Error !!!\n");
                                                con_printf(" Written %u bytes",nb);
                                                break ;
                        case cvCommError : con_printf(" Communication Error !!!");
                                                break ;
                        default         : con_printf(" Unknown Error !!!");
                                                break ;
                        }
                }
```

```
        old_ret = ret;

    }

if(man->ncyc == 0)
        clear_line(Y_COMM+2);
}



/*
        ---------------------------------------------------------------------------

        Name            :       CaenVmeReadBlt
        Description :   Vme Block Transfer Read

        Input           :       BHandle                 ->      Board Id (V1718,V2718)
                                man                     ->      pointer to 'man' structure
        Output          :       None
        Release         :       1.0

        ---------------------------------------------------------------------------
*/

void CaenVmeReadBlt(int32_t BHandle, man_par_t *man)
{
int                             nb;
uint32_t                        i,j ;
CVAddressModifier       am;
```

```
CVErrorCodes            ret,old_ret=cvSuccess;



if(man->am == cvA16_U)
        {
   con_printf(" Can't execute a A16 BLT Read Cycle");
        return ;
        }
if(man->am == cvCR_CSR)
        {
   con_printf(" Can't execute a CR/CSR BLT Read Cycle");
        return ;
        }


if (man->dtsize == cvD64)
        {
        if (man->am == cvA24_U_DATA)
                am = cvA24_U_MBLT;
        else
                am = cvA32_U_MBLT;
        }
else
        {
        if (man->am == cvA24_U_DATA)
                am = cvA24_U_BLT;
        else
                am = cvA32_U_BLT;
        }
```

```
if(man->ncyc == 0)                                  // Infinite Loop
        con_printf_xy(X_COMM,Y_COMM+2," Running ...    Press any key to stop.");

for (i=0; ((man->ncyc==0) || (i<man->ncyc)) && !con_kbhit(); i++)
        {
        for (j=0;j<(man->blts/4);j++)
                man->buff[j]=0;

        ret = CAENVME_BLTReadCycle(BHandle,man->addr,(char *)man->buff,man->blts,am,man->dtsize,&nb);

        if((i==0) || (ret != old_ret))
                {
                gotoxy(X_COMM,Y_COMM) ;

                switch (ret)
                        {
                        case cvSuccess   : con_printf(" Cycle(s) completed normally\n");
                                                con_printf(" Read b% bytes\n");
                                                break ;
                        case cvBusError  : con_printf(" Bus Error !!!\n");
                                                con_printf(" Read b% bytes\n");
                                                break ;
                        case cvCommError : con_printf(" Communication Error !!!");
                                                break ;
                        default          : con_printf(" Unknown Error !!!");
                                                break ;
                        }
                }
```

```
        old_ret = ret;

    }

if(man->ncyc == 0)
        clear_line(Y_COMM+2);
}



/*
        ---------------------------------------------------------------------------

        Name            :       CaenVmeWrite
        Description :   Vme Write access

        Input           :       BHandle                 ->      Board Id (V1718,V2718)
                                man                     ->      pointer to 'man' structure
        Output          :       None
        Release         :       1.0

        ---------------------------------------------------------------------------
*/

void CaenVmeWrite(int32_t BHandle, man_par_t *man)
{
uint32_t                i ;
CVErrorCodes    ret,old_ret=cvSuccess;
```

```
if(man->dtsize == cvD64)
        {
   con_printf(" Can't execute a D64 Write Cycle");
        return ;
        }

con_printf_xy(X_COMM,Y_COMM+1," Write Data [hex] : ") ;
con_scanf("%x",&man->data) ;

if(man->ncyc == 0)                                  // Infinite Loop
        con_printf_xy(X_COMM,Y_COMM+2," Running ...    Press any key to stop.");


for (i=0; ((man->ncyc==0) || (i<man->ncyc)) && !con_kbhit(); i++)
        {

        ret = CAENVME_WriteCycle(BHandle,man->addr,&man->data,man->am,man->dtsize);

        if((i==0) || (ret != old_ret))
                {
                gotoxy(X_COMM,Y_COMM) ;

                switch (ret)
                        {
                        case cvSuccess   : con_printf(" Cycle(s) completed normally\n");
                                                break ;
                        case cvBusError  : con_printf(" Bus Error !!!");
                                                break ;
```

```
                        case cvCommError : con_printf(" Communication Error !!!");
                                                    break ;
                        default         : con_printf(" Unknown Error !!!");
                                                    break ;
                        }
                }

        old_ret = ret;

        if(man->autoinc)
                {
                man->addr += man->dtsize;                   // Increment address (+1 or +2 or +4)
                con_printf_xy(X_ADDR,Y_ADDR,"%08X]",man->addr);    // Update the screen
                }
   }

if(man->ncyc == 0)
        clear_line(Y_COMM+2);
}

////////////////////
////////////////////
////////////////////
/* MARCO FUNCTION: WRITE SOMETHING IN HEX TO A GIVEN ADDRESS */
////////////////////
////////////////////
////////////////////
void MarcoVmeWrite(int32_t BHandle, man_par_t *man, uint32_t marco_data)
{
```

```
uint32_t                        i ;
CVErrorCodes    ret,old_ret=cvSuccess;



if(man->dtsize == cvD64)
        {
   con_printf(" Can't execute a D64 Write Cycle");
        return ;
        }

//con_printf_xy(X_COMM,Y_COMM+1," Write Data [hex] : ") ;
//con_scanf("%x",&man->data) ;



if(man->ncyc == 0)                                      // Infinite Loop
        con_printf_xy(X_COMM,Y_COMM+2," Running ...    Press any key to stop.");



for (i=0; ((man->ncyc==0) || (i<man->ncyc)) && !con_kbhit(); i++)
        {

//      ret = CAENVME_WriteCycle(BHandle,man->addr,&man->data,man->am,man->dtsize);
        ret = CAENVME_WriteCycle(BHandle,man->addr,&marco_data,man->am,man->dtsize);

        if((i==0) || (ret != old_ret))
                {
                gotoxy(X_COMM,Y_COMM) ;

                switch (ret)
```

```
                        {
                        case cvSuccess   : con_printf(" Cycle(s) completed normally\n");
                                                        break ;
                        case cvBusError  : con_printf(" Bus Error !!!");
                                                        break ;
                        case cvCommError : con_printf(" Communication Error !!!");
                                                        break ;
                        default          : con_printf(" Unknown Error !!!");
                                                        break ;
                        }
                }

        old_ret = ret;

        if(man->autoinc)
                {
                man->addr += man->dtsize;                   // Increment address (+1 or +2 or +4)
                con_printf_xy(X_ADDR,Y_ADDR,"%08X]",man->addr);    // Update the screen
                }
   }

if(man->ncyc == 0)
        clear_line(Y_COMM+2);
}


/*
        ---------------------------------------------------------------------------
```

```
        Name            :       CaenVmeRead
        Description :   Vme Read access

        Input           :       BHandle             ->      Board Id (V1718,V2718)
                                man                 ->      pointer to 'man' structure
        Output          :       None
        Release         :       1.0

        ------------------------------------------------------------------------
*/

void CaenVmeRead(int32_t BHandle, man_par_t *man)
{
uint32_t                i,old_data=0 ;
CVErrorCodes    ret,old_ret=cvSuccess;



if(man->dtsize == cvD64)
        {
   con_printf(" Can't execute a D64 Read Cycle");
        return ;
        }

if(man->ncyc == 0)                              // Infinite Loop
        con_printf_xy(X_COMM,Y_COMM+2," Running ...    Press any key to stop.");

for (i=0; ((man->ncyc==0) || (i<man->ncyc)) && !con_kbhit(); i++)
        {
```

```
        ret = CAENVME_ReadCycle(BHandle,man->addr,&man->data,man->am,man->dtsize);
    // ret = CAENVME_ReadCycle(BHandle,man->addr,&marco_data,man->am,man->dtsize);
        if((i==0) || (ret != old_ret))
                {
                gotoxy(X_COMM,Y_COMM) ;

                switch (ret)
                        {
                        case cvSuccess   : con_printf(" Cycle(s) completed normally\n");
                                                        if((i==0) || (old_data != man->data))
                    {
                    if( man->dtsize == cvD32 )
                                                                con_printf(" Current Value :
%08d\n",man->data);
                    //voltage uses this if statement
                    if( man->dtsize == cvD16 )
                                                                con_printf(" Current Value :
%04d\n",man->data & 0xffff);
                    if( man->dtsize == cvD8 )
                                                                con_printf(" Current Value :
%02d\n",man->data & 0xff);
                    }
                                                        break ;
                        case cvBusError  : con_printf(" Bus Error !!!");
                                                        break ;
                        case cvCommError : con_printf(" Communication Error !!!");
                                                        break ;
                        default          : con_printf(" Unknown Error !!!");
                                                        break ;
                        }
```

```
                }

        old_data = man->data;
        old_ret = ret;

        if(man->autoinc)
                {
                man->addr += man->dtsize;                   // Increment address (+1 or +2 or +4)
                con_printf_xy(X_ADDR,Y_ADDR,"%08X]",man->addr);    // Update the screen
                }
   }

if(man->ncyc == 0)
        clear_line(Y_COMM+2);
}




/*
        ---------------------------------------------------------------------------

        Name            :       ViewReadBltData
        Description :   Display the content of 'man->buff' buffer

        Input           :       man     ->      pointer to 'man' structure
        Output          :       None
        Release         :       1.0
```

```
        ------------------------------------------------------------------------
*/

void ViewReadBltData(man_par_t *man)
{
ushort          i,j,line, page=0, gotow, dtsize ;
uint32_t        ndata;
uint32_t        *d32;
ushort          *d16;
uchar           *d8;
char            key = 0;
char            msg[80];
FILE            *fsave;

d32 = man->buff ;
d16 = (ushort *)man->buff ;
d8  = (uchar *)man->buff ;

dtsize = man->dtsize ;

while( key != 'Q' )                                 // Loop. Exit if 'Q' is pressed
        {
   ndata = man->blts / dtsize;

        clrscr() ;
        con_printf("\n VIEW BUFFER\n\n") ;
        con_printf(" Num.    Addr     Hex        Dec \n\n");

                                                                    // Write a page
```

```
for( line=0, i=page * 16; (line<16) && (i<ndata); line++, i++)
                {
   if( dtsize == cvD32 || dtsize == cvD64)
                        con_printf(" %05u   %04X    %08X    %-10d \n",i,i*4,d32[i],d32[i]);

                if( dtsize == cvD16)
                        con_printf(" %05u   %04X    %04X       %-6d \n",i,i*2,d16[i],d16[i]);

                if( dtsize == cvD8)
      con_printf(" %05u   %04X    %02X         %-4d \n",i,i,d8[i],d8[i]);
   }

                        // Print the line menu
con_printf("\n[Q] Quit  [D] Data_size  [S] Save  [G] Goto");
if( page != 0 )
                con_printf("  [P] Previous");
if( i != ndata )
   con_printf("  [N] Next");

        key=toupper(con_getch()) ;          // Wait for command

        clear_line(22);

        switch (key)
                {
                case 'N' :          if(i<ndata)          // Next page
                                                page++;
                                        break ;
                case 'P' :          if(page>0)           // Previous page
```

```
                    page--;
                break ;
case 'D' :      dtsize = dtsize * 2;    // Change data size (8,16,32)
                if(dtsize > 4)
                    dtsize = 1;
                page = 0;
                break ;
case 'G' :      con_printf("Insert data number (dec) : ") ; // Go to data
                con_scanf("%d",(ushort *)&gotow) ;

                if(gotow>ndata)
                    {
                    clear_line(22) ;
                  con_printf(" Invalid data number ");
                    }
                else
                    page=gotow/16;
                break ;

case 'S' :      clear_line(23) ;
                con_printf_xy(1,23," File Name : ") ; // Save buffer to file
                if (con_scanf("%s",msg) == EOF)
                    break ;

                if((fsave=fopen(msg,"w")) == NULL)
                    {
                    clear_line(23) ;
                    con_printf_xy(1,23," Can't open file ");
                    }
```

```
                                        else
                                                {
                                                for(j=0;j<ndata;j++)
                {
                if( dtsize == cvD32 || dtsize == cvD64)
                   fprintf(fsave,"%05u\t%08X\t%-10d\n",j,d32[j],d32[j]);
                if( dtsize == cvD16)
                   fprintf(fsave,"%05u\t%04X\t%-6d\n",j,d16[j],d16[j]);
                if( dtsize == cvD8)
                   fprintf(fsave,"%05u\t%02X\t%-4d\n",j,d8[j],d8[j]);
                }

                                                fclose(fsave);
                                                }
                                        break ;
                 default   :  break ;

                 }
   }
}


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//MarcoVmeread//////////////////////////////////////////////
//by will lash
////////////////////////////////////////////////////////////////

void MarcoVmeRead(int32_t BHandle, man_par_t *man, uint32_t marco_data)
```

```
{
uint32_t                    i,old_data=0 ;
CVErrorCodes    ret,old_ret=cvSuccess;



if(man->dtsize == cvD64)
        {
   con_printf(" Can't execute a D64 Read Cycle");
        return ;
        }

if(man->ncyc == 0)                                      // Infinite Loop
        con_printf_xy(X_COMM,Y_COMM+2," Running ...    Press any key to stop.");

for (i=0; ((man->ncyc==0) || (i<man->ncyc)) && !con_kbhit(); i++)
        {

        ret = CAENVME_ReadCycle(BHandle,man->addr,&man->data,man->am,man->dtsize);
      //ret = CAENVME_ReadCycle(BHandle,man->addr,&marco_data,man->am,man->dtsize);
        if((i==0) || (ret != old_ret))
                {
                gotoxy(X_COMM,Y_COMM) ;

                switch (ret)
                        {
                        case cvSuccess   : con_printf(" Cycle(s) completed normally\n");
                                                        if((i==0) || (old_data != man->data))
                        {
                        if( man->dtsize == cvD32 )
```

```
                                                con_printf(" Current value :
%08d",man->data);
                        if( man->dtsize == cvD16 )
                                                con_printf(" Current value :
%04d",man->data & 0xffff);
                        if( man->dtsize == cvD8 )
                                                con_printf(" Current value :
%02d",man->data & 0xff);
                        }
                                                        break ;
                        case cvBusError  : con_printf(" Bus Error !!!");
                                                        break ;
                        case cvCommError : con_printf(" Communication Error !!!");
                                                        break ;
                        default         : con_printf(" Unknown Error !!!");
                                                        break ;
                        }
                }

        old_data = man->data;
        old_ret = ret;

        if(man->autoinc)
                {
                man->addr += man->dtsize;                   // Increment address (+1 or +2 or +4)
                con_printf_xy(X_ADDR,Y_ADDR,"%08X]",man->addr);    // Update the screen
                }
  }

if(man->ncyc == 0)
```

```
	clear_line(Y_COMM+2);
}

/*
	---------------------------------------------------------------------------

	Name		:	CaenVmeManual
	Description :	V1718 & v2718 manual controller

	Input		:	BHandle		->	Board Id (V1718,V2718)
				first_call	->	flag for default settings
	Output		:	None
	Release		:	1.0

	---------------------------------------------------------------------------
*/

void CaenVmeManual(int32_t BHandle, short first_call)
{

static  man_par_t	man	;
char			key,dis_main_menu ;
uint32_t		value ;

uint32_t channel_base_address;
int ii;
float val;
```

```
if (first_call)
        {                                                       // Default Value
        man.addr = 0;
        man.level = 1 ;
        man.am = cvA32_U_DATA ;
        man.dtsize = cvD16 ;
        man.basaddr = 0x32100000;//839909376 ;
        man.blts = 256 ;
        man.ncyc = 1 ;
        man.autoinc = 0 ;

        // Allocate 32K for the software buffer containing data for blt

   man.buff = (uint32_t *)malloc(16*1024*1024);
   if (man.buff == NULL)
                {
                con_printf(" !!! Error. Can't allocate memory for BLT buffer. ");
                exit(-1);
                }
        }

for (;;)
        {
        dis_main_menu = 0 ;

        clrscr() ;

        dis_main_menu = 0 ;
```

```
con_printf("\n     CAEN VME Manual Controller \n\n") ;

con_printf(" R - READ\n") ;
con_printf(" W - WRITE\n") ;
con_printf(" B - BLOCK TRANSFER READ\n") ;
con_printf(" T - BLOCK TRANSFER WRITE\n") ;
con_printf(" I - CHECK INTERRUPT\n") ;
con_printf(" 1 - ADDRESS                [%08"PRIX32"]\n",man.addr) ;
con_printf(" 2 - BASE ADDRESS           [%08"PRIX32"]\n",man.basaddr) ;
con_printf(" 3 - DATA FORMAT            [") ;
if (man.dtsize == cvD8)
            con_printf("D8]\n") ;
if (man.dtsize == cvD16)
            con_printf("D16]\n") ;
if (man.dtsize == cvD32)
            con_printf("D32]\n") ;
if (man.dtsize == cvD64)
            con_printf("D64]\n") ;

con_printf(" 4 - ADDRESSING MODE        [") ;
if (man.am == cvA16_U)
            con_printf("A16]\n") ;
if (man.am == cvA24_U_DATA)
            con_printf("A124]\n") ;
if (man.am == cvA32_U_DATA)
            con_printf("A32]\n") ;
if (man.am == cvCR_CSR)
            con_printf("CR/CSR]\n") ;
```

```
con_printf(" 5 - BLOCK TRANSFER SIZE      [%"PRIu32"]\n",man.blts) ;
con_printf(" 6 - AUTO INCREMENT ADDRESS   [") ;
if (man.autoinc)
        con_printf("ON] \n") ;
else
        con_printf("OFF]\n") ;

con_printf(" 7 - NUMBER OF CYCLES        [") ;
if (man.ncyc)
        con_printf("%d]\n",man.ncyc) ;
else
        con_printf("Infinite\n") ;

con_printf(" 8 - VIEW BLT DATA\n") ;
con_printf(" F - FRONT PANEL I/O\n") ;
con_printf(" Q - QUIT MANUAL CONTROLLER\n") ;

con_printf("\n 0 - Interactive menu\n");



do
        {
        gotoxy(X_COMM,Y_COMM) ;

        key = toupper(con_getch()) ;

        clear_line(Y_COMM) ;
        clear_line(Y_COMM+1) ;
        clear_line(Y_COMM+2) ;
```

```
                gotoxy(X_COMM,Y_COMM) ;

                switch (key)
                        {
                        case  'R' :          CaenVmeRead(BHandle, &man) ;
                                                break ;

                        case  'W' : CaenVmeWrite(BHandle, &man) ;
                                                break ;

                        case  'B' : CaenVmeReadBlt(BHandle, &man) ;
                                                break ;

                        case  'T' : CaenVmeWriteBlt(BHandle, &man) ;
                                                break ;

                        case  'I' : CaenVmeIrqCheck(BHandle, &man) ;
                                                break ;

                        case  '1' : con_printf(" Please enter new Address : ") ;
if (con_scanf("%"SCNx32,&value) != EOF)
                                                        man.addr = man.basaddr + value ;

        con_printf_xy(X_ADDR,Y_ADDR,"%08"PRIX32,man.addr) ;
                                                clear_line(Y_COMM) ;

                                                break ;
```

```
/*
if (man.addr = 000000000);
{man.addr = 321000090;};
break;
if (man.addr = 111111111);
{man.addr = 321000110;};
break;
if (man.addr = 222222222);
{man.addr = 321000190;};
break;
if (man.addr = 333333333);
{man.addr = 321000210;};
break;
if (man.addr = 444444444);
{man.addr = 321000290;};
break;
if (man.addr = 555555555);
{man.addr = 321000310;};
break;
*/


                    case  '2' :          con_printf(" Please enter new Base Address : ") ;

                                                        if (con_scanf("%"SCNx32,&value) != EOF)
                                                                {
                    man.addr -= man.basaddr ;
                                                                man.basaddr = value ;
                                                                man.addr += man.basaddr ;
```

```
                                        }


con_printf_xy(X_BADDR,Y_BADDR,"%08"PRIX32,man.basaddr) ;

con_printf_xy(X_ADDR,Y_ADDR,"%08"PRIX32,man.addr) ;

                                        clear_line(Y_COMM) ;

                                        break ;

                case  '3' : gotoxy(X_DSIZE,Y_DSIZE) ;

                                        switch(man.dtsize)
                                                {
                                                case cvD8  : man.dtsize = cvD16 ;
                                                                        con_printf("D16]\n") ;
                                                                        break ;
                                                case cvD16 : man.dtsize = cvD32 ;
                                                                        con_printf("D32]\n") ;
                                                                        break ;
                                                case cvD32 : man.dtsize = cvD64 ;
                                                                        con_printf("D64]\n") ;
                                                                        break ;
                                                case cvD64 : man.dtsize = cvD8 ;
                                                                        con_printf("D8] \n") ;
                                                                        break ;

                                                case cvD16_swapped :
```

```
                                        case cvD32_swapped :
                                        case cvD64_swapped :
                                        default :
                                                        break;
                                        }

                                break ;

                case  '4' : gotoxy(X_AM,Y_AM) ;

                                switch(man.am)
                                        {
                                        case cvA16_U      : man.am = cvA24_U_DATA ;

        con_printf("A24]\n") ;
                                                                                        break ;
                                        case cvA24_U_DATA : man.am =
cvA32_U_DATA ;

        con_printf("A32]\n") ;
                                                                                        break ;
                                        case cvA32_U_DATA : man.am = cvCR_CSR ;

        con_printf("CR/SCR]\n") ;
                                                                                        break ;
                                        case cvCR_CSR     : man.am = cvA16_U ;

        con_printf("A16]   \n") ;
                                                                                        break ;
                                        }
```

```
                                        break ;

                        case  '5' : con_printf(" Please enter Block Transfer Size : ") ;

                                        if (con_scanf("%"SCNu32,&value) != EOF)
                                                man.blts = value ;
                                        con_printf_xy(X_BLTSIZE,Y_BLTSIZE,"%"PRIu32"]
\n",man.blts) ;

                                        clear_line(Y_COMM) ;

                                        break ;

                        case  '6' : gotoxy(X_AUTOINC,Y_AUTOINC) ;

                                        man.autoinc ^= 1 ;
                                        if (man.autoinc)
                                                con_printf("ON] \n") ;
                                        else
                                                con_printf("OFF]\n") ;

                                        break ;

                        case  '7' : con_printf(" Please enter Number of Cycles : ") ;

                                        if (con_scanf("%"SCNu32,&value) != EOF)
                                                man.ncyc = (ushort)value ;
                                        gotoxy(X_NCYCLES,Y_NCYCLES) ;
```

```
                                        if (man.ncyc)
                                                con_printf("%"PRIu32"]      \n",man.ncyc) ;
                                        else
                                                con_printf("Infinite]\n",man.ncyc) ;

                                        clear_line(Y_COMM) ;

                                        break ;

                        case  '8' :         ViewReadBltData(&man);
                                        dis_main_menu = 1;      // Display Main Menu
                                        break ;

                        case  'Q' : free(man.buff);                     // Release the memory buffer for BLT
                                        return ;

                        case '0' : con_printf(" Which channel do you want to work with? (0..5)\n");
con_printf("Enter desired Channel:");
if(con_scanf("%"SCNx32,&value) != EOF) {
        switch (value) {
                   case 0 : channel_base_address = 0x80;
                break;
         case 1 : channel_base_address = 0x100;
                break;
         case 2 : channel_base_address = 0x180;
                break;
         case 3 : channel_base_address = 0x200;
                break;
         case 4 : channel_base_address = 0x280;
```

```
                    break;
            case 5 : channel_base_address = 0x300;
                                break;
// make the other cases for each channel
                    default : break;
            }
}

con_printf(" What do you want to do with channel %d?\n",value);
con_printf("  -> 0 - OFF\n") ;
con_printf("  -> 1 - ON\n") ;
con_printf("  -> 2 - Change voltage\n") ;
con_printf("  -> 3 - Change current\n") ;
con_printf("Enter choice:");
// Read voltage set
// Read voltage monitor

//.... Write down all the other options

if(con_scanf("%"SCNx32,&value) != EOF) {
            switch (value) {
                    case 0 :
                                man.addr = man.basaddr + channel_base_address + 0x10;
                                con_printf_xy(X_ADDR,Y_ADDR,"%08"PRIX32,man.addr) ;
                                MarcoVmeWrite(BHandle, &man, 0);
                    //con_printf("Channel %d Off\n");
                     // delay(1000);
                                break;
             case 1 :
```

```
                man.addr = man.basaddr + channel_base_address + 0x10;
                        con_printf_xy(X_ADDR,Y_ADDR,"%08"PRIX32,man.addr) ;
                        MarcoVmeWrite(BHandle, &man, 1);
               // con_printf("Channel %d On\n");
               //  delay(1000);
                        break;
                   case 2 :
clear_line(Y_COMM);
                        man.addr = man.basaddr + channel_base_address + 0x00;
//                      con_printf_xy(X_ADDR,Y_ADDR,"%08"PRIX32,man.addr) ;
                  CaenVmeRead(BHandle, &man);
                        con_printf("Input voltage: ");
                        if(con_scanf("%f",&val) != EOF) {
                                value = (int)(val*10);
//                              con_printf("Typed: %d",value);
                                MarcoVmeWrite(BHandle, &man, value);
break;
          case 3 :
                        man.addr = man.basaddr + channel_base_address + 0x04;
//                      con_printf_xy(X_ADDR,Y_ADDR,"%08"PRIX32,man.addr) ;
                   CaenVmeRead(BHandle, &man);
                        con_printf("Input Current:\n ");
                        if(con_scanf("%f",&val) != EOF) {
                                value = (int)(val);
//                              con_printf("Typed: %d",value);
                                MarcoVmeWrite(BHandle, &man, value);
               }
                        }
                        break;
```

```
                    // ADD OTHER CASES!
                    default : break;
          }
}



/*        if(value == 1) {
                    man.addr = man.basaddr + 144;
                    con_printf_xy(X_ADDR,Y_ADDR,"%08"PRIX32,man.addr) ;
                    MarcoVmeWrite(BHandle, &man, 1);
          }
*/
for(ii=0; ii<10; ii++) clear_line(Y_COMM+ii) ;
break ;

                              default   : break ;
                              }


                    }
          while (!dis_main_menu) ;


          }                                                                                  // End 'for(;;)'
}
```