OS-9/68K
SOFTWARE SUPPORT MANUAL

CC143 SCF Driver pack

VERSION 1.0 March 1991

Documentation history

date	version	change / description
91/03/07	1.0	first release

OS-9/68XXX CC143 SCF SUPPORT

TABLE OF CONTENTS

CHAPTER 1

General Information

1.1 Introduction

The CC143 SCF Driver support package is designed to give programmers a hardware independant interface to the CC143 functions. The package consists of 4 drivers :
'scvid143' is the driver that controls the video part.
'scptr143' is the driver that controls the pointer(mouse) part.
'sckbd143' is the driver that controls the keyboard part.
'sccc143' is a standard SCF driver for the mouse port.

The drivers are implemented as a pseudo-SCF drivers. For the video and the pointer part, the standard Read and Write calls of the driver are not used, but every call is going through SetStat and Getstat calls. The package comes with a C-library which gives C programmers a comfortable interface to the drivers.

1.2 General information

1.3 Library Interface

An application which want to use the CC143 functions, first opens the device which has to be used . The devices are :
'/vid' for the video part.
'/ptr' for the pointer part.
'/kbd' for the keyboard part.
'/sm' for the SCF path on the mouse port.
As an example the video part this can be opened with the call :

```
                   vidpath = open("/vid",S_IREAD+S_IWRITE);
```

The 'vidpath' which is returned has to be used in every call to the CC143 library.

CHAPTER 2

High level C Functions

2.1 Introduction

These functions are the video functions from the vidlib.l library. this library has to be used in combination with the CC143 video driver.

2.2 Video functions

SYNOPSIS:

```
int _vd_scrsiz(path,width,height,pages)
int path; /* path number of video device */
int *width; /* pointer to width variable */
int *height; /* pointer to height variable */
int *pages; /* pointer to nr of pages variable */
```

DESCRIPTION: This function returns the size of the display in the variables pointed to by width and height. It also returns the maximum number of screen pages that can be used.

_vd_reqdmmem() request drawmap memory _vd_reqdmmem()

SYNOPSIS:

```
unsigned char *_vd_reqdmmem(path,size)
int path; /* path number of video device */
int size;
```

DESCRIPTION: This function requests drawmap memory from the CC143 video memory. The size which is passed is the size of the drawmap in bytes. It must match the size of one screen. _vd_reqdmmem() returns a pointer to the drawmap which can be used.
If an error occurs _vd_reqdmmem() returns -1 as its value and the appropriate error code in the global variable errno.

CAVEATS: The size can be obtained using the _vd_scrsiz() call

SYNOPSIS:

```
int _vd_retdmmem(path,drawmap)
int path; /* path number of video device */
unsigned char *drawmap;
```

DESCRIPTION: This function returns drawmap memory to the CC143 video memory. The memory must be requested first with the _vd_reqdmmem call.
If an error occurs _vd_retdmmem() returns -1 as its value and the appropriate error code in the global variable errno.

SYNOPSIS:

```
int _vd_actsn(path,drawmap)
int path; /* path number of video device */
unsigned char *drawmap;
```

DESCRIPTION: This function activates the drawmap memory which is passed, as being the current displayed screen.

SYNOPSIS:

```
int _vd_snoff(path)
int path; /* path number of video device */
```

DESCRIPTION: This function disables the screen.

_vd_getclut()   Get a specific CLUT value   _vd_getclut()

SYNOPSIS:

```
int _vd_getclut(path,clut)
int path; /* path number of video device */
int clut; /* CLUT register number */
```

DESCRIPTION: This function returns the value of the CLUT register specified by in the following format:

```
 ---------------------------------
 |   0   |  red  | green | blue   |
 ---------------------------------
bit31                           bit0
```

If an error occurs it returns -1 as its value and the appropriate error code in the global variable errno.

_vd_getcluts() Get a range of CLUT values _vd_getcluts()

SYNOPSIS:

```
int _vd_getcluts(path,stclut,numcluts,clutvals)
int path; /* path number of video device */
int stclut; /* start CLUT register number */
int numcluts; /* number of CLUTS to read */
char *clutvals; /* array to hold CLUT values */
```

DESCRIPTION: This function reads the specified number of CLUT register values (numcluts) starting at stclut into the array pointed to by clutvals. The individual CLUT values have the following format:

```
 _________________________
| red   | green  | blue    |
--------------------------
  byte0   byte1    byte2
```

If an error occurs it returns -1 as its value and the appropriate error code in the global variable errno.

SYNOPSIS:

```
int _vd_setclut(path,clut,value)
int path; /* path number of video device */
int clut; /* CLUT register number */
int value; /* CLUT color value to set */
```

DESCRIPTION: This function sets one CLUT register specified by clut to the value given in value. clut is in the following format:

```
     ----------------------------------
     |    0    |   red   | green | blue  |
     ----------------------------------
   bit31                               bit0
```

If an error occurs it returns -1 as its value and the appropriate error code in the global variable errno.

_vd_setcluts() Set a range of CLUT values _vd_setcluts()

SYNOPSIS:

```
int _vd_setcluts(path,stclut,numcluts,clutvals)
int path; /* path number of video device */
int stclut; /* start CLUT register number */
int numcluts; /* number of CLUTS to be set */
char *clutvals; /* pointer to buffer of  CLUT  color
values */
```

DESCRIPTION: This function sets numcluts CLUT values to the CLUT registers of the hardware. CLUT values will be set starting at the stclut register. The data will be copied from the buffer pointed to by clutvals. The individual color values have the following format:

red	green	blue
byte0	byte1	byte2

If an error occurs it returns -1 as its value and the appropriate error code in the global variable errno.

2.3 Keyboard functions

_kb_ssig()Send signal when a new key value is available_kb_ssig()

SYNOPSIS:

```
int _kb_ssig(kbdpath,sigcode)
int kbdpath;
short sigcode;
```

DESCRIPTION: This function sets up a signal to be sent to the calling process when a new key value is available. As soon as a new value is available, the signal 'sigcode' is sent to the calling process.

_kb_ssig() must be called each time the signal is sent if it is to be used again.

If an error occurs _kb_ssig() returns -1 as its value and the appropriate error code in the global variable errno.

_kb_rel() Release signal to be sent _kb_rel()

SYNOPSIS:

```
int _kb_rel(kbdpath)
int kbdpath;
```

DESCRIPTION: This function cancels the signal to be sent to the calling process when a new key becomes available. The function _kb_ssig() enables this function.

If an error occurs _kb_rel() returns -1 as its value and the appropriate error code in the global variable errno.

CAVEATS: The signal request is also cancelled when the issuing process dies or closes the path to the device.

2.4 Pointer functions

_pt_ssig()Send signal when a new key value is available_pt_ssig()

SYNOPSIS:

```
int _pt_ssig(ptdpath,sigcode)
int ptdpath;
short sigcode;
```

DESCRIPTION: This function sets up a signal to be sent to the calling process when a new key value is available. As soon as a new value is available, the signal 'sigcode' is sent to the calling process.

_pt_ssig() must be called each time the signal is sent if it is to be used again.

If an error occurs _pt_ssig() returns -1 as its value and the appropriate error code in the global variable errno.

_pt_rel()   Release signal to be sent   _pt_rel()

SYNOPSIS:

```
int _pt_rel(ptdpath)
int ptdpath;
```

DESCRIPTION: This function cancels the signal to be sent to the calling process when a new key becomes available. The function _pt_ssig() enables this function.

If an error occurs _pt_rel() returns -1 as its value and the appropriate error code in the global variable errno.

CAVEATS: The signal request is also cancelled when the issuing process dies or closes the path to the device.

CHAPTER 4

Device Drivers

4.5 scvid143

This is the video driver for the CC143. It has functions to initialize the hardware, write and read the CLUT table, and extract memory.
All these functions are implemented using OS-9 GetStat and SetStat calls. These functions can be called using the C-interface which is described in chapter 3. The Read and Write entries are empty entries, and should not be used,

4.6 scptr143

This is the pointer(mouse) driver for the CC143. It has functions to initialize the hardware, and to read the mouse position. The driver assumes that a logitech (or compatible) mouse is connected to the mouse port of the CC143
The functions are implemented using OS-9 GetStat and SetStat calls. These functions can be called using the C-interface which is described in chapter 3. The Read and Write entries are empty entries, and should not be used,

4.7 sckbd143

This is the keyboard driver for the CC143. It has functions to initialize the hardware, and to read key kodes. The driver assumes that a IBM/PS2 or compatible keyboard is connected to the keyboard port of the CC143
The functions are implemented using OS-9 GetStat and SetStat calls. These functions can be called using the C-interface which is described in chapter 3. The Read and Write entries are empty entries, and should not be used,

4.8 sccc143

This is a standard SCF driver for the mouse port of the CC143. If another device than a mouse is connected, e.g. a terminal, this driver can be used in combination with the 'sm' device descriptor.

CHAPTER 5

Device Descriptors

5.9 Descriptors

The Device Descriptors of the video part are called:

```
   vid_***_R***x***_*M
        |     |    |  |___ X-Tal frequency of CC143.
        |     |    |        5 or 8 MHz.
        |     |    |
        |     |    |______ Vertical resolution
        |     |
        |     |__________ Horizontal resolution
        |
        |_______________ CPU-module type
```

Example : vid_cc112_R1024x768_5M is the video descriptor for a CC112 and a screen resolution of 1024 by 768 for a CC143 with a 5MHz X-tal.
The correct 'vid' descriptors has to be loaded into memory for proper operation. The device name is called 'vid'.

The Device Descriptors of the ptr part are called:

```
   ptr_***
        |
        |_______________ CPU-module type
```

Example : ptr_cc112 is the ptr descriptor for a CC112.

The Device Descriptors of the kbd part are called:

```
   kbd_***
        |
        |_______________ CPU-module type
```

Example : kbd_cc112 is the kbd descriptor for a CC112.

The source of the desciptors can be found in the 'DESCRIPTORS' directory of your distribution disk. The various resolutions are described in the 'resolutions.d' file. The CPU dependant values are described in the files systype_*.d, where * is the CPU type. If new descriptors have to be added, create a new systype_*.d , and change the makefile. The objects of the device descriptors can be found in the 'OBJS' directory of your distribution disk.