01 : Volume table of contents

02 : Product Identification

54 : General Software Design

55 : Software Module overview

02.1. Download New Cd-i Keyboard drivers for Cd-i 450 & 380

The CD-i 450, Goldstar 450 and CD-i FW380 type players all have the need for a special I/O splitter cable to connect more than one CD-i (pointing) device, thus resulting in 2 different ports (1 & 2).

Port 1 is internally connected to IKAT 1 or the UART of the 68070 (depending on the particular device attached).

Port 2 is internally connected to IKAT2 and is dedicated to wired pointing devices.

On the 450 and 380 players, Port 2 is NOT capable of handling CD-i Keyboard input according to the FFGB spec. Only a CD-i keyboard connected to Port 1 will be recognized..
This means that is is impossible on the 450 and 380 players to run applications that use both a Modem device and a CD-i Keyboard device. e.g. The CD-Online internet application.

To enable the use of a Keyboard on Port 2 of a 450/380 player, A New CD-i Keyboard specification was written. (Keyboard specification : V0.92 , Jos Schepers, PIA Hasselt, 6/27/96).
This new specification defines a new mode ('T' mode), where the Keyboard will announce itself as a Graphic Tablet, and will send its data in Graphic Tablet packet format. This mode will be for exclusive use on the 450/380 players.
The new CD-Online keyboard, developed by Technico Taiwan, will support both modes through an external switch (switch to 'K' mode or 'T' mode)..
In order to support a 'T'mode keyboard on the 450/380 players, the application has to replace the existing keyboard and pointing device2 driver/descriptor with new drivers, supporting the New keyboard specification.
These new drivers were supplied by PIA Hasselt.

This document describes a C-coded(K&R) CD-i test application with following functionality :

- unique check to verify whether the Player is of type CD-i 450, GOLDSTAR 450, CD-i FW380
- verify whether a CD-i Keyboard in 'T' mode is connected.
- download new drivers in case of 450/380 players.
- initialize paths for the new drivers and implement a sample signal handler for keycodes and cursor control.

For use in the CD-Online application I suggest following method :

- copy following functions : int Cdi450_380()
  int CSD_OK()
- copy all #defines / prototypes for these functions.
- use the part of the main() routine which calls the latter functions and does the appropriate detaching / replacement of drivers.
- open the paths to /kb1 and /pt2 in the same order as the main() routine does.

54.2. Source Description / Remarks

see : Appendix A : Makefile
see : Appendix B : new_drivers_subr.c - Documented Source

Remarks :

* The CDi450_380() call will uniquely discriminate the CD-i 450 / CD-i 380 players from the rest of the CD-i player park. The call will return following values :

- 'CDI_450' defined value : player is of type CD-i 450, GOLDSTAR 450.
- 'CDI_380'defined value : player is of type CD-i FW380.
- 'FALSE' defined value : any other player.

This function was seperately tested using a mastered disc on all players of the PIMC test departement. Test results can be obtained at PIMC (see 4. Resources/Electronic information).

* It is necessary to differentiate between the CD-i 450 and the CD-i 380 players because the new kb1driv keyboard driver is different for the 450 / 380 player.

In case of a CD-i 450 player the file 'mods_450' is loaded. 'mods_450' is a concatenation of following OS9 modules :

- pt2driv :	CRC=$4956A7	; new pointing device 2 driver for CD-i 450/380
- kb1driv :	CRC=$E61C3	; new 'T'mode keyboard driver for CD-i 450
- pt2 :	CRC=$C7624D	; new pointing device 2 descriptor for CD-i 450/380

In case of a CD-i 380 player the file 'mods_380' is loaded. 'mods_380' is a concatenation of following OS9 modules :

- pt2driv :	CRC=$4956A7	; new pointing device 2 driver for CD-i 450/380
- kb1driv :	CRC=$4066D	; new 'T'mode keyboard driver for CD-i 380
- pt2 :	CRC=$C7624D	; new pointing device 2 descriptor for CD-i 450/380

* After correctly loading the appropriate drivers, the new keyboard will be activated if the paths are opened/accessed in the correct order :

1. Open /kb1 path
2. Open /pt2 path
3. initialize keyboard, etc...
4. initialize cursor (/pt2) if F1-F8 cursor functionality should be supported.

54.3. Usage of 'new_drivers_subr.c' within the CD-Online application

For use in the CD-Online application I suggest following method :

- copy following functions : int Cdi450_380()
  int CSD_OK()
- copy all #defines / prototypes for these functions.
- use the part of the main() routine which calls the latter functions and does the appropriate detaching / replacement of drivers.
- open the paths to /kb1 and /pt2 in the same order as the main() routine does.

54.4. Resources/Electronic information

Test results, additional information and (electronic) copies of the driver files, makefile and source can be obtained by contacting PIMC :

Jan Matejka
Philips Interactive Media Centre (PIMC)
Maastrichterstraat 63
3500 Hasselt
BELGIUM

tel : +32 11 242 546
fax : +32 11 242 273
email : janm@pimc.be

APPENDIX A : Makefile

```
CC = cc68

# CDRTOS 1.0 directories
CDILIB  = /usr/local/os9/lib
CDIDEFS = /usr/local/os9/defs
CDRTOSLIB = /usr/local/os9/lib
CDRTOSDEFS = /usr/local/os9/defs

INFILE = new_drivers_subr

PROJ  = .
SRC   = $(PROJ)/src
OBJ   = $(PROJ)/obj
DEFS  = $(PROJ)/inc
LIB   = $(PROJ)/lib
LIBS  = -l=$(CDILIB)/clib.l \
      -l=$(CDILIB)/cio.l  \
      -l=$(CDILIB)/usr.l  \
      -l=$(LIB)/mvcdi.l   \
      -l=$(LIB)/macdi.l   \
      -l=$(LIB)/fmv.l     \
      -l=$(LIB)/mvcdin.l  \
      -l=$(CDRTOSLIB)/cdi.l  \
      -l=$(CDRTOSLIB)/sys.l



# Cflags
#-------
CFLAGS = -v=$(CDIDEFS) -v=$(CDRTOSDEFS) -v=$(DEFS) -v=$(PROJ)

RFILES = $(OBJ)/$(INFILE).r


$(PROJ)/$(INFILE)  :  $(RFILES)
        $(CC) -g -m=4 $(RFILES) $(LIBS) -f=$(PROJ)/$(INFILE)

# Building the object modules
#----------------------------
$(OBJ)/$(INFILE).r : $(SRC)/$(INFILE).c
        $(CC) $(CFLAGS) $(SRC)/$(INFILE).c -r=$(OBJ)
```

APPENDIX B : New_drivers_subr.c

```
/*************************************************************
*
* new_drivers_subr.c
*
*
* download and test CD-i 450/380 drivers for CD-Online Keyboard
* in 'T' mode.
*
*
* Jan Matejka, PIMC
* June 1996
*
*************************************************************/

#include <stdio.h>
#include <modes.h>
#include <signal.h>
#include <csd.h>
#include <cdfm.h>
#include <ucm.h>
#include <module.h>
#include <sysio.h>
#include <memory.h>
#include <errno.h>
#include <time.h>

#define PIMC_DEBUG    1

#if PIMC_DEBUG
#define DEBUG PIMC_DEBUG
#define DEBUG_ON(STATEMENT) {STATEMENT}
#define DEBUG_OFF(STATEMENT)
#else
#define DEBUG_ON(STATEMENT)
#define DEBUG_OFF(STATEMENT)
#endif

#define        SYSERR          (-1)
#define        TRUE        1
#define        FALSE       0

#define PT2_SIG        260

#define        KEYBOARD_SIG            300

#define        KB_MODE_WAIT            0x01
#define        KB_MODE_REMOVE          0x02

#define        KB_KEY_PRESSED          0x01
#define        KB_KEY_AUTO_REPEAT      0x02
```

```
#define         KB_KEY_RELEASED         0x04

#define         CONFIG_MODNAME          "config"
#define         PT2DRIV_MODNAME         "pt2driv"
#define         KB1DRIV_MODNAME         "kb1driv"
#define         CRC_KB1DRIV_450         (0x5Ef36C)
#define         CDI_450                 2
#define         CDI_380                 3
#define         NEW_KB1DRIV_450         "mods_450"/* "new_kb1driv_450, pt2driv, pt2" */
#define         NEW_KB1DRIV_380         "mods_380"/* "new_kb1driv_380, pt2driv, pt2" */


/*****************************************
*
* player and corresponding config module CRC.
* Information obtained from release notes. Some were
* not available, in which case CRC is marked ?.
*
* Jan Matejka,
* PIMC, may 96
*
*       FW380i, R1.1                    CRC config : $4B6FDD
*       FW380i, R1.3                    CRC config : $4B6FDD ?
*       GOLDSTAR 450, US, R1.3                  CRC config : $4B6FDD
*
*       GOLDSTAR 450, KOREA, R1.3       CRC config : $7466BF
*       450 HIGHSCREEN, R1.1            CRC config : $7466BF
*       450 R1.3                        CRC config : $7466BF
*       450 R1.4                        CRC config : $7466BF
*       450/11 R1.4                     CRC config : $7466BF
*       GOLDSTAR 450, UK R1.2                   CRC config : $7466BF
*       GOLDSTAR 450, US R1.2                   CRC config : $7466BF
*       GOLDSTAR 450, US R1.3                   CRC config : $7466BF
*
*       GOLDSTAR 450 R1.1               CRC config : $C010C9
*       450 R1.1                        CRC config : $C010C9
*
*****************************************/

/*      globals         */
extern int      errno;


int             video_path;
int             pt2_path;
int             kb1_path;
unsigned short  ntsc_flag;

int             tp;
int             screenpos;
unsigned short  textout_flag;

int             btnstate,ptx,pty;
```

```
short           inkey;
unsigned char   type;


/* proto's */
int             CDi450_380();
int             CSD_OK();

int             init_videopath();
void            detach_ptr2();
int             init_pt2path();
int             init_kb1path();
void            flush_keyboard_buffer();
void            init_cursor();
void            Wait_n_Ticks();
void            signal_handler();


/*****************************************************************************/

main()
{
int     i;
int     CDI_Type;

video_path = init_videopath();


CDI_Type = CDi450_380();                        /* returns type CDI_450, CDI_380 or FALSE */

if ( (CDI_Type==CDI_450)||(CDI_Type==CDI_380) )  /* test if the device is a CD-i 450 or CD-i 380 */
        {
        if (CSD_OK() )                          /* test if keyboard in 'T' mode is attached */
                {
                if( CDI_Type == CDI_450 )       /* test if the device is a CD-i 450 */
                        {
                        detach_ptr2();
                        modload(NEW_KB1DRIV_450, 1);
                        attach( "/pt2", READ_);
                        DEBUG_ON( printf("new pt2driv, kb1driv, pt2 loaded for CDI 450 player\n"); );
                        }
                if (CDI_Type == CDI_380 )       /* else it must be a FW380i */
                        {
                        detach_ptr2();
                        modload(NEW_KB1DRIV_380, 1);
                        attach( "/pt2", READ_);
                        DEBUG_ON( printf("new pt2driv, kb1driv, pt2 loaded for CDI 380 player\n"); );
                        }
                }
                else
                {
                if( CDI_Type == CDI_450 )       /* test if the device is a CD-i 450 */
                        {
                        DEBUG_ON( printf("no drivers loaded for CDI 450 player :\n"); );
```

```
                }
            if (CDI_Type == CDI_380 )               /* else it must be a FW380i */
                {
                DEBUG_ON( printf("no drivers loaded for CDI 380 player :\n"); );
                }
            DEBUG_ON( printf("keyboard in 'I' mode is not attached\n"); );
            }
        }

TEST:

kb1_path = init_kb1path();
pt2_path = init_pt2path();

kb_repeat(kb1_path,25,5);               /* default auto-repeat is on */
flush_keyboard_buffer(kb1_path);
init_cursor();                          /* show and position upper left */

intercept(signal_handler);
if (kb1_path) kb_ssig(kb1_path, KEYBOARD_SIG);
if (pt2_path) pt_ssig(pt2_path, PT2_SIG);

EXIT:
while(1);

close(video_path);
exit(0);
}

/*****************************************************************************
*
* int CDi450_380()
*
* This function will try to link to the CONFIG module of the CD-i player,
* and if present it will verify its CRC. From the CD-i player release notes
* information, the CONFIG module has proven to be a good discriminator to isolate
* whether the player is of type CD-i 450, 380, GOLDSTAR 450, or any other type
* apart from these 3. The player is of type CD-i 450, 380 or GOLDSTAR 450 if the
* CRC is one of these : 0x4B6FDD, 0x7466BF, 0xC010C9.
* To differentiate a CDi 450/GOLDSTAR 450 player from a CD-i FW380 player,
* The kb1driv module CRC is used. This CRC is 0x5ef36C for the CD-i 450 / GOLDSTAR
* 450 range of players.
*
* value returned : int
*               int = CDI_450           for CD-i 450 and GOLDSTAR 450 range of players
*               int = CDI_380           for CD-i FW380 range of players
*               int = FALSE             for any other player.
*
*****************************************************************************/

int CDi450_380()
{
unsigned int    crc;
mod_exec        *modptr;
```

```
int                 return_value;

if ( (int) (modptr=(mod_exec *)modlink(CONFIG_MODNAME, mktypelang(MT_PROGRAM,ML_OBJECT))) ==
SYSERR)
        {
        DEBUG_ON( printf("Unable to link to config module\n"); );
droid64

        return(FALSE);
        }

crc = *(unsigned int *)((char *)modptr + modptr->_mh._msize - 4);
DEBUG_ON ( printf("CRC of config module : %x\n", crc); );

munlink(modptr);

switch(crc)
        {
        case 0x4B6FDD:
        case 0x7466BF:
        case 0xC010C9:
                DEBUG_ON( printf("CRC of config module is one of 0x4B6FDD, 0x7466BF, 0xC010C9 : \n"); );
                DEBUG_ON( printf("Player is of type CD-i 450, Goldstar 450 or CD-i FW380\n"); );
                break;

        default:
                DEBUG_ON( printf("CRC of config module is NONE of 0x4B6FDD, 0x7466BF, 0xC010C9 : \n"); );
                DEBUG_ON( printf("Player is of OTHER type than CD-i 450, Goldstar 450 or CD-i FW380\n"); );
                return(FALSE);
                break;
        }

if ( (int) (modptr=(mod_exec *)modlink(KB1DRIV_MODNAME, mktypelang(MT_DEVDRVR,ML_OBJECT))) ==
SYSERR)
        {
        DEBUG_ON( printf("Unable to link to kb1driv module\n"); );
        return(FALSE);
        }
        else
        {
        crc = *(unsigned int *)((char *)modptr + modptr->_mh._msize - 4);
        DEBUG_ON ( printf("CRC of kb1driv module : %x\n", crc); );

        munlink(modptr);

        if (crc==CRC_KB1DRIV_450)
                {
                DEBUG_ON( printf("Player is definitely CD-i 450\n"); );
                return(CDI_450);
                }
                else
                {
                DEBUG_ON( printf("Player is definitely CD-i FW380\n"); );
                return(CDI_380);
```

```
                }
        }
}

/*************************************************************************
*
* int CSD_OK()
*
* This function will check whether a Keyboard in 'T' mode is attached.
* Only in this case the new drivers should be downloaded.
* A keyboard in 'T' mode will announce itself as an absolute pointing
* device, thus resulting in a CSD entry of type
*       5:/pt2:CL="a" or
*       5:/pt2:CL="b".
* Additionally it is checked whether the /pt2 absolute pointing device descriptor
* found is referencing correcty  to the pt2driv driver.
*
* value returned : int TRUE or FALSE
*       TRUE = 'T'mode keyboard has an entry in the CSD
*       FALSE = 'T'mode keyboard entry not found, or referencing to wrong driver
*
*************************************************************************/

int CSD_OK()
{
int             i=0;
char            *pt2_name;
char            *pt2_param;
mod_dev                 *pt2_modp;
char            *pt2driv_name;

while(1)
        {
        pt2_name=csd_devname(DT_PTR, ++i);              /* try to find /pt2 descriptor in next CSD entry */
        DEBUG_ON( printf("pt2_name : %s\n", pt2_name); );

        if (!(int)pt2_name) return(FALSE);          /* if not found return FALSE  */

        if (strcmp(pt2_name, "/pt2") == 0)
                {                                       /* yes, it is /pt2 */
                pt2_param = csd_devparam(pt2_name);
                DEBUG_ON( printf("pt2_param : %s\n", pt2_param); );

                if (    findstr(1, pt2_param, "CL=\"a\"") || /* check if  it's an absolute p. device */
                        findstr(1, pt2_param, "CL=\"b\"") )
                        {
                        pt2_modp=(mod_dev *)modlink((char *)pt2_name+1,
mktypelang(MT_DEVDESC,ML_ANY));
                        if ( (int)pt2_modp == SYSERR) return(FALSE);

                        DEBUG_ON( printf("pt2_modp : %x\n", pt2_modp); );
                        pt2driv_name = (char *)((int)pt2_modp + (int)pt2_modp->_mpdev);

                        /* check it the /pt2 descriptor is linked to the pt2driv driver */
```

```
                        if (!strcmp(pt2driv_name, PT2DRIV_MODNAME) )
                                {
                                munlink(pt2_modp);
                                return(TRUE);
                                }
                                else
                                {
                                munlink(pt2_modp);
                                return(FALSE);
                                }
                        }
                        else
                        {
                        munlink(pt2_modp);
                        return(FALSE);
                        }
                }
        }
}

/*****************************************************************************/

void    detach_ptr2()
{

struct devicetbl    *pt2devtbl;

pt2devtbl = attach( "/pt2", READ_);
while (pt2devtbl->V_usrs > 0)
        {
        detach(pt2devtbl);
        }
}

/*****************************************************************************/

int     init_pt2path()
{

if ((pt2_path = open("/pt2", S_IREAD)) == SYSERR)
        {
        DEBUG_ON( printf("Error on getting pt2 path\n"); );
        exit(3);
        }
return(pt2_path);
}

/*****************************************************************************/

int     init_kb1path()
{
int     kb1_path;

if ((kb1_path = open("/kb1", S_IREAD)) == SYSERR)
```

```
        {
        DEBUG_ON(printf("Error on opening keyboard path \n");)
        return(FALSE);
        }
        else
        {
        return(kb1_path);
        }
}

/*****************************************************************************/

int     init_kb1(kb1_path)
int     kb1_path;
{

if ((kb1_path = open("/kb1", S_IREAD)) == SYSERR)
        {
        DEBUG_ON(printf("Error on opening keyboard path \n");)
        return(FALSE);
        }
        else
        {
        kb_repeat(kb1_path,25,5);           /* default auto-repeat is on */
        flush_keyboard_buffer(kb1_path);
        }

return(kb1_path);
}

/*******************************************************************************/

void    flush_keyboard_buffer(kb1_path)
int     kb1_path;
{
short           inkey;
unsigned char   type;


do kb_read(kb1_path,KB_MODE_REMOVE,&inkey,&type);
        while (type);
Wait_n_Ticks(2);
do kb_read(kb1_path,KB_MODE_REMOVE,&inkey,&type);
        while (type);
}

/*****************************************************************************/

int     init_videopath()
{
char    *video_device_name;
char    *vidparam;
int     video_path;
```

```
if ((video_device_name=csd_devname(DT_VIDEO,1)) == 0)
    {
    DEBUG_ON(printf("Error on getting device name"));
        exit(3);
    }
if ((video_path=open(video_device_name,S_IREAD)) == -1)
        {
        DEBUG_ON(printf("cannot open vpath error\n"));
        exit( errno );
        }

if(vidparam = csd_devparam(video_device_name))
        {
        if(findstr(1,vidparam,"625"))
                {
                dc_setcmp(video_path, 0);
                ntsc_flag = 0;
        }
    else
                {
                ntsc_flag = 1;
                if (findstr(1, vidparam, "TV"))
                        {
                        dc_setcmp(video_path, 0);
                        }
                else
                        {
                        dc_setcmp(video_path, 1);
                        }
                }
        free(vidparam);
        }
free(video_device_name);
dc_intl(video_path,0);
return(video_path);
}

/*****************************************************************
* INIT CURSOR                                                    *
*****************************************************************
*                                                                *
* Initialize cursor settings                                     *
*                                                                *
*****************************************************************/
void init_cursor()
{
        gc_blnk(video_path, 0x00000030); /* cursor blink rate */
        gc_col (video_path, 0x80A0A0A0);        /* cursor color */
        gc_org(video_path,0,0);                 /* cursor origin -- upper left point */
        gc_pos(video_path,0,0);
#if 0
        pt_org(pt2_path,0,0);                   /* set pointer origin  */
        pt_pos(pt2_path,0,0);           /* position pointer to x,y  */
#endif
```

```
        gc_show(video_path);                    /* hide the cursor */
#if 0
        gc_hide(video_path);
#endif
}


/*****************************************************************************/

void Wait_n_Ticks(n)
int n;
{

unsigned int        tickcount;

tickcount=(_getsys(D_Ticks,4));
while( (_getsys(D_Ticks,4) - tickcount) < n);

}


/*****************************************************************
* SIGNAL_HANDLER                                                 *
*****************************************************************/

void signal_handler(signum)
register int        signum;
{

        switch (signum)
                {

        case    KEYBOARD_SIG:
                        kb_read(kb1_path,KB_MODE_REMOVE,&inkey,&type);
                        DEBUG_ON( printf("inkey : %d --- type : %d\n",inkey,type); );
                        kb_ssig(kb1_path, KEYBOARD_SIG);
                        break;

                case PT2_SIG:
                        pt_coord(pt2_path,&btnstate,&ptx,&pty);
                        gc_pos(video_path,ptx,pty);
                        DEBUG_ON( printf("pt2sig received - btnstate = %0d\n", btnstate&3); );
                pt_ssig(pt2_path,PT2_SIG);
                        break;


                case SIGQUIT:
                case SIGINT:
                        exit(0);
                        break;
                default:
                        DEBUG_ON(printf("\nSignal_Handler: Unknown signal [%d]", signum););
```

```
        }

}

/*****************************************************************************/
```