PlayStation®2 EE Library Overview
Release 2.4

Movie Libraries

Summary Table of Contents

About This Manual

This is the Runtime Library Release 2.4 version of the *PlayStation®2 EE Library Overview - Movie Libraries* manual.

The purpose of this manual is to provide overview-level information about the PlayStation®2 movie libraries. For related descriptions of the PlayStation®2 movie library structures and functions, refer to the *PlayStation®2 EE Library Reference - Movie Libraries.*

Changes Since Last Release

Chapter 2: MPEG Library

- In the " Library Overview" section, the size described as 720x480 has been changed to 720x576 and a description of picture structure has been added.

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

Library Overview

libipu is a low-level library used for controlling the image data processor (IPU). The library defines functions for executing individual IPU commands as well as functions for controlling overall IPU operations.

libmpeg is provided as a higher-level library than libipu for handling MPEG streams.

Related Files

The following files are required for libipu:

Table 1-1

Category	Filename
Library file	libipu.a
Header file	libipu.h

Reference Material

For details on IPU features and commands, please see "6. IPU: image data processor" from the "EE User's Manual".

Sample Programs

The following sample programs that use libipu are provided:

- **sce/ee/sample/ipu/ezmovie**
  Shows how to play back an IPU stream from main memory.
- **sce/ee/sample/ipu/ezcube**
  Shows how to use an IPU stream in main memory as a texture.
- **sce/ee/sample/ipu/ezcube3**
  Shows how to simultaneously use multiple IPU streams in main memory as textures.

Creating IPU Stream Files (.ipu)

For the IPU stream files (.ipu) used in the above sample programs, create MPEG2 video stream data (.m2v) consisting only of I-Pictures using commercial MPEG encoder software. This data can then be converted and .ipu files created using the stream converter (ps2str).

The procedure is shown below when using the Windows GUI version of the stream converter (ps2strw).

1. **Create .mv2 data consisting only of I-Pictures**

   Use commercial MPEG encoder software to create MPEG2 video stream data.
   At this time set the data to "I-Picture" only.

2. **Start ps2strw**

3. **Verify file format**

   Open the Info tab and either enter the filename of the created .m2v file in the File column or drag and drop the file from Explorer. Then, click the Show button to display the file information. If the following display appears, I-Picture-only data will have been created.

```
IPU compliant    :    yes
```

If instead, this appears:

```
IPU compliant    :    no
```

then reconfirm the output settings of the encoder.

4. **Read and convert .m2v files**

   Open the IPU Convert tab, click the Add... button and either select the .m2v file or drag and drop the file from Explorer. As necessary, enter the folder name of the output destination in the Output Folder column and click the Start button to create the .ipu file.

Chapter 2: MPEG Library

Library Overview

libmpeg is a library for decoding MPEG2/MPEG1 video. Compatible MPEG2 (MPEG1) streams that are supported are as follows:

- Main Profile at Main Level(MP@ML)
- Simple Profile at Main Level(SP@ML)
- Size: 32x32 minimum, 720x576 maximum, height and width (in pixels) are multiples of 16
- Picture structure: frame structure (progressive frames preferred) and field structure

Related Files

The following files are required to use libmpeg. libmpeg uses libipu internally, so libipu library files are also required.

Table 2-1

Category	File Name
Library files	libmpeg.a
	libipu.a
Header file	libmpeg.h

Resources Used

In addition to main memory, libmpeg uses the following hardware resources to perform decoding operations.

Table 2-2

Resources	Usage
DMA ch4	Throughout
DMA ch3	Only in sceMpegInit(), sceMpegGetPicture(), sceMpegGetPictureRAW8()
ScratchPad RAM	Only in sceMpegGetPicture(), sceMpegGetPictureRAW8()

Make sure that these resources will not be used elsewhere while they are being used by libmpeg.

Sample Programs

The sample programs that use libmpeg are as follows:

- **sce/ee/sample/mpeg/ezmpeg**
  Shows how to play back MPEG2 video elementary streams (m2v) from main memory.
- **sce/ee/sample/mpeg/mpegstr**
  Shows how to play back an MPEG2 stream (PSS) from a CD/DVD, or a hard disk on the host. The IPU is used for color conversion.
- **sce/ee/sample/mpeg/mpegvu1**
  Shows how to play back an MPEG2 stream (PSS) from a CD/DVD, or a hard disk on the host. VU1 is used for color conversion.

Creating MPEG2 Program Stream Files (.pss)

The .pss files used in the sample program mpegstr above can be created using the stream converter ps2str, from video streams and wav files created using commercial MPEG encoder software. The procedure is shown below for the Windows GUI version of the stream converter (ps2strw).

1. Create video stream (.m2v)

Use commercial MPEG encoder software to create data that satisfies the following conditions.

- Main Profile at Main Level(MP@ML)
- Simple Profile at Main Level(SP@ML)
- Size: Min. 32 X 32 ~ Max. 720 X 576   Height and width (number of pixels) is a multiple of 16

2. Create audio stream (.ads)

Prepare a 48 KHz .wav file, use ps2strw to convert the file and create an .ads file. The procedure is as follows.

1. Start ps2strw.
2. Open the SPU Encode tab.
3. Read the .wav file

   Either click the Add... button to open the .wav file or drag and drop from Explorer.
4. Convert the file

As necessary, enter the folder name of the output destination in the Output Folder column and click the Start button.

3. Multiplex the video stream and audio stream

Use ps2strw to multiplex the .m2v and .ads files to create a .pss file.

1. Start ps2strw.
2. Open the Mux tab.
3. Click the Add New button.

   The Program Stream Setting dialog box will be displayed.
4. Click the Add button to register the .m2v file.

   The Stream Setting dialog box will be displayed. Enter the filename of the .m2v file in the Source File column. Confirm that the Stream column changes to Video and click the OK button.
5. Click the Add button once again to register the .ads file.

   The Stream Setting dialog box will be displayed. Enter the filename of the .ads file in the Source File column. Confirm that the Stream column changes to PCM and click the OK button.
6. Save the registered file.

   Click the OK button to close the Program Stream Setting dialog box. Registration details can be saved here. As necessary, save the file.
7. Click the Start button

   Click the Start button to return to the Mux tab. The file will be converted and a .pss file created.

Using ADPCM Sound

The sample program mpegstr is only compatible with straight PCM as sound data although .pss files and the stream converter are compatible with ADPCM streams.

When using ADPCM the frequency of original .wav files is arbitrary. Use caution when specifying ADPCM in the Stream column when registering an .ads file in the Stream Setting dialog box, if the AD-PCM checkbox in the SPU Encode tab is checked when creating an .ads file.

Restrictions and Notes

- Only one sceMpeg structure can be created in a program. Creating multiple structures will result in unpredictable behavior.
- The current version does not support the clock (STC) feature that uses to the PTS/DTS timestamp. If this is required, the PTS/DTS recorded in the PSS must be fetched.

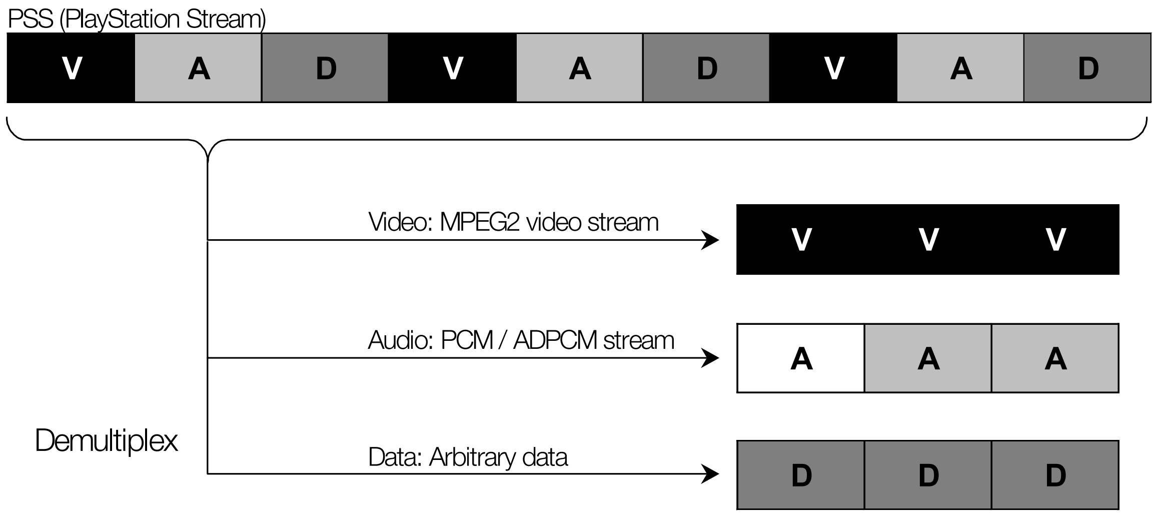
Description of Features

Demultiplexing Streams

libmpeg can demultiplex multiplexed MPEG2 Program Streams and extract elementary streams of video/audio/other data.

Streams are structured as follows:

Figure 2-1



Multiplexed Streams

- PSS (PlayStation Stream): PlayStation MPEG2 Program Stream with audio/data
- PSS: (MPEG2 video) + (ADS) + (DATA)

Streams within the PSS are identified by stream type and a stream number.

Audio/data are recorded as MPEG2 private streams.

Elementary Streams

- Video: MPEG2 video m2v extension, etc. Standard MPEG2 video stream
- Audio: ADS(Audio Data Stream) PS2 streaming audio
  a. PCM format: straight PCM 16bit 48KHz 2ch
  b. ADPCM format: SPU2 ADPCM (same as VAG)
- Data: any type of data

The DVD stream structure is indicated below for comparison with PSS.

Multiplexed Streams

- DVD stream (vob): MPEG2 Program Stream with DVD audio/sub-picture
- DVD stream (vob): (MPEG2 video)+(DVD audio)+(sub-picture) ...

(Non-MPEG2 audio) audio/sub-pictures are recorded as MPEG2 private streams.

Elementary Streams

- Video: MPEG2 video
- Audio: MPEG2 audio / Dolby AC3 / PCM / DTS / SDDS
- Sub-pictures: data (captions, etc.)

Callback Mechanism

libmpeg implements continuous data decoding (streaming). In the PlayStation 2, this type of operation requires buffering at various levels in the EE/IOP.

Since there are many possible ways of implementing buffering, libmpeg does not assume a specific buffering system. The buffering system can be set up freely by the application. libmpeg provides a callback mechanism to allow the buffer management performed by the application to work together with libmpeg operations. In callbacks, an application function (callback function) is registered beforehand to be called when a particular event takes place during a libmpeg operation. The callback types and triggers supported by libmpeg are shown below.

Table 2-3

Callback Type	Cause
sceMpegCbError	Decode error generated
sceMpegCbNodata	No more input data
sceMpegCbStopDMA	Application must interrupt DMA transfer to the IPU
sceMpegCbRestartDMA	Application must resume DMA transfer to the IPU
sceMpegCbBackground	CSC processing begun by IPU
sceMpegCbTimeStamp	Data being decoded needs corresponding time stamp
sceMpegCbStr	Target stream data found

The parameters passed to callback functions and the return values from callback functions vary according to the callback type. For details, please refer to the structures defined for individual callbacks as described in the reference documents. The following is a description of the different callback types.

sceMpegCbError

This callback is generated when an error takes place during decoding. If no callback function is registered, an error message is output to the terminal.

sceMpegCbNodata

This callback is generated when no more input data is available for decoding. When this is received by the application, new data must be sent through DMA ch4 to the IPU.

sceMpegCbStopDMA

This callback is generated when DMA ch4 is interrupted due to CSC processing. When decoding MPEG2 streams, the IPU is used twice for a single picture since bitstream decoding and CSC (color conversion) are performed separately.

DMA transfers for bitstream decoding are handled by the application, but DMA transfers for CSC are handled and performed automatically by the decoder (within libmpeg). Thus, when the decoder is performing CSC, this callback is generated to request the application to release DMA ch4 temporarily. The application receives this and temporarily stops transfers through DMA ch4. The application must save the current decoding position.

sceMpegCbRestartDMA

This callback is generated when the decoder finishes CSC processing and waits for the next bitstream. The application must receive this, resume communication on DMA ch4 and send data following the saved decoding position to the IPU.

sceMpegCbBackground

This callback is generated when the decoder begins CSC processing. While CSC processing is taking place, there is no load on the EE core from the decoding operation. Thus, the application can receive this and perform appropriate processing in the background. Note that this background processing cannot make use of the IPU or DMA ch3, 4.

sceMpegCbTimeStamp

This callback is generated when the decoder finds the start of a picture. The application receives this, examines D4_MADR, IPU_CTRL, IPU_BP, determines the decoding position, and provides the decoder with the associated PTS/DTS.

sceMpegCbStr

This callback is generated when the decoder finds a specific elementary stream when demultiplexing PSS. The application receives this and performs appropriate processing corresponding to the elementary stream that was found.

With the other callback types, only one callback function can be registered, but sceMpegCbStr allows a callback function to be registered for each elementary stream. Elementary streams are identified by stream type and a stream number. The valid ranges for stream types and stream numbers are as follows.

Table 2-4

Elementary stream	Stream type	Stream number
MPEG2 video stream	sceMpegStrM2V	0-15
IPU stream	sceMpegStrIPU	0-65535
PCM stream	sceMpegStrPCM	0-65535
ADPCM stream	sceMpegStrADPCM	0-65535
DATA stream	sceMpegStrDATA	0-65535