PlayStation®2 EE Library Reference
Release 2.4.3

Movie Libraries

Summary Table of Contents

About This Manual **v**

Changes Since Last Release v

Related Documentation v

Typographic Conventions v

Developer Support v

Chapter 1: IPU Library **1-1**

Structures 1-3

Functions 1-4

Chapter 2: MPEG Library **2-1**

Structures 2-3

Functions 2-9

About This Manual

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

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

Changes Since Last Release

Chapter 2: MPEG Library

- In sceMpegCreate(), the description in the "Notes" section has been deleted.

Related Documentation

Library specifications for the IOP can be found in the *PlayStation®2 IOP Library Reference* manuals and the *PlayStation®2 IOP 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: IPU Library
Table of Contents

Structures **1-3**
sceIpuDmaEnv 1-3

Functions **1-4**
sceIpuBCLR 1-4
sceIpuBDEC 1-5
sceIpuCSC 1-6
sceIpuFDEC 1-7
sceIpuGetFVdecResult 1-8
sceIpuIDEC 1-9
sceIpuInit 1-10
sceIpuIsBusy 1-11
sceIpuIsError 1-12
sceIpuPACK 1-13
sceIpuReset 1-14
sceIpuRestartDMA 1-15
sceIpuSETIQ 1-16
sceIpuSETTH 1-17
sceIpuSETVQ 1-18
sceIpuStopDMA 1-19
sceIpuSync 1-20
sceIpuVDEC 1-21

Structures

sceIpuDmaEnv

State when IPU-related DMA is stopped

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Structure

typedef struct {	
u_int *d4madr;*	ToIPU channel transfer address (D4_MADR register value)
u_int *d4tadr;*	ToIPU channel tag address (D4_TARD register value)
u_int *d4qwc;*	ToIPU channel transfer data size (D4_QWC register value)
u_int *d4chcr;*	ToIPU channel control information (D4_CHCR register value)
u_int *d3madr;*	FromIPU channel transfer address (D3_MADR register value)
u_int *d3qwc;*	FromIPU channel transfer data size (D3_QWC register value)
u_int *d3chcr;*	FromIPU channel control information (D3_CHCR register value)
u_int *ipubp;*	Bit stream position (IPU_BP register value)
u_int *ipuctrl;*	IPU status (IPU_CTRL register value)
} sceIpuDmaEnv;	

Description

This is a structure for saving the DMAC and IPU state when an IPU-related DMA transfer is stopped by calling the sceIpuStopDMA() function.

The sceIpuRestartDMA() function can be used to return and restart the transfer.

Functions

sceIpuBCLR

Execute BCLR command

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuBCLR(	
int *bp***)**	Bit position for starting decoding among first 128 bits

Calling conditions

Can be called from a thread

Not multithread safe

Description

Clears input FIFO by executing the BCLR command.

The DMA to the input FIFO (toIPU:ch-4) must be stopped before calling this function.

Return value

None

sceIpuBDEC

Execute BDEC command

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuBDEC(

int *mbi,*	Macroblock Intra
	SCE_IPU_BDEC_NONINTRA(0) : Non-intra macroblock
	SCE_IPU_BDEC_INTRA(1): Intra macroblock
int *dcr,*	DC Reset
	SCE_IPU_BDEC_NODCRESET(0): Do not reset DC prediction value
	SCE_IPU_BDEC_DCRESET(1): Reset DC prediction value
int *dt,*	DCT Type
	SCE_IPU_BDEC_FRAMEDCT(0) : frame DCT
	SCE_IPU_BDEC_FIELDDCT(1) : field DCT
int *qsc,*	Quantizer Step Code
int *fb***)**	Forward Bit

Calling conditions

Can be called from a thread

Not multithread safe

Description

Performs block decoding by executing the BDEC command.

Return value

None

sceIpuCSC

Execute CSC command

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuCSC(	
int *ofm,*	Output Format
	SCE_IPU_CSC_RGB32(0): RGB32
	SCE_IPU_CSC_RGB16(1): RGB16
int *dte,*	Dither Enable
	SCE_IPU_CSC_NODITHER(0): No dither
	SCE_IPU_CSC_DITHER(1): Dither
	(Valid only when *ofm* = RGB16)
int *mbc***)**	Macroblock Count
	Number of macroblocks to be converted

Calling conditions

Can be called from a thread

Not multithread safe

Description

Performs color space conversion by executing the CSC command.

Return value

None

sceIpuFDEC

Execute FDEC command

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuFDEC(	
int *fb***)**	Forward Bit

Calling conditions

Can be called from a thread

Not multithread safe

Description

Decodes fixed-length data by executing the FDEC command.

The decoding result can be obtained using the sceIpuGetFdecResult() function.

Return value

None

sceIpuGetFVdecResult

Get execution result of FDEC or VDEC command

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

u_int sceIpuGetFVdecResult(void)

Calling conditions

Can be called from a thread

Not multithread safe

Description

Reads the execution result of the FDEC or VDEC command that was executed immediately before this command.

Return value

Data that was decoded by the preceding FDEC or VDEC command

sceIpuIDEC

Execute IDEC command

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuIDEC(

int *ofm,*	Output Format
	SCE_IPU_IDEC_RGB32(0): RGB32
	SCE_IPU_IDEC_RGB16(1): RGB16
int *dte,*	Dither Enable
	SCE_IPU_IDEC_NODITHER(0): No dither
	SCE_IPU_IDEC_DITHER(1): Dither
	(Valid only when *ofm* = RGB16)
int *sgn,*	Pseudo Sign Offset
	SCE_IPU_IDEC_NOOFFSET(0): No offset
	SCE_IPU_IDEC_OFFSET(1): offset -128
int *dtd,*	DT Decode
	SCE_IPU_IDEC_NODTDECODE(0): Do not decode Dct Type
	SCE_IPU_IDEC_DTDECODE(1): Decode Dct Type
int *qsc,*	Quantizer Step Code
int *fb***)**	Forward Bit

Calling conditions

Can be called from a thread

Not multithread safe

Description

Performs intra-decoding by executing the IDEC command.

Return value

None

sceIpuInit

Initialize IPU and IPU library

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuInit(void)

Calling conditions

Can be called from a thread

Not multithread safe

Description

Resets the IPU and clears the IPU_in_FIFO.

Initializes the IPU library.

Return value

None

sceIpuIsBusy

Check IPU state

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

int sceIpuIsBusy(void)

Calling conditions

Can be called from a thread

Not multithread safe

Description

Returns a value indicating whether the IPU is busy.

Return value

0: Stopped

Other: Busy

sceIpuIsError

Check IPU state

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

int sceIpuIsError(*void***)**

Calling conditions

Can be called from a thread

Not multithread safe

Description

Returns a value indicating whether an error occurred during IPU processing.

This value is automatically cleared every time an IPU command is executed.

Return value

0: No error

Other: Error

sceIpuPACK

Execute PACK command

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuPACK(

int *ofm,*	Output Format
	SCE_IPU_PACK_INDX4(0): INDX4
	SCE_IPU_PACK_RGB16(1): RGB16
int *dte,*	Dither Enable
	SCE_IPU_PACK_NODITHER(0): No dither
	SCE_IPU_PACK_DITHER(1): Dither
int *mbc*)	Macroblock Count
	Number of macroblocks to be converted

Calling conditions

Can be called from a thread

Not multithread safe

Description

Performs format conversion by executing the PACK command.

Return value

None

sceIpuReset

Reset IPU

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuReset(void)

Calling conditions

Can be called from a thread

Not multithread safe

Description

Resets the IPU.

Return value

None

sceIpuRestartDMA

Restart IPU-related DMA

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuRestartDMA(	
sceIpuDmaEnv ***env*)**	Pointer to the structure in which the DMA and IPU states were previously saved

Calling conditions

Can be called from a thread

Not multithread safe

Description

Restarts toIPU(ch-4) and fromIPU(ch-3) DMA operations using the saved DMA and IPU states.

Return value

None

sceIpuSETIQ

Execute SETIQ command

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuSETIQ(

int *iqm,*	Intra IQ Matrix
	SCE_IPU_SETIQ_INTRA(0): Intra quantization matrix
	SCE_IPU_SETIQ_NONINTRA(1): Non-intra quantization matrix
int *fb*)	Forward Bit

Calling conditions

Can be called from a thread

Not multithread safe

Description

Sets the IQ table by executing the SETIQ command.

Return value

None

sceIpuSETTH

Execute SETTH command

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuSETTH(

int *th1,*	Semi-transparent threshold
int *th0***)**	Transparent threshold

Calling conditions

Can be called from a thread

Not multithread safe

Description

Sets threshold values by executing the SETTH command.

These threshold values are used when performing color conversion using the CSC command.

Return value

None

sceIpuSETVQ

Execute SETVQ command

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuSETVQ(void)

Calling conditions

Can be called from a thread

Not multithread safe

Description

Sets the VQCLUT table by executing the SETVQ command.

Return value

None

sceIpuStopDMA

Stop IPU-related DMA

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuStopDMA(	
sceIpuDmaEnv **env***)**	Pointer to structure for saving internal state

Calling conditions

Can be called from a thread

Not multithread safe

Description

Safely stops toIPU(ch-4) and fromIPU(ch-3) DMA operations then saves the DMA state and IPU internal state.

Return value

None

sceIpuSync

Synchronizes with IPU processing

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

int sceIpuSync(

int *mode,*	0: Blocks while IPU is busy. 1: Terminates immediately and returns IPU status. Performs the same operation as the sceIpuIsBusy() function.
unsigned short *timeout***)**	Specifies the timeout value when *mode* = 0. The units of the specified value are in number of horizontal lines. *timeout* = 0: Use the default timeout value in the library. *timeout* > 0: Use the specified value as the timeout value.

Calling conditions

Can be called from a thread

Not multithread safe

Description

Determines whether the IPU is busy and waits for IPU operation to end.

Determines whether the IPU is busy and returns IPU status.

Return value

When mode = 0

Non-negative value: Normal termination
Negative value: Abnormal termination (timeout occurred)

When mode = 1

0: IPU is not busy
Positive value: IPU is busy

sceIpuVDEC

Execute VDEC command

Library	*Introduced*	*Documentation last modified*
libipu	1.1	March 26, 2001

Syntax

void sceIpuVDEC(

int *tbl,*	VLC table
	SCE_IPU_VDEC_MBAI(0): Macroblock Address Increment
	SCE_IPU_VDEC_MBTYPE(1): Macroblock Type
	SCE_IPU_VDEC_MOTIONCODE(2): Motion Code
	SCE_IPU_VDEC_DMVECTOR(3): DMVector
int *fb***)**	Forward Bit

Calling conditions

Can be called from a thread

Not multithread safe

Description

Decodes the symbols specified by *tbl* by executing the VDEC command.

The decoding result can be obtained using the sceIpuGetVdecResult() function.

Return value

None

Chapter 2: MPEG Library
Table of Contents

Structures **2-3**

sceMpeg 2-3

sceMpegCbData 2-5

sceMpegCbDataError 2-6

sceMpegCbDataStr 2-7

sceMpegCbDataTimeStamp 2-8

Functions **2-9**

sceMpegAddBs 2-9

sceMpegAddCallback 2-10

sceMpegAddStrCallback 2-11

sceMpegCreate 2-12

sceMpegDelete 2-13

sceMpegDemuxPss 2-14

sceMpegDemuxPssRing 2-15

sceMpegGetDecodeMode 2-16

sceMpegGetPicture 2-17

sceMpegGetPictureRAW8 2-19

sceMpegInit 2-20

sceMpegIsEnd 2-21

sceMpegIsRefBuffEmpty 2-22

sceMpegReset 2-23

sceMpegSetDecodeMode 2-24

Structures

sceMpeg

MPEG decoder

Library	*Introduced*	*Documentation last modified*
libmpeg	1.1	March 31, 2000

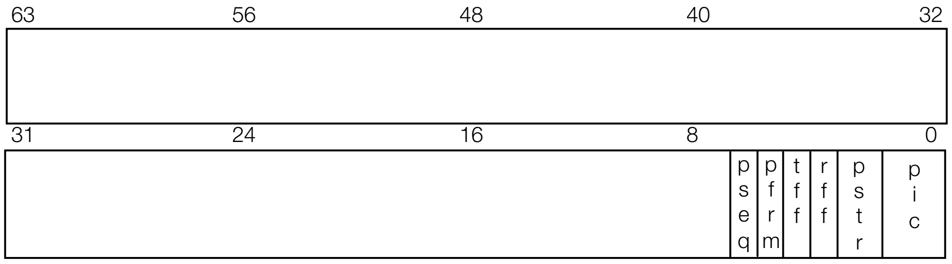
Structure

typedef struct {

int *width;*	Width of decode image (set up after decoding first picture)
int *height;*	Height of decode image (set up after decoding first picture)
int *frameCount;*	Number of frames in decode image (changes according to decode)
long *pts;*	Presentation Time Stamp; indicates the time at which image is to be displayed
long *dts;*	Decoding Time Stamp; indicates the time at which image is to be decoded
u_long *flags;*	Flags related to decoding (see figure and table below)
long *pts2nd;*	reserved for future use
long *dts2nd;*	reserved for future use
u_long *flags2nd;*	reserved for future use
void **sys;*	System data (used internally by the decoder)

} sceMpeg;

Figure 2-1: Flags



<pic>: picture_coding_type

000: Reserved
001: I picture
010: P picture
011: B picture
100: D picture (mpeg1)
101: Reserved
110: Reserved
111: Reserved

<pstr>: picture_structure

00: reserved
01: Top Field
10: Bottom Field
11: Frame Picture

<rff>: **repeat_first_field**

<tff>: **top_field_first**

<pfrm>: **progressive_frame**

<pseq>: **progressive_sequence**

Description

The sceMpeg structure is a structure used for the implementation of an MPEG decoder. The *width* and *height* members are set up when the first picture is decoded. *frameCount* is a running count from the start of the picture that was decoded previously. The value changes as decoding progresses.

pts is the PTS (Presentation Time Stamp) of the picture that was decoded previously. *dts* is the DTS (Decoding Time Stamp) of the picture that was decoded previously. PTS and DTS are timestamps defined in MPEG2 and are used to determine the timing for presentation and decoding. These values are expressed as values for an imaginary clock operating at 90 KHz. In other words, 1 sec = 90000 ticks. *flags* holds various attributes for each decoded sequence and decoded image. *sys* is used by the system during decoding.

sceMpegCbData

MPEG callback data

Library	*Introduced*	*Documentation last modified*
libmpeg	1.3	February 29, 2000

Structure

typedef union {	
sceMpegCbType *type;*	Callback type
sceMpegCbDataError *error;*	Structure associated with sceMpegCbError
sceMpegCbDataTimeStamp *ts;*	Structure associated with sceMpegCbTimeStamp
sceMpegCbDataStr *str;*	Structure associated with sceMpegCbStr
} sceMpegCbData;	

Description

sceMpegCbData is a shared structure having the structures shown above as members. All members have type as the first member which allows the callback type to be determined. sceMpegCbData is used when a callback function of a type that does not define special structures is called. sceMpegCbData is passed as the second argument to the function.

Sample callback function:

```
int mpegNodata(sceMpeg *mp,
             sceMpegCbData *cbdata, void *anyData) {

    ... // get next data and send it to IPU

    return 1;
}
```

sceMpegCbDataError

MPEG error callback

Library	*Introduced*	*Documentation last modified*
libmpeg	1.3	February 29, 2000

Structure

typedef struct {	
sceMpegCbType *type;*	Callback type (sceMpegCbError)
char **errMessage;*	Error message
} sceMpegCbDataError;	

Description

sceMpegCbDataError is a callback data structure associated with sceMpegCbError. If the callback function associated with sceMpegCbError is called, it is passed as the second argument to the callback function.

The sceMpegCbError callback function is called when some sort of error has taken place in the decoder. An error message is stored in errMessage.

Sample callback function:

```
int mpegError(sceMpeg *mp,
       sceMpegCbDataError *cberror, void *anyData) {
    printf("%s\n", cberror->errMessage);
    return 1;
}
```

sceMpegCbDataStr

MPEG stream callback data

Library	*Introduced*	*Documentation last modified*
libmpeg	1.3	February 29, 2000

Structure

typedef struct {

sceMpegCbType *type;*	Callback type (sceMpegCbStr)
u_char **header;*	Start of packet header
u_char **data;*	Start of packet data
u_int *len;*	Length of packet data
long *pts;*	PTS (Presentation time stamp) value; valid only when pts >= 0
long *dts;*	DTS (Decoding time stamp) value; valid only when dts >= 0

} *sceMpegCbDataStr;*

Description

sceMpegCbDataStr is a callback data structure associated with sceMpegCbStr. When the callback function associated with sceMpegCbStr is called, it is passed on to the callback function as the second argument.

The sceMpegCbStr callback function is called when stream data that was registered beforehand is found while PSS data is being demultiplexed. header contains the starting position of the PES packet, *data* contains the starting position of the data in the PES packet, *len* contains the size of the data, and *pts/dts* contain PTS/DTS respectively. The callback function must use *data* and *len* to extract the data section and save it to a separate area. For future operations, it is necessary to associate the data position and *pts/dts* in some manner. The callback function must return a 0 if demultiplexing is to be stopped and a 1 otherwise.

Sample callback function:

```
int func() {
    sceMpeg theMepg;
    // choose MPEG video 0
    sceMpegAddStrCallback(&theMpeg,
       sceMpegStrM2V,      // MPEG2 video stream
       0,                  // stream number 0
       videoCallback,      // callback function; see below
       NULL
    );
}

// callback function for sceMpegCbStr
int videoCallback(sceMpeg *mp,
       sceMpegCbDataStr *cbstr, void *anyData) {

    if (/* video input buff is not full*/) {

       ... // copy (cbstr->data, cbstr->len)
       ... // to video input buffer
       ... // save cbstr->pts/cbstr->dts value
       return 1;
    }
    return 0;
}
```

sceMpegCbDataTimeStamp

MPEG timestamp callback data

Library	*Introduced*	*Documentation last modified*
libmpeg	1.3	February 29, 2000

Structure

typedef struct {	
sceMpegCbType *type;*	Callback type (sceMpegCbTimeStamp)
long *pts;*	PTS (Presentation time stamp) value; valid only when pts >= 0
long *dts;*	DTS (Decoding time stamp) value; valid only when dts >= 0
} sceMpegCbDataTimeStamp;	

Description

sceMpegCbDataTimeStamp is a callback data structure associated with sceMpegCbTimeStamp. If a callback function associated with sceMpegCbTimeStamp is called, it is passed as the second argument to the callback function.

The sceMpegCbTimeStamp callback function is called when the decoder wants to get the current PTS and DTS associated with the data being decoded. The callback function must analyze the data position being decoded from D4_MADR, IPU_CTRL, IPU_BP, etc., and return the PTS/DTS corresponding to that position. Members pts and dts are used.

Sample callback function:

```
int mpegTimeStamp(sceMpeg *mp,
       sceMpegCbDataTimeStamp *cbts, void *anyData) {
    long pts_value, dts_value;

    pts_value = ....
    dts_value = ....

    cbts->pts = pts_value;
    cbts->dts = dts_value;
    return 1;
}
```

Functions

sceMpegAddBs

Add input bit stream to MPEG decoder

Library	*Introduced*	*Documentation last modified*
libmpeg	1.1	March 26, 2001

Syntax

int sceMpegAddBs(

sceMpeg **mp,*	MPEG decoder
u_long128 **bs,*	Bit stream to be decoded
int *bs_size***);**	Size of bit stream to be decoded

Calling conditions

Can be called from a thread

Not multithread safe

Description

Sets up an MPEG2/MPEG1 bit stream for the decoder.

The decoder internally sets up DMA ch4.

Return value

None

sceMpegAddCallback

Set up callback function

Library	*Introduced*	*Documentation last modified*
libmpeg	1.3	March 26, 2001

Syntax

sceMpegCallback sceMpegAddCallback(

sceMpeg **mp,*	MPEG decoder
sceMpegCbType *type,*	Callback type
sceMpegCallback *callback,*	Callback function to be registered
void **anyData*)	Arbitrary data

Calling conditions

Can be called from a thread

Not multithread safe

Description

Registers a callback function for an MPEG decoder. *type* is the type of the callback being registered. *callback* is the callback function being registered. *anyData* can be any data and is passed to the callback function as the third parameter. *anyData* can be used freely by the application.

Return value

Callback already registered

sceMpegAddStrCallback

Set up stream callback

Library	*Introduced*	*Documentation last modified*
libmpeg	1.3	March 26, 2001

Syntax

sceMpegCallback sceMpegAddStrCallback(

sceMpeg **mp,*	MPEG decoder
sceMpegStrType *strType,*	Stream type
int *strNumber,*	Stream number
sceMpegCallback *callback,*	Callback function to be registered
void **anyData***)**	Arbitrary data

Calling conditions

Can be called from a thread

Not multithread safe

Description

Registers callback function for MPEG decoder. The type of callback is sceMpegCbStr. Functions can be registered for individual stream types. Stream types are specified by *strType* and *strNumber*. *callback* indicates the callback function to be registered. *anyData* can be any arbitary data and is passed to the callback function as the third parameter. *anyData* can be used freely by the application.

Return value

Callback already registered

sceMpegCreate

Create MPEG decoder

Library	*Introduced*	*Documentation last modified*
libmpeg	1.1	January 4, 2002

Syntax

int sceMpegCreate(	
sceMpeg **mp,*	Structure associated with decoder to be created
u_char **work_area,*	Work area for decoder
int *work_area_size***)**	Size of work area for decoder

Calling conditions

Can be called from a thread

Not multithread safe

Description

sceMpegCreate() creates an MPEG decoder. A pointer to an sceMpeg structure assigned by the application is passed as the mp parameter. The work area for MPEG decoding and the size of the work area are passed to the *work_area* and *work_area_size* parameters.

The work area must be allocated by the application. The size of the work area can be determined using the following macro:

```
SCE_MPEG_BUFFER_SIZE(w, h)    // w: max_width, h: max_height
```

After a decoder has been created using sceMpegCreate(), an application can identify the MPEG decoder using the sceMpeg structure specified by mp.

Return value

None

sceMpegDelete

Delete MPEG decoder

Library	*Introduced*	*Documentation last modified*
libmpeg	1.1	March 26, 2001

Syntax

int sceMpegDelete(	
sceMpeg ****mp*)**	MPEG decoder

Calling conditions

Can be called from a thread

Not multithread safe

Description

Deletes the specified MPEG decoder.

Return value

None

sceMpegDemuxPss

Demultiplex PSS

Library	*Introduced*	*Documentation last modified*
libmpeg	1.3	March 26, 2001

Syntax

int sceMpegDemuxPss(	
sceMpeg **mp,*	MPEG decoder
u_char **pss,*	Pointer to PSS data
int *pss_size*)	Size of PSS data

Calling conditions

Can be called from a thread

Not multithread safe

Description

The data area provided by *pss*, *pss_size* is analyzed. If a registered stream data is found, the associated callback function is called. The target stream and the associated callback function must be registered beforehand using sceMpegAddStrCallback().

sceMpegDemuxPss() is generally used for demultiplexing PSS. sceMpegDemuxPss() continues processing until the end of the specified area is reached or until a callback function that was called returns a 0. The return value is the length of the processed data in bytes.

Return value

Length of processed data (in bytes)

sceMpegDemuxPssRing

Demultiplex PSS in ring buffer

Library	*Introduced*	*Documentation last modified*
libmpeg	1.3	March 26, 2001

Syntax

int sceMpegDemuxPssRing(

sceMpeg **mp,*	MPEG decoder
u_char **pss,*	Pointer to PSS data
int *pss_size,*	Size of PSS data
u_char **buf_top,*	Pointer to top of ring buffer
int buf_*size***)**	Size of ring buffer

Calling conditions

Can be called from a thread

Not multithread safe

Description

The data area provided by *pss*, *pss_size* is analyzed. If a registered stream data is found, the associated callback function is called. It is assumed that data is placed in a ring buffer specified by *buf_top* and *buf_size*. In other words, once the data at the position *buf_top* + *buf_size* - 1 is processed, processing is continued from *buf_top*. The target stream and the associated callback function must be registered beforehand using sceMpegAddStrCallback(). sceMpegDemuxPssRing() is generally used for demultiplexing PSS.

sceMpegDemuxPssRing() continues processing until the end of the specified area is reached or until a callback function that was called returns a 0. The return value is the length of the processed data in bytes.

Return value

Length of processed data (in bytes)

sceMpegGetDecodeMode

Get decode mode

Library	*Introduced*	*Documentation last modified*
libmpeg	1.3	March 26, 2001

Syntax

void sceMpegGetDecodeMode(	
sceMpeg **mp,*	MPEG decoder
int **ni,*	Area to store number of I-pictures played back in 1 GOP
int **np,*	Area to store number of P-pictures played back in 1 GOP
int **nb***)**	Area to store number of B-pictures played back in 1 GOP

Calling conditions

Can be called from a thread

Not multithread safe

Description

Gets the decode mode. For information about the decode mode, refer to the description for sceMpegeSetDecodeMode().

Return value

None

sceMpegGetPicture

Decode 1 picture with MPEG decoder (RGB32)

Library	*Introduced*	*Documentation last modified*
libmpeg	1.1	March 26, 2001

Syntax

int sceMpegGetPicture(

sceMpeg **mp,*	MPEG decoder
sceIpuRGB32 **rgb32,*	Area storing decoded picture data
int *mbcount***)**	Size of area storing decoded picture data (unit: number of sceIpuRGB32 = number of macroblocks)

Calling conditions

Can be called from a thread

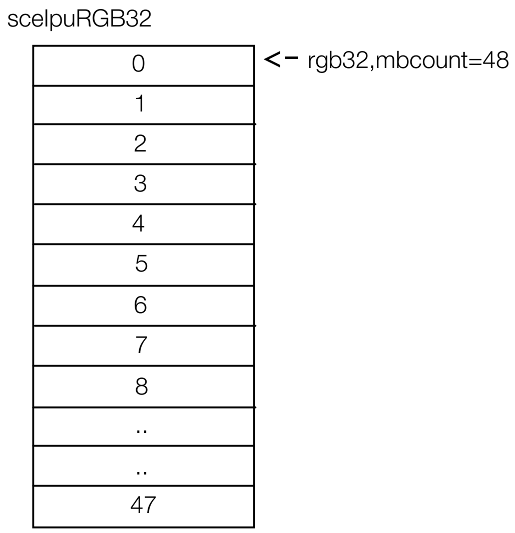
Not multithread safe

Description

sceMpegGetPicture() decodes one picture's worth of data. The decoded picture is stored in memory as a data array in the sceIpuRGB32 format. The area in which the data is stored is specified by the *rgb32* parameter. The sequence in which data is stored in the sceIpuRGB32 memory and the sequence of corresponding macroblocks in the image are as shown below.

Example for 128x96:

= Sequence in memory =



= Sequence of macroblocks in image =

0	6	12	18	24	30	36	42
1	7	13	19	25	31	37	43
2	8	14	20	26	32	38	44
3	9	15	21	27	33	39	45
4	10	16	22	28	34	40	46
5	11	17	23	29	35	41	47

To display the decoded picture data correctly, data must be rearranged correctly using Source Chain DMA. For information about the sceIpuRGB32 format itself, please refer to the libipu documentation.

Return value

Non-negative: Successful completion

Negative: Failed

sceMpegGetPictureRAW8

Decode 1 picture with MPEG decoder (RAW8)

Library	*Introduced*	*Documentation last modified*
libmpeg	1.3	March 26, 2001

Syntax

int sceMpegGetPictureRAW8(	
sceMpeg **mp,*	MPEG decoder
sceIpuRAW8 **raw8,*	Area in which decoded picture data is stored
int *mbcount***);**	Size of area in which decoded picture data is stored (unit: number of sceIpuRAW8s = number of macroblocks)

Calling conditions

Can be called from a thread

Not multithread safe

Description

sceMpegGetPictureRAW8() decodes one picture's worth of data. The decoded picture is stored in memory as a data array in the sceIpuRAW8 format. The area in which the data is stored is specified by the *raw8* parameter. The sequence in which data is stored in the sceIpuRAW8 memory and the sequence of the corresponding macroblocks in the image are the same as with sceIpuRGB32 for sceMpegGetPicture().

Return value

Non-negative: Successful completion

Negative: Failed

sceMpegInit

Initialize libmpeg

Library	*Introduced*	*Documentation last modified*
libmpeg	1.1	March 26, 2001

Syntax

int sceMpegInit(void)

Calling conditions

Can be called from a thread

Not multithread safe

Description

Initializes the MPEG library.

Initializes DMA ch3, ch4.

Return value

None

sceMpegIsEnd

Check to see if MPEG bit stream is finished

Library	*Introduced*	*Documentation last modified*
libmpeg	1.1	March 26, 2001

Syntax

int sceMpegIsEnd(	
sceMpeg **mp***)**	MPEG decoder

Calling conditions

Can be called from a thread

Not multithread safe

Description

Checks to see if bit stream has been decoded to the end (sequence_end_code).

Return value

Non-0: Decoding has been performed to the end of the bit stream

0: Decoding has not been performed to the end of the bit stream

sceMpegIsRefBuffEmpty

Check to see if the reference image buffer in the MPEG decoder is empty

Library	*Introduced*	*Documentation last modified*
libmpeg	1.3	March 26, 2001

Syntax

int sceMpegIsRefBuffEmpty(	
sceMpeg **mp***)**	MPEG decoder

Calling conditions

Can be called from a thread

Not multithread safe

Description

Checks to see if the reference image buffer in the MPEG decoder is empty or not. The MPEG decoder stores a reference image used for decoding in a reference image buffer. The reference image buffer is empty before decoding is begun and after decoding is finished and all images have been output.

Return value

Non-0: reference image buffer is empty

0: reference image buffer is not empty

sceMpegReset

Reset MPEG decoder

Library	*Introduced*	*Documentation last modified*
libmpeg	1.1	March 26, 2001

Syntax

int sceMpegReset(	
sceMpeg **mp***)**	MPEG decoder

Calling conditions

Can be called from a thread

Not multithread safe

Description

Reinitializes the specified MPEG decoder.

Return value

None

sceMpegSetDecodeMode

Set up decode mode

Library	*Introduced*	*Documentation last modified*
libmpeg	1.3	March 26, 2001

Syntax

void sceMpegSetDecodeMode(

sceMpeg **mp,*	MPEG decoder
int *ni,*	Number of I-pictures played back in 1 GOP constant SCE_MPEG_DECODE_ALL or 0
int *np,*	Number of P-pictures played back in 1 GOP constant SCE_MPEG_DECODE_ALL or a number 0 or higher
int *nb*)	Number of B-pictures played back in 1 GOP constant SCE_MPEG_DECODE_ALL or a number 0 or higher

Calling conditions

Can be called from a thread

Not multithread safe

Description

A decode mode is set up. The decode mode is a mode that determines how many I-, P-, and B-pictures are played back in 1 GOP. For normal playback, specify the constant SCE_MPEG_DECODE_ALL for *ni*, *np*, and *nb*.

If sceMpegSetDecodeMode() is not called, the default value for the decoder will be this value.

When performing fast-forward playback, using the sample setting shown below will allow decoding of intermediate images to be skipped.

<Skip B-pictures>

ni = SCE_MPEG_DECODE_ALL

np = SCE_MPEG_DECODE_ALL

nb = 0

<Skip B-pictures, decode only 2 P-pictures>

ni = SCE_MPEG_DECODE_ALL

np = 2

nb = 0

Return value

None