Library Overview

© 2001 Sony Computer Entertainment Inc.

Publication date: October 2001

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 *Library Overview* manual is supplied pursuant to and subject to the terms of the Sony Computer Entertainment PlayStation® license agreements.

The *Library Overview* 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 *Library Overview* 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.

Table of Contents

About This Manual

This is the Runtime Library Release 2.4 version of the *Library Overview* manual.

This document was formerly named “Software Architecture Overview” and describes the PlayStation 2 software structure, kernel, libraries, etc.

Changes Since Last Release

- The contents of this document have been substantially modified.

  It now deals chiefly with overviews of the libraries and modules. Descriptions of tools have been deleted.

Related Documentation

The “Hardware Architecture Overview” (SysHard) document provides information on hardware architecture and dataflows.

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)

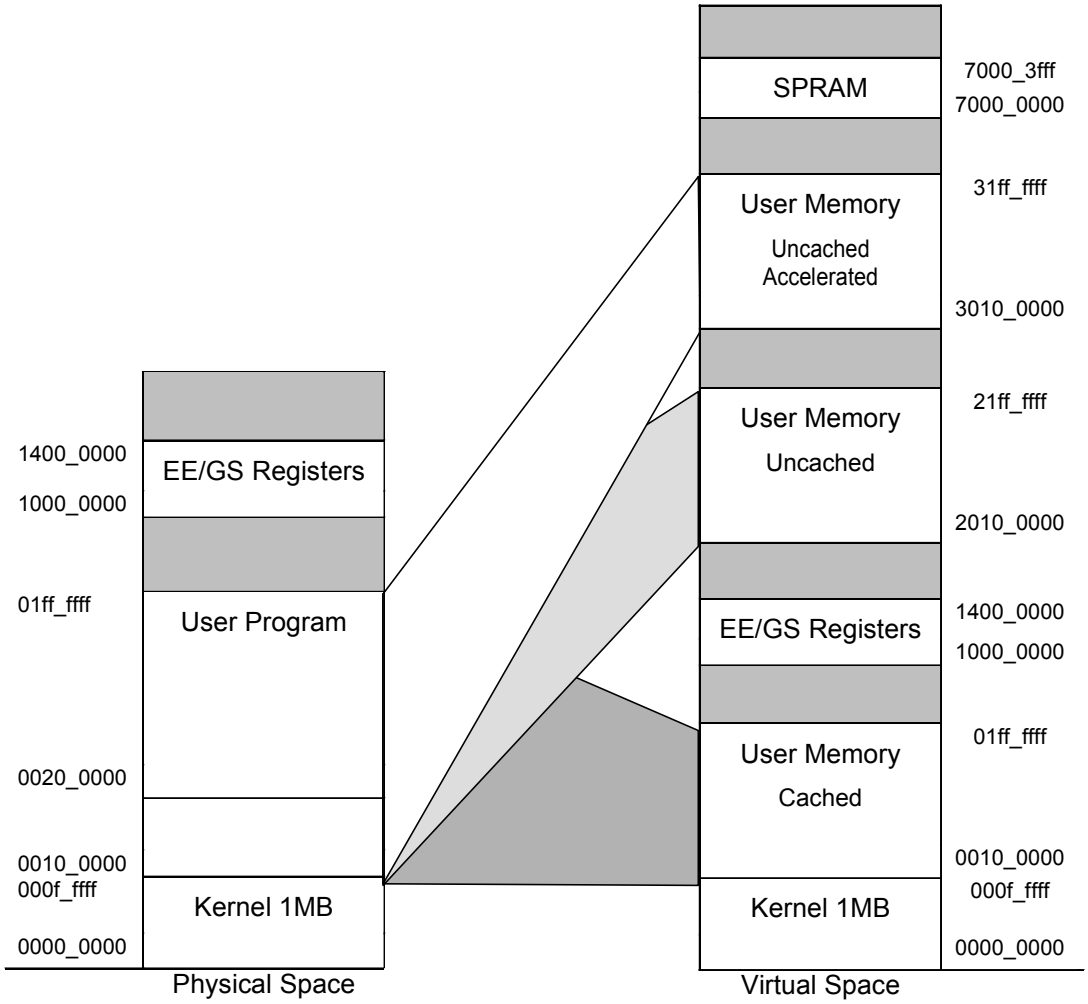
Software Architecture Basics

EE Memory Map

The memory map of the EE kernel is shown below.

User memory is mapped to three overlapping virtual address spaces that have different cache usage modes. By using these three virtual address spaces, the programmer can fine-tune cache usage.

Figure 1: EE Kernel Memory Map



IOP Module Structure

IOP software is constructed as modules that are based on the premise of dynamic linking. When an application starts, each application selects the necessary modules and loads them into memory.

A module is stored as a relocatable object file in ROM or on a CD/DVD-ROM. The file normally has an ".irx" extension and is also referred to as an IRX file.

Modules can be classified as non-resident modules, which are executed once and removed from memory when they have finished their duties, and resident modules, which remain in memory and perform processing by interrupts or requests from other modules. Resident modules include a further sub-

classification known as unloadable resident modules. These are modules which can be removed from memory when they are no longer needed.

Some resident modules provide subroutines for other modules. These are known as resident libraries.

The main IOP modules that are provided by SCE are shown below.

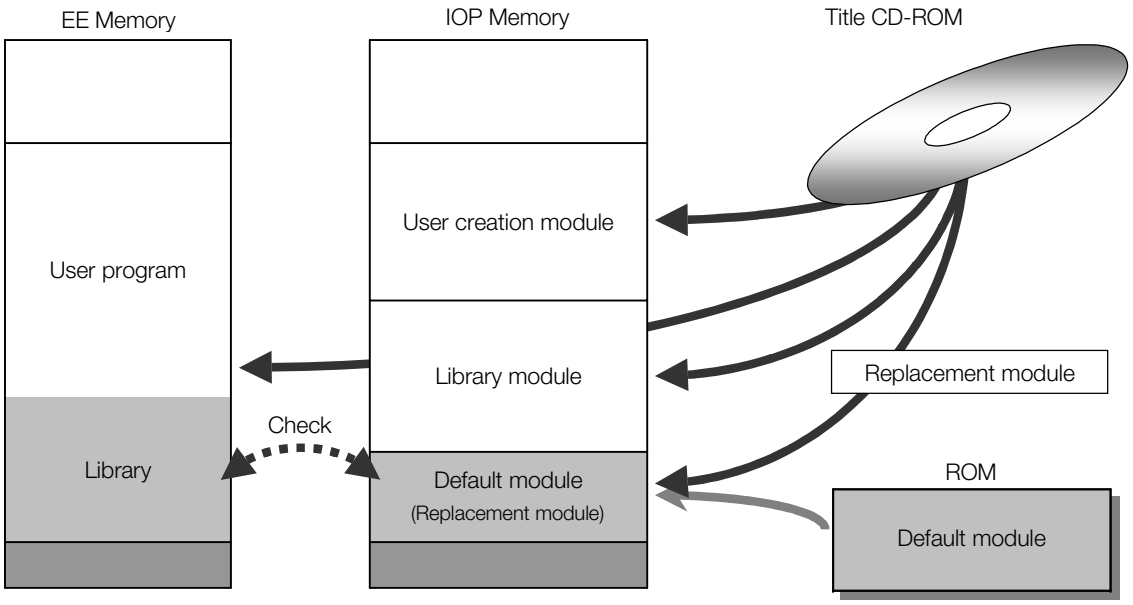
Table 1

Module	Use or Function
sio2man.irx	Serial communication basic module
padman.irx	Controller control module (for libpad2)
dbcman.irx	Device control module (for libpad2)
sio2d.irx	Serial communication module (for libpad2)
dgco.irx, ds1u.irx, ds2u.irx, ds1o.irx, ds2o.irx	Controller driver module (for libpad2)
mtapman.irx	Multitap control module
mcman.irx, mcserv.irx	Memory card control module
sksound.irx, skhsynth.irx, skmidi.irx, sksesq.irx, skmsin.irx	Standard kit library/sound system module
mcxman.irx,mcxserv.irx	PDA control module
libsd.irx, sdrdrv.irx	Low-level sound control module
spucodec.irx	Waveform data code processing module that supports the SPU2
modmidi.irx	CSL: MIDI sequencer
modmsin.irx	CSL: MIDI stream generation
modsesq.irx	CSL: Sound effect sequencer
modsein.irx	CSL: Sound effect stream generation
modssyn.irx	CSL: Software synthesizer
modhsyn.irx	CSL: Hardware synthesizer
modmono.irx	CSL: MIDI monophonic
moddelay.irx	CSL: MIDI latency adjustment
usbd.irx	USB driver module
usbmload.irx	USB module loader
ilink.irx,ilsock.irx	i.LINK driver module
inet.irx	Network library (INET protocol stack)
inetctl.irx	Network (INET) configuration library
netcnf.irx	Common network configuration library
smap.irx	HDD Ethernet driver module
pppoe.irx	PPPoE driver module
atad.irx	Device module to which network adapter is connected
dev9.irx	Hard disk driver module
hdd.irx	Partition management module
pfs.irx	PlayStation File System module

Replacing Default Modules

The most basic modules are loaded from ROM at startup (from flash ROM in the DTL-T10000). These modules are called default modules. However, because the versions of the default modules loaded in ROM of the actual machine are old, title applications must reboot the IOP immediately after startup so that other library modules with the same versions as the default modules (replacement modules) can be loaded from the CD/DVD-ROM.

Figure 2: Replacing IOP default modules



The specific names of replacement modules are shown below.

cdvdman.irx, cdvdfsv.irx, eesync.irx, fileio.irx, ioman.irx, loadcore.irx, loadfile.irx, moduleloader.irx, romdrv.irx, sifcmd.irx, sifman.irx, stdio.irx, sysclib.irx, timermani.irx, and threadman.irx.

These modules are grouped together in an image file (ioprpxx.img for the actual PlayStation 2 console, and t10000-relxx.bin for a development unit) which is provided together with the libraries.

Specific information about default module replacement processing appears in the “SIF System” document.

Checking the Versions of Default Modules

The following EE library functions have a feature for checking if the library version matches that of the default modules.

- sceOpen()
- sceSifLoadModule()
- sceSifLoadElf()
- sceSifLoadElfPart()

These functions compare their own version with the version of loadfile.irx, which is one of the replacement modules. If the versions do not match, -SCE_EVERSIONMISS is returned. When this error is returned, the version of the image file or flash ROM should be confirmed.

Unloading Modules

Resident modules that can be unloaded from memory when they are no longer needed are known as unloadable resident modules. To unload one of these modules, first perform stop processing, which frees resources such as memory and threads that were allocated immediately after the module was started. Once these resources have been properly freed, perform deletion processing, which removes the module from memory.

Kernel

Although there are differences in the details of the EE and IOP system software (kernel), they both provide an operating system that supports multithreading. The kernel handles system booting, program loading, multithread schedule management, inter-thread communication, exception handling, memory management, cache management, device management, and debugger support.

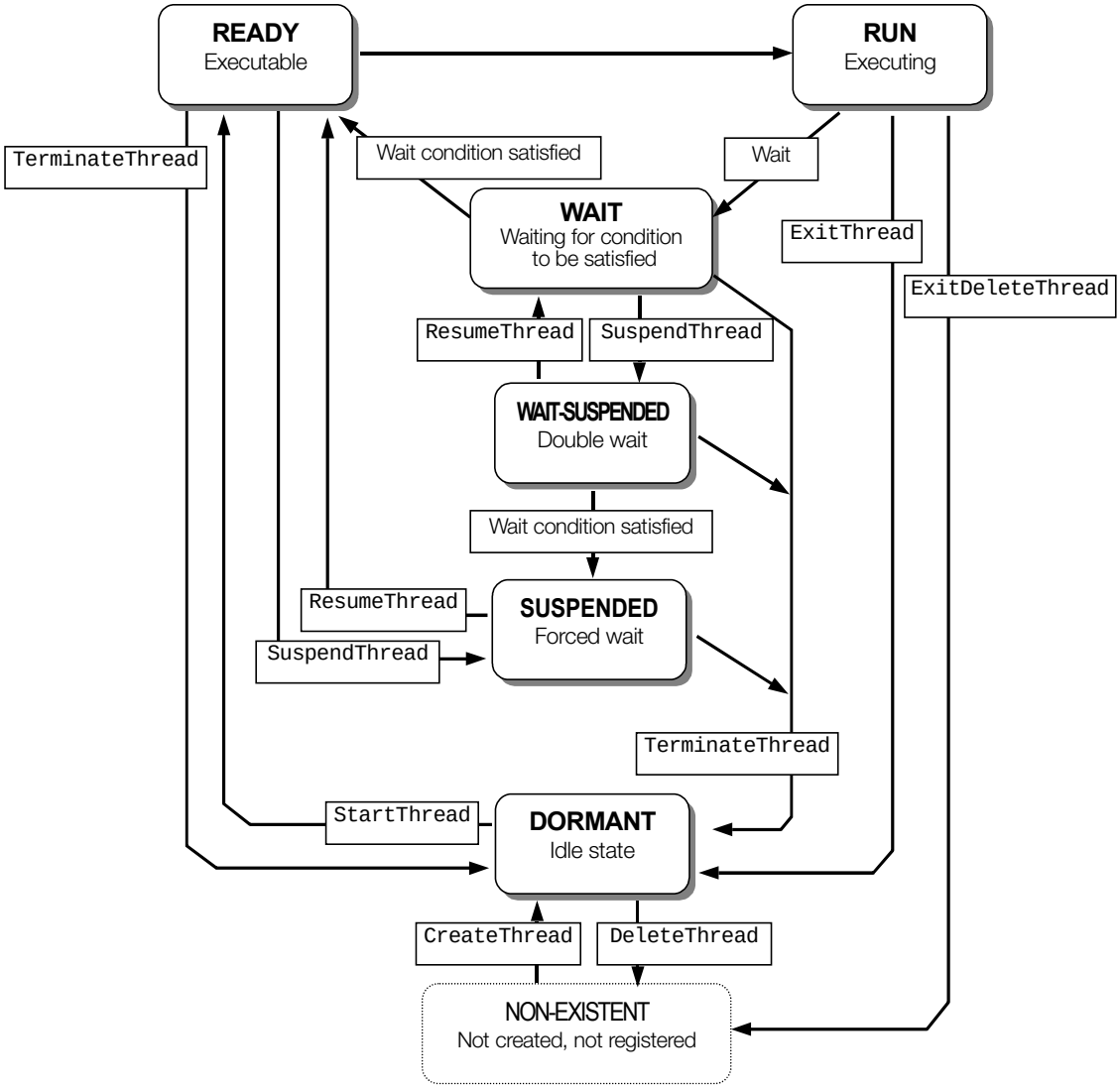
Threads

From the standpoint of concurrent processing, threads are the logical units of the program. The contents of applications programs to be processed are divided into multiple threads and can then be processed concurrently. Although the processing is termed ‘concurrent’, strictly speaking only one thread can be executed at any one time. and threads are switched and executed successively according to kernel management scheduling.

Multithread Management

Multithread scheduling is performed by the kernel using the thread state and the thread priorities that are assigned when threads are created. The kernel does not implement a time-sharing system, so execution priority cannot be implicitly taken away from a high-priority thread. If an external interrupt is received, the original thread will still continue executing upon return from the interrupt, unless the interrupt handler changes the execution priority of the thread or a higher priority thread becomes ready for execution. The six thread states are shown below:

Figure 3: State Transitions of Threads



A thread that has been created is initially in DORMANT state. When it is activated it enters READY state and is added to the READY queue. The READY queue is ordered by priority. Threads having the same priority are queued in the order they entered READY state.

The thread at the start of the ready queue is executed and is placed in RUN state. A thread in RUN state changes to WAIT state if it is waiting for the condition of a semaphore, etc. to be satisfied. When the wait condition is satisfied, the thread enters READY state again.

If a thread in READY state is forcibly suspended from outside the thread, it is placed in SUSPEND state. If a thread in WAIT state is forcibly suspended it is placed in WAIT-SUSPEND state.

Priority

When the EE Kernel is started, the priority of user programs is set to 1 (maximum). Each thread of a user program can have a priority of 1-127.

On the IOP, the priority of a user program must be in the range USER_HIGHEST_PRIORITY (9) to USER_LOWEST_PRIORITY (123). Also, the device drivers provided by SCEI use the priorities indicated below:

Table 2

Priority	Device Driver
8	loader
10	reboot
16	sio2man
18	(reserved)
20	padman, mtapman(high), dbcman
24	rspu2drv, sdrdrv
28	i.link(high)
30	usb(high)
32	sksound
34	i.link(low)
36	usb(low)
40	smap, an986
42	mcman, mcxman
46	padman, mtapman(low), sio2d, dgco, ds1u, ds2u, dslo, ds2o
48	inet, inetctl, netcnf
68	ppp, pppoe
81	cdvdfsv
88	loadmodule
96	fileio
104	mcserv, mcxserv

Inter-thread Communication

The EE kernel provides semaphores and event flags to handle inter-thread communication and synchronization. The IOP kernel also provides event flags and message boxes. Semaphores are used to restrict the number of threads that can simultaneously use a certain device or memory area. A maximum value is specified before a semaphore is created. Threads that want to use a device, etc. first retrieve a semaphore resource (for allowing use of a device) by calling the WaitSema() function. If the resource is unavailable, the thread is added to a semaphore wait queue. When another thread calls the SignalSema() function and releases the semaphore, the thread at the start of the wait queue is given the semaphore and returns to READY state.

The event flags allows a wider variety of synchronization operations. An initial flag value is set up beforehand when an event flag is created. By calling WaitEventFlag(), a thread that is waiting for a condition can specify a release condition and be added to a wait queue. When another thread changes the flag value via SetEventFlag(), the threads for which the release conditions have now been satisfied will return to READY state.

A message box is a service which supports the exchange of data between threads and can be thought of as the combination of a semaphore and ring buffer. First, create a message box using CreateMbx() (multiple boxes can be created). A thread which is waiting for a message specifies a message box number, issues ReceiveMbx() and enters WAIT state. When SendMbx() or iSendMbx() is called from another thread or a thread-independent section, a message box number is specified and a message will be sent. Then, the thread waiting on that message box will transition to READY state.

Kernel API

The API of the kernel is defined as a set of C language functions. Note that separate APIs are available for calls that are made from within a thread and for calls that are made from a thread-independent section, e.g., an interrupt handler. These APIs perform roughly identical functions but use different internal operations. Functions called from interrupt handlers, etc. are prefixed with an “I”, as in TerminateTh and iTerminateTh.

Features of the Libraries

In order to support creation of PlayStation 2 software, libraries for controlling blocks of hardware features and libraries to support software development are provided.

EE Library - Kernel API

Kernel API

The kernel API provides various functions such as priority-based multithread scheduling, inter-thread communication using semaphores, exception handler management and COP0 control.

I/O Service

The I/O service is implemented as a kernel API extension and provides CD(DVD)-ROM and hard disk support, as well as file I/O functions on the development computer. In addition to the generic open/close, read/write, seek and ioctl APIs, a simple printf function and a function to reset SIF RPC bind information are also provided.

Standard IOP Service

The standard IOP service is implemented as a kernel API extension and consists of function groups that load IOP modules into IOP memory, load elf files into EE memory and manage heap space in IOP memory.

EE Library - Peripheral Device Libraries

System Configuration Library - libscf

This library obtains system configuration information set by the user in the advanced settings screen of the PlayStation 2. Examples of the information include the display language and the aspect ratio of the television screen. The library has functions which hide the differences between the overseas versions and the Japanese versions of the PlayStation 2 and DTL-T10000.

CD(DVD)-ROM Library - libcdvd

This library controls the CD(DVD)-ROM drive and performs data reading. It is provided with a ring buffer in IOP memory so it can continuously read data in order to support streaming. A function to read the current date and time (from the internal realtime clock) is also provided in this library.

Memory Card Library - libmc

This library detects the presence of a memory card and performs formatting, file management and data read/write for the memory card.

PDA Library - libmcx

This library controls the PocketStation (PDA), acquires / updates PDA information, acquires / updates user interface information and performs memory access.

Controller Library - libpad

The libpad library gets button data from the controllers, changes the controller mode and operates (vibrates) actuators. The library supports various types of commercial controllers including the Dualshock 2.

Controller Library 2 - libpad2

This library has a partitioned structure enabling developers to limit memory usage. Both EE and IOP memory can be saved if the types of controllers and their usage are restricted, because separate driver modules are provided for each type of controller. For instance, the vibration control function has been isolated in a separate library.

Vibration Library - libvib

This library controls the vibration function of the controller and is used together with libpad2.

Multitap library - libmtap

This library controls the PS2 multitap. When the controller library (libpad) and the memory card library are used together, the controller connected to the multitap and the PS2 memory card can be controlled.

USB Keyboard Library - libusbkb

This library controls USB keyboards. It presumes the use of character input and has functions that can perform keycode substitution and repetition.

Hard Disk Library

This library is used for hard disk drives. It contains two modules: one for drive control and partition management and the other for pfs (Playstation File System) operations. Both of these use the I/O service API.

EE Library - Graphics Libraries

GS Basic Library - libgraph

This library generates and shapes environmental information required for display and drawing as well as provides initial settings for the GS. Because this library can hide differences between GS versions, it must be used to perform initial settings.

High-Level Graphics Library - libhig

libhig is a graphics library with a simple plugin format enabling functions to be easily added or combined. Processing is handled through a master data format and each plugin is called using a data structure known as a plugin block. This allows complete processing to be performed by the application such as reading authored data, kicking plugin blocks and DMA-transferring generated display lists to the GIF.

Low-level routines are also provided for each plugin such as memory management service functions, DMA service functions and GS management service functions to support the use of independently developed plugins.

Graphics Plugin Library - libhip

libhip is a plugin group for use with the high-level graphics library. Various plugins are provided that perform microprogram transfers to VU1, texture data transfers to the GS, drawing of shape data, matrix generation of hierarchy structures and coordinate conversions, linear interpolation between keyframes, shape generation using shared vertices and shared normals, bump maps, environment maps and shadowing.

EE Library - Animation Libraries

MPEG Library - libmpeg

This library performs decoding of MPEG2 and MPEG1 images. It provides control of data transfers and MC compensation processing. Data transfers are controlled using callbacks enabling the application to select a suitable buffering method.

IPU Library - libipu

libipu is a low-level library for controlling the IPU. APIs are provided to execute IPU commands and for controlling the operation of the entire IPU.

EE Library - Sound Libraries

Low-level Sound Library - libsdr

This library remotely controls the SPU2. It operates in conjunction with driver modules present on the IOP and provides EE applications with functions identical to that of the low-level sound library (libsd) for the IOP. In order to improve processing efficiency, the library is provided with a batch execution mechanism that collects and issues commands for the SPU2.

CSL Lineout - liblout

liblout is a CSL module that transfers PCM streams to the SPU2 via libsdr for playback.

CSL MIDI Stream Generation - libmsin

libmsin is a CSL module that generates MIDI Streams in realtime by calling APIs from an application program. It can also generate extended MIDI messages for controlling sound effects.

CSL Software Synthesizer - libssyn

libssyn is a software sound source module that inputs MIDI Streams from modssyn on the IOP, then uses phenome data in EE memory to output PCM Stream data. It also has APIs for generating sound that can be called from an EE application.

CSL Sound Effect Stream Generation - libsein

libsein is a CSL module that generates sound effect streams in realtime by calling APIs from an application program.

SPU2 Local Memory Management Library - libspu2m

This library performs memory management by keeping the status of SPU2 local memory in a management table in EE memory. The size of the management table can be specified when initializing the library according to how many total blocks the local memory will be divided into for management.

Reserved areas (such as sound data I/O areas) will not be managed by libspu2m. Work areas for effect processing can be reserved as necessary.

Standard Kit Library / Sound System - SKSS

This library is able to perform sound processing on the EE only, without requiring any programming on the IOP. It supports playback of MIDI sequences, sound effect sequences and one-shot sound generation.

EE Library - Network Libraries

Network Socket Library - libinsck

This library calls the network library (INET) on the IOP and provides a socket API to an EE application.

Common Network Setting GUI Library - ntgui

This library supports the creation of configuration functions for network hardware and network service providers. It has functions to read and write common network configuration files as well as user interface functions and can be used to easily create the Network Configuration Screen.

EE Library - Development Support

Debugger Support Library for Development - libdev

This library has functions for controlling VU0, VU1, VIF0, VIF1 and GIF and for reading internal registers. It also includes functions to output messages from the PlayStation 2 monitor screen as a pseudo-console.

DMA Basic Library

This library has functions for controlling DMA transfers as well as functions to generate various tags for use in Chain mode, in order to support the creation of transfer lists.

Packet Library

This library supports the creation of DMA packets for Source Chain mode, VIF UNPACK packets, VIF DIRECT packets and GIF packets (GS primitives). It has a function to semi-automatically generate a header for each packet and perform suitable alignment.

Performance Counter Library

This library uses a performance counter internal to the EE core as a resource for program tuning, and to count and measure internal events of the CPU / pipeline during program execution.

VU0 Library

This is a library of vector operations and matrix operations that use VU0 macro instructions. It is provided along with source code.

IOP Library - Kernel API

IOP Kernel

The IOP kernel API provides various functions such as priority-based multithread scheduling, synchronous interthread communication between threads using semaphores, event flags and message boxes, interrupt handler management, memory pool management, timer management and V-blank synchronization.

IOP Library - Peripheral Device Libraries

CD(DVD)-ROM Library - libcdvd

This library controls the CD(DVD)-ROM drive and performs data reading. It is provided with a ring buffer in IOP memory so it can continuously read data in order to support streaming. A function to read the current date and time (from the internal realtime clock) is also provided in this library. This library uses almost the same functions as those of the corresponding EE library.

Hard Disk Library

This library is used to support the hard disk drive. It contains two modules: one for drive control and partition management and the other for pfs (Playstation File System) operations. Both of these use the I/O service API. This library uses almost the same functions as those of the corresponding EE library.

IOP Library - Sound Libraries

Low-level Sound Library - libsd

This library directly controls the SPU2. It has functions for performing SPU2 register access, transfer of waveform data / sound data in SPU2 local memory, setting up of processes to handle interrupts from the SPU2, and functions for getting and setting effect attributes. It also has functions to perform a series of SPU2 register accesses as a batch.

Waveform Encode Library - spucodec

This library encodes 16-bit straight PCM data into waveform data for the SPU2.

CSL MIDI Stream Generation - modmsin

modmsin is a CSL module that generates MIDI Streams in realtime by calling APIs from an application program. It can also generate extended MIDI messages for controlling sound effects.

CSL MIDI Sequencer - modmidi

modmidi is a CSL module that inputs SQ data to generate MIDI Streams. It can play back multiple SQ data simultaneously.

CSL Hardware Synthesizer - modhsyn

modhsyn is a CSL module that inputs MIDI Stream data for sound generation using the SPU2. It also supports extended MIDI messages for sound effects and sound effect messages. This module also has a function for obtaining SPU2 access history for use in debugging.

CSL Hardware Synthesizer - modssyn

modssyn is a CSL module that inputs MIDI Stream data for sound generation using libssyn on the EE.

CSL MIDI Delay Module - moddelay

moddelay is a CSL module that performs delay processing of MIDI Streams. It can be used to compensate for different latencies.

CSL MIDI Monophonic - modmono

modmono is a CSL module that assigns MIDI Streams to monophonic.

CSL Sound Effect Stream Generation - modsein

modsein is a CSL module that enables an application to generate sound effect streams in realtime using API calls.

CSL sound effect sequencer - modsesq

modsesq is a CSL module that inputs SQ data for generating sound effect streams.

IOP Library - i.Link Libraries

i.Link Driver Library

This driver library monitors and controls the 1394 bus. It also supports asynchronous transfers.

Socket Library

This library supports socket communication as an upper-level layer of the i.Link driver library.

IOP Library - USB Libraries

USB Driver Library

This driver library monitors and controls the USB bus. It has three functions: HCD (Host Controller Driver), USBD (USB Driver) and Hub Driver, and is responsible for device-independent processing for target devices. In order to use a specific USB device, an LDD must be created that is responsible for processing for that device.

USB Module Autoloader

This driver loads and unloads the appropriate LDD whenever the associated USB device is inserted or removed. When a driver database is created that describes the correspondence between the USB device and corresponding LDD file, together with an LDD that supports autoload, a suitable LDD will be determined and automatically loaded when the USB device is connected. An LDD that supports autoloading is necessary however. During streaming, the autoload can be postponed if it cannot be done conveniently.

IOP Library - Network Libraries

Network Library - INET

This library (INET protocol stack) provides a data send / receive service using TCP / UDP protocols.

Network (UNET) Setting Library - INETCTL

This library handles settings related to the operation of the INET protocol stack and managment of the operating state.

Common Network Configuration Library - NETCNF

This library supports reading and writing of common network configuration files. A common network configuration file is a file that allows each application to share the use of different types of configuration information related to telephone lines and LANs in order to connect to the network.

Network Device Library - NETDEV

This library is used when developing device drivers for devices connected to the network.

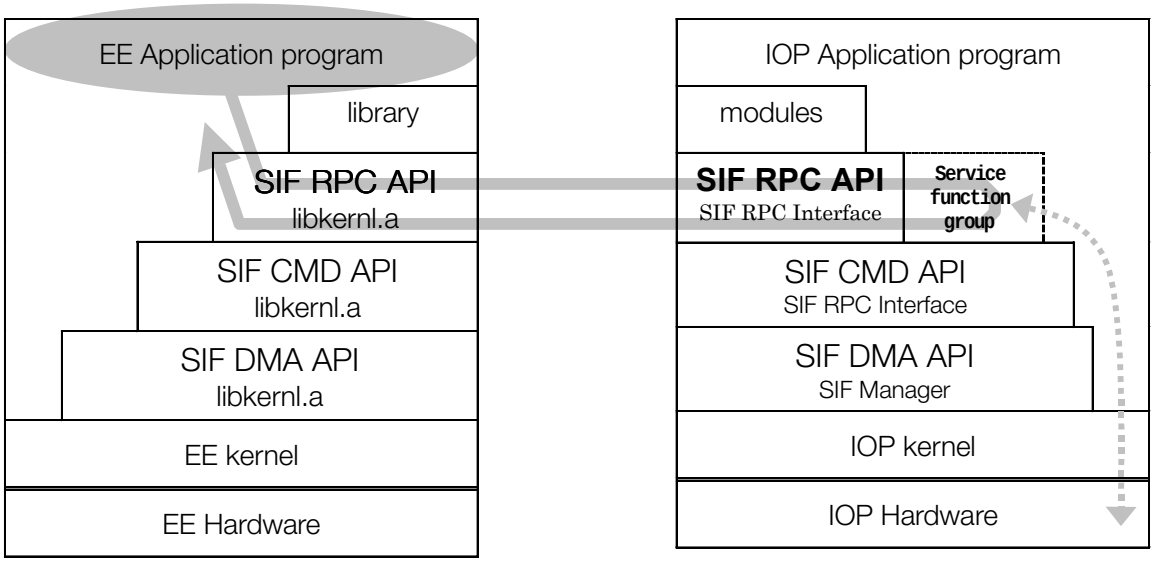
Modem Development Library

This library is used for developing modem drivers.

Library Structure for EE-IOP Communication (SIF RPC)

In order to support cooperative processing between the EE and IOP, a remote procedure call library (SIF RPC) is provided that creates a simple client/server system between the EE and IOP. This library is provided together with low-level libraries. Functions including hard disk drive control, file I/O and IOP control are also provided as standard services. SIF RPC is almost always used internally in libraries implemented on the EE and libraries related to peripheral devices such as libpad / libmc / libcdvd.

Figure 4:EE-IOP Client/Server Hierarchy Diagram



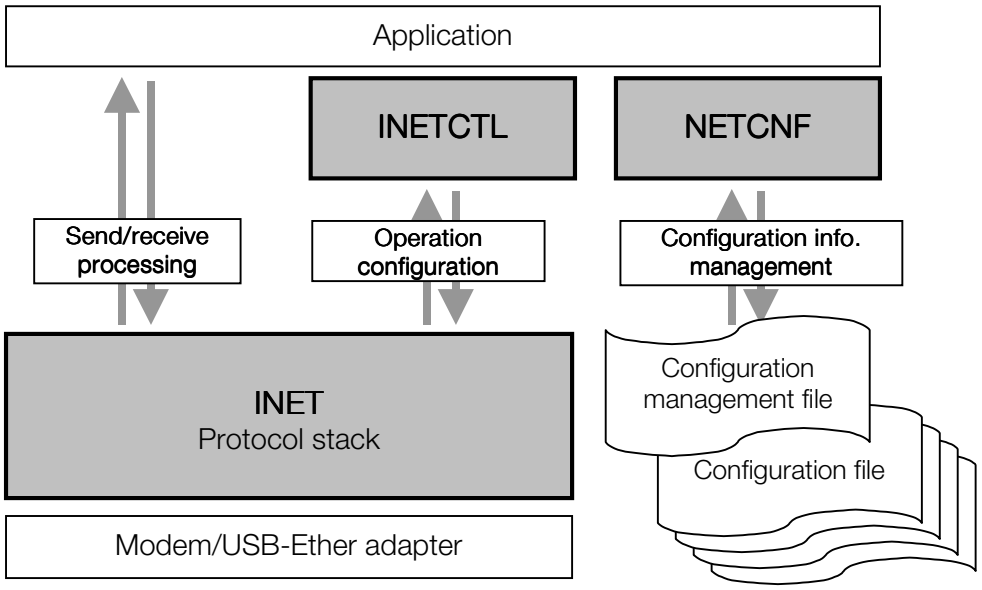
While using SIF RPC, semaphores must be used for arbitration to prevent calling service functions in an interrupt-inhibited state and to prevent calling the same group of service functions simultaneously from multiple threads. The same precautions must be taken for libraries using SIF RPC internally.

If there is a large amount of overhead (e.g. an application that uses the network), a multithreaded version (MSIF RPC) that has an arbitration function can be used instead.

Structure of the Network Libraries

Three IOP libraries, INET / INETCTL / NETCNF are provided in order to connect the PlayStation 2 to a network (Internet).

Figure 5: Organization of network support library



This INET protocol stacks are TCP and UDP, with IP / ARP / PPP as their lower level protocols. The library also supports hardware for the following three network connection devices.

- Dialup connection using USB Modem
- LAN connection using network adaptor
- LAN connection using USB-Ethernet adaptor (for development use only)

Each piece of information that must be set by the user in order to connect to the network such as IP addresses and subnet masks, as well as information such as the telephone number of the provider and user account, is designed to be stored and managed in common network configuration files. These files pass through the NETCNF library and allow applications to have common access to the network.

When using a common network configuration file, the read / write processing of the file along with the user interface must be implemented according to prescribed guidelines. The EE library that supports this is the network configuration GUI library (ntgui).

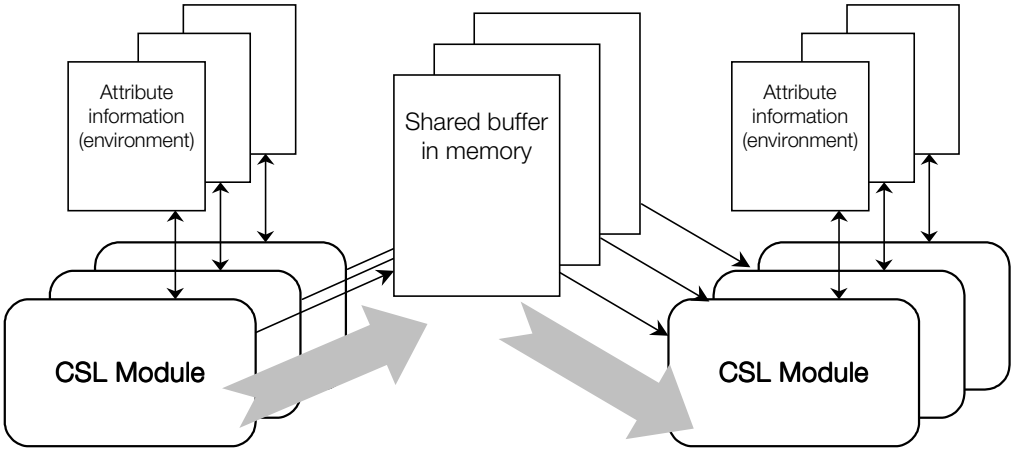
In addition, a socket communication library is provided so that EE applications can make use of the network.

Structure of the Component Sound Library (CSL)

CSL is a standardized architecture that is used to "componentize" sound generation processes. High-level sound processing can be easily implemented by optimally combining CSL modules to create an independent CSL module as necessary.

Cooperation between CSL modules is implemented by sharing the output buffer of a certain module as the input buffer of another module. Each CSL module is called within a fixed time interval (called a "tick"). If there is data in the input buffer, an amount equivalent to 1 tick is written to the output buffer, which then becomes the input data of the next CSL module. Data streams are delivered through the shared buffers in this manner.

Figure 6: Cooperation of CSL Modules Using Buffer Sharing



Although CSL modules function on both the IOP and the EE, they are divided into categories depending on the combinations of the input data formats. The following CSL modules are provided by SCE.

Table 3

Module	Category	Function
Hardware synthesizer (modhsyn)	IOP ; MIDI → Original	Sound source using SPU2 voice
Software synthesizer (modssyn)	IOP ; MIDI → PCM	Multi-function WaveTable sound source IOP-side module
MIDI sequencer (modmidi)	IOP ; Original → MIDI	Conversion from SQ file to MIDI stream
Sound effects sequencer (modsesq)	IOP ; Original → MIDI	Dedicated sequencer for sound effects
Sound effects stream generation (modsein)	IOP ; Original → SE	Generates sound effects streams
MIDI stream creation (modmsin)	IOP ; Original → MIDI	MIDI stream creation
MIDI monophonic (modmono)	IOP ; MIDI → MIDI	Conversion to monophonic
MIDI delay (moddelay)	IOP ; MIDI → MIDI	Latency adjustments

Module	Category	Function
MIDI Stream creation (libmsin)	EE ; Original → MIDI	MIDI stream creation
Software synthesizer (libssyn)	EE ; MIDI → PCM	Multifunction WaveTable sound source EE-side module
Line out (liblout)	EE ; PCM → Original	PCM streaming playback using SPU2
Sound effects stream generation (libsein)	EE ; Original → SE	Generates sound effects streams

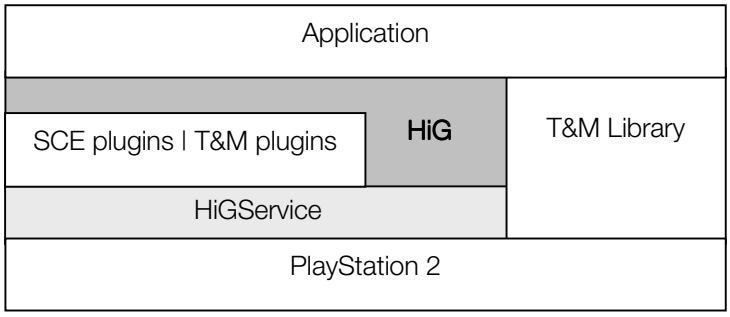
As described above, independent sound processes can be implemented by creating independent CSL modules and combining them with the CSL modules provided by SCE. The CSL standard is based loosely on the buffer structure, primarily. Processing within a module is of course established only by the recommended specification in the API. Basically, almost the entire implementation is left to the creator.

Structure of the High-level Graphics Library (HiG)

A high-level graphics library (HiG) is provided that manages graphics functions using a plugin format. The plugin format can be used to implement various graphics techniques with simple procedures. The HiG is provided together with data formats, converters, basic plugins and low-level libraries which support independent plugin creation.

The HiG itself has management functions such as plugin registration, data block interpretation and calling plugins. Each plugin also performs graphic processing. The different parameters for the plugins are kept within data blocks and user programs handle hardware resource operations such as allocating memory. A structure with a high level of independence like this makes it easier to reuse graphics processing routines and to provide them as middleware.

Figure 7: Software Hierarchy of the High-level Graphics Library



A graphics framework function and an equivalent graphics library HiP are provided as basic plugins. In addition, a tool called "esConv" is provided together with the eS package as a data converter. This tool converts from eS format (described later) to HiG format.

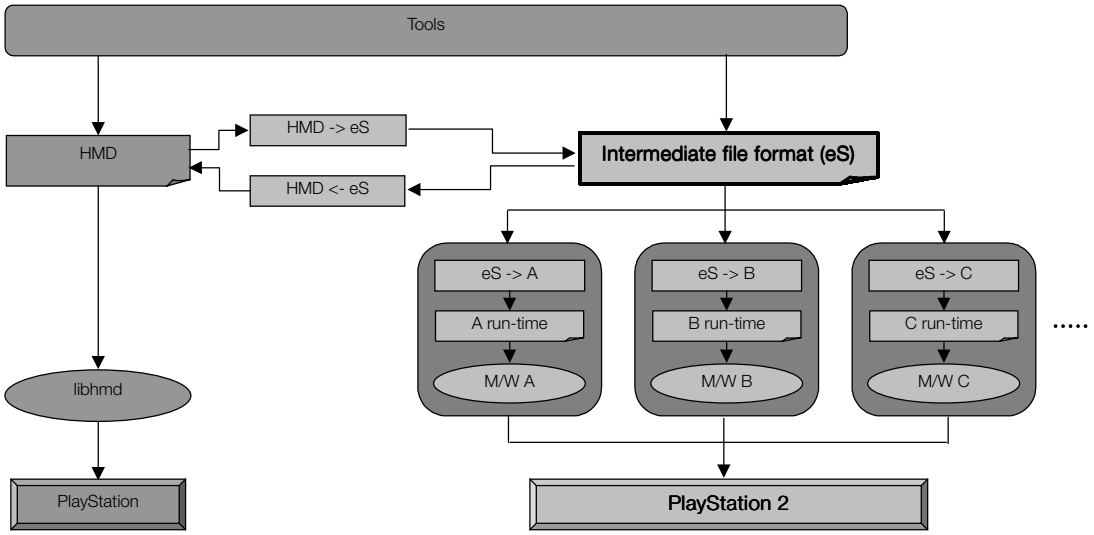
Middleware

Middleware is software mainly provided by third parties that allows the capabilities of the PlayStation 2 to be used to create high-quality expressions. Middleware is positioned higher than libraries.

Providing high-quality expressions suitable for computer entertainment applications involves the use of 3D graphics, animation, and sound, and requires a high degree of specialized skill and knowledge in the respective field. The creation of high-quality expressions suitable for next-generation computer entertainment applications is made possible by having skilled third parties provide middleware in their fields of expertise.

An intermediate file format "eS" must be specified for each piece of middleware. The eS format can uniformly handle geometry, texture, animation and backgrounds for regions supported by the conventional HMD format. The eS format can also be easily extended to handle individual information in each application / middleware. It can integrate data that uses multiple pieces of middleware into one file and extract each portion of the data to process each piece of middleware.

Figure 8: eS and Middleware



Using a Controller

libpad and libpad2

libpad and libpad2 + libvib are provided as libraries for managing controllers. Among these, libpad is a library that obtains the connection state and actuator information, reads button data, controls the actuator and provides overall support for the controller.

libpad2 is a library that mainly divides every function of libpad for developers who wish to reduce the amount of memory used. In addition to isolating the control functions of the actuator as libvib, libpad2 is designed to control each individual controller when the driver modules on the IOP are also separate for each type of controller.

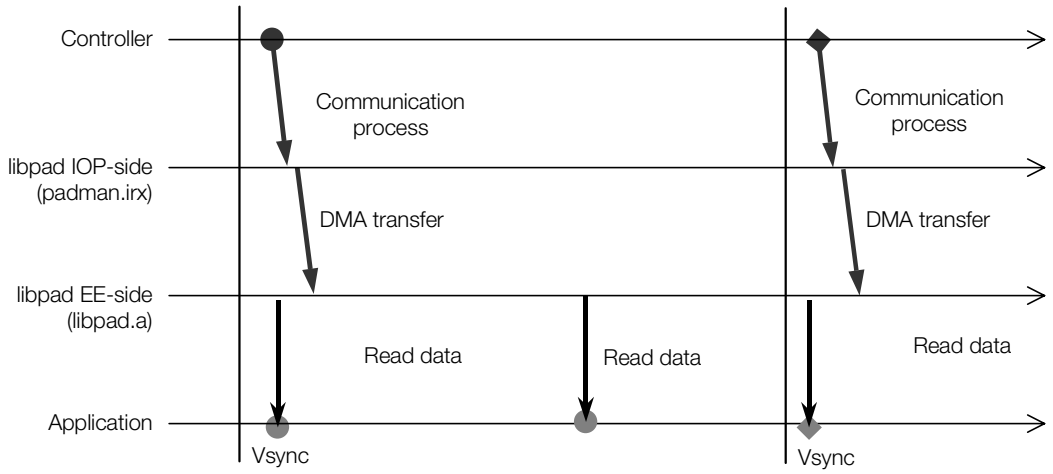
The amount of memory used can be reduced since this module need be loaded only when necessary, as determined by the application's needs, what controllers are supported, and number of supported connections (for example, when no vibration processing is performed, or when the pressure sensitivity function of a button is not used, or when a different controller is used that is not DUALSHOCK2 compatible, or in the case of a one-player game, etc.).

Process Timing

The controller library consists of an EE program and an IOP program. The mechanism for communicating with controllers is such that the IOP program is synchronized with VSync each frame, the data that was read is DMA transferred to EE memory, where it can be read by the EE program.

Although applications running on the EE can read controller data at arbitrary times, the data may reflect the state 1 Vsync earlier.

Figure 9: Controller Data Read Timing



Using a Multitap

By using a multitap, up to eight controllers and PS2 memory cards can be connected to the PlayStation 2.

To use a multitap, libpad/libmc services must be accessed via libmtap (libpad2 does not currently support the multitap). The controllers and PS2 memory cards connected to the multitap can be accessed by first loading the libmtap module in the IOP and opening the port to which the multitap is connected. Then the various libpad/libmc functions can be called.

Using the CD/DVD-ROM

The PlayStation 2's internal CD/DVD-ROM drive has a maximum speed of 24x (CD-ROM) / 4x (DVD-ROM) and uses the CAV method of uniform rotational velocity, however, it does not use a fixed data transfer rate. Two modes can be set for controlling the rotational velocity. One is a high-speed mode in which the speed will be reduced and a retry attempted if an error occurs during reading. The other mode is for streaming. This mode reduces the speed initially to try and prevent errors from occurring.

The libcdvd library is provided for accessing the CD/DVD-ROM. The file system conforms to ISO-9660 level1. UDF is not supported for DVD. File and directory names have the following restrictions.

Table 4

Item	Restriction
Number of directories	40 directories/system
Depth of directory tree	8 levels maximum
Number of files	30 files/directory
Valid characters	0 - 9, A - Z, _
Format of filenames	(8-character filename).(3-character extension);1 * long filenames not permitted
Format of directory names	(8-character directory name) * extensions not permitted

Files are accessed by first obtaining the sector number from the filename, then using the sector number to read the data. Data read operations are non- blocking and can be performed in parallel with other operations. By using the separate CD/DVD-ROM Generator, mapping tables for files and sector numbers can be generated. This eliminates the step of obtaining the sector number from the filename during execution.

CD-XA and CD-DA (music CDs) are not supported. CD-DA data requires a pitch conversion from 44.1 KHz to 48 KHz, which puts a heavy load on the system.

Using a Memory Card

The PlayStation 2 supports PS2 memory cards with an expanded capacity of 8 MB. The 128 KB memory cards can be used only for copying PS data files using the browser.

Memory Card Specifications

The PlayStation 2 memory card specifications are listed below.

Table 5

Item	Specifications	Specifications of 128 KB memory card
Capacity	64Mbit (8MB)	1Mbit (128KB)
Management units	Cluster (1024 bytes)	Slot (8192 bytes)
Access units	Page (512 bytes)	Sector (128bytes)
Delete units	Block (8192 bytes)	
Write time	60ms/block	2200ms/block
Read time	25ms/block	2200ms/block

With the larger capacity, there are changes in the file system such as support for hierarchical directories as well as changes to icons and file information.

Memory Card Library

The libmc library is provided to support access to the PS2 memory card. libmc provides support for formatting of PS2 memory cards, obtaining card information, obtaining file lists, manipulating hierarchical directories, reading/writing files, changing file attributes, etc. The libmc operations are non-blocking, thus allowing other operations to be carried out in parallel, while polling to check for completion of an operation.

Related Tools

A viewer is provided that displays a preview of icon data for the memory card.

Using a PocketStation

The PocketStation (PDA) is a small computer that uses an ARM7TDMI for the processor core and has 128 KB of flash memory, 2 KB of RAM, a 32 X 32 dot monochrome LCD and other I/O devices. The PocketStation fits into the memory card slot of the PlayStation 2. Program files (PDA applications) are saved in the same manner as with a memory card and execute out of ARM7 virtual address space.

The EE library "libmcx" is provided as a library to support access to the PocketStation. Used together with libmc functions, libmcx can perform file access using standard I/O, partition management and file system management. In addition, an API is provided to read and write information related to serial numbers and executing PDA applications, read and write user interface information like alarm settings and speaker volume, read and write internal flash memory, and control the display during file transfers.

As a restriction on PocketStation use, access to the PocketStation is limited to PlayStation 2 title applications containing PDA applications. File formats which can be created are also limited to the PDA application format.

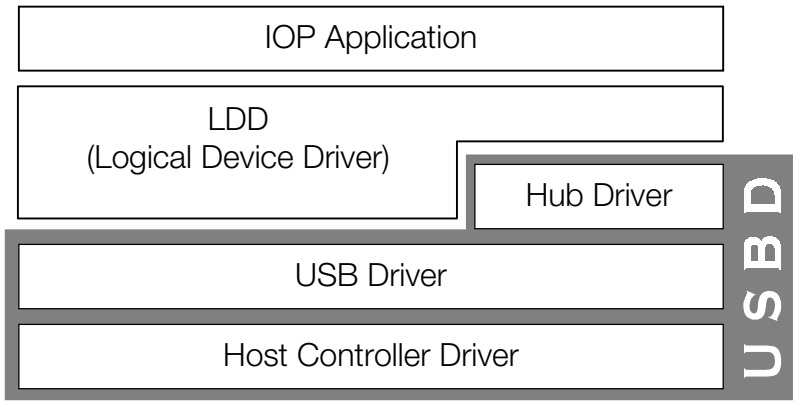
Using USB

The USBD that runs on the IOP is provided as a low-level library of the USB specification. The USBD library can be used for connecting many different types of peripheral devices such as a mouse, keyboard, microphone, modem, printer and others.

USBD and LDD

USBD is a low-level library that includes host controller control functions (OpenHCI) and USB hub control functions. In order to use specific USB devices such as a mouse or modem, drivers (LDD: Logical Device Driver) which correspond to each device (or device class) must be created for each device.

Figure 10: Hierarchical Structure of USBD and LDD



The five functions shown below must be implemented in an LDD.

Table 6

Function	Function/Operation
start function	Initializes LDD, registers to USBD
probe function	When a USB device is detected, USBD calls the probe function of each registered LDD in order. The probe function determines if it can handle the connected device and returns the information to the USBD.
attach function	USBD calls the attach function of the appropriate LDD according to the return value from the probe function for each LDD. The attach function handles preparations for communication with the USB device and issues data transfer requests for the USBD.
callback function	The callback function is called when data transfer completes. It handles the relevant data processing and re-issues data transfer requests when necessary.
detach function	The detach function is called when USBD detects that a USB device was disconnected.

When creating an LDD, you must understand the USB standard, the definitions of the applicable device class and the specifications of the specific USB device. Information on the first two can be obtained at the USB Implementers Forum Inc. web site (http://www.usb.org/).

USB Module Autoloader

The USB module autoloader is provided as a function to load an LDD when the USB device it is associated with is connected.

A configuration file is prepared in advance that contains LDD filenames and their corresponding USB devices. When a USB device is connected, the relevant LDD for that device is loaded, according to the configuration file. The autoloader can also be stopped momentarily, such as during streaming, when it might be inconvenient.

Using i.Link

An i.Link driver library and socket library are provided for the IOP to support communication via i.Link (IEEE 1394 bus). In addition to providing data conversion between i.Link compatible devices, the i.Link enables competitive processing between two connected PlayStation 2s as well as data conversion between PCs.

The i.Link driver conforms to the IEEE 1394.a 2.0 specification implementing node capabilities at the transaction level. Basically, this means the driver can handle async communication and Config ROM reading / writing. The socket library supports packet communication with a specified destination.

Regardless of which library is installed on the IOP-side, be sure to independently provide an RPC service function to use the library from an EE application. (Since a large load is placed on the IOP, there are no plans to provide a general purpose library.)

Figure 11: Hierarchical Structure of i.Link Related Libraries