PlayStation®2 IOP Library Reference
Release 2.4.3

Device Libraries

© 2002 Sony Computer Entertainment Inc.

Publication date: January 2002

Sony Computer Entertainment Inc.
1-1, Akasaka 7-chome, Minato-ku
Tokyo 107-0052, Japan

Sony Computer Entertainment America
919 E. Hillsdale Blvd.
Foster City, CA 94404, U.S.A.

Sony Computer Entertainment Europe
30 Golden Square
London W1F 9LD, U.K.

The *PlayStation®2 IOP Library Reference - Device Libraries* manual is supplied pursuant to and subject to the terms of the Sony Computer Entertainment PlayStation® license agreements.

The *PlayStation®2 IOP Library Reference - Device Libraries* manual is intended for distribution to and use by only Sony Computer Entertainment licensed Developers and Publishers in accordance with the PlayStation® license agreements.

Unauthorized reproduction, distribution, lending, rental or disclosure to any third party, in whole or in part, of this book is expressly prohibited by law and by the terms of the Sony Computer Entertainment PlayStation® license agreements.

Ownership of the physical property of the book is retained by and reserved by Sony Computer Entertainment. Alteration to or deletion, in whole or in part, of the book, its presentation, or its contents is prohibited.

The information in the *PlayStation®2 IOP Library Reference - Device Libraries* manual is subject to change without notice. The content of this book is Confidential Information of Sony Computer Entertainment.

and PlayStation are registered trademarks of Sony Computer Entertainment Inc. All other trademarks are property of their respective owners and/or their licensors.

Summary Table of Contents

About This Manual

This is the Runtime Library Release 2.4.3 version of the *PlayStation®2 IOP Library Reference - Device Libraries* manual.

The purpose of this manual is to define all available PlayStation®2 IOP device library structures and functions. The companion *PlayStation®2 IOP Library Overview - Device Libraries* describes the structure and purpose of the libraries.

Changes Since Last Release

Chapter 2: Hard Disk Library (for IOP)

- In the "Return Value" section of the devctl command HDIOC_STATUS, the description of return value 2 has been changed.

Chapter 4: i.LINK Socket Library

- A simplified explanation of each function has been added under the function name.

Chapter 6: USB Driver Library

- Descriptions of the following structures have been added.

  sceUsbdIsochronousPswLen()

  sceUsbdMultiIsochronousRequest()

- A description of the sceUsbdMultiIsochronousTransfer() function has been added.
- Notes have been added in the "Description" sections of sceUsbdOpenPipe() and sceUsbdOpenPipeAligned().

Chapter 7: USB Module Autoloader

- In the "Structure" and "Members" sections of the USBDEV_t() structure, descriptions for the following members have been added.

  modid
  modname
  load_result

- In the "Structure" and "Members" sections of the USBDEV_t() structure, the description of the load_flag member has been deleted.

Related Documentation

Library specifications for the EE can be found in the *PlayStation®2 EE Library Reference* manuals and the *PlayStation®2 EE Library Overview* manuals.

Note: the Developer Support Web site posts current developments regarding the Libraries and also provides notice of future documentation releases and upgrades.

Typographic Conventions

Certain Typographic Conventions are used throughout this manual to clarify the meaning of the text:

Convention	Meaning
`courier`	Indicates literal program code.
italic	Indicates names of arguments and structure members (in structure/function definitions only).
medium bold	Indicates data types and structure/function names (in structure/function definitions only).
blue	Indicates a hyperlink.

Developer Support

Sony Computer Entertainment America (SCEA)

SCEA developer support is available to licensees in North America only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
In North America:	*In North America:*
Attn: Developer Tools Coordinator Sony Computer Entertainment America 919 East Hillsdale Blvd. Foster City, CA 94404, U.S.A. Tel: (650) 655-8000	E-mail: PS2_Support@playstation.sony.com Web: http://www.devnet.scea.com/ Developer Support Hotline: (650) 655-5566 (Call Monday through Friday, 8 a.m. to 5 p.m., PST/PDT)

Sony Computer Entertainment Europe (SCEE)

SCEE developer support is available to licensees in Europe only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
In Europe:	*In Europe:*
Attn: Production Coordinator Sony Computer Entertainment Europe 30 Golden Square London W1F 9LD, U.K. Tel: +44 (0) 20 7859-5000	E-mail: ps2_support@scee.net Web: https://www.ps2-pro.com/ Developer Support Hotline: +44 (0) 20 7859-5777 (Call Monday through Friday, 9 a.m. to 6 p.m., GMT)

Chapter 1: CD(DVD)-ROM Library (for IOP)
Table of Contents

Structures

sceCdCLOCK

Structure which stores the date and time

Library	*Introduced*	*Documentation last modified*
libcdvd	1.5	January 22, 2001

Structure

typedef struct {

u_char *stat;*	0: normal. Any other: error (e.g. internal battery is dead)
u_char *second;*	Second (BCD value)
u_char *minute;*	Minute (BCD value)
u_char *hour;*	Hour (BCD value)
u_char *pad;*	Padding data produced by alignment
u_char *day;*	Day (BCD value)
u_char *month;*	Month (BCD value)
u_char *year;*	Year (BCD value)

} **sceCdCLOCK;**

Description

Stores the date and time with a BCD value.

See also

sceCdReadClock()

sceCdlFILE

File descriptor

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	April 16, 2001

Structure

typedef struct {

u_int *lsn;*	Logical sector number of file
u_int *size;*	File size (in bytes)
char *name[16];*	Filename
u_char *date[8];*	1st: Seconds 2nd: Minutes 3rd: Hours 4th: Date 5th: Month 6th 7th: Year (4 digits)
u_int *flag;*	Bits 0-7 are the ISO9660 file flag; other bits are reserved

} **sceCdlFILE;**

Description

Structure representing CD(DVD)-ROM file position and size.

See also

sceCdSearchFile()

sceCdlLOCCD

CD-ROM read location

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	December 23, 1999

Structure

```
typedef struct {
    u_char minute;      Minutes
    u_char second;      Seconds
    u_char sector;      Sector
    u_char track;       Track number
} sceCdlLOCCD;
```

Description

Structure representing read position (head position) on the CD-ROM.

Notes

Provided solely to calculate the CD read location using minutes/seconds/sectors.

See also

sceCdIntToPos(), sceCdPosToInt()

sceCdRMode

CD(DVD)-ROM read mode

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	October 11, 2001

Structure

typedef struct {

u_char *trycount;*	Read try count (No. of error retries + 1) (0: 256 tries)
u_char *spindlctrl;*	SCECdSpinStm: Recommended stream rotation speed. SCECdSpinNom: Starts reading data at maximum rotational velocity and if a read error occurs, the rotational velocity is reduced.
u_char *datapattern;*	SCECdSecS2048: Data size 2048 bytes SCECdSecS2328: 2328 bytes SCECdSecS2340: 2340 bytes
u_char *pad;*	Padding data produced by alignment

} **sceCdRMode;**

Description

This structure is used to specify the CD(DVD)-ROM read mode.

datapattern for DVD media reads is effective only with SCECdSecS2048.

See also

sceCdRead()

sceCdStmInit

File I/O functions: Stream initialization structure

Library	*Introduced*	*Documentation last modified*
libcdvd	2.4	October 11, 2001

Structure

typedef struct {

u_int *bufmax***;**	Capacity of stream buffer, in its entirety (in number of 2048-byte sectors)
u_int *bankmax***;**	Number of subdivisions of the stream buffer (i.e. number of ring buffers) For a buffer that has been subdivided into 3 more parts, the desired buffer size is approximately 16 sectors.
u_int *iop_bufaddr***;**	IOP memory address of stream buffer

} sceCdStmInit;

Description

This structure is used to specify initial values of the stream for the sceDevctl() file I/O functions.

See also

CDIOC_STREAMINIT

Functions

sceCdBreak

Break command

Library	*Introduced*	*Documentation last modified*
libcdvd	1.4	October 11, 2001

Syntax

int sceCdBreak (void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Breaks the executing command (e.g., sceCdPause(), sceCdRead(), sceCdSeek(), sceCdStandby(), sceCdSstatus(), sceCdstop()).

The sceCdSync() function is used to confirm that break processing has ended.

Breaks the processing of each command and calls the callback function, if one is set.

SCECdErABRT will be set for drive error information.

Return value

0 if command issue failed.

1 if command issue succeeded.

sceCdCallback

Define sceCdSync callback function

Library	*Introduced*	*Documentation last modified*
libcdvd	1.3	August 31, 2001

Syntax

int sceCdCallback (

void *(*func)***(int))** Address of callback function

Calling conditions

Can be called from a thread

Multithread safe

Description

Sets the callback *func* called when a non-blocking function terminates.

When the callback is set, the function *func* is called when the non-blocking function terminates.

The function *func* is called by the interrupt handler.

If *func* is set to 0 or the command fails to issue, the callback does not occur.

Moreover, a callback cannot be set when a function that has already caused a callback is executing.

The function code of the cause of the callback is passed to the callback function in the first argument, as shown below.

SCECdFuncRead	sceCdRead() function has terminated.
SCECdFuncSeek	sceCdSeek () function has terminated.
SCECdFuncStandby	sceCdStandby() function has terminated.
SCECdFuncStop	sceCdStop() function has terminated.
SCECdFuncPause	sceCdPause() function has terminated.

Note about callback functions

- The callback function is called in the interrupt handler while interrupts are inhibited. Consequently, processing must be completed as quickly as possible. In addition, if a dedicated interrupt processing function is provided by the library, that function must be used.

Return value

Returns the address of the previously set callback function, or 0 if no callback was set.

sceCdChangeThreadPriority

Change the IOP thread priority of an EE-side request processing module

Library	*Introduced*	*Documentation last modified*
libcdvd	2.0	July 2, 2001

Syntax

int sceCdChangeThreadPriority(

int *priority***)** Value of IOP thread priority for EE-side request processing module

Calling conditions

Can be called from a thread

Not multithread safe

Description

This function sets the IOP thread priority of an EE-side request processing module.

The default value for IOP thread priority of an EE-side request processing module is 81.

When changing the IOP thread priority, careful consideration must be given to the priorities of other modules. Therefore, the IOP thread priority value should not be changed carelessly.

Return value

If command issue failed, the KernelErrorCode from the IOP is returned.
0 is returned if the command was successfully issued.

sceCdDiskReady

Check drive status

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	July 2, 2001

Syntax

int sceCdDiskReady (

int *mode)* — Check mode (0: blocking, 1: non-blocking)
When mode is set to non-blocking, the operating conditions of other threads must be thoroughly considered.

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Checks the drive status and determines if a command can be issued. If there is no media in the drive, SCECdNotReady is returned.

If the mode argument is set to blocking, and the drive rotation is not stable, the function waits until the drive rotation is stable, then it returns. In the non-blocking mode, the function returns immediately after the status is checked.

When this function is used for polling in non-blocking mode in a multithreaded environment, a function such as DelayThread() must be used so that there is sufficient room for other threads to operate.

Return value

SCECdComplete	Drive state allows commands to be issued
SCECdNotReady	Drive cannot accept commands

sceCdGetDiskType

Get media format

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	August 31, 2001

Syntax

int sceCdGetDiskType (void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets the media format

Return value

SCECdIllgalMedia	Disc cannot be played
SCECdPS2DVD	Disc is a PlayStation 2 DVD
SCECdPS2CD	Disc is a PlayStation 2 CD
SCECdPS2CDDA	Disc is a PlayStation 2 CD (with CDDA)
SCECdPSCD	Disc is a PlayStation CD
SCECdPSCDDA	Disc is a PlayStation CD (with CDDA)
SCECdDVDV	Disc is DVD Video
SCECdCDDA	Disc is a music CD
SCECdDETCT	Analyzing disc
SCECdNODISC	No disc mounted
SCECdUNKNOWN	Undistinguishable disk

sceCdGetError

Get drive error information

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	October 11, 2001

Syntax

int sceCdGetError (void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Gets drive error information.

Return value

Table 1-1

Return value	Meaning
SCECdErFAIL	sceCdGetError() function issue failed
SCECdErNO	No error
SCECdErEOM	Outermost track reached during playback
SCECdErTRMOPN	Cover opened during playback
SCECdErREAD	Problem occurred during read
SCECdErCUD	Not appropriate for disc in drive
SCECdErNORDY	Processing command
SCECdErABRT	Abort command received
SCECdErREADCF	Read command issue failed

sceCdGetReadPos

Check the progress of sceCdRead()

Library	*Introduced*	*Documentation last modified*
libcdvd	1.3	July 2, 2001

Syntax

u_int sceCdGetReadPos (void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

If the media is a CD, values in 1 sector units (a multiple of 2048) will be returned. If the media is a DVD, values in 16 sector units (a multiple of 32768) will be returned.

Return value

Returns the progress of the sceCdRead() function as the size of the data transferred to the buffer.

When sceCdRead() terminates, 0 is returned.

sceCdGetToc

Read TOC

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	July 2, 2001

Syntax

int sceCdGetToc (

u_char **toc)* — Address returned by location table information (a 1024 byte area is required).

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Gets TOC sector information from CD-ROM.

Return value

1 is returned if the command was successfully issued, else 0 is returned.

sceCdInit

Initialize the CD(DVD)-ROM subsystem

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	July 2, 2001

Syntax

int sceCdInit (

int *init_mode)* Library initialization mode

SCECdINIT: Initialize library and block until commands can be issued.

SCECdINoD: Initialize library only

SCECdEXIT: Close library

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Initializes the CD(DVD)-ROM subsystem.

Notes

sceCdInit must be used for initialization first even if stdio (e.g., sceRead()) will be used.

After performing initialization with sceCdInit(), be sure to call sceCdMmode() to specify the type of media (CD or DVD).

If this function is used when cdvdman.irx and cdvdfsv.irx have not been replaced within the IOP default module, 2 is returned.

Return value

0: Initialization failed.

1: Initialization was performed normally.

2: Although initialization was performed, the default module was detected on the IOP side.

See also

sceCdMmode()

sceCdIntToPos

Get CD-ROM's minutes/seconds/sectors from logical sector

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	July 2, 2001

Syntax

sceCdlLOCCD *sceCdIntToPos (

int *i,*	Logical sector number
sceCdlLOCCD **p)*	Minutes/seconds/sectors

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Calculates minutes/seconds/sectors from logical sector number.

Not meaningful when the media is DVD.

Return value

Returns the address of CdlLOCCD.

sceCdMmode

Specify the media for reading

Library	*Introduced*	*Documentation last modified*
libcdvd	2.0	July 2, 2001

Syntax

int sceCdMmode(

int *media***)**

Read media	
SCECdCD	Specify CD as the read media.
SCECdDVD	Specify DVD as the read media.

Calling conditions

Can be called from a thread

Not multithread safe

Description

This function is used to specify the read media for the CD(DVD)-ROM subsystem.

Notes

This function must be used to specify the read media after the sceCdInit() function is called.

Return value

0 is returned if command issue failed. 1 is returned if the command was successfully issued.

See also

sceCdInit()

sceCdPause

Pause CD(DVD)-ROM head

Library	*Introduced*	*Documentation last modified*
libcdvd	1.2	July 2, 2001

Syntax

int sceCdPause (void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

The head is put in a pause state at its current position on the CD(DVD)-ROM.

Notes

Since the function is a non-blocking function, the actual pausing of the head must be detected with sceCdSync().

Return value

1 is returned if the command was successfully issued, else 0 is returned.

See also

sceCdSync()

sceCdPOffCallback

Set PlayStation 2 power off callback function

Library	*Introduced*	*Documentation last modified*
libcdvd	2.2.2	August 31, 2001

Syntax

int sceCdPOffCallback (

void (**func*)**(void *)**	Address of the callback function
void ****addr*)**	Address of the callback argument

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt enabled state)

Description

For compatibility with the hard disk drive (EXPANSION BAY type), in order to use the hard disk this function must be used to perform hard disk power-off processing. It is only for an EXPANSION BAY type hard disk drive.

The function sets the callback *func* that is to be called when the power-off operation is performed.

When a callback is set, the function *func* is called when the power-off operation is performed.

The function *func* is called by the interrupt handler.

If 0 is specified for *func*, no callback will occur.

Return value

Address of the callback function set previously. 0 is returned if the callback has not been set.

sceCdPosToInt

Get CD-ROM's logical sector number from minutes/seconds/sectors

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	July 2, 2001

Syntax

int sceCdPosToInt (

sceCdlLOCCD **p)* Minutes/seconds/sectors

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Calculates logical sector number from minutes/seconds/sectors value.

Not meaningful when the media is DVD.

Return value

Logical sector number

sceCdPowerOff

PlayStation 2 power OFF

Library	*Introduced*	*Documentation last modified*
libcdvd	2.2.2	August 31, 2001

Syntax

int sceCdPowerOff (

int **stat***)** Status

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function issues a PlayStation 2 PowerOff request.

This function must be used in power-off processing when a hard disk drive or HDD Ethernet (smap.irx) is used.

For details about power-off processing, refer to the CD(DVD)-ROM library, PlayStation File System (pfs) and network (inet) overviews.

Notes

When calling this function, make sure it is executed after an interrupt is detected by sceCdPOffCallback() and the hard disk drive is powered off.

<Sample power-off processing function calling sequence when a hard disk drive is used>

```
printf("power off request has come.\n");
/* close all files */
devctl("pfs:", PDIOC_CLOSEALL, NULL, 0, NULL, 0);
/* dev9 power off, need to power off PS2 */
while(devctl("hdd:", HDIOC_DEV9OFF, NULL, 0, NULL, 0) < 0);
/* PS2 power off */
while(!sceCdPowerOff(&stat) || stat);
while(1);
```

Notes:

- With a hard disk drive (EXPANSION BAY type), if the RESET button on the system unit is pressed between the time hard disk power-off processing is performed and PlayStation 2 system unit power-off processing is performed, the PlayStation 2 system unit will be reset.
- When cdvdfsv.irx (cdvd_ee_driver) has been unloaded, use the devctl command CDIOC_POWEROFF.

Return value

0 if command issued failed

stat return value bit7: 1 Command error

sceCdRead

Read data

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	July 2, 2001

Syntax

int sceCdRead (

u_int *lsn,*	Logical sector number at which to begin reading
u_int *sectors,*	Number of sectors to read
void **buf,*	Read buffer
sceCdRMode **mode***)**	Read mode

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

A seek is performed to the starting read position indicated by lsn.

The number of sectors of data specified by the sectors argument is read from StartPoint on the CD(DVD)-ROM and placed in the memory specified by buf. The head is then put in the pause state.

Notes

CD-DA and DVD-video data cannot be read.

Since this is a non-blocking function, the actual completion of the data transfer must be detected using sceCdSync().

Note on using this function

- If the surface of the disk is severely damaged, it may not be possible to detect the completion of this function using sceCdSync(). In this case, use the sceCdBreak() function to abort this function.

Return value

1 is returned if the command was successfully issued, otherwise 0 is returned.

See also

sceCdSync()

sceCdReadClock

Get date and time

Library	*Introduced*	*Documentation last modified*
libcdvd	1.5	July 2, 2001

Syntax

int sceCdReadClock (

sceCdCLOCK **rtc)* Address of structure where date and time are stored

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Gets the date and time from the PlayStation 2's built-in real-time clock.

Notes

For this function to use a controller which performs drive-related processing, an interval of 300(msec) must be cleared when calling it continuously.

Also, the following values are returned in the stat member of the rtc time storage structure.

bit 1: Clock battery monitoring voltage problem

bit 7: Command error

Return value

1 is returned if the command was successfully issued, otherwise 0 is returned.

sceCdSearchFile

Get position and size from filename

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	July 2, 2001

Syntax

int sceCdSearchFile (

sceCdlFILE **fp,*	Pointer to CD(DVD)-ROM file structure
const char **name***)**	Filename

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Determines absolute position LSN (logical sector number) and size from a filename on the CD(DVD)-ROM. The result is stored in *fp*.

Notes

Filenames must be specified fully using absolute paths.

Position information for files in the same directory as the specified file is cached in memory.

Return value

0: No file was found.

1: File structure pointer was successfully obtained.

sceCdSeek

Move CD(DVD)-ROM head

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	July 2, 2001

Syntax

int sceCdSeek (

u_int *lsn***)** Target logical sector number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Seeks CD(DVD)-ROM head to target position and puts head in PAUSE state.

Notes

Since this is a non-blocking function, sceCdSync() must be used to determine completion of the head seek.

Return value

1 is returned if the command was successfully issued, otherwise 0 is returned.

See also

sceCdSync()

sceCdStandby

Start rotation of the media

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	July 2, 2001

Syntax

int sceCdStandby(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Spins up the CD(DVD)-ROM media and puts the head in PAUSE state at the innermost track.

Notes

Since this is a non-blocking function, sceCdSync() must be used to determine when the actual operation is completed.

Return value

1 is returned if the command was successfully issued, otherwise 0 is returned.

See also

sceCdSync()

sceCdStatus

Get drive status

Library	*Introduced*	*Documentation last modified*
libcdvd	1.2	July 2, 2001

Syntax

int sceCdStatus(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Returns current status of drive.

Return value

A -1 is returned if the command was not successfully issued. If the command was successfully issued, the status is returned according to the list below.

Table 1-2

Return value	Meaning
SCECdStatShellOpen	Tray is OPEN
SCECdStatStop	Stopped
SCECdStatSpin	Spindle is spinning
SCECdStatRead	Reading
SCECdStatPause	Paused (unreferenced)
SCECdStatSeek	Seeking
SCECdStatEmg	Abnormal termination

sceCdStInit

Initialize stream

Library	*Introduced*	*Documentation last modified*
libcdvd	1.4	July 2, 2001

Syntax

int sceCdStInit(

u_int *bufmax,*	Capacity of entire stream buffer (specified using number of sectors, in 2048-byte units)
u_int *bankmax,*	Number of stream buffer partitions (number of ring buffers) A buffer with three or more partitions should have a capacity of approximately 16 sectors.
u_int *iop_bufaddr***)**	IOP memory address of stream buffer

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Initializes stream and registers the stream buffer (creates ring buffer).

Notes

CD-DA data and DVD-video data cannot be read.

Return value

0 if command issue failed.

1 if command issue succeeded.

See also

sceSifAllocIopHeap(), sceCdStRead(), sceCdStSeek(), sceCdStSeekF(), sceCdStStart(), sceCdStStat(), sceCdStStop(), sceCdStPause(), sceCdStResume()

sceCdStop

Stop rotation of the media

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	July 2, 2001

Syntax

int sceCdStop(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Stops rotation of the CD(DVD)-ROM media.

Notes

Since the function is a non-blocking function, sceCdSync()must be used to determine when the actual operation is finished.

Return value

1 is returned if the command was successfully issued, otherwise 0 is returned.

See also

sceCdSync()

sceCdStPause

Pause stream

Library	*Introduced*	*Documentation last modified*
libcdvd	1.6	July 2, 2001

Syntax

int sceCdStPause(void)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Pauses the reading of stream data while maintaining the contents of the stream buffer.

Notes

Use sceCdStResume() to restart the reading of stream data.

Return Value

If command issue failed, 0 is returned. If it succeeded, 1 is returned.

See Also

sceCdStInit(), sceCdRead(), sceCdStSeek(), sceCdStSeekF(), sceCdStStart(), sceCdStStat(), sceCdStStop(), sceCdStResume()

sceCdStRead

Read stream data

Library	*Introduced*	*Documentation last modified*
libcdvd	1.4	July 2, 2001

Syntax

int sceCdStRead(

u_int *sectors,*	Number of sectors of data to read from stream buffer
u_int **buf,*	Data read address
u_int *mode,*	Data read mode STMNBLK: Returns only data currently in stream buffer. STMBLK: Block reads are performed until the specified number of sectors of data are read or an error occurs.
u_int **err)*	Error code storage address Error code is the same as that obtained using sceCdGetError().

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Reads data from the stream buffer.

CD-DA data and DVD-video data cannot be read.

Return value

Returns the number of sectors read (2048-byte units).

See also

sceCdStInit(), sceCdStSeek(), sceCdStStart(), sceCdStStat(), sceCdStSeekF(), sceCdStStop(), sceCdStPause(), sceCdStResume()

sceCdStResume

Restart stream

Library	*Introduced*	*Documentation last modified*
libcdvd	1.6	July 2, 2001

Syntax

int sceCdStResume(void)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Restarts the reading of stream data (cancels a pause due to the sceCdStPause() function).

Return value

If command issue failed, 0 is returned. If it succeeded, 1 is returned.

See Also

sceCdStInit(), sceCdStRead(), sceCdStSeek(), sceCdStSeekF(), sceCdStStart(), sceCdStStat(), sceCdStStop(), sceCdStPause()

sceCdStSeek

Change stream position

Library	*Introduced*	*Documentation last modified*
libcdvd	1.4	July 2, 2001

Syntax

int sceCdStSeek(

u_int *lsn)* — Changed stream position (specified according to logical sector number)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Destroys contents of stream buffer and changes the current stream position.

Return value

0 if command issue failed. 1 if command issue succeeded.

See also

sceCdStInit(), sceCdStRead(), sceCdStSeekF(), sceCdStStart(), sceCdStStat(), sceCdStStop(), sceCdStPause(), sceCdStResume()

sceCdStSeekF

Change stream position (high-speed version)

Library	*Introduced*	*Documentation last modified*
libcdvd	2.1	July 2, 2001

Syntax

int sceCdStSeekF(

u_int *lsn)* Changed stream position (specified according to logical sector number)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

This function discards the stream buffer contents and changes the current stream position.

This entire function has improved performance over sceCdStSeek().

Return value

0 is returned if command issue failed. 1 is returned if it was successful.

See also

sceCdStInit(), sceCdStRead(), sceCdStSeek(), sceCdStStart(), sceCdStStat(), sceCdStStop(), sceCdStPause(), sceCdStResume()

sceCdStStart

Start streaming

Library	*Introduced*	*Documentation last modified*
libcdvd	1.4	October 11, 2001

Syntax

int sceCdStStart(

u_int *lsn,*	Stream start position (specified using logical sector number)
sceCdRMode **mode***)**	Read mode

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Starts reading from the specified stream start position into the stream buffer.

After streaming starts, data is read from the CD(DVD) into the streaming buffer recurrently in the background. This means that functions like the file control functions and sceCDRead() cannot be used to read from the CD(DVD)-ROM until streaming has been stopped with sceCdStStop().

The only value that can be specified for datapattern *mode* is SCECdSecS2048.

CD-DA data and DVD-video data cannot be read.

Return value

0 if command issue failed. 1 if command issue succeeded.

See also

sceCdStInit(), sceCdStRead(), sceCdStSeek(), sceCdStSeekF(), sceCdStStat(), sceCdStStop(), sceCdStPause(), sceCdStResume()

sceCdStStat

Get stream data read status

Library	*Introduced*	*Documentation last modified*
libcdvd	1.4	July 2, 2001

Syntax

int sceCdStStat (void)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Gets current read status of stream data.

Return value

0 if command issue failed. On success, returns the number of sectors of data that have been accumulated in the stream (in 2048-byte units).

See also

sceCdStInit(), sceCdStRead(), sceCdStSeek(), sceCdStSeekF(), sceCdStStart(), sceCdStStop(), sceCdStPause(), sceCdStResume()

sceCdStStop

Stop streaming

Library	*Introduced*	*Documentation last modified*
libcdvd	1.4	July 2, 2001

Syntax

int sceCdStStop (void)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Stops streaming.

Return value

0 if command issue failed. 1 if command issue succeeded.

See also

sceCdStInit(), sceCdStRead(), sceCdStSeek(), sceCdStSeekF(), sceCdStStart(), sceCdStStat(), sceCdStPause(), sceCdStResume()

sceCdSync

Wait for command completion

Library	*Introduced*	*Documentation last modified*
libcdvd	1.1	July 2, 2001

Syntax

int sceCdSync (

int *mode)*

0x00: Wait for completion of command (blocking)

0x01: Check current status and return immediately (non-blocking).
When using this mode, the operating conditions of other threads must be thoroughly considered.

0x10: Wait for completion of command including completion of command issued from the EE (blocking).

0x11: Check and immediately return the current state, including the state of the command issued from the EE (non-blocking). When using this mode, the operating conditions of other threads must be thoroughly considered.

Calling conditions

The blocking type cannot be called in interrupt-disabled state.

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

When *mode* is set to 0x00, this function waits for the command being executed to complete and returns 0.

When *mode* is set to 0x01, this function checks the execution state of the command and returns either 0 or 1.

When *mode* is set to 0x10, this function waits for the command being executed to complete, including completion of the command issued from the EE, and returns 0.

When *mode* is set to 0x11, this function checks the execution state of the command, including the state of the command issued from the EE, and returns either 0 or 1.

When this function is used for polling in non-blocking mode in a multithreaded environment, a function such as DelayThread() must be used so that there is sufficient room for other threads to operate.

Return value

0: Completed, 1: Not completed

See also

sceCdRead(), sceCdSeek(), sceCdStop(), sceCdStandby(), sceCdGetToc()

sceCdTrayReq

Open and close the tray

Library	*Introduced*	*Documentation last modified*
libcdvd	1.3	July 2, 2001

Syntax

int sceCdTrayReq(

int *mode,*	Tray control mode SCECdTrayOpen: Open tray SCECdTrayClose: Close tray SCECdTrayCheck: Get tray state change
u_int **traycnt***)**	Address for returning whether or not there was a tray state change 0: Tray was not opened. 1: Tray was opened.

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

This function opens or closes the tray of the CD(DVD)-ROM drive according to the specified mode.

When *mode* is SCECdTrayCheck, the mode for getting the tray state change is set, and information indicating whether or not the tray was opened since the previous time this command was called in this mode is returned in **traycnt*.

Notes

Use sceCdDiskReady() to determine whether or not commands can be received after a disk has been inserted.

Return value

0 if command issue failed. 1 if command issue succeeded.

File Control Functions

File control functions

Library	*Introduced*	*Documentation last modified*
libcdvd	2.4	October 11, 2001

Syntax

The following file control functions are supported.

#include <sifdev.h> Refer to the documentation on the standard I/O functions for the arguments.

sceClose()

sceDclose()

sceDevctl()

sceDopen()

sceDread()

sceIoctl2()

sceLseek()

sceOpen() Additional arguments for the sceOpen() function:

filename cdrom0: + filename (ISO9660 Level 1)

flags Access mode. Specify either of the following constants.

SCE_RDONLY	Open only for reading
SCE_CdSTREAM	Open only for reading a stream

sceRead()

Description

File-based I/O functions are supported.

Precautions when the file is opened with SCE_CdSTREAM for reading a stream:

1. The size argument of the sceRead() function must be specified as a multiple of 2048.
2. The CDIOC_GETERROR command must be used to obtain the read error.
3. After the file is opened, data is recursively read from the CD(DVD)-ROM to the streaming buffer in the background. Therefore, the file control functions and functions such as sceCdRead() cannot be used to read from the CD(DVD)-ROM until streaming is terminated using the sceClose() function.

devctl Commands

CDIOC_BREAK

Interrupt command

Library	*Introduced*	*Documentation last modified*
libcdvd	2.4	October 11, 2001

Arguments

arg	Reserved. Set to NULL.
arglen	Reserved. Set to 0.
bufp	Reserved. Set to NULL.
buflen	Reserved. Set to 0.

Description

This command interrupts a currently executing command (such as sceRead(), CDIOC_STANDBY, CDIOC_STOP, CDIOC_PAUSE, sceCdPause(), sceCdRead(), sceCdSeek(), sceCdStandby(), sceCdSstatus(), or sceCdstop()).

When a command is interrupted, a callback function is called if one was previously set.

SCECdErABRT will be set for drive error information.

Return value

If processing succeeds, 0 is returned.

On error, the product of errno and -1 is returned.

CDIOC_DISKRDY

Check drive state

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	August 31, 2001

Arguments

arg	Check mode (0: Blocking, 1: Non-blocking) storage address
arglen	sizeof(int)
bufp	Drive state storage address
buflen	sizeof(int)

Description

The CDIOC_DISKRDY command checks the following drive states to determine whether or not a command can be issued.

SCECdComplete is the drive state that allows commands to be issued, and SCECdNotReady is the state in which the drive cannot accept commands. The state becomes SCECdNotReady when there is no media in the drive.

For the blocking case, if the drive rotation is unstable, the function will wait until the rotation becomes stable before returning. For the non-blocking case, the function will return immediately after checking the status.

When this function is used for non-blocking polling in a multithread environment, the DelayThread() (or equivalent) function must be used to make sure there is room for other threads to run.

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

CDIOC_GETDISKTYP

Get media format

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	August 31, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Reserved. Specify 0.
bufp	Media type storage address
buflen	sizeof(int)

Description

This command obtains one of the following media formats.

SCECdIllgalMedia	Play-prohibited disc
SCECdPS2DVD	Disc is a PlayStation 2 DVD
SCECdPS2CD	Disc is a PlayStation 2 CD
SCECdPS2CDDA	Disc is a PlayStation 2 CD (with CDDA)
SCECdPSCD	Disc is a PlayStation CD
SCECdPSCDDA	Disc is a PlayStation CD (with CDDA)
SCECdDVDV	Disc is a DVD Video
SCECdCDDA	Disc is a music CD
SCECdDETCT	Format detection in progress
SCECdNODISC	No disc has been inserted
SCECdUNKNOWN	Unknown disc format

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

CDIOC_GETERROR

Get drive error information

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	October 11, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Reserved. Specify 0.
bufp	Error information storage address
buflen	sizeof(int)

Description

This command obtains one of the following kinds of drive error information.

SCECdErFAIL	Processing for issuing sceCdGetError() function failed
SCECdErNO	No error occurred
SCECdErEOM	Reached outermost periphery during play
SCECdErTRMOPN	Drive was opened during play
SCECdErREAD	Problem occurred while reading
SCECdErCUD	Improper disc in drive
SCECdErNORDY	Command is being processed
SCECdErABRT	Command aborted
SCECdErREADCF	Read command issue failed

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

CDIOC_GETTOC

Read TOC

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	August 31, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Reserved. Specify 0.
bufp	TOC storage address. 1024-byte area is required.
buflen	1024

Description

This command gets the TOC sector information of the CD-ROM.

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

CDIOC_MMODE

Specify read media

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	August 31, 2001

Arguments

arg	Read media storage address
arglen	sizeof(int)
bufp	Reserved. Specify NULL.
buflen	Reserved. Specify 0.

Description

This command specifies one of the following read media types for the CD(DVD-ROM) subsystem.

SCECdCD	Specifies CD as the read media.
SCECdDVD	Specifies DVD as the read media.

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

CDIOC_PAUSE

Pause CD(DVD)-ROM head

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	August 31, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Reserved. Specify 0.
bufp	Reserved. Specify NULL.
buflen	Reserved. Specify 0.

Description

This command pauses the CD(DVD)-ROM head at its current location.

This call will block until processing ends.

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

CDIOC_POWEROFF

Power off PlayStation 2

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	August 31, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Reserved. Specify 0.
bufp	Status storage address
buflen	sizeof(int)

Description

This command issues a request to power off the PlayStation 2. For details, see the sceCdPowerOff() function reference.

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

CDIOC_READCLOCK

Get date and time

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	August 31, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Reserved. Specify 0.
bufp	Address of date/time storage structure sceCdClock for storing the date and time
buflen	sizeof(sceCdClock)

Description

This command gets the date and time. See the description of sceCdReadClock().

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

CDIOC_SPINNOM

Set adaptive speed control for the standard I/O media spin rate

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	August 31, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Reserved. Specify 0.
bufp	Reserved. Specify NULL.
buflen	Reserved. Specify 0.

Description

This command sets adaptive speed control for the standard I/O media spin rate. This causes data reading to begin at the highest spin rate, and when a read error occurs, it lowers the spin rate until reading can be performed properly.

The initial value for the standard I/O spin rate is adaptive speed control.

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

CDIOC_STANDBY

Start media rotation

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	August 31, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Reserved. Specify 0.
bufp	Reserved. Specify NULL.
buflen	Reserved. Specify 0.

Description

This command causes the CD(DVD)-ROM media to rotate, positions the head at the innermost circumference, and sets pause state.

This call blocks until processing ends.

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

CDIOC_STATUS

Get drive state

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	August 31, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Reserved. Specify 0.
bufp	Drive status storage address
buflen	sizeof(int)

Description

This command returns one of the following as the current drive state.

SCECdStatShellOpen	Tray is open
SCECdStatStop	Stop state
SCECdStatSpin	Spindle is rotating
SCECdStatRead	Read operation is executing (cannot be referenced)
SCECdStatPause	Pause state (cannot be referenced)
SCECdStatSeek	Seeking
SCECdStatEmg	Emergency stop

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

CDIOC_STOP

Stop media rotation

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	August 31, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Reserved. Specify 0.
bufp	Reserved. Specify NULL.
buflen	Reserved. Specify 0.

Description

This command stops the rotation of the CD(DVD)-ROM media.

This call blocks until processing ends.

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

CDIOC_STREAMINIT

Initialize streamer for file I/O functions

Library	*Introduced*	*Documentation last modified*
libcdvd	2.4	October 11, 2001

Arguments

arg	Reserved. Set to NULL.
arglen	Reserved. Set to 0.
bufp	Address of sceCdStmInit, the file I/O function stream initialization structure
buflen	sizeof(sceCdStmInit)

Description

This command initializes the streamer for file I/O functions and registers the stream buffer (creates a ring buffer).

Return value

On error, the product of errno and -1 is returned.

CDIOC_TRAYREQ

Open/close tray

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	August 31, 2001

Arguments

arg	Tray control mode
arglen	sizeof(int)
bufp	Address where tray state change, if present, is returned
buflen	sizeof(u_int)

Description

This command opens or closes the CD(DVD)-ROM drive tray according to the tray control mode specification.

If SCECdTrayCheck was specified for the tray control mode, the mode will become tray state change acquisition mode, and whether or not the tray was opened since the last time this command was called in tray state change acquisition mode is returned in the tray state change address.

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

CDIOC_TRYCNT

Set media read retry count for standard I/O

Library	*Introduced*	*Documentation last modified*
libcdvd	2.3.4	August 31, 2001

Arguments

arg	Read retry count storage address (0 <= Retry count <= 255; 0: 256 times)
arglen	sizeof(u_char)
bufp	Reserved. Specify NULL.
buflen	Reserved. Specify 0.

Description

This command sets the media read retry count for standard I/O. The initial value is set to 16 times.

Return value

If processing succeeds, 0 is returned.

If an error occurs, the product of errno and -1 is returned.

ioctl2 Commands

CDIOSTREAMSTAT

Get stream data read status

Library	*Introduced*	*Documentation last modified*
libcdvd	2.4	October 11, 2001

Arguments

arg	Reserved. Set to NULL.
arglen	Size of arg.
bufp	Reserved. Set to NULL.
buflen	Size of bufp.

Description

This command gets the current stream data read status.

Return value

On error, the product of errno and -1 is returned.

If processing succeeds, the amount of data already accumulated in the streamer is returned as the number of sectors (2048-byte units).

CIOCSTREMPAUSE

Pause stream

Library	*Introduced*	*Documentation last modified*
libcdvd	2.4	October 11, 2001

Arguments

arg	Reserved. Set to NULL.
arglen	Size of arg.
bufp	Reserved. Set to NULL.
buflen	Size of bufp.

Description

This command pauses the reading of stream data while maintaining the contents of the stream buffer.

Notes

Stream data reading can be resumed with CIOCSTREMRESUME.

Return value

If processing succeeds, 0 is returned.

On error, the product of errno and -1 is returned.

CIOCSTREMRESUME

Resume stream

Library	*Introduced*	*Documentation last modified*
libcdvd	2.4	October 11, 2001

Arguments

arg	Reserved. Set to NULL.
arglen	Size of arg.
bufp	Reserved. Set to NULL.
buflen	Size of bufp.

Description

This function resumes the reading of stream data (cancels a pause set by CIOCSTREMPAUSE).

Notes

To obtain IOP memory such as the stream buffer from the EE, use a function such as sceSifAllocIopHeap().

CD-DA data and DVD-video data cannot be read.

Return value

If processing succeeds, 0 is returned.

On error, the product of errno and -1 is returned.

Chapter 2: Hard Disk Library (for IOP)
Table of Contents

Structures

sce_dirent

Partition table entry

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Structure

struct sce_dirent {

struct sce_stat *d_stat;*	Partition status
char *d_name*[256];	Partition ID
void **d_private* **};**	Reserved

Description

This structure stores an entry of the partition table.

See also

sceDread(), dread()

sce_stat

Partition status

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Structure

struct sce_stat {

unsigned int *st_mode;*	Filesystem type of the partition
unsigned int *st_attr;*	bit 0 Sub-partition
unsigned int *st_size;*	Number of sectors in the partition
unsigned char *st_ctime*[8];	Creation time of the partition
unsigned char *st_atime*[8];	byte 0 reserved
unsigned char *st_mtime*[8];	byte 1 Seconds byte 2 Minutes byte 3 Hours byte 4 Day byte 5 Month byte 6-7 Year (4 digits)
unsigned int *st_hisize;*	
unsigned int *st_private*[6] };	word 0 For the main partition, represents the number of sub-partitions. For a sub-partition, represents the sub-partition number starting from 1.

Description

This structure stores partition status.

See also

struct sce_dirent, getstat()

Functions

close

Close main partition

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int close(
 int fd)                                  Previously open file descriptor
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Closes the opened partition and frees the file descriptor.

Return value

0 if successful.

-1 times errno if an error occurred.

EBADF *fd* is not a valid open descriptor.

dclose

Close partition table

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

```
#include <dirent.h>
int dclose(
 int fd)                    File descriptor
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Closes the opened partition table and frees the file descriptor.

Return value

0 if successful.

-1 times errno if an error occurred.

EBADF *fd* is not a valid open descriptor.

devctl

Special operations for a device

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

```
#include <sys/mount.h>
int devctl(
 const char *name,
 int cmd,
 void *arg,
 int arglen,
 void *bufp,
 size_t buflen)
```

name: Device name (hdd0:, hdd1:).

cmd: Operation command.

Any of the following constants can be specified.

- HDIOC_MAXSECTOR
- HDIOC_TOTALSECTOR
- HDIOC_IDLE
- HDIOC_FLUSH
- HDIOC_SWAPTMP
- HDIOC_DEV9OFF
- HDIOC_STATUS
- HDIOC_FORMATVER

arg: Command arguments. Depends on *cmd*.

arglen: Size of *arg*.

bufp: Arguments received from the command. Depends on *cmd*.

buflen: Size of *bufp*.

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Performs special operations for a device. For details regarding each of the commands, refer to the "devctl command list".

Return value

If successful, returns a command-dependent value.

If an error occured, returns -1 times errno.

The errors that are common to each of the commands are as follows.

EMFILE	Reached the maximum number of descriptors that can be opened.
ENODEV	Specified device does not exist.

dopen

Open partition table

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

#include <dirent.h>

int dopen(

const char **name***)** Device name (hdd0:, hdd1:)

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Opens a partition table. For obtaining information about all the partitions present on the disk, the partition table is viewed as a simulated directory.

Return value

Returns file descriptor on normal completion (value > 0).

Returns -1 times errno if an error occurred.

EMFILE	Reached the maximum number of descriptors that can be opened.
ENODEV	Specified device does not exist.

dread

Read partition table

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

```
#include <dirent.h>
int dread(
 int fd,                          File descriptor
 struct sce_dirent *buf)          Address of the buffer that stores the data that was read.
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

The next entry from the partition entry stream indicated by *fd* is copied to the sce_dirent structure *buf*. Returns 0 when reaches the end of entries.

Return value

Returns the length of the partition ID string on success. Returns 0 when the end of entries is reached.

Returns -1 times errno if an error occurred.

EBADF	*fd* is not a valid open descriptor.
EIO	I/O error.
ENOMEM	Not enough free memory.
ENOTDIR	*fd* is not a descriptor of the partition table.

format

Format hard disk drive

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	October 11, 2001

Syntax

```
#include <sys/mount.h>
int format(
 const char *devname,
 const char *blockdevname,
 void *arg,
 int arglen)
```

Device name (hdd0:, hdd1:)

Reserved. Specify NULL.

Reserved. Specify NULL.

Size of *arg*.

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Formats the hard disk drive with the specified unit number. The required partition is created in advance by the system.

Notes

For use only during title development and should not be incorporated within a title. Care should be taken when using this command as this operation initializes the information of all partitions on the disk.

Return value

0 if successful. -1 times errno if an error occurred.

EIO	I/O error.
EMFILE	Reached the maximum number of descriptors that can be opened.
ENODEV	Specified device does not exist.
ENXIO	Disk for the specified unit number does not exist.

getstat

Get partition status

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

```
#include <sys/stat.h>
int getstat(
 const char *name,
 struct sce_stat *buf)
```

name, — Partition identifier string. If a password has been set then minimally, the read-only password must be specified.

buf) — buffer for storing the status.

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Copies partition information to the sce_dirent structure buf.

Return value

Returns zero on success.

-1 times errno if an error occurred.

EACCES	No access rights.
EINVAL	Incorrect arguments were specified.
EIO	I/O error.
EMFILE	Reached the maximum number of descriptors that can be opened.
ENODEV	Specified device does not exist.
ENOMEM	Not enough free memory.

ioctl2

Special operations for a partition

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int ioctl2(
```

int *fd,*	Target file descriptor
long *cmd,*	Operation command. Any of the following constants can be specified. HIOCADDSUB HIOCDELSUB HIOCNSUB HIOCFLUSH
void **arg,*	Command arguments. Depends on *cmd*.
size_t *arglen,*	Size of *arg*.
void **bufp,*	Arguments received from the command. Depends on *cmd*.
size_t *buflen*)	Size of the *bufp*.

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Performs special operations on a partition. For details regarding each *cmd*, refer to the "Ioctl2 command table".

Return value

Returns a command-dependent value if successful.

-1 times errno if an error occurred.

The errors that are common to each of the commands are as follows.

EBADF	*fd* is not a valid open descriptor.
EINVAL	Incorrect arguments specified.

lseek

Move extended attribute area file pointer of partition

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

#include <stdio.h>

int lseek(

int *fd,*	File descriptor of partition for which the pointer will be moved
long *offset,*	Distance to move pointer (multiple of 512 bytes)
int *whence***)**	Reference position of *offset* in the extended attribute area of the partition. Any of the following constants can be specified. SEEK_SET Starting position SEEK_CUR Current position SEEK_END End

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

The offset of file descriptor *fd* is changed to the position specified by the *offset* argument, and according to *whence*. The offset cannot be set to a position exceeding the end of the extended attribute area of the partition.

Return value

On success, returns the updated value of the file pointer.

On error, returns -1 times errno.

EBADF	*fd* is not a valid open descriptor.
EINVAL	The specified size is not a multiple of 512. *whence* is an incorrect value or an *offset* beyond the EOF was specified.
EIO	I/O error.

open

Create, open main partition

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int open(
 const char *name,
 int flags)
```

Partition identifier string.

Access mode.

Any of the following constants can be specified.

A logical OR is performed if more than one is specified.

O_RDONLY	Open as read-only.
O_RDWR	Open as read/write.
O_CREAT	Create a new partition if one does not exist.

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Creates, opens the main partition. Assigns a file descriptor to the file that has been opened. Certain partitions cannot be opened simultaneously. The partition identifier string consists of device name + unit number + ':'+ a string made from a sequence of the following strings separated by commas.

Partition ID	This is a unique ID for the entire disk and is formally issued by SCE, however, any character string can be used as long as it is unique during the creation stage and is no more than 32 characters long.
Full password	The password required for read/write access. The password can be up to 8 characters long.
Read-only password	The password for read-only access. The password can be up to 8 characters long.
Partition size	Character string which specifies the size of the partition. The valid characters strings are shown below: 128M, 256M, 512M, 1G, 2G, 4G, 8G, 16G, 32G
Filesystem name	At present, only "PFS" is valid.

All of these need to be specified for creation, except for the passwords. To open a partition, specify up to the required password.

Example 1: Creation with password specifications

```
sceOpen("hdd0:BISLPS-XXXXX,fpwd,rpwd,128M,PFS",SCE_CREAT|SCE_RDWR);
```

Example 2: Creation without password specifications

```
sceOpen("hdd0:BISLPS-XXXXX,,,128M,PFS", SCE_CREAT|SCE_RDWR);
```

Example 3: Open with a password

```
sceOpen("hdd0:BISLPS-XXXXX,fpwd", SCE_RDWR);
```

Example 4: Open without a password

```
sceOpen("hdd0:BISLPS-XXXXX", SCE_RDWR);
```

Example 5: Open with a read-only password

```
sceOpen("hdd0:BISLPS-XXXXX,,rpwd", SCE_RDONLY);
```

Notes

If an opened partition is not closed before the filesystem driver performs a format or mount of the partition, then an EBUSY error is returned.

Return value

Returns the file descriptor on normal completion (value > 0).

-1 times errno if an error occurred.

EACCES	No access rights.
EBUSY	The specified partition is already open.
EINVAL	Incorrect arguments were specified.
EIO	I/O error.
EMFILE	Reached maximum number of descriptors that can be opened.
ENODEV	Specified device does not exist.
ENOENT	Specified partition does not exist.
ENOMEM	Not enough free memory.
ENOSPC	No free space.

read

Read from the extended attribute area of a partition

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int read(
 int fd,                    File descriptor of the read target
 void *buf,                 Address of the buffer that will store the read data
 size_t count)              Read data size (multiple of 512 bytes)
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Reads a maximum of *count* bytes from the extended attribute area of the partition that was previously opened, into the buffer starting from the address specified by *buf*. *count* must be a multiple of 512. Specifying any other value than this results in an error.

Return value

On success, the number of bytes read are returned. The file position is advanced by this amount only. A return value of 0 means end of file. If an error occurred, -1 times errno is returned.

EBADF	*fd* is not a valid open descriptor.
EINVAL	The specified size is not a multiple of 512.
EIO	I/O error.

remove

Delete partition

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int remove(
const char *name)
```

name — Partition identifier string. If a password is specified, specifications are required up to the full password.

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Deletes the specified partition. All sub-partitions that were added are also deleted.

Return value

Returns zero on success.

-1 times errno if an error occurred.

EACCES	No access rights.
EBUSY	The specified partition is already open.
EINVAL	Incorrect arguments were specified.
EIO	I/O error.
EMFILE	Reached maximum number of descriptors that can be opened.
ENODEV	Specified device does not exist.
ENOMEM	Not enough free memory.

write

Write to the extended attribute area of partition

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int write(
 int fd,                    File descriptor of the write target.
 const void *buf,           Address of the buffer that stores the write data
 size_t count)              Write data size (multiple of 512 bytes)
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Writes a maximum of *count* bytes from the buffer indicated by *buf* into the extended attribute area of the partition referenced by the file descriptor *fd*. *count* must be a multiple of 512. Specifying any other value than this results in an error.

Return value

On success, returns the number of bytes written. The file position is advance by this amount only.

If an error occurred, -1 times errno is returned.

EACCES	No write permission.
EBADF	*fd* is not a valid open descriptor.
EINVAL	The specified size is not a multiple of 512.
EIO	I/O error.

devctl Commands

HDIOC_DEV9OFF

Power OFF device

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	October 11, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Size of *arg*.
bufp	Reserved. Specify NULL.
buflen	Size of *bufp*.

Description

Powers off the entire dev9 device to which the hard disk drive is connected.

This processing should be performed before powering off the main unit.

Note: When this processing is performed, other devices connected to dev9 (the network adapter) are also powered off.

Return value

0 if successful.

HDIOC_FLUSH

Flush the disk cache

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

arg	Reserved. Specify NULL.
arglen	Size of *arg*.
bufp	Reserved. Specify NULL.
buflen	Size of *bufp*.

Description

Flushes the cache on the disk. Usually, the application is not required to perform this operation.

Return value

0 if successful.

-1 times errno if an error occurred.

EIO I/O error.

HDIOC_FORMATVER

Get partition system version

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

arg	Reserved. Specify NULL.
arglen	Size of *arg*.
bufp	Reserved. Specify NULL.
buflen	Size of *bufp*.

Description

Gets the version of the formatted partition system. Usually, the application is not required to verify the version.

Return value

Returns the version of the partition system.

HDIOC_FREESECTOR

Get installable size

Library	*Introduced*	*Documentation last modified*
hdd	2.3.1	July 27, 2001

Syntax

arg	Reserved. Specify NULL.
arglen	Size of *arg*.
bufp	Pointer to an unsigned 32-bit integer, for storing installable size, in sectors.
buflen	Size of *bufp*.

Description

Returns the installable size in *bufp*. The installable size will be equal to the disk's free space as indicated in the browser. All free space over 1 GB is added, and for free space less than 1 GB (512M, 256M and 128M) an addition is performed for up to one area, respectively. If there is more than one free area less than 1 GB, those areas will not be added as such, but as aggregates of smaller areas. However, in this case, any area that has been counted once will not be counted again.

For example, assume there are two areas of 512 MB and one of 128 MB. In this case, the first 512 MB area is simply counted; the second 512 MB area is counted as areas of 256 MB and 128 MB. Upon finding the next 128 MB area, since 128 MB has already been counted once, that area is not counted -it is ignored.

Note: As shown in the example above, the return value for this devctl command does not simply give the amount of free space. Rather, it may return a value that is smaller than the actual partition size that can be created. To find the actual partition size that can be created, the partition list and the capacity of the entire disk would need to be obtained.

Return value

Returns 0 if successful

On error, returns -1 times errno

EIO: input/output error.

HDIOC_IDLE

Set idle mode

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	October 11, 2001

Syntax

arg	Pointer to an 8-bit variable that stores the setting value.
arglen	Size of *arg*.
bufp	Reserved. Specify NULL.
buflen	Size of *bufp*.

Description

Sets the amount of time after which the hard disk drive will transition to idle mode.

The default time for the hdd module to transition to idle mode is 21 minutes and 15 seconds. The settable values are shown below.

0x00	timeout disable
0x01 - 0xf0	(value * 5) s
0xf1 - 0xfb	((value - 240) * 30) min
0xfc	21 min
0xfd	Period between 8 and 12 hours
0xfe	Reserved
0xff	21 min 15 s

Example:

```
u_char standbytimer = 0xff;
devctl("hdd0:", HDIOC_IDLE, &standbytimer, sizeof(char), NULL, 0);
```

Return value

0 if successful.

-1 times errno if an error occurred.

EIO	I/O error.

HDIOC_MAXSECTOR

Get maximum size of partition that can be created

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

arg	Reserved. Specify NULL.
arglen	Size of *arg*.
bufp	Reserved. Specify NULL.
buflen	Size of *bufp*.

Description

Gets the maximum size of a partition that can be created (in units of sectors).

Return value

Returns a value that is a power of 2 (2 ^ n) as an unsigned 32-bit integer.

HDIOC_SMARTSTAT

Check for hard disk drive failure

Library	*Introduced*	*Documentation last modified*
hdd	2.3	October 11, 2001

Syntax

arg	Reserved. Specify NULL.
arglen	Size of *arg*.
bufp	Reserved. Specify NULL.
buflen	Size of *bufp*.

Description

Checks for the presence of a failure using the hard disk drive SMART function.

Return value

Returns 0 if there is no failure and 1 if there is a failure.

In case of an error, -EIO is returned.

HDIOC_STATUS

Get hard disk drive status

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	January 4, 2002

Syntax

arg	Reserved. Specify NULL.
arglen	Size of *arg.*
bufp	Reserved. Specify NULL.
buflen	Size of *bufp*.

Description

Gets hard disk drive status.

Return value

Returns the following status:

3: Hard disk drive not connected.
2: (Reserved)
1: Not formatted.
0: Normal

HDIOC_SWAPTMP

Exchange partition information with _tmp

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

arg	Partition identifer string.
arglen	Size of *arg.*
bufp	Reserved. Specify NULL.
buflen	Size of *bufp.*

Description

Exchanges partition information with the _tmp partition.

Creates a new partition and copies the contents of the existing partition, then deletes the copy source. Can be used as a substitute for defrag, etc. Processing is similar to that of the filesystem rename(), etc., however, this command uses _tmp instead of creating a partition with a new partition ID.

Return value

0 if successful.

-1 times errno if an error occurred.

EACCES	No access rights.
EINVAL	Incorrect arguments were specified.
EIO	I/O error.
ENOMEM	Not enough free memory.

HDIOC_TOTALSECTOR

Get total number of sectors on the disk

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

arg	Reserved. Specify NULL.
arglen	Size of *arg.*
bufp	Reserved. Specify NULL.
buflen	Size of *bufp.*

Description

Gets total number of sectors on the disk.

Return value

Returns an unsigned 32-bit integer.

ioctl2 Commands

HIOCADDSUB

Add sub-partition

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

arg	Pointer to the partition size string.
arglen	Size of *arg*.
bufp	Reserved. Specify NULL.
buflen	Size of *bufp*.

Description

Adds a sub-partition.

Example:

```
char chsize[] = "128M";
ioctl2(fd, HIOCADDSUB, chsize, strlen(chsize)+1, NULL, 0);
```

Return value

0 if successful.

-1 times errno if an error occurred.

EACCES	No access rights.
EFBIG	Already reached number of sub-partitions that can be added.
EIO	I/O error.
ENOMEM	Not enough free memory.
ENOSPC	No free space.

HIOCDELSUB

Delete sub-partition

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

arg	Reserved. Specify NULL.
arglen	Size of *arg.*
bufp	Reserved. Specify NULL.
buflen	Size of *bufp.*

Description

Deletes a sub-partition. The sub-partition that was added last is deleted. If a filesystem has already been created in this partition and if a sub-partition is deleted without first reducing the size of the filesystem, then the filesystem will be destroyed. Usually, the application is not required to perform this operation directly.

Return value

Returns 0 on success.

-1 times errno if an error occurred.

EACCES	No access rights.
EIO	I/O error.
ENOENT	Partition to be deleted does not exist.
ENOMEM	Not enough free memory.

HIOCFLUSH

Flush the disk cache

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

arg	Reserved. Specify NULL.
arglen	Size of *arg*.
bufp	Reserved. Specify NULL.
buflen	Size of *bufp*.

Description

Flushes the cache on the disk. Usually, the application is not required to perform this operation as the disk cache is flushed appropriately by the filesystem.

Return value

0 if successful.

HIOCNSUB

Get number of sub-partitions

Library	*Introduced*	*Documentation last modified*
hdd	2.2.2	April 16, 2001

Syntax

arg	Reserved. Specify NULL.
arglen	Size of *arg*.
bufp	Reserved. Specify NULL.
buflen	Size of *bufp*.

Description

Gets number of sub-partitions that were added.

Return value

Returns number of sub-partitions on success.

Chapter 3: i.LINK Driver Library
Table of Contents

i.LINK Device Driver Operations

sce1394ChangeThreadPriority

Change priority of thread that is using the i.LINK driver

Library	*Introduced*	*Documentation last modified*
ilink	2.2	March 26, 2001

Syntax

int sce1394ChangeThreadPriority (

int *priority_hi,*	Thread priority (high)
int *priority_lo);*	Thread priority (low)

Calling conditions

Can be called from a thread

Multithread safe

Description

Changes the priority of the thread created by the i.LINK driver.

The value of *priority_hi* should be higher than that of priority_lo.

The settable priority values are in the range USER_HIGHEST_PRIORITY ~ USER_LOWEST_PRIORITY (including endpoint values). These are defined in thread.h.

Notes

The thread priority can also be specified when ilink.irx is loaded by using sceSifLoadModule(). As an example, if the loading is as shown below, then the thread priority of ilink.irx will be priority_hi =20, priority_hi =22.

```
unsigned char *param = "thpri=20,22";
sceSifLoadModule( "host0:/usr/local/sce/iop/modules/ilink.irx", s
trlen(param)+1, param);
```

Return value

KE_OK:	Success
KE_ILLEGAL_PRIORITY:	Incorrect thread priority was specified

sce1394ConfGet

Get i.LINK device driver operation parameter

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394ConfGet(

int *member,* Any of the values in the table below representing the parameter type.

...);

Calling conditions

Can be called from a thread

Multithread safe

Table 3-1

Value	Parameter type
SCE1394CF_PRIORITY_HI	Driver thread priority (high)
SCE1394CF_PRIORITY_LO	Driver thread priority (low)
SCE1394CF_TRSIZE_MASTER	Maximum payload size (bytes) of transactions to be sent
SCE1394CF_TRSIZE_SLAVE	Maximum payload size (bytes) of transactions to be received
SCE1394CF_NODE_CAPABILITY	Node capability

Description

Returns the current operations parameters of the i.LINK device driver.

Return value

Returns the current operation parameter of the i.LINK device driver. Either of the following error codes can be returned as an error:

- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_INVALID_ARGUMENT

sce1394ConfSet

Set i.LINK device driver operation parameter

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394ConfSet(

int *member,*	Any of the values in the table below representing the parameter type.
int *value);*	Value to be set as parameter.

Table 3-2

Value	Parameter type
SCE1394CF_PRIORITY_HI	Driver thread priority (high)
SCE1394 CF_PRIORITY_LO	Driver thread priority (low)
SCE1394CF_TRSIZE_MASTER	Maximum payload size (bytes) of transactions to be sent
SCE1394CF_TRSIZE_SLAVE	Maximum payload size (bytes) of transactions to be received
SCE1394CF_NODE_CAPABILITY	Node capability

Calling conditions

Can be called from a thread

Not multithread safe

Description

Sets an operation parameter of the i.LINK device driver. The initial value of the maximum payload size is 122. A specification of 0 means the initial value. If the specified value exceeds the values that can be handled by the hardware, SCE1394ERR_INVALID_ARGUMENT is returned. If a hardware resource is being used for another purpose, SCE1394ERR_RESOURCE_UNAVAILABLE is returned.

The node capability (irmc, cmc, isc, bmc, or pmc) is specified by the 5th byte of Bus_Info_Block in the configuration ROM. If an unimplemented feature is specified, an error will occur and the state will not change.

Currently, only cmc is implemented. If the node is the root and cmc is enabled, sending of the cycle start packet is started.

Return value

The value of the previous operation parameter is returned.

Any of the following error codes can be returned as an error:

- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_INVALID_ARGUMENT
- SCE1394ERR_RESOURCE_UNAVAILABLE

sce1394Destroy

Stop use of device driver

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394Destroy();

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Stops the use of the i.LINK device driver. Specifically, this function decrements the number of driver references by 1 and if this number is 0, it resets the device driver and releases resources.

Return value

The number of new driver references is returned.

SCE1394ERR_NOT_INITIALIZED can be returned as an error.

sce1394Initialize

Load and initialize i.LINK device driver

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394Initialize(

void **arg);* Must be set to NULL.

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Loads and initializes the i.LINK device driver. Specifically, this function increments the number of driver references by 1 and returns the previous number of references. If the i.LINK device driver has already been initialized, initialization processing is not performed.

Driver threads (multiple) are created with priorities of 28 or 34.

Immediately after initialization, packets cannot be sent or received until sce1394SbEnable() is called because the serial bus will have been disabled.

Return value

The previous number of driver references is returned.

Either of the following error codes can be returned as an error:

- SCE1394ERR_ERROR
- SCE1394ERR_NO_MEMORY

sce1394UnitAdd

Add unit directory

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394UnitAdd(

int *size,*	Number of quadlets of image data
u_int **dir);*	Unit directory image (host byte order)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Adds a unit directory to the root directory of Config-ROM and returns the identifier (id > 0).

Although the directory size must be included at the beginning of the image data, CRC is automatically generated.

Subdirectories may be included in the unit directory. The CRC of the subdirectories must be set in advance. The number of units and the image size are limited.

Return value

The unit directory identifier (id > 0) is returned.

Any of the following error codes can be returned as an error:

- SCE1394ERR_NO_MEMORY
- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_RESOURCE_UNAVAILABLE

sce1394UnitDelete

Delete unit directory

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

Sce1394ErrorCode sce1394UnitDelete(

int *id);* Unit directory identifier

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Deletes the unit directory specified by identifier *id*.

Return value

Any of the following error codes can be returned as an error:

- SCE1394ERR_OK
- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_INVALID_ID

SB_CONTROL.request

sce1394SbCycleTime

Get cycle time

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

u_int sce1394SbCycleTime(

u_int **bus_time);* Buffer for copying BUS_TIME value or NULL

Calling conditions

Can be called from a thread

Not multithread safe

Description

Returns (u_int)SCE1394ERR_RESOURCE_UNAVAILABLE if the node itself is not cycle-master and no cycle-start packet could be received since the previous call until the present.

Return value

This function gets the CYCLE_TIME CSR and BUS_TIME CSR values. If processing is normal, the value that is returned will not exceed 0xfff3fbff.

Either of the following error codes can be returned as an error:

- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_RESOURCE_UNAVAILABLE

sce1394SbDisable

Disable serial bus

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394SbDisable(

int *reset);*

Whether or not bus-reset is invoked

0: Do not invoke bus-reset

1: Invoke bus-reset

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Stops packet transmission/reception on the serial bus so that no replies are sent for packets from external sources. A bus-reset can be invoked at the same time according to the *reset* specification.

Return value

The previous bus state is returned (0: disabled, 1: enabled).

SCE1394ERR_NOT_INITIALIZED can be returned as an error.

sce1394SbEnable

Enable serial bus

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394SbEnable();

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Enables packet transmission/reception on the serial bus.

Immediately after the first time that sce1394Initialize() is executed, the serial bus is in a disabled state.

Return value

The previous bus state is returned (0: disabled, 1: enabled).

SCE1394ERR_NOT_INITIALIZED can be returned as an error.

sce1394SbEui64

Get EUI64 of local node

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

Sce1394ErrorCode sce1394SbEui64(

u_int **buff);*

2-quadlet buffer for returning the EUI64
buff[0] High-order 32 bits
buff[1] Low-order 32 bits

Calling conditions

Can be called from a thread

Multithread safe

Description

Gets the 64-bit Extended Unique ID of the local node itself and stores it in host byte order in the area pointed to by *buff*.

Return value

Any of the following error codes is returned:

- SCE1394ERR_OK
- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_INVALID_ARGUMENT

sce1394SbGenNumber

Get generation number

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394SbGenNumber();

Calling conditions

Can be called from a thread

Multithread safe

Description

Returns the generation number of the bus.

The generation number is a value that is incremented each time a bus-reset is detected. If the value changes before this function is called a second time, the bus configuration is changed and information that was obtained during this interval may be invalid.

Return value

The generation number (>= 0) is returned.

SCE1394ERR_NOT_INITIALIZED can be returned as an error.

sce1394SbNodeCount

Get number of nodes

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394SbNodeCount();

Calling conditions

Can be called from a thread

Multithread safe

Description

Returns the number of nodes, which could be obtained from the Self-ID data that was received immediately before this function was executed.

Return value

Returns the number of nodes. If Self-ID cannot be received, this function returns 0.

Also, SCE1394ERR_NOT_INITIALIZED can be returned as an error.

sce1394SbNodeID

Get node ID of local node

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394SbNodeID();

Calling conditions

Can be called from a thread

Multithread safe

Description

Returns the node ID (16-bit value formed by concatenating BUS-ID and PHY-ID) of the local node itself.

Return value

Returns the node ID of the local node itself.

SCE1394ERR_NOT_INITIALIZED can be returned as an error.

sce1394SbPhyPacket

Send PHY packet (unimplemented)

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394SbPhyPacket(

int *request,*	PHY packet type RAWPACKET LINKON GAPCOUNT ROOTHOLD
u_int *arg);*	Parameter included in PHY packet or PHY packet itself

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Sends a PHY packet.

Return value

Any of the following error codes can be returned:

- SCE1394ERR_OK
- SCE1394ERR_ERROR
- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_NOT_SUPPORTED

sce1394SbReset

Reset bus

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

Sce1394ErrorCode sce1394SbReset(

int *flag);* Must be set to 0.

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Requests a (short) bus-reset for PHY.

Return value

Either of the following error codes can be returned:

- SCE1394ERR_OK
- SCE1394ERR_NOT_INITIALIZED

sce1394SbSelfId

Get Self-ID

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394SbSelfId(

int *nquad,*	Buffer size (quadlet)
u_int **buff);*	Transfer destination buffer address (4-byte adjusted)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

Copies the Self-ID data that was received immediately before this function was executed to the buffer in host byte order and returns the actual number of quadlets in Self-ID.

Return value

Returns the number of quadlets in Self-ID. If Self-ID could not be received, this function returns 0.

Also, SCE1394ERR_INVALID_ARGUMENT can be returned as an error.

SB_EVENT.indication

sce1394EvAlloc

Create event buffer

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394EvAlloc();

Calling conditions

Can be called from a thread

Not multithread safe

Description

Creates an event buffer and returns its id (> 0).

Return value

Returns the id (> 0) of the event buffer that was created.

Any of the following error codes can be returned as an error:

- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_NO_MEMORY
- SCE1394ERR_RESOURCE_UNAVAILABLE

sce1394EvFree

Delete event buffer

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

Sce1394ErrorCode sce1394EvFree(

int *id);* Event buffer identifier

Calling conditions

Can be called from a thread

Not multithread safe

Description

Deletes an event buffer.

Return value

Any of the following error codes can be returned as an error:

- SCE1394ERR_OK
- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_INVALID_ID

sce1394EvPoll

Check for occurrence of event

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394EvPoll(

int *id,* Event buffer id

int *eventMask);* Type of event to check for (see table below)

Table 3-3

Event	Bit
SCE1394EV_BUS_RESET_START	1
SCE1394EV_BUS_RESET_COMPLETE	2
SCE1394EV_SELF_ID	3
SCE1394EV_COMMAND_RESET	4
SCE1394EV_BUS_OCCUPANCY_VIOLATION	8
SCE1394EV_CYCLE_TOO_LONG	9
SCE1394EV_CABLE_POWER_FAIL	10
SCE1394EV_DUPLICATE_CHANNEL	11
SCE1394EV_HEADER_CRC_ERROR	12
SCE1394EV_REQUEST_DATA_ERROR	13
SCE1394EV_RESPONSE_ACK_MISSING	14
SCE1394EV_RESPONSE_DATA_ERROR	15
SCE1394EV_RESPONSE_FORMAT_ERROR	16
SCE1394EV_RESPONSE_RETRY_FAILED	17
SCE1394EV_UNEXPECTED_CHANNEL	18
SCE1394EV_UNKNOWN_TCODE	19
SCE1394EV_UNSOLICITED_RESPONSE	20

Calling conditions

Can be called from a thread

Multithread safe

Description

Checks whether the specified event is occurring and returns a bit mask of the event that is occurring. If the specified event is not occurring, 0 is returned. The specified event is removed from the buffer. This function cannot be called for an event buffer that is already being blocked by the sce1394EvWait() function.

Return value

A bit mask representing the event that is occurring is returned. If the specified event is not occurring, 0 is returned. Any of the following error codes can be returned as an error:

- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_INVALID_ID
- SCE1394ERR_RESOURCE_UNAVAILABLE

sce1394EvWait

Wait for occurrence of event

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394EvWait(

int *id,* Event buffer id

int *eventMask);* Bit mask of event to wait for (see table below)

Table 3-4

Event	Bit
SCE1394EV_BUS_RESET_START	1
SCE1394EV_BUS_RESET_COMPLETE	2
SCE1394EV_SELF_ID	3
SCE1394EV_COMMAND_RESET	4
SCE1394EV_BUS_OCCUPANCY_VIOLATION	8
SCE1394EV_CYCLE_TOO_LONG	9
SCE1394EV_CABLE_POWER_FAIL	10
SCE1394EV_DUPLICATE_CHANNEL	11
SCE1394EV_HEADER_CRC_ERROR	12
SCE1394EV_REQUEST_DATA_ERROR	13
SCE1394EV_RESPONSE_ACK_MISSING	14
SCE1394EV_RESPONSE_DATA_ERROR	15
SCE1394EV_RESPONSE_FORMAT_ERROR	16
SCE1394EV_RESPONSE_RETRY_FAILED	17
SCE1394EV_UNEXPECTED_CHANNEL	18
SCE1394EV_UNKNOWN_TCODE	19
SCE1394EV_UNSOLICITED_RESPONSE	20

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Checks whether the specified event is occurring and returns a bit mask of the event that is occurring. If the specified event is not occurring, this function blocks the event buffer until the event occurs. The specified event is removed from the buffer. Multiple threads cannot wait simultaneously by using one event buffer.

Return value

A bit mask representing the event that is occurring is returned.

Any of the following error codes can be returned as an error:

- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_INVALID_ID
- SCE1394ERR_RESOURCE_UNAVAILABLE

TR_DATA.indication / TR_DATA.response

callbackFunc

Simple callback function

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int callbackFunc(

u_int *off,*	offset_L field value of request
u_int *size,*	length field value of request
u_int **payload,*	Data part starting address (network byte order) of request
int *pb);*	Packet block handle of request

Description

This is a simple callback function that is called from a callback-specific thread when a request packet is received.

This function must immediately return the required rcode.

Detailed information related to the request packet can be obtained by using the following API.

```
int sce1394PbGetGenNumber(int pb);
int sce1394PbGetSpeed(int pb);
int sce1394PbGetDest(int pb);
int sce1394PbGetSource(int pb);
int sce1394PbGetExTCode(int pb);
```

Return value

rcode that must be returned.

sce1394PbGet

Get detailed information related to request packet

Library	*Introduced*	*Documentation last modified*
ilink	2.3	July 2, 2001

Syntax

int sce1394PbGet(

int *pb,*	Packet block handle (passed to callbackFunc)
int *member,*	Any of the values below that represent a parameter class
... **);**	Pointer to destination storage location, etc. (if necessary)

Calling conditions

Can be called from a thread

Multithread safe

Can only be called within a callback function

Table 3-5

Parameter class value	Detailed information
SCE1394PB_BUFFER	System reserved
SCE1394PB_GENNUMBER	generation number
SCE1394PB_SIZE	System reserved
SCE1394PB_SPEED	Transfer rate (0:S100, 1:S200, 2:S400)
SCE1394PB_ACK	Ack code
SCE1394PB_STATUS	System reserved
SCE1394PB_EXTENSION	System reserved
SCE1394PB_TCODE	Transaction code
SCE1394PB_TLABEL	Transaction label
SCE1394PB_DEST	Target node ID (16-bits linking BUS-ID and PHY-ID)
SCE1394PB_SOURCE	Transmission source node ID (16-bits linking BUS-ID and PHY-ID)
SCE1394PB_OFFH	Upper target offset (16 bits)
SCE1394PB_OFFL	Lower target offset (32 bits) (value is sent to the location the 3rd argument points to)
SCE1394PB_LENGTH	Block payload size (bytes) (only valid for block write / lock)
SCE1394PB_EXTCODE	Extended transaction code
SCE1394PB_RCODE	System reserved
SCE1394PB_QUAD_PAYLOAD	Payload value (only valid for quadlet write) (value is sent to the location the 3rd argument points to)
SCE1394PB_BLOCK_PAYLOAD	Payload end address (only valid for block write / lock) (value is sent to the location the 3rd argument points to)

Description

Gets detailed information related to a receive request packet.

Can only be called within a callback function.

Return value

As a general rule, detailed information specified by the *member* argument is returned.

For classes where detailed information exceeds 31 bits, SCE1394ERR_OK is returned and the detailed information is stored at the location indicated by the pointer of the 3rd argument.

sce1394TrDataInd

Register callback function

Library	*Introduced*	*Documentation last modified*
ilink	1.6	July 2, 2001

Syntax

typedef int (*sce1394TrDataIndProc)
(int pb, void *arg);

int sce1394TrDataInd(

int *offH,*	offset_H field value of request to be detected
u_int *offL,*	Minimum value of offset_L field value of request to be detected
u_int *size,*	offset_L field range of request to be detected
sce1394TrDataIndProc *write,* **void** **wArg,*	NULL must be specified. Simple callback function
sce1394TrDataIndProc *read,* **void** **rArg,*	NULL must be specified. Simple callback function
sce1394TrDataIndProc *lock,* **void** **lArg);*	NULL must be specified. Simple callback function

Calling conditions

Can be called from a thread

Multithread safe

Description

Registers the callback function that is called when a request for the local node is received and returns the registration id (> 0). Specifying values for *offL* and *size* which exceed *offH* is not permitted.

For definitions of the callback functions, refer to the descriptions of callbackFunc().

In CSR space, etc., where the i.LINK driver responds by default, the space will be handled as if it were initially registered by this function. Thus, if the user later sets up a space that overlaps with this space, the user setting will be given priority.

sce1394TrDataInd() requires that all write/read/lock callback functions must be registered together. For example, passing a NULL for the lock argument indicates "invalid handler registered" rather than "no callback function registered". Thus, in normal usage, at least two callback functions should be specified for read and write.

Return value

The registration id (> 0) of the callback function is returned.

Any of the following error codes can be returned as an error:

- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_NO_MEMORY
- SCE1394ERR_RESOURCE_UNAVAILABLE
- SCE1394ERR_INVALID_ARGUMENT

sce1394TrDataUnInd

Delete callback function

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

Sce1394ErrorCode sce1394TrDataUnInd(

int *id);* Registration id

Calling conditions

Can be called from a thread

Multithread safe

Description

Deletes a registered callback function.

Return value

Any of the following error codes can be returned:

- SCE1394ERR_OK
- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_INVALID_ID

TR_DATA.request / TR_DATA.confirmation

sce1394TrAlloc

Create transaction handle

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394TrAlloc(

int *nodeId,*	target node ID (16-bit value formed by concatenating BUS-ID and PHY-ID)
int *mode);*	Must be set to 0.

Calling conditions

Can be called from a thread

Multithread safe

Description

Creates a transaction context and returns its identifier (transaction context handle). The transaction context includes the following kind of data.

generation number
target node ID (16-bit value formed by concatenating BUS-ID and PHY-ID)
transfer rate
end status
split transfer size
user data

The generation number is initialized using the value returned by sce1394SbGenNumber(). A value that can be used is set as the default for the transfer rate. The split transfer size is initialized to 0 (not split). The user data is initialized to NULL. Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

The created transaction context handle (> 0) is returned.

Any of the following error codes can be returned as an error:

- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_NO_MEMORY
- SCE1394ERR_INVALID_ARGUMENT
- SCE1394ERR_RESOURCE_UNAVAILABLE

sce1394TrFree

Release transaction context

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

Sce1394ErrorCode sce1394TrFree(

int *tc);* Transaction context handle

Calling conditions

Can be called from a thread

Multithread safe

Description

Releases the transaction context.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

Either of the following error codes can be returned as an error:

- SCE1394ERR_OK
- SCE1394ERR_INVALID_ID

sce1394TrGetBlockSize

Get block size

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394TrGetBlockSize(

int *tc);* Transaction context

Calling conditions

Can be called from a thread

Multithread safe

Description

Returns the split transfer size that is set in the transaction context.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

The split transfer size that is currently set is returned.

SCE1394ERR_INVALID_ID can be returned as an error.

sce1394TrGetDest

Get target node ID

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394TrGetDest(

int *tc);* Transaction context

Calling conditions

Can be called from a thread

Multithread safe

Description

Returns the target node ID that is set in the transaction context.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

The target node ID that is set in tc is returned.

SCE1394ERR_INVALID_ID can be returned as an error.

sce1394TrGetExtension

Get user data

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

Sce1394ErrorCode sce1394TrGetExtension(

int *tc,*	Transaction context
void ***extension);*	Pointer to area for storing obtained value

Calling conditions

Can be called from a thread

Multithread safe

Description

Gets the user data that is set in the transaction context.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

Any of the following error codes can be returned:

- SCE1394ERR_OK
- SCE1394ERR_INVALID_ID
- SCE1394ERR_INVALID_ARGUMENT

sce1394TrGetGenNumber

Get generation number

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394TrGetGenNumber(

int *tc);* Transaction context

Calling conditions

Can be called from a thread

Multithread safe

Description

Gets the generation number that is set in the transaction context.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

The generation number that is set in tc is returned.

SCE1394ERR_INVALID_ID can be returned as an error.

sce1394TrGetSpeed

Get transmission rate

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394TrGetSpeed(

int *tc);* Transaction context

Calling conditions

Can be called from a thread

Multithread safe

Description

Returns the transmission rate that is set in the transaction context.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

Any of the following values indicating the transmission rate can be returned.

Table 3-6

Value	Transmission rate
0	S100(100Mbit/sec)
1	S200(200Mbit/sec)
2	S400(400Mbit/sec)

SCE1394ERR_INVALID_ID can be returned as an error.

sce1394TrLock

Lock transaction

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394TrLock(

int *tc,*	Transaction context
int *offH,* **u_int** *offL,*	Target address offset
int *len,*	Number of transmit data bytes
void **buff,*	Transmit/receive data buffer address
int *ex_tCode);*	Lock type

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Starts up the lock transaction based on the parameters that are set in the transaction context *tc* and waits for the end of the transaction.

The data is treated as network byte order (BigEndian) data. If an attempt is made to start up the transaction when the generation number that is set in the transaction context does not match the value that is obtained by sce1394SbGenNumber(), the error SCE1394ERR_RESET_DETECTED will occur.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

The number of received payload bytes is returned.

Any of the following error codes can be returned as an error:

- SCE1394ERR_ERROR
- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_NOT_SUPPORTED
- SCE1394ERR_NO_MEMORY
- SCE1394ERR_RESOURCE_UNAVAILABLE
- SCE1394ERR_INVALID_ID
- SCE1394ERR_INVALID_ARGUMENT
- SCE1394ERR_INVALID_SIZE
- SCE1394ERR_INVALID_ADDRESS
- SCE1394ERR_TRANSACTION_ERROR
- SCE1394ERR_RESET_DETECTED
- SCE1394ERR_REQUEST_DISABLED
- SCE1394ERR_ERROR_RESPONSE
- SCE1394ERR_TIMEOUT
- SCE1394ERR_ACK_MISSING

- SCE1394ERR_RETRY_LIMIT
- SCE1394ERR_DATA_ERROR
- SCE1394ERR_INVALID_PARAMETER

sce1394TrRead / sce1394TrReadV

Read transaction

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

```
int sce1394TrRead(
 int tc,                          Transaction context
 int offH, u_int offL,            Target address offset
 int len,                         Number of read bytes
 void *buff );                    Data buffer address
int sce1394TrReadV(
 int tc,                          Transaction context
 int offH, u_int offL,            Target address offset
 int count,                       Number of vec elements
 sce1394Iov *vec );               Buffer list
                                  typedef sce1394Iov {
                                      int   iov_len;  //Number of bytes  (multiple of 4,
                                                        excluding last element)
                                      void  *iov_base; //Data buffer address (4-byte
                                                        alignment)
                                  } sce1394Iov;
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Starts up the read transaction based on the parameters that are set in the transaction context *tc* and waits for the end of the transaction.

The data is treated as network byte order (BigEndian) data. The sce1394TrRead() function sets the split transfer size as the maximum payload length, repeats the transaction until *len* bytes are reached, and returns the total number of transfer bytes. When the transfer is split, the target address offset is automatically incremented. However, values that would result in an advance to *offH* cannot be specified.

The sce1394TrReadV() function starts up a single transaction and transfers received data to non-contiguous data areas. It returns the number of received payload bytes. If an attempt is made to start up the transaction when the generation number that is set in the transaction context does not match the value that is obtained by sce1394SbGenNumber(), the error SCE1394ERR_RESET_DETECTED will occur.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

For sce1394TrRead(), the total number of transfer bytes is returned. For sce1394TrReadV(), the number of received payload bytes is returned.

Any of the following error codes can be returned as an error:

- SCE1394ERR_ERROR
- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_NOT_SUPPORTED
- SCE1394ERR_NO_MEMORY
- SCE1394ERR_RESOURCE_UNAVAILABLE
- SCE1394ERR_INVALID_ID
- SCE1394ERR_INVALID_ARGUMENT
- SCE1394ERR_INVALID_SIZE
- SCE1394ERR_INVALID_ADDRESS
- SCE1394ERR_TRANSACTION_ERROR
- SCE1394ERR_RESET_DETECTED
- SCE1394ERR_REQUEST_DISABLED
- SCE1394ERR_FAILED_RESPONSE
- SCE1394ERR_TIMEOUT
- SCE1394ERR_ACK_MISSING
- SCE1394ERR_RETRY_LIMIT
- SCE1394ERR_DATA_ERROR
- SCE1394ERR_INVALID_PARAMETER

sce1394TrSetBlockSize

Set block size

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394TrSetBlockSize(

int *tc,*	Transaction context
int *size);*	Split transfer size (bytes)

Calling conditions

Can be called from a thread

Multithread safe

Description

Sets the split transfer size in the transaction context and returns the value that had just been set.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

The split transfer size that had just been set is returned.

Either of the following error codes can be returned as an error:

- SCE1394ERR_INVALID_ID
- SCE1394ERR_INVALID_ARGUMENT

sce1394TrSetDest

Set target node ID

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394TrSetDest(

int *tc,*	Transaction context
int *nodeId* **);**	Target node ID

Calling conditions

Can be called from a thread

Multithread safe

Description

Sets the target node ID in the transaction context and returns the target node ID that had just been set.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

The target node ID that had just been set is returned.

Either of the following error codes can be returned as an error:

- SCE1394ERR_INVALID_ID
- SCE1394ERR_INVALID_ARGUMENT

sce1394TrSetExtension

Set user data

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

Sce1394ErrorCode sce1394TrSetExtension(

int *tc*,	Transaction context
void ***extension);*	Pointer to area for storing value that is set or obtained

Calling conditions

Can be called from a thread

Multithread safe

Description

Sets a pointer as user data in the transaction context and gets the value that had just been set.

The driver does not use the user data. Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

Any of the following error codes can be returned:

- SCE1394ERR_OK
- SCE1394ERR_INVALID_ID
- SCE1394ERR_INVALID_ARGUMENT

sce1394TrSetGenNumber

Set generation number

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394TrSetGenNumber(

int *tc,*	Transaction context
int *genNumber);*	Value to be set

Calling conditions

Can be called from a thread

Multithread safe

Description

Sets the generation number in the transaction context and returns the value that had just been set.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

The generation number that had just been set is returned.

Either of the following error codes can be returned as an error:

- SCE1394ERR_INVALID_ID
- SCE1394ERR_INVALID_ARGUMENT

sce1394TrSetSpeed

Set transmission rate

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394TrSetSpeed(

int *tc,*	Transaction context
int *speed);*	Any of the values representing the transmission rate (see table below)

Table 3-7

Value	Transmission rate
0	S100(100Mbit/sec)
1	S200(200Mbit/sec)
2	S400(400Mbit/sec)

Calling conditions

Can be called from a thread

Multithread safe

Description

Sets the transmission rate in the transaction context and returns the value that had just been set.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

The transmission rate that had just been set is returned.

Either of the following error codes can be returned as an error:

- SCE1394ERR_INVALID_ID
- SCE1394ERR_INVALID_ARGUMENT

sce1394TrStatus

Get response code and end status

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

Sce1394ErrorCode sce1394TrStatus(

int *tc,*	Transaction context
int **rCode);*	Pointer to area for storing rCode that had been included in preceding response, or NULL

Calling conditions

Can be called from a thread

Multithread safe

Description

Stores the response code of the preceding transaction in the area pointed to by *rCode* and returns the end status.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

Any of the following error codes can be returned as an error:

- SCE1394ERR_INVALID_ID
- SCE1394ERR_TRANSACTION_ERROR
- SCE1394ERR_RESET_DETECTED
- SCE1394ERR_REQUEST_DISABLED
- SCE1394ERR_FAILED_RESPONSE
- SCE1394ERR_TIMEOUT
- SCE1394ERR_ACK_MISSING
- SCE1394ERR_RETRY_LIMIT
- SCE1394ERR_DATA_ERROR
- SCE1394ERR_INVALID_PARAMETER

sce1394TrWrite / sce1394TrWriteV

Write transaction

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

```
int sce1394TrWrite(
 int tc,                          Transaction context
 int offH, u_int offL,            Target address offset
 int len,                         Number of write bytes
 void *buff );                    Data buffer address
int sce1394TrWriteV(
 int tc,                          Transaction context
 int offH, u_int offL,            Target address offset
 int count,                       Number of vec elements
 sce1394Iov *vec );               Buffer list
                                  typedef sce1394Iov {
                                      int    iov_len;   //Number of bytes  (multiple of 4,
                                                        excluding last element)
                                      void *iov_base; //Data buffer address (4-byte
                                                        alignment)
                                  } sce1394Iov;
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Starts up the write transaction based on the parameters that are set in the transaction context *tc* and waits for the end of the transaction. The data is treated as network byte order (BigEndian) data.

The sce1394TrWrite() function sets the split transfer size as the maximum payload length, repeats the transaction until *len* bytes are reached, and returns the total number of transfer bytes. When the transfer is split, the target address offset is automatically incremented. However, values that would result in an advance to *offH* cannot be specified.

The sce1394TrWriteV() function concatenates distributed data in memory and starts up a transaction as a single packet. It returns the number of concatenated payload bytes.

If an attempt is made to start up the transaction when the generation number that is set in the transaction context does not match the value that is obtained by sce1394SbGenNumber(), the error SCE1394ERR_RESET_DETECTED will occur.

Since the request that specified the transaction context is not reentrant, exclusive control must be performed to share the same transaction context among multiple threads.

Return value

For sce1394TrWrite(), the total number of transfer bytes is returned. For sce1394TrWriteV(), the number of concatenated payload bytes is returned.

Any of the following error codes can be returned as an error:

- SCE1394ERR_ERROR
- SCE1394ERR_NOT_INITIALIZED
- SCE1394ERR_NOT_SUPPORTED
- SCE1394ERR_NO_MEMORY
- SCE1394ERR_RESOURCE_UNAVAILABLE
- SCE1394ERR_INVALID_ID
- SCE1394ERR_INVALID_ARGUMENT
- SCE1394ERR_INVALID_SIZE
- SCE1394ERR_INVALID_ADDRESS
- SCE1394ERR_TRANSACTION_ERROR
- SCE1394ERR_RESET_DETECTED
- SCE1394ERR_REQUEST_DISABLED
- SCE1394ERR_FAILED_RESPONSE
- SCE1394ERR_TIMEOUT
- SCE1394ERR_ACK_MISSING
- SCE1394ERR_RETRY_LIMIT
- SCE1394ERR_DATA_ERROR
- SCE1394ERR_INVALID_PARAMETER

Config-ROM Access Support

sce1394CrCapability

Get node capability information

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394CrCapability(

int *nodeId);* — Node ID (16-bit value formed by concatenating BUS-ID and PHY-ID)

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Returns the 5th byte of Bus_Info_Block, which represents the functions implemented at the node.

If 0x3ff is specified in the high-order 10 bits (BUS-ID) of *nodeId*, the cached data is returned. If 0x3f is specified in the low-order 6 bits (PHY-ID), this will be a specification of the local node.

Return value

The specified node capability is returned.

SCE1394ERR_ERROR can be returned as an error.

sce1394CrEui64

Get EUI64

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

Sce1394ErrorCode sce1394CrEui64(

int *nodeId,*	Node ID (16-bit value formed by concatenating BUS-ID and PHY-ID)
u_int **eui64);*	8-byte buffer for returning EUI64

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Gets the EUI64 of the node specified by *nodeId* and stores it in host byte order in the area pointed to by *eui64*. If 0xffff is specified for *nodeId*, this function gets the EUI64 of the local node itself.

Return value

Either of the following error codes can be returned as an error:

- SCE1394ERR_ERROR
- SCE1394ERR_INVALID_ARGUMENT

sce1394CrFindNode

Find node ID

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394CrFindNode(

u_int ****eui64* **); Buffer for maintaining the EUI64 (host byte order) of the node

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Returns the node ID (16-bit value formed by concatenating BUS-ID and PHY-ID) of the node that has the 64-bit Extended Unique ID specified by *eui64*.

Return value

The node ID of the relevant node is returned.

SCE1394ERR_ERROR can be returned as an error.

sce1394CrFindUnit

Find unit directory

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394CrFindUnit(

int *unit,*	Sequence number
int *specId,*	Unit_Spec_ID
int *version,*	Unit_SW_Version
u_int **offset);*	Directory offset (quadlet)

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Finds the directory for which the combination of the entries Unit_Spec_ID and Unit_SW_Version match *specID* and *version* and returns the node ID (16-bit value formed by concatenating BUS-ID and PHY-ID) of the node where the directory that was found in the sequence number position indicated by unit resides.

The offset from the beginning of Config-ROM of the directory that was found is set for *offset*.

Return value

The node ID of the relevant node is returned.

SCE1394ERR_ERROR can be returned as an error.

sce1394CrGenNumber

Get generation number of Config-ROM information

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394CrGenNumber();

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Gets the generation number of the Config-ROM information inside the driver.

If the generation number had been changed, the driver is reconfigured.

Return value

This function gets the generation number of the Config-ROM information.

SCE1394ERR_ERROR can be returned as an error.

sce1394CrInvalidate

Clear Config-ROM information cache

Library	*Introduced*	*Documentation last modified*
ilink	2.2	March 26, 2001

Syntax

int sce1394CrInvalidate();

Calling conditions

Can be called from a thread

Multithread safe

Description

Clears the Config-ROM information cache (related to other nodes) within the driver.
(Not necessary for normal programming. Used in debugging)

Return value

Returns the generation number of the Config-ROM information that has been invalidated.

sce1394CrMaxRec

Get max_rec information

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394CrMaxRec(

int *nodeId);* — Node ID (16-bit value formed by concatenating BUS-ID and PHY-ID)

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Gets the max_rec information of the node specified by *nodeId*, converts it to a number of bytes, and returns the result.

If 0x3ff is specified in the high-order 10 bits (BUS-ID) of *nodeId*, the cached data is returned. If 0x3f is specified in the low-order 6 bits (PHY-ID), this will be a specification of the local node.

Return value

The max_rec information of the specified node is converted to a number of bytes and returned.

- SCE1394ERR_ERROR can be returned as an error.

sce1394CrMaxSpeed

Get maximum transfer rate

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394CrMaxSpeed(

int *nodeId);* — Node ID (16-bit value formed by concatenating BUS-ID and PHY-ID)

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Returns the maximum applicable transfer rate for the node specified by *nodeID*.

If 0x3ff is specified in the high-order 10 bits (BUS-ID) of *nodeId*, the cached data is returned. If 0x3f is specified in the low-order 6 bits (PHY-ID), this will be a specification of the local node.

Return value

Any of the following values indicating the maximum transfer rate of the specified node can be returned.

Table 3-8

Value	Transmission rate
0	S100(100Mbit/sec)
1	S200(200Mbit/sec)
2	S400(400Mbit/sec)

SCE1394ERR_ERROR can be returned as an error.

sce1394CrRead

Get Config-ROM contents

Library	*Introduced*	*Documentation last modified*
ilink	1.6	March 26, 2001

Syntax

int sce1394CrRead(

int *nodeId,*	Node ID (16-bit value formed by concatenating BUS-ID and PHY-ID)
u_int *off,*	Offset (quadlet) from beginning of Config-ROM
int *nquad,*	Transfer size (quadlet)
u_int **buff);*	Buffer address (4-byte alignment)

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Gets the Config-ROM contents of the node specified by *nodeId* and returns the number of quadlets that were transferred from that node.

The data order is converted to host byte order.

If 0x3ff is specified in the high-order 10 bits (BUS-ID) of *nodeId*, the cached data is returned. If 0x3f is specified in the low-order 6 bits (PHY-ID), this will be a specification of the local node.

Return value

The number of quadlets that were transferred from the node is returned.

SCE1394ERR_ERROR can be returned as an error.

Chapter 4: i.LINK Socket Library
Table of Contents

Structures

sceILsock_addr

Socket address

Library	*Introduced*	*Documentation last modified*
ilsock	1.6	March 26, 2001

Structure

```
typedef unsigned int sceILsock_addr_t;
typedef unsigned short sceILsock_port_t;
typedef struct sceILsock_addr {
    unsigned char  sock_len;
    unsigned char  sock_family;
    sceILsock_port_t  sock_port;
    struct  eui64 {
            sceILsock_addr_t  eui64_hi;
            sceILsock_addr_t  eui64_lo;
    } sock_addr;
    char  sock_zero [4];
};
```

sock_len; — Address structure size

sizeof(struct sceILsock_addr)

sock_family; — Address family

Only SCEILSOCK_AF can be specified.

sock_port; — Port number

Values from 0 to 1024 are reserved for use by system services, etc.

Values from SCEILSOCK_PORT_ANONMIN to SCEILSOCK_PORT _ANONMAX are temporarily used by the system.

} *sock_addr;* — Node unique ID (64 bits)

SCEILSOCK_ADDR_ANY_HI, SCEILSOCK_ADDR_ANY_LO:

Indicates the node's own node unique ID.

SCEILSOCK_ADDR_BROADCAST_HI,

SCEILSOCK_ADDR_BROADCAST_LO:

Indicates a broadcast during transmission.

For members that are at least two bytes in size, both of these types of IDs must be set using network byte order (big-endian).

Functions

sceILsockBind

Assign address to socket

Library	*Introduced*	*Documentation last modified*
ilsock	1.6	March 26, 2001

Syntax

enum sceILsockErrorCode sceILsockBind(

int *sock,*	Descriptor obtained by using sceILsockOpen().
struct sceILsock_addr **name,*	Pointer to address structure.
int *namelen);*	Size of address structure.

Calling conditions

Can be called from a thread

Multithread safe

Description

Assigns an address to socket. This enables the socket to receive datagrams sent to this address from other sockets. This address is also used as the source address when sending datagrams.

- When 0 is specified for sock_port

  An unused port number between SCEILSOCK_PORT_ANONMIN and SCEILSOCK_PORT_ANONMAX is found and treated as if it were the specified value.
- When sceILsockBind() is omitted

  Each time sceILsockSend() or sceILsockSendTo() is called, an unused port number between SCEILSOCK_PORT_ANONMIN and SCEILSOCK_PORT_ANONMAX is found and used as the port number of the sender.
- If the specified port number is already being used, an error will occur.

The order in which sceILsockBind and sceILsockConnect appear is irrelevant. Only SCEILSOCK_ADDR_ANY_HI/LO or the socket's own eui64 can be specified. SCEILSOCK_ADDR_ANY_HI/LO is interpreted as the socket's own eui64.

Return value

Any of the following values may be returned.

- SCEILSOCKERR_OK
- SCEILSOCKERR_NOT_INITIALIZED
- SCEILSOCKERR_INVALID_ID
- SCEILSOCKERR_INVALID_REQUEST

sceILsockClose

Close socket

Library	*Introduced*	*Documentation last modified*
ilsock	1.6	March 26, 2001

Syntax

enum sceILsockErrorCode sceILsockClose(

int *sock);* Descriptor obtained by using sceILsockOpen()

Calling conditions

Can be called from a thread

Multithread safe

Description

Closes the communication socket and releases resources.

Return value

Any of the following values may be returned.

- SCEILSOCKERR_OK
- SCEILSOCKERR_NOT_INITIALIZED
- SCEILSOCKERR_INVALID_ID

sceILsockConnect

Set address of send/receive target

Library	*Introduced*	*Documentation last modified*
ilsock	1.6	March 26, 2001

Syntax

enum sceILsockErrorCode sceILsockConnect(

int *sock,*	Descriptor obtained by using sceILsockOpen().
struct sceILsock_addr **name,*	Pointer to the name structure. Copied internally.
int *namelen);*	Size of name structure.

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

For sceILsockSend(), sets the destination of the datagrams that are to be sent from this socket, and for sceILsockRecv(), sets the source of the datagrams that are to be received by this socket.

If SCEILSOCK_AF_UNSPEC is specified for sock_family of name, the connect state is canceled.

Return value

Any of the following values may be returned.

- SCEILSOCKERR_OK
- SCEILSOCKERR_NOT_INITIALIZED
- SCEILSOCKERR_INVALID_ID
- SCEILSOCKERR_INVALID_ARGUMENT
- SCE1394ERR_...
- KE_...

sceILsockInit

Initialize socket driver

Library	*Introduced*	*Documentation last modified*
ilsock	1.6	March 26, 2001

Syntax

enum sceILsockErrorCode sceILsockInit(

int *maxsock,*	Number of sockets that can be open simultaneously.
int *maxsize);*	Maximum size (bytes) of datagrams to be handled.

Calling conditions

Can be called from a thread

Multithread safe

Description

Initializes the socket driver.

Reserves memory to be used by the driver and creates event flags, etc. The second or subsequent time this function is called, it returns SCEILSOCKERR_OK without doing anything. If 0 is specified for maxsize, SCEILSOCK_MAX_PAYLOAD_SIZE_DEFAULT(=44) is assumed to have been specified. If a negative number is specified, SCEILSOCK_MAX_PAYLOAD_SIZE(=492) is assumed to have been specified.

Return value

Any of the following values may be returned.

- SCEILSOCKERR_OK
- SCEILSOCKERR_NO_MEMORY
- SCE1394ERR_...
- KE_...

sceILsockOpen

Create socket

Library	*Introduced*	*Documentation last modified*
ilsock	1.6	March 26, 2001

Syntax

int sceILsockOpen(

int *domain,*	Communication domain Only SCEILSOCK_PF can be specified.
int *type,*	Communication semantics Only SCEILSOCK_DGRAM can be specified. The maximum size of a datagram is the value specified by the sceILsockInit() function. Although factors such as the sequence or arrival of datagrams are not guaranteed, the absence of data corruption is guaranteed. (This does not hold for a global network. For a home network, both the sequence and arrival of datagrams should be considered to be guaranteed.)
int *protocol);*	Protocol number Only 0 can be specified.

Calling conditions

Can be called from a thread

Multithread safe

Description

Creates a communication socket and returns a descriptor (>=0). Since operations for descriptors are not reentrant, sockets having the same descriptor cannot be shared among multiple threads. If sharing is necessary, operations must be performed exclusively. Sending and receiving by a single thread can share a socket.

Return value

Any of the following values may be returned.

- SCEILSOCKERR_NOT_INITIALIZED
- SCEILSOCKERR_NO_MEMORY
- SCEILSOCKERR_RESOURCE_UNAVAILABLE
- SCEILSOCKERR_INVALID_ARGUMENT
- SCE1394ERR_...
- KE_...

scelLsockRecv

Receive data

Library	*Introduced*	*Documentation last modified*
ilsock	1.6	March 26, 2001

Syntax

int scelLsockRecv(

int *sock,*	Descriptor obtained by using scelLsockOpen().
char **buf,*	Starting address of transfer destination buffer.
int *len,*	Number of bytes in transfer destination buffer.
int *flags);*	Specify 0.

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Extracts datagrams that were received for socket and returns the number of bytes.

When scelLsockConnect() has already been executed, receives only data from the relevant address.

Any portion that does not fit in the buffer is discarded.

The byte order of the received data is not changed.

If no data is received, blocking is performed until data is received.

Return value

Any of the following values may be returned.

- SCEILSOCKERR_NOT_INITIALIZED
- SCEILSOCKERR_NO_MEMORY
- SCEILSOCKERR_INVALID_ID
- SCEILSOCKERR_INVALID_ARGUMENT
- KE_...

sceILsockRecvFrom

Receive data and get target address

Library	*Introduced*	*Documentation last modified*
ilsock	1.6	March 26, 2001

Syntax

int sceILsockRecvFrom(

int *sock,*	Descriptor obtained by using sceILsockOpen().
char **buf,*	Starting address of transfer destination buffer.
int *len,*	Number of bytes in transfer destination buffer.
int *flags,*	Specify 0.
struct sceILsock_addr **from,*	Starting address of target's address buffer.
int **fromlen);*	Address of variable for specifying the number of bytes in the address buffer. On return, the valid number of bytes in the receive address is returned.

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Extracts received datagrams and the sending address and returns the number of transferred bytes. When sceILsockConnect() has already been executed, receives only data from the relevant address.

Any portion that does not fit in the buffer is discarded.

The byte order of the received data is not changed.

If no data is received, blocking is performed until data is received.

Return value

Any of the following values may be returned.

- SCEILSOCKERR_NOT_INITIALIZED
- SCEILSOCKERR_NO_MEMORY
- SCEILSOCKERR_INVALID_ID
- SCEILSOCKERR_INVALID_ARGUMENT
- KE_...

sceILsockReset

Reset socket driver

Library	*Introduced*	*Documentation last modified*
ilsock	1.6	March 26, 2001

Syntax

void sceILsockReset(void);

Calling conditions

Can be called from a thread

Multithread safe

Description

Makes socket driver unavailable.

Releases memory, event flags, and other resources that were being used by the driver.

Open sockets are closed.

Return value

None

sceILsockSend

Send data

Library	*Introduced*	*Documentation last modified*
ilsock	1.6	March 26, 2001

Syntax

int sceILsockSend(

int *sock,*	Descriptor obtained by using sceILsockOpen().
char **buf,*	Starting address of buffer for datagrams to be sent.
int *len,*	Number of bytes in datagrams to be sent.
int *flags);*	Specify 0.

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Sends datagrams from a socket for which sceILsockConnect() has been executed and returns the number of bytes that were sent.

The byte order of the data that is to be sent is interpreted as network byte order.

For broadcasting, if a bus-reset occurs during transmission, an error is returned.

When either ack_data_error or ack_busy is detected, up to 32 retries are attempted.

Return value

Any of the following values may be returned.

- SCEILSOCKERR_NOT_INITIALIZED
- SCEILSOCKERR_NO_MEMORY
- SCEILSOCKERR_INVALID_ID
- SCEILSOCKERR_INVALID_ARGUMENT
- SCEILSOCKERR_INVALID_REQUEST
- SCEILSOCKERR_INVALID_SIZE
- SCEILSOCKERR_NO_SUCH_NODE
- SCE1394ERR_...
- KE_...

sceILsockSendTo

Specify target address and send data

Library	*Introduced*	*Documentation last modified*
ilsock	1.6	March 26, 2001

Syntax

int sceILsockSendTo(

int *sock,*	Descriptor obtained by using sceILsockOpen().
char **buf,*	Starting address of buffer for datagrams to be sent.
int *len,*	Number of bytes in datagrams to be sent.
int *flags,*	Specify 0.
struct sceILsock_addr **to,*	Address of target's address structure.
int *tolen);*	Number of bytes in address structure.

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Specifies the target's address, sends datagrams from socket, and returns the number of bytes that were sent.

The byte order of the data that is to be sent is interpreted as network byte order.

For broadcasting, if a bus-reset occurs during transmission, an error is returned.

When either ack_data_error or ack_busy is detected, up to 32 retries are attempted.

Return value

Any of the following values may be returned.

- SCEILSOCKERR_NOT_INITIALIZED
- SCEILSOCKERR_NO_MEMORY
- SCEILSOCKERR_INVALID_ID
- SCEILSOCKERR_INVALID_ARGUMENT
- SCEILSOCKERR_INVALID_REQUEST
- SCEILSOCKERR_INVALID_SIZE
- SCEILSOCKERR_NO_SUCH_NODE
- SCE1394ERR_...
- KE_...

Chapter 5: PlayStation File System (for IOP)
Table of Contents

Structures

sce_dirent

Directory entry

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Structure

struct sce_dirent {

struct sce_stat *d_stat;*	File status
char *d_name***[256];**	Filename
void ***d_private*};**	Reserved

Description

This structure stores a directory entry.

See also

dread()

sce_stat

File status

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Structure

struct sce_stat {

unsigned int *st_mode;* — File mode

- bit 0 Execute permission (other)
- bit 1 Write permission (other)
- bit 2 Read permission (other)
- bit 3 Execute permission (group)
- bit 4 Write permission (group)
- bit 5 Read permission (group)
- bit 6 Execute permission (user)
- bit 7 Write permission (user)
- bit 8 Read permission (user)
- bit 9 Reserved
- bit10 Reserved
- bit11 Reserved
- bit12-15 File type
  - 1 Directory
  - 2 Normal file
  - 4 Symbolic link

unsigned int *st_attr;* — Flag compatible with memory card mode

unsigned int *st_size;*

unsigned char *st_ctime***[8];** — Creation time

unsigned char *st_atime***[8];** — This field is updated at the same time as last access time and last update time.

unsigned char *st_mtime***[8];** — Last update time

- byte0 Reserved
- byte1 Seconds
- byte2 Minutes
- byte3 Hours
- byte4 Day
- byte5 Month
- byte6-7 (4 digits)

unsigned int *st_hisize;* — File size (64 bit)

unsigned int *st_private***[6]};** — word0 uid

word1 gid

word2 Number of zones used by the file

Description

This structure stores file status.

See also

struct sce_dirent

getstat()

Functions

chdir

Change current directory

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int chdir(
 const char *name)          File path name
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Changes current directory.

Return value

Returns zero on success. If an error occurred, returns -1 times errno.

EACCES	No access permission.
EIO	I/O error.
ELOOP	Too many symbolic links encountered when resolving the path name.
EMFILE	Reached the maximum number of descriptors that can be opened.
ENAMETOOLONG	File path name is too long.
ENODEV	Specified device does not exist.
ENOENT	Specified file does not exist.
ENOMEM	Not enough free memory.
ENOTDIR	*name* is not a directory.

chstat

Change status of file/directory

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <sys/stat.h>
int chstat(
 const char *name,
 struct sce_stat *buf,
 unsigned int cbit)
```

name: File path name (including device name + ':')

buf: Buffer for storing the status

cbit: Macro specifying the field to be changed. Any of the following constants can be specified.

- SCE_CST_MODE
- SCE_CST_ATTR
- SCE_CST_SIZE
- SCE_CST_CT
- SCE_CST_AT
- SCE_CST_MT
- SCE_CST_PRVT

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Changes the status of the specified file/directory. The members of the sce_stat structure that can be changed by this function are: bits except for the file type of the file mode and each time, bits except for the subdirectory bit of the memory card compatibility flag and the close completion flag.

Return value

Returns zero on success. If an error occurred, returns -1 times errno.

EACCES	No access permission.
EIO	I/O error.
ELOOP	Too many symbolic links encountered when resolving the path name.
EMFILE	Reached the maximum number of descriptors that can be opened.
ENAMETOOLONG	File path name is too long.
ENODEV	Specified device does not exist.
ENOENT	Specified file does not exist.
ENOMEM	Not enough free memory.
EROFS	Write access was requested for a file from a read-only filesystem.

close

Close file

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int close(
 int fd)                    Previously opened file descriptor
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Closes an open file and frees the file descriptor.

Return value

Returns 0 on success. On error, returns -1 times errno.

EBADF *fd* is not a valid open descriptor.

dclose

Close directory

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

#include <dirent.h>

int dclose(

int *fd***)** File descriptor

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Closes an open directory and frees the file descriptor.

Return value

Returns zero on success. On error returns -1 times errno.

EBADF *fd* is not a valid open descriptor.

devctl

Special operations for a device

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

#include <sys/mount.h>

int devctl(

const char **name,*	Filesystem device name
int *cmd,*	Operation command. Any of the following constants can be specified. PDIOC_ZONESZ PDIOC_ZONEFREE PDIOC_CLOSEALL
void **arg,*	Command arguments. Depends upon *cmd*.
int *arglen,*	Size of *arg*
void **bufp,*	Arguments received from command. Depends upon *cmd*.
size_t *buflen***)**	Size of *bufp*

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Performs special operations for a device. For details regarding each of the commands, refer to the "devctl command list."

Return value

If successful, returns a command-dependent value.

If an error occured, returns -1 times errno.

The errors that are common to each of the commands are as follows.

EINVAL	A non-existent *cmd* was specified.
EMFILE	Reached the maximum number of descriptors that can be opened.
ENODEV	Specified device does not exist.

dopen

Open a directory

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <dirent.h>
int dopen(
 const char *name)          Directory path name(including device name + ':')
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Opens a directory. Assigns a file descriptor to the open directory. Directory path name is "pfs" + unit number + ':' + character string.

Return value

Returns file descriptor on normal completion (value > 0). Returns -1 times errno if an error occurred.

EACCES	No access permission.
EIO	I/O error.
ELOOP	Too many symbolic links encountered when resolving the path name.
EMFILE	Reached maximum number of descriptors that can be opened.
ENAMETOOLONG	File path name is too long.
ENODEV	Specified device does not exist.
ENOENT	Specified directory not found.
ENOMEM	Not enough free memory.
ENOTDIR	Specified file is not a directory.

dread

Read directory entry

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <dirent.h>
int dread(
 int fd,                          File descriptor
 struct sce_dirent *buf)          Address of the buffer that stores the data that was read.
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

The next entry from the directory entry stream indicated by *fd* is copied to the sce_dirent structure *buf*. Returns 0 when the end of entries is reached.

Return value

Returns the length of the filename on success. Returns 0 when the end of entries is reached.

Returns -1 times errno if an error occurred.

EBADF	*fd* is not a valid open descriptor.
EIO	I/O error.
ENOMEM	Not enough free memory.
ENOTDIR	*fd* is not a descriptor for a directory.

format

Format filesystem

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	July 2, 2001

Syntax

```
#include <sys/mount.h>
int format(
```

const char *****_devname,_	Filesystem device name (pfs:)
const char *****_blockdevname,_	Block device name of partition created in advance. (Example: 'hdd0:BISLPS-XXXXXX,fpwd")
void *****_arg,_	Pointer to zone size variable and fragment option.
int _arglen_**)**	Size of _arg_.

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Builds a new filesystem. The specified zone size must be a power of 2 (2 ^ n) and in the range 2K - 128K. Efficiency will be improved if the zone size is set to a smaller value if most of the files to be created are small, and to a larger value if most of the files to be created are large.

Example:

```
int zonesz = 8192;
format("pfs:", "hdd0:BISLPS-XXXXXX,fpwd", &zonesz, sizeof(int));
```

In addition, formatting can also be performed if fragmentation was intentionally done for verification purposes during development.

```
int arg[3];
arg[0] = 8192;              // zone size
arg[1] = 0x00002d66;        // -f
arg[2] = 0x01030f0f;        // fragment bit pattern
sceFormat("pfs:", "hdd0:test", &arg, sizeof(arg));
```

Each bit of the bit pattern corresponds to a zone. For example, if 0x0f0f0f0f is specified, formatting will be performed with a repeated pattern in which four zones are used and four zones are empty.

Notes

Note that when this operation is performed, a filesystem previously created on the specified partition will be initialized.

Return value

On success, returns 0. On error, returns -1 times errno.

EACCES	No access permission.
EBUSY	Specified partition is already open.
EINVAL	An invalid argument was specified.
EIO	I/O error.

EMFILE	Reached the maximum number of descriptors that can be opened.
ENODEV	Specified device does not exist.
ENOENT	Specified partition does not exist.
ENOMEM	Not enough free memory.
ENXIO	Not a supported device.

getstat

Get file/directory status

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <sys/stat.h>
int getstat(
 const char *name,                File path name (including device name + ':')
 struct sce_stat *buf)            Buffer for storing the status
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Copies file information to the sce_dirent structure *buf*. The file path name is "pfs" + unit number + ':' + character string.

Return value

On success, returns zero. On error, returns -1 times errno.

EACCES	No access permission.
EIO	I/O error.
ELOOP	Too many symbolic links encountered when resolving the path name.
EMFILE	Reached the maximum number of descriptors that can be opened.
ENAMETOOLONG	File path name is too long.
ENODEV	Specified device does not exist.
ENOENT	Specified file does not exist.
ENOMEM	Not enough free memory.

ioctl2

Special operations for a file

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

#include <stdio.h>

int ioctl2(

int *fd,*	Target file descriptor
long *cmd,*	Operation command. Any of the following constants can be specified. PIOCALLOC PIOCFREE PIOCATTRADD PIOCATTRDEL PIOCATTRLOOKUP PIOCATTRREAD
**void ****arg,*	Command arguments. Depends on *cmd*.
size_t *arglen,*	Size of *arg.*
**void ****bufp,*	Arguments received from the command. Depends on *cmd*.
size_t *buflen***)**	Size of the *bufp*.

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Performs special operations on a partition. For details regarding each of the commands, refer to the "ioctl2 command list."

Return value

Returns a command-dependent value if successful.

-1 times errno if an error occurred.

The errors that are common to each of the commands are as follows.

EBADF	*fd* is not a valid open descriptor.
EINVAL	Specified command not found.

lseek

Move file pointer

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

#include <stdio.h>

int lseek(

int *fd,*	File descriptor
long *offset,*	Distance to move pointer
int *whence***)**	Reference position of *offset* in the extended attribute area of the partition.
	Any of the following constants can be specified.
	SEEK_SET Starting position
	SEEK_CUR Current position
	SEEK_END End

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

The offset of file descriptor *fd* is changed to the position specified by the *offset* argument, and according to *whence*. The offset cannot be set to a position exceeding the end of file.

Return value

On success, returns the new setting of the file pointer.

On error, returns -1 times errno.

EBADF	*fd* is not a valid open descriptor.
EINVAL	whence is an incorrect value or an offset beyond the EOF was specified.
EIO	I/O error.
EISDIR	The request was made for a directory.

lseek64

Move file pointer (64-bit compatible)

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int lseek64(
 int fd,
 long long offset,
 int whence)
```

fd — File descriptor

offset — Distance to move pointer

whence — Reference position of offset in the extended attribute area of the partition. Any of the following constants can be specified:

SEEK_SET	Starting position
SEEK_CUR	Current position
SEEK_END	End

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

The offset of file descriptor *fd* is changed to the position specified by the offset argument, and according to *whence*. The offset cannot be set to a position exceeding the end of file. This function supports a 64-bit file size.

Return value

On success, returns the new setting of the file pointer.

On error, returns -1 times errno.

EBADF	*fd* is not a valid open descriptor.
EINVAL	whence is an incorrect value or an offset beyond the EOF was specified.
EIO	I/O error.
EISDIR	The request was made for a directory.

mkdir

Create directory

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int mkdir(
 const char *name,
 int mode)
```

name: Directory path name (including device name + ':')

mode: File mode

- bit 0 Execute permission (other)
- bit 1 Write permission (other)
- bit 2 Read permission (other)
- bit 3 Execute permission (group)
- bit 4 Write permission (group)
- bit 5 Read permission (group)
- bit 6 Execute permission (user)
- bit 7 Write permission (user)
- bit 8 Read permission (user)
- bit 9 Reserved
- bit10 Reserved
- bit11 Reserved

Macros for each mode are provided in sys/file.h. However, using octal codes such as 0777, 0755, etc. is also an easy way to specify the mode.

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Creates a directory. The path name is "pfs" + unit number + ':' + character string.

Notes

If the *mode* is not properly specified when the directory is created, it might not be possible to access the directory. In the current library, umask, uid and gid cannot be changed. The value of umask is fixed at 0002 and the values of uid and gid are fixed at 0xffff.

Return value

Returns zero on success. On error returns -1 times errno.

EACCES	No access permission.
EEXIST	File already exists.
EIO	I/O error.
ELOOP	Too many symbolic links encountered when resolving the path name.
EMFILE	Reached the maximum number of descriptors that can be opened.

ENAMETOOLONG	File path name is too long.
ENODEV	Specified device does not exist.
ENOENT	Directory not found in the specified path.
ENOMEM	Not enough free memory.
ENOSPC	No free space.
EROFS	Write access was requested for a file from a read-only filesystem.

mount

Mount device

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

#include <sys/mount.h>

int mount(

**const char ****fsname,*	Character string which specifies filesystem device name and unit number after mounting.
**const char ****devname,*	Character string which identifies the required device that will be used to open the block device to be mounted.
int *flag,*	Mount flag. Any of the following constants can be specified. For multiple specifications, take the logical OR. SCE_MT_RDWR Mount as read/write enabled. SCE_MT_RDONLY Mount as read-only. SCE_MT_ROBUST Mount in ROBUST node. SCE_MT_ERRCHECK Set an error if there is anything abnormal in the filesystem when mounting.
void **arg,*	Reserved
int *arglen***)**	Size of *arg*

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Mounts the block device specified by *devname* on the filesystem logical device specified by *fsname*.

devname usually specifies a string that identifies a previously created partition. If the mount is done read-only, providing only a read-only password is acceptable, but to allow mounting for read/write, a full password is required.

When SCE_MT_ROBUST is specified, filesystem information will always be updated. When any operation that causes a change to the filesystem (such as mkdir(), write()) is performed, it is immediately synchronized with the disk. Furthermore, updating of the close completion flag for memory card compatibility is only performed in ROBUST mode.

When SCE_MT_ERRCHECK is specified and an abnormality occurs in the filesystem, an EIO error will be returned. When an abnormality is seen in the filesystem, a prompt filesystem check is recommended. Even with an abnormality in the filesystem, the trouble-free portion of the filesystem is still readable, provided that the filesystem is not updated. However, writing should not be performed because it may worsen the problem.

Examples:

```
mount("pfs0:", "hdd0:tst1,fpwd1", SCE_MT_RDWR, NULL, 0);
mount("pfs1:", "hdd0:tst2,fpwd2", SCE_MT_RDWR|SCE_MT_ROBUST, NULL, 0);
mount("pfs2:", "hdd0:tst3,,rpwd3", SCE_MT_RDONLY, NULL, 0);
```

Return value

Returns zero on success. On error returns -1 times errno.

EACCES	No access permission.
EBUS	The specified partition is already open.
EINVAL	An invalid argument was specified.
EIO	I/O error.
EMFILE	Reached the maximum number of descriptors that can be opened.
ENODEV	Specified device does not exist.
ENOENT	Specified partition not found.
ENOMEM	Not enough free memory.
ENXIO	Not a supported device.

open

Create, open file

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

#include <stdio.h>

int open(

const char **name,*
File path name (including device name + ':')

int *flags,*
Access mode. Any of the following constants can be specified.

For multiple specifications, take the logical OR.

O_RDONLY	Open as read only
O_WRONLY	Open as write only
O_RDWR	Open for read/write
O_APPEND	Always perform writes at the end of file
O_CREAT	Create a new file if the file does not exist
O_TRUNC	Discard previous file contents
O_EXCL	When specified with O_CREAT, if a file exists with the same name, an error will occur

unsigned short *mode***)**
File mode

bit 0	Execute permission (other)
bit 1	Write permission (other)
bit 2	Read permission (other)
bit 3	Execute permission (group)
bit 4	Write permission (group)
bit 5	Read permission (group)
bit 6	Execute permission (user)
bit 7	Write permission (user)
bit 8	Read permission (user)
bit 9	Reserved
bit10	Reserved
bit11	Reserved

Macros for each mode are provided in sys/file.h. However, using octal codes such as 0777, 0755, etc. is also an easy way to specify the mode.

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Creates, opens a file. Assigns a file descriptor to the file that was opened. The file path name is "pfs" + unit number + ':' + character string.

Example:

```
open("pfs0:/foo", O_CREAT|O_RDWR, SCE_STM_RWXUGO);
```

Notes

If the *mode* was not properly specified when the file was created, it may not be possible to open the file. In the current library, umask, uid and gid cannot be changed. The value of umask is fixed at 0002 and the values of uid and gid are fixed at 0xffff.

Return value

Returns the file descriptor on normal completion (value > 0).

-1 times errno if an error occurred.

EACCES	No access permission.
EEXIST	Both O_CREAT and O_EXCL were specified and the file already exists.
EINVAL	An invalid argument was specified.
EIO	I/O error.
EISDIR	The file is a directory.
ELOOP	Too many symbolic links encountered when resolving the path name.
EMFILE	Reached the maximum number of descriptors that can be opened.
ENAMETOOLONG	File path name is too long.
ENODEV	Specified device does not exist.
ENOENT	Specified file does not exist.
ENOMEM	Not enough free memory.
ENOSPC	No free space.
EROFS	Write access was requested for a file from a read-only filesystem.

read

Read file

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int read(
 int fd,                    File descriptor of the read target
 void *buf,                 Address of the buffer that will store the read data
 size-t count)              Read data size
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Reads a maximum of *count* bytes from the file that was previously opened, into the buffer starting from the address specified by *buf*. Performance will improve if count is a multiple of 512. It is recommended that a multiple of 512 be used as much as possible. Even if reading in 512-byte units is not possible, reads should be performed in at least 4-byte units. To the degree that transfers are performed once in large units, performance will improve even more.

If an EIO|0x10000 error occurs, either overwrite the file or delete it completely without performing a filesystem check.

Return value

On success, returns the number of bytes read. The file position is advanced by this amount only. A return value of 0 means end of file. If an error occurred, -1 times errno is returned.

EBADF	*fd* is not a valid open descriptor.
EINVAL	An invalid argument was specified.
EIO	I/O error.
EIO|0x10000	Bad sector was found while reading the file contents.
ENOMEM	Not enough free memory.

readlink

Read symbolic link value

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int readlink(
 const char *path,
 char *buf,
 size_t bufsiz)
```

File path name
Buffer for writing contents
Size of *buf* (up to 1023 bytes)

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Stores the contents of the symbolic link specified by path into *buf*. *bufsiz* specifies the size of *buf*. readlink does not add null (NUL) characters to *buf*. If the buffer is too small to store the entire contents, the contents are truncated to fit into *bufsiz* bytes.

Return value

On success, returns the number of characters stored in the buffer. On error, returns -1 times errno.

EACCES	No access permission.
EEXIST	newname already exists.
EINVAL	Invalid argument was specified, or not a symbolic link.
EIO	I/O error.
ELOOP	Too many symbolic links encountered when resolving the path name.
EMFILE	Reached the maximum number of descriptors that can be opened.
ENAMETOOLONG	File path name is too long.
ENODEV	Specified device does not exist.
ENOENT	Specified file does not exist.
ENOMEM	Not enough free memory.

remove

Delete file

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

#include <stdio.h>
int remove(
const char ****name*)** File path name (including device name + ':')

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Deletes the specified file. The file path name is "pfs" + unit number + ':' + character string.

Return value

Returns zero on success. On error returns -1 times errno.

EACCES	No access permission.
EBUSY	The file is open.
EIO	I/O error.
EISDIR	The file is a directory.
ELOOP	Too many symbolic links encountered when resolving the path name.
ENAMETOOLONG	File path name is too long
EMFILE	Reached the maximum number of descriptors that can be opened.
ENODEV	Specified device does not exist.
ENOENT	Specified file does not exist.
ENOMEM	Not enough free memory.
EROFS	Write access was requested for a file from a read-only filesystem.

rename

Change file/directory name

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int rename(
 const char *oldname,        Name of file/directory before change
 const char *newname)        Name of file/directory after change
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Renames file. If required, performs movement between directories. If *newname* already exists, it is automatically replaced if the following conditions are met.

- *oldname* is a file and *newname* is also a file.
- *oldname* is a directory and *newname* is also a directory.
- *newname* is a directory and it is empty.
- *newname* is not open.

If *newname* exists, it is guaranteed that the original *newname* will remain unchanged even if the operation fails for any reason.

Return value

Returns zero on success. On error returns -1 times errno.

EACCES	No access permission.
EBUSY	The file is open or it is a working directory.
EINVAL	"." or ".." was specified, or newname includes part of the path of oldname. In other words, tried to change a directory into its own subdirectory.
EIO	I/O error.
EISDIR	oldname is a file and newname is a directory.
ELOOP	Too many symbolic links encountered when resolving the path name.
EMFILE	Reached the maximum number of descriptors that can be opened.
ENAMETOOLONG	File path name is too long.
ENODEV	Specified device does not exist.
ENOENT	Specified file does not exist.
ENOMEM	Not enough free memory.
ENOSPC	No free space.
ENOTDIR	oldname is directory but newname is not a directory.
ENOTVACANT	newname is a directory but the directory is not empty.
EROFS	Write access was requested for a file from a read-only filesystem.

rmdir

Delete directory

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

#include <stdio.h>

int rmdir(

const char **name***)** Directory path name (including device name + ':')

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Deletes the specified directory. The directory to be deleted must be empty. Directory path name is "pfs" + unit number + ':' + character string.

Return value

Returns zero on success. On error, returns -1 times errno.

EACCES	No access permission.
EBUSY	Directory is open or it is a working directory.
EIO	I/O error.
ELOOP	Too many symbolic links encountered when resolving the path name.
EMFILE	Reached the maximum number of descriptors that can be opened.
ENAMETOOLONG	File path name is too long.
ENODEV	Specified device does not exist.
ENOENT	Specified directory not found.
ENOMEM	Not enough free memory.
ENOTDIR	Specified file is not a directory.
ENOTVACANT	Directory is not empty.
EROFS	Write access was requested for a file from a read-only filesystem.

symlink

Create symbolic link

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int symlink(
 const char *oldname,          Original filename
 const char *newname)          New filename
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Creates a symbolic link named *newname* to *oldname*. The symbolic link is interpreted during execution when locating files or directories, by following its contents and replacing them. A symbolic link might specify an existing file, or a file which does not exist. "..." may be included in the path. If *newname* already exists, it will not be replaced.

Return value

Returns zero on success. On error, returns -1 times errno.

EACCES	No access permission.
EEXIST	*newname* already exists.
EINVAL	Invalid argument was specified.
EIO	I/O error.
ELOOP	Too many symbolic links encountered when resolving the path name.
EMFILE	Reached the maximum number of descriptors that can be opened.
ENAMETOOLONG	File path name is too long.
ENODEV	Specified device does not exist.
ENOENT	Specified file does not exist.
ENOMEM	Not enough free memory.
ENOSPC	No free space.
EROFS	Write access was requested for a file from a read-only filesystem.

sync

Synchronize buffer cache and disk

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int sync(
 const char *name,          Device name
 int flag)                  Flag (Reserved, not used)
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

To avoid reading and writing to a slow disk, the filesystem keeps data in memory. This function flushes the contents of the filesystem's buffer cache in this memory to the disk. Flushing also includes the cache on the disk as well.

Return value

Returns zero on success. On error returns -1 times errno.

EIO	I/O error.
ENODEV	Specified device does not exist.

umount

Unmount filesystem

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

#include <sys/mount.h>

int umount(

const char ****fsname*)** Character string specifying filesystem device name and unit number during mounting.

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Unmounts the filesystem. The contents of the buffer cache in memory are flushed.

Return value

Returns zero on success. On error returns -1 times errno.

EBUSY	File is open.
EMFILE	Reached the maximum number of descriptors that can be opened.
ENODEV	Specified device does not exist.

write

Write to file

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Syntax

```
#include <stdio.h>
int write(
 int fd,                    File descriptor of the file to be written.
 const void *buf,           Address that stores the data to be written.
 size_t count)              Size of the data to be written.
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Writes a maximum of count bytes from the buffer indicated by buf into the file referenced by the file descriptor fd. Performance will improve if count is a multiple of 512. It is recommended that a multiple of 512 be used as much as possible. Even if reading in 512-byte units is not possible, reads should be performed in at least 4-byte units. To the degree that transfers are performed once in large units, performance will improve even more.

If an EIO|0x10000 error occurs, delete the file without performing a filesystem check.

Return value

On success, returns the number of bytes written. The file position is advanced by this amount only.

If an error occurred, -1 times errno is returned.

EBADF	*fd* is not a valid open descriptor.
EINVAL	Invalid argument was specified.
EIO	I/O error.
EIO |0x10000	Bad sector was found while writing the file contents.
ENOMEM	Not enough free memory.
ENOSPC	No free space.

devctl Commands

PDIOC_CLOSEALL

Close all files

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Size of *arg.*
bufp	Reserved. Specify NULL.
buflen	Size of *bufp.*

Description

Closes all files on all mounted filesystems. However, file descriptors do not get freed, so use this function only when powering off, etc.

Return value

Returns 0.

PDIOC_CLRFSCKSTAT

Clear FSCK status

Library	*Introduced*	*Documentation last modified*
pfs	2.3	July 2, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Size of *arg.*
bufp	Reserved. Specify NULL.
buflen	Size of *bufp.*

Description

This command clears the state of the filesystem that was updated by fsck.

Return value

0 if processing succeeds.

PDIOC_GETFSCKSTAT

Check FSCK status

Library	*Introduced*	*Documentation last modified*
pfs	2.3	July 2, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Size of *arg.*
bufp	Reserved. Specify NULL.
buflen	Size of *bufp.*

Description

This command returns 1 only when a problem of some kind was found in the filesystem and the filesystem state was updated. Once this state occurs, it is held until cleared with PDIOC_CLRFSCKSTAT.

Return value

If fsck had previously corrected a problem in the filesystem (i.e. the filesystem state was updated), 1 is returned.

Otherwise, 0 is returned.

PDIOC_ZONEFREE

Get free zones

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Size of *arg.*
bufp	Reserved. Specify NULL.
buflen	Size of *bufp.*

Description

Gets the number of available free zones.

Return value

The number of free zones is returned as an unsigned 32 bit integer.

PDIOC_ZONESZ

Get zone size

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Size of *arg.*
bufp	Reserved. Specify NULL.
buflen	Size of *bufp.*

Description

Gets the zone size.

Return value

Returns the zone size.

ioctl2 Commands

PIOCALLOC

Allocate area

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Arguments

arg	Number of allocated zones.
arglen	Size of *arg.*
bufp	Reserved. Specify NULL.
buflen	Size of *bufp.*

Description

Allocates an area that can be used by a file. In pfs, the speed of allocating an area for a new file is not very high. When the file size is approximately known and a large amount of data is to be continuously written to a file, write performance will be improved if the area is allocated in advance before writing.

Example:

```
u_int size = 1024*1024;
ioctl2(fd, PIOCALLOC, &size, sizeof(int), NULL, 0);
```

Return value

Returns zero on success. On error returns -1 times errno.

EACCES	No access permission.
EINVAL	Invalid argument was specified.
EIO	I/O error.
ENOMEM	Not enough free memory.
ENOSPC	No free space.

PIOCATTRADD

Add extended file attribute entry

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Arguments

arg	Buffer that stores a 256 byte key and a 256 byte value.
arglen	Size of *arg.*
bufp	Reserved. Specify NULL.
buflen	Size of *bufp.*

Description

Adds an entry to the extended file attribute area.

Example:

```
struct {
        char key[0x100];
        char value[0x100];
} attr;
strcpy(key, "application");
strcpy(value, "x-compressed");
ioctl2(fd, PIOCATTRADD, &attr, 0x100*2, NULL, 0);
```

Return value

Returns zero on success. On error returns -1 times errno.

EACCES	No access permission.
EEXIST	Specified key already exists.
EINVAL	Invalid argument was specified.
EIO	I/O error.
ENOMEM	Not enough free memory.
ENOSPC	No free space.

PIOCATTRDEL

Delete extended file attribute entry

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Arguments

arg	Buffer that stores the key string.
arglen	Size of *arg.*
bufp	Reserved. Specify NULL.
buflen	Size of *bufp.*

Description

Deletes an entry from the extended file attribute area.

Example:

```
char key[] = "application";
ioctl2(fd, PIOCATTRDEL, key, strlen(key)+1, NULL, 0);
```

Return value

Returns zero on success. On error returns -1 times errno.

EACCES	No access permission.
EIO	I/O error.
ENOENT	Entry not found.
ENOMEM	Not enough free memory.

PIOCATTRLOOKUP

Lookup extended file attribute entry

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Arguments

arg	Buffer that stores the key string.
arglen	Size of *arg.*
bufp	Buffer that will store the value.
buflen	Size of *bufp.*

Description

Searches for the specified key from the extended file attribute area and stores the value in *bufp*.

Example:

```
char key[] = "application";
char value[0x100];
ioctl2(fd, PIOCATTRLOOKUP, key, strlen(key)+1, value, 0x100);
```

Return value

Returns zero on success.

On error, returns -1 times errno.

EIO	I/O error.
ENOENT	Entry not found.
ENOMEM	Not enough free memory.

PIOCATTRREAD

Read extended file attribute entry

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Size of *arg.*
bufp	512 byte buffer that will store the key and value
buflen	Size of *bufp.*

Description

Copies the next entry from the attribute entry stream into *bufp*. Returns 0 when reaches the end of entries.

Example:

```
struct {
        char key[0x100];
        char value[0x100];
} attr;
while ((r = ioctl2(fd, PIOCATTRREAD, NULL, 0, &attr, 0)) > 0)
        printf("%s/%s\n", attr.key, attr.value);
```

Return value

On success, returns length of key. Returns zero if reaches the end of entries.

On error, returns -1 times errno.

EIO	I/O error.
ENOMEM	Not enough free memory.

PIOCFREE

Free area

Library	*Introduced*	*Documentation last modified*
pfs	2.2.2	April 16, 2001

Arguments

arg	Reserved. Specify NULL.
arglen	Size of *arg.*
bufp	Reserved. Specify NULL.
buflen	Size of *bufp.*

Description

Frees areas not being used by files.

Return value

Returns zero on success. On error returns -1 times errno.

EACCES	No access permission.
EINVAL	Invalid argument was specified.
EIO	I/O error.
ENOMEM	Not enough free memory.
ENOSPC	No free space.

Chapter 6: USB Driver Library
Table of Contents

Structures

sceUsbdIsochronousPswLen

Multiple isochronous transfer packet structure

Library	*Introduced*	*Documentation last modified*
usbd	2.4.3	January 4, 2002

Structure

typedef struct _sceUsbdIsochronousPswLen {

u_short *len***:11;**	Transfer byte count
u_short *reserved***:1;**	Reserved area
u_short *PSW***:4;**	Transfer completion code (ITD.CC)

} sceUsbdIsochronousPswLen;

Description

This structure is used in the sceUsbdMultiIsochronousRequest structure of the sceUsbdMultiIsochronousTransfer function.

It specifies the number of transfer bytes for each packet.

After the transfer ends, the result is stored in PSW.

sceUsbdLddOps

LDD management structure

Library	*Introduced*	*Documentation last modified*
usbd	1.5	October 6, 2000

Structure

typedef struct _sceUsbdLddOps {

struct _sceUsbdLddOps **forw, *back;*	forw	Bi-directional link used internally by USBD, set to NULL.
	back	Bi-directional link used internally by USBD, set to NULL.
char **name;*		Specifies name string for LDD (naming is arbitrary).
int *(*probe)(***int dev_id);**		Specifies device probing function (when the device is inserted).
int *(*attach)***(int dev_id);**		Specifies device attaching function (when communicating with the device).
int *(*detach)***(int dev_id);**		Specifies device detaching function (when the device is removed).
void * *reserved[5];*		Reserved for future extensions. Must be all zeros.
u_int *gp;*		$gp value used when calling the probe, attach, or detach function from USBD. USBD sets this to the value of $gp when sceUsbdRegisterLdd() is called.

} sceUsbdLddOps;

Description

The USBD uses this structure for LDD (Logical Device Driver) management. It is specified as an argument of sceUsbdRegisterLdd().

sceUsbdMultiIsochronousRequest

Multiple isochronous transfer request structure

Library	*Introduced*	*Documentation last modified*
usbd	2.4.3	January 4, 2002

Structure

```
#define sceUsbd_MAX_ISOCH_PACKETS 8
typedef struct _sceUsbdMultiIsochronousRequest {
    void *buffer_base;                          Starting address of send data to be transferred (or of receive buffer)
    int relative_start_frame;                   Relative value of frame number (this is similar to the "relative value of the frame number" in the description of the sceUsbdTransferPipe function)
    int num_packets;                            Number of packets
    sceUsbdIsochronousPswLen Packets[sceUsbd_MAX_ISOCH_PACKETS];   Number of bytes for each packet and completion code
} sceUsbdMultiIsochronousRequest;
```

Description

This structure is used by the sceUsbdMultiIsochronousTransfer function.

It indicates details of the transfer request.

The Packets member indicates the number of bytes in each packet and its purpose is to store the transfer result.

LDD External Public Functions

xxxAttach

Function to attach device by LDD

Library	*Introduced*	*Documentation last modified*
usbd	1.5	October 6, 2000

Syntax

int xxxAttach(

int *dev_id);* **Device ID**

Description

This function is used to attach a device if xxxProbe() returned a non-zero value (corresponding to the attach member in the sceUsbdLddOps structure).

The main operations required for xxxAttach() are as follows.

- Use sceUsbdScanStaticDescriptor() to get the necessary information while scanning the contents of the static descriptor and checking for consistency.
- Open the necessary pipe within the LDD (sceUsbdOpenPipe)
- Associate LDD-dependent data with the device (sceUsbdSetPrivateData)
- Put device in Configured state (sceUsbdSetConfiguration)

Notes

Currently, operations are the same regardless of the xxxAttach() return value. In the future, there are plans for turning off or disabling the device if an error takes place.

Return value

A zero is returned for normal exits.

A non-zero value is returned if an error takes place.

xxxDetach

Function to detach device by LDD

Library	*Introduced*	*Documentation last modified*
usbd	1.5	October 6, 2000

Syntax

int xxxDetach(

int *dev_id);* Device ID

Description

xxxDetach() is a detach-handling function called when a device is detached (corresponding to the detach member in the sceUsbdLddOps structure). The function performs LDD-dependent disconnect operations (e.g. freeing of private data).

Operations such as closing related pipes are performed by the USBD so they do not need to be handled separately.

Return value

A zero is returned for normal exits.

A non-zero value is returned if an error takes place.

xxxProbe

Function to probe device by LDD

Library	*Introduced*	*Documentation last modified*
usbd	1.5	October 6, 2000

Syntax

int xxxProbe(

int *dev_id);* Device ID

Description

xxxProbe() provides device detection for LDDs (corresponding to the probe member in the sceUsbdLddOps structure).

This function determines whether the LDD should be used to handle the dev_id device.

Before calling the xxxProbe function, the USBD stores a static descriptor in its own buffer. Thus, information about the type of device with dev_id can be obtained using sceUsbdScanStaticDescriptor(). This function gets the static descriptor retrieved by the USBD when the device is recognized.

Whether or not this device is to be handled is based on the following:

- Decide based on Device Descriptor or the Class, SubClass, Protocol of the Interface Descriptor.
- Decide based on idVendor, idProduct of the Device Descriptor.

xxxProbe() is called under one of the following conditions:

- If there is no device associated with an LDD when sceUsbdRegisterLdd() is called.
- If a new device is connected.

The device ID is the ID used to specify the device in the USBD. In the current implementation, the device ID always uses the same value as the address value on the USB bus. Since this address changes dynamically, it has no connection with the location information of the port number, etc. Use sceUsbdGetDeviceLocation() to get device location information.

Notes

Once a decision is made to "handle" the device, this cannot be revoked later. Also, a single device cannot be shared with other LDDs.

Return value

If a device indicated by dev_id is handled by the corresponding LDD, a nonzero value is returned. Otherwise, a zero is returned.

USBD Functions

sceUsbdChangeThreadPriority

Change the priority of athread used by the USBD

Library	*Introduced*	*Documentation last modified*
usbd	2.0	March 26, 2001

Syntax

int sceUsbdChangeThreadPriority(

int *prio1,*	Priority of interrupt handling thread
int *prio2);*	Priority of thread calling callback function

Calling conditions

Can be called from a thread

Not multithread safe

Description

Changes the priority of a thread generated by the USBD.

Priority Prio1 should be higher than priority Prio2.

Return Value

See Completion/Error codes

sceUsbdClosePipe

Close pipe

Library	*Introduced*	*Documentation last modified*
usbd	1.5	March 26, 2001

Syntax

int sceUsbdClosePipe(

int *pipe_id);* Pipe ID of pipe to be closed

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

This function closes the specified pipe *(pipe_id).*

Return value

See Completion/Error Codes

sceUsbdGetDeviceLocation

Get device location information

Library	*Introduced*	*Documentation last modified*
usbd	1.6	July 2, 2001

Syntax

int sceUsbdGetDeviceLocation(

int *dev_id,*	Device ID
u_char **locations);*	Pointer for storing location information (indicates the start of an ordinary 7-byte area)

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Transfers location information of the specified device *(dev_id)* to the specified area *(locations)*. The destination pointed to by *locations* is always a 7-byte area, and the port numbers from PlayStation 2 to the specified device are set sequentially as follows, with the 0x00 that appears first representing the specified device.

- When the specified device is connected to USB port 1 of the PlayStation 2 console

  0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

- When Hub-A is connected to USB port 2 of the PlayStation 2 console, Hub-B is connected to port 3 of Hub-A, and the specified device is connected to port 4 of Hub-B

  0x02, 0x03, 0x04, 0x00, 0x00, 0x00, 0x00

According to the USB standard, at most five Hubs can be passed through between the PlayStation 2 console and the device. If this restriction is violated, sceUsbdGetDeviceLocation() will return an sceUsbd_INVAL_HUB_DEPTH error.

If sceUsbdGetDeviceLocation() is issued to the ID of a device that is not connected, an sceUsbd_INVAL_CONTEXT error will be returned.

Return value

See Completion/Error Codes

sceUsbdGetPrivateData

Get associated private data

Library	*Introduced*	*Documentation last modified*
usbd	1.5	March 26, 2001

Syntax

void *sceUsbdGetPrivateData(

int *dev_id);* Device ID

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

This function gets the private data associated with the specified device *(dev_id).*

Return value

See Completion/Error Codes

sceUsbdGetReportDescriptor

Access report descriptor

Library	*Introduced*	*Documentation last modified*
usbd	2.3	July 2, 2001

Syntax

int sceUsbdGetReportDescriptor(

int *dev_id,*	Device ID
int *cfgnum,*	Configuration number
	Configuration descriptor's bConfigurationValue.
int *ifnum,*	Interface number
	Interface descriptor's bInterfaceNumber.
void ***desc,*	Destination storage location for the pointer that points to the acquired report descriptor.
	If NULL is specified for *desc*, nothing will be stored.
int **len***);**	Destination storage location for length of the acquired report descriptor.
	If NULL is specified for *len*, nothing will be stored.
	Stored length information is in bytes.

Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

Description

Accesses the report descriptor (one type of HID sub-descriptor) held in the USBD.

The usbd.irx start-up option reportd=1 must be specified to use this function.

Since this function can be executed during a probe function, it can be used to judge a device using the report descriptor.

Return value

Refer to completion / error codes.

sceUsbdNOERR is returned on normal end.

sceUsbdMultiIsochronousTransfer

Perform multiple isochronous transfers

Library	*Introduced*	*Documentation last modified*
usbd	2.4.3	January 4, 2002

Syntax

```
typedef void (*sceUsbdMultiIsochronousDoneCallback)(
    int result,                                          Completion / error code
    sceUsbdMultiIsochronousRequest *req,                 Pointer for receiving the transfer
                                                         request information structure from
                                                         the USBD
    void *arg                                            Pointer to LDD-dependent private
                                                         data
);          /* Callback function */

int sceUsbdMultiIsochronousTransfer(
 int pipe_id,                                            Pipe ID
 sceUsbdMultiIsochronousRequest *req,                    Pointer for passing the transfer
                                                         request information structure to the
                                                         USBD
 sceUsbdMultiIsochronousDoneCallback done_cb,            Transfer completion callback function
 void *arg);                                             Pointer to LDD-dependent private
                                                         data
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

This function requests up to eight isochronous transfers at one time for the specified pipe (*pipe_id*). Since the processing for performing multiple transfers at once is implemented in hardware, the interrupt and callback processing frequency is reduced to (1 / *req*->num_packets) compared with using sceUsbdTransferPipe().

The function (*done_cb*) is a callback function that is called when the transfer completes. The *result* argument is the completion code (sceUsbd_XXX), the *req* argument is a pointer to the transfer completion information, and the *arg* argument, which is the *arg* argument of sceUsbdMultiIsochronousTransfer(), is used for LDD-dependent processing.

The PSW.CC field within the *result* argument of the transfer completion callback function is always zero when this sceUsbdMultiIsochronousTransfer() function is used. For the completion code of each individual transfer, see *req*->Packets[i].PSW (where, i is from 0 to (*req*->num_packets - 1)).

The starting address of the area that is transferred first is *req*->buffer_base, and the specified area byte count is *req*->Packets[0].len.

The starting address of the area that is transferred next (and subsequently) is the address obtained by adding the byte count for the specified area of the previous transfer to the starting address of the previous transfer.

The number of bytes that were actually transferred during an input transfer can be seen from the req argument of the callback function as *req*->Packets[i].len.

The specification of the $gp register value and the relative value of the frame number are the same as those of the sceUsbdTransferPipe() function. Refer to the descriptions given there.

However, since multiple transfers are performed, the frame number is incremented by *req*->num_packets instead of 1 when the specified multiple transfers end.

Return value

See Completion/error codes.

When processing completes normally, sceUsbdNOERR is returned.

sceUsbdOpenPipe

Open pipe

Library	*Introduced*	*Documentation last modified*
usbd	1.5	January 4, 2002

Syntax

int sceUsbdOpenPipe(

int *dev_id,*	Device ID
UsbEndpointDescriptor **edesc);*	Pointer to endpoint descriptor of pipe to be opened

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

This function opens a pipe with the specified endpoint (*edesc*) for the specified device (*dev_id*).

A packet having a length of 63 or 64 bytes will automatically have its length rounded to 62 bytes for the pipe opened by this function.

This is done to prevent a hardware bug (see the section on restrictions in the Overview).

If you do not want the length rounded to 62 bytes, use the sceUsbdOpenPipeAligned() function.

Return value

If *edesc* is NULL , the pipe ID of the Control Pipe is returned.

If *edesc* is non-NULL, a pipe is opened based on the Endpoint Descriptor, and the pipe ID is returned.

A -1, indicating an error, is returned if one of the following is true.

- The value of *dev_id* is invalid.
- The Endpoint count exceeds the maximum value.

sceUsbdOpenPipeAligned

Open pipe on a word boundary

Library	*Introduced*	*Documentation last modified*
usbd	1.6	January 4, 2002

Syntax

int sceUsbdOpenPipeAligned(

int *dev_id,*	Device ID
UsbEndpointDescriptor **edesc);*	Pointer to endpoint descriptor of pipe to be opened

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Opens the pipe with the specified endpoint *(edesc)* of the specified device *(dev_id).*

sceUsbdOpenPipeAligned() differs from sceUsbdOpenPipe() as follows:

- When the maximum packet size is 64 bytes, no processing for rounding it to 62 bytes is performed (see "Restrictions and Precautions" in the USB driver library overview).
- If the starting address of the data is not a word boundary when sceUsbdTransferPipe() is executed for the opened pipe, the sceUsbd_INVAL_ALIGN error will occur.

Processing for determining whether an sceUsbd_INVAL_ALIGN error has occurred is performed using only the starting address value, and does not depend on transfer direction or transfer size.

However, the method of determining whether a word boundary error has occurred is different when sceUsbdTransferPipe() is executed for the Control pipe.

The sceUsbdOpenPipeAligned() function can be used even for the Control pipe to prevent rounding to 62 bytes.

Either sceUsbdOpenPipe() or sceUsbdOpenPipeAligned() can be used to determine whether a word boundary error has occurred.

An sceUsbd_INVAL_ALIGN error is returned for the Control pipe only when all of the following conditions are satisfied.

- The starting address of the data is not on a word boundary.
- The transfer direction is host -> device.
- The MaxPacketSize of the control endpoint descriptor is 64.
- The packet size is at least 63 bytes.

The reason for this is that the Control pipe is automatically opened by the USBD.

When a USB device is connected, the USBD automatically opens the Control pipe to obtain descriptor information. Therefore, the user cannot select whether to enable or disable the word boundary check (that is, the Control pipe is opened before the probe function is called).

As a result, a word boundary check is always performed when sceUsbdTransferPipe() is executed for the Control pipe.

Note that simply looking at the word boundary will not generate an error. An error will not occur unless the four conditions described above for generating the hardware bug are true.

Return value

When edesc is NULL, the pipe ID of the Control pipe is returned.

When edesc is not NULL, the pipe is opened according to the Endpoint Descriptor, and the pipe ID of that pipe is returned. At this time, an alignment check is executed.

Any of the following cases indicates an error, and -1 is returned.

- When the dev_id value is illegal
- When the number of Endpoints exceeds the maximum value
- When the alignment of the pipe that was opened is invalid

sceUsbdRegisterLdd

Register LDD

Library	*Introduced*	*Documentation last modified*
usbd	1.5	March 26, 2001

Syntax

int sceUsbdRegisterLdd(

sceUsbdLddOps **lddops);* LDD registration information.
Refer to the sceUsbdLddOps structure for contents.

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Registers LDD (Logical Device Driver) to USBD.

Return value

Table 6-1

Status	Meaning
sceUsbd_NOERR	Normal exit
sceUsbd_BUSY	Error lddops->forw, lddops->back were non-NULL, or were already registered at the head of the LDD list in the USBD
sceUsbd_INVAL_LDDOPS	Error lddops->name was NULL

sceUsbdScanStaticDescriptor

Scan static descriptors

Library	*Introduced*	*Documentation last modified*
usbd	1.5	July 2, 2001

Syntax

void *sceUsbdScanStaticDescriptor(

int *dev_id,*	Device ID
void **ptr,*	Location of descriptor scan
u_char *type);*	Type of descriptor to be scanned (bDescriptorType of the standard descriptor structure)

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

This functions scans static descriptors stored in USBD.

If the scan starting position *(ptr)* is NULL, scanning takes place from the beginning. Otherwise, scanning begins with the descriptor "after" the descriptor indicated by *ptr.*

type specifies the descriptor type to be scanned (equivalent to the standard descriptor structure bDescriptorType). All descriptors will be scanned if zero.

The arrangement of static descriptors stored in the USBD is as follows:

Device Descriptor (always one)
Configuration Descriptor 1
Configuration Descriptor 2
:
Configuration Descriptor N (=bNumConfigurations of Device Descriptor)

Each Configuration Descriptor includes 0, or 1 or more Interface Descriptor, Endpoint Descriptor or Class Specific Descriptor.

The size of the Configuration Descriptors is wTotalLength bytes of the Configuration Descriptor. There is no alignment between Configuration Descriptors

Return value

If scan is successful, that descriptor's start address is returned. If the applicable descriptor does not exist, NULL is returned.

sceUsbdSetPrivateData

Associate private data with device

Library	*Introduced*	*Documentation last modified*
usbd	1.5	March 26, 2001

Syntax

int sceUsbdSetPrivateData(

int *dev_id,*	Device ID
void **priv);*	Pointer to LDD-dependent private data

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

This function associates the specified device *(dev_id)* with LDD-dependent private data *(priv).*

Any type of data can be indicated by the private data. For example, an address for a dynamic data area used for LDD-dependent operations can be associated, and this area can be freed by xxxDetach().

Return value

See Completion/Error Codes

sceUsbdTransferPipe

Transfer data with pipe

Library	*Introduced*	*Documentation last modified*
usbd	1.5	March 26, 2001

Syntax

```
typedef void (*sceUsbdDoneCallback)
 (int result, int count, void *arg);
int sceUsbdTransferPipe(
 int pipe_id,                    Pipe ID
 void *ptr,                      Pointer to transmit data or receive buffer
 int len,                        Transfer length (in bytes)
 void *option,                   Option (meaning differs according to the transfer mode)
 sceUsbdDoneCallback done_cb,    Callback function that is called when the transfer ends.
                                 The arguments of the callback function are as follows.
                                     result   Completion/error code
                                     count    Transferred data size (in bytes)
                                     arg      Pointer to LDD-dependent
                                              private data
 void *arg);                     Pointer to LDD-dependent private data
```

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

Transfers data *(ptr, len)* for the specified pipe *(pipe_id).*

The specified option *(option)* is a transfer type-dependent parameter (described later) that is used during Control transfers and Isochronous transfers.

The function *(done_cb)* is a callback function that is called when the transfer is completed. The arguments are used for LDD-dependent processing. The *result* argument is the completion code (sceUsbd_XXX), the *count* argument is the transferred data byte count, and the *arg* argument is the *arg* argument of sceUsbdTransferPipe().

The $gp value when *done_cb* is called is the $gp value when sceUsbdTransferPipe() is called.

sceUsbdTransferPipe() has no connection to the (Control/Bulk/Interrupt/Isochronous) transfer method, and it can be called multiple times before the callback comes.

Control transfer

Specify the pointer to UsbDeviceRequest for the option *(option).*

Isochronous transfer

Specify the relative value of the frame number for the option ((int) *option*). Usually, 0 is specified. However, in this case, the frame number is treated as the sum of 2 added to the current frame number when that frame is the beginning (of the set of frames that are transfered without interruption), and the transfer is performed after 1 to 2 ms. As long as data transfers are not interrupted for the next and subsequent frames, the transfers are performed according to frame numbers with an increment value of 1.

For example, to perform an Isochronous transfer of m [frame] data after n [frame], specify (n-2) for the option when performing the first transfer, and specify 0 for the remaining options.

Table 6-2

Frame	Option
1	n-2
2	0
3	0
:	:
m-1	0
m	0

Normally, Isochronous transfers are performed every frame. However, to perform these kinds of transfers "without interruption," you must "accumulate multiple transfer requests" by calling sceUsbdTransferPipe multiple times before calling the callback function.

Since the frame numbers that are managed within the USBD are 16 bits, you cannot specify a frame number that exceeds 0x7ffc (approximately 30 seconds).

Interrupt transfer

The option *(option)* is not used. Always specify NULL.

Bulk transfer

The option *(option)* is not used. Always specify NULL.

Notes

1. Several macros for calling sceUsbdTransferPipe() are provided in usbd.h. Those macros can be used for standard tranfers that are not Class-dependent or Vendor-dependent.
2. No function is provided for interrupting the transfer after a transfer is requested using sceUsbdTransferPipe(). If you need to interrupt the transfer, use sceUsbdClosePipe() to close the pipe and then use sceUsbdOpenPipe() to open the pipe again.
3. If the actual transfer length (count) is less than the specified length (len) in an Isochronous IN transfer, the completion code (result) will be 0x90 (Data Underrun). Except for this case, there is no non-0x0 completion code that should be considered as a "non-error status."

The priority of the thread executing done_cb can be set using sceUsbdChangeThreadPriority().

Return value

See Completion/Error codes.

sceUsbdUnregisterLdd

Unregister LDD

Library	*Introduced*	*Documentation last modified*
usbd	1.5	March 26, 2001

Syntax

int sceUsbdUnregisterLdd(

sceUsbdLddOps **lddops);* Pointer to LDD management structure

Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

Description

This function cancels the registration of an LDD (Logical Device Driver).

All pipes for an active device (a device that is already attached) will be closed. Before this function is called, LDD exit operations should be performed (e.g. releasing private data).

Return value

See Completion/Error Codes

Completion/Error Codes

Error Codes

Error codes returned by the USBD

Library	*Introduced*	*Documentation last modified*
usbd	1.5	January 4, 2002

```
#define sceUsbd_INVAL_DEVICE        0x101   /* Invalid device id */
#define sceUsbd_INVAL_PIPE          0x102   /* Invalid pipe id */
#define sceUsbd_INVAL_LENGTH        0x103   /* Invalid length */
#define sceUsbd_INVAL_LDDOPS        0x104   /* Invalid LDD ops */
#define sceUsbd_INVAL_CONTEXT       0x105   /* Invalid context */
#define sceUsbd_INVAL_ALIGN         0x106   /* Invalid argument */
#define sceUsbd_INVAL_HUB_DEPTH     0x107   /* Invalid hub depth */
#define sceUsbd_NO_ED               0x111   /* No space (ED) */
#define sceUsbd_NO_IOREQ            0x112   /* No space (IOREQ) */
#define sceUsbd_NO_OPTION           0x113   /* No Option */
#define sceUsbd_BUSY                0x121   /* Busy */
#define sceUsbd_ABORTED             0x122   /* Aborted */
#define sceUsbd_NOT_IMP             0x131   /* Not yet implemented */
#define sceUsbd_ERROR               0x132   /* Error (unknown reason) */
```

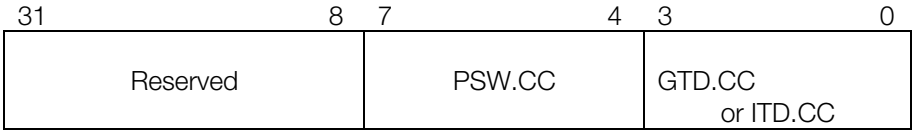
Completion Codes

Completion codes returned by OHCI (GTD.CC or ITD.CC)

Library	*Introduced*	*Documentation last modified*
usbd	1.5	January 4, 2002

The completion code bitmap is shown below.

Figure 6-1



1. GTD.CC (Completion Code of General Transfer Descriptor)
   Completion code for control transfers, bulk transfers, and interrupt transfers.
2. ITD.CC (Completion Code of Isochronous Transfer Descriptor)
   Completion code for isochronous transfers.
3. PSW.CC (Completion Code of Packet Status Word)
   Data overruns, data underruns, buffer overruns, and buffer underruns from isochronous transfers are indicated here.

Notes:

- Reserved is all zeros.
- In isochronous transfers, if the transferred data length is smaller than the specified data length, a 0x90 (Data Underrun) will be returned. This should be treated as a status indicator rather than an error.

CC is defined as shown below.

```
#define sceUsbd_NOERR       0x000   /* No Error */
#define sceUsbd_CRC         0x001   /* CRC */
#define sceUsbd_BFV         0x002   /* Bit Stuffing Violation */
#define sceUsbd_DTM         0x003   /* Data Toggle Mismatch */
#define sceUsbd_STALL       0x004   /* Stall */
#define sceUsbd_NOTRESP     0x005   /* Device Not Responding */
#define sceUsbd_PIDCF       0x006   /* PID Check Failure */
#define sceUsbd_UEPID       0x007   /* Unexpected PID */
#define sceUsbd_DOR         0x008   /* Data Overrun */
#define sceUsbd_DUR         0x009   /* Data Underrun */
#define sceUsbd_RSVDA       0x00a   /* (reserved) */
#define sceUsbd_RSVDB       0x00b   /* (reserved) */
#define sceUsbd_BOR         0x00c   /* Buffer Overrun */
#define sceUsbd_BUR         0x00d   /* Buffer Underrun */
#define sceUsbd_NOTACC1     0x00e   /* (not accessed) */
```

```
#define sceUsbd_NOTACC2         0x00f    /* (not accessed) */
```

The following macros are provided to retrieve PSW.CC.

```
/* PSW.CC (returned as an OR with ITD.CC above during Isochronous transfers) */
#define sceUsbd_PSW_BITS        0x0f0    /* PSW.CC (Isochronous) */
#define sceUsbd_PSW_SHIFT       4
```

Chapter 7: USB Module Autoloader
Table of Contents

Structures

USBDEV_t

Driver data management structure

Library	*Introduced*	*Documentation last modified*
usbml	2.1	January 4, 2002

Structure

```
#define NUM_OF_ARGV 8
 typedef struct _USBDEV_t {
    struct _USBDEV_t* forw;     /* For unidirectional list */
     char* dispname;            /* Display name  */
     int vendor;                /* Device vendor */
     int product;               /* Device product */
     int release;               /* Device USB spec release */
     int class;                 /* Interface class */
     int subclass;              /* Interface subclass */
     int protocol;              /* Interface protocol */
     char* category;            /* Category */
     char* path;                /* Driver file path */
     char* argv[NUM_OF_ARGV];   /* Driver parameters */
     int argc;                  /* Number of parameters */
     char activate_flag;        /*Activate ON/OFF*/
     /*Information of Loaded Module*/
     int modid;                 /*Loaded Module ID*/
     char modname[56];          /*Loaded Module Name*/
     int load_result;           /*Result of LoadStartModule()*/
     } USBDEV_t;
} USBDEV_t;
```

Unidirectional link used within USBMLOAD

Driver name

Vendor ID

Product ID

USB release

Class code

Subclass code

Protocol code

Category

Driver file path

Arguments passed to driver

Number of arguments passed to driver (not including "lmode=AUTOLOAD")

Activate flag used within USBMLOAD

Member which stores the ID of the loaded module

Member which stores the name of the loaded module

Member which stores the LoadStartModule() result.

Description

This is a structure for the internal management of driver data by USBMLOAD.

It is also used when the user registers a driver.

If you want to specify a wildcard for the *vendor*, *product*, *release*, *class*, *subclass*, or *protocol* member, substitute WILDCARD_INT, which is defined in usbmload.h.

Functions

sceUsbmlActivateCategory

Activate specified category

Library	*Introduced*	*Documentation last modified*
usbml	2.1	March 26, 2001

Syntax

int sceUsbmlActivateCategory(

const char* *category)* Category (string)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

This function enables the autoloading of drivers that belong to the specified category (the default is disabled).

As with sceUsbmlEnable(), a search is performed for devices that do not have an LDD, but no search is performed when autoloading is disabled.

Return value

1 or more	Number of activated drivers
USBML_NG (-1)	No driver was found that belongs to the specified category.

sceUsbmlChangeThreadPriority

Change autoloader thread priority

Library	*Introduced*	*Documentation last modified*
usbml	2.1	March 26, 2001

Syntax

int sceUsbmlChangeThreadPriority(

int *prio1)* Thread priority

Calling conditions

Can be called from a thread

Not multithread safe

Description

This function changes the priority of the autoloader thread.

Return value

USBML_OK	Normal termination
USBML_NG	Error

sceUsbmlDisable

Disable automatic loading function

Library	*Introduced*	*Documentation last modified*
usbml	2.1	March 26, 2001

Syntax

int sceUsbmlDisable(void)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

This function disables the autoload function. The default is disabled.

This function is used to disable autoloading in states when it cannot be performed (e.g., during movie streaming).

Return value

USBML_OK (0)	Processing was successful
USBML_NG (-1)	Processing failed

sceUsbmlEnable

Enable autoload function

Library	*Introduced*	*Documentation last modified*
usbml	2.1	March 26, 2001

Syntax

int sceUsbmlEnable(void)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

This function enables the autoload function. The default is disabled.

When the state transitions from disabled to enabled, this function searches for devices without LDDs. If such a device is found, the usbmload.irx probe function is called.

Consequently, connected devices can be autoloaded in the disabled state.

Return value

USBML_OK (0)	Processing was successful
USBML_NG (-1)	Processing failed

sceUsbmlInactivateCategory

Inactivate specified category

Library	*Introduced*	*Documentation last modified*
usbml	2.1	March 26, 2001

Syntax

int sceUsbmlInactivateCategory(

const char* *category)* Category (string)

Calling conditions

Can be called from a thread

Not multithread safe (must be called in an interrupt-enabled state)

Description

This function disables the autoloading of drivers that belong to the specified category.

Return value

1 or more	Number of drivers that were inactivated
USBML_NG (-1)	No driver was found that belongs to the specified category.

sceUsbmlLoadConffile

Load driver database file

Library	*Introduced*	*Documentation last modified*
usbml	2.1	March 26, 2001

Syntax

int sceUsbmlLoadConffile(

const char* *conffile)* File path of driver database file (string)

Calling conditions

Can be called from a thread

Not multithread safe

Description

This function makes the autoloader load the driver database file.

Even if the driver database file was not loaded when usbmload.irx was started up, this API can be used to load it.

Multiple files can be loaded by calling this function repeatedly.

Return value

USBML_OK (0) Processing was successful

USBML_NG (-1) Processing failed

sceUsbmlRegisterDevice

Register driver data

Library	*Introduced*	*Documentation last modified*
usbml	2.1	March 26, 2001

Syntax

int sceUsbmlRegisterDevice(

USBDEV_t* *device)* Driver data

Calling conditions

Can be called from a thread

Not multithread safe

Description

This function registers in the autoloader, the driver data specified in the device argument just as if it were loaded from the driver database file.

Registration cannot be deleted.

The string information is copied into the autoloader during registration.

Return value

USBML_OK (0) Processing was successful

USBML_NG (-1) Processing failed

sceUsbmlRegisterLoadFunc

Register class driver loading function

Library	*Introduced*	*Documentation last modified*
usbml	2.1	August 31, 2001

Syntax

int sceUsbmlRegisterLoadFunc

(sceUsbmlLoadFunc *loadfunc)*	Pointer to function that loads the class driver
typedef USBDEV_t* (*sceUsbmlPopDevinfo)(void);	
typedef void (*sceUsbmlLoadFunc)	
(sceUsbmlPopDevinfo *pop_devinfo);*	Pointer to function that gets the filename of the candidate class driver

Calling conditions

Can be called from a thread

Not multithread safe

Description

This function allows class drivers to be loaded with a user-created routine. You might want to do this, for example, if you wanted to convey details about the loading status to the EE.

This function enables you to register a user-created class driver loading routine.

If registration is successful, the registered routine will be called instead of the default processing routine. Only one function can be registered.

Notes

A routine equivalent to the default processing routine is shown in the listing, below.

Since the specifications were changed beginning with Release 2.3.4, the function must be rewritten.

Previously, a flag was used to prevent duplicate loading. However, beginning with Release 2.3.4, duplicate loading is prevented using the module name.

This is because consideration has been given to modules that will unload themselves.

```
#define AUTOLOAD_PARAM "lmode=AUTOLOAD"
#define ARGP_MAX       256

void default_loadfunc(sceUsbmlPopDevinfo pop_devinfo)
{
  USBDEV_t *device;
  char argp[ARGP_MAX];
  int result;
  int modid = -1;
  int i;
  int argp_len = 0;
  int argv_len = 0;
  ModuleStatus modstat;

  while((device = pop_devinfo()) != NULL) {
    if (device->modid >= 0) { /* In initial state modid=-1 */
      if (ReferModuleStatus(modid,&modstat) == KE_OK) {
     if (strcmp(modstat.name,device->modname) == 0) {
```

```
                /* A module with the same ID and same name exists so no
        reading takes place */
                continue;
              }
               }
            }

            /* Process to combine arguments */
            argp_len = 0;
            for(i=0; i<device->argc; i++) {
              argv_len = strlen(device->argv[i]) + 1;
              if ((argp_len + argv_len) > (ARGP_MAX - sizeof(AUTOLOAD_PAR
          AM) - 1))
               { break; }
              strcpy(argp+argp_len, device->argv[i]);
              argp_len += argv_len;
            }
            strcpy(argp+argp_len, AUTOLOAD_PARAM);
            argp_len += sizeof(AUTOLOAD_PARAM);

            /* Load module */
            modid = LoadStartModule(device->path, argp_len, argp, &result)
          ;
            if (modid >= 0) {
              /* Made resident so copy module information */
              device->modid = modid;
              device->load_result = result;
              ReferModuleStatus(modid,&modstat);
              strcpy(device->modname,modstat.name);
            }
          }
          return;
        }
```

The pop_devinfo function obtains device information for candidate devices that the probe function of usbmload.irx inserted in the ring buffer.

More than one device candidate can be considered, so the function will continue to get information until the buffer has been exhausted.

Return value

USBML_OK (0)	Registration was successful
USBML_NG (-1)	Registration failed

sceUsbmlUnregisterLoadFunc

Delete registration of registered class driver loading function

Library	*Introduced*	*Documentation last modified*
usbml	2.1	March 26, 2001

Syntax

void sceUsbmlUnregisterLoadFunc(void)

Calling conditions

Can be called from a thread

Not multithread safe

Description

This function deletes the function that was registered by sceUsbmlRegisterLoadFunc().

After the function is deleted, drivers will be loaded by the default processing routine.

Return value

None