PlayStation®2 EE Library Reference
Release 2.4

Other Libraries

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.

Summary Table of Contents

About This Manual **v**
- Changes Since Last Release v
- Related Documentation v
- Typographic Conventions v
- Developer Support v

Chapter 1: Debugging Support Library **1-1**
- Structures 1-3
- VIF1 Control Functions 1-8
- VIF0 Control Functions 1-16
- GIF Control Functions 1-23
- VU1 Control Functions 1-32
- VUO Control Functions 1-43
- Pseudo Console Control Functions 1-54

Chapter 2: DMA Basic Library **2-1**
- Structures 2-3
- Functions 2-6

Chapter 3: Performance Counter Library **3-1**
- Functions 3-3

Chapter 4: Packet Library **4-1**
- Structures 4-5
- Functions 4-9

Chapter 5: System Configuration Library **5-1**
- Structures 5-3
- Functions 5-5

About This Manual

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

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

Changes Since Last Release

Chapter 2: DMA Basic Library

- In the "Description" sections of sceDmaRecv() and sceDmaSend(), information about transfer to the SPR 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: Debugging Support Library
Table of Contents

Structures **1-3**
- sceDevGifCnd 1-3
- sceDevVif0Cnd 1-4
- sceDevVif1Cnd 1-5
- sceDevVu0Cnd 1-6
- sceDevVu1Cnd 1-7

VIF1 Control Functions **1-8**
- sceDevVif1Continue 1-8
- sceDevVif1GetCnd 1-9
- sceDevVif1GetErr 1-10
- sceDevVif1GetFifo 1-11
- sceDevVif1Pause 1-12
- sceDevVif1PutErr 1-13
- sceDevVif1PutFifo 1-14
- sceDevVif1Reset 1-15

VIF0 Control Functions **1-16**
- sceDevVif0Continue 1-16
- sceDevVif0GetCnd 1-17
- sceDevVif0GetErr 1-18
- sceDevVif0Pause 1-19
- sceDevVif0PutErr 1-20
- sceDevVif0PutFifo 1-21
- sceDevVif0Reset 1-22

GIF Control Functions **1-23**
- sceDevGifContinue 1-23
- sceDevGifGetCnd 1-24
- sceDevGifGetImtMode 1-25
- sceDevGifGetP3msk 1-26
- sceDevGifPause 1-27
- sceDevGifPutFifo 1-28
- sceDevGifPutImtMode 1-29
- sceDevGifPutP3msk 1-30
- sceDevGifReset 1-31

VU1 Control Functions **1-32**
- sceDevVu1Continue 1-32
- sceDevVu1Exec 1-33
- sceDevVu1GetCnd 1-34
- sceDevVu1GetDBit 1-35
- sceDevVu1GetTBit 1-36
- sceDevVu1GetTpc 1-37
- sceDevVu1Pause 1-38
- sceDevVu1PutCnd 1-39
- sceDevVu1PutDBit 1-40
- sceDevVu1PutTBit 1-41
- sceDevVu1Reset 1-42

VUO Control Functions **1-43**
sceDevVu0Continue 1-43
sceDevVu0Exec 1-44
sceDevVu0GetCnd 1-45
sceDevVu0GetDBit 1-46
sceDevVu0GetTBit 1-47
sceDevVu0GetTpc 1-48
sceDevVu0Pause 1-49
sceDevVu0PutCnd 1-50
sceDevVu0PutDBit 1-51
sceDevVu0PutTBit 1-52
sceDevVu0Reset 1-53

Pseudo Console Control Functions **1-54**
sceDevConsAttribute 1-54
sceDevConsClear 1-55
sceDevConsClearBox 1-56
sceDevConsClose 1-57
sceDevConsDraw 1-58
sceDevConsDrawS 1-59
sceDevConsFrame 1-60
sceDevConsGet 1-61
sceDevConsInit 1-62
sceDevConsLocate 1-63
sceDevConsMessage 1-64
sceDevConsMove 1-65
sceDevConsOpen 1-66
sceDevConsPrintf 1-67
sceDevConsPut 1-68
sceDevConsRef 1-69
sceDevConsRollup 1-70
sceDevConsSetColor 1-71

Structures

sceDevGifCnd

GIF register set

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

Structure

typedef struct sceDevGifCnd {

u_long128 *tag;*	Previously processed GIFtag
u_int *stat;*	GIF_STAT register
u_int *count;*	GIF_CNT register
u_int *p3count;*	GIF_P3CNT register
u_int *p3tag;*	GIF_P3TAG register
u_int *pad;*	Padding

}

Description

This structure contains the registers of the GIF.

sceDevVif0Cnd

VIF0 register set

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

Structure

typedef struct sceDevVif0Cnd {

u_int row[*4];*	VIF0_R[0-3] registers
u_int col[*4];*	VIF0_C[0-3] registers
u_int *mask;*	VIF0_MASK register
u_int *code;*	VIF0_CODE register
u_int *stat;*	VIF0_STAT register
u_short *itop,itops;*	VIF0_ITOP register, VIF0_ITOPS register
u_short *mark;*	VIF0_MARK register
u_short *num;*	VIF0_NUM register
u_char *error;*	VIF0_ERR register
u_char cl,*wl;*	VIF0_CYCLE register
u_char *cmod;*	VIF0_MODE register
u_char *pad;*	Padding

}

Description

This structure contains the registers of VIF0.

sceDevVif1Cnd

VIF1 register set

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

Structure

typedef struct sceDevVif1Cnd {

u_int row[*4];*	VIF1_R[0-3] registers
u_int col[*4];*	VIF1_C[0-3] registers
u_int *mask;*	VIF1_MASK register
u_int *code;*	VIF1_CODE register
u_int *stat;*	VIF1_STAT register
u_short *itop,itops;*	VIF1_ITOP register, VIF1_ITOPS register
u_short *base,offset;*	VIF1_BASE register, VIF1_OFFSET register
u_short *top,tops;*	VIF1_TOP register, VIF1_TOPS register
u_short *mark;*	VIF1_MARK register
u_short *num;*	VIF1_NUM register
u_char *error;*	VIF1_ERR register
u_char *cl,wl;*	VIF1_CYCLE register
u_char *cmod;*	VIF1_MODE register
u_char *pad;*	Padding

}

Description

This structure contains the registers of VIF1.

sceDevVu0Cnd

VU0 register set

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

Structure

typedef struct sceDevVu0Cnd {

u_long128 *vf[32];*	Floating-point registers VF00 to VF31
u_int *status;*	Status flag
u_int *mac;*	MAC flag
u_int *clipping;*	Clipping flag
u_int *r, i, q;*	R register, I register, and Q register
u_short *vi[16];*	Integer registers VI00 to VI15

}

Description

This structure contains the registers of VU0.

sceDevVu1Cnd

VU1 register set

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

Structure

typedef struct sceDevVu1Cnd {

u_long128 *vf[32];*	Floating-point registers VF00 to VF31
u_int *status;*	Status flag
u_int *mac;*	MAC flag
u_int *clipping;*	Clipping flag
u_int *r,i,q,p;*	R register, I register, Q register, and P register
u_short *vi[16];*	Integer registers VI00 to VI15

}

Description

This structure contains the registers of VU1.

VIF1 Control Functions

sceDevVif1Continue

Cancel pause of VIF1

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

Syntax

#include <devvif1.h>

int sceDevVif1Continue(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Restarts processing of VIF1, which had been stopped by the sceDevVif1Pause function.

Return value

0: Restart failed

1: Restart succeeded

sceDevVif1GetCnd

Get all registers

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

Syntax

```
#include <devvif1.h>
int sceDevVif1GetCnd(
 sceDevVif1Cnd *cnd)          Structure in which the state that was obtained is stored
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Checks the state of VIF1.

The state can be obtained only when the VIF has stalled (STALL status) or is idle (IDLE status).

Return value

0: State could not be obtained

1: State was obtained

sceDevVif1GetErr

Get error mask

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

Syntax

#include <devvif1.h>

u_int sceDevVif1GetErr(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Gets the VIF1 error mask.

Return value

- bit0
  0: UNMASK interrupt
  1: MASK interrupt
- bit1
  0: Stall due to DMAtag Mismatch error
  1: Ignore DMAtag Mismatch error
- bit2
  0: Stall due to VIFcode
  1: Ignore VIFcode error

sceDevVif1GetFifo

Read data from FIFO

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

Syntax

```
#include <devvif1.h>
int sceDevVif1GetFifo(
 u_long128 *addr,          Address for storing data that was read
 int n)                    Number of data items to be read (qword units)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Reads data from VIF1-FIFO using an I/O operation.

Return value

Number of data items that were read

sceDevVif1Pause

Pause VIF1

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

Syntax

#include <devvif1.h>

int sceDevVif1Pause(

int *mode***)**

0: Pause immediately (Force Break)

1: Pause after completion of the VIFcode that is being processed

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Pauses VIF1.

Return value

0: Pause request failed

1: Pause request succeeded

sceDevVif1PutErr

Set error mask

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

Syntax

```
#include <devvif1.h>
u_int sceDevVif1PutErr(
 int interrupt,     0: Enable i bit interrupt
                    1: Disable i bit interrupt
 int miss1,         0: Stall due to DMAtag Mismatch error
                    1: Ignore DMAtag Mismatch error
 int miss2)         0: Stall due to VIFcode error
                    1: Ignore VIFcode error
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Sets the VIF1 error mask.

Return value

Bit pattern that was set in VIF1_ERR register

sceDevVif1PutFifo

Write data to FIFO

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

Syntax

```
#include <devvif1.h>
int sceDevVif1PutFifo(
 u_long128 *addr,        Starting address of data to be written
 int n)                  Number of data items to be written (qword units)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Writes data to VIF1-FIFO using an I/O operation.

Return value

Number of data items that were written

sceDevVif1Reset

Reset VIF1

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

Syntax

#include <devvif1.h>

void sceDevVif1Reset(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Resets the entire VIF1 including VIF1-FIFO.

The following bits are set in the error mask (VIF1_ERR register):

- Enable i bit interrupt
- Ignore DMAtag Mismatch error
- Ignore VIFcode error

Return value

None

VIF0 Control Functions

sceDevVif0Continue

Cancel pause of VIF0

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

Syntax

#include <devvif0.h>

int sceDevVif0Continue(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Restarts processing of VIF0, which had been stopped by the sceDevVif0Pause function.

Return value

0: Restart failed

1: Restart succeeded

sceDevVif0GetCnd

Get all registers

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

Syntax

#include <devvif0.h>
int sceDevVif0GetCnd(
sceDevVif0Cnd ****cnd*)** Structure in which the state that was obtained is stored

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Checks the state of VIF0.

The state can be obtained only when the VIF has stalled (STALL status) or is idle (IDLE status).

Return value

0: State could not be obtained

1: State was obtained

sceDevVif0GetErr

Get error mask

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

Syntax

#include <devvif0.h>

u_int sceDevVif0GetErr(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Gets the VIF0 error mask.

Return value

Value of VIF0_ERR register

sceDevVif0Pause

Pause VIF0

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

Syntax

#include <devvif0.h>

int sceDevVif0Pause(

int *mode***)**

0: Pause immediately (Force Break)

1: Pause after completion of the VIFcode that is being processed (Stop)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Pauses VIF0.

Return value

0: Pause request failed

1: Pause request succeeded

sceDevVif0PutErr

Set error mask

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

Syntax

#include <devvif0.h>

u_int sceDevVif0PutErr(

int *interrupt,*	0: Enable i bit interrupt
	1: Disable i bit interrupt
int *miss1,*	0: Stall due to DMAtag Mismatch error
	1: Ignore DMAtag Mismatch error
int *miss2***)**	0: Stall due to VIFcode error
	1: Ignore VIFcode error

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Sets the VIF0 error mask.

Return value

Bit pattern that was set.

sceDevVif0PutFifo

Write data to FIFO

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

Syntax

```
#include <devvif0.h>
int sceDevVif0PutFifo(
 u_long128 *addr,          Starting address of data to be written
 int n)                    Number of data items to be written (qword units)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Writes data to VIF0-FIFO using an I/O operation.

Return value

Number of data items that were written.

sceDevVif0Reset

Reset VIF0

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

Syntax

#include <devvif0.h>

void sceDevVif0Reset(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Resets the entire VIF0 including VIF0-FIFO.

The following bits are set in the error mask (VIF0_ERR register):

- Enable i bit interrupt
- Ignore DMAtag Mismatch error
- Ignore VIFcode error

Return value

None

GIF Control Functions

sceDevGifContinue

Restart GIF processing

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

Syntax

#include <devgif.h>

int sceDevGifContinue(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Restarts GIF processing, which had been stopped by the sceDevGifPause function.

Return value

0: Restart failed

1: Restart succeeded

sceDevGifGetCnd

Get all registers

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

Syntax

```
#include <devgif.h>
int sceDevGifGetCnd(
 sceDevGifCnd *cnd)
```

cnd — Structure in which the state that was obtained is stored

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Checks the GIF state.

The state can be obtained only when the GIF has been paused (PAUSE status) or is idle (IDLE status).

Return value

0: State could not be obtained

1: State was obtained

sceDevGifGetImtMode

Get PATH3 transfer mode

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

Syntax

#include <devgif.h>

u_int sceDevGifGetImtMode(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Gets the PATH3 transfer mode.

Return value

0: Continuous transfer mode (PATH3 image data transfers are performed continuously)

1: Intermittent transfer mode (PATH3 image data transfers are stopped every 8 qwords. If there is another data transfer request, an interrupt occurs and that data is transferred)

sceDevGifGetP3msk

Get PATH3 mask status

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

Syntax

#include <devgif.h>

int sceDevGifGetP3msk(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Gets the PATH3 mask status.

Return value

0: PATH3 unmasked

1: PATH3 masked

sceDevGifPause

Pause GIF

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

Syntax

#include <devgif.h>

int sceDevGifPause(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Pauses the GIF.

Return value

0: Pause failed

1: Pause succeeded

sceDevGifPutFifo

Write data to FIFO

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

Syntax

```
#include <devgif.h>
int sceDevGifPutFifo(
 u_long128 *addr,        Starting address of data to be written
 int n)                  Number of data items to be written (qword units)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Writes data to GIF-FIFO using an I/O operation.

Return value

Number of data items that were written

sceDevGifPutImtMode

Set PATH3 transfer mode

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

Syntax

#include <devgif.h>

void sceDevGifPutImtMode(

int *mode***)**

PATH3 transfer mode

0: Continuous transfer mode (PATH3 image data transfers are performed continuously)

1: Intermittent transfer mode (PATH3 image data transfers are stopped every 8 qwords. If there is another data transfer request, an interrupt occurs and that data is transferred)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Sets the PATH3 transfer mode.

Return value

None

sceDevGifPutP3msk

Mask PATH3

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

Syntax

```
#include <devgif.h>
int sceDevGifPutP3msk(
 int enable)
```

PATH3 mask value
0: unmask
1: mask

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Masks PATH3.

The PATH3 mask can be set only when GIF/VIF1 are IDLE.

Return value

0: Setting failed

1: Setting succeeded

sceDevGifReset

Reset GIF

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

Syntax

#include <devgif.h>

void sceDevGifReset(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Resets the entire GIF.

Return value

None

VU1 Control Functions

sceDevVu1Continue

Restart VU1 processing

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

Syntax

#include <devvu1.h>

int sceDevVu1Continue(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Restarts VU1 processing, which had been stopped by the sceDevVu1Pause function.

Return value

0: Restart failed

1: Restart succeeded

sceDevVu1Exec

Re-execute microprogram

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

Syntax

#include <devvu1.h>

void sceDevVu1Exec(

u_short *addr***)** Re-execution address

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Re-executes VU1 starting at *addr*.

Return value

None

sceDevVu1GetCnd

Read all registers

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

Syntax

```
#include <devvu1.h>
int sceDevVu1GetCnd(
 sceDevVu1Cnd *cnd)
```

cnd — Address of structure which will contain read data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Reads the contents of the registers of VU1.

Return value

0: Read failed

1: Read succeeded

sceDevVu1GetDBit

Get state of debugging function

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

Syntax

#include <devvu1.h>

int sceDevVu1GetDBit(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Gets the state of the stop function according to the Dbit of VU1.

Return value

0: disabled

1: enabled

sceDevVu1GetTBit

Get state of debugging function

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

Syntax

#include <devvu1.h>

int sceDevVu1GetTBit(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Gets the state of the stop function according to the Tbit of VU1.

Return value

0: disabled

1: enabled

sceDevVu1GetTpc

Read TPC register

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

Syntax

#include <devvu1.h>

u_short sceDevVu1GetTpc(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Reads the TPC register of VU1.

Return value

Data that was read

sceDevVu1Pause

Force Break VU1

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

Syntax

#include <devvu1.h>

int sceDevVu1Pause(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Performs a Force Break operation on VU1.

Return value

0: Break failed (debugging was already stopped)

1: Break succeeded

sceDevVu1PutCnd

Write to all registers

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

Syntax

```
#include <devvu1.h>
int sceDevVu1PutCnd(
 sceDevVu1Cnd *cnd)
```

cnd Address of structure which contains the data to be written

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Writes data to the registers of VU1.

Return value

0: Write failed

1: Write succeeded

sceDevVu1PutDBit

Set debugging function

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

Syntax

#include <devvu1.h>

void sceDevVu1PutDBit(

int *dbit***)** 0: Disable Dbit
1: Enable Dbit

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Enables or disables the stop function according to the Dbit of VU1.

Return value

None

sceDevVu1PutTBit

Set debugging function

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

Syntax

#include <devvu1.h>

void sceDevVu1PutTBit(

int *tbit***)** 0: Disable Tbit
1: Enable Tbit

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Enables or disables the stop function according to the Tbit of VU1.

Return value

None

sceDevVu1Reset

Reset VU1

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

Syntax

#include <devvu1.h>

void sceDevVu1Reset(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Resets VU1.

Return value

None

VUO Control Functions

sceDevVu0Continue

Restart VU0 processing

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

Syntax

#include <devvu0.h>

int sceDevVu0Continue(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Restarts VU0 processing after VU0 was stopped due to debug or the sceDevVu0Pause function.

Return value

0: Restart failed

1: Restart succeeded

sceDevVu0Exec

Re-execute microprogram

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

Syntax

#include <devvu0.h>

void sceDevVu0Exec(

u_short *addr***)** Re-execution address

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Re-executes VU0 starting at *addr*.

Return value

None

sceDevVu0GetCnd

Read all registers

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

Syntax

```
#include <devvu0.h>
int sceDevVu0GetCnd(
 sceDevVu0Cnd *cnd)          Address of structure which will contain read data
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Reads the contents of the registers of VU0.

Return value

0: Read failed

1: Read succeeded

sceDevVu0GetDBit

Get state of debugging function

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

Syntax

#include <devvu0.h>

int sceDevVu0GetDBit(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Gets the state of the stop function according to the Dbit of VU0.

Return value

0: disabled

1: enabled

sceDevVu0GetTBit

Get state of debugging function

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

Syntax

#include <devvu0.h>

int sceDevVu0GetTBit(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Gets the state of the stop function according to the Tbit of VU0.

Return value

0: disabled

1: enabled

sceDevVu0GetTpc

Read TPC register

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

Syntax

#include <devvu0.h>

u_short sceDevVu0GetTpc(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Reads the TPC register of VU0.

Return value

Data that was read

sceDevVu0Pause

Force Break VU0

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

Syntax

#include <devvu0.h>

int sceDevVu0Pause(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Performs a Force Break operation on VU0.

Return value

0: Break failed (debugging was already stopped)

1: Break succeeded

sceDevVu0PutCnd

Write to all registers

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

Syntax

```
#include <devvu0.h>
int sceDevVu0PutCnd(
 sceDevVu0Cnd *cnd)
```

cnd Address of structure which contains the data to be written

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Writes data to the registers of VU0.

Return value

0: Write failed

1: Write succeeded

sceDevVu0PutDBit

Set debugging function

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

Syntax

#include <devvu0.h>

void sceDevVu0PutDBit(

int *dbit,)* 0: Disable Dbit
1: Enable Dbit

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Enables or disables the stop function according to the Dbit of VU0.

Return value

None

sceDevVu0PutTBit

Set debugging function

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

Syntax

#include <devvu0.h>

void sceDevVu0PutTBit(

int *tbit,)* 0: Disable Tbit
1: Enable Tbit

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Enables or disables the stop function according to the Tbit of VU0.

Return value

None

sceDevVu0Reset

Reset VU0

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

Syntax

#include <devvu0.h>

void sceDevVu0Reset(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Resets VU0.

Return value

None

Pseudo Console Control Functions

sceDevConsAttribute

Change attributes (color)

Library	*Introduced*	*Documentation last modified*
libdev	1.5	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsAttribute(
 int cd,                    Console identifier
 u_char col)                Attribute (color code)
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Changes the default attributes of the characters output to the console.

Return value

None

sceDevConsClear

Clear console

Library	*Introduced*	*Documentation last modified*
libdev	1.4	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsClear(
 int cd);                    Console identifier
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Clears the console.

Return value

None

sceDevConsClearBox

Clear the box

Library	*Introduced*	*Documentation last modified*
libdev	1.5	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsClearBox(
```

int *cd,*	Console identifier
int *x,*	Drawing X coordinate
int *y,*	Drawing Y coordinate
u_int *w,*	Width
u_int *h)*	Height

Calling conditions

Can be called from a thread

Not multithread safe

Description

Clears the box on the console, i.e. fills it with spaces.

Return value

None

sceDevConsClose

Close console

Library	*Introduced*	*Documentation last modified*
libdev	1.4	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsClose(
 int cd);                    Console identifier
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Closes the console.

Return value

None

sceDevConsDraw

Draw console image

Library	*Introduced*	*Documentation last modified*
libdev	1.4	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsDraw(
 int cd)                    Console identifier
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Uses functions such as sceDevConsPrintf() to draw an output console image.

While the packet is being built, because direct DMA kicks are performed internally, processing will not return until drawing ends.

Return value

None

sceDevConsDrawS

Draw console image (using scratch pad)

Library	*Introduced*	*Documentation last modified*
libdev	1.4	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsDrawS(
 int cd);                    Console identifier
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Uses functions such as sceDevConsPrintf() to draw an output console image.

While the packet is being built, because direct DMA kicks are performed internally, processing will not return until drawing ends.

Scratch pad is used as the work area.

Return value

None

sceDevConsFrame

Draw frame

Library	*Introduced*	*Documentation last modified*
libdev	1.5	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsFrame(
 int cd,                    Console identifier
 int x,                     X coordinate
 int y,                     Y coordinate
 u_int w,                   Width
 u_int h);                  Height
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Draws a specific size box to a specific location on the console.

Return value

None

sceDevConsGet

Get 1 character from console location

Library	*Introduced*	*Documentation last modified*
libdev	1.5	March 26, 2001

Syntax

#include <devfont.h>

u_short sceDevConsGet(

int *cd)* Console identifier

Calling conditions

Can be called from a thread

Not multithread safe

Description

Gets console location character code and attribute information.

Return Value

Character code to the lower 8 bits

Attributes to the upper 8 bits

sceDevConsInit

Initialize console library

Library	*Introduced*	*Documentation last modified*
libdev	1.4	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsInit(void)
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Initializes the internal variables of the console library. Only call this function once.

Return value

None

sceDevConsLocate

Change cursor location

Library	*Introduced*	*Documentation last modified*
libdev	1.5	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsLocate(
 int cd,                    Console identifier
 u_int lx,                  Console X coordinate
 u_int ly);                 Console Y coordinate
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Changes the cursor location on the console.

Return value

None

sceDevConsMessage

Displays message with frame

Library	*Introduced*	*Documentation last modified*
libdev	1.5	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsMessage(
 int cd,                    Console identifier or -1
 int x,                     Character drawing X coordinate or frame GS coordinate
 int y,                     Character drawing Y coordinate or frame GS coordinate
 char const* str);          Character string
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

When cd is specified, drawing is performed so that the upper left point of the character string is located at the specified position on the console.

When cd is -1, x and y become GS primitive coordinates as in sceDevConsOpen() and drawing is performed so that the upper left point of the frame enclosing the character string is positioned in that location.

Return value

None

sceDevConsMove

Moves the box

Library	*Introduced*	*Documentation last modified*
libdev	1.5	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsMove(
```

int *cd,*	Console identifier
int *dx,*	Transfer destination X coordinate
int *dy,*	Transfer destination Y coordinate
u_int *sx,*	Transfer source X coordinate
u_int *sy*	Transfer source Y coordinate
u_int *w,*	Transfer width
u_int *h);*	Transfer height

Calling conditions

Can be called from a thread

Not multithread safe

Description

Transfers the specified box on the console to the specified coordinates.

It is uncertain what happens when the transfer source and transfer destination overlap.

Return value

None

sceDevConsOpen

Open console

Library	*Introduced*	*Documentation last modified*
libdev	1.4	March 26, 2001

Syntax

```
#include <devfont.h>
int sceDevConsOpen(
  u_int gs_x,        Console drawing X-coordinate (GS primitive coordinate)
  u_int gs_y,        Console drawing Y-coordinate (GS primitive coordinate)
  u_int chr_w,       Console width
  u_int chr_h);      Console height
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Opens the console. Only one console can be open at a time and no memory is allocated.

The maximum size which can be specified by *chr_w* and *chr_h* is 80 x 40 characters.

Return value

Console identifier

sceDevConsPrintf

Output characters to console

Library	*Introduced*	*Documentation last modified*
libdev	1.4	March 26, 2001

Syntax

```
#include <devfont.h>
u_int sceDevConsPrintf(
 int cd,                    Console identifier
 const char* str,           Formatted character string
 ...);                      Parameters
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Outputs character strings to the console.

Return value

Number of characters that were output

sceDevConsPut

Output 1 character to console location

Library	*Introduced*	*Documentation last modified*
libdev	1.5	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsPut(
 int cd,                    Console identifier
 u_char c,                  Character code
 u_char a);                 Attribute (color code)
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Outputs one character to the cursor location on the console.

Return value

None

sceDevConsRef

Attach console image to packet

Library	*Introduced*	*Documentation last modified*
libdev	1.4	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsRef(
 int cd                          Console identifier
 sceGifPacket* pPacket);         Initialized GIF packet structure
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Uses functions such as sceDevConsPrintf() to build an output console image within a GIF packet. Approximately 112 qwords are required per character.

Return value

None

sceDevConsRollup

Roll up

Library	*Introduced*	*Documentation last modified*
libdev	1.5	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsRollup(
 int cd,                    Console identifier
 u_int line);               Number of lines
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Rolls up only the specified number of lines on the console.

Return value

None

sceDevConsSetColor

Change font color table

Library	*Introduced*	*Documentation last modified*
libdev	1.4	March 26, 2001

Syntax

```
#include <devfont.h>
void sceDevConsSetColor(
 int cd,                    Console identifier
 u_char c,                  Color code (0 to 7)
 u_char r,                  Red (0 to 255)
 u_char g,                  Green (0 to 255)
 u_char b);                 Blue (0 to 255)
```

Calling conditions

Can be called from a thread

Not multithread safe

Description

Changes the color table of the characters to be output to the console.

The default color table is as follows:

0, 0x00, 0x00, 0x00
1, 0x00, 0x00, 0xff
2, 0xff, 0x00, 0x00
3, 0xff, 0x00, 0xff
4, 0x00, 0xff, 0x00
5, 0x00, 0xff, 0xff
6, 0xff, 0xff, 0x00
7, 0xff, 0xff, 0xff

Return value

None

Chapter 2: DMA Basic Library
Table of Contents

Structures **2-3**

sceDmaChan 2-3
sceDmaEnv 2-4
sceDmatag 2-5

Functions **2-6**

sceDmaAddCall 2-6
sceDmaAddCont 2-7
sceDmaAddDest 2-8
sceDmaAddDests 2-9
sceDmaAddEnd 2-10
sceDmaAddExpress 2-11
sceDmaAddICall 2-12
sceDmaAddICont 2-13
sceDmaAddIDest 2-14
sceDmaAddIDests 2-15
sceDmaAddIEnd 2-16
sceDmaAddINext 2-17
sceDmaAddIRef 2-18
sceDmaAddIRefe 2-19
sceDmaAddIRefs 2-20
sceDmaAddIRet 2-21
sceDmaAddNext 2-22
sceDmaAddRef 2-23
sceDmaAddRefe 2-24
sceDmaAddRefs 2-25
sceDmaAddRet 2-26
sceDmaDebug 2-27
sceDmaGetChan 2-28
sceDmaGetEnv 2-29
sceDmaGetNextTag 2-30
sceDmaPause 2-31
sceDmaPutEnv 2-32
sceDmaPutStallAddr 2-33
sceDmaRecv 2-34
sceDmaRecvI 2-35
sceDmaRecvN 2-36
sceDmaReset 2-37
sceDmaRestart 2-38
sceDmaSend 2-39
sceDmaSendI 2-40
sceDmaSendN 2-41
sceDmaSync 2-42
sceDmaWatch 2-43

Structures

sceDmaChan

Individual channel state

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

Structure

```
typedef struct {
    tD_CHCR chcr; u_int p0[3];          Channel control Dn_CHCR
    void *madr; u_int p1[3];            Transfer memory address Dn_MADR
    u_int qwc; u_int p2[3];             Transfer size Dn_QWC
    sceDmaTag *tadr; u_int p3[3];       Transfer tag address Dn_TADR
    void *as0; u_int p4[3];             Address stack 0 Dn_ASR0
    void *as1; u_int p5[3],             Address stack Dn_ASR1
    u_int p6[4],                        pad
    u_int p7[4];                        pad
    void *sadr; u_int p8[3];            SPR address Dn_SADR
} sceDmaChan;
```

Description

This structure is used to get/set state for an individual channel. Each member corresponds to the register shown to the right of the description.

The sceDmaGetChan() function can be used to obtain the address of the structure corresponding to each channel. Since this address is a physical register address, values are reflected in the DMAC register as soon as they are set for each member of the structure.

sceDmaEnv

Common channel state

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

Structure

typedef struct {

u_char *sts;*	Stall control source channel D_CTRL.STS
u_char *std;*	Stall control drain channel D_CTRL.STD
u_char *mfd;*	MFIFO drain channel D_CTRL.MFD
u_char *rcycle;*	Release cycle D_CTRL.RCYC
u_short *express;*	Express channel setting (LSB=CH0) D_PCR.CDE
u_short *notify;*	Specification of channel for sending termination to COP0 (LSB=CH0) D_PCR.CPC
u_short *sqwc;*	Skip word count in interleave mode D_SQWC.SQWC
u_short *tqwc;*	Transfer word count in interleave mode D_SQWC.TQWC
void * *rbadr;*	Starting address of MFIFO Ring buffer D_RBOR.ADDR
u_int *rbmsk;*	MFIFO Ring buffer size mask D_RBSR.RMSK

} sceDmaEnv;

Description

This structure is used to get/set state that is common to all channels.

Each member corresponds to the register shown to the right of the description.

The PutDEnv() and GetDenv() functions can be used to set and get member values.

sceDmatag

DMAtag

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

Structure

```
typedef struct _sceDmaTag {
    u_short qwc;                    Packet size
    u_char mark;                    Mark value
    u_char id;                      Tag ID and flag
    struct _sceDmaTag *next;        ADDR field
    u_int p[2];                     Padding
} sceDmaTag __attribute__ ((aligned(16)));
```

Description

This structure describes a DMAtag.

Functions

sceDmaAddCall

Add CALL tag

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

Syntax

void *sceDmaAddCall(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new CALL tag
void ****ctag*)**	Address information to be written in new CALL tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new CALL tag having calling destination address *ctag* and size *qwc* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the tag (=*ctag*) following the new CALL tag, the next tag can be created in succession.

The address of the data that is transferred with the new CALL tag will be returned.

Return value

The address of the packet body is returned.

sceDmaAddCont

Add CNT tag

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

Syntax

void *sceDmaAddCont(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc***)**	Size information to be written in new CNT tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new CNT tag (for Source Chain) having size *qwc* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the tag address (=address following packet body) following the new CNT tag, the next tag can be created in succession.

The address of the data that is transferred with the new CNT tag will be returned.

Return value

The address of the packet body is returned.

sceDmaAddDest

Add DEST tag

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

Syntax

void *sceDmaAddDest(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new DEST tag
void ****addr*)**	Address information to be written in new DEST tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new Destination Chain DEST tag having address *addr* and size *qwc* in memory pointed to by **tag*.

The address of the data that is transferred with the new DEST tag will be returned.

Since **tag* is updated so that it points to the address following the packet body, the next tag can be created in succession.

Return value

The address of the packet body is returned.

sceDmaAddDests

Add DESTS tag

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

Syntax

void *sceDmaAddDests(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new CNTS tag
void ****addr*)**	Address information to be written in new CNTS tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a Destination Chain DESTS tag having address *addr* and size *qwc* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the address following the packet body, the next tag can be created in succession.

The address of the data that is transferred with the new DESTS tag will be returned.

Return value

The address of the packet body is returned.

sceDmaAddEnd

Add END tag

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

Syntax

void *sceDmaAddEnd(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new END tag
void ****addr*)**	Transfer destination address (for Destination Chain)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new END tag having size *qwc* in memory pointed to by **tag*. To create a Destination Chain END tag, specify the memory address of the transfer destination in *addr*.

Since **tag* is updated so that it points to the address following the packet body, the next tag can be created following the packet body.

The address of the data that is transferred with the new END tag will be returned.

Return value

The address of the packet body is returned.

sceDmaAddExpress

Specify express (priority) transfer

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

Syntax

void sceDmaAddExpress(

sceDmatag ****tag*)** Relevant DMAtag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds the express transfer attribute to a DMAtag that is registered in the transfer list.

For details about express transfer, refer to the description of the D_PCR register in the "EE User's Manual".

Return value

None

sceDmaAddICall

Add CALL tag

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

Syntax

void *sceDmaAddICall(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new CALL tag
void ****ctag*)**	Address information to be written in new CALL tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new CALL tag with interrupt having calling destination address *ctag* and size *qwc* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the tag address (=*ctag*) following the new CALL tag, the next tag can be created in succession.

The address of the data that is transferred with the new CALL tag will be returned.

Return value

The address of the packet body is returned.

sceDmaAddICont

Add CNT tag

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

Syntax

void *sceDmaAddICont(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc***)**	Size information to be written in new CNT tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new CNT tag with interrupt having size *qwc* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the tag address (=address following packet body) following the new CNT tag, the next tag can be created in succession.

The address of the data that is transferred with the new CNT tag will be returned.

Return value

The address of the packet body is returned.

sceDmaAddIDest

Add IDEST tag

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

Syntax

void *sceDmaAddIDest(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new CNT tag
void ****addr*)**	Address information to be written in new CNT tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new Destination Chain CNT tag with interrupt having address *addr* and size *qwc* in memory pointed to by **tag*.

The address of the data that is transferred with the new CNT tag will be returned.

Since **tag* is updated so that it points to the address following the packet body, the next tag can be created following the packet body.

Return value

The address of the packet body is returned.

sceDmaAddIDests

Add IDESTS tag

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

Syntax

void *sceDmaAddIDests(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new CNTS tag
void ****addr*)**	Address information to be written in new CNTS tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new Destination Chain CNTS tag with interrupt having address *addr* and size *qwc* in memory pointed to by **tag*.

The address of the data that is transferred with the new CNTS tag will be returned.

Since **tag* is updated so that it points to the address following the packet body, the next tag can be created following the packet body.

Return value

The address of the packet body is returned.

sceDmaAddIEnd

Add END tag

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

Syntax

void *sceDmaAddIEnd(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new END tag
void ****addr*)**	Transfer destination address (for Destination Chain)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new END tag with interrupt having size *qwc* in memory pointed to by **tag*. To create a Destination Chain END tag, specify the memory address of the transfer destination in *addr*.

The address of the data that is transferred with the new END tag will be returned.

Since **tag* is updated so that it points to the address following the packet body, the next tag can be created following the packet body.

Return value

The address of the packet body is returned.

sceDmaAddINext

Add NEXT tag

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

Syntax

void *sceDmaAddINext(

**sceDmaTag ****tag,*	Tag pointer address
int *qwc,*	Size information to be written in new NEXT tag
void ***addr)**	Address information to be written in new NEXT tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new NEXT tag with interrupt having address *addr* and size *qwc* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the tag address (=*addr*) following the new NEXT tag, the next tag can be created in succession.

The address of the data that is transferred with the new NEXT tag will be returned.

Return value

The address of the packet body is returned.

sceDmaAddIRef

Add REF tag

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

Syntax

void *sceDmaAddIRef(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in the new REF tag
void ****addr*)**	Address information to be written in the new REF tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new REF tag with interrupt for referencing the data of size *qwc* at address *addr* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the address following the new REF tag, the next tag can be created in succession.

The address of the data that is transferred with the new REF tag will be returned.

Return value

The address (=*addr*) of the packet body is returned.

sceDmaAddIRefe

Add REFE tag

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

Syntax

void *sceDmaAddIRefe(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new REFE tag
void ****addr*)**	Address information to be written in new REFE tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new REFE tag with interrupt for referencing the data of size *qwc* at address *addr* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the address following the new REFE tag, the next tag can be created in succession.

The address of the data that is transferred with the new REFE tag will be returned.

Return value

The address (=*addr*) of the packet body is returned.

sceDmaAddIRefs

Add REFS tag

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

Syntax

void *sceDmaAddIRefs(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new REFS tag
void ****addr*)**	Address information to be written in new REFS tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new REFS tag with interrupt for referencing the data of size *qwc* at address *addr* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the address following the new REFS tag, the next tag can be created in succession.

The address of the data that is transferred with the new REFS tag will be returned.

Return value

The address (=*addr*) of the packet body is returned.

sceDmaAddIRet

Add RET tag

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

Syntax

void *sceDmaAddIRet(

**sceDmaTag ****tag,*	Tag pointer address
int *qwc***)**	Size information to be written in new RET tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new RET tag with interrupt having size *qwc* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the tag address following the new RET tag, the next tag can be created in succession.

The address of the data that is transferred with the new RET tag will be returned.

Return value

The address of the packet body is returned.

sceDmaAddNext

Add NEXT tag

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

Syntax

void *sceDmaAddNext(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new NEXT tag
void ****addr*)**	Address information to be written in new NEXT tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new NEXT tag having tag address *addr* and size *qwc* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the tag following the new NEXT tag, the next tag can be created in succession.

The address of the data that is transferred with the new NEXT tag will be returned.

Return value

The address of the packet body is returned.

sceDmaAddRef

Add REF tag

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

Syntax

void *sceDmaAddRef(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new REF tag
void ****addr*)**	Address information to be written in new REF tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new REF tag for referencing data of size *qwc* at address *addr* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the address following the new REF tag, the next tag can be created in succession.

The address of the data that is transferred with the new REF tag will be returned.

Return value

The address (=*addr*) of the packet body is returned.

sceDmaAddRefe

Add REFE tag

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

Syntax

void *sceDmaAddRefe(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new REFE tag
void ****addr*)**	Address information to be written in new REFE tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new REFE tag for referencing data of size *qwc* at address *addr* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the address following the new REFE tag, the next tag can be created in succession.

The address of the data that is transferred with the new REFE tag will be returned.

Return value

The address (=*addr*) of the packet body is returned.

sceDmaAddRefs

Add REFS tag

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

Syntax

void *sceDmaAddRefs(

**sceDmaTag *******tag,*	Tag pointer address
int *qwc,*	Size information to be written in new REFS tag
void ****addr*)**	Address information to be written in new REFS tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new REFS tag for referencing data of size *qwc* at address *addr* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the address following the new REFS tag, the next tag can be created in succession.

The address of the data that is transferred with the new REFS tag will be returned.

Return value

The address (=*addr*) of the packet body is returned.

sceDmaAddRet

Add RET tag

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

Syntax

void *sceDmaAddRet(

**sceDmaTag ****tag,*	Tag pointer address
int *qwc***)**	Size information to be written in new RET tag

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Creates a new RET tag having size *qwc* in memory pointed to by **tag*.

Since **tag* is updated so that it points to the tag following the new RET tag, the next tag can be created in succession.

The address of the data that is transferred with the new RET tag will be returned.

Return value

The address of the packet body is returned.

sceDmaDebug

Set debugging mode

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

Syntax

int sceDmaDebug(

int *mode***)** Debugging mode

0: Debugging OFF
1: Debugging ON

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Sets the debugging mode. If 1 is specified, debugging is turned on and an argument consistency check will be performed by each libdma function.

Return value

Previous mode value

sceDmaGetChan

Get channel structure

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

Syntax

sceDmaChan *sceDmaGetChan(

int *id***)** Channel number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Returns the address of the sceDmaChan structure corresponding to the channel number *id*. Since this address is the address of the DMA register that was mapped to memory, the DMAC register can be directly read or written by using the return value as a pointer.

Also, to specify the relevant DMA channel when starting or stopping DMA transfers, use the value returned by this function, not the channel number. The values of id correspond to DMA channels as follows:

Table 2-1

id	DMA channel
0	VIF0
1	VIF1
2	GIF
3	fromIPU
4	toIPU
5	SIF0
6	SIF1
7	SIF2
8	fromSPR
9	toSPR

Notes

Since data transfers involving the IPU and SIF must be performed carefully, fromIPU/toIPU control should be performed using libipu and SIF0/SIF1/SIF2 control should be performed using libsif.

Access to memory-mapped registers is uncached. Note that using this as a normal structure may cause processing speed to decrease.

Return value

The address of the sceDmaChan structure corresponding to the channel number *id* is returned.

If the *id* value is invalid, an error occurs, and 0 is returned.

sceDmaGetEnv

Get common channel registers

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

Syntax

sceDmaEnv *sceDmaGetEnv(

sceDmaEnv **denv*) Address of structure into which register values are to be read

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Reads the values of the DMAC common channel registers and stores them in the structure specified by *denv*.

Return value

denv is returned.

sceDmaGetNextTag

Get next transfer list tag

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

Syntax

sceDmaTag *sceDmaGetNextTag(

sceDmaTag ***tag*)** Tag address in transfer list

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Returns the address of the DMAtag that is transferred following tag.

Return value

The next tag address is returned. If *tag* specifies an invalid tag, 0 is returned.

sceDmaPause

Pause DMA transfer

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

Syntax

u_int sceDmaPause(

sceDmaChan ****d*)** DMA channel to be paused

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Pauses DMA transfer for the specified channel.

DMA transfer can be restarted by the scdDmaRestart() function.

Return value

Returns the value of the Dn_CHCR register when it was stopped. This value is passed to the sceDmaRestart function argument.

sceDmaPutEnv

Set common channel registers

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

Syntax

int sceDmaPutEnv(

sceDmaEnv ****env*)** Address of structure containg values to be written to registers

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Sets the DMAC registers with the contents of the structure pointed to by *env*.

In order to ensure system stability, the settings for common registers are first placed in memory in the sceDmaEnv structure, then this function is used to set the contents of those registers.

Since DMAC registers are shared between threads, if several threads set these registers separately, there is a danger that resource collisions will occur.

Return value

On normal termination, 0 is returned. If the contents of the structure pointed to by *env* are invalid, a negative number is returned.

sceDmaPutStallAddr

Set stall address

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

Syntax

void *sceDmaPutStallAddr(

void ***addr*)** Address at which stall is to occur

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Sets the value of *addr* in the D_STADR register, which indicates the DMA stall address.

A positive value is normally specified for *addr*. However, if -1 is specified, no stall address is set, and the value that is currently set is returned.

Return value

The previous stall address is returned.

sceDmaRecv

Start DMA transfer (Destination Chain Mode)

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

Syntax

void sceDmaRecv(

sceDmaChan ****d*)** DMA channel for performing transfer

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Starts DMA transfer from a device to memory using Destination Chain Mode.

The destination memory address is specified with tag information from the device.

The processing performed internally by this function only initiates the DMA transfer. The actual transfer is performed in the background. Also, queuing is not performed so the function is executed immediately.

When transfering to the SPR, the DMA tag address MSB must be set to 1.

Return value

None

sceDmaRecvI

Start DMA transfer (Interleave Mode, SPR->memory)

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

Syntax

void sceDmaRecvI(

**sceDmaChan ***d,*	DMA channel for performing transfer (fromSPR)
**void ***addr,*	Memory address of transfer destination
int *size***)**	Size of data to be transferred (qword)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Starts DMA transfer from SPR to memory using Interleave Mode. fromSPR (id=8) must be specified for *d*.

The SPR address of the transfer source is specified by *d->sadr*, and the memory address of the transfer destination is specified by *addr*. Also, parameters indicating a small rectangular area must be set in advance for the DMAC via *sqwc* and *tqwc* of the sceDmaEnv structure.

The processing performed internally by this function only initiates the DMA transfer. The actual transfer is performed in the background. Also, queuing is not performed so the function is executed immediately.

Return value

None

sceDmaRecvN

Start DMA transfer (Normal Mode, device->memory)

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

Syntax

void sceDmaRecvN(

**sceDmaChan ***d,*	DMA channel for performing transfer
**void ***addr,*	Memory address of transfer destination
int *size***)**	Size of data to be transferred (qword)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Starts DMA transfer from a device to memory using Normal Mode. The memory address that is the transfer destination is specified by *addr*, and the size of the data to be transferred is specified by *size*.

The processing performed internally by this function only initiates the DMA transfer. The actual transfer is performed in the background. Also, queuing is not performed so the function is executed immediately.

Return value

None

sceDmaReset

Reset DMAC

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

Syntax

int sceDmaReset(

int *mode***)** Reset mode

0: disable
1: enable

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Resets the DMAC.

After waiting for termination of all DMA transfers, this function clears all end-of-transfer handlers, initializes the DMAC, and enables or disables the DMAC according to the mode.

Return value

Previous mode value

sceDmaRestart

Restart paused DMA transfer

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

Syntax

int sceDmaRestart(

sceDmaChan ****d*,**	DMA channel for restarting transfer
u_int *chdr*)	Return value of sceDmaPause function

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Restarts transfer processing for the specified DMA channel.

Return value

If the transfer was stopped, 0 is returned. If it was already operating, 1 is returned.

sceDmaSend

Start DMA transfer (Source Chain Mode)

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

Syntax

void sceDmaSend(

**sceDmaChan ****d,*	DMA channel for performing transfer
void ****tag*)**	Starting address of transfer list

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Starts DMA transfer from memory to a device using Source Chain Mode. DMAtag, which is the beginning of the data to be transferred, is specified by *tag*, and the destination device (DMA channel) is specified by *d*.

The processing performed internally by this function only initiates the DMA transfer. The actual transfer is performed in the background. Also, queuing is not performed so the function is executed immediately.

In order to transfer from the SPR, the address MSB included in the DMA tag must be set to 1.

Return value

None

sceDmaSendI

Start DMA transfer (Interleave Mode, memory->SPR)

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

Syntax

void sceDmaSendI(

**sceDmaChan ***d,*	DMA channel for performing transfer (to SPR only)
**void ***addr,*	Address of transfer data
int *size***)**	Size of transfer data (qword)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Starts DMA transfer from memory to SPR using Interleave Mode. The channel structure address of the SPR channel (id=9) must be specified for *d*.

Also, parameters indicating a small rectangular area of the transfer source must be set in advance for the DMAC via *sqwc* and *tqwc* of the sceDmaEnv structure. The SPR address, which is the transfer destination, is specified by *d->sadr*.

The processing performed internally by this function only initiates the DMA transfer. The actual transfer is performed in the background. Also, queuing is not performed so the function is executed immediately.

Return value

None

sceDmaSendN

Start DMA transfer (Normal Mode, memory->device)

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

Syntax

void sceDmaSendN(

**sceDmaChan ***d,*	DMA channel for performing transfer
**void ***addr,*	Transfer starting address
int *size***)**	Size of data to be transferred (qword)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Starts DMA transfer from memory to a device using Normal Mode. The starting address of the transfer data is specified by *addr*, and the data size is specified by *size*.

The processing performed internally by this function only initiates the DMA transfer. The actual transfer is performed in the background. Also, queuing is not performed so the function is executed immediately.

Return value

None

sceDmaSync

Wait for end of DMA transfer and check status

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

Syntax

int sceDmaSync(

**sceDmaChan ***d,*	Relevant DMA channel
int *mode,*	Block mode 0: Block 1: Non-block
int *timeout***)**	Maximum wait interval (Hsync units)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Detects the end of DMA transfer for the channel specified by *d*.

When 0 is specified for *mode*, block mode is used, and control does not return from this function until the DMA transfer ends. When 1 is specified for *mode*, non-block mode is used, and control returns as soon as the status has been checked.

When block mode is used, a timeout occurs if the wait interval exceeds the horizontal synchronization count specified by *timeout*. (This function is not implemented in the current version.)

Return value

If the DMA transfer is in process, 1 is returned. If the DMA transfer has terminated, 0 is returned.

sceDmaWatch

Wait for transfer of specific address and check status

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

Syntax

int sceDmaWatch(

**sceDmaChan ***d,*	Relevant DMA channel
**void ***addr,*	Memory address to be checked
int *mode,*	Block mode 0: Block 1: Non-block
int *timeout***)**	Maximum wait interval (Hsync units)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Checks whether or not the data at address *addr* has been transferred using the channel specified by *d*.

When 0 is specified for *mode*, block mode is used, and control does not return from this function until the specified address has been transferred. When 1 is specified for *mode*, non-block mode is used, and control returns as soon as the status has been checked.

When block mode is used, a timeout occurs if the wait interval exceeds the horizontal synchronization count specified by timeout. (This function is not implemented in the current version.)

Return value

If the address has been transferred, 1 is returned. Otherwise, 0 is returned.

Chapter 3: Performance Counter Library
Table of Contents

Functions **3-3**

scePcGetCounter0 3-3

scePcGetCounter1 3-4

scePcStart 3-5

scePcStop 3-6

Functions

scePcGetCounter0

Get Counter 0 value

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

Syntax

int scePcGetCounter0(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Gets the value of Counter 0.

Return value

Counter 0 value

scePcGetCounter1

Get Counter 1 value

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

Syntax

int scePcGetCounter1(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Gets the value of Counter 1.

Return value

Counter 1 value

scePcStart

Start Performance Counter

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

Syntax

void scePcStart(

int *control,*	Setting of events to be counted
int *counter0,*	Counter 0 initial value
int *counter1***)**	Counter 1 initial value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Starts the Performance Counter.

The control argument specifies the event to be counted and the logical OR of the processor modes for which that event is to be counted, for each counter. See the list of event codes for the corresponding event values. The processor mode values and counter enable are represented as follows:

```
SCE_PC_EXL0   (1  <<  1)    // exception  mode for counter0
SCE_PC_K0     (1  <<  2)    // kernel mode for counter0
SCE_PC_S0     (1  <<  3)    // supervisor mode for counter0
SCE_PC_U0     (1  <<  4)    // user mode for counter0
SCE_PC_EXL1   (1  << 11)    // exception mode for counter1
SCE_PC_K1     (1  << 12)    // kernel mode for counter1
SCE_PC_S1     (1  << 13)    // supervisor mode for counter1
SCE_PC_U1     (1  << 14)    // user mode for counter1

SCE_PC_CTE    (1  << 31)    // counter enable
```

If counter enable is not specified, counting will not start.

For example, to count processor cycles in all modes using Counter 0 and to count D$ misses only in user mode using Counter 1, the value of control would be set as follows.

```
control  = SCE_PC0_CPU_CYCLE | (SCE_PC_U0|SCE_PC_S0|SCE_PC_K0|SCE_PC_EXL0);
control |= SCE_PC1_DCACHE_MISS | (SCE_PC_U1);
control |= SCE_PC_CTE;
```

If only Counter 0 or only Counter 1 is to be used, be sure to specify NO_EVENT for the counter event of the unused counter.

Return value

None

scePcStop

Stop Performance Counter

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

Syntax

void scePcStop(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Stops the Performance Counter.

Return value

None

Chapter 4: Packet Library
Table of Contents

Structures **4-5**

sceDmaPacket 4-5
sceGifPacket 4-6
sceVif0Packet 4-7
sceVif1Packet 4-8

Functions **4-9**

sceDmaPkAddData 4-9
sceDmaPkAddDataN 4-10
sceDmaPkCall 4-11
sceDmaPkCnt 4-12
sceDmaPkDump 4-13
sceDmaPkEnd 4-14
sceDmaPkInit 4-15
sceDmaPkNext 4-16
sceDmaPkRef 4-17
sceDmaPkRefe 4-18
sceDmaPkRefs 4-19
sceDmaPkReserve 4-20
sceDmaPkReset 4-21
sceDmaPkRet 4-22
sceDmaPkSize 4-23
sceDmaPkTerminate 4-24
sceGifPkAddGsAD 4-25
sceGifPkAddGsData 4-26
sceGifPkAddGsDataN 4-27
sceGifPkAddGsPacked 4-28
sceGifPkAddGsPackedN 4-29
sceGifPkCall 4-30
sceGifPkCloseGifTag 4-31
sceGifPkCnt 4-32
sceGifPkDump 4-33
sceGifPkEnd 4-34
sceGifPkInit 4-35
sceGifPkNext 4-36
sceGifPkOpenGifTag 4-37
sceGifPkRef 4-38
sceGifPkRefe 4-39
sceGifPkRefLoadImage 4-40
sceGifPkRefs 4-41
sceGifPkReserve 4-42
sceGifPkReset 4-43
sceGifPkRet 4-44
sceGifPkSize 4-45
sceGifPkTerminate 4-46
sceVif0PkAddCode 4-47
sceVif0PkAddData 4-48
sceVif0PkAddDataN 4-49

sceVif0PkAddUpkData128 4-50
sceVif0PkAddUpkData128N 4-51
sceVif0PkAddUpkData32 4-52
sceVif0PkAddUpkData32N 4-53
sceVif0PkAddUpkData64 4-54
sceVif0PkAddUpkData64N 4-55
sceVif0PkAlign 4-56
sceVif0PkCall 4-57
sceVif0PkCloseUpkCode 4-58
sceVif0PkCnt 4-59
sceVif0PkDump 4-60
sceVif0PkEnd 4-61
sceVif0PkInit 4-62
sceVif0PkNext 4-63
sceVif0PkOpenUpkCode 4-64
sceVif0PkRef 4-65
sceVif0PkRefe 4-66
sceVif0PkRefMpg 4-67
sceVif0PkRefs 4-68
sceVif0PkReserve 4-69
sceVif0PkReset 4-70
sceVif0PkRet 4-71
sceVif0PkSize 4-72
sceVif0PkTerminate 4-73
sceVif1PkAddCode 4-74
sceVif1PkAddData 4-75
sceVif1PkAddDataN 4-76
sceVif1PkAddDirectData 4-77
sceVif1PkAddDirectDataN 4-78
sceVif1PkAddGsAD 4-79
sceVif1PkAddGsData 4-80
sceVif1PkAddGsDataN 4-81
sceVif1PkAddGsPacked 4-82
sceVif1PkAddGsPackedN 4-83
sceVif1PkAddUpkData128 4-84
sceVif1PkAddUpkData128N 4-85
sceVif1PkAddUpkData32 4-86
sceVif1PkAddUpkData32N 4-87
sceVif1PkAddUpkData64 4-88
sceVif1PkAddUpkData64N 4-89
sceVif1PkAlign 4-90
sceVif1PkCall 4-91
sceVif1PkCloseDirectCode 4-92
sceVif1PkCloseDirectHLCode 4-93
sceVif1PkCloseGifTag 4-94
sceVif1PkCloseUpkCode 4-95
sceVif1PkCnt 4-96
sceVif1PkDump 4-97
sceVif1PkEnd 4-98
sceVif1PkInit 4-99
sceVif1PkNext 4-100

sceVif1PkOpenDirectCode 4-101
sceVif1PkOpenDirectHLCode 4-102
sceVif1PkOpenGifTag 4-103
sceVif1PkOpenUpkCode 4-104
sceVif1PkRef 4-105
sceVif1PkRefe 4-106
sceVif1PkRefLoadImage 4-107
sceVif1PkRefMpg 4-109
sceVif1PkRefs 4-110
sceVif1PkReserve 4-111
sceVif1PkReset 4-112
sceVif1PkRet 4-113
sceVif1PkSize 4-114
sceVif1PkTerminate 4-115

Structures

sceDmaPacket

Packet management information

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

Structure

```
typedef struct _DMAPACKET{
    u_int *pCurrent;        Address of last packet
    u_long128 *pBase;       Base address of packet area
    u_long128 *pDmaTag;     Address of active DMAtag
    u_int pad03;
} sceDmaPacket;
```

Description

This structure contains packet management information.

sceGifPacket

Packet management information

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

Structure

```
typedef struct {
    u_int *pCurrent;          Address of last packet
    u_long128 *pBase;         Base address of packet area
    u_long128 *pDmaTag;       Address of active DMAtag
    u_long *pGifTag;          Opened GIFtag
} sceGifPacket;
```

Description

This structure contains packet management information.

sceVif0Packet

Packet management information

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

Structure

```
typedef struct{
    u_int *pCurrent;          Address of last packet
    u_long128 *pBase;         Base address of packet area
    u_long128 *pDmaTag;       Address of active DMAtag
    u_int *pVifCode;          Address of opened VIFcode
    u_int numlen;
    u_int pad11;
    u_int pad12;
    u_int pad13;
}sceVif0Packet;
```

Description

This structure contains packet management information.

sceVif1Packet

Packet management information

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

Structure

```
typedef struct{
    u_int *pCurrent;          Address of last packet
    u_long128 *pBase;         Base address of packet area
    u_long128 *pDmaTag;       Address of active DMAtag
    u_int *pVifCode;          Address of opened VIFcode
    u_int numlen;
    u_long *pGifTag;          Address of opened GIFtag
    u_int pad12;
    u_int pad13;
}sceVif1Packet;
```

Description

This structure contains packet management information.

Functions

sceDmaPkAddData

Add data

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

Syntax

void sceDmaPkAddData(

sceDmaPacket ***pPacket,** Pointer to packet management information structure

u_long128 *data***);** Data to be added to packet

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds 1qword of data to the end of the packet.

Return value

None

sceDmaPkAddDataN

Add multiple items of data

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

Syntax

void sceDmaPkAddDataN(

**sceDmaPacket ****pPacket,*	Pointer to packet management information structure
u_long128* *pData,*	Pointer to data array to be added to packet
u_int *count***);**	Number of data items to be added to packet (in qwords)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds *count* qwords of data to the end of the packet.

Return value

None

sceDmaPkCall

Add CALL tag

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

Syntax

void sceDmaPkCall(

**sceDmaPacket ****pPacket,*	Pointer to packet management information structure
**u_long128 ****pCall,*	Address of next DMAtag
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a CALL tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceDmaPkCnt

Add CNT tag

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

Syntax

void sceDmaPkCnt(

**sceDmaPacket ***_pPacket,_	Pointer to packet management information structure
u_int _opt1,_	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int _opt2,_	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int _flag_**);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a CNT tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceDmaPkDump

Dump contents of packet

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

Syntax

void sceDmaPkDump(

sceDmaPacket ***pPacket*****);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Sends a hex dump of the specified packet to standard output.

This function is provided for debugging.

Return value

None

sceDmaPkEnd

Add END tag

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

Syntax

void sceDmaPkEnd(

**sceDmaPacket ***_pPacket,_	Pointer to packet management information structure
u_int _opt1,_	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int _opt2,_	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int _flag_**);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds an END tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceDmaPkInit

Initialize packet

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

Syntax

void sceDmaPkInit(

**sceDmaPacket ****pPacket,*	Pointer to packet management information structure
u_long128 ****pBase*);**	Base address of packet area

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Initializes packets.

pBase, *pDmaTag*, and *pCurrent* are initialized in the packet management information structure.

Notes

Enough memory must be reserved in advance for the packet area.

Return value

None

sceDmaPkNext

Add NEXT tag

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

Syntax

void sceDmaPkNext(

sceDmaPacket ******pPacket,*	Pointer to packet management information structure
u_long128 ******pNext,*	Address of next tag
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a NEXT tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceDmaPkRef

Add REF tag

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

Syntax

void sceDmaPkRef(

sceDmaPacket ******pPacket,*	Pointer to packet management information structure
u_long128 ******pRef,*	Address of transfer data
u_int *size,*	Size of transfer data (in qwords)
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a REF tag to the end of the packet.

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceDmaPkRefe

Add REFE tag

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

Syntax

void sceDmaPkRefe(

sceDmaPacket ******pPacket,*	Pointer to packet management information structure
u_long128 ******pRef,*	Address of transfer data
u_int *size,*	Size of transfer data (in qwords)
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a REFE tag to the end of the packet.

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceDmaPkRefs

Add REFS tag

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

Syntax

void sceDmaPkRefs(

**sceDmaPacket ****pPacket,*	Pointer to packet management information structure
**u_long128 ****pRef,*	Address of transfer data
u_int *size,*	Size of transfer data (in qwords)
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a REFS tag to the end of the packet.

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceDmaPkReserve

Add data area

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

Syntax

u_int *sceDmaPkReserve(

**sceDmaPacket ***pPacket,*	Pointer to packet management information structure
u_int *count***);**	Amount of space to reserve (in words)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Reserves an area with size count (in words) at the end of the packet area.

If there is an active DMAtag, the size of the reserved area is added to the size field in the DMAtag.

Return value

Base address of reserved area

sceDmaPkReset

Reset packet management information

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

Syntax

void sceDmaPkReset(

sceDmaPacket ****pPacket*);** Pointer to packet management information structure to be reset

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Resets packet management information.

The value of *pBase* will be copied to *pCurrent*, which indicates the last packet address.

Return value

None

sceDmaPkRet

Add RET tag

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

Syntax

void sceDmaPkRet(

**sceDmaPacket ***_pPacket,_	Pointer to packet management information structure
u_int _opt1,_	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int _opt2,_	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int _flag_**);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a RET tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceDmaPkSize

Get packet size

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

Syntax

u_int sceDmaPkSize(

sceDmaPacket ****pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Returns the size of the packet area (in qwords).

The size can be expressed as *pCurrent - pBase*.

Return value

None

sceDmaPkTerminate

Terminate packet

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

Syntax

u_long128 *sceDmaPkTerminate(

sceDmaPacket ***pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Terminates the packet.

In other words, size management for the active DMAtag is terminated.

Return value

Next address after terminated packet.

sceGifPkAddGsAD

Add GS packed mode A+D command

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

Syntax

void sceGifPkAddGsAD(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
u_int *address,*	GS command to add (register address)
u_long *data***);**	Parameter for GS command to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a GS packed mode A+D command to the packet. The GIFtag must be opened in A+D packed mode.

Return value

None

sceGifPkAddGsData

Add GS data

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

Syntax

void sceGifPkAddGsData(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
u_long *data***);**	Data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds GS data to the packet. The GIFtag must be open.

Return value

None

sceGifPkAddGsDataN

Add GS data

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

Syntax

void sceGifPkAddGsDataN(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
u_long* *pData,*	Base address of data to be added
u_int *count***);**	Number of data units to be added (in 64-bit units)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds GS data to the packet.

The GIFtag must be open.

Return value

None

sceGifPkAddGsPacked

Add GS packed mode data

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

Syntax

void sceGifPkAddGsPacked(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
u_long128 *data***);**	Parameter to GS command to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds data to be transferred in the GS packed mode (non A+D command) to the packet.

The GIFtag must be opened in packed mode.

Return value

None

sceGifPkAddGsPackedN

Add GS packed mode data

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

Syntax

void sceGifPkAddGsPackedN(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
**u_long128 ****pData,*	Address of parameter for GS command to be added
u_int *count***);**	Number of GS command parameters to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds multiple sets of data to the packet (commands other than A+D) that will be transferred using the packed mode of the GS.

The GIFtag must be opened in packed mode.

Return value

None

sceGifPkCall

Add CALL tag

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

Syntax

void sceGifPkCall(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
**u_long128 ****pCall,*	Address of next DMAtag
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a CALL tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceGifPkCloseGifTag

Close GIFtag

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

Syntax

void sceGifPkCloseGifTag(

sceGifPacket ****pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Closes the GIFtag.

Notes

GIFtags must be closed before terminating a packet.

Return value

None

sceGifPkCnt

Add CNT tag

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

Syntax

void sceGifPkCnt(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a CNT tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceGifPkDump

Dump contents of packet

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

Syntax

void sceGifPkDump(

sceGifPacket ****pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Sends a hex dump of the specified packet to standard output.

This function is provided for debugging.

Return value

None

sceGifPkEnd

Add END tag

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

Syntax

void sceGifPkEnd(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds an END tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceGifPkInit

Initialize packet

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

Syntax

void sceGifPkInit(

**sceGifPacket ****pPacket,*	Pointer to packet structure
u_long128 ****pBase*);**	Base address of packet area

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Initializes the packet.

Return value

None

sceGifPkNext

Add NEXT tag

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

Syntax

void sceGifPkNext(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
**u_long128 ****pNext,*	Address of next DMAtag
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a Next tag to the end of the packet.

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceGifPkOpenGifTag

Open and add GIFtag

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

Syntax

void sceGifPkOpenGifTag(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
u_long128 *gifTag***);**	GIFtag to be added (the NLOOP field must be set to 0)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Opens the GIFtag and adds it to the end of the packet.

Size management will be performed on the GIFtag until the GIFtagis closed.

Notes

An appropriate DMAtag must be opened before opening the GIFtag.

Return value

None

sceGifPkRef

Add REF tag

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

Syntax

void sceGifPkRef(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
**u_long128 ****pRef,*	Address of transfer data
u_int *size,*	Size of transfer data (in qwords)
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a REF tag to the end of the packet.

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceGifPkRefe

Add REFE tag

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

Syntax

void sceGifPkRefe(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
**u_long128 ****pRef,*	Address of transfer data
u_int *size,*	Size of transfer data (in qwords)
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a REFE tag to the end of the packet.

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceGifPkRefLoadImage

Add REF tag and image data to be transferred to the GS

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

Syntax

void
sceGifPkRefLoadImage(

sceGifPacket ***pPacket,**	Pointer to packet management information structure
u_short *bp,*	Base address of destination buffer (actual address will be bp x 64)
u_char *psm,*	Data format

Constant	Value	Format
SCE_GS_PSMCT32	0	PSMCT32
SCE_GS_PSMCT24	1	PSMCT24
SCE_GS_PSMCT16	2	PSMCT16
SCE_GS_PSMCT16S	10	PSMCT16S
SCE_GS_PSMT8	19	PSMT8
SCE_GS_PSMT4	20	PSMT4
SCE_GS_PSMT8H	27	PSMT8H
SCE_GS_PSMT4HL	36	PSMT4HL
SCE_GS_PSMT4HH	44	PSMT4HH
SCE_GS_PSMZ32	48	PSMZ32
SCE_GS_PSMZ24	49	PSMZ24
SCE_GS_PSMZ16	50	PSMZ16
SCE_GS_PSMZ16S	58	PSMZ16S

u_short *bw,*	Destination buffer width (actual width will be bw x 64)
u_long128 ***image,**	Base address of data to be transferred
u_int *size,*	Size of data to be transferred (in qwords)
u_int *x,* **u_int** *y,*	Upper-left coordinate of destination area
u_int *w,* **u_int** *h***);**	Width, height of transfer area

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds image data to be transferred to GS local memory to the packet. If the data size is large, the data will be automatically partitioned.

Notes

When transferring texture data to the GS, the texture page buffer must be disabled.

Return value

None

sceGifPkRefs

Add REFS tag

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

Syntax

void sceGifPkRefs(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
**u_long128 ****pRef,*	Address of transfer data
u_int *size,*	Size of transfer data (in qwords)
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a REFS tag to the end of the packet.

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceGifPkReserve

Add data area

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

Syntax

u_int *sceGifPkReserve(

sceGifPacket * *pPacket,*	Pointer to packet management information structure
u_int *count***);**	Amount of space to reserve (in words)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Reserves an area of size count (in words) at the end of the packet.

If there is an active DMAtag or an opened GIFtag, the value isadded to the respective size field (qwc, NLOOP).

Return value

Base address of reserved area

sceGifPkReset

Reset packet management information

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

Syntax

void sceGifPkReset(

sceGifPacket ****pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Resets the packet.

The contents of the packet will be discarded.

Return value

None

sceGifPkRet

Add RET tag

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

Syntax

void sceGifPkRet(

**sceGifPacket ****pPacket,*	Pointer to packet management information structure
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a RET tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceGifPkSize

Get packet size

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

Syntax

u_int sceGifPkSize(

sceGifPacket ***_pPacket_);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Returns the packet size (in qwords).

The size can be expressed as *pCurrent - pBase*.

Return value

None

sceGifPkTerminate

Terminate packet

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

Syntax

u_long128 *sceGifPkTerminate(

sceGifPacket ***pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Terminates the packet.

In other words, size management for the active DMAtag is terminated.

Notes

Any open GIFtags must be closed before terminating the packet.

Return value

Address after terminated packet

sceVif0PkAddCode

Add VIF code

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

Syntax

void sceVif0PkAddCode(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
u_int *code***);**	VIFcode to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a VIF code to the end of the packet.

Return value

None

sceVif0PkAddData

Add VIF data

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

Syntax

void sceVif0PkAddData(

**sceVif0Packet ***_pPacket,_	Pointer to packet management information structure
u_int _data_**);**	Data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds VIF data to the end of the packet.

Return value

None

sceVif0PkAddDataN

Add VIF data

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

Syntax

void sceVif0PkAddDataN(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
u_int* *pData,*	Base address of data to be added
u_int *count***);**	Number of sets of data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds VIF data to the end of the packet.

Return value

None

sceVif0PkAddUpkData128

Add UNPACK data

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

Syntax

void sceVif0PkAddUpkData128(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
u_long128 *data***);**	Data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds 128-bit data to the open VIF UNPACK command.

Return value

None

sceVif0PkAddUpkData128N

Add UNPACK data

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

Syntax

void sceVif0PkAddUpkData128N(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
**u_long128 ****pData,*	Base address of data to be added
u_int *count***);**	Number of sets of data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds multiple sets of 128-bit data to the open VIF UNPACK command.

Return value

None

sceVif0PkAddUpkData32

Add UNPACK data

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

Syntax

void sceVif0PkAddUpkData32(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
u_int *data***);**	Data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds 32-bit data to the open VIF UNPACK command.

Return value

None

sceVif0PkAddUpkData32N

Add UNPACK data

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

Syntax

void sceVif0PkAddUpkData32N(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
**u_int ****pData,*	Base address of data to be added
u_int *count***);**	Number of sets of data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds multiple sets of 32-bit data to the open VIF UNPACK command.

Return value

None

sceVif0PkAddUpkData64

Add UNPACK data

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

Syntax

void sceVif0PkAddUpkData64(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
u_long *data***);**	Data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds 64-bit data to the open VIF UNPACK command.

Return value

None

sceVif0PkAddUpkData64N

Add UNPACK data

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

Syntax

void sceVif0PkAddUpkData64N(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
**u_long ****pData,*	Base address of data to be added
u_int *count***);**	Number of sets of data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds multiple sets of 64-bit data to the open VIF UNPACK command.

Return value

None

sceVif0PkAlign

Adjust VIF code alignment

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

Syntax

void sceVif0PkAlign(

**sceVif0Packet ***pPacket,*	Pointer to packet management information structure
u_int *bit,*	Alignment unit 0 : 32bit 1 : 64bit 2 : 128bit
u_int *pos***);**	Alignment position 0 - 3 (in 32-bit units)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds the necessary number of VIF NOPs to the end of the packet in order to adjust the alignment.

This function must be used immediately before adding the VIF MPG command to the end of the packet. The *bit* and *pos* arguments indicate the conditions for adjusting the alignment. They are specified as follows.

Table 4-1

Next VIFcode	*bit*	*pos*
MPG	1	1
Other	Alignment adjustment not needed	

Return value

None

sceVif0PkCall

Add CALL tag

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

Syntax

void sceVif0PkCall(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
**u_long128 ****pCall,*	Address of next DMAtag
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a CALL tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif0PkCloseUpkCode

Close VIF UNPACK packet

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

Syntax

void sceVif0PkCloseUpkCode(

sceVif0Packet ****pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Closes the VIF UNPACK command

Return value

None

sceVif0PkCnt

Add CNT tag

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

Syntax

void sceVif0PkCnt(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a CNT tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif0PkDump

Dump contents of packet

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

Syntax

void sceVif0PkDump(

sceVif0Packet ****pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Sends a hex dump of the specified packet to standard output.

This function is provided for debugging.

Return value

None

sceVif0PkEnd

Add END tag

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

Syntax

void sceVif0PkEnd(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds an END tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif0PkInit

Initialize packet

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

Syntax

void sceVif0PkInit(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
u_long128 ****pBase*);**	Base address of packet area

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Initializes the packet.

Return value

None

sceVif0PkNext

Add NEXT tag

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

Syntax

void sceVif0PkNext(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
**u_long128 ****pNext,*	Address of next DMAtag
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a NEXT tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif0PkOpenUpkCode

Open VIF UNPACK packet

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

Syntax

void sceVif0PkOpenUpkCode(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
u_short *vuaddr,*	Address of VU Mem0 for data transfer destination
u_int *upkcmd,*	UNPACK command code (8 bit)
u_int *cl,*	Cycle register (cl) count
u_int *wl***);**	Cycle register (wl) count

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Opens the VIF UNPACK command and adds it to the end of the packet.

Subsequently, the NUM field in the VIFcode will be managed until the command is closed using sceVif0PkCloseUpkCode(). When data is added to the packet with a function such as sceVif0PkAddUpkData32(), the field will be set according to the data size.

Notes

An appropriate DMAtag must be added to the packet beforehand.

Return value

None

sceVif0PkRef

Add REF tag

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

Syntax

void sceVif0PkRef(

sceVif0Packet ***pPacket,**	Pointer to packet management information structure
u_long128 ***pRef,**	Address of transfer data
u_int *size,*	Size of transfer data (in qwords)
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a REF tag to the end of the packet.

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif0PkRefe

Add REFE tag

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

Syntax

void sceVif0PkRefe(

sceVif0Packet ***pPacket,**	Pointer to packet management information structure
u_long128 ***pRef,**	Address of transfer data
u_int *size,*	Size of transfer data (in qwords)
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a REFE tag to the end of the packet.

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif0PkRefMpg

Add REF tag and microprogram

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

Syntax

void sceVif0PkRefMpg(

**sceVif0Packet ****pPacket,*	Pointer to packet management information structure
u_short *vuaddr,*	MicroMem0 address of destination (in 64-bit units)
**u_long128 ****pMicro,*	Destination address
u_int *size,*	Size of microprogram (in qword units)
u_int *opt1***);**	VIFcode to be placed in empty areas. Usually VIF NOP.

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Loads the indicated microprogram to MicroMem0.

A REF tag is generated and added to the end of the packet.

Return value

None

sceVif0PkRefs

Add REFS tag

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

Syntax

void sceVif0PkRefs(

sceVif0Packet ***pPacket,**	Pointer to packet management information structure
u_long128 ***pRef,**	Address of transfer data
u_int *size,*	Size of transfer data (in qwords)
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a REFS tag to the end of the packet.

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif0PkReserve

Add data area

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

Syntax

u_int *sceVif0PkReserve(

sceVif0Packet **pPacket,	Pointer to packet management information structure
u_int *count***);**	Amount of space to reserve (in words)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Reserves an area of size count (in words) at the end of the packet.

If an active DMAtag or an open VIF UNPACK command is available, the size of the reserved area will be reflected in their respective size fields (qwc, NUM).

Return value

Base address of reserved area

sceVif0PkReset

Reset packet management information

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

Syntax

void sceVif0PkReset(

sceVif0Packet ****pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Resets packet management information.

The value of pBase is copied to pCurrent, which holds the end of the packet.

Return value

None

sceVif0PkRet

Add RET tag

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

Syntax

void sceVif0PkRet(

**sceVif0Packet ***_pPacket,_	Pointer to packet management information structure
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a RET tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif0PkSize

Get packet size

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

Syntax

u_int sceVif0PkSize(

sceVif0Packet ****pPacket*)** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Returns the size of the data (in qwords) registered in the packet.

The size can be expressed as pCurrent - pBase.

Return value

None

sceVif0PkTerminate

Terminate packet

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

Syntax

u_long128 *sceVif0PkTerminate(

sceVif0Packet * *pPacket* **);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function terminates the packet.

In other words, size management for the active DMAtag is terminated.

Notes

If a VIF UNPACK code is open, it must be closed before the packet is terminated.

Return value

Address after terminated packet

sceVif1PkAddCode

Add VIF code

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

Syntax

void sceVif1PkAddCode(

**sceVif1Packet ***_pPacket,_	Pointer to packet management information structure
u_int _code_**);**	VIFcode to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds the VIF code to the end of the packet.

Return value

None

sceVif1PkAddData

Add VIF data

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

Syntax

void sceVif1PkAddData(

**sceVif1Packet ***_pPacket,_	Pointer to packet management information structure
u_int _data_**);**	Data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds VIF data to the end of the packet.

Return value

None

sceVif1PkAddDataN

Add VIF data

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

Syntax

void sceVif1PkAddDataN(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
u_int* *pData,*	Base address of data to be added
u_int *count***);**	Number of sets of data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds multiple sets of VIF data to the end of the packet.

Return value

None

sceVif1PkAddDirectData

Add DIRECT data

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

Syntax

void sceVif1PkAddDirectData(

**sceVif1Packet ***_pPacket,_	Pointer to packet management information structure
u_long128 _data_**);**	Data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds data to the open VIF DIRECT command.

Return value

None

sceVif1PkAddDirectDataN

Add DIRECT data

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

Syntax

void sceVif1PkAddDirectDataN(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
u_long128* *pData,*	Base address of data to be added
u_int *count***);**	Number of sets of data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds multiple sets of data to the open VIF DIRECT command.

Return value

None

sceVif1PkAddGsAD

Add GS packed mode A+D command

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

Syntax

void sceVif1PkAddGsAD(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
u_int *address,*	GS command number
u_long *data***);**	GS command parameter

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds the GS packed mode A+D command to the packet.

The GIFtag must be open in packed mode A+D.

Return value

None

sceVif1PkAddGsData

Add GS data

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

Syntax

void sceVif1PkAddGsData(

**sceVif1Packet ***_pPacket,_	Pointer to packet management information structure
u_long _data_**);**	Data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds GS data to the packet.

The GIFtag must be open.

Return value

None

sceVif1PkAddGsDataN

Add GS data

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

Syntax

void sceVif1PkAddGsDataN(

**sceVif1Packet ***_pPacket,_	Pointer to packet management information structure
u_long* _pData,_	Base address of data to be added
u_int _count_**);**	Number of sets of data to be added (in 64-bit units)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds multiple sets of GS data to the packet.

The GIFtag must be open.

Return value

None

sceVif1PkAddGsPacked

Add GS packed mode data

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

Syntax

void sceVif1PkAddGsPacked(

**sceVif1Packet ***_pPacket,_	Pointer to packet management information structure
u_long128 _data_**);**	GS command parameter to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds data to be transferred in GS packed mode to the packet (commands other than A+D).

The GIFtag must be open in packed mode.

Return value

None

sceVif1PkAddGsPackedN

Add GS packed mode data

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

Syntax

void sceVif1PkAddGsPackedN(

sceVif1Packet ***pPacket,**	Pointer to packet management information structure
u_long128 ***pData,**	Address of GS command parameter to be added
u_int *count***);**	Number of GS command parameters to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds multiple sets of data to be transferred in GS packed mode to the packet (commands other than A+D).

The GIFtag must be open in packed mode.

Return value

None

sceVif1PkAddUpkData128

Add UNPACK data

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

Syntax

void sceVif1PkAddUpkData128(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
u_long128 *data***);**	Data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds 128-bit data to the open VIF UNPACK command.

Return value

None

sceVif1PkAddUpkData128N

Add UNPACK data

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

Syntax

void sceVif1PkAddUpkData128N(

sceVif1Packet ***pPacket,**	Pointer to packet management information structure
u_long128 ***pData,**	Base address of data to be added
u_int *count***);**	Number of sets of data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds multiple sets of 128-bit data to the open VIF UNPACK command.

Return value

None

sceVif1PkAddUpkData32

Add UNPACK data

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

Syntax

void sceVif1PkAddUpkData32(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
u_int *data***);**	Data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds 32-bit data to the open VIF UNPACK command.

Return value

None

sceVif1PkAddUpkData32N

Add UNPACK data

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

Syntax

void sceVif1PkAddUpkData32N(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
**u_int ****pData,*	Base address of data to be added
u_int *count***);**	Number of sets of data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds multiple sets of 32-bit data to the open VIF UNPACK command.

Return value

None

sceVif1PkAddUpkData64

Add UNPACK data

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

Syntax

void sceVif1PkAddUpkData64(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
u_long *data***);**	Data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds 64-bit data to the open VIF UNPACK command.

Return value

None

sceVif1PkAddUpkData64N

Add UNPACK data

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

Syntax

void sceVif1PkAddUpkData64N(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
**u_long ****pData,*	Base address of data to be added
u_int *count***);**	Number of sets of data to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds multiple sets of 64-bit data to the open VIF UNPACK command.

Return value

None

sceVif1PkAlign

Adjust alignment of VIFcode

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

Syntax

void sceVif1PkAlign(

**sceVif1Packet ***pPacket,*	Pointer to packet management information structure
u_int *bit,*	Alignment unit 0 : 32bit 1 : 64bit 2 : 128bit
u_int *pos***);**	Alignment position 0- 3 (in 32-bit units)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds the necessary number of VIF NOPs to the end of the packet in order to adjust the alignment.

This function must be used immediately before adding the VIF MPG/VIF DIRECT/VIF DIRECTHL commands to the end of the packet.

The *bit* and *pos* arguments indicate the conditions for adjusting the alignment. They are specified as follows.

Table 4-2

Next VIFcode	*bit*	*pos*
MPG	1	1
DIRECT	2	3
DIRECTHL	2	3
Other	Alignment adjustment not needed	

Return value

None

sceVif1PkCall

Add CALL tag

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

Syntax

void sceVif1PkCall(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
**u_long128 ****pCall,*	Address of next DMAtag
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a CALL tag.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif1PkCloseDirectCode

Close VIF DIRECT packet

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

Syntax

void sceVif1PkCloseDirectCode(

sceVif1Packet ****pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Closes the VIF DIRECT command.

Notes

If there are any open GIFtags, they must be closed before this function is used to close the VIF DIRECT command.

Return value

None

sceVif1PkCloseDirectHLCode

Close VIF DIRECTHL packet

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

Syntax

void sceVif1PkCloseDirectHLCode(

sceVif1Packet ****pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function closes the VIF DIRECTHL command.

Notes

If there are any open GIFtags, they must be closed before this function is used to close the VIF DIRECTHL command.

Return value

None

sceVif1PkCloseGifTag

Close GIFtag

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

Syntax

void sceVif1PkCloseGifTag(

sceVif1Packet ***pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Closes the GIFtag.

Return value

None

sceVif1PkCloseUpkCode

Close VIF UNPACK packet

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

Syntax

void sceVif1PkCloseUpkCode(

sceVif1Packet ***pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Closes the VIF UNPACK command.

Return value

None

sceVif1PkCnt

Add CNT tag

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

Syntax

void sceVif1PkCnt(

**sceVif1Packet ***_pPacket,_	Pointer to packet management information structure
u_int _flag_**);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a CNT tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif1PkDump

Dump contents of packet

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

Syntax

void sceVif1PkDump(

sceVif1Packet ***pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Sends a hex dump of the specified packet to standard output.

This function is provided for debugging.

Return value

None

sceVif1PkEnd

Add END tag

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

Syntax

void sceVif1PkEnd(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds an END tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif1PkInit

Initialize packet

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

Syntax

void sceVif1PkInit(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
u_long128 ****pBase*);**	Base address of packet

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Initializes the packet.

Return value

None

sceVif1PkNext

Add NEXT tag

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

Syntax

void sceVif1PkNext(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
**u_long128 ****pNext,*	address for next DMAtag
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a NEXT tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif1PkOpenDirectCode

Open VIF DIRECT packet

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

Syntax

void sceVif1PkOpenDirectCode(

**sceVif1Packet ***_pPacket,_	Pointer to packet management information structure
int _stall_**);**	Interrupt flag (i-bit) setting 0: No interrupts (i=0) 1: Generate interrupt (i=1)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Opens the VIF DIRECT command after adjusting the alignment and adds it to the end of the packet.

Subsequently, the NUM field in the VIFcode will be managed until the command is closed using sceVif1PkCloseDirectCode(). When data is added to the packet with a function such as sceVif1PkAddDirectData(), the field will be set according to the data size.

Return value

None

sceVif1PkOpenDirectHLCode

Open VIF DIRECTHL packet

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

Syntax

void sceVif1PkOpenDirectHLCode(

**sceVif1Packet ***_pPacket,_	Pointer to packet management information structure
int _stall_**);**	Interrupt flag (i-bit) setting 0: No interrupts (i=0) 1: Generate interrupt (i=1)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Opens the VIF DIRECTHL command after adjusting the alignment and adds it to the end of the packet.

Subsequently, the NUM field in the VIFcode will be managed until the command is closed using sceVif1PkCloseDirectHLCode(). When data is added to the packet with a function such as sceVif1PkAddDirectData(), the field will be set according to the data size.

Return value

None

sceVif1PkOpenGifTag

Open GIFtag

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

Syntax

void sceVif1PkOpenGifTag(

**sceVif1Packet ***_pPacket,_	Pointer to packet management information structure
u_long128 _gifTag_**);**	GIFtag to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function opens the GIFtag and adds it to an already open VIF DIRECT command.

Subsequently, the qwc field in the GIFtag will be managed until the command is closed using sceVif1PkCloseGifTag(). When data is added to the packet with a function such as sceVif1PkAddGsData(), the field will be set according to the data size.

Return value

None

sceVif1PkOpenUpkCode

Open VIF UNPACK packet

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

Syntax

void sceVif1PkOpenUpkCode(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
u_short *vuaddr,*	Address of VU Mem1 for data transfer destination
u_int *upkcmd,*	UNPACK command code (8 bit)
u_int *cl,*	Cycle register (cl) count
u_int *wl***);**	Cycle register (wl) count

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Opens the VIF UNPACK command and adds it to the end of the packet.

Subsequently, the NUM field in the VIFcode will be managed until the command is closed using sceVif1PkCloseUpkCode(). When data is added to the packet with a function such as sceVif1PkAddUpkData32(), the field will be set according to the data size.

Notes

An appropriate DMAtag must be added to the packet beforehand.

Return value

None

sceVif1PkRef

Add REF tag

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

Syntax

void sceVif1PkRef(

sceVif1Packet ***pPacket,**	Pointer to packet management information structure
u_long128 ***pRef,**	Address of transfer data
u_int *size,*	Size of transfer data (in qwords)
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a REF tag to the end of the packet.

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif1PkRefe

Add REFE tag

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

Syntax

void sceVif1PkRefe(

sceVif1Packet ***pPacket,**	Pointer to packet management information structure
u_long128 ***pRef,**	Address of transfer data
u_int *size,*	Size of transfer data (in qwords)
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a REFE tag to the end of the packet.

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif1PkRefLoadImage

Add REF tag and image data to be transferred to the GS

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

Syntax

void sceVif1PkRefLoadImage(

sceVif1Packet ***pPacket,** — Pointer to packet management information

u_short *bp,* — Base address of destination buffer (actual address will be bp x 64)

u_char *psm,* — Data format

Constant	Value	Format
SCE_GS_PSMCT32	0	PSMCT32
SCE_GS_PSMCT24	1	PSMCT24
SCE_GS_PSMCT16	2	PSMCT16
SCE_GS_PSMCT16S	10	PSMCT16S
SCE_GS_PSMT8	19	PSMT8
SCE_GS_PSMT4	20	PSMT4
SCE_GS_PSMT8H	27	PSMT8H
SCE_GS_PSMT4HL	36	PSMT4HL
SCE_GS_PSMT4HH	44	PSMT4HH
SCE_GS_PSMZ32	48	PSMZ32
SCE_GS_PSMZ24	49	PSMZ24
SCE_GS_PSMZ16	50	PSMZ16
SCE_GS_PSMZ16S	58	PSMZ16S

u_short *bw,* — Destination buffer width (actual width will be bw x 64)

u_long128 ***image,** — Base address of data to be transferred

u_int *size,* — Size of data to be transferred (in qwords)

u_int *x,* **u_int** *y,* — Upper-left coordinate of destination area

u_int *w,* **u_int** *h***);** — Width, height of transfer area

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds image data to be transferred to GS local memory along with a REF tag to the packet. If the data size is large, the data will be automatically partitioned.

Notes

When transferring texture data to the GS, the texture page buffer must be disabled.

Return value

None

sceVif1PkRefMpg

Add REF tag and microprogram

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

Syntax

void sceVif1PkRefMpg(

**sceVif1Packet ****pPacket,*	Pointer to packet management information structure
u_short *vuaddr,*	MicroMem1 address of transfer destination (in 64-bit units)
**u_long128 ****pMicro,*	Destination address
u_int *size,*	Size of microprogram (in qword units)
u_int *opt1***);**	VIFcode to be placed in empty areas. usually VIF NOP.

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Loads the specified microprogram to MicroMem1.

A REF tag is generated and added to the end of the packet.

Return value

None

sceVif1PkRefs

Add REFS tag

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

Syntax

void sceVif1PkRefs(

sceVif1Packet ***pPacket,**	Pointer to packet management information structure
u_long128 ***pRef,**	Address of transfer data
u_int *size,*	Size of transfer data (in qwords)
u_int *opt1,*	Data to be placed in high-order bits of DMAtag (bits 95-64)
u_int *opt2,*	Data to be placed in high-order bits of DMAtag (bits 127-96)
u_int *flag***);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a REFS tag to the end of the packet.

opt1, *opt2* are data to be placed in the 64 high-order bits of DMAtag.

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif1PkReserve

Add data area

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

Syntax

u_int *sceVif1PkReserve(

**sceVif1Packet ***pPacket,*	Pointer to packet management information structure
u_int *count***);**	Amount of space to reserve (in words)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Reserves an area of size count (in words) at the end of the packet.

If an active DMAtag or an open VIFcode/GIFtag is available, the size of the reserved area will be reflected in their respective size fields.

Return value

Base address of reserved area

sceVif1PkReset

Reset packet management information

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

Syntax

void sceVif1PkReset(

sceVif1Packet ***_pPacket_);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Resets packet management information.

The value of pBase is copied to pCurrent, which holds the end of the packet.

Return value

None

sceVif1PkRet

Add RET tag

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

Syntax

void sceVif1PkRet(

**sceVif1Packet ***_pPacket,_	Pointer to packet management information structure
u_int _flag_**);**	Bit pattern specifying IRQ, PCE

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Adds a RET tag to the end of the packet.

Subsequently, the size of data added to the packet is counted and reflected in the size information (QWC field).

Bit 31 of *flag* indicates the IRQ and bits 26-27 indicate the PCE.

The 32 low-order bits of DMAtag are OR'ed so all other bits must be set to 0.

Return value

None

sceVif1PkSize

Get packet size

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

Syntax

u_int sceVif1PkSize(
sceVif1Packet ****pPacket*);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Returns the size of the packet (in qwords).

The size can be expressed as pCurrent - pBase.

Return value

None

sceVif1PkTerminate

Terminate packet

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

Syntax

u_long128 *sceVif1PkTerminate(

sceVif1Packet * *pPacket* **);** Pointer to packet management information structure

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Terminates the packet.

In other words, size management for the active DMAtag is terminated.

Notes

If there are any open VIFcodes or GIFtags, they must be closed before the packet is terminated.

Return value

Address after the terminated packet

Chapter 5: System Configuration Library
Table of Contents

Structures **5-3**

sceScfT10kConfig 5-3

Functions **5-5**

sceScfGetAspect 5-5
sceScfGetDateNotation 5-6
sceScfGetGMTfromRTC 5-7
sceScfGetLanguage 5-8
sceScfGetLocalTimefromRTC 5-9
sceScfGetSpdif 5-10
sceScfGetSummerTime 5-11
sceScfGetTimeNotation 5-12
sceScfGetTimeZone 5-13
sceScfSetT10kConfig 5-14

Structures

sceScfT10kConfig

Structure for setting up DTL-T10000 system configuration information

Library	*Introduced*	*Documentation last modified*
libscf	2.0	January 2, 2001

Structure

```
#include <libscf.h>
typedef struct {
    short  TimeZone;        Difference between local standard time and GMT (in minutes)
    u_char Aspect;          Aspect ratio setting for television screen
    u_char DateNotation     Date format
    u_char Language;        Language setting
    u_char Spdif;           SPDIF setting
    u_char SummerTime;      Daylight savings setting
    u_char TimeNotation;    Time format
} sceScfT10kConfig;
```

Description

This structure is used to set up system configuration information for the DTL-T10000.

The allowable values for each of the members are given below.

<TimeZone>

The difference, in minutes, between the local standard time (local time if daylight savings is not in effect) and GMT.

Example: Tokyo 540, London 0

<Aspect>

SCE_ASPECT_43	4:3
SCE_ASPECT_FULL	Full-screen
SCE_ASPECT_169	16:9

<DateNotation>

SCE_DATE_YYYYMMDD	Year/Month/Day (SPCH-10000, SPCH-15000)
SCE_DATE_MMDDYYYY	Month/Day/Year
SCE_DATE_DDMMYYYY	Day/Month/Year

< Language>

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

<Spdif>

SCE_SPDIF_ON	SPDIF ON
SCE_SPDIF_OFF	SPDIF OFF

<SummerTime>

SCE_SUMMERTIME_OFF	No daylight savings (SPCH-10000, SPCH-15000)
SCE_SUMMERTIME_ON	Daylight savings

<TimeNotation>

SCE_TIME_24HOUR	24-hour format (SPCH-10000,SPCH-15000)
SCE_TIME_12HOUR	12-hour (AM/PM) format

See also

sceScfSetT10kConfig

Functions

sceScfGetAspect

Get the aspect ratio for the television screen

Library	*Introduced*	*Documentation last modified*
libscf	2.0	March 26, 2001

Syntax

#include <libscf.h>

int sceScfGetAspect (void)

Calling conditions

Can be called from a thread

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

Description

This function gets the video screen aspect ratio from the detailed settings in the system configuration information.

Return value

Table 5-1

Return value	Meaning
SCE_ASPECT_43	4:3
SCE_ASPECT_FULL	Full-screen
SCE_ASPECT_169	16:9

sceScfGetDateNotation

Get date format

Library	*Introduced*	*Documentation last modified*
libscf	2.0	March 26, 2001

Syntax

#include <libscf.h>

int sceScfGetDateNotation (void)

Calling conditions

Can be called from a thread

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

Description

This function returns the date display format from the system configuration information.

The current Japanese models of the PlayStation 2 (SPCH-10000, SPCH-15000) will return SCE_DATE_YYYMMDD.

Return value

Table 5-2

Return value	Description
SCE_DATE_YYYYMMDD	Year/Month/Day (SPCH-10000, SPCH-15000)
SCE_DATE_MMDDYYYY	Month/Day/Year
SCE_DATE_DDMMYYYY	Day/Month/Year

sceScfGetGMTfromRTC

Convert from JST to GMT

Library	*Introduced*	*Documentation last modified*
libscf	2.0	March 26, 2001

Syntax

#include <libscf.h>

void sceScfGetGMTfromRTC (

sceCdCLOCK **rtc*) Address of structure holding date and time

Calling conditions

Can be called from a thread

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

Description

The structure pointed to by the rtc parameter will be treated as JST (Japan Standard time) and will be converted to GMT (Greenwich Mean Time). The converted values will be written back to the same structure.

The sceCdCLOCK structure can only store dates from January 1, 2000 through December 31, 2099.

For example, if Japan Standard Time is 6:00 January 1, 2000, GMT would be 21:00 December 31, 1999. This function would write 99/12/31 21:00.

Return value

None

sceScfGetLanguage

Get the language setting

Library	*Introduced*	*Documentation last modified*
libscf	2.0	March 26, 2001

Syntax

#include <libscf.h>

int sceScfGetLanguage (void)

Calling conditions

Can be called from a thread

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

Description

This function gets the language setting from the system configuration information.

Return value

Table 5-3

Return value	Description
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

sceScfGetLocalTimefromRTC

Convert from JST to local time

Library	*Introduced*	*Documentation last modified*
libscf	2.0	March 26, 2001

Syntax

#include <libscf.h>

void sceScfGetLocalTimefromRTC (

sceCdCLOCK **rtc)* Address of structure holding date and time

Calling conditions

Can be called from a thread

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

Description

The structure pointed to by the rtc parameter will be treated as JST (Japan Standard time) and will be converted to local time, taking the time zone and daylight savings time into consideration. The converted values will be written back to the same structure.

The sceCdCLOCK structure can only store dates from January 1, 2000 through December 31, 2099.

For example, if Japan Standard Time is 6:00 January 1, 2000, then U.S. Eastern standard time (GMT-5 hours) would be 16:00 December 31, 1999. This function would write 99/12/31 16:00.

Return value

None

sceScfGetSpdif

Return whether SPDIF is enabled

Library	*Introduced*	*Documentation last modified*
libscf	2.0	March 26, 2001

Syntax

#include <libscf.h>

int sceScfGetSpdif (void)

Calling conditions

Can be called from a thread

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

Description

This function gets information about whether the SPDIF is enabled in the system configuration information.

Return value

Table 5-4

Return value	Meaning
SCE_SPDIF_ON	SPDIF is ON
SCE_SPDIF_OFF	SPDIF is OFF

sceScfGetSummerTime

Return whether daylight savings is in effect

Library	*Introduced*	*Documentation last modified*
libscf	2.0	March 26, 2001

Syntax

#include <libscf.h>

int sceScfGetSummerTime (void)

Calling conditions

Can be called from a thread

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

Description

This function returns whether or not daylight savings is set in the system configuration information.

The current Japanese models of the PlayStation 2 (SPCH-10000, SPCH-15000) will return SCE_SUMMERTIME_OFF.

Return value

Table 5-5

Return value	Meaning
SCE_SUMMERTIME_OFF	No daylight savings (SPCH-10000, SPCH-15000)
SCE_SUMMERTIME_ON	Daylight savings in effect

sceScfGetTimeNotation

Return time format

Library	*Introduced*	*Documentation last modified*
libscf	2.0	March 26, 2001

Syntax

#include <libscf.h>

int sceScfGetTimeNotation (void)

Calling conditions

Can be called from a thread

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

Description

This function returns the time format in the system configuration information.

If SCE_TIME_24HOUR is returned, the time will be displayed in 24-hour format (e.g., 15:23). If SCE_TIME_12HOUR is returned, time will be displayed in 12-hour format (e.g., 3:23PM).

The current Japanese models of the PlayStation 2 (SPCH-10000, SPCH-15000) will return SCE_TIME_24HOUR.

Return value

Table 5-6

Return value	Meaning
SCE_TIME_24HOUR	24-hour format (SPCH-10000,SPCH-15000)
SCE_TIME_12HOUR	12-hour (AM/PM) format

sceScfGetTimeZone

Return time zone information

Library	*Introduced*	*Documentation last modified*
libscf	2.0	March 26, 2001

Syntax

#include <libscf.h>

int sceScfGetTimeZone (void)

Calling conditions

Can be called from a thread

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

Description

The current Japanese models of the PlayStation 2 (SPCH-10000, SPCH-15000) will return 540.

Return value

The difference, in minutes, between local time and GMT will be returned.

sceScfSetT10kConfig

Set up DTL-T10000 system configuration information

Library	*Introduced*	*Documentation last modified*
libscf	2.0	March 26, 2001

Syntax

#include <libscf.h>

void sceScfSetT10kConfig(

sceScfT10kConfig **config*) Address of system configuration information structure

Calling conditions

Can be called from a thread

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

Description

This function sets up system configuration information for the DTL-T10000.

It will have no effect on units other than the DTL-T10000.

Return value

None