PlayStation®2 EE Library Overview
Release 2.4

Other Libraries

Summary Table of Contents

About This Manual

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

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

Changes Since Last Release

None

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:
Development Debugging Support Library

Library Overview

libdev is a library that provides support for debugging during development.

libdev allows execution of the main devices (VU0/VU1/VIF0/VIF1/GIF) to be started and stopped, and allows the status of these devices to be obtained. Debugging messages can be output using the monitor of the PlayStation 2 as a virtual console.

Related Files

Files related to libdev are as follows:

Table 1-1

Category	Filename
Library file	libdev.a
Header files	devgif.h
	devvif0.h
	devvif1.h
	devvu0.h
	devvu1.h
	devfont.h

List of Functions

Except for functions relating to virtual consoles, libdev functions are provided so that functions having the same features are available for each device. The following list shows the functions arranged by feature and target device.

VIF0 / VIF1 / GIF Control

Table 1-2

Description	VIF0	VIF1	GIF
Reset	Vif0Reset	Vif1Reset	GifReset
Pause	Vif0Pause	Vif1Pause	GifPause
Continue	Vif0Continue	Vif1Continue	GifContinue
Read all registers	Vif0GetCnd	Vif1GetCnd	GifGetCnd
Write data to FIFO	Vif0PutFifo	Vif1PutFifo	GifPutFifo
Read data from FIFO	---	Vif1GetFifo	---
Set error mask	Vif0PutErr	Vif1PutErr	---
Get error mask	Vif0GetErr	Vif1GetErr	---
Set PATH3 transfer mode	---	---	GifPutImtMode
Get PATH3 transfer mode	---	---	GifGetImtMode
Set PATH3 mask	---	---	GifPutP3msk
Get PATH3 mask state	---	---	GifGetP3msk

Note: The “sceDev” at the start of each function name has been omitted to reduce complexity.

VU0 / VU1 Control

Table 1-3

Description	VU0	VU1
Reset	Vu0Reset	Vu1Reset
Force Break	Vu0Pause	Vu1Pause
Continue	Vu0Continue	Vu1Continue
Set up debugging	Vu0PutDBit	Vu1PutDBit
Set up debugging	Vu0PutTBit	Vu1PutTBit
Get debugging status	Vu0GetDBit	Vu1GetDBit
Get debugging status	Vu0GetTBit	Vu1GetTBit
Re-execute microprogram	Vu0Exec	Vu1Exec
Read TPC register	Vu0GetTpc	Vu1GetTpc
Read all registers	Vu0GetCnd	Vu1GetCnd
Write all registers	Vu0PutCnd	Vu1PutCnd

Note: The “sceDev” at the start of each function name has been omitted to reduce complexity.

Reference Material

For VU0 / VU1 controls, see Chapters 3 and 5 of the "VU User's Manual". For VIF0 / VIF1, see Chapter 4 of the "EE User's Manual". For GIF, see Chapter 5 of the "EE User's Manual".

Chapter 2: Basic DMA Library

Library Overview

libdma is a library that provides support for DMA transfer operations. In addition to performing DMA transfers by controlling the DMAC, the library can generate Chain mode tags to support the creation of transfer lists.

Related Files

The following files are required for libdma:

Table 2-1

Category	Filenames
Library file	libdma.a
Header file	libdma.h

List of Functions

Among the libdma functions, the functions that generate tags and add to a transfer list can be categorized as follows based on the type of DMA tags generated and the interrupt bits.

Table 2-2

Non-interrupt functions	Interrupt functions	Description (generated tag)
SceDmaAddRef	SceDmaAddIRef	Add REF tag (for Source Chain)
SceDmaAddRefe	SceDmaAddIRefe	Add REFE tag (for Source Chain)
SceDmaAddRefs	SceDmaAddIRefs	Add REFS tag (for Source Chain)
SceDmaAddCont	SceDmaAddICont	Add CNT tag (for Source Chain)
SceDmaAddNext	SceDmaAddINext	Add NEXT tag (for Source Chain)
SceDmaAddCall	SceDmaAddICall	Add CALL tag (for Source Chain)
SceDmaAddRet	SceDmaAddIRet	Add RET tag (for Source Chain)
SceDmaAddEnd	SceDmaAddIEnd	Add END tag (for Source Chain)
SceDmaAddDest	SceDmaAddIDest	Add CNT tag (for Destination Chain)
SceDmaAddDests	SceDmaAddIDests	Add CNTS tag (for Destination Chain)

DMA termination interrupt handlers can be registered using the AddDmacHandler() kernel API.

Reference Material

For more information on DMA transfers, see Chapter 3 in the "EE User's Manual".

Chapter 3: Performance Counter Library

Library Overview

The libpc library uses the performance counter within the EE core to provide a means to collect statistical information on internal events of the pipeline or CPU during program execution. For example, it can be used to get data in order to perform program tuning, such as checking the status when a cache miss has been generated.

In the EE core, there are counter 0, counter 1 and two independent counters. The value of each counter is increased by one each time their corresponding counter events occur. Each counter can count up to a maximum of 0x7fffffff events. If a counter overflows beyond this value, a Performance Counter exception is generated.

Details on events which can be counted with the performance counter are described below.

Related Files

libpc requires the following files:

Table 3-1

Category	File name
Library file	libpc.a
Header file	libpc.h

Countable Events

Events and event codes which can be counted by Counter 0 are as follows:

Table 3-2

Event Code	Value	Event
SCE_PC0_RESERVED	(0 << 5)	(reserved)
SCE_PC0_CPU_CYCLE	(1 << 5)	Processor cycle
SCE_PC0_SINGLE_ISSUE	(2 << 5)	Single instructions issue
SCE_PC0_BRANCH_ISSUED	(3 << 5)	Branch issued
SCE_PC0_BTAC_MISS	(4 << 5)	BTAC miss
SCE_PC0_ITLB_MISS	(5 << 5)	ITLB miss
SCE_PC0_ICACHE_MISS	(6 << 5)	Instruction cache miss
SCE_PC0_DTLB_ACCESSED	(7 << 5)	DTLB accessed
SCE_PC0_NONBLOCK_LOAD	(8 << 5)	Non-blocking load
SCE_PC0_WBB_SINGLE_REQ	(9 << 5)	WBB single request
SCE_PC0_WBB_BURST_REQ	(10 << 5)	WBB burst request
SCE_PC0_ADDR_BUS_BUSY	(11 << 5)	CPU address bus busy
SCE_PC0_INST_COMP	(12 << 5)	Instruction completed
SCE_PC0_NON_BDS_COMP	(13 << 5)	Non-BDS instruction completed
SCE_PC0_COP2_COMP	(14 << 5)	COP2 instruction completed
SCE_PC0_LOAD_COMP	(15 << 5)	Load completed
SCE_PC0_NO_EVENT	(16 << 5)	No event

Events and event codes which can be counted by Counter 1 are as follows:

Table 3-3

Event code	Value	Event
SCE_PC1_LOW_BRANCH_ISSUED	(0 << 15)	Low-order branch issued
SCE_PC1_CPU_CYCLE	(1 << 15)	Processor cycle
SCE_PC1_DUAL_ISSUE	(2 << 15)	Dual instructions issue
SCE_PC1_BRANCH_MISS_PREDICT	(3 << 15)	Branch miss-predicted
SCE_PC1_TLB_MISS	(4 << 15)	TLB miss
SCE_PC1_DTLB_MISS	(5 << 15)	DTLB miss
SCE_PC1_DCACHE_MISS	(6 << 15)	Data cache miss
SCE_PC1_WBB_SINGLE_UNAVAIL	(7 << 15)	WBB single request unavailable
SCE_PC1_WBB_BURST_UNAVAIL	(8 << 15)	WBB burst request unavailable
SCE_PC1_WBB_BURST_ALMOST	(9 << 15)	WBB burst request almost full
SCE_PC1_WBB_BURST_FULL	(10 << 15)	WBB burst request full
SCE_PC1_DATA_BUS_BUSY	(11 << 15)	CPU data bus busy
SCE_PC1_INST_COMP	(12 << 15)	Instruction completed
SCE_PC1_NON_BDS_COMP	(13 << 15)	Non-BDS instruction completed
SCE_PC1_COP1_COMP	(14 << 15)	COP1 instruction completed
SCE_PC1_STORE_COMP	(15 << 15)	Store completed
SCE_PC1_NO_EVENT	(16 << 15)	No event

Sample Program

The sample program that uses libpc is as follows:

- **/sce/ee/sample/pc/dcache**

Event Details

The generation conditions of each event are described below.

Almost all events are counted unconditionally when the event occurs. That is, there is no guarantee that the counted event was generated from an instruction that was actually completed. Even if the count was interrupted, it is possible that several events may occur and be counted. Although the instruction in the branch delay slot of the branch-likely instruction may be invalidated if the branch conditions are not completed, it is possible that several events may occur and be counted in this case also. Details about each type of event are presented below.

The "branch" indicated refers to the branch which goes with the branch prediction, in other words, the conditional branch instruction and J/JAL instructions. The JR/JALR/ERET/SYSCALL/BREAK/TRAP instructions are not included.

Low-order branch issued - SCE_PC1_LOW_BRANCH_ISSUED

This event counts the total number of branches that were issued in the low-order (even) part of an instruction pair fetch. Only these instructions are subject to BTAC lookup.

Processor cycle - SCE_PC0_CPU_CYCLE/SCE_PC1_CPU_CYCLE

This event is generated for each processor clock cycle.

Single instruction issued - SCE_PC0_SINGLE_ISSUE

This event is always generated when an instruction is issued by only one pipeline of the two logical pipelines in the EE core.

Two instructions issued at same time - SCE_PC1_DUAL_ISSUE

This event is always generated if instructions are issued using the two logical pipelines in the EE core.

Branch issued - SCE_PC0_BRANCH_ISSUED

This event is always generated when a branch is issued. A branch may be interrupted if the previous instruction caused an exception.

Branch prediction miss - SCE_PC1_BRANCH_MISS_PREDICT

This event is always generated when the destination address of the branch prediction is invalid.

BTAC miss - SCE_PC0_BTAC_MISS

This event is always generated when the instruction address lookup for the BTAC fails.

TLB miss - SCE_PC1_TLB_MISS

This event is always generated when a TLB miss is detected.

ITLB miss - SCE_PC0_ITLB_MISS

This event is always generated when an ITLB miss is detected.

DTLB miss - SCE_PC1_DTLB_MISS

This event is always generated when a DTLB miss is detected.

Instruction cache miss - SCE_PC0_ICACHE_MISS

This event is always generated when an instruction cache miss is detected. This does not include uncached fetches.

Data cache miss - SCE_PC1_DCACHE_MISS

This event is always generated when a data cache miss is detected. This is limited not only to normal data cache misses, but also includes all uncached load and store operations.

DTLB accessed - SCE_PC0_DTLB_ACCESSED

This event is always generated when a load or store is executed. It is not generated when a load or store command is interrupted.

WBB single request unavailable - SCE_PC1_WBB_SINGLE_UNAVAIL

This event is generated when there is a request to the WBB (write back buffer), but the WBB does not have enough free entries to process the request.

Non-blocking load - SCE_PC0_NONBLOCK_LOAD

This event is always generated when a non-blocking cache miss occurs due to a load instruction. This event is not defined in blocking mode.

WBB burst request unavailable (1) - SCE_PC1_WBB_BURST_UNAVAIL

This event is generated when there is a burst request to the WBB, but the WBB does not have enough free entries to process the request.

WBB single request - SCE_PC0_WBB_SINGLE_REQ

This event is generated when there is a single request to the WBB.

WBB burst request unavailable (2) - SCE_PC1_WBB_BURST_ALMOST

This event is generated when there is a burst request to the WBB but the WBB does not have enough free entries to process the request.

WBB burst request - SCE_PC0_WBB_BURST_REQ

This event is generated when there is a burst request to the WBB.

WBB burst request unavailable (3) - SCE_PC1_WBB_BURST_FULL

This event is generated when there is a burst request to the WBB, but the WBB is completely full.

CPU address bus busy - SCE_PC0_ADDR_BUS_BUSY

This event is generated for each BUSCLK (this is not the CPU clock), which indicates that the CPU address bus is unavailable. The CPU address bus is always considered to be unavailable when the bus is busy, and it is also considered to be unavailable when two addresses are issued, but data from the first address is not returned.

CPU data bus busy - SCE_PC1_DATA_BUS_BUSY

This event is generated for each BUSCLK (this is not the CPU clock), which indicates that the CPU data bus is unavailable. The CPU address bus is always considered to be unavailable when the bus is busy, and it is also considered to be unavailable when two addresses are issued, but data from the first address is not returned.

Instruction completed - SCE_PC0_INST_COMP/SCE_PC1_INST_COMP

This event is generated when an instruction is completed.

It is not generated when the instruction is interrupted due to an exception.

However, some instructions (such as SYSCALL or TEQ) which generate an exception as part of their normal operation, are excluded.

It is considered that such an instruction completes even if a "normal" exception is not about to be generated.

A branch delay slot (BDS) branch-likely instruction is counted as completed because the BDS instruction is invalidated.

The EE can execute up to two instructions simultaneously. If the two instructions are completed, the counter is incremented by two.

Non-BDS instruction completed - SCE_PC0_NON_BDS_COMP/SCE_PC1_NON_BDS_COMP

This event is generated when an instruction that does not have a branch delay slot (i.e. an instruction other than a branch or jump instruction) is completed.

For a branch-likely instruction, a branch delay slot instruction is generated even if the branch-likely instruction is invalidated.

COP2 instruction completed - SCE_PC0_COP2_COMP

This event is generated when a COP2 instruction is completed. It is generated even if the COP2 instruction completed but was later invalidated because it was in the branch delay slot of a branch-likely instruction.

COP1 instruction completed - SCE_PC1_COP1_COMP

This event is generated when a COP1 instruction is completed. It is generated even if the COP1 instruction completed but was later invalidated because it was in the branch delay slot of a branch-likely instruction.

Load completed - SCE_PC0_LOAD_COMP

This event is generated when a load instruction is completed. It is generated even if the load instruction is in the branch delay slot of a branch-likely instruction and is invalidated.

Store completed - SCE_PC1_STORE_COMP

This event is generated when a store instruction is completed. It is generated even if the store instruction was invalidated because it was in the branch delay slot of a branch-likely instruction.

No event - SCE_PC0_NO_EVENT/SCE_PC1_NO_EVENT

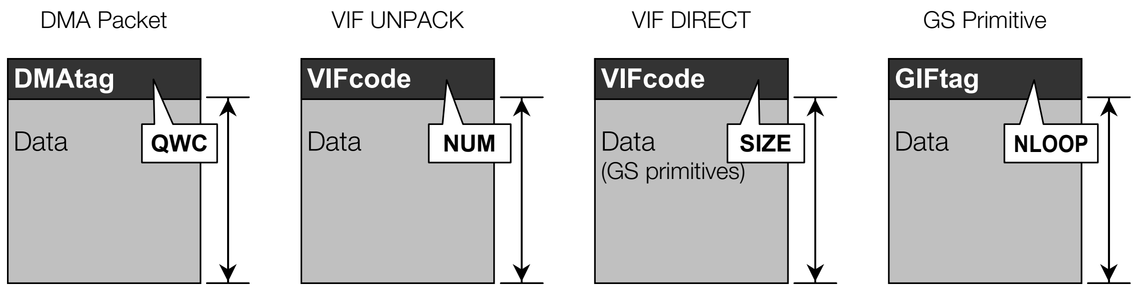
Specifying this instruction in a dummy event prevents the corresponding counter from being incremented. It is specified when it is necessary to make only one of the two counters active.

Chapter 4: Packet Library

Library Overview

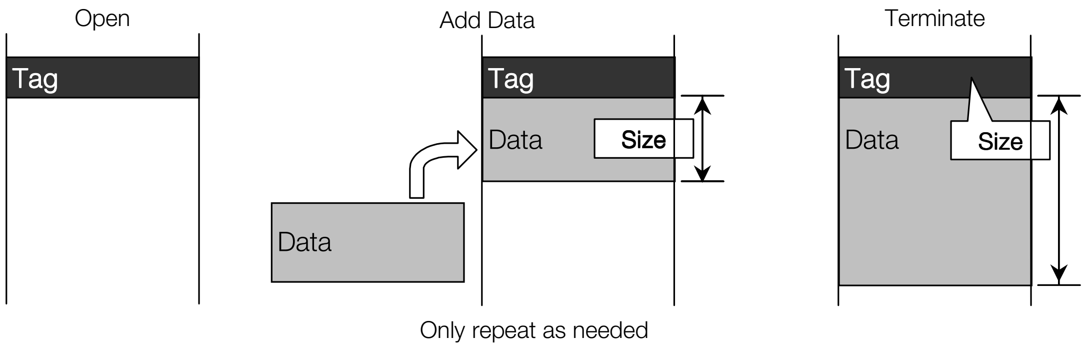
The libpkt library provides support for creation of Source Chain Mode DMA packets, VIF UNPACK packets, VIF DIRECT packets, and GIF packets (GS primitives). These packets all have tag information at the beginning and the data size is recorded in the tag. Thus, the size of the data must be known when creating these packets.

Figure 4-1



When libpkt is used, the steps performed are: open packet; add data; close (terminate) packet. Thus, the data size added to the packet can be successively counted and appropriate tag information can be automatically generated. At the same time, adjustments can be made automatically for proper data alignment.

Figure 4-2



As shown below, libpkt is divided into functions which are grouped by DMA channel (devices). This also provides a grouping based on the hierarchical structure of packets handled by the different DMA channels. The opening and closing of packets must be performed according to this packet hierarchy.

Table 4-1

Function group	DMA Channel	Device	Control target
sceVif0Pk	ch-0	VIF0	DMA packet/VIF packet
sceVif1Pk	ch-1	VIF1	DMA packet/VIF packet/GIF packet
sceGifPk	ch-2	GIF	DMA packet/GIF packet
sceDmaPk	other	---	DMA packet

In addition to libpkt, libdma is also provided as a library that supports creation of DMA packets.

Related Files

The files required for libpkt are shown below.

Table 4-2

Category	File Name
Library file	libpkt.a
Header file	libdmapk.h,libgifpk.h,libvifpk.h

Reference Material

For information on DMA packets, see "Source Chain Mode" and "DMAtag" from the "EE User's Manual".

For information on VIF packets, see "Data transfers with VIF" from the "EE User's Manual".

For information on GIF packets, see "Data formats" from the "EE User's Manual".

Chapter 5: System Configuration Library

Library Overview

The libscf library is used to retrieve user-defined system configuration information, such as the language setting and TV screen aspect ratio, from the Detailed Settings screen on the PlayStation 2.

The system configuration information for the overseas PlayStation 2, the Japanese PlayStation 2, and the DTL-T10000 are different, but libscf hides these differences.

Related Files

The following files are needed to use the system configuration library API.

Table 5-1

Category	Filename
Header file	libscf.h
Library file	libscf.a
Library file	libkernl.a (library release 2.0 and later)

Notes

The libscf functions are not thread-safe. Calling these functions simultaneously from more than one thread can cause problems. When using libscf functions with multiple threads, make sure to provide a locking mechanism, e.g., with semaphores.

Contents of System Configuration Information

Differences Between Models

On the overseas PlayStation 2, the date format, time zone, and daylight savings settings can be set.

On the current Japanese PlayStation 2 (SPCH-10000, SPCH-15000) system settings and the DTL-T10000, these items cannot be set and will always return the constant values shown below.

Future Japanese PlayStation 2 versions will be able to have the same settings as the overseas versions.

Table 5-2

Function name	DTL-T10000 development tool	SPCH-10000 and SPCH-15000	Overseas and future Japanese version PlayStation 2's
SceScfGetLanguage	SCE_JAPANESE _LANGUAGE	Settable	Settable
SceScfGetAspect	SCE_ASPECT_43	Settable	Settable
SceScfGetSpdif	SCE_SPDIF_ON	Settable	Settable
SceScfGetDate Notation	SCE_DATE _YYYYMMDD	SCE_DATE _YYYYMMDD	Settable
sceScfGet SummerTime	SCE_SUMMER TIME_OFF	SCE_SUMMER TIME_OFF	Settable
sceScfGetTime Notation	SCE_TIME _24HOUR	SCE_TIME _24HOUR	Settable
sceScfGetTimeZone	540	540	Settable
SceScfGetLocal TimefromRTC	Settable	Settable	Settable
SceScfGetGMT fromRTC	Settable	Settable	Settable

Time Setting

How the PlayStation 2 handles time settings

All Japanese and overseas PlayStation 2 models keep time by saving JST (Japan Standard Time) in the RTC.

How to set time on the PlayStation 2

When the unit is turned on with no game disk inserted, the Detailed Settings screen is displayed. The RTC time can be set from there.

The time set here would be the local time. This is converted to JST and saved in the RTC.

How to set time on the DTL-T10000

Start up a web browser on the development computer, access the DTL-T10000, and open the Management Tools screen. Click on "PlayStation 2 RTC Settings" and select "Select Time". Enter the year, month, day, hours, and seconds in JST.

Note: Before performing these settings, the "DTL-T10000 management tool PStoolSetup Ver.1.2" package must be downloaded from the developer support website and installed.

JST can be calculated as GMT (Greenwich Mean Time) + 9 hours.

Getting local time, GMT

First, get the RTC value using sceCdReadClock.

To determine local time, convert the RTC value with sceScfGetLocalTimefromRTC.

To determine GMT, convert the RTC value with sceScfGetGMTfromRTC.

Languages

The language setting can be retrieved using the sceScfGetLanguage function.

The return value will be one of the following.

• SCE_JAPANESE_LANGUAGE	Japanese
• SCE_ENGLISH_LANGUAGE	English
• SCE_FRENCH_LANGUAGE	French
• SCE_SPANISH_LANGUAGE	Spanish
• SCE_GERMAN_LANGUAGE	German
• SCE_ITALIAN_LANGUAGE	Italian
• SCE_DUTCH_LANGUAGE	Dutch
• SCE_PORTUGUESE_LANGUAGE	Portuguese

TV Screen Aspect Ratio

The aspect ratio of the TV screen can be retrieved with the sceScfGetAspect function.

The return value will be one of the following.

• SCE_ASPECT_43	4:3
• SCE_ASPECT_FULL	FULL SCREEN
• SCE_ASPECT_169	16:9

SPDIF

The SPDIF setting can be retrieved with the sceScfGetSpdif function. If SPDIF is enabled, SCE_SPDIF_ON will be returned. If disabled, SCE_SPDIF_OFF will be returned.

Date Format

The date format can be retrieved with sceScfGetDateNotation.

The return value will be SCE_DATE_YYYYMMDD, SCE_DATE_MMDDYYYY, or SCE_DATE_DDMMYYYY. When displaying the date in a title application, follow this format.

As an example, October 23, 2000 would be displayed as:

SCE_DATE_YYYYMMDD	2000/10/23
SCE_DATE_MMDDYYYY	10/23/2000
SCE_DATE_DDMMYYYY	23/10/2000

Time Format

The time format can be retrieved with the sceScfGetTimeNotation function.

The return value will be either SCE_TIME_24HOUR or SCE_TIME_12HOUR. When displaying the time in a title application, follow this format.

As an example, 15:00:00 would be displayed as:

SCE_TIME_24HOUR 24-hour time	15:00:00
SCE_TIME_12HOUR 12-hour time	03:00:00 PM

Daylight Savings Time

The daylight savings setting can be retrieved with the sceScfGetSummerTime function.

If the return value is SCE_SUMMERTIME_ON, the standard time for the time zone is advanced by one hour. The time stays the same if the return value is SCE_SUMMERTIME_OFF.